A Globally Optimal Energy-Efficient Power Control Framework and its Efficient Implementation in Wireless Interference Networks

Bho Matthiesen, *Member, IEEE,* Alessio Zappone, *Senior Member, IEEE,* Karl-L. Besser, *Student Member, IEEE,* Eduard A. Jorswieck, *Fellow, IEEE,* Merouane Debbah, *Fellow, IEEE*

***Abstract*—This work develops a novel power control framework for energy-efficient power control in wireless networks. The proposed method is a new branch-and-bound procedure based on problem-specific bounds for energy-efficiency maximization that allow for faster convergence. This enables to find the global solution for all of the most common energy-efficient power control problems with a complexity that, although still exponential in the number of variables, is much lower than other available global optimization frameworks. Moreover, the reduced complexity of the proposed framework allows its practical implementation through the use of deep neural networks. Specifically, thanks to its reduced complexity, the proposed method can be used to train an artificial neural network to predict the optimal resource allocation. This is in contrast with other power control methods based on deep learning, which train the neural network based on suboptimal power allocations due to the large complexity that generating large training sets of optimal power allocations would have with available global optimization methods. As a benchmark, we also develop a novel first-order optimal power allocation algorithm. Numerical results show that a neural network can be trained to predict the optimal power allocation policy.**

***Index Terms*—Energy Efficiency, Non-Convex Optimization, Branch-and-Bound, Sum-of-Ratios, Interference Networks, Deep Learning, Artificial Neural Network**

I. Introduction

Energy management is known to be one of the crucial issues for the sustainability of future wireless communication networks, whose bit-per-Joule energy efficiency (EE) is required to increase by a factor 2000 compared to present networks [3]. To this end, several energy management techniques have been proposed, such as energy-efficient network deployment, the use of renewable energy sources, as well as the development of resource allocation techniques aimed at EE maximization [4]. This work focuses on the last of these energy management approaches, and in particular on the issue of energy-efficient power control.

Due to the fractional nature of energy-efficient performance metrics, traditional convex optimization theory can not directly handle energy-efficient power control problems. Instead, the mathematical frameworks of generalized concavity theory and fractional programming provide a suitable set of optimization methods. However, these optimization tools come with a limited complexity only when the fractional function to maximize fulfills certain mathematical assumptions, such as the concavity of the numerator [5]. Unfortunately, this requirement is not fulfilled whenever an interference-limited network needs to be optimized, and indeed in these cases EE maximization problems are typically NP-hard [6]. Thus, several suboptimal methods have been proposed for energy-efficient resource allocation in wireless interference networks. The simplest approach is to resort to interference cancelation techniques or to orthogonal transmission schemes, thus falling back into the noise-limited regime [7]–[9]. However, this either leads to a poor resource efficiency, or to noise enhancement effects and/or non-linear receive schemes. A more recent approach is instead of trying to develop energy-efficient power control algorithms that, although not provably optimal, enjoy limited (typically polynomial) complexity. In [10], the maximization of the system global energy efficiency (GEE) is pursued by merging fractional programming with alternating optimization, decomposing the problem into a series of simpler sub-problems. A similar approach is used in [11], where the minimum of the users' EEs is maximized, and in [12], where the sum of the individual users' EEs is considered. In [13], fractional programming is merged with sequential optimization theory to develop power control algorithms for the maximization of the system GEE or minimum of the users' EEs. Unlike previous contributions, the method proposed there guarantees first-order optimality, and has been numerically shown to achieve global optimality in small network setups in [14]. Results for networks with a larger number of users are not available at present, mainly due to the fact that, in order to compute the global

Parts of this paper were presented at IEEE WCNC 2018 [1] and IEEE SPAWC 2019 [2].

The work of B. Matthiesen is supported in part by the German Research Foundation (DFG) in the Collaborative Research Center 912 "Highly Adaptive Energy-Efficient Computing," and under Germany's Excellence Strategy (EXC 2077 at University of Bremen, University Allowance). The work of K.-L. Besser and E. Jorswieck is supported in part by the German Research Foundation (DFG) under grant JO 801/23-1. The work of A. Zappone and M. Debbah was funded by the European Commission through the H2020-MSCA IF-BESMART project under Grant Agreement 749336.

B. Matthiesen is with the Department of Communications Engineering, University of Bremen, 28359 Bremen, Germany (e-mail: matthiesen@uni-bremen.de). A. Zappone is with DIEI, University of Cassino and Southern Lazio, Cassino, FR, Italy (e-mail: alessio.zappone@unicas.it). K.-L. Besser and E. A. Jorswieck are with the Department of Information Theory and Communication Systems, TU Braunschweig, Germany (e-mail: k.besser@tu-bs.de, e.jorswieck@tu-bs.de). M. Debbah is with Université Paris-Saclay, CNRS, CentraleSupélec, Laboratoire des signaux et systèmes, 91190, Gif-sur-Yvette, France (e-mail: merouane.debbah@l2s.centralesupelec.fr) and with the Mathematical and Algorithmic Sciences Laboratory, France Research Center, Huawei Technologies, Paris, France. Part of this research was conducted while B. Matthiesen and E. A. Jorswieck were with Technische Universität Dresden and A. Zappone was with Université Paris-Saclay, CNRS, CentraleSupélec, Laboratoire des signaux et systèmes, 91190, Gif-sur-Yvette, France.

We thank the Center for Information Services and High Performance Computing (ZIH) at TU Dresden for generous allocations of computer time.

maximum of the GEE in networks with more than a handful of users, available global optimization frameworks require a computational complexity that is impractical even for offline simulation. This issue is even more severe when other energy-efficient metrics are considered, such as the product or the sum of the users' EEs. In particular, the maximization of the sum of the users' EEs is acknowledged as the hardest energy-efficient power control problem [5], and it has been numerically shown in [1] that, unlike what was shown for the GEE in [14], state-of-the-art first-order optimal methods with polynomial complexity have a gap to the global solution even in networks with a few users. Moreover, it is known that weighted sum energy efficiency (WSEE) maximization is an NP-complete optimization problem even if each ratio has concave numerator and linear denominator [15], due to the fact that addition is not guaranteed to preserve properties like pseudo-concavity or quasi-concavity.

Thus, the analysis of the state-of-the-art shows a significant gap regarding the availability of optimization frameworks that allow the computation of the global maximum of energy-efficient performance metrics in wireless interference networks with a complexity that is at least affordable for offline simulation. Besides having its own theoretical value, developing a framework for efficient offline global optimization is of interest also because it provides a practical way to benchmark the performance of any sub-optimal, low-complexity optimization routine. Finally, having efficient global optimization algorithms represents a key requirement also for the use of deep learning in wireless communications, a topic that has been gaining momentum recently [16], [17]. Indeed, an efficient global optimization algorithm provides a feasible way to generate large training sets, which is a critical requirement in order for artificial neural networks (ANNs) to perform well.

Specifically, the main contributions of the work are as follows:

- We develop a novel and improved branch-and-bound (BB)-based algorithm to globally solve the most common energy-efficient power control problems with a complexity that is much lower than available global optimization approaches for these kind of problems. This is achieved thanks to the development of improved bounding techniques that significantly accelerate the convergence of the algorithm to the global solution. This makes it possible to globally solve NP-hard optimization problems in a time that is fully affordable for offline simulations.
- In addition to the newly proposed BB-based algorithm, we also propose a novel power control algorithm that is guaranteed to obtain a first-order optimal solution of the WSEE maximization problem with polynomial complexity. Although not enjoying global optimality, this approach provides a practical benchmark to assess the performance of other optimization methods.
- Finally, we also develop a feedforward ANN-based energy-efficient power control method that works in tandem with the proposed BB-based algorithm and leverages the universal approximation property of ANNs. Specifically, the lower complexity of the proposed BB-based algorithm enables the offline generation a large training set containing optimal power allocations for many different channel realizations, which is then used to train an ANN to learn the optimal map between the network channel realization and the corresponding optimal power control policy. Afterwards, the trained ANN is able to provide a power control policy for new realizations of the network channels not contained in the training set with an extremely limited computational complexity that essentially amounts to performing one forward propagation of the ANN. Thus, the trained ANN can be applied as an effective and low-complexity online power allocation method. It is to be stressed how this approach differs from previous related works, which train the ANN based on suboptimal power allocation routines [18], [19], and are thus intrinsically limited by the sub-optimality of the data in the training set. Instead, the reduced complexity of our proposed BB-based algorithm makes it practical to generate offline large training sets with optimal power allocations.
- Extensive numerical results are provided to assess the performance of the proposed approaches. Interestingly, it is found that both the proposed first-order approach and the ANN-based approach achieve near-optimal performance but at very different computational cost. Moreover, our numerical analysis shows that the proposed ANN-based method is even robust to mismatches in the channel statistics between the training data and the actual scenario in which the ANN is tested.

The rest of the work is organized as follows. Section II describes the system model and formulates the power control problem. Section III introduces the proposed globally optimal algorithm. Section IV introduces the proposed first-order optimal power control algorithm, while Section V develops the ANN-based power control method that employs an ANN trained on the globally optimal algorithm from Section III. Section VI illustrates the numerical performance assessment of the proposed algorithms, while concluding remarks are provided in Section VII.

Notation: Vectors $\boldsymbol{a}$ and matrices $\boldsymbol{A}$ are typeset in bold face, sets and maps in calligraphic letters $\mathcal{A}$. The sets of real and complex numbers are denoted by $\mathbb{R}$ and $\mathbb{C}$, respectively. We define the sets $\{a_i\}_{i=1}^n = \{a_1, a_2, \ldots, a_n\}$ and vectors $(a_i)_{i=1}^n = (a_1, a_2, \ldots, a_n)$ where we might omit the index bounds if clear from the context. $[\boldsymbol{a}, \boldsymbol{b}] = \{\boldsymbol{x} \,|\, a_i \leq x_i \leq b_i, \text{ for all } i = 1, \ldots, n\}$, while $\boldsymbol{a} \leq \boldsymbol{b}$ means $a_i \leq b_i$ for all $i = 1, \ldots, n$, $\mathbf{0}$ denotes a zero vector of appropriate dimension, and scalar functions applied to vectors are to be applied element-wise, e.g., $\log \boldsymbol{a} = (\log a_i)_{i=1}^n$. The operators $\|\cdot\|$, $|\cdot|$, $(\cdot)^T$, and $(\cdot)^H$ denote the L^2-norm, absolute value, transpose, and conjugate transpose, respectively. Logarithms are, unless noted otherwise, to the base 2, and $\ln$ denotes the natural logarithm. Definitions and approximations are marked by $:=$ and $\approx$, respectively.

II. System Model & Problem Statement

Consider the uplink of a multi-cell interference network with L single antenna user equipments (UEs) served by M base

station (BS), equipped with n_R antennas each. Let $a(i)$ be the BS serving user i. Then, the received signal at BS $a(i)$ is

$$\boldsymbol{y}_{a(i)} = \sum_{j=1}^{L} \boldsymbol{h}_{a(i),j} x_j + \boldsymbol{z}_{a(i)} \tag{1}$$

wherein $\boldsymbol{h}_{a(i),j} \in \mathbb{C}^{n_R}$ is the channel from UE j to BS $a(i)$, $x_j \in \mathbb{C}$ the symbol transmitted by UE j, and $\boldsymbol{z}_{a(i)}$ zero-mean circularly symmetrical complex Gaussian noise with power $\sigma^2_{a(i)}$. Each UE is subject to an average transmit power constraint, i.e., $p_j \leq P_j$ where p_j is the average power of x_j. Upon matched-filter reception, and under the assumption of Gaussian codebooks, the achievable rate from UE i to its intended BS is

$$R_i = B \log \left(1 + \frac{\alpha_i p_i}{1 + \sum_{j \neq i} \beta_{i,j} p_j} \right) \tag{2}$$

with B the communication bandwidth, $\alpha_i = \frac{\left\| \boldsymbol{h}_{a(i),i} \right\|^2}{\sigma^2_{a(i)}}$, and $\beta_{i,j} = \frac{\left| \boldsymbol{h}^H_{a(i),i} \boldsymbol{h}_{a(i),j} \right|^2}{\sigma^2_{a(i)} \left\| \boldsymbol{h}_{a(i),i} \right\|^2}$.

In this context, the EE of the link between UE i and its intended BS is defined as the benefit-cost ratio in terms of the link's achievable rate and power consumption necessary to operate the link, i.e.,

$$\mathrm{EE}_i = \frac{B \log \left(1 + \frac{\alpha_i p_i}{1 + \sum_{j \neq i} \beta_{i,j} p_j} \right)}{\mu_i p_i + P_{c,i}} \tag{3}$$

with μ_i the inefficiency of UE i's power amplifier and $P_{c,i}$ the total static power consumption of UE i.

A. Problem Formulation and Motivation

The goal of this work is to develop an efficient and effective algorithm to solve energy-efficient power control problems. The four fundamental energy-efficient metrics that have received most research attention in the open literature are the GEE, the WSEE, the weighted product energy efficiency (WPEE), and the weighted minimum energy efficiency (WMEE), defined as

$$\mathrm{GEE} = \frac{\sum_{i=1}^{L} \log \left(1 + \frac{\alpha_i p_i}{1 + \sum_{j \neq i} \beta_{i,j} p_j} \right)}{\sum_{i=1}^{L} \left(\mu_i p_i + P_{c,i} \right)} \tag{4}$$

$$\mathrm{WSEE} = \sum_{i=1}^{L} w_i \frac{\log \left(1 + \frac{\alpha_i p_i}{1 + \sum_{j \neq i} \beta_{i,j} p_j} \right)}{\mu_i p_i + P_{c,i}} \tag{5}$$

$$\mathrm{WPEE} = \prod_{i=1}^{L} \left(\frac{\log \left(1 + \frac{\alpha_i p_i}{1 + \sum_{j \neq i} \beta_{i,j} p_j} \right)}{\mu_i p_i + P_{c,i}} \right)^{w_i} \tag{6}$$

$$\mathrm{WMEE} = \min_{1 \leq i \leq L} w_i \frac{\log \left(1 + \frac{\alpha_i p_i}{1 + \sum_{j \neq i} \beta_{i,j} p_j} \right)}{\mu_i p_i + P_{c,i}} . \tag{7}$$

In order to consider a specific case-study, the optimization algorithms to be developed in the following will be stated primarily with reference to the problem of WSEE maximization. Nevertheless, it must be stressed that both the global optimization framework and the ANN-based method to be developed are general enough to apply to all four energy-efficient metrics above. This will be explicitly shown in Section III with reference to the global optimization method, and in Section V with reference to the ANN-based method.

Regarding the motivation for choosing the WSEE as main case-study, several arguments can be made. First of all, as already mentioned, the WSEE is the hardest to maximize among energy-efficient metrics. Thus, showing that our optimization framework performs well in this case represents a strong motivation for its use to tackle other energy-efficient power control problems, too. Among the reasons that make Problem (P1) so challenging to handle by traditional optimization techniques, the following observations hold:

- The objective of (P1) is a sum of fractions, a functional form that is NP-complete in general [15], and which indeed can not be tackled with polynomial complexity by any available fractional programming technique.
- Each summand of the objective of (P1) is a fraction with a non-concave numerator, which would make (P1) NP-hard even if only the weighted sum-rate were to be maximized (i.e., if $\mu_i = 0$ for all $i = 1, \ldots, L$) [6].

Another reason to motivate the consideration of the WSEE lies in its different operational meaning compared to the more widely-considered GEE metric. While the latter is meant to optimize the EE of the network as a whole, the former enables to balance the EE levels among the UEs, thanks to the use of the weights $w_i \geq 0$, prioritizing the EE of some links over others. For example, some terminals might not have a reliable energy source, e.g., due to being powered by a renewable energy source. In this case, it can be useful to prioritize the individual EE of these users over that of the other users.

Further elaborating on the prioritization of the users' individual EEs, it is important to mention that Problem (P1) can be cast into the framework of multi-objective optimization. Formally speaking, let us consider the problem of jointly maximizing all of the users' individual EEs, namely

$$\max_{\boldsymbol{0} \leq \boldsymbol{p} \leq \boldsymbol{P}} [\mathrm{EE}_1(\boldsymbol{p}), \mathrm{EE}_2(\boldsymbol{p}), \ldots, \mathrm{EE}_L(\boldsymbol{p})] \tag{8}$$

with $\boldsymbol{p} = [p_1, \ldots, p_L]$ and $\boldsymbol{P} = [P_1, \ldots, P_L]$. A moment's thought shows that the individual EEs are conflicting objectives, since, for all i, (3) is strictly decreasing in p_j for all $j \neq i$, i.e., $p_j = 0$ for all $j \neq i$ maximizes (3), but the optimal p_i is strictly positive. Thus, no power allocation vector exists that simultaneously maximizes all individual EEs. In this context, in order to define a suitable solution concept for (8), the notion of energy-efficient Pareto region is defined as the set of all EE vectors $[\mathrm{EE}_1(\boldsymbol{p}), \mathrm{EE}_2(\boldsymbol{p}), \ldots, \mathrm{EE}_L(\boldsymbol{p})]$ which can be attained by a feasible power allocation vector $\boldsymbol{p}$. The outer boundary of the Pareto region is called Pareto boundary, and provides all feasible operating points at which it is not possible to improve the EE of one user, without decreasing the EE of another user [20], [21]. For this reason, the points on the Pareto-boundary are called Pareto-optimal and are commonly understood as the solutions of the multi-objective optimization problem.

Having said this, a popular method of determining Pareto-optimal solutions is the so-called scalarization approach, which consists of maximizing a weighted sum of the objectives.

Therefore, Problem (P1) is the scalarized version of the multi-objective Problem (8), and thus solving (P1) yields a Pareto-optimal[1] solution of (8).

A third motivation to focus on the WSEE is that it is a direct generalization of the system weighted sum-rate

$$\text{WSR} = \sum_{i=1}^{L} w_i \log\left(1 + \frac{\alpha_i p_i}{1 + \sum_{j \neq i} \beta_{i,j} p_j}\right), \tag{9}$$

obtained from the WSEE by setting $\mu_i = 0$ and $P_{c,i} = 1$ for all $i = 1, \ldots, L$. In addition, we emphasize that the global optimization approach and the ANN-based approach to be developed in Sections III and V, respectively, are not restricted to the WSEE, and will be shown to encompass all four major energy-efficient metrics defined above.

Finally, based on all the above considerations, the main optimization problem to be considered in the following sections is the maximization of the WSEE subject to maximum power constraints, which is formulated as

$$\begin{cases} \max\limits_{\boldsymbol{p}} & \sum\limits_{i=1}^{L} w_i \dfrac{\log\left(1 + \frac{\alpha_i p_i}{1+\sum_{j\neq i}\beta_{i,j}p_j}\right)}{\mu_i p_i + P_{c,i}} \\ \text{s.t.} & 0 \leq p_i \leq P_i, \quad \text{for all } i = 1, 2, \ldots, L. \end{cases} \tag{P1}$$

In the following, we denote the objective of (P1) as $f(\boldsymbol{p})$. The coming section develops a novel BB-based method to obtain the global solution of (P1), with a lower complexity than other global optimization frameworks.

III. Globally Optimal Power Control

Problem (P1) has, in general, multiple locally optimal solutions and is known to be NP-complete [15]. Thus, traditional optimization approaches like gradient descent or interior-point methods are not able to solve (P1) globally. Instead, in this section we develop a novel BB procedure that obtains an ε-optimal solution of (P1), i.e., a power allocation $\bar{\boldsymbol{p}}$ that satisfies $f(\bar{\boldsymbol{p}}) \geq f(\boldsymbol{p}) - \varepsilon$ for all $\boldsymbol{p} \in [\boldsymbol{0}, \boldsymbol{P}]$. While still scaling exponentially in the number of variables, the developed method has significantly lower complexity than BB methods relying on general-purpose bounds, e.g., monotonic optimization [22] or mixed monotonic programming [23].

The core idea behind BB is to successively partition the feasible set and compute, for each partition element $\mathcal{M}$, an upper bound on the objective value in $\mathcal{M}$ and a candidate solution from a point in $\mathcal{M}$. Clearly, the maximum bound over all partition elements is an upper bound on the solution of (P1) and the best encountered candidate solution, the current best value (CBV), is a lower bound on the optimal value. As the partition is refined, the bounds become tighter, better candidate solutions are encountered, and partition elements can be pruned because their upper bound is worse than the CBV, i.e., they cannot contain a better solution. Ultimately, the maximum upper bound and the CBV converge and the algorithm is terminated.[2]

[1]It can be proved that solving a scalarized problem for varying combinations of the weights enables to obtain the complete convex hull of the Pareto-boundary.

[2]We refer the interested reader to [24, Chapter 3], [25, Chapter 4], and [26, Chapter 6] for a thorough introduction to BB methods.

We partition the set $[\boldsymbol{0}, \boldsymbol{P}] = [0, P_i]^L$ into L-dimensional hyper-rectangles of the form

$$\mathcal{M}^k = \{\boldsymbol{p} \,:\, r_i^{(k)} \leq p_i \leq s_i^{(k)}, \forall\, i = 1, \ldots, L\} \triangleq [\boldsymbol{r}^{(k)}, \boldsymbol{s}^{(k)}]\,. \tag{10}$$

In each hyper-rectangle $\mathcal{M}^k$, an upper bound $\beta(\mathcal{M}^k)$ on the values that $f(\boldsymbol{p})$ can take in $\mathcal{M}^k$ is computed, i.e., $\beta(\mathcal{M}^k) \geq \max_{\boldsymbol{p}\in\mathcal{M}^k} f(\boldsymbol{p})$ for all $\mathcal{M}^k \subseteq [\boldsymbol{0}, \boldsymbol{P}]$. Such a bound over the box $\mathcal{M}^k = [\boldsymbol{r}^{(k)}, \boldsymbol{s}^{(k)}]$ is

$$\begin{aligned} &\max_{\boldsymbol{p}\in\mathcal{M}^k} \sum_{i=1}^{L} w_i \frac{\log\left(1 + \frac{\alpha_i p_i}{1+\sum_{j\neq i}\beta_{i,j}p_j}\right)}{\mu_i p_i + P_{c,i}} \\ &\quad\leq \sum_{i=1}^{L} w_i \max_{\boldsymbol{p}\in\mathcal{M}^k} \frac{\log\left(1 + \frac{\alpha_i p_i}{1+\sum_{j\neq i}\beta_{i,j}p_j}\right)}{\mu_i p_i + P_{c,i}} \\ &\quad= \sum_{i=1}^{L} w_i \max_{r_i^{(k)} \leq p_i \leq s_i^{(k)}} \underbrace{\frac{\log\left(1 + \frac{\alpha_i p_i}{1+\sum_{j\neq i}\beta_{i,j}r_j^{(k)}}\right)}{\mu_i p_i + P_{c,i}}}_{:=\overline{\text{EE}}_i(p_i, \mathcal{M}^k)} \qquad (11) \\ &\quad=: \beta(\mathcal{M}^k) \qquad (12) \end{aligned}$$

where the last step is due to EE_i being decreasing in p_j for all $j \neq i$. Thus, we compute the bound by maximizing each EE_i with respect to $\boldsymbol{p}$ over $\mathcal{M}^k$. This provides an excellent accuracy-complexity trade-off and leads to fast convergence, as confirmed by the numerical analysis reported in Section VI. At the same time, the bound can be computed with reasonably low computational complexity. Indeed, $\overline{\text{EE}}_i(p_i, \mathcal{M}^k)$ is a strictly pseudo-concave function of p_i, being the ratio of a strictly concave over an affine function [5]. Thus, its global maximizer is obtained as the unique zero of its derivative, namely the unique solution of the equation

$$\frac{\alpha_i(\mu_i p_i + P_{c,i})}{1 + \sum_{j\neq i}\beta_{i,j} r_j^{(k)} + \alpha_i p_i} = \mu_i \ln\left(1 + \frac{\alpha_i p_i}{1 + \sum_{j\neq i}\beta_{i,j} r_j^{(k)}}\right), \tag{13}$$

which is denoted by $\hat{p}_i^{(k)}$. This point might be outside $\mathcal{M}^k$. Due to the pseudo-concavity of $\overline{\text{EE}}_i(p_i, \mathcal{M}^k)$, the optimal solution of the maximization in (11) is

$$\widetilde{p}_i^{(k)} = \begin{cases} r_i^{(k)} & \text{if } \hat{p}_i^{(k)} \leq r_i^{(k)} \\ \hat{p}_i^{(k)} & \text{if } r_i^{(k)} < \hat{p}_i^{(k)} < s_i^{(k)} \\ s_i^{(k)} & \text{otherwise.} \end{cases} \tag{14}$$

Equation (13) can be solved numerically with any root finding algorithm, e.g. with Newton-Raphson's or Halley's method. However, due to $\frac{\mathrm{d}}{\mathrm{d}p_i}\overline{\text{EE}}_i$ approaching zero quickly as $p_i \to \infty$ these methods might suffer from numerical problems. Instead, a more stable numerical solution of (13) is computed in Appendix A as

$$\hat{p}_i^{(k)} = \frac{1}{\widetilde{\alpha}_i}\left(\frac{\frac{\widetilde{\alpha}_i}{\mu_i}P_{c,i} - 1}{W_0\left(\left(\frac{\widetilde{\alpha}_i}{\mu_i}P_{c,i} - 1\right)e^{-1}\right)} - 1\right), \tag{15}$$

where $\widetilde{\alpha}_i = \frac{\alpha_i}{1+\sum_{j\neq i}\beta_{i,j}r_j^{(k)}}$ and $W_0(\cdot)$ is the principal branch of the Lambert W function.

The bound in (12) is tight at $\boldsymbol{r}^{(k)}$, i.e., $\text{EE}_i(\boldsymbol{r}^{(k)}) =$

Algorithm 1 Global optimal solution of (P1)

1: Initialize $\mathscr{R}_0 = \{[\mathbf{0}, \boldsymbol{P}]\}$, $\gamma = -\infty$, $k = 0$

2: **repeat**
3: Select $\mathcal{M}^k \in \arg\max\{\beta(\mathcal{M}) \,|\, \mathcal{M} \in \mathscr{R}_k\}$.
4: Bisect $\mathcal{M}^k$ via $(\boldsymbol{v}^k, j_k)$ with $(\boldsymbol{v}^k, j_k)$ as in (16) and
5: Let $\mathscr{P}_k = \{\mathcal{M}^k_-, \mathcal{M}^k_+\}$ with $\mathcal{M}^k_-$, $\mathcal{M}^k_+$ as in (17).

6: **for all** $\mathcal{M} \in \mathscr{P}_k$ **do**
7: $\mathscr{S}_k \leftarrow \emptyset$
8: **if** $\beta(\mathcal{M}) > \gamma + \varepsilon$ **then**
9: Add $\mathcal{M}$ to $\mathscr{S}_k$.
10: Let $\boldsymbol{r}$ such that $\mathcal{M} = [\boldsymbol{r}, \boldsymbol{s}]$.
11: **if** $f(\boldsymbol{r}) > \gamma$ **then**
12: $\bar{\boldsymbol{p}} \leftarrow \boldsymbol{r}$
13: $\gamma \leftarrow f(\boldsymbol{r})$
14: **end if**
15: **end if**
16: **end for**

17: $\mathscr{R}_{k+1} \leftarrow \mathscr{S}_k \cup \mathscr{R}_k \setminus \{\mathcal{M}^k\}$
18: $k \leftarrow k + 1$
19: **until** $\mathscr{R}_k = \emptyset$

20: **return** $\bar{\boldsymbol{p}}$ as the optimal solution

$\overline{\mathrm{EE}}_i(r_i^{(k)}, \mathcal{M}^k)$, and the bounding procedure generates a point $\widetilde{\boldsymbol{p}}^{(k)}$ that is often in the interior of $\mathcal{M}^k$. This allows the use of an adaptive bisection rule that drives the bound directly towards an objective value. Instead, typical exhaustive bisection strives to reduce the size of each partition element towards a singleton which results in much slower convergence. In particular, in each iteration k the hyper-rectangle $\mathcal{M}^k = [\boldsymbol{r}^{(k)}, \boldsymbol{s}^{(k)}]$ with the best bound is selected and then bisected via $(\boldsymbol{v}^{(k)}, j_k)$ where

$$\boldsymbol{v}^{(k)} = \frac{1}{2}(\widetilde{\boldsymbol{p}}^{(k)} + \boldsymbol{r}^{(k)}), \quad j_k = \arg\max_j \left| \widetilde{p}_j^{(k)} - r_j^{(k)} \right|. \quad (16)$$

The partition sets are given by the subrectangles $\mathcal{M}^k_-$ and $\mathcal{M}^k_+$ determined by the hyperplane $p_{j_k} = v_{j_k}$ as

$$\begin{aligned} \mathcal{M}^k_- &= \{\boldsymbol{x} \,|\, r^{(k)}_{j_k} \le x_{j_k} \le v^{(k)}_{j_k},\ r_i^{(k)} \le x_i \le s_i^{(k)}\ (i \ne j_k)\} \\ \mathcal{M}^k_+ &= \{\boldsymbol{x} \,|\, v^{(k)}_{j_k} \le x_{j_k} \le s^{(k)}_{j_k},\ r_i^{(k)} \le x_i \le s_i^{(k)}\ (i \ne j_k)\}. \end{aligned} \quad (17)$$

The final procedure is stated in Algorithm 1. The set $\mathscr{R}_k$ holds the current partition, i.e., it contains all hyper-rectangles to be examined, and γ is the CBV. In lines 3–5, the box $\mathcal{M}^k$ with the best bound is selected and bisected. This bisection is stored in $\mathscr{P}_k$. Later, in line 17, the hyper-rectangle $\mathcal{M}^k$ is removed from $\mathscr{R}_k$ since it is replaced by its subdivision. Each of the resulting subrectangles in $\mathscr{P}_k$ is examined in lines 6–16. If its bound is not better than the CBV (plus the absolute tolerance ε), it is ignored. Otherwise, it is added to $\mathscr{R}_k$ via $\mathscr{S}_k$ in line 17. Then, the algorithm selects a feasible point from $\mathcal{M}^k$ and updates the current best solution (CBS) $\bar{\boldsymbol{p}}$ if the objective value for this point is better than the CBV. When $\mathscr{R}_k$ is empty, all partition elements have been pruned because their bound was less than the CBV and the problem is solved. The absolute tolerance in line 11 is necessary to ensure termination of Algorithm 1 within a finite number of iterations.

Convergence of Algorithm 1 is stated formally below.

Proposition 1: For every $\varepsilon > 0$, Algorithm 1 converges in a finite number of iterations towards a point with objective value within an ε-region of the global optimal value of (P1).

Proof: First, observe that the branching procedure does not generate any box containing infeasible points. Hence, no feasibility checks are necessary during Algorithm 1.

After each iteration, $\mathscr{R}_k$ contains all boxes that might hold a better solution than the CBS $\bar{\boldsymbol{p}}$. If the algorithm terminates, then all boxes $\mathcal{M}$ generated from $\mathscr{R}_k$ after the last update of γ had a bound $\beta(\mathcal{M}) \le \gamma + \varepsilon$. Since $\bar{\boldsymbol{p}}$ is feasible and satisfies $f(\bar{\boldsymbol{p}}) > \beta(\mathcal{M}) - \varepsilon \ge \max_{\boldsymbol{p} \in \mathcal{M}} f(\boldsymbol{p}) - \varepsilon$ for every $\mathcal{M}$, $\bar{\boldsymbol{p}}$ is a global ε-optimal solution. Thus, it remains to show that the algorithm is finite, i.e., that the termination criterion $\mathscr{R}_k = \emptyset$ occurs at some point. By virtue of [26, Prop. 6.2], this is the case if both points $\widetilde{\boldsymbol{p}}^{(k)}, \boldsymbol{r}^{(k)}$ in (16) are in $\mathcal{M}^k$, $\widetilde{\boldsymbol{p}}^{(k)}$ is feasible, and the bounding procedure satisfies

$$f\big(\boldsymbol{r}^{(k_v)}\big) - \beta\big(\mathcal{M}^{k_v}\big) = o\big(\big\|\widetilde{\boldsymbol{p}}^{(k)} - \boldsymbol{r}^{(k)}\big\|\big). \quad (18)$$

Since, by [26, Thm. 6.4], there exists a subsequence $\{k_v\}_v$ such that $\boldsymbol{r}^{(k_v)}$ and $\widetilde{\boldsymbol{p}}^{(k_v)}$ approach a common limit $\boldsymbol{x}$ as $v \to \infty$, it holds that

$$\begin{aligned} &f(\boldsymbol{r}^{(k_v)}) - \beta(\mathcal{M}^{k_v}) \\ &= \sum_{i=1}^{L} w_i \overline{\mathrm{EE}}_i(r_i^{(k_v)}, \mathcal{M}^{k_v}) - \sum_{i=1}^{L} w_i \overline{\mathrm{EE}}_i(\widetilde{p}_i^{(k_v)}, \mathcal{M}^{k_v}) \to 0, \end{aligned}$$

and, thus, $\beta(\mathcal{M}^{k_v}) \to f(\boldsymbol{x})$. This point is feasible for (P1) and, hence, $\beta(\mathcal{M}^{k_v}) \ge f(\boldsymbol{x})$. Therefore, $\beta(\mathcal{M}^{k_v}) \to f(\boldsymbol{x}^*)$ with $\boldsymbol{x}^*$ the optimal solution of (P1) as $v \to \infty$. ∎

The following remarks are in order.

Remark 1: Inspecting Algorithm 1, it can be seen that each step towards the computation of the optimal power allocation is continuous with respect to the channel parameters $\{\alpha_i, \beta_{i,j}\}_{i,j}$, with the exception of (15), which has a discontinuity in $\alpha_i = 0$. Therefore, in order to claim the continuity of the map (26), which will be required by Proposition 3, it is necessary to assume that α_i is bounded below by a strictly positive quantity, i.e., $\alpha_i > \omega_i$, for some $\omega_i > 0$ and $i = 1, \ldots, L$. This assumption does not seem restrictive for any practical system, recalling that α_i is the channel-to-noise ratio between the i-th UE and the associated BS.

Remark 2: If it is desired to use a relative tolerance instead of an absolute tolerance, it is sufficient to replace "$\beta(\mathcal{M}) > \gamma + \varepsilon$" by "$\beta(\mathcal{M}) > (1 + \varepsilon)\gamma$" in line 8 of Algorithm 1.

Remark 3: The novel part about Algorithm 1 is the bounding approach. It can be modified to include Quality of Service (QoS) constraints by introducing suitable feasibility checks. Please refer to [23], [24], [27] for detailed discussions and examples.

A. Application to other Energy Efficiency Metrics

This section explicitly shows how to apply the proposed framework to other performance metrics. It was already observed in Section II-A that the WSEE generalizes the system weighted sum-rate defined in (9), and thus the proposed approach naturally applies to the maximization of the weighted sum-rate with $\widetilde{\boldsymbol{p}}^{(k)} = \boldsymbol{s}^{(k)}$ as can be easily seen from (12).

Instead, the application of our framework to other EE metrics requires more effort. Specifically, we will consider the WPEE, WMEE, and GEE functions, as defined in (6), (7), and (4).

1) WPEE and WMEE maximization: Let us consider the WPEE function. The main issue regarding the applicability of the proposed BB procedure is the derivation of a bound for the WPEE over the generic hyper-rectangle $\mathcal{M}^k$, that is tight, satisfies (18), and is simple to maximize. To this end, observe that we can write

$$\begin{aligned}
&\max_{\boldsymbol{p}\in\mathcal{M}^k} \prod_{i=1}^{L} \frac{\log\left(1+\frac{\alpha_i p_i}{1+\sum_{j\neq i}\beta_{i,j}p_j}\right)}{\mu_i p_i + P_{c,i}} \\
&\quad\leq \prod_{i=1}^{L} \max_{\boldsymbol{p}\in\mathcal{M}^k} \frac{\log\left(1+\frac{\alpha_i p_i}{1+\sum_{j\neq i}\beta_{i,j}p_j}\right)}{\mu_i p_i + P_{c,i}} \\
&\quad= \prod_{i=1}^{L} \max_{r_i^{(k)}\leq p_i\leq s_i^{(k)}} \frac{\log\left(1+\frac{\alpha_i p_i}{1+\sum_{j\neq i}\beta_{i,j}r_j^{(k)}}\right)}{\mu_i p_i + P_{c,i}} \qquad (19) \\
&\quad=: \beta(\mathcal{M}^k). \qquad (20)
\end{aligned}$$

Note that, for any $i=1,\ldots,L$, the upper-bound $\beta(\mathcal{M}^k)$ in (19) coincides with that obtained for the WSEE function in (12). Thus, the same approach used for WSEE maximization applies also to the maximization of the WPEE.

Moreover, it can be seen that the same bounding technique applies also to the WMEE function. Indeed, all steps above can be made also if the product of the EEs is replaced by the minimum of the EEs. Indeed, both are increasing functions of the individual EEs and no differentiability assumption is required by the proposed BB procedure.

2) GEE Maximization: The GEE function does not explicitly depend on the individual EEs, which slightly complicates the bounding technique. Nevertheless, a similar approach applies. Defining $P_c=\sum_{i=1}^{L}P_{c,i}$, we have

$$\begin{aligned}
&\max_{\boldsymbol{p}\in\mathcal{M}^k} \frac{\sum_{i=1}^{L}\log\left(1+\frac{\alpha_i p_i}{1+\sum_{j\neq i}\beta_{i,j}p_j}\right)}{P_c+\sum_{i=1}^{L}\mu_i p_i} \\
&\quad\leq \sum_{i=1}^{L} \max_{\boldsymbol{p}\in\mathcal{M}^k} \frac{\log\left(1+\frac{\alpha_i p_i}{1+\sum_{j\neq i}\beta_{i,j}p_j}\right)}{P_c+\sum_{i=1}^{L}\mu_i p_i} \\
&\quad= \sum_{i=1}^{L} \max_{s_i^{(k)}\leq p_i\leq r_i^{(k)}} \frac{\log\left(1+\frac{\alpha_i p_i}{1+\sum_{j\neq i}\beta_{i,j}r_j^{(k)}}\right)}{\mu_i p_i + P_c + \sum_{j\neq i}\mu_j r_j^{(k)}} \qquad (21) \\
&\quad=: \beta(\mathcal{M}^k), \qquad (22)
\end{aligned}$$

where the last step is due to $\frac{\log\left(1+\frac{\alpha_i p_i}{1+\sum_{j\neq i}\beta_{i,j}p_j}\right)}{P_c+\sum_{\ell=1}^{L}\mu_\ell p_\ell}$ being decreasing[3] in p_j for all $j\neq i$. Although slightly different from that obtained for the WSEE, WPEE, and WMEE, the bound in (21) is formally equivalent to (12) as a function of p_i. Thus, the same maximization procedure applies to GEE maximization, too.

[3]This can be verified easily from the first-order derivative.

IV. First-order Optimal Power Control

This section is devoted to developing a power control algorithm with guaranteed convergence to a first-order optimal point of Problem (P1). Besides providing an alternative power control approach with polynomial complexity, the method developed here also provides a theoretically solid benchmark for other approaches. The algorithm proposed in this section is inspired by the successive pseudo-concave framework from [28], which tackles the maximization of a function f by maximizing a sequence of approximate functions $\{\widetilde{f}_t\}_t$ fulfilling the following properties for all j:

1) $\widetilde{f}_t(\boldsymbol{p};\boldsymbol{p}^{(t)})$ is pseudo-concave in $\boldsymbol{p}$ for any feasible $\boldsymbol{p}^{(t)}$.
2) $\widetilde{f}_t(\boldsymbol{p};\boldsymbol{p}^{(t)})$ is continuously differentiable in $\boldsymbol{p}$ for any feasible $\boldsymbol{p}^{(t)}$ and continuous in $\boldsymbol{p}^{(t)}$ for any feasible $\boldsymbol{p}$.
3) $\nabla_{\boldsymbol{p}}\widetilde{f}_t(\boldsymbol{p}^{(t)};\boldsymbol{p}^{(t)})=\nabla_{\boldsymbol{p}}f(\boldsymbol{p}^{(t)})$.
4) $\widetilde{f}_t(\boldsymbol{p};\boldsymbol{p}^{(t)})$ has a non-empty set of maximizers in the feasible set.
5) Given any convergent subsequence $\{\boldsymbol{p}^{(t)}\}_{t\in\mathcal{T}}$ where $\mathcal{T}\subseteq\{1,2,\ldots\}$, the sequence $\{\arg\max_{\boldsymbol{p}\in\mathcal{P}}\ \widetilde{f}_t(\boldsymbol{p};\boldsymbol{p}^{(t)})\}_{t\in\mathcal{T}}$ is bounded.

It is seen that the approximate functions are parametrized by $\boldsymbol{p}^{(t)}$ and the properties above need to hold for any feasible $\boldsymbol{p}^{(t)}$. In practice, $\boldsymbol{p}^{(t)}$ is updated according to a specific rule after each iteration, as will be explained in the sequel.

Under the assumptions above, [28] shows that every limit point of the sequence $\boldsymbol{p}_t^*$ of maximizers of $\widetilde{f}_t$ with respect to $\boldsymbol{p}$, converges to a first-order optimal point for the original problem of maximizing f. Moreover, due to the first property above, for all t, the maximization of $\widetilde{f}_t$ can be accomplished with polynomial complexity by standard optimization methods [29].[4] Thus, the crucial point when employing the above framework is about finding suitable approximate functions $\{\widetilde{f}_t\}_t$ that fulfill all above assumptions. Next, we develop such approximate functions for the objective $f(\boldsymbol{p})$ of the WSEE maximization Problem (P1).

Consider the function

$$\begin{aligned}
\widetilde{\mathrm{EE}}_i(p_i;\boldsymbol{p}^{(t)}) &= w_i\frac{R_i(p_i,\boldsymbol{p}_{-i}^{(t)})}{\mu_i p_i^{(t)}+P_{c,i}} + (p_i-p_i^{(t)}) \\
&\quad\cdot\left(w_i\frac{-\mu_i R_i(\boldsymbol{p}^{(t)})}{(\mu_i p_i^{(t)}+P_{c,i})^2} + \sum_{j\neq i} w_j\frac{\frac{\partial}{\partial p_i}R_j(\boldsymbol{p}^{(t)})}{\mu_i p_i^{(t)}+P_{c,i}}\right), \qquad (23)
\end{aligned}$$

wherein $R_i(p_i,\boldsymbol{p}_{-i}^{(t)})$ denotes the i-th user's rate as a function of the i-th user's power p_i, while all other powers are fixed to $\boldsymbol{p}_{-i}^{(t)}=\{p_j^{(t)}\}_{j\neq i}$. The idea behind this function is to preserve the concave part of EE_i in $f(\boldsymbol{p})$ and linearize it in all other terms and variables with respect to p_k at $\boldsymbol{p}=\boldsymbol{p}^{(t)}$. Combining these functions into $\widetilde{f}_t$ as

$$\widetilde{f}_t=\sum_{i=1}^{L}\widetilde{\mathrm{EE}}_i(p_i;\boldsymbol{p}^{(t)}) \qquad (24)$$

results in an approximate function for $f(\boldsymbol{p})$ that satisfies the properties in 1) – 5) above. This is formally established in

[4]Recall that pseudo-concave functions are differentiable by definition and every stationary point is a global maximizer.

Appendix B. We obtain the approximate problem

$$\begin{cases} \max\limits_{\boldsymbol{p}} & \sum\limits_{i=1}^{L} \widetilde{\mathrm{EE}}_i(p_i;\boldsymbol{p}^{(t)}) \\ \text{s.t.} & 0 \le p_i \le P_i, \quad \text{for all } i=1,2,\ldots,L, \end{cases} \tag{P2}$$

which is a concave maximization problem due to $\widetilde{\mathrm{EE}}_i(p_i;\boldsymbol{p}^{(t)})$ being concave in p_i and constant in $\boldsymbol{p}_{-i}$. It can be solved in polynomial time using standard convex optimization tools [30]. Moreover, since $\widetilde{\mathrm{EE}}_i$ depends only on p_i for all i, Problem (P2) can be decoupled over the users, and each transmit power p_i can be optimized separately by solving the scalar problem:

$$\begin{cases} \max\limits_{p_i} & \widetilde{\mathrm{EE}}_i(p_i;\boldsymbol{p}^{(t)}) \\ \text{s.t.} & 0 \le p_i \le P_i, \end{cases} \tag{P3}$$

Thus, by the successive pseudo-concave optimization framework, the original Problem (P1) is tackled by solving a sequence of problems of the form of (P2), updating the point $\boldsymbol{p}^{(t)}$ after each iteration according to the formula

$$\boldsymbol{p}^{(t+1)} = \boldsymbol{p}^{(t)} + \gamma^{(t)}(\mathbb{B}\boldsymbol{p}^{(t)} - \boldsymbol{p}^{(t)}) \,, \tag{25}$$

with $\mathbb{B}\boldsymbol{p}^t$ an optimal solution of (P2) and $\gamma^t = \beta^{m_t}$ to be determined by the Armijo rule, where m_t is the smallest nonnegative integer such that

$$f(\boldsymbol{p}^{(t)} + \beta^{m_t}(\mathbb{B}\boldsymbol{p}^{(t)} - \boldsymbol{p}^{(t)})) \\ \ge f(\boldsymbol{p}^{(t)}) + \alpha\beta^{m_t}\nabla f(\boldsymbol{x}^{(t)})^T(\mathbb{B}\boldsymbol{p}^{(t)} - \boldsymbol{p}^{(t)})$$

with $0 < \alpha < 1$ and $0 < \beta < 1$ being scalar constants.[5]

The overall procedure is summarized in Algorithm 2, whose convergence is formally stated below.

Proposition 2: Any limit point of $\{\boldsymbol{p}^{(t)}\}_t$ obtained by Algorithm 2 is a stationary point of (P1).

Proof: Please refer to the Appendix. ■

Algorithm 2 Successive convex approximation algorithm

1: Initialize $t = 0$, $\boldsymbol{p}^{(0)} \in [\boldsymbol{0}, \boldsymbol{P}]$, $\alpha, \beta \in (0,1)$.
2: **repeat**
3: $(\mathbb{B}\boldsymbol{p}^{(t)})_i \leftarrow \arg\max_{0\le p_i\le P_i} \widetilde{r}_i(p_i;\boldsymbol{p}^{(t)}), \quad i=1,\ldots,L.$
4: $\gamma^{(t)} \leftarrow 1$
5: **while** $f(\boldsymbol{p}^{(t)} + \gamma^{(t)}(\mathbb{B}\boldsymbol{p}^{(t)} - \boldsymbol{p}^{(t)})) < f(\boldsymbol{p}^{(t)}) + \alpha\gamma^{(t)}\nabla f(\boldsymbol{p}^{(t)})^T(\mathbb{B}\boldsymbol{p}^{(t)} - \boldsymbol{p}^{(t)})$ **do**
6: $\gamma^{(t)} \leftarrow \beta\gamma^{(t)}$
7: **end while**
8: $\boldsymbol{p}^{(t+1)} \leftarrow \boldsymbol{p}^{(t)} + \gamma^{(t)}(\mathbb{B}\boldsymbol{p}^{(t)} - \boldsymbol{p}^{(t)})$
9: $t \leftarrow t+1$
10: **until** convergence.

Remark 4: QoS constraints can be incorporated into Algorithm 2 by employing standard techniques. Please refer, e.g., to [14] and [32].

V. ANN-Based Power Control

While the global optimization method from Section III is aimed at providing an efficient way to solve offline Problem (P1), this section proposes an approach that is able to tackle Problem (P1) with a complexity that is amenable to an online implementation, i.e., with a complexity that enables to update the optimal power control vector with the same frequency as the variations of the fading channel realizations. This will be achieved by merging the global optimization method from Section III with an ANN-based procedure. The main idea is based on reformulating the optimization problem (P1) as the problem of determining the map

$$\mathcal{F} : \mathbf{a} = (\alpha_i, \beta_{i,j}, P_i)_{i,j} \in \mathbb{R}^{L(L+1)} \mapsto \boldsymbol{p}^* \in \mathbb{R}^L, \tag{26}$$

with $\boldsymbol{p}^*$ the optimal power allocation corresponding to $\mathbf{a}$. Thus, ANNs can be used to learn the map (26), since ANNs, and in particular feedforward neural networks with fully-connected layers, are universal function approximators [33].

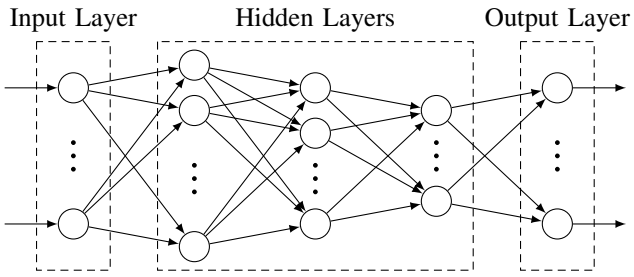

Fig. 1. General scheme of a deep feedforward ANN with fully-connected layers.

A. Network Architecture

Before proceeding further, let us first briefly introduce the architecture of the considered ANN. We employ a fully-connected feedforward network which takes as input a realization of the parameter vector $\mathbf{a}$, producing as output a power allocation vector $\hat{\boldsymbol{p}}$, which estimates the optimal power allocation vector $\mathbf{p}^*$ corresponding to $\mathbf{a}$. Between the input and output layer, K hidden layers are present. For all $k = 1,\ldots,K+1$, the k-th layer has N_k neurons, with neuron n computing

$$\zeta_k(n) = f_{n,k}\big(\boldsymbol{\gamma}_{n,k}^T\boldsymbol{\zeta}_{k-1} + \delta_{n,k}\big) \tag{27}$$

wherein $\boldsymbol{\zeta}_k = (\zeta_k(1),\ldots,\zeta_k(N_{k+1}))$ denotes the $N_{k+1}\times 1$ output vector of layer k, $\boldsymbol{\gamma}_{n,k} \in \mathbb{R}^{N_{k-1}}$ and $\delta_{n,k} \in \mathbb{R}$ are neuron-dependent weights and bias terms, respectively, while $f_{n,k}$ is the activation function[6] of neuron n in layer k. Apparently, each neuron performs quite simple operations. Nevertheless, combining the processing of multiple neurons, ANNs can perform very complex tasks and obtain an overall input-output map that emulates virtually any function. Formally, the following universal approximation result holds [33, Theorem 1].

Proposition 3: Consider a single layer of an ANN with $n\times 1$ input vector $\boldsymbol{\xi}$, $m\times n$ weight matrix $\widetilde{\boldsymbol{\Gamma}}$, $m\times 1$ bias

[5]Please refer to [31, Sec. 1.2.1] for more details on the Armijo rule and how to choose α and β properly.

[6]In principle, any function can be considered as activation function, even though widely accepted choices are sigmoidal functions, rectified linear units (ReLU), and generalized ReLU functions. The specific choices considered in this work are discussed in Section VI.

vector $\widetilde{\boldsymbol{\delta}}$, and activation functions $\boldsymbol{f} = [f_1, \ldots, f_m]$. The set of all input-output maps that can be obtained by the ANN is

$$\mathcal{A}_n = \left\{ \boldsymbol{f}\left(\widetilde{\boldsymbol{\Gamma}}\boldsymbol{\xi} + \widetilde{\boldsymbol{\delta}}\right) \,\middle|\, \widetilde{\boldsymbol{\Gamma}} \in \mathbb{R}^{n\times m}, \widetilde{\boldsymbol{\delta}} \in \mathbb{R}^m \right\} . \tag{28}$$

Then, $\mathcal{A}_n$ is dense in the set of continuous functions if and only if $\boldsymbol{f}$ is not an algebraic polynomial.

Proposition 3 formally proves that the input-output relationship of an ANN can emulate any continuous map.[7] In addition to Proposition 3, [34] provides bounds for the number of neurons to be used to obtain a given approximation accuracy. However, although of great theoretical importance, neither of these results is constructive, in the sense that they do not provide any guidance as to the topology of the ANN to use and how to configure the weights and biases to achieve a desired approximation accuracy. In practice, it has been empirically shown that deep architectures, i.e., ANN with multiple layers, tend to require less neurons [35, Sec. 6.4.1] to achieve the same level of accuracy, which motivates us to employ a deep ANN. Moreover, it is intuitively clear that the number of neurons to employ increases with the size of the problem, i.e., with the dimension of the domain and co-domain of the map to estimate. For the case of (26), it is needed to estimate a map from an $L(L+1)$-dimensional space to an L-dimensional space.

B. Training Procedure, Normalization, and Data Augmentation

At this point, the problem remains of how to tune the weights and biases to reliably estimate (26). To this end, the weights $\boldsymbol{\Gamma} = \{\boldsymbol{\gamma}_{n,k}\}_{n,k}$, and the biases $\boldsymbol{\delta} = \{\delta_{n,k}\}_{n,k}$ are adjusted in a supervised learning fashion by training the ANN. This requires the use of a training set, i.e., a set of the form $\{(\boldsymbol{a}_n, \boldsymbol{p}_n^*) \,|\, n = 1, \ldots, N_T\}$ with N_T training tuples $(\boldsymbol{a}_n, \boldsymbol{p}_n^*)$, wherein $\boldsymbol{p}_n^*$ is the optimal power allocation vector corresponding to $\boldsymbol{a}_n$. In other words, the training set contains examples of desired power allocation vectors corresponding to some possible configurations of system parameters $\boldsymbol{a}_n$. By exploiting these examples, the ANN learns to predict the power allocation also for new realizations of $\boldsymbol{a}_n$ that are not contained in the training set. Mathematically speaking, the training process consists of adjusting the weights and biases of the ANN in order to minimize the loss between actual and desired output, namely considering the problem:

$$\min_{\boldsymbol{\Gamma},\boldsymbol{\delta}} \; \frac{1}{N_T} \sum_{n=1}^{N_T} \mathcal{L}(\hat{\boldsymbol{p}}_n(\boldsymbol{\Gamma}, \boldsymbol{\delta}), \boldsymbol{p}_n^*) \tag{29}$$

with $\mathcal{L}(\cdot,\cdot)$ being any suitable measure of the error incurred when the actual output of the ANN corresponding to the n-th training input is $\hat{\boldsymbol{p}}_n$, while the desired output was $\mathbf{p}_n^*$. A widely-used error measure is the squared error $\|\hat{\boldsymbol{p}}_n(\boldsymbol{\Gamma}, \boldsymbol{\delta}) - \boldsymbol{p}_n^*\|^2$ [35] which is also employed here. The minimization of (29) can be tackled by state-of-the-art, off-the-shelf stochastic gradient descent methods specifically developed for training ANNs [35], and therefore will not be discussed here. Instead, it is interesting to note that the learning process can be simplified by normalizing the transmit powers before running the stochastic gradient descent training algorithm. Specifically, applying the variable change $p_i \to \tilde{p}_i P_i$, for all $i = 1, \ldots, L$, we normalize the transmit power to lie in the interval $[0, 1]$, which leads to the following equivalent reformulation of Problem (P1)

$$\begin{cases} \max\limits_{\tilde{\boldsymbol{p}}} & \displaystyle\sum_{i=1}^{L} w_i \frac{\log\left(1 + \frac{\tilde{\alpha}_i \tilde{p}_i}{1+\sum_{j\neq i} \tilde{\beta}_{i,j}\tilde{p}_j}\right)}{\mu_i P_i \tilde{p}_i + P_{c,i}} \\ \text{s.t.} & 0 \leq \tilde{p}_i \leq 1, \quad \text{for all } i = 1, 2, \ldots, L, \end{cases} \tag{P4}$$

wherein $\tilde{\alpha}_i = \alpha_i P_i$, $\tilde{\beta}_i = \beta_i P_i$, $\tilde{\mu}_i = \mu_i P_i$, for all $i = 1, \ldots, L$. Then, the normalized training set then is $\mathcal{S}_T = \{(\tilde{\boldsymbol{a}}_n, \tilde{\boldsymbol{p}}_n^*) \,|\, n = 1, \ldots, N_T\}$ with parameter vector $\tilde{\boldsymbol{a}} = (\tilde{\alpha}_i, \tilde{\beta}_{i,j}, P_i)_{i,j}$. The advantage of this reformulation is that, despite the values $P_1, \ldots, P_L$, the transmit powers are always in the set $[0,1]$. Intuitively, this simplifies the dependence of the optimal power allocation on the maximum power constraints, thereby making it easier for the ANN to grasp the optimal power allocation structure as a function of the maximum power constraints.

When doing this, the use of realistic numbers for the receive noise power and propagation channels might lead to coefficients $\{\tilde{\alpha}_i, \tilde{\beta}_{i,j}\}_{i,j}$ with quite a large magnitude, which often cause numerical problems to the stochastic gradient descent training algorithm. We have observed that this issue is solved by expressing the parameter vectors $\tilde{\boldsymbol{a}}$ in the training set in logarithmic units rather than in a linear scale. A similar problem occurs for the output powers, which in some cases might be close to zero due to the normalization by $P_{\max}$. This issue is also resolved by expressing the output powers in logarithmic scale. On the other hand, logarithms cause numerical problems when the optimal transmit powers are very close to zero. In order to avoid this issue, a suitable approach is to clip logarithmic values approaching $-\infty$ at $-M$ for $M > 0$. Thus, the considered normalized training set is

$$\mathcal{S}_T = \{(\log_{10} \tilde{\boldsymbol{a}}_n, \max\{-M, \log_{10} \tilde{\boldsymbol{p}}_n^*\}) \,|\, n = 1, \ldots, N_T\}.$$

Additionally, we can use the fact that the problem is invariant under permutation of the users for data augmentation, i.e., we can increase the size of the training set $\mathcal{S}_T$ during training. In order to do this, the rows and columns of a channel matrix (and the corresponding power allocations) from the training set can be permuted to generate a new training sample. In mathematical terms, given a permutation σ of the index set $\mathcal{I} = \{1, 2, \ldots, L\}$, the elements of the new matrix $\tilde{H}$ are given as $\tilde{h}_{i,j} = h_{\sigma(i),\sigma(j)}$ by permuting the indices of a channel matrix H from the training set. In each training step, a new random permutation is generated and used to permute the training samples and corresponding labels. Therefore, the ANN automatically learns the invariance of the problem against permutation of the users.

An important aspect for good training performance is the choice of the output layer where the proposed ANN deploys a linear activation function. This seems to contrast with the fact that the transmit powers need to be constrained in the interval $[0, 1]$. However, enforcing this constraint directly in the output activation function might mislead the ANN. Indeed, it could lead to low training errors simply thanks to the use of cut-off levels in the activation function, instead of being the result of proper adjustment of the hidden layer weights and biases.

[7] The continuity of (26) has been analyzed in Section III.

In this case, the ANN would not be able to learn that the training and validation errors are acceptable only because the desired power level is close to either 1 or 0, and the clipping at the output layer provides by construction such a power level, regardless of the configuration adopted in the hidden layers. Instead, a linear output activation function allows the ANN to learn whether the present configuration of weights and biases is truly leading to a small error. At the end of the training phase, the output variables are clipped to the interval $[0, 1]$.

After the training phase, all weights and biases of the ANN are configured and the ANN essentially provides a closed-form estimate of the map (26). Indeed, once the weights and biases have been set, the input-output relationship of the ANN can be written in closed-form as the composition of the affine combinations and activation functions of the neurons in the ANN. This effectively provides a closed-form expression for the map (26), within an approximation accuracy that can be made small at will by properly designing and training the ANN. Thus, as $\mathbf{a}$ changes due to channel fading, the corresponding power allocation can be obtained without having to solve Problem (P1) again, but simply computing the output of the ANN when the input is the new realization of $\mathbf{a}$. This grants a large complexity reduction during the online operation of the method, as compared to other approaches that employ iterative methods where several convex problems need to be solved for each instance of the channel realizations. This point is analyzed in more detail in the next section.

C. Computational Complexity

The main advantage of the proposed ANN-based method is that it allows performing most of the computations towards solving (P1) offline, leaving only a few operations to be executed online, i.e., only a few operations need to be repeated when the system channel realizations vary, while most of the computations need to be performed only sporadically. This is in contrast to available online power control methods based on the traditional use of optimization theory, which need to be run from scratch every time one or more system channel realizations have changed. To elaborate, the complexity of the proposed ANN-based power allocation can be divided into an online and an offline complexity, as explained next:

(a) **Online complexity.** This is the complexity that is required to use the trained ANN for the online computation of the power allocation vector. As discussed below, this is the complexity that is incurred during the online operation of the method, i.e., when the trained ANN is being used to output the optimal power control policy following the variations of the channel fading realizations.

(b) **Offline complexity.** This is the complexity that is required to build the training set and to implement the training procedure. As discussed below, these tasks can be executed at a much longer time-scale than that with which the channel fading realizations change.

Online phase. It requires computing the output $\zeta_k(n)$ of each neuron in the ANN, moving from the first layer to the output layer, which in turn requires $\sum_{k=1}^{K+1} N_{k-1}N_k$ real multiplications,[8] and evaluating $\sum_{k=1}^{K+1} N_k$ scalar activation functions $f_{n,k}$. Despite being typically non-linear, the activation functions are elementary functions whose computation does not pose any significant computational issue. Thus, with respect to the number L of active users' in the network, the exact complexity of computing a forward propagation scales with L^2, since the variable L appears in the size of the first and last layer, for which we have $N_0 = L(L+1)$ and $N_{K+1} = L$. Thus, obtaining the output of the trained ANN for any given input vector entails a negligible complexity, since it requires only the computation of a forward propagation of the trained ANN. For this reason, the online complexity of the proposed ANN-based method is much lower than the complexity of the first-order optimal method from Section IV, which instead requires solving convex problems in each iteration. With this respect, we observe that while there are no formulas that provide the exact computational complexity of a convex problem, a popular results from [36] estimates that the asymptotic complexity of convex problem scales in general with the fourth power of the number of variables, i.e. L^4. Moreover, the first-order optimal method from Section IV requires solving a sequence of convex problems, and thus its overall complexity will be much higher than that of computing a simple forward propagation in the ANN.

Offline phase. It requires the generation of the training set and its use to train the ANN. Among these two tasks, the most complex is the generation of the training set, since the execution of the training algorithm is conveniently performed by off-the-shelf stochastic gradient descent algorithms, which ensure a fast convergence. Moreover gradient computation is performed by the backpropagation algorithm, which further reduces the computational complexity [35]. Instead, generating the training set requires actually solving the NP-complete Problem (P1) N_T times, i.e., for N_T different realizations of the system parameter vector $\mathbf{a}$. At a first sight, this might seem to defeat the purpose of using the proposed ANN-based approach, but actually this is not the case for three main reasons:

- The whole training phase (including both generation and use of the training set) can be performed *offline*. Thus, a much higher complexity can be afforded, and it is not needed to complete the training process within the channels coherence time.
- The training set can be updated at a *much longer time-scale* than the channels coherence time. In other words, the training set can be updated sporadically compared with the frequency with which Problem (P1) should be solved if traditional optimization approaches were used.
- Despite the first two points, it can be argued that Problem (P1) is NP-complete, and thus generating a large training set appears a daunting task even if it can be performed offline. However, the global optimization method that is proposed in Section III eases this issue, making it possible to globally solve (P1) in practical wireless networks, with a complexity that is affordable for offline implementations.

[8]The complexity related to additions is neglected as it is much smaller than that required for multiplications.

Finally, we explicitly observe that, as anticipated, the proposed ANN-based approach is not restricted to the maximization of the WSEE. Indeed, any power allocation problem can be cast as in (26) and the BB method developed in Section III is not limited to the maximization of the WSEE, but encompasses all major energy-efficient metrics, as addressed in detail in Section III-A.

VI. Numerical Evaluation

We consider the uplink of a wireless interference network. At first, we consider that $L = 4$ interfering single-antenna UEs are placed in a square area with edge 2 km and communicate with 4 access points placed at coordinates $(0.5, 0.5)$ km, $(0.5, 1.5)$ km, $(1.5, 0.5)$ km, $(1.5, 1.5)$ km, and equipped with $n_R = 2$ antennas each. It should be stressed that the parameter L does not denote the total number of active users in the network, but rather the number of users sharing the same resource block, i.e. the parameter L denotes the network resource reuse factor. The total number of active users in the considered four-cell network is obtained multiplying L by the total number of orthogonal resource blocks employed by the network operator. Moreover, it can be seen that the uplink power control problem considered in this work decouples over orthogonal resource blocks, thus implying that all of the optimization methods developed in this work can be applied in parallel to different orthogonal resource blocks.

The path-loss is modeled following [37], with carrier frequency 1.8 GHz and power decay factor equal to 4.5, while fast fading terms are modeled as realizations of zero-mean, unit-variance circularly symmetric complex Gaussian random variables. The circuit power consumption and power amplifier inefficiency terms are equal to $P_{c,i} = 1\,\mathrm{W}$ and $\mu_i = 4$ for all $i = 1, \ldots, L$, respectively. The noise power at each receiver is generated as $\sigma^2 = F\mathcal{N}_0 B$, wherein $F = 3\,\mathrm{dB}$ is the receiver noise figure, $B = 180\,\mathrm{kHz}$ is the communication bandwidth, and $\mathcal{N}_0 = -174\,\mathrm{dBm/Hz}$ is the noise spectral density. All users have the same maximum transmit powers $P_1 = \cdots = P_L = P_{\max}$.

The proposed ANN-based solution of Problem (P1) is implemented through a feedforward ANN with $K + 1$ fully-connected layers, with the $K = 5$ hidden layers having 128, 64, 32, 16, 8 neurons, respectively. In order to generate the training set, Problem (P1) needs to be solved for different realizations of the vector $\tilde{\boldsymbol{a}} = (\tilde{\alpha}_i, \tilde{\beta}_{i,j}, P_{\max})_{i,j}$. The data is converted to logarithmic units as explained in Section V-B with clipping at $-M = -20$.[9] As for the activation functions, ReLU and its generalizations are the most widely used choice. Our experiments verify that they also perform well in this application. Specifically, the first hidden layer has an exponential linear unit (ELU) activation, motivated by the need to compensate for the logarithmic conversion in the training set. This choice, together with the logarithmic normalization of the data set, has proven itself essential for good training performance. The other hidden layers alternate ReLU and ELU activation functions while the output layer deploys a linear activation function (cf. Section V-B).

[9]Note that, although using a logarithmic scale, the transmit powers are not expressed in dBW, since the logarithmic values are not multiplied by 10. Thus $-M = -20$, corresponds to -200 dBW.

A. Training Performance

The ANN is implemented in Keras 2.2.4 [38] with TensorFlow 1.12.0 [39] as backend. Training is performed on a Nvidia GeForce GTX 1080 Ti over 500 epochs with batches of size 128 and shuffling of the training data before each epoch. Initialization is performed by Keras with default parameters, i.e., Glorot uniform initialization [40] for the kernel and zero biases. The optimization problem (29) is solved by the Adam optimizer with Nesterov momentum [41], initialized by Keras default parameters, with the squared error as the loss function in (29). Source code and data sets are available online [42].

The training set is generated from 2000 independent and identically distributed (i.i.d.) realizations of UEs' positions and propagation channels. Users are randomly placed in the service area and channels are generated according to the channel model described above. Each UE i is associated to the access point towards which it enjoys the strongest effective channel α_i. For each channel realization, we apply Algorithm 1 to solve (P1) for $P_{\max} = -30, \ldots, 20\,\mathrm{dB}$ in 1 dB steps with relative tolerance $\varepsilon = 0.01$. This yields a training set of 102,000 samples.

Besides the training set, also a validation set and a test set are required. The validation set is used during training to estimate the generalization performance of the ANN, i.e., the performance on a data set the ANN was not trained on. The validation loss is the central metric for hyperparameter tuning, i.e., choosing all parameters of the ANN other than weights and biases (e.g. number of layers, activation functions, batch size). Since during this process information about the validation set leaks into the ANN model, another set for the final testing of the ANN is required, the test set. It is essential that the test set is never used during training and tuning of the ANN [43]. The validation and test sets have been independently generated from 200 and 10,000 i.i.d. channel realizations, respectively, with the same procedure used for the training set. This results in 10,200 samples for the validation set, i.e., 10% of the training set, and 510,000 samples for the test set. The final performance to be shown in Section VI-B will be averaged over the test set samples. Thus, using a test set based on 10,000 channel scenarios means that the performance presented in Section VI-B is what is obtained by using the trained ANN for 10,000 channel coherence times. This confirms that the training phase needs to be performed only sporadically.

Considering training, validation, and test sets, 622,200 data samples were generated, which required solving the NP-complete problem (P1) 622,200 times. This has been accomplished by the newly proposed BB method developed in Algorithm 1, which has been implemented in C++ employing the Intel® MKL, OpenMP, and the Lambert W library published at https://github.com/CzB404/lambert_w. Computing the complete data set (622,200 samples including training, validation, and test sets) took 8.4 CPU hours on Intel Haswell nodes with Xeon E5-2680 v3 CPUs running at 2.50 GHz. The mean and median times per sample are 48.7 ms and 4.8 ms, respectively, which shows the effectiveness of the proposed

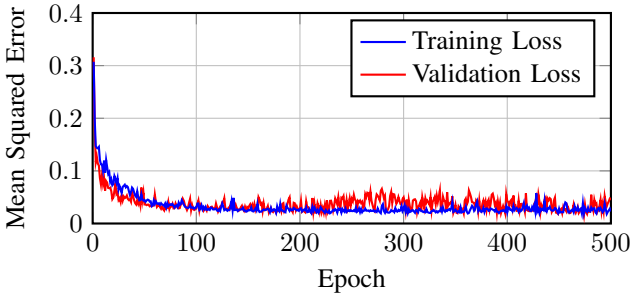

Fig. 2. Training and validation loss.

Algorithm 1, and in turn supports the argument that the offline generation of a suitable training set for the proposed ANN-based power control method is quite affordable.

Due to the random initialization, shuffling of the training data, and the inherent randomness of the optimizer, the weights and biases of the ANN are realizations of a random process. Thus, all performance results reported for the ANN are averaged over 10 realizations of the network obtained by training the ANN *on the same training set* with different initialization of the underlying random number generator.[10] The average training and validation losses for the final ANN are shown in Fig. 2. It can be observed that both errors quickly approach a small value of the same order of magnitude as the tolerance of Algorithm 1. Moreover, neither of the losses increases over time which leads to the conclusion, that the adopted training procedure fits the training data well, without underfitting or overfitting.

B. Testing Performance

The average performance of the final ANN on the test set is reported in Fig. 3. Recall that this test set is never used during training and, thus, the ANN has no information about it except for its statistical properties gathered from the training set (and, possibly, the validation set due to hyperparameter tuning). It can be seen from Fig. 3 that the gap to the optimal value is virtually non-existent which is confirmed by the empirical cumulative distribution function (CDF) of the relative approximation error displayed in Fig. 4. Its mean and median values are 0.0133 and 0.00739 respectively.

In addition to near-optimal performance and low computational complexity, the proposed ANN-based approach also outperforms several baseline approaches. Specifically, we have included a comparison with the following benchmarks:

- **SCAos:** The first-order optimal method based on the sequential convex approximation method developed in Section IV. For each value of $P_{\max}$, the algorithm initializes the transmit power to $p_i = P_{\max}$, for all $i = 1, \ldots, L$.
- **SCA:** This is again the first-order optimal method based on sequential convex approximation developed in Section IV, but in this case a double-initialization approach is used. Specifically, at $P_{\max} = -30\,\mathrm{dBW}$ once again maximum power initialization is used. However, for all values of $P_{\max} > -30\,\mathrm{dBW}$, the algorithm is run twice, once with the maximum power initialization, and once initializing the transmit powers with the optimal solution obtained for the previous $P_{\max}$ value. Then, the power allocation achieving the better WSEE value is retained.
- **Max. Power:** All UEs transmit with $p_i = P_{\max}$, for all $i = 1, \ldots, L$. This strategy is known to perform well in interference networks for low $P_{\max}$ values.
- **Best only:** Only one UE is allowed to transmit, specifically that with the best effective channel. This approach is motivated for high $P_{\max}$ values, as a naive way of nulling out multi-user interference.

[10]Note that this is not equivalent to *model ensembling* [43, Sect. 7.3.3] or *bagging* [35, Sect. 7.1].

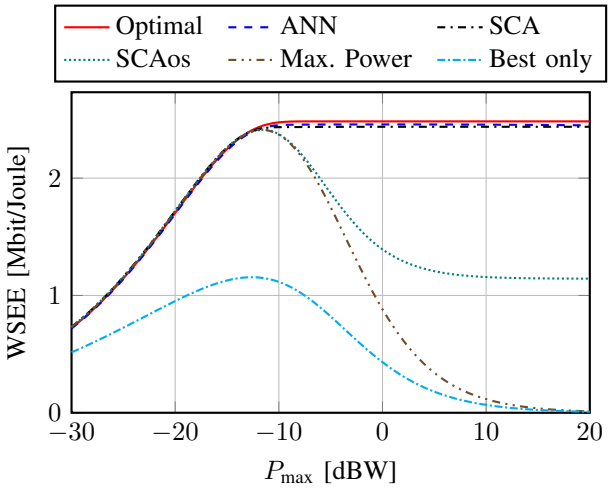

Fig. 3. Performance on the test set compared to the global optimal solution, first-order optimal solutions, and fixed power allocations.

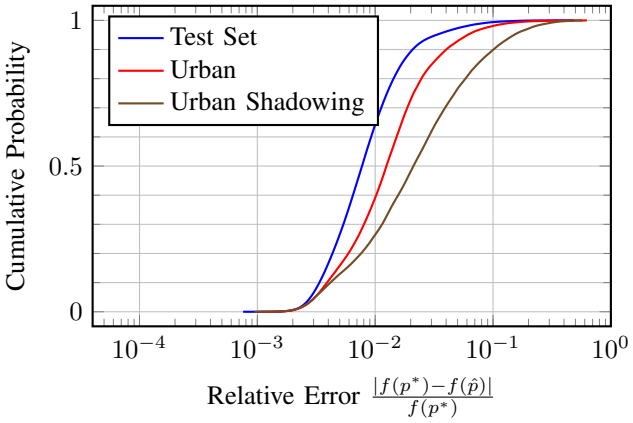

Fig. 4. Empirical CDF of the relative approximation error.

The results show that the proposed ANN-based approach outperforms all other suboptimal schemes. The only exception is the SCA approach which shows similar (but still worse) performance. However, as described above, this method relies on a sophisticated initialization rule, which requires to solve the WSEE maximization problem twice and for the complete range of $P_{\max}$ values. This is clearly not suitable for obtaining a "one-shot" solution, i.e., when the WSEE needs to be maximized only for one specific value of $P_{\max}$, as is required for online resource allocation. Moreover, it requires some calibration depending on the channel statistics since it performs well provided the $P_{\max}$ range starts sufficiently far away from the WSEE saturation region, i.e., the range of $P_{\max}$ values for which the WSEE keeps constant, starting from $P_{\max} \approx -10\,\mathrm{dBW}$ in Fig. 3. Thus, the SCA approach has a quite higher complexity than the ANN-

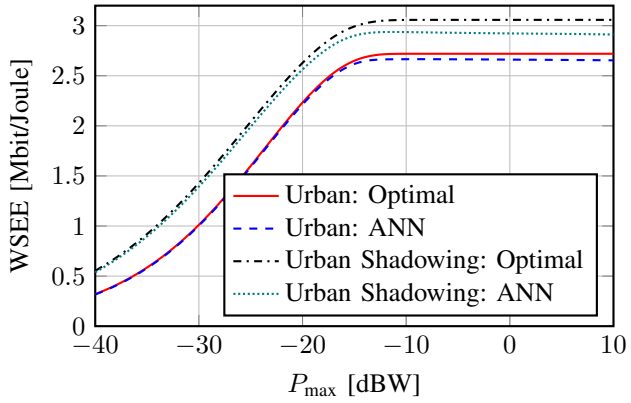

Fig. 5. Performance on different channel distributions: Hata-COST231 Urban propagation model with and without 8 dB shadowing.

based method, but, despite this, it performs slightly worse. In conclusion, we can argue that the ANN approach is much better suited to online power allocation than state-of-the-art approaches, including Algorithm 2.

C. Resilience against Channel Modeling Mismatches

Previous results consider a test set whose samples are independently generated from the training and validation sets, but following the same statistical distribution. Instead, now we analyze how robust the ANN performance is to changes in the channel statistics. To this end, in the following we consider a new test set, whose samples are generated according to a different statistical distribution. Specifically, we generate path-loss effects according to the Hata-COST231 propagation model [44], [45] for urban (non-metropolitan) areas with carrier frequency 1.9 GHz and base station height 30 m. However, we do not repeat the training based on the new channel model, but instead use the same ANN trained as described above. Remarkably, as indicated by the prediction performance reported in Fig. 5 and verified by the distribution of the relative error in Fig. 4 under the label "Urban," the performance degrades only slightly compared to the case in which the test and training samples come from the same distribution.

Next, we further modify the channel generation procedure of the test set, by also introducing log-normal shadowing [45] with 8 dB standard deviation. It can be observed from Fig. 4 that the median relative error increases by roughly half an order of magnitude. Given that the underlying channel distribution is quite different from the trained channel model, the performance can still be considered good. This is also verified by the WSEE performance shown in Fig. 5. Indeed, the performance is still better than with SCAos. Based on these observations, we conclude that training the ANN on synthetic data based on simple channel models is quite robust and performs well also in more sophisticated channels scenarios. Of course, the performance tends to degrade as the mismatch between the training and test set distributions increases.

D. Performance of an ANN with Reduced Size

Next, we evaluate the performance of a much smaller ANN trained with the same data as before. Specifically, we only

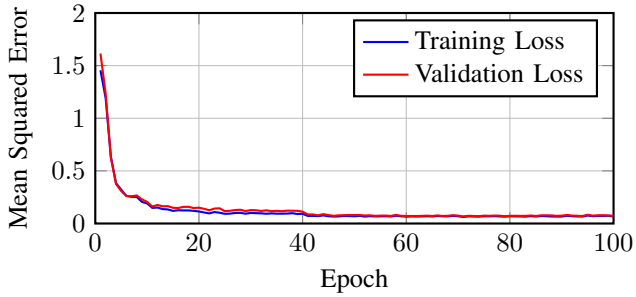

Fig. 6. Training and validation loss of the smaller ANN.

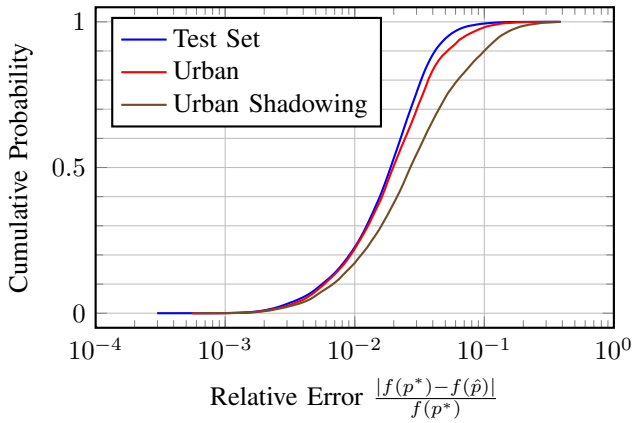

Fig. 7. Empirical CDF of the relative approximation error made by the smaller ANN.

consider 2 hidden layers having 16 and 8 neurons, respectively, with activation functions ELU and ReLU and no permutation of the training data. This further reduces the computational complexity for online resource allocation. Again, the output layer has 4 nodes and a linear activation function. Training is performed in batches of size 128. While the counterpart of Fig. 3 looks identical (and is, therefore, not reproduced), the difference between the two ANNs is best studied from the training loss in Fig. 6 and the distribution of the relative error in Fig. 7. First, observe from Fig. 6 that the training and validation losses stall on a value clearly greater than those of the original (larger) ANN. The CDF of the relative error reflects this as well, being shifted significantly to the right. Still, the mean error on the test set is so small that no difference would be observed in terms of achieved WSEE value. Instead, the performance on the test set generated from the Hata-COST231 Urban model with shadowing differs noticeably from the original ANN as can be seen from Fig. 8. However, the performance can still be considered good. Thus, although the smaller ANN has worse performance than the original, the practical implications are limited and its reduced complexity might be worth the downsides.

E. Increasing the Number of Users

In the previous sections, a scenario with $L = 4$ users has been considered. In this section, we increase the number of users to demonstrate that resource allocation with ANNs can be scaled to a higher number of users and further showcase the performance of the proposed global optimization algorithm. In particular, consider the same scenario as before but with $L = 7$ UEs and $n_R = 4$ antennas per access point.

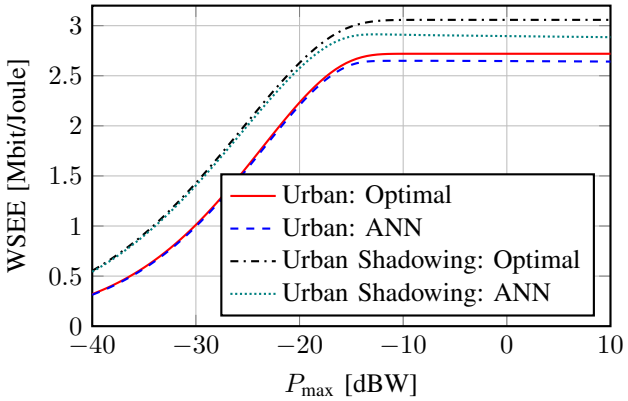

Fig. 8. Performance of the smaller ANN on different channel distributions: Hata-COST231 Urban propagation model with and without 8 dB shadowing.

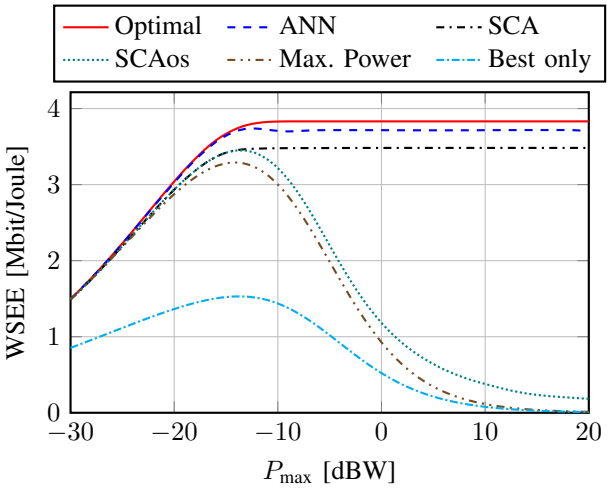

Fig. 9. Performance on the test set compared to the global optimal solution, first-order optimal solutions, and fixed power allocations for the scenario with seven users.

To account for the increased randomness in the data set, the sizes of the ANN and data sets need to be increased. An ANN with nine hidden layers having (1024, 4096, 1024, 512, 256, ..., 16) nodes, respectively, ELU activation functions on the first and last layers, and ReLU in all others layers has shown the best performance in our numerical experiments. The training set has been generated from 6000 i.i.d. channel realizations, while the validation and test sets were generated from 600 and 1000 channels, respectively. This results in a total of 387,600 samples, labelled with Algorithm 1 with mean and median computation times of 260.6 s and 24.1 s per sample, respectively.

Figure 9 shows the performance on the test set compared to the globally optimal solution, first-order optimal solutions and fixed power allocations. Remarkably, the curves look similar to the ones of the smaller problem presented in Fig. 3. In particular, it can be seen that the ANN performs virtually optimal in the low signal-to-noise ratio (SNR) region, outperforming all of the other reference algorithms. At a SNR of approximately -15 dBW the ANN solution is slightly worse than the globally optimal solution and the ANN performs similar to SCA with the two-stage initialization rule discussed in Section VI-B. The results in the previous subsections indicate that this gap can be closed by extended hyperparameter tuning and possibly increasing the size of the training set. Thus, it is possible to increase the size of the considered problem and still use the presented framework to obtain a near-optimal solution.

VII. Conclusions

This work has developed a power control framework for energy-efficient power control in wireless networks. The proposed method is based on a novel branch-and-bound procedure wherein specific bounds for energy-efficient problems are derived, which leads to a much faster convergence than other available global optimization methods. Moreover, this complexity reduction allows to train an ANN using a large dataset of optimal power allocations, which provides a practical power control algorithm, with affordable online complexity. Numerical results have shown that the proposed ANN-based method achieves near-optimal performance, also being robust against mismatches between the training set and the real testing conditions. Moreover, the proposed ANN-based method has a much lower complexity than first-order optimal methods, which tackle power control in interference network by solving a sequence of pseudo-convex relaxations, while at the same time yielding comparable or even better performance. This essentially shows that ANN-based online resource allocation is feasible and a viable method to implement near-optimal power control. However, further work is necessary to bring this proof-of-concept into practise, one important topic being the implementation and strict adherence to QoS constraints.

Appendix A
Derivation of Equation (15)

We derive (15) from (13) and start by substituting $\tilde{\alpha}_i = \frac{\alpha_i}{1+\sum_{j\neq i}\beta_{i,j}r_j^{(k)}}$ in (13). Then,

$$\frac{\tilde{\alpha}_i\mu_i p_i + \tilde{\alpha}_i P_{c,i}}{1+p_i\tilde{\alpha}_i} = \mu_i \ln\left(1+\tilde{\alpha}_i p_i\right) \tag{30}$$

which is equivalent to

$$\frac{\tilde{\alpha}_i P_{c,i}}{\mu_i} - 1 = \left(1+p_i\tilde{\alpha}_i\right)\left[\ln\left(1+\tilde{\alpha}_i p_i\right) - 1\right]. \tag{31}$$

Following [46, p. 337] and [47], we obtain

$$e^{-1}\left(\frac{\tilde{\alpha}_i P_{c,i}}{\mu_i} - 1\right) = e^{\ln(1+p_i\tilde{\alpha}_i)-1}\left[\ln\left(1+\tilde{\alpha}_i p_i\right) - 1\right]. \tag{32}$$

The Lambert W function is defined as the inverse function of xe^x, i.e., $W(xe^x) = x$. Thus,

$$W\left(e^{-1}\left(\frac{\tilde{\alpha}_i P_{c,i}}{\mu_i} - 1\right)\right) = \ln\left(1+\tilde{\alpha}_i p_i\right) - 1. \tag{33}$$

Since $\ln\left(1+\tilde{\alpha}_i p_i\right) \geq 0$, the left-hand side must satisfy $W\left(e^{-1}\left(\frac{\tilde{\alpha}_i P_{c,i}}{\mu_i} - 1\right)\right) \geq -1$. For real numbers, this holds only for the principal branch of the Lambert W function [46, p.330f]. Hence,

$$\exp\left[1 + W_0\left(e^{-1}\left(\frac{\tilde{\alpha}_i P_{c,i}}{\mu_i} - 1\right)\right)\right] = 1+\tilde{\alpha}_i p_i. \tag{34}$$

This result can be further transformed to facilitate efficient computation. From the definition $W(x)e^{W(x)} = x$ [46], follows the identify $e^{W(x)} = \frac{x}{W(x)}$. Then,

$$\exp(1)\frac{e^{-1}\left(\frac{\tilde{\alpha}_i P_{c,i}}{\mu_i} - 1\right)}{W_0\left(e^{-1}\left(\frac{\tilde{\alpha}_i P_{c,i}}{\mu_i} - 1\right)\right)} = 1 + \tilde{\alpha}_i p_i \tag{35}$$

and

$$\frac{1}{\tilde{\alpha}_i}\left(\frac{\frac{\tilde{\alpha}_i P_{c,i}}{\mu_i} - 1}{W_0\left(e^{-1}\left(\frac{\tilde{\alpha}_i P_{c,i}}{\mu_i} - 1\right)\right)} - 1\right) = p_i \tag{36}$$

which is the result in (15).

Appendix B
Proof of Theorem 2

Problem (P1) has a closed convex feasible set and its objective is a proper, continuously differentiable function. Thus, the result follows from [28, Thm. 1] if we can show that (24) satisfies all five technical conditions stated in Section IV.

Conditions 1) and 2) are clearly satisfied, and the boundedness of the feasible set $[\mathbf{0}, \boldsymbol{P}]$ is sufficient for conditions 4) and 5) to hold [28].

Condition 3) is equivalent to

$$\frac{\partial}{\partial p_k}\widetilde{f}_t(\boldsymbol{p};\boldsymbol{p}^{(t)})\big|_{\boldsymbol{p}=\boldsymbol{p}^{(t)}} = \frac{\partial}{\partial p_k} f(\boldsymbol{p})\big|_{\boldsymbol{p}=\boldsymbol{p}^{(t)}}$$

for all $k = 1, \ldots, K$. The left-hand side can be expressed as

$$\begin{aligned}
&\frac{\partial}{\partial p_k}\widetilde{f}_t(\boldsymbol{p};\boldsymbol{p}^{(t)})\big|_{\boldsymbol{p}=\boldsymbol{p}^{(t)}} = \frac{\partial}{\partial p_k}\widetilde{\mathrm{EE}}_i(p_k;\boldsymbol{p}^{(t)})\big|_{\boldsymbol{p}=\boldsymbol{p}^{(t)}} \\
&= \Bigg[\frac{\partial}{\partial p_k}\frac{w_k R_k(p_k, \boldsymbol{p}_{-i}^{(t)})}{\mu_k p_k^{(t)} + P_{c,k}} - \frac{w_k \mu_k R_k(\boldsymbol{p}^{(t)})}{(\mu_k p_k^{(t)} + P_{c,k})^2} \\
&\qquad + \sum_{j\neq k}\frac{w_j \frac{\partial}{\partial p_k} R_j(\boldsymbol{p}^{(t)})}{\mu_j p_j^{(t)} + P_{c,j}}\Bigg]_{\boldsymbol{p}=\boldsymbol{p}^{(t)}} \\
&= w_k\left(\frac{\frac{\partial}{\partial p_k}R_k(\boldsymbol{p}^{(t)})}{\mu_k p_k^{(t)} + P_{c,k}} - \frac{\mu_k R_k(\boldsymbol{p}^{(t)})}{(\mu_k p_k^{(t)} + P_{c,k})^2}\right) + \sum_{j\neq k}\frac{w_j \frac{\partial}{\partial p_k} R_j(\boldsymbol{p}^{(t)})}{\mu_j p_j^{(t)} + P_{c,j}}.
\end{aligned} \tag{37}$$

The right-hand side is

$$\frac{\partial}{\partial p_k} f(\boldsymbol{p})\big|_{\boldsymbol{p}=\boldsymbol{p}^{(t)}} = \sum_i w_i \frac{\partial}{\partial p_k}\mathrm{EE}_i\big|_{\boldsymbol{p}=\boldsymbol{p}^{(t)}}$$

where the partial derivates are

$$\begin{aligned}
\frac{\partial}{\partial p_k}\mathrm{EE}_k(\boldsymbol{p}) &= \frac{\partial}{\partial p_k}\frac{R_k(\boldsymbol{p})}{\mu_k p_k + P_{c,k}} \\
&= \frac{\frac{\partial}{\partial p_k}R_k(\boldsymbol{p})}{\mu_k p_k + P_{c,k}} - \frac{\mu_k R_k(\boldsymbol{p})}{(\mu_k p_k + P_{c,k})^2}
\end{aligned}$$

and, for $i \neq k$, $\frac{\partial}{\partial p_k}\mathrm{EE}_i(\boldsymbol{p}) = \frac{\frac{\partial}{\partial p_k}R_i(\boldsymbol{p})}{\mu_k p_i + P_{c,k}}$. Hence,

$$\begin{aligned}
&\frac{\partial}{\partial p_k} f(\boldsymbol{p})\big|_{\boldsymbol{p}=\boldsymbol{p}^{(t)}} \\
&= w_k\left(\frac{\frac{\partial}{\partial p_k}R_k(\boldsymbol{p}^{(t)})}{\mu_k p_k^{(t)} + P_{c,k}} - \frac{\mu_k R_k(\boldsymbol{p}^{(t)})}{(\mu_k p_k^{(t)} + P_{c,k})^2}\right) + \sum_{j\neq k}\frac{w_j \frac{\partial}{\partial p_k} R_j(\boldsymbol{p}^{(t)})}{\mu_k p_j^{(t)} + P_{c,k}}
\end{aligned}$$

Clearly, this is equivalent to (37) and, thus, Condition 3) is satisfied.

References

[1] B. Matthiesen, Y. Yang, and E. A. Jorswieck, "Optimization of weighted individual energy efficiencies in interference networks," in *Proc. IEEE Wireless Commun. Netw. Conf. (WCNC)*, Barcelona, Spain, Apr. 2018.

[2] B. Matthiesen, A. Zappone, E. A. Jorswieck, and M. Debbah, "Deep learning for real-time energy-efficient power control in mobile networks," in *Proc. IEEE SPAWC*, Cannes, France, Jul. 2019.

[3] "NGMN 5G white paper," NGMN Alliance, Tech. Rep., Mar. 2015.

[4] S. Buzzi, C.-L. I, T. E. Klein, H. V. Poor, C. Yang, and A. Zappone, "A survey of energy-efficient techniques for 5G networks and challenges ahead," *IEEE J. Sel. Areas Commun.*, vol. 34, no. 5, 2016.

[5] A. Zappone and E. Jorswieck, *Energy Efficiency in Wireless Networks via Fractional Programming Theory*, ser. Found. Trends Commun. Inf. Theory. Now Publishers, 2015, vol. 11, no. 3-4.

[6] Z.-Q. Luo and S. Zhang, "Dynamic spectrum management: Complexity and duality," *IEEE J. Sel. Areas Commun.*, vol. 2, no. 1, Feb. 2008.

[7] D. W. K. Ng, E. S. Lo, and R. Schober, "Energy-efficient resource allocation in multi-cell OFDMA systems with limited backhaul capacity," *IEEE Trans. Wireless Commun.*, vol. 11, no. 10, Oct. 2012.

[8] Q. Xu, X. Li, H. Ji, and X. Du, "Energy-efficient resource allocation for heterogeneous services in OFDMA downlink networks: Systematic perspective," *IEEE Trans. Veh. Technol.*, vol. 63, no. 5, June 2014.

[9] J. Xu and L. Qiu, "Energy efficiency optimization for MIMO broadcast channels," *IEEE Trans. Wireless Commun.*, vol. 12, no. 2, Feb. 2013.

[10] S. He, Y. Huang, S. Jin, and L. Yang, "Coordinated beamforming for energy efficient transmission in multicell multiuser systems," *IEEE Trans. Commun.*, vol. 61, no. 12, pp. 4961–4971, Dec. 2013.

[11] B. Du, C. Pan, W. Zhang, and M. Chen, "Distributed energy-efficient power optimization for CoMP systems with max-min fairness," *IEEE Commun. Lett.*, vol. 18, no. 6, pp. 999–1002, 2014.

[12] S. He, Y. Huang, L. Yang, and B. Ottersten, "Coordinated multicell multiuser precoding for maximizing weighted sum energy efficiency," *IEEE Trans. Signal Process.*, vol. 62, no. 3, pp. 741–751, Feb. 2014.

[13] A. Zappone, L. Sanguinetti, G. Bacci, E. A. Jorswieck, and M. Debbah, "Energy-efficient power control: A look at 5G wireless technologies," *IEEE Trans. Signal Process.*, vol. 64, no. 7, pp. 1668–1683, Apr. 2016.

[14] A. Zappone, E. Björnson, L. Sanguinetti, and E. Jorswieck, "Globally optimal energy-efficient power control and receiver design in wireless networks," *IEEE Trans. Signal Process.*, vol. 65, no. 11, Jun. 2017.

[15] R. W. Freund and F. Jarre, "Solving the sum-of-ratios problem by an interior-point method," *J. Global Optim.*, vol. 19, no. 1, 2001.

[16] T. O'Shea and J. Hoydis, "An introduction to deep learning for the physical layer," *IEEE Trans. Cogn. Commun. Netw.*, vol. 3, no. 4, 2017.

[17] M. Chen, U. Challita, W. Saad, C. Yin, and M. Debbah, "Machine learning for wireless networks with artificial intelligence: A tutorial on neural networks," 2017. [Online]. Available: http://arxiv.org/pdf/1710.02913.pdf

[18] H. Sun, X. Chen, Q. Shi, M. Hong, X. Fu, and N. D. Sidiropoulos, "Learning to optimize: Training deep neural networks for interference management," *IEEE Trans. Signal Process.*, vol. 66, no. 20, Oct. 2018.

[19] F. Liang, C. Shen, W. Yu, and F. Wu, "Towards optimal power control via ensembling deep neural networks," 2018, arXiv:1807.10025.

[20] L. A. Zadeh, "Optimality and non-scalar-valued performance criteria," *IEEE Trans. Autom. Control*, vol. 8, no. 1, pp. 59–60, Jan. 1963.

[21] K. Miettinen, *Nonlinear Multiobjective Optimization*. Springer, 1999.

[22] H. Tuy, F. Al-Khayyal, and P. T. Thach, *Essays and Surveys in Global Optimization*. Springer, 2005, pp. 39–78.

[23] B. Matthiesen, C. Hellings, E. A. Jorswieck, and W. Utschick, "Mixed monotonic programming for fast global optimization," *IEEE Trans. Signal Process.*, 2020, accepted for publication. [Online]. Available: https://arxiv.org/abs/1910.07853

[24] B. Matthiesen, "Efficient globally optimal resource allocation in wireless interference networks," Ph.D. Thesis, Technische Universität Dresden, Dresden, Germany, Nov. 2019. [Online]. Available: https://nbn-resolving.org/urn:nbn:de:bsz:14-qucosa2-362878

[25] R. Horst and H. Tuy, *Global Optimization: Deterministic Approaches*, 3rd ed. New York; Berlin, Germany; Vienna, Austria: Springer-Verlag, 1996.

[26] H. Tuy, *Convex Analysis and Global Optimization*. Springer, 2016.

[27] B. Matthiesen and E. A. Jorswieck, "Efficient global optimal resource allocation in non-orthogonal interference networks," *IEEE Trans. Signal Process.*, vol. 67, no. 21, pp. 5612–5627, Nov. 2019.

[28] Y. Yang and M. Pesavento, "A unified successive pseudoconvex approximation framework," *IEEE Trans. Signal Process.*, vol. 65, no. 13, pp. 3313–3328, Jul. 2017.

[29] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge University Press, 2004.
[30] Y. Nesterov and A. Nemirovskii, *Interior-Point Polynomial Algorithms in Convex Programming*. SIAM, 1994.
[31] D. P. Bertsekas, *Nonlinear Programming*, 2nd ed. Athena Sci, 1999.
[32] Y. Yang, M. Pesavento, S. Chatzinotas, and B. Ottersten, "Energy efficiency optimization in MIMO interference channels: A successive pseudoconvex approximation approach," *IEEE Trans. Signal Process.*, vol. 67, no. 15, pp. 4107–4121, Aug. 2019.
[33] M. Leshno, V. Y. Lin, A. Pinkus, and S. Schocken, "Multilayer feedforward networks with a nonpolynomial activation function can approximate any function," *Neural Netw.*, vol. 6, pp. 861–867, 1993.
[34] A. E. Barron, "Universal approximation bounds for superpositions of a sigmoidal function," *IEEE Trans. Inf. Theory*, vol. 39, no. 3, 1993.
[35] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT, 2016.
[36] A. Ben-Tal and A. Nemirovski, *Lectures on Modern Convex Optimization: Analysis, Algorithms, and Engineering Applications*, ser. MPS-SIAM Ser. Optim. Philadelphia, PA, USA: SIAM, 2001.
[37] G. Calcev *et al.*, "A wideband spatial channel model for system-wide simulations," *IEEE Trans. Veh. Technol.*, vol. 56, no. 2, March 2007.
[38] F. Chollet *et al.*, "Keras," https://keras.io, 2015.
[39] M. Abadi *et al.*, "TensorFlow: Large-scale machine learning on heterogeneous systems," https://tensorflow.org, 2015.
[40] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Int. Conf. Artificial Intell. Statis.*, 2010.
[41] I. Sutskever, J. Martens, G. Dahl, and G. Hinton, "On the importance of initialization and momentum in deep learning," in *Int. Conf. Mach. Learn.*, 2013, pp. 1139–1147.
[42] B. Matthiesen, K.-L. Besser, and A. Zappone. (2019) Source code. [Online]. Available: https://github.com/bmatthiesen/deep-EE-opt
[43] F. Chollet, *Deep Learning with Python*. Manning, Nov. 2017.
[44] 3GPP, "Digital cellular telecommunications systems (phase 2+); radio network planning aspects," Tech. Rep. TR 43.030 V9.0.0 R9, 2010.
[45] T. S. Rappaport, *Wirless Communications*, 2nd ed. Prentice-Hall, 2002.
[46] R. M. Corless, G. H. Gonnet, D. E. G. Hare, D. J. Jeffrey, and D. E. Knuth, "On the Lambert$W$ function," *Adv. Comput. Math.*, vol. 5, no. 1, pp. 329–359, Dec. 1996.
[47] C. Isheden, Z. Chong, E. A. Jorswieck, and G. P. Fettweis, "Framework for link-level energy efficiency optimization with informed transmitter," *IEEE Trans. Wireless Commun.*, vol. 11, no. 8, pp. 2946–2957, 8 2012.