EURASIP Journal on Wireless Communications and Networking

RESEARCH Open Access

Deep learning-based optimal placement of a mobile HAP for common throughput maximization in wireless powered communication networks

Hong-Sik Kim and Inwhee Joe*

*Correspondence:
iwjoe@hanyang.ac.kr
Department of Computer Software, Hanyang University, Seoul, Republic of South Korea

Abstract

Hybrid access point (HAP) is a node in wireless powered communication networks (WPCN) that can distribute energy to each wireless device and also can receive information from these devices. Recently, mobile HAPs have emerged for efficient network use, and the throughput of the network depends on their location. There are two kinds of metrics for throughput, that is, sum throughput and common throughput; each is the sum and minimum value of throughput between a HAP and each wireless device, respectively. Likewise, two types of throughput maximization problems can be considered, sum throughput maximization and common throughput maximization. In this paper, we focus on the latter to propose a deep learning-based methodology for common throughput maximization by optimally placing a mobile HAP for WPCN. Our study implies that deep learning can be applied to optimize a complex function of common throughput maximization, which is a convex function or a combination of a few convex functions. The experimental results show that our approach provides better performance than mathematical methods for smaller maps.

Keywords: Hybrid access point (HAP), Wireless powered communication network (WPCN), Deep learning, Mobile, Optimal placement

1 Introduction

In WPCN, there is Access Point (AP) mechanism [1] which contains energy nodes (ENs), wireless devices (WDs) and access points (APs). First, energy nodes send energy to each wireless device. When ENs receive the energy, it sends information to the APs using the energy. That is, ENs send energy to the WDs, and these WDs send information to the APs. We can encapsulate an AP and an EN into a Hybrid Access Point (HAP) and so can describe HAP mechanism. In this mechanism, the HAP sends energy to each WD, and each WD sends information to the HAP. The HAP allocates time slots for sending energy to each WD and itself, and for sending information to each WD, so time allocation for itself and each WD is also an important issue.

Springer Open

© The Author(s), 2021. **Open Access** This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit http://creativecommons.org/licenses/by/4.0/.

Because the distance between the HAP and each WD is different among each WD, there is an energy efficiency gap between the WDs caused by the difference of throughput for each WD. That is, a WD near to the HAP receives more energy from the HAP and uses less energy to transmit information, and another WD far from the HAP receives less energy but uses more energy to transmit information. To solve this unfairness problem, the worst case, a WD which receives the least energy and uses the most energy, is very important. In this case, we use the concept of common throughput which is the minimum value of throughput among the throughput values of each WD, and we concentrate on maximizing the common throughput value in the WPCN environment.

In [2], Bi and Zhang researched the placement optimization of energy and information access points in WPCN using the bi-section search method, Greedy algorithm, Trial-and-error method and alternating method for joint AP-EN placement. There can be more than 1 HAPs in the supposed WPCN environment of this paper. Its methodology repeatedly adds HAPs and check if each WD satisfies conditions in the environment.

Normally, mathematical methodologies are suitable to solve optimization problems by minimizing relatively simple functions. On the other hand, deep learning is suitable to solve these problems by minimizing relatively complex functions. Some mathematical methods can be suitable to solve some relatively simple problems, and deep learning performs better when there are many and various cases of inputs and corresponding outputs. For this problem, because there are so many cases of how the devices are located in a WPCN environment, the computation and optimization of the common throughput would be more complex if there are many devices, so mathematical methods have some limits to solving this kind of problem. So, although the method in [2] is suitable to solve this problem, it would be worth trying to apply the deep learning method here for comparative purposes. We can make many and various cases of data that the inputs are the vector or tensor with the location of devices, and the outputs are the common throughput for when the HAP is located at each point. So, the motivation of this paper is to introduce deep learning to optimize the placement of HAP in the relatively complex WPCN environment. This paper introduces a methodology to place an HAP in a WPCN environment to maximize common throughput when time allocation is optimized, by using deep learning, and shows that this methodology has a meaningful contribution to solving this problem and shows better performance than the mathematical methodology already studied, such as [2].

Section 3 describes our HAP placement model, data preparation and algorithm for training, and how to find the best HAP placement. Section 4 describes our design and an environment for the experiments and the experimental results of our model. Section 5 describes our analysis of the results. Finally, Sect. 6 describes the conclusion of this paper.

2 Related works

Our system has only one HAP, and the goal of our system is to maximize the common throughput of devices. Considering the system, Song et al., Lee, Kim et al., Kwan and Fapojuwo and Thomas and Malarvizhi [3–7] have an HAP and many devices with their systems as same as this research. In detail, the HAP and devices in the system of [3] have antennas. In [4], the spectrum of HAP and devices for both DL WET and

UL WIT are the same. The system of [5] consists of a primary WIT and a secondary WPCN system, and the HAP and the devices are in the latter. The system of [6] uses radio frequency (RF) to harvest energy. The system of [7] consists of not only HTT (harvest-then-transmit) but also backscatter mode. Tang et al. [8] have many UAV (unmanned aerial vehicle)s and many devices, Hwan et al. [9] have many HAPs and many devices, Xie et al. [10] have a UAV and many devices, Biason and Zorzi [11] have an AP(access point) and two devices recharged by the AP, Cao [12] have a relay communication system and many devices. Chi et al. [13] compares the performance of TDMA- and NOMA-based WPCN for EP (energy provision) minimization problem with network throughput constraints, Kwan and Fapojuwo [14] tries to maximize the sum throughput of the wireless sensor network using three protocols, and [15] tries to optimize time allocation for backscatter-assisted WPCN to maximize total throughput. Considering the objective functions and constraints of variables, Tang et al., Xie et al. and Biason and Zorzi [8, 10, 11] try to maximize common throughput, in another word, minimum throughput and [11] tries to maximize long-term minimum throughput. Hwan et al. [9] tries to maximize the sum-rate performance. Song et al., Lee, Kim et al. Kwan and Fapojuwo, Thomas and Malarvizhi, Cao, Kwan and Fapojuwo and Ramezani and Jamalipour [3–7, 12, 14, 15] try to maximize sum throughput. In detail, Song et al. and Kim et al. [3, 5] also use transmit covariance matrix for DL-WET. Lee [4] defines the problem as maximizing the sum throughput for U-CWPCN and O-CWPCN (two overlay-based cognitive WPCN models). Kwan and Fapojuwo [6, 14] uses bandwidth allocation to optimize it. Chi et al. [13] tries to minimize EP of H-sink. Cao [12] have 3 divided time slots as variables with constraints. Ramezani and Jamalipour [15] uses the achievable throughput of both the users and EIRs in two phases. Thomas and Malarvizhi [7] define the sum throughput of all users as the sum of the throughput of two modes, HTT and backscatter mode. Therefore, our research can be compared with [10] because both the system model and the variable to maximize (or minimize) are the same.

Considering the methods, Song et al., Lee, Kim et al., Xie et al., Cao and Chi et al. [3–5, 10, 12, 13] just applied mathematical optimization methods using convex optimization methods like CVX [16] and transforming non-convex problems to convex problems. In detail, Song et al. and Kim et al. and Chi et al. [3, 5, 13] use golden section method, and [4] uses Newton's method for time allocation. Kim et al. and Xie et al. [5, 10] use Lagrange dual method and subgradient-based methods such as the ellipsoid method. Chi et al. [13] also uses the bisection method for time allocation, and [5] also uses a line search method. Cao [12] uses SDP (Semi-Definite Programming) relaxation to derive the optimal solution. Tang et al. [8] used Multi-agent deep Q learning (DQL), Hwan et al. [9] used multi-agent deep reinforcement learning (MADRL) and distributed reinforcement learning, Biason and Zorzi [11] used Markov Chain and Markov Decision Process, Kwan and Fapojuwo [14] used its own three protocols, and [6] used MS-BABF/Hybrid-STF method. The method applied to [15] is similar to the mathematical optimization methods used in [3–5, 10, 12, 13] but combined with Block Coordinate Descent (BCD) method. Thomas and Malarvizhi [7] describe no particular methods for finding the solution.

Consequently, the research that can be compared with ours, Xie et al. [10], does not use machine learning methods. So, we can apply machine learning methods to solve this problem, and this can be an improved method to find the optimal placement of the HAP.

3 Methods: using HAP placement model

3.1 Overview

Figure 1 describes the system architecture of the model. Let us explain our model using the definitions above. Mobile HAP can be placed at any location in the environment and can move to any other location in the environment. The goal is to maximize common throughput that is defined as the minimum throughput between the HAP and each WD by optimizing the HAP placement. So, the HAP needs to move to the location where the minimum throughput is maximized. So, in WDs placement map, the HAP can be located at any grid in the map and should be located at the best throughput point. The rightmost figure of Fig. 1 describes computed minimum throughput for each grid when the HAP is located at the grid and the best throughput point.

Figure 2 is the flow chart of the HAP placement model. The model is composed of three phases. First, "making data" is to create training and test data. Next, "training using data" is to process the data to convert to training and test data for the deep learning model, and train using the model. Last, "finding the best point" is to find the best HAP placement point using the throughput map derived from this model.

In this paper, we map the physical wireless channel environment into a 2-dimensional array. As in [1], we assume that the environment is located in the free space, so the path loss follows the rule for the free space. Just one exception for this is, for each WD, when the distance between the HAP and the WD is less than a specific value, we compute the throughput as when the distance is the value. Detailed discussion about it will be discussed in the Sect. 3.2.

From now on, we use the definitions here. WDs placement map means the grid map representing the environment, as in Fig. 1. N and M mean the number of rows and columns of the WDs placement map, respectively, and K means the number of wireless devices in the WDs placement map. Block means each grid in the WDs placement

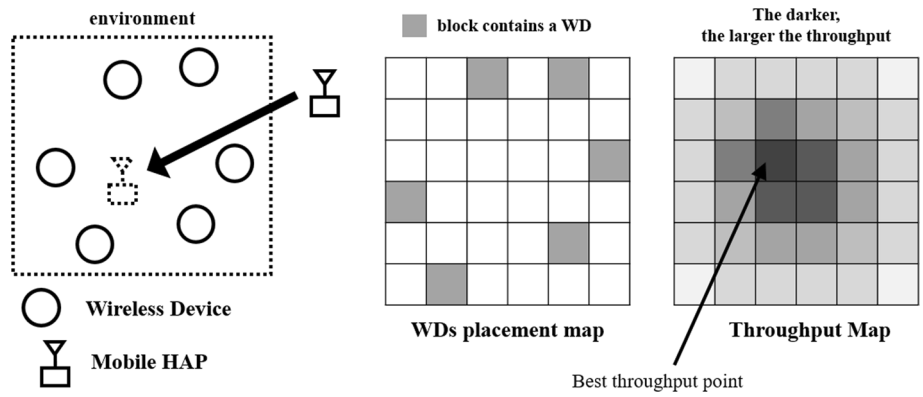

Fig. 1 The system architecture of our model. Wireless devices are placed in the environment, and we can represent the location of WDs as a WDs placement map. A mobile HAP is placed in the environment, and the throughput value of the environment is calculated and represented as a throughput map

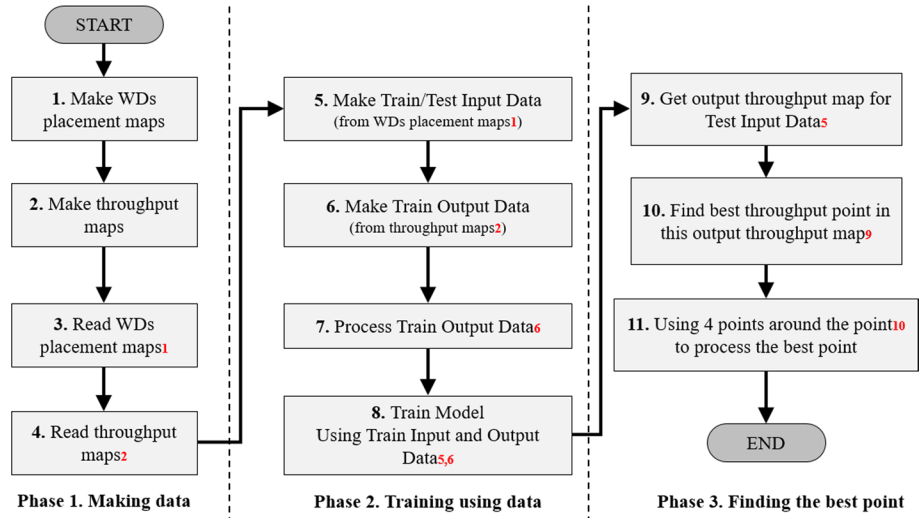

Fig. 2 Flow chart of HAP placement model. Note that the number after the description of each phase means the order of the stage whose data are used in this stage

map, so there are $N \times$M blocks in the WDs placement map with N rows and M columns. K blocks among these $N \times$M blocks in the map, randomly set at the training stage, contain a WD. None of the pairs of two WDs out of all the WDs occupy the same block in the map. From now on, we call the i-th wireless placement map with N rows and M columns $\mathrm{WDPM}_i(N, M, K)$. Throughput map means the grid map with N rows and M columns, and each block contains the throughput value where the HAP is located in this block of $\mathrm{WDPM}_i(N, M, K)$. From now on, we call the throughput map corresponding to $\mathrm{WDPM}_i(N, M, K)$ $\mathrm{TM}_i(N, M)$. Best throughput point means the position of HAP that maximizes throughput value in $\mathrm{TM}_i(N, M)$, derived from our model, so it could be not a real position that maximizes the throughput value. We will call the best throughput point corresponding to $\mathrm{TM}_i(N, M)$ $\mathrm{BTP}_i(N, M)$.

3.2 Making data

We computed and used Eq. (1) by combining Eqs. (7) and (8) in [17] for the throughput. To make $\mathrm{WDPM}_i(N, M, K)'$s, $\mathrm{i} = 0, \ldots, \mathrm{m_{total}} - 1$, where m_{total} is the sum of the number of training and test maps, first define a grid map with N rows and M columns, $N \times$M blocks in total. Then repeat placing a WD on randomly selected point without a HAP K times. To make $\mathrm{TM}_i(N, M)'$s, $\mathrm{i} = 0, \ldots, \mathrm{m_{total}} - 1$ using these $\mathrm{WDPM}_i(N, M, K)'$s, place HAP at each point in $\mathrm{WDPM}_i(N, M, K)'$s and compute throughput for the location of HAP and each WD using Algorithm 1 because the throughput is computed using (1). Procedure getThrput finds optimal time allocation given $\mathrm{WDPM}_i(N, M, K)$. Because we supposed that $\zeta = 1.0$, $\mathrm{h_i} = 0.001\mathrm{p_i^2 d^{-\alpha_d}}$where $\alpha_d = 2.0$, $g_i = 0.001 p_i^2 d^{-\alpha_u}$, $\alpha_u = 2.0$, $p_i = 1.0$, $P_A = 20.0$, $\Gamma = 9.8$ and $\sigma = 0.001$ where d is the distance from the HAP and each WD, this formula can be converted into (2). To prevent divide by 0 error and consider the limit of throughput, we supposed that distance is 1.0 when actual distance is less than 1.0.

$$R_i(\tau) = \tau_i \log_2\left(1 + \frac{\zeta h_i g_i P_A}{\Gamma \sigma^2}\frac{\tau_0}{\tau_i}\right), i = 1, \ldots, K \tag{1}$$

$$R_i(\tau) = \tau_i \log_2\left(1 + \frac{100 p_i^4}{49 \times \max(d,1)^4}\frac{\tau_0}{\tau_i}\right), i = 1, \ldots, K \tag{2}$$

Then, because $\mathrm{WDPM}_i(N, M, K)'$s and $\mathrm{TM}_i(N, M)'$s are saved as text files, the model must read them before using them.

Algorithm 1 Finding throughput for a HAP location and each WD's location

Input: list of the location of each WD $wdList$, location (Y-axis and X-axis) of HAP $HAPpoint$
Output: throughput value $finalThrput$
$lr \leftarrow 5.0$
$timeList \leftarrow [1.0, 1.0, ..., 1.0]$
for $i \leftarrow 1$ to 1000 **do**
  $tpChange \leftarrow []$
  $thrput \leftarrow$ GETTHRPUT($wdList$, $HAPpoint$, $timeList$)
  for each x in (HAP, $wdList$) **do**
    $timeListCopy \leftarrow timeList$
    $timeListCopy[x] \leftarrow timeListCopy[x] + 1$
    $newThrput \leftarrow$ GETTHRPUT($wdList$, $HAPpoint$, $timeListCopy$)
    $difThrput \leftarrow \log_2(newThrput/thrput)$
    append max($0.01 \times \log_2(difThrput)$, $\log_2(difThrput)$) to $tpChange$
    $timeList[x] \leftarrow timeList[x] \times 2^{lr \times tpChange[x]}$
  end for
end for
$finalThrput \leftarrow$ GETTHRPUT($wdList$, $HAPpoint$, $timeList$)
return $finalThrput$

function GETTHRPUT($wdList, HAPpoint, timeList$)
  (HAP_y, HAP_x) $\leftarrow$ (Y axis of $HAPpoint$, X axis of $HAPpoint$)
  $sumOfTime \leftarrow$ sum($timeList$)
  $HAPtime \leftarrow timeList[0]/sumOfTime$
  $result \leftarrow INFINITE$
  for each WDi **do**
    (WD_y, WD_x) $\leftarrow$ (Y axis of $wdList[i]$, X axis of $wdList[i]$)
    $dist \leftarrow \sqrt{(WD_y - HAP_y)^2 + (WD_x - HAP_x)^2}$
    $chargeTime \leftarrow timeList[1+i]/sumOfTime$
    $throughput \leftarrow chargeTime \times \log_2(1 + \frac{100 p_i^4}{49 \times (max(dist,1))^4} \times \frac{HAPtime}{chargeTime})$
    $result \leftarrow$ min($result$, $throughput$)
  end for
  return $result$
end function

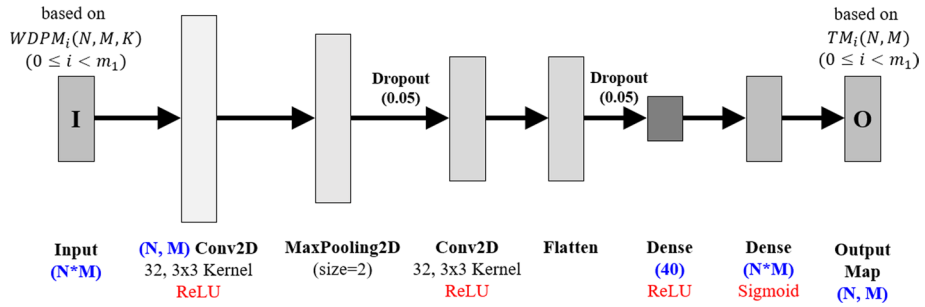

Fig. 3 Architecture of deep learning model for common throughput maximization: we use convolutional neural network (CNN) [19] for our methodology

3.3 Training

First, make input data for training and testing based on $\mathrm{WDPM}_i(N, M, K)$, $i = 0, \ldots, m_1 + m_2 - 1$, supposing that the number of training and testing data is m_1 and m_2 each. The model considers first m_1 maps as training data and next m_2 maps as test data. The input data are the $N \times M$ map whose value at each block of the map is -1 when a WD is on this block and 0 otherwise. Then make output data for training based on $\mathrm{TM}_i(N, M)'\mathrm{s}$, $\mathrm{i} = 0, \ldots, \mathrm{m}_1 - 1$ corresponding to $\mathrm{WDPM}_i(N, M, K)'\mathrm{s}$, $\mathrm{i} = 0, \ldots, \mathrm{m}_1 - 1$.

The output data are the $N \times M$ map whose value at each block, whose row index is n, and column index is m that is $V^{i''}_{n,m}$, defined below. We define $V^{i}_{n,m}$, $i = 0, \ldots, m_1 - 1$, $n = 0, \ldots, N - 1$, $m = 0, \ldots, M - 1$, where the value at the block at the intersection of n-th row and m-th column of the map is the maximum throughput where HAP is placed at this block, and the wireless devices are placed as $\mathrm{WDPM}_i(N, M, K)$, $i = 0, \ldots, m_1 - 1$. The following is the procedure to compute $V^{i''}_{n,m}$. First, find maximum common throughput value $\max\left(\mathrm{V}^{\mathrm{i}}_{\mathrm{n,m}}\right)$, $n = 0, \ldots, N - 1$, $m = 0, \ldots, M - 1$ for each training output map $i = 0, \ldots, m_1 - 1$ using Algorithm 1, and then divide each value $V^{i}_{n,m}$, $i = 0, \ldots, m_1 - 1$, $n = 0, \ldots, N - 1$, $m = 0, \ldots, M - 1$ by $\max\left(\mathrm{V}^{\mathrm{i}}_{\mathrm{n,m}}\right)$. Last, transform each value $V^{i}_{n,m}$ at each block using (3).

$$V^{i'}_{n,m} = \mathrm{sigmoid}\left(2\mathrm{V}^{\mathrm{i}}_{\mathrm{n,m}} - 1\right) \tag{3}$$

In (3), sigmoid(x) is defined as $1/(1 + \exp(-\mathrm{x}))$. Then train using input data $\mathrm{WDPM}_i(N, M, K)'\mathrm{s}$, $i = 0, \ldots, m_1 - 1$ and corresponding m_1 output data made based on $\mathrm{TM}_i(N, M)'\mathrm{s}, \mathrm{i} = 0, \ldots, \mathrm{m}_1 - 1$ using the deep learning model described in Fig. 3 with Adam optimizer [18] with learning rate 0.0001 and 1000 epochs.

3.4 Finding the best points

Using test input data, the model finds best point for HAP placement. For each test input data created using $\mathrm{WDPM}_i(N, M, K), i = m_1, \ldots, m_1 + m_2 - 1$, input these data into the model trained in Sect. 3.3 and get output maps corresponding to $\mathrm{TM}_i(N, M), i = m_1, \ldots, m_1 + m_2 - 1$. For each value $V^{i'}_{n,m} = \mathrm{sigmoid}\left(2\mathrm{V}^{\mathrm{i}}_{\mathrm{n,m}} - 1\right)$ at each block in each output map is converted by (4) using the inverse function of the sigmoid

function, to convert them from the form of $V^{i'}_{n,m}$= sigmoid$\left(2\mathrm{V}^{\mathrm{i}}_{\mathrm{n,m}} - 1\right)$ into the form of $V^{i''}_{n,m} = 2V^{i}_{n,m} - 1$ form, where $V^{i}_{n,m}$ is the estimated common throughput value.

$$V^{i''}_{n,m} = \mathrm{invSigmoid}\left(\mathrm{V}^{\mathrm{i}'}_{\mathrm{n,m}}\right) \tag{4}$$

In (4), invSigmoid(x) is the inverse function of sigmoid(x) and defined as $\ln(\mathrm{x}/(1-\mathrm{x}))$. Then, for each output map, the model finds the maximum value among values in blocks of this map. Let's call row and column axis of this value in the map n_{M} and m_{M} , respectively, and call the maximum value $V^{i''}_{n_{\mathrm{M}},m_{\mathrm{M}}}$. Then the row axis n_{optimal} and column axis m_{optimal} of optimal HAP location are computed by (5) and (6) each, and $\mathrm{BTP}_i(N,\ M)$ is computed by (7), described in Fig. 4.

$$n_{\mathrm{optimal}} = n_{\mathrm{M}} + \frac{V^{i''}_{n_{\mathrm{M}}+1,\ m_{\mathrm{M}}} - V^{i''}_{n_{\mathrm{M}}-1,\ m_{\mathrm{M}}}}{V^{i''}_{n_{\mathrm{M}}-1,\ m_{\mathrm{M}}} + V^{i''}_{n_{\mathrm{M}},\ m_{\mathrm{M}}} + V^{i''}_{n_{\mathrm{M}}+1,\ m_{\mathrm{M}}}} \tag{5}$$

$$m_{\mathrm{optimal}} = m_{\mathrm{M}} + \frac{V^{i''}_{n_{\mathrm{M}},\ m_{\mathrm{M}}+1} - V^{i''}_{n_{\mathrm{M}},\ m_{\mathrm{M}}-1}}{V^{i''}_{n_{\mathrm{M}},\ m_{\mathrm{M}}-1} + V^{i''}_{n_{\mathrm{M}},\ m_{\mathrm{M}}} + V^{i''}_{n_{\mathrm{M}},\ m_{\mathrm{M}}+1}} \tag{6}$$

$$\mathrm{BTP}_i(N,\ M) = \left(n_{\mathrm{optimal}},\ m_{\mathrm{optimal}}\right) \tag{7}$$

If $V^{i''}_{n_{\mathrm{M}}+1,\ m_{\mathrm{M}}}$ is greater than $V^{i''}_{n_{\mathrm{M}}-1,\ m_{\mathrm{M}}}$, n_{optimal} moves down from original position, and otherwise, it moves up. Similarly, if $V^{i''}_{n_{\mathrm{M}},\ m_{\mathrm{M}}+1}$ is greater than $V^{i''}_{n_{\mathrm{M}},\ m_{\mathrm{M}}-1}$, m_{optimal} moves right, and otherwise, it moves left. Because original common throughput $V^{i}_{n,m}$ and $2V^{i}_{n,m} - 1$ can be converted into each other by just a linear transmission, there is no difference of n_{optimal} and m_{optimal} between when converted $V^{i''}_{n,m}$ into $V^{i}_{n,m}$ and do not convert $V^{i''}_{n,m}$ into any other form.

4 Experiments and results

4.1 Experiment design and test metrics

Figure 5 is the flow chart for our experiment. For each estimated optimal HAP placement point for each test map $\mathrm{BTP}_i(N,\ M) = \left(n_{\mathrm{optimal}},\ m_{\mathrm{optimal}}\right), i = m_1, \ldots, m_1 + m_2 - 1$ derived by Sect. 3.4, corresponding to $\mathrm{TM}_i(N,\ M)'\mathrm{s}, \mathrm{i} = \mathrm{m}_1, \ldots, \mathrm{m}_1 + \mathrm{m}_2 - 1$, first compute common throughput value $C_i, i = m_1, \ldots, m_1 + m_2 - 1$ using this point. Because we use $\mathrm{TM}_i(N,\ M)'\mathrm{s}, \mathrm{i} = \mathrm{m}_1, \ldots, \mathrm{m}_1 + \mathrm{m}_2 - 1$ only for computing the difference when testing, the throughput maps as generated using the output of the model, called $\mathrm{TM}'_i(N,\ M)'\mathrm{s}, \mathrm{i} = \mathrm{m}_1, \ldots, \mathrm{m}_1 + \mathrm{m}_2 - 1$ in this section, are not equal to corresponding $\mathrm{TM}_i(N,\ M)'\mathrm{s}, \mathrm{i} = \mathrm{m}_1, \ldots, \mathrm{m}_1 + \mathrm{m}_2 - 1$. Then compare the throughput value with $MC_i, i = m_1, \ldots, m_1 + m_2 - 1$, the maximum common throughput value among all points $(n,\ m),\ n = 0, \ldots, N-1,\ m = 0, \ldots, M-1$ in corresponding $\mathrm{TM}_i(N,\ M)$. Then the test metrics are defined as and computed using (8), (9) and (10).

$$\mathrm{CT.AVERAGE} = \frac{\sum_{\mathrm{i}=\mathrm{m}_1}^{\mathrm{m}_1+\mathrm{m}_2-1} \mathrm{C_i}}{\mathrm{m}_2} \tag{8}$$

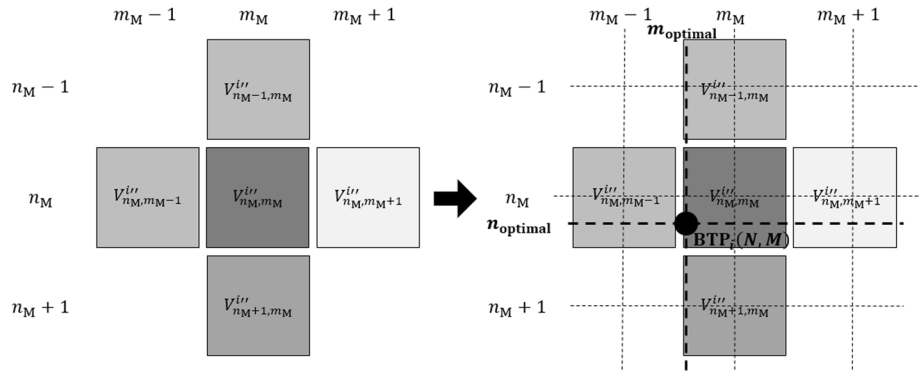

Fig. 4 Decision algorithm for $n_{optimal}$ and $m_{optimal}$. For each picture on the left and right, $n_M - 1$, n_M, $n_M + 1$, $m_M - 1$, m_M and $m_M + 1$ means the Y and X-axis of the environment, respectively. Each rectangle with the value means the common throughput value at the point of these Y and X-axis

$$\text{CT.AVGMAX} = \frac{\sum_{i=m_1}^{m_1+m_2-1} \text{MC}_i}{m_2} \tag{9}$$

$$\text{CT.RATE} = \frac{\sum_{i=m_1}^{m_1+m_2-1} \text{C}_i}{\sum_{i=m_1}^{m_1+m_2-1} \text{MC}_i} \tag{10}$$

CT.AVERAGE means average common throughput for each test map with corresponding $\text{BTP}_i(N, M), i = m_1, \ldots, m_1 + m_2 - 1$, and CT.AVGMAX means maximum common throughput value for each throughput map corresponding to each test map, and CT.RATE means the rate between the sum of C_i and the sum of MC_i for all test maps. It also means the rate between CT.AVERAGE and CT.AVGMAX. We also define performance rate *PR* as (11) meaning how well our methodology is compared to the methodology used in the original paper, and the original paper in (11) means [2].

$$\text{PR} = \frac{(\text{CT.AVERAGE of } \text{M}_1)}{(\text{CT.AVERAGE of } \text{M}_0)} \tag{11}$$

In (11), M_1 is our methodology, and M_0 is the methodology in the original paper. CT.RATE can be larger than 1.0 because CT.AVGMAX means the average of largest value among the value at discrete blocks from corresponding TM_i, but CT.AVERAGE means the average of common throughput value with non-discrete HAP location.

4.2 Experimental environment

The computer system information for our experiment is as the following. The operating system is Window 10 Pro 64bit (10.0, build 18363), system manufacturer is LG Electronics, the system model is 17ZD90N-VX5BK, the BIOS is C2ZE0160 X64, the processor is Intel(R) Core i5-1035G7 CPU @ 1.20 GHz (8 CPUs), ~1.5 GHz, and the memory is 16384MB RAM. The programming language is Python 3.7.4, and used NumPy [20],

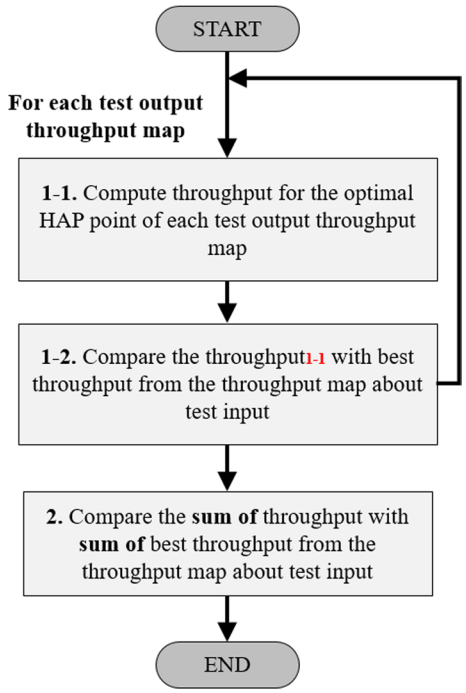

Fig. 5 Flow chart of design of testing. Note that 1-1 after the description of stage 1-2 means the data of stage 1-1 is used in stage 1-2

Tensorflow [21] and Keras as libraries. You can download the experiment code from https://github.com/WannaBeSuperteur/2020/tree/master/WPCN.

4.3 Experimental results

Table 1 describes CT.RATE (%) and CT.AVERAGE values for our methodology and the methodology in the original paper. We used $f_d = 9.15\times10^8$, $P_0 = 1.0$, $A_d = 3.0$, $\eta = 0.51$, $d_D = 2.2$, $\delta = 20$, $\sigma = 10^{-6}$ and $\beta = A_d\left(\frac{3\times10^8}{4\pi f_d}\right)^{d_D}$ with $\pi = 3.141592654$ for the methodology in [2], and the algorithm to solve (20) in [2]is described in Algorithm 2. For our methodology, CT.RATE value increases when the number of WDs increases and decreases when the size of maps increases, and CT.AVERAGE decreases when both the number of WDs and the size of maps increases. For the methodology in the original paper, CT.RATE increases when the size of maps increases, but has no significant correlation with the number of WDs. Table 2 shows the values of CT.AVGMAX and *PR* for each size and number of WDs. The unit for size is one block, as mentioned in Sect. 3. For example, the size of 12×12 means that the environment contains 12 rows, and each row contains 12 blocks. CT.AVGMAX decreases when both the number of WDs and the size of maps increases and *PR* decreases when the size of maps increases, but has no significant correlation with the number of WDs. For smaller sizes, our methodology shows significantly better performance (PR $>$ 1) than the methodology in the original paper, but for 12×12 size, these two methods show almost the same performance. (PR $\approx$ 1), and for 16×16 size, our methodology shows worse performance. (PR $<$ 1) Fig. 6. is the line chart representation of

Fig. 6 Line chart version of Tables 1 and 2. For the 4 tables in the left, the upper 2 table shows the result of CT.RATE and CT.AVERAGE of our methodology, and the lower 2 table shows the result of them of the methodology in the original paper. The 2 tables in the right show the values of CT.AVGMAX and PR, respectively

Tables 1 and 2, and Fig. 7. is the bar chart for comparison of our methodology and the methodology in the original paper.

Algorithm 2 Finding throughput for a HAP location and each WD's location

Input: λ_k, $a_{1,k}$, $a_{2,k}$, w_k, d_U and v_{1_k}, as defined in the original paper
Output: optimal value of v_{1_k} (by Gradient Descent Method)
$lr \leftarrow 3.0 \times 10^8$
for $i = 1$ to 7000 **do**
$t_{min}^{v_{1_k}} \leftarrow min(\lambda_k - a_{1,k} - a_{2,k}||v_{1_k} - w_k||^{d_U}), k = 1, ..., K$
$v_{1_k}^0 \leftarrow [v_{1_k}[0] + 10^{-6}, v_{1_k}[1]]$
$v_{1_k}^1 \leftarrow [v_{1_k}[0], v_{1_k}[1] + 10^{-6}]$
$t_{min}^{v_{1_k}^0} \leftarrow min(\lambda_k - a_{1,k} - a_{2,k}||v_{1_k}^0 - w_k||^{d_U}), k = 1, ..., K$
$t_{min}^{v_{1_k}^1} \leftarrow min(\lambda_k - a_{1,k} - a_{2,k}||v_{1_k}^1 - w_k||^{d_U}), k = 1, ..., K$
$v_{1_k}[0] \leftarrow v_{1_k}[0] + lr \times (t_{min}^{v_{1_k}^0} - t_{min}^{v_{1_k}})$
$v_{1_k}[1] \leftarrow v_{1_k}[1] + lr \times (t_{min}^{v_{1_k}^1} - t_{min}^{v_{1_k}})$
end for
return v_{1_k}

5 Discussion

Our method shows higher CT.RATE for smaller maps, and the methodology in the original paper shows higher CT.RATE for larger maps. The reason for the former is, first, that common throughput usually depends on the WDs near the boundary of the environment, and these WDs usually enlarge the minimum value of the maximum possible distance between the HAP and each WD. For larger maps, the influence on the learning of the blocks with these WDs decreases, because the number of blocks influencing the learning is larger, so the influence of each block on the learning decreases. Second, there are fewer possible cases for smaller maps because the number of blocks is fewer for them, so our model could be more accurate. The reason for the latter is that the locations of WDs are not realistic for smaller maps because both x and y-axis of them are always an integer, so the methodology in the original paper is not so accurate.

Tables 3, 4 and 5 describes the average, standard deviation and 95% confidence interval of some variables from the experimental result using 100 test dataset samples, that is, $\mathrm{WDPM}_i(N, M, K)$ and $\mathrm{TM}_i(N, M, K), i = m_1, \ldots, m_1 + m_2 - 1$ where m_1=900 and m_2 =100. When the value of 'rows' is r, it means the size of the grid map is r×r. We computed the confidence interval using (12) where $\bar{X}$ and σ is average (refer to Table 3 to check the values) and standard deviation (refer to Table 4 to check the values) of the sample values, respectively, and n is the number of samples for each case, that is 100 for the experiment.

$$(95\%\ \text{confidence interval}) = \left[\bar{\mathrm{X}} - 1.96 \times \frac{\sigma}{\sqrt{\mathrm{n}}}, \bar{\mathrm{X}} + 1.96 \times \frac{\sigma}{\sqrt{\mathrm{n}}}\right] \tag{12}$$

According to Table 5, the portion of time allocated to the HAP (HAPtime) has a positive correlation with the size of the grid map. The values of Y/size and X/size when the size

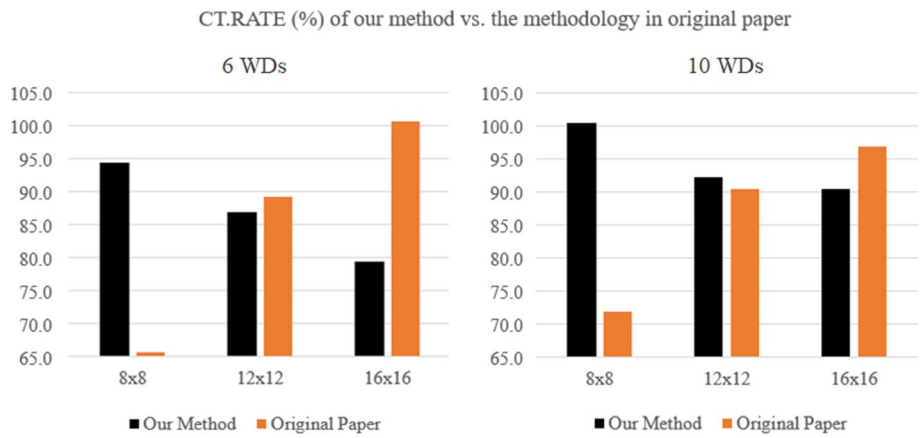

Fig. 7 Comparison of CT.RATE(%) of our methodology and the methodology in the original paper. Our methodology shows better performance than the methodology in the original paper for size 8×8, but the two methods show nearly the same performance for size 12×12, and our methodology shows worse performance for size 16×16

Table 1 CT.RATE and CT.AVERAGE values of our methodology and the methodology in the original paper

Size / WDs	CT.RATE (%)		CT.AVERAGE	
	6 WDs	10 WDs	6 WDs	10 WDs
Our methodology				
8×8	94.2787	100.4858	0.010222	0.006884
12×12	86.9357	92.1310	0.002565	0.001648
16×16	79.4420	90.4991	0.000790	0.000530
Methodology in the original paper				
8×8	65.5779	71.9627	0.007110	0.004930
12×12	89.1676	90.3931	0.002631	0.001617
16×16	100.6565	96.8606	0.001001	0.00568

The upper table is about our methodology, and lower table represents CT.RATE and CT.AVERAGE about the methodology in the original paper. Size means the size of the board representing the environment of WPCN

Table 2 The values of CT.AVGMAX and PR

CT.AVGMAX			PR		
Size/WDs	6 WDs	10 WDs	Size/WDs	6 WDs	10 WDs
8×8	0.010842	0.006851	8×8	1.437660	1.396360
12×12	0.002951	0.001789	12×12	0.974970	1.019226
16×16	0.000994	0.000586	16×16	0.789239	0.934323

The table on the left and on the right represents CT.AVGMAX and PR, respectively, for each option (size and the number of WDs). There can be some errors for PR values because the values of CT.AVGMAX have less significant figures than 6

of the grid map is 8×8 is smaller than those of when the size is larger, but these values when the size of the grid map is 12×12 and 16×16 are not so different.

If the size of the grid map is $r \times r$, both Y and X-axis values of the center of the top-left cell are 0.0, and the ones of the center of the bottom right cell are $r - 1$. Because we randomly put wireless devices on the grid map, the average values of both Y and

Table 3 Average values for each variable

Rows	WDs	Y	X	HAPtime	Y/rows	X/rows
8	6	3.5351	3.4875	0.9330	0.4419	0.4359
8	10	3.4885	3.4285	0.9201	0.4361	0.4286
12	6	5.5705	5.3169	0.9566	0.4642	0.4431
12	10	5.5630	5.5734	0.9503	0.4636	0.4645
16	6	7.4428	7.4439	0.9685	0.4652	0.4652
16	10	7.4846	7.5141	0.9627	0.4678	0.4696

This table describes the average value of some variables from the experimental result using 100 test dataset samples. For each sample $\mathrm{WDPM}_i(N, M, K)$ and $\mathrm{TM}_i(N, M, K)$, $i = 900, \ldots, 999$ in the test dataset, rows, WDs, Y, X, HAPtime, Y/rows and X/rows means the number of rows and wireless devices in $\mathrm{WDPM}_i(N, M, K)$, the value of n_{optimal} and m_{optimal}, the portion of time allocated to HAP computed with GETTHRPUT function in Algorithm 1 with HAPpoint as $[n_{\mathrm{optimal}}, m_{\mathrm{optimal}}]$, the value of n_{optimal}/rows and m_{optimal}/rows, respectively

Table 4 Standard deviation for each variable

Rows	WDs	Y	X	HAPtime	Y/rows	X/rows
8	6	0.5937	0.7304	0.0122	0.0742	0.0913
8	10	0.4670	0.4403	0.0113	0.0584	0.0550
12	6	1.0814	0.9942	0.0068	0.0901	0.0828
12	10	0.6123	0.7403	0.0073	0.0510	0.0617
16	6	1.3271	1.6320	0.0042	0.0829	0.1020
16	10	0.8325	0.8926	0.0035	0.0520	0.0558

This table describes the standard deviation of some variables from the experimental results using test dataset, are same to as Table 3.

Table 5 95% Confidence interval for each variable

Rows	WDs	Y	X	HAPtime	Y/rows	X/rows
8	6	[3.4187, 3.6514]	[3.3443, 3.6307]	[0.9306, 0.9354]	[0.4273, 0.4564]	[0.4180, 0.4538]
8	10	[3.3970, 3.5800]	[3.3422, 3.5148]	[0.9179, 0.9223]	[0.4246, 0.4475]	[0.4178, 0.4393]
12	6	[5.3586, 5.7825]	[5.1221, 5.5118]	[0.9552, 0.9579]	[0.4465, 0.4819]	[0.4268, 0.4593]
12	10	[5.4430, 5.6830]	[5.4283, 5.7185]	[0.9489, 0.9518]	[0.4536, 0.4736]	[0.4524, 0.4765]
16	6	[7.1827, 7.7029]	[7.1240, 7.7638]	[0.9677, 0.9694]	[0.4489, 0.4814]	[0.4453, 0.4852]
16	10	[7.3214, 7.6478]	[7.3392, 7.6891]	[0.9620, 0.9634]	[0.4576, 0.4780]	[0.4587, 0.4806]

This table describes the 95% confidence interval of some variables from the experimental results using test dataset, are same to as Tables 3 and 4

X-axis maximizing the common throughput should be $(r-1)/2$. In Table 5, one can see that all the confidence intervals for both Y and X-axis values include $(r-1)/2$ for all the cases with $r = 8$, $r = 12$ and $r = 16$. Y/rows and X/rows should have positive correlation with r where r means the number of rows, because Y/r and $X/r = ((r-1)/2)/r = (r-1)/2r$ increases when the value of r increases. In Table 5, one can see that it is true and when comparing the cases for $rows = 8$ and $rows = 16$, the confidence intervals is not overlapped for the cases that $WDs = 10$. These guarantee that our method randomly put wireless devices on the grid maps for test data. In addition, One can see that the portion of time allocated to HAP (HAPtime) has a positive correlation with the number of rows in the grid map (rows), and in Table 5, the confidence intervals are always not overlapped when the number of rows differs. It

indicates that when the number of rows in the grid map increases, the portion of time allocated to HAP also increases.

6 Conclusion

We showed that our deep learning-based method shows better performance than the mathematical method in the original paper [2] when the size is smaller than 12×12. Although our method may show worse performance if the size is larger than 12×12, our approach to find the optimal placement and time allocation for HAP using deep-learning is meaningful because there is no attempt to apply deep-learning to this problem yet. In addition, we found that with HAP locations derived by our method, the portion of time allocated to HAP has a positive correlation with the size of the grid map (8×8, 12×12 and 16×16). There are some limits to our study. First, our study has an advantageous point for our method that it uses only 1 HAP which is fitted to the experimental environment, but the method in the original paper may and commonly uses more than 1 HAPs. Second, we studied with just a few conditions, 3 options for map size and 2 options for the number of WDs. So, some future research should be done for many options in terms of the map size and the number of WDs, and the number of HAPs.

Abbreviations

AP: Access point; DL: Downlink; EIR: Energy and information receiver; EN: Energy node; HAP: Hybrid access point; NOMA: Non-orthogonal multiple access; UL: Uplink; TDMA: Time-division multiplexing access; WD: Wireless device; WET: Wireless energy transfer; WIT: Wireless information transmission; WPCN: Wireless powered communication networks.

Acknowledgements

This work was supported partly by the Institute for Information & Communications Technology Promotion (IITP) grant funded by the Korea government (MSIP) (No. 2020-0-00107, Development of the technology to automate the recommendations for big data analytic models that define data characteristics and problems), and partly by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (No. NRF-2019R1A2C1009894).

Authors' contributions

HSK conducted all the experiments for writing this paper and wrote this paper. IJ is the corresponding author, and he corrected the paper. Both authors read and approved the final manuscript.

Availability of data and materials

The data used for writing this paper are available from https://github.com/WannaBeSuperteur/2020/tree/master/WPCN..

Delcarations

Consent for publication

Not applicable

Competing interests

The authors declare that they have no competing interests.

Received: 27 January 2021 Accepted: 12 September 2021
Published online: 02 October 2021

References

1. S. Bi, Y. Zeng, R. Zhang, Wireless powered communication networks: an overview, in *IEEE Wireless Communications* (2015). arXiv: 1508.06366
2. S. Bi, R. Zhang, Placement optimization of energy and information access points in wireless powered communication networks. IEEE Trans. Wirel. Commun. **15**(3), 2351–2364 (2016)
3. D. Song, J. Lee, H. VincentPoor, Sum-throughput maximization in noma-based wpcn: a cluster-specific beamforming approach. IEEE Internet Things J. **8**(13), 10543–10556 (2021)
4. S. Lee, Cognitive wireless powered network: spectrum sharing models and throughput maximization. IEEE Trans. Cogn. Commun. Netw. **1**(3), 335–346 (2015)
5. J. Kim, H. Lie, C. Song et al., Sum throughput maximization for multi-user MIMO cognitive wireless powered communication networks. IEEE Trans. Wirel. Commun. **16**(2), 913–923 (2017)

6. J.C. Kwan, A.O. Fapojuwo, Sum-throughput and fairness optimization of a wireless energy harvesting sensor network. IEEE (2019). https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=8891499
7. R.M. Thomas, S. Malarvizhi, Throughput maximization in WPCN: assisted by backscatter communication with initial energy, in *International Conference on Intelligent Computing and Applications*, pp. 143–151 (2018). https://link.springer.com/chapter/10.1007/978-981-13-2182-5_15
8. J. Tang, J. Song, J. Ou, et al., Minimum throughput maximization for multi-uav enabled wpcn: a deep reinforcement learning method, in *IEEE Access* (2020). https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=8950047
9. S. Hwang, H. Kim, H. Lee, et al., in *Multi-Agent Deep Reinforcement Learning for Distributed Resource Management in Wirelessly Powered Communication Networks* (2020). arxiv: 2010.09171
10. L. Xie, J. Xu, R. Zhang, Throughput maximization for UAV-enabled wireless powered communication networks. IEEE Internet Things J. **6**(2), 1690–1703 (2019)
11. A. Biason, M. Zorzi, Long-term throughput optimization in WPCN with battery-powered devices, in *Workshop on Wireless Powered Communication Networks: From Theory to Industrial Challenges, WPCNets 2016* (2016). https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=7564702
12. Z. Cao, in *Maximum Throughput Design for WPCN Systems*, UNSW Australia, School of Electrical Engineering and Telecommunications (2020). http://www2.ee.unsw.edu.au/~derrick/WPCN_Cao_Thesis_2020.pdf
13. K. Chi, Z. Chen, K. Zhang et al., Energy provision minimization in wireless powered communication networks with network throughput demand: TDMA or NOMA? IEEE Trans. Commun. **67**(9), 6401–6414 (2019)
14. J.C. Kwan, A.O. Fapojuwo, Sum-throughput maximization in wireless sensor networks with radio frequency energy harvesting and backscatter communication. IEEE Sens. J. **18**(17), 7325–7339 (2018)
15. P. Ramezani, A. Jamalipour, Optimal resource allocation in backscatter assisted WPCN with practical energy harvesting model. IEEE Trans. Veh. Technol. **68**(12), 12406–12410 (2019)
16. M. Grant, S. Boyd, CVX: MATLAB software for disciplined convex programming, version 2.1. (2014). http://cvxr.com/cvx
17. H. Ju, R. Zhang, in *Throughput Maximization in Wireless Powered Communication Networks* (2014). arxiv: 1304.7886v4
18. D.P. Kingma, J. Ba, ADAM: a method for stochastic optimization, in *ICLR 2015*. arxiv: 1412.6980
19. S. Albawi, T.A. Mohammed, S. Al-Zawi, Understanding of a convolutional neural network, in *ICET 2017*. https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8308186
20. IEEE, The NumPy array: a structure for efficient numerical computation. Sci. Python. https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5725236
21. M. Abadi, P. Barham, J. Chen et al., TensorFlow: a system for large-scale machine learning, Google Brain. https://www.usenix.org/system/files/conference/osdi16/osdi16-abadi.pdf

Publisher's Note

Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Submit your manuscript to a SpringerOpen journal and benefit from:

- ► Convenient online submission
- ► Rigorous peer review
- ► Open access: articles freely available online
- ► High visibility within the field
- ► Retaining the copyright to your article

Submit your next manuscript at ► springeropen.com