Meta-learning Based Beamforming Design for MISO Downlink

Jingyuan Xia
School of Electrical Technology
National University of Defense Technology
Changsha, China 410072
Email: j.xia16@imperial.ac.uk

Deniz Gunduz
Department of Electrical and Electronic Engineering
Imperial College London
London, UK
Email: d.gunduz@imperial.ac.uk

2021 IEEE International Symposium on Information Theory (ISIT) | 978-1-5386-8209-8/21/$31.00 ©2021 IEEE | DOI: 10.1109/ISIT45174.2021.9518251

Abstract—**Downlink beamforming is an essential technology for wireless cellular networks; however, the design of beamforming vectors that maximize the weighted sum rate (WSR) is an NP-hard problem and iterative algorithms are typically applied to solve it. The weighted minimum mean square error (WMMSE) algorithm is the most widely used one, which iteratively minimizes the WSR and converges to a local optimal. Motivated by the recent developments in meta-learning techniques to solve non-convex optimization problems, we propose a meta-learning based iterative algorithm for WSR maximization in a MISO downlink channel. A long-short-term-memory (LSTM) network based meta-learning model is built to learn a dynamic optimization strategy to update the variables iteratively. The learned strategy aims to optimize each variable in a less greedy manner compared to WMMSE, which updates variables by computing their first order stationary points at each iteration step. The proposed algorithm outperforms WMMSE significantly in the high signal to noise ratio (SNR) regime and achieves comparable performance when the SNR is low.**

I. Introduction

Downlink beamforming is an essential technology for multi-antenna cellular networks. Optimization of the beamforming vectors is typically formulated as a weighted sum rate (WSR) maximization problem under a total transmit power constraint. However, the WSR maximization problem is non-convex and NP-hard [1]. Therefore, it is typically solved by iterative methods which either adopt convex approximations [2]–[4], or convert the original problem into an alternative formulation with closed-form solutions [5]–[7]. Among them, the iterative weighted minimum mean square error (WMMSE) algorithm is the most widely used approach that provides superior performance in most scenarios.

The WMMSE algorithm converts the WSR maximization problem into a weighted sum mean square error minimization problem under the total transmit power constraint. The equivalent minimization problem is convex with respect to each individual variable when the others are fixed. Therefore, a locally optimal solution can be obtained by minimizing the objective function with respect to individual variables in an iterative manner. The WMMSE algorithm has been shown to achieve state-of-the-art performance in a wide range of scenarios, and is often considered as the benchmark in the literature.

Our goal in this paper is to obtain a less greedy and more flexible iterative updating strategy, such that the solution of each individual variable moves away from the locally optimal solution and closer to the globally optimal solution. To achieve such an adaptive algorithm, we incorporate machine learning techniques. Using machine-learning-based solutions to improve the WMMSE algorithm has received significant recent attention. However, the common approaches are built on an end-to-end learning model, where the wireless channel coefficients are given as input to a neural network, which then outputs the estimation of the transmitter beamformer [8]–[10]. These approaches convert the original model-based problem into a data-driven learning-based problem, and the performance depends on the the network architecture. Alternatively, more recent works in [11], [12] propose unfolding the WMMSE algorithm in order to design a model-inspired computational structure for the neural network, and learn a specific optimization rule for updating the WMMSE parameters by training on a set of channel samples. The unfolded WMMSE algorithm follows the original mathematical model of the WSR optimization problem, and designs a specific network structure to map the variables and the parameters in the original problem; therefore, the optimization of the WSR problem is converted into the optimization of the designed neural network. However, the performance of the aforementioned approaches [8], [9], [11]–[13] do not surpass the WMMSE algorithm.

An alternative approach is to utilize meta-learning techniques to solve the underlying optimization problem. The work in [14] uses meta-learner networks to learn the variable updating strategy in the form of gradient descent, which surpasses the benchmark approaches including Adam [15] and RMSProp [16] in convergence speed. In [17], [18], meta-learning is applied to the design of solution algorithms for different gradient-descent-based optimization problems. Another popular model-agnostic meta-learning (MAML) algorithm [19], [20] uses meta-learning for finding a superior initialization, which can lead to a faster convergence. This idea has been recently employed in [13] for the WSR maximization problem, where there can be a mismatch between test and training SNRs. Meta-learning allows fine-tuning the NN parameters with few training samples. A meta-learning based global scope optimization (GSO) method is proposed in [21] for solving

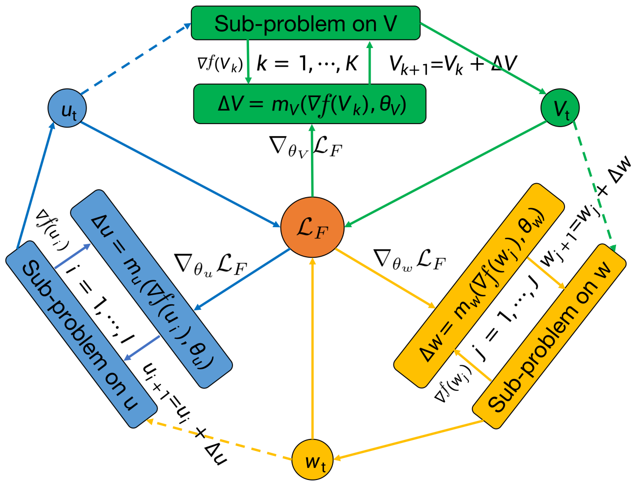

Fig. 1. The MLBF algorithm builds a bridge of information flow between the updates on each variable and the global objective function (denoted by $\mathcal{L}_F$). Each color represents the gradient flow for one individual variable. We use solid arrows to denote information flow, and dashed arrows to indicate the lack of gradient flow. It shows that the MLBF algorithm achieves a circular information flow between the variable update steps within each sub-problem and the global objective function throughout iterations.

non-convex multi-variable optimization problems. The GSO algorithm exhibits significantly better performance in solving non-convex problems, such as matrix completion without rank information or Gaussian mixture model with high dimensionality, than the traditional alternating minimization-based methods.

In this paper, we propose a meta-learning aided beamformer (MLBF) algorithm to solve the WSR maximization problem. The MLBF algorithm is built upon the iterative block-coordinate descent method, but instead of greedily optimizing each individual variables at each step, it can adaptively optimize each variable with respect to the geometry of the objective function. The proposed MLBF algorithm is established by three long-short-term-memory (LSTM) neural networks corresponding to the three complex variables to be optimized. The general structure of the MLBF algorithm for the WSR maximization problem is depicted in Fig.1. The notations and the detailed explanation will be presented in Section IV. It can be seen that the MLBF algorithm builds a bridge between the accumulated global loss function $\mathcal{L}_F$ and the update rules on each variable. This allows the gradient flow to circulate between the inner update loops for individual variable (referring to the update steps i, j and k for each variable) and the outer iterative steps for different variables (referring to different t).

The main contribution of this work is the proposed MLBF algorithm that can learn a less greedy update strategy for the variables, and therefore, achieves a better performance than the WMMSE algorithm. The extensive numerical results show that the proposed MLBF algorithm outperforms the WMMSE algorithm in terms of WSR when the SNR is high and performs equally at low SNR. We observe that the gain from the proposed MLBF algorithm in terms of the achieved WSR compared to the WMMSE algorithm increase with the channel SNR.

II. Problem Formulation

In this paper, we study a multiple-input single output (MISO) downlink channel, where a base station with M transmit antennas serves N single-antenna users. Let $x_i \sim \mathcal{CN}(0,1)$ denote the independent data symbol transmitted to the i^{th} user, while $\boldsymbol{h}_i \sim \mathcal{CN}(\boldsymbol{0}, \boldsymbol{I}_M)$ denote the channel between the base station and the i^{th} user. Then the received signal at the i^{th} user is given by

$$y_i = \boldsymbol{h}_i^H \boldsymbol{v}_i x_i + \sum_{j=1, j\neq i}^{N} \boldsymbol{h}_i^H \boldsymbol{v}_j x_j + n_i, \tag{1}$$

where $\boldsymbol{v}_i \in \mathbb{C}^M$ is the transmit beamforming vector for user i and $n_i \in \mathcal{CN}(0, \sigma^2)$ denotes a complex circularly symmetric zero mean Gaussian noise with variance σ^2. The signal-to-interference-plus-noise-ratio (SINR) is given by

$$\text{SINR}_i = \frac{|\boldsymbol{h}_i^H \boldsymbol{v}_i|^2}{\sum_{j=1, j\neq i} |\boldsymbol{h}_i^H \boldsymbol{v}_j|^2 + \sigma^2}. \tag{2}$$

Given the channel knowledge, the WSR maximization problem of the downlink channel with respect to a total transmit power constraint is formulated as

$$\begin{aligned} &\max_{\boldsymbol{V}} \sum_{i=1}^{N} \alpha_i \log_2(1 + \text{SINR}_i) \\ &\quad \text{s.t. } \text{Tr}(\boldsymbol{V}\boldsymbol{V}^H) \leq P, \end{aligned} \tag{3}$$

where α_i is the user weight specified by the system designer, P is the maximum transmit power, $\boldsymbol{V} \triangleq [\boldsymbol{v}_1, \ldots, \boldsymbol{v}_N]^T$, and $\text{Tr}(\cdot)$ denotes the trace operator.

III. WMMSE Algorithm

As mentioned earlier, the WSR maximization problem in (3) is NP hard and non-convex. The iterative WMMSE algorithm is typically implemented to solve it in the following steps: First, problem (3) is converted into the equivalent weighted sum mean square error minimization problem:

$$\begin{aligned} &\min_{\boldsymbol{u},\boldsymbol{w},\boldsymbol{V}} \alpha_i \sum_{i=1}^{N} (w_i e_i - \log_2 w_i) \\ &\quad \text{s.t. } \text{Tr}(\boldsymbol{V}\boldsymbol{V}^H) \leq P, \end{aligned} \tag{4}$$

where e_i is the mean-square error given by

$$\begin{aligned} e_i &= \mathbb{E}_{\boldsymbol{x}, n_i,}[|\hat{x}_i - x_i|^2] \\ &= |u_i \boldsymbol{h}_i^H \boldsymbol{v}_i - 1|^2 + \left(\sum_{j\neq i, j=1}^{N} |u_i \boldsymbol{h}_i^H \boldsymbol{v}_j|^2 \right) + \sigma^2 |u_i|^2, \end{aligned} \tag{5}$$

u_i denotes the receiver gain, w_i is a positive user weight, $\boldsymbol{w} = [w_1, \ldots, w_N]^T$, $\boldsymbol{u} = [u_1, \ldots, u_N]^T$, and n_i is the noise term. Although this new formulation is also non-convex, problem (4) is convex with respect to each individual variable when the others are fixed. Therefore, this new formulation can be solved by block coordinate descent. Indeed, a closed-form solution with respect to the first-order stationary point for each variable can be obtained. Then, the WMMSE algorithm iteratively updates each variable $i = 1, \ldots, N$ by computing their first order stationary points as follows

$$w_i = \frac{\sum_{j=1}^{N} |\boldsymbol{h}_i^H \boldsymbol{v}_j|^2 + \sigma^2}{\sum_{j\neq i, j=1}^{N} |\boldsymbol{h}_i^H \boldsymbol{v}_j|^2 + \sigma^2}, \tag{6}$$

$$u_i = \frac{\boldsymbol{h}_i^H \boldsymbol{v}_i}{\sum_{j=1}^{N} |\boldsymbol{h}_i^H \boldsymbol{v}_j|^2 + \sigma^2}, \tag{7}$$

$$\boldsymbol{v}_i = \alpha_i u_i w_i \boldsymbol{h}_i (\boldsymbol{A} + \mu \boldsymbol{I})^{-1}, \tag{8}$$

where $\boldsymbol{A} \triangleq \sum_{i=1}^{N} \alpha_i w_i |u_i|^2 \boldsymbol{h}_i \boldsymbol{h}_i^H$, and $\mu \geq 0$ is a Langrange multiplier. In a nutshell, the WMMSE algorithm first initializes $\boldsymbol{V}$ so that it satisfies the power constraint, then iteratively update $\boldsymbol{w}$, $\boldsymbol{u}$, and $\boldsymbol{V}$ following the equations (6)-(8) until a stop criterion is met. Parameter μ is computed by bisection search. Further details of the WMMSE algorithm can be found in [7].

IV. A Meta-learning based Beamforming (MLBF) Algorithm

We propose a meta-learning based WMMSE algorithm to learn an adaptive updating strategy of each variable in problem (4). In particular, for each variable, the update term is generated by a meta-learner neural network whose parameters are continuously updated with the objective of minimizing the global loss function $F(\boldsymbol{u}, \boldsymbol{w}, \boldsymbol{V})$. We refer to the minimization of $F(\boldsymbol{u}, \boldsymbol{w}, \boldsymbol{V})$ as the overall problem while to the minimization of $f(\boldsymbol{u})$,$f(\boldsymbol{w})$ and $f(\boldsymbol{V})$ as the three sub-problems to be optimized sequentially in each iteration. Following (4) the overall problem can be written as

$$\min F(\boldsymbol{u}, \boldsymbol{w}, \boldsymbol{V}) \triangleq \alpha_i \sum_{i=1}^{N} (w_i e_i - \log_2 w_i) + \mu \mathrm{Tr}(\boldsymbol{V}\boldsymbol{V}^H) - \mu P. \tag{9}$$

Three LSTM networks $m_{\boldsymbol{V}}$, $m_{\boldsymbol{w}}$ and $m_{\boldsymbol{u}}$ are established for updating $\boldsymbol{V}$, $\boldsymbol{w}$, and $\boldsymbol{u}$, respectively. We define the variable update process within each sub-problem as inner loops and the iterative steps over different sub-problems as outer loops. We correspondingly denote by $\boldsymbol{\theta}_{\boldsymbol{V}}$, $\boldsymbol{\theta}_{\boldsymbol{u}}$ and $\boldsymbol{\theta}_{\boldsymbol{w}}$ the parameters of the three LSTM networks, and by $\boldsymbol{C}_{\boldsymbol{V}}$, $\boldsymbol{C}_{\boldsymbol{u}}$, and $\boldsymbol{C}_{\boldsymbol{w}}$ their cell states. The inputs of the LSTM networks are the gradients of the sub-problems and the previous states of the variables which are represented by the corresponding $\nabla f()$ and $\boldsymbol{C}$, respectively. The outputs of the LSTM networks include updating steps for variables, along with the update of the cell state. The cell states contain the previous variables' information, and can be adaptively adjusted by the inner control gates inside the corresponding LSTM networks. Further details can be found in [21]. Denoting the inner loop update steps by $k = 1, \ldots, K$, $i = 1, \ldots, I$, and $j = 1, \ldots, J$ as superscripts and the outer loop steps by $t = 1, \ldots, T$ as subscripts for each sub-problem, where K, I and J denote the maximum number of steps of the inner loops for each variable, and T denotes the maximum number of steps of the outer loop, respectively. The meta-learner networks $m_{\boldsymbol{V}}$, $m_{\boldsymbol{w}}$ and $m_{\boldsymbol{u}}$ carry out variable updates in the inner loops in the following form:

$$\begin{aligned} \boldsymbol{V}^{(k)} &= \boldsymbol{V}^{(k-1)} + m_{\boldsymbol{V}}(\nabla f(\boldsymbol{V}^{(k-1)}), \boldsymbol{C}_{\boldsymbol{V}^{(k-1)}}, \boldsymbol{\theta}_{\boldsymbol{V}}), \\ \boldsymbol{u}^{(i)} &= \boldsymbol{u}^{(i-1)} + m_{\boldsymbol{u}}(\nabla f(\boldsymbol{u}^{(i-1)}), \boldsymbol{C}_{\boldsymbol{u}^{(i-1)}}, \boldsymbol{\theta}_{\boldsymbol{u}}), \\ \boldsymbol{w}^{(j)} &= \boldsymbol{w}^{(j-1)} + m_{\boldsymbol{w}}(\nabla f(\boldsymbol{w}^{(j-1)}), \boldsymbol{C}_{\boldsymbol{w}^{(j-1)}}, \boldsymbol{\theta}_{\boldsymbol{w}}). \end{aligned} \tag{10}$$

Within the inner loops, the parameters of these three networks are fixed, and are used to generate the updates for the variables as the input of the variable update function. In the outer loops, we update the network parameters through backpropagation with respect to the accumulated global loss, given by

$$\mathcal{L}_F^s = \frac{1}{t_{up}} \sum_{t_s=(s-1)t_{up}+1}^{st_{up}} \omega_{t_s} F(\boldsymbol{u}_{t_s}, \boldsymbol{w}_{t_s}, \boldsymbol{V}_{t_s}), \tag{11}$$

where t_{up} is the update interval; that is, for every t_{up} outer loop iterations, the accumulated global loss $\mathcal{L}_F^s$ is used to update $\boldsymbol{\theta}_{\boldsymbol{V}}$, $\boldsymbol{\theta}_{\boldsymbol{u}}$ and $\boldsymbol{\theta}_{\boldsymbol{w}}$, $s = 1, 2, \ldots, S$, and $\omega_{t_s} \in \mathbb{R}_{\geq 0}$ denotes the weights associated with each outer loop step. $T = St_{up}$ denotes the maximum number of outer loop steps. Every t_{up} outer loop iterations, the parameters of the LSTM networks are updated by the Adam [15] optimizer using the accumulated global loss $\mathcal{L}_F^s$:

$$\begin{aligned} \boldsymbol{\theta}_{\boldsymbol{V}}^{s+1} &= \boldsymbol{\theta}_{\boldsymbol{V}}^{s} + \alpha_{\boldsymbol{V}} \cdot \mathrm{Adam}(\boldsymbol{\theta}_{\boldsymbol{V}}^{s}, \nabla_{\boldsymbol{\theta}_{\boldsymbol{V}}^{s}} \mathcal{L}_F^s), \\ \boldsymbol{\theta}_{\boldsymbol{u}}^{s+1} &= \boldsymbol{\theta}_{\boldsymbol{u}}^{s} + \alpha_{\boldsymbol{u}} \cdot \mathrm{Adam}(\boldsymbol{\theta}_{\boldsymbol{u}}^{s}, \nabla_{\boldsymbol{\theta}_{\boldsymbol{u}}^{s}} \mathcal{L}_F^s), \\ \boldsymbol{\theta}_{\boldsymbol{w}}^{s+1} &= \boldsymbol{\theta}_{\boldsymbol{w}}^{s} + \alpha_{\boldsymbol{w}} \cdot \mathrm{Adam}(\boldsymbol{\theta}_{\boldsymbol{w}}^{s}, \nabla_{\boldsymbol{\theta}_{\boldsymbol{w}}^{s}} \mathcal{L}_F^s), \end{aligned} \tag{12}$$

where $\alpha_{\boldsymbol{V}}$, $\alpha_{\boldsymbol{u}}$ and $\alpha_{\boldsymbol{w}}$ denote the learning rate for the LSTM networks $m_{\boldsymbol{V}}$, $m_{\boldsymbol{u}}$ and $m_{\boldsymbol{w}}$, respectively.

To strictly satisfy the total power constraint when neural networks are used to update the variables, we define $\mathcal{D} \triangleq \{\boldsymbol{V} | \mathrm{Tr}(\boldsymbol{V}\boldsymbol{V}^H) \leq P\}$ as the set of beamforming matrices that satisfy the power constraint. Then, we apply the following projection of the beamforming matrix $\boldsymbol{V}$ to set $\mathcal{D}$ in each outer loop step when the inner loop updating for $\boldsymbol{V}$ is completed.

$$\Omega_{\mathcal{D}}\{\boldsymbol{V}\} = \begin{cases} \boldsymbol{V}, & \text{if } \boldsymbol{V} \in \mathcal{D}, \\ \frac{\boldsymbol{V}}{||\boldsymbol{V}||_F} \sqrt{P}, & \text{otherwise}. \end{cases} \tag{13}$$

In conclusion, the MLBF algorithm builds a connection between the variable update functions and the global loss function. Specifically, at the outer loops, the three networks update their parameters with the objective of maximizing the WSR (equivalently minimizing the global loss function in (9)), which then determine the generation of the variable update terms with the inner loops. In this way, the knowledge of the global loss is reflected on the update rules of each variable within the inner loops; that is, $\boldsymbol{\theta}_{\boldsymbol{V}}$, $\boldsymbol{\theta}_{\boldsymbol{u}}$ and $\boldsymbol{\theta}_{\boldsymbol{w}}$ convey the global loss knowledge from the outer loops for the variable update functions. This allows the individual variable update functions to be less-greedy, and to return updates that are more aligned with the global loss function instead of the first-order stationary points of the corresponding sub-problems. Therefore, the learned updating strategy may not necessarily minimize the loss function of each sub-problem, but the overall problem could be better accommodated compared to the WMMSE algorithm. The general structure of the proposed MLBF algorithm for the WSR maximization problem is presented in **Algorithm** 1.

We highlight that, here the proposed MLBF algorithm is applied directly as a solution algorithm. For each WSR problem to be optimized, the LSTM networks' parameters of the implemented MLBF algorithm start from scratch and are trained during the iterative steps, such that the finally obtained variables are assigned as the solutions. We note that the computational complexity of the MLBF algorithm is higher than the WMMSE algorithm, nevertheless, it is directly used as a solution algorithm for WSR problem. In this way, the proposed MLBF approach is less time-consuming and resource-dependent to be applied than the previous deep-learning based approaches [9]–[12], as the training process is essentially embedded in the solution process while the traditional deep-learning based approaches have to train the parameters of the networks with a set of training samples for a certain period of time (which is typically quite long) before they can be used to solve a particular instance of the problem.

Algorithm 1: General Structure of the MLBF algorithm for WSR maximization problem.

```
1  Given: global loss function $F(\boldsymbol{u},\boldsymbol{w},\boldsymbol{V})$, $\boldsymbol{H}$.
2  Initialized: $\boldsymbol{V}_0$, $\boldsymbol{u}_0$, and $\boldsymbol{w}_0$.
3  for $t \leftarrow 1,2,\ldots,T$ do
4      $i,j,k,s = 1$
5      while $i \le I$ do
6          $\Delta\boldsymbol{u} = m_{\boldsymbol{u}}(\nabla f(\boldsymbol{u}^{(i-1)}), \boldsymbol{C}_{\boldsymbol{u}}^{(i-1)}, \boldsymbol{\theta}_{\boldsymbol{u}}^{s})$
7          $\boldsymbol{u}^{(i)} \leftarrow \boldsymbol{u}^{(i-1)} + \Delta\boldsymbol{u}$ $i = i+1$
8      end
9      $\boldsymbol{u}_t = \boldsymbol{u}^{(I)}$
10     generate $f(\boldsymbol{w})$with $\boldsymbol{u}_t, \boldsymbol{V}_{t-1}$
11     while $j \le J$ do
12         $\Delta\boldsymbol{w} = m_{\boldsymbol{w}}(\nabla f(\boldsymbol{w}^{(j-1)}), \boldsymbol{C}_{\boldsymbol{w}}^{(j-1)}, \boldsymbol{\theta}_{\boldsymbol{w}}^{s})$
13         $\boldsymbol{w}^{(j)} \leftarrow \boldsymbol{w}^{(j-1)} + \Delta\boldsymbol{w}$ $j = j+1$
14     end
15     $\boldsymbol{w}_t = \boldsymbol{w}^{(J)}$
16     generate $f(\boldsymbol{V})$with $\boldsymbol{w}_t, \boldsymbol{u}_t$
17     while $k \le K$ do
18         $\Delta\boldsymbol{V} = m_{\boldsymbol{V}}(\nabla f(\boldsymbol{V}^{(k-1)}), \boldsymbol{C}_{\boldsymbol{V}}^{(k-1)}, \boldsymbol{\theta}_{\boldsymbol{V}}^{s})$
19         $\tilde{\boldsymbol{V}}^{(k)} \leftarrow \boldsymbol{V}^{(k-1)} + \Delta\boldsymbol{V}$
20         $\boldsymbol{V}^{(k)} = \Omega_{\mathcal{D}}\{\tilde{\boldsymbol{V}}^{(k)}\}$
21         $k = k+1$
22     end
23     $\boldsymbol{V}_t = \boldsymbol{V}^{(K)}$
24     generate $f(\boldsymbol{u})$with $\boldsymbol{V}_t, \boldsymbol{w}_t$
25     $F(\boldsymbol{V}_t, \boldsymbol{u}_t, \boldsymbol{w}_t) \leftarrow \boldsymbol{u}_t, \boldsymbol{w}_t, \boldsymbol{V}_t$
26     while $s \le t/t_{up}$ do
27         $\mathcal{L}_F^s = \frac{1}{t_{up}} \sum_{t_s=(s-1)t_{up}+1}^{st_{up}} \omega_{t_s} F(\boldsymbol{V}_{t_s}, \boldsymbol{w}_{t_s}, \boldsymbol{u}_{t_s})$
28         $\boldsymbol{\theta}_{\boldsymbol{V}}^{s+1} = \boldsymbol{\theta}_{\boldsymbol{V}}^{s} - \alpha_{\boldsymbol{V}} \nabla_{\boldsymbol{\theta}_{\boldsymbol{V}}^{s}} \mathcal{L}_F^s$
29         $\boldsymbol{\theta}_{\boldsymbol{u}}^{s+1} = \boldsymbol{\theta}_{\boldsymbol{u}}^{s} - \alpha_{\boldsymbol{u}} \nabla_{\boldsymbol{\theta}_{\boldsymbol{u}}^{s}} \mathcal{L}_F^s$
30         $\boldsymbol{\theta}_{\boldsymbol{w}}^{s+1} = \boldsymbol{\theta}_{\boldsymbol{w}}^{s} - \alpha_{\boldsymbol{w}} \nabla_{\boldsymbol{\theta}_{\boldsymbol{w}}^{s}} \mathcal{L}_F^s$
31         $s = s+1$
32     end
33 end
```

V. Simulation Results

In this section, the performance of the proposed MLBF algorithm is evaluated by extensive simulations. MLBF algorithm is implemented in Python 3.6 with Pytorch 1.6. The WMMSE algorithm is also implemented in Python 3.6 for comparison. We consider the number of transmit antennas $M = 4$ and the number of single-antenna users $N = 4$, with user weights $\alpha_i = 1, \forall i$. We initialize the MLBF and WMMSE with the same initialization of $\boldsymbol{V}$ such that $\text{Tr}(\boldsymbol{V}\boldsymbol{V}^H) \le P$. Both methods are directly evaluated on 1000 channel realizations generated independent and identically distributed (i.i.d.) from a complex standard Gaussian distribution.

For the MLBF algorithm, we set the maximum number of outer loops to $T = 500$ when solving each problem and set the maximum number of inner loops to $K = I = J = 10$ when updating each variable within the inner loop. The update interval t_{up} is set to 5, thus the maximum update times within each sample in the test set is $S = 100$. Our meta-learners employ two-layer LSTM networks with 200 hidden units in each layer. The learning rates of Adam [15] for updating the LSTM networks are set to 10^{-4}. The weight of the outer loop step ω_{t_s} is set to 1 for all t_s. We note that our MLBF algorithm is trained during the optimization process and the final obtained results are presented to evaluate the performance. At the beginning of solving each problem, our networks start from scratch. In this way, the MLBF algorithm is implemented as a solution algorithm to the optimization problem at hand, relieved from the time-consuming training process.

Since complex variables are currently not supported by the deep learning tools, we decompose the complex variables into their real and imaginary parts. Thus we make an alternative reformulation that maps the complex computations to the equivalent computations in reals. Let the real matrix $\boldsymbol{V}^{'} = [\boldsymbol{V}_{re}, \boldsymbol{V}_{im}]$ denote the variables updated by the meta-learner network. We employ $\boldsymbol{M}_{re}$ and $\boldsymbol{M}_{im}$ as the masks matrices to recover the real and imaginary parts, respectively; that is, we set $\boldsymbol{V}_{re} = \boldsymbol{V}^{'}\boldsymbol{M}_{re}$ and $\boldsymbol{V}_{im} = \boldsymbol{V}^{'}\boldsymbol{M}_{im}$. Then, it is explicit that $\boldsymbol{V} = \boldsymbol{V}_{re} + j\boldsymbol{V}_{im}$, and the computations in the complex space, such as in (9), can be reformulated as real computations with matrices $\boldsymbol{V}_{re}$ and $\boldsymbol{V}_{im}$.

The vanilla WMMSE algorithm is implemented to solve the beamforming problem following the formulation in [11], including the stopping criterion that the deviation ϵ in the WSR between two iterations satisfies $\epsilon \le 10^{-4}$, and employing the bisection search method for computing μ. Differently from [11], to ensure the convergence of the WMMSE algorithm, we set the iterative steps L of the WMMSE algorithm to 100 (the maximum number of iterative steps for WMMSE is set to 6 in previous works [11], [12]). Furthermore, for each sample, we randomly initialize the algorithm for 10 times and choose the best performance for both of the tested algorithms as the obtained result to mitigate the potential impact of poor initializations.

In Fig. 2, we present the obtained WSR results of the MLBF and WMMSE algorithms averaged over 1000 channel realizations. The red solid line is the WSR results over iterations when using the MLBF algorithm and the dash blue line is from the WMMSE algorithm that is considered as the baseline. It shows that the MLBF algorithm clearly surpasses the WMMSE performance in the high SNR regime of SNR=20-40dB, and achieves comparable performance when SNR=10dB. It can be seen that, at higher SNR, the gap between the WSR obtained by the MLBF algorithm and the WMMSE algorithm becomes larger. In Fig. 3 we further show the impact of channel SNR on the obtained WSR by the MLBF and WMMSE algorithms, respectively. The red solid line with triangles denotes the averaged WSR obtained by the MLBF algorithm as a function of the channel SNR and the blue dash line with circles refers to the results from the WMMSE algorithm. Each point corresponds to the averaged WSR across 1000 channel realizations. It can be seen that when SNR$\leq$ 15dB, the WSR obtained by the MLBF and WMMSE algorithms are approximately equivalent. With the increase in SNR, the advantage of the MLBF algorithm is more significant. We infer that the low SNR allows both the MLBF and the WMMSE algorithms converge successfully to the optimal solution, thus both algorithms obtain similar WSR results. On the other hand, with the increasing SNR, the non-convexity in the WSR maximization problem is also amplified. Therefore, more local optimal points might exist in the geometry surface of the WSR maximization problem, and the WMMSE is surpassed by the MLBF algorithm significantly as the latter has a more effective search algorithm along the complex geometry.

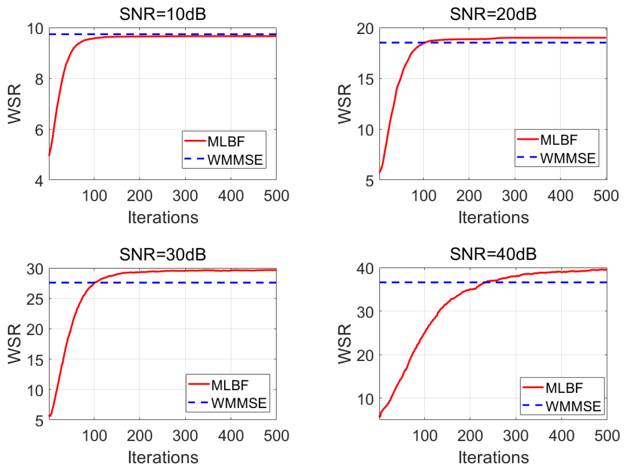

Fig. 2. Red curve denotes the average WSR obtained by the MLBF algorithm, while the blue dash line is the average WSR obtained by the WMMSE algorithm, with L=100. Both curves are the average WSR results across 1000 channel realizations.

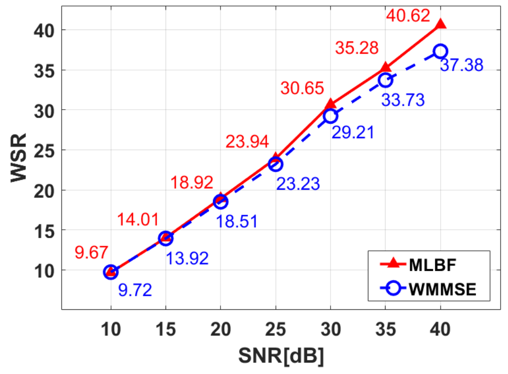

Fig. 3. WSR obtained by the MLBF and WMMSE algorithms as a function of the channel SNR. Each point represents the average WSR across 1000 channel realizations.

VI. Conclusion

In this paper, we proposed a meta-learning based solution to the WSR maximization problem in downlink MISO broadcast channels. The proposed MLBF algorithm is built upon the iterative block-coordinate descent method, but instead of greedily optimizing for each individual variable at each step, it can adaptively optimize each variable with respect to the geometry of the objective function. In this way, the proposed MLBF algorithm seeks to find a better update of individual variables for the NP-hard and non-convex WSR maximization problem compared to exhaustively minimizing each sub-problem. Simulation results confirm that the MLBF algorithm outperforms WMMSE particularly in the high SNR regime, and achieves a comparable performance when SNR is small. Future works will focus on extending the MLBF algorithm to the multiple-input multiple-out (MIMO) and interference channel scenarios, and the consideration of more general utility functions as studied in [7]. We expect that the gains from the proposed meta-learning approach will be even more significant in those problems due to the increased non-convexity and complexity.

VII. Acknowledgement

This work is supported by National Natural Science Foundation of China, projects 61921001 and 62022091, and by the European Research Council (ERC) project BEACON (grant no.677854). We also acknowledge the code from [11] for the WMMSE algorithm implementation that has been shared by the authors.

References

[1] Z.-Q. Luo and S. Zhang, "Dynamic spectrum management: Complexity and duality," *IEEE journal of selected topics in signal processing*, vol. 2, no. 1, pp. 57–73, 2008.

[2] C. B. Peel, B. M. Hochwald, and A. L. Swindlehurst, "A vector-perturbation technique for near-capacity multiantenna multiuser communication-part i: channel inversion and regularization," *IEEE Transactions on Communications*, vol. 53, no. 1, pp. 195–202, 2005.

[3] C. T. Ng and H. Huang, "Linear precoding in cooperative MIMO cellular networks with limited coordination clusters," *IEEE Journal on Selected Areas in Communications*, vol. 28, no. 9, pp. 1446–1454, 2010.

[4] M. G. Kibria, H. Murata, and S. Yoshida, "Coordinated linear precoding in downlink multicell mu-miso ofdma networks," in *2013 IEEE 78th Vehicular Technology Conference (VTC Fall)*. IEEE, 2013, pp. 1–5.

[5] S. S. Christensen, R. Agarwal, E. De Carvalho, and J. M. Cioffi, "Weighted sum-rate maximization using weighted MMSE for MIMO-bc beamforming design," *IEEE Transactions on Wireless Communications*, vol. 7, no. 12, pp. 4792–4799, 2008.

[6] D. A. Schmidt, C. Shi, R. A. Berry, M. L. Honig, and W. Utschick, "Minimum mean squared error interference alignment," in *2009 Conference Record of the Forty-Third Asilomar Conference on Signals, Systems and Computers*. IEEE, 2009, pp. 1106–1110.

[7] Q. Shi, M. Razaviyayn, Z.-Q. Luo, and C. He, "An iteratively weighted MMSE approach to distributed sum-utility maximization for a MIMO interfering broadcast channel," *IEEE Transactions on Signal Processing*, vol. 59, no. 9, pp. 4331–4340, 2011.

[8] 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 Transactions on Signal Processing*, vol. 66, no. 20, pp. 5438–5453, 2018.

[9] W. Xia, G. Zheng, Y. Zhu, J. Zhang, J. Wang, and A. P. Petropulu, "A deep learning framework for optimization of miso downlink beamforming," *IEEE Transactions on Communications*, vol. 68, no. 3, pp. 1866–1880, 2019.

[10] H. Huang, W. Xia, J. Xiong, J. Yang, G. Zheng, and X. Zhu, "Unsupervised learning-based fast beamforming design for downlink MIMO," *IEEE Access*, vol. 7, pp. 7599–7605, 2018.

[11] P. Lissy, M. Bengtsson, and J. Jaldén, "Deep unfolding of the weighted MMSE beamforming algorithm," *arXiv preprint arXiv:2006.08448*, 2020.

[12] A. Chowdhury, A. S. Gunjan Verma, Chirag Rao, and S. Segarra, "Unfolding wmmse using graph neural networks for efficient power allocation," *arXiv preprint arXiv:2009.10812*, 2020.

[13] Y. Yuan, G. Zheng, K.-K. Wong, B. Ottersten, and Z.-Q. Luo, "Transfer learning and meta learning based fast downlink beamforming adaptation," *IEEE Transactions on Wireless Communications*, 2020.

[14] M. Andrychowicz, M. Denil, S. Gomez, M. W. Hoffman, D. Pfau, T. Schaul, B. Shillingford, and N. De Freitas, "Learning to learn by gradient descent by gradient descent," in *Advances in neural information processing systems*, 2016, pp. 3981–3989.

[15] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[16] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[17] K. Li and J. Malik, "Learning to optimize," *arXiv preprint arXiv:1606.01885*, 2016.

[18] ——, "Learning to optimize neural nets," *arXiv preprint arXiv:1703.00441*, 2017.

[19] F. Chelsea and L. Sergey, "Meta-learning and universality: Deep representations and gradient descent can approximate any learning algorithm," *arXiv preprint arXiv:1710.11622*, 2017.

[20] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," in *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org, 2017, pp. 1126–1135.

[21] J. Xia, S. Li, J.-J. Huang, I. Jaimoukha, and X. Liu, "Meta-learning for multi-variable non-convex optimization problems: Iterating non-optimums makes optimum possible," *arXiv preprint arXiv:2009.04899*, 2020.