Learning to Optimize Non-Convex Sum-Rate Maximization Problems

Qingyu Song [1] **Guochen Liu** [2] **Hong Xu** [1]

Abstract

Solving optimization problems through machine learning is a promising research direction. In this position paper, we sketch a general framework motivated by first-order necessary conditions to solve non-convex sum-rate optimization problems arising from practical resource allocation problems in cellular networks. We construct two parameter matrices to update matrix-form decision variables of the given objective function. We inherently enhance the learning efficiency by increasing the dimensionality of decision variables with a learnable parameter matrix. Our preliminary evaluation shows that our approach achieves up to 98% optimality over state-of-the-art numerical algorithms while being up to 38× faster in various settings.

1. Introduction

Learning to Optimize (L2O) has garnered significant interest in AI for Science. Despite the remarkable progress in deep learning, their effectiveness in solving non-convex optimization remains limited. L2O can be categorized into two distinct regimes: online and offline. In the online learning regime, L2O methods usually serve as a parameter-updating strategy to improve the optimality of the solutions. Several deep learning techniques, such as LSTM (Chen et al., 2020) and model-based reinforcement learning (Li & Malik, 2017), have been employed. However, online learning has low efficiency: it works within the same iterative process as in conventional optimization algorithms (i.e., SGD (Robbins & Monro, 1951), WMMSE (Shi et al., 2011), etc.) when solving a problem instance, which renders its running time much longer than conventional methods (Xia & Gunduz, 2021). Thus we focus on offline learning, which strives to learn a mapping from a concrete problem instance to its potential optimal solutions offline so that the learned model can directly output a solution with one forward pass.

This work examines a fundamental non-convex optimization problem called sum-rate maximization that arises in many domains, in particular in wireless communications (Shi et al., 2011). Specifically, the problem is adjusting the channel and power allocation for downlink from the base stations to users to maximize the total throughput across users (Luo & Zhang, 2008).

The problem formulation is shown in Equation (1). We consider a cellular network with B base stations (BSes), each with N_t transmitting antennas. Each base station serves U users (UE), each equipped with N_r receiving antennas. The channel state between UE u and BS b is denoted by matrix $\mathbf{H}_{b,u} \in \mathbb{C}^{N_r \times N_t}, b = 1, \ldots, B, u = 1, \ldots, U$, while the decision variable (including beamforming and power allocation) is denoted by matrix $\mathbf{V}_{b,u}$. We optimize each $\mathbf{V}_{b,u} \in \mathbb{C}^{N_t \times N_r}$ (referred to as the precoding matrix) with the given channel state $\mathbf{H}$ and total power constraint P to maximize the sum rate of users. Notably, the non-convexity of the objective function arises from the inversion of the outer product, which captures the potential interference among all users, including intra-BS and inter-BS interference. This problem is known to be NP-hard (Luo & Zhang, 2008).

Instead of using a generic solver, a numerical algorithm known as the Weighted Mean-Square Error (WMMSE) algorithm (Shi et al., 2011) is widely regarded as the state-of-the-art method in terms of optimality to specifically tackle Problem (1). Generally, WMMSE formulates a dual problem with two types of auxiliary variables, enabling numerical solvability via a blocked coordinate descent algorithm. Consistent with prior work, we also benchmark our method against WMMSE.

In addition to conventional algorithms, deep learning has recently been explored to improve the efficiency and (hopefully) optimality of solving the non-convex rate maximization. There are two lines of work here in general. The first is a *learning-assisted* approach, which utilizes learning to reduce the empirical running time of WMMSE and other conventional methods. For instance, Hu et al. (2021)

[1]Department of Computer Science and Engineering, CUHK, Hong Kong, China [2]Decision Making and Reasoning Laboratory, Huawei Noah's Ark Lab, Shenzhen, China. Correspondence to: Qingyu Song <qysong21@cse.cuhk.edu.hk>, Guochen Liu <liuguochen1@huawei.com>, Hong Xu <hongxu@cuhk.edu.hk>.

Accepted after peer-review at the 1st workshop on Synergy of Scientific and Machine Learning Modeling, SynS & ML ICML, Honolulu, Hawaii, USA. July, 2023.

$$\max_{\mathbf{V}} \sum_{b=1}^{B}\sum_{u=1}^{U} \log_2 \det\left(\mathbf{I} + \left(\mathbf{H}_{b,u}\mathbf{V}_{b,u}\right)\left(\mathbf{H}_{b,u}\mathbf{V}_{b,u}\right)^H \left(\sum_{\tilde{b},\tilde{u};\tilde{b},\tilde{u}\neq b,u}^{B,U} \left(\mathbf{H}_{\tilde{b},u}\mathbf{V}_{\tilde{b},\tilde{u}}\right)\left(\mathbf{H}_{\tilde{b},u}\mathbf{V}_{\tilde{b},\tilde{u}}\right)^H + \sigma^2\mathbf{I}\right)^{-1}\right) \quad (1)$$
$$\text{s.t.} \sum_{u=1}^{U} \mathrm{Tr}\left(\mathbf{V}_{b,u}\mathbf{V}_{b,u}^H\right) \leq P, b = 1, 2, \ldots, B.$$

employ two parameter matrices to approximate the matrix inversions in WMMSE, while Schynol & Pesavento (2023) replace the $\mathbf{W}$-update and $\mathbf{V}$-update steps in WMMSE with graph convolution neural networks. The experimental results in these works achieve more than 90% optimality of WMMSE and offer salient speedups, especially for large problem sizes.

Nonetheless, it is essential to note that learning-assisted methods inherit the same iterative solving process of WMMSE (or other conventional optimization algorithms), and the neural networks are used to replace specific calculation steps in this process. Their performance is intrinsically bounded by WMMSE's inherent limitations. The optimality is clearly upper-bounded by WMMSE. Moreover, there is no robustness guarantee due to the random initialization. Although the most recent WMMSE variant proposed by Zhao et al. (2023) utilizes interference zero-forcing (Gao et al., 2011) in conjunction with singular value decomposition, it remains non-deterministic.

In response to the limitations above, recent studies have explored *learning-only* approach that does not follow the framework of conventional optimization algorithms. Various neural network techniques, such as vanilla DNN (Sun et al., 2018), deep reinforcement learning (Zhao et al., 2021), and graph neural networks (Shen et al., 2021; 2022), have been employed in this regard. However, most methods oversimplify the problem in Equation (1) by degenerating the decision variable $\mathbf{V}_{b,u}$ from a matrix to a scalar (i.e., no beamforming). Consequently, in our practice, the model presented in (Shen et al., 2021) shows less than 70% optimality of WMMSE for the original matrix form.

To our best knowledge, current learning-only approaches ignore the intrinsic structure and characteristics of the problem. They essentially use black-box models to approximate the solutions to Equation (1). Our method is based on two key insights. First, motivated by the workflow of numerical optimization algorithms, we adopt two parameter matrices to update each matrix-form solution $\mathbf{V}_{b,u}$ by matrix multiplications on both sides. We construct these parameter matrices using DNN models with selected intermediate results of the objective function. Second, motivated by orthogonalization-based interference reduction methods in wireless MIMO communications (Gao et al., 2011), we improve the solvability of Equation (1) by mapping its solution space to a specific higher dimensional space such that interference reduction can be readily applied to solve it. Another DNN is then used to learn the mapping from this higher dimensional space back to the original space. We show that our method is able to provide over 90% optimality of WMMSE and comparable efficiency to learning-assisted algorithms, outperforming all existing learning-only methods.

2. Design: A Structured Learning Framework

In this section, we introduce our end-to-end structured learning framework as depicted in Figure 1. Briefly, in each training step after initialization, the initial solution $\mathbf{V}_0$ and the channel state $\mathbf{H}$ are transformed into a higher dimensional space by a learnable matrix $\mathbf{D}$ for solving. In this higher dimensional space, the solution $\mathbf{V}_0$ is improved for a fixed number of iterations according to $\mathbf{H}$. The obtained solution is then projected back to the original space with $\mathbf{D}$ again so we can evaluate the loss and gradients, and this completes one training step. Our framework has two key novelties, both of which involve the use of machine learning: First, structured solution updates. Each update to the solution variable $\mathbf{V}$ is done in the form of two matrix multiplications on its left and right-hand side with $\mathbf{W}_L$ and $\mathbf{W}_R$, both learned from two DNNs of the same structure. Second, the dimension expanding and shrinking with $\mathbf{D}$, which is also learned using DNN. For simplicity, throughout this work, we use fully connected layers unless stated otherwise. We explain more details of the two key novelties below.

2.1. Structured Solution Updates with DNN

Our structured updates to the solution $\mathbf{V}$ are motivated by the first-order necessary condition in the WMMSE algorithm. WMMSE performs a matrix inversion with several matrix multiplications of intermediate solutions from the last iteration. We mimic this with two learnable parameter matrices generated by DNNs. Specifically, given a feasible solution $\mathbf{V}_{b,u}^{t-1}$ at iteration t, we utilize two squared parameter matrices denoted as $\mathbf{W}_L^t \in \mathbb{C}^{N_t \times N_t}$ and $\mathbf{W}_R^t \in \mathbb{C}^{N_t \times N_r}$ to update the solution by $\mathbf{W}_L^t \mathbf{V}_{b,u}^{t-1} \mathbf{W}_R^t$. We employ several neural networks to construct $\mathbf{W}_L^t$ and $\mathbf{W}_R^t$ in the initialization and solution update phases.

Initialization. In Equation (1), each $\mathbf{V}_{b,u}$ is multiplied with $\mathbf{H}_{b,u}$ in the numerator and multiplied with $\mathbf{H}_{\tilde{b},\tilde{u}}^H, \tilde{b}, \tilde{u} \neq b, u$ in the denominator. Such fractions imply that an ideal decision variable $\mathbf{V}_{b,u}$ should achieve parallelism in all existences on the numerator and orthogonality in all existences on the denominator. Moreover, since each $\mathbf{V}_{b,u}$ is linearly

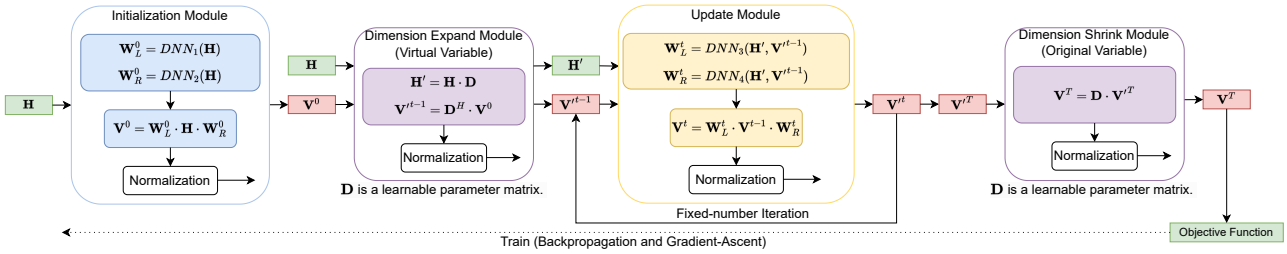

Figure 1: Overall framework of our proposed model.

multiplied with related $\mathbf{H}$, the value of $\mathbf{H}_{b,u}\mathbf{H}^H_{\tilde{b},\tilde{u}}, \tilde{b}, \tilde{u} \neq b, u$ illustrates the extend that $\mathbf{V}_{b,u}$ achieves. Hence, we add the value of such product along with original $\mathbf{H}_{b,u}$ and $\mathbf{H}^H_{\tilde{b},\tilde{u}}, \tilde{b}, \tilde{u} \neq b, u$ to the input of the DNN models. Our initialization module achieves up to 80% optimality of WMMSE if it is standalone trained. Our initialization only takes $\mathbf{H}$ as input and can be readily implemented in a distributed setting with a single communication among base stations.

Solution Update. In this phase, we improve each initial $\mathbf{V}^0_{b,u}$ using a fixed number of iterations through another DNN. This approach allocates learning tasks of improving each $\mathbf{V}_{b,u}$ to each iteration.

We construct the inputs of DNN models as the values of $\mathbf{H}_{b,u}\mathbf{V}^{t-1}_{b,u}$ in the numerator of the objective function and $\mathbf{H}_{b,u}\mathbf{V}^{t-1}_{\tilde{b},\tilde{u}}, \tilde{b}, \tilde{u} \neq b, u$ in the denominator, along with the corresponding original matrices.

To achieve multi-variable coordination explicitly, we introduce a message-passing neural network (MPNN) (Gilmer et al., 2017) that collects and aggregates the parameter matrices $\mathbf{W}_L^{t-1}$ and $\mathbf{W}_R^{t-1}$ of all b, u, using a sum-pooling method. From the perspective of one single user, the MPNN collects the parameter matrices of all other users, i.e., the current base station's all other users and other base stations' users. Then, the MPNN aggregates such matrices with the mean-pooling method, where we calculate the mean values of parameter matrices of all other users. We utilize two fully connected layers with GELU activation functions to enhance such aggregation by learning.

2.2. Dimension Expanding and Shrinking

As introduced, we raise the dimension of the original problem in Equation (1), i.e., coefficient $\mathbf{H}$ and solution $\mathbf{V}$, and try to find a solution in the higher dimensional space to improve efficiency. First, based on Theorem A.1 and Corollary A.2 in Appendix A.1, it is sufficient to construct a feasible solution of Equation (1) by finding a feasible solution in a higher dimensional space and then projecting it back to original space. For example, any arbitrary one fat unitary matrice is feasible since they keep values of the matrix multiplications in Equation (1). Second, finding a (near) optimal solution in a higher dimensional space is more solvable than in the original space as long as the dimension of the new space is sufficiently large. This is essentially due to the characteristics of MIMO communications: when the transmitting antennas are sufficient, it is possible to find a precoding matrix $\mathbf{V}$ that effectively cancels the interference. Appendix A.2 has more detailed explanations.

We perform the dimension expanding and shrinking by matrix multiplications with a learnable parameter matrix $\mathbf{D}$. We utilize a learning approach to construct feasible solutions in higher dimensional space. Thus, they are integrated into one end-to-end learning process. We further eliminate the unitary constraint on the dimension-transformation matrix $\mathbf{D}$ in Theorem A.1. The process achieves a trade-off between finding a feasible solution in a higher dimensional space and transforming it into a feasible solution in the original space. We discuss its potential effectiveness in Appendix A.3.

Finally, note that we apply a normalization step in each phase of our framework as in Figure 1 to ensure that the intermediate solutions are feasible.

2.3. Module Architectures

Our approach employs an adaptive model construction tailored to the dimension scalings defined in Equation (1). For initialization module, the output dimension of the DNN model feeding $\mathbf{H}_{b,u}\mathbf{H}^H_{\tilde{b},\tilde{u}}, \tilde{b}, \tilde{u} \neq b, u$, $\mathbf{H}_{b,u}$, and $\mathbf{H}_{\tilde{b},\tilde{u}}$ are $N_t/4$ times, N_t times, and N_t times of their input dimensions respectively. For the embeddings of $\mathbf{H}_{b,u}\mathbf{H}^H_{\tilde{b},\tilde{u}}$ and $\mathbf{H}_{\tilde{b},\tilde{u}}$, they are sum-pooled to keep an identical shape with embedding of $\mathbf{H}_{b,u}$. This manipulation also achieves permutation-equivalence with different numbers and orders of users.

Further, we utilize two three-layer fully connected models to generate vector-form $\mathbf{W}_L^0$ and $\mathbf{W}_R^0$ from the concatenation of all embeddings. They are then reshaped to N_t-by-N_t and N_r-by-N_r dimensions matrices as our solution updating framework parameters. The dimensions of the output layers are N_t^2-N_t^2-$N_t^2 t$ and $N_r^2$8-$N_r^2 * 4$-N_r^2 respectively.

For update module, the dimension expandings for $\mathbf{H}_{b,u}\mathbf{V}_{b,u}$ $\mathbf{H}_{\tilde{b},\tilde{u}}\mathbf{V}_{b,u}, \tilde{b},\tilde{u} \neq b,u$, $\mathbf{V}_{b,u}$, and $\mathbf{V}_{\tilde{b},\tilde{u}}$ are 4, 4, $N_t/2$, and $N_t/2$ respectively. We apply one layer MPNN with another one fully-connected layer to output vector-form $\mathbf{W}_L^t$, which takes all concatenated embeddings as input. The output dimensions are $(N_r * U_b * B)^2$-$(N_r * U_b * B)^2$. For the DNN model for $\mathbf{W}_R^t$, we apply the same architecture of as that for $\mathbf{W}_R^0$.

3. Preliminary Evaluation

We conduct experiments using Intel Xeon Gold 5320 CPU and an NVIDIA RTX 3090 GPU. Our proposed framework is evaluated in simulated scenarios of varying scales, including small, moderate, and large. These scales are defined by eight, sixteen, and thirty-two $\mathbf{V}$ arranged on 8-by-2, 16-by-2, and 32-by-2 matrix spaces, respectively.

The datasets for the three scales have 100k, 120k, and 120k samples, respectively, with a ratio of 8:1:1 for training, evaluation, and testing. We construct the loss function as negative objective value of Problem (1). The mini-batch sizes are 512, 512, and 200, respectively. The training optimizer is AdamW. The learning rate is set to 0.001 and decays with epochs to 0.00001. We train 400 epochs for all scenarios.

To generate data ($\mathbf{H}$ in Equation (1)), we follow the configurations outlined in the latest WMMSE paper (Zhao et al., 2023). Moreover, we consider two scenarios with distinct inter-cell interference levels based on inter-base station distances. We set the inter-base station distance for the small inter-cell interference scenario to 2.8 km (Zhao et al., 2023). For the large inter-cell interference scenario, we set it to 0.5 km. The scalar in constraints is set to be $P = 10[W]$. The noise power for each base stations b is set to be $\sigma_b^2 = 10^{\frac{1}{U}\sum_u \log_{10}\frac{1}{N_r}\|\mathbf{H}_{b,u}\|_F^2} \times 10^{-\frac{\mathrm{SNR}}{10}}$ (Zhao et al., 2023), where the SNR is set to be $\mathrm{SNR} = 5[dB]$. Input channels (coefficients of Equation (1)) are randomly sampled upon circularly-symmetric standard complex Gaussian distribution with path loss between the users and the base stations. We set the users to be uniformly located in a 0.1 $\sim$ 0.3 km range to its base stations. The WMMSE algorithm is implemented with a threshold of 0.01 for iteration termination and a maximum iteration limit of 50.

Our complex-number neural network implementations are based on an open-source codebase developed by Popoff (2022) on PyTorch (Paszke et al., 2019). In our initialization module, each matrix-form input is flattened and fed into a unique fully-connected embedding layer, producing higher-dimensional outputs (embeddings). These embeddings are then concatenated and used to create two distinct multi-layer fully-connected models, which generate $\mathbf{W}_L^0$ and $\mathbf{W}_R^0$. We similarly constructed our update module. For a more detailed description of our model architecture, please refer to Appendix 2.3.

	Small	Moderate	Large
Small Interference	98.6%	96.9%	93.0%
Large Interference	96.9%	95.7%	91.5%

Table 1: Percentage of optimal objective value achieved of our method over WMMSE (SOTA). Columns indicate different problem scales; rows different inter-cell interference levels.

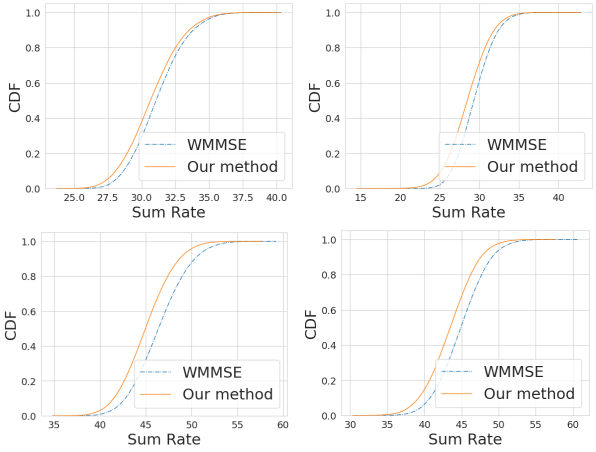

Figure 2: Empirical cumulative distribution function of objective values across two inter-cell interference scenarios in small scale and moderate scale. Left: Small interference scenarios. Right: Large interference scenarios.

We first evaluate the optimality attained by our proposed approach. We compare the objective values, i.e., sum rates, in Figure 2 and Table 1. Our findings indicate that our method achieves over 90% optimality compared to WMMSE. Moreover, we observe that larger inference scenarios result in higher variances of the input coefficients in Equation (1), leading to a marginal decline in performance when compared to smaller scenarios.

To evaluate the empirical efficiency of our proposed method, we quantify the execution time by determining the duration

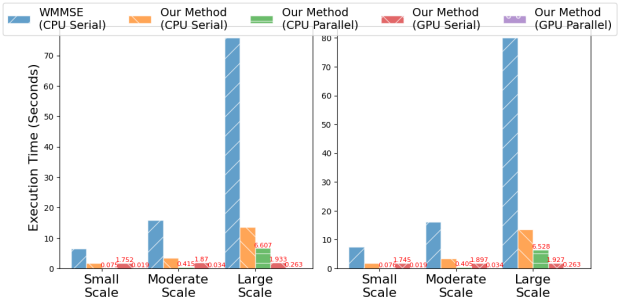

Figure 3: Execution time (seconds) on moderate scale scenario. 'Serial' means each sample executes individually. 'Parallel' means 100 samples execute concurrently.

required to solve 100 problems with varying coefficients $\mathbf{H}$ in Equation (1). Additionally, we conduct a parallel execution time assessment of the 100 problems for our proposed method. In contrast, the WMMSE algorithm cannot achieve parallel execution due to the impossibility of having the same number of iteration rounds in all cases.

Figure 3 presents the average time the WMMSE consumes and our proposed method over ten runs. Our structured framework employs five iterations during the updating phase, whereas the WMMSE algorithm requires an average of 28.16, 28.47, and 24.23 iterations across three different scales, respectively. Our proposed method achieves a $5\times$ speed-up over the WMMSE algorithm across all three scenarios when executed on a CPU. When run on a GPU, the speed-up over the WMMSE algorithm is $5\times$, $8\times$, and $38\times$. Moreover, parallelizing 100 samples on a GPU leads to a substantial acceleration of hundreds of times over the WMMSE algorithm.

4. Conclusion

This study proposes a structured framework to learn a mapping from given coefficients to optimal solutions of the non-convex sum rate maximization problem. Our approach involves using manually constructed parameter matrices to achieve a fixed-number updating process on each decision variable with matrix multiplications, where we employ DNN models to non-linearly generate the parameter matrices. To systematically construct the DNN models, we base them on the objective function. Additionally, we utilize another learnable parameter matrix to transform the optimization problem's dimensionalities to improve the solvability of the non-convex problem. We evaluate the effectiveness of our proposed method on several synthetic datasets. The preliminary experimental results demonstrate that our method achieves up to 98% optimality compared to state-of-the-art conventional optimization algorithms, with up to 38 $\times$ faster execution times.

5. Acknowledgment

This work is supported in part by funding from Huawei (CUHK #7010691).

6. Broader impact

Despite the impressive performance of large black-box models like GPT in computer vision and natural language processing, we argue that precise and systematic manipulations are still essential. This study introduces a systematic deep-learning design to solve a mathematical optimization problem. Our findings offer insights into how deep learning algorithms can be enhanced to tackle continuous non-convex optimization problems more effectively, and we hope they will inspire further research in this area.

References

Chen, T., Zhang, W., Zhou, J., Chang, S., Liu, S., Amini, L., and Wang, Z. Training stronger baselines for learning to optimize. In *NeurIPS*, 2020.

Clerckx, B. and Oestges, C. Chapter 12 - multi-user mimo. In Clerckx, B. and Oestges, C. (eds.), *Mimo Wireless Networks (Second Edition)*, pp. 419–523. Academic Press, Oxford, second edition edition, 2013. URL https://www.sciencedirect.com/science/article/pii/B9780123850553000122.

Gao, X., Edfors, O., Rusek, F., and Tufvesson, F. Linear Pre-Coding Performance in Measured Very-Large MIMO Channels. In *2011 IEEE Vehicular Technology Conference (VTC Fall)*, pp. 1–5, 2011.

Gilmer, J., Schoenholz, S. S., Riley, P. F., Vinyals, O., and Dahl, G. E. Neural message passing for quantum chemistry. In *ICML*, pp. 1263–1272, 2017.

Hu, Q., Cai, Y., Shi, Q., Xu, K., Yu, G., and Ding, Z. Iterative Algorithm Induced Deep-Unfolding Neural Networks: Precoding Design for Multiuser MIMO Systems. *IEEE TWC*, 20(2):1394–1410, 2021.

Li, K. and Malik, J. Learning to optimize. In *ICLR*, 2017.

Luo, Z.-Q. and Zhang, S. Dynamic Spectrum Management: Complexity and Duality. *IEEE JSAC*, 2(1):57–73, 2008.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *NeurIPS*, pp. 8024–8035. 2019.

Popoff, S. M. complexpytorch. https://github.com/wavefrontshaping/complexPyTorch, 2022.

Robbins, H. and Monro, S. A stochastic approximation method. *The annals of mathematical statistics*, pp. 400–407, 1951.

Schynol, L. and Pesavento, M. Coordinated Sum-Rate Maximization in Multicell MU-MIMO With Deep Unrolling. *IEEE JSAC*, 41(4):1120–1134, 2023.

Shen, Y., Shi, Y., Zhang, J., and Letaief, K. B. Graph Neural Networks for Scalable Radio Resource Management: Architecture Design and Theoretical Analysis. *IEEE JSAC*, 39(1):101–115, 2021.

Shen, Y., Zhang, J., Song, S., and Letaief, K. B. Graph Neural Networks for Wireless Communications: From Theory to Practice. *IEEE TWC*, pp. 1–1, 2022.

Shi, Q., Razaviyayn, M., Luo, Z. Q., and He, C. An iteratively weighted MMSE approach to distributed sum-utility maximization for a MIMO interfering broadcast channel. *IEEE TWC*, 59(9):4331–4340, 2011.

Sun, H., Chen, X., Shi, Q., Hong, M., Fu, X., and Sidiropoulos, N. D. Learning to optimize: Training deep neural networks for interference management. *IEEE TSP*, 66 (20):5438–5453, 2018.

Xia, J. and Gunduz, D. Meta-learning based beamforming design for MISO downlink. In *2021 IEEE International Symposium on Information Theory (ISIT)*, pp. 2954–2959, 2021.

Zhao, X., Lu, S., Shi, Q., and Luo, Z.-Q. Rethinking WMMSE: Can Its Complexity Scale Linearly With the Number of BS Antennas? *IEEE TSP*, 71:433–446, 2023.

Zhao, Y., Niemegeers, I. G., and De Groot, S. M. Dynamic Power Allocation for Cell-Free Massive MIMO: Deep Reinforcement Learning Methods. *IEEE Access*, 9: 102953–102965, 2021.

A. Appendix

A.1. Theorem and Corollary

Suppose $\mathbf{V}_{b,u} \in \mathbb{C}^{N_t \times N_r}$. Suppose a parameter matrix $\mathbf{D} \in \mathbb{C}^{N_t \times (N_r U_b B)}$ and $\mathbf{D}\mathbf{D}^H = \mathbf{I}$. Then, $\mathbf{H}'_{b,u} = \mathbf{H}_{b,u}\mathbf{D}, b = 1, \ldots, B, u = 1, \ldots, U$ generates a new problem in higher dimensional space from Problem (1). We have the following simple theorem:

Theorem A.1. *If $\mathbf{V}_{b,u}, b = 1, \ldots, B, u = 1, \ldots, U$ is a feasible solution of original space, $\mathbf{D}^H\mathbf{V}_{b,u}$ is a feasible solution of the new space.*

The Theorem A.1 demonstrates that for any objective value of the original problem, there exists a feasible solution in a higher dimensional space to achieve the same value.

Suppose $\mathbf{V}'_{b,u} \in \mathbb{C}^{(N_r U_b B) \times N_r}$, based on Theorem A.1, we derive the following corollary:

Corollary A.2. *If $\mathbf{V}'_{b,u}$ is a feasible solution, $\mathbf{V}_{b,u} = \mathbf{D}\mathbf{V}'_{b,u}, \mathbf{V}_{b,u} \in \mathbb{C}^{N_t \times N_r}$ is a feasible solution of original Problem (1).*

The Corollary A.2 demonstrates that any feasible solution in the higher dimensional space can be transformed into a feasible solution in Problem (1) in the original space.

A.2. Discussion of Solvability

Increasing the dimensionalities of the decision variable enhances solvability. The zero-forcing (Gao et al., 2011) method demonstrates that Problem (1) can be solved through singular value decomposition (SVD) when the number of rows of decision variables $\mathbf{V}$ is greater than or equal to $N_r \times U_b \times B$ (Shi et al., 2011).

Specifically, define the interference space as $[\mathbf{H}^H_{\tilde{b},\tilde{u}}, \tilde{u} \neq b, u]^H \in \mathbb{C}^{N_r(BU-1) \times N_t}$, the condition ensure the equation $[\mathbf{H}^H_{\tilde{b},\tilde{u}}, \tilde{u} \neq b, u]^H\mathbf{V}_{b,u} = 0$ has more than one solutions. It is feasible to find a basis, denote as $\tilde{\mathbf{V}}_{b,u}$, in the null space of $[\mathbf{H}^H_{\tilde{b},\tilde{u}}, \tilde{u} \neq b, u]^H$ with SVD. Further, define the target space as $\mathbf{H}_{b,u} \in \mathbb{C}^{N_r \times N_t}$, another basis, denote as $\mathbf{V}'_{b,u}$, is also achievable with SVD. Hence, a feasible $\mathbf{V}_{b,u}$ is calculated by $\mathbf{V}_{b,u} = \mathbf{V}'_{b,u}\tilde{\mathbf{V}}_{b,u}$. The detailed demonstrations are in (Clerckx & Oestges, 2013).

A.3. Discussion of Trade-Off

Our end-to-end learning process intuitively involves a trade-off among the proposed three steps. First, infinite optimal solutions exist in the increased dimensional space. For instance, the basis of null space by SVD is not unique. Hence, it is necessary for the linear mapping with $\mathbf{D}$ to be sufficiently robust to map all of these optimal solutions to the optimal solutions of the original Problem (1). It is only possible if the original problem has a finite number of optimal solutions. Even if the number of optimal solutions in the original space is infinite, achieving a mapping $\mathbf{D}$ that builds an equivalence between NP-hardness and P-complement is generally impossible. We use our structured learning framework to construct a single feasible solution to avoid the dilemma of multiple optimal solutions and the corresponding many-to-one dimension-transformation problem. The optimality is implicitly achieved by training.