DEEP INITIALIZATION FOR GUARANTEED UNIMODULAR QUADRATIC PROGRAMMING

Amrutha Varshini Ramesh

Department of Computer Science,
University of British Columbia, Canada

Mojtaba Soltanalian

ECE Department,
University of Illinois at Chicago, USA

ABSTRACT

In this work, we study a deep learning-based initialization approach for unimodular quadratic programs (UQPs), that are concerned with the maximization of a quadratic form over a set of complex unimodular vectors. UQPs have shown prevalent presence in many signal processing and design problems such as in wireless communications and active sensing. Stemming from their NP-hard nature, prior works on UQPs have focused on proposing approximate solutions that generally traded-off speed for obtaining theoretically sound approximations. With the aim of improving the computational efficiency of existing UQP solvers and equipped with highly-scalable deep learning frameworks as a backbone, we propose a novel hybrid solver, which we refer to as Deep-INIT. The proposed data-driven initialization approach makes use of deep learning to automatically learn "good" initializations for an underlying model-based solver called MERIT, that provides strong optimality guarantees for UQP solutions; thereby, speeding up an existing optimality-certificate producing solver for UQPs. In fact, apart from achieving a significant speed-up over the underlying UQP solver, a fundamental characteristic of Deep-INIT is that it preserves the guarantees that emerge from the model-based solver. Our numerical results reaffirm the speed-up potential that Deep-INIT offers.

Index Terms— Complex quadratic programming, deep learning, non-convex optimization, optimality guarantees.

1. INTRODUCTION

Optimal signal design for signal transmission plays an important role in improving the overall performance of active sensing and communication systems. Specifically, unimodular signals (also known as constant-modulus signals) are used to help maximize the signal-to-noise ratio (SNR) of such systems while maintaining an optimal (*i.e.*, unity) peak-to-average-power ratio (PAR). For instance, radar systems widely use unimodular signals to satisfy their transmission requirements and perform effective range compression [1, 2]. Due to such a wide applicability of unimodular signals in practice, many works in the past have studied communication or sensing performance optimization over unimodular signals, usually through unimodular quadratic programming (UQP), and have developed computational algorithms to efficiently compute their solutions or approximate them [1, 3–16]. A UQP is often formulated as a maximization of a quadratic form over a set of complex unimodular vectors; more precisely,

$$\max_{\mathbf{s}\in\Omega^n} \ \mathbf{s}^H\mathbf{R}\mathbf{s} \tag{1}$$

where $\mathbf{R} \in \mathbb{C}^{\mathrm{n}\times\mathrm{n}}$ is a given Hermitian matrix, and $\mathbf{s}$ is a complex unimodular vector, with each element lying on the unit circle $\Omega = \{s : \ |s| = 1\}$. This problem can be shown to be NP-hard in general by applying a reduction from a well-known NP-complete matrix partitioning problem [7].

1.1. Approximate Solutions to UQP

As mentioned earlier, several approximation methods have been studied for solving UQPs. Such methods carefully construct reformulations of the UQP problem (1) in such a way that they may result in approximate solutions are *guaranteed* to be somewhat close to the true UQP solution. As a result, they may come with strong theoretical properties, usually in the form of sub-optimality guarantees. One of the popular methods for approximating UQP solutions is semidefinite relaxation (e.g. [6–16]), which are shown to have a sub-optimality guarantee of $\pi/4$. The associated sub-optimality is defined as: $\gamma = v_{SDR}/v_{opt}$, where γ is the sub-optimality coefficient, and v_{SDR} and v_{opt} denote the objective values at the obtained solution from SDR and at the global optimum.

1.2. MERIT for UQP

Recently, another work provided an iterative solution to UQPs, referred to as MERIT [1], or monotonically error-bound improving technique for mathematical optimization, leading to tighter sub-optimality guarantees than those offered by SDR. MERIT aims to successively approximate (and get closer to) the original UQP problem instance and its global optimum of interest via constructing a sequence of problem instances whose corresponding global optima are readily known. Notably, the authors of [1] were able show that the MERIT solution for UQP enjoys data-dependent sub-optimality guarantees,

This work was supported in part by the NSF Grant ECCS-1809225.
Correspondence: avramesh@cs.ubc.ca

which in most cases are better than the *a priori* known analytical guarantees.

Since this work will rely heavily on the MERIT formulation, we present some of the required details in the following. The authors of [1] begin their study by defining a convex cone $\mathcal{K}(\mathbf{s})$ of all input matrices $\mathbf{R}$ for which a given $\mathbf{s}$ is the global optimizer of the UQP. Then, they develop an approximate characterization of $\mathcal{K}(\mathbf{s})$, using two other convex cones $\mathcal{C}_{\mathbf{s}}$ and $\mathcal{C}(\mathbf{V}_{\mathbf{s}})$, where: 1) $\mathcal{C}_{\mathbf{s}}$ represents the convex cone of all matrices for which $\mathbf{s}$ is the dominant eigenvector, and, 2) $\mathcal{C}(\mathbf{V}_{\mathbf{s}})$ represents a convex cone of matrices $\boldsymbol{V}_s = \boldsymbol{D} \odot (\boldsymbol{s}\boldsymbol{s}^H)$ where $\boldsymbol{D}$ is any real-valued symmetric matrix with non-negative off-diagonal entries. Specifically, [1] derives the following relation addressing the accuracy of this characterization:

$$\mathbf{R} + \alpha_0 \mathbf{s}\mathbf{s}^H = \mathbf{Q}_{\mathbf{s}} + \mathbf{P}_{\mathbf{s}} = (\mathbf{Q}_{\mathbf{1}} + \mathbf{P}_{\mathbf{1}}) \odot \mathbf{s}\mathbf{s}^H \qquad (2)$$

where $\mathbf{Q}_{\mathbf{s}} \in \mathcal{C}_{\mathbf{s}}$, $\mathbf{P}_{\mathbf{s}} \in \mathcal{C}(\mathbf{V}_{\mathbf{s}})$, $\mathbf{1}$ is the all-one vector, $\odot$ denotes the Hadamard product operator, and $\alpha_0 \geq 0$. The vector $\mathbf{s}$ that satisfies (2) is guaranteed to be the global optimum of the UQP associated with $\mathbf{R}$ when $\alpha_0 = 0$, and a local optimum when $\alpha_0 > 0$. Note that the smaller the α_0, the stronger the characterization.

In the case of $\alpha_0 = 0$, one deals with the following optimization problem:

$$\min_{\mathbf{s}\in\Omega^n, \mathbf{Q}_{\mathbf{1}}\in\mathcal{C}_\infty, \mathbf{P}_{\mathbf{1}}\in\mathcal{C}(\mathbf{V}_{\mathbf{1}})} ||\mathbf{R} - \underbrace{(\mathbf{Q}_{\mathbf{1}} + \mathbf{P}_{\mathbf{1}}) \odot \mathbf{s}\mathbf{s}^H}_{\triangleq \mathbf{R}_{\mathbf{s}}}||_F \qquad (3)$$

The MERIT algorithm, solves (3) by first optimizing for $\mathbf{Q}_{\mathbf{1}}$ and $\mathbf{P}_{\mathbf{1}}$, which is a convex problem and then uses *power method-like iterations* [1,4] to approximate $\mathbf{s}$ with the optimal $\mathbf{Q}_{\mathbf{1}}$ and $\mathbf{P}_{\mathbf{1}}$ that have been obtained. In the case of $\alpha_0 > 0$, MERIT finds the local optimum discussed below (2) iteratively by defining $\mathbf{R}' = \mathbf{R} + \alpha_0 \mathbf{s}\mathbf{s}^H$ for increasing values of α_0 (found using a bisection algorithm) and solving the objective (3) with $\mathbf{R} = \mathbf{R}'$. The obtained value of α_0 is then used to provide data-dependent sub-optimality guarantees, given as [1]:

$$\gamma = \frac{\mathbf{s}^H \mathbf{R} \mathbf{s}}{\mathbf{s}^H \mathbf{R}_{\mathbf{s}} \mathbf{s}} = 1 - \frac{\alpha_0 n^2}{\mathbf{s}^H \mathbf{R}_{\mathbf{s}} \mathbf{s}}.$$

where n is the length of $\mathbf{s}$.

1.3. Contributions of This Work

There is a strong need for UQP solvers that are both computationally efficient and reliable. In recent years, deep learning-based methods have been deployed in an overwhelming number of engineering applications. However, because of their black-box nature, they are generally devoid of the theoretical advantages that model-based methods often provide [19, 20]. As discussed before, this work aims to speed-up the existing model-based methods while preserving their theoretical properties. In order to achieve this goal, we focus on a smart initialization of MERIT through deep learning, thus preserving the advantages that comes with MERIT while making it faster.

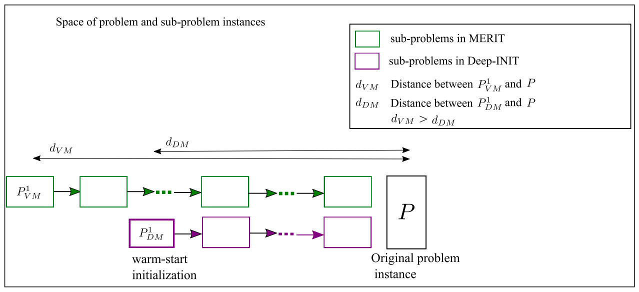
Space of problem and sub-problem instances
sub-problems in MERIT
sub-problems in Deep-INIT
d_{VM} Distance between P^1_{VM} and P
d_{DM} Distance between P^1_{DM} and P
$d_{VM} > d_{DM}$
d_{VM}
d_{DM}
P^1_{VM}
P^1_{DM}
P
warm-start initialization
Original problem instance

Fig. 1: MERIT initialized by Deep-INIT is well-positioned to successfully approach a local optimum of the problem in fewer number of iterations than what MERIT typically requires when it employs a random initialization. The "closeness" between a sub-problem and the original problem instance is measured using a matrix distance measure *d*; see e.g., (6).

2. DEEP-INIT FOR UQP

The central idea behind smart initialization for our problem is that the number of iterations required by MERIT depends on the proximity of the initial sub-problem to the original problem instance. Recall that, in each iteration, a new sub-problem is created by MERIT that is closer to the original problem instance than the sub-problem constructed in the previous iteration. Therefore, one could potentially speed up MERIT by starting with a sub-problem that is closer to the original problem instance as illustrated in Fig. 1.

Herein, we will achieve this goal by re-parameterizing the initial values of the MERIT variables $\mathbf{s}$ and $\mathbf{Q}_{\mathbf{1}}$ using warm-start auxiliary parameters $[\mathbf{W}_1, \mathbf{W}_2]$. Let $\mathbf{Q}_{\mathbf{1}}^0$ and $\mathbf{s}^0$ denote the initial values of $\mathbf{Q}_{\mathbf{1}}$ and $\mathbf{s}$, respectively, that will be used by MERIT. Suppose $\mathbf{Q}_{\mathbf{1}}^{\text{rand}}$ is a random initial value of $\mathbf{Q}_{\mathbf{1}}$ in $\mathcal{C}(\mathbf{V}_{\mathbf{1}})$ and $\mathbf{s}_1^{\text{rand}}$ is a random unimodular vector. We let $\mathbf{Q}_{\mathbf{1}}^0 = \mathbf{W}_1 \mathbf{Q}_{\mathbf{1}}^{rand}, \mathbf{s}^0 = e^{j\arg(\mathbf{W}_2 \mathbf{s}_1^{rand})}$. Given this chose of parameters, MERIT will use a value for $\mathbf{P}_{\mathbf{1}}$ that is given as $\mathbf{P}_{\mathbf{1}}^0 = |\boldsymbol{R}| \odot \boldsymbol{R}_{c+}$, where $\boldsymbol{R}_c = \cos(\arg(\boldsymbol{R}) - \arg(\mathbf{s}^0(\mathbf{s}^0)^H))$. For any new UQP instance, the learned $\boldsymbol{\Theta} \triangleq [\mathbf{W}_1, \mathbf{W}_2]$ can be used to "warm-start" the optimization problem (3).

We hypothesize that after training, the sub-problem constructed using the "warm-started" parameters is more likely to be closer to the original problem instance than a random initialization. The reason for this is that *our learned weights reshape the distribution of* $\mathbf{Q}_{\mathbf{1}}^0$ *and* $\mathbf{s}_1^0$*, in a way that make them more likely to be close to the true* $\mathbf{Q}_{\mathbf{1}}$ *and* $\mathbf{s}$. Our numerical experiments show that this is indeed the case, even without making any assumptions on the problem instance or the training data, such as similarity between training and test distributions.

To the best of our knowledge, our work is one of the first few that proposes a data-driven initialization scheme to accelerate a model-based algorithm. Similar initialization methods have been studied for other non-convex problems in the literature [21–24]. Although the method introduced in this work is tailor-made for UQPs, the proposed technique can serve as a general recipe for many optimization problem in which proper

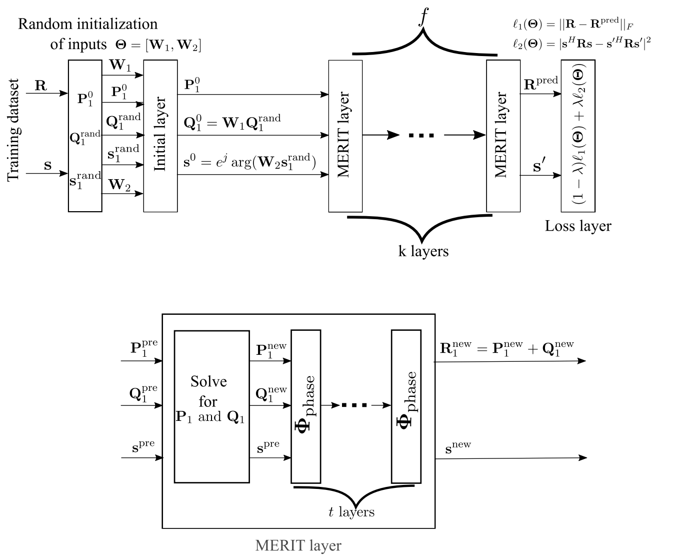
Random initialization
of inputs $\Theta = [\mathbf{W}_1, \mathbf{W}_2]$
Training dataset
$\mathbf{R}$
$\mathbf{s}$
$\mathbf{P}_1^0$
$\mathbf{Q}_1^{\text{rand}}$
$\mathbf{s}_1^{\text{rand}}$
$\mathbf{W}_1$
$\mathbf{W}_2$
Initial layer
$\mathbf{Q}_1^0 = \mathbf{W}_1\mathbf{Q}_1^{\text{rand}}$
$\mathbf{s}^0 = e^j \arg(\mathbf{W}_2\mathbf{s}_1^{\text{rand}})$
MERIT layer
MERIT layer
f
k layers
$\ell_1(\Theta) = ||\mathbf{R} - \mathbf{R}^{\text{pred}}||_F$
$\ell_2(\Theta) = |\mathbf{s}^H\mathbf{R}\mathbf{s} - \mathbf{s}'^H\mathbf{R}\mathbf{s}'|^2$
$\mathbf{R}^{\text{pred}}$
$\mathbf{s}'$
$(1-\lambda)\ell_1(\Theta) + \lambda\ell_2(\Theta)$
Loss layer
$\mathbf{P}_1^{\text{pre}}$
$\mathbf{Q}_1^{\text{pre}}$
$\mathbf{s}^{\text{pre}}$
Solve for $\mathbf{P}_1$ and $\mathbf{Q}_1$
$\mathbf{P}_1^{\text{new}}$
$\mathbf{Q}_1^{\text{new}}$
$\mathbf{s}^{\text{pre}}$
Φ_{phase}
Φ_{phase}
t layers
$\mathbf{R}_1^{\text{new}} = \mathbf{P}_1^{\text{new}} + \mathbf{Q}_1^{\text{new}}$
$\mathbf{s}^{\text{new}}$
MERIT layer

Fig. 2: An illustration of the training architecture of the proposed Deep-INIT model. The **top** figure shows the high-level architecture of Deep-INIT training. The image at the **bottom** gives more details about the MERIT layer. Here, Φ_{phase} denotes one iteration of the internal MERIT optimizaion process over $\mathbf{s}$.

initialization could potentially speed-up convergence. We now outline the new deep architecture and training details.

2.1. The Deep-INIT Architecture

The training architecture of Deep-INIT is presented in Fig. 2. The input parameters to the Deep-INIT training pipeline are a collection of training data points $\{(\boldsymbol{R}_i, \boldsymbol{s}_i)\}$. Then, for each $(\boldsymbol{R}_i, \boldsymbol{s}_i)$, the MERIT parameters $\boldsymbol{P}_1^0, \boldsymbol{s}_1^{\text{rand}}, \boldsymbol{Q}_1^{\text{rand}}$ are randomly generated within their respective constraint sets. The *initial layer* generates improved MERIT parameters for training by multiplying (random initializations of) $\boldsymbol{Q}$ and $\boldsymbol{s}$ with weights $\boldsymbol{\Theta} = [\boldsymbol{W}_1, \boldsymbol{W}_2]$. Next, we have k MERIT layers for a given $\boldsymbol{R}_i$, which is used to compute the loss function (discussed below) in the *loss layer*.

Loss Function for Training: The training for "warm-starting" MERIT can be formulated by the following objective:

$$\begin{aligned} \min_{\boldsymbol{\Theta}} \min_{\mathbf{P_1},\mathbf{Q_1},\mathbf{s}} \quad & \ell(f_k(\mathbf{R}), \mathbf{R}, \boldsymbol{s}) \qquad (4) \\ \text{s.t.} \quad & \mathbf{Q_1^0} = \mathbf{W}_1\mathbf{Q_1^{\text{rand}}}; \quad \mathbf{s}^0 = \exp(j \arg(\mathbf{W}_2\mathbf{s}_1^{\text{rand}})) \end{aligned}$$

where $f_k(\mathbf{R})$ is the MERIT approximation of the matrix $\mathbf{R}$ at the k^{th} iteration (thus truncating the number of iterations to exactly k) and $\ell(.)$ is a specific loss function introduced below.

Note that one can easily evaluate $f_k(\mathbf{R})$ by looking at the MERIT objective in (3). To form our loss function, motivated by the MERIT objective, we define the functions $\ell_1(.)$ and $\ell_2(.)$ as follows:

$$\ell_1(\boldsymbol{\Theta}) = ||\mathbf{R} - f_k(\mathbf{R})||_F; \quad \ell_2(\boldsymbol{\Theta}) = |\Psi(\mathbf{s}) - \Psi(\mathbf{s}^0)|^2, \quad (5)$$

where $\Psi(\mathbf{x}) = \mathbf{x}^H\mathbf{R}\mathbf{x}$ represents the original UQP objective. In the above, while $\ell_1(.)$ promotes learning initializations that are closer to the original UQP problem instance, the function $\ell_2(.)$ further ensures that the learned solutions ultimately lead to a *good* UQP solution as well. In our experiments, we define the loss function $\ell(.)$ to be a convex combination of two losses $\ell_1(.)$ and $\ell_2(.)$, where $\lambda \in (0, 1)$ is the mixing parameter: $\ell(\boldsymbol{\Theta}) = (1-\lambda)\ell_1(\boldsymbol{\Theta}) + \lambda\ell_2(\boldsymbol{\Theta})$.

Algorithm 1 for Training Data Generation

```
1: Input: $\boldsymbol{\Sigma} \in \mathbb{R}^{n\times n}$ diagonal matrix with positive entries in decreasing order
2: Randomly generate $\mathbf{s}_1 \in \Omega^n$
3: Generate an orthogonal basis $\mathbf{U}$ with $\mathbf{s}_1$ as its first column
4: Construct $\boldsymbol{R}_1 \leftarrow \mathbf{U}\boldsymbol{\Sigma}\mathbf{U}^H$
5: Return $(\boldsymbol{R}_1, \mathbf{s}_1)$
```

Algorithm 2 for Training Data Generation

```
1: Input: $(\mathbf{R}_1, \mathbf{s}_1)$ pair obtained from Algorithm 1
2: Randomly generate $\mathbf{s}_2 \in \Omega^n$
3: Compute $\mathbf{s}_0 = \mathbf{s}_1^* \odot \mathbf{s}_2$;   $\mathbf{R}_2 = \mathbf{R}_1 \odot \mathbf{s}_0\mathbf{s}_0^H$
4: Return $(\mathbf{R}_2, \mathbf{s}_2)$
```

Training data: Due to the unavailability of any benchmark datasets, in this section, we use a technique to synthetically generate training data for Deep-INIT. The proposed approach relies on a characterization of the UQP problem instances whose global optima are readily known or could be computed quite efficiently. There are already families of such problems introduced in [1]; in particular, we base our first method of data generation on Theorem 2 (see Algorithm 1), and then use Theorem 1 to further enrich the set of problems that is used for training (see Algorithm 2).

2.2. Inference

At the time of execution and testing, we are given a new matrix $\mathbf{R}_{\text{test}}$ and we are required to predict the UQP solution corresponding to $\mathbf{R}_{\text{test}}$. In order to recover the optimal $\boldsymbol{s}$, we run the full MERIT algorithm where $\mathbf{Q_1}$ and $\boldsymbol{s}$ are initialized using the learned weights $\boldsymbol{\Theta}$. In our experiments, we observe that the learned weights glean sufficient clues on the sub-problem structure of MERIT, helping us to "jump ahead" in the iterative sequence of MERIT.

A key feature of the proposed Deep-INIT algorithm is that it is able to preserve the optimality guarantees provided by the MERIT algorithm [1] during inference—as observed below.

3. NUMERICAL RESULTS

In this section, we will empirically validate our hypothesis that the weights learned through the Deep-INIT model is able to significantly reduce the number of iterations, and thus, speed-up the optimization process.

3.1. Experimental Setup

We conduct two numerical experiments to verify the effectiveness of our proposed Deep-INIT model and training procedure for approximating UQPs. *Experiment 1* is used to validate the speed-up benefits of Deep-INIT. *Experiment 2* tests our hypothesis that the warm-start initializations that Deep-INIT provides, creates sub-problems that are closer to the original problem instance than the initial random initializations.

We evaluate the MERIT truncated function $f_k(.)$ by running the optimization over $\mathbf{P_1}$ and $\mathbf{Q_1}$ with k times. At each iteration, the function runs the local optimization over $\mathbf{s}$ for t sub-iterations. In our experiments, we set $k = 3$ and $t = 10$ during training, to achieve the desired results.

Metrics. We define the following two metrics:

1. *Iteration reduction ratio* (η): Let n_v and n_d denote the number of iterations required by MERIT in the case of a random initialization and the case of initialization with Deep-INIT, to converge to a local optimum for a given UQP problem instance. Then η is defined as: $\eta = \frac{n_v - n_d}{n_v}$. For instance, if Deep-INIT does not reduce the number of iterations, η will assume a zero value.

2. *Wall-clock time (t)*: Measures the time taken (in seconds) by MERIT in each case.

Experiment 1. We first train our Deep-INIT model with synthetic data generated using Algorithm 1 and Algorithm 2. In particular, we generated random positive definite input matrices. We generated 500 data points for training and 50 data points for inference with dimensionality $d \in \{8, 16, 24, 32\}$, creating a total of four variations. For each variation, we ran the training and testing 10 different times and recorded the mean and standard deviation of our metrics.

Experiment 2. For this experiment, we fix $d = 8$ and generate 10 different training (with 500 data points) and test (with 50 data points) sets. For each set, we first train the Deep-INIT algorithm and obtain the warm-start initializations $(Q_1, \boldsymbol{s})$. Using these values, we run one iteration of MERIT. Additionally, we run one iteration of MERIT with the random initialization. In both cases, we measure the approximation strength, measured as the difference between the constructed and true $\mathbf{R}$. Assuming $\mathbf{R}'$ denotes the contructed version of $\mathbf{R}$, we measure the problem-instance construction strength (δ) via the criterion:

$$\delta(\mathbf{R}, \mathbf{R}') = ||\mathbf{R} - \mathbf{R}'||_F. \tag{6}$$

3.2. Discussion

From Table 1, we can observe that, across all variations, Deep-INIT achieves a significant speed-up over vanilla MERIT, without sacrificing the quality of optimization guarantees.

The scatter plot in Fig. 3 may be interpreted as follows. The $y-$axis represents the values of δ for MERIT initialized by Deep-INIT and $x-$axis denotes the values of δ for vanilla MERIT (i.e. initialized at random). If δ for Deep-INIT is smaller overall than MERIT, then we expect to see more points below the diagonal reference line on the scatter plot. This is indeed what we observe from the plots.

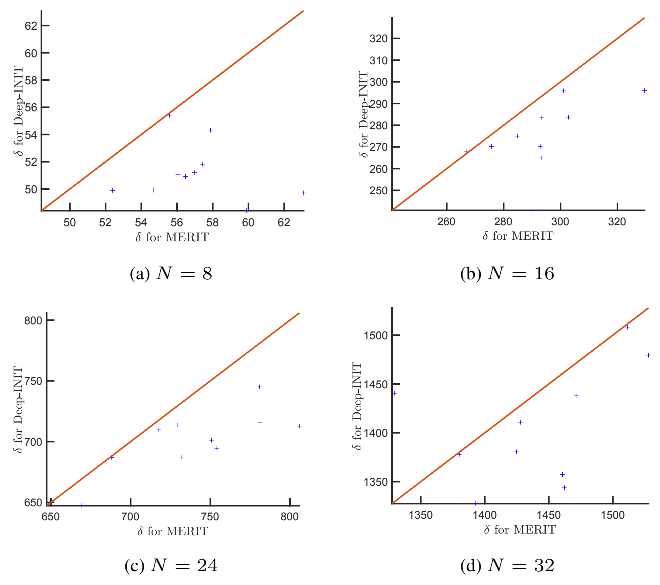

Fig. 3: Scatter plot of problem-instance construction gap δ of vanilla MERIT vs MERIT initialized by Deep-INIT, measured for matrix sizes $N \in \{8, 16, 24, 32\}$, after running one iteration of MERIT. The results are plotted for 10 different tests in each case.

N	η (%)	$\gamma_{\text{Deep-INIT}}$	γ_{MERIT}	$t_{\text{Deep-INIT}}$	t_{MERIT}
8	15±6	0.989	0.987	0.33±0.04	0.43
16	19±5	0.954	0.949	17.7±2.64	22.62
24	14±6	0.93	0.924	51.54±4.4	54.8
32	18±7	0.9	0.9	149.18±6	195.47

Table 1: Performance of Deep-INIT's warm-start over 10 trials, measured by percentage reduction in the number of required MERIT iterations and wall-clock time. Here, N denotes the matrix size, or equivalently, the dimension of each sample point.

4. CONCLUSION

In this work, we developed a smart initialization scheme for the parameters of a guaranteed UQP solver. Our proposed algorithm, Deep-INIT, learns the proper initialization for the parameters of a model-based solver for UQP, called MERIT. With the learned initialization, Deep-INIT helps MERIT to provably converge to a locally optimal solution for UQP, in a fewer number of iterations and less time. We also proposed a data generation algorithm for obtaining training data to train our Deep-INIT architecture.

5. REFERENCES

[1] M. Soltanalian and P. Stoica, "Designing unimodular codes via quadratic optimization," *IEEE Transactions on Signal Processing*, vol. 62, no. 5, pp. 1221–1234, 2014.

[2] P. Stoica, J. Li, and Y. Xie, "On probing signal design for MIMO radar," *IEEE Transactions on Signal Processing*, vol. 55, no. 8, pp. 4151–4161, 2007.

[3] S. Ragi, E. K. Chong, and H. D. Mittelmann, "Polynomial-time methods to solve unimodular quadratic programs with performance guarantees," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 55, no. 5, pp. 2118–2127, 2018.

[4] M. Soltanalian, B. Tang, J. Li, and P. Stoica, "Joint design of the receive filter and transmit sequence for active sensing," *IEEE Signal Processing Letters*, vol. 20, no. 5, pp. 423–426, 2013.

[5] S. Ragi, E. K. Chong, and H. D. Mittelmann, "Heuristic methods for designing unimodular code sequences with performance guarantees," in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2017, pp. 3221–3225.

[6] J. Jaldén, C. Martin, and B. Ottersten, "Semidefinite programming for detection in linear systems-optimality conditions and space-time decoding," in *2003 IEEE International Conference on Acoustics, Speech, and Signal Processing, 2003. Proceedings.(ICASSP'03).*, vol. 4. IEEE, 2003, pp. IV–9.

[7] S. Zhang and Y. Huang, "Complex quadratic optimization and semidefinite programming," *SIAM Journal on Optimization*, vol. 16, no. 3, pp. 871–890, 2006.

[8] A. T. Kyrillidis and G. N. Karystinos, "Rank-deficient quadratic-form maximization over m-phase alphabet: Polynomial-complexity solvability and algorithmic developments," in *2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2011, pp. 3856–3859.

[9] S. Verdú, "Computational complexity of optimum multiuser detection," *Algorithmica*, vol. 4, no. 1-4, pp. 303–312, 1989.

[10] W.-K. Ma, B.-N. Vo, T. N. Davidson, and P.-C. Ching, "Blind ml detection of orthogonal space-time block codes: Efficient high-performance implementations," *IEEE Transactions on Signal Processing*, vol. 54, no. 2, pp. 738–751, 2006.

[11] T. Cui and C. Tellambura, "Joint channel estimation and data detection for ofdm systems via sphere decoding," in *IEEE Global Telecommunications Conference, 2004. GLOBECOM'04.*, vol. 6. IEEE, 2004, pp. 3656–3660.

[12] S. Boyd, S. P. Boyd, and L. Vandenberghe, *Convex optimization*. Cambridge university press, 2004.

[13] M. X. Goemans and D. P. Williamson, "Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming," *Journal of the ACM (JACM)*, vol. 42, no. 6, pp. 1115–1145, 1995.

[14] A. De Maio, Y. Huang, M. Piezzo, S. Zhang, and A. Farina, "Design of optimized radar codes with a peak to average power ratio constraint," *IEEE Transactions on Signal Processing*, vol. 59, no. 6, pp. 2683–2697, 2011.

[15] Z.-Q. Luo, W.-K. Ma, A. M.-C. So, Y. Ye, and S. Zhang, "Semidefinite relaxation of quadratic optimization problems," *IEEE Signal Processing Magazine*, vol. 27, no. 3, pp. 20–34, 2010.

[16] A. M.-C. So, J. Zhang, and Y. Ye, "On approximating complex quadratic optimization problems via semidefinite programming relaxations," *Mathematical Programming*, vol. 110, no. 1, pp. 93–110, 2007.

[17] C. G. Tsinos and B. Ottersten, "An efficient algorithm for unit-modulus quadratic programs with application in beamforming for wireless sensor networks," *IEEE Signal Processing Letters*, vol. 25, no. 2, pp. 169–173, 2017.

[18] F. Jiang, J. Chen, and A. L. Swindlehurst, "Estimation in phase-shift and forward wireless sensor networks," *IEEE transactions on signal processing*, vol. 61, no. 15, pp. 3840–3851, 2013.

[19] Ö. Aslan, X. Zhang, and D. Schuurmans, "Convex deep learning via normalized kernels," in *Advances in Neural Information Processing Systems*, 2014, pp. 3275–3283.

[20] V. Ganapathiraman, Z. Shi, X. Zhang, and Y. Yu, "Inductive two-layer modeling with parametric bregman transfer," in *International Conference on Machine Learning*, 2018, pp. 1636–1645.

[21] P. Netrapalli, P. Jain, and S. Sanghavi, "Phase retrieval using alternating minimization," *IEEE Transactions on Signal Processing*, vol. 63, no. 18, pp. 4814–4826, 2015.

[22] A. Agarwal, A. Anandkumar, P. Jain, P. Netrapalli, and R. Tandon, "Learning sparsely used overcomplete dictionaries," in *Conference on Learning Theory*, 2014, pp. 123–137.

[23] A. Agarwal, A. Anandkumar, P. Jain, and P. Netrapalli, "Learning sparsely used overcomplete dictionaries via alternating minimization," *SIAM Journal on Optimization*, vol. 26, no. 4, pp. 2775–2799, 2016.

[24] Y. Chen and E. Candes, "Solving random quadratic systems of equations is nearly as easy as solving linear systems," in *Advances in Neural Information Processing Systems*, 2015, pp. 739–747.