NEURAL INFORMATION PROCESSING SYSTEMS

LEARNING TO SOLVE QUADRATIC UNCONSTRAINED BINARY OPTIMIZATION IN A CLASSIFICATION WAY

Ming Chen

College of Systems Engineering, National University of Defense Technology

cmself@163.com

Problem Description

NEURAL INFORMATION PROCESSING SYSTEMS

Quadratic Unconstrained Binary Optimization (QUBO) a highly challenging subject in mathematical programming and operations research. The purpose of QUBO is to optimize an unconstrained quadratic function:

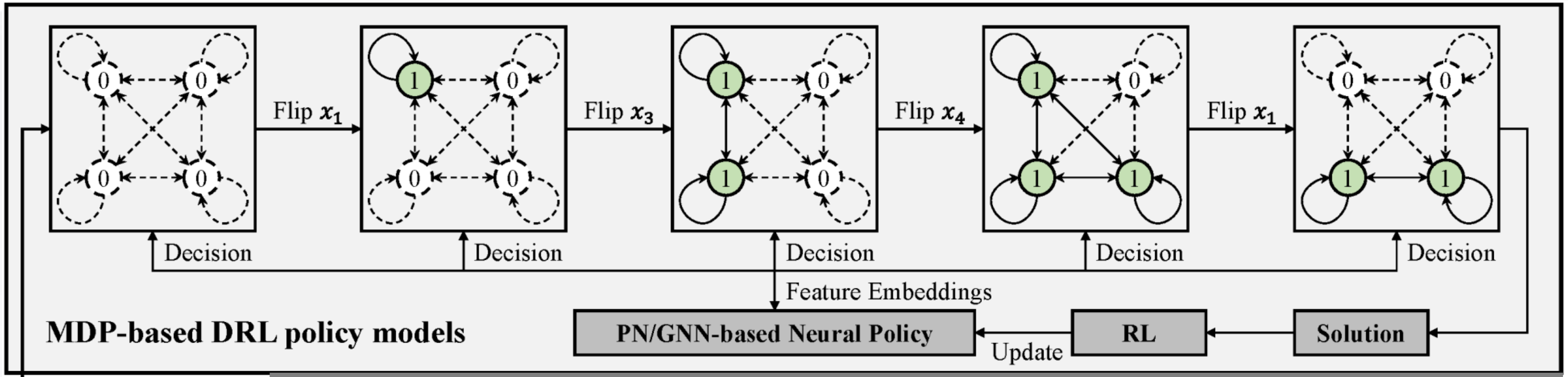
$$\max\backslash\min OFV = f(x) = x^{\top} Q x = \sum_{i=1}^{n} \sum_{j=1}^{n} q_{ij} x_i x_j$$

where Q is a **symmetric matrix** with $n \times n$ coefficients, while x is a binary (0-1) n-dimensional column vector, i.e., $x_i \in \{0,1\}, i = 1, \ldots, n$. OFV is short for objective function value.

This simple formulation is able to represent a remarkable spectrum of applications in combinatorial optimization.

Current Limitations

- **Current Learning-based methods: MDP-based Sequential decision process**

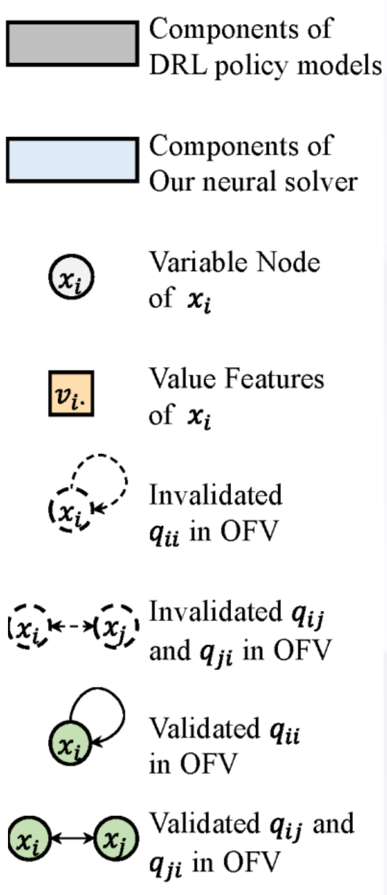


Learning-based methods in solving QUBO encounter limitations:

- Pointer-Network **(PN)-based** DRL models. Difficulties in processing $n \times n$-dimensional input matrix of the QUBO directly. (such as DRLH)
- **GNN-based** DRL models. Significant computational and storage costs due to the repeated updating of hidden states at each MDP step. (such as S2V-DQN and ECO-DQN)
- The repeated OFV evaluation throughout the MDP leads to significant **computational overhead**.

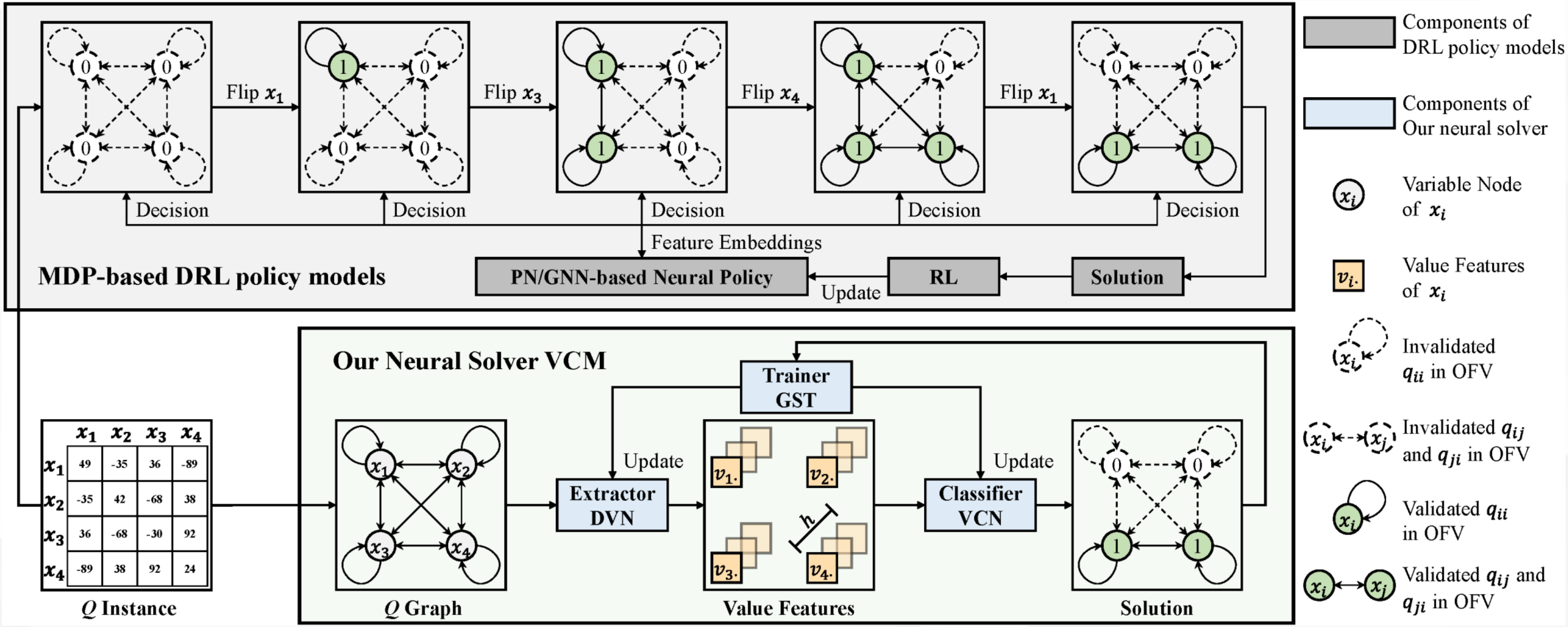
	x_1	x_2	x_3	x_4
x_1	49	-35	36	-89
x_2	-35	42	-68	38
x_3	36	-68	-30	92
x_4	-89	38	92	24

Q Instance



Contributions

- **Value Classification Model (VCM):** A **neural solver** that formulates the solution process from a **classification perspective.**



Contributions

NEURAL INFORMATION PROCESSING SYSTEMS

- **Value Classification Model** (**VCM**). A neural solver that formulates the solution process from a **classification perspective**, which includes the Extractor, Classifier, and Trainer. It can achieve **near-optimal solutions in milliseconds.**

- **Extractor (Depth Value Network, DVN).** Based on graph convolutional, DVN exploits the **symmetry property** in Q to auto-grasp value features.

- **Classifier (Value Classification Network, VCN).** VCN directly generates **classification solutions.**

- **Trainer (Greedy-guided Self Trainer, GST).** A highly efficient model-tailored trainer that **does not require priori optimal labels**.

- A VCM trained at a specific DVN depth can **steadily find better solutions by simply extending the testing depth**.

The proposed VCM

- **Main Idea**

In VCM, each x_j accompanied by a **value feature vector $\boldsymbol{v}_j$.**

Let $\boldsymbol{v}_j$ be the value features of the j-th column of Q (denoted as v_j^{col}), then each element in row j can then be expressed as $\boldsymbol{v}_j^{col} q_{ij}$. Thus, the value feature of i-th row can be computed as follows:

$$\boldsymbol{v}_j^{row} = \sum_{j=1}^{n} q_{ij} \boldsymbol{v}_j^{col}$$

Given that **$\boldsymbol{Q}$ is symmetric**, the value of each column should match the value of its corresponding row. This requirement translates to unifying $\boldsymbol{V} = Q\boldsymbol{V}$**.**

The proposed VCM

- **The Depth Value Network, DVN**

In DVN, we take the $Q\boldsymbol{V}$ and current $\boldsymbol{V}$ as inputs and propose a learning function $\mathcal{F}_E$ to iterate the value feature. The iteration at depth d is updated as follows:

$$\boldsymbol{V}^{(d+1)} = \mathcal{F}_E\left(\boldsymbol{V}^{(d)}, Q\boldsymbol{V}^{(d)}\right)$$

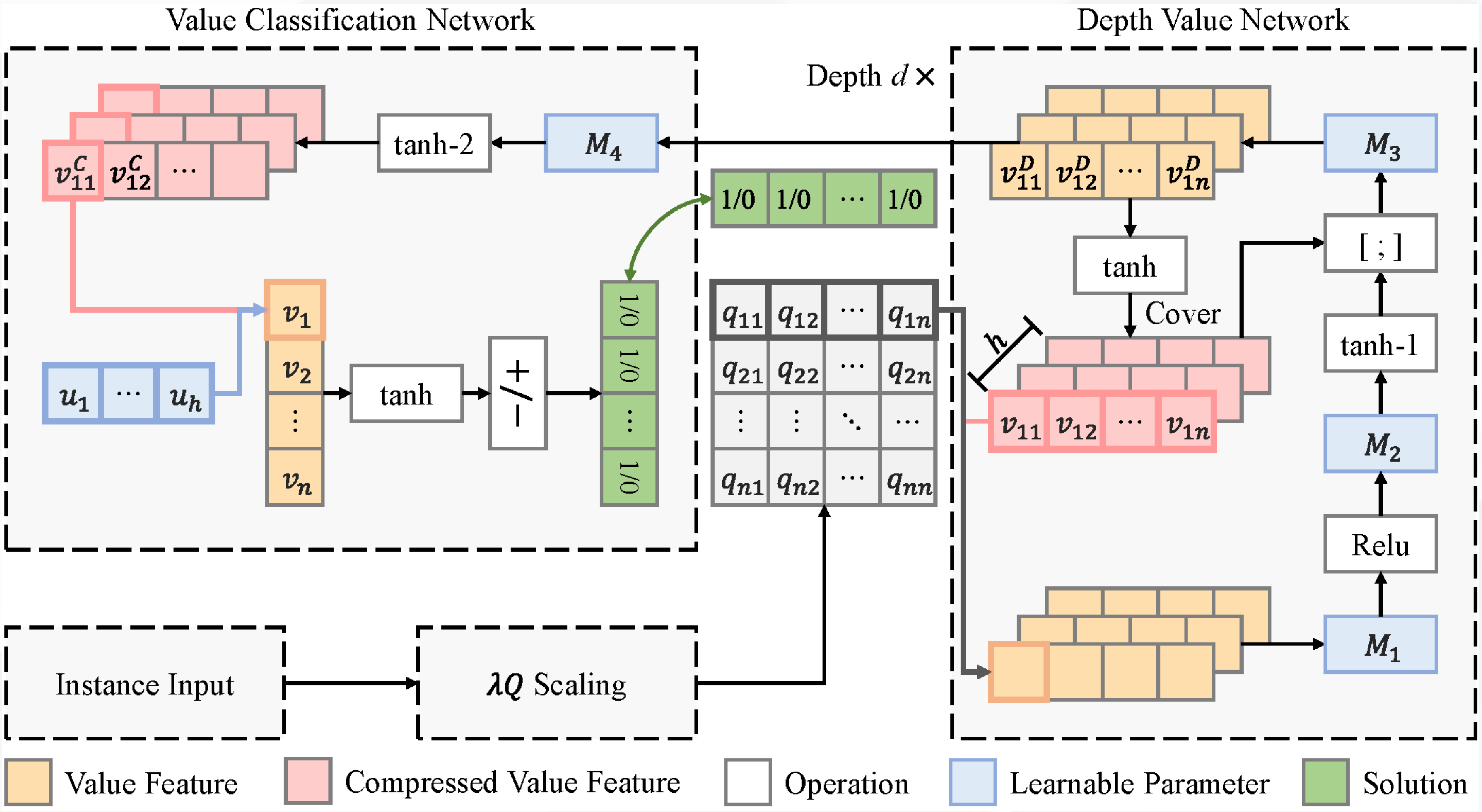
- **The Value Classification Network, VCN**

Based on the obtained value features $V^{(d),D}$ from DVN, we propose a learning function $\mathcal{F}_C$ which serves as the classifier to generate the solution x.

$$x = \mathcal{F}_C\left(\boldsymbol{V}^{(d),D}\right)$$

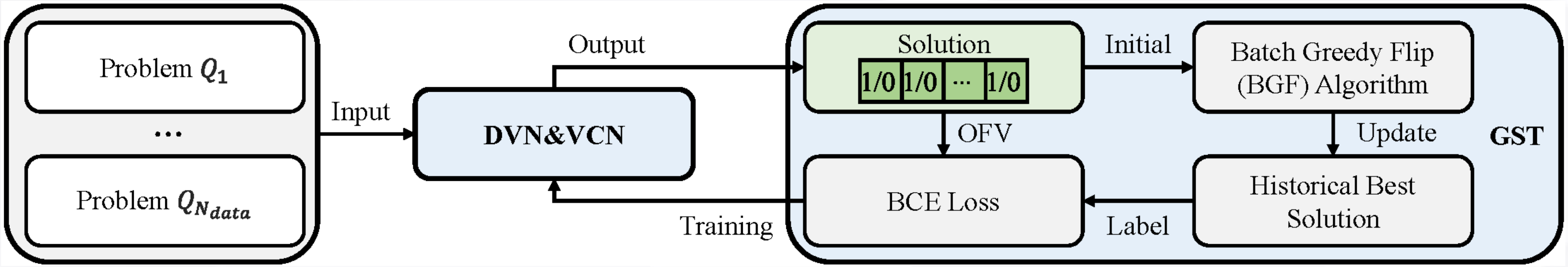
The proposed VCM

- **The Neural Architecture of DVN+VCN**



The proposed VCM

- **The Greedy-guided Self Trainer, GST**



Algorithm 1 The Greedy-guided Self Trainer

Input: The training dataset D with N_{data} instances, the epoch size E, the batch size B, the training steps per epoch $N_{steps} = ceil(N_{data}/B)$
Output: The trained VCM with parameters θ^*
Initialize the VCM parameters θ, Adam optimizer and the label set $X^L_{steps} = X^L_1 \cup ... \cup X^L_{N_{steps}}$
for $epoch = 1$ **to** E **do**
  for $step = 1$ **to** N_{steps} **do**
    $data_{step} \leftarrow SampleInput(D)$
    $X^L_{step} \leftarrow SampleInput(X^L_{steps})$
    $state_{step}, X^{VCM}_{step} \leftarrow VCM(data_{step})$
    $X^G_{step} \leftarrow BatchGreedyFlip(X^{VCM}_{step}, data_{step})$
    $X^L_{step} \leftarrow \arg\max_{X \in \{X^G_{step}, X^L_{step}\}} OFV(X)$
    $Loss \leftarrow BCELoss(X^L_{step}, (state_{step} + 1)/2)$
    $\theta \leftarrow Adam(\theta, Loss)$
  end for
end for

- VCM itself
- A Batch Greedy Flip (BGF) algorithm based on VCM (VCM-BGF)
- A historical best solution set (HB)

Validation

- **Details**
  - Dataset

    - Generated instances (**G**), benchmarks (**B**), and large well-known instances (**P**)

    - For the G set, the Q matrix is uniformly generated at random within [-100,100], following the benchmark data format.

    - The B set is B2500(10) consisting of ten ORLIB[1] instances of size 2500

    - The P set includes 21 very-large instances[2] including P3000(5), P4000(5), P5000(5), P6000(3), and P7000(3).

  - Train

    - 512,000 G instances 10, 20, 50, and 100 variables.
    - Hidden size 128 and DVN depth 40.
    - Batch size 512 and epoch size 100.

Validation

- **Main Competitor**
  - ◆ **The exact optimizer**:
  - **Gurobi**[1]. We set the max allowed time to **1s** and **1h.**
  - ◆ **Heuristic classification methods**:
  - **Diag** and **SR** from [2].
  - ◆ **Heuristic construction algorithm** :
  - **The proposed BGF, part of GST.**
  - ◆ **Learning-based model**:
  - PN-based DRL model:

    DRLH[2]**-B** (“**-B**” means it enhanced by our Batch OFV increment computation process)
  - GNN-based DRL model:

    S2V-DQN[3]**-B** and **ECO-DQN**[4]**-B**.
  - The physics-inspired neural solver:

    PI-GNN[5].

[1] LLC Gurobi Optimization. Gurobi optimizer reference manual, 2024.

[2] Heuristic algorithms based on deep reinforcement learning for quadratic unconstrained binary optimization. *Knowledge-Based Systems*. 2020

[3] Learning combinatorial optimization algorithms over graphs. *Advances in neural information processing systems*. 2017

[4] Exploratory combinatorial optimization with reinforcement learning. *In Proceedings of the AAAI conference on artificial intelligence*. 2020

[5] Combinatorial optimization with physicsinspired graph neural networks. *Nature Machine Intelligence*. 2022

Validation

- **The Performance of VCM**

Algorithm	B2500(10)		P3000(5)		P4000(5)		P5000(5)		P6000(3)		P7000(3)	
	Objective Function Value Baseline (Optimal)											
	1479921.4		5134727.2		7869134.2		10973791.4		13950582.0		17725010.33	
	GAP (%)	ART (ms)	GAP (%)	ART (ms)	GAP (%)	ART (ms)	GAP (%)	ART (ms)	GAP (%)	ART (ms)	GAP (%)	ART (ms)
Diag	81.842	0.7	99.262	1.7	97.374	2.9	98.256	4.4	99.273	11.1	98.843	14.3
SR	26.867	0.1	32.861	0.0	33.510	0.1	33.428	0.2	33.679	0.0	32.758	0.4
VCM10	0.368	7.8	**0.566**	7.9	**0.440**	8.1	**0.508**	8.5	**0.645**	8.8	**0.569**	9.1
VCM20	0.488	7.7	0.598	7.9	0.673	8.1	0.657	8.3	0.712	8.6	0.646	9.1
VCM50	**0.362**	7.7	0.861	7.9	0.669	8.1	0.783	8.1	0.806	8.7	0.702	9.1
VCM100	0.401	7.8	0.763	7.9	0.668	8.0	0.803	8.0	0.914	8.7	0.772	9.1
BGF	0.807	800.8	0.979	984.4	0.834	999.4	0.914	994.7	0.966	999.9	0.730	983.6
DRLH-B	1.640	1.5E+03	2.044	2.4E+03	1.884	5.0E+03	1.853	8.9E+03	2.030	1.5E+04	1.825	2.2E+04
S2V-DQN-B	21.657	1.6E+04	13.172	2.7E+04	13.255	6.3E+04	14.050	1.2E+05	14.403	2.1E+05	-	-
ECO-DQN-B	0.937	2.8E+04	1.371	5.4E+04	1.333	1.2E+05	1.270	2.3E+05	1.467	4.0E+05	-	-
VCM10-BGF	0.230	40.8	0.382	70.6	0.297	107.4	0.322	196.3	0.374	444.4	0.335	613.6
VCM20-BGF	0.227	52.3	0.260	87.8	0.342	135.8	0.310	282.6	0.316	506.8	0.312	634.7
VCM50-BGF	**0.136**	44.1	0.277	100.2	0.214	170.0	0.262	326.0	**0.267**	523.3	0.224	770.2
VCM100-BGF	0.139	49.2	**0.203**	106.0	**0.178**	186.7	**0.245**	298.2	0.271	549.7	**0.212**	770.4
PI-GNN(2-Layer)	1.689	4.4E+04	2.13	6.2E+04	1.636	7.1E+04	1.418	8.6E+04	1.986	1.1E+05	1.437	1.6E+05
PI-GNN(3-Layer)	1.909	5.7E+04	2.523	7.2E+04	2.092	8.0E+04	1.945	1.0E+05	2.18	1.3E+05	2.076	1.9E+05
PI-GNN(5-Layer)	3.28	1.2E+05	3.047	1.0E+05	2.761	1.1E+05	2.463	1.3E+05	2.584	1.8E+05	2.289	2.7E+05
Gurobi-1s	0.034	1.0E+03	0.070	1.0E+03	0.108	1.0E+03	0.091	1.0E+03	0.122	1.0E+03	0.145	1.0E+03
Gurobi-1h	**0.0023**	3.6E+06	**0.0028**	3.6E+06	**0.0109**	3.6E+06	**0.0096**	3.6E+06	**0.0144**	3.6E+06	**0.0169**	3.6E+06
VCM-BGF-HB	0.012	1.9E+04	0.020	3.6E+04	0.027	6.0E+04	0.040	1.1E+05	0.030	2.0E+05	0.057	2.8E+05

[1] The **Bold** indicates the best average result in different classes of methods.

The **state-of-the-art neural solver** of QUBO.

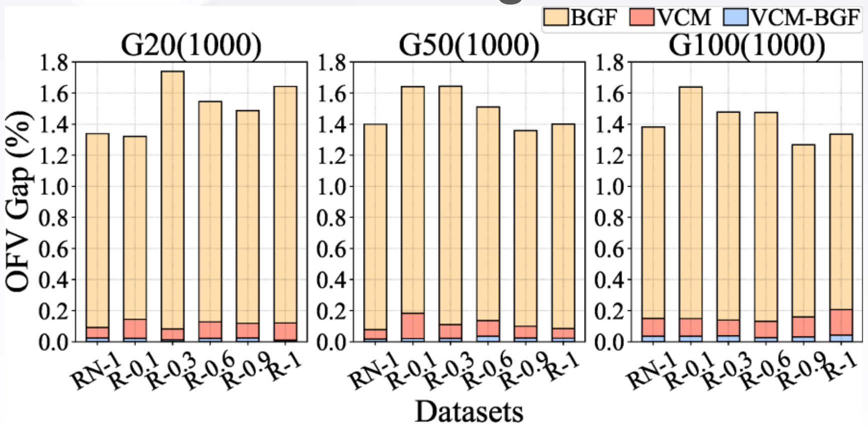
Validation

- **The Performance of VCM**

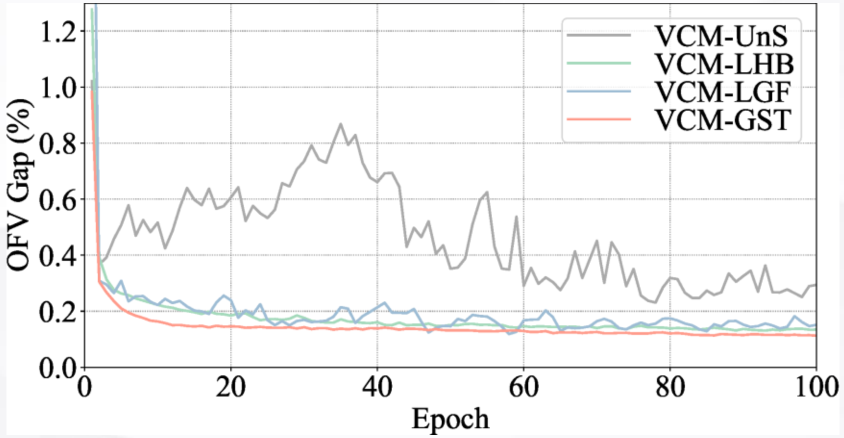
◆ The size generalization ability.

(a) The average gap in % of VCMs.

◆ The distribution generalization ability.



◆ Efficiency of GST



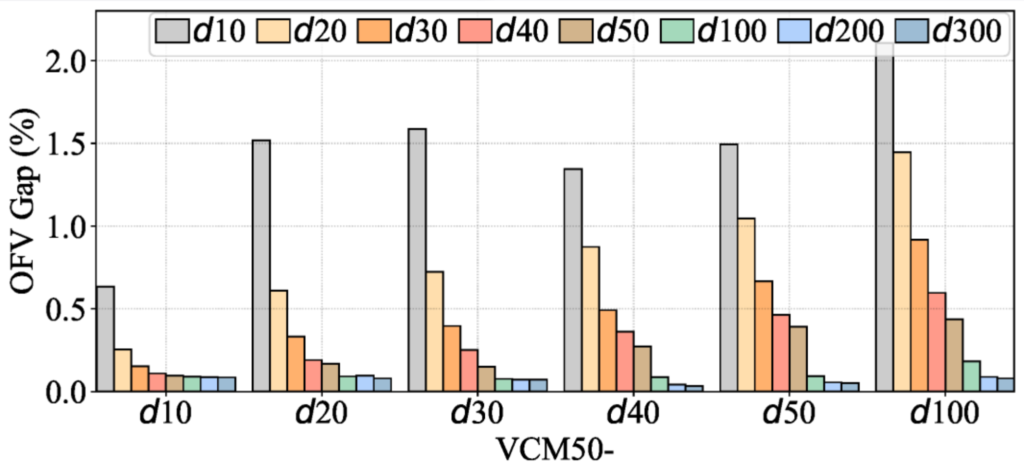
UnS: An **un-supervised** training method from [1].
LHB: Can be regarded as a **supervised** method.
LGF: GST without HB.

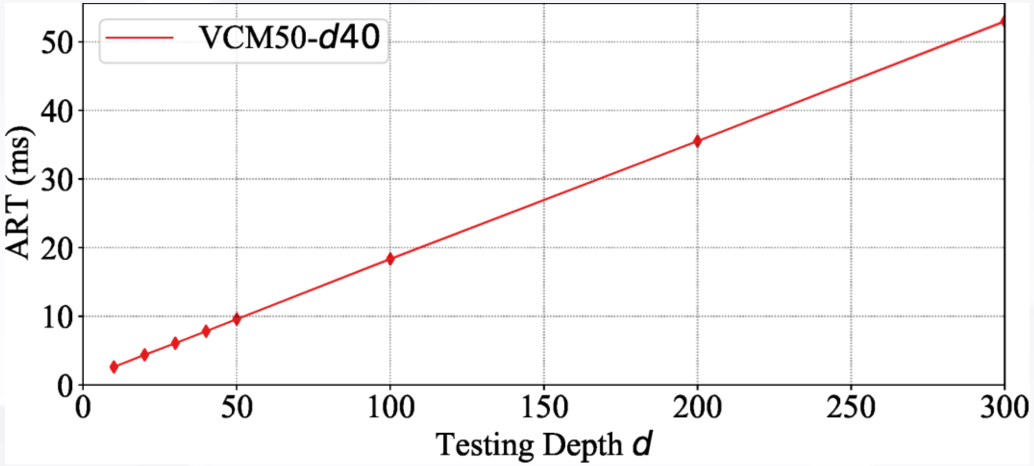
GST enables a **rapid, stable, and adaptive formation** of VCM

[1] Combinatorial optimization with physicsinspired graph neural networks. *Nature Machine Intelligence*. 2022

Validation

- **DVN Study**
  - **VCM50 (training under different depth) with different testing depth on B2500(10)**



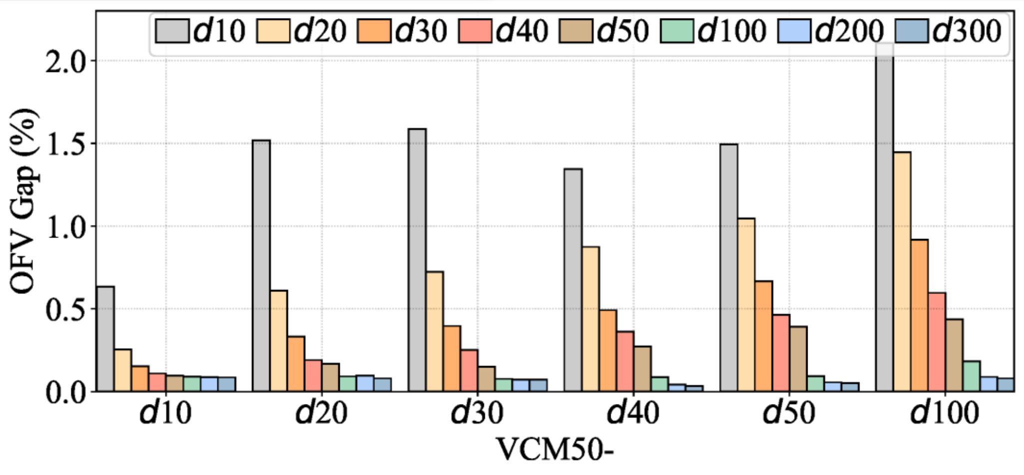


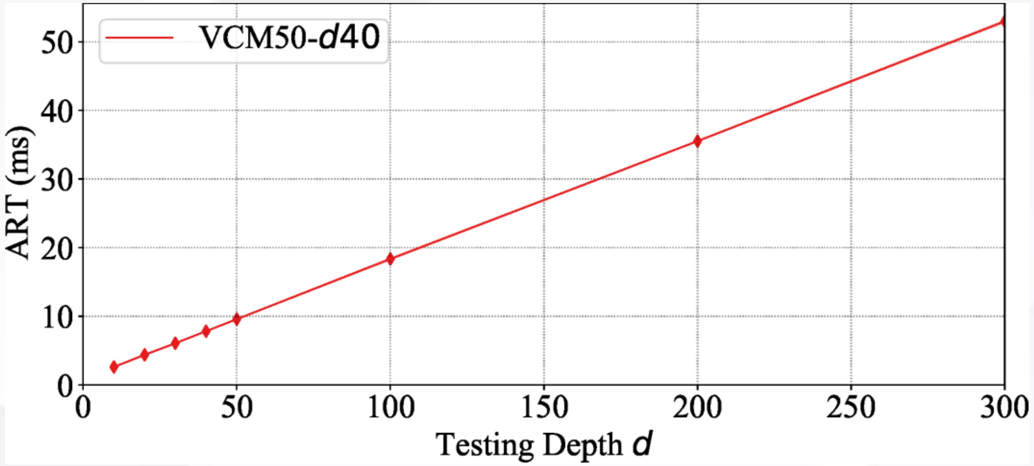
VCM can **steadily find better solutions** by **simply extending the testing depth** of DVN.

Computational time increases **linearly (on average 0.17ms per depth) as the testing depth enlarges.**

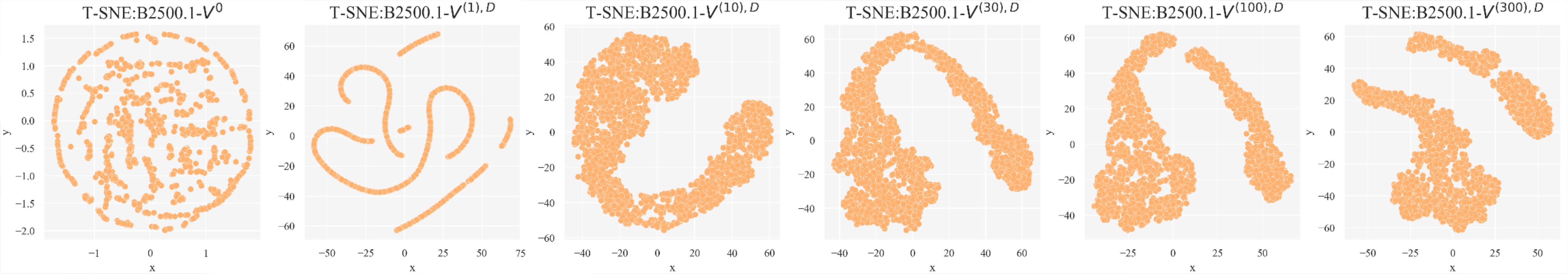
Validation

- **DVN Study**
  - ◆ VCM50 (training under different depth) with different testing depth on B2500(10)





  - ◆ t-SNE of the DVN output



THANKS

Ming Chen

College of Systems Engineering, National University of Defense Technology

Feel free to contact me:

WeChat:

Email: cmself@163.com