BIPNN: Learning to Solve Binary Integer Programming via Hypergraph Neural Networks

Sen Bai[1], Chunqi Yang[1], Xin Bai[2], Xin Zhang[1], Zhengang Jiang[1]

[1] Changchun university of science and technology

[2] Huawei Technologies Co. Ltd.

BIP (Binary Integer Programming) Problem

BIP is an optimization problem where the decision variables $\boldsymbol{x} = (x_1, x_2, \cdots, x_m)$ are restricted to binary values (0 or 1). Below is the general formulation.

$$
\begin{aligned}
\min \quad & O_{\mathrm{BIP}} = f(\boldsymbol{x}) \\
\text{s.t.} \quad & g_k(\boldsymbol{x}) \leq 0 \quad \text{for all} \quad k = 1,2,\cdots,K \\
& q_l(\boldsymbol{x}) = 0 \quad \text{for all} \quad l = 1,2,\cdots,L \\
& x_i \in \{0,1\} \quad \text{for all} \quad i = 1,2,\cdots,n
\end{aligned}
$$

-- Traditional solvers for BIPs:

Gurobi, **SCIP**

- rely on linear relaxation, which introduces a number of auxiliary variables
- suffer from prohibitive **computational cost**

-- To this end, we propose **BIPNN** (**B**inary **I**nteger **P**rogramming **N**eural **N**etwork).

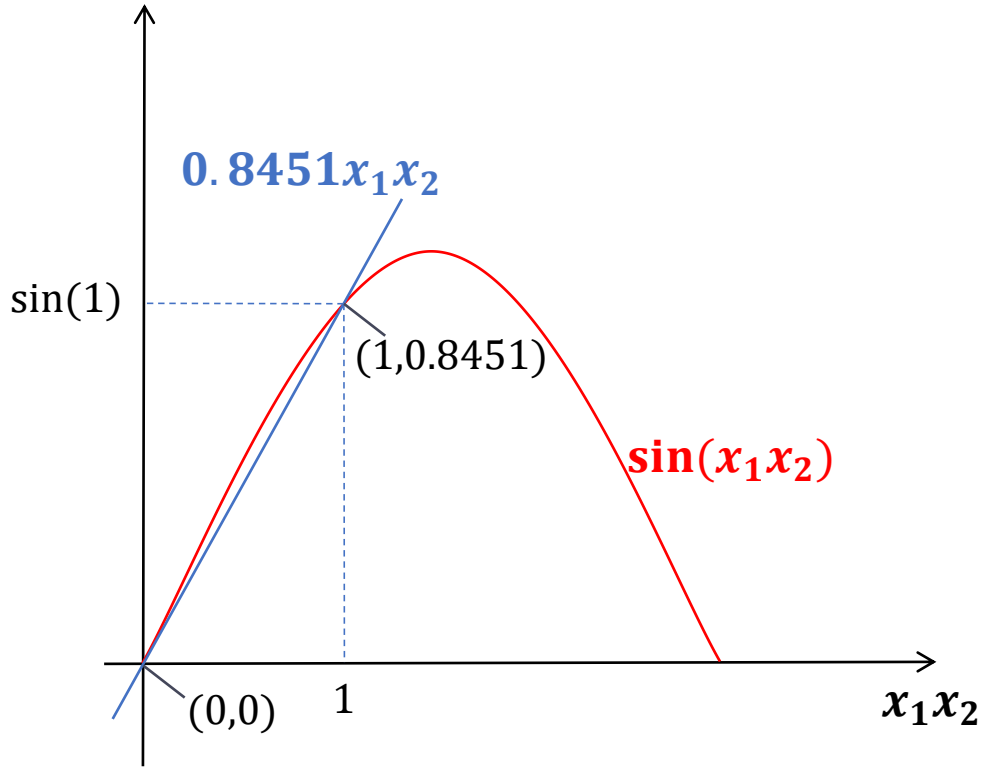
BIPNN: Polynomial Reformulation

BIP problem -- an example:

$$\begin{aligned} \min \quad & -x_1 - \mathbf{sin}(\boldsymbol{x_1 x_2}) + x_3 x_4 + 2x_4 \\ \text{s.t.} \quad & 2x_1 + \boldsymbol{e^{x_2}} + 3x_1 x_3 \leq 5 \\ & x_i \in \{0,1\} \quad \text{for} \quad i = 1,2,3,4 \end{aligned}$$

-- Polynomial Reformulation:

$$\begin{aligned} \min \quad & -x_1 - \mathbf{0.8451 x_1 x_2} + x_3 x_4 + 2x_4 \\ \text{s.t.} \quad & 2x_1 + \mathbf{1.7183 x_2} + 3x_1 x_3 \leq 4 \\ & x_i \in \{0,1\} \quad \text{for} \quad i = 1,2,3,4 \end{aligned}$$



Polynomial Reformulation:

For $x_1x_2 = 0$ or 1, $\sin(0) = 0$ or $\sin(1) = 0.8451$

-- Therefore, we can use $(0,0)$ and $(1,0.8451)$ to fit a linear function $0.8451x_1x_2$ to replace $\sin(x_1x_2)$.

BIPNN: Unconstrained Reformulation

-- Previously, only a few penalty terms were discovered.

Classical Constraint	Equivalent Penalty
$x + y \leq 1$	$P(xy)$
$x + y \geq 1$	$P(1 - x - y + xy)$
$x + y = 1$	$P(1 - x - y + 2xy)$
$x \leq y$	$P(x - xy)$
$x_1 + x_2 + x_3 \leq 1$	$P(x_1x_2 + x_1x_3 + x_2x_3)$
$x = y$	$P(x + y - 2xy)$

-- The penalty function should satisfy the following two requirements:

- If the current solution satisfies the constraints, the value of the penalty function should be zero.
- If the current solution does not satisfy the constraints, the penalty function should yield a large positive number.

(x_1, x_2)	$x_1 + 2x_2 - 2 \leq 0$
(0,0)	✓
(1,0)	✓
(0,1)	✓
(1,1)	✗

-- **Observation**: For more general constraints such as $x_1 + 2x_2 - 2 \leq 0$, the violating set $\{x_1 = 1, x_2 = 1\}$ corresponds to penalty term $\lambda(x_1x_2)$.

-- This motives us to propose "Minimal violation subset" based penalty methods.

BIPNN: Unconstrained Reformulation

-- Polynomial Reformulation:

$$\begin{aligned} \min \quad & -x_1 - 0.8451x_1x_2 + x_3x_4 + 2x_4 \\ \text{s. t.} \quad & \mathbf{2x_1 + 1.7183x_2 + 3x_1x_3 \le 4} \\ & x_i \in \{0,1\} \quad \text{for} \quad i = 1,2,3,4 \end{aligned}$$

-- Unconstrained Reformulation:

$$\begin{aligned} \min \quad & -x_1 - 0.8451x_1x_2 + x_3x_4 + 2x_4 + \lambda \mathbf{x_1x_3} \\ & x_i \in \{0,1\} \quad \text{for} \quad i = 1,2,3,4 \end{aligned}$$

Minimal violation subset:

(x_1, x_2, x_3)	$2x_1 + 1.7183x_2 + 3x_1x_3$	≤ 4
(0,0,0)	0	✓
(1,0,0)	0	✓
(0,1,0)	1.7183	✓
(0,0,1)	0	✓
(1,1,0)	3.7183	✓
(1, 0, 1)	5	✗
(0,1,1)	1.7183	✓
(1,1,1)	6.7183	✗

Unconstrained Reformulation:

-- As the table shows, we propose the **Minimal violation subset** to construct the penalty terms.

-- Our work makes significant improvements to the penalty-based methods.

BIPNN: Modeling BIPs via Hypergraphs

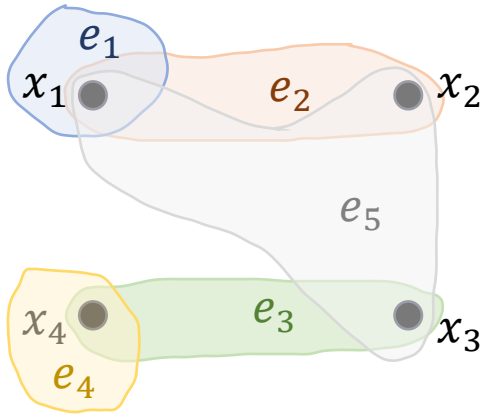
-- One-to-one mapping between polynomial BIPs and hypergraph structures.

Coefficients: $[c_1, c_2, c_3, c_4, c_5]$

$$\min \quad O_{\text{BIP}} = c_1x_1 + c_2x_1x_2 + c_3x_3x_4 + c_4x_4 + c_5x_1x_2x_3$$

Polynomial terms: $e_1 \sim x_1$ $e_2 \sim x_1x_2$ $e_3 \sim x_3x_4$ $e_4 \sim x_4$ $e_5 \sim x_1x_2x_3$

-- Hypergraph structures



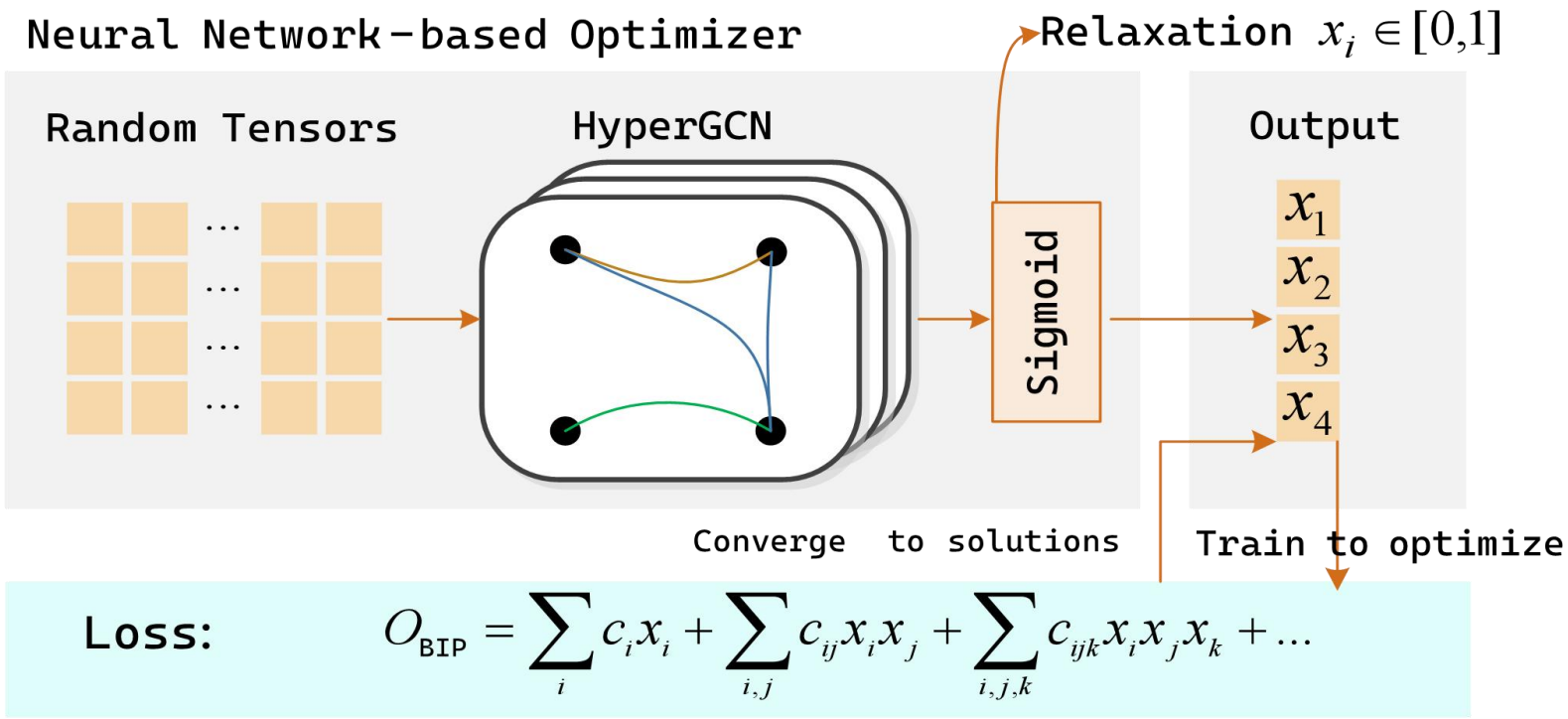
-- Incidence matrix

	e_1	e_2	e_3	e_4	e_5
x_1	1	1	0	0	1
x_2	0	1	0	0	1
x_3	0	0	1	0	1
x_4	0	0	1	1	0

This modeling enables us to solve BIPs via unsupervised hypergraph neural networks.

BIPNN: Neural Network-based Optimizer

-- Training workflow

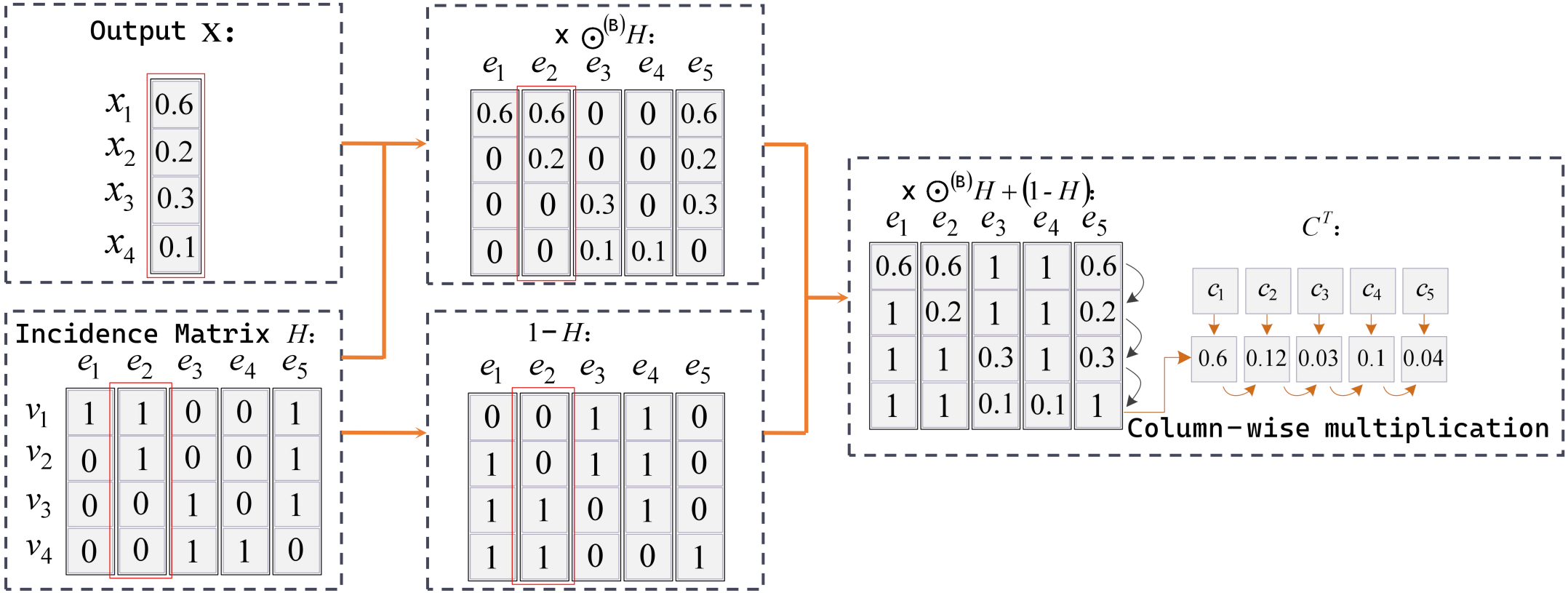


-- BIPNN leverages unsupervised training of HyperGCNs to optimize BIPs.

-- However, we encounter severe obstacles of low computational efficiency in polynomial losses O_{BIP} with numerous variables.

BIPNN: GPU-accelerated Training

-- We address the challenge via matrix operations on the incidence matrices of hypergraphs.

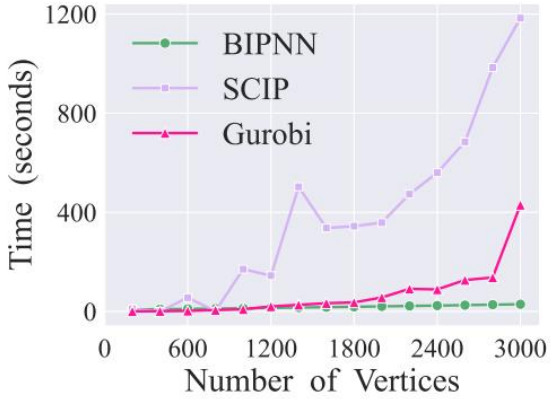


-- That is,

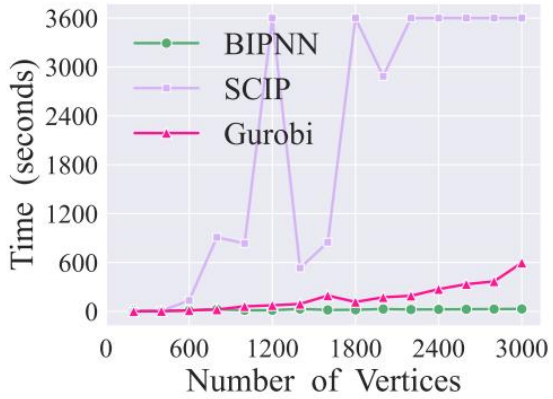
$$O_{\text{BIP}} = \text{ColM}\left(\text{x} \odot^{(\text{B})} H + (1-H)\right) C^T$$

Experimental Results

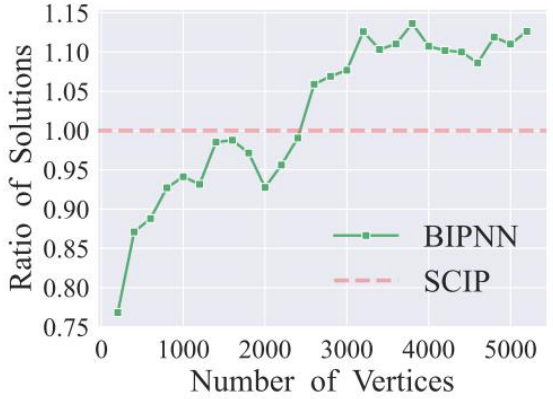
-- Compared with Gurobi and SCIP.



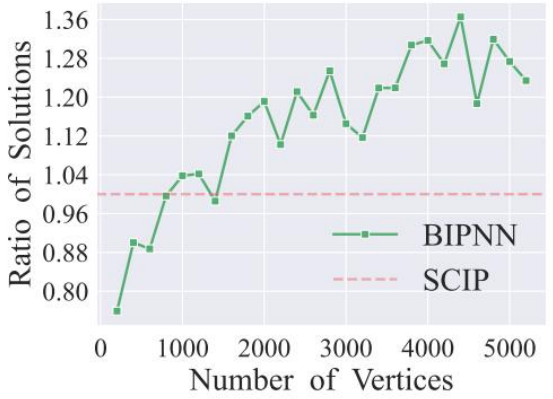
(a) $d = 4$.



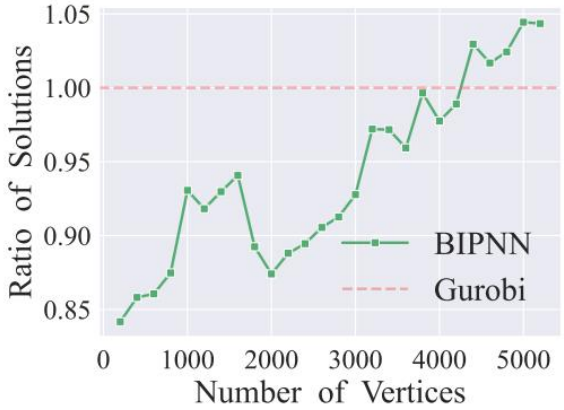
(b) $d = 6$.



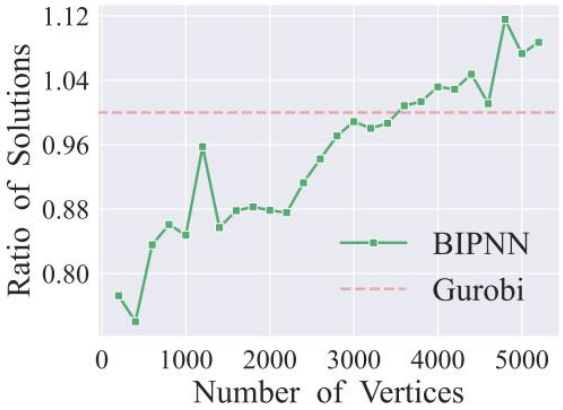
(c) $d = 4$.



(d) $d = 6$.

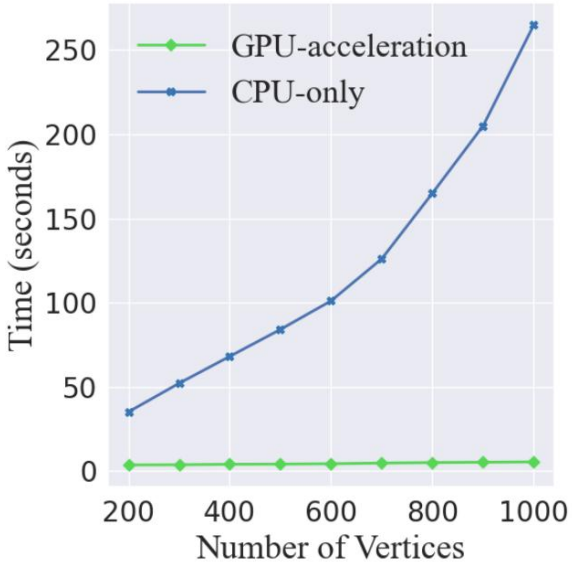


(e) $d = 4$.



(f) $d = 6$.

-- The training time for BIPNN with or without GPU acceleration.



(a)(b) show the solving time required for BIPNN, SCIP, and Gurobi to obtain the same solution. (c)(d) show the ratio of solutions.

Thank you!