PONTIFICIA
UNIVERSIDAD
CATÓLICA
DE CHILE

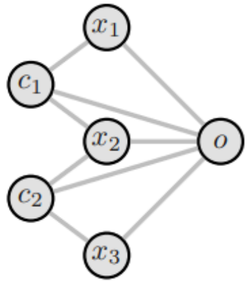
DCC
Departamento de
Ciencia de la computación

Exploring the Power of Graph Neural Networks in Solving Linear Optimization Problems

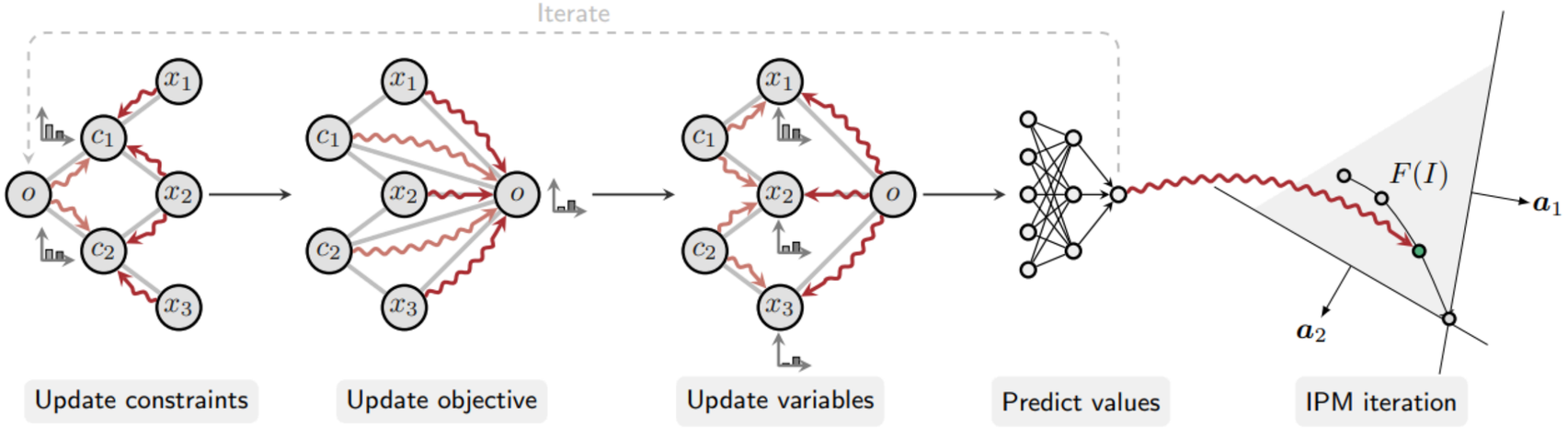
Presentado por:

Alexander Infanta

Introducción



(a) Representing an LP instance with two constraints and three variables as a tripartite graph.



(b) IPM-MPNNs emulate interior-point methods.

Figure 1: Overview of our IPM-MPNN framework.

Background

Linear optimization problem (LP)

$$\begin{aligned} \text{mín} \quad & c^T x \\ \text{s.a.} \quad & Ax \leq b \end{aligned}$$

Linear optimization problem (LP)

Una instancia I de un LP es una tupla (A, b, c)

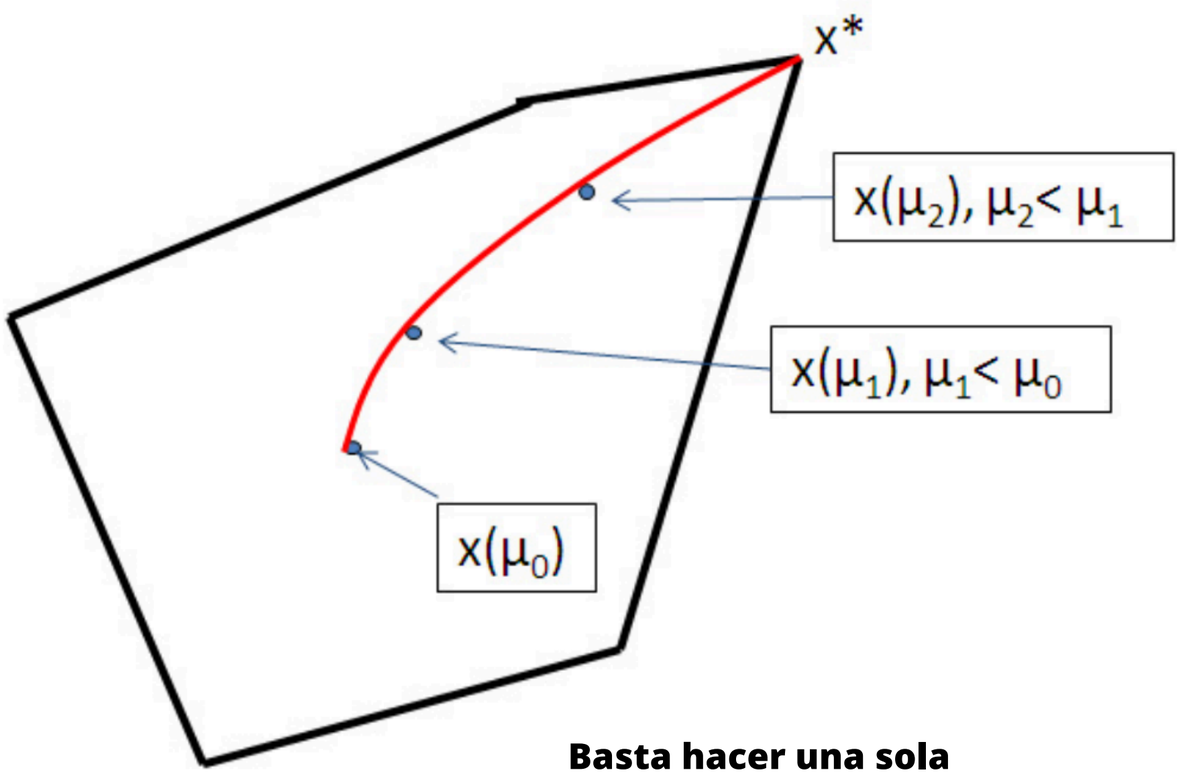
- A matriz $\in Q^{m \times n}$,
- $b \in Q^m$
- $c \in Q^n$

$$\begin{aligned} &\text{mín} \quad c^T x \\ &\text{s.a.} \; Ax \leq b \end{aligned}$$

Buscamos encontrar el vector x^* en Q^n que minimiza $c^\top x^*$ sobre $F(I)$

$$F(I) = \{x \in Q^n \mid A_j x \leq b_j \text{ for } j \in [m] \text{ and } x_i \geq 0 \text{ for } i \in [n]\}$$

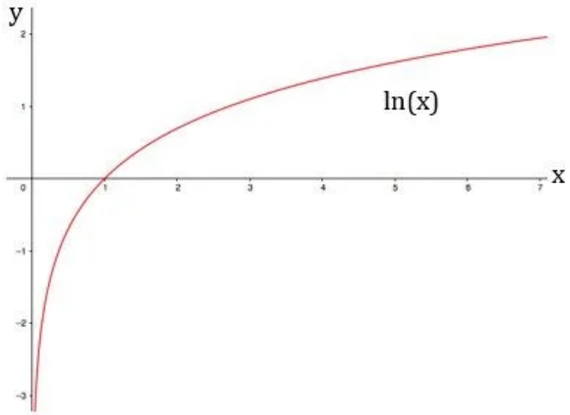
Interior-point methods for linear optimization (IPMs)



Basta hacer una sola iteración del Método de Newton para mantener la cercanía adecuada a la trayectoria central

$$P_\mu) \quad \min c^T x + \mu\Phi(x)$$

$$\Phi(x) = -\sum_{i=1}^{m} \log\left(b_i - \alpha_i^T x\right)$$



$$\lim_{b_i \to \alpha_i^T x} -\log\left(b_i - \alpha_i^T x\right) \to \lim_{\delta \to 0} -\log(\delta) \to \infty$$

Interior-point methods for linear optimization (IPMs)

La version perturbada de la LP (1),

$$\min_{x \in Q^n} c^\top x - \mu[1^\top \log(Ax - b) + 1^\top \log(x)] \tag{2}$$

Interior-point methods for linear optimization (IPMs)

La version perturbada de la LP (1),

$$\min_{x \in Q^n} c^\top x - \mu[1^\top \log(Ax - b) + 1^\top \log(x)] \tag{2}$$

para un $\mu > 0$. Introducimos las variables:

$$s_i = \mu / x_i \qquad r_j = A_j x - b_j \qquad w_j = \mu / r_j$$

Las condiciones de primer orden para (2) pueden ser esscritos como:

$$Ax - r^* = b$$
$$A^\top w^* + s^* = c$$
$$x_i^* s_i^* = \mu \quad i \in [n],$$
$$w_j^* r_j^* = \mu \quad j \in [m],$$

con $x^*, w^*, s^*, r^* \geq 0$.

Interior-point methods for linear optimization (IPMs)

Sea $\sigma \in (0,1)$. Los dos algoritmos parten desde un punto inicial positivo $(x_0, w_0, s_0, r_0) > 0$, y alternan entre computar un paso de Newton para el problema perturbado (2) con parametro de barrera $\sigma\mu$.

$$\begin{bmatrix} A & 0 & 0 & -I \\ 0 & A^\top & I & 0 \\ D(s) & 0 & D(x) & 0 \\ 0 & D(r) & 0 & D(w) \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta w \\ \Delta s \\ \Delta r \end{bmatrix} = \begin{bmatrix} b - Ax + r \\ c - A^\top w - s \\ \sigma\mu 1 - D(x)D(s)1 \\ \sigma\mu 1 - D(w)D(r)1 \end{bmatrix}$$

y dando un paso en esa direccion de tamaño $\alpha > 0$, se obtiene el punto:

$$(x', w', s', r') = (x, w, s, r) + \alpha(\Delta x, \Delta w, \Delta s, \Delta r)$$

que satisface $(x', w', s', r') > 0$.

Interior-point methods for linear optimization (IPMs)

Podemos simplificar el sistema de ecuaciones:

$$\Delta s = \sigma\mu D(x)^{-1}1 - s - D(x)^{-1}D(s)\Delta x, \tag{3}$$

$$\Delta r = \sigma\mu D(w)^{-1}1 - r - D(w)^{-1}D(r)\Delta w, \tag{4}$$

lo que implica

$$A\Delta x + D(w)^{-1}D(r)\Delta w = b - Ax + \sigma\mu D(w)^{-1}1,$$

$$A^\top \Delta w - D(x)^{-1}D(s)\Delta x = c - A^\top w - \sigma\mu D(x)^{-1}1.$$

con esto tenemos Δx y $Q\Delta w$

$$\Delta x = D(s)^{-1}D(x)\left[A^\top \Delta w - c + A^\top w + \sigma\mu D(x)^{-1}1\right] \tag{5}$$

$$Q\Delta w = b - Ax + \sigma\mu D(w)^{-1}1 + AD(s)^{-1}D(x)\left[c - A^\top w - \sigma\mu D(x)^{-1}1\right] \tag{6}$$

para $Q = AD(s)^{-1}D(x)A^\top + D(w)^{-1}D(r)$.

Resolviendo el sistema de ecuaciones (6), podemos encontrar Δw, y entonces Δx, Δr, y Δs a través de las ecuaciones (3)-(5).

Interior-point methods for linear optimization (IPMs)

Algorithm 1 Theoretical IPM for LPs

Input: An LP instance $(\boldsymbol{A}, \boldsymbol{b}, \boldsymbol{c})$, a barrier reduction hyperparameter $\sigma \in (0,1)$, a neighborhood hyperparameter $\gamma \in (0,1]$ and initial values $(\boldsymbol{x}_0, \boldsymbol{w}_0, \boldsymbol{s}_0, \boldsymbol{r}_0)$ such that $\boldsymbol{A}\boldsymbol{x}_0 - \boldsymbol{r}_0 = \boldsymbol{b}$, $\boldsymbol{A}^\mathsf{T}\boldsymbol{w}_0 + \boldsymbol{s}_0 = \boldsymbol{c}$, $(\boldsymbol{x}_0, \boldsymbol{w}_0, \boldsymbol{s}_0, \boldsymbol{r}_0) > 0$ and $\min_i x_{0i} w_{0i} \geq \gamma\mu_0$, $\min_i w_{0i} r_{0i} \geq \gamma\mu_0$ for $\mu_0 = (\boldsymbol{x}_0^\mathsf{T}\boldsymbol{s}_0 + (\boldsymbol{w}_0^\mathsf{T})_0)/(n+m)$.

repeat

$\mu \leftarrow (\boldsymbol{x}^T\boldsymbol{s} + \boldsymbol{w}^\mathsf{T}\boldsymbol{r})/(n+m)$

Compute $\Delta\boldsymbol{w}$ by solving the linear system

$$\boldsymbol{Q}\Delta\boldsymbol{w} = \boldsymbol{b} - \boldsymbol{A}\boldsymbol{x} + \sigma\mu\boldsymbol{D}(\boldsymbol{w})^{-1}\mathbf{1} + \boldsymbol{A}\boldsymbol{D}(\boldsymbol{s})^{-1}\boldsymbol{D}(\boldsymbol{x})[\boldsymbol{c} - \boldsymbol{A}^\mathsf{T}\boldsymbol{w} - \sigma\mu\boldsymbol{D}(\boldsymbol{x})^{-1}\mathbf{1}]$$

for $\boldsymbol{Q} = \boldsymbol{A}\boldsymbol{D}(\boldsymbol{s})^{-1}\boldsymbol{D}(\boldsymbol{x})\boldsymbol{A}^\mathsf{T} + \boldsymbol{D}(\boldsymbol{w})^{-1}\boldsymbol{D}(\boldsymbol{r})$

$\Delta\boldsymbol{x} \leftarrow \boldsymbol{D}(\boldsymbol{s})^{-1}\boldsymbol{D}(\boldsymbol{x})[\boldsymbol{A}^\mathsf{T}\Delta\boldsymbol{w} - \boldsymbol{c} + \boldsymbol{A}^\mathsf{T}\boldsymbol{w} + \sigma\mu\boldsymbol{D}(\boldsymbol{x})^{-1}\mathbf{1}]$

$\Delta\boldsymbol{s} \leftarrow \sigma\mu\boldsymbol{D}(\boldsymbol{x})^{-1}\mathbf{1} - \boldsymbol{s} - \boldsymbol{D}(\boldsymbol{x})^{-1}\boldsymbol{D}(\boldsymbol{s})\Delta\boldsymbol{x}$

$\Delta\boldsymbol{r} \leftarrow \sigma\mu\boldsymbol{D}(\boldsymbol{w})^{-1}\mathbf{1} - \boldsymbol{r} - \boldsymbol{D}(\boldsymbol{w})^{-1}\boldsymbol{D}(\boldsymbol{r})\Delta\boldsymbol{w}$

Compute the largest $\alpha \in (0,1)$ such that

$$\min_{i,j}\{(\boldsymbol{x} + \alpha\Delta\boldsymbol{x})_i(\boldsymbol{s} + \alpha\Delta\boldsymbol{s})_i, (\boldsymbol{w} + \alpha\Delta\boldsymbol{w})_j(\boldsymbol{r} + \alpha\Delta\boldsymbol{r})_j\} \geq \gamma\frac{(\boldsymbol{x} + \alpha\Delta\boldsymbol{x})^\mathsf{T}(\boldsymbol{s} + \alpha\Delta\boldsymbol{s}) + (\boldsymbol{w} + \alpha\Delta\boldsymbol{w})^\mathsf{T}(\boldsymbol{r} + \alpha\Delta\boldsymbol{r})}{n+m}.$$

Update $(\boldsymbol{x}, \boldsymbol{w}, \boldsymbol{s}, \boldsymbol{r}) \mathrel{+}= \alpha(\Delta\boldsymbol{x}, \Delta\boldsymbol{w}, \Delta\boldsymbol{s}, \Delta\boldsymbol{r})$

until convergence of $(\boldsymbol{x}, \boldsymbol{w}, \boldsymbol{s}, \boldsymbol{r})$

return the point $\boldsymbol{x}$, which solves 1.

$$\mathcal{O}((n+m)\log(1/\epsilon)) \text{ iterations}$$

Interior-point methods for linear optimization (IPMs)

Algorithm 2 Practical IPM for LPs

Input: An LP instance $(\boldsymbol{A}, \boldsymbol{b}, \boldsymbol{c})$, a barrier reduction hyperparameter $\sigma \in (0, 1)$ and initial values $(\boldsymbol{x}_0, \boldsymbol{w}_0, \boldsymbol{s}_0, \boldsymbol{r}_0, \mu_0)$ such that $(\boldsymbol{x}_0, \boldsymbol{w}_0, \boldsymbol{s}_0, \boldsymbol{r}_0) > 0$ and $\mu_0 = (\boldsymbol{x}_0^\mathsf{T}\boldsymbol{s}_0 + \boldsymbol{w}_0^\mathsf{T}\boldsymbol{r}_0)/(n+m)$.

repeat

Compute $\Delta\boldsymbol{w}$ by solving the linear system

$$\boldsymbol{Q}\Delta\boldsymbol{w} = \boldsymbol{b} - \boldsymbol{A}\boldsymbol{x} + \sigma\mu\boldsymbol{D}(\boldsymbol{w})^{-1}\boldsymbol{1} + \boldsymbol{A}\boldsymbol{D}(\boldsymbol{s})^{-1}\boldsymbol{D}(\boldsymbol{x})[\boldsymbol{c} - \boldsymbol{A}^\mathsf{T}\boldsymbol{w} - \sigma\mu\boldsymbol{D}(\boldsymbol{x})^{-1}\boldsymbol{1}]$$

for $\boldsymbol{Q} = \boldsymbol{A}\boldsymbol{D}(\boldsymbol{s})^{-1}\boldsymbol{D}(\boldsymbol{x})\boldsymbol{A}^\mathsf{T} + \boldsymbol{D}(\boldsymbol{w})^{-1}\boldsymbol{D}(\boldsymbol{r})$.

$\Delta\boldsymbol{x} \leftarrow \boldsymbol{D}(\boldsymbol{s})^{-1}\boldsymbol{D}(\boldsymbol{x})[\boldsymbol{A}^\mathsf{T}\Delta\boldsymbol{w} - \boldsymbol{c} + \boldsymbol{A}^\mathsf{T}\boldsymbol{w} + \sigma\mu\boldsymbol{D}(\boldsymbol{x})^{-1}\boldsymbol{1}]$

$\Delta\boldsymbol{s} \leftarrow \sigma\mu\boldsymbol{D}(\boldsymbol{x})^{-1}\boldsymbol{1} - \boldsymbol{s} - \boldsymbol{D}(\boldsymbol{x})^{-1}\boldsymbol{D}(\boldsymbol{s})\Delta\boldsymbol{x}$

$\Delta\boldsymbol{r} \leftarrow \sigma\mu\boldsymbol{D}(\boldsymbol{w})^{-1}\boldsymbol{1} - \boldsymbol{r} - \boldsymbol{D}(\boldsymbol{w})^{-1}\boldsymbol{D}(\boldsymbol{r})\Delta\boldsymbol{w}$

Find the largest $\alpha > 0$ such that

$$\min_{i,j}\{(\boldsymbol{x} + \alpha\Delta\boldsymbol{x})_i(\boldsymbol{s} + \alpha\Delta\boldsymbol{s})_i, (\boldsymbol{w} + \alpha\Delta\boldsymbol{w})_j(\boldsymbol{r} + \alpha\Delta\boldsymbol{r})_j\} \geq 0$$

Update $(\boldsymbol{x}, \boldsymbol{w}, \boldsymbol{s}, \boldsymbol{r}) \mathrel{+}= 0.99\alpha(\Delta\boldsymbol{x}, \Delta\boldsymbol{w}, \Delta\boldsymbol{s}, \Delta\boldsymbol{r})$

$\mu \leftarrow \sigma\mu$

until convergence of $(\boldsymbol{x}, \boldsymbol{w}, \boldsymbol{s}, \boldsymbol{r})$

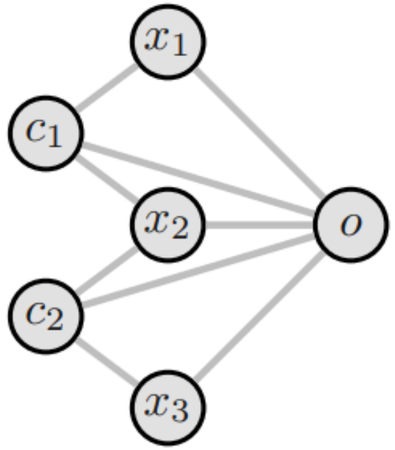
return the point $\boldsymbol{x}$, which solves 1.

Simular IPMs con MPNNs

¿Como las MPNNs simulan el comportamiento de los algoritmos anteriores?

$$\boldsymbol{h}_v^{(t)} := \mathsf{UPD}^{(t)}\left(\boldsymbol{h}_v^{(t-1)}, \mathsf{MSG}^{(t)}\left(\{\!\{\boldsymbol{h}_u^{(t-1)} \mid u \in N(v)\}\!\}\right)\right)$$

Representación de LPs como grafos



(a) Representing an LP instance with two constraints and three variables as a tripartite graph.

Representación de LPs como grafos

Sea $I = (A, b, c)$

Grafo tripartito ponderado $G(I) := (V(I), C(I), \{o\}, E(I)_{vc}, E(I)_{vo}, E(I)_{co})$.

$V(I) := \{v_i \mid i \in [n]\}$ conjunto de nodos que representan las variables de I.

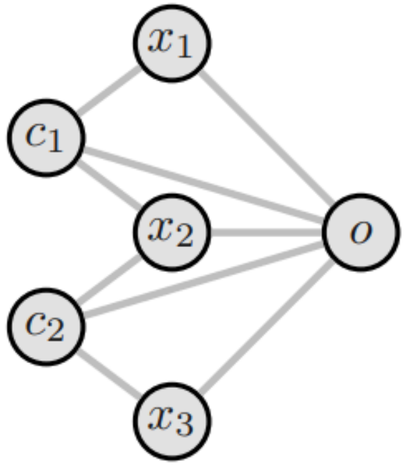
$C(I) := \{c_i \mid i \in [m]\}$ conjunto de nodos que representa las restricciones de I.

$o :=$ Nodo que representa el objetivo. Está conectado a todos los demás nodos.

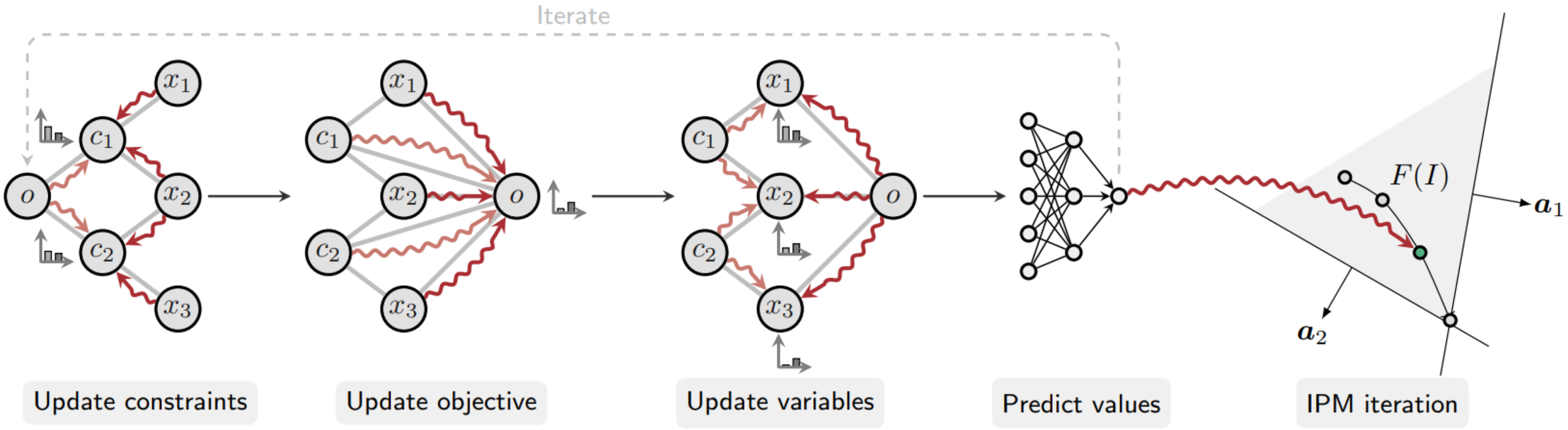
$E(I)_{vc} := \{(v_i, c_j) \mid A_{ij} \neq 0\}$ conjunto de aristas que modela la interacción variable-restricción, cada arista (v_i, c_j) está anotada con el peso A_{ij}.

$E(I)_{vo} := \{(o, v_i) \mid v_i \in V(I)\}$ conjunto de aristas que modela la interacción variable-objetivo, cada arista (o, v_i) está anotada con el peso c_i.

$E(I)_{co} := \{(o, c_i) \mid c_i \in C(I)\}$ conjunto de aristas que modela la interacción restricción-objetivo, cada arista (o, c_i) está anotada con el peso b_i.



Simular IPMs con MPNNs



(b) IPM-MPNNs emulate interior-point methods.

Simular IPMs con MPNNs

Theorem 1. There exists an MPNN $f_{\text{MPNN,IPM1}}$ composed of $\mathcal{O}(m)$ message-passing steps that reproduces an iteration of Algorithm 1, in the sense that for any LP instance $I = (A, b, c)$ and any iteration step $t \geq 0$, $f_{\text{MPNN,IPM1}}$ maps the graph $G(I)$ carrying $[x_t, s_t]$ on the variable nodes and $[w_t, r_t]$ on the constraint nodes to the same graph $G(I)$ carrying $[x_{t+1}, s_{t+1}]$ on the variable nodes and $[w_{t+1}, r_{t+1}]$ on the constraint nodes.

This implies, by composing several instances of $f_{\text{MPNN,IPM1}}$, that Algorithm 1 can be simulated by an MPNN with a number of layers proportional to the number of iterations taken by the algorithm.

Simular IPMs con MPNNs

Proposition 2. There exists an MPNN $f_{\text{MPNN,IPM2}}$ composed of $\mathcal{O}(m)$ message-passing steps that reproduces each iteration of Algorithm 2, in the sense that for any LP instance $I = (A, b, c)$ and any iteration step $t \geq 0$, $f_{\text{MPNN,IPM2}}$ maps the graph $G(I)$ carrying $[x_t, s_t]$ on the variable nodes, $[w_t, r_t]$ on the constraint nodes and $[\mu_t]$ on the objective node to the same graph $G(I)$ carrying $[x_{t+1}, s_{t+1}]$ on the variable nodes, $[w_{t+1}, r_{t+1}]$ on the constraint nodes and $[\mu_{t+1}]$ on the objective node.

IPM-MPNN Framework

IPM-MPNN Framework (Embeddings)

MPNN asincrónica que opera sobre el grafo tripartito $G(I)$

$h_c^{(t)} \in R^d$: Node feature de una restricción $c \in C(I)$ en la iteración $t > 0$.

$h_v^{(t)} \in R^d$: Node feature de una variable $v \in V(I)$ en la iteración $t > 0$.

$h_o^{(t)} \in R^d$: Node feature del nodo objetivo o en la iteración t.

e_{co}, e_{vc}, e_{vo}: Edge features.

En $t = 0$: Se establecen con transformaciones lineales los datos iniciales.

IPM-MPNN Framework (Embeddings)

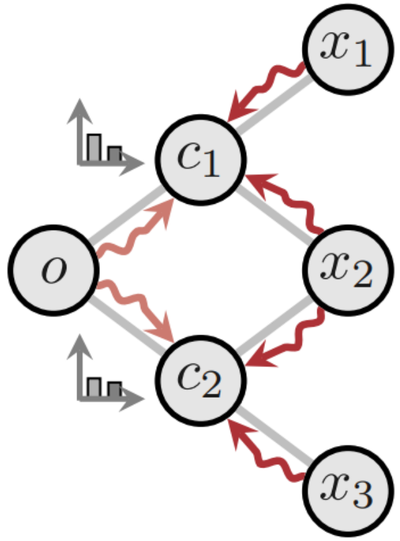
$$\boldsymbol{h}_c^{(t)} := \mathrm{UPD}_{\mathrm{c}}^{(t)}\left[\boldsymbol{h}_c^{(t-1)}, \mathrm{MSG}_{\mathrm{o}\to\mathrm{c}}^{(t)}\left(\boldsymbol{h}_o^{(t-1)}, \boldsymbol{e}_{oc}\right), \mathrm{MSG}_{\mathrm{v}\to\mathrm{c}}^{(t)}\left(\{\!\{(\boldsymbol{h}_v^{(t-1)}, \boldsymbol{e}_{vc}) \mid v \in N(c) \cap V(I)\}\!\}\right)\right]$$

$$\boldsymbol{h}_{\mathrm{o}}^{(t)} := \mathrm{UPD}_{\mathrm{o}}^{(t)}\left[\boldsymbol{h}_o^{(t-1)}, \mathrm{MSG}_{\mathrm{c}\to\mathrm{o}}^{(t)}\left(\{\!\{\boldsymbol{h}_c^{(t)}, \boldsymbol{e}_{co} \mid c \in C(I)\}\!\}\right) \mathrm{MSG}_{\mathrm{v}\to\mathrm{o}}^{(t)}\left(\{\!\{\boldsymbol{h}_v^{(t-1)}, \boldsymbol{e}_{vo} \mid v \in V(I)\}\!\}\right)\right].$$

$$\boldsymbol{h}_v^{(t)} := \mathrm{UPD}_{\mathrm{v}}^{(t)}\left[\boldsymbol{h}_v^{(t-1)}, \mathrm{MSG}_{\mathrm{o}\to\mathrm{v}}^{(t)}\left(\boldsymbol{h}_o^{(t)}, \boldsymbol{e}_{ov}\right), \mathrm{MSG}_{\mathrm{c}\to\mathrm{v}}^{(t)}\left(\{\!\{\boldsymbol{h}_c^{(t)}, \boldsymbol{e}_{cv} \mid c \in N(v) \cap V(C)\}\!\}\right)\right]$$

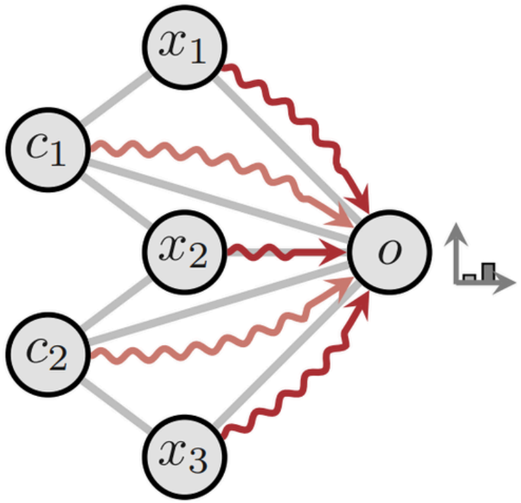
IPM-MPNN Framework (Embeddings)

$$\boldsymbol{h}_c^{(t)} := \mathsf{UPD}_{\mathrm{c}}^{(t)}\left[\boldsymbol{h}_c^{(t-1)}, \mathsf{MSG}_{\mathrm{o}\to\mathrm{c}}^{(t)}\left(\boldsymbol{h}_o^{(t-1)}, \boldsymbol{e}_{oc}\right), \mathsf{MSG}_{\mathrm{v}\to\mathrm{c}}^{(t)}\left(\left\{\!\!\left\{(\boldsymbol{h}_v^{(t-1)}, \boldsymbol{e}_{vc}) \mid v \in N(c) \cap V(I)\right\}\!\!\right\}\right)\right]$$



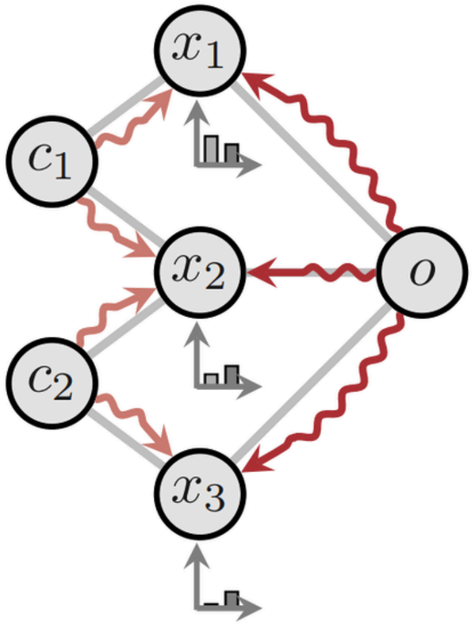
IPM-MPNN Framework (Embeddings)

$$\boldsymbol{h}_{\mathrm{o}}^{(t)} := \mathsf{UPD}_{\mathrm{o}}^{(t)}\Big[\boldsymbol{h}_{o}^{(t-1)}, \mathsf{MSG}_{\mathrm{c}\to\mathrm{o}}^{(t)}\Big(\{\!\{\boldsymbol{h}_{c}^{(t)}, \boldsymbol{e}_{co} \mid c \in C(I)\}\!\}\Big) \\ \mathsf{MSG}_{\mathrm{v}\to\mathrm{o}}^{(t)}\Big(\{\!\{\boldsymbol{h}_{v}^{(t-1)}, \boldsymbol{e}_{vo} \mid v \in V(I)\}\!\}\Big)\Big].$$



IPM-MPNN Framework (Embeddings)

$$\boldsymbol{h}_v^{(t)} := \mathsf{UPD}_{\mathrm{v}}^{(t)}\left[\boldsymbol{h}_v^{(t-1)}, \mathsf{MSG}_{\mathrm{o}\to\mathrm{v}}^{(t)}\left(\boldsymbol{h}_o^{(t)}, \boldsymbol{e}_{ov}\right), \mathsf{MSG}_{\mathrm{c}\to\mathrm{v}}^{(t)}\left(\left\{\!\!\left\{\boldsymbol{h}_c^{(t)}, \boldsymbol{e}_{cv} \mid c \in N(v) \cap V(C)\right\}\!\!\right\}\right)\right]$$

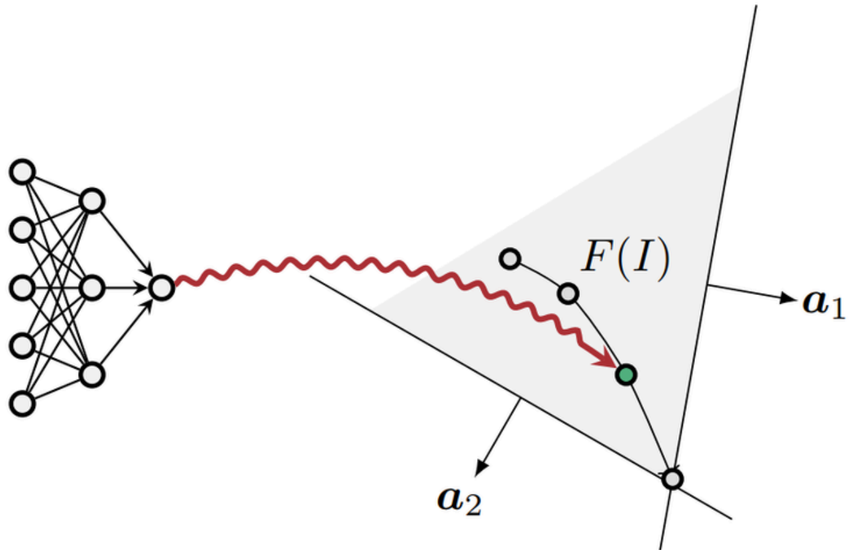


IPM-MPNN Framework (Embeddings)

Todo el proceso se ejecuta de manera asincrónica.

Mapeamos cada característica del nodo variable $h_v^{(t)}$ a $MLP(h_v^{(t)}) \in R$.

Concatenamos todos los $MLP(h_v^{(t)})$ resultantes, para obtener la predicción final $z^{(t)} \in R^n$.



IPM-MPNN Framework (Loss function)

$$\mathcal{L}_{\text{var}} := \frac{1}{N} \sum_{i=1}^{N} \sum_{t=1}^{T} \alpha^{T-t} \| \boldsymbol{y}_i^{(t)} - \boldsymbol{z}_i^{(t)} \|_2^2, \qquad (7)$$

α^{T-t} disminuye conforme t disminuye.

IPM-MPNN Framework (Loss function)

$$\mathcal{L}_{\text{obj}} := \frac{1}{N} \sum_{i=1}^{N} \sum_{t=1}^{T} \alpha^{T-t} \left[\boldsymbol{c}^{\mathsf{T}} \left(\boldsymbol{y}_i^{(t)} - \boldsymbol{z}_i^{(t)} \right) \right]^2. \qquad (8)$$

α^{T-t} disminuye conforme t disminuye.

IPM-MPNN Framework (Loss function)

$$\mathcal{L}_{\text{cons}} := \frac{1}{N} \sum_{i=1}^{N} \sum_{t=1}^{T} \alpha^{T-t} \left\| \mathsf{ReLU}\left(\boldsymbol{A}_i \boldsymbol{z}_i^{(t)} - \boldsymbol{b}_i \right) \right\|_2^2. \quad (9)$$

α^{T-t} disminuye conforme t disminuye.

IPM-MPNN Framework (Loss function)

$$\mathcal{L} := w_{\text{var}}\mathcal{L}_{\text{var}} + w_{\text{obj}}\mathcal{L}_{\text{obj}} + w_{\text{cons}}\mathcal{L}_{\text{cons}}, \qquad (10)$$

$w_{\text{var}}, w_{\text{obj}}, \text{y } w_{\text{cons}} > 0$ son hyper-parametros.

Estudio Experimental

Estudio Experimental (Dataset)

Combinatorial optimization problems:

- Set cover problem
- Maximal independent set
- Combinatorial auction
- Capacitated facility location

Estudio Experimental (Dataset)

Table 4: Sizes of Setcover.

Size	Num. Row	Num. Col.	Density	Num. instances
Mini	[15, 20]	[15, 20]	0.15	1000
Small	[30, 50]	[50, 70]	0.05	10 000
Large	[300, 500]	[500, 700]	0.01	10 000

Estudio Experimental (Dataset)

Table 5: Sizes of maximal independent set instances.

Size	Num. nodes	Affinity	Num. instances
Mini	[10, 20]	2	1000
Small	[50, 70]	2	10 000
Large	[500, 700]	2	10 000

Estudio Experimental (Dataset)

Table 6: Sizes of combinatorial auction instances.

Size	Num. items	Num. bids	Num. instances
Mini	20	20	1000
Small	[50, 80]	[50, 80]	10 000
Large	[300, 500]	[300, 500]	10 000

Estudio Experimental (Dataset)

Table 7: Sizes of capacitated facility location instances.

Size	Num. customers	Num. facilities	Ratio	Num. instances
Mini	[3, 5]	[3, 5]	5	1000
Small	10	10	5	10 000
Large	[20, 30]	[20, 30]	5	10 000

Estudio Experimental (Dataset)

Sea A_i la fila i de la matriz de restricciones y $A_{.,j}$ como la columna j.

Para $v_j \in V(I)$: Sus características iniciales se establecen con la media y la desviación estándar del vector columna $A_{.,j}$, resultando en dos características.

Para $c_i \in C(I)$: Derivamos características iniciales a partir de las propiedades estadísticas del vector fila A_i.

Para o: Características se construyen de manera similar usando el vector c.

Estudio Experimental

División de datos: Se dividio los conjuntos de datos en entrenamiento, validación y prueba con proporciones 0.8, 0.1 y 0.1.

Experimentos: Se realizaron evaluando cada conjunto de datos múltiples veces, realizando tres ejecuciones independientes con semillas aleatorias distintas. Los resultados reportados representan los promedios sobre el conjunto de prueba a lo largo de estas ejecuciones y las desviaciones estándar correspondientes.

Hardware: Ejecución de todos los experimentos en una sola tarjeta gráfica NVIDIA A100 de 80GB

Resultados

Resultados

Q1 Can MPNNs properly imitate the performance of IPM solvers in practice?
Q2 How is MPNNs' performance compared with competing neural-network-based solutions?
Q3 What advantage does our MPNN architecture hold compared with traditional IPM solvers?
Q4 Do MPNNs generalize to instances larger than seen during training?

Resultados

Table 1: Results of our proposed IPM-MPNNs (✓) versus bipartite representation ablations (✗). We report the relative objective gap and the constraint violation, averaged over all three runs. We print the best results per target in bold.

	Tri.	MPNN	Small instances				Large instances			
			Setcover	Indset	Cauc	Fac	Setcover	Indset	Cauc	Fac
Objective gap [%]	✓	GEN	**0.319±0.020**	0.119±0.003	**0.612±0.049**	**0.549±0.112**	0.629±0.086	0.158±0.035	**0.306±0.047**	**0.747±0.083**
		GCN	0.418±0.008	**0.103±0.006**	0.682±0.029	0.578±0.015	**0.420±0.047**	**0.094±0.005**	0.407±0.038	0.914±0.141
		GIN	0.478±0.038	0.146±0.011	0.632±0.036	0.810±0.221	0.711±0.115	0.126±0.021	0.378±0.052	0.911±0.132
	✗	GEN	8.310±1.269	0.735±0.032	1.417±0.009	2.976±0.013	15.170±6.844	0.320±0.008	0.851±0.122	2.531±0.025
		GCN	5.523±0.133	0.639±0.009	1.394±0.081	3.031±0.059	6.092±0.456	0.298±0.009	0.766±0.093	2.535±0.034
		GIN	5.592±0.179	0.634±0.021	1.202±0.016	2.996±0.031	5.835±1.917	0.290±0.005	0.810±0.140	2.660±0.062
Constraint violation	✓	GEN	**0.002±0.0002**	0.0006±0.00003	0.003±0.0007	0.002±0.001	0.009±0.001	0.0015±0.0003	**0.0004±0.0002**	0.002±0.001
		GCN	0.002±0.001	**0.0003±0.0001**	0.002±0.00007	**0.002±0.0002**	0.009±0.001	**0.0005±0.00004**	0.001±0.0005	**0.001±0.0004**
		GIN	0.004±0.001	0.0006±0.00008	**0.001±0.0001**	0.002±0.0005	**0.008±0.002**	0.0006±0.0001	0.002±0.0008	0.002±0.0007
	✗	GEN	0.181±0.023	0.006±0.0003	0.006±0.001	0.011±0.004	0.309±0.025	0.004±0.0002	0.006±0.001	0.003±0.001
		GCN	0.207±0.006	0.004±0.001	0.002±0.001	0.006±0.0003	0.267±0.049	0.003±0.0004	0.004±0.001	0.002±0.0003
		GIN	0.211±0.007	0.003±0.0002	0.003±0.001	0.008±0.002	0.236±0.014	0.003±0.0004	0.004±0.002	0.003±0.0002

Resultados

$$\frac{1}{N}\sum_{i=1}^{N}\left|\frac{\boldsymbol{c}^{\mathsf{T}}\left(\boldsymbol{y}_i^{(T)}-\boldsymbol{z}_i^{(T)}\right)}{\boldsymbol{c}^{\mathsf{T}}\boldsymbol{y}_i^{(T)}}\right|\times 100$$

mean absolute relative objective gap of the last step T

$$\frac{1}{N}\sum_{i=1}^{N}\frac{1}{m_i}\left\|\mathsf{reLU}\left(\boldsymbol{A}_i\boldsymbol{z}_i^{(T)}-\boldsymbol{b}_i\right)\right\|_1,$$

mean absolute constraint violation of the last step T

Resultados

	Method	MPNN	Setcover	Indset	Cauc	Fac
Obj. gap [%]	ODE	GEN	14.915±0.425	6.225±0.097	13.845±0.554	20.560±0.059
		GCN	14.545±0.055	6.148±0.071	12.945±0.385	20.690±0.037
		GIN	15.050±0.228	6.474±0.114	13.470±1.145	21.010±0.529
	Ours	GEN	2.555±0.122	1.580±0.095	**2.733±0.074**	1.449±0.255
		GCN	**2.375±0.062**	1.447±0.152	2.769±0.091	1.478±0.154
		GIN	2.740±0.3184	**1.404±0.153**	2.847±0.091	**1.328±0.201**
Constraint vio.	ODE	GEN	0.072±0.006	0.046±0.002	0.025±0.008	0.020±0.001
		GCN	0.049±0.012	0.048±0.008	0.025±0.0002	0.020±0.0005
		GIN	0.064±0.005	0.043±0.008	0.024±0.005	0.014±0.004
	Ours	GEN	**0.023±0.002**	0.005±0.0001	0.015±0.003	0.013±0.003
		GCN	0.030±0.003	0.005±0.0006	0.017±0.002	**0.005±0.0006**
		GIN	0.023±0.005	**0.005±0.0003**	**0.014±0.001**	0.006±0.0006
Time [s]	ODE	GEN	47.829	51.283	63.068	96.298
		GCN	57.196	80.133	79.606	34.297
		GIN	55.918	64.628	39.904	62.448
	Ours	GEN	10.177	9.617	9.946	11.124
		GCN	18.964	8.688	**7.368**	**8.834**
		GIN	**6.042**	**8.096**	8.881	10.771
Memory (GB)	ODE	GEN	16.455	25.931	23.354	44.520
		GCN	16.489	34.003	23.805	10.640
		GIN	18.238	30.101	13.482	24.713
	Ours	GEN	**0.091**	0.088	0.101	0.148
		GCN	0.201	0.134	**0.069**	**0.142**
		GIN	0.094	**0.073**	0.148	0.187

Table 2: Comparing between IPM-MPNNs and the Wu and Lisser (2023) method on 1000 mini-sized instances. We report the average relative objective gap, constraint violation, training time over three runs, and maximal GPU memory allocated. We print the best results per target in bold.

Resultados

Table 3: Comparing IPM-MPNNs' inference time to SciPy's IPM implementation and our Python-based IPM solver. We report mean and standard deviation in seconds over three runs. We print the best results per target in bold.

Instances	SciPy Solver	Our Solver	GEN	GCN	GIN
Small setcover	**0.006±0.004**	0.071±0.015	0.033±0.001	0.029±0.001	0.017±0.001
Large setcover	0.390±0.098	3.696±2.141	0.033±0.001	0.030±0.001	**0.021±0.001**
Small indset	**0.008±0.067**	0.089±0.024	0.033±0.001	0.031±0.002	0.021±0.001
Large indset	0.226±0.087	1.053±0.281	0.033±0.002	0.030±0.001	**0.021±0.001**
Small cauc	**0.012±0.005**	0.151±0.035	0.033±0.001	0.028±0.001	0.021±0.001
Large cauc	0.282±0.065	3.148±0.880	0.033±0.001	0.029±0.001	**0.021±0.001**
Small fac	**0.017±0.011**	2.025±1.854	0.029±0.001	0.029±0.001	0.022±0.001
Large fac	0.732±0.324	6.229±2.672	0.030±0.001	0.031±0.001	**0.022±0.001**

Resultados

Table 11: Size generalization. We report the relative objective gap and constraint violation on larger test instances. Numbers represent mean and standard deviation across multiple pretrained models.

	Train size		Inference size		GEN		GCN		GIN	
	Rows	Cols	Rows	Cols	Obj. (%)	Cons.	Obj. (%)	Cons.	Obj. (%)	Cons.
Setc.	[300, 500]	[500, 700]	500	700	0.717±0.158	0.516±0.010	0.511±0.047	0.509±0.004	1.034±0.237	0.486±0.023
			550	750	0.917±0.317	0.552±0.012	0.871±0.252	0.543±0.003	2.318±1.411	0.497±0.032
			600	700	0.993±0.211	0.573±0.015	0.705±0.125	0.565±0.012	1.491±0.512	0.521±0.045
			500	800	0.902±0.323	0.528±0.008	1.058±0.441	0.509±0.004	12.538±16.027	0.485±0.050
			600	800	1.004±0.407	0.589±0.014	1.556±0.588	0.568±0.005	12.217±14.715	0.486±0.071
Indset.	[584, 990]	[300, 500]	[978, 994]	500	0.128±0.027	0.299±0.001	0.099±0.008	0.303±0.001	0.129±0.031	0.304±0.001
			[1028, 1044]	525	0.157±0.063	0.300±0.001	0.101±0.013	0.304±0.001	0.111±0.017	0.305±0.001
			[1076, 1094]	550	0.300±0.186	0.301±0.002	0.096±0.022	0.303±0.001	0.177±0.097	0.304±0.001
			[1128, 1144]	575	1.402±1.036	0.305±0.006	0.146±0.044	0.304±0.001	0.380±0.367	0.304±0.002
			[1178, 1194]	600	4.552±3.153	0.317±0.015	0.408±0.317	0.304±0.001	0.647±0.725	0.304±0.002
Cauc.	[320, 562]	[300, 499]	[530, 564]	500	0.333±0.134	0.257±0.001	0.318±0.048	0.259±0.001	0.344±0.108	0.259±0.001
			[596, 646]	500	0.363±0.131	0.267±0.002	0.519±0.069	0.270±0.003	0.576±0.165	0.271±0.001
			[652, 720]	500	0.524±0.039	0.284±0.001	1.255±0.523	0.289±0.007	0.944±0.114	0.289±0.001
			[559, 596]	600	7.325±3.615	0.257±0.002	0.587±0.268	0.255±0.001	1.014±0.845	0.263±0.006
			[633, 677]	600	7.965±3.941	0.263±0.002	0.868±0.441	0.258±0.003	1.375±0.693	0.269±0.005
Fac.	[441, 900]	[420, 870]	961	930	0.912±0.251	0.178±0.006	1.154±0.206	0.173±0.007	1.452±0.528	0.178±0.003
			936	900	1.320±0.347	0.148±0.009	1.615±0.322	0.145±0.009	1.736±0.558	0.153±0.004
			936	910	0.964±0.063	0.209±0.005	1.538±0.526	0.211±0.007	1.538±0.422	0.215±0.006
			1116	1080	1.502±0.704	0.163±0.009	3.540±3.134	0.161±0.006	2.288±0.659	0.167±0.005
			1296	1260	1.808±0.566	0.173±0.009	7.629±7.577	0.179±0.008	13.522±8.027	0.163±0.021

Anexos

Anexos

Algorithm 3 Conjugate gradient algorithm for IPMs

Input: An instance $I = (\boldsymbol{A}, \boldsymbol{b}, \boldsymbol{c})$ and a point $(\boldsymbol{x}, \boldsymbol{w}, \boldsymbol{s}, \boldsymbol{r}) > 0$.

$\boldsymbol{p} \leftarrow \boldsymbol{b} - \boldsymbol{A}\boldsymbol{x} + \sigma\mu\boldsymbol{D}(\boldsymbol{w})^{-1}\boldsymbol{1} + \boldsymbol{A}\boldsymbol{D}(\boldsymbol{s})^{-1}\boldsymbol{D}(\boldsymbol{x})[\boldsymbol{c} - \boldsymbol{A}^{\mathsf{T}}\boldsymbol{w} - \sigma\mu\boldsymbol{D}(\boldsymbol{x})^{-1}\boldsymbol{1}]$
$\boldsymbol{v} \leftarrow -\boldsymbol{p}$
$\Delta\boldsymbol{w} \leftarrow \boldsymbol{0}$
for m iterations **do**
  $\boldsymbol{u} \leftarrow [\boldsymbol{A}\boldsymbol{D}(\boldsymbol{s})^{-1}\boldsymbol{D}(\boldsymbol{x})\boldsymbol{A}^{\mathsf{T}} + \boldsymbol{D}(\boldsymbol{w})^{-1}\boldsymbol{D}(\boldsymbol{r})]\boldsymbol{p}$
  $\alpha \leftarrow \boldsymbol{v}^{\mathsf{T}}\boldsymbol{v}/\boldsymbol{p}^{\mathsf{T}}\boldsymbol{u}$
  $\Delta\boldsymbol{w} \mathrel{+}= \alpha\boldsymbol{p}$
  $\boldsymbol{v}_{\text{new}} \leftarrow \boldsymbol{v} + \alpha\boldsymbol{u}$
  $\beta \leftarrow \boldsymbol{v}_{\text{new}}^{\mathsf{T}}\boldsymbol{v}_{\text{new}}/\boldsymbol{v}^{\mathsf{T}}\boldsymbol{v}$
  $\boldsymbol{v} \leftarrow \boldsymbol{v}_{\text{new}}$
  $\boldsymbol{p} \leftarrow -\boldsymbol{v} + \beta\boldsymbol{p}$
end for
return A direction $\Delta\boldsymbol{w}$ that solves the system (6).

Anexos

We now move on with the proofs of Theorem 1 and Proposition 2.

Theorem 4. There exists an MPNN $f_{\mathsf{MPNN,IPM1}}$ composed of $\mathcal{O}(m)$ message-passing steps that reproduces an iteration of Algorithm 1, in the sense that for any LP instance $I = (\boldsymbol{A}, \boldsymbol{b}, \boldsymbol{c})$ and any iteration step $t \geq 0$, $f_{\mathsf{MPNN,IPM1}}$ maps the graph $G(I)$ carrying $[\boldsymbol{x}_t, \boldsymbol{s}_t]$ on the variable nodes and $[\boldsymbol{w}_t, \boldsymbol{r}_t]$ on the constraint nodes to the same graph $G(I)$ carrying $[\boldsymbol{x}_{t+1}, \boldsymbol{s}_{t+1}]$ on the variable nodes and $[\boldsymbol{w}_{t+1}, \boldsymbol{r}_{t+1}]$ on the constraint nodes.

Proof. We need to check that every step can be computed by message-passing steps over $G(I)$.

- Step 2 can be written as $\boldsymbol{h}_1 \leftarrow \boldsymbol{x}^\mathsf{T}\boldsymbol{s}$, $\boldsymbol{h}_2 \leftarrow \boldsymbol{w}^\mathsf{T}\boldsymbol{r}$, $\mu = (\boldsymbol{h}_1 + \boldsymbol{h}_2)/(n+m)$. These can be realized as a variable-to-objective message-passing step, a constraints-to-objective message-passing step, and a local operation on the objective node, respectively.
- Step 3 can be written as message-passing steps by Lemma 3.
- Step 4 can be broken down as follows. We can compute $\boldsymbol{h}_1 \leftarrow \boldsymbol{A}^\mathsf{T}[\boldsymbol{w} + \Delta\boldsymbol{w}]$, $\boldsymbol{h}_2 \leftarrow \mu\mathbf{1}_n$ and $\boldsymbol{h}_3 \leftarrow \boldsymbol{c}$ by a constraints-to-variables and two objective-to-variables message-passing steps, respectively. Then, one can compute $\Delta\boldsymbol{x} \leftarrow \boldsymbol{D}(\boldsymbol{s})^{-1}\boldsymbol{D}(\boldsymbol{x})\boldsymbol{h}_1 - \boldsymbol{h}_3 + \sigma\boldsymbol{D}(\boldsymbol{x})^{-1}\boldsymbol{h}_2$ by a local operation on variable nodes.
- Step 5 can be realized by taking an objective-to-variables message-passing step $\boldsymbol{h}_1 \leftarrow \mu\mathbf{1}_n$, and computing $\Delta\boldsymbol{s} \leftarrow \sigma\boldsymbol{D}(\boldsymbol{x})^{-1}\boldsymbol{h}_1 - \boldsymbol{s} - \boldsymbol{D}(\boldsymbol{x})^{-1}\boldsymbol{D}(\boldsymbol{s})\Delta\boldsymbol{x}$.

Anexos

- Step 6 can be realized by taking an objective-to-constraints message-passing step $\boldsymbol{h}_1 \leftarrow \mu \mathbf{1}_m$, and computing $\Delta \boldsymbol{r} \leftarrow \sigma \boldsymbol{D}(\boldsymbol{w})^{-1}\boldsymbol{h}_1 - \boldsymbol{r} - \boldsymbol{D}(\boldsymbol{w})^{-1}\boldsymbol{D}(\boldsymbol{r})\Delta \boldsymbol{w}$.
- Step 7 can be performed by message-passing steps as follows. We need to find the largest $\alpha \in (0, 1)$ such that

$$(\boldsymbol{x} + \alpha \Delta \boldsymbol{x})_i (\boldsymbol{s} + \alpha \Delta \boldsymbol{s})_i \tag{11}$$

$$\geq \gamma \frac{(\boldsymbol{x} + \alpha \Delta \boldsymbol{x})^{\mathsf{T}}(\boldsymbol{s} + \alpha \Delta \boldsymbol{s}) + (\boldsymbol{w} + \alpha \Delta \boldsymbol{w})^{\mathsf{T}}(\boldsymbol{r} + \alpha \Delta \boldsymbol{r})}{n + m} \tag{12}$$

for every $i \in [n]$ and

$$(\boldsymbol{w} + \alpha \Delta \boldsymbol{w})_j (\boldsymbol{r} + \alpha \Delta \boldsymbol{r})_j \} \tag{13}$$

$$\geq \gamma \frac{(\boldsymbol{x} + \alpha \Delta \boldsymbol{x})^{\mathsf{T}}(\boldsymbol{s} + \alpha \Delta \boldsymbol{s}) + (\boldsymbol{w} + \alpha \Delta \boldsymbol{w})^{\mathsf{T}}(\boldsymbol{r} + \alpha \Delta \boldsymbol{r})}{n + m} \tag{14}$$

Anexos

for every $j \in [m]$. Equivalently, for each $i \in [n]$, we can find the largest $\alpha_i < 1$ such that 12 holds, that is such that

$$\begin{aligned}
&\alpha_i^2 \left(\Delta x_i \Delta s_i - \gamma \frac{\Delta \boldsymbol{x}^\mathsf{T} \Delta \boldsymbol{s} + \Delta \boldsymbol{w}^\mathsf{T} \Delta \boldsymbol{r}}{n+m} \right) \\
&+ \alpha_i \left(x_i \Delta s_i + \Delta x_i s_i - \gamma \frac{\boldsymbol{x}^\mathsf{T} \Delta \boldsymbol{s} + \Delta \boldsymbol{x}^\mathsf{T} \boldsymbol{s} + \boldsymbol{w}^\mathsf{T} \Delta \boldsymbol{r} + \Delta \boldsymbol{w}^\mathsf{T} \boldsymbol{r}}{n+m} \right) \\
&+ \left(x_i s_i - \gamma \frac{\boldsymbol{x}^\mathsf{T} \boldsymbol{s} + \boldsymbol{w}^\mathsf{T} \boldsymbol{r}}{n+m} \right) \geq 0
\end{aligned}$$

holds; and similarly, find the largest $\bar{\alpha}_j < 1$ such that 14 holds, that is such that

$$\begin{aligned}
&\bar{\alpha}_j^2 \left(\Delta w_j \Delta r_j - \gamma \frac{\Delta \boldsymbol{x}^\mathsf{T} \Delta \boldsymbol{s} + \Delta \boldsymbol{w}^\mathsf{T} \Delta \boldsymbol{r}}{n+m} \right) \\
&+ \bar{\alpha}_j \left(w_j \Delta r_j + \Delta w_j r_j - \gamma \frac{\boldsymbol{x}^\mathsf{T} \Delta \boldsymbol{s} + \Delta \boldsymbol{x}^\mathsf{T} \boldsymbol{s} + \boldsymbol{w}^\mathsf{T} \Delta \boldsymbol{r} + \Delta \boldsymbol{w}^\mathsf{T} \boldsymbol{r}}{n+m} \right) \\
&+ \left(w_j r_j - \gamma \frac{\boldsymbol{x}^\mathsf{T} \boldsymbol{s} + \boldsymbol{w}^\mathsf{T} \boldsymbol{r}}{n+m} \right) \geq 0
\end{aligned}$$

holds; then $\alpha = \min_{i,j}\{\alpha_i, \bar{\alpha}_j\}$.

Anexos

This can be computed by message-passing steps as follows. First, we can compute $\boldsymbol{h}_1 \leftarrow \Delta\boldsymbol{x}^\mathsf{T}\Delta\boldsymbol{s}$, $\boldsymbol{h}_2 \leftarrow \Delta\boldsymbol{x}^\mathsf{T}\boldsymbol{s}$, $\boldsymbol{h}_3 \leftarrow \boldsymbol{x}^\mathsf{T}\Delta\boldsymbol{s}$ and $\boldsymbol{h}_4 \leftarrow \boldsymbol{x}^\mathsf{T}\boldsymbol{s}$ by variable-to-objective message-passing steps; and similarly $\bar{\boldsymbol{h}}_1 \leftarrow \Delta\boldsymbol{w}^\mathsf{T}\Delta\boldsymbol{r}$, $\bar{\boldsymbol{h}}_2 \leftarrow \Delta\boldsymbol{w}^\mathsf{T}\boldsymbol{r}$, $\bar{\boldsymbol{h}}_3 \leftarrow \boldsymbol{w}^\mathsf{T}\Delta\boldsymbol{r}$, $\bar{\boldsymbol{h}}_4 \leftarrow \boldsymbol{w}^\mathsf{T}\boldsymbol{r}$ by constraints-to-objective message-passing steps. The quantities $t_1 \leftarrow \gamma(\boldsymbol{h}_1 + \bar{\boldsymbol{h}}_1)/(n+m)$, $t_2 \leftarrow \gamma(\boldsymbol{h}_2 + \boldsymbol{h}_3 + \bar{\boldsymbol{h}}_2 + \bar{\boldsymbol{h}}_3)/(n+m)$ and $t_3 \leftarrow \gamma(\boldsymbol{h}_4 + \bar{\boldsymbol{h}}_4)/(n+m)$ can then be computed by local operations on the objective node, and returned to the variable and constraint nodes by objective-to-variables message-passing steps $\boldsymbol{t}_1 \leftarrow t_1\mathbf{1}_n$, $\boldsymbol{t}_2 \leftarrow t_2\mathbf{1}_n$, $\boldsymbol{t}_3 \leftarrow t_3\mathbf{1}_n$ and objective-to-constraint message-passing steps $\bar{\boldsymbol{t}}_1 \leftarrow t_1\mathbf{1}_n$, $\bar{\boldsymbol{t}}_2 \leftarrow t_2\mathbf{1}_n$, $\bar{\boldsymbol{t}}_3 \leftarrow t_3\mathbf{1}_n$. Then, on each variable node v_i, we can solve

$$\begin{aligned}\alpha_i = \max\{\alpha \in (0,1) \mid \alpha^2(\Delta x_i \Delta s_i - t_{1i}) \\ + \alpha(x_i\Delta s_i + \Delta x_i s_i - t_{2i}) + (x_i s_i - t_{3i}) \geq 0\}\end{aligned}$$

as a local operation, and similarly, on each constraint node c_j, we can find

$$\begin{aligned}\bar{\alpha}_j = \max\{\alpha \in (0,1) \mid \alpha^2(\Delta w_i \Delta r_i - \bar{t}_{1i}) \\ + \alpha(w_i\Delta r_i + \Delta w_i r_i - \bar{t}_{2i}) + (w_i r_i - \bar{t}_{3i}) \geq 0\}\end{aligned}$$

as a local operation. Finally, we can compute $\alpha_v \leftarrow \min_i \alpha_i$ as a variables-to-objective message-passing step, and $\alpha_c \leftarrow \min_j \bar{\alpha}_j$ as a constraints-to-objective message-passing step, and finally take $\alpha = \min(\alpha_v, \alpha_c)$ as a local operation on the objective node.

- Finally, step 8 can be performed by taking objective-to-variables, objective-to-constraints message-passing steps $\boldsymbol{h}_1 \leftarrow \alpha\mathbf{1}_n$, $\bar{\boldsymbol{h}}_1 \leftarrow \alpha\mathbf{1}_m$, and taking local operations $(\boldsymbol{x}, \boldsymbol{s}) \leftarrow (\boldsymbol{x} + \boldsymbol{D}(\boldsymbol{h}_1)\Delta\boldsymbol{x}, \boldsymbol{s} + \boldsymbol{D}(\boldsymbol{h}_1)\Delta\boldsymbol{s})$ and $(\boldsymbol{w}, \boldsymbol{r}) \leftarrow (\boldsymbol{w} + \boldsymbol{D}(\bar{\boldsymbol{h}}_1)\Delta\boldsymbol{w}, \boldsymbol{r} + \boldsymbol{D}(\bar{\boldsymbol{h}}_1)\Delta\boldsymbol{r})$ on variable and constraint nodes respectively.

Counting the number of successive message-passing steps, we find that all steps can be realized in 23 message-passing steps, plus the $\mathcal{O}(m)$ steps of Step 3, completing the proof. □

Anexos

C Details of IPM-MPNNs

In the following, we outline details with regard to the specific MPNN layers used in Section 5. We follow the notation outlined in Section 3. Furthermore, let MLP be a multi-layer perceptron, whose subscript denotes its role. Specifically, MLP_* is for node initialization or node updating after gathering message function, $\mathsf{MLP}_{*\to *}$ is for message vector mapping, and MLP_{**} is for edge feature embedding in each layer. At the initialization $t = 0$, we obtain node embeddings by

$$\begin{aligned}
\boldsymbol{h}_v^{(0)} &\coloneqq \mathsf{MLP}_{\mathrm{v}}^{(0)}(\boldsymbol{x}_v), \forall v \in V(I) \\
\boldsymbol{h}_c^{(0)} &\coloneqq \mathsf{MLP}_{\mathrm{c}}^{(0)}(\boldsymbol{x}_c), \forall c \in C(I) \\
\boldsymbol{h}_o^{(0)} &\coloneqq \mathsf{MLP}_{\mathrm{o}}^{(0)}(\boldsymbol{x}_o).
\end{aligned}$$

Anexos

Then, a GCN layer updates the constraint, objective, and variable nodes as follows:

$$
\begin{aligned}
\boldsymbol{h}_c^{(t)} \coloneqq \mathsf{MLP}_{\mathrm{c}}^{(t)} \Bigg[& \mathsf{MLP}_{\mathrm{c}\to\mathrm{c}}^{(t)}\left(\boldsymbol{h}_c^{(t-1)}\right) + \\
& \mathsf{MLP}_{\mathrm{o}\to\mathrm{c}}^{(t)}\left(\frac{1}{\sqrt{d_o d_c}}\left(\boldsymbol{h}_o^{(t-1)} + \mathsf{MLP}_{\mathrm{oc}}^{(t)}(\boldsymbol{e}_{oc})\right)\right) + \\
& \mathsf{MLP}_{\mathrm{v}\to\mathrm{c}}^{(t)}\left(\sum_{v \in N_c \cap V(I)} \frac{1}{\sqrt{d_v d_c}}\left(\boldsymbol{h}_v^{(t-1)} + \mathsf{MLP}_{\mathrm{vc}}^{(t)}(\boldsymbol{e}_{vc})\right)\right)\Bigg] \\
\boldsymbol{h}_o^{(t)} \coloneqq \mathsf{MLP}_{\mathrm{o}}^{(t)} \Bigg[& \mathsf{MLP}_{\mathrm{o}\to\mathrm{o}}^{(t)}\left(\boldsymbol{h}_o^{(t-1)}\right) + \\
& \mathsf{MLP}_{\mathrm{c}\to\mathrm{o}}^{(t)}\left(\sum_{c \in C(I)} \frac{1}{\sqrt{d_o d_c}}\left(\boldsymbol{h}_c^{(t)} + \mathsf{MLP}_{\mathrm{co}}^{(t)}(\boldsymbol{e}_{co})\right)\right) + \\
& \mathsf{MLP}_{\mathrm{v}\to\mathrm{o}}^{(t)}\left(\sum_{v \in V(I)} \frac{1}{\sqrt{d_o d_v}}\left(\boldsymbol{h}_v^{(t-1)} + \mathsf{MLP}_{\mathrm{vo}}^{(t)}(\boldsymbol{e}_{vo})\right)\right)\Bigg]
\end{aligned}
$$

Anexos

$$
\begin{aligned}
\boldsymbol{h}_v^{(t)} := \mathrm{MLP}_{\mathrm{v}}^{(t)} \Bigg[& \mathrm{MLP}_{\mathrm{v}\to\mathrm{v}}^{(t)}\left(\boldsymbol{h}_v^{(t-1)}\right) + \\
& \mathrm{MLP}_{\mathrm{o}\to\mathrm{v}}^{(t)}\left(\frac{1}{\sqrt{d_o d_c}}\left(\boldsymbol{h}_o^{(t)} + \mathrm{MLP}_{\mathrm{ov}}^{(t)}(\boldsymbol{e}_{ov})\right)\right) + \\
& \mathrm{MLP}_{\mathrm{c}\to\mathrm{v}}^{(t)}\left(\sum_{c\in N_v \cap C(I)} \frac{1}{\sqrt{d_c d_v}}\left(\boldsymbol{h}_c^{(t)} + \mathrm{MLP}_{\mathrm{cv}}^{(t)}(\boldsymbol{e}_{cv})\right)\right)\Bigg].
\end{aligned}
$$

Anexos

Similarly, for the GIN layer, we have:

$$
\begin{aligned}
\boldsymbol{h}_c^{(t)} \coloneqq \mathsf{MLP}_{\mathrm{c}}^{(t)} \Bigg[& \left(1+\epsilon_c^{(t)}\right) \mathsf{MLP}_{\mathrm{c}\to\mathrm{c}}^{(t)}\left(\boldsymbol{h}_c^{(t-1)}\right) + \\
& \mathsf{MLP}_{\mathrm{o}\to\mathrm{c}}^{(t)}\left(\boldsymbol{h}_o^{(t-1)} + \mathsf{MLP}_{\mathrm{oc}}^{(t)}(\boldsymbol{e}_{oc})\right) + \\
& \mathsf{MLP}_{\mathrm{v}\to\mathrm{c}}^{(t)}\left(\sum_{v\in N_c\cap V(I)} \left(\boldsymbol{h}_v^{(t-1)} + \mathsf{MLP}_{\mathrm{vc}}^{(t)}(\boldsymbol{e}_{vc})\right)\right)\Bigg] \\
\boldsymbol{h}_o^{(t)} \coloneqq \mathsf{MLP}_{\mathrm{o}}^{(t)} \Bigg[& \left(1+\epsilon_o^{(t)}\right) \mathsf{MLP}_{o\to o}^{(t)}\left(\boldsymbol{h}_o^{(t-1)}\right) + \\
& \mathsf{MLP}_{\mathrm{c}\to\mathrm{o}}^{(t)}\left(\sum_{c\in C(I)} \left(\boldsymbol{h}_c^{(t)} + \mathsf{MLP}_{\mathrm{co}}^{(t)}(\boldsymbol{e}_{co})\right)\right) + \\
& \mathsf{MLP}_{\mathrm{v}\to\mathrm{o}}^{(t)}\left(\sum_{v\in V(I)} \left(\boldsymbol{h}_v^{(t-1)} + \mathsf{MLP}_{\mathrm{vo}}^{(t)}(\boldsymbol{e}_{vo})\right)\right)\Bigg] \\
\boldsymbol{h}_v^{(t)} \coloneqq \mathsf{MLP}_{\mathrm{v}}^{(t)} \Bigg[& \left(1+\epsilon_v^{(t)}\right) \mathsf{MLP}_{\mathrm{v}\to\mathrm{v}}^{(t)}\left(\boldsymbol{h}_v^{(t-1)}\right) + \\
& \mathsf{MLP}_{\mathrm{o}\to\mathrm{v}}^{(t)}\left(\boldsymbol{h}_o^{(t)} + \mathsf{MLP}_{\mathrm{ov}}^{(t)}(\boldsymbol{e}_{ov})\right) + \\
& \mathsf{MLP}_{\mathrm{c}\to\mathrm{v}}^{(t)}\left(\sum_{c\in N_v\cap C(I)} \left(\boldsymbol{h}_c^{(t)} + \mathsf{MLP}_{\mathrm{cv}}^{(t)}(\boldsymbol{e}_{cv})\right)\right)\Bigg].
\end{aligned}
$$

Anexos

For the GEN layer, we have:

$$
\begin{aligned}
\boldsymbol{h}_c^{(t)} \coloneqq \mathsf{MLP}_{\mathrm{c}}^{(t)} \Bigg[& \mathsf{MLP}_{\mathrm{c}\to\mathrm{c}}^{(t)}\left(\boldsymbol{h}_c^{(t-1)}\right) + \\
& \mathsf{MLP}_{\mathrm{o}\to\mathrm{c}}^{(t)}\left(\boldsymbol{h}_o^{(t-1)} + \mathsf{MLP}_{\mathrm{oc}}^{(t)}(\boldsymbol{e}_{oc}) + \epsilon_{\mathrm{o}\to\mathrm{c}}^{(t)}\right) + \\
& \mathsf{MLP}_{\mathrm{v}\to\mathrm{c}}^{(t)}\left(\mathsf{MSG}\left(\left\{\!\!\left\{\boldsymbol{h}_v^{(t-1)} + \mathsf{MLP}_{\mathrm{vc}}^{(t)}(\boldsymbol{e}_{vc}) + \epsilon_{\mathrm{v}\to\mathrm{c}}^{(t)} \mid v \in N_c \cap V(I)\right\}\!\!\right\}\right)\right) \Bigg] \\
\boldsymbol{h}_o^{(t)} \coloneqq \mathsf{MLP}_{\mathrm{o}}^{(t)} \Bigg[& \mathsf{MLP}_{\mathrm{o}\to\mathrm{o}}^{(t)}\left(\boldsymbol{h}_o^{(t-1)}\right) + \\
& \mathsf{MLP}_{\mathrm{c}\to\mathrm{o}}^{(t)}\left(\mathsf{MSG}\left(\left\{\!\!\left\{\boldsymbol{h}_c^{(t-1)} + \mathsf{MLP}_{\mathrm{co}}^{(t)}(\boldsymbol{e}_{co}) + \epsilon_{\mathrm{c}\to\mathrm{o}}^{(t)} \mid c \in C(I)\right\}\!\!\right\}\right)\right) + \\
& \mathsf{MLP}_{\mathrm{v}\to\mathrm{o}}^{(t)}\left(\mathsf{MSG}\left(\left\{\!\!\left\{\boldsymbol{h}_c^{(t-1)} + \mathsf{MLP}_{\mathrm{vc}}^{(t)}(\boldsymbol{e}_{vc}) + \epsilon_{\mathrm{v}\to\mathrm{o}}^{(t)} \mid v \in V(I)\right\}\!\!\right\}\right)\right) \Bigg] \\
\boldsymbol{h}_v^{(t)} \coloneqq \mathsf{MLP}_{\mathrm{v}}^{(t)} \Bigg[& \mathsf{MLP}_{\mathrm{v}\to\mathrm{v}}^{(t)}\left(\boldsymbol{h}_v^{(t-1)}\right) + \\
& \mathsf{MLP}_{\mathrm{o}\to\mathrm{v}}^{(t)}\left(\boldsymbol{h}_o^{(t)} + \mathsf{MLP}_{\mathrm{ov}}^{(t)}(\boldsymbol{e}_{ov}) + \epsilon_{\mathrm{o}\to\mathrm{v}}^{(t)}\right) +
\end{aligned}
$$

Anexos

$$\mathsf{MLP}^{(t)}_{\mathrm{c}\to\mathrm{v}}\Big(\mathsf{MSG}\Big(\Big\{\!\!\Big\{\boldsymbol{h}_c^{(t-1)} + \mathsf{MLP}^{(t)}_{\mathrm{cv}}(\boldsymbol{e}_{cv}) + \epsilon^{(t)}_{\mathrm{c}\to\mathrm{v}} \mid c \in N_v \cap C(I)\Big\}\!\!\Big\}\Big)\Big)\Big],$$

where MSG is the softmax aggregation with $\tau = 1$, i.e.,

$$\mathsf{softmax}(\mathcal{X} \mid \tau) = \sum_{\boldsymbol{x}_i \in \mathcal{X}} \frac{\exp(\tau \cdot \boldsymbol{x}_i)}{\sum_{\boldsymbol{x}_j \in \mathcal{X}} \exp(\tau \cdot \boldsymbol{x}_j)} \cdot \boldsymbol{x}_i.$$

Anexos

D Details of datasets

In the following, we describe our datasets.

D.1 Combinatorial optimization problems

In the following, we briefly the combinatorial optimization problems.

Set cover problem The set cover problem aims cover the universe U with a collection of given subsets $S = \{S_1, S_2, \ldots, S_m\}$ satisfying $\cup_{i=1}^{m} S_i = U$, with the target of minimizing the objective function. Formally, let x_i be the variable deciding whether subset S_i is selected, and c_i the cost per subset, we have:

$$\begin{aligned} &\min_{\boldsymbol{x}} \sum_{i=1}^{m} c_i x_i \\ &\text{s.t.} \sum_{i:u \in S_i} x_i \geq 1, \forall u \in U \\ &\quad x_i \in \{0,1\}, \forall i \in [m]. \end{aligned}$$

Anexos

Maximal independent set Given an undirected graph G with node set $V(G)$ and edge set $E(G)$, the goal of the maximal independent set problem is to find a set of nodes where no pairs of them are connected. If we use x_i to denote a node i is selected or not, we have:

$$
\begin{aligned}
&\max_{\boldsymbol{x}} \sum_{i \in V} x_i \\
&\text{s.t. } x_u + x_v \leq 1, \forall (u,v) \in E(G), u, v \in V(G) \\
&\quad x_i \in \{0,1\}, \forall i \in V(G).
\end{aligned}
$$

Anexos

Combinatorial auction Suppose there are a set of items M and one of bidders N. Each bidder $i \in N$ maintains a set of bids B_i, each bid $b \in B_i$ is associated with a subset $S_{ib} \subseteq M$ of items and a value v_{ib} that the bidder i is willing to pay for this subset. The binary decision variable x_{ib} is 1 if the bid b by bidder i is accepted or 0 otherwise. The MILP formulation of the problem is as follows:

$$\begin{aligned} &\max_{\boldsymbol{x}} \sum_{i \in N} \sum_{b \in B_i} v_{ib} x_{ib} \\ &\text{s.t.} \sum_{i \in N} \sum_{b \in B_i : j \in S_{ib}} x_{ib} \leq 1, \forall j \in M, \\ &\quad x_{ib} \in \{0,1\}, \forall i \in N, b \in B_i. \end{aligned} \tag{15}$$

Anexos

Capacitated facility location Given a set of facilities M and another of customers N, we aim to build facilities and satisfy the demand of the customers at minimum cost. Let $y_j, j \in M$ be the binary decision of building the facility j or not, and x_{ij} be a continuous variable indicating the fraction of demand facility j sends to customer $i \in N$. Let $d_i \in \mathbb{R}^+$ be the amount of demand of customer i, and v_j be the volume of the facility j, c_{ij} the cost of shipment and f_j the cost of building facility j, we formulate the problem as follows:

$$
\begin{aligned}
\min_{\boldsymbol{x}} & \sum_{j \in M} f_j y_j + \sum_{i \in N} \sum_{j \in M} c_{ij} x_{ij} \\
\text{s.t.} & \sum_{j \in M} x_{ij} = 1, \forall i \in N \\
& \sum_{i \in N} d_i x_{ij} \leq v_j y_j, \forall i \in N, j \in M \\
& y_j \in \{0,1\}, x_{ij} \in [0,1], \forall i \in N, j \in M.
\end{aligned}
\tag{16}
$$

Anexos

D.2 Generation of instances

We propose various sizes of generated instances; see Tables 4 to 7 for the size parameters of each dataset. The generation of instances follows the setting of Gasse et al. (2019). For the set covering instances, problems are generated with 15-20 rows and columns and a constraint matrix density of 0.15 for mini instances. For small instances, we used 30-50 rows and 50-70 columns with a density of 0.05. Large instances have 300-500 rows and 500-700 columns with a density of 0.01. We employ the Erdős–R'enyi random graph as the foundational graph when generating maximal independent set instances, designating 10-20 nodes for mini instances, 50-70 nodes for small instances, and 500-700 nodes for large instances. For combinatorial auction instances, we modulate the size by varying the number of items and bids: specifically, 20 items and bids are set for mini instances, 50-80 for small instances, and 500-800 for large instances. Lastly, for the capacitated facility location instances, we set 3-5 customers and facilities for mini instances, 10 for both in small instances, and 20-30 in large ones.

Anexos

E Training parameters

For all the experiments, we train the neural networks with Adam optimizer with default hyperparamters, and run for at most 1000 epochs. During training, we leverage learning rate decay with right to the validation objective gap metric with a decay ratio of 0.5 and patience 50. We terminate the run at patience 100 epochs. Besides, we display the other task-specific hyperparameters in Table 8, which are the batch size, number of MPNN layers as well as the number of sampled IPM solver steps, the step decay factor introduced in Equation (7), the loss weight combination in Equation (10), plus the weight decay of the optimizer.

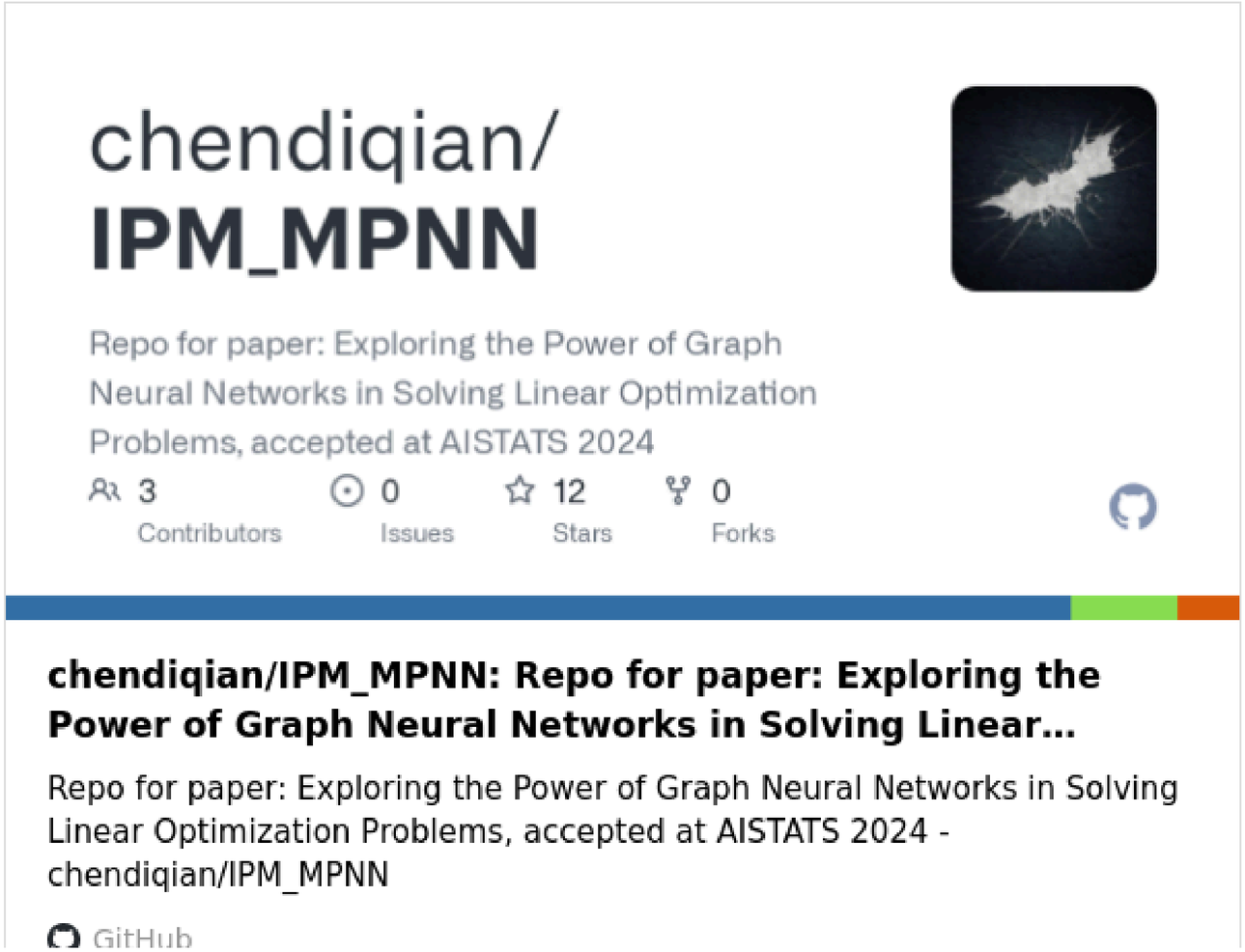
With regard to the bipartiteness ablation study, we also tune the hyperparameters for the sake of fair comparison. The choosen hyperparameters are listed in Table 9. Moreover, the hyperparameter configurations for the ODE approach baseline Wu and Lisser (2023) are shown in Table 10.

Anexos

Table 10: Training hyperparameters of the ODE approach baseline. The experiments are done on the mini-sized instances with GEN-based MPNNs, v^a represents $v \times 10^a$.

Instances	MPNN	Candidate	Batch size	Num. layers	Hidden dim.	α	w_{var}	w_{obj}	w_{cons}	Weight decay
Setcover	GEN	Ours	512	3	128	0.3	1.0	3.5	1.3	3.3^{-3}
		Baseline	8	3	128	-	-	-	-	0.0
	GCN	Ours	512	3	180	0.8	1.0	6.1	1.6	1.0^{-6}
		Baseline	8	3	180	-	-	-	-	0.0
	GIN	Ours	512	3	180	0.4	1.0	3.8	1.4	5.5^{-6}
		Baseline	8	3	180	-	-	-	-	0.0
Indset	GEN	Ours	512	3	128	0.4	1.0	7.1	6.2	1.0^{-6}
		Baseline	8	3	128	-	-	-	-	0.0
	GCN	Ours	512	3	180	0.8	1.0	3.5	5.6	2.1^{-6}
		Baseline	8	3	180	-	-	-	-	0.0
	GIN	Ours	512	3	180	0.4	1.0	5.9	3.9	9.7^{-6}
		Baseline	8	3	180	-	-	-	-	0.0
Cauc	GEN	Ours	512	3	128	0.8	1.0	9.6	7.1	8.0^{-5}
		Baseline	8	3	128	-	-	-	-	0.0
	GCN	Ours	512	3	180	0.7	1.0	4.5	5.0	3.7^{-6}
		Baseline	8	3	180	-	-	-	-	0.0
	GIN	Ours	512	3	128	0.9	1.0	6.4	5.0	1.0^{-6}
		Baseline	8	3	128	-	-	-	-	0.0
Fac	GEN	Ours	512	3	128	0.7	1.0	5.3	0.8	1.7^{-6}
		Baseline	8	3	128	-	-	-	-	0.0
	GCN	Ours	512	3	128	0.8	1.0	2.9	3.7	9.2^{-6}
		Baseline	8	3	128	-	-	-	-	0.0
	GIN	Ours	512	3	180	0.8	1.0	1.8	1.5	3.6^{-7}
		Baseline	8	3	180	-	-	-	-	0.0

Anexos


chendiqian/
IPM_MPNN
Repo for paper: Exploring the Power of Graph Neural Networks in Solving Linear Optimization Problems, accepted at AISTATS 2024
3
Contributors
0
Issues
12
Stars
0
Forks
chendiqian/IPM_MPNN: Repo for paper: Exploring the Power of Graph Neural Networks in Solving Linear...
Repo for paper: Exploring the Power of Graph Neural Networks in Solving Linear Optimization Problems, accepted at AISTATS 2024 - chendiqian/IPM_MPNN
GitHub