One Model, Any CSP: Graph Neural Networks as Fast Global Search Heuristics for Constraint Satisfaction

Jan Tönshoff, Jakob Lindner, Berke Kisin, Martin Grohe

RWTH Aachen

toenshoff@informatik.rwth-aachen.de

October 28, 2022
Neural Combinatorial Optimization

Learn heuristics for combinatorial optimization with Graph Neural Networks:
Neural Combinatorial Optimization

Learn heuristics for combinatorial optimization with Graph Neural Networks:

- **Pros**
  - Learn novel algorithms from scratch.
Neural Combinatorial Optimization

Learn heuristics for combinatorial optimization with Graph Neural Networks:

Pros

- Learn novel algorithms from scratch.
- Data-driven fine-tuning for specific input distributions.

Graph Neural Network
Neural Combinatorial Optimization

Learn heuristics for combinatorial optimization with Graph Neural Networks:

Pros
- Learn novel algorithms from scratch.
- Data-driven fine-tuning for specific input distributions.

Cons
- Computationally expensive.
Neural Combinatorial Optimization

Learn heuristics for combinatorial optimization with Graph Neural Networks:

Pros
- Learn novel algorithms from scratch.
- Data-driven fine-tuning for specific input distributions.

Cons
- Computationally expensive.
- Problem specific graph reductions, architectures and training procedures.
Constraint Satisfaction Problems
Constraint Satisfaction Problems

CSP-Instance: $\mathcal{I} = (\mathcal{X}, \mathcal{C}, \mathcal{D})$
Constraint Satisfaction Problems

CSP-Instance: \( \mathcal{I} = (\mathcal{X}, C, D) \)

- Variables \( \mathcal{X} = \{X_1, \ldots, X_n\} \) with finite domains \( D = \{D(X_1), \ldots, D(X_n)\} \)
Constraint Satisfaction Problems

CSP-Instance: $\mathcal{I} = (\mathcal{X}, \mathcal{C}, \mathcal{D})$

- Variables $\mathcal{X} = \{X_1, \ldots, X_n\}$ with finite domains $\mathcal{D} = \{D(X_1), \ldots, D(X_n)\}$
- Constraints $\mathcal{C} = \{C_1, \ldots, C_m\}$ of the form $C = (s^C, R^C)$:
Constraint Satisfaction Problems

CSP-Instance: $\mathcal{I} = (\mathcal{X}, \mathcal{C}, \mathcal{D})$

- Variables $\mathcal{X} = \{X_1, \ldots, X_n\}$ with finite domains $\mathcal{D} = \{\mathcal{D}(X_1), \ldots, \mathcal{D}(X_n)\}$
- Constraints $\mathcal{C} = \{C_1, \ldots, C_m\}$ of the form $C = (s^C, R^C)$:
  
  $s^C = (X_1^C, \ldots, X_\ell^C)$
  
  $R^C \subseteq \mathcal{D}(X_1^C) \times \cdots \times \mathcal{D}(X_\ell^C)$
Constraint Satisfaction Problems

CSP-Instance: $\mathcal{I} = (\mathcal{X}, \mathcal{C}, \mathcal{D})$

- Variables $\mathcal{X} = \{X_1, \ldots, X_n\}$ with finite domains $\mathcal{D} = \{\mathcal{D}(X_1), \ldots, \mathcal{D}(X_n)\}$
- Constraints $\mathcal{C} = \{C_1, \ldots, C_m\}$ of the form $C = (s^C, R^C)$:
  $$s^C = (X_1^C, \ldots, X_\ell^C)$$
  $$R^C \subseteq \mathcal{D}(X_1^C) \times \cdots \times \mathcal{D}(X_\ell^C)$$

Variable assignment $\alpha$: $\alpha(X) \in \mathcal{D}(X)$
Constraint Satisfaction Problems

CSP-Instance: $\mathcal{I} = (\mathcal{X}, \mathcal{C}, \mathcal{D})$

- Variables $\mathcal{X} = \{X_1, \ldots, X_n\}$ with finite domains $\mathcal{D} = \{\mathcal{D}(X_1), \ldots, \mathcal{D}(X_n)\}$
- Constraints $\mathcal{C} = \{C_1, \ldots, C_m\}$ of the form $C = (s^C, R^C)$:
  $$s^C = (X_1^C, \ldots, X_\ell^C)$$
  $$R^C \subseteq \mathcal{D}(X_1^C) \times \cdots \times \mathcal{D}(X_\ell^C)$$

Variable assignment $\alpha$: $\alpha(X) \in \mathcal{D}(X)$

$$\alpha \models C \iff (\alpha(X_1^C), \ldots, \alpha(X_\ell^C)) \in R^C$$
Constraint Satisfaction Problems

Quality of assignment $\alpha$ for instance $I = (X, C, D)$:

$$Q_I(\alpha) = \frac{|\{C \in C : \alpha \models C\}|}{|C|}$$
Constraint Satisfaction Problems

Quality of assignment $\alpha$ for instance $\mathcal{I} = (\mathcal{X}, \mathcal{C}, \mathcal{D})$:

$$Q_{\mathcal{I}}(\alpha) = \frac{|\{C \in \mathcal{C} : \alpha \models C\}|}{|\mathcal{C}|}$$

Decision problem for $\mathcal{I}$:

$$\exists \alpha : Q_{\mathcal{I}}(\alpha) = 1?$$
Constraint Satisfaction Problems

Quality of assignment $\alpha$ for instance $\mathcal{I} = (\mathcal{X}, \mathcal{C}, \mathcal{D})$:

$$Q_{\mathcal{I}}(\alpha) = \frac{|\{C \in \mathcal{C} : \alpha \models C\}|}{|\mathcal{C}|}$$

Decision problem for $\mathcal{I}$:

$$\exists \alpha : Q_{\mathcal{I}}(\alpha) = 1?$$

Maximization problem for $\mathcal{I}$:

$$\alpha^* = \arg\max_{\alpha} Q_{\mathcal{I}}(\alpha)$$
Constraint Satisfaction Problems

Boolean SAT formula $f = (X_1 \lor X_2) \land (\neg X_1 \lor X_3 \lor X_2)$. 
Constraint Satisfaction Problems

Boolean SAT formula \( f = (X_1 \lor X_2) \land (\neg X_1 \lor X_3 \lor X_2). \)

Equivalent CSP \( \mathcal{I} = (\mathcal{X}, \mathcal{C}, \mathcal{D}) \):

\[
\mathcal{X} = \{X_1, X_2, X_3\}, \quad \mathcal{D}(X_i) = \{0, 1\}
\]
Constraint Satisfaction Problems

Boolean SAT formula \( f = (X_1 \lor X_2) \land (\neg X_1 \lor X_3 \lor X_2). \)

Equivalent CSP \( \mathcal{I} = (\mathcal{X}, \mathcal{C}, \mathcal{D}): \)

\[
\mathcal{X} = \{X_1, X_2, X_3\}, \quad \mathcal{D}(X_i) = \{0, 1\}
\]

\[
C_1 = ((X_1, X_2), \{0, 1\}^2 \setminus (0, 0))
\]
Constraint Satisfaction Problems

Boolean SAT formula $f = (X_1 \lor X_2) \land (\neg X_1 \lor X_3 \lor X_2)$.

Equivalent CSP $\mathcal{I} = (\mathcal{X}, \mathcal{C}, \mathcal{D})$:

\[
\mathcal{X} = \{X_1, X_2, X_3\}, \quad \mathcal{D}(X_i) = \{0, 1\}
\]

\[
\mathcal{C}_1 = ((X_1, X_2), \{0, 1\}^2 \setminus (0, 0))
\]

\[
\mathcal{C}_2 = ((X_1, X_3, X_2), \{0, 1\}^3 \setminus (1, 0, 0))
\]
Constraint Satisfaction Problems

Graph Colouring instance \((G, k)\) with \(G = (V, E)\) and \(k\) colours.
Constraint Satisfaction Problems

Graph Colouring instance \((G, k)\) with \(G = (V, E)\) and \(k\) colours.

Equivalent CSP \(\mathcal{I} = (\mathcal{X}, \mathcal{C}, \mathcal{D})\):

\[
\mathcal{X} = V, \quad \mathcal{D}(X) = \{1, \ldots, k\}
\]
Constraint Satisfaction Problems

Graph Colouring instance \((G, k)\) with \(G = (V, E)\) and \(k\) colours.
Equivalent CSP \(\mathcal{I} = (\mathcal{X}, \mathcal{C}, \mathcal{D})\):

\[
\mathcal{X} = V, \quad \mathcal{D}(X) = \{1, \ldots, k\} \\
\mathcal{C} = \{((u, v), R \neq) : \forall uv \in E\}
\]
RLSAT

Learning local search for SAT with GNNs (Yolcu and Póczos, 2019):
RLSAT

Learning local search for SAT with GNNs (Yolcu and Póczos, 2019):

\[(X_1 \lor X_2) \land (\neg X_1 \lor X_3 \lor X_2)\]

\[(X_1, X_2, X_3) \mapsto (1, 0, 0)\]

Instance + Assignment
RLSAT

Learning local search for SAT with GNNs (Yolcu and Póczos, 2019):

\[(X_1 \lor X_2) \land (\neg X_1 \lor X_3 \lor X_2)\]

\[(X_1, X_2, X_3) \mapsto (1, 0, 0)\]
RLSAT

Learning local search for SAT with GNNs (Yolcu and Póczos, 2019):

\[(X_1 \lor X_2) \land (\neg X_1 \lor X_3 \lor X_2)\]

\[(X_1, X_2, X_3) \rightarrow (1, 0, 0)\]
Related Work

Overview: (Cappart et al., 2021)

SAT:
- RLSAT (Yolcu and Póczos, 2019)
- PDP (Amizadeh et al., 2019)

MaxCut:
- S2V (Khalil et al., 2017)
- ECO-DQN (Barrett et al., 2020)
- ECORD (Barrett et al., 2022)

Binary CSPs:
- RUNCSP (Tönshoff et al., 2021)
ANYCSP

ANYCSP: Are Neural Networks great heuristics? Yes, for CSPs!
ANYCSP: Are Neural Networks great heuristics? Yes, for CSPs!

Objectives:

- Design unified graph representation and GNN architecture for all CSPs.
ANYCSP: Are Neural Networks great heuristics? Yes, for CSPs!

Objectives:
- Design unified graph representation and GNN architecture for all CSPs.
- No restrictions to domain size or relations.
ANYCSP

ANYCSP: Are Neural Networks great heuristics? Yes, for CSPs!

Objectives:

• Design unified graph representation and GNN architecture for all CSPs.
• No restrictions to domain size or relations.
• Trained unsupervised with reinforcement learning.
ANYCSP: Are Neural Networks great heuristics? Yes, for CSPs!

Objectives:

- Design unified graph representation and GNN architecture for all CSPs.
- No restrictions to domain size or relations.
- Trained unsupervised with reinforcement learning.
- Utilizes a global search action space.
CSP Instance $\mathcal{I}$:

$\mathcal{X} = \{X, Y, Z\}$

$D_X = \{1, 2, 3\}$
$D_Y = \{1, 2\}$
$D_Z = \{1, 2\}$

$C_1 : X \leq Y$
$C_2 : Y \neq Z$
Constraint Value Graph

CSP Instance $\mathcal{I}$:

$\mathcal{X} = \{X, Y, Z\}$

$D_X = \{1, 2, 3\}$
$D_Y = \{1, 2\}$
$D_Z = \{1, 2\}$

$C_1 : X \leq Y$
$C_2 : Y \neq Z$

Assignment $\alpha = (2, 1, 2)$
Constraint Value Graph

CSP Instance $\mathcal{I}$:

$\mathcal{X} = \{X, Y, Z\}$

$D_X = \{1, 2, 3\}$
$D_Y = \{1, 2\}$
$D_Z = \{1, 2\}$

$C_1 : X \leq Y$
$C_2 : Y \neq Z$

Assignment $\alpha = (2, 1, 2)$
CSP Instance $\mathcal{I}$:

$\mathcal{X} = \{X, Y, Z\}$

$D_X = \{1, 2, 3\}$

$D_Y = \{1, 2\}$

$D_Z = \{1, 2\}$

$C_1 : X \leq Y$

$C_2 : Y \neq Z$

Assignment $\alpha = (2, 1, 2)$

$G(\mathcal{I}, \alpha) = (V, E, L_D, L_C)$
CSP Instance $\mathcal{I}$:

$\mathcal{X} = \{X, Y, Z\}$

$D_X = \{1, 2, 3\}$

$D_Y = \{1, 2\}$

$D_Z = \{1, 2\}$

$C_1 : X \leq Y$

$C_2 : Y \neq Z$

Assignment $\alpha = (2, 1, 2)$

$G(\mathcal{I}, \alpha) = (V, E, L_D, L_C)$
Constraint Value Graph

CSP Instance $\mathcal{I}$:

$\mathcal{X} = \{X, Y, Z\}$

$D_X = \{1, 2, 3\}$

$D_Y = \{1, 2\}$

$D_Z = \{1, 2\}$

$C_1 : X \leq Y$

$C_2 : Y \neq Z$

Assignment $\alpha = (2, 1, 2)$

$G(\mathcal{I}, \alpha) = (V, E, L_D, L_C)$

$L_C(v, C) = 1 \iff \alpha[X_v = v] \models C$
Constraint Value Graph

CSP Instance $\mathcal{I}$:

$\mathcal{X} = \{X, Y, Z\}$

$D_X = \{1, 2, 3\}$

$D_Y = \{1, 2\}$

$D_Z = \{1, 2\}$

$C_1 : X \leq Y$

$C_2 : Y \neq Z$

Assignment $\alpha^{(1)} = (2, 2, 2)$

$G(\mathcal{I}, \alpha^{(1)}) = (V, E, L_D, L_C)$

$L_C(v, C) = 1 \iff \alpha[X_v = v] \models C$
CSP Instance $\mathcal{I}$:

$\mathcal{X} = \{X, Y, Z\}$

$D_X = \{1, 2, 3\}$

$D_Y = \{1, 2\}$

$D_Z = \{1, 2\}$

$C_1 : X \leq Y$

$C_2 : Y \neq Z$

Assignment $\alpha^{(2)} = (2, 2, 1)$

Constraint Value Graph

$G(\mathcal{I}, \alpha^{(2)}) = (V, E, L_D, L_C)$

$L_C(v, C) = 1 \iff \alpha[X_v = v] \models C$
Policy GNN

Our GNN $\pi_{\theta}$ is a trainable stochastic global search policy:

- **Input:** $G(I, \alpha(t)), h(t) : \mathcal{D} \rightarrow \mathbb{R}^d$.
- **Output:** Soft assignment $\phi(t+1) : \mathcal{D} \rightarrow [0, 1]$.
- **Next assignment:** $\alpha(t+1) \sim \phi(t+1)$.
Our GNN $\pi_\theta$ is a trainable stochastic global search policy:
Our GNN $\pi_\theta$ is a trainable stochastic global search policy:

- **Input:** $G(I, \alpha(t))$, recurrent states $h(t) : \mathcal{D} \rightarrow \mathbb{R}^d$. 

---

**Policy GNN**
Our GNN $\pi_\theta$ is a trainable stochastic global search policy:

- **Input**: $G(I, \alpha(t))$, recurrent states $h(t) : \dot{\mathcal{D}} \rightarrow \mathbb{R}^d$.
- **Output**: Soft assignment $\varphi^{(t+1)} : \dot{\mathcal{D}} \rightarrow [0, 1]$
Policy GNN

Our GNN $\pi_\theta$ is a trainable stochastic global search policy:

- **Input**: $G(I, \alpha(t))$, recurrent states $h(t) : \hat{D} \rightarrow \mathbb{R}^d$.
- **Output**: Soft assignment $\varphi(t+1) : \hat{D} \rightarrow [0, 1]$
- **Next assignment**: $\alpha(t+1) \sim \varphi(t+1)$
Stochastic Global Search

Input: CSP instance $\mathcal{I}$, number of iterations $T$. 
Stochastic Global Search

Input: CSP instance $\mathcal{I}$, number of iterations $T$.

$\alpha^{(0)}$

$h^{(0)}$
Stochastic Global Search

Input: CSP instance $\mathcal{I}$, number of iterations $T$.

$$\alpha^{(0)} \rightarrow G(\mathcal{I}, \alpha^{(0)})$$

$$h^{(0)}$$
Stochastic Global Search

Input: CSP instance $\mathcal{I}$, number of iterations $T$.

\[
\begin{align*}
\alpha^{(0)} & \rightarrow G(\mathcal{I}, \alpha^{(0)}) \\
 h^{(0)} & \rightarrow \text{GNN } \pi_{\theta} \\
 \phi^{(1)} & \rightarrow \alpha^{(1)} \\
 & \rightarrow \alpha^{(2)} \\
 & \rightarrow \alpha^{(3)} \\
 & \rightarrow \text{Output: Sequence of assignments } \alpha^{(1)}, \ldots, \alpha^{(T)}.
\end{align*}
\]
Stochastic Global Search

Input: CSP instance $\mathcal{I}$, number of iterations $T$.

\[ \alpha^{(0)} \rightarrow G(\mathcal{I}, \alpha^{(0)}) \]

\[ h^{(0)} \rightarrow \text{GNN } \pi_{\theta} \]

\[ \varphi^{(1)} \]

\[ \zeta \]

\[ \alpha^{(1)} \]
Stochastic Global Search

Input: CSP instance $\mathcal{I}$, number of iterations $T$.
Stochastic Global Search

Input: CSP instance $\mathcal{I}$, number of iterations $T$.

\[ \alpha^{(0)} \rightarrow G(\mathcal{I}, \alpha^{(0)}) \rightarrow G(\mathcal{I}, \alpha^{(1)}) \]

\[ h^{(0)} \rightarrow \text{GNN} \pi_\theta \rightarrow \text{GNN} \pi_\theta \]

\[ \varphi^{(1)} \leftarrow \zeta \leftarrow \alpha^{(1)} \]

\[ \varphi^{(2)} \leftarrow \zeta \leftarrow \alpha^{(2)} \]

Output: Sequence of assignments $\alpha = \alpha^{(1)}, \ldots, \alpha^{(T)}$. 
Stochastic Global Search

Input: CSP instance $\mathcal{I}$, number of iterations $T$.

\[ \alpha^{(0)} \xrightarrow{G(\mathcal{I}, \alpha^{(0)})} \pi_\theta \xrightarrow{\varphi^{(1)}} \zeta \]

\[ h^{(0)} \xrightarrow{\text{GNN} \pi_\theta} G(\mathcal{I}, \alpha^{(0)}) \]

\[ \alpha^{(1)} \xrightarrow{G(\mathcal{I}, \alpha^{(1)})} \pi_\theta \xrightarrow{\varphi^{(2)}} \zeta \]

\[ \zeta \xrightarrow{\varphi^{(3)}} \alpha^{(3)} \]

Output: Sequence of assignments $\alpha = \alpha^{(1)}, \ldots, \alpha^{(T)}$. 
Stochastic Global Search

Input: CSP instance $\mathcal{I}$, number of iterations $T$.

Output: Sequence of assignments $\alpha = \alpha^{(1)}, \ldots, \alpha^{(T)}$. 
Stochastic Global Search

Input: CSP instance $\mathcal{I}$, number of iterations $T$.

Output: Sequence of assignments $\alpha = \alpha(1), \ldots, \alpha(T)$.
Rewarding Improvements
Rewarding Improvements

Highest quality achieved before iteration $t$:

$$q^{(t)} = \max_{0 \leq t' < t} Q_I(\alpha^{(t')})$$

(1)
Rewarding Improvements

Highest quality achieved before iteration $t$:

$$q(t) = \max_{0 \leq t' < t} Q_I(\alpha(t'))$$  \hspace{1cm} (1)

Reward in iteration $t$:

$$r(t) = \begin{cases} 
\end{cases}$$  \hspace{1cm} (2)
Rewarding Improvements

Highest quality achieved before iteration $t$:

$$q(t) = \max_{0 \leq t' < t} Q_I(\alpha(t'))$$  \hspace{1cm} (1)

Reward in iteration $t$:

$$r(t) = \begin{cases} 0 & \text{if } Q_I(\alpha(t)) \leq q(t) \end{cases}$$  \hspace{1cm} (2)
Rewarding Improvements

Highest quality achieved before iteration $t$:

$$q(t) = \max_{0 \leq t' < t} \; Q_I(\alpha(t'))$$

(1)

Reward in iteration $t$:

$$r(t) = \begin{cases} 
0 & \text{if } Q_I(\alpha(t)) \leq q(t) \\
Q_I(\alpha(t)) - q(t) & \text{if } Q_I(\alpha(t)) > q(t) 
\end{cases}$$

(2)
Rewarding Improvements

Highest quality achieved before iteration $t$:

$$q(t) = \max_{0 \leq t' < t} Q_I(\alpha(t'))$$

(1)

Reward in iteration $t$:

$$r(t) = \begin{cases} 
0 & \text{if } Q_I(\alpha(t)) \leq q(t) \\
Q_I(\alpha(t)) - q(t) & \text{if } Q_I(\alpha(t)) > q(t) 
\end{cases}$$

(2)

For the total reward after $T$ iterations we observe:

$$\sum_{t=1}^{T} r(t) = q(T+1) - Q_I(\alpha(0))$$

(3)
Training

Assume training distribution of CSP instances $\Omega$. 

\begin{align*}
\text{Objective:} & \quad \theta^* = \arg \max_{\theta} \mathbb{E}_{I \sim \Omega} \alpha \sim \pi(\theta)(I) \\
& \quad h_T X_t = 1 \lambda_t - 1 r(t) \\
& \quad L(I, \alpha, \phi(\theta)) = -T \sum_{t=1}^{X_T} \log P(\alpha(t) | \phi(t) \theta), \\
& \quad G_t = T \sum_{k=t}^{X_T} \lambda_k - t r(k)
\end{align*}
Training

Assume training distribution of CSP instances $\Omega$. Objective:

\[
\theta^* = \arg \max_\theta \mathbb{E}_{\mathcal{I} \sim \Omega, \alpha \sim \pi_\theta(I)} \left[ \sum_{t=1}^{T} \lambda^{t-1} r(t) \right]
\]  

(4)
Training

Assume training distribution of CSP instances Ω. Objective:

\[
\theta^* = \arg \max_{\theta} \mathbb{E}_{\mathcal{I} \sim \Omega, \alpha \sim \pi_\theta(\mathcal{I})} \left[ \sum_{t=1}^{T} \lambda^{t-1} r(t) \right]
\] (4)

We use REINFORCE (Williams, 1992) to learn network parameters θ with SGD:

\[
\mathcal{L}(\mathcal{I}, \alpha, \varphi_\theta) = -\sum_{t=1}^{T} G_t \log P(\alpha^{(t)} | \varphi_\theta^{(t)}), \quad G_t = \sum_{k=t}^{T} \lambda^{k-t} r^{(k)}
\] (5)
Training

Note that our action space $A = \mathcal{D}(X_1) \times \cdots \times \mathcal{D}(X_n)$ is exponentially large!
Training

Note that our action space $A = \mathcal{D}(X_1) \times \cdots \times \mathcal{D}(X_n)$ is exponentially large!

However, we only use efficient parallelizable operations:

$$\alpha^{(t)} \sim \varphi^{(t)}$$

(6)

$$\log P(\alpha^{(t)} | \varphi^{(t)}) = \sum_X \log \varphi^{(t)}(\alpha^{(t)}(X))$$

(7)
Training

Note that our action space $A = \mathcal{D}(X_1) \times \cdots \times \mathcal{D}(X_n)$ is exponentially large!

However, we only use efficient parallelizable operations:

$$\alpha^{(t)} \sim \varphi^{(t)}$$  \hspace{1cm} (6)

$$\log P(\alpha^{(t)}|\varphi^{(t)}) = \sum_{X} \log \varphi^{(t)}(\alpha^{(t)}(X))$$  \hspace{1cm} (7)

Policy Gradient methods allow us to handle very large action spaces.
Experiments
Experiments

We train heuristics for the following CSPs:

- Model RB ($\Omega_{RB}$)
- Graph Coloring ($\Omega_{COL}$)
- MAXCUT ($\Omega_{MCUT}$)
- 3-SAT ($\Omega_{3SAT}$)
- MAX-$k$-SAT ($\Omega_{MSAT}$)
Experiments

We train heuristics for the following CSPs:

- Model RB ($\Omega_{RB}$)
- Graph Coloring ($\Omega_{COL}$)
- MaxCut ($\Omega_{MCUT}$)
- 3-SAT ($\Omega_{3SAT}$)
- Max-$k$-SAT ($\Omega_{MSAT}$)

Training Setup:

- Generated random instances with $|\mathcal{X}| \leq 100$. 
We train heuristics for the following CSPs:

- Model RB ($\Omega_{RB}$)
- Graph Coloring ($\Omega_{COL}$)
- MaxCut ($\Omega_{MCUT}$)
- 3-SAT ($\Omega_{3SAT}$)
- Max-$k$-SAT ($\Omega_{MSAT}$)

Training Setup:

- Generated random instances with $|X| \leq 100$.
- $T_{train} = 40$ search iterations.
Experiments

We train heuristics for the following CSPs:

- Model RB ($\Omega_{RB}$)
- Graph Coloring ($\Omega_{COL}$)
- MaxCut ($\Omega_{MCUT}$)
- 3-SAT ($\Omega_{3SAT}$)
- Max-$k$-SAT ($\Omega_{MSAT}$)

Training Setup:

- Generated random instances with $|\mathcal{X}| \leq 100$.
- $T_{\text{train}} = 40$ search iterations.
- Train for 500K steps of SGD ($\sim 48h$) and batch size 25.
Model RB Benchmarks

![Graph showing model RB benchmarks]

- ANYCSP
- Picat
- ACE
- CoSoCo

The graph compares the number of solved instances over runtime in seconds for different models.

**Graph Details:**
- **X-axis:** Runtime in seconds
- **Y-axis:** Number of solved instances
- **Legend:**
  - ANYCSP
  - Picat
  - ACE
  - CoSoCo
# MaxCut

Instances: (Unweighted) GSet graphs

Metric: Mean absolute deviation from known optimum cut value.

| Method     | $|V|=800$ | $|V|=1K$ | $|V|=2K$ | $|V|\geq3K$ |
|------------|----------|----------|----------|-------------|
| Greedy     | 411.44   | 359.11   | 737.00   | 774.25      |
| SDP        | 245.44   | 229.22   | -        | -           |
| RUNCSP     | 185.89   | 156.56   | 357.33   | 401.00      |
| ECO-DQN    | 65.11    | 54.67    | 157.00   | 428.25      |
| ECORD      | 8.67     | 8.78     | 39.22    | 187.75      |
| ANYCSP     | **1.22** | **2.44** | **13.11**| **51.63**   |
Graph Colouring

Instances: Structured $k$-Colouring instances ($4 \leq k \leq 73$, $|V| \leq 2000$, $|E| \leq 20000$)

Metric: Number of optimally coloured graphs.

<table>
<thead>
<tr>
<th>Method</th>
<th>$\text{COL}_{&lt;10}$</th>
<th>$\text{COL}_{\geq10}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>RUNCSP</td>
<td>33</td>
<td>-</td>
</tr>
<tr>
<td>CoSoCo</td>
<td>49</td>
<td>33</td>
</tr>
<tr>
<td>Picat</td>
<td>49</td>
<td>38</td>
</tr>
<tr>
<td>Greedy</td>
<td>16</td>
<td>15</td>
</tr>
<tr>
<td>DSatur</td>
<td>38</td>
<td>28</td>
</tr>
<tr>
<td>HybridEA</td>
<td>50</td>
<td>40</td>
</tr>
<tr>
<td>ANYCSP</td>
<td>50</td>
<td>40</td>
</tr>
</tbody>
</table>
Graph Colouring

t = 0
Graph Colouring

\[ t = 1 \]
Graph Colouring

\[ t = 2 \]
Graph Colouring

$t = 3$
Graph Colouring

$t = 4$
Graph Colouring

\[ t = 5 \]
Graph Colouring

t = 6
Graph Colouring

t = 7
Graph Colouring

\[ t = 8 \]
SAT

Instances: Random 3SAT Instances from SATLIB.

Metric: Number of satisfied instances.

<table>
<thead>
<tr>
<th>Method</th>
<th>SL50</th>
<th>SL100</th>
<th>SL150</th>
<th>SL200</th>
<th>SL250</th>
</tr>
</thead>
<tbody>
<tr>
<td>RLSAT</td>
<td>100</td>
<td>87</td>
<td>67</td>
<td>27</td>
<td>12</td>
</tr>
<tr>
<td>PDP</td>
<td>93</td>
<td>79</td>
<td>72</td>
<td>57</td>
<td>61</td>
</tr>
<tr>
<td>WalkSAT</td>
<td>100</td>
<td>100</td>
<td>97</td>
<td>93</td>
<td>87</td>
</tr>
<tr>
<td>ProbsAT</td>
<td>100</td>
<td>100</td>
<td>97</td>
<td>87</td>
<td>92</td>
</tr>
<tr>
<td>ANYCSP</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>97</td>
<td>99</td>
</tr>
</tbody>
</table>
Max-5-SAT

![Graph showing performance of different algorithms for Max-5-SAT problem](image-url)
Conclusion

ANYCSP:

- Constraint Value Graphs: A generic and compact representation for CSPs.
- REINFORCE applied to exponential action spaces.
Conclusion

ANYCSP:
• Constraint Value Graphs: A generic and compact representation for CSPs.
• REINFORCE applied to exponential action spaces.

Empirical Observations:
• CSP heuristics can be obtained purely through data-driven training.
• GNNs parameterize a powerful and versatile class of global search heuristic.


## Max-\(k\)-SAT

Instances: CNF formulas with 10K variables and 75K-300K clauses.

Metric: Mean number of unsatisfied clauses.

<table>
<thead>
<tr>
<th>Method</th>
<th>3CNF</th>
<th>4CNF</th>
<th>5CNF</th>
</tr>
</thead>
<tbody>
<tr>
<td>WalkSAT</td>
<td>2145.28</td>
<td>1556.68</td>
<td>1685.10</td>
</tr>
<tr>
<td>CCLS</td>
<td>1567.24</td>
<td>1323.14</td>
<td>1315.96</td>
</tr>
<tr>
<td>SATLike</td>
<td>1595.86</td>
<td>1188.56</td>
<td>1152.88</td>
</tr>
<tr>
<td>ANYCSP</td>
<td><strong>1537.46</strong></td>
<td><strong>1126.44</strong></td>
<td><strong>1103.14</strong></td>
</tr>
</tbody>
</table>
Cross-Comparison

Training Distribution $\Omega$ vs Test CSPs:

<table>
<thead>
<tr>
<th></th>
<th>RB50</th>
<th>$\text{COL}_{&lt;10}$</th>
<th>Gset800</th>
<th>SL250</th>
<th>$\text{Max-5-CNF}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\Omega_{\text{RB}}$</td>
<td>42</td>
<td>50</td>
<td>655.56</td>
<td>98</td>
<td>6192.18</td>
</tr>
<tr>
<td>$\Omega_{\text{COL}}$</td>
<td>15</td>
<td>50</td>
<td>868.22</td>
<td>96</td>
<td>5076.16</td>
</tr>
<tr>
<td>$\Omega_{\text{MCUT}}$</td>
<td>0</td>
<td>0</td>
<td>1.22</td>
<td>0</td>
<td>9048.64</td>
</tr>
<tr>
<td>$\Omega_{\text{3SAT}}$</td>
<td>0</td>
<td>19</td>
<td>1213.11</td>
<td>99</td>
<td>5001.72</td>
</tr>
<tr>
<td>$\Omega_{\text{MSAT}}$</td>
<td>0</td>
<td>15</td>
<td>1217.67</td>
<td>66</td>
<td>1103.14</td>
</tr>
</tbody>
</table>
Ablation

![Ablation Graph](image-url)
\( \pi_\theta \): Message Passing Scheme

\begin{align*}
(1) & \quad \begin{array}{c}
X \\downarrow \quad Y
\end{array} \\
C_1 & \quad \begin{array}{c}
1 & 0 & 0 & 1 & 0 \\
0 & 0 & 1 & 0 & 1
\end{array} \\
(2) & \quad \begin{array}{c}
X \quad Y
\end{array} \\
C_1 & \quad \begin{array}{c}
1 & 0 & 0 & 1 & 0 \\
0 & 0 & 1 & 0 & 1
\end{array} \\
(3) & \quad \begin{array}{c}
X \quad Y
\end{array} \\
C_1 & \quad \begin{array}{c}
1 & 0 & 0 & 1 & 0 \\
0 & 0 & 1 & 0 & 1
\end{array} \\
(4) & \quad \begin{array}{c}
X \quad Y
\end{array} \\
C_1 & \quad \begin{array}{c}
1 & 0 & 0 & 1 & 0 \\
0 & 0 & 1 & 0 & 1
\end{array} \\
(5) & \quad \begin{array}{c}
X \quad Y
\end{array} \\
C_1 & \quad \begin{array}{c}
1 & 0 & 0 & 1 & 0 \\
0 & 0 & 1 & 0 & 1
\end{array} \\
(6) & \quad \begin{array}{c}
X \quad Y
\end{array} \\
C_1 & \quad \begin{array}{c}
1 & 0 & 0 & 1 & 0 \\
0 & 0 & 1 & 0 & 1
\end{array} \\
\end{align*}
The formal Markov Decision Process of ANYCSP:

- State at time $t$: $s(t) = (G(I, \alpha(t)), q(t))$, with $q(t) = \max_{t' < t} Q_I(\alpha(t'))$.
- Initial assignment $\alpha(0)$ is drawn uniformly, $q(0) = 0$.
- Action space $A$: Set of all assignments of $I$.
- Transition function: $(s(t), \alpha(t+1)) \mapsto (G(I, \alpha(t+1)), \max\{q(t), Q_I(\alpha(t))\})$.
- Reward: $r(t) = \max\{0, Q_I(\alpha(t)) - q(t)\}$