# Automated Learning from Graph-Structured Data

Quanming Yao[1], Huan Zhao[2], Yongqi Zhang[2]

[1]*Department of Electronic Engineering, Tsinghua University*

[2]*4Paradigm Inc.*

https://quanmingyao.github.io/AutoML.github.io/aaai22-tutorial.html



# Schedule at a Glance

| Time | Event |
|---|---|
| 00-45 minutes | **Part 1**: Automated Graph Neural Network |
| | **Speaker**: Huan Zhao |
| 45-90 minutes | **Part 2**: Hyper-parameter Tuning for Graph-Structured Data |
| | **Speaker**: Quanming Yao |
| 90-135 minutes | **Part 3**: Automated Knowledge Graph Reasoning: from Triplets to Subgraphs |
| | **Speaker**: Yongqi Zhang |
| 135-150 minutes | **Part 4**: Discussion |

Code Repo.

# Hyper-parameter Tuning for Graph-Structured Data

## - An Example with Knowledge Graph

Dr. Quanming Yao

*Assistant professor, EE Tsinghua*

Joint work with 4Paradigm Inc. (Under review)

# Outline

- Background
- A comprehensive understanding of HP in KGE
- An efficient two-stage HP search algorithm
- Experiments
- Key takeaway and future directions

# Outline

- Background
  - What is knowledge graph embedding
  - What is automated machine learning (AutoML)
  - Limitation of existing works
- A comprehensive understanding of HP in KGE
- An efficient two-stage HP search algorithm
- Experiments
- Key takeaway and future directions

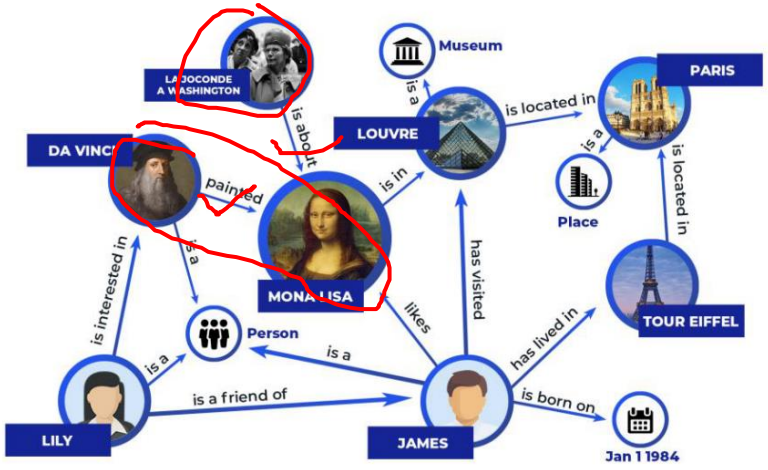
# Background – Knowledge Graph (KG)

## A knowledge graph

- Mainly describe real world entities and relations, organized in a graph
- Allows potentially interacting arbitrary entities with each other

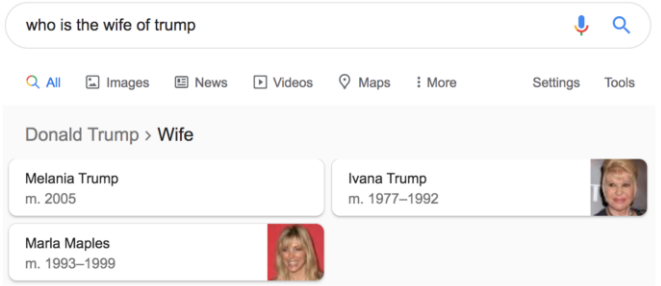
## Preliminaries

- Graph representation: $\mathcal{G} = (\mathcal{E}, \mathcal{R}, \mathcal{F})$
- Entities $\mathcal{E}$
  - real world objects or concepts
- Relations $\mathcal{R}$
  - interactions between entities
- Facts $\mathcal{F}$
  - the basic unit in form of $(s, r, o)$
  - (subject entity, relation, object entity)

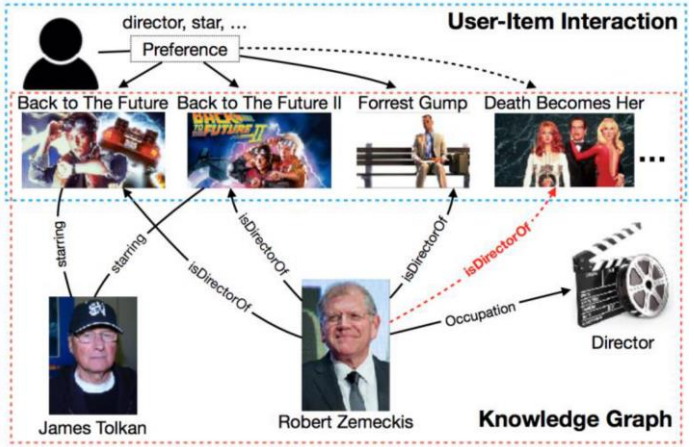



## Applications

KGQA:



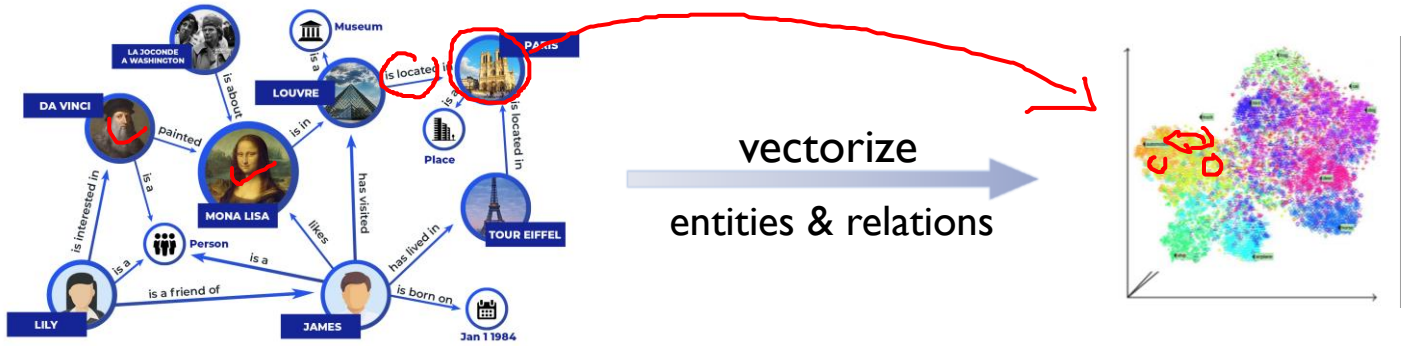

Recommendation:



# Background – Knowledge Graph Embedding (KGE)



- Knowledge Graph Embedding
  - Encode entities and relations in KG into low-dimensional vectors space
  - while capturing nodes' and edges' connection properties




- Most KGE models based on embeddings define a scoring function $F$ to estimate the plausibility of any fact $(s, r, o)$ using their embeddings:

$$F(\boldsymbol{s}, \boldsymbol{r}, \boldsymbol{o}) = \mathrm{p}(\mathbf{o}|\mathbf{s}, \mathbf{r}) \text{ or } \mathrm{p}(\mathbf{s}|\mathbf{o}, \mathbf{r}) \text{ or } \mathrm{p}(\mathbf{r}|\mathbf{s}, \mathbf{o})$$

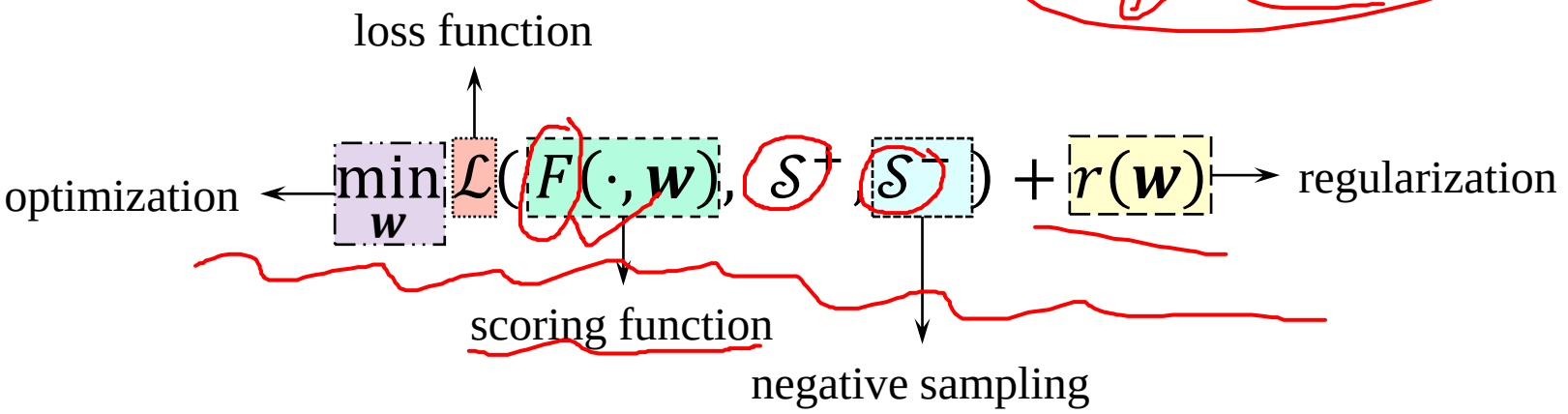
# Machine learning on Knowledge Graph



For setting up a KGE system, we need

- A **scoring function** $F$ to discriminate positive/negative triples
- A sampling scheme to generate **negative samples** $S^-$
- A **loss function** $L$ and **regularization** $r$ to define the learning problem
- An **optimization** strategy for convergence procedure

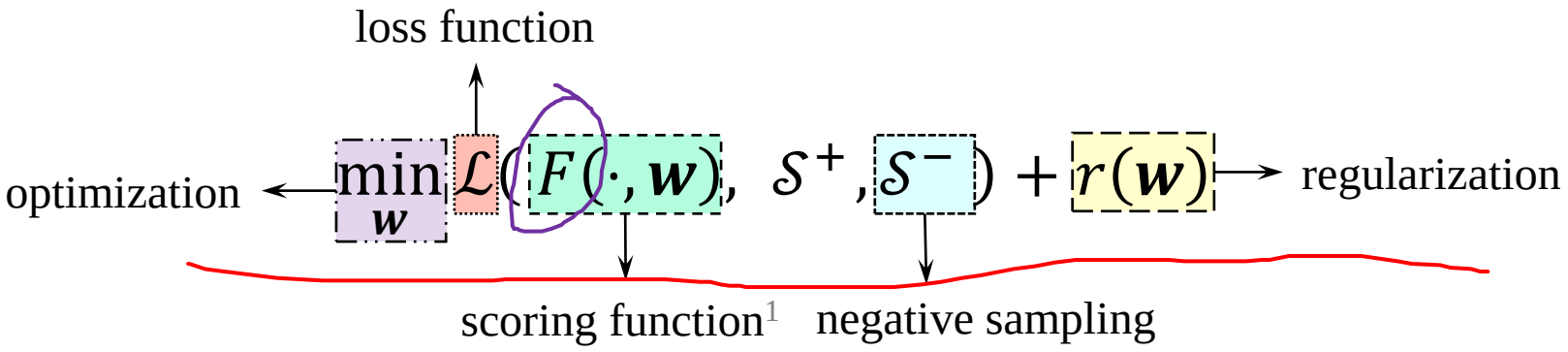
We can formulate the learning framework as:



# Machine learning on Knowledge Graph



## KGE components and related hyper-parameters (HPs)




| component | name | type | range |
|---|---|---|---|
| negative sampling | # negative samples | cat | {32, 128, 512, 2048, 1VsAll, kVsAll} |
| loss function | loss function | cat | {MR, BCE_(mean, sum, adv), CE} |
| | gamma (MR) | float | [1, 24] |
| | adv. weight (BCE_adv) | float | [0.5, 2.0] |
| regularization | regularizer | cat | {FRO, NUC, DURA, None} |
| | reg. weight (not None) | float | $[10^{-12}, 10^{2}]$ |
| | dropout rate | float | [0, 0.5] |
| optimization | optimizer | cat | {Adam, Adagrad, SGD} |
| | learning rate | float | $[10^{-5}, 10^{0}]$ |
| | initializer | cat | {uniform, normal, xavier_uniform, xavier_norm} |
| | batch size | int | {128, 256, 512, 1024} |
| | dimension size | int | {100, 200, 500, 1000, 2000} |
| | inverse relation | bool | {True, False} |

1: Note that HPs in SF are not covered here

# Machine learning on Knowledge Graph

AAAI-2022

## KGE components and related hyper-parameters

**hyper-parameter**

| component | name | type | range |
|---|---|---|---|
| negative sampling | # negative samples | cat | {32, 128, 512, 2048, 1VsAll, kVsAll} |
| loss function | loss function | cat | {MR, BCE_(mean, sum, adv), CE} |
| | gamma (MR) | float | [1, 24] |
| | adv. weight (BCE_adv) | float | [0.5, 2.0] |
| regularization | regularizer | cat | {FRO, NUC, DURA, None} |
| | reg. weight (not None) | float | $[10^{-12}, 10^{2}]$ |
| | dropout rate | float | [0, 0.5] |
| optimization | optimizer | cat | {Adam, Adagrad, SGD} |
| | learning rate | float | $[10^{-5}, 10^{0}]$ |
| | initializer | cat | {uniform, normal, xavier_uniform, xavier_norm} |
| | batch size | int | {128, 256, 512, 1024} |
| | dimension size | int | {100, 200, 500, 1000, 2000} |
| | inverse relation | bool | {True, False} |

**A configuration**

| | |
|---|---|
| # negative samples | 512 |
| loss function | BCE_adv |
| gamma | 0.00 |
| adv. weight | 0.57 |
| regularizer | DURA |
| reg. weight | $8.64 * 10^{-3}$ |
| dropout rate | 0.25 |
| optimizer | Adam |
| learning rate | $1.77 * 10^{-3}$ |
| initializer | xavier_norm |
| batch size | 512 |
| dimension size | 1000 |
| inverse relation | False |

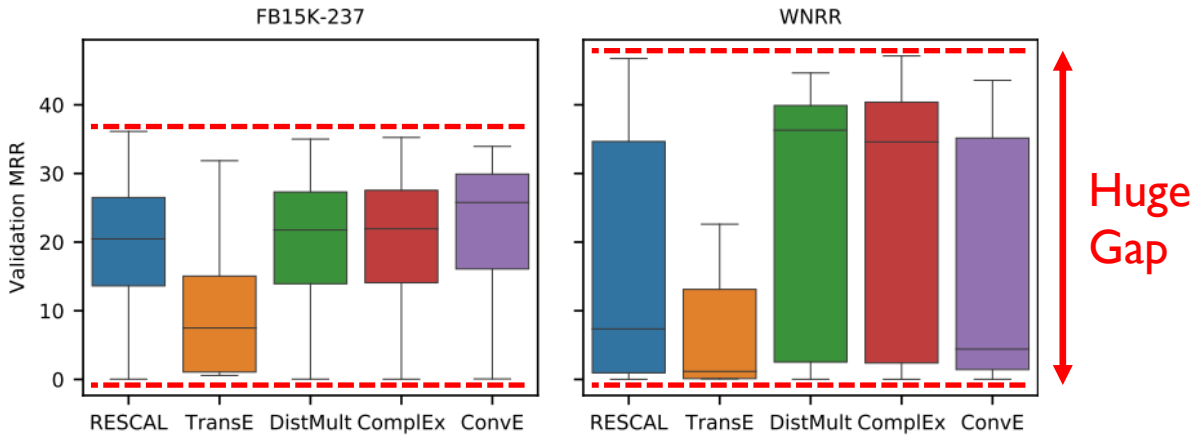
# Review of KGE Models

AAAI-2022

| | | FB15k | | | | WN18 | | | | FB15k-237 | | | | WN18RR | | | | YAGO3-10 | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | H@1 | H@10 | MR | MRR | H@1 | H@10 | MR | MRR | H@1 | H@10 | MR | MRR | H@1 | H@10 | MR | MRR | H@1 | H@10 | MR | MRR |
| Tensor Decomposition Models | DistMult | 73.61 | 86.32 | 173 | 0.784 | 72.60 | 94.61 | 675 | 0.824 | 22.44 | 49.01 | 199 | 0.313 | 39.68 | 50.22 | 5913 | 0.433 | 41.26 | 66.12 | 1107 | 0.501 |
| | ComplEx | 81.56 | 90.53 | 34 | 0.848 | 94.53 | 95.50 | 3623 | 0.949 | 25.72 | 52.97 | 202 | 0.349 | 42.55 | 52.12 | 4907 | 0.458 | 50.48 | 70.35 | 1112 | 0.576 |
| | ANALOGY | 65.59 | 83.74 | 126 | 0.726 | 92.61 | 94.42 | 808 | 0.934 | 12.59 | 35.38 | 476 | 0.202 | 35.82 | 38.00 | 9266 | 0.366 | 19.21 | 45.65 | 2423 | 0.283 |
| | SimplE | 66.13 | 83.63 | 138 | 0.726 | 93.25 | 94.58 | 759 | 0.938 | 10.03 | 34.35 | 651 | 0.179 | 38.27 | 42.65 | 8764 | 0.398 | 35.76 | 63.16 | 2849 | 0.453 |
| | HolE | 75.85 | 86.78 | 211 | 0.800 | 93.11 | 94.94 | 650 | 0.938 | 21.37 | 47.64 | 186 | 0.303 | 40.28 | 48.79 | 8401 | 0.432 | 41.84 | 65.19 | 6489 | 0.502 |
| | TuckER | 72.89 | 88.88 | 39 | 0.788 | 94.64 | 95.80 | 510 | 0.951 | 25.90 | 53.61 | 162 | 0.352 | 42.95 | 51.40 | 6239 | 0.459 | 46.56 | 68.09 | 2417 | 0.544 |
| Geometric Models | TransE | 49.36 | 84.73 | 45 | 0.628 | 40.56 | 94.87 | 279 | 0.646 | 21.72 | 49.65 | 209 | 0.31 | 2.79 | 49.52 | 3936 | 0.206 | 40.57 | 67.39 | 1187 | 0.501 |
| | STransE | 39.77 | 79.60 | 69 | 0.543 | 43.12 | 93.45 | 208 | 0.656 | 22.48 | 49.56 | 357 | 0.315 | 10.13 | 42.21 | 5172 | 0.226 | 3.28 | 7.35 | 5797 | 0.049 |
| | CrossE | 60.08 | 86.23 | 136 | 0.702 | 73.28 | 95.03 | 441 | 0.834 | 21.21 | 47.05 | 227 | 0.298 | 38.07 | 44.99 | 5212 | 0.405 | 33.09 | 65.45 | 3839 | 0.446 |
| | TorusE | 68.85 | 83.98 | 143 | 0.746 | 94.33 | 95.44 | 525 | 0.947 | 19.62 | 44.71 | 211 | 0.281 | 42.68 | 53.35 | 4873 | 0.463 | 27.43 | 47.44 | 19455 | 0.342 |
| | RotatE | 73.93 | 88.10 | 42 | 0.791 | 94.30 | 96.02 | 274 | 0.949 | 23.83 | 53.06 | 178 | 0.336 | 42.60 | 57.35 | 3318 | 0.475 | 40.52 | 67.07 | 1827 | 0.498 |
| Deep Learning Models | ConvE | 59.46 | 84.94 | 51 | 0.688 | 93.89 | 95.68 | 413 | 0.945 | 21.90 | 47.62 | 281 | 0.305 | 38.99 | 50.75 | 4944 | 0.427 | 39.93 | 65.75 | 2429 | 0.488 |
| | ConvKB | 11.44 | 40.83 | 324 | 0.211 | 52.89 | 94.89 | 202 | 0.709 | 13.98 | 41.46 | 309 | 0.230 | 5.63 | 52.50 | 3429 | 0.249 | 32.16 | 60.47 | 1683 | 0.420 |
| | ConvR | 70.57 | 88.55 | 70 | 0.773 | 94.56 | 95.85 | 471 | 0.950 | 25.56 | 52.63 | 251 | 0.346 | 43.73 | 52.68 | 5646 | 0.467 | 44.62 | 67.33 | 2582 | 0.527 |
| | CapsE | 1.93 | 21.78 | 610 | 0.087 | 84.55 | 95.08 | 233 | 0.890 | 7.34 | 35.60 | 405 | 0.160 | 33.69 | 55.98 | 720 | 0.415 | 0.00 | 0.00 | 60676 | 0.000 |
| | RSN | 72.34 | 87.01 | 51 | 0.777 | 91.23 | 95.10 | 346 | 0.928 | 19.84 | 44.44 | 248 | 0.280 | 34.59 | 48.34 | 4210 | 0.395 | 42.65 | 66.43 | 1339 | 0.511 |
| | AnyBURL | 81.09 | 87.86 | 288 | 0.835 | 94.63 | 95.96 | 233 | 0.951 | 24.03 | 48.93 | 480 | 0.324 | 44.93 | 55.97 | 2530 | 0.485 | 45.83 | 66.07 | 815 | 0.528 |

No best models

| | | RESCAL | TransE | DistMult | ComplEx | ConvE |
|---|---|---|---|---|---|---|
| *FB15K-237* | Valid. MRR | *36.1* | *31.5* | *35.0* | *35.3* | *34.3* |
| | Emb. size | 128 *(-0.5)* | 512 *(-3.7)* | 256 *(-0.2)* | 256 *(-0.3)* | 256 *(-0.4)* |
| | Batch size | 512 *(-0.5)* | 128 *(-7.1)* | 1024 *(-0.2)* | 1024 *(-0.3)* | 1024 *(-0.4)* |
| | Train type | 1vsAll *(-0.8)* | NegSamp – | NegSamp *(-0.2)* | NegSamp *(-0.3)* | 1vsAll *(-0.4)* |
| | Loss | CE *(-0.9)* | CE *(-7.1)* | CE *(-3.1)* | CE *(-3.8)* | CE *(-0.4)* |
| | Optimizer | Adam *(-0.5)* | Adagrad *(-3.7)* | Adagrad *(-0.2)* | Adagrad *(-0.5)* | Adagrad *(-1.5)* |
| | Initializer | Normal *(-0.8)* | XvNorm *(-3.7)* | Unif. *(-0.2)* | Unif. *(-0.5)* | XvNorm *(-0.4)* |
| | Regularizer | None *(-0.5)* | L2 *(-3.7)* | L3 *(-0.2)* | L3 *(-0.3)* | L3 *(-0.4)* |
| | Reciprocal | No *(-0.5)* | Yes *(-9.5)* | Yes *(-0.3)* | Yes *(-0.3)* | Yes – |
| *WNRR* | Valid. MRR | *46.8* | *22.6* | *45.4* | *47.6* | *44.3* |
| | Emb. size | 128 *(-1.0)* | 512 *(-5.1)* | 512 *(-1.1)* | 128 *(-1.0)* | 512 *(-1.2)* |
| | Batch size | 128 *(-1.0)* | 128 *(-5.1)* | 1024 *(-1.1)* | 512 *(-1.0)* | 1024 *(-1.3)* |
| | Train type | KvsAll *(-1.0)* | NegSamp – | KvsAll *(-1.1)* | 1vsAll *(-1.0)* | KvsAll *(-1.2)* |
| | Loss | CE *(-2.0)* | CE *(-5.1)* | CE *(-2.4)* | CE *(-3.5)* | CE *(-1.4)* |
| | Optimizer | Adam *(-1.2)* | Adagrad *(-5.8)* | Adagrad *(-1.5)* | Adagrad *(-1.5)* | Adam *(-1.4)* |
| | Initializer | Unif. *(-1.0)* | XvNorm *(-5.1)* | Unif. *(-1.3)* | Unif. *(-1.5)* | XvNorm *(-1.4)* |
| | Regularizer | L3 *(-1.2)* | L2 *(-5.1)* | L3 *(-1.1)* | L2 *(-1.0)* | L1 *(-1.2)* |
| | Reciprocal | Yes *(-1.0)* | Yes *(-5.9)* | Yes *(-1.1)* | No *(-1.0)* | Yes – |



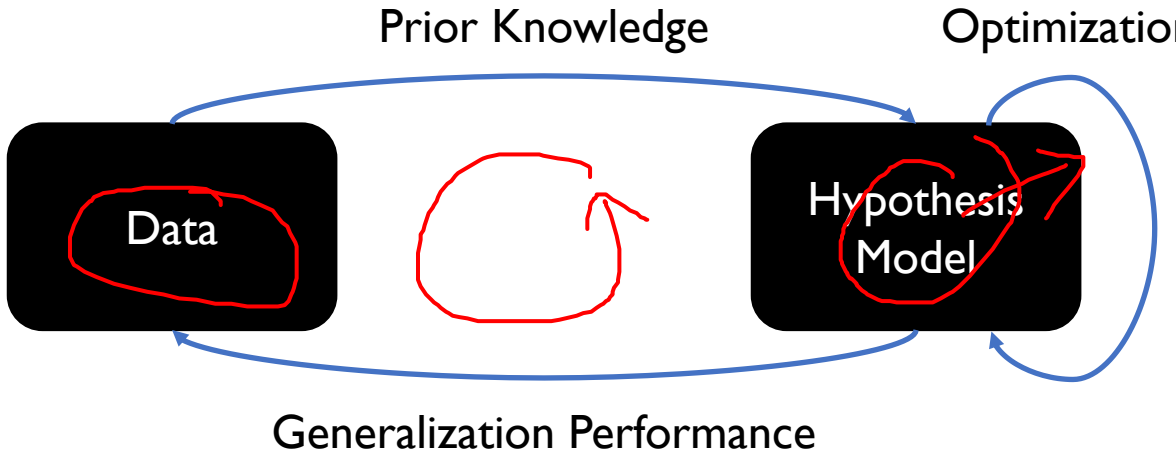

No best hyper-parameters

# Outline

- Background
  - What is knowledge graph embedding
  - What is automated machine learning (AutoML)
  - Limitation of existing works
- A comprehensive understanding of HP in KGE
- An efficient two-stage HP search algorithm
- Experiments
- Key takeaway and future directions

# The Iterative Loop of Machine Learning

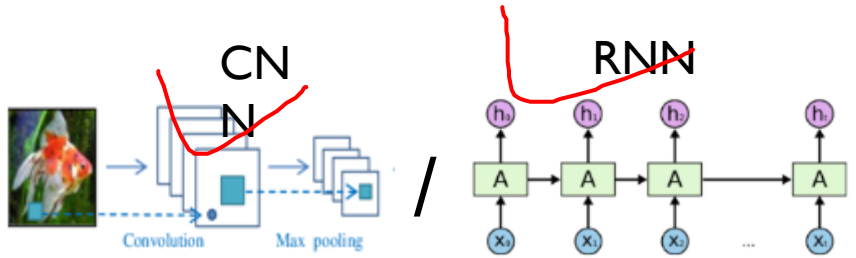




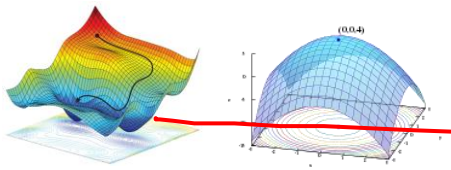

The Advancement of Learning

- An iteration between theory and practice

- A feedback loop




Generalization: What kind of $f$ should we use?



SGD v.s. Adagrad[1]

Optimization: How can we find such $f$?

*Prior knowledge*

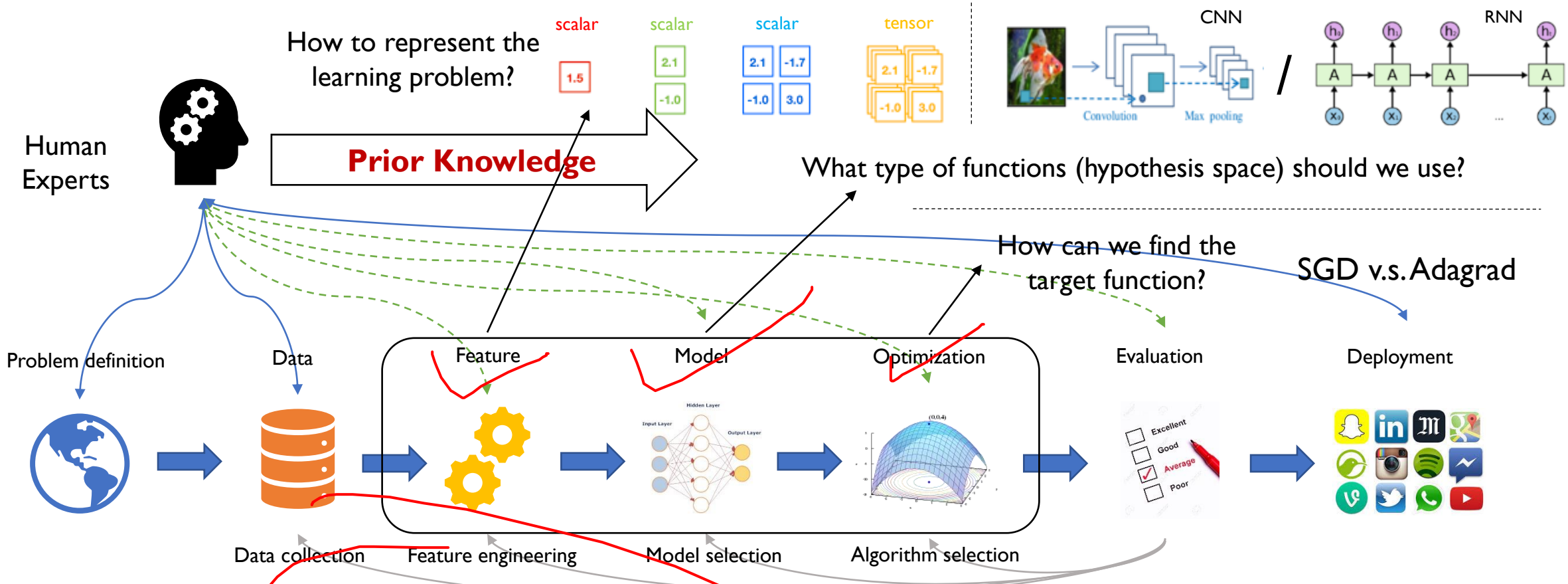
"All models are wrong, but some are useful"[2]

**Better understanding of prior knowledge → Better hypothesis → Better generalization performance**

[1]. Image Source: A. Amini et al. "Spatial Uncertainty Sampling for End-to-End Control". NeurIPS Bayesian Deep Learning 2018

[2] G. Box, Science and statistics, JASA 1976

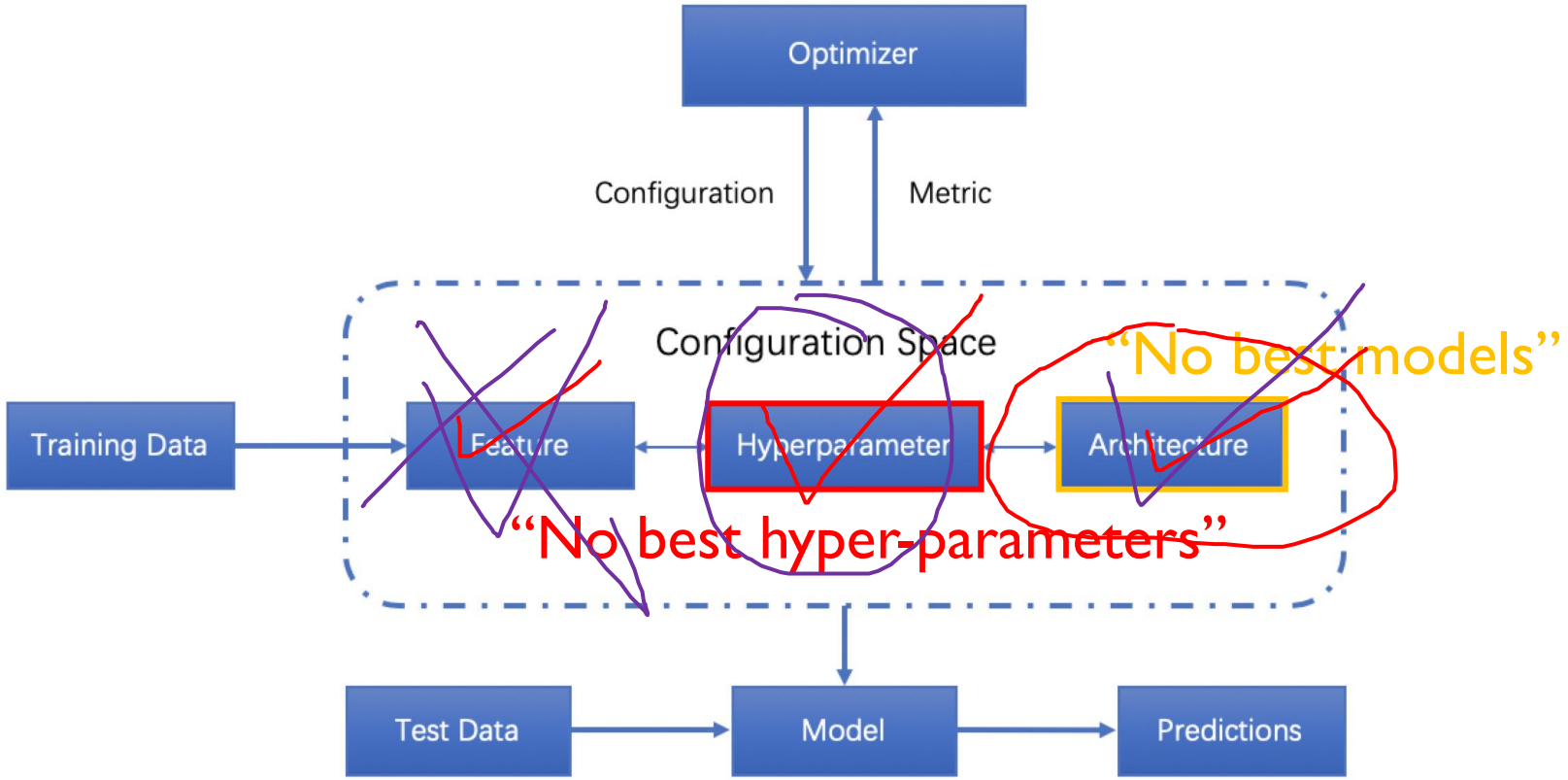
# What is Automated Machine Learning (AutoML)




As a consequence
- Human participations can be naturally replaced by computation power
- total error of machine learning can be reduced (generalization can be improved)

# Current Scope of AutoML






https://awesomeopensource.com/project/hibayesian/awesome-automl-papers

# How to use AutoML


1. Define an AutoML problem

- Derive a search space from insights in specific domains
- Search objective is usually validation performance
- Search constraint is usually resource budgets
- Training objective usually comes from classical learning models

$$\min_{\lambda \in \mathcal{S}} M(F(w^*; \lambda), D_{\text{val}})$$ ← Search Objective

Search Space → $\lambda \in \mathcal{S}$

$$\min_{w} L(F(w; \lambda), D_{\text{tra}})$$ ← Training Objective

s. t.

$$G(\lambda) \leq C$$ ← Search Constraints

2. Design or select proper search algorithm

- Reduce model training cost (time to get $w^*$)

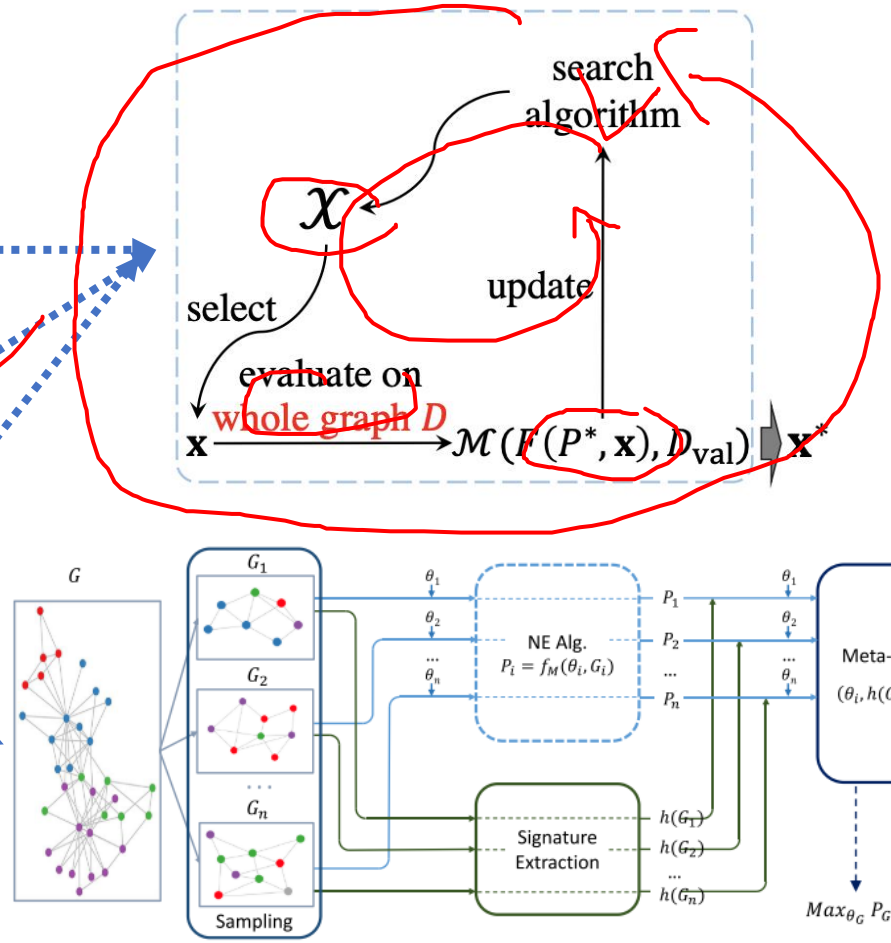
# Outline

- Background
  - What is knowledge graph embedding
  - What is automated machine learning (AutoML)
  - Limitation of existing works
- A comprehensive understanding of HP in KGE
- An efficient two-stage HP search algorithm
- Experiments
- Key takeaway and future directions

# Background – review of HPO methods

AAAI-2022

| Taxonomy | Examples | Cons |
|---|---|---|
| Sampled-based | Grid search | Low efficiency<br>Can not learn from historical records |
| | Random search | |
| Bayesian optimization | Hyperopt (TPE) [1] | Slow feedback from the original KG |
| | SMAC (RF) [2] | |
| | Ax (GP) [3] | |
| | AutoNE [4] | (Subgraph-based)<br>No specialized designs for KGE |
| | e-AutoGR [5] | |
| | $\phi(HP\ configuration) \rightarrow Performance$ | |





[1] Hyperopt: A python library for optimizing the hyperparameters of machine learning algorithms.
[2] Sequential model-based optimization for general algorithm configuration.
[3] https://github.com/facebook/Ax
[4] Autone: Hyperparameter optimization for massive network embedding.
[5] Explainable automated graph representation learning with hyperparameter importance.

# Motivation and Objective



**Major disadvantages of existing works**

Low efficiency in searching for HP configuration

- usually in a time-consuming trial-and-error way
- interaction, importance, and tunability of HPs are unclear
- lacking understanding of KGE components

***Objective of KGbench:***

- *Design a searching algorithm,*
- *for any given dataset and embedding model with limited budget,*
- *to **efficiently** search for the hyper-parameter configuration.*

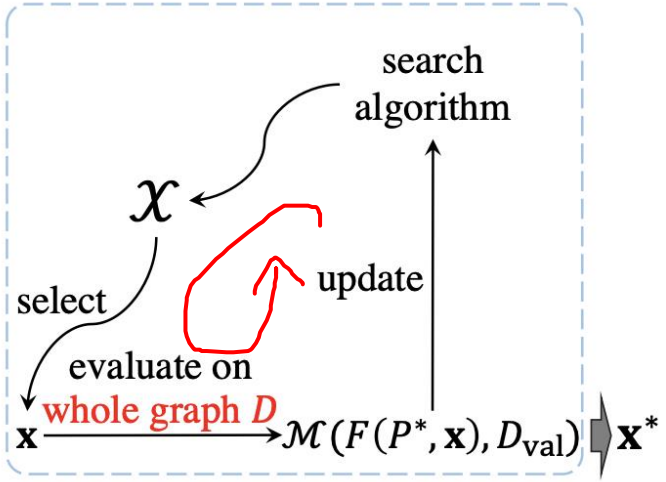
# Motivation and Objective

## HP searching problem setup

**Definition 1** (Hyper-parameter search for KG embedding). *The problem of HP search for KG embedding model is formulated as*

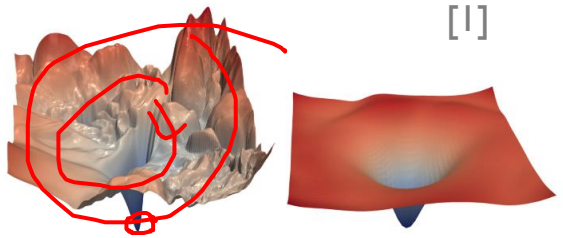
$$\mathbf{x}^* = \arg\max_{\mathbf{x}\in\mathcal{X}} \mathcal{M}\big(F(\boldsymbol{P}^*, \mathbf{x}), D_{val}\big), \quad (2)$$

$$\boldsymbol{P}^* = \arg\min_{\boldsymbol{P}} \mathcal{L}\big(F(\boldsymbol{P}, \mathbf{x}), D_{tra}\big). \quad (3)$$




Three major aspects for efficiency in Def. 1

1. the **size** of search space $\chi$
2. the validation **curvature** of $\mathcal{M}$
3. the evaluation **cost** in solving $argmin_{\mathcal{P}}$



[1]

[1] Visualizing the loss landscape of neural nets

# Outline

- Background
- A comprehensive understanding of HP in KGE
  - search space
  - evaluation cost
  - validation curvature
- An efficient two-stage HP search algorithm
- Experiments
- Key takeaway and future directions

# Understanding the HP in KGE



Three major aspects for efficiency in Def. 1

1. the size of search space $\chi$
2. the validation curvature of $\mathcal{M}$
3. the evaluation cost in solving $argmin_{\mathcal{P}}$

## Recall the search space

| name | type | range |
|---|---|---|
| # negative samples | cat | {32, 128, 512, 2048, 1VsAll, kVsAll} |
| loss function | cat | {MR, BCE_(mean, sum, adv), CE} |
| gamma (MR) | float | [1, 24] |
| adv. weight (BCE_adv) | float | [0.5, 2.0] |
| regularizer | cat | {FRO, NUC, DURA, None} |
| reg. weight (not None) | float | $[10^{-12}, 10^{2}]$ |
| dropout rate | float | [0, 0.5] |
| optimizer | cat | {Adam, Adagrad, SGD} |
| learning rate | float | $[10^{-5}, 10^{0}]$ |
| initializer | cat | {uniform, normal, xavier_uniform, xavier_norm} |
| batch size | int | {128, 256, 512, 1024} |
| dimension size | int | {100, 200, 500, 1000, 2000} |
| inverse relation | bool | {True, False} |

**Questions to be answered**

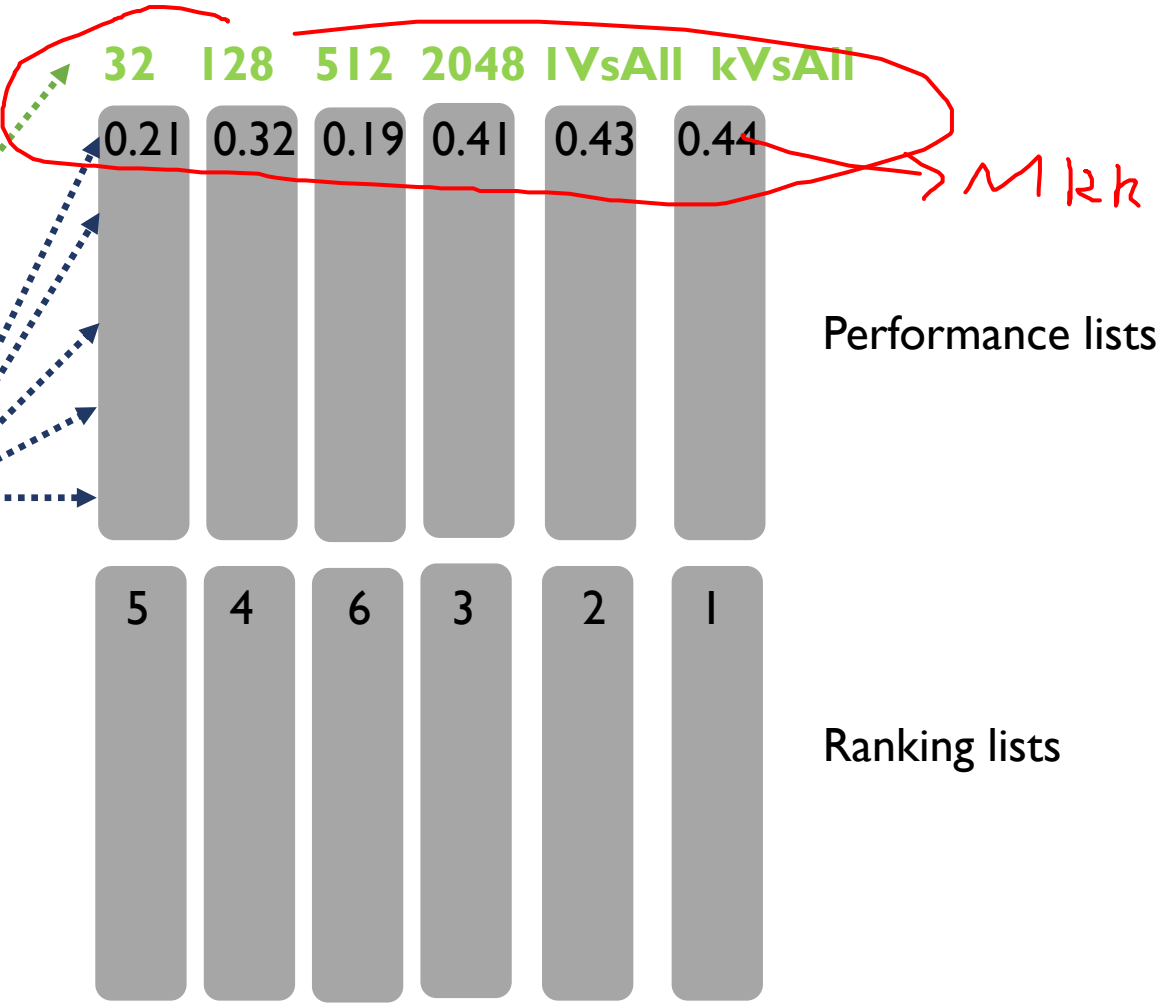
- What are the properties of each HP?
  - ranking distribution
  - consistency
  - computing cost
- Can we decrease the range for each HP?
- Can we decouple some HPs?

# Understanding the HP in KGE

AAAI-2022

## Excavating properties of HPs

| name | type | range |
|---|---|---|
| # negative samples | cat | {32, 128, 512, 2048, 1VsAll, kVsAll} |
| loss function | cat | {MR, BCE_(mean, sum, adv), CE} |
| gamma (MR) | float | [1, 24] |
| adv. weight (BCE_adv) | float | [0.5, 2.0] |
| regularizer | cat | {FRO, NUC, DURA, None} |
| reg. weight (not None) | float | $[10^{-12}, 10^{2}]$ |
| dropout rate | float | [0, 0.5] |
| optimizer | cat | {Adam, Adagrad, SGD} |
| learning rate | float | $[10^{-5}, 10^{0}]$ |
| initializer | cat | {uniform, normal, xavier_uniform, xavier_norm} |
| batch size | int | {128, 256, 512, 1024} |
| dimension size | int | {100, 200, 500, 1000, 2000} |
| inverse relation | bool | {True, False} |




enumerating

sampling [1]

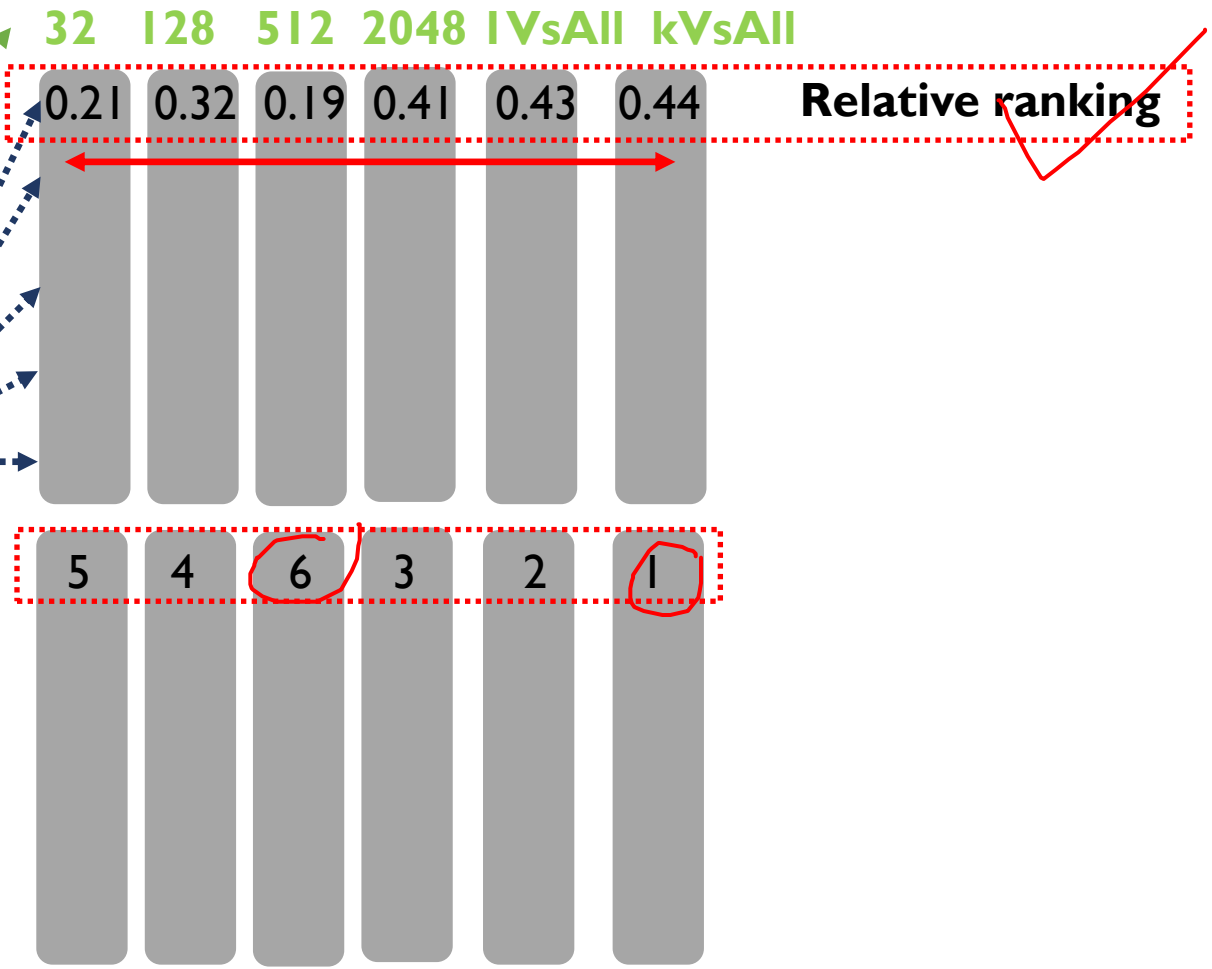
[1] Design Space for Graph Neural Networks

# Understanding the HP in KGE



## Excavating properties of HPs

| name | type | range |
|---|---|---|
| # negative samples | cat | {32, 128, 512, 2048, 1VsAll, kVsAll} |
| loss function | cat | {MR, BCE_(mean, sum, adv), CE} |
| gamma (MR) | float | [1, 24] |
| adv. weight (BCE_adv) | float | [0.5, 2.0] |
| regularizer | cat | {FRO, NUC, DURA, None} |
| reg. weight (not None) | float | $[10^{-12}, 10^{2}]$ |
| dropout rate | float | [0, 0.5] |
| optimizer | cat | {Adam, Adagrad, SGD} |
| learning rate | float | $[10^{-5}, 10^{0}]$ |
| initializer | cat | {uniform, normal, xavier_uniform, xavier_norm} |
| batch size | int | {128, 256, 512, 1024} |
| dimension size | int | {100, 200, 500, 1000, 2000} |
| inverse relation | bool | {True, False} |




enumerating

sampling [1]

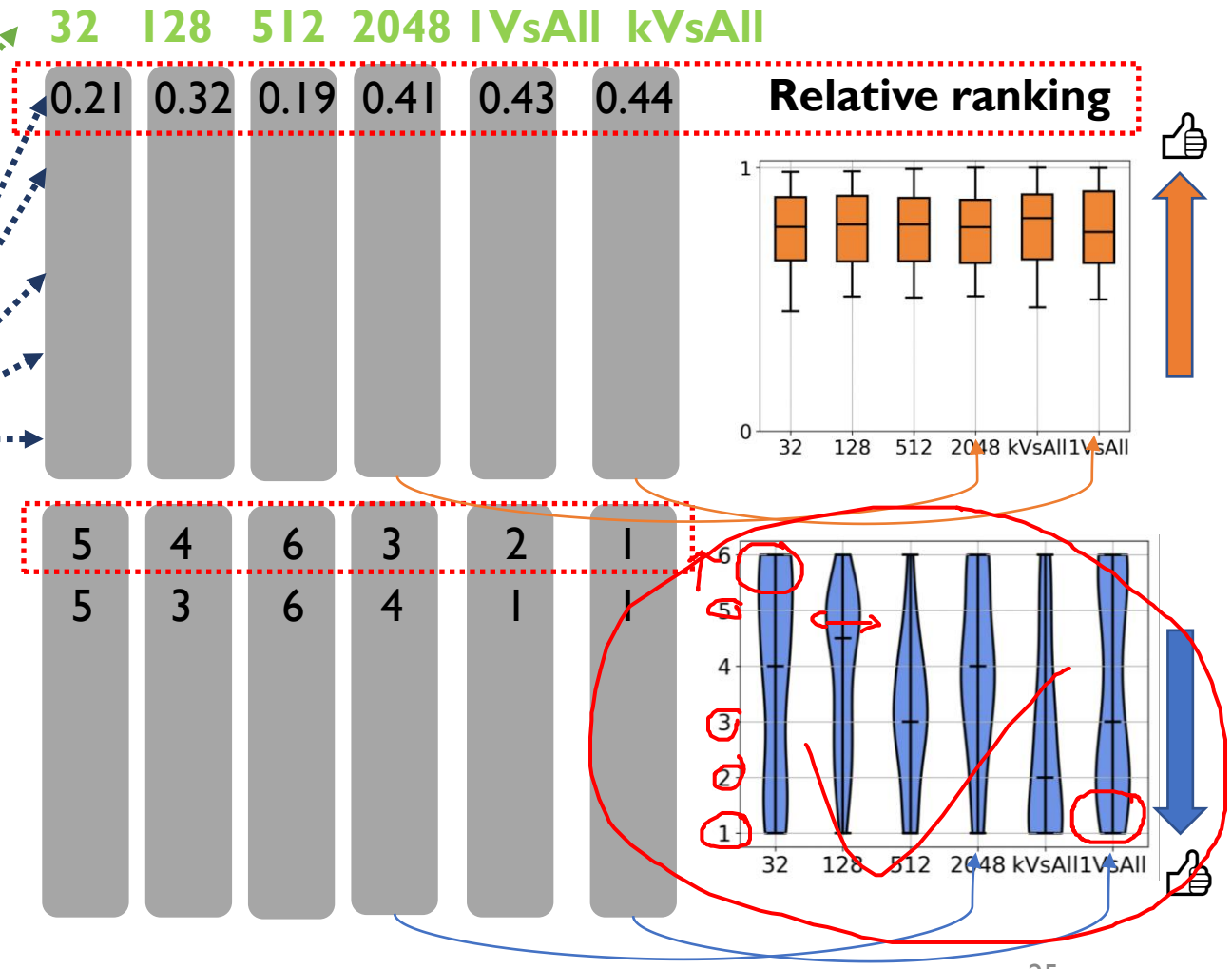
[1] Design Space for Graph Neural Networks

# Understanding the HP in KGE

AAAI-2022

## Excavating properties of HPs

| name | type | range |
|---|---|---|
| # negative samples | cat | {32, 128, 512, 2048, 1VsAll, kVsAll} |
| loss function | cat | {MR, BCE_(mean, sum, adv), CE} |
| gamma (MR) | float | [1, 24] |
| adv. weight (BCE_adv) | float | [0.5, 2.0] |
| regularizer | cat | {FRO, NUC, DURA, None} |
| reg. weight (not None) | float | $[10^{-12}, 10^{2}]$ |
| dropout rate | float | [0, 0.5] |
| optimizer | cat | {Adam, Adagrad, SGD} |
| learning rate | float | $[10^{-5}, 10^{0}]$ |
| initializer | cat | {uniform, normal, xavier_uniform, xavier_norm} |
| batch size | int | {128, 256, 512, 1024} |
| dimension size | int | {100, 200, 500, 1000, 2000} |
| inverse relation | bool | {True, False} |




enumerating

sampling [1]

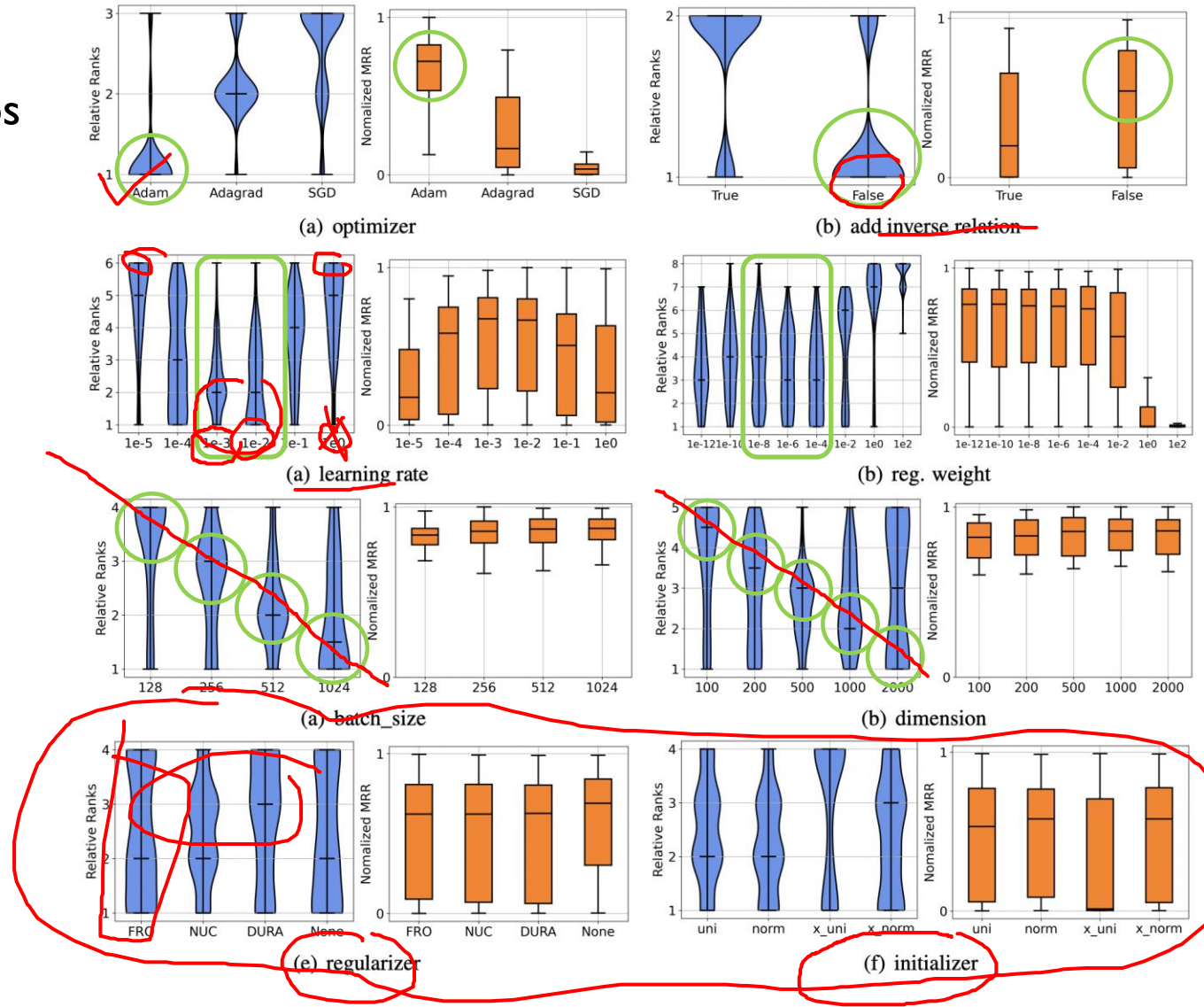
[1] Design Space for Graph Neural Networks

# Understanding the HP in KGE

## **Excavating properties of HPs** | Ranking/Performance distribution

The HPs can be classified into 4 groups

1. fixed choice
2. limited range
3. monotonously related
4. no obvious patterns




✗ lower rankings

✓ higher rankings

# Understanding the HP in KGE

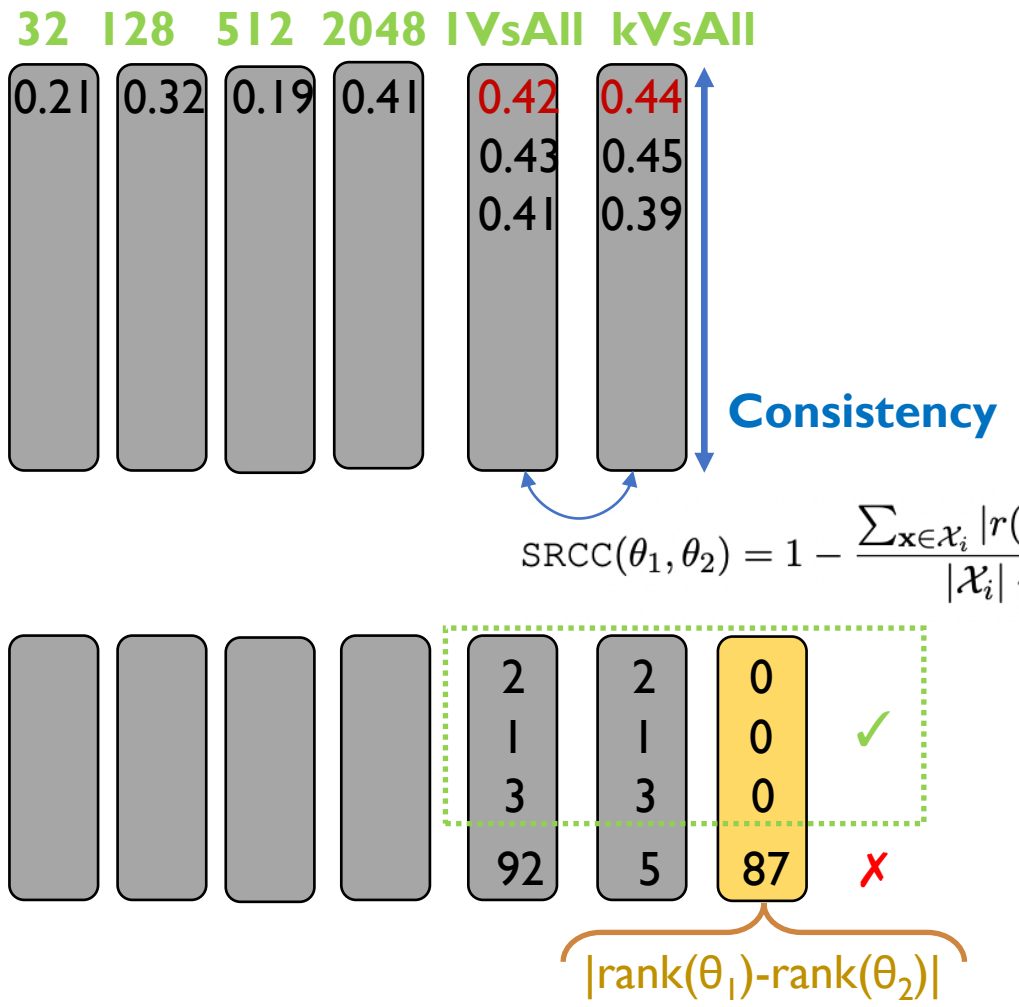
AAAI-2022

## Excavating properties of HPs

| name | type | range |
|---|---|---|
| # negative samples | cat | {32, 128, 512, 2048, 1VsAll, kVsAll} |
| loss function | cat | {MR, BCE_(mean, sum, adv), CE} |
| gamma (MR) | float | [1, 24] |
| adv. weight (BCE_adv) | float | [0.5, 2.0] |
| regularizer | cat | {FRO, NUC, DURA, None} |
| reg. weight (not None) | float | $[10^{-12}, 10^{2}]$ |
| dropout rate | float | [0, 0.5] |
| optimizer | cat | {Adam, Adagrad, SGD} |
| learning rate | float | $[10^{-5}, 10^{0}]$ |
| initializer | cat | {uniform, normal, xavier_uniform, xavier_norm} |
| batch size | int | {128, 256, 512, 1024} |
| dimension size | int | {100, 200, 500, 1000, 2000} |
| inverse relation | bool | {True, False} |

enumerating

sampling

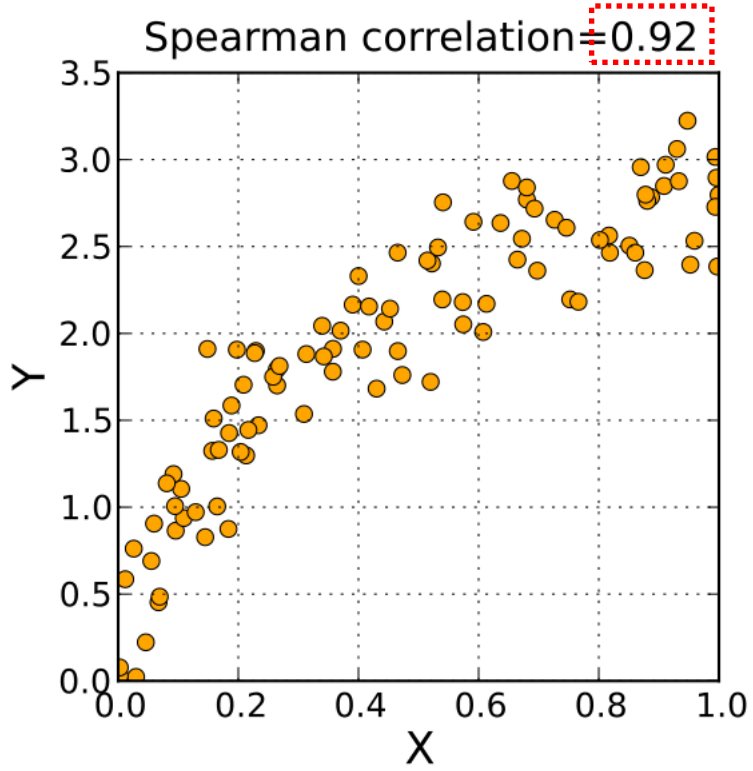


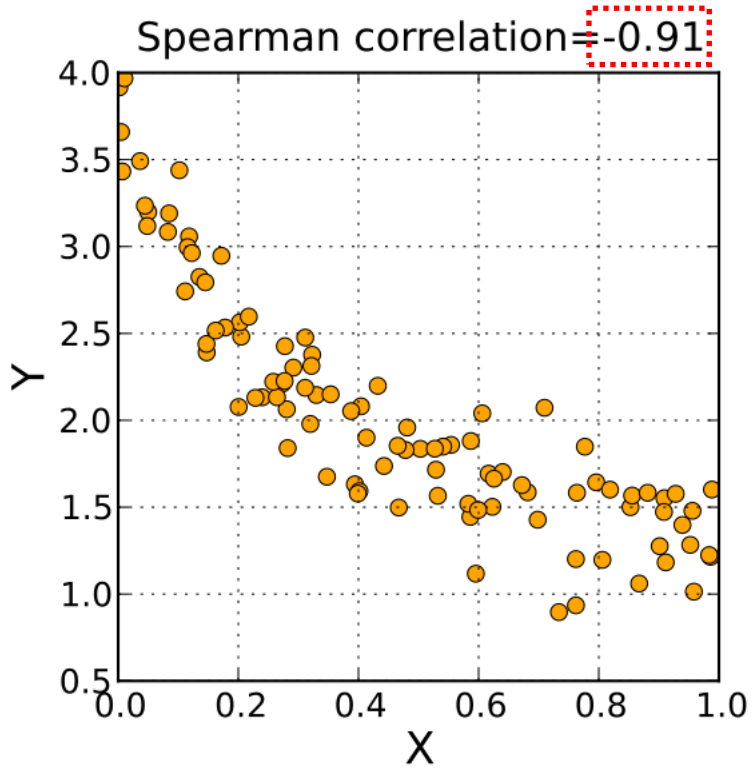
# Understanding the HP in KGE

**Positive and negative Spearman rank correlations**

$$\mathrm{SRCC}(\theta_1, \theta_2) = 1 - \frac{\sum_{\mathbf{x} \in \mathcal{X}_i} |r(\mathbf{x}, \theta_1) - r(\mathbf{x}, \theta_2)|}{|\mathcal{X}_i| \cdot (|\mathcal{X}_i|^2 - 1)}$$

Positive




negative

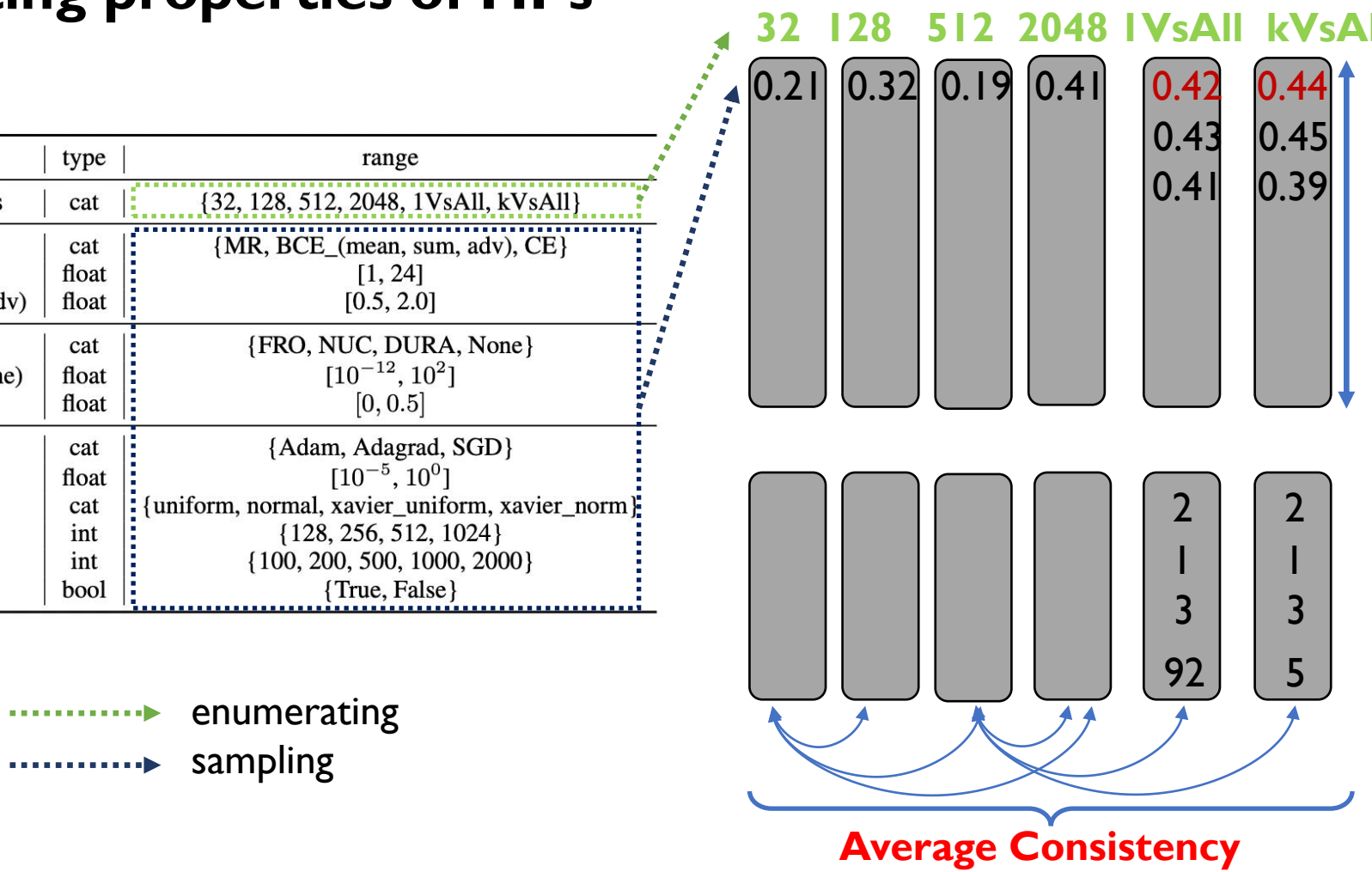
https://en.wikipedia.org/wiki/Spearman%27s_rank_correlation_coefficient

# Understanding the HP in KGE

AAAI-2022

## Excavating properties of HPs

| name | type | range |
|---|---|---|
| # negative samples | cat | {32, 128, 512, 2048, 1VsAll, kVsAll} |
| loss function | cat | {MR, BCE_(mean, sum, adv), CE} |
| gamma (MR) | float | [1, 24] |
| adv. weight (BCE_adv) | float | [0.5, 2.0] |
| regularizer | cat | {FRO, NUC, DURA, None} |
| reg. weight (not None) | float | $[10^{-12}, 10^{2}]$ |
| dropout rate | float | [0, 0.5] |
| optimizer | cat | {Adam, Adagrad, SGD} |
| learning rate | float | $[10^{-5}, 10^{0}]$ |
| initializer | cat | {uniform, normal, xavier_uniform, xavier_norm} |
| batch size | int | {128, 256, 512, 1024} |
| dimension size | int | {100, 200, 500, 1000, 2000} |
| inverse relation | bool | {True, False} |




enumerating

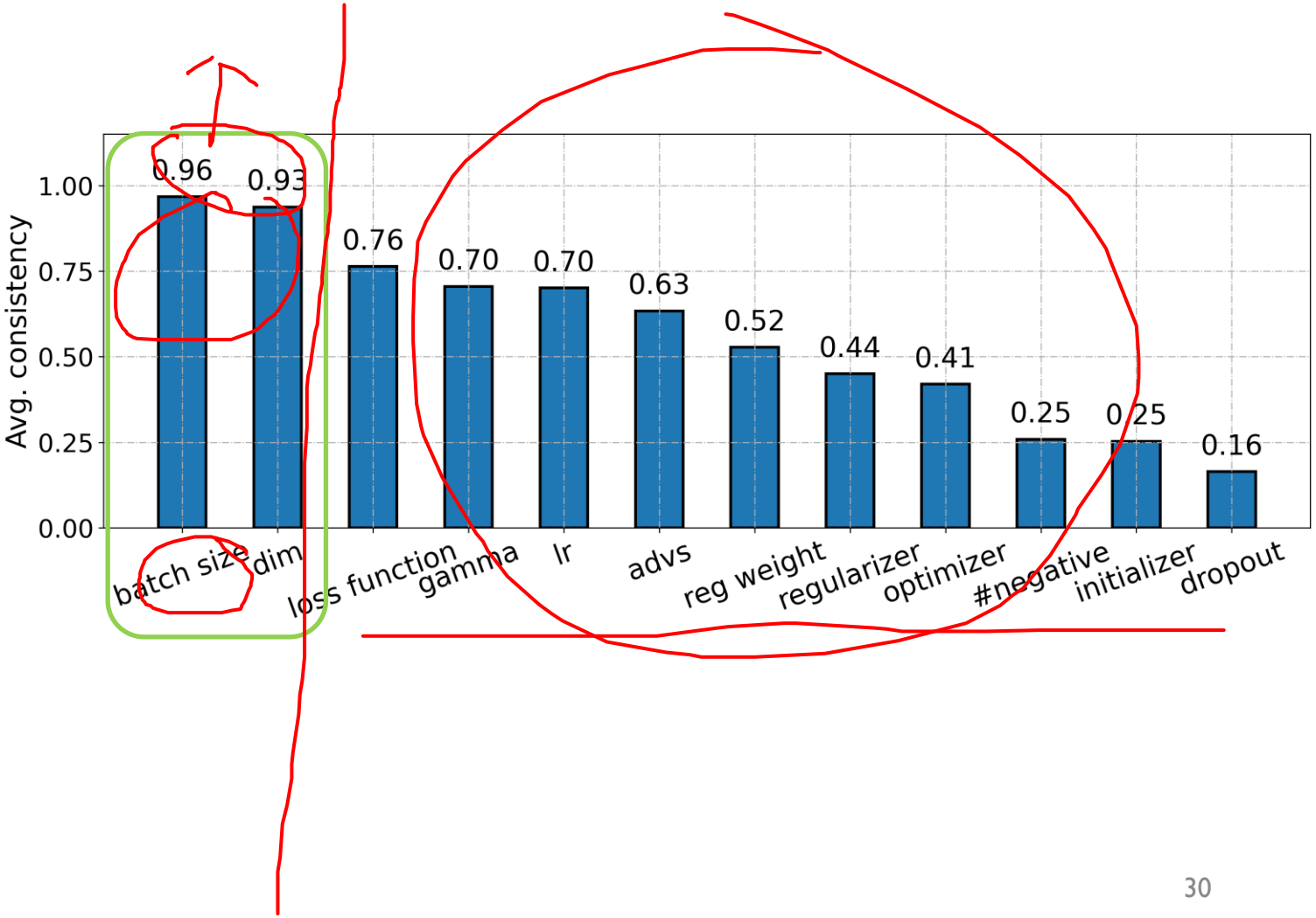
sampling

# Understanding the HP in KGE



## **Excavating properties of HPs** | Consistency

**Observation:**
the batchsize and dimension show higher consistency than the other HPs.



# Outline

- Background
- A comprehensive understanding of HP in KGE
  - search space
  - validation curvature
  - evaluation cost
- An efficient two-stage HP search algorithm
- Experiments
- Key takeaway and future directions

# Understanding the HP in KGE




**Excavating properties of HPs** | from the aspect of predictor

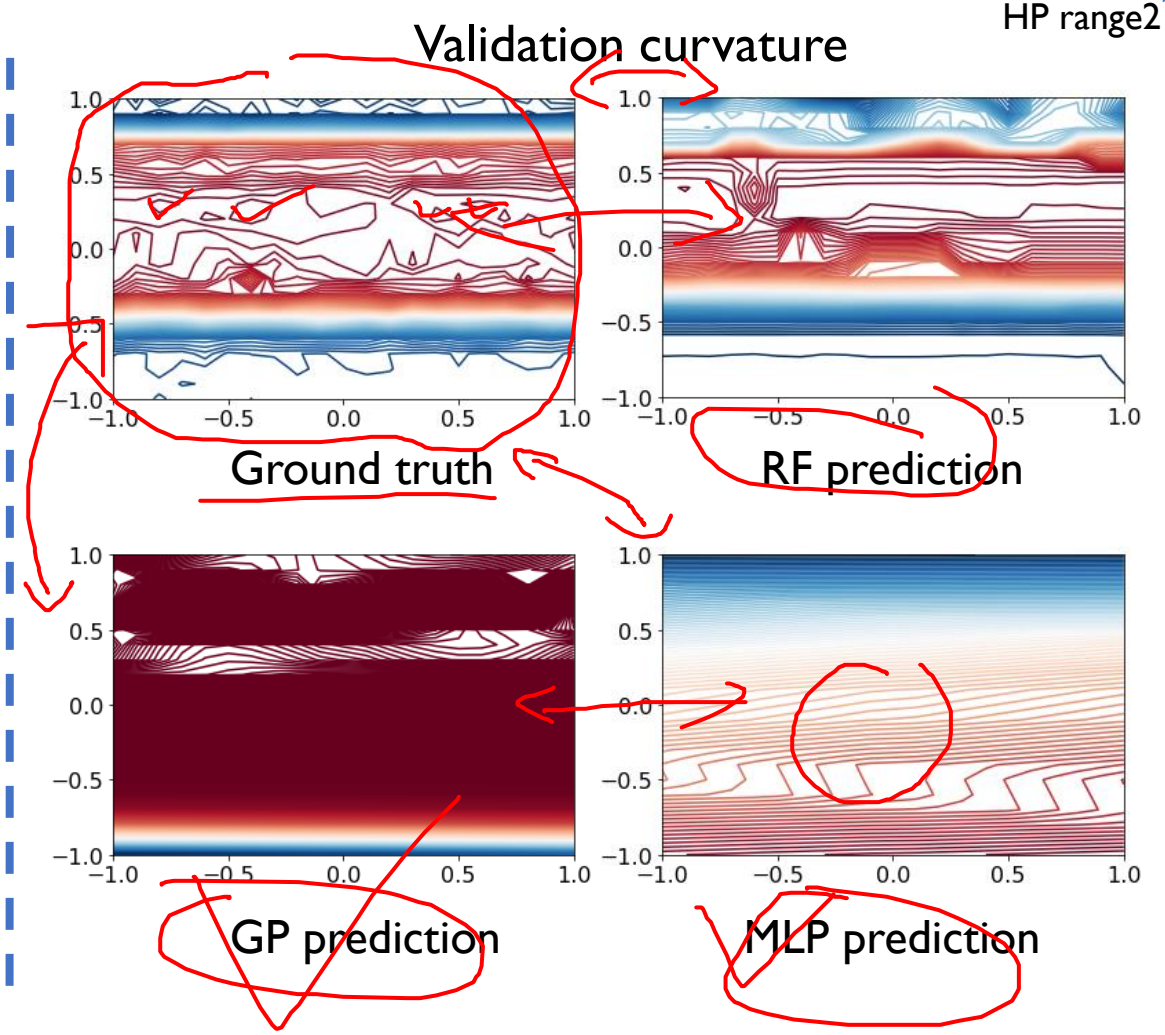
Predictors

- Gaussian process (GP)
- Multi layer perceptron (MLP)
- Random forest (RF)

Observation:
RF is better in approximating the curvature

| # train configurations | 10 | 20 | 30 |
|---|---|---|---|
| GP | 0.0693±0.02 | 0.029±0.01 | 0.019±0.01 |
| MLP | 2.121±0.4 | 2.052±0.3 | 0.584±0.1 |
| RF | **0.003±0.002** | **0.002±0.001** | **0.001±0.001** |

MSE results

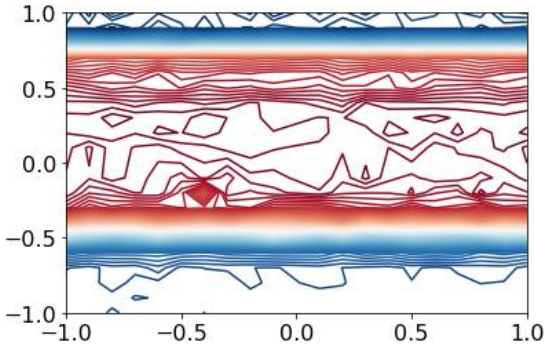


# Understanding the HP in KGE

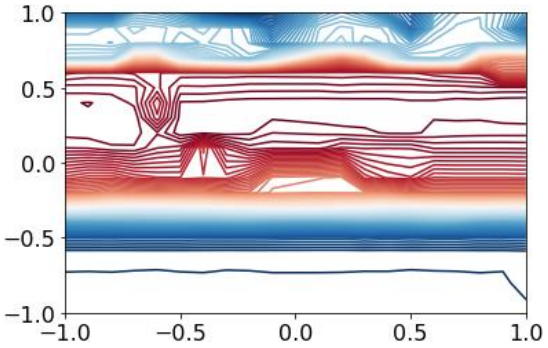
AAAI-2022

## **Excavating properties of HPs** | from the aspect of predictor

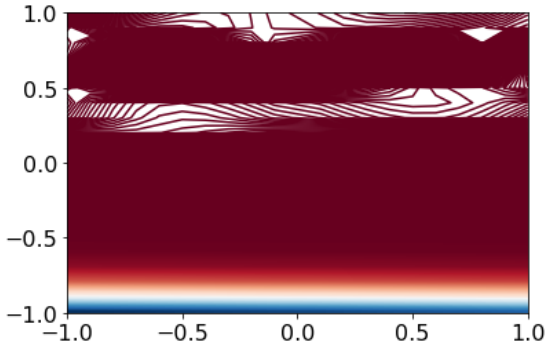
Ground truth



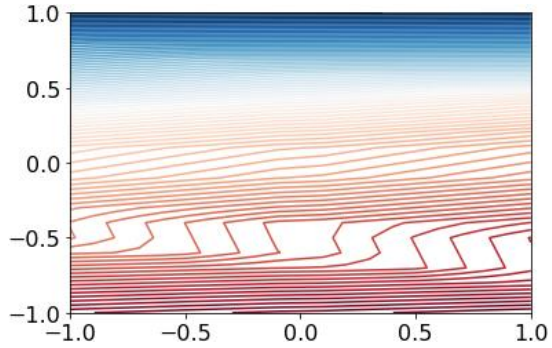
RF prediction

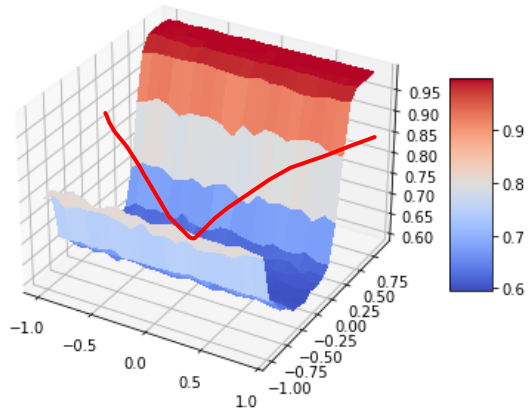


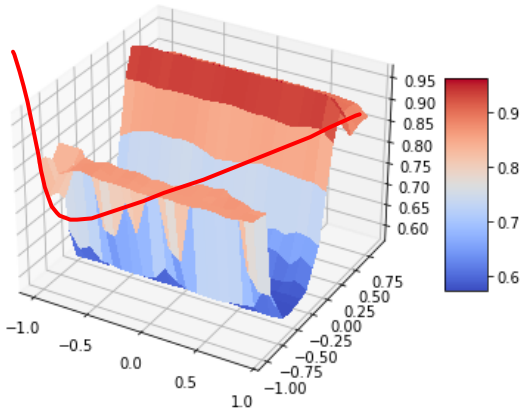
GP prediction

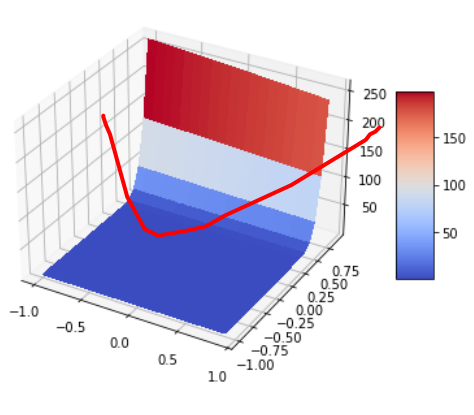


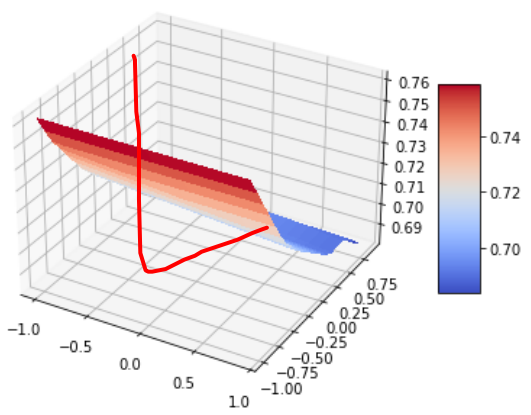
MLP prediction











# Outline

- Background
- A comprehensive understanding of HP in KGE
  - search space
  - validation curvature
  - evaluation cost
- An efficient two-stage HP search algorithm
- Experiments
- Key takeaway and future directions

# Understanding the HP in KGE

## **Excavating properties of HPs** | Time cost

Three major aspects for efficiency in Def. 1

1. the size of search space $\chi$
2. the validation curvature of $\mathcal{M}$
3. the evaluation cost in solving $argmin_{\mathcal{P}}$

| dataset | #entity | #relation | #train | #validate | #test |
|---|---|---|---|---|---|
| WN18RR (Dettmers et al., 2017) | 41k | 11 | 87k | 3k | 3k |
| FB15k-237 (Toutanova and Chen, 2015) | 15k | 237 | 272k | 18k | 20k |
| ogbl-biokg (Hu et al., 2020) | 94k | 51 | 4,763k | 163k | 163k |
| ogbl-wikikg2 (Hu et al., 2020) | 2,500k | 535 | 16,109k | 429k | 598k |

Average evaluation time cost:

~2.1h

~3.5h

~17.3h

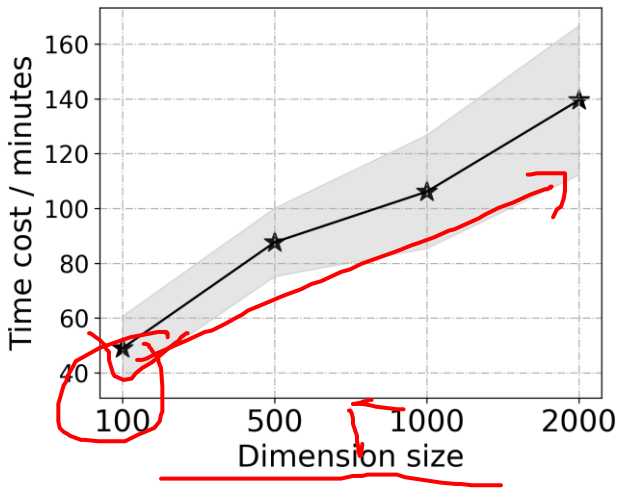
~21.7h

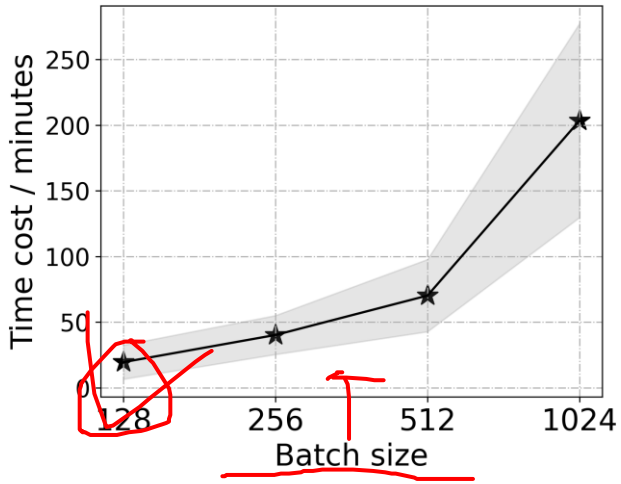
# Understanding the HP in KGE

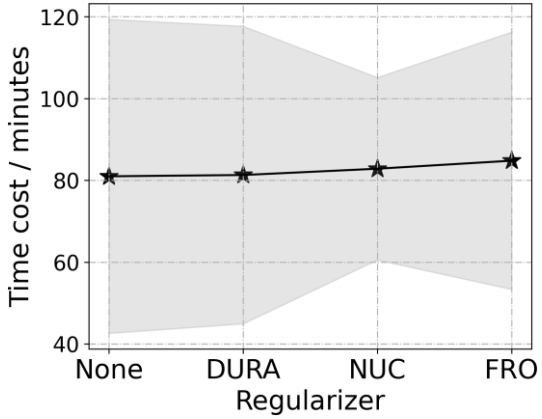
## **Excavating properties of HPs** | Time cost[1]

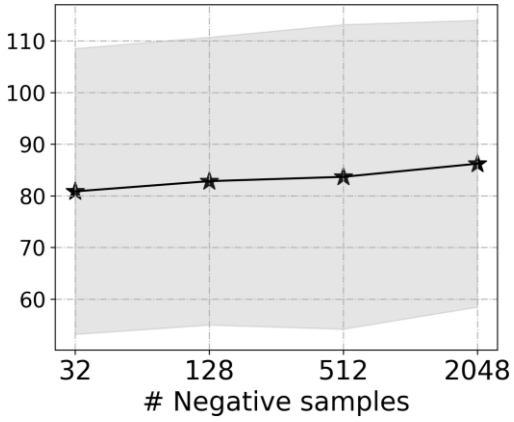
Three major aspects for efficiency in Def. 1
1. the size of search space $\chi$
2. the validation curvature of $\mathcal{M}$
3. the evaluation cost in solving $argmin_{\mathcal{P}}$

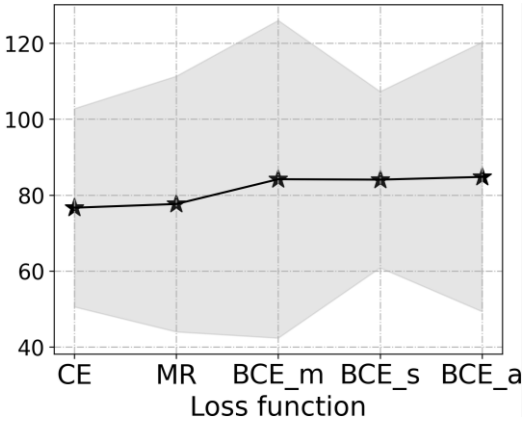
Observation:
batchsize↑ or dimension↑
⇒ time cost↑



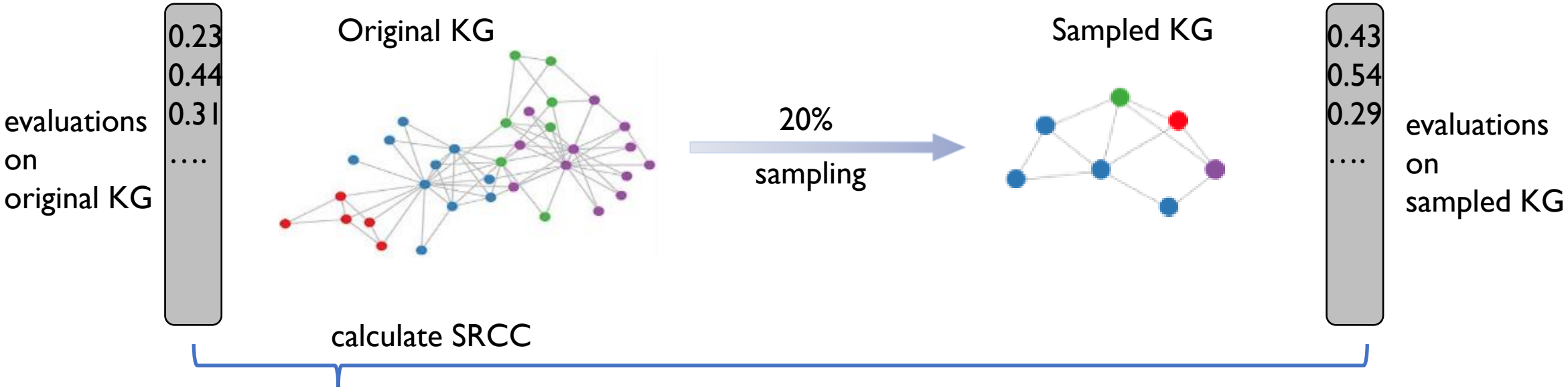



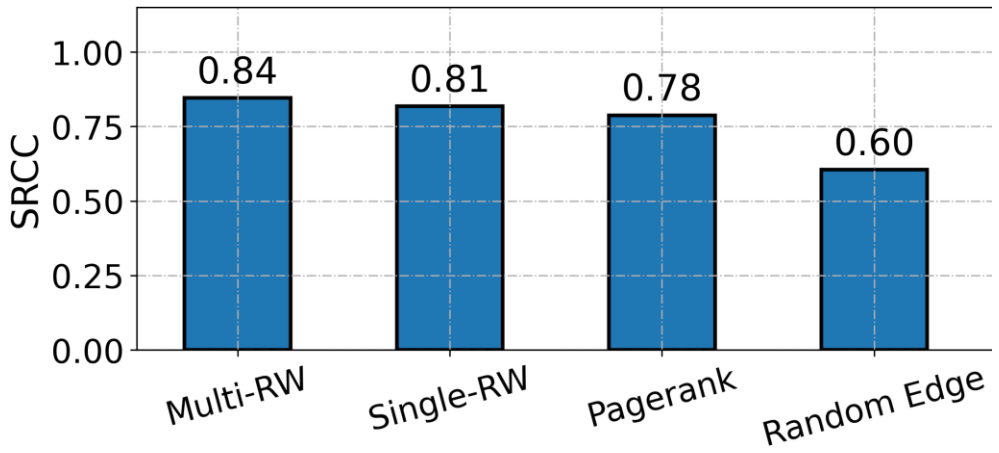




1: The experiments are implemented with PyTorch framework,
on a machine with Intel Xeon 6230R CPUs, 754 GB memory and RTX 3090 GPUs with 24 GB.

# Understanding the HP in KGE

## **Excavating properties of HPs** | Transferability of subgraphs



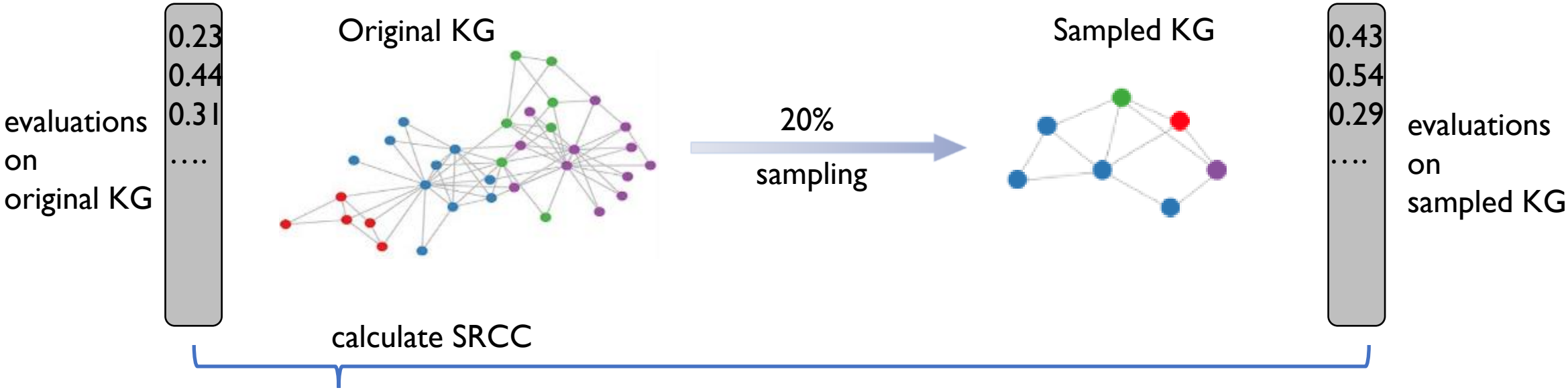



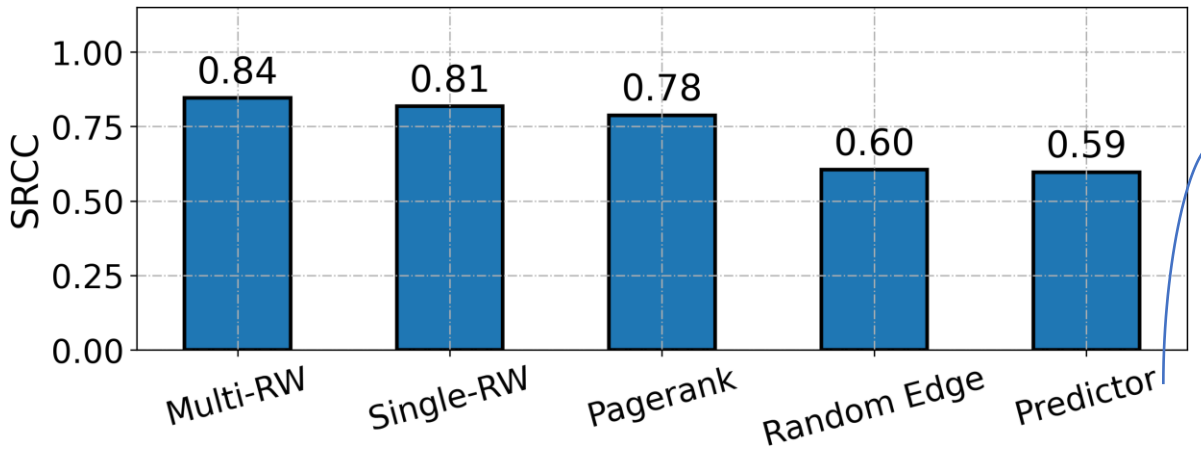

Observation:

- Good correlation between subgraph and original graph
- Multi-start random walk is the best choice

# Understanding the HP in KGE

## **Excavating properties of HPs** | Transferability of subgraphs

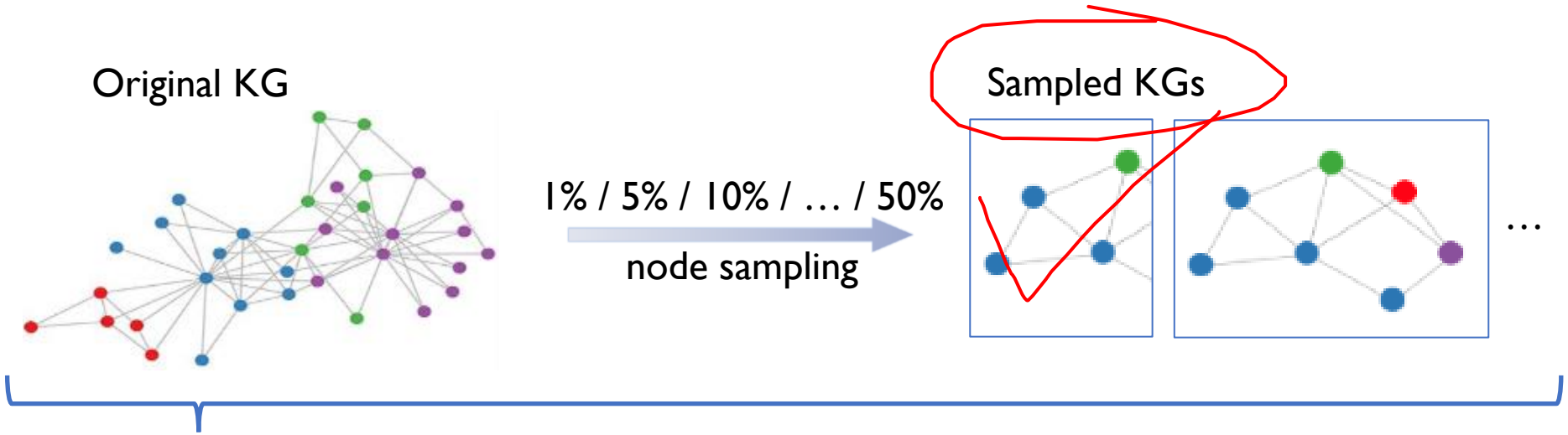


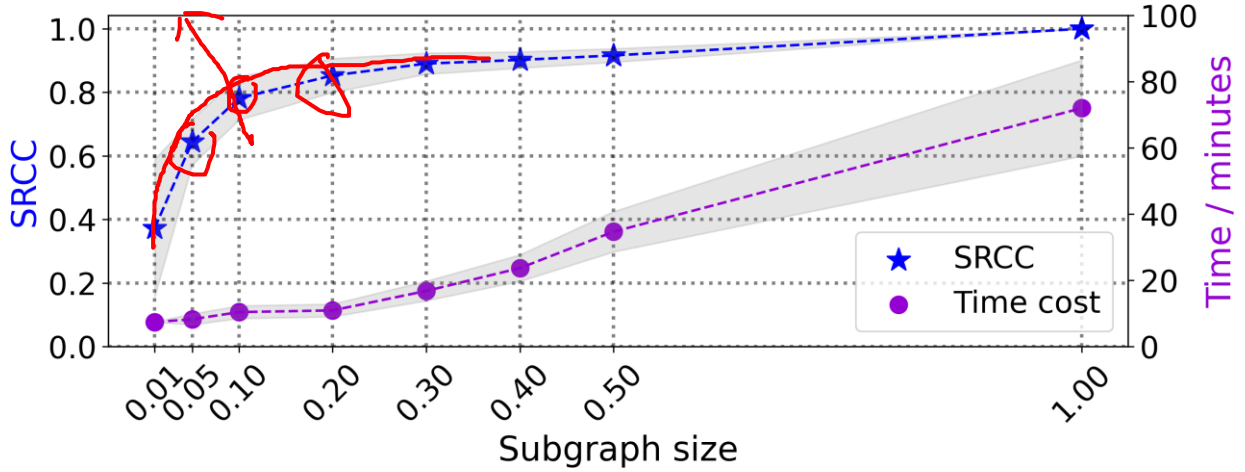




Predictor
- train with sampled KG + original KG
- lower SRCC compared with direct transfer
- similar cases for GP/MLP/RF

# Understanding the HP in KGE

## **Excavating properties of HPs** | Transferability of subgraphs




Observation:

- When the subgraph size↑, the consistency↑ and the cost↑
- To balance the consistency and cost, the subgraphs with 20% nodes are the better choices

# Understanding the HP in KGE



## Summary of the observations

- Ranking distribution/consistency for each HP's values
  - dimension/batchsize
- Full HP range can be reduced and decoupled

- The validation curvature is pretty complex
- RF is better than GP/MLP as the predictor

- Sampling with multi-start random walk can reduce cost while possessing high performance consistency

Three major aspects for efficiency in Def. 1
1. the size of search space $\chi$
2. the validation curvature of $\mathcal{M}$
3. the evaluation cost in solving $argmin_{\mathcal{P}}$

$$\mathbf{x}^* = \arg\max_{\mathbf{x}\in\mathcal{X}} \mathcal{M}\big(F(\boldsymbol{P}^*, \mathbf{x}), D_{val}\big),$$
$$\boldsymbol{P}^* = \arg\min_{\boldsymbol{P}} \mathcal{L}\big(F(\boldsymbol{P}, \mathbf{x}), D_{tra}\big).$$

***How to design algorithm based on the above observations?*** 🤔

# Outline

- Background
- A comprehensive understanding of HP in KGE
- An efficient two-stage HP search algorithm
- Experiments
- Key takeaway and future directions

# Efficient two-stage HP search algorithm



## Reducing the search space

| name | ranges in the whole space | revised ranges |
|---|---|---|
| optimizer | {Adam, Adagrad, SGD} | Adam |
| learning rate | $[10^{-5}, 10^{0}]$ | $[10^{-4}, 10^{-1}]$ |
| reg. weight | $[10^{-12}, 10^{2}]$ | $[10^{-8}, 10^{-2}]$ |
| dropout rate | [0, 0.5] | [0, 0.3] |
| inverse relation | {True, False} | {False} |
| batch size | {128, 256, 512, 1024} | 128 |
| dimension size | {100, 200, 500, 1000, 2000} | 100 |

reduced range HPs:
can be searched more exactly

decoupled HPs:
can be directly tuned
apart from other HPs

Reduced space = shrinkage range HPs + decoupled HPs
The reduced space is about **700 times smaller** than the full space

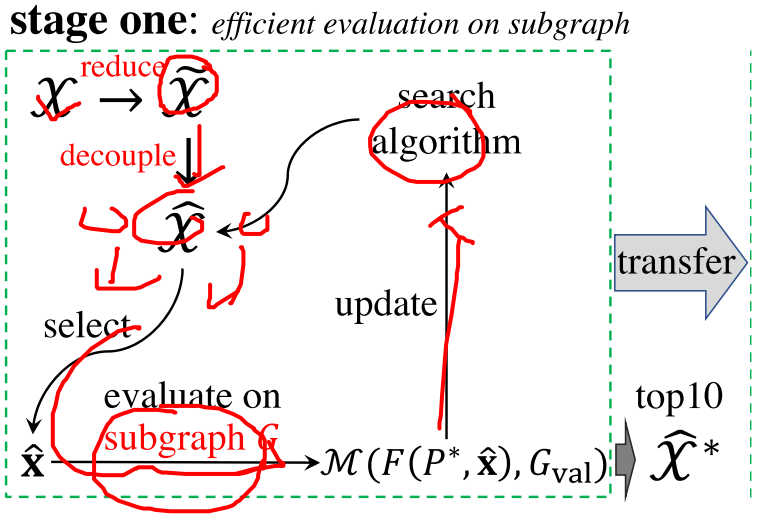
# Efficient two-stage HP search algorithm



## TwO-Stage Search algorithm (TOSS)

**Stage 1:** exploration on reduced space

- quickly search HP on sampled KG
- with predictor RF and acquisition BORE




**Algorithm 1** TOSS: two-stage search algorithm

**Require:** KG embedding model $F$, dataset $D$, and budget $B$;
reduce the search space $\mathcal{X}$ to $\tilde{\mathcal{X}}$ and decouple $\tilde{\mathcal{X}}$ to $\hat{\mathcal{X}}$;
**# state one**: *efficient evaluation on subgraph*
sample a subgraph (with 20% entities) $G$ from $D_{\text{tra}}$ by multi-start random walk;
**repeat**
  sample a configuration $\hat{\mathbf{x}}$ from $\hat{\mathcal{X}}$ by RF+BORE;
  evaluate $\hat{\mathbf{x}}$ on the subgraph $G$ to get the performance;
  update the RF with record $\left(\hat{\mathbf{x}}, \mathcal{M}(F(P^*, \hat{\mathbf{x}}), G_{\text{val}})\right)$;
**until** $B/2$ budget exhausted;
save the top10 configurations in $\hat{\mathcal{X}}^*$;
**# state two**: *fine-tune the top configurations*
increase the batch/dimension size in $\hat{\mathcal{X}}^*$ to get $\tilde{\mathcal{X}}^*$;
set $y^* = 0$ and initialize the RF surrogate;
**repeat**
  select a configuration $\tilde{\mathbf{x}}^*$ from $\tilde{\mathcal{X}}^*$ by RF+BORE;
  evaluate on whole graph $G$ to get the performance;
  update the RF with record $\left(\tilde{\mathbf{x}}^*, \mathcal{M}(F(P^*, \tilde{\mathbf{x}}^*), D_{\text{val}})\right)$;
  **if** $\mathcal{M}(F(P^*, \tilde{\mathbf{x}}^*), D_{\text{val}}) > y^*$ **then**
  $y^* \leftarrow \mathcal{M}(F(P^*, \tilde{\mathbf{x}}^*), D_{\text{val}})$ and $\mathbf{x}^* \leftarrow \tilde{\mathbf{x}}^*$; **end if**
**until** $B/2$ budget exhausted;
**return** $\mathbf{x}^*$.

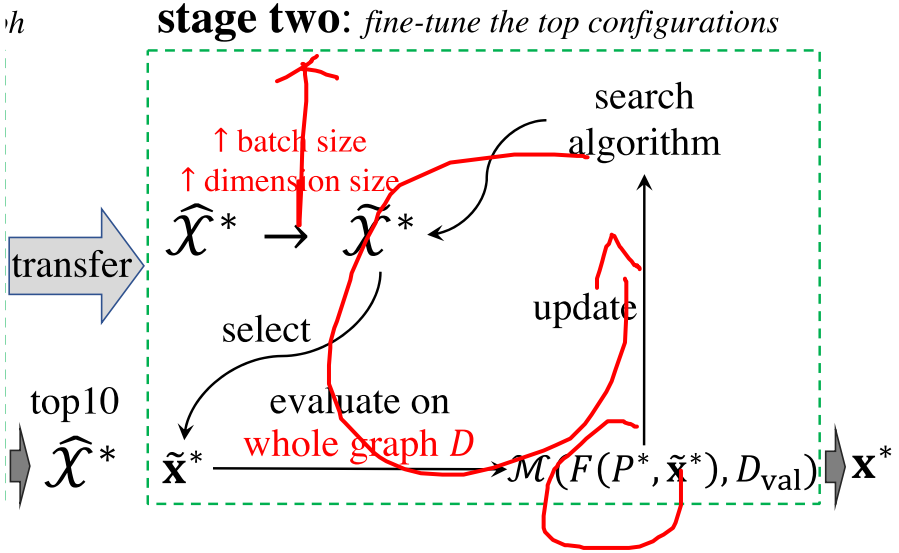
# Efficient two-stage HP search algorithm



## TwO-Stage Search algorithm (TOSS)

**Stage2:** exploitation with fine-tuning

- transfer top10 configurations from stage 1
- finetune configuration on original KG
  - with higher dimension and batchsize




**Algorithm 1** TOSS: two-stage search algorithm

**Require:** KG embedding model $F$, dataset $D$, and budget $B$;

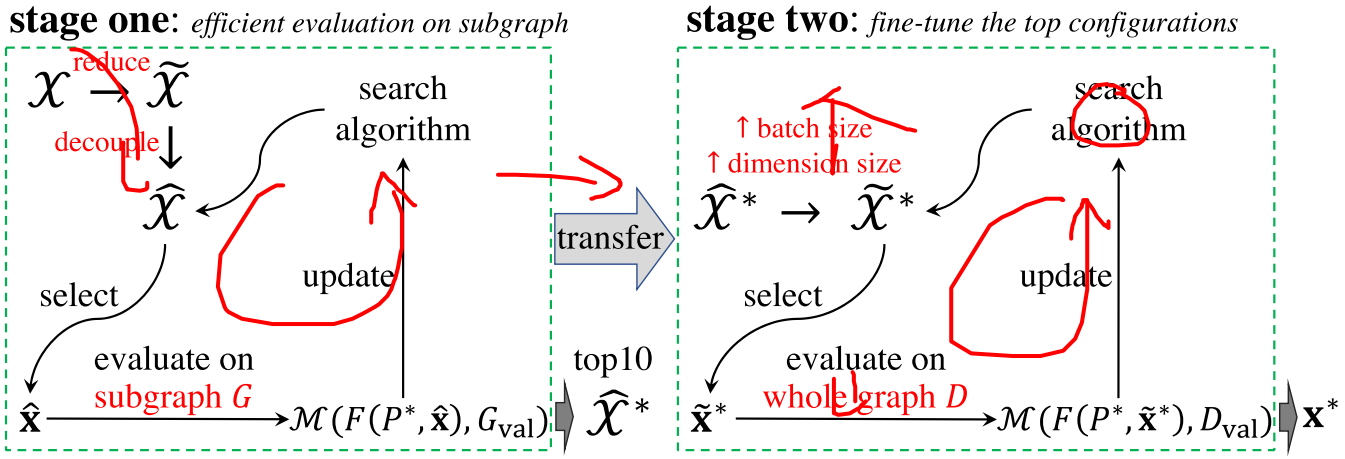
reduce the search space $\mathcal{X}$ to $\tilde{\mathcal{X}}$ and decouple $\tilde{\mathcal{X}}$ to $\hat{\mathcal{X}}$;
**# state one**: *efficient evaluation on subgraph*
sample a subgraph (with 20% entities) $G$ from $D_{\text{tra}}$ by multi-start random walk;
**repeat**
  sample a configuration $\hat{\mathbf{x}}$ from $\hat{\mathcal{X}}$ by RF+BORE;
  evaluate $\hat{\mathbf{x}}$ on the subgraph $G$ to get the performance;
  update the RF with record $\left(\hat{\mathbf{x}}, \mathcal{M}(F(P^*, \hat{\mathbf{x}}), G_{\text{val}})\right)$;
**until** $B/2$ budget exhausted;
save the top10 configurations in $\hat{\mathcal{X}}^*$;
**# state two**: *fine-tune the top configurations*
increase the batch/dimension size in $\hat{\mathcal{X}}^*$ to get $\tilde{\mathcal{X}}^*$;
set $y^* = 0$ and initialize the RF surrogate;
**repeat**
  select a configuration $\tilde{\mathbf{x}}^*$ from $\tilde{\mathcal{X}}^*$ by RF+BORE;
  evaluate on whole graph $G$ to get the performance;
  update the RF with record $\left(\tilde{\mathbf{x}}^*, \mathcal{M}(F(P^*, \tilde{\mathbf{x}}^*), D_{\text{val}})\right)$;
  **if** $\mathcal{M}(F(P^*, \tilde{\mathbf{x}}^*), D_{\text{val}}) > y^*$ **then**
    $y^* \leftarrow \mathcal{M}(F(P^*, \tilde{\mathbf{x}}^*), D_{\text{val}})$ and $\mathbf{x}^* \leftarrow \tilde{\mathbf{x}}^*$; **end if**
**until** $B/2$ budget exhausted;
**return** $\mathbf{x}^*$.

# Efficient two-stage HP search algorithm



**Stage1:** exploration on reduced space
- quickly search HP on sampled KG
- with predictor RF and acquisition BORE

**Stage2:** exploitation with fine-tuning
- transfer top10 configurations from stage 1
- finetune configuration on original KG



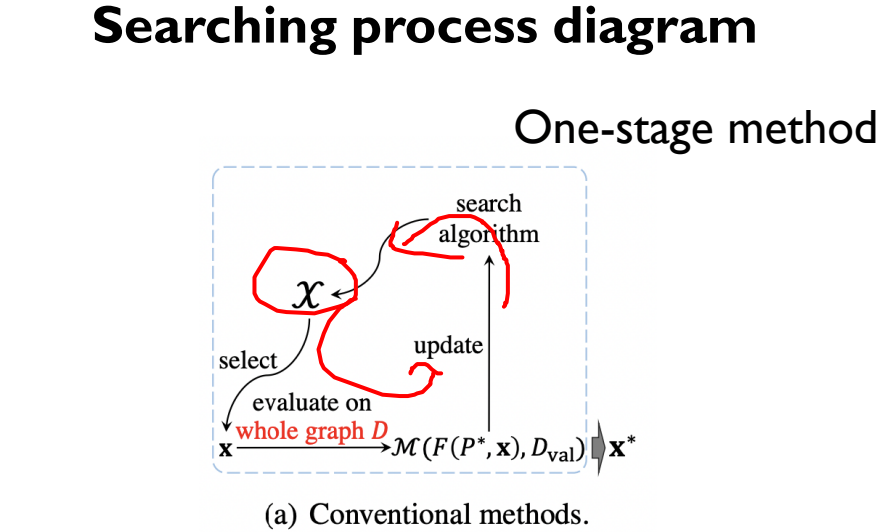

**Algorithm 1** TOSS: two-stage search algorithm

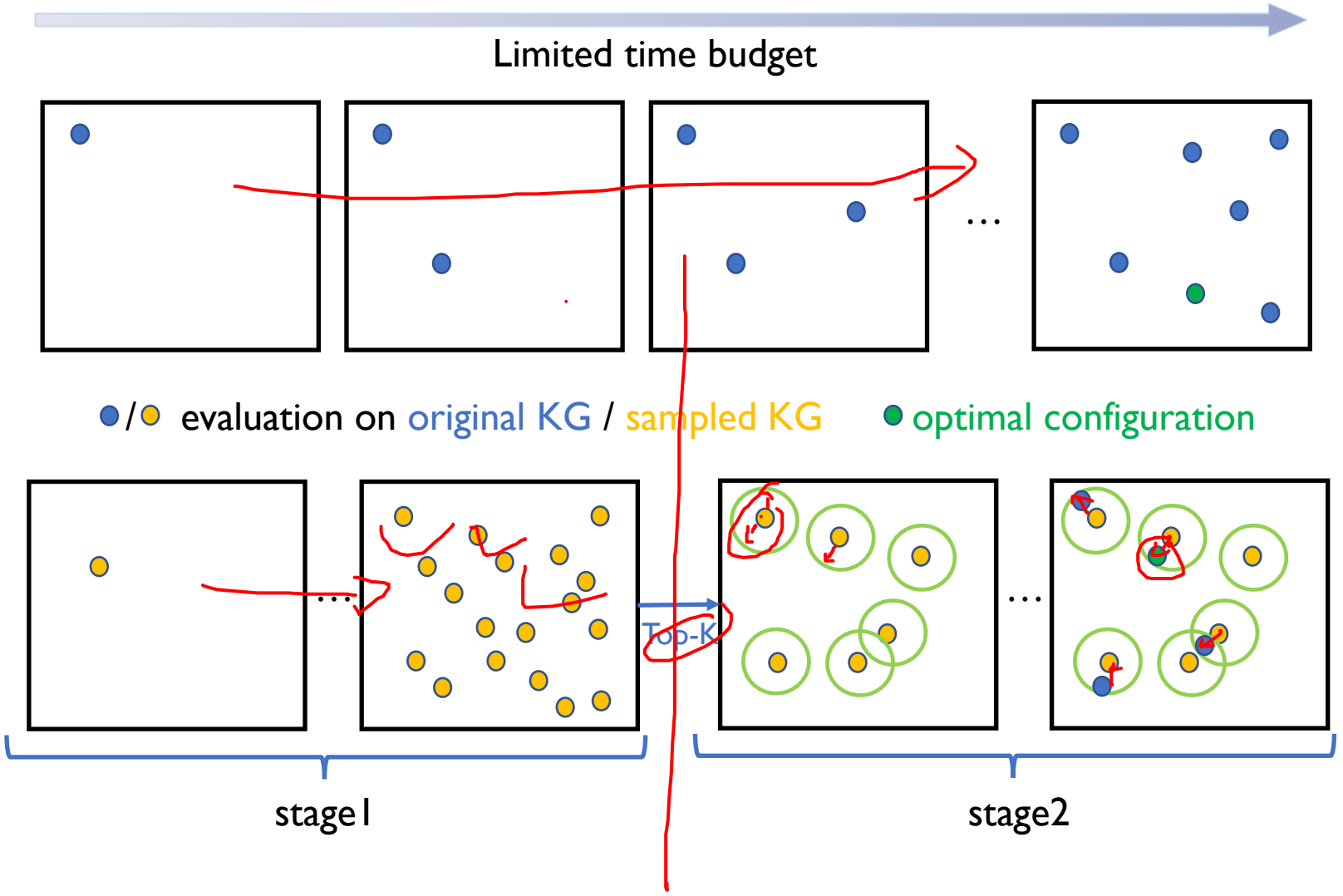
**Require:** KG embedding model $F$, dataset $D$, and budget $B$;
1: reduce the search space $\mathcal{X}$ to $\tilde{\mathcal{X}}$ and decouple $\tilde{\mathcal{X}}$ to $\hat{\mathcal{X}}$;
**# state one**: *efficient evaluation on subgraph*
2: sample a subgraph (with 20% entities) $G$ from $D_{\text{tra}}$ by multi-start random walk;
3: **repeat**
4: sample a configuration $\hat{\mathbf{x}}$ from $\hat{\mathcal{X}}$ by RF+BORE;
5: evaluate $\hat{\mathbf{x}}$ on the subgraph $G$ to get the performance;
6: update the RF with record $\big(\hat{\mathbf{x}}, \mathcal{M}(F(P^*, \hat{\mathbf{x}}), G_{\text{val}})\big)$;
7: **until** $B/2$ budget exhausted;
8: save the top10 configurations in $\hat{\mathcal{X}}^*$;
**# state two**: *fine-tune the top configurations*
9: increase the batch/dimension size in $\hat{\mathcal{X}}^*$ to get $\tilde{\mathcal{X}}^*$;
10: set $y^* = 0$ and initialize the RF surrogate;
11: **repeat**
12: select a configuration $\tilde{\mathbf{x}}^*$ from $\tilde{\mathcal{X}}^*$ by RF+BORE;
13: evaluate on whole graph $G$ to get the performance;
14: update the RF with record $\big(\tilde{\mathbf{x}}^*, \mathcal{M}(F(P^*, \tilde{\mathbf{x}}^*), D_{\text{val}})\big)$;
15: **if** $\mathcal{M}(F(P^*, \tilde{\mathbf{x}}^*), D_{\text{val}}) > y^*$ **then** $y^* \leftarrow \mathcal{M}(F(P^*, \tilde{\mathbf{x}}^*), D_{\text{val}})$ and $\mathbf{x}^* \leftarrow \tilde{\mathbf{x}}^*$; **end if**
16: **until** $B/2$ budget exhausted;
17: **return** $\mathbf{x}^*$.

# Efficient two-stage HP search algorithm

AAAI-2022

## Searching process diagram



# Outline

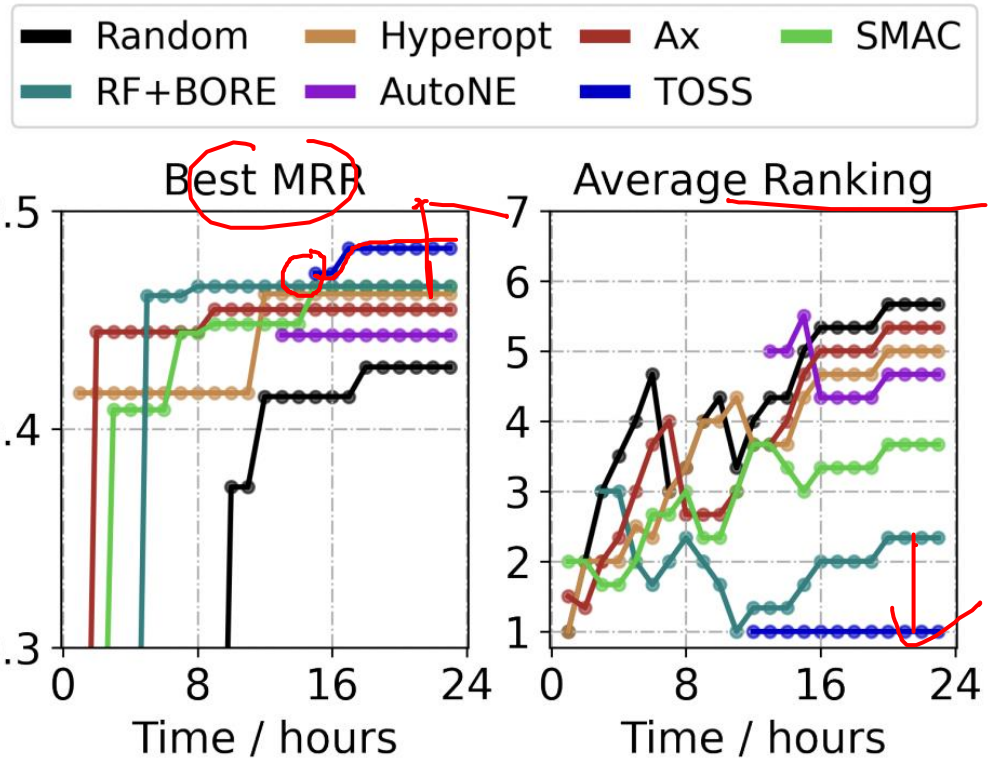
- Background
- A comprehensive understanding of HP in KGE
- An efficient two-stage HP search algorithm
- Experiments
- Key takeaway and future directions

# Experiment

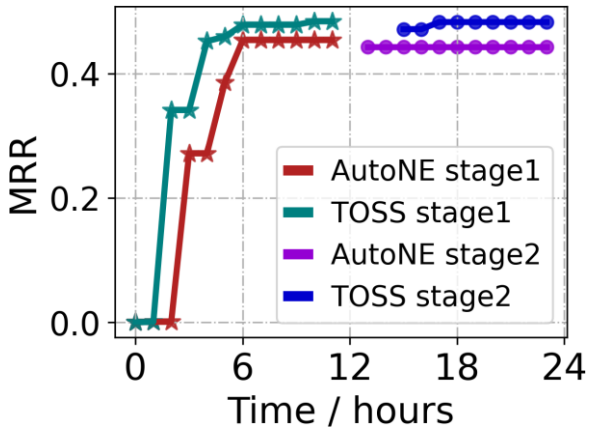
## Search algorithm comparison

**Observation**

- Random search is the worst due to the full randomness.
- SMAC and RF+BORE achieve better performance than Hyperopt and Ax since RF can fit the space better than TPE and GP.
- Due to the weak approximation and transferability, AutoNE also performs bad.
- TOSS is much better than all the baselines




| | search space reduce | search space decouple | surrogate model | fast evaluation |
|---|---|---|---|---|
| Random | × | × | × | × |
| Hyperopt | × | × | TPE | × |
| Ax | × | × | GP | × |
| SMAC | × | × | RF | × |
| RF+BORE | × | × | RF | × |
| AutoNE | × | × | GP | √ |
| TOSS | √ | √ | RF | √ |



# Experiment

AAAI-2022

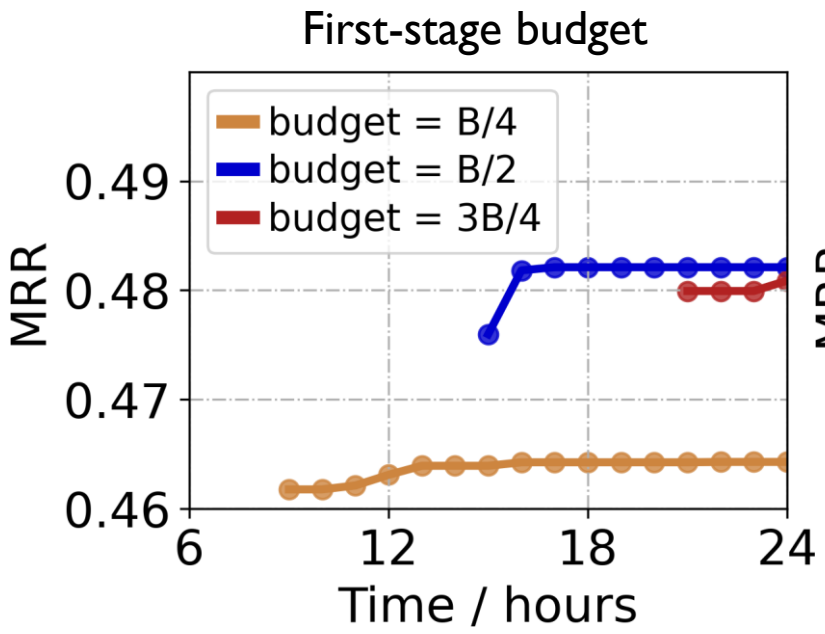
## Searched configuration performance

| models | | ogbl-biokg | ogbl-wikikg2 |
|---|---|---|---|
| OGB board | TransE | 0.7452 | 0.4256 |
| | RotatE | 0.7989 | 0.2530 |
| | DistMult | 0.8043 | 0.3729 |
| | ComplEx | 0.8095 | 0.4027 |
| TOSS | TransE | 0.7771 (4.28%↑) | 0.4745 (11.48%↑) |
| | RotatE | 0.8018 (0.36%↑) | 0.2845 (12.45%↑) |
| | DistMult | 0.8254 (2.62%↑) | 0.4866 (30.49%↑) |
| | ComplEx | 0.8383 (3.55%↑) | 0.4898 (21.62%↑) |
| average improvement | | 2.70% | 19.01% |

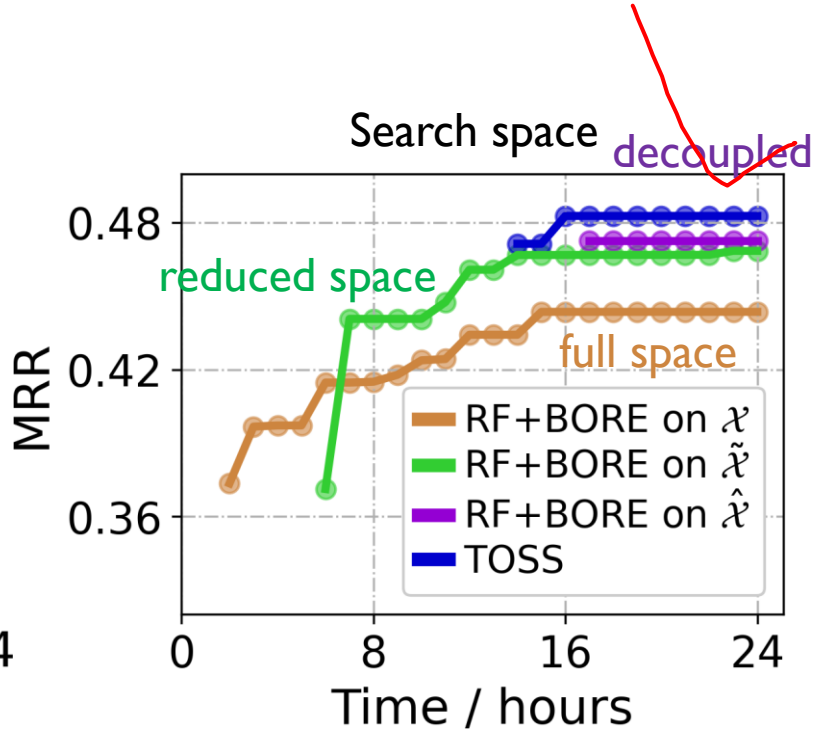
| | | WN18RR | | | | FB15k-237 | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | MRR | Hit@1 | Hit@3 | Hit@10 | MRR | Hit@1 | Hit@3 | Hit@10 |
| Original | ComplEx | 0.440 | 0.410 | 0.460 | 0.510 | 0.247 | 0.158 | 0.275 | 0.428 |
| | DistMult | 0.430 | 0.390 | 0.440 | 0.490 | 0.241 | 0.155 | 0.263 | 0.419 |
| | RESCAL | 0.420 | - | - | 0.447 | 0.270 | - | - | 0.427 |
| | ConvE | 0.430 | 0.400 | 0.440 | **0.520** | 0.325 | 0.237 | 0.356 | 0.501 |
| | TransE | 0.226 | - | - | 0.501 | 0.294 | - | - | 0.465 |
| | RotatE | 0.476 | **0.428** | 0.492 | 0.571 | 0.338 | 0.241 | **0.375** | **0.533** |
| | TuckER | 0.470 | **0.443** | 0.482 | 0.526 | **0.358** | **0.266** | **0.394** | **0.544** |
| LibKGE (Ruffinelli et al., 2019) | ComplEx | 0.475 | 0.438 | 0.490 | 0.547 | 0.348 | 0.253 | 0.384 | **0.536** |
| | DistMult | 0.452 | **0.413** | 0.466 | 0.530 | 0.343 | 0.250 | **0.378** | **0.531** |
| | RESCAL | 0.467 | **0.439** | 0.480 | 0.517 | 0.356 | 0.263 | 0.393 | **0.541** |
| | ConvE | **0.442** | **0.411** | **0.451** | 0.504 | **0.339** | **0.248** | **0.369** | 0.521 |
| | TransE | 0.228 | **0.053** | 0.368 | 0.520 | 0.313 | 0.221 | 0.347 | 0.497 |
| TOSS (ours) | ComplEx | **0.483** | **0.439** | **0.501** | **0.564** | **0.355** | **0.266** | **0.389** | 0.533 |
| | DistMult | **0.453** | 0.406 | **0.468** | **0.545** | **0.344** | **0.253** | 0.377 | 0.525 |
| | RESCAL | **0.478** | 0.434 | **0.499** | **0.559** | **0.359** | **0.272** | **0.395** | 0.532 |
| | ConvE | 0.435 | 0.405 | 0.444 | 0.500 | 0.330 | 0.237 | 0.362 | **0.522** |
| | TransE | **0.232** | 0.033 | **0.394** | **0.540** | **0.328** | **0.229** | **0.369** | **0.523** |
| | RotatE | **0.479** | 0.426 | **0.499** | **0.585** | **0.339** | **0.246** | 0.372 | 0.527 |
| | TuckER | **0.480** | 0.437 | **0.501** | **0.556** | 0.347 | 0.255 | 0.382 | 0.535 |

# Experiment | Ablation study

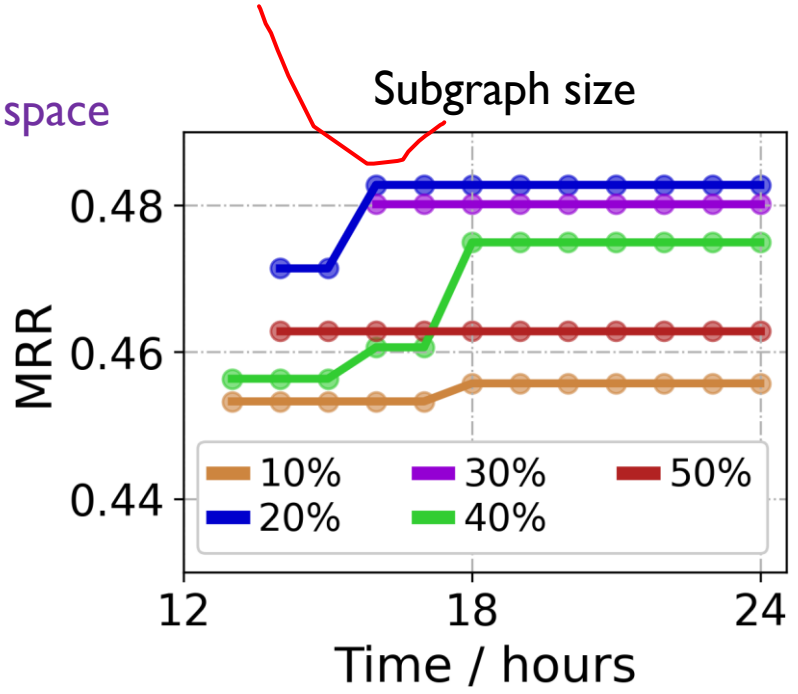





budget = B/2 👍




Subgraph + Decouple 👍




Subgraph size = 20%/30% 👍

# Experiment

## Searched optimal configurations

*Table 9: Searched optimal hyperparameters for the WN18RR dataset.*

| HP/Model | ComplEx | DistMult | RESCAL | ConvE | TransE | RotatE | TuckER |
|---|---|---|---|---|---|---|---|
| # negative samples | 512 | 128 | 128 | 1VsAll | 128 | 2048 | 128 |
| loss function | BCE_adv | BCE_adv | BCE_mean | BCE_sum | CE | BCE_adv | CE |
| gamma | 0.00 | 0.00 | 0.00 | 0.00 | 6.00 | 3.10 | 0.00 |
| adv. weight | 0.57 | 1.41 | 0.00 | 0.00 | 0.00 | 1.93 | 0.00 |
| regularizer | DURA | NUC | DURA | FRO | FRO | FRO | DURA |
| reg. weight | $8.64 * 10^{-3}$ | $9.58 * 10^{-3}$ | $1.76 * 10^{-3}$ | $1.00 * 10^{-4}$ | $1.00 * 10^{-4}$ | $6.51 * 10^{-6}$ | $1.42 * 10^{-3}$ |
| dropout rate | 0.25 | 0.29 | 0.00 | 0.00 | 0.20 | 0.00 | 0.00 |
| optimizer | Adam | Adam | Adam | Adam | Adam | Adam | Adam |
| learning rate | $1.77 * 10^{-3}$ | $4.58 * 10^{-3}$ | $1.73 * 10^{-3}$ | $1.00 * 10^{-3}$ | $1.00 * 10^{-3}$ | $6.43 * 10^{-4}$ | $1.37 * 10^{-3}$ |
| initializer | xavier_norm | norm | uniform | uniform | uniform | norm | uniform |
| batch size | 512 | 1024 | 512 | 1024 | 512 | 512 | 512 |
| dimension size | 1000 | 2000 | 1000 | 2000 | 1000 | 1000 | 200 |
| inverse relation | False | False | False | False | False | False | False |

Limited range

Monotonously

*Table 10: Searched optimal hyperparameters for the FB15k-237 dataset.*

| HP/Model | ComplEx | DistMult | RESCAL | ConvE | TransE | RotatE | TuckER |
|---|---|---|---|---|---|---|---|
| # negative samples | 512 | kVsAll | 2048 | 512 | 512 | 2048 | 2048 |
| loss function | BCE_adv | CE | CE | BCE_sum | BCE_adv | BCE_adv | BCE_adv |
| gamma | 0.00 | 0.00 | 0.00 | 0.00 | 6.76 | 7.58 | 0.00 |
| adv. weight | 1.93 | 0.00 | 0.00 | 0.00 | 1.99 | 1.57 | 1.94 |
| regularizer | DURA | FRO | DURA | DURA | FRO | DURA | DURA |
| reg. weight | $9.75 * 10^{-3}$ | $1.00 * 10^{-4}$ | $9.01 * 10^{-3}$ | $6.42 * 10^{-3}$ | $2.16 * 10^{-3}$ | $5.12 * 10^{-3}$ | $1.47 * 10^{-4}$ |
| dropout rate | 0.22 | 0.30 | 0.00 | 0.08 | 0.03 | 0.02 | 0.02 |
| optimizer | Adam | Adam | Adam | Adam | Adam | Adam | Adam |
| learning rate | $9.70 * 10^{-4}$ | $1.00 * 10^{-3}$ | $1.19 * 10^{-3}$ | $2.09 * 10^{-3}$ | $2.66 * 10^{-4}$ | $2.98 * 10^{-4}$ | $3.19 * 10^{-4}$ |
| initializer | uniform | normal | xavier_norm | normal | xavier_norm | uniform | normal |
| batch size | 1024 | 1024 | 512 | 1024 | 512 | 512 | 512 |
| dimension size | 2000 | 2000 | 500 | 500 | 1000 | 1000 | 500 |
| inverse relation | False | False | False | False | False | False | False |

Fixed choice

# Outline

- Background
- A comprehensive understanding of HP in KGE
- An efficient two-stage HP search algorithm
- Experiments
- Key takeaway and future directions

# Key takeaways



| Recall the difficulties | KGbench |
| --- | --- |
| Lacking understanding of KGE components | A comprehensive understanding of HPs |
| Low efficiency in searching for hyperparameter | An efficient two-stage HP search algorithm |

# Limitation and future directions

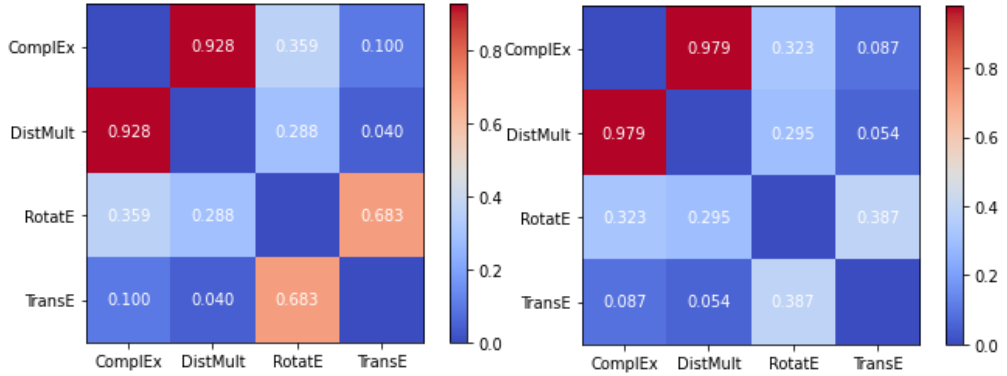


## Limitation

- Limited to pure embedding models
- Not considering HPs inside the SF model
- Lacking of theoretical analysis and guarantees

## Potential directions

- apply with GNN to solve the scaling problem
- combine HPO with NAS
- transferability across datasets/models/tasks




WN18RR FB15k-237

# Q&A

Thanks for your attention!