AutoNE: Hyperparameter Optimization for Massive Network Embedding

Ke Tu
Tsinghua U

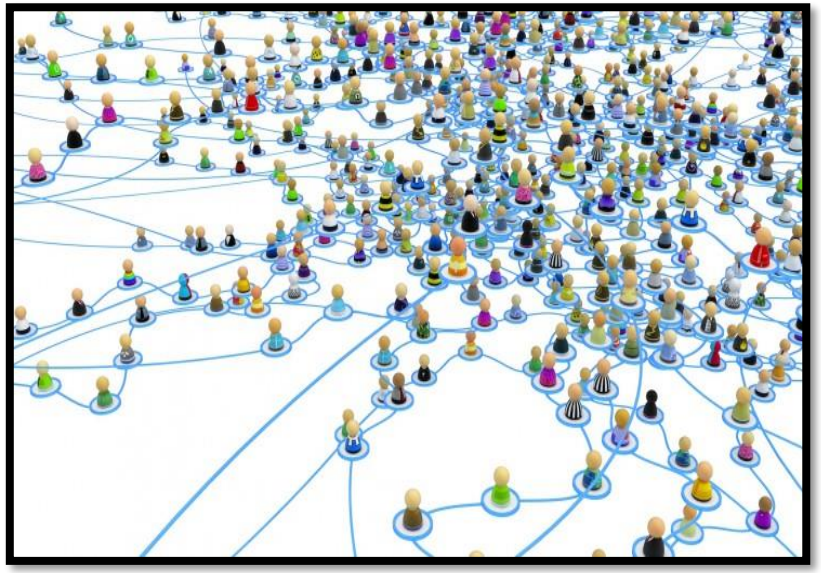
Jianxin Ma
Tsinghua U

Peng Cui
Tsinghua U

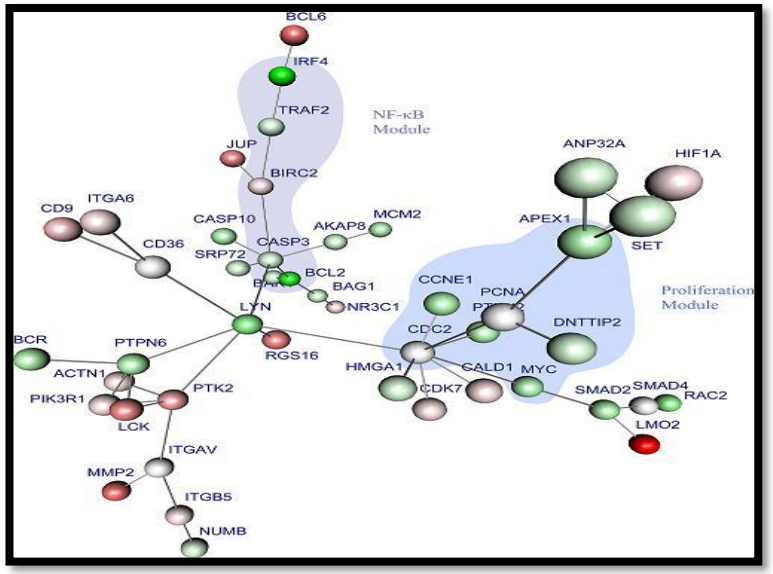
Jian Pei
SFU&JD

Wenwu Zhu
Tsinghua U

Network Analytics



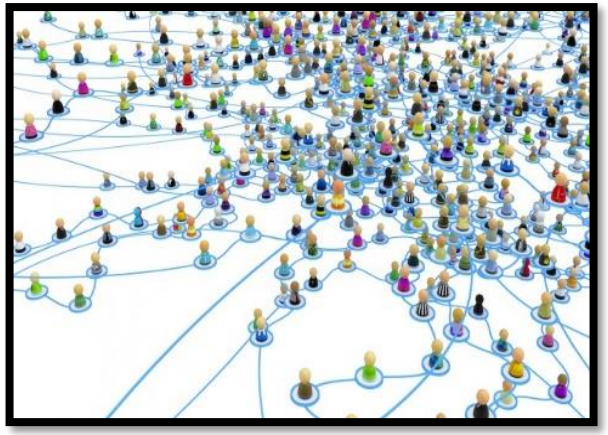
Social Networks



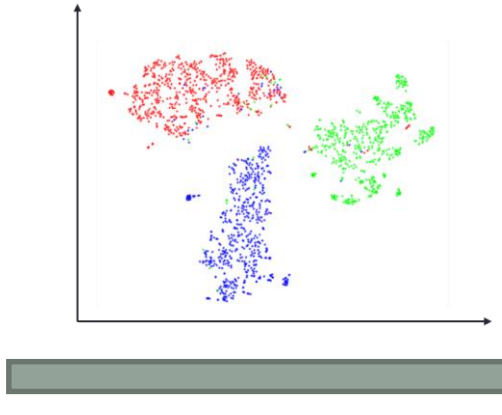
Biology Networks

Networks are widely used to represent the rich pairwise relationships of data objects

Network Embedding



Origin network

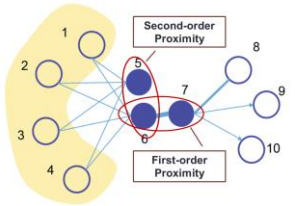


- in vector space

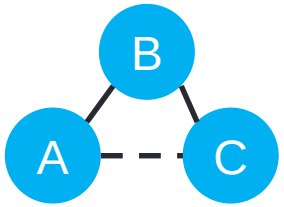
- Link Prediction
- Community Detection
- Node Classification
- Network Distance
- Node Importance
- …

Networks embedding aims to learn a low-dimensional representation for each node

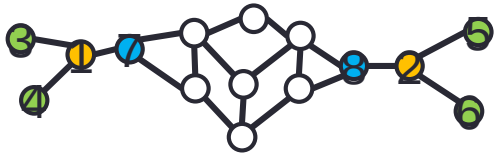
Existing Embedding Methods



High-order



Transitivity



Global position

Various network properties

- Link Prediction
- Community Detection
- Node Classification
- Network Distance
- Node Importance
- …

Various applications

- Leading to a large number of hyperparameters
  - E.g. Deepwalk: number of walks, walk length, window size
- Must be carefully tuned

AutoML

AutoML

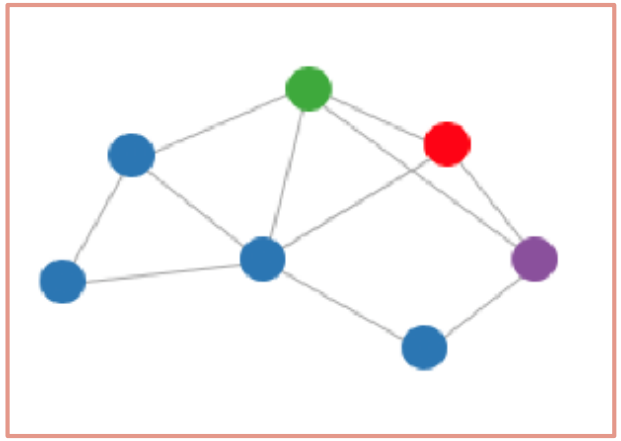
- Ease the adoption of machine learning and reduce the reliance on human experts
  - e.g., hyperparameter optimization

- Network data remains largely unexplored
- Large scale issue:
  - Complexity of Network Embedding is usually at least $O(E)$
    - E is the number of edges (can be 10 billion)
  - Total complexity: $O(E\mathrm{T})$, T is the times searching for optimal hyperparameter

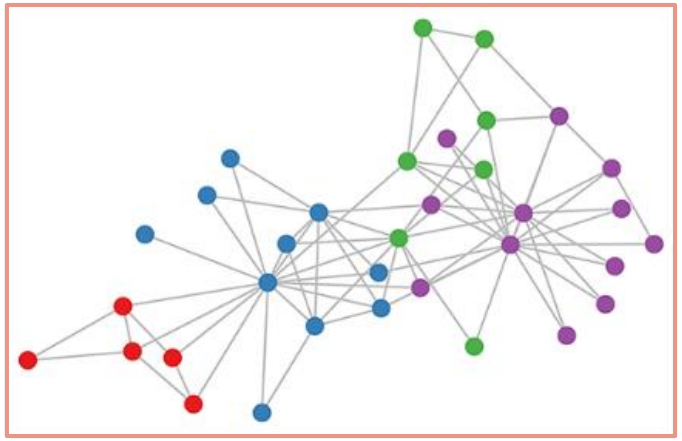
How to incorporate AutoML into massive network embedding efficiently? (reduce E and T)

incorporating AutoML into NE

- A straightforward way: configuration selection on sampled sub-networks



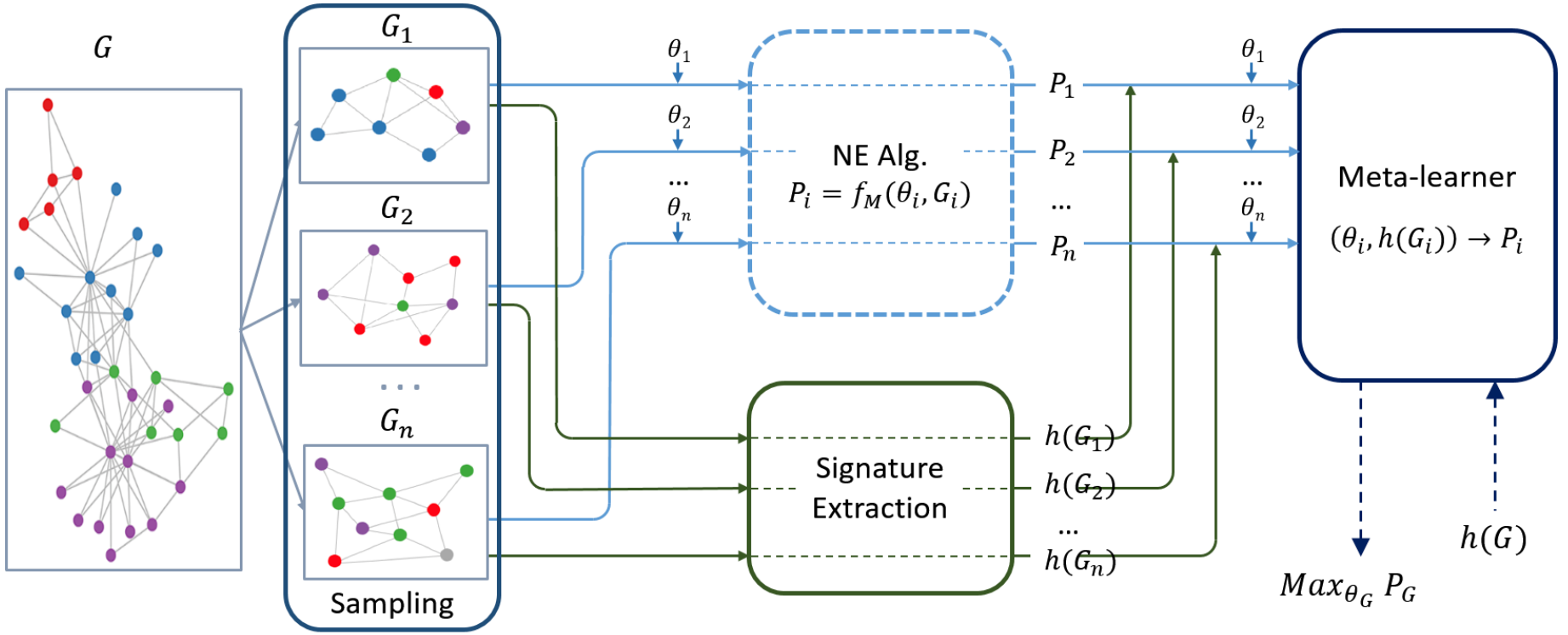
Sampled sub-network
Optimal configuration θ^*



Origin massive network
using θ^*

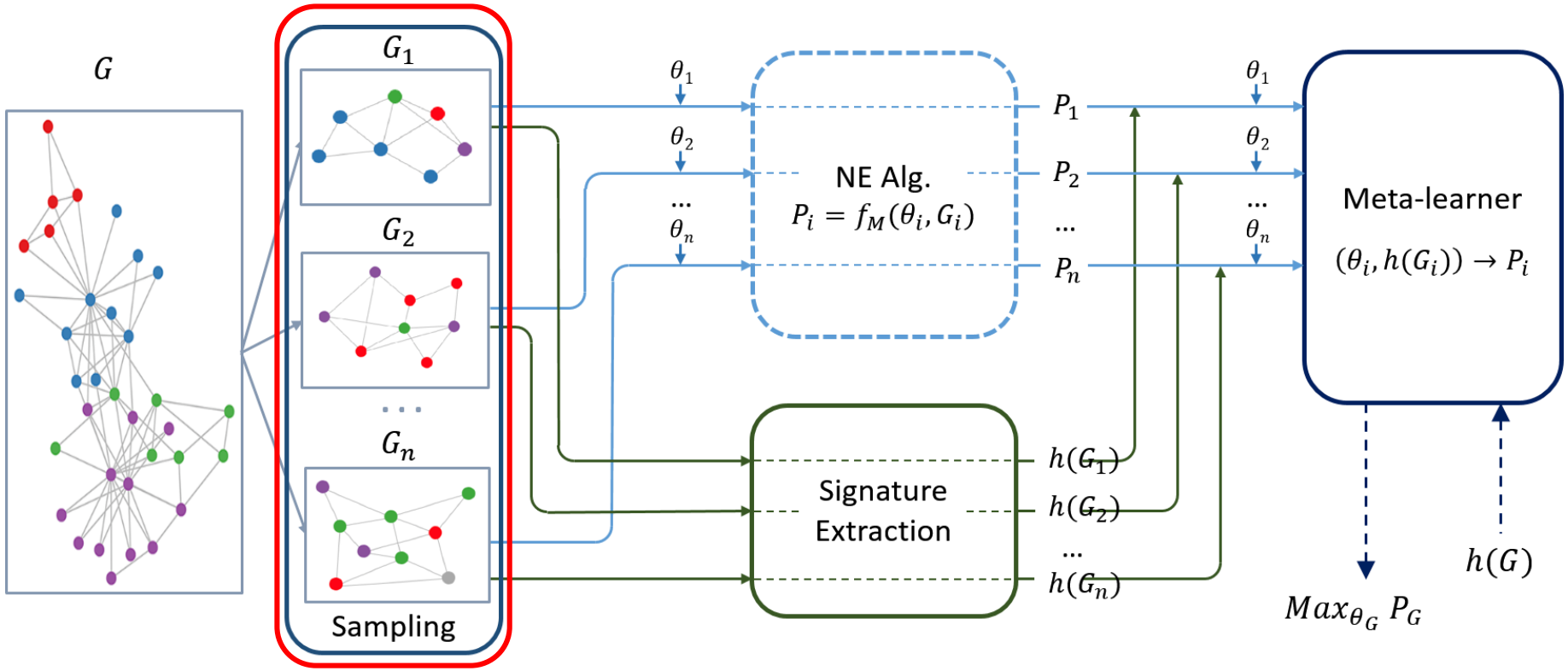
- **Transferability**
  - $\theta \neq$ optimal configuration on origin network
- **Heterogeneity**
  - several highly heterogeneous components => carefully designed sampling

AutoNE



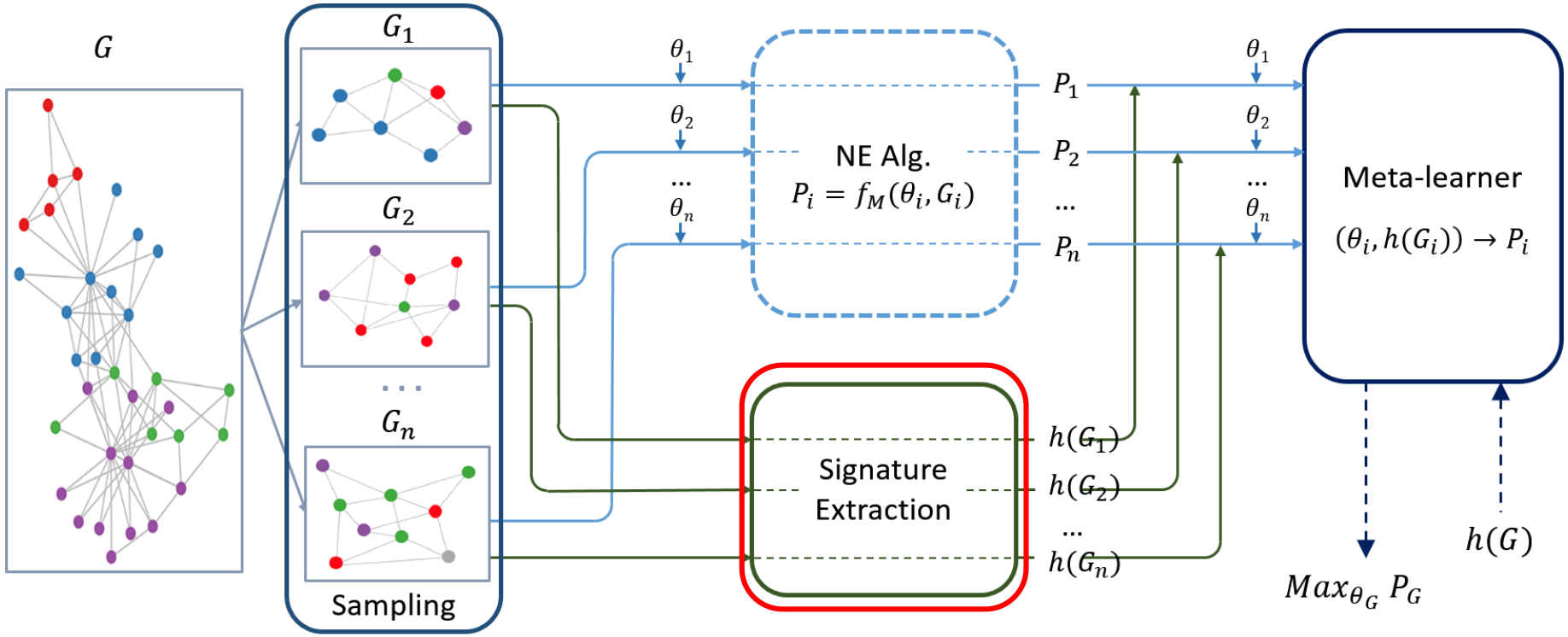
Transfer the knowledge about optimal hyperparameters from the sub-networks to the original massive network

AutoNE



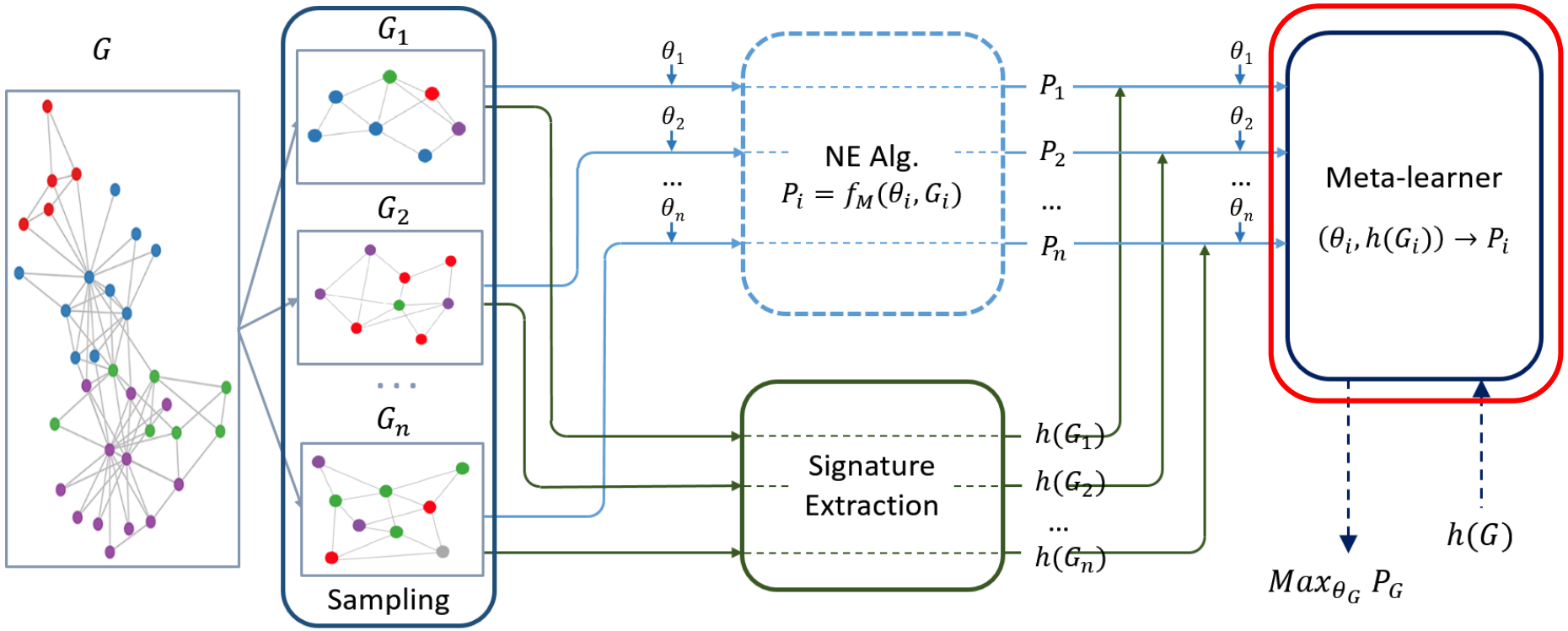
Transfer the knowledge about optimal hyperparameters from the sub-networks to the original massive network

AutoNE



Transfer the knowledge about optimal hyperparameters from the sub-networks to the original massive network

AutoNE



Transfer the knowledge about optimal hyperparameters from the sub-networks to the original massive network

Sampling Module

- **Goal**: Sample a series of representative sub-networks that share similar properties with the original large-scale network
- **Heterogeneity** issue: preserve diversity of the origin network

- Origin network $G = (V, E)$
- Random walk w $= (v_1 \rightarrow v_2 \rightarrow \ ... \rightarrow v_n)$
- Sub-graph G_s based random walk w:
  - Nodes: $\mathrm{V}' = \{\mathrm{v}_1, v_2, \ldots, v_n\}$
  - Edges: $\mathrm{E}' = \{(v_i, v_j) | \ i, j \in [1, n], \ (v_i, v_j) \in E\}$

- The starting points v_1
  - Supervised: several nodes with different labels
  - Unsupervised: from the different communities， e.g., a greedy algorithm that maximizes modularity

Signature Extraction Module

- The signature of a network is a vector descriptor that encodes the various properties of the whole network

- Graph signature $\mathrm{h}(G)$
  - $Similarity(G_1, G_2) = Distance(h(G_1), \mathrm{h}(G_2))$

- **Challenge**: The signatures should be comprehensive enough
  - Based on spectral graph theory, a large number of network properties are decided by the spectrum of a network, e.g. the normalized cuts used by spectral clustering
  - We choose NetLSD [Tsitsulin et al. KDD18]
    - heat diffusion process on a network
    - $h_t(G) = tr(H_t) = tr(e^{-tL}) = \sum_j e^{-t\lambda_j}$

Meta-Learning Module

- Performance function $f_M(\theta, G) \rightarrow P$
- **Transferability** issue:
  - Assumption: Two similar network will has similar optimal hyperparameter on the same network embedding method
- Learning Performance function on small sampling networks, and predict on origin massive network
- We choose f_M as **Gaussian Process** like Bayesian optimization
- The log likelihood: (K is the kernel function)

$$\ln p(\mathbf{f} \mid \mathbf{X}) = -\frac{1}{2}\mathbf{f}^\top K(\mathbf{X},\mathbf{X})^{-1}\mathbf{f} - \frac{1}{2}\ln\det(K(\mathbf{X},\mathbf{X})) + constant.$$

Meta-Learning Module

- Given a new test point $x_* = (\theta_*, h(G_*))$ and the observed values f
  - The predicted performance f_* and f follow a joint normal distribution
- The posterior distribution $p(f_M(\theta_*, h(G_*) \mid x_*, f, X)$ is a normal distribution:

$$f_M(\theta_*, G_*) \mid \mathbf{x}_*, \mathbf{f}, \mathbf{X} \sim \mathcal{N}(\mu_*, \sigma_*^2),$$
$$\mu_* = K(\mathbf{x}_*, \mathbf{X})K(\mathbf{X}, \mathbf{X})^{-1}\mathbf{f},$$
$$\sigma_*^2 = K(\mathbf{x}_*, \mathbf{x}_*) - K(\mathbf{x}_*, \mathbf{X})K(\mathbf{X}, \mathbf{X})^{-1}K(\mathbf{X}, \mathbf{x}_*).$$

- Optimize θ_*: Upper confidence bound: $\arg\max_{\theta_*} \mu_* + \kappa\sigma_*$.
- Kernel function K:
  - Measure the similarity between two sets of hyperparameters and the similarity between two networks

$$k\left((\theta_1, G_1), (\theta_2, G_2)\right) = k_\theta\left(\theta_1, \theta_2\right) \cdot k_g\left(h(G_1), h(G_2)\right).$$

Kernel for hyperparameter

Kernel for graph

Experiment Setting --- datasets

Datasets	Node	Edge	Label	Feature (For GCN)
BlogCatalog	10,312	333,983	39	-
Wikipedia	4,777	184,812	40	-
Pubmed	19,717	44,338	3	500
TopCat	1,791,489	28,511,807	-	-

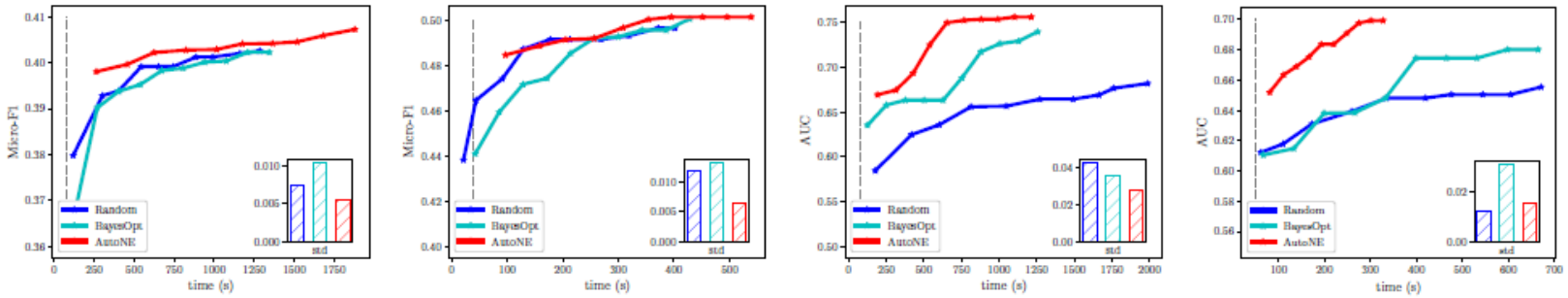
Experiment Setting --- NE

- Network Embedding method: Three classes
  - **Sampling-based NE: Deepwalk** [Perozzi, Bryan, et.al. KDD14]
    - Number of random walks
    - Length of each random walk
    - Windows size
  - **Factorization based NE: AROPE** [Zhang, Ziwei, et al. KDD18]
    - The weights of the different orders(Max 4)
  - **Deep neural network-based NE: GCN** [Thomas N. Kipf, et al. ICLR17]
    - Learning rate
    - Size of each hidden layer
    - Number of training epochs
    - The dropout rate
    - The weight decay

Experiment Setting --- Baselines and Tasks

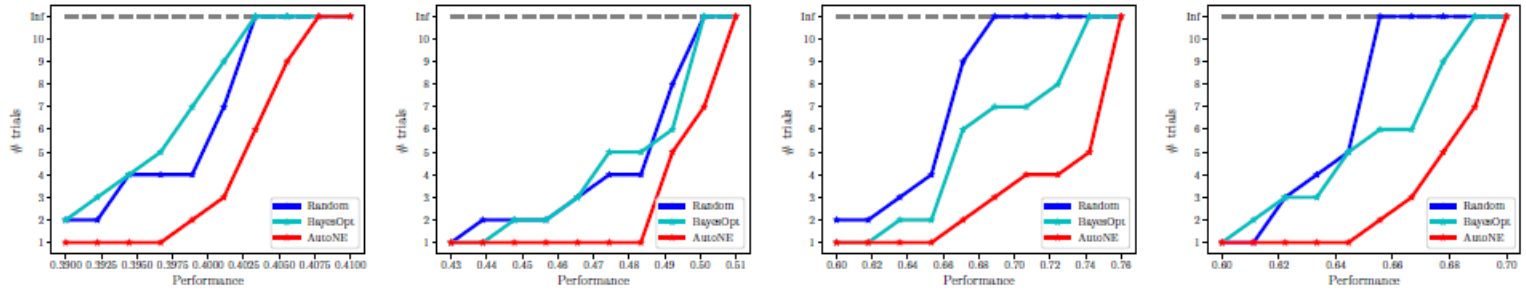
- Baselines
  - **Random search**
    - find the optimal solution as long as the time budget is large enough
    - Instead of grid search
      - explore larger configuration space more efficiently
      - find better solutions faster
  - **Bayesian optimization (BayesOpt)**
    - the most used method in AutoML framework
    - performs many trials on the original data
      - makes it inefficient at handling large-scale networks.
- Tasks
  - Link prediction and node classification

Experiment --- Sampling-Based NE



(a) Classification on BlogCatalog (b) Classification on Wikipedia (c) Link prediction on BlogCatalog (d) Link prediction on Wikipedia

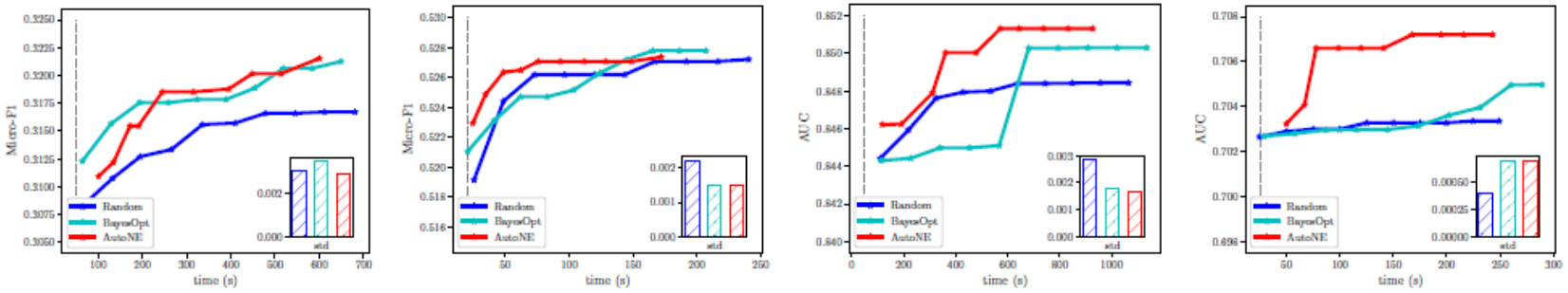
The performance achieved within various time thresholds.



(a) Classification on BlogCatalog (b) Classification on Wikipedia (c) Link prediction on BlogCatalog (d) Link prediction on Wikipedia

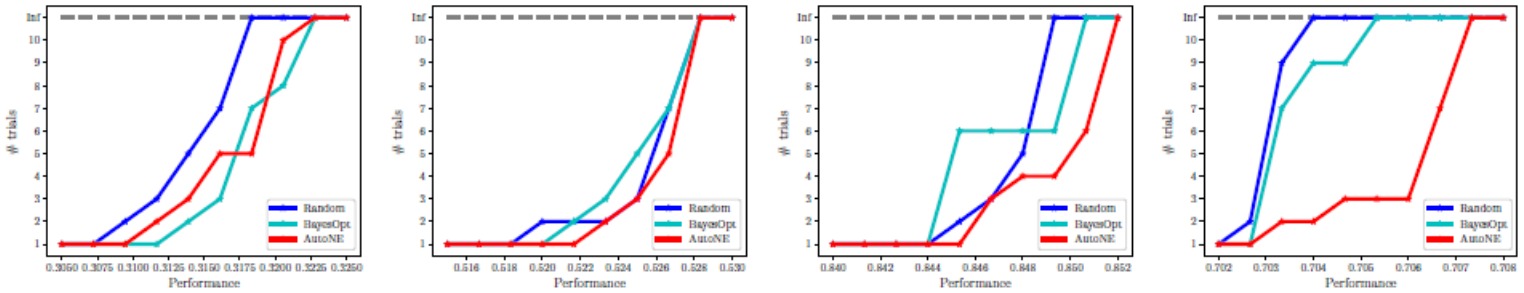
The number of trials to reach a certain performance threshold

Experiment---Factorization-Based NE



(a) Classification on BlogCatalog (b) Classification on Wikipedia (c) Link prediction on BlogCatalog (d) Link prediction on Wikipedia

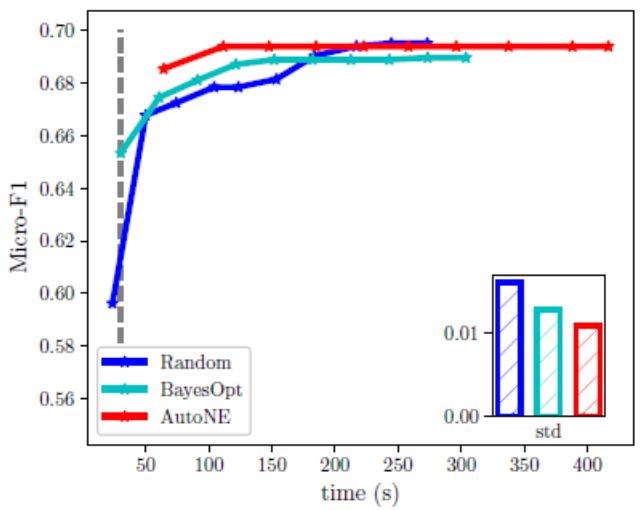
The performance achieved within various time thresholds.



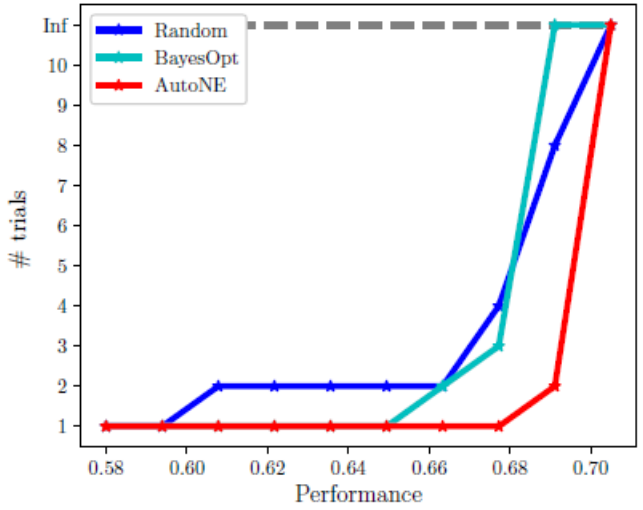
(a) Classification on BlogCatalog (b) Classification on Wikipedia (c) Link prediction on BlogCatalog (d) Link prediction on Wikipedia

The number of trials to reach a certain performance threshold

Experiment --- Deep NN-Based NE



(a) The performance achieved by each method within various time thresholds.



(b) The number of trials required to reach a certain performance threshold.

Node classification on Pubmed.

Experiment --- Large-Scale

Table 1: Results on a massive network with around thirty million edges, where we can only afford to run a NE algorithm on the whole network for a few times.

Method	Trial 1		Trial 2		Trial 3	
	AUC	Time(s)	AUC	Time(s)	AUC	Time(s)
AutoNE	0.717	1067.9	0.726	1856.2	**0.769**	2641.9
Random	0.714	698.3	0.727	1426.3	0.715	2088.6
BayesOpt	0.715	702.5	0.714	1405.1	0.727	2307.7

Summary

- Investigate the pressing problem of incorporating AutoML into NE
- Propose a novel framework AutoNE
  - Automate hyperparameter optimization for NE.
- Can scale up to massive real-world networks
  - Utilizing the meta-knowledge transferred from sampled sub-networks.
- Extensive experiment on real-world networks
  - Four real-world dataset
  - Three representative network embedding methods
  - Two baselines

Thanks!

Ke Tu, Tsinghua University

tuke1993@gmail.com

AutoNE: Hyperparameter Optimization for
Massive Network Embedding