AliGraph: An Extremely Large Scale Graph Representation Learning in Practice

Xiaoyong Liu
Hongxia Yang
Wei Lin

Alibaba Group

Part 1：About Alibaba

Our Mission

Alibaba Group 阿里巴巴集团

Our Vision

Alibaba Group 阿里巴巴集团

MEET

Enabling millions of commercial and social interactions

WORK

Empowering our customers with data and infrastructure to manage their business

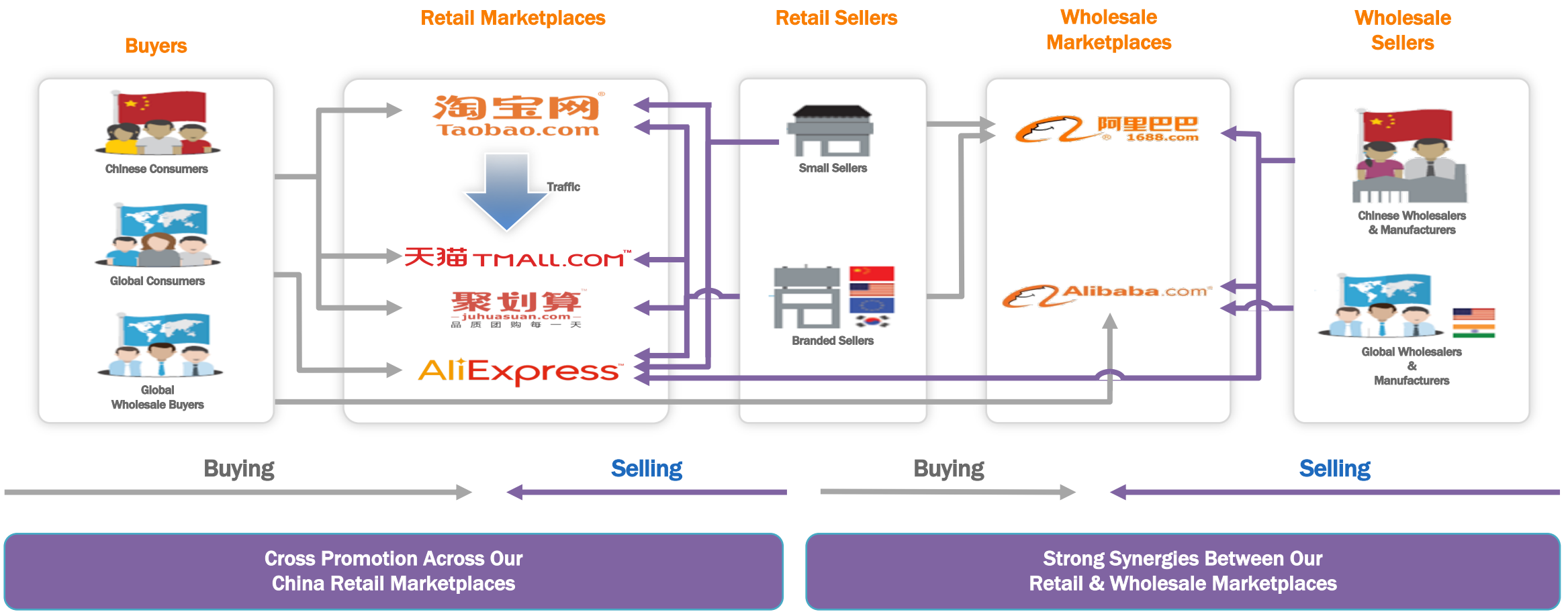
LIVE

To become central to the everyday lives of our customers

@ Alibaba

Alibaba Ecosystem

Our Marketplaces are Networked



Part 2：Why GE and Two works

Why Graph and Graph Embedding?

Alibaba Group 阿里巴巴集团

- Graph computing models are very popular in big data companies, especially IT companies, as they are the most straightforward solutions to many practical problems
- Traditional recommendation and CTR/CVR prediction problems can be equivalently modeled with the attributed user-item bipartite graphs
- Objective functions are more general: global optimization vs conditional independent optimization
- Introducing high-level proximity samples and their modeling brings both risks (e.g., noise) and benefits (e.g., more generalization and exploration, predictive graph changes)
- Pure deep learning is mature, graph embedding that combines both deep learning and graph computing integrating end-to-end learning with inductive reasoning, which is expected to solve the relational reasoning that deep learning cannot perform[1].

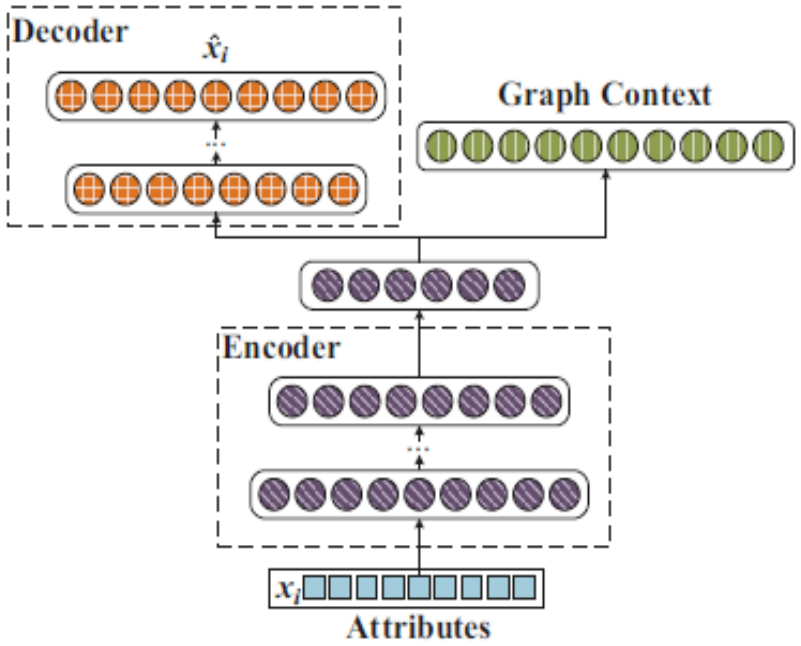
[1] Relational inductive biases, deep learning and graph networks, Battaglia etc, arxiv, 2018.

Representative Work: Graph Embedding in Fraud Detection[1]

Alibaba Group 阿里巴巴集团

Challenges of mobile fraud detection:

- Billions of mobile logging records per day
- Device ids are missing
- Devices are abnormal:
  - Dual sim cards dual standby: one device has two imei, imsi[2] with random switching
  - Replace the sim card: one device corresponds to multiple imsi
  - System Restore: One device corresponds to multiple device ids
  - Simulator: One device corresponds to a large number of device ids
  - Cottage machine: a large number of devices share one same id
- Devices contain rich attributes



We propose a distributed deep learning-based representation learning model, ANRL, which maps nodes in the graph to representation vectors in low-dimensional space to facilitate subsequent processing of related tasks

- Combining both network structure and node attribute information to better maintain network characteristics
- Neural networks can mine deeper correlations between the two

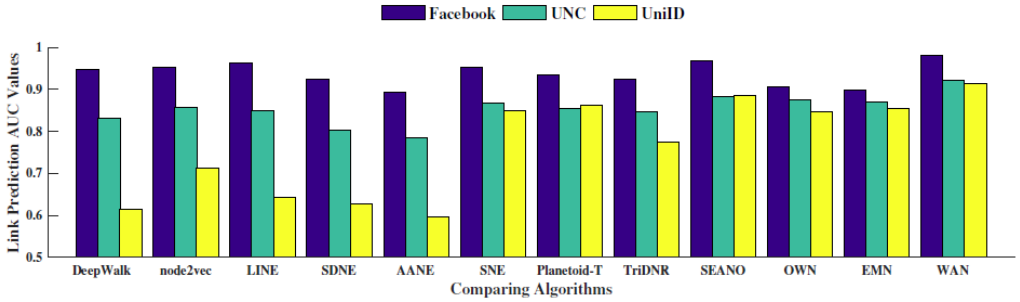
[1] **ANRL: Attributed Network Representation Learning via Deep Neural Networks, IJCAI 2018**
[2] **imei: international mobile equipment identity; imsi: international mobile subscriber identity**

Representative Work: Graph Embedding in Fraud Detection

Alibaba Group 阿里巴巴集团

- Objective Functions:

$$
\begin{aligned}
\mathcal{L} &= \mathcal{L}_{sg} + \alpha\mathcal{L}_{ae} + \beta\mathcal{L}_{reg} \\
&= -\sum_{i=1}^{n}\sum_{c\in C}\sum_{-b\le j\le b, j\neq 0} \log\frac{\exp(\mathbf{u}_{i+j}^{\mathrm{T}}\mathbf{y}_i^{(K)})}{\sum_{v=1}^{n}\exp(\mathbf{u}_v^{\mathrm{T}}\mathbf{y}_i^{(K)})} \\
&+ \alpha\sum_{i=1}^{n}\|\hat{\mathbf{x}}_i - T(v_i)\|_2^2 + \frac{\beta}{2}\sum_{k=1}^{K}(\|\mathbf{W}^{(k)}\|_F^2 + \|\hat{\mathbf{W}}^{(k)}\|_F^2)
\end{aligned}
$$



- Existing methods generally incorporate attribute information into representation learning by matrix decomposition or by adding auxiliary nodes. This shallow level model can't capture the deep inner relationship between the two.

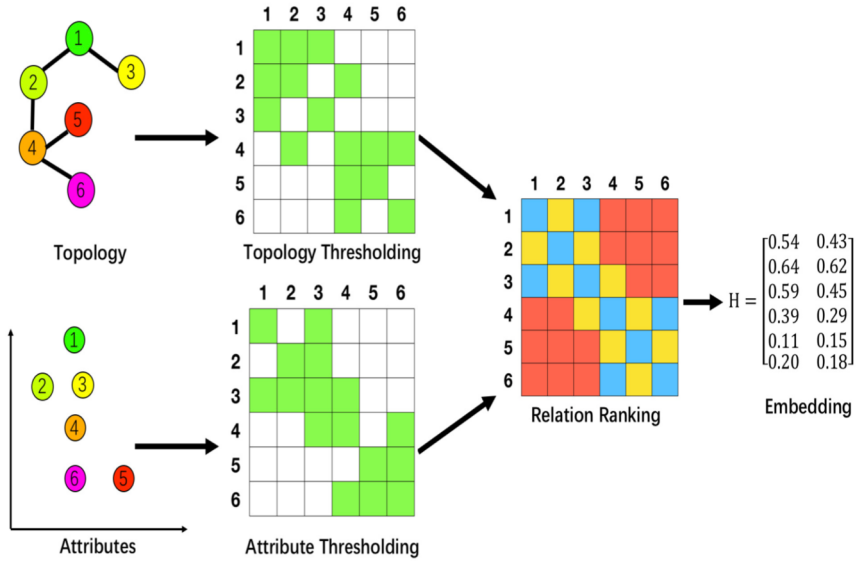
- Compared with the existing state-of-the-art method, our model ANRL has a certain degree of improvement in tasks such as ink prediction and node classification by using the deep learning model.

Datasets	Citeseer		Pubmed		Fraud Detection	
Evaluation	Micro-F1	Macro-F1	Micro-F1	Macro-F1	Micro-F1	Macro-F1
SVM	0.667	0.626	0.856	0.855	0.725	0.719
autoencoder	0.630	0.565	0.792	0.800	0.732	0.726
DeepWalk	0.583	0.534	0.809	0.795	0.509	0.464
node2vec	0.607	0.561	0.815	0.802	0.571	0.519
LINE	0.542	0.512	0.766	0.749	0.659	0.654
SDNE	0.569	0.528	0.699	0.677	0.662	0.656
AANE	0.579	0.541	0.784	0.765	0.654	0.643
SNE	0.632	0.615	0.803	0.797	0.662	0.654
Planetoid-T	0.656	0.594	0.851	0.847	0.692	0.693
TriDNR	0.633	0.587	0.843	0.824	0.686	0.685
SEANO	0.713	0.662	0.859	0.848	0.703	0.704
ANRL-OWN	0.652	0.606	0.842	0.845	0.724	0.720
ANRL-EMN	0.716	0.668	0.865	0.867	0.733	0.731
ANRL-WAN	**0.729**	**0.673**	**0.876**	**0.871**	**0.759**	**0.755**

Representative Work: Graph Embedding in Entity Recognition[1]

Challenges of mobile entity recognition:

- Data contains large-scale sparse network structure information and rich attribute information
- Both information can be used to describe the relationship among the nodes, however distance measures of the two are probably not monotonically increasing or decreasing of the same nodes
- Consider a non-positive relationship, we can capture more information to help improve the learning performance



We propose to investigate attributed network embedding through taking uncorrelation between topology structure and attribute into account and make the following contributions.

- We recognize the uncorrelation phenomenon between topology structure and attribute which affects the actual proximity among nodes in the embedding space.
- We propose a Personalized node Relation Ranking Embedding (PRRE) model such that the resulting network embedding can capture the uncorrelation as well as preserve good proximity among nodes.
- Sampling with Mini-batch Gradient Descent for efficient iteration and update.

[1] **PRRE: Personalized Relation Ranking Embedding for Attributed Networks, CIKM 2018**

Representative Work: Graph Embedding in Entity Recognition

Alibaba Group 阿里巴巴集团

- Objective function and algorithm framework

$$\mathcal{J}(H,\theta_T,\theta_A)=\prod_{i\in\mathcal{V}}\left(\prod_{p\in P}\prod_{a\in A}P(p\geq_i a|\theta_A,\theta_T,H)\prod_{a\in A}\prod_{\prod_{n\in N}}P(a\geq_i n|\theta_A,\theta_T,H)\right).$$

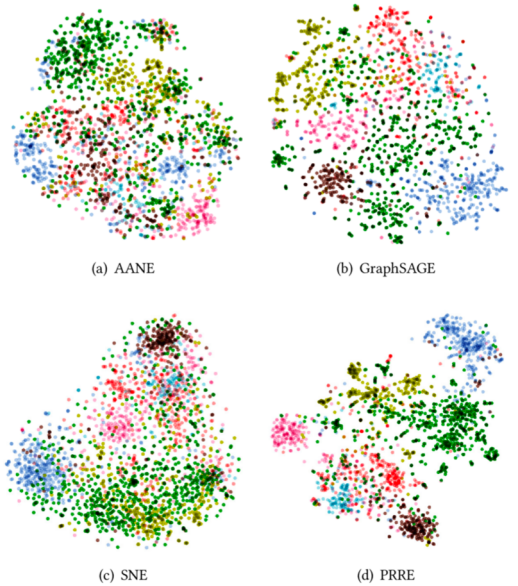
$$\mathcal{J}(H,\theta_T,\theta_A)=ln\mathcal{J}(H,\theta_T,\theta_A)-\lambda_h\sum_{i\in\mathcal{V}}||h_i||^2$$

$$=\sum_{i\in\mathcal{V}}\left(\sum_{p\in P}\sum_{a\in A}\frac{1}{1+g(\theta_T,\theta_A)}ln\frac{\sigma_{ip}-\sigma_{ia}+1}{2}+\sum_{a\in A}\sum_{n\in N}\frac{1}{1+g(\theta_T,\theta_A)}ln[\frac{\sigma_{ia}-\sigma_{in}+1}{2}]\right)-\lambda_h\sum_{i\in\mathcal{V}}||h_i$$

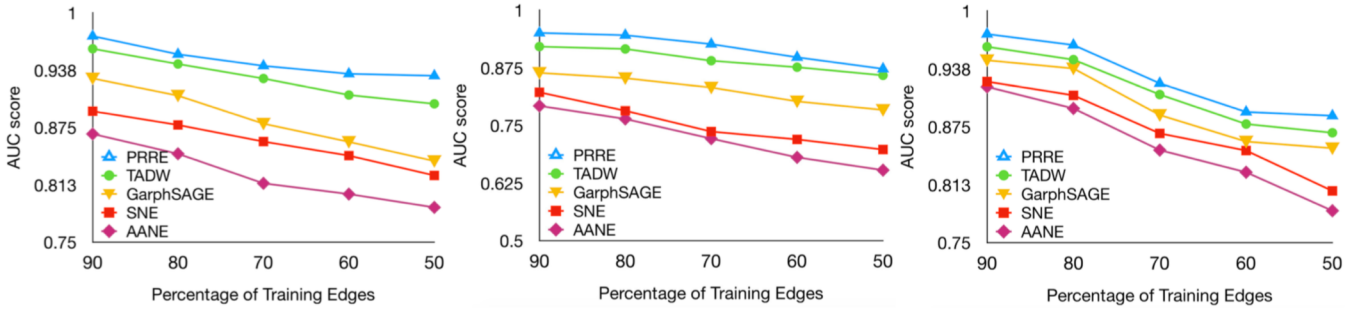
Algorithm 1 PRRE for attributed networks

Input: $G=\{\mathcal{V},E,A\},d$
Output: $H\in R^{n\times d}$

1: Compute similarity matrix $S_A\in R^{n\times n}$ and $S_T\in R^{n\times n}$ using selected similarity measure
2: Initialize θ_A,θ_T and $H\sim U(0,1)$
3: **while** t < max_iter and $\triangle\mathcal{J}<\epsilon$ **do**
4: Sample batch B with size s,
5: **for** $v_i\in\mathcal{V}$ **do**
6: Compute Positive/Ambiguous/Negative pairs using current thresholds θ_T,θ_A
7: **end for**
8: Compute the gradients for h_i,h_p,h_a,h_n by Equation [13], [14], [15], [16].
9: Update H by Equation [19]
10: Compute the gradients for θ_T,θ_A by Equation [17],[18].
11: Update θ_T,θ_A by Equation [19]
12: **end while**



Nodes are mapped into the 2-D space using t-SNE with learned embeddings



Other Graph Embedding Representative Works

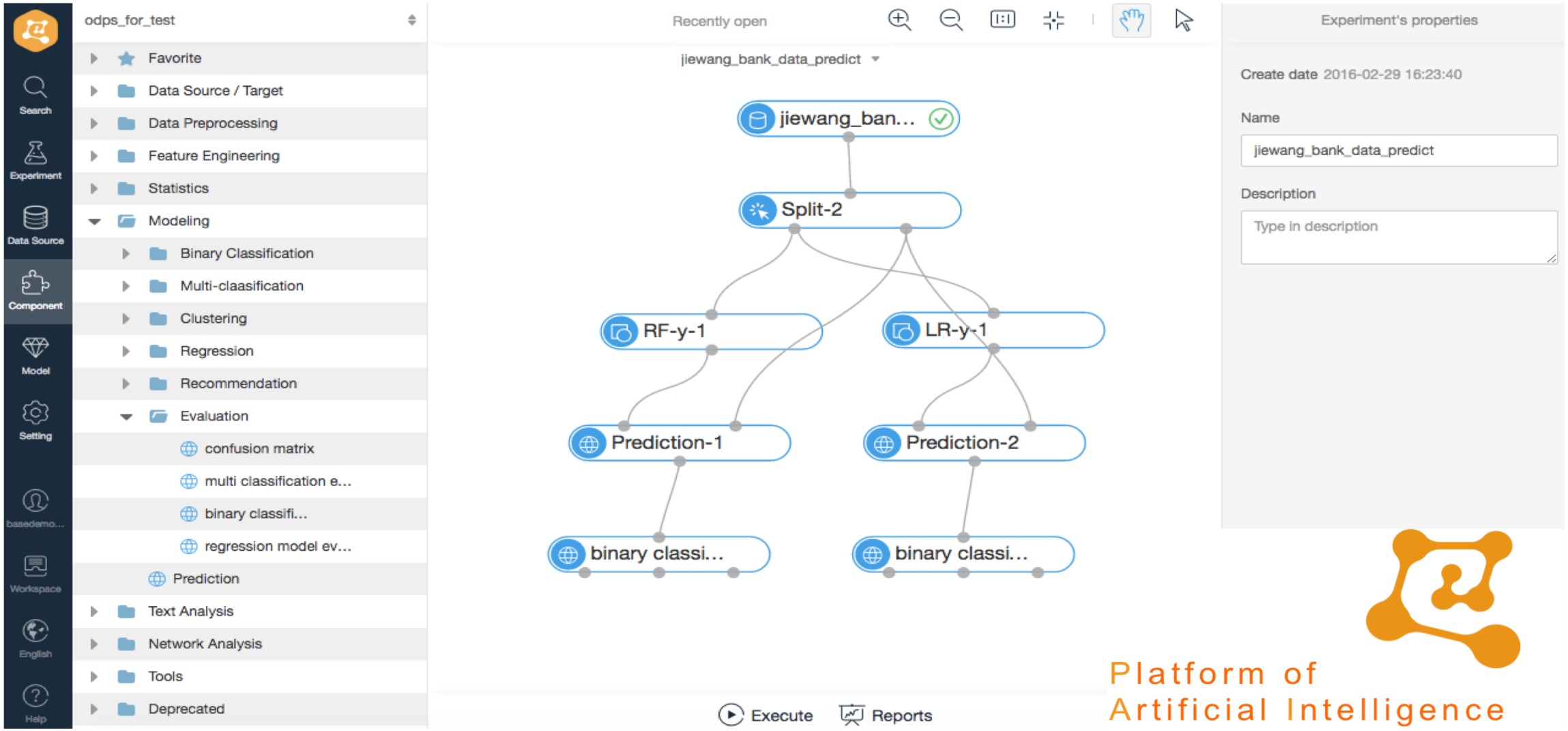
Alibaba Group 阿里巴巴集团

1. Subgraph-augmented Path Embedding for Semantic User Search on Heterogeneous Social Network, WWW, 2018.
2. ANRL: Attributed Network Representation Learning via Deep Neural Networks, IJCAI, 2018.
3. Adversarial Detection with Model Interpretation. **KDD, 2018.**
4. SPARC: Self-Paced Network Representation for Few-Shot Rare Category Characterization. **KDD, 2018.**
5. Mobile access record resolution on large-scale identifier-linkage graphs. **KDD, 2018.**
6. Interactive Paths Embedding for Semantic Proximity Search on Heterogeneous Graphs. **KDD, 2018.**
7. PRRE: Personalized Relation Ranking Embedding for Attributed Network. 27th ACM International Conference on Information and Knowledge Management (CIKM), 2018.
8. Heterogeneous Embedding Propagation for Large-scale E-Commerce User Alignment, 2018 IEEE International Conference on Data Mining (ICDM), 2018
9. Local Algorithm for User Action Prediction Towards Display Ads. 23rd ACM SIGKDD Conference on Knowledge Discovery and Data Mining **(KDD), 2017**.
10. Hybrid Framework for Text Modeling with Convolutional RNN. 23rd ACM SIGKDD Conference on Knowledge Discovery and Data Mining **(KDD), 2017**.
11. Bayesian Heteroscedastic Matrix Factorization for Conversion Rate Prediction. 26th ACM International Conference on Information and Knowledge Management (CIKM), 2017.
12. Will Triadic Closure Strengthen Ties in Social Networks?, ACM Transactions on Knowledge Discovery from Data (TKDD), 2017.

Part 3：PAI - AliGraph Algorithm System

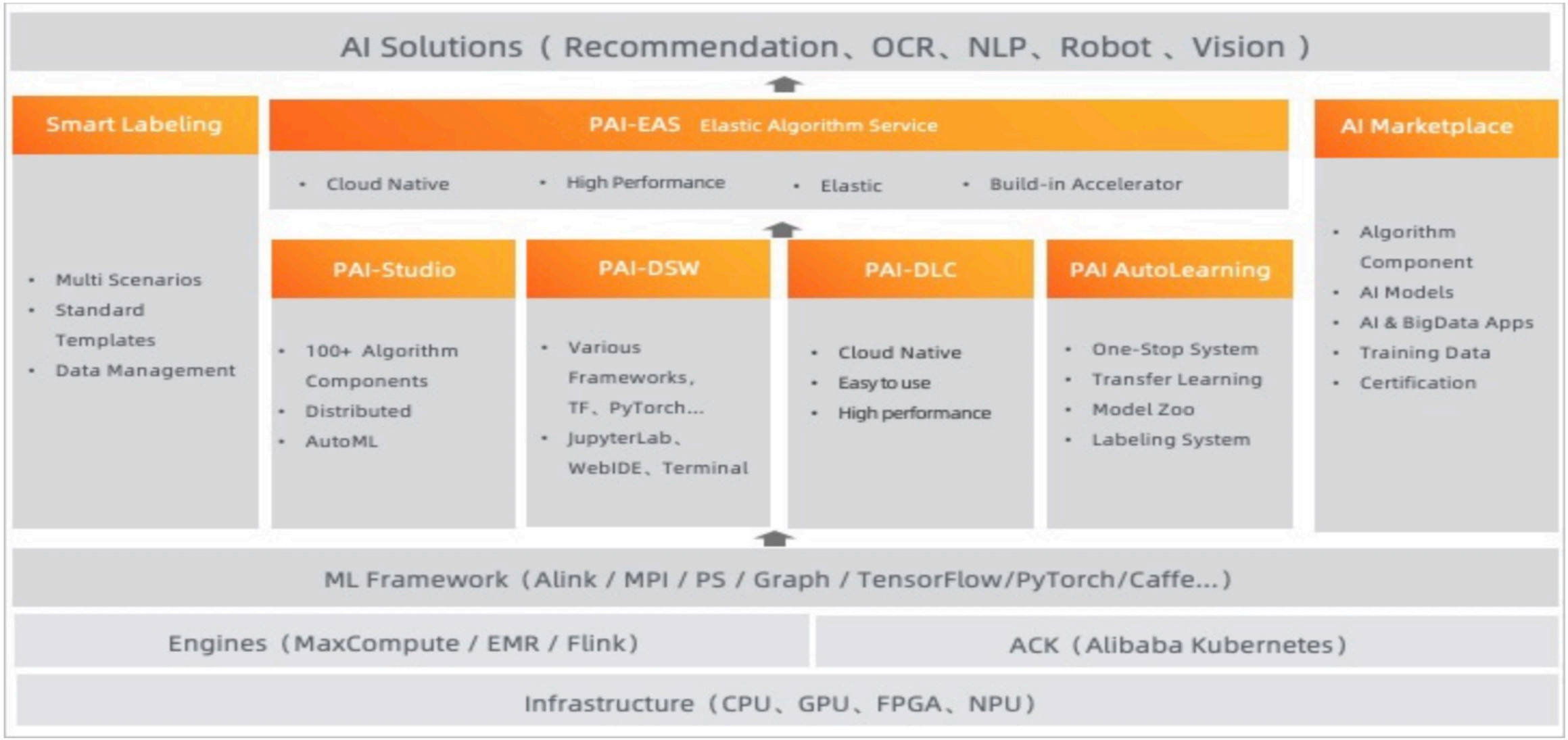
PAI : Alibaba Machine Learning Platform of AI

Alibaba Group 阿里巴巴集团



Platform of
Artificial Intelligence

PAI Architecture



https://www.alibabacloud.com/product/machine-learning

PAI-AliGraph Algorithm Framework

Algorithm 1: GNN Framework

Input: network $\mathcal{G}$, embedding dimension $d \in \mathbb{N}$, a vertex feature $\mathbf{x}_v$ for each vertex $v \in \mathcal{V}$ and the maximum hops of neighbors $k_{max} \in \mathbb{N}$.

Output: embedding result $\mathbf{h}_v$ of each vertex $v \in \mathcal{V}$

$\mathbf{h}_v^{(0)} \leftarrow \mathbf{x}_v$

for $k \leftarrow 1$ *to* k_{max} **do**

  for *each vertex* $v \in \mathcal{V}$ **do**

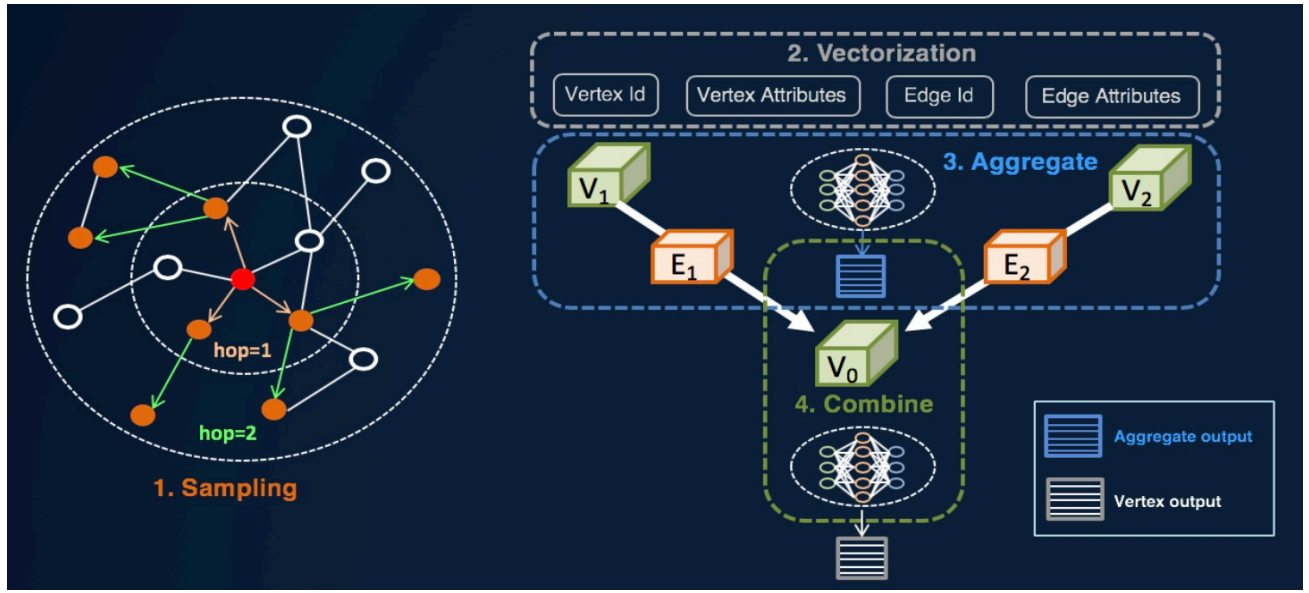
    $S_v \leftarrow \text{SAMPLE}(Nb(v))$

    $\mathbf{h}'_v \leftarrow \text{AGGREGATE}(\mathbf{h}_u^{(k-1)}, \forall u \in S)$

    $\mathbf{h}_v^{(k)} \leftarrow \text{COMBINE}(\mathbf{h}_v^{(k-1)}, \mathbf{h}'_v)$

  normalize all embedding vectors $\mathbf{h}_v^{(k)}$ for all $v \in \mathcal{V}$

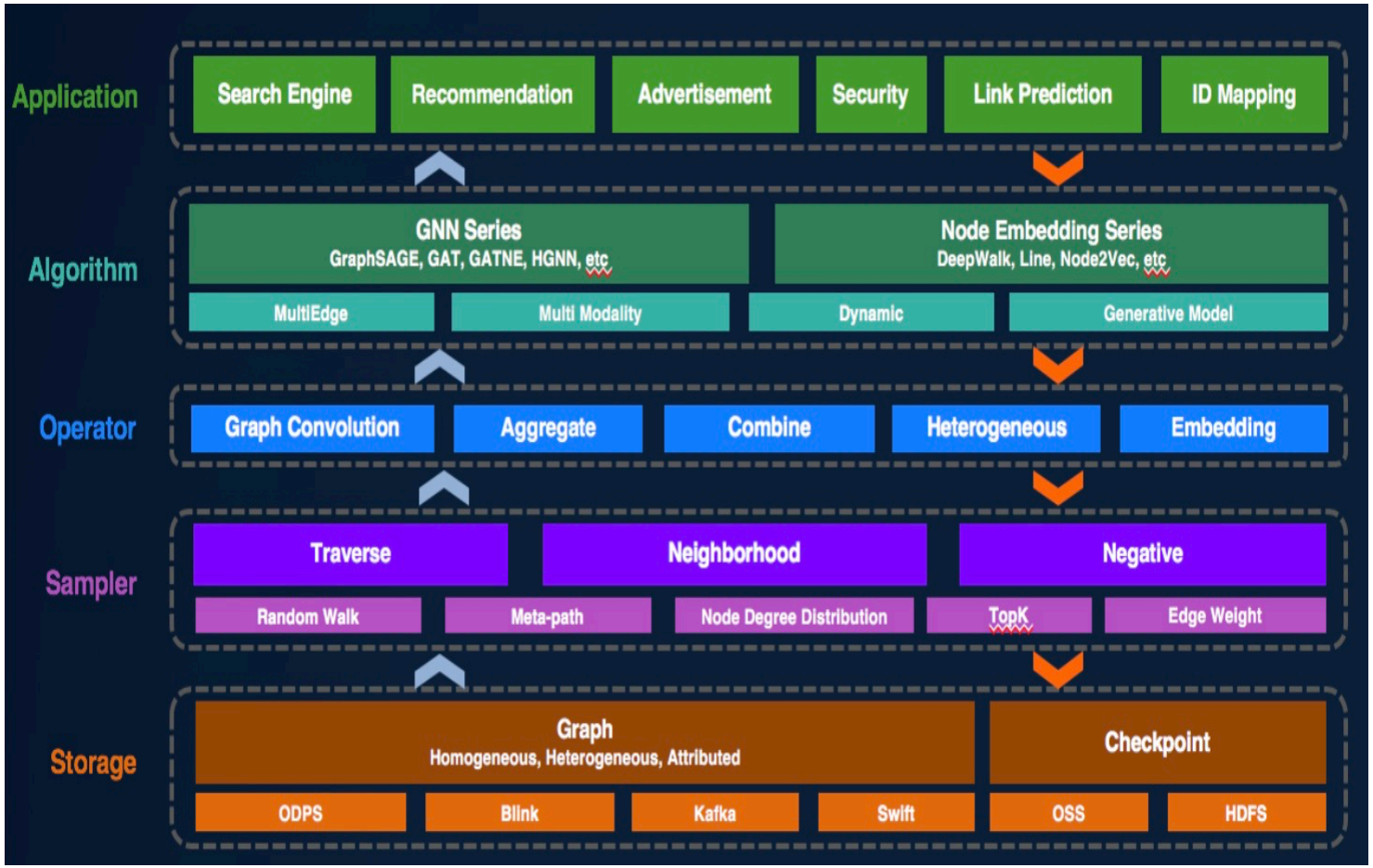
$\mathbf{h}_v \leftarrow \mathbf{h}_v^{(k_{max})}$ for all $v \in \mathcal{V}$ return $\mathbf{h}_v$ as the embedding result for all $v \in \mathcal{V}$



- Business needs to carry out more model innovations based on the AliGraph Algorithm Framework
- Need more flexible programming framework to support algorithm innovation
  - PAI Tensorflow

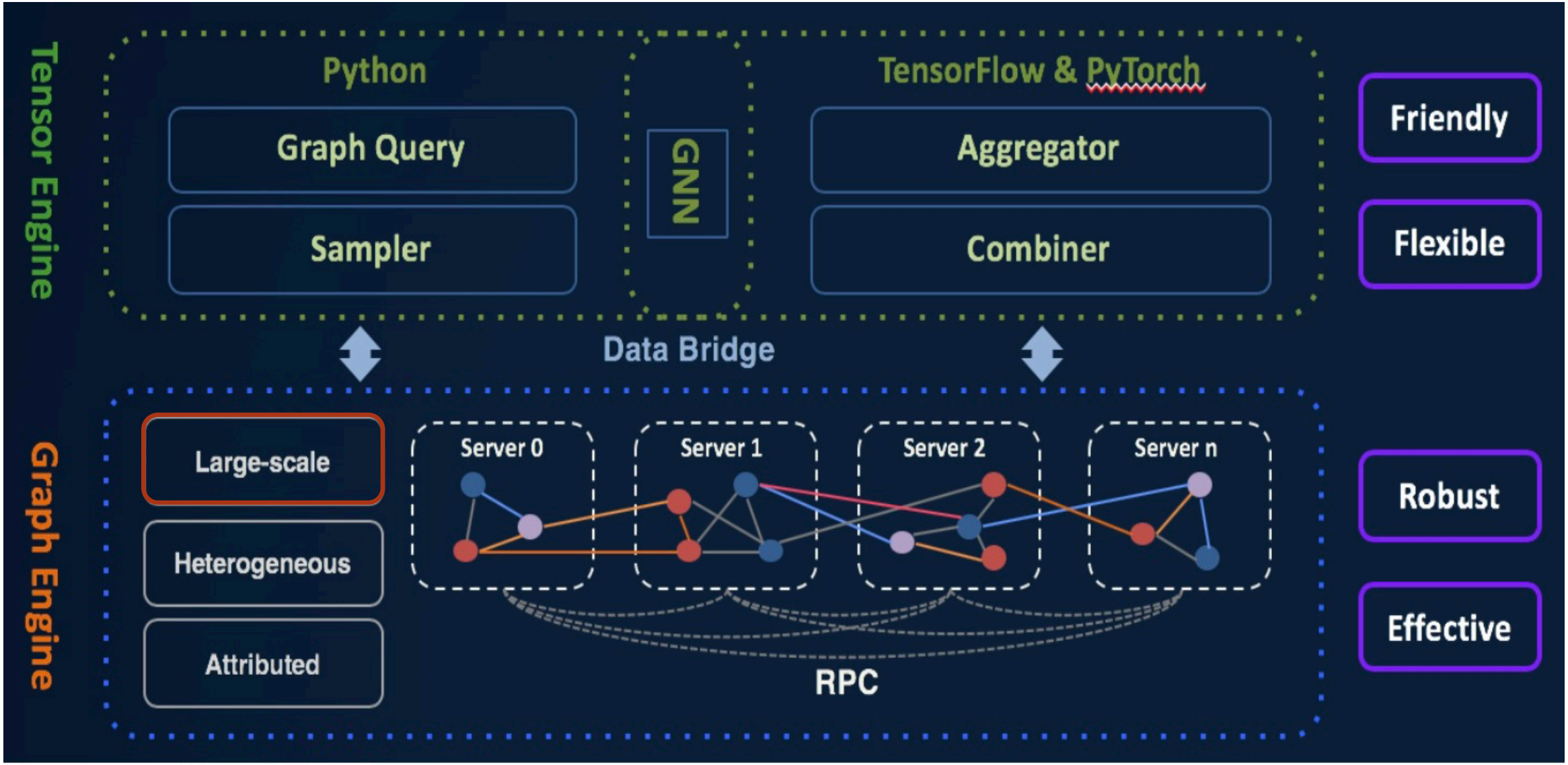
AliGraph-System Overview

- Graph storage + sampling + operator need to be tightly combined for deep learning together

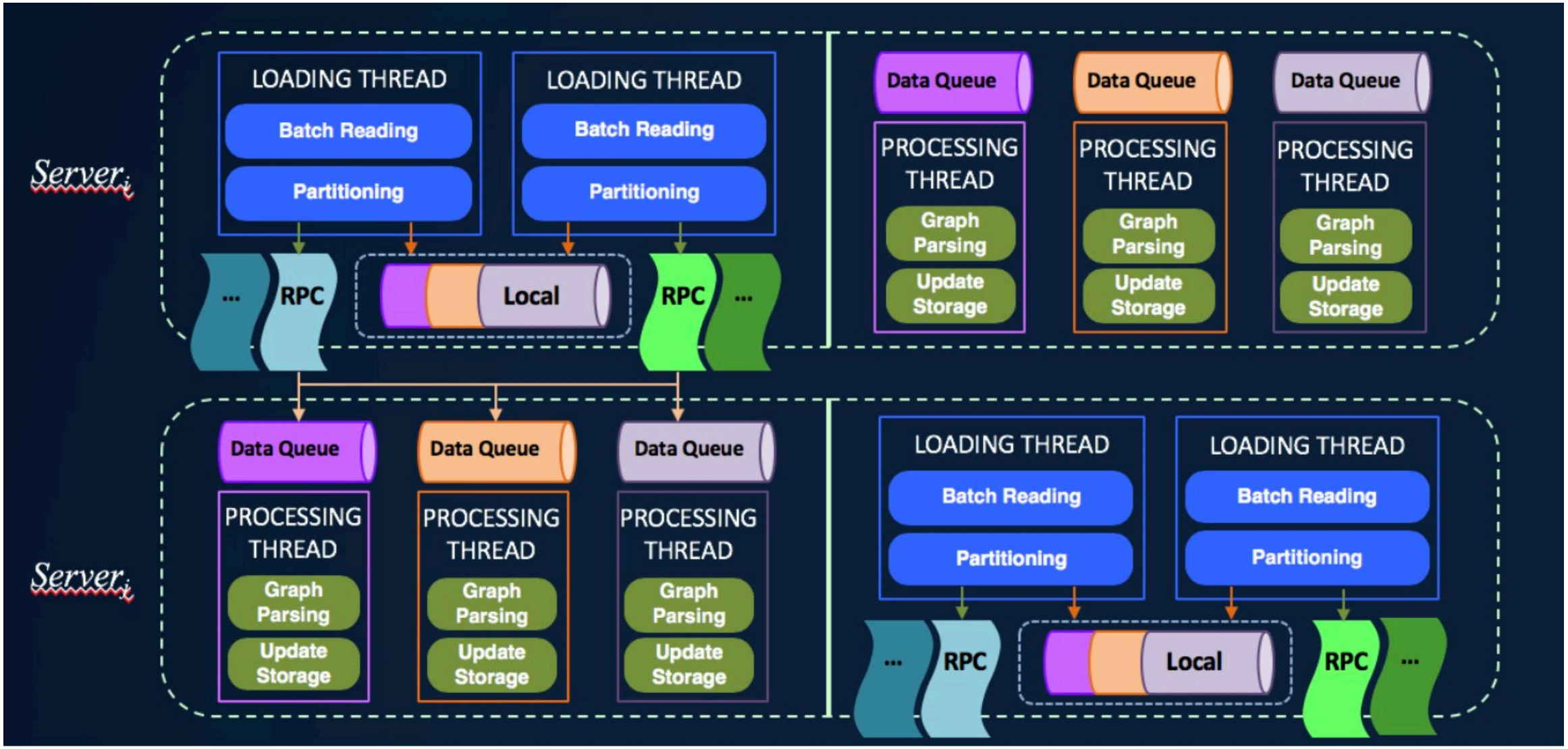


- Aligraph Algorithm Framework based on PAI-Tensorflow extension
  - Use TF's flexible composition method
  - Use TF's good design structure for effective expansion
  - Most of the existing algorithm innovations are based on TF

AliGraph-The Flow



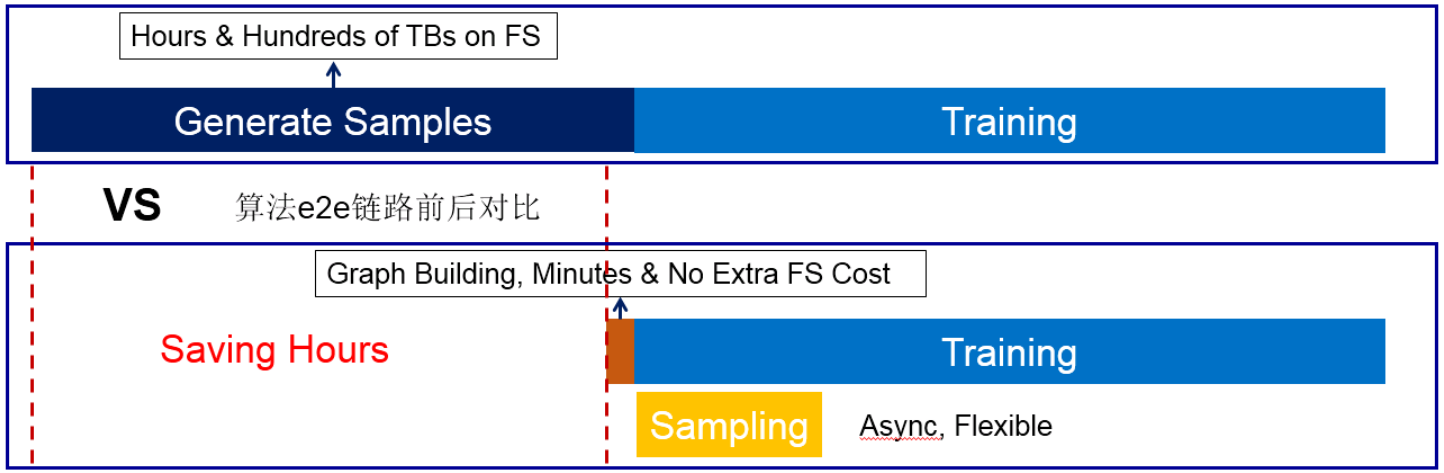
AliGraph-Graph Building



Sample

- Various sampling modes require flexible support
- Provide traverse, neighborhood, negative three ways
- Overlap sampling and calculation

```
Define a TRAVERSE sampler as s1
Define a NEIGHBORHOOD sampler as s2
Define a NEGATIVE sampler as s3
...
def sampling(s1, s2, s3, batch_size):
    vertex = s1.sample(edge_type, batch_size)
    # hop_nums contains neighbor count at each hop
    context = s2.sample(edge_type, vertex, hop_nums)
    neg = s3.sample(edge_type, vertex, neg_num)
    return vertex, context, neg
```

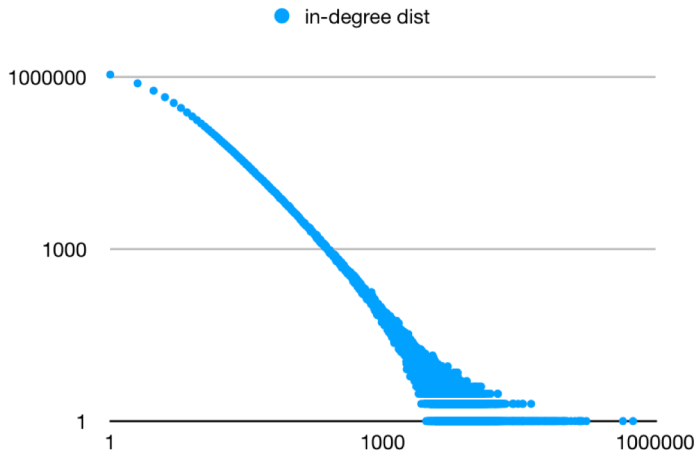


Graph Storage (Memory Representation)

Alibaba Group 阿里巴巴集团

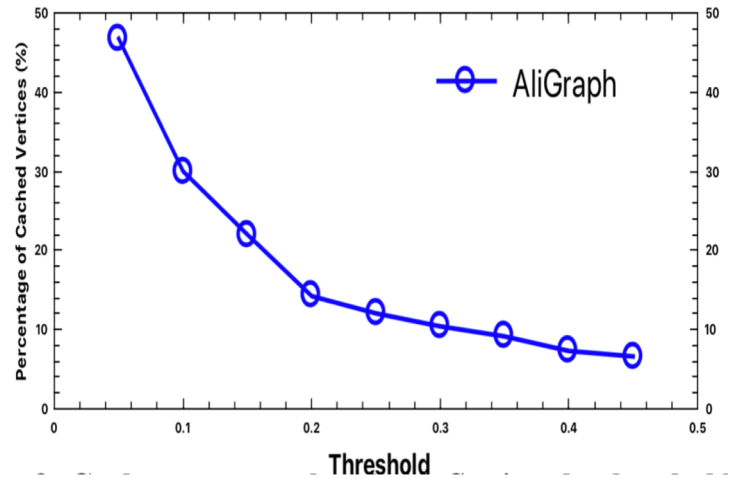
- Tens of billions of edge relations
- The adjacency relationship is the basis of the GNN algorithm, and locality needs to be fully utilized
- Tradeoff between data fragmentation and communication
  - Effective caching strategy: importance metric based on vertices
- Easy to traverse and sample by batch

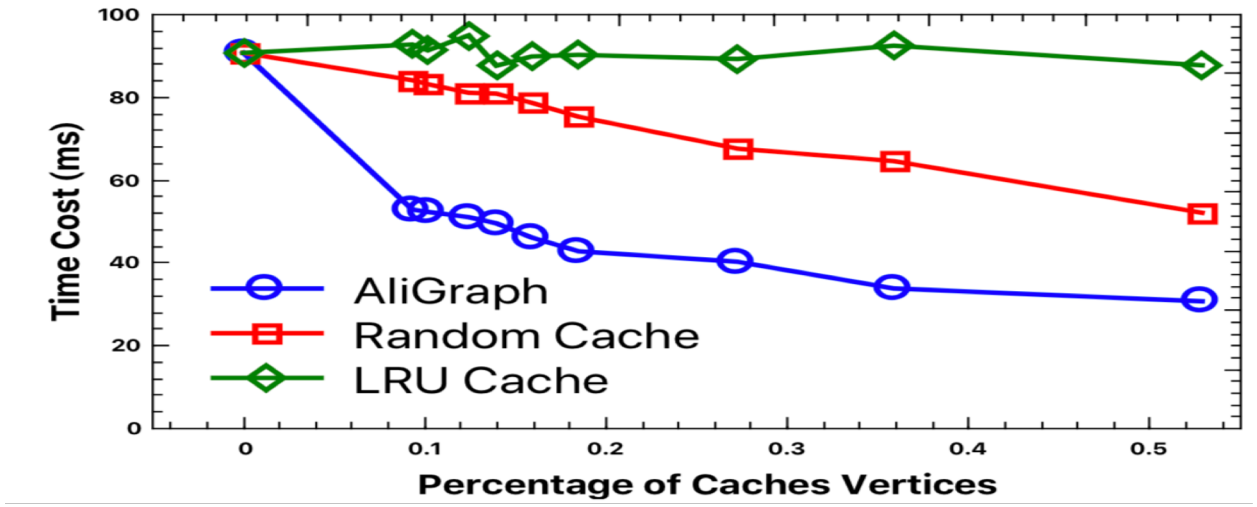
Cache/Memory Strategy



Input: graph $\mathcal{G}$, partition number p, cache depth h, threshold $\tau_1, \tau_2, \ldots, \tau_h$
Output: p subgraphs

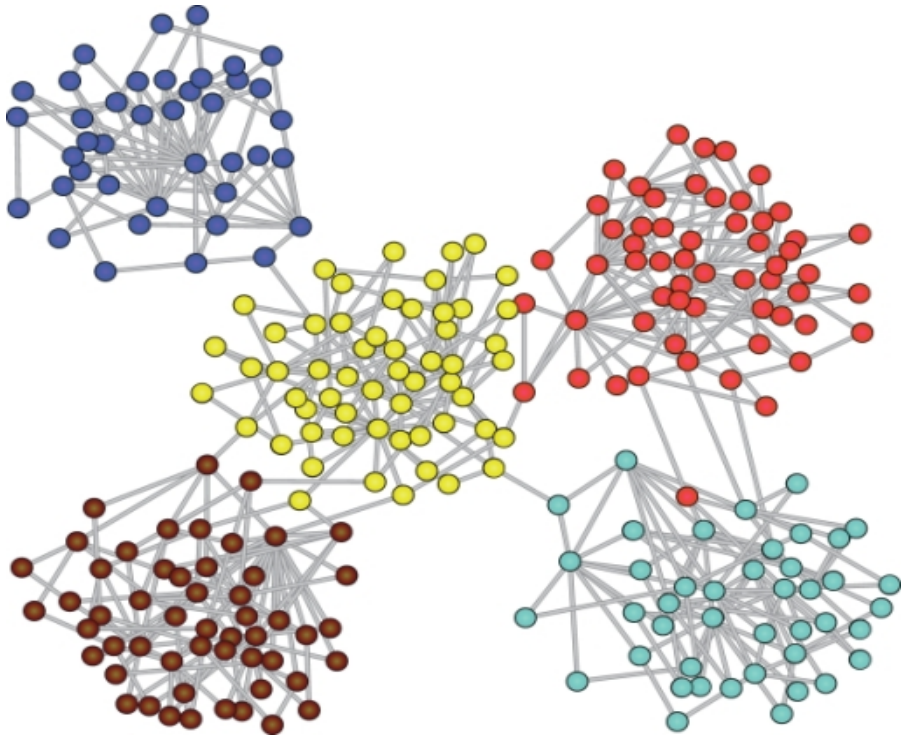
```
Initialize p graph servers
for each edge e = (u, v) ∈ E do
    j = ASSIGN(u)
    Send edge e to the j-th partition
for each vertex v ∈ V do
    for k ← 1 to h do
        Compute D_i^(k)(v) and D_o^(k)(v)
        if D_i^(k)(v) / D_o^(k)(v) ≥ τ_k then
            Cache the 1 to k-hop out-neighbors of v on each partition where
              v exists
```

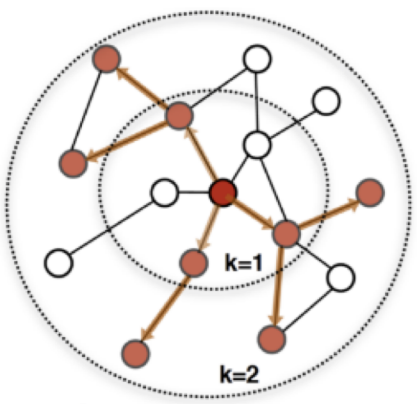


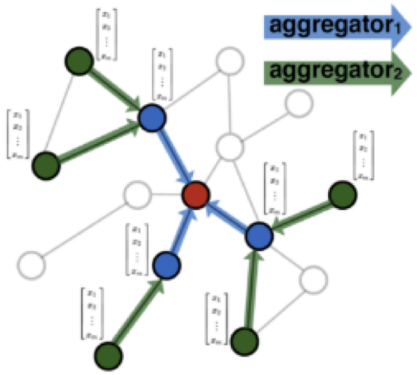


Cache acceleration: 50% faster than random method, 60% faster than LRU method

Operator Optimization

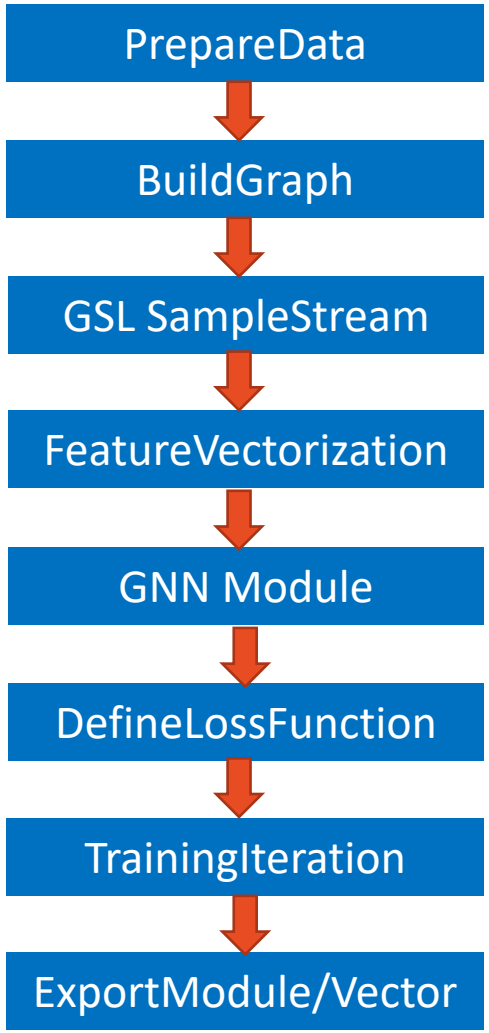






- Reduce double calculation by caching all levels of h
- Co-locate model parameters, all levels of h and the graph itself, greatly reduce network communication and delay
- Algorithm and System co-design: need to cooperate with training strategy and upper-level algorithm

AliGraph-Module Development Demo

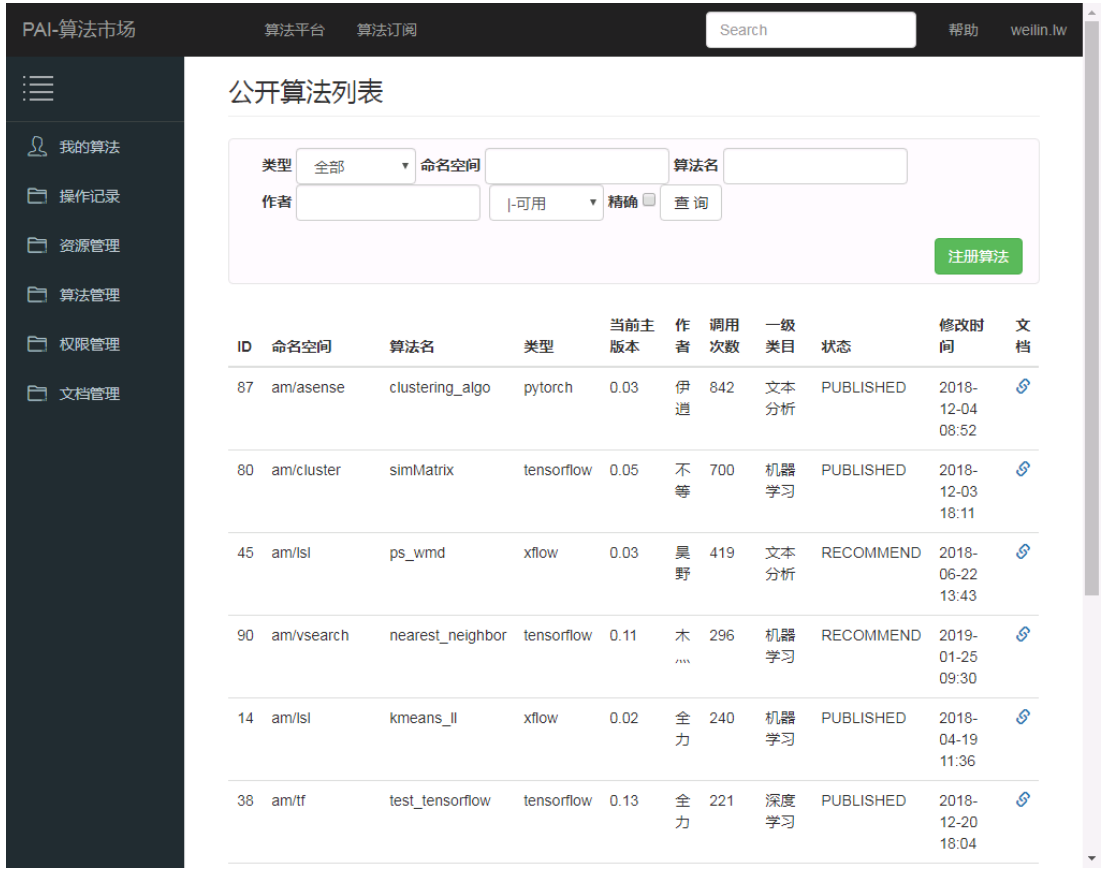


example.py

```python
import graphlearn as gl
import graphlearn.python.nn.tfm as tfm
import tensorflow as tf

# Define graph object
g = gl.Graph()

# Add data source
g.node(i_path, 'i', decoder=gl.Decoder(attr_types=['float'] * 4, attr_dims=[10] * 4, labeled=True)) \
 .edge(i2i_path, ('i', 'i', 'i-i'), decoder=gl.Decoder()) \
 .init()

# Construct GSL Query
query = g.V('i').batch(10).alias('i') \
        .outV('i-i').sample(5).by('topk').alias('hop1') \
        .outV('i-i').sample(5).by('random').alias('hop2') \
        .values()
df = tfm.DataFlow(query)

# Construct Module
dims = np.array([4, 16, 8])
model = tfm.HomoEgoGraphSAGE(dims, bn_fn=None, active_fn=tf.nn.relu, droput=0.1)

# Module computing, Generate embedding
embeddings = model.forward(df.get_ego_graph('i'))

# Construct Node Classifer
nc = tfm.NodeClassifier(dims=[8, 4], class_num=2)
logits, loss = nc.forward(embeddings, eg.nodes.labels)

# Train
trainer = tfm.Trainer()
trainer.minimize(loss)

def trace(ret):
  print('loss = %f' % ret[1])

trainer.step_to_epochs(10, [logits, loss], trace)

g.close()
```

AliGraph-Algorithm Deployment and Share

- PAI provides algorithm warehouse for algorithm package, management and release



Part 4：Algorithm Warehouse

- Representative and negative sampling
- Borrow idea from importance sampling
- Extend node-wise to batch-wise sampling
- Type-dependent & -fusion sampling with self-normalization
- Computational complexity drops from O(|E|+|V|) to O(|V|)

Algorithm 2 Training type-fusion strategy with self-normalization (one batch)

Input: same as Algorihtm 1;
Output: same as Algorithm 1;

compute p and the sampler q by Eq.(16);
for each $t \in T$ **do**
  sample n_t neighborhoods with the sampler q_t;
end for
for each v_i **do**
  for each $t \in T$ **do**
    normalize $\{\pi(v_j)|v_j \in Ng(v_i, t)\}$;
    compute the estimator $\hat{\boldsymbol{g}}_{i,t}$ by Eq. (17);
  end for
  compute the reconstructed $\tilde{\boldsymbol{h}}_i$ with $\{\hat{\boldsymbol{g}}_{i,t}|t \in T\}$ by Eq. (6);
  compute $L_{EPS,k}(v_i)$ based on $\tilde{\boldsymbol{h}}_i$, $\boldsymbol{h}_i$;
end for
minimize $L_k(\boldsymbol{H}, \boldsymbol{\Theta}^{(k)})$ and perform gradient updates;
output the optimized $\boldsymbol{\Theta}^{(k+1)}$ and $\boldsymbol{H}$.

Algorithm 1 Training with type-dependent strategy (one batch)

Input: target nodes $\{v_i\}$; neighborhood $\{v_j\}$; sampling size of each type $\{n_t|t \in T\}$; embeddings $\boldsymbol{H}$; parameters $\boldsymbol{\Theta}^{(k)}$.
Output: the optimized embedding $\boldsymbol{H}$ and parameters $\boldsymbol{\Theta}^{(k+1)}$.

for each $t \in T$ **do**
  compute p and the sampler q_t by Eq. (14);
  sample n_t neighbors with the sampler q_t;
end for
for each v_i **do**
  for each $t \in T$ **do**
    compute the estimator $\hat{\boldsymbol{g}}_{i,t}$ by Eq. (9);
  end for
  compute the reconstructed $\tilde{\boldsymbol{h}}_i$ with $\{\hat{\boldsymbol{g}}_{i,t}|t \in T\}$ by Eq. (6);
  compute $L_{EPS,k}(v_i)$ based on $\tilde{\boldsymbol{h}}_i$ and $\boldsymbol{h}_i$;
end for
minimize $L_k(\boldsymbol{H}, \boldsymbol{\Theta}^{(k)})$ and perform gradient updates;
output the optimized $\boldsymbol{\Theta}^{(k+1)}$ and $\boldsymbol{H}$.

Table 4: Micro/Macro-F1 scores for multi-class classification on Aminer. Excluding HEP, the best method is bolded and the second best is underlined. Percentages in parenthesis indicate the performance level, treating HEP-nil as 0% and HEP as 100%.

	Micro-F1				Macro-F1			
Sampling size	512 ~ 3%	1024 ~ 6%	2048 ~ 12%	4096 ~ 24%	512 ~ 3%	1024 ~ 6%	2048 ~ 12%	4096 ~ 24%
HEP-Nil	0.2084 (intuitive lower bound)				0.2027 (intuitive lower bound)			
HEP	0.9566 (intuitive upper bound)				0.9551 (intuitive upper bound)			
AS-GCN	0.2866 (10%)	0.2901 (11%)	0.2951 (12%)	0.3050 (13%)	0.2721 (9%)	0.2770 (9%)	0.2788 (9%)	0.2818 (10%)
Unif-TD	0.3089 (13%)	0.3934 (25%)	0.5227 (42%)	0.6861 (64%)	0.2864 (10%)	0.3679 (21%)	0.4978 (39%)	0.6642 (61%)
Unif-TF	0.2877 (11%)	0.4010 (26%)	0.5574 (47%)	0.7329 (70%)	0.2897 (11%)	0.3941 (25%)	0.5371 (44%)	0.7162 (68%)
Unif-TD-SN	0.2283 (3%)	0.2955 (12%)	0.4816 (37%)	0.7260 (69%)	0.1768 (-4%)	0.2286 (3%)	0.4431 (31%)	0.7209 (68%)
Unif-TF-SN	0.1631 (-6%)	0.2951 (12%)	0.4489 (32%)	0.7101 (67%)	0.1315 (-10%)	0.2655 (8%)	0.4302 (30%)	0.6954 (65%)
VarR-TD	0.3423 (18%)	0.4911 (38%)	0.5734 (49%)	0.6408 (58%)	0.3258 (16%)	0.4663 (34%)	0.5516 (46%)	0.6185 (55%)
VarR-TF	**0.6200 (55%)**	0.7472 (72%)	0.7931 (78%)	0.7920 (78%)	**0.6087 (54%)**	0.7388 (71%)	0.7812 (77%)	0.7822 (77%)
VarR-TD-SN	0.2353 (4%)	0.3245 (16%)	0.5210 (42%)	0.7078 (67%)	0.1999 (-1%)	0.2853 (10%)	0.5008 (39%)	0.6921 (65%)
VarR-TF-SN	0.5960 (52%)	**0.8124 (81%)**	**0.9008 (93%)**	**0.9311 (97%)**	0.5843 (50%)	**0.8170 (81%)**	**0.9047 (93%)**	**0.9327 (97%)**

Table 5: F1 and AUC scores for binary purchase prediction on Alibaba. Excluding HEP, the best method is bolded and the second best is underlined. Percentages in parenthesis indicate the performance level, treating HEP-nil as 0% and HEP as 100%

	F1-score				AUC			
Sampling size	512 ~ 2.5%	1024 ~ 5%	2048 ~ 10%	4096 ~ 20%	512 ~ 2.5%	1024 ~ 5%	2048 ~ 10%	4096 ~ 20%
HEP-Nil	0.3994 (intuitive lower bound)				0.5134 (intuitive lower bound)			
HEP	0.5793 (intuitive upper bound)				0.7777 (intuitive upper bound)			
AS-GCN	(unable to complete due to memory constraint)							
Unif-TD	0.3883 (-6%)	0.4353 (20%)	0.4692 (39%)	0.4891 (50%)	0.5999 (33%)	0.6616 (56%)	0.7158 (77%)	0.7463 (88%)
Unif-TF	0.4051 (3%)	0.4281 (16%)	0.4558 (31%)	0.4858 (48%)	0.6207 (41%)	0.6584 (55%)	0.6992 (70%)	0.7413 (86%)
Unif-TD-SN	0.3884 (-6%)	0.4322 (18%)	0.4470 (26%)	0.4920 (51%)	0.5982 (32%)	0.6635 (57%)	0.6873 (66%)	0.7500 (90%)
Unif-TF-SN	0.3928 (-4%)	0.4219 (13%)	0.4435 (25%)	0.4917 (51%)	0.6028 (34%)	0.6496 (52%)	0.6836 (64%)	0.7489 (89%)
VarR-TD	0.4476 (27%)	0.4518 (29%)	0.4852 (48%)	0.4743 (42%)	0.6829 (64%)	0.6969 (69%)	0.7314 (82%)	0.7317 (83%)
VarR-TF	**0.4774 (43%)**	0.4863 (48%)	**0.5019 (57%)**	0.5090 (61%)	**0.7374 (85%)**	**0.7466 (88%)**	**0.7513 (90%)**	0.7548 (91%)
VarR-TD-SN	0.4293 (17%)	0.4497 (28%)	0.4720 (40%)	0.5038 (58%)	0.6645 (57%)	0.6941 (68%)	0.7293 (82%)	0.7501 (90%)
VarR-TF-SN	0.4766 (43%)	**0.4896 (50%)**	0.4970 (54%)	**0.5097 (61%)**	0.7347 (84%)	0.7453 (88%)	0.7496 (89%)	**0.7551 (91%)**

Representation Learning for Attributed Multiplex Heterogeneous Network, KDD 2018

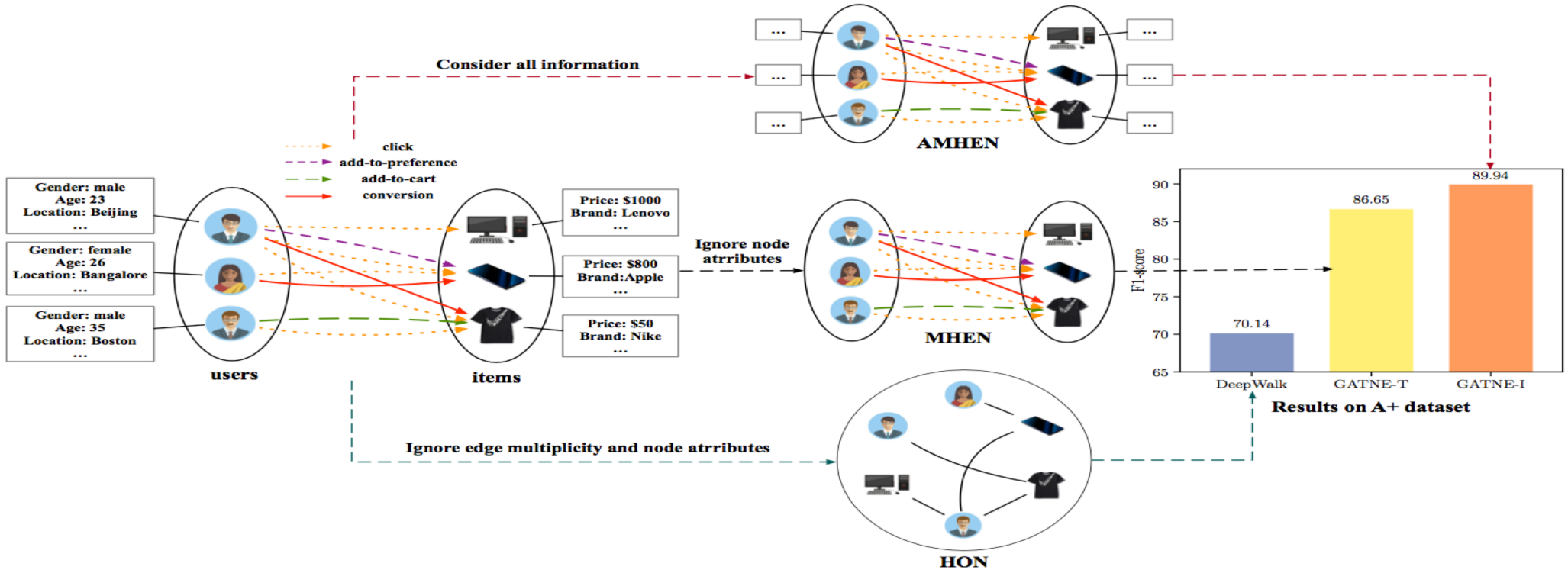


Figure 1: The left illustrates an example of an attributed multiplex heterogeneous network. Users in the left part of the figure are associated with attributes including gender, age, and location. Similarly, items in the left part of the figure include attributes such as price and brand. The edge types between users and items are from four interactions, including *click*, *add-to-preference*, *add-to-cart* and *conversion*. The three subfigures in the middle represent different ways of setting up the graphs, including HON, MHEN, and AMHEN from the bottom to the top. The right part shows the performance improvement of the proposed models over DeepWalk on the A+ dataset. As can be seen, *GATNE*-I achieves a +28.23% performance lift compared to DeepWalk.

- Transudative Model:

For *GATNE*-T, the overall embedding for node v_i on edge type r is:

$$\mathbf{v}_{i,r} = \mathbf{b}_i + \alpha_r \mathbf{M}_r^T \mathbf{U}_i \mathbf{a}_{i,r} = \mathbf{b}_i + \alpha_r \mathbf{M}_r^T \sum_{p=1}^{m} \lambda_p \mathbf{u}_{i,p},$$

where λ_p denotes the p-th element of $\mathbf{a}_{i,r}$ and is computed as:

$$\lambda_p = \frac{\exp(\mathbf{w}_r^T \tanh(\mathbf{W}_r \mathbf{u}_{i,p}))}{\sum_t \exp(\mathbf{w}_r^T \tanh(\mathbf{W}_r \mathbf{u}_{i,t}))}.$$

- Inductive Model:

$$\mathbf{v}_{i,r} = \mathbf{h}_z(\mathbf{x}_i) + \alpha_r \mathbf{M}_r^T \mathbf{U}_i \mathbf{a}_{i,r} + \beta_r \mathbf{D}_z^T \mathbf{x}_i,$$

	Amazon			YouTube			Twitter			A+ dataset (small)		
	ROC-AUC	PR-AUC	F1	ROC-AUC	PR-AUC	F1	ROC-AUC	PR-AUC	F1	ROC-AUC	PR-AUC	F1
DeepWalk	94.20	94.03	87.38	71.11	70.04	65.52	69.42	72.58	62.68	59.39	60.62	56.10
node2vec	94.47	94.30	87.88	71.21	70.32	65.36	69.90	73.04	63.12	62.26	63.40	58.49
LINE	81.45	74.97	76.35	64.24	63.25	62.35	62.29	60.88	58.18	53.97	54.65	52.85
metapath2vec	94.15	94.01	87.48	70.98	70.02	65.34	69.35	72.61	62.70	60.94	61.40	58.25
ANRL	95.41	94.19	89.60	75.93	73.21	70.65	70.04	67.16	64.69	64.76	61.67	61.37
PMNE(n)	95.59	95.48	89.37	65.06	63.59	60.85	69.48	72.66	62.88	62.23	63.35	58.74
PMNE(r)	88.38	88.56	79.67	70.61	69.82	65.39	62.91	67.85	56.13	55.29	57.49	53.65
PMNE(c)	93.55	93.46	86.42	68.63	68.22	63.54	67.04	70.23	60.84	51.57	51.78	51.44
MVE	92.98	93.05	87.80	70.39	70.10	65.10	72.62	73.47	67.04	60.24	60.51	57.08
MNE	90.28	91.74	83.25	82.30	82.18	75.03	91.37	91.65	84.32	62.79	63.82	58.74
GATNE-T	**97.44**	**97.05**	**92.87**	**84.61**	81.93	**76.83**	**92.30**	91.77	**84.96**	66.71	67.55	62.48
GATNE-I	96.25	94.77	91.36	84.47	**82.32**	**76.83**	92.04	**91.95**	84.38	**70.87**	**71.65**	**65.54**