Accelerated Attributed Network Embedding

报告人:王昕毅

Why we present our work:

1) While existing network embedding algorithms focus on pure networks, nodes in real-world networks are often associated with a rich set of features or attributes. Network topological structure and node attributes are often strongly correlated with each other. Thus modeling and incorporating node attribute proximity into network embedding could be potentially helpful in learning better vector representations.

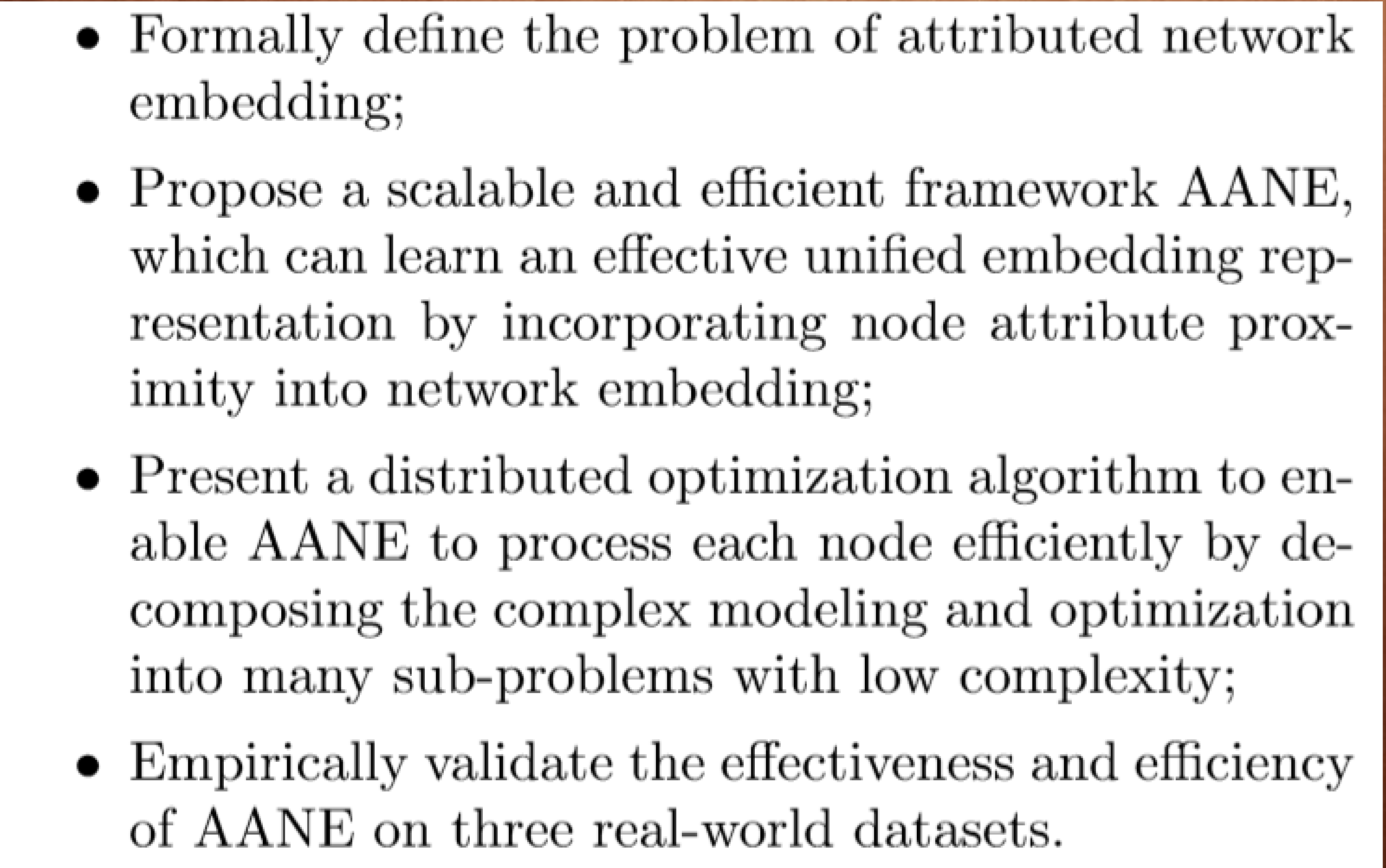
2) Meanwhile, real-world networks often put demands on the scalability of embedding algorithms. To bridge the gap, in this paper, we propose an accelerated attributed network embedding algorithm AANE, which enables the joint learning process to be done in a distributed manner by decomposing the complex modeling and optimization into many sub-problems

Attributed network embedding is challenging in three aspects as follows:
(1) High computational time requirement might be a bottleneck that limits the usage of algorithms in practice

(2) Assessing node proximity in the joint space of network and attribute is challenging due to the bewildering combination of heterogeneous information. In addition, as the size of network scales up, node attribute proximity is often too large to be stored on a single machine, not to mention the operations on it.

(3) Both sources could be incomplete and noisy, which further exacerbates the embedding representation learning problem.

To tackle the above challenges， We made our effort and summarize the contributions of this paper as follows:



- Formally define the problem of attributed network embedding;
- Propose a scalable and efficient framework AANE, which can learn an effective unified embedding representation by incorporating node attribute proximity into network embedding;
- Present a distributed optimization algorithm to enable AANE to process each node efficiently by decomposing the complex modeling and optimization into many sub-problems with low complexity;
- Empirically validate the effectiveness and efficiency of AANE on three real-world datasets.

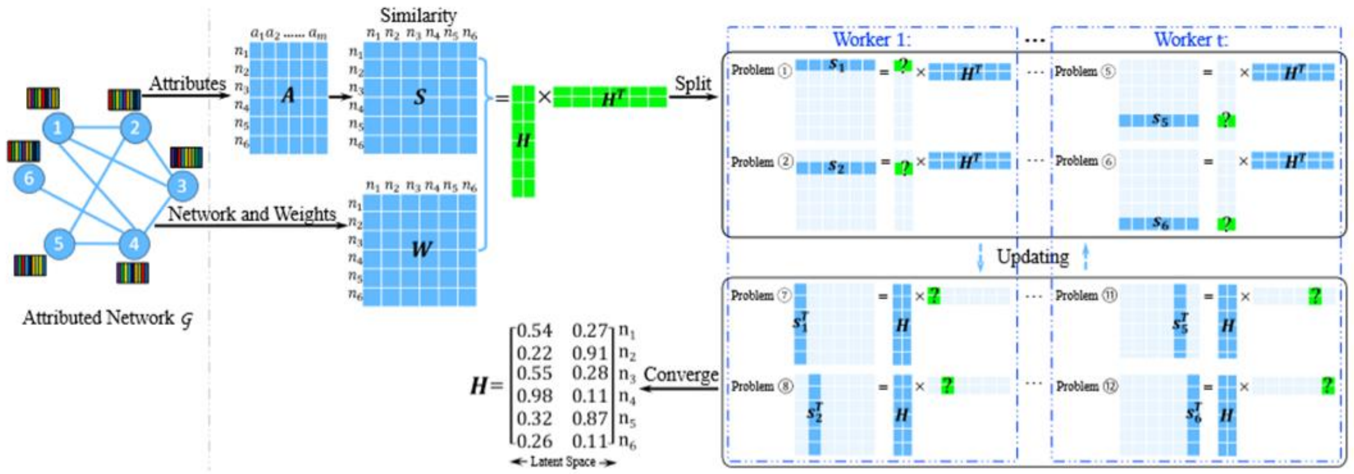
We formally define the problem of attributed network embedding as follows: Given a set of **n** nodes connected by a network **G** associated with edge weights **W** and node attributes **A**, we aim to represent each node **i** $\in$**V** as a d-dimensional vector $\mathbf{h}_i$, such that this embedding representation **H** can preserve the node proximity both in topological structure **G** and node attributes **A**. As a result, **H** could achieve better performance than original features in terms of advancing other learning tasks.

In this section, we describe how AANE jointly models network structure and attribute proximity in an effective way. Figure 1 illustrates the basic idea. Given a network with n = 6 nodes.

1)Attribute Proximity Modeling: It first decomposes attribute affinity S into the product of H and H^T.

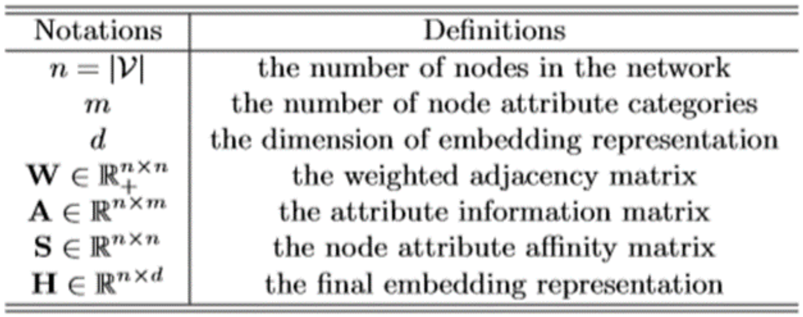
2) Network Structure Modeling: Meanwhile, it imposes an edge-based penalty into this decomposition such that connected nodes are close to each other in H, and the closeness is controlled by the edge weights in W

3) Acceleration: To accelerate the optimization, a distributed algorithm is proposed to separate the original problem into 2n = 12 sub-problems with low complexity



$$H=\begin{bmatrix}0.54 & 0.27\\0.22 & 0.91\\0.55 & 0.28\\0.98 & 0.11\\0.32 & 0.87\\0.26 & 0.11\end{bmatrix}\begin{matrix}n_1\\n_2\\n_3\\n_4\\n_5\\n_6\end{matrix}$$

Figure 1: The basic idea of AANE is to represent nodes as continuous vectors, based on the decomposition of attribute affinity matrix and the penalty of embedding difference between connected nodes. The optimization is split into $2n$ sub-problems with low complexity, which can be solved separately by t workers.



Notations	Definitions
$n = \lvert\mathcal{V}\rvert$	the number of nodes in the network
m	the number of node attribute categories
d	the dimension of embedding representation
$\mathbf{W} \in \mathbb{R}_+^{n\times n}$	the weighted adjacency matrix
$\mathbf{A} \in \mathbb{R}^{n\times m}$	the attribute information matrix
$\mathbf{S} \in \mathbb{R}^{n\times n}$	the node attribute affinity matrix
$\mathbf{H} \in \mathbb{R}^{n\times d}$	the final embedding representation

Table 1: Main symbols and definitions.

3.1 Network Structure Modeling

To preserve the node proximity in G, the proposed framework is based on two hypotheses .

First, a graph-based mapping is assumed to be smooth across edges, especially for the regions of high density [7]. This is in line with the cluster hypothesis [25].

Second, nodes with more similar topological structures or connected by higher weights are more likely to have similar vector representations.

Loss function:

$$\mathcal{J}_{\mathcal{G}} = \sum_{(i,j)\in\mathcal{E}} w_{ij} \|\mathbf{h}_i - \mathbf{h}_j\|_2, \tag{3.1}$$

where rows h_i and h_j are vector representations of node i and node j, and w_{ij} is the edge weight between the two. The key idea is that in order to minimize the penalty w_{ij} $||h_i − h_j||_2$, a larger weight w_{ij} is more likely to enforce the difference of h_i and h_j to be smaller.

3.2 Attribute Proximity Modeling

Attribute information of nodes is tightly hinged with network topological structure. Node proximity in network space should be consistent with the one in node attribute space. Thus, we investigate how to make the embedding representation **H** also well preserve the node attribute proximity .

The basic idea is to enforce the dot product of vector representations $\mathbf{h}_i$ and $\mathbf{h}_j$ to be the same as corresponding attribute similarity $\mathbf{s}_{ij}$. Mathematically, the loss function is defined as

$$\mathcal{J}_A = \|\mathbf{S} - \mathbf{H}\mathbf{H}^\top\|_\mathrm{F}^2 = \sum_{i=1}^{n}\sum_{j=1}^{n}(\mathbf{s}_{ij} - \mathbf{h}_i\mathbf{h}_j^\top)^2, \tag{3.2}$$

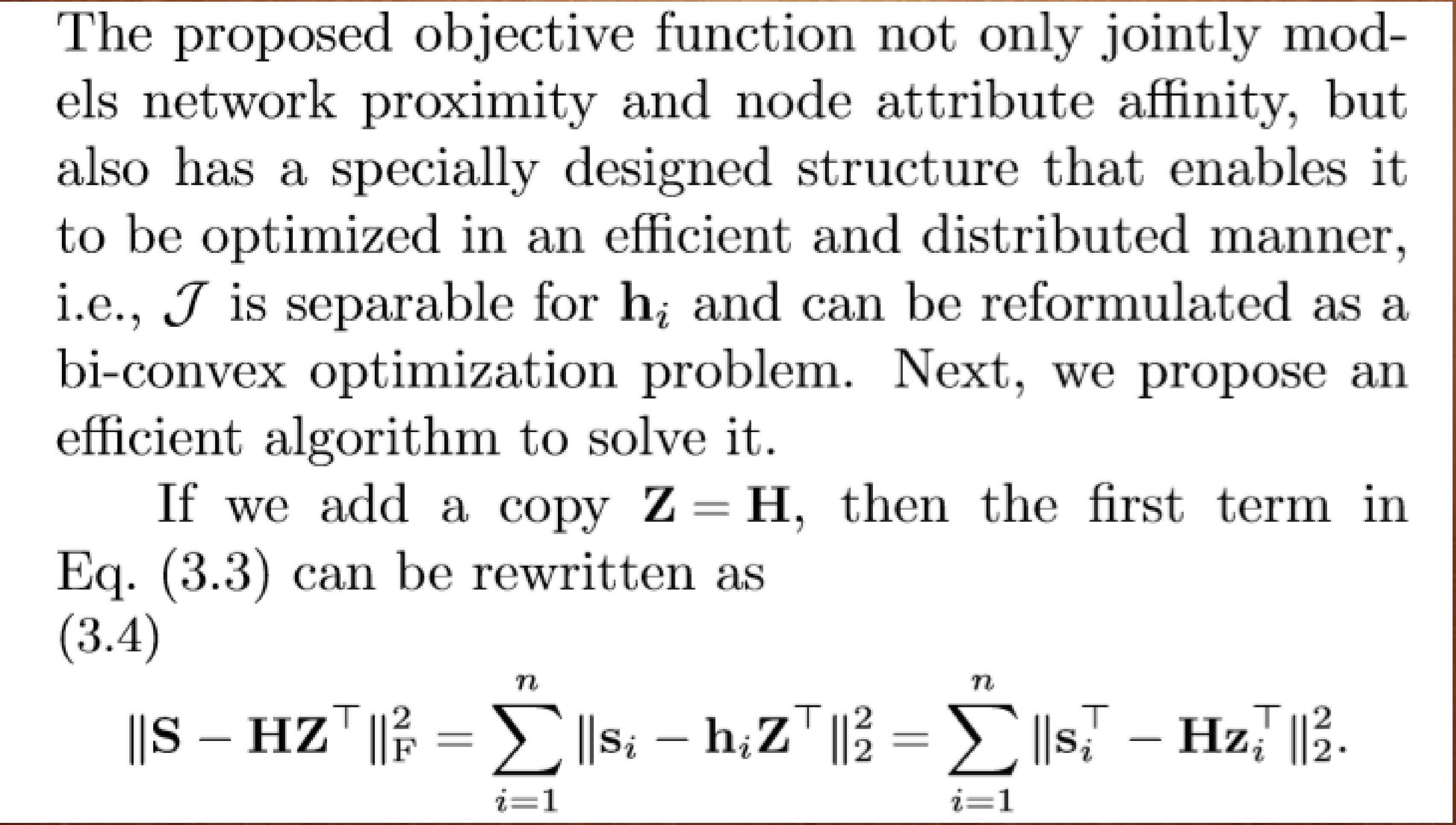
3.3 Joint Embedding Representation Learning

To make network structure modeling and attribute proximity modeling complement each other towards a unified robust and informative space, we jointly model the two types of information in the following optimization problem:

$$(3.3)\quad \min_{\mathbf{H}} \quad \mathcal{J} = \|\mathbf{S} - \mathbf{H}\mathbf{H}^{\top}\|_{\mathrm{F}}^{2} + \lambda \sum_{(i,j)\in\mathcal{E}} w_{ij}\|\mathbf{h}_i - \mathbf{h}_j\|_2$$

Scalar λ serves as an overall parameter that defines a trade-off between the contributions of network and attribute information.

3.4 Accelerated and Distributed Algorithm

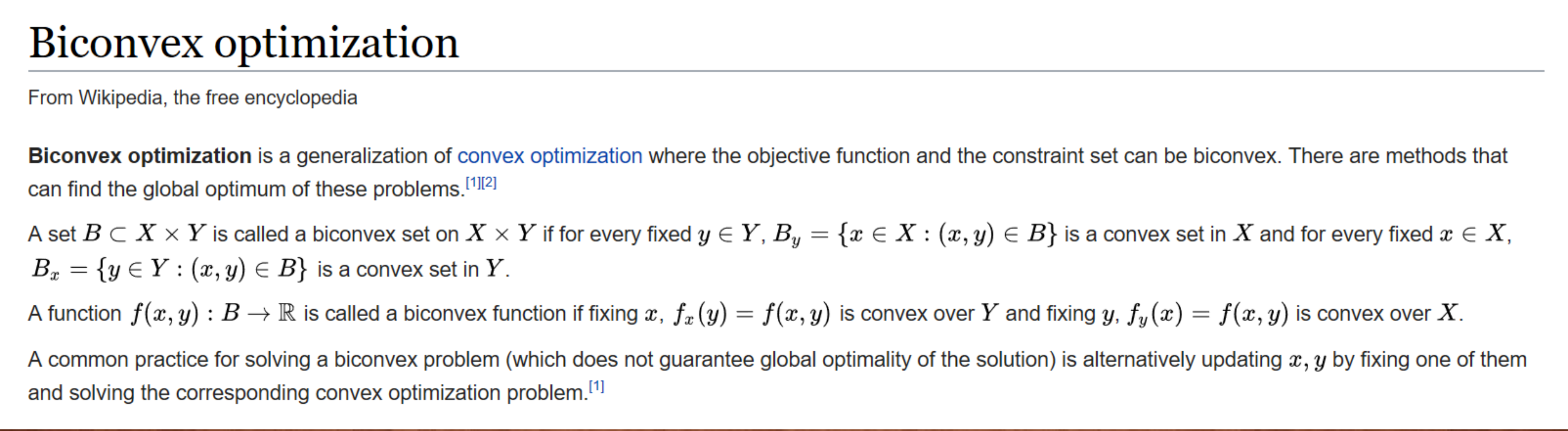


The proposed objective function not only jointly models network proximity and node attribute affinity, but also has a specially designed structure that enables it to be optimized in an efficient and distributed manner, i.e., $\mathcal{J}$ is separable for $\mathbf{h}_i$ and can be reformulated as a bi-convex optimization problem. Next, we propose an efficient algorithm to solve it.

If we add a copy $\mathbf{Z} = \mathbf{H}$, then the first term in Eq. (3.3) can be rewritten as

(3.4)

$$\|\mathbf{S} - \mathbf{H}\mathbf{Z}^\top\|_{\mathrm{F}}^2 = \sum_{i=1}^{n} \|\mathbf{s}_i - \mathbf{h}_i\mathbf{Z}^\top\|_2^2 = \sum_{i=1}^{n} \|\mathbf{s}_i^\top - \mathbf{H}\mathbf{z}_i^\top\|_2^2.$$



Biconvex optimization

From Wikipedia, the free encyclopedia

Biconvex optimization is a generalization of convex optimization where the objective function and the constraint set can be biconvex. There are methods that can find the global optimum of these problems.[1][2]

A set $B \subset X \times Y$ is called a biconvex set on $X \times Y$ if for every fixed $y \in Y$, $B_y = \{x \in X : (x, y) \in B\}$ is a convex set in X and for every fixed $x \in X$, $B_x = \{y \in Y : (x, y) \in B\}$ is a convex set in Y.

A function $f(x, y) : B \to \mathbb{R}$ is called a biconvex function if fixing x, $f_x(y) = f(x, y)$ is convex over Y and fixing y, $f_y(x) = f(x, y)$ is convex over X.

A common practice for solving a biconvex problem (which does not guarantee global optimality of the solution) is alternatively updating x, y by fixing one of them and solving the corresponding convex optimization problem.[1]

(3.3) $$\min_{\mathbf{H}} \; \mathcal{J} = \|\mathbf{S} - \mathbf{H}\mathbf{H}^\top\|_F^2 + \lambda \sum_{(i,j)\in\mathcal{E}} w_{ij}\|\mathbf{h}_i - \mathbf{h}_j\|_2$$

(3.4)
$$\|\mathbf{S} - \mathbf{H}\mathbf{Z}^\top\|_F^2 = \sum_{i=1}^{n} \|\mathbf{s}_i - \mathbf{h}_i\mathbf{Z}^\top\|_2^2 = \sum_{i=1}^{n} \|\mathbf{s}_i^\top - \mathbf{H}\mathbf{z}_i^\top\|_2^2.$$

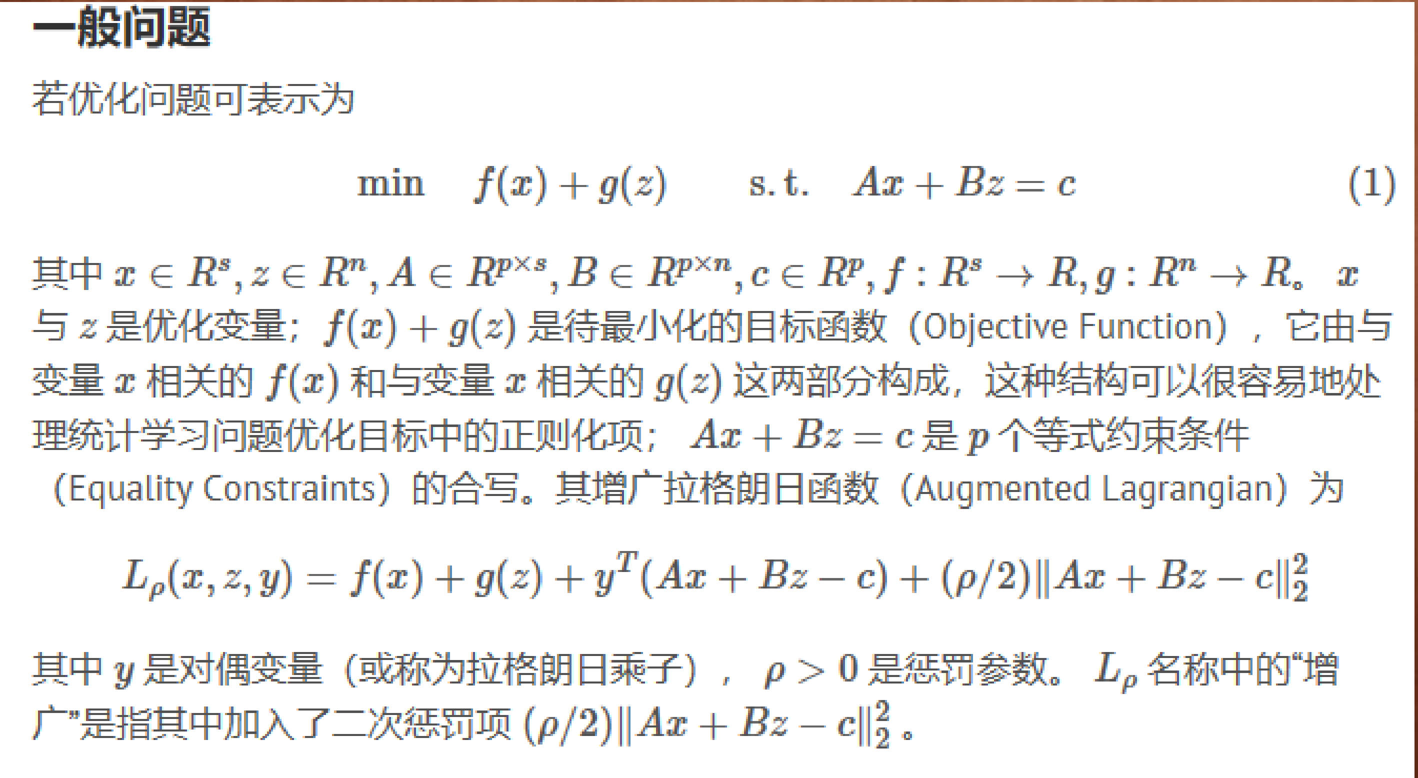
(3.5)
$$\begin{aligned} \min_{\mathbf{H}} \quad & \sum_{i=1}^{n} \|\mathbf{s}_i - \mathbf{h}_i\mathbf{Z}^\top\|_2^2 + \lambda \sum_{(i,j)\in\mathcal{E}} w_{ij}\|\mathbf{h}_i - \mathbf{z}_j\|_2, \\ \text{subject to} \quad & \mathbf{h}_i = \mathbf{z}_i, \; i = 1, \ldots, n. \end{aligned}$$

This indicates that $\mathcal{J}$ is separable for both $\mathbf{h}_i$ and $\mathbf{z}_i$. Since ℓ_2-norm is convex, it is easy to verify that Eq. (3.5) is also bi-convex, i.e., convex w.r.t. $\mathbf{h}_i$ when $\mathbf{z}_i$ is fixed and convex w.r.t. $\mathbf{z}_i$ when $\mathbf{h}_i$ is fixed. Thus, we are able to split the original complex embedding problem into $2n$ small convex optimization sub-problems.

It is infeasible to obtain closed-form solutions for these $2n$ sub-problems. Motivated by the distributed convex optimization technique - Alternating Direction Method of Multipliers (ADMM) [2, 9], we accelerate the optimization by converting it into $2n$ updating steps and one matrix updating step. The proposed solution enjoys several nice properties. First, it enables the n updating steps of $\mathbf{h}_i$ (or $\mathbf{z}_i$) independent of each other. Thus in each iteration, the global coordination can assign tasks to workers and collect the solutions from them without a fixed order. Second, all updating steps have low complexity. Third, it converges to a modest solution fast [2]. We now introduce the details.

ADMM简介：

交替方向乘子法（Alternating Direction Method of Multipliers，ADMM）是一种求解优化问题的计算框架, 适用于求解分布式凸优化问题，特别是统计学习问题。 ADMM 通过分解协调（Decomposition-Coordination）过程，将大的全局问题分解为多个较小、较容易求解的局部子问题，并通过协调子问题的解而得到大的全局问题的解。



一般问题

若优化问题可表示为

$$\min \quad f(x)+g(z) \qquad \text{s.t.} \quad Ax+Bz=c \tag{1}$$

其中 $x \in R^s, z \in R^n, A \in R^{p\times s}, B \in R^{p\times n}, c \in R^p, f: R^s \to R, g: R^n \to R$。 x 与 z 是优化变量； $f(x)+g(z)$ 是待最小化的目标函数（Objective Function），它由与变量 x 相关的 $f(x)$ 和与变量 x 相关的 $g(z)$ 这两部分构成，这种结构可以很容易地处理统计学习问题优化目标中的正则化项； $Ax+Bz=c$ 是 p 个等式约束条件（Equality Constraints）的合写。其增广拉格朗日函数（Augmented Lagrangian）为

$$L_\rho(x,z,y)=f(x)+g(z)+y^T(Ax+Bz-c)+(\rho/2)\|Ax+Bz-c\|_2^2$$

其中 y 是对偶变量（或称为拉格朗日乘子）， $\rho>0$ 是惩罚参数。 L_ρ 名称中的“增广”是指其中加入了二次惩罚项 $(\rho/2)\|Ax+Bz-c\|_2^2$ 。

则该优化问题的 ADMM 迭代求解方法为

$$
\begin{aligned}
x^{k+1} &:= \arg\min_x L_\rho(x, z^k, y^k) \\
z^{k+1} &:= \arg\min_z L_\rho(x^{k+1}, z, y^k) \\
y^{k+1} &:= y^k + \rho(Ax^{k+1} + Bz^{k+1} - c)
\end{aligned}
$$

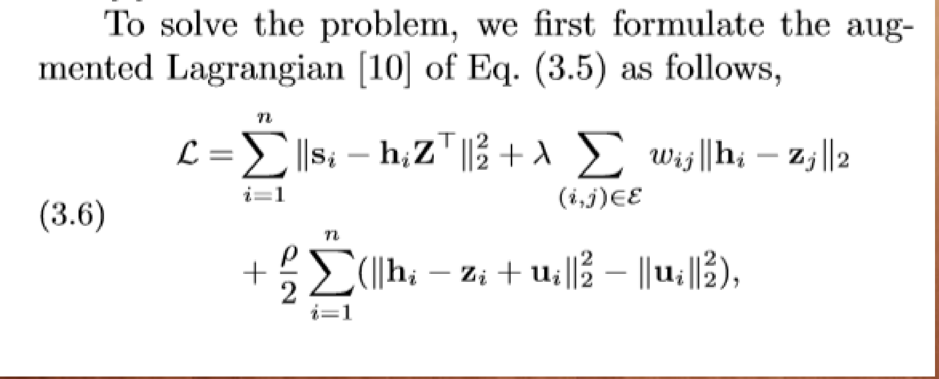
令 $u = (1/\rho)y$，并对 $Ax + Bz - c$ 配方，可得表示上更简洁的缩放形式（Scaled Form）

$$
\begin{aligned}
x^{k+1} &:= \arg\min_x \left(f(x) + (\rho/2)\|Ax + Bz^k - c + u^k\|_2^2\right) \\
z^{k+1} &:= \arg\min_z \left(g(z) + (\rho/2)\|Ax^{k+1} + Bz - c + u^k\|_2^2\right) \\
u^{k+1} &:= u^k + Ax^{k+1} + Bz^{k+1} - c
\end{aligned}
$$

可以看出，每次迭代分为三步：

1. 求解与 x 相关的最小化问题，更新变量 x
2. 求解与 z 相关的最小化问题，更新变量 z
3. 更新对偶变量 u

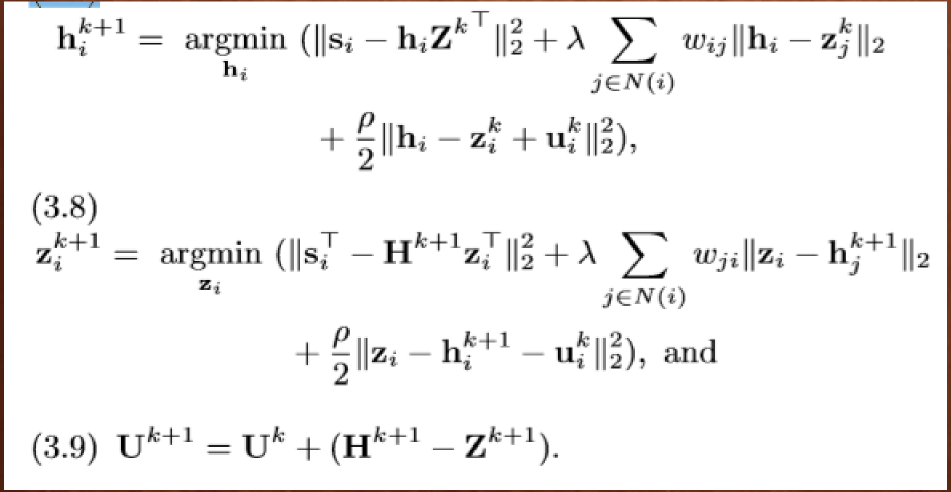
ADMM名称中的“乘子法”是指这是一种使用增广拉格朗日函数（带有二次惩罚项）的对偶上升（Dual Ascent）方法，而“交替方向”是指变量 x 和 z 是交替更新的。两变量的交替更新是在 $f(x)$ 或 $g(z)$ 可分时可以将优化问题分解的关键原因。



To solve the problem, we first formulate the augmented Lagrangian [10] of Eq. (3.5) as follows,

$$\begin{aligned}\mathcal{L} = &\sum_{i=1}^{n} \|\mathbf{s}_i - \mathbf{h}_i \mathbf{Z}^\top\|_2^2 + \lambda \sum_{(i,j)\in\mathcal{E}} w_{ij}\|\mathbf{h}_i - \mathbf{z}_j\|_2 \\ &+ \frac{\rho}{2}\sum_{i=1}^{n}(\|\mathbf{h}_i - \mathbf{z}_i + \mathbf{u}_i\|_2^2 - \|\mathbf{u}_i\|_2^2),\end{aligned} \tag{3.6}$$

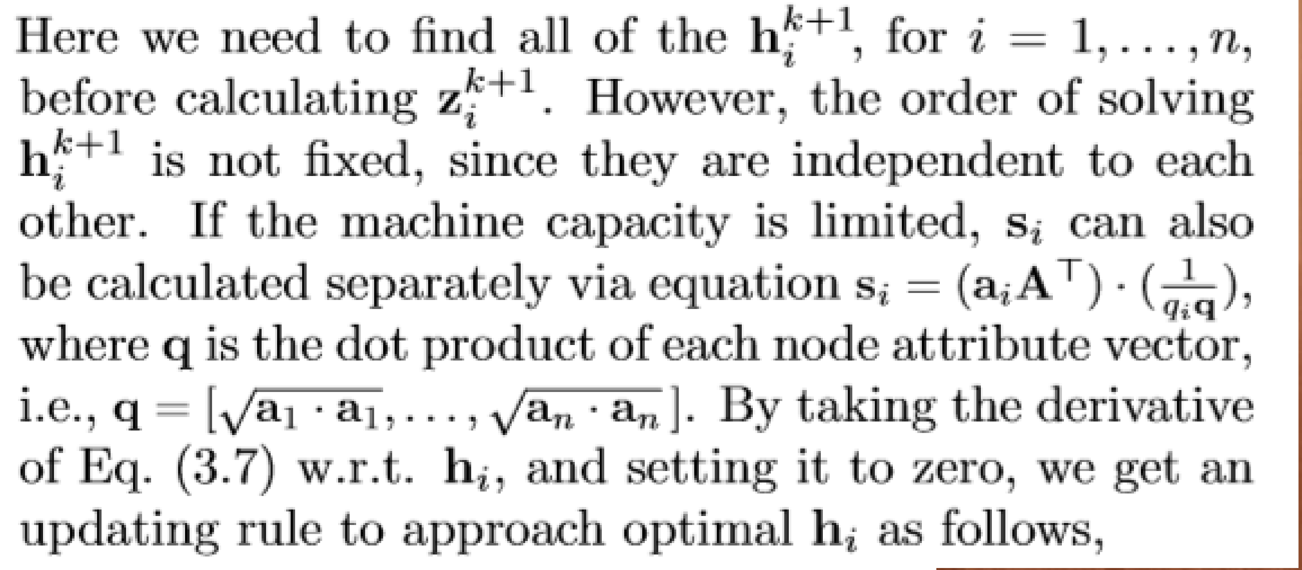
The corresponding optimization problems in terms of each node i at iteration k +1 are written as (3.7)



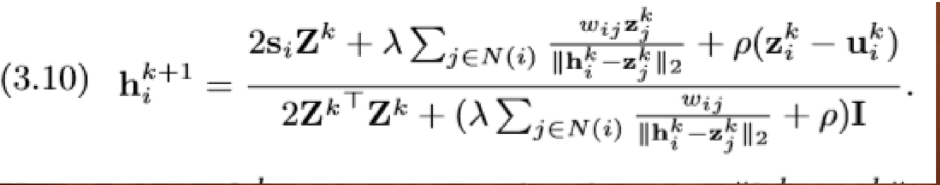
$$\begin{aligned}\mathbf{h}_i^{k+1} = &\operatorname*{argmin}_{\mathbf{h}_i} (\|\mathbf{s}_i - \mathbf{h}_i {\mathbf{Z}^k}^\top\|_2^2 + \lambda \sum_{j\in N(i)} w_{ij}\|\mathbf{h}_i - \mathbf{z}_j^k\|_2 \\ &+ \frac{\rho}{2}\|\mathbf{h}_i - \mathbf{z}_i^k + \mathbf{u}_i^k\|_2^2),\end{aligned}$$

$$\begin{aligned}\mathbf{z}_i^{k+1} = &\operatorname*{argmin}_{\mathbf{z}_i} (\|\mathbf{s}_i^\top - \mathbf{H}^{k+1}\mathbf{z}_i^\top\|_2^2 + \lambda \sum_{j\in N(i)} w_{ji}\|\mathbf{z}_i - \mathbf{h}_j^{k+1}\|_2 \\ &+ \frac{\rho}{2}\|\mathbf{z}_i - \mathbf{h}_i^{k+1} - \mathbf{u}_i^k\|_2^2), \text{ and}\end{aligned} \tag{3.8}$$

$$\mathbf{U}^{k+1} = \mathbf{U}^k + (\mathbf{H}^{k+1} - \mathbf{Z}^{k+1}). \tag{3.9}$$

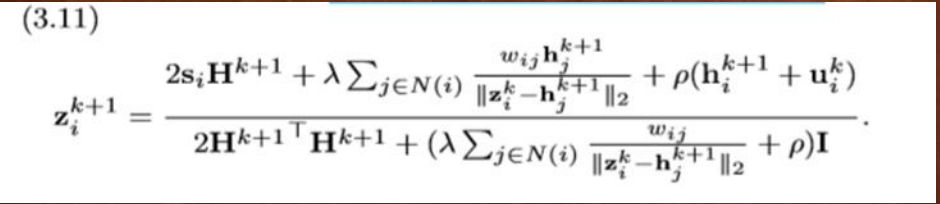


Here we need to find all of the $\mathbf{h}_i^{k+1}$, for $i = 1, \ldots, n$, before calculating $\mathbf{z}_i^{k+1}$. However, the order of solving $\mathbf{h}_i^{k+1}$ is not fixed, since they are independent to each other. If the machine capacity is limited, $\mathbf{s}_i$ can also be calculated separately via equation $\mathbf{s}_i = (\mathbf{a}_i \mathbf{A}^\top) \cdot (\frac{1}{q_i \mathbf{q}})$, where $\mathbf{q}$ is the dot product of each node attribute vector, i.e., $\mathbf{q} = [\sqrt{\mathbf{a}_1 \cdot \mathbf{a}_1}, \ldots, \sqrt{\mathbf{a}_n \cdot \mathbf{a}_n}]$. By taking the derivative of Eq. (3.7) w.r.t. $\mathbf{h}_i$, and setting it to zero, we get an updating rule to approach optimal $\mathbf{h}_i$ as follows,

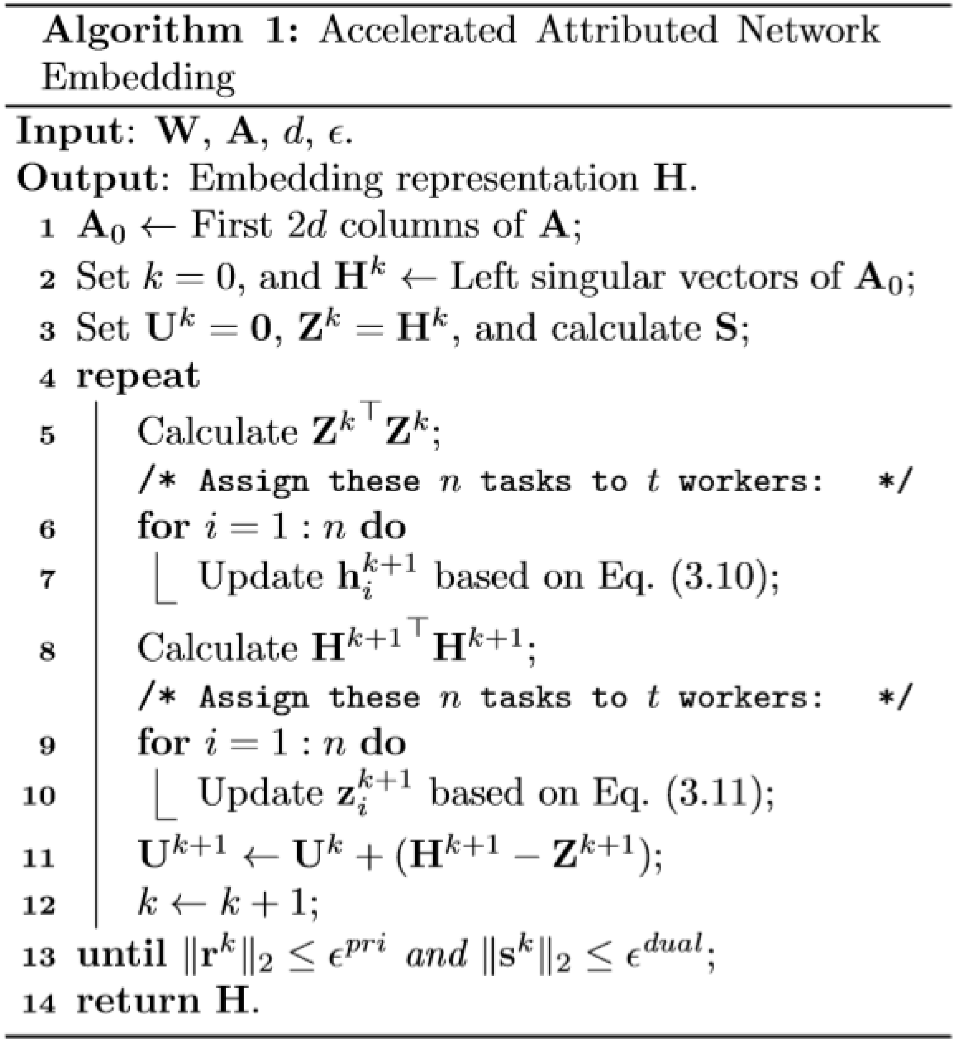


$$\mathbf{h}_i^{k+1} = \frac{2\mathbf{s}_i \mathbf{Z}^k + \lambda \sum_{j \in N(i)} \frac{w_{ij} \mathbf{z}_j^k}{\|\mathbf{h}_i^k - \mathbf{z}_j^k\|_2} + \rho(\mathbf{z}_i^k - \mathbf{u}_i^k)}{2\mathbf{Z}^{k\top} \mathbf{Z}^k + (\lambda \sum_{j \in N(i)} \frac{w_{ij}}{\|\mathbf{h}_i^k - \mathbf{z}_j^k\|_2} + \rho)\mathbf{I}}. \tag{3.10}$$

We obtain a similar rule for z_i:



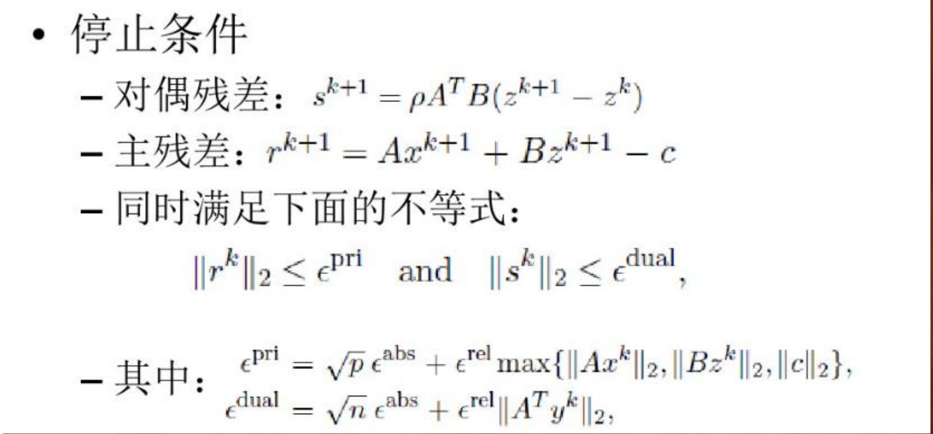
$$\mathbf{z}_i^{k+1} = \frac{2\mathbf{s}_i \mathbf{H}^{k+1} + \lambda \sum_{j \in N(i)} \frac{w_{ij} \mathbf{h}_j^{k+1}}{\|\mathbf{z}_i^k - \mathbf{h}_j^{k+1}\|_2} + \rho(\mathbf{h}_i^{k+1} + \mathbf{u}_i^k)}{2\mathbf{H}^{k+1\top} \mathbf{H}^{k+1} + (\lambda \sum_{j \in N(i)} \frac{w_{ij}}{\|\mathbf{z}_i^k - \mathbf{h}_j^{k+1}\|_2} + \rho)\mathbf{I}}. \tag{3.11}$$



Algorithm 1: Accelerated Attributed Network Embedding

```
Input: $\mathbf{W}$, $\mathbf{A}$, $d$, $\epsilon$.
Output: Embedding representation $\mathbf{H}$.
1  $\mathbf{A}_0 \leftarrow$ First $2d$ columns of $\mathbf{A}$;
2  Set $k = 0$, and $\mathbf{H}^k \leftarrow$ Left singular vectors of $\mathbf{A}_0$;
3  Set $\mathbf{U}^k = \mathbf{0}$, $\mathbf{Z}^k = \mathbf{H}^k$, and calculate $\mathbf{S}$;
4  repeat
5      Calculate ${\mathbf{Z}^k}^\top \mathbf{Z}^k$;
       /* Assign these $n$ tasks to $t$ workers:   */
6      for $i = 1 : n$ do
7          Update $\mathbf{h}_i^{k+1}$ based on Eq. (3.10);
8      Calculate ${\mathbf{H}^{k+1}}^\top \mathbf{H}^{k+1}$;
       /* Assign these $n$ tasks to $t$ workers:   */
9      for $i = 1 : n$ do
10         Update $\mathbf{z}_i^{k+1}$ based on Eq. (3.11);
11     $\mathbf{U}^{k+1} \leftarrow \mathbf{U}^k + (\mathbf{H}^{k+1} - \mathbf{Z}^{k+1})$;
12     $k \leftarrow k + 1$;
13 until $\|\mathbf{r}^k\|_2 \le \epsilon^{pri}$ and $\|\mathbf{s}^k\|_2 \le \epsilon^{dual}$;
14 return $\mathbf{H}$.
```

To have an appropriate initialization, we set H as the left singular vectors of A_0 in the first iteration. A_0 is a matrix that collects the first 2d columns of A. The optimization is split into 2n sub-problems, and we solve them iteratively. In each iteration, the n updating steps of h_i (or z_i) can be assigned to t workers in a distributed way as illustrated in Figure 1.

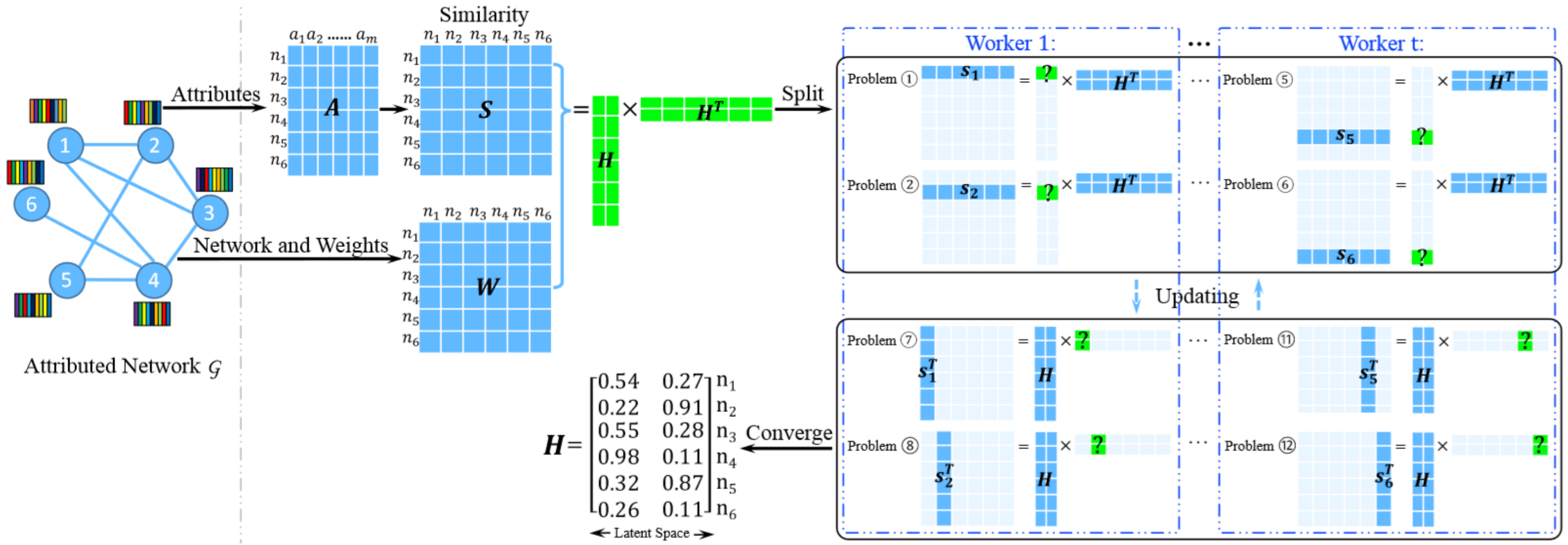


- 停止条件
  - 对偶残差： $s^{k+1} = \rho A^T B(z^{k+1} - z^k)$
  - 主残差： $r^{k+1} = Ax^{k+1} + Bz^{k+1} - c$
  - 同时满足下面的不等式：

$$\|r^k\|_2 \le \epsilon^{\text{pri}} \quad \text{and} \quad \|s^k\|_2 \le \epsilon^{\text{dual}},$$

  - 其中： $$\epsilon^{\text{pri}} = \sqrt{p}\,\epsilon^{\text{abs}} + \epsilon^{\text{rel}} \max\{\|Ax^k\|_2, \|Bz^k\|_2, \|c\|_2\},$$ $$\epsilon^{\text{dual}} = \sqrt{n}\,\epsilon^{\text{abs}} + \epsilon^{\text{rel}} \|A^T y^k\|_2,$$

奇异值分解：
https://www.zhihu.com/question/22237507


Attributed Network $\mathcal{G}$
Attributes
Network and Weights
$a_1 a_2 \ldots\ldots a_m$
n_1 n_2 n_3 n_4 n_5 n_6
A
Similarity
S
W
H
H^T
Split
Worker 1:
Worker t:
Problem ① s_1 = ? × H^T
Problem ② s_2 = ? × H^T
Problem ⑤ s_5 = ? × H^T
Problem ⑥ s_6 = ? × H^T
Updating
Problem ⑦ s_1^T = H × ?
Problem ⑧ s_2^T = H × ?
Problem ⑪ s_5^T = H × ?
Problem ⑫ s_6^T = H × ?
Converge
Latent Space

$$H = \begin{bmatrix} 0.54 & 0.27 \\ 0.22 & 0.91 \\ 0.55 & 0.28 \\ 0.98 & 0.11 \\ 0.32 & 0.87 \\ 0.26 & 0.11 \end{bmatrix} \begin{array}{l} n_1 \\ n_2 \\ n_3 \\ n_4 \\ n_5 \\ n_6 \end{array}$$

Figure 1: The basic idea of AANE is to represent nodes as continuous vectors, based on the decomposition of attribute affinity matrix and the penalty of embedding difference between connected nodes. The optimization is split into $2n$ sub-problems with low complexity, which can be solved separately by t workers.

4 Experiments

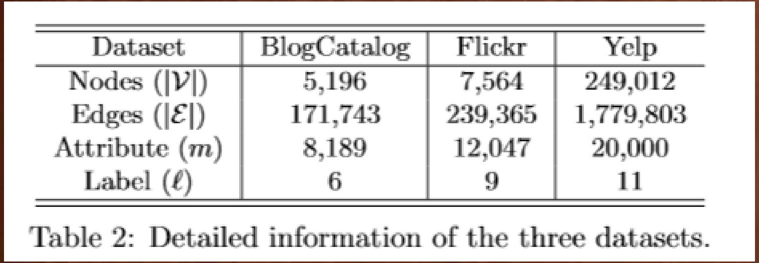
In this section, we empirically evaluate the effectiveness and efficiency of the proposed framework AANE. We aim at answering three questions as follows.

(1) What are the impacts of node attributes on network embedding?

(2) How effective is the embedding representation learned by AANE compared with other learning methods on real-world attributed networks?

(3) How efficient is AANE compared with the state-of-the-art methods? We first introduce the datasets used in the experiments.

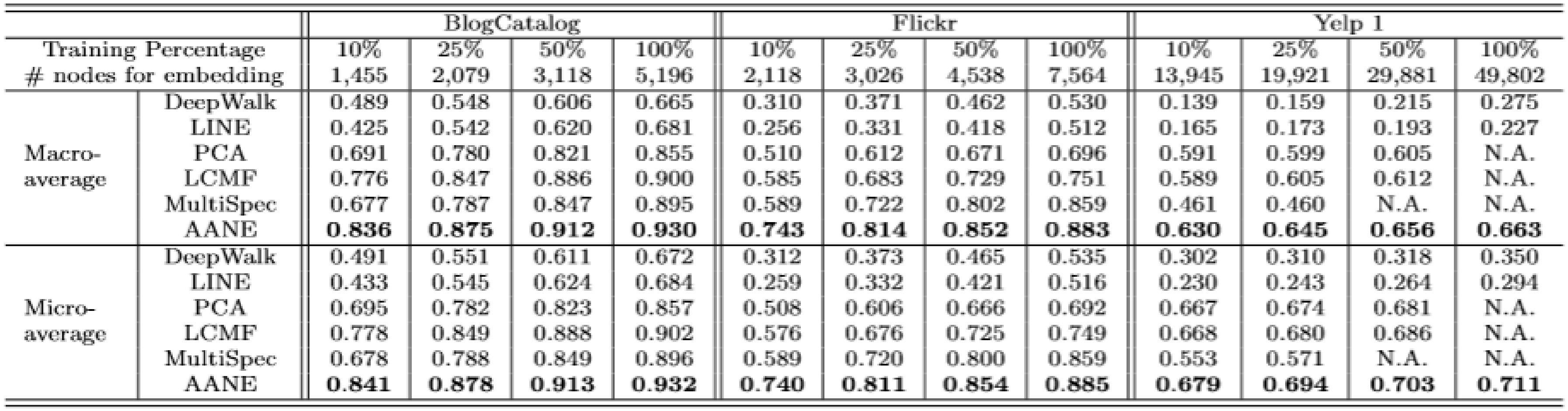
4.1 Datasets



Dataset	BlogCatalog	Flickr	Yelp
Nodes ($\|\mathcal{V}\|$)	5,196	7,564	249,012
Edges ($\|\mathcal{E}\|$)	171,743	239,365	1,779,803
Attribute (m)	8,189	12,047	20,000
Label (ℓ)	6	9	11

Table 2: Detailed information of the three datasets.

Classification performance



		BlogCatalog				Flickr				Yelp 1			
Training Percentage		10%	25%	50%	100%	10%	25%	50%	100%	10%	25%	50%	100%
# nodes for embedding		1,455	2,079	3,118	5,196	2,118	3,026	4,538	7,564	13,945	19,921	29,881	49,802
Macro-average	DeepWalk	0.489	0.548	0.606	0.665	0.310	0.371	0.462	0.530	0.139	0.159	0.215	0.275
	LINE	0.425	0.542	0.620	0.681	0.256	0.331	0.418	0.512	0.165	0.173	0.193	0.227
	PCA	0.691	0.780	0.821	0.855	0.510	0.612	0.671	0.696	0.591	0.599	0.605	N.A.
	LCMF	0.776	0.847	0.886	0.900	0.585	0.683	0.729	0.751	0.589	0.605	0.612	N.A.
	MultiSpec	0.677	0.787	0.847	0.895	0.589	0.722	0.802	0.859	0.461	0.460	N.A.	N.A.
	AANE	**0.836**	**0.875**	**0.912**	**0.930**	**0.743**	**0.814**	**0.852**	**0.883**	**0.630**	**0.645**	**0.656**	**0.663**
Micro-average	DeepWalk	0.491	0.551	0.611	0.672	0.312	0.373	0.465	0.535	0.302	0.310	0.318	0.350
	LINE	0.433	0.545	0.624	0.684	0.259	0.332	0.421	0.516	0.230	0.243	0.264	0.294
	PCA	0.695	0.782	0.823	0.857	0.508	0.606	0.666	0.692	0.667	0.674	0.681	N.A.
	LCMF	0.778	0.849	0.888	0.902	0.576	0.676	0.725	0.749	0.668	0.680	0.686	N.A.
	MultiSpec	0.678	0.788	0.849	0.896	0.589	0.720	0.800	0.859	0.553	0.571	N.A.	N.A.
	AANE	**0.841**	**0.878**	**0.913**	**0.932**	**0.740**	**0.811**	**0.854**	**0.885**	**0.679**	**0.694**	**0.703**	**0.711**

Table 3: Classification performance of different methods on different datasets with $d = 100$.

As shown in Tables 3, LCMF, MultiSpec and AANE outperform the two pure network embedding methods. AANE consistently achieves better performance than all baseline methods.

Efficiency Evaluation

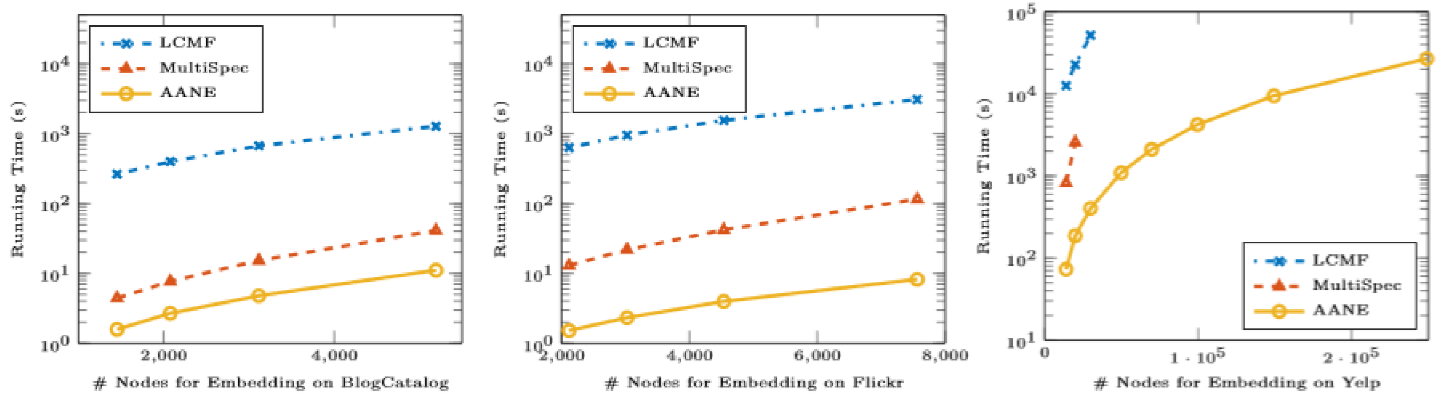


Figure 2: Running time of LCMF, MultiSpec and AANE w.r.t. the number of input nodes on the three datasets.

in Figure 2, we observe that our method AANE takes much less running time than LCMF and MultiSpec consistently. Furthermore, while the three methods are running in single-thread for a fair comparison, AANE could be implemented in multi-thread as illustrated in Figure 1. This strategy could further reduce the computation time of AANE.

4.6 Parameter Analysis

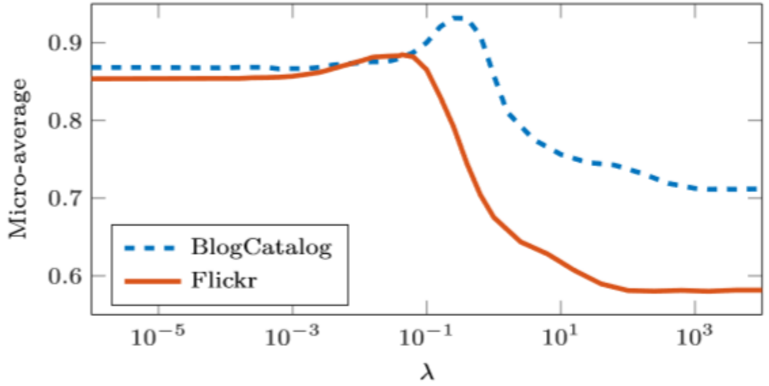


Figure 3: Impact of regularization parameter λ on the proposed framework AANE.

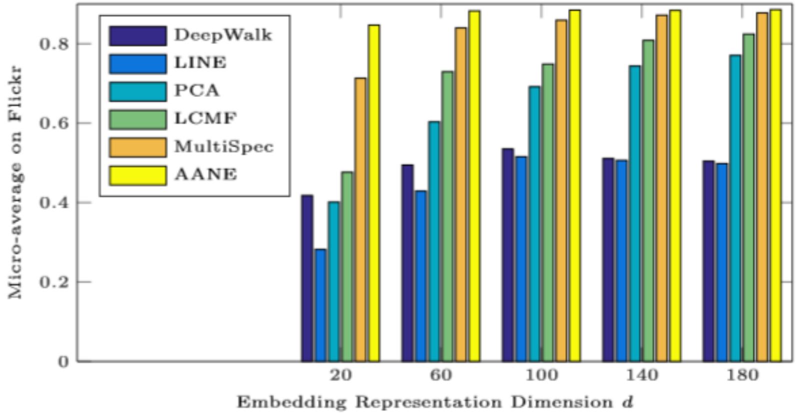


Figure 4: Classification performance on Flickr dataset w.r.t. the embedding dimension d.

Thank you