图嵌入系统的研究与方法综述

赵景林[1)]

[1)](华中科技大学 计算机科学与技术学院，武汉市 中国 430074)

摘 要 我们身处的世界关系都可以建模为图的结构，用图的节点或节点属性表示真实网络系统中的实体或实体标签，用图中的边表示真实网络中的实体关系。因此，通过图嵌入方法可以提高效率并且解决多种多样的下游任务。近些年不论是同时基于 CPU 和 GPU 还是单纯基于 CPU 的系统，都有很多高效率的图嵌入算法诞生。下面是对近些年比较经典的算法的简要综述，算法包括 NetSMF 、ProNE、PBG、 GraphVite 和 LightNE。研究并总结这些算法有助于我们了解图嵌入系统的前沿研究进展，进行更好地学习。

关键词 图嵌入；图神经网络；图嵌入系统

A review of research and methods for graph embedding systems

Jinglin Zhao[1)]

[1)](Huazhong University of Science and Technology, Wuhan, China)

Abstract The relationships in our world can be modelled as graph structures, with nodes or node attributes of the graph representing entities or entity labels in real network systems, and edges in the graph representing entity relationships in real networks. Thus, graph embedding methods can be used to improve efficiency and solve a wide variety of downstream tasks. In recent years, many efficient graph embedding algorithms have been created for both CPU and GPU based systems as well as purely CPU based systems. The following is a brief review of some of the more classic algorithms from recent years, including NetSMF, ProNE, PBG, GraphVite and LightNE, which can be studied and summarised to help us understand the cutting edge of graph embedding systems and learn better.

Key words Graph Embedding; Graph Neural Networks; Graph Embedding Systems

1 绪论

1.1 研究背景

电子商务和社交网络公司今天面临着分析和挖掘具有数十亿个节点和数百亿到数万亿条边的图形的挑战。近年来，一种流行的学习方法是应用网络嵌入技术来获得每个节点的向量表示。这些学习的表示或嵌入可以很容易地在下游的机器学习和推荐算法中使用。这些表示在各种在线服务中广泛使用频繁更新。例如，阿里巴巴的一个核心项目推荐系统拥有数十亿个项目和用户，当新用户和项目都在线时，需要频繁重新嵌入，并且必须快速重新计算基础嵌入。LinkedIn 的一个类似系统离线计算数百万个人（节点）的嵌入，必须定期重新发布该图以保持高精度。在这两种情况下，计算嵌入都必须以低延迟可伸缩的方式完成。

尽管对开发复杂的网络嵌入算法进行了大量研究，但使用可能牺牲大量准确性的简单和可扩展的嵌入解决方案仍然是行业中处理大规模图形的主要选择。例如，LinkedIn 使用 LINE[32]进行嵌入，这只捕获节点 1 跳邻域内的局部结构信息。阿里巴巴嵌入了一个 6000 亿节点的商品图，首先将其划分为 12000 个 5000 万节点的子图，然后用运行 DeepWalk 的 100 个 GPU 分别嵌入每个子图[21]。原因是在实践中，图形经常更新，嵌入算法通常需要每隔几个小时运行一次。

尽管文献中提出了许多新的嵌入系统，这些系

统证明了下游应用的高精度，但高延迟、有限的可扩展性和高计算成本阻止了这些技术在大规模数据集上的大规模部署或商业使用。例如，GraphVite 是基于 DeepWalk 的 CPU-GPU 混合系统，使用 4 个 P100 GPU 在 Friendster 图（65M 节点和 1.8B 边缘）上训练需要 20 小时。GraphVite 在该图上获得嵌入的成本是 210 美元，按云虚拟机租金计算。人们可以估计，使用 GraphVite 嵌入 10000 张这样的图（遵循阿里巴巴的方法），每次运行将超过 200 万美元，这是非常昂贵的。

1.2 研究现状

下面通过网络嵌入算法和网络嵌入系统两个方向的叙述相关工作。**网络嵌入算法。**在过去十年中，网络嵌入算法得到了广泛的研究。调查见。从优化方面来看，最近的网络嵌入算法可分为三大类。第一类使用通用随机梯度下降来优化损失函数，并遵循跳格模型框架。迄今为止，这些方法的样本效率和收敛速度的唯一界限需要进行额外的条件假设。第二类使用奇异值分解（SVD）来获得最佳低秩近。此类方法的示例包括 NetMF、NetSMF 和 ProNE。LightNE 也属于此类别。图神经网络代表了网络嵌入算法的第三种。此类方法包括 GCN、GAT 等。这些算法通常依赖于顶点属性以及监督信息。

网络嵌入系统。由于大型图嵌入带来的效率问题，现在已经开发了几种用于嵌入大型图的系统。GraphVite 是基于 DeepWalk[6]和 LINE[7]的 CPU-GPU 混合网络嵌入系统。它使用 CPU 进行图形运算，使用 GPU 进行线性代数运算。PyTorch BigGraph 是基于 DeepWalk 和 LINE 的分布式内存系统。它使用图形分区进行负载平衡，使用共享参数服务器进行同步。LightNE 是为共享内存机器设计的，在共享内存机器中，通信比分布式内存系统便宜得多。NetSMF 是一种基于稀疏矩阵分解的网络嵌入系统。

2 相关研究

2.1 NetSMF

稀疏矩阵分解（NetSMF）作为网络嵌入学习的解决方案。主要包括三个步骤。首先，它利用谱图稀疏化技术为网络的随机游动矩阵多项式找到稀疏化器。其次，它使用这种稀疏器来构造一个非零矩阵，该矩阵的非零数量明显少于原始 NetMF 矩阵，但在频谱上接近原始 NetMF。最后，它执行随机奇异值分解，以有效分解稀疏化的 NetSMF 矩阵，生成网络的嵌入。通过这种设计，NetSMF 提供了效率和有效性的保证，因为稀疏矩阵的近似误差在理论上是有界的。改论文作者在五个代表不同规模和类型的网络中进行实验。实验结果表明，对于百万规模或更大的网络，NetSMF 比 NetMF 实现了数量级的加速，同时保持了顶点分类任务的竞争性能。换句话说，NetSMF 和 NetMF 都优于公认的网络嵌入基准（即 DeepWalk、LINE 和 node2vec），但 NetSMF 解决了 NetMF 面临的计算挑战。

下面是具体的算法流程：

步骤 1：随机游走矩阵多项式稀疏化。算法通过重复 PathSampling 算法 M 次来构造具有 O（M）个非零的稀疏器。

Algorithm 2: PathSampling algorithm as described in [8].

1 **Procedure** PathSampling($e = (u, v), r$)
2 Uniformly pick an integer $k \in [r]$
3 Perform $(k-1)$-step random walk from u to u_0
4 Perform $(r-k)$-step random walk from v to u_r
5 Keep track of $Z(\boldsymbol{p})$ along the length-r path $\boldsymbol{p}$ between u_0 and u_r according to Eq. (7)
6 **return** u_0, u_r, $Z(\boldsymbol{p})$

图 1 NetSMF 的系统设计

步骤 2：构造 NetMF 矩阵稀疏器。在构造稀疏器$\tilde{L}$之后，我们可以将其插入到等式（1）中，以获得如等式（2）所示的 NetMF 矩阵稀疏器（算法 1，第 9-10 行）。该步骤不会改变稀疏器中非零的数量级。

$$\widetilde{M} = D^{-1}(D - \tilde{L})\,D^{-1} \quad (1)$$

$$trunc\_log\left(\frac{vol(G)}{b}\widetilde{M}\right) \quad (2)$$

其中$\tilde{L}$是经过稀疏采样后形成的拉普拉斯矩阵。

Algorithm 1: NetSMF

Input : A social network $G = (V, E, \boldsymbol{A})$ which we want to learn network embedding; The number of non-zeros M in the sparsifier; The dimension of embedding d.

Output: An embedding matrix of size $n \times d$, each row corresponding to a vertex.

$\widetilde{G} \leftarrow (V, \emptyset, \widetilde{\boldsymbol{A}} = \boldsymbol{0})$

/* Create an empty network with $E = \emptyset$ and $\widetilde{\boldsymbol{A}} = 0$. */

for $i \leftarrow 1$ **to** M **do**

  Uniformly pick an edge $e = (u, v) \in E$

  Uniformly pick an integer $r \in [T]$

  $u', v', Z \leftarrow \texttt{PathSampling}(e, r)$

  Add an edge $\left(u', v', \frac{2rm}{MZ}\right)$ to $\widetilde{G}$

  /* Parallel edges will be merged into one edge, with their weights summed up together. */

end

Compute $\widetilde{\boldsymbol{L}}$ to be the unnormalized graph Laplacian of $\widetilde{G}$

Compute $\widetilde{\boldsymbol{M}} = \boldsymbol{D}^{-1}\left(\boldsymbol{D} - \widetilde{\boldsymbol{L}}\right)\boldsymbol{D}^{-1}$

$\boldsymbol{U}_d, \Sigma_d, \boldsymbol{V}_d \leftarrow \texttt{RandomizedSVD}\left(\text{trunc\_log}^{\circ}\left(\frac{\text{vol}(G)}{b}\widetilde{\boldsymbol{M}}\right), d\right)$

return $\boldsymbol{U}_d\sqrt{\Sigma_d}$ as network embeddings

图 2　NetSMF 算法

步骤 3：截断奇异值分解。最后一步是对构建的 NetMF 矩阵稀疏器（等式（2））执行截断奇异值分解（SVD）。然而，即使只有 O（M）个非零稀疏器，执行精确的 SVD 仍然很耗时。在这项工作中，我们利用了现代随机矩阵近似技术随机化 SVD。由于空间限制，该算法无法包含许多细节。简而言之，该算法通过高斯随机矩阵将原始矩阵投影到低维空间。只需要在 d×d 小矩阵上执行传统 SVD（例如 Jacobi SVD）。

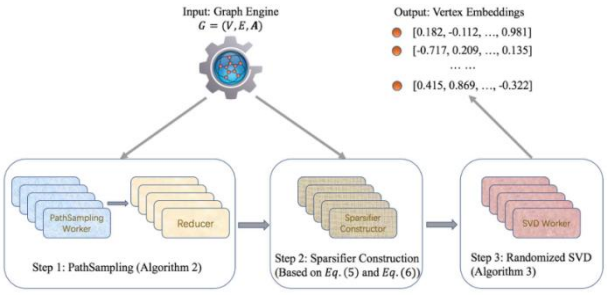

图 3 NetSMF 的系统设计

我们通过实验结果图 4 进行 NetSMF 和 NetMF 的比较，因为 NetSMF 的目标是解决 NetMF 的效率和可扩展性问题，同时保持其效率优势。对于 YouTube 和 OAG（两者都包含超过 100 万个顶点），NetMF 由于空间和内存消耗过大而无法完成，而 NetSMF 能够分别在四小时和一天内完成。对于中等规模的网络 Flickr，这两种方法都可以在一周内完成，但是 NetSMF 速度快 2.5 倍（即 48 分钟比 2 小时）。对于小型网络，NetMF 在 BlogCatalog 中比 NetSMF 更快，在运行时间方面与 PPI 中的 NetSMF 相当。这是因为当输入网络仅包含数千个顶点时，稀疏矩阵构造和因子分解相对于其密集替代方案所带来的优势，可能会被工作过程中的其他组件边缘化。

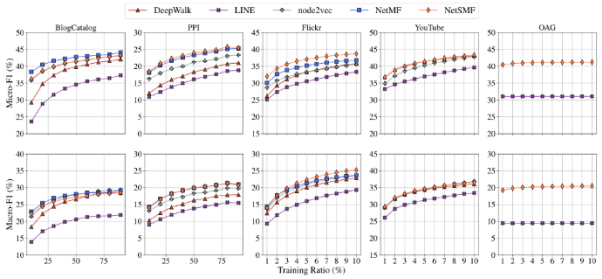

图 4　实验结果展示

2.2　ProNE

为了（预）训练非常大规模的网络嵌入，清华大学的教授提出了 ProNE：一个快速、可扩展和有效的模型，其单线程版本比 20 个线程的高效网络嵌入基准标记快 10 - 400 倍，包括 LINE、DeepWalk、node2vec、GraRep 和 HOPE。举一个具体的例子，单线程的 ProNE 只需要 29 个小时就能嵌入一个上亿个节点的网络，而使用 20 线程的 LINE 需要几周，Deep-Walk 需要几个月。为了实现这个目标，ProNE 首先通过将任务表述为稀疏矩阵分解，对网络嵌入进行高效的初始化。ProNE 的第二步是通过在频谱调制空间中传播嵌入的方式来实现嵌入。在不同规模和类型的网络上进行的大量实验表明，与上述基线相比，ProNE 实现了有效性和显著的效率优势。此外，ProNE 的嵌入增强步骤也可以在速度上通用于改进其他模型，例如，对于所使用的基线，相对增益大于 10%。

ProNE 主要由展示在图 5 中的两个步骤组成。首先，它将图嵌入表示的问题化为稀疏矩阵分解问题，以高效地实现初始的结点的嵌入表示。接着，它借助高阶的 Cheeger 不等式来调制图的谱空间，然后把第一步学到的嵌入表示在调整后的图上传播，从而达到将局域的平滑信息（localized smoothing information）和全局的聚类信息（global clustering information）融合进图嵌入表示中。

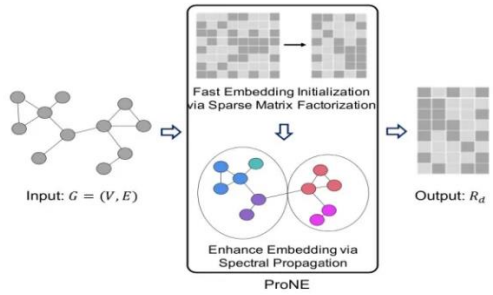

图 5 ProNE 模型。1）用于快速嵌入初始化的稀疏矩阵分解，和 2）用于嵌入增强的调制网络中的频谱传播。

与其他流行的方法类似，例如 DeepWalk 和 LINE，通过稀疏矩阵分解学习的图嵌入表示一般仅能捕获局域的结构信息。为了进一步整合全局网络属性，例如社区结构，三角形结构，等等，作者提出可以通过根据高阶 Cheeger 不等式对图结构进行谱空间的调制，使得调整后的图更加强调局域的平滑信息和全局的聚类信息，然后将上一节中训练得到的初始图嵌入在在新网络中传播，就可以很自然地整合了这些高阶的结构（通过传播，结点的图嵌入会共享到和它同属于一个高阶结构的点的嵌入信息）。因为整个传播过程的关键是先对图做谱空间的调制，所以将整个步骤命名为“谱传播”。

注意到谱传播策略是一种通用的将高阶的图信息进一步整合进图嵌入的方法，它一般不依赖所传播的初始图嵌入的具体训练方式。所以，谱传播可以用来提升现有的图嵌入算法。

Algorithm 1 ProNE

输入: $G = (V, E)$, 维数 d, 位置参数 μ, 尺度参数 θ

输出: 图嵌入表示矩阵 R_d

1: 依据式子 (5-13) 构建相似度矩阵 M
2: 生成高斯随机矩阵 Ω, 计算得到 $Y = M\Omega$
3: 对 Y 进行 QR 分解, 分解结果为 $QR = Y$，即找到了算法需要的 Q
4: 对 $H = Q^T M$ 进行 SVD 分解，$S_d\Sigma_d V_d^T \leftarrow \text{SVD}(H, d)$
5: $R_d \leftarrow QS_d\Sigma_d^{1/2}$
6: 通过式子 (4-21) 以 Chebyshev 的递归方式 (4-22) 对 R_d 进行谱传播
7: 通过 SVD 对 R_d 进行正交化
8: **返回**图嵌入表示矩阵 R_d

图 6 ProNE 算法

算法 ProNE 的复杂度为 O(|V|d2 + k |E|), 所以算法非常高效。关于空间复杂度，由于算法 ProNE 在计算过程中只涉及到稀疏矩阵和图嵌入矩阵的存储，所以空间复杂度是 O(|V|d + |E|)

2.3 PBG

PyTorch BigGraph（PBG），这是一种图嵌入系统，它对传统的多关系嵌入系统进行了一些修改，使其能够扩展到具有数十亿节点和数万亿条边的图。PBG 使用图形划分来训练单个机器或分布式环境中任意大的嵌入。在通用基准测试上展示了与现有嵌入系统相当的性能，同时允许缩放到任意大的图形并在多台机器上进行并行化。

点和边的划分。在图 7（左），节点分为 P 个分区，其数量根据内存调整。边根据源节点和目标节点的划分划分为桶。在分布式模式下，可以并行执行具有非重叠分区的多个桶（红色方块）。在图 7（中），节点类型较少的话可以不必分区;如果用于尾节点的所有节点都是未分区的，则边缘可以仅基于源节点分区划分为 P 个桶。在图 7（右），“由内而外”的桶顺序保证了桶至少有一个以前训练过的嵌入分区。根据经验，这种排序比其他方式（如随机)产生更好的嵌入。

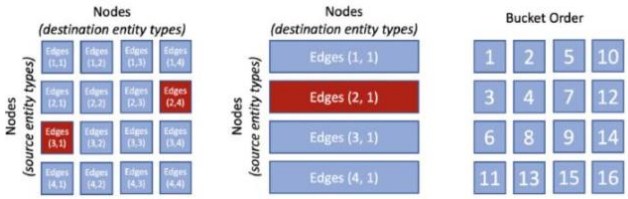

图 7 PBG 划分方案示例

分布式执行。现有的分布式嵌入系统通常使用参数服务器架构。在这个架构中，参数服务器包含嵌入的键值存储。在每次 SGD 迭代时，每一批数据都需要从参数服务器请求所需的嵌入参数，然后将梯度发送到服务器更新参数。参数服务器范式对于训练大型稀疏模型是有效的，但它有很多缺点。一个问题是基于参数服务器的嵌入框架需要太多的网络带宽，因为每批量的边及其相应的负样本的嵌入必须在每个 SGD 步骤传递更新。此外，我们发现高效的搜寻是必须以使相同的模型可以在一台机器上运行或分布式执行，但参数服务器架构限制可以在单个计算机上运行的模型的大小。给定分块后的节点和边，PBG 使用一个并行化方案，该方案将锁定分块后的嵌入，多个桶可以平行训练，只要当它们在不相交的分区集上操作时，如中所示图 7（左）。训练可以并行在 P/2 台机器上。

图 8 为 PBG 分布式模块框图。箭头表示 Rank 2 训练器为训练一个分块而进行的通信。首先，训练器从 Rank 1 上的锁定服务器请求一个桶，该桶锁定对应的分区。然后，训练器会保存不再使用的所有分区，并从分片服务器加载所需的新分区，此

时它可以在锁服务器上释放其旧分区。然后从共享文件系统加载边，并且在没有线程间同步的情况下在多个线程上进行训练。在单独的线程中，少量共享参数与分片参数服务器持续同步。模型检查点偶尔会从训练器写入共享文件系统。

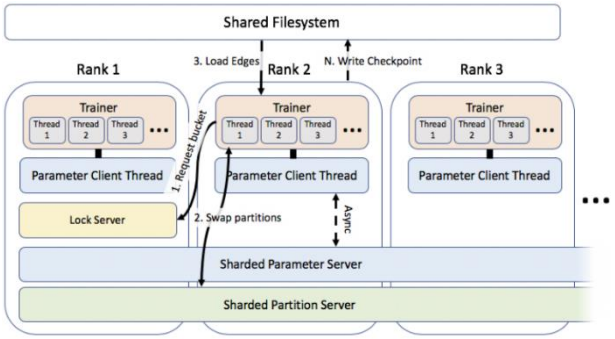

图 8　PBG 分布式架构

分批负采样。小批量的 Bn 采样源节点或目的节点可以重用来构造多个负样本。在典型设置中，PBG 接受来着自训练集的边的一批 B=1000 的正样本，将其分成 50 条边的块。目的节点（等价，源节点）的嵌入从每个块中连接 50 个嵌入（从尾节点类型中均匀采样）。具有 200 个采样节点的 50 个正样本的外积中等效于 9900 个负样本。训练计算如图 9 所示。这种方法比给每批都抽样负样本代价更低。对于每批 B 个正样本，3B 嵌入从内存中获取和以及 3B*Bn 条边的分数（点积）被计算。每一批的边的分数可以计算为一个批次的 Bn×Bn 矩阵乘法，可以更高效执行。

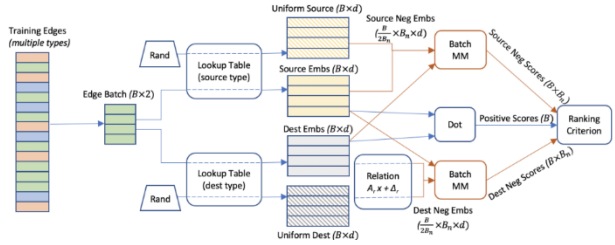

图 9　内存高效的成批负采样

2.4　GraphVite

图嵌入算法通常包括两个阶段：网络增强和嵌入训练。网络增强阶段先对原始网络用随机游走产生增强网络，然后从增强网络中采样正边，这一阶段涉及到大量的随机访问，而 CPU 在随机访问上 GPU 性能好出太多，所以网络增强阶段应该放在 CPU 上进行。嵌入训练主要是通过上一阶段生成的样本训练节点嵌入。这一阶段主要的运算是矩阵计算，所以应该放在 GPU 上。

系统的整体架构如图 4 所示：CPU 负责图结构和完整的参数矩阵，进行正样本采样；GPU 存储部分参数矩阵，进行梯度下降和负样本采样；CPU 和 GPU 之间通过样本池传输样本数据。

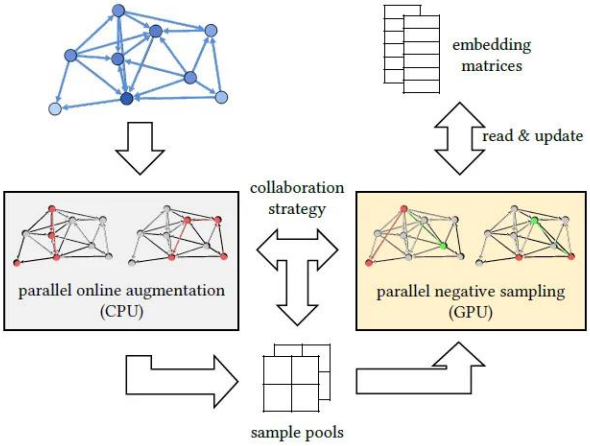

图 10 GraphVite 系统架构

GraphVite 的主要贡献有：（1）并行在线增强（2）并行负采样（3）协同策略。

并行在线增强。为了解决增强网络太大这个问题，GraphVite 直接生成增强边样本，从而避免了生成增强网络：首先选中一个节点作为出发点，然后用随机游走算法生成一个长度为 s 的样本。相同的随机游走（算法）生成的边样本之间往往具有相关性，这会影响最后的优化性能。为了避免这个问题，用一个样本池来收集所有的样本，然后随机打乱样本池中的样本，再传输到 GPU 中进行训练。当有多个 CPU 时，给每个线程预先分配一个样本池，然后进行并行计算。整个过程伪代码如图 11 所示。

用随机算法打乱样本池的样本这个操作会大幅降低网络增强阶段的速度——因为随机内存访问这一步不能通过 CPU 缓存加速。但值得注意的是，最具相关性的边样本在相同的随机游走中共享源节点或目标节点。将样本池分成个连续的块，将相关样本分到不同的块中，然后每次都将样本添加到各个块的末尾，这样就能利用 CPU 缓存加速了。最后把各个块拼起来，组成最终的样本池。

并行负采样。在训练阶段，将训练任务分割为 n（n 是 GPU 数量）部分，并将这 n 部分放到 n 个 GPU 中并行计算。由于节点嵌入训练的稀疏性，在网络中存在大量梯度可交换的训练数据对，所以定义了ε梯度可交换。例如，对于数据X_1和X_2，只要它们不共享任何源节点和目标节点，那它们就是梯度可交换的。即X_1和X_2共享某些节点，在有限的迭代

次数和限制的学习率下它们仍然是ε梯度可交换的。

Algorithm 2 Parallel Online Augmentation

```
function ParallelOnlineAugmentation(num_CPU)
    for i ← 0 to num_CPU − 1 do                    ▷ paralleled
        pool[i] ← ∅
        while pool is not full do
            x ← DepartureSampling(G)
            for u, v ∈ RandomWalkSampling(x) do
                if Distance(u, v) <= s then
                    pool.append((u, v))
                end if
            end for
        end while
        pool[i] ← Shuffle(pool[i])
    end for
    return Concatenate(pool[·])
end function
```

图 11 并行在线增强算法伪代码

基于节点嵌入中的梯度可交换性，GraphVite 提出并行负采样算法。图 12 是一个典型例子：对 n 个 GPU，将顶点矩阵和上下文矩阵分别分割成 n 部分，于是样本池被分 n×n 个块，每条边属于其中一块。不同行或列的块是梯度可交换的，相同行或列的块是 梯度可交换的。在每一步中，选取 n 个正交块和它们对应的顶点部分、上下文部分放到 n 个 GPU 中。在每个 GPU 中进行异步梯度下降，然后再将块传回 CPU。另外，由于各个 GPU 里的数据几步共行也不共列，负采样只在 GPU 内部进行，这样就不用进行 GPU 间同步数据。这也是并行负采样名字的由来。

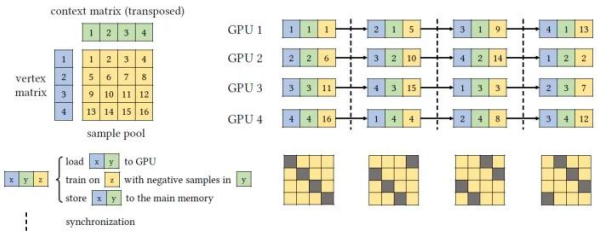

图 12 4GPU 并行负采样示例

协同策略。由于 CPU 和 GPU 是共享样本池的，这就意味着，当 GPU 在工作时 CPU 就会闲置；反之，CPU 工作时 GPU 也会闲置。对于这个问题，在主存中配置两个样本池，当第一个样本池满了之后送到 GPU 进行训练，然后在 CPU 上进行第二个样本池的并行在线增强，同时在 GPU 中进行并行负采样。当第二个样本池满了之后，交换两个样本池。

2.5 LightNE

分布式架构和 GPU 是高质量大规模网络嵌入所必需的，但是该文章与主流观点不同，作者证明了使用共享内存、仅 CPU 的架构，依然可以实现更高的质量、更好的可扩展性、更低的成本和更快的运行时间。LightNE 结合了 NetSMF 和 ProNE 两种理论中的嵌入方法。并首次将以下技术引入到网络嵌入中：（1）一种新提出的下采样方法，以降低 NetSMF 的采样复杂度，同时保持其理论优势；（2）高性能并行图形处理堆栈 GBBS，以实现高内存效率和可扩展性；（3） 稀疏并行哈希表，以在存储器中聚合和维护矩阵稀疏器；以及（4）Intel MKL，用于高效的随机化 SVD 和频谱传播。

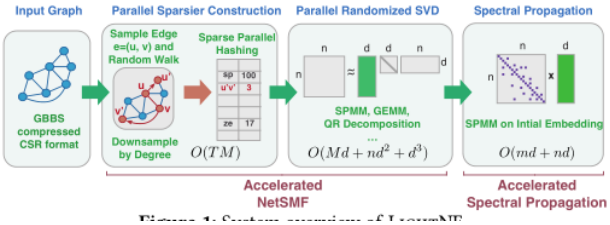

图 13 LightNE 的系统概述。

从算法设计的角度来看，LightNE 设计结合了 NetSMF 和 ProNE。第一步是使用新的边缘下采样算法的 NetSMF，这显著提高了样本复杂性。第二步是使用 ProNE 的频谱传播增强 NetSMF 嵌入。

稀疏并行图形处理。通过调用 GBSS 库中的函数，将图压缩为 CSR 格式，并采用了 Ligra+中的并行字节格式。Ligra+中的并行字节格式将高阶顶点的邻居划分为块，其中每个块包含可配置数量的邻居。每个块相对于源进行内部差分编码。

Algorithm 2: Downsampled Per-Edge PathSampling.

```
Procedure DownSampledPerEdgePathSampling(G, T)
    G.MapEdges(function (e = (u, v)) →
        n_e ← ⌊M/m⌋+random variable from Bernoulli ({M/m})
        for i ← 1 to n_e do
            p ← Uniform[0, 1]; r ← Uniform[1, T]
            if p < p_e then
                (u', v') ← PathSampling(G, u, v, r)
                Add (u', v') with weight 1/p_e to the sparsifier )
```

图 14 边的下采样算法。

GBBS 的并行每边路径采样采用 NetSMF 中的 PathSampling 算法。PathSampling 算法在第一篇文章的综述中已经详细介绍了，通过一定的概率构建一定权重的稀疏采样的边集。。采样完边之后，使用稀疏并行哈希表进行存储，并采用线性探测来解决冲突问题。

随机 SVD 与 Spectral Propagation。随机化 SVD 加速了运算过程，减少了内存的使用。由于该过程涉及过多的线性代数运算，而“英特尔 MKL”库非常支持并高度优化了这些运，所以直接调用该库完成 SVD 算法。Spectral Propagation 采用的是 ProNE 中的谱增强算法，步骤也涉及线性代数运算并且是高效的。它不需要计算 L 的高次幂，而是只在稀疏矩阵之间应用重复的稀疏矩阵乘法（SPMM），这些步骤也可以由 MKL 的稀疏 BLAS 例程处理完成。

作者在两个非常大的 1000 亿规模的图数据集 ClueWebSym 和 Hyperlink2014Sym 上测试了 LightNE，现有的网络嵌入系统中没有一个能够在单个机器中处理如此大的图，能够处理这种量级的图，依靠的是 CBSS 的压缩技术、NetSMF 中的下采样算法以及随机 SVD 的使用。

3 总结和展望

近几年，图数据处理引起了众多学者的广泛关注，节点分类和图分类是复杂网络研 究中的重要研究分支。目前大部分任务都是处理图结构数据，为了完成图相关任务，首 先需要将图中节点和边的信息学习为低维向量，从而完成下游机器学习任务。

虽然图嵌入算法在处理高维稀疏数据、计算效率和嵌入效果上已有大幅提升，但面对不断发展和变化的数据及应用要求，图嵌入算法仍需进一步发展和创新。

NetSMF 这篇文章以实现效率和有效性为目标来研究网络嵌入。为了解决 NetMF 模型面临的可扩展性挑战，作者建议将大规模网络嵌入研究为稀疏矩阵分解。我们提出了 NetSMF 算法，它实现了（密集）NetMF 矩阵的稀疏化。稀疏矩阵的构造和分解都足够快，可以支持非常大规模的网络嵌入学习。广泛的实验结果表明，NetSMF 稀疏学习的嵌入与 NetMF 矩阵分解的嵌入一样有效，因此它优于常见的网络嵌入基准 DeepWalk、LINE 和 node2vec。

ProNE 是一个在大规模图结构数据上快速可扩展的图嵌入算法。首先，ProNE 将图嵌入表示的问题化为稀疏矩阵分解问题，以高效地实现初始的结点的嵌入表示，从而刻画结点的分布相似度。然后，它借助高阶的 Cheeger 不等式来调制图的谱空间，把第一步学到的嵌入表示在调整后的图上传播，从而达到将局域的平滑信息和全局的聚类信息融合进图嵌入表示中。

ProNE 在运行效率和精度效果上都比现在流行的基线算法，比如 DeepWalk, LINE, node2vec, GraRep 和 HOPE，都要更优秀。最亮眼的是，主体单线程的 ProNE 模型比 20 个线程加速的基线算法们快上大约 10—400 倍。

PyTorch BigGraph 是一个嵌入系统，可以扩展到具有数十亿个节点和数万亿条边的图。PBG 支持多实体、多关系图，具有每个关系配置，如边权重和关系运算符的选择。为了节省内存使用并允许并行化，PBG 将邻接矩阵进行块分解，将其分解为 N 个桶，每次对一个桶的边缘进行训练。

用 PBG 训练的嵌入的质量与现有的嵌入系统相当，而且需要的训练时间更少。实验表明，对 Freebase 图进行分区可以在不降低嵌入质量的情况下减少 88%的内存消耗，并且在 8 台机器上的分布式执行可以将训练速度提高 4 倍。

GraphVite 是一个通用的图嵌入系统/框架，主要应用在多 CPU 多 GPU 快速/超大规模训练上。这是一种单机框架，不支持分布式集群，主要是为没有分布式集群的用户打造。最大特点是速度快：与目前最快的框架相比，GraphVite 在性能几乎没有任何牺牲的情况下，速度提高了约 50 倍。其次，GraphVite 在单机四卡上，最大可以训练一张 6 千万点，18 亿边的图，训到收敛也只需要不到 1 天的时间。

LightNE 与最近流行的三个网络嵌入系统 GraphVite、PyTorch BigGraph 和 NetSMF 相比，在大规模图嵌入上展现出了前所未有的性能。LightNE 结合了两种先进的网络嵌入算法 NetSMF 和 ProNE，在九个基准图形数据集上实现了最先进的性能。通过结合稀疏并行图形处理技术和其他并行算法技术（如稀疏并行哈希和高性能并行线性代数），LightNE 能够在几个小时内学习具有数千亿条边的图形的高质量嵌入，所有这些都是以适中的成本完成的。

未来，作者计划研究减少哈希表和 SVD 实现上的内存瓶颈的技术。为这些数据结构设计高效的压缩技术是一种很有前途的研究方向，可以为大规模的真实网络实现更好的性能。作者还想研究流媒体或动态环境中的大规模网络嵌入问题。

参 考 文 献

[1] Qiu J, Dong Y, Ma H, et al. Netsmf: Large-scale network embedding as sparse matrix factorization[C]//The World Wide Web Conference. 2019: 1509-1520.

[2] Lerer A , Wu L , Shen J , et al. PyTorch-BigGraph: A Large-scale Graph Embedding System[J]. 2019.

[3] Zhu Z , Xu S , Qu M , et al. GraphVite: A High-Performance CPU-GPU Hybrid System for Node Embedding:, 10.1145/3308558.3313508[P]. 2019.

[4] Qiu J , Dhulipala L , Tang J , et al. LightNE: A Lightweight Graph Processing System for Network Embedding[C]// SIGMOD/PODS '21: International Conference on Management of Data. 2021.

[5] Perozzi B, Al-Rfou R, Skiena S. Deepwalk: Online learning of social representations[C]//Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining. 2014: 701-710.

[6] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. 2015.Line: Large-scale information network embedding. In WWW ’15. 1067–1077.