图嵌入技术研究综述

摘　要 图是一种重要的数据表示形式，也出现在各种各样的现实场景中，基于图的分析可以使用户能够更深入的了解数据背后的含义，从而通过相应的应用程序处理。问题在于如今大多数基于图的分析计算和空间成本高昂。图嵌入是一种基于图随机游走的重要应用，用来进行节点分类，链接预测、节点推荐等，它是将图数据转换到一个低维空间中，在该空间中最大限度地保存图的结构信息和属性。在本文综述中，我们先介绍图嵌入的定义以及相关概念，然后说明图嵌入相关方向的研究和应用场景。通过近年来三篇文章的三个不同研究方向来具体说明图嵌入相关技术的最新进展，一是网络嵌入方向；二是随机游走方向；三是 CPU-GPU 混合系统方向。最后，总结图嵌入的相关应用并展望。

关键词 图嵌入；图数据；网络嵌入；随机游走；混合系统

引言

图在如今的现实场景中被广泛运用，例如社交媒体网络中的社交图、用户兴趣图、研究领域中的引用图，还有如今火热的知识图。通过分析这些图形可以深入了解如何利用图形中的隐藏信息，正因为可以分析一些现实应用，所以在过去几十年中图分析受到了广泛的关注。有效的图分析可以使许多应用程序受益，如节点分类、节点聚类、节点推荐以及链接预测等等。通过分析社交网络(如微博、QQ)中用户交互构建的图，我们可以对其中的用户进行分类，检测相应的社区、推荐可能的朋友，并预测两个用户之间是否会发生相应的交互。

图 1 现实图型转为图处理用的数据

尽管图分析非常实用，但现有的图分析方法都存在则计算量大和空间开销大的问题，目前大量的研究致力于如果高效地进行图分析，并降低相应的成本，如一些图数据处理框架(GraphX，GraphLab)、一些节省空间的图存储格式(CSR、DIA)等。

图嵌入技术通过将图转换到低维空间，即一种将图数据（通常为高维稠密的矩阵）映射为低微稠密向量的过程，能够很好地解决图数据难以高效输入机器学习算法的问题。由于图的类型不同，可以大致分为四个类型，分别是同构图、异构图、辅助信息图、非关系数据构建图，因此图嵌入的输入在不同场景下是不同的，图嵌入的输出也可能表示图的一部分或者整个图的低维向量，输出也分为四种类型，分别是节点嵌入、边嵌入、混合嵌入以及全图嵌入。

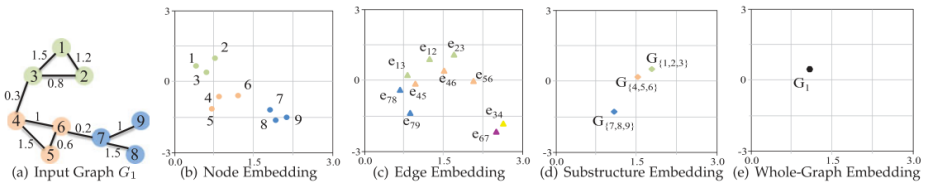

图 2 不同的粒度将图嵌入为 2D 空间

早期图嵌入算法主要是通过假设数据位于低维流形中降低非关系数据的高维性。给定一组非相关的高维数据特征，基于成对特征相似度构造相似图，然后将图中每个节点嵌入到低维空间中。

自从 2010 年开始，随着图在各个领域的大量普及，图嵌入的研究开始以图为主，利用相应的辅助信息来促进嵌入。一方面，其中一些关注于图的一部分(如节点、边、子结构)表示为一个向量。为了获取这种嵌入，要么采用最先进的深度学习，要么设计目标函数以优化边缘重构概率。另一方面，也有一些工作将整个图作为一个整体向量嵌入到以图为级别的应用程序中，例如蛋白质结构作为一个整体观察蛋白质之间的相似性等。

图嵌入与两个传统的研究问题相关，分别是图分析和表征学习。图分析旨在从图形数据中挖掘有用的信息，而表征学习则是为了获得数据表征，从而在构建分类器或预测器时能够更容易提取有用的信息。图嵌入在这个两个问题的汇交处，并侧重于学习低维表征。图表征学习不同于图嵌入，前者不需要将表征变为低维向量。

将图嵌入到低维空间是一项复杂的工作，图嵌入挑战取决于问题的设置，包括嵌入输入和嵌入输出。不同类型的嵌入输入携带不同的信息保存在嵌入空间中，例如当嵌入仅包含结构信息的图时，节点之间的连接关系需要保留在低维空间中；又或者对于具有节点标签或属性信息的图时，嵌入过程中可能要考虑这些信息的使用。对于嵌入输出是任务驱动的，最常见的嵌入输出类型为节点嵌入，将相近的节点表示为类似的向量。节点嵌入可以方便一些如节点分类、节点聚类的任务处理；然而一些任务与图的更高粒度挂钩，例如子图和整图。

原理和优势

图嵌入的输入和输出各有四种基本类型，每个类型都有各自的特性和优势。

图嵌入输入

1. 同质图

同质图可以进一步分为加权(或有向)和非加权(或无向)图，无向非加权图是最基本的图嵌入输入。图 3(a)V1 到 V2 的权重比 V1 到 V3 权重更大，在嵌入到低维空间中 V1 和 V2 两点间的距离更近，图 3(b)V4 和 V5 是双向连接，V6 和 V4 是单向连接，在嵌入到低维空间中 V4 和 V5 距离更近。

图 3 权重图和有向图

同质图在社交用户链接中，每个用户与其他用户都有相应的链接关系，并且可以添加相应的权重信息(用户的被关注度等)，从而构建基本的有向权重图。

2. 异构图

异构图主要存在于三种场景中：一是基于社区的问答网站，即用户在网站发布问题，由其他用户回答，节点包含三种类型(问题、答案、用户)；二是多媒体网络，多

媒体网络包含多媒体数据，例如图像文本等，嵌入包含两种类型的节点(图像和文本)和三种类型的链接(图像-文本、图像-图像、文本-文本)；三是知识图谱，

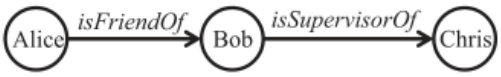

图 4 知识图谱举例

图 4 中可以观察到实体(节点)和关系(边)通常具有不同的类型，实体类型包含 Alice、Bob 和 Chiris，关系类型包含朋友、监护人。

3. 辅助信息图

辅助信息图除了包含节点的结构关系外，还包含节点、边或全图的辅助信息，一般辅助信息主要有五种类型属性，分别为：

标签：具有不同标签的节点嵌入后应该远离彼此，可以通过对不同标签节点之间相似度进行相应的惩罚。例如书有多个标签子类别，“作者”和“价格”等。

属性：与标签不同的是，属性值可以是连续的，在将节点属性表示为高维向量时(社交网络用户属性特征)，可以考虑处理节点和边的离散和连续属性。

节点特征：大多数节点特征均为文本，要么作为每个节点的特征向量，要么作为文档，后者可以利用词包、主题建模等技术对文档处理从而提取特征向量。

信息传播：节点收到信息并且能够传播给他们的邻居，模型可模拟用户如何影响网络中的每个人

知识库：比如百度百科、Wikipedia 等，概念为用户提出的实体，文本是与该实体相关联的文章，通过知识库可以从网络中学习相应的知识图。

4. 非关系数据构建图

假设输入数据位于低维流形时，通常通过不同的策略从非关系输入数据构造嵌入，而非直接由图提供嵌入数据。

图嵌入输出

1. 节点嵌入

最常见的嵌入输出设置，将每个节点表示为一个低维空间的向量，不同的图嵌入方法主要区别在于如何定义两个节点之间的相似性。计算成对节点相似度的两种常用度量为一阶相似度和二阶相似度。

2. 边嵌入

将一条边表示为一个低维向量，主要应用场景如知识图嵌入，三元组<h,r,t>需要学习将关系 r 保留在 h 和 t 两个实体中，在只知道两个分量的情况下可以正确预测缺失的实体或关系，即链接预测、关系预测等。

3. 混合嵌入

混合嵌入是将不同类型的图元素如节点+边，节点+社区等组合在一起进行嵌入、前者在两个可能遥远的节点间嵌入图结构，用来支持语义相似搜索；后者考虑社区感知的相似性，可以用来评估网络的相应成员之间的相似性的度量。

4. 全图嵌入

通常是小图，例如蛋白质、分子等。在这种情况下，一个图表示为一个向量，全图嵌入为图分类任务提供了一种简单、高效的计算图相似性的方法。

研究进展

网络嵌入方向

网络嵌入最新进展大致分为三类：基于矩阵分解的方案；基于 skip-gram 的模型；图神经网络。前两种方法预先训练的嵌入被输入到下游任务的学习模型中。因此，为了服务于大规模的实际网络应用，高效地生成有效的网络嵌入是至关重要的。然而，现有的大多数模型侧重于提高嵌入的有效性，使其效率和可扩展性受到限制。例如在基于矩阵分解的模型中，GraRep 的时间复杂度为 $O(n^3)$，其中 n 是网络中的节点数，这使得大型网络的计算成本过高；在基于 skip-gram 的模型中，使用默认参数设置，学习 1 亿个节点和节点组成的网络的嵌入以 LINE、DeepWalk、node2vec 为模型至少也需要以周为单位的时间。

基于现有模型嵌入的效率和可扩展较差，目前研究中提出一种用于大规模嵌入的快速和可扩展的模型——ProNE。

ProNE 总体思想是首先以有效的方式初始化网络嵌入，然后增强这些嵌入的表示能力。受大多数真实网络的长尾分布及其产生的网络稀疏性的启发，第一步通过将网络嵌入表示为稀疏矩阵分解来实现；第二步是利用高阶 Cheeger 不等式对初始嵌入进行谱传播，以获取网络的局部平滑和全局聚类信息。

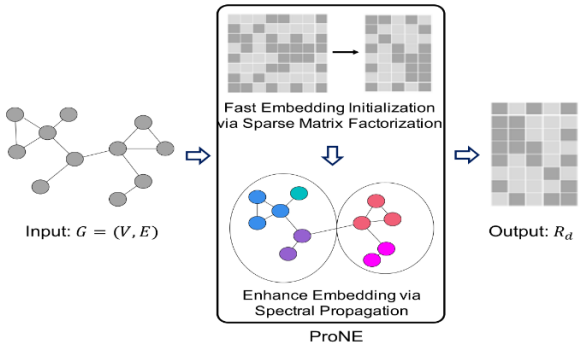

图 6 ProNE 模型

如图 6 所示，ProNE 模型包含两个部分：1）用于快速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谱传播

第一步矩阵分解基于奇异值分解完成，节点相似性通常由结构-上下文建模。利用最简单的结构-边来表示节点-上下文对。基于分布相似性的网络嵌入目标被转化为矩阵分解，使用谱方法截断奇异值分解最后获得特征向量。

由于嵌入只能捕获局部结构信息，为了进一步结合全局网络特性，即社区结构，所以受高阶 Cheeger 不等式启发，在频谱调制网络中传播初始嵌入。谱传播是增强网络嵌入的一般框架，可以用由 LINE、node2vec 等生成的嵌入作为输入，谱传播策略为为所有嵌入提供了 10%以上的相对改进。

表 1 基于运行时间(s)的的效率对比

Dataset	*DeepWalk*	*LINE*	*node2vec*	*ProNE*
PPI	272	70	828	**3**
Wiki	494	87	939	**6**
BlogCatalog	1,231	185	3,533	**21**
DBLP	3,825	1,204	4,749	**24**
Youtube	68,272	5,890	>5days	**627**

表 1 显示了 ProNE 和三条现有最快基准 DeepWalk、LINE、node2vec 的运行时间（IO 和计算时间）对比。DeepWalk 和 node2vec 基于谱矩阵分解，运行时间最慢。结果表明对于小型图 PPI 和 Wiki，ProNE 需要不到 10s 的时间完成，比最快的 baseline 也要快 14 倍。对于大型图，ProNE 对比使用 20 个线程的 LINE 也快大约 10 倍。

随机游走方向

现代计算机具有为数据密集型应用而设计的复杂的内存层次结构。多级 CPU 缓存、高 DRAM 带宽以及硬件预取和内存行缓冲区等特性，都可以透明地帮助程序从时间和空间的局域性中获益，前者多级 CPU 缓存和高 DRAM 带宽允许数据重用，后者简化了大型顺序访问。然而在大型工作集中，大量执行随机访问的应用程序从最先进的 CPU 中获得的收益缺非常少。

一个重要图工作负载 Graph Random Walk 已经被诸多公司如阿里巴巴、谷歌、Facebook 等大量使用，给定一个输入图，随机遍历应用程序根据特定的转移概率规范从当前顶点取样一条边，从而发出一定数量 walker，每个 walker 在顶点之间游走，随机游走的一个应用即图嵌入，可以用于不同的应用像节点分类、链接预测、分类等，随机游走也被用于图分析、聚合、排序和数据集成等。图上的随机游走的过程可以看成是一个有限的马尔可夫链，无向图上的随机游走可以看成是时间可逆的马尔可夫链。

目前许多的图随机游走都通常使用随机梯度下降的方法，因此一般图嵌入框架都采用在 CPU 上执行图随机游走，在 GPU 上执行嵌入训练。GPU 相比较于 CPU 的计算能力增长速度快得多，给主机端随机游走带来持续压力，从而跟上 GPU 嵌入计算的速度。而且机器学习应用中往往采用 GPU 节点，其中少数 GPU 可以使用不同的超参数同时训练独立的图嵌入。在大且不规则的图数据集中，概率操作的固有随机性使得大多数内存访问都是随机的，现有的系统依次独立地处理每个 walker，在游走中对一条边进行采样，并在 DRAM 中移动到其他地方。

如今在大型工作集上大量执行随机访问的应用程序从最先进的 CPU 硬件中获得的好处非常少。图的随机游走被假定为几乎没有局部性，对大型、不规则图形数据集的概率操作固有的随机性使得大多数内存访问都是随机的。

表 2 访问延迟在不同的缓存和 DRAM 上

Location	L1 cache	L2 cache	L3 cache	Local mem	Remote mem
Sequential read	0.42ns	0.41ns	0.44ns	0.76ns	1.51ns
Random Read	0.77ns	0.95ns	2.60ns	18.35ns	24.35ns
Pointer-chasing	1.69ns	5.26ns	19.26ns	116.90ns	194.26ns

Access latency at different cache/DRAM layers

表 2 中展示了在不同缓存和 DRAM 上的访问延迟，可以看到对于随机读写 DRAM 访问延迟明显增高，而对于 Pointer-chasing(链表遍历)来说，由于下一个元素不在缓存中，导致遍历过程会不断引起内存操作，访问延迟更加明显。由图可以得出的结论：(1)尽管内存硬件具有随机访问的性质，但顺序访问和随机访问之间存在很大的延迟差距；(2) 顺序流甚至可以从 NUMA 节点的远程内存中获得可承受的延迟，而顺序随机性能差距随着层次结构的深入而快速增长；(3)指针跟踪的成本很高，其成本使得三级缓存内的访问速度比对 DRAM 的简单随机访问慢。

由于图数据具有倾斜性和幂律性，少数高度数顶点会吸引大量 walker，FlashMob 提出一种方案：对于高度数顶点采用预采样处理(PS)，低度数顶点采用直接采样处理(DS)。

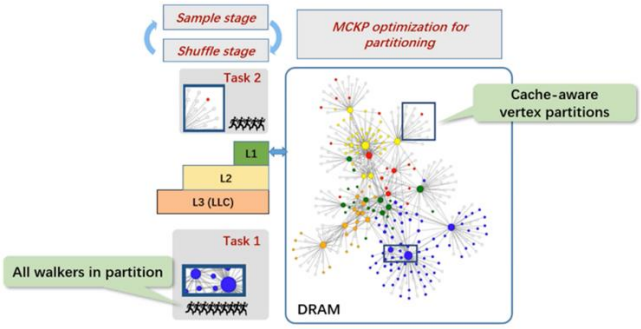

图 8 FlashMob 整体框架示意图

FlashMob 先对图按照度数排序，按照度数顶点分成不同的任务。将随机游走的执行变为每轮迭代只走一步，游走过程分为采样和洗牌两个阶段，前者负责采样，后者将 walker 发送到对应分区。分区太小，缓存友好但是洗牌开销变大；分区太大则相反。

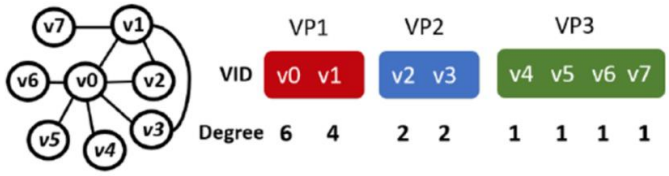

图 9 采样顶点分区举例

图 9 举例 FlashMob 依据顶点的度数进行排序后进行分区，然后根据分区内的情况选择不同的采样策略。

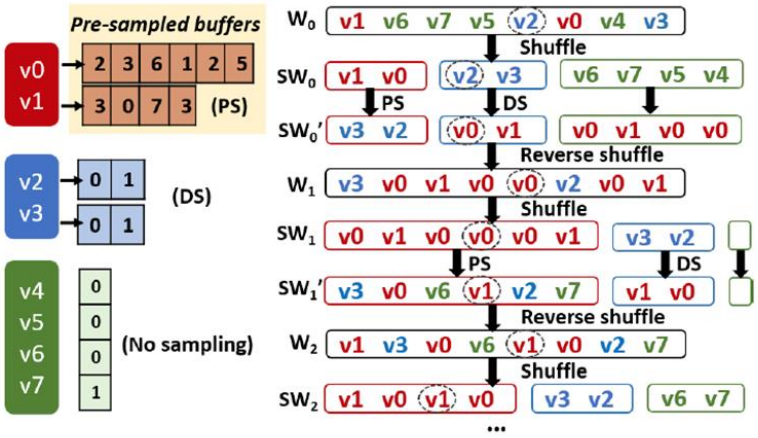

图 10 FlashMob walker 游走方法

图 10 中 W0 为保存 walker 当前驻留的顶点，洗牌作用是将 walker 发送到对应分区，相当于一次 map，然后针对 VP1 采用预采样，VP2 直接采样，VP3 由于无边或度数为 1，不采样。PS 通过连续掷骰子的方式，用边缘转移概率重新填充器预采样的边缘缓冲区，DS 当场掷骰子，需要将 VP 中所有边放入缓存中

W1 和 W2 分别在执行第一步和第二步后，以原始 walker ID（WID）顺序存储所有 walker 的位置 W1 和 W2 分别在第一步和第二步后，按照原始的 walker ID 顺序进行存储对应的位置。

对于缓存友好的边缘采样，FlashMob 需要将顶点切割成连续的 VP，主要的折衷在于 VP 大小和技术之间，较小的 VP 适合更快的缓存，更多的 VP 经过不适合的缓存或增加洗牌级别来增加洗牌开销。将排好序的顶点分成 G 组，组内的顶点再次切割为 2 的幂次方大小的 VP，顶点分区大小可能因组而异。

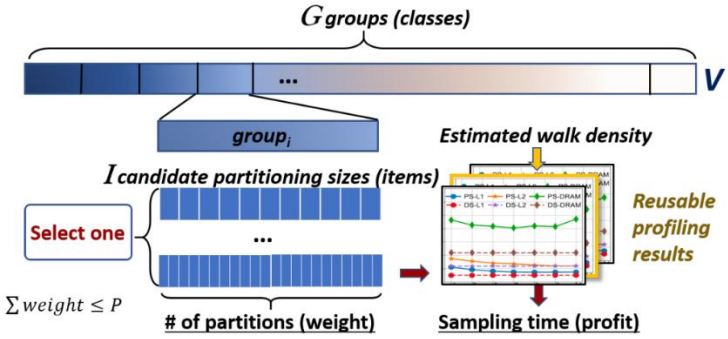

图 11 FlashMob 的 MCKP 问题放大一组示例

图 7 显示了 MCKP 问题公式，G 组映射到 G classes。每个类别中候选 items 是 VP 大小和每个 VP 的抽烟策略。

将顶点按度数排序后分成相同大小组，每个组按照 L1, 2, 3 和 DRAM 大小分区，针对组内分区采用 PS 或 DS 方法，采样方法根据预实验得到采样时间最短的采样方法确定，每个组权重为组内分区数，收益针对组内所有分区采样时间的负值(时间越短，收益越高)，利润定义为总采样成本的负值：组内所有 VP 采样时间之和。优化目标即收益最大，即背包问题 MCKP。

洗牌阶段 P 参数作为超参数，给出总重量限制，FlashMob 将每个外部级别的洗牌任务放入二级缓存(二级缓存大小在容量和速度之间提供了一个很好的平衡，决定了单个洗牌级别容纳的 VP 数量，即到 SW 的并发顺序写入流的数量。如果 VP 数量超过限制，那么需要添加一级洗牌，增加总体开销)。

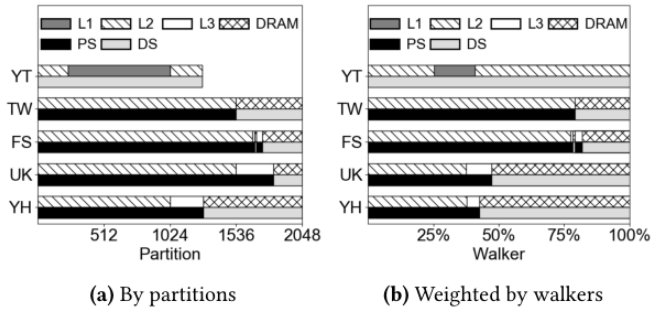

图 12 FlashMob 自动配置 VP 大小解决方案

图 12 顶部栏给出了排序顶点数组 VP 大小的决策，底部栏给出了采样策略的决策。条形图左侧高阶顶点被切割成更小的(主要为 L2)VP，并分配给使用 PS；最低顶点通常使用

DP，拟合到 R3 没有帮助。MCKP 对 L2 的偏好高于 L3。由于每个 L3 和 DRAM 大小的 VP 都包含更多的顶点，但是通过 walker 进行加权，L2 大小的 VP 在份额中后退，说明 DP 算法偏向最高阶顶点。

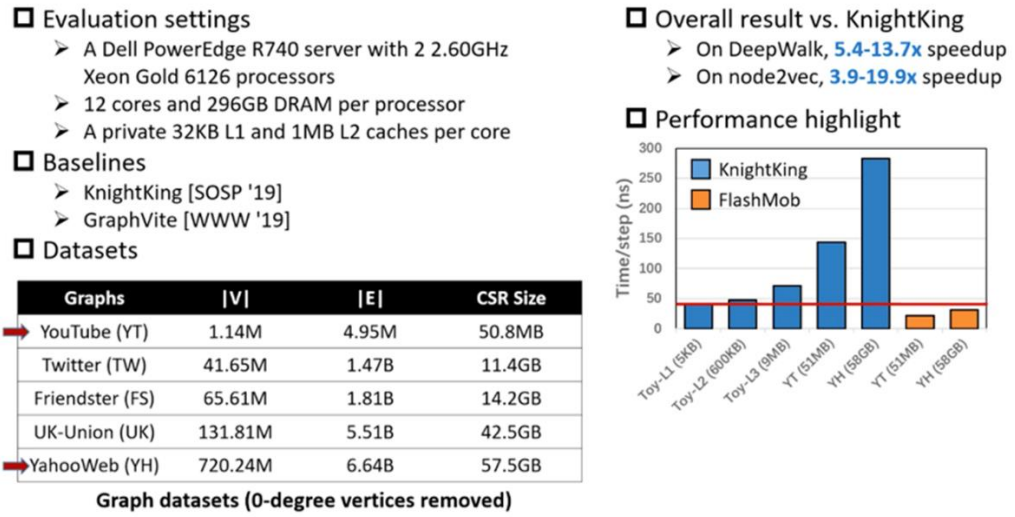

图 13 评估效果

实验测试结果与 KnightKing 和 GraphVite 对比后有明显的性能提升，图 13 显示了实验测试结果的具体数据和测试所用配置。

CPU-GPU 混合系统

网络在现实世界中是无处不在的，人们对图表示的学习越来越感兴趣，其目的是在低维空间中保留网络的结构。在网络分析中，拥有有效的节点表示是至关重要的，因为这些表示在很大程度上决定了许多下游任务的性能。这类方法已经在各种应用中被证明是成功的，如节点分类、链接预测和网络可视化。在这一领域已经有许多工作被提出，如嵌入方法 DeepWalk,LINE 和 node2vec。通过预测邻居来学习有效的节点嵌入可以通过异步随机梯度下降法（ASGD）进行有效优化。

目标条件是在具有多核 CPU 的单一机器上，它们能够处理具有一个或几百万个节点的网络，但现实世界的网络很容易达到数千万个节点和近数十亿条边的规模，如何使节点嵌入方法适应如此大规模的网络？一般将现有的方法扩展到分布式设置，但是大型 CPU 集群的成本对许多用户来说仍然是难以承受的。可以利用高度并行的硬件来加速节点嵌入的训练，然而直接采用 GPU 进行节点嵌入可能是低效的。因为节点嵌入中的抽样程序需要对网络结构进行过多的随机内存访问。CPU 在执行随机内存访问方面的能力要强得多。

我们需要设计一个系统，利用 CPU 和 GPU 的独特优势，协同使用它们来有效地训练节点嵌入。面临的主要挑战有：1）有限的 GPU 内存，节点嵌入的参数矩阵相当大，而单个 GPU 的内存却非常小；2）有限的总线带宽，总线的带宽比 GPU 的计算速度慢得多。这样如果 GPU 经常与主存储器交换数据，就会出现严重的延迟；3）巨大的同步成本，大量的数据在 CPU 和 GPU 之间传输。CPU-GPU 或 GPU 之间的同步都是非常昂贵的。

明智的做法是同时使用 CPU 和 GPU 来训练节点嵌入，提出 GraphVite 是一个针对节点嵌入的高性能的 CPU-GPU 混合系统因此。GraphVite 通过共同优化节点嵌入算法和系统，充分利用了 CPU 和 GPU 的优势。

节点嵌入方法通常由两个阶段组成：1）针对 CPU 进行网络增强，在原始网络上进行随机游走，在增强后的网络上对正边进行采样，会导致过度的随机访问；2）针对 GPU 进行嵌入训练，根据样本进行训练，主要由矩阵计算组成。

算法 1 节点嵌入通用框架

```
E'← E
for v∈ V do // parallelizable
    for u∈ Walk(v) do
        E'← E'∪ {(v, u, Weight(v, u))}
    end for
end for

for each iteration do // parallelizable
    v, u← EdgeSampling(E')
    Train(vertex[v], context[u], label = 1)
    for u'∈ NegativeSampling(V ) do
        Train(vertex[v], context[u'], label = 0)
    end for
end for
```

算法 1 总结了现有节点嵌入的总体框架，第一阶段可以很容易的并行化，第二阶段可以通过异步随机梯度下降法并行化，在大多数现有的节点嵌入系统中，两个阶段是按顺序执行的，每个阶段由一组 CPU 线程并行。

GraphVite 提出了一种并行在线扩充，用于在 CPU 上实现高效网络扩充的方案，同时引入一个并行负采样来配合多个 GPU 进行嵌入训练。

算法 2 并行在线扩充

```
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
```

由于扩充网络通常比原始网络大一到两个数量级，如果原始网络非常大，则不可能将其加载到主内存中。因此引入了一种并行在线扩充网络，可以在不进行显式网络增强的情况下动态生成增强的边缘样本。算法 2 给出了详细步骤：首先决定一个出发节点，其概率与每个节点的度数成正比；然后从出发节点开始进行随机游走，在一个特定的增强距离 s 内挑选节点作为边缘样本；接着收集边缘样本到一个样本池中，并对样本池进行洗牌，再将样本池转移到 GPU 进行嵌入训练。

洗牌属于伪洗牌，因为洗牌样本池对优化很重要，会减慢网络扩充阶段。原因是一般随机洗牌由大量随机内存访问组成，不能通过 CPU 缓存加速。如果服务器有多个 CPU 插槽，则损失速度会更加严重。伪洗牌能以方便缓存的方式洗牌，从而显著提高系统速度。多数相关性来自在同一随机游走中共享源节点或目标节点的边采样。由于这种相关性发生在一组增加距离 s 的 s 个样本中，我们将样本池划分为 s 个连续块，并将相关样本分散到

不同的块中。对于每个块，我们总是在最后按顺序追加样本，这可以从 CPU 缓存中受益匪浅。连接 s 块以形成最终样本池

在嵌入训练阶段，将训练任务分割成小块，并将其分配给多个 GPU。子任务的设计必须使用很少的共享数据，从而最小化 GPU 之间的同步成本。

$$\begin{cases} \theta_1 \leftarrow \theta_0 - \alpha \nabla L(X_1; \theta_0) \\ \theta_2 \leftarrow \theta_1 - \alpha \nabla L(X_2; \theta_1) \end{cases} \quad (1)$$

$$\begin{cases} \theta_1' \leftarrow \theta_0 - \alpha \nabla L(X_2; \theta_0) \\ \theta_2' \leftarrow \theta_1' - \alpha \nabla L(X_1; \theta_1') \end{cases} \quad (2)$$

i.e. $\|\theta_2 - \theta_2'\| \leq \epsilon$ *is true for the above equations.*

如果两个边样本集不共享任何源节点或目标节点，则两个边样本集计算梯度可交换。即使边样本集共享一些节点，如果学习率和迭代次数有界，仍然可以 ϵ-梯度交换。

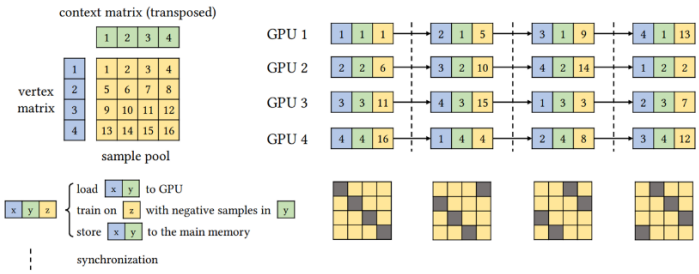

图 14 并行负采样

基于上述梯度可交换性，提出一种用于嵌入训练阶段的并行负采样算法，如图 14 对于 n 个 GPU，将顶点行和上下文分别划分为 n 个分区，将为样本池生成一个 n×n 的的分区网格，其中每个边都属于其中的一个块。任何一对不共享行或列的块都可以进行梯度交换。只要限制每个块上的迭代次数，同一行或同一列中的块是可 ϵ-梯度交换的。图 14 给出的是一个带有 4 个分区的并行负采样示例，每个 GPU 使用异步随机梯度下降法更新自己的嵌入分区，由于梯度可交换，因此多个 GPU 可以并发执行异步随机梯度下降法，同时我们限制只能从当前 GPU 的上下文行中提取负样本，因为如果 GPU 必须相互通信获取负样本嵌入，则可能非常耗时。

算法 3 并行负采样

```
function ParallelNegativeSampling(num_GPU )
    vertex_partitions← Partition(vertex)
    context_partitions← Partition(context)
    while not converge do
        pool← ParallelOnlineAugmentation(num_CPU )
        block[·][·]← Redistribute(pool )
        for offset← 0 to num_GPU- 1 do
            for i← 0 to num_GPU- 1 do ▷ paralleled
                vid← i
                cid← (i + offset) mod num_GPU
                send vertex_partitions[vid] to GPU i
                send context_partitions[cid] to GPU i
                train block[vid][cid] on GPU i
                receive vertex_partitions[vid] from GPU i
                receive context_partitions[cid] from GPU i
            end for
        end for
    end while
end function
```

并行负采样使不同的 GPU 能够同时训练节点嵌入，只需要在片段之间进行同步。但是样本池也在 CPU 和 GPU 之间共享，如果他们在样本池上进行同步，那么只有同一阶段的工作者可以同时访问样本池，这意味着硬件有一半的时间是空闲的。

为了解决这个问题，提出一种协作策略来降低同步成本。我们在主内存中分配两个样本池，让 CPU 和 GPU 始终在不同的内存池中工作。

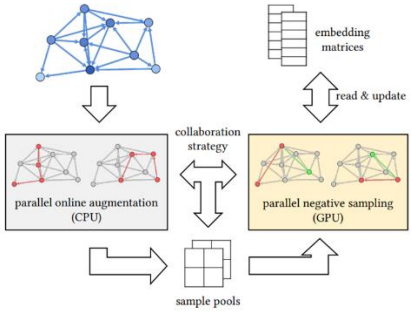

图 15 协作策略分配两个样本池

实验结果如表 3 显示 GraphVite 即使在当个 GPU 上，速度仍然使 LINE 的 19 倍。

表 3 Youtube 上不同系统的时间结果

Method	CPU threads	GPU	Training time	Preprocessing time
LINE [32]	20	-	1.24 hrs	17.4 mins
DeepWalk [23]	20	-	1.56 hrs	14.2 mins
node2vec [8]	20	-	47.7 mins	25.9 hrs
LINE in OpenNE [2]	1	1	> 1 day	2.14 mins
GraphVite	6	1	**3.98 mins**(18.7×)	**7.37 s**
GraphVite	24	4	**1.46 mins**(50.9×)	**16.0 s**

总结及展望

在本文中，我们先给出了图嵌入的定义以及相关的基本概念。然后我们介绍了四种类型的嵌入输入和四种类型的嵌入输出，并总结了每种设置中的优势和应用场景，紧接着我们讲述了图嵌入方向中所用到的相关技术。再从三个方向，分别是网络嵌入方向、随机游走方向以及 CPU-GPU 混合系统方向，展示了目前图嵌入相关的最新研究进展。

未来随着图相关应用的不断发展和推进，图相关技术不可或缺，通过图嵌入技术为核心来构造特征向量，可以方便我们更好地理解图背后数据的含义。图嵌入领域的四个未来方向，包括 1）计算，因为图没有网格结构来让输入数据位于一维或二维网格中，导致采用集合输入的深层体系结构存在效率低的问题，需要为图嵌入设计深层架构来寻找替代方案，从而提高模型效率；2）问题设置，动态图是一种很有前途的图嵌入设置，生活场景中的图不总是静态的；3）技术，结构感知是基于图嵌入边重构的重要内容，现有的基于边重构的图嵌入方法主要是只基于边，存在则效率低下的问题，需要设计相应的结构感知图嵌入优化解决方案；4）应用场景，从不同的角度可以为传统问题提供有效的解决方案，但是应用场景仍有着广阔的前景。