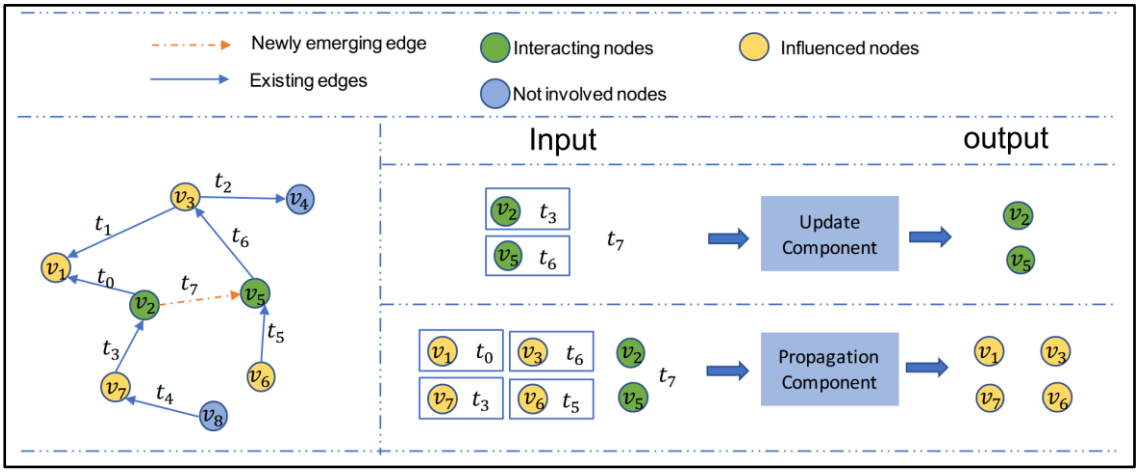
《Streaming Graph Neural Networks》

《流图神经网络》

摘要：

1. **本文的背景：**图是许多真实世界数据（如社交网络）的基本表示形式，而在此之前大多数图神经网络模型都是针对静态图设计的，但现实世界中的许多图本身却是动态的。这就使得当前的图神经网络模型无法有效地利用动态图中的动态信息，然而动态信息已经被证明是可以提高许多图分析任务的性能的，如社区检测和链接预测。因此，迫切需要为动态图设计一种专用的图神经网络模型。
2. **本文的贡献：**提出了一种新型动态图神经网络模型——$DGNN$，它可以随着图的演化对动态信息进行建模。特别是该框架可以通过捕获边的序列信息、边之间的时间间隔和信息传播的连贯性来不断更新节点的信息。
3. **主要创新点：**设计了两个组件：**更新组件**和**传播组件**。在引入新边时，更新组件可以通过捕获边的创建顺序信息和交互之间的时间间隔来保持节点信息的更新；而传播组件能通过考虑影响强度，将新的交互信息传播到受影响的节点的邻居上。
4. **实验结果：**在三个真实的动态图数据集上取得了 SOTA 结果。对链路预测和节点分类任务的实验结果表明了动态信息的重要性，以及所提出的更新和传播组件在捕获动态信息方面的有效性。

问题定义：



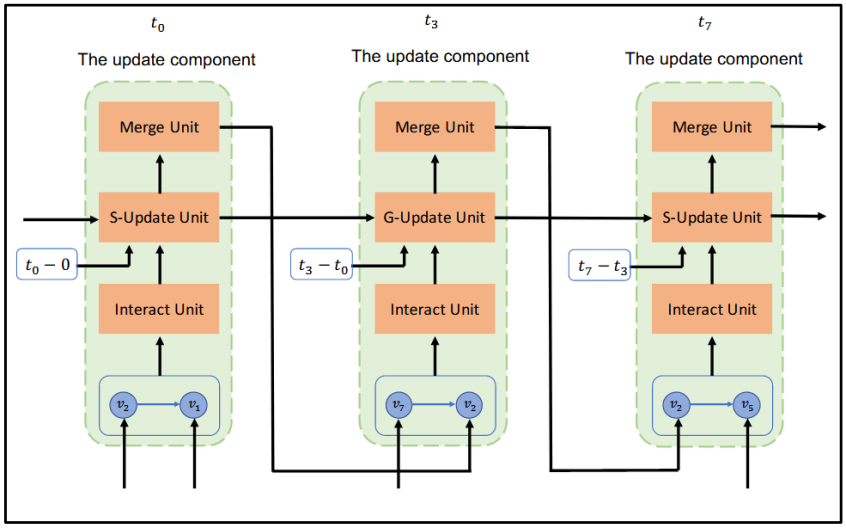
动态图由一组节点组成，假设有N个节点$V = \{v_1, v_2, \ldots, v_N\}$。当出现新的边和节点时，动态图就会发生变化。如上图左侧所示，其中有 8 个节点，而从时间t_0到t_7一共出现了 8 条交互边。【**注意：**在本项工作中，我们只考虑新的边和节点的出现，不考虑现有边和节点删除的情况。】

其中，有向边e可以进一步表示为(v_s, v_g, t)，描述在t时刻从节点v_s到节点v_g的交互。而为了方便起见，我们把参与交互的两个节点称为**“交互节点”**，如上图绿色节点所示。此外，新的交互不仅会影响两个交互节点，还会影响与交互节点“相邻”的其他节点，我们称之为**“受影响节点”**（一跳邻居），如上图黄色节点所示。

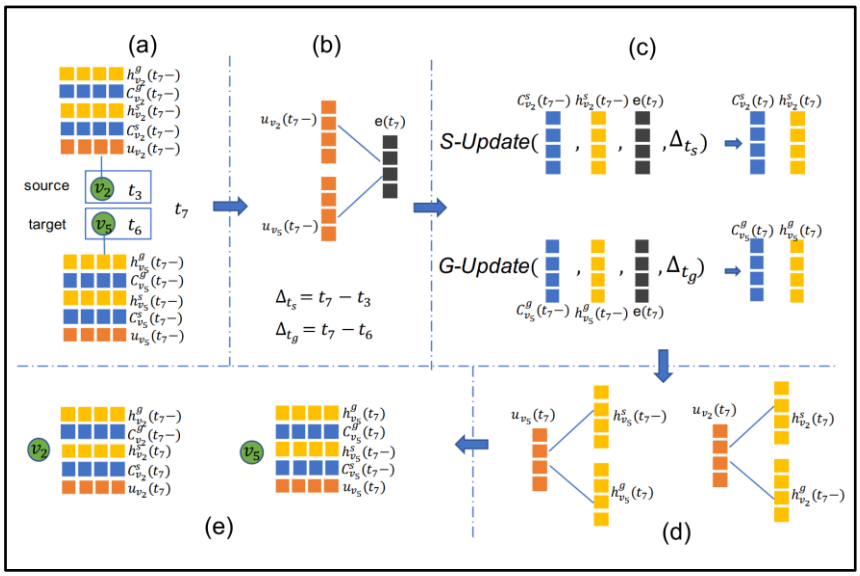
因此，我们不仅需要在两个交互节点上更新新的交互信息，还需要将此信息传播给“受影响节点”。而这就分别涉及了所提出模型的**“更新组件”**和**“传播组件”**两个模块，如上图右侧所示。

更新组件（The update component）：

在更新组件中，有三种单元：交互单元（Interact Unit），更新单元（Update Unit），合并单元（Merge Unit），如下图所示：【更新组件的操作，以v_2和它的所有交互为例】



而具体的更新步骤过程，以下图v_2和v_5的交互为例：



（1）如上图(a)所示，每个节点v都储存着五种信息，而所有信息的符号解释如下：

C表示为单元记忆存储（cell memory）；h表示为隐藏层状态（hidden state）；u表示为该节点的一般特征（general feature）；上标g表示为该节点作为目标节点（target role of node）；上标s表示为该节点作为源节点（source role of node）；下标v_i表示节点的编号；(t_i-)表示接近t_i时刻，但是没有到达；(t_i)表示在t_i时刻。

图中举例的部分说明如下：

举例 1：$h_{v_2}^{g}(t_7)$表示为在t_7时刻，节点v_2作为目标节点的隐藏层状态

举例 2：$C_{v_5}^{s}(t_7-)$表示为在接近t_7时刻时，节点v_5作为源节点的单元记忆储存

举例 3：$u_{v_2}(t_7-)$表示为在接近t_7时刻时，节点v_2的一般特征表示

（2）如上图(b)所示，是交互单元（Interact Unit）的操作：

将$u_{v_2}(t_7-)$和$u_{v_5}(t_7-)$作为输入，而输出的$e(t_7)$中包含交互v_2、v_5和t_7的信息，具体计算公式如下：

$$e(t) = act(W_1 \cdot u_{v_s}(t-) + W_2 \cdot u_{v_g}(t-) + b_e)$$

其中，W_1, W_2, b_e是模型的可训练参数，$act(\cdot)$是激活函数，例如$sigmoid$或者$tanh$。并在此计算两个时间差：

$$\Delta_{t_s} = t_7 - t_3$$

$$\Delta_{t_g} = t_7 - t_6$$

这两个时间差将会在下文的更新单元中用作为输入。

（3）如上图(c)所示，是更新单元（Update Unit）的操作：

更新单元分为两种$S-Update$和$G-Update$，是为了处理当节点作为不同的角色（源节点或目标节点）时的区别。$S-Update$将此次交互(v_2, v_5, t_7)的源节点v_2在(t_7-)时刻相关的状态$h_{v_2}^s(t_7-), C_{v_2}^s(t_7-)$以及交互信息的$e(t_7)$，还有该节点时间间隔$\Delta_{t_s}$作为输入，输出为在$(t_7)$时刻的更新状态$h_{v_2}^s(t_7), C_{v_2}^s(t_7)$。同理，$G-Update$更新目标节点$v_5$的相关状态$h_{v_5}^g(t_7), C_{v_5}^g(t_7)$。

而$S-Update$和$G-Update$的具体结构是一个$LSTM$的变种，如下图所示：

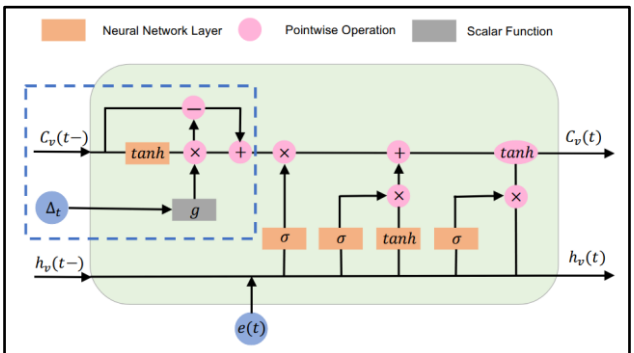


Figure 4: An illustration of the update unit

其中，蓝色框中的具体公式操作如下：

$$C_v^I(t-1) = tanh(W_d \cdot C_v(t-1) + b_d) \quad (2)$$

$$\hat{C}_v^I(t-1) = C_v^I(t-1) * g(\Delta_t) \quad (3)$$

$$C_v^T(t-1) = C_v(t-1) - C_v^I(t-1) \quad (4)$$

$$C_v^*(t-1) = C_v^T(t-1) + \hat{C}_v^I(t-1) \quad (5)$$

$C_v^I(t-1)$表示短期记忆部分，$C_v^T(t-1)$表示长期记忆部分，$g(\Delta_t)$为遗忘函数，它是一个递减函数，意味着时间间隔越大，保持的短期记忆越少。而上述所有步骤的目的是为得到一个加入了时间间隔信息的综合$C_v^*(t-1)$。

接着，是剩下的$LSTM$单元的具体公式表达，它和常见的$LSTM$单元类似：

$$f_t = \sigma(W_f \cdot e(t) + U_f \cdot h_v(t-1) + b_f) \quad (6)$$

$$i_t = \sigma(W_i \cdot e(t) + U_i \cdot h_v(t-1) + b_i) \quad (7)$$

$$o_t = \sigma(W_o \cdot e(t) + U_o \cdot h_v(t-1) + b_o) \quad (8)$$

$$\tilde{C}_v(t) = tanh(W_c \cdot e(t) + U_c \cdot h_v(t-1) + b_c) \quad (9)$$

$$C_v(t) = f_t * C_v^*(t-1) + i_t * \tilde{C}_v(t) \quad (10)$$

$$h_v(t) = o_t * tanh(C_v(t)). \quad (11)$$

因此，整个 Update Unit 模块可以缩写为：

$$C_v(t), h_v(t) = \text{Update}(C_v(t-1), h_v(t-1), \Delta_t, e(t)) \qquad (12)$$

注意：$S-Update$只处理这次交互的源节点，$G-Update$只处理这次交互的目标节点。

（4）如上图(d)所示，是合并单元（Merge Unit）的操作：

将交互(v_2, v_5, t_7)经过更新后的信息$h^s_{v_2}(t_7), h^g_{v_5}(t_7)$和先前该节点未更新且对应的信息$h^g_{v_2}(t_7-), h^s_{v_5}(t_7-)$合并为一般特征$u_{v_2}(t_7), u_{v_5}(t_7)$，具体操作公式如下：

$$u_{v_s}(t) = W^s \cdot h^s_{v_s}(t) + W^g \cdot h^g_{v_s}(t-) + b_u \qquad (13)$$

$$u_{v_g}(t) = W^s \cdot h^s_{v_g}(t-) + W^g \cdot h^g_{v_g}(t) + b_u \qquad (14)$$

其中，W^s, W^g, b_u是模型的可训练参数。

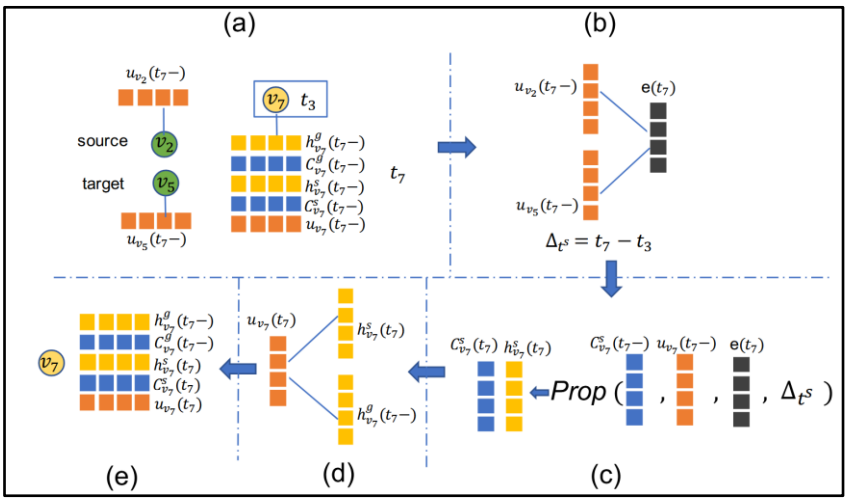
传播组件（The propagation component）：

传播组件基于以下假设：

1）由于交互不会直接影响受影响的节点，所以**假设交互不会干扰受影响节点的历史，只会带来新的信息。**因此，在模型中我们不需要像在更新组件中那样衰减或减少历史信息（单元内存），而只需要增量地向其添加新信息。

2）与较早的交互应该对最近的节点信息影响较小的直觉类似，**假设交互应该对较早的受影响节点影响较小。**因此，还需要考虑传播分量中相互作用的时间间隔。

传播组件的具体传播流程如下：



如上图所示，可见传播组件中也包含三个单元：交互单元（Interact Unit），更新单元（Update Unit），合并单元（Merge Unit），但和之前的更新组件相比，只有更新单元存在区别。所以，这里仅主要介绍传播组件的更新单元，如上图(c)中所示。

符号说明：

$N(v_s)$表示源节点v_s所影响的邻居，例如节点v_2影响的邻居为v_2的一跳邻居v_1, v_7（除开交互包含点v_5）；$N(v_g)$表示目标节点v_g所影响的邻居，例如节点v_5影响的邻居为v_5的一跳邻居v_3, v_6（除开交互包含点v_2）；$\Delta_t^s = t - t_x$指节点v_x与源节点v_s交互时，当前时间t和最后交互时间t_x之间的间隔；$g(\Delta_t^s)$和上面更新组件中的相似，是一个衰减函数；$h(\Delta_t^s)$是为了限制交互时间间隔非常长的邻居（extremely old neighbors），从直觉上来看，如果将交互信息传播给这些邻居，将会引入噪声。因此，引入一个函数$h(\Delta_t^s)$是为了过滤一些“受影响节点”，其具体的定义如下：

$$h(\Delta_t^s) = \begin{cases} 1, & \Delta_t^s \le \tau, \\ 0, & \text{otherwise.} \end{cases}$$

其中，τ是一个预定义的阈值。

传播组件的更新单元的具体公式如下：

$$C_{v_x}^s(t) = C_{v_x}^s(t-) + f_a(u_{v_x}(t-), u_{v_s}(t-)) \cdot g(\Delta_t^s) \cdot h(\Delta_t^s) \cdot \hat{W}_s^s \cdot e(t) \quad (15)$$

$$h_{v_x}^s(t) = \tanh(C_{v_x}^s(t)) \quad (16)$$

其中，$\hat{W}_s^s$表示一个变换矩阵（神经网络训练参数）；$f_a(u_{v_x}(t-), u_{v_s}(t-))$是一个注意力函数，用来捕获两节点之间的连接强度关系，具体函数如下（$softmax$）：

$$f_a(u_{v_x}(t-), u_{v_s}(t-)) = \frac{\exp(u_{v_x}(t-)^T u_{v_s}(t-))}{\sum_{v \in N^s(v_s)} \exp(u_v(t-)^T u_{v_s}(t-))} \quad (17)$$

参数学习：

在$DGNN$中每个节点有一组一般特征，但是在链路预测中每个节点有两个不

同的角色，于是引入两个投影矩阵P^s, P^g，这两个矩阵负责将一般特征投影为对应角色的特征。对于一组交互(v_s, v_g, t)，将最近的一般特征$u_{v_s}(t-), u_{v_g}(t-)$投影为$u^s_{v_s}(t-), u^g_{v_g}(t-)$：

$$u^s_{v_s}(t-) = P^s \cdot u_{v_s}(t-)$$
$$u^g_{v_g}(t-) = P^g \cdot u_{v_g}(t-)$$

从v_s到v_g的概率被建模为$\sigma(u^s_{v_s}(t-)^T u^g_{v_g}(t-))$，其中$\sigma(\cdot)$是$sigmoid$函数。

此外，模型链接预测任务的**损失函数定义**如下：

$$J((v_s, v_g, t)) = -log(\sigma(u^s_{v_s}(t-)^T u^g_{v_g}(t-))) - Q \cdot \mathbb{E}_{v_n \sim P_n(v)} log(\sigma(u^s_{v_s}(t-)^T u^g_{v_n}(t-)))$$

即常见的负采样训练，其中Q是负样本数，$P_n(v)$是负采样分布。因此，直到时间T的总损失可以表示为：

$$\sum_{e \in \mathcal{E}(T)} J(e);$$

最后，模型节点分类任务的**损失函数定义**如下：

$$J(v, t) = -\sum_{i=0}^{N_c - 1} y[i] \log \left(\frac{\exp(u^c_v(t))[i]}{\sum_{j=0}^{N_c - 1} \exp(u^c_v(t))[j]} \right);$$

其中，$y \in 0,1^{N_c}$表示为类型y取值为 0,1；上标N_c表示为分类种类的数量；$u^c_v(t)$是把原来的一般特征$u_v(t)$进行一个投影，映射得到N_c维度上，而每一个维度就是一种特征。此外，在训练中可以并不是所有节点都有标签，所以是一个半监督的任务。但在进行传播和更新的时候，所有节点都要参与。

实验及结果：

1）数据集

UCI：有向图，表示加利福尼亚大学欧文分校学生在线社区用户之间的消息通信。

DNC：有向图，这是 2016 年民主党全国委员会电子邮件泄漏事件中电子邮件通信的情况。

$Epinions$：有向图，表示产品审查平台 Epinions 中用户之间的信任关系。

Table 1: Statistics of datasets

	UCI	DNC	Epinions
number of nodes	1,899	2,029	6,224
number of edges	59,835	39,264	19496
time duration	194 days	982 days	936 days
number of labels	\	\	15

2）实验设置

在链接预测任务中，使用一小部分交互作为历史，并预测未来会出现哪些新的边缘；训练集/验证集/测试集的比例：8/1/1；对于在测试集里的每个交互边(v_s, v_g, t)，固定节点v_s，使用图中所有节点去替换v_g，然后用余弦相似度进行相似度排序。再反过来，固定节点v_g，同所有节点替换v_s，并计算相似度排名。

在节点分类任务中，随机抽取一部分节点并隐藏它们的标签。这些隐藏标签的节点将被视为验证和测试集。其余节点被视为训练集；训练集/验证集/测试集的比例：8/1/1。

3）评价指标

$$\textbf{MRR} = \frac{1}{|H|}\sum_{i=1}^{H}\frac{1}{rank_i}$$

$$\textbf{Recall@k} = \frac{1}{|H|}\sum_{i}^{|H|}\mathbf{1}\{rank_i \le k\}$$

4）链接预测实验结果

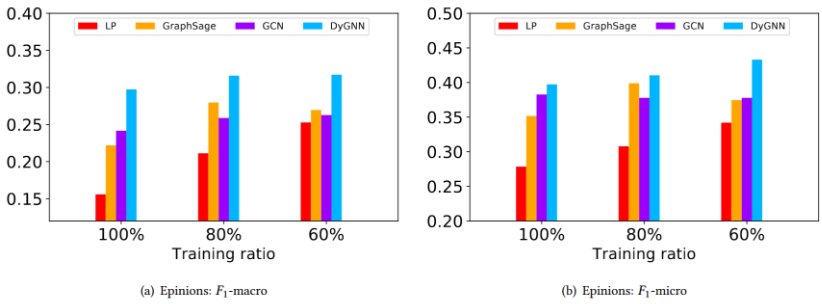
Table 2: Performance comparison of link prediction.

Baselines	UCI			DNC			Epinions		
	MRR	Recall@20	Recall@50	MRR	Recall@20	Recall@50	MRR	Recall@20	Recall@50
DGNN	**0.0342**	**0.1284**	**0.2547**	**0.0536**	0.1852	**0.3884**	**0.0204**	**0.0848**	**0.1894**
GCN	0.0138	0.0632	0.1176	0.0447	**0.2032**	0.3291	0.0045	0.0071	0.0119
GraphSage	0.0060	0.0161	0.0578	0.0167	0.0576	0.1781	0.0035	0.0072	0.0108
node2vec	0.0056	0.0184	0.0309	0.0202	0.0719	0.178	0.0135	0.0571	0.1240
DynGEM	0.0146	0.0773	0.1455	0.0271	0.0971	0.2356	0.0150	0.0657	0.1233
CPTM	0.0138	0.0921	0.1082	0.0109	0.0072	0.0108	0.0036	0.0060	0.0125
DANE	0.0040	0.0110	0.0233	0.0128	0.0270	0.0432	0.0040	0.0100	0.0120
DynamicTriad	0.0150	0.0610	0.1236	0.0146	0.0414	0.0665	0.0170	0.0729	0.1629

可以看到，所提出的动态图神经网络模型$DGNN$的表现是优于现有的两个有代表性的GNN，即GCN和$GraphSage$的。而由于$DGNN$是针对动态网络的，而GCN和$GraphSage$忽略了动态信息，这进一步支持了捕捉动态信息的重要性。此外，

在DNC网络上的表现不如预期，而这可能是因为它最初是为属性网络（attributed networks）设计的。

5）节点分类实验结果



(a) Epinions: F_1-macro (b) Epinions: F_1-micro

Figure 6: Performance Comparison of Node classification on Epinions dataset

可以看到，随着标记节点数量的增加，分类性能趋于提高。并且，$GraphSage$、GCN和$DGNN$在所有设置下都优于LP，这表明GNN在半监督学习中的有效性。此外，$DGNN$在所有三种设置下都优于$GraphSage$和GCN，这表明了时间信息在节点分类中的重要性。

6）模型组件分析：

Table 3: Comparison of variants on the link prediction task.

Baselines	UCI			DNC			Epinions		
	MRR	Recall@20	Recall@50	MRR	Recall@20	Recall@50	MRR	Recall@20	Recall@50
DGNN	0.0342	0.1284	0.2547	0.0536	0.185	0.3884	0.0204	0.0848	0.1894
DGNN-prop	0.0103	0.0444	0.1087	0.0046	0	0	0.0171	0.0633	0.1514
DGNN-ti	0.0174	0.0918	0.2118	0.0050	0	0.0054	0.0157	0.0591	0.1589
DGNN-att	0.0200	0.0844	0.2235	0.0562	0.1547	0.3219	0.0177	0.0651	0.1655

可以看到，更新组件和传播组件都是有效果的，并且我们还能得出以下结论：1）有必要将交互信息传播到受影响的节点；2）重要的是要考虑时间间隔信息；3）捕捉各种影响可以提高性能。

7）参数敏感性分析：

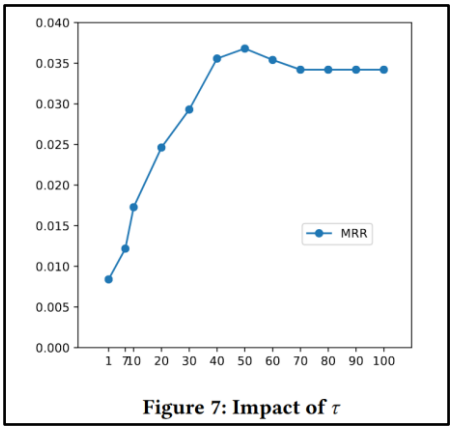


Figure 7: Impact of τ

可以看到，传播过程确实有助于向“受影响节点”传递必要的信息，因为当阈值增加时，性能首先得到改善。但是，将交互信息传播给“非常老的邻居”可能没有帮助，甚至可能带来噪音。综上，只需要使用少量“受影响节点”来执行传播，这样不仅能提高效率，而且模型性能也更好。

总结：

本文提出了一种用于动态图的新型图神经网络模型$DGNN$。它提供了两个关键组件——更新组件和传播组件。在引入新的 Edge 时，更新组件可以通过捕获 Edge 的创建顺序信息和交互之间的时间间隔来保持节点信息的更新。而传播组件将通过考虑影响强度将新的交互信息传播到受影响的节点。而在实验中，作者以链路预测和节点分类为例，说明了如何利用$DGNN$推进图挖掘任务，并在 3 个真实世界的动态图上进行了实验。而从链路预测和节点分类等方面的实验结果表明了动态信息的重要性以及所提出的更新和传播组件在捕获动态信息方面的有效性。

对本文的感悟：

$DGNN$更偏向于消息传递机制，其中涉及节点嵌入表示的训练又使用到了RNN中的$LSTM$结构，这是一种很好地融合，但缺点是会导致模型串行化，使参数优化变得非常困难。总而言之，这篇论文的写作逻辑非常值得我们学习。