流图中基于草图的异常检测



Siddharth Bhatia

TurboML siddharth@turboml.com

尼尔·沙阿

Mohit Wadhwa

mailmohitwadhwa@gmail.com

俞思宇

伊利诺伊大学芝加哥分校psyu@uic.ed

uKenji川口

新加坡国立大学kenji@comp.nus.edu.sg

Bryan Hooi

新加坡国立大学bhooi@comp.nus.edu.sg

ABSTRACT

给定来自动态图的图边流，我们如何以在线方式为边和子图分配异常分数，以检测异常行为，使用恒定的时间和内存?例如，在入侵检测中，现有的工作试图检测异常边或异常子图，但不能同时检测两者。在本文中，我们首先将count-min草图数据结构扩展到高阶草图。这个高阶草图具有保留密集子图结构的有用性质(输入中的密集子图变成数据结构中的密集子矩阵)。然后，我们提出了4种利用这种增强数据结构的在线算法，它们(a)检测边缘和图的异常;(b)以恒定的内存和每个新到达的边缘的恒定更新时间处理每个边缘和图，并且;(c)在4个真实数据集上优于最先进的基线。我们的方法是第一个采用密集子图搜索来检测恒定内存和时间内的图异常的流方法。

CCS CONCEPTS

•计算方法→异常检测。

KEYWORDS

异常检测，动态图形，流，草图

ACM Reference Format:

Siddharth Bhatia, Mohit Wadhwa, Kenji Kawaguchi, Neil Shah, Philip S. Yu和Bryan Hooi. 2023。基于草图的流图异常检测。第29届ACM SIGKDD知识发现与数据挖掘会议论文集(KDD ' 23)， 2023年8月6-10日，USA，加州长滩。ACM，USA纽约，12页。https://doi.org/10.1145/3580305。3599504

1 INTRODUCTION

考虑一个入侵检测系统，在这个系统中，异常行为可以被描述为一个人或一组攻击者与一组目标机器建立大量连接，以限制可访问性或寻找潜在漏洞。我们可以将其建模为动态图，其中节点对应于机器，每个边表示从一台机器到另一台机器的带有时间戳的连接。

IMAGE

This work is licensed under a Creative Commons Attribution International 4.0 License.

KDD ’23, August 6–10, 2023, Long Beach, CA, USA ©2023 Copyright held by the owner/author(s). ACM ISBN 979-8-4007-0103-0/23/08. https://doi.org/10.1145/3580305.3599504

在这个图中，边缘异常包括个别连接，这些连接明显比网络中其他连接更具恶意。此外，异常行为通常还采取密集子图的形式，该子图可以表示一组恶意节点，这些节点以一种与图的其余部分相比不寻常的方式相互通信，如[1-3]中的几个真实数据集所示。同时检测这些边缘和子图异常提供了对网络结构和行为的有价值的见解，并且可以帮助识别仅考虑一种异常类型可能不明显的趋势或模式。

类似地，在金融系统中，边缘异常可能包括比系统中其他交易明显更大或更频繁的交易。子图异常可能包括比系统其余部分更有可能参与欺诈活动的个人或企业群体。同时识别这两种类型的异常可以帮助检测欺诈活动并防止经济损失。

因此，我们提出了一个问题:给定来自动态图的图边流，我们如何以在线方式为边和子图分配异常分数，以检测异常行为，使用恒定的内存和每个新到达的边的恒定更新时间?

有几种方法[4-10]旨在检测图集中的异常。然而，这些方法关注的是静态图，而许多现实世界的图本质上是时间进化的。在流或在线图场景中，一些方法可以检测异常边的存在，[3,11 - 13]，而另一些方法可以检测异常子图[1,2,14]。然而，现有的所有方法都局限于异常边缘或图检测，而不能同时检测这两种异常，如表1所示。正如我们在第7节中讨论的那样，我们的方法在准确率和运行时间上都优于现有方法;并且在异常边缘和子图检测场景上都是如此。此外，我们的方法是唯一一种利用密集子图搜索来检测图异常的流方法，同时只需要恒定的内存和时间。

我们首先将二维草图扩展到高阶草图，使其能够在图中嵌入源节点和目标节点之间的关系。高阶草图具有保留密集子图结构的有用性质;输入中的密集子图在这种数据结构中变成密集子矩阵。因此，从大图中检测密集子图的问题简化为在常数大小的矩阵中寻找密集子矩阵，这可以在常数时间内实现。高阶草图允许我们提出几种算法来同时检测两者

93

Siddharth Bhatia et al.

KDD ’23, August 6–10, 2023, Long Beach, CA, USA

以流的方式检测异常边和子图。我们引入了两种边缘异常检测方法AnoEdge-G和AnoEdge-L，以及两种图异常检测方法AnoGraph和angraph - k，它们使用相同的数据结构来检测密集子矩阵的存在，从而分别检测异常边或子图。我们所有的方法都在恒定的时间内处理边和图，并且与图的大小无关，也就是说，它们需要恒定的内存。我们还提供了关于高阶草图估计和子矩阵密度度量的理论保证。综上所述，我们论文的主要贡献有:

(1)高阶草图(第4节):我们通过将count-min草图(CMS)数据结构扩展到高阶草图，将密集子图检测问题转化为寻找密集子矩阵(可以在常数时间内实现)。

(2)流式异常检测(第5,6节):我们提出了四种新的在线方法来实时检测异常边缘和图形，具有恒定的内存和更新时间。此外，这是第一个结合密集子图搜索来检测恒定内存/时间内的图异常的流工作。

(3)有效性(第7节):我们在四个真实数据集上优于所有最先进的流边缘和图异常检测方法。

可重复性:我们的代码和数据集可在https://github.com/Stream-AD/AnoGraph上获得。

⌚RELATEDwork

我们的工作与图上的异常检测[15-25]和流[26-35]以及流算法[36 - 40]等领域密切相关。[40]中讨论了高阶草图，然而，它们仅限于计数草图和非图设置。

[41-43]讨论了基于深度学习的异常检测，[44 - 48]讨论了密集子图发现，然而，这些方法无法以流方式检测异常。[4-10]仅限于静态图中的异常检测。然而，在本节中，我们将审查仅限于检测动态图中的边缘和图异常的方法;请参阅[49]进行广泛的调查。

边缘流方法:热点[50]检测节点的自我网突然变化。RHSS[51]专注于稀疏连接的图部分。CAD[52]使用comm -mute时间距离测量来定位异常变化。最近，DenseStream[1]维护和更新张量流中的密集子张量。SedanSpot[11]基于边缘发生、优先依恋和相互邻居来识别边缘异常。PENminer[12]探索活动片段的持久性，即边缘更新序列重复出现的长度和规律性。F-FADE[13]旨在通过分解其频率来检测异常交互模式。MIDAS[3,53]识别基于微簇的异常。然而，所有这些方法都无法检测到图异常。

图流方法:DTA/STA[54]使用矩阵分解近似当前快照的邻接矩阵。Copycatch[55]定位近二部核，其中每个节点在短时间内密集地连接到同一核中的其他节点。SPOT/DSPOT[33]使用极值理论自动设置异常阈值。IncGM+[56]采用增

量方法处理图更新。最近，DenseAlert识别短时间内创建的子张量，并利用增量方法更有效地处理图更新或子图。SpotLight[2]发现具有密集双团的异常图，但使用随机方法而不搜索密集子图。受PageRank[57]的启发，AnomRank[14]迭代更新两个分数向量，并计算异常分数。然而，这些方法速度很慢，并且不能检测边缘异常。此外，它们不会在恒定的内存/时间内搜索密集子图。

3 PROBLEM

设e ={𝑒´2，···}是来自时间演化图g的加权边流，每个到达的边是一个元组，(𝑢)，𝑖,𝑡𝑖)由一个源节点，一个目标节点，一个权值和一个发生时间组成，这个时间是边被添加到图中的时间。例如，在网络流量流中，边𝑒𝑖could表示从源IP地址𝑢𝑖to到目的IP地址𝑣𝑖at时间𝑡𝑖的连接。我们不假设顶点集合V是先验已知的:例如，新的IP地址或用户id可能在流的过程中被创建。

我们将G建模为一个有向图。无向图可以通过将传入的无向边处理为两个同时存在的有向边，每个方向各一条来处理。我们还允许G是一个多图:在同一对节点之间可以多次创建边。允许边同时到达:即𝑡𝑖+1≥𝑡𝑖，因为在许多应用中𝑡𝑖is作为离散时间刻度给出。

我们的算法的期望属性如下:

•检测异常边:在线检测边缘是否为异常子图的一部分。能够在更细粒度的边缘上检测异常，可以早期发现，以便尽快开始恢复，并将恶意活动的影响降至最低。

•检测异常图:以在线方式检测异常子图(由一段时间内接收的边组成)的存在，因为这样的子图通常对应于意外行为，例如协调攻击。

•恒定内存和更新时间:为了确保可扩展性，内存使用和更新时间不应随着节点数量或流的长度而增长。因此，对于新到达的边缘，我们的算法应该在恒定的内存和更新时间下运行。

4 高阶草图和符号

计数最小草图(Count-min sketch, CMS)[58]是一种流行的流数据结构，被许多在线算法所使用。CMS使用多个哈希函数将事件映射到频率，但与哈希表不同的是，它只使用次线性空间，代价是由于碰撞而对某些事件进行过多计数。频率被近似为所有哈希函数中的最小值。如图1(a)所示，CMS表示为二维矩阵，其中每行对应一个散列函数，散列对应相同数量的桶(列)。

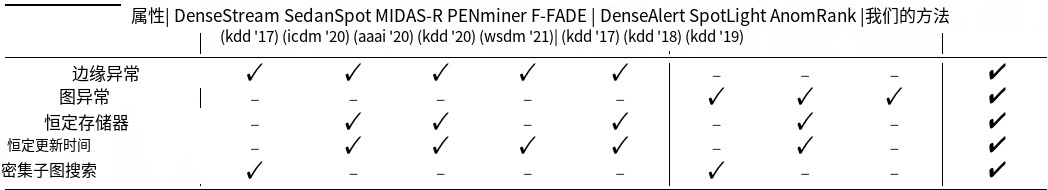
我们引入了一个高阶CMS (H-CMS)数据结构，其中每个哈希函数将多维输入映射到一个通用的十元，而不是映射到一个行向量。H-CMS通过单独散列实体的各个组件来增强CMS

94

KDD ’23, August 6–10, 2023, Long Beach, CA, USA

Sketch-Based Anomaly Detection in Streaming Graphs

表1:相关异常检测方法的比较。



维护更多的信息。图1(b)显示了一个三维H-CMS，可用于将二维实体(如图边)散列到矩阵中。源节点被散列到草图矩阵的第一个维度，目标节点被散列到草图矩阵的另一个维度，而原始CMS将整个边缘散列到一维行向量(图1(a))。

我们使用三维H-CMS(操作在算法1中描述)，其中哈希函数的数量用𝑛𝑟表示，矩阵M𝑗corresponding到𝑗-th哈希函数ℎ𝑗is的维数𝑛𝑏×𝑛𝑏，即一个方阵。对于每个𝑗∈[𝑛𝑟]]，用ℎ𝑗(𝑢，𝑣表示的𝑗-th哈希函数将一个边(𝑢，𝑣)映射到一个矩阵索引(′𝑗(𝑢),ℎ𝑗′′(𝑣)))，即源节点映射到一个行索引，目的节点映射到一个列索引。即，<s:1>𝑗(𝑢，𝑣)= (<s:1> '𝑗(𝑢)，<s:1> '𝑗'(𝑣))。因此，三维H-CMS中的每个矩阵都捕获了图邻接性的本质

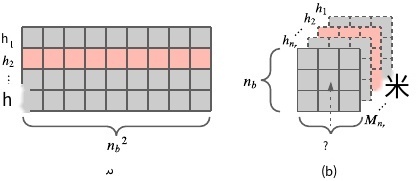


图1:(a)原始CMS，每个哈希函数对应𝑛𝑏2桶;(b)高阶CMS，每个哈希函数对应𝑛𝑏x𝑛𝑏buckets桶。

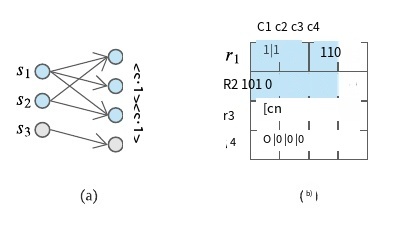


图2:(a)原图中的稠密子图-源节点𝑠1，𝑠2和目标节点𝑑1，𝑑2，𝑑3之间转换为H-CMS中𝑟1，𝑟2行和𝑐1，𝑐2，𝑐3列之间的稠密子矩阵(b)。

矩阵。因此，密集子图检测可以转化为密集子矩阵检测问题(如图2所示)，其中矩阵的大小是一个小常数，与边的数量或图的大小无关。

对于任意(𝑢，𝑣)，设(𝑢，𝑣)为迄今为止观测到的(𝑢，𝑣)的真实计数，而(𝑢，𝑣)= min𝑗∈[𝑛𝑟]M𝑗[𝑗'𝑢(𝑗)][𝑣' '()][' '()]为通过三维H-CMS对计数的估计。由于H-CMS可能会因可能的碰撞而高估计数(但不会低估，因为我们更新并保留了每个哈希函数的所有计数)，因此我们有(𝑢，𝑣)≤(𝑢，𝑣)。我们定义𝑀to为到目前为止所有观测值的个数;即𝑀=Í𝑢，𝑣(𝑢，𝑣)。以下定理表明三维H-CMS具有与CMS相似的估计保证:

定理1。(附录A中的证明)对于所有𝑘∈[𝑛𝑟]]，letℎ𝑘(𝑢，𝑣)= (<s:1>𝑘'(𝑢)，<s:2>𝑘"(𝑣))，其中每个哈希函数都是从一个成对独立的族中均匀随机选择的。在这里，我们允许?两种情况下都存在，分别是:<s:2> ' = <e:1> '和'≠<e:1> '。固定𝛿> 0，设??𝑟=?ln𝛿1和??𝑏=≤≤𝑒。然后，以至少1−??的概率，综上所示，粽(𝑢，𝑣)≤粽(𝑢，??)+粽(𝑀)。

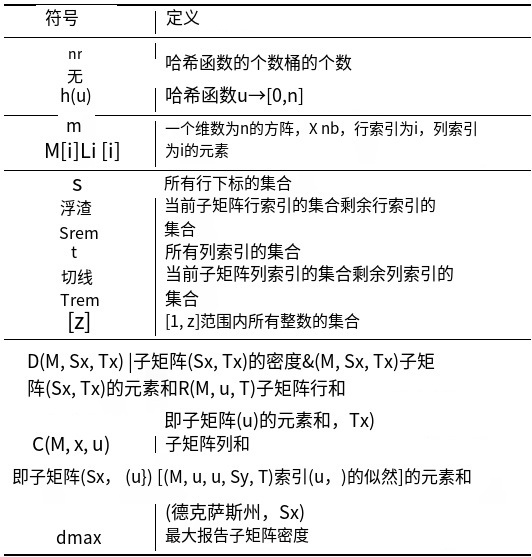
定理1表明，即使我们对源节点和目标节点(即ℎ′=ℎ′′))使用相同的哈希函数，我们也有估计保证。因此，在滥用符号的情况下，我们通过在右侧设置<s:1> = <s:1> ' = <e:1> '，写出了当<s:1> ' = <e:1> "时的<s:1>(𝑢，𝑣)= (<s:3>(𝑢)，<s:3>(𝑣))。另一方面，在ℎ′≠ℎ′′,的情况下

95

Siddharth Bhatia et al.

KDD ’23, August 6–10, 2023, Long Beach, CA, USA

表2:符号表。



在Theo-rem 1中有可能改进估计保证。例如，如果我们可以从一个弱全称的哈希函数集合(通过为ℎ′andℎ′′undersome定义相应的分布族)中均匀随机地选择

√︃

条件)，则根据定理1的证明，可以令𝑛𝑏=≤𝑒𝜖⌉具有与定理1相同的估计保证。对这种潜在改进的分析作为一个开放问题留给未来的工作。

表2中讨论了常用的符号，我们利用[60]中讨论的子图密度度量来定义子矩阵(𝑆𝑥,𝑇𝑥)密度)。

Definition 1. Given matrix M, density of a submatrix of M represented by 𝑆𝑥⊆𝑆and 𝑇𝑥⊆𝑇, is:

FORMULA

5 EDGE ANOMALIES

在本节中，使用H-CMS数据结构，我们提出了AnoEdge-G和AnoEdge-L，通过检查接收到的边缘在映射到草图矩阵元素时是否属于密集子矩阵的一部分来检测边缘异常。AnoEdge-G找到一个全局密集子矩阵，AnoEdge-L在矩阵元素周围维护和更新一个局部密集子矩阵。

5.1 AnoEdge-G

如算法2中所述，AnoEdge-G维持一个暂时衰减的H-CMS，即每当一个单位时间过去时，我们将所有H-CMS计数乘以一个固定因子𝛼(lines2,4)。这种衰减模拟了逐渐“遗忘”旧的，因此，更过时的信息。当一条边(𝑢，𝑣)到达时，将𝑢，𝑣are分别映射到每个散列函数的矩阵索引(𝑢)，(𝑣)，并更新相应的H-CMS计数(第5行)。然后调用边缘-子矩阵-密度过程(如下所述)来计算围绕((𝑢)，(𝑣))的密集子矩阵的密度。密度被报告为边缘的异常分数;密度越大，说明边缘异常的可能性越大。

边-子矩阵-密度过程计算给定索引周围的密集子矩阵的密度((𝑢)，(𝑣))。用𝑆𝑐𝑢𝑟and𝑇𝑐𝑢𝑟表示的1×1子矩阵，用行索引<e:1>(𝑢)和列索引<e:1>(𝑣)(第9行)进行初始化。通过贪婪地选择一行𝑢𝑝from𝑆𝑟𝑒𝑚(or一列𝑣𝑝from𝑇𝑟𝑒𝑚)来迭代展开子矩阵，该列获得与当前子矩阵的最大行(或列)和(第11、12行)。这个选定的行𝑢𝑝(or列𝑣𝑝)从𝑆𝑟𝑒𝑚(or𝑇𝑟𝑒𝑚)中删除，并添加到𝑆𝑐𝑢𝑟(or𝑇𝑐𝑢𝑟)中(第14、16行)。重复此过程，直到𝑆𝑟𝑒𝑚and𝑇𝑟𝑒𝑚are为空(第10行)。在子矩阵展开过程的每次迭代中计算当前子矩阵的密度，并返回所有贪婪形成的子矩阵密度的最大值(第17、18行)。

96

KDD ’23, August 6–10, 2023, Long Beach, CA, USA

Sketch-Based Anomaly Detection in Streaming Graphs

提案一。(证明见附录B.1)演算法2的时间复杂度为𝑂(| |∗𝑛𝑟∗𝑛𝑏2)1。算法2的内存复杂度为𝑂(𝑛𝑟∗𝑛𝑏2)。

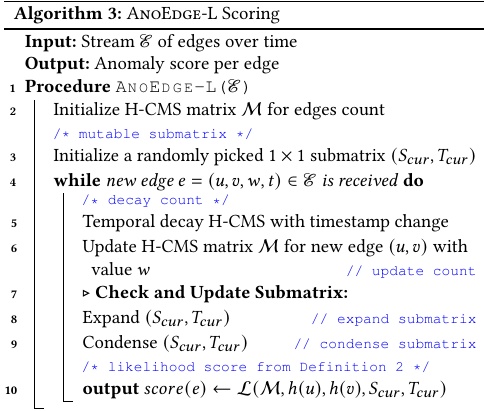
5.2 AnoEdge-L

受定义1的启发，我们定义了矩阵索引(𝑆(𝑢)，𝑆(𝑣))相对于子矩阵(𝑥,𝑇𝑥))的似然度量，作为子矩阵(𝑥,𝑇𝑥))中要么与索引ℎ′′(𝑣)共享行，要么与索引ℎ′(𝑢)共享列的元素之和除以此类元素的总数。

Definition 2. Given matrix M, likelihood of an index ℎ(𝑢, 𝑣) with respect to a submatrix represented by 𝑆𝑥⊆𝑆and 𝑇𝑥⊆𝑇, is:

FORMULA

如算法3中所述，AnoEdge-L维持一个随时间衰减的H-CMS来存储边缘计数。我们还用随机元素初始化一个大小为1 ×1的可变子矩阵，并将其表示为(𝑆𝑐𝑢𝑟,𝑇𝑐𝑢𝑟))。当我们处理边时，我们贪婪地更新(𝑆𝑐𝑢𝑟,𝑇𝑐𝑢𝑟))以维持它作为一个密集的子矩阵。当边缘到达时，首先更新H-CMS计数，然后使用接收到的边缘检查是否扩展当前子矩阵(第7行)。如果子矩阵密度随着行(或列)的增加而增加，则将行索引(𝑢)(或列索引(𝑣))添加到当前子矩阵(𝑆𝑐𝑢𝑟,𝑇𝑐𝑢𝑟))中。为了移除随时间衰减的行(s)和列(s)，该过程迭代地选择行和(或列和)最小的行(或列)，直到移除它增加当前的子矩阵密度。这确保了当前的子矩阵尽可能地浓缩(第9行)。正如定义2中所定义的，AnoEdge-L计算相对于(𝑆𝑐𝑢𝑟，𝑇𝑐𝑢𝑟))的边缘的似然评分(第10行)。更高的似然度量意味着该边更有可能是异常的。



1This is for processing all edges; the time per edge is constant.

命题2。(证明见附录B.2)演算法3的时间复杂度为𝑂(𝑛𝑟∗𝑛𝑏2 + | |∗𝑛𝑟∗𝑛𝑏)。算法3的内存复杂度为𝑂(𝑛𝑟∗𝑛𝑏2)。

⌚GRAPH异常

我们现在提出AnoGraph和AnoGraph- k，通过首先将图映射到高阶草图，然后检查密集子矩阵来检测图异常。这些是第一批利用密集子图搜索在恒定内存和时间内检测图异常的流算法。AnoGraph贪婪地寻找密度测度上具有2近似保证的稠密子矩阵。AnoGraph- k利用算法2中的Edge-Submatrix-Density在𝐾周围寻找一个密集的子矩阵，在实践中表现同样良好。

6.1 AnoGraph

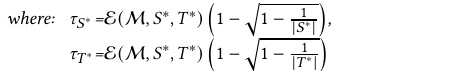
如算法4中所述，AnoGraph维护一个H-CMS来存储每当新图到达时重置的边缘计数。首先处理边缘以更新H-CMS计数。然后调用Ano-Graph-Density过程(如下所述)来查找密集的子矩阵。AnoGraph将异常评分报告为检测到的(密集)子矩阵的密度;更大的密度意味着图更有可能是异常的。

anography - density过程计算矩阵M的密集子矩阵的密度。当前的密集子矩阵被初始化为矩阵M，然后贪婪地从当前子矩阵中删除行(列)和最小的行(列)。重复这个过程，直到𝑆𝑐𝑢𝑟and𝑇𝑐𝑢𝑟are为空(第11行)。在子矩阵展开过程的每次迭代中计算当前子矩阵的密度，并返回所有密度的最大值(第18、19行)。

算法4是在有向图问题[60]中寻找最密集子图的特殊情况，其中有向图被表示为邻接矩阵，检测最密集子图本质上意味着检测密集的子矩阵。现在我们对密度测度提供了一个保证。

引理1。设𝑆∗和𝑇∗是密度D(M，𝑆∗，𝑇∗)=𝑑𝑜𝑝𝑡的M的最优密度子矩阵解。则∀𝑢∈𝑆∗及∀𝑣∈𝑇∗,

FORMULA



证明。利用[60]的证明，我们假设∃𝑢∈𝑆∗具有R(M，𝑢，𝑇∗)<𝑆∗。去除后的子矩阵密度

E(M，𝑆∗,𝑇∗)−R(M，𝑢，𝑇∗)E(M，𝑆∗,𝑇∗)−𝑆𝑆∗𝑢=√(|𝑇∗−1|)|𝑆∗|其大于√∗∗=(|𝑇−1|)| |

𝑑𝑜𝑝𝑡, and that is not possible. Hence, R(M,𝑢,𝑇 ∗) ≥ 𝜏𝑆∗. C(M,𝑆 ∗, 𝑣) ≥𝜏𝑇∗can be proved in a similar manner. □

定理2。算法4中的AnoGraph-Density过程实现了最密集子矩阵问题的2逼近保证。

97

Siddharth Bhatia et al.

KDD ’23, August 6–10, 2023, Long Beach, CA, USA

证明。利用[60]的证明，我们贪婪地删除行和(或列和)最小的行(或列)。在贪婪过程的某个迭代中，∀𝑢∈𝑆𝑐𝑢𝑟;∀𝑣∈𝑇𝑐𝑢𝑟，R(M，𝑢，𝑇𝑐𝑢𝑟)≥𝜏𝑆∗andC(M，𝑆𝑐𝑢𝑟，𝑣)≥𝜏𝑇∗。因此，E(M，𝑆𝑐𝑢𝑟,𝑇𝑐𝑢𝑟)≥|𝑆𝑐𝑢𝑟|𝜏𝑆∗andE(M，𝑆𝑐𝑢𝑟,𝑇𝑐𝑢𝑟)≥|𝑇𝑐𝑢|𝑟𝑇∗。这就意味着密度D(M，𝑆𝑐𝑢𝑟，𝑇𝑐𝑢𝑟)≥√︃|𝑆𝑐𝑢𝑟|𝑆∗|𝑇𝑐𝑢𝑟|𝑇∗√

= \* \*

\* |𝑆||𝑇|𝑆𝑇. .输入的值为:𝑆和𝑇∗𝑐𝑢𝑟𝑐𝑢𝑟

从引理1中，并设置|𝑆∗| = 1，|𝑇∗| = 1,sin2

E(M,S,T)我们得到D(M，𝑆𝑐𝑢𝑟，𝑇𝑐𝑢𝑟)≥√∗∗(1−cos时延)(1−cos时延)≥|𝑆||𝑇∗| sin时延sin时延

𝑑𝑜𝑝𝑡𝑑𝑜𝑝𝑡

rgr

时延≥2。

□

2 cos cos

2 2

提案3。(证明见附录C.1)算法4的时间复杂度为𝑂(|𝒢|∗𝑛𝑟∗𝑛𝑏2 +| <s:1> |∗𝑛𝑟)。算法4的内存复杂度为𝑂(𝑛𝑟∗𝑛𝑏2)。

6.2 AnoGraph-K

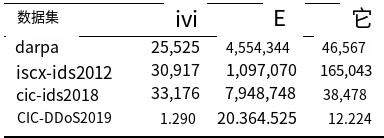
与AnoGraph类似，AnoGraph-K维护一个H-CMS，每当新图形到达时，它就会重置。它使用AnoGraph-K-Density过程(如下所述)来查找密集的子矩阵。AnoGraph-K在算法5中进行了总结。

AnoGraph-K-Density计算矩阵m的密集子矩阵的密度。直觉来自于启发式，即具有较高值的矩阵元素更有可能是密集子矩阵的一部分。因此，该方法考虑矩阵M的𝐾largest元素，并调用算法2中的Edge-Submatrix-Density来获得每个元素周围的密集子矩阵(第13行)。返回所考虑的𝐾dense子矩阵上的最大密度。

提案4。(证明见附录C.2) al - galgorithm 5的时间复杂度为𝑂(|𝒢|∗𝐾∗𝑛𝑟∗𝑛𝑏2 + | <s:1> |∗𝑛𝑟)。算法5的内存复杂度为𝑂(𝑛𝑟∗𝑛𝑏2)。

7 EXPERIMENTS

表3:数据集统计。



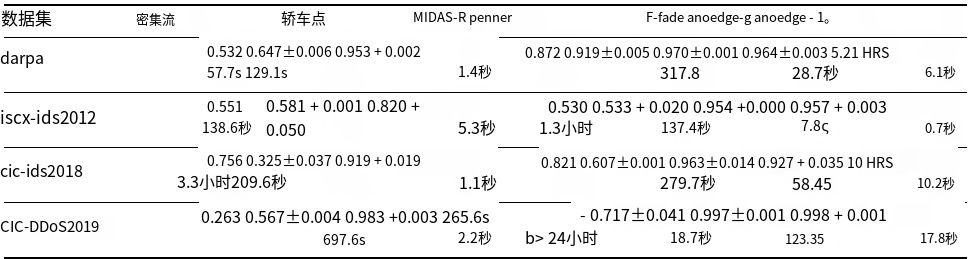
在本节中，与表1中讨论的所有基线相比，我们评估了我们的方法的性能，旨在回答以下问题:

98

KDD ’23, August 6–10, 2023, Long Beach, CA, USA

Sketch-Based Anomaly Detection in Streaming Graphs

表4:检测边缘异常时的AUC和运行时间。平均超过5次运行。



q1。边缘异常:与基线相比，AnoEdge-G和AnoEdge-L检测边缘异常的准确性如何?它们是否快速且可扩展?

q2 .;图异常:AnoGraph和angraph - k检测图异常(即异常图快照)的准确性如何?它们是否快速且可扩展?

表3显示了我们使用的四个真实世界数据集的统计摘要:DARPA[61]和ISCX-IDS2012[62]是基线用于评估其算法的图异常检测的流行数据集;[63]调查了30多个数据集，并建议使用较新的包含现代攻击场景的CIC-IDS2018和CIC-DDoS2019数据集[66,65]。| <s:1> |对应边缘记录总数，| |和|𝑇|分别为唯一节点数和唯一时间戳数。

与基线论文类似，我们报告了ROC曲线下的面积(AUC)和运行时间。AUC是通过绘制不同分类阈值下的真阳性率(TPR)和假阳性率(FPR)，然后计算得出的受试者工作特征(ROC)曲线下的面积来计算的。异常检测系统的适当分类阈值将取决于具体的应用程序以及假阳性和假阴性的成本，然而，由于AUC独立于分类阈值，因此可以评估系统的整体性能，而不必选择特定的阈值。除非明确指定，否则包括基线上的实验在内的所有实验都要重复5次，并报告平均值。

附录D描述了实验设置。基线的超参数在附录e中提供。所有基于边(或图)的方法输出每个边(或图)的异常分数，更高的分数意味着更多的异常。

7.1 Edge Anomalies

精度:边缘异常检测基线、AnoEdge-G、AnoEdge-L的AUC如表4所示。我们报告DenseStream和PENminer的单个值，因为它们是非随机方法。PENminer无法在24小时内完成大型CIC-DDoS2019。SedanSpot使用个性化的PageRank来检测异常，但并不总是能够检测到密集块模式中出现的异常边缘，而PENminer无法检测到结构异常。在基线中，MIDAS-R是

最准确的，然而，当有大量的时间戳时，如ISCX-IDS2012，它的性能较差。请注意，AnoEdge-G和AnoEdge-L在所有数据集上都优于所有基线。

运行时间:表4显示了AnoEdge-G和AnoEdge-L的运行时间(不包括I/O)和实时性能。由于AnoEdge-L维持一个局部密集子矩阵，因此比AnoEdge-G更快。DenseStream为每个到来的元组增量地维护密集块，并在满足更新条件时更新密集子张量，从而限制了检测速度。SedanSpot需要几个子进程(哈希、随机行走、重新排序、采样等)，PENminer和F-FADE需要主动提取每次图更新的模式，导致大量的计算时间。当ISCX-IDS2012中有大量的时间戳时，MIDAS-R的性能比最快的AnoEdge-L慢。

AUC与运行时间:图3绘制了ISCX-IDS2012数据集上的精度(AUC)与运行时间(对数尺度，以秒为单位，不包括I/O)。与所有基线相比，AnoEdge-G和AnoEdge-L的精度要高得多，同时运行速度也快得多。

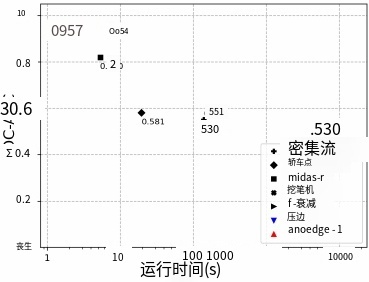


图3:ISCX-IDS2012上检测边缘anom-alies时的AUC与运行时间

可扩展性:图4(a)和图4(b)分别绘制了ISCX-IDS2012数据集上哈希函数和边数量增加时的运行时间。这证明了AnoEdge-G和AnoEdge-L的可扩展性。

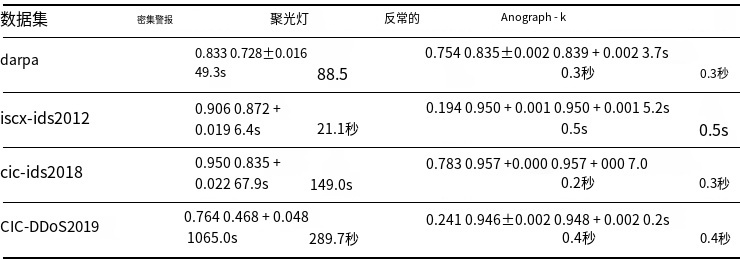
AnoEdge-G vs AnoEdge-L: AnoEdge-G找到一个全局密集子矩阵，因此比AnoEdge-L更准确

99

Siddharth Bhatia et al.

KDD ’23, August 6–10, 2023, Long Beach, CA, USA

表5:检测图异常时的AUC和运行时间。平均超过5次运行。



显示在CIC-IDS2018的表现中。另一方面，AnoEdge-L维护和更新矩阵元素周围的局部密集子矩阵，因此具有更好的时间复杂度和对更大数据集的可扩展性。

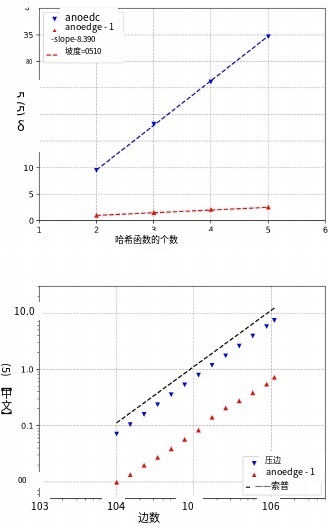


图4:(a)具有多个哈希函数的线性可扩展性。(b)具有边数的线性可扩展性。

7.2 图异常

精度:表5给出了图异常检测基线、AnoGraph和AnoGraph- k的AUC。我们报告DenseAlert和AnomRank的单个值，因为这些是非随机方法。AnomRank不适用于流场景，因此AUC较低。DenseAlert一次只能估计一个子张量，SpotLight使用随机方法，而不需要

对密集子图进行任何实际搜索。请注意，AnoGraph和Ano-Graph-K在所有数据集上都优于所有基线，同时使用简单的草图数据结构来合并密集的子图搜索，而不是基线。

运行时间:表5显示了运行时间(不包括I/O)。DenseAlert具有𝑂(|(|))更差的情况时间复杂度(每个传入边)。AnomRank需要计算一个全局PageRank，它不适合流处理。请注意，AnoGraph和AnoGraph- k的运行速度比所有基线都快得多。

AUC与运行时间:图5绘制了准确度(AUC)与运行时间(对数尺度，以秒为单位，不包括I/O)在CIC-DDoS2019数据集上的关系。与基线相比，AnoGraph和AnoGraph- k实现了更高的精度，同时运行速度也明显更快。

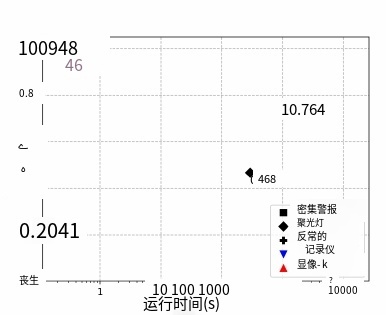


图5:在CIC-DDoS2019上检测图形异常时的AUC与运行时间

可扩展性:图6(a)、6(b)和6(c)分别绘制了CIC-DDoS2019数据集上的运行时间与增加因子𝐾(算法5中用于top-𝐾)、哈希函数数量和边数量的关系。这证明了AnoGraph和AnoGraph- k的可扩展性。

AnoGraph与AnoGraph- k: AnoGraph贪婪地寻找一个密集的子矩阵，其时间复杂度为𝑂(|𝒢|∗𝑛𝑟∗𝑛𝑏2 + | |∗𝑛𝑟))，而AnoGraph- k贪婪地在𝐾strategically选定的矩阵元素周围寻找一个密集的子矩阵，其时间复杂度为𝐾strategically

100

KDD ’23, August 6–10, 2023, Long Beach, CA, USA

Sketch-Based Anomaly Detection in Streaming Graphs

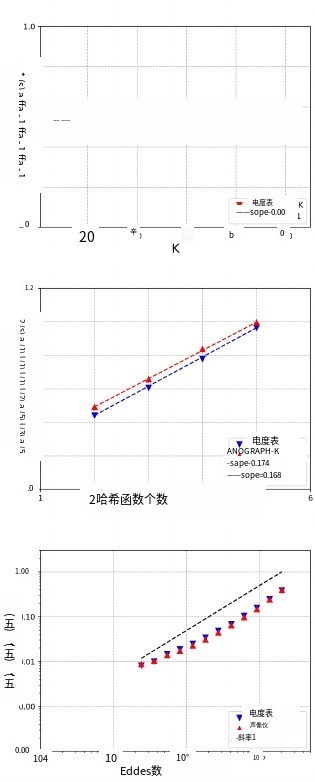


图6:(a) AnoGraph-K与因子𝐾呈线性关系。(b)哈希函数个数的线性可扩展性。(c)具有边数的线性可扩展性。

的𝑂(|𝒢|∗𝐾∗𝑛𝑟∗𝑛𝑏2 + | <s:1> |∗𝑛𝑟)。因此，当𝐾is明显小于𝑛时AnoGraph-K K的速度更快。AnoGraph-K也更健壮，因为它只考虑少量的矩阵元素。

7.3 超参数研究

表6显示了AnoGraph和AnoGraph- k在多个时间窗口和边缘阈值下的性能。边缘阈值的变化方式使得在时间窗内存在足够数量的异常。AnoGraph和AnoGraph- k的结果与表5中的结果相当。表7显示了AnoEdge-G和AnoEdge-L在改变时间衰减因子时的鲁棒性。

8 CONCLUSION

在本文中，我们将CMS数据结构扩展到一个高阶草图，以捕获图数据中的复杂关系，并减少

表6:检测图异常时，时间窗和边缘阈值对ROC-AUC的影响。

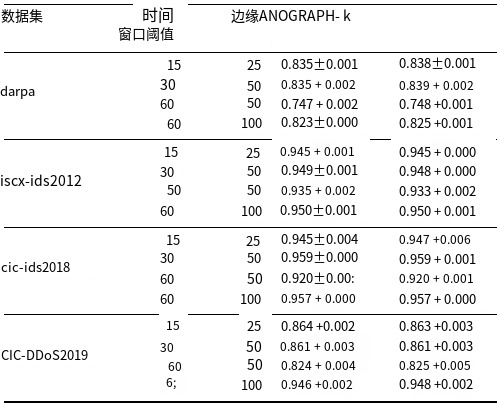
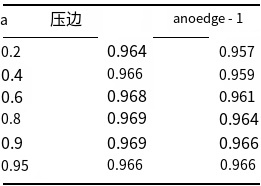


表7:时间衰减因子𝛼on对AnoEdge-G和AnoEdge-L ROC-AUC对DARPA的影响。



检测可疑密集子图以在恒定时间内找到密集子矩阵的问题。然后，我们提出了四种基于草图的流方法来检测恒定更新时间和内存中的边缘和子图异常。此外，我们的方法是第一个结合密集子图搜索来检测恒定内存和时间内的图异常的流工作。我们还对子矩阵密度度量提供了理论保证，并证明了所有方法的时间和空间复杂性。在四个真实数据集上的实验结果证明了我们的有效性，而不是流行的最先进的流边缘和图基线。未来的工作可以考虑结合矩形H-CMS矩阵、节点和边缘表示、更一般的数据类型(包括张量)和并行计算来处理具有大量传入边的大型动态图。

致谢

本研究由新加坡国家研究基金会、NCS私人有限公司、新加坡国立大学NUS-NCS联合实验室(Grant A-0008542-00-00)和美国国家科学基金会(NSF)资助(Grant SaTC-1930941)。

101

Siddharth Bhatia et al.

KDD ’23, August 6–10, 2023, Long Beach, CA, USA

参考文献

[1] K. Shin, B. Hooi, J. Kim, and C. Faloutsos, “Densealert: Incremental dense- subtensor detection in tensor streams,” KDD, 2017.

[2] D. Eswaran, C. Faloutsos, S. Guha, and N. Mishra, “Spotlight: Detecting anomalies in streaming graphs,” in KDD, 2018.

[3] S. Bhatia, B. Hooi, M. Yoon, K. Shin, and C. Faloutsos, “Midas: Microcluster-based detector of anomalies in edge streams,” in AAAI, 2020.

[4] L. Akoglu, M. McGlohon, and C. Faloutsos, “Oddball: Spotting anomalies in weighted graphs,” in PAKDD, 2010.

[5] D. Chakrabarti, “Autopart: Parameter-free graph partitioning and outlier detec-tion,” in PKDD, 2004.

[6] B. Hooi, K. Shin, H. A. Song, A. Beutel, N. Shah, and C. Faloutsos, “Graph-based fraud detection in the face of camouflage,” TKDD, 2017.

[7] M. Jiang, P. Cui, A. Beutel, C. Faloutsos, and S. Yang, “Catching synchronized behaviors in large networks: A graph mining approach,” TKDD, 2016.

[8] J. M. Kleinberg, “Authoritative sources in a hyperlinked environment,” JACM,1999.

[9] K. Shin, T. Eliassi-Rad, and C. Faloutsos, “Patterns and anomalies in k-cores of real-world graphs with applications,” KAIS, 2018.

[10] H. Tong and C.-Y. Lin, “Non-negative residual matrix factorization with applica- tion to graph anomaly detection,” in SDM, 2011.

[11] D. Eswaran and C. Faloutsos, “Sedanspot: Detecting anomalies in edge streams,” in ICDM, 2018.

[12] C. Belth, X. Zheng, and D. Koutra, “Mining persistent activity in continually evolving networks.,” in KDD, 2020.

[13] Y.-Y. Chang, P. Li, R. Sosic, M. Afifi, M. Schweighauser, and J. Leskovec, “F-fade: Frequency factorization for anomaly detection in edge streams,” in WSDM, 2021.

[14] M. Yoon, B. Hooi, K. Shin, and C. Faloutsos, “Fast and accurate anomaly detection in dynamic graphs with a two-pronged approach,” in KDD, 2019.

[15] J. Zhang, S. Liu, W. Yu, W. Feng, and X. Cheng, “Eigenpulse: Detecting surges in large streaming graphs with row augmentation,” in PAKDD, 2019.

[16] P. Bogdanov, C. Faloutsos, M. Mongiovì, E. E. Papalexakis, R. Ranca, and A. K. Singh, “Netspot: Spotting significant anomalous regions on dynamic networks,” in SDM, 2013.

[17] N. Shah, A. Beutel, B. Hooi, L. Akoglu, S. Gunnemann, D. Makhija, M. Kumar, and C. Faloutsos, “Edgecentric: Anomaly detection in edge-attributed networks,” in ICDMW, 2016.

[18] B. Perozzi and L. Akoglu, “Discovering communities and anomalies in attributed graphs: Interactive visual exploration and summarization,” TKDD, 2018.

[19] F. Bonchi, I. Bordino, F. Gullo, and G. Stilo, “The importance of unexpectedness: Discovering buzzing stories in anomalous temporal graphs,” Web Intelligence,2019.

[20] F. Bonchi, I. Bordino, F. Gullo, and G. Stilo, “Identifying buzzing stories via anomalous temporal subgraph discovery,” in WI, 2016.

[21] A. Bojchevski and S. Günnemann, “Bayesian robust attributed graph clustering: Joint learning of partial anomalies and group structure,” in AAAI, 2018.

[22] W. Yu, W. Cheng, C. Aggarwal, K. Zhang, H. Chen, and W. Wang, “Netwalk: A flexible deep embedding approach for anomaly detection in dynamic networks,” KDD, 2018.

[23] A. Kumagai, T. Iwata, and Y. Fujiwara, “Semi-supervised anomaly detection on attributed graphs,” IJCNN, 2021.

[24] Y. Liu, S. Pan, Y. G. Wang, F. Xiong, L. Wang, and V. C. Lee, “Anomaly detection in dynamic graphs via transformer,” ArXiv, vol. abs/2106.09876, 2021.

[25] M. Shao, J. Li, F. Chen, and X. Chen, “An efficient framework for detecting evolving anomalous subgraphs in dynamic networks,” INFOCOM, 2018.

[26] S. Bhatia, A. Jain, P. Li, R. Kumar, and B. Hooi, “Mstream: Fast anomaly detection in multi-aspect streams,” The Web Conference (WWW), 2021.

[27] S. Bhatia, A. Jain, S. Srivastava, K. Kawaguchi, and B. Hooi, “Memstream: Memory- based anomaly detection in multi-aspect streams with concept drift,” The Web Conference (WWW), 2022.

[28] E. A. Manzoor, H. Lamba, and L. Akoglu, “xstream: Outlier detection in feature- evolving data streams,” in KDD, 2018.

[29] S. C. Tan, K. M. Ting, and T. F. Liu, “Fast anomaly detection for streaming data,” in IJCAI, 2011.

[30] D. Jankov, S. Sikdar, R. Mukherjee, K. Teymourian, and C. Jermaine, “Real-time high performance anomaly detection over data streams: Grand challenge,” DEBS,2017.

[31] S. Zou, Y. Liang, H. V. Poor, and X. Shi, “Nonparametric detection of anomalous data streams,” IEEE Transactions on Signal Processing, 2017.

[32] M. Moshtaghi, J. C. Bezdek, C. Leckie, S. Karunasekera, and M. Palaniswami,“Evolving fuzzy rules for anomaly detection in data streams,” IEEE Transactions on Fuzzy Systems, 2015.

[33] A. Siffer, P.-A. Fouque, A. Termier, C. Largouet, and C. Largouët, “Anomaly detection in streams with extreme value theory,” KDD, 2017.

[34] M. U. Togbe, M. Barry, A. Boly, Y. Chabchoub, R. Chiky, J. Montiel, and V.-T. Tran, “Anomaly detection for data streams based on isolation forest using scikit-multiflow,” in ICCSA, 2020.

[35] J. Zhang, S. Liu, W. Hou, S. Bhatia, H.-W. Shen, W. Yu, and X. Cheng, “Augsplicing: Synchronized behavior detection in streaming tensors,” AAAI, 2021.

[36] S. Pan, X. Zhu, C. Zhang, and S. Y. Philip, “Graph stream classification using labeled and unlabeled graphs,” in ICDE, 2013.

[37] W. Wang, X. Guan, and X. Zhang, “Processing of massive audit data streams for real-time anomaly intrusion detection,” Computer communications, 2008.

[38] A. K. Menon, G. V. A. Pham, S. Chawla, and A. Viglas, “An incremental data- stream sketch using sparse random projections,” in SDM, 2007.

[39] P. Zhao, C. C. Aggarwal, and M. Wang, “gsketch: On query estimation in graphstreams,” VLDB, 2011.

[40] Y. Shi and A. Anandkumar, “Higher-order count sketch: Dimensionality reduction that retains efficient tensor operations,” DCC, 2020.

[41] R. Chalapathy and S. Chawla, “Deep learning for anomaly detection: A survey,” ArXiv, vol. abs/1901.03407, 2019.

[42] G. Pang, C. Shen, L. Cao, and A. van den Hengel, “Deep learning for anomaly detection: A review,” arXiv preprint arXiv:2007.02500, 2020.

[43] S. Bhatia, A. Jain, and B. Hooi, “Exgan: Adversarial generation of extreme samples,” AAAI Conference on Artificial Intelligence (AAAI), 2021.

[44] C. Ma, Y. Fang, R. Cheng, L. Lakshmanan, W. Zhang, and X. Lin, “Efficient algorithms for densest subgraph discovery on large directed graphs,” SIGMOD,2020.

[45] A. Epasto, S. Lattanzi, and M. Sozio, “Efficient densest subgraph computation in evolving graphs,” WWW, 2015.

[46] S. Sawlani and J. Wang, “Near-optimal fully dynamic densest subgraph,” STOC,2020.

[47] A. Mcgregor, D. Tench, S. Vorotnikova, and H. Vu, “Densest subgraph in dynamic graph streams,” MFCS, 2015.

[48] H. Esfandiari and M. Mitzenmacher, “Metric sublinear algorithms via linearsampling,” FOCS, 2018.

[49] L. Akoglu, H. Tong, and D. Koutra, “Graph based anomaly detection and descrip- tion: A survey,” Data mining and knowledge discovery, 2015.

[50] W. Yu, C. C. Aggarwal, S. Ma, and H. Wang, “On anomalous hotspot discovery in graph streams,” in ICDM, 2013.

[51] S. Ranshous, S. Harenberg, K. Sharma, and N. F. Samatova, “A scalable approach for outlier detection in edge streams using sketch-based approximations,” in SDM,2016.

[52] K. Sricharan and K. Das, “Localizing anomalous changes in time-evolving graphs,” in SIGMOD, 2014.

[53] S. Bhatia, R. Liu, B. Hooi, M. Yoon, K. Shin, and C. Faloutsos, “Real-time anom- aly detection in edge streams,” Transactions on Knowledge Discovery from Data (TKDD), 2022.

[54] J. Sun, D. Tao, and C. Faloutsos, “Beyond streams and graphs: dynamic tensoranalysis,” in KDD, 2006.

[55] A. Beutel, W. Xu, V. Guruswami, C. Palow, and C. Faloutsos, “Copycatch: stopping group attacks by spotting lockstep behavior in social networks,” in WWW, 2013.

[56] E. Abdelhamid, M. Canim, M. Sadoghi, B. Bhattacharjee, Y.-C. Chang, and P. Kal- nis, “Incremental frequent subgraph mining on large evolving graphs,” TKDE,2017.

[57] L. Page, S. Brin, R. Motwani, and T. Winograd, “The pagerank citation ranking : Bringing order to the web,” in WWW, 1999.

[58] G. Cormode and S. Muthukrishnan, “An improved data stream summary: the count-min sketch and its applications,” Journal of Algorithms, 2005.

[59] A. Mcgregor, “Graph stream algorithms: a survey,” SIGMOD Record, 2014.

[60] S. Khuller and B. Saha, “On finding dense subgraphs,” in ICALP, 2009.

[61] R. Lippmann, R. K. Cunningham, D. J. Fried, I. Graf, K. R. Kendall, S. E. Web- ster, and M. A. Zissman, “Results of the darpa 1998 offline intrusion detectionevaluation.,” in Recent advances in intrusion detection, 1999.

[62] A. Shiravi, H. Shiravi, M. Tavallaee, and A. A. Ghorbani, “Toward developing a systematic approach to generate benchmark datasets for intrusion detection,” computers & security, 2012.

[63] M. Ring, S. Wunderlich, D. Scheuring, D. Landes, and A. Hotho, “A survey of network-based intrusion detection data sets,” Computers & Security, 2019.

[64] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, “Toward generating a new intrusion detection dataset and intrusion traffic characterization,” in ICISSP, 2018.

[65] I. Sharafaldin, A. H. Lashkari, S. Hakak, and A. A. Ghorbani, “Developing realistic distributed denial of service (ddos) attack dataset and taxonomy,” in ICCST, 2019.

[66] R. C. Forest. https://github.com/aws/random-cut-forest-by-aws, 2021.

[67] J. L. Carter and M. N. Wegman, “Universal classes of hash functions,” Journal of computer and system sciences, 1979.

102

KDD ’23, August 6–10, 2023, Long Beach, CA, USA

Sketch-Based Anomaly Detection in Streaming Graphs

附录

一个H-CMS估计保证证明

定理1。对于所有𝑘∈[𝑛𝑟]，让<s:1>𝑘(𝑢，𝑣)= (<s:1>𝑘'(𝑢)，<s:2>𝑘"(𝑣))，其中每个哈希函数都是从一个成对独立的族中均匀随机选择的。在这里，?我们允许两种情况都存在，即:<s:2> ' = <e:1> '和'≠<e:1> '。固定??> 0，设𝑛𝑟= ln𝛿and𝑛𝑏=≤≤≤𝑒;则，用1?

概率至少为1−𝛿，粽(𝑢，𝑣)≤粽(𝑢，𝑣)+粽(𝑀)。

Proof. Fix 𝑗∈[𝑛 𝑟]. Let 𝑎= (𝑢 𝑎, 𝑣𝑎) and 𝑏= (𝑢 𝑏, 𝑣𝑏) such that 𝑎≠𝑏. This implies that at least one of the following holds: 𝑢𝑎≠𝑢𝑏or 𝑣𝑎≠𝑣𝑏. Sinceℎ ′𝑗(andℎ ′𝑗′) is chosen uniformly at random from a pairwise-independent family, 𝑃(ℎ ′𝑗(𝑢 𝑎) = ℎ′𝑗(𝑢 𝑏)) = 𝑛1𝑏or 𝑃(ℎ ′𝑗′(𝑣 𝑎) = ℎ′𝑗′(𝑣 𝑏)) = 𝑛1𝑏. If 𝑃(ℎ ′𝑗(𝑢 𝑎) = ℎ𝑗′(𝑢 𝑏)) = 𝑛1𝑏, we have that 𝑃(ℎ 𝑗(𝑎) = ℎ𝑗(𝑏)) = 𝑃(ℎ ′𝑗(𝑢 𝑎) = ℎ′𝑗(𝑢 𝑏) ∧ℎ′𝑗′(𝑣 𝑎) = ℎ′𝑗′(𝑣 𝑏)) = 𝑃(ℎ ′𝑗(𝑢 𝑎) = ℎ′𝑗(𝑢 𝑏))𝑃(ℎ ′𝑗′(𝑢 𝑎) = ℎ′𝑗′(𝑢 𝑏)|ℎ ′𝑗(𝑢 𝑎) = ℎ′𝑗(𝑢 𝑏)) ≤ 1 = 𝜖𝑒. Similarly, if 𝑃(ℎ ′′ ′′ 1

𝑛𝑏

𝑗(𝑣𝑎)= <s:1>(𝑗)(𝑣𝑏))=𝑛𝑏，<s:2>(𝑗)(𝑎)=

△△△△△△△△△△△△△△△△△△△△△△△△△△△△△△△△△△△△△△△△△△△△△△△△△△△△△△△△△△△△△△△△△△△△△△△△△△△△△△△△△△△△△△△△△△△因此，在这两种情况下，碰撞的概率为:< < < < < < < < < < < < < < < < < < < > > > > < < < < < < < < > > > > < < < < < > > > > < < < < < > > > > < < < < < > > > > < < < < > > > > < < < < > > >由此，通过定义𝑋𝑎，𝑗= Í𝑏1{𝑎≠𝑏∧<s:1>𝑗(𝑎)= <s:2>𝑗(𝑏)}综(𝑏)，则E[𝑋𝑎，𝑗]≤Í𝑏(𝑎)E[1{𝑏≠

𝑏∧ℎ𝑗(𝑎) = ℎ𝑗(𝑏)}] ≤𝜖𝑒𝑀. Since 𝑦ˆ(𝑎) = min𝑗𝑦(𝑎) + 𝑋𝑎,𝑗 , this implies that 𝑃( 𝑦ˆ(𝑎) > 𝑦(𝑎) + 𝜖𝑀)) = 𝑃(min 𝑗𝑦(𝑎) + 𝑋𝑎,𝑗 > 𝑦(𝑎) + 𝜖𝑀)) = 𝑃(min 𝑗𝑋𝑎,𝑗 > 𝜖𝑀)) ≤𝑃(min 𝑗𝑋𝑎,𝑗 > 𝑒E[𝑋 𝑎,𝑗 ])). By the Markov’s inequality on the right-hand side, we have that 𝑃( 𝑦ˆ(𝑎) > 𝑦(𝑎) +𝜖𝑀)) ≤𝑃(min 𝑗𝑋𝑎,𝑗 > 𝑒E[𝑋 𝑎,𝑗 ])) ≤𝑒−𝑑≤𝛿. □

B边缘异常证明

B.1 AnoEdge-G

提案1。算法2的时间复杂度为𝑂(| <s:1> |∗𝑛𝑟∗??𝑏2)。算法2的内存复杂度为𝑂(𝑛𝑟∗𝑛𝑏2)。

证明。边-子矩阵-密度迭代删除行(或列)，可删除的行和列总数为𝑛𝑏+𝑛𝑏−2。在每次迭代中，该方法执行以下三个操作:(a)选择行和最小的行;(b)选取列和最小的列;(c)计算密度。我们保留𝑛𝑏-sized数组用于标记删除的行(或列)，并用于维护行和(或列和)。操作(a)和(b)取最大值𝑛𝑏steps来选择并标记行和(或列和)最小的行。基于选中的行(或列)更新列和(或行和)再次使用最大值𝑛𝑏steps。因此，(a)和(b)的时间复杂度为𝑂(𝑛𝑏))。密度是直接计算的，基于减去被删除的行和(或列和)，并从早期的密度值中减少行计数(或列计数)。Row-count和column-count被保留为单独的变量。因此，密度计算步骤的时间复杂度为𝑂(1)。边-子矩阵-密度过程的总时间复杂度为𝑂((𝑛𝑏+𝑛𝑏−2)∗(𝑛𝑏+𝑛𝑏+ 1))=𝑂(𝑛𝑏2)。

初始化和衰减H-CMS数据结构的时间复杂度为𝑂(𝑛𝑟∗𝑛𝑏2)。时间衰减操作在时间戳发生变化时应用，而不是对每个接收到的边都应用。更新计数操作为𝑛𝑟个矩阵更新一个矩阵元素值(𝑂(1)操作)，这一步的时间复杂度为𝑂(𝑛𝑟))。每条边的异常评分基于子矩阵密度计算过程，即

𝑂(𝑛𝑏2);𝑛𝑟matrices的时间复杂度变为𝑂(𝑛𝑟∗𝑛𝑏2)。因此，算法2的总时间复杂度为𝑂(| <s:1> |∗(𝑛??+??𝑟∗𝑛𝑏2))=𝑂(| <s:1> |∗𝑛𝑟∗𝑛𝑏2)。

对于过程边-子矩阵-密度，我们保留一个𝑛𝑏-sized数组来标记属于当前子矩阵的行和列，并维护行和和。边-子矩阵-密度过程的总内存复杂度为𝑂(4∗𝑛𝑏)=𝑂(𝑛𝑏)。

H-CMS数据结构的内存复杂度为𝑂(𝑛𝑟∗𝑛𝑏2)。密集子矩阵搜索和密度计算过程需要𝑂(𝑛𝑏))内存。对于𝑛𝑟matrices，这就变成了𝑂(𝑛𝑟∗𝑛𝑏))。因此，算法2的总内存复杂度为𝑂(𝑛𝑟∗𝑛𝑏2 +𝑛𝑟∗𝑛𝑏)=𝑂(𝑛𝑟∗𝑛𝑏2)。

□

B.2 AnoEdge-L .

提案二。算法3的时间复杂度为𝑂(𝑛𝑟∗𝑛𝑏2 + | <s:1> |∗𝑛𝑟∗𝑛𝑏)。算法3的内存复杂度为𝑂(𝑛𝑟∗𝑛𝑏2)。

证明。如命题1所示，H-CMS的时间复杂度为𝑂(𝑛𝑟∗𝑛𝑏2)，更新操作为𝑂(𝑛𝑟)。当前子矩阵(𝑆𝑐𝑢𝑟,𝑇𝑐𝑢𝑟))是基于展开和压缩子矩阵运算更新的。(a)我们保留一个𝑛𝑏-sized数组来标记当前子矩阵的行(或列)，同时也维护行和(或列和)。展开子矩阵的操作依赖于行(𝑢)和列(𝑣)中的元素，并且密度是通过考虑这些元素来计算的，因此需要最大的𝑛𝑏steps。在添加行(或列)时，相关的列和(或行和)也被更新，取最大值𝑛𝑏steps。因此展开操作的时间复杂度为𝑂(𝑛𝑏))。(b)压缩子矩阵运算迭代地移除行和列。通过在𝑂(𝑛𝑏))时间内选择行和(或列和)最小的行(或列)来执行行(或列)消除。删除的行(或列)会影响相关的列和(或行和)，并在𝑂(𝑛𝑏)time)中更新。因此，删除一行(或列)的时间复杂度为𝑂(𝑛𝑏))。压缩子矩阵删除曾经通过扩展子矩阵操作添加的行(或列)，在更坏的情况下是𝑂| <e:1> |。

Expand and condense submatrix operations are performed for 𝑛𝑟matrices. Likelihood score calculation depends on elements from rowℎ(𝑢) and columnℎ(𝑣), and takes𝑂(𝑛 𝑟∗𝑛𝑏) time for𝑛 𝑟matrices. Therefore, the total time complexity of Algorithm 3 is 𝑂(𝑛 𝑟∗𝑛𝑏2 + |ℰ| ∗𝑛𝑟+ |ℰ| ∗𝑛𝑟∗𝑛𝑏+ |ℰ| ∗𝑛𝑟∗𝑛𝑏+ |ℰ| ∗𝑛𝑟∗𝑛𝑏) = 𝑂(𝑛 𝑟∗𝑛𝑏2 + |ℰ| ∗𝑛𝑟∗𝑛𝑏).

Memory complexity of the H-CMS data structure is 𝑂(𝑛 𝑟∗𝑛𝑏2 ). To keep current submatrix information, we utilize 𝑛𝑏-sized arrays similar to Proposition 1. For 𝑛𝑟matrices, submatrix information requires 𝑂(𝑛 𝑟∗𝑛𝑏) memory. Hence, total memory complexity of Algorithm 3 is 𝑂(𝑛 𝑟∗𝑛𝑏2 + 𝑛𝑟∗𝑛𝑏) = 𝑂(𝑛 𝑟∗𝑛𝑏2 ). □

C图异常证明

C.1 AnoGraph

提案3。算法4的时间复杂度为𝑂(|𝒢|∗𝑛𝑟∗𝑛𝑏2 + | <s:1> |∗𝑛𝑟)。算法4的内存复杂度为𝑂(𝑛𝑟∗𝑛𝑏2)。

Proof. Procedure AnoGraph-Density iteratively removes row (or column) with minimum row-sum (or column-sum). Maximum

103

Siddharth Bhatia et al.

KDD ’23, August 6–10, 2023, Long Beach, CA, USA

可删除的行数和列数为𝑛𝑏+𝑛𝑏−2。我们保留𝑛𝑏-sized数组来存储当前子矩阵的行和列，以及行和和和。在每次迭代中，选择行和(或列和)最小的行(或列)需要𝑂(𝑛𝑏))时间，更新相关的行和(或列和)也需要𝑂(𝑛𝑏))时间。密度是基于当前子矩阵行和和列和在𝑂(𝑛𝑏)time)中计算的。每次迭代占用𝑂(𝑛𝑏+𝑛𝑏+𝑛𝑏)=𝑂(𝑛𝑏)time)。因此，AnoGraph-Density过程的总时间复杂度为𝑂((𝑛𝑏+𝑛𝑏−2)∗𝑛𝑏)=𝑂(𝑛𝑏2)。

Initializing the H-CMS data structure takes𝑂(𝑛 𝑟∗𝑛𝑏2 ) time. When a graph arrives, AnoGraph: (a) resets counts that take 𝑂(𝑛 𝑟∗𝑛𝑏2 ) time; (b) updates counts taking 𝑂(1) time for every edge update; (c) computes submatrix density that follows from procedure Ano-Graph-Density and takes 𝑂(𝑛 𝑏2 ) time. Each of these operations is applied for 𝑛𝑟matrices. Therefore, the total time complexity of Algorithm 4 is 𝑂(𝑛 𝑟∗𝑛𝑏2 + |𝒢| ∗𝑛𝑟∗𝑛𝑏2 + |ℰ| ∗𝑛𝑟+ |𝒢| ∗𝑛𝑟∗𝑛𝑏2 ) = 𝑂(|𝒢| ∗𝑛𝑟∗𝑛𝑏2 + |ℰ| ∗𝑛𝑟), where |ℰ| is the total number of edges over graphs 𝒢.

对于过程AnoGraph-Density，我们保留𝑛𝑏-sized数组来标记当前子矩阵的行和列，并维护行和和。因此，AnoGraph-Density过程的内存复杂度为𝑂(4∗𝑛𝑏)=𝑂(𝑛𝑏))。

H-CMS数据结构需要𝑂(𝑛𝑟∗𝑛𝑏2)内存。密度计算依赖于AnoGraph-Density程序，占用𝑂(𝑛𝑏))内存。因此，算法4的总内存复杂度为𝑂(𝑛𝑟\*𝑛𝑏2)。

□

C.2 证明:AnoGraph-K

命题4。算法5的时间复杂度为𝑂(|𝒢|∗𝐾∗𝑛𝑟∗𝑛𝑏2 + | <s:1> |∗𝑛𝑟)。算法5的内存复杂度为𝑂(𝑛𝑟∗𝑛𝑏2)。

证明。AnoGraph-K-Density程序中的相关操作直接遵循边-子矩阵-密度程序，该程序具有𝑂(𝑛𝑏2)时间复杂度。边-子矩阵-密度法称为𝐾times，因此，AnoGraph-K-Density法的总时间复杂度为𝑂(𝐾∗𝑛𝑏2)。

对于算法5，我们初始化一个耗时𝑂(𝑛𝑟∗𝑛𝑏2)的H-CMS数据结构。当一个图到达时，AnoGraph-K:(a)重置花费𝑂(𝑛𝑟∗𝑛𝑏2)时间的计数;(b)更新每次边缘更新花费𝑂(1)时间的计数;(c)计算从程序AnoGraph-K-Density得出的子矩阵密度，并花费𝑂(𝐾∗𝑛𝑏2)时间。这些操作中的每一个都应用于𝑛𝑟matrices。因此，算法5的总时间复杂度为𝑂(𝑛𝑟∗𝑛𝑏2 + |𝒢|∗𝐾∗𝑛𝑟∗𝑛𝑏2 + | <s:1> |∗𝑛𝑟+ |𝒢|∗𝑛𝑟∗𝑛2)=𝑏(|𝑂|∗𝒢𝐾∗𝑛2 + |𝑟|∗𝑛)，其中|??|为图上的边总数𝑛。

𝐾submatrices的密度是独立计算的，算法过程AnoGraph-K-Density的存储复杂度与边-子矩阵-密度过程即𝑂(𝑛𝑏))的存储复杂度相同。

Maintaining the H-CMS data structure requires 𝑂(𝑛 𝑟∗𝑛𝑏2 ) mem-ory. Density computation relies on AnoGraph-K-Density proce-dure, and it requires 𝑂(𝑛 𝑏) memory. Therefore, the total memory complexity of Algorithm 5 is 𝑂(𝑛 𝑟∗𝑛𝑏2 ). □

D实验设置

所有实验均在2.4𝐺𝐻<s:1> Intel Core𝑖9处理器，32𝐺RAM，运行OS𝑋10.15.3上进行。对于我们的方法，我们保留𝑛𝑟=2和𝑛𝑏=32，以便与使用𝑛𝑏2= 1024桶的MIDAS进行公平的比较。算法2和算法3的时间衰减因子𝛼=0.9。我们为算法5保留𝐾=5。图异常的AUC显示为边缘阈值，DARPA为50，其他数据集为100。DARPA数据集的时间窗口为30分钟，其他数据集的时间窗口为60分钟。

E基线

我们使用作者提供的DenseStream[1]、SedanS-pot[11]、MIDAS-R [3] (c++)、PENminer[12]、F-FADE[13]、DenseAlert[1]和AnomRank[14]的开源实现，参数设置按照原论文建议。对于聚光灯 b[2]，我们使用了随机切割森林[66]和Carter Wegman哈希[67]的开源实现。

E.1 边缘异常

(1) SedanSpot: sample\_size = 10000, num\_walk = 50, restart\_prob 0.15

(2) MIDAS:所有测试的cms大小为2行乘1024列。对于MIDAS-R，衰减因子= 0.6。

(3) PENminer: ws = 1, ms = 1, view = id, alpha = 1, beta = 1, gamma = 1

(4) DenseStream:我们保留默认参数，即order = 3。

(5) F-FADE: embedding\_size = 200, W\_upd = 720, T\_th = 120, alpha = 0.999, M = 100

对于t\_setup，我们总是使用数据集10𝑡ℎpercentile处的时间戳值。

E.2 图异常

(1)聚光灯:K = 50, p = 0.2, q = 0.2

(2) DenseAlert:我们保持默认参数，即order = 3和window=60。

(3) AnomRank:保持默认参数，即阻尼因子c = 0.5，节点评分向量阈值epsilon的L1变化= 10−3。我们保留了1/4𝑡ℎnumber的图，用于初始化各自论文中提到的均值/方差。

104