基于 Storm 的分布式黑洞检测系统

第一章 绪论

1.1 研究背景

传感技术的进步带来了大量的人类移动数据，例如出租车流量、地铁刷卡数据、自行车行程数据和呼叫详细记录(CDR)，这些数据可以构成一张时空图(Spatio-Temporal Graph)。这是一个有向图，其中顶点和边分别具有地理空间位置和空间长度，并且与时空属性相关联。如图 1 所示，在道路网络中，具有地理位置的 POI 被视为顶点，POI 之间连接的道路被视为边缘；POI 之间的交通流是随着时间而变化的，同时可以根据需要从时空图中删除或添加边缘。基于这个特性，利用时空图的信息可以得到局部流量是否出现异常，为流量分流等社会流动管理提供数据支持。

在本问题中，定义了两种特殊的流量状态。一种被称为是“黑洞”的异常，即总流入量大于总流出量阈值。另一种是“火山”，即总流出量大于流入量阈值。

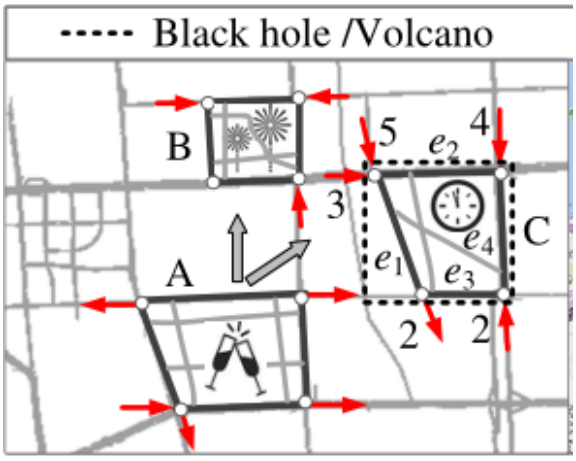


图 1 上海外滩 STG 图

如图 1 所示，在 2015 年新年倒计时临近时候，大多数人开始离开酒吧前往外滩参加倒计时活动，或者前往烟花表演观赏区，因此形成了火山 A，黑洞 B 以及黑洞 C；由于没有足够的空间容纳这过度的人流量，黑洞 C 区域出现了严重的踩踏事件。这实际上是可以避免的，在灾难发生的前几个小时就能捕获到区域 C 的“黑洞”信息，根据此信息对游客进行引导分流或者设置临时的交通管制，可以避免事故的发生。

因此，本文基于时空图提出了一个高效索引的索引方式，并在此基础上提出了一种在时间间隔内从 STG 中检测出黑洞的有效算法。

1.2 目前的挑战

在现实中，黑洞通常是受空间和流约束的动态图，同时由于其多条边和多个顶点的组合，其存在许许多多的子图。同时，由于黑洞的影响范围和持续时间都会随着时间的推移而变化，这为算法设计带来不小的挑战。

1.3 当前研究工作的不足之处

当前常用的流量黑洞/火山检测算法大多没有考虑到城市出行流量数据的固有特性—流式数据、规模大等。流式数据，对计算实时性要求较高，且规模较大不易储存。

同时，由于城市流量分布不均匀地特性，如果将所有数据统一汇总运算，需要大量的算力，这在某种程度上是一种浪费。例如，城中村，远离 CBD 的区域等，此类区域的流量明显呈现阶段性特征，如上下班高峰期；因此，在非上下班高峰期时段可以减少对该区域数据的算力投入。因此，分布式计算是必要的。可以将城市划分为多个网格单元，相邻近的、流量上具有高度关联的网格单元设置为一个处理组，每个处理组单独对应一个处理组件，该组件只进行检测该处理组内部网格单元的流量情况。

1.4 本文的工作要解决的问题以及方法

本文所构建的城市流量黑洞检测系统基于 Storm 的分布式计算架构，将城市 STG 图划分为多个网格单元，将相邻近的网格单元汇聚成处理组，每个处理组对应单独的 Bolt 进行实时分析检测。系统采用 TCP 与数据提供方进行通信，在经过对数据的处理后分发到对应的 Spout 中。该架构有利于控制算力的具体投入，更精准地监测某些具有阶段性流量特征的区域。

第二章 系统框架

2.1 Storm 基本介绍

Storm 是开源的分布式实时计算系统，在实时分析、在线机器学习、连续计算、分布式 RPC、ETL 等场景中广泛使用。Storm 集成了多种消息队列技术和数据库技术。 Storm 具有以下特性：

1. 用例广泛：Storm 可以处理用户消息，并更新数据库，完成数据流的持续查询任务，然后将结果流送到客户端。

2. 可伸缩性：Storm 每秒处理大量信息，当需要拓展拓扑时，只需要向拓扑中添加主机，增加拓扑的并行设置。

3. 可靠性：通过 ACK 机制，Storm 对每条消息的处理是可靠的。

4. 容错性：当计算中发生错误时，Storm 只需要重新分配任务再执行即可。

5. 高性能低延迟性：原生的流处理系统，可以做到毫秒级处理。

2.2 Storm 系统架构

Storm 集群中存在两种类型的节点：运行 Nimbus 服务的主节点和运行 Supervisor 服务的工作节点。Storm 集群由一个主节点和多个工作节点组成，主节点上运行一个名为“Nimbus”的守护进程，用于分配代码、布置任务及故障检测；每个工作节点上则运行一个名为“Supervisor”的守护进程，用于监听工作、开始并终止工作进程。Nimbus 和 Supervisor 之间的协调工作是通过 zookeeper 来完成的。Nimbus 和 Supervisor 是无状态的，其元数据存储在 Zookeeper 中，使得系统具有很高的容错性。

Worker 由 Supervisor 负责启动，一个 worker 可以有多个 Executor 执行线程，每个 Executor 又可以包含一个或多个 Task，其中 Task 为 Storm 中最小处理单元。每个 Executor 都会启动一个消息循环线程，用于接收、处理和发送消息，当 Executor 收到属于其下某一 Task 的消息后，就会调用该 Task 对应的处理逻辑对消息进行处理。

2.3 分布式黑洞检测系统架构

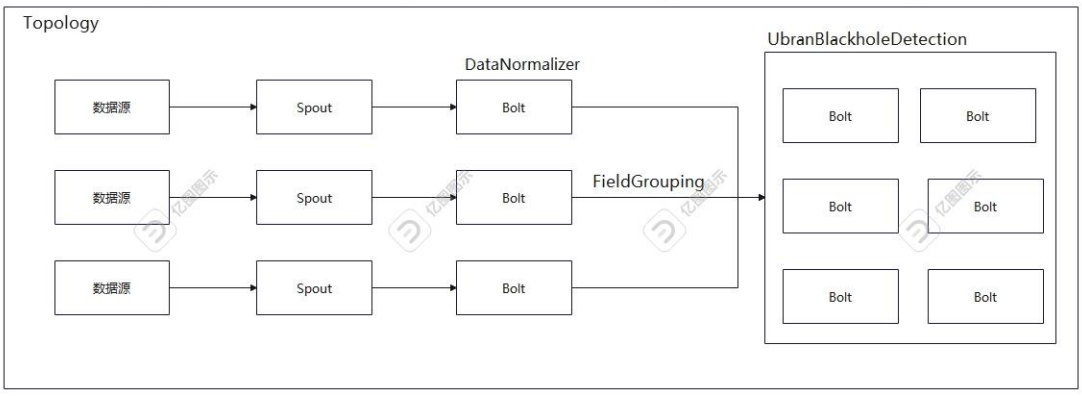


图 2 分布式黑洞检测系统架构

在 Storm 中，Spout 有 3 中获取数据的模式：直接连接、消息队列和 DRPC。在本系统是采用直接连接的模式，Spout 与数据源 TCP 直接通信，接收城市流量数据流。

在 Storm 中，Bolt 是一个组件，以元组作为输入，生成元组作为输出。在客户端主机中创建 Bolt，序列化到拓扑，并提交到集群的主控节点。集群启动 Worker，反序列化 Bolt，调用并处理元组。在本系统中，有两类 Bolt，一种是接收 Spout 发来的元组，对其进行数据格式处理的 DataNormalizer；

元组经过 DataNormalizer 的处理，得到数据：time(时间间隔)、nodef(起始目标点)、nodef.lat(起始点维度)、nodef.lon(起始点经度)、nodet(终止目标点)、nodet.lat(终止点维度)、nodet.lon(终止点经度)、flow(流量)；其中，格局节点的经纬度可以判断出当前节点所在的网格单元；在进行元组分发的时候按字段分组 Fields grouping，相同的网格单元所在的处理组分发至相同的 Bolt 进行处理。

另一种 Bolt 是进行流量黑洞检测的组件，收集 DataNormalize 发来的元组，当信息达到一定阈值之后开始查找当前时间间隔内的城市流量黑洞，并将相关黑洞信息反馈给客户端。

2.3 相关术语的定义

2.3.1 时空图(STG)$G = (V, E)$是一个有向图，其中$v \in V$具有实际上的地理空间位置，$e \in E$有实际空间上的长度，且每个顶点/边之间属性随着时间而变化，如流入和流出。

2.3.2 每个边缘$e \in E$，时间间隔t内的e流入量$f_{in}(e,t) = \sum_{e' \in (E-e)} f(e', e, t)$，其中$f(e', e, t)$表示在时间间隔$t$从$e'$流向$e$的流量。相似地，流出量定义为$f_{out}(e,t) = \sum_{e' \in (E-e)} f(e', e, t)$。

2.3.3 S 的流入量定义为$f_{in}(S,t) = \sum_{e \in E \cap e' \in (G.E-S.E)} f(e', e, t)$。相似地，S 的流出量定义为$f_{out}(S,t) = \sum_{e \in E \cap e' \in (G.E-S.E)} f(e', e, t)$

2.3.4 在 STG 中，边e的实际流量为$f_{a\alpha} = f_{in}(e,t) - f_{out}(e,t)$。相似地，子图 S 的实际流量定义为$S.f_a = f_{in}(S,t) - f_{out}(S,t)$。

2.3.5 在 STG 中，子图 S 是黑洞当且仅当：$S.f_a \geq \tau$，并且MBB$(S) \leq d$，其中τ和d是流量阈值和空间阈值，MBB 表示空间最小边界框。在本文中，将 d 设置为 MBB 对角线长度；在给定 d 约束的条件下，τ表示该区域路网的通行能力，即$\tau = \alpha \times \sum_{e \in S.E} e.l \times e.n/L$，其中$e.n$是车道数，L 是车辆的平均长度，α是一个∈ [0,1]的系数。

2.3.6 在 STG 中，子图 S 是火山当且仅当：$-S.f_a \geq \tau$，并且MBB$(S) \leq d$。

2.4 问题定义

给定 STG G，时间间隔 t，流量阈值τ以及空间阈值d，在 G 检测出黑洞就是找到一组满足下列条件的子图集合$BH \in S_1, S_2, \ldots, S_n$：

(1) $\forall S \in BH$满足定义 5

(2) $\forall S \in \{G - BH\}$不满足定义 5

(3) $\forall S_i, S_j \in BH$，当$i \neq j$时，$S_i \cap S_j = \emptyset$，且$S_i \cup S_j$不满足定义 5。

定理 1：在 STG 中检测黑洞等价于检测火山。

证明：假设$G' = (V', E')$是图 G 的逆图，则可知$f_{in}(e', t) = f_{out}(e, t)$。如果

S 是 G 中的黑洞，则$f_{in}(S,t)-f_{out}(S,t)\geq\tau$。因此，相对应的$S'$在$G'$中是$f_{out}(S',t)-f_{in}(S,t)\geq\tau$，这就是火山的定义。因此，可以得知在 STG 中检测黑洞相当于在 G 的逆图中检测火山；类似地，可以证明在 STG 中检测火山就是在逆图中检测黑洞。

定理 2：在 STG 中检测黑洞是 NP-Complete 类问题。

证明：一个时间间隔内的 STG 可以被视为一个加权有向图，其中每条边上的流表示边的权重。考虑以下已知为 NP 完全的局部密度图聚类问题：给定加权有向图$G=(V,E)$，从 G 中找出一组子图，每个子图 S 具有 k 个顶点，且其密度为$\delta_{int}(S)=\frac{1}{|S|(|S|-1)}\sum_{e\in S.E}e.f_{\mathrm{a}}\geq r$。黑洞检测可以被限制为局部密度问题，只允许 d 被设置为由 k 个顶点形成的最大可能 MBB，且$\tau=r\times k(k-1)$。因此，局部密度问题是黑洞检测问题的一个特例，它证明了从 STG 中检测黑洞是 NP-Completeness。

2.5 黑洞检测算法

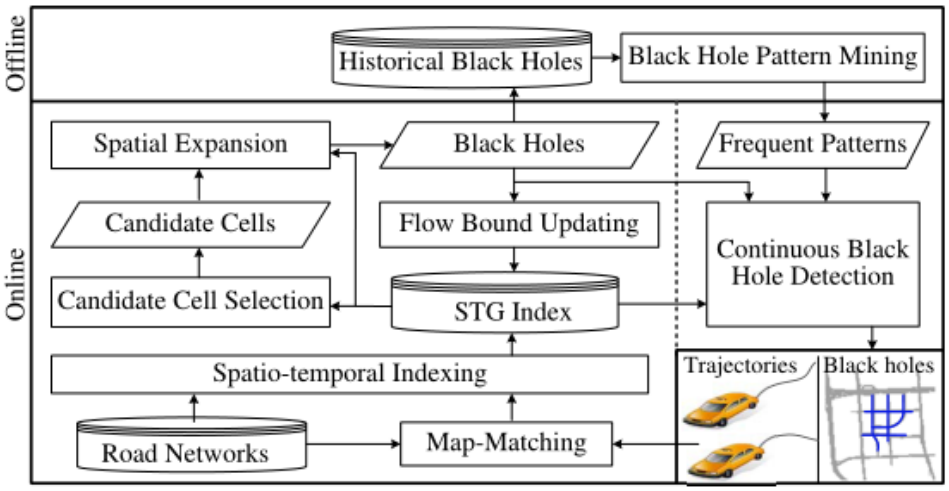


图 3 黑洞检测算法框架

图 3 显示了整个检测方法的框架，他由两部分组成：在单个时间间隔内的黑洞检测以及从连续时间间隔内的检测。

单个时间间隔内黑洞检测：先使用地图匹配算法将最近时间间隔内接收到的 GPS 轨迹映射到道路网络上，然后计算出间隔内每个路段的流入/流出。使用时空索引算法，将城市划分为多个不相交的网络单元，每个网格单元可以覆盖几个路段，并构建 STG 索引，该索引保持每个单元的实际流量上限和属于该单元的每个路段的实际流量。基于 STG 索引，候选单元选择算法找到可能包含黑洞的候选单元。空间拓展算法从具有最大实际流量的候选单元开始，然后拓展单元中具有最大实际流量的路段到黑洞中，逐渐增加相邻路段，直到不再满足

空间和流量限制。一旦检测到黑洞，重新计算剩余候选单元的实际流量上限，根据更新的上限，从候选集合中删除单元。不断重复空间拓展算法，直到所有的候选单元都被检查完毕。

连续黑洞检测：该部分通过使用历史先验进一步提高黑洞检测的效率和有效性。因为，虽然黑洞会随时时间而改变，但是在两个连续的时间间隔内地理空间不会产生巨大的变化。其次，黑洞的出现遵循一定的模式。从历史先验中可以检测到频繁出现的子图模式，这些黑洞是在同一时间间隔但不同时间段（如几天、几月等）内被检测到的。因此，在进行 t+1 的空间拓展算法时，可以利用时间间隔 t 内检测到的黑洞信息以及 t+1 的频繁黑洞模式作为初始图，进而快速识别 t+1 中的新黑洞，无需从根本上重建黑洞。

第三章 STG 索引

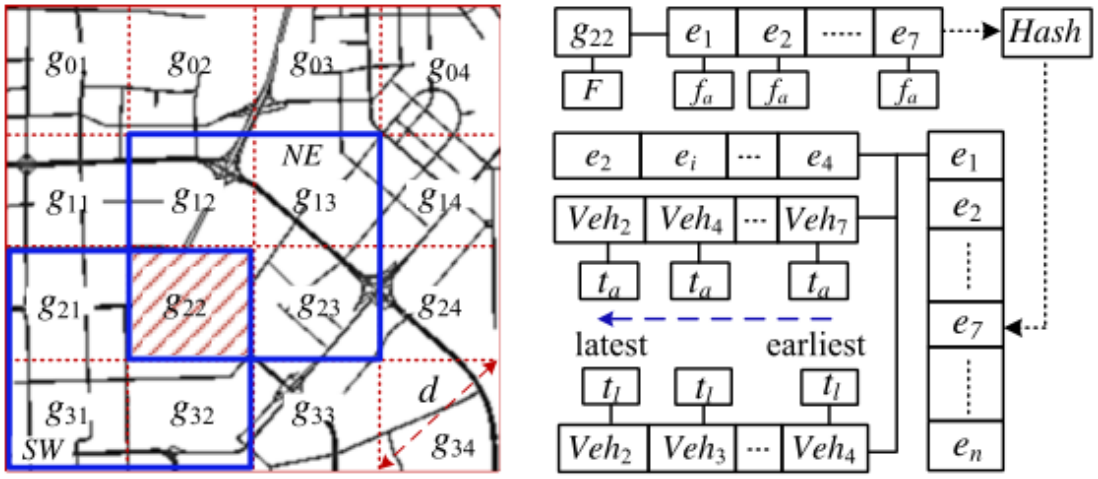


图 4 STG 索引

STG 索引包含了 STG 的结构、每条边的动态流信息以及不同边之间的空间关系。如图 4 所示，将城市划分为不相交的统一网格单元。不同网格单元之间的空间关系由矩阵保持。例如，给定关系矩阵，可以快速确定 g22 的邻域是 g11~g33。同时，对于每个网格单元 g 都对一个一个列表 L，该列表存储网格单元的路段 ID。g.L 中的每个路段 e 都与最近时间间隔内自身实际流量$e.f_\alpha$相关联。对于每个网络单元 g，我们通过下述公式计算实际流量 g.F：

$$g.F = \sum_{e_i \in g.L \cap e_i.f_a > 0} e_i.f_a$$

同时，建立一个邻接表来管理 STG 的结构和动态流。对于相邻列表中的每个路段 e，分别维护三个列表。第一个列表存储了和路段 e 连接的路段信息；第

二个列表存储的是按到达 e 的时间进行排序的车辆 ID 列表；第三个列表存储的是按照离开 e 的时间进行排序的车辆 ID 列表。第一个列表的静态的，不随着时间进行变化；第二、第三个列表在每次地图匹配后进行更新。在实际应用中，只需要存储最近时间间隔的车辆 ID。在计算一个时间间隔内边缘的实际流量时，只需要计算到达时间在时间间隔内而离开时间在时间间隔之外的车辆数量。在 g.L 中每个路段的 ID 通过哈希进行索引，连接到邻接表中的相应记录。

第四章 候选单元选择

候选单元选择算法通过检查每个网格单元及其相邻网格单元的正实际流量来快速选择可能具有黑洞的单元。

具体而言，如图 3 所示，通过将每个单元的大小设置为黑洞的空间约束 d，确保每个黑洞最多可以同时与四个单元相交。如果四个单元的实际流量仍小于给定的流量阈值τ，则不可能在四个单元中找到任何黑洞。给定一个网格单元，可以检验四个方向的四个单元组合的实际流量：西北(NW)、东北(NE)、东南(SE)和西南(SW)。对于每个方向，可以获得的实际流量分别表示为$F_{NW}(g)$、$F_{NE}(g)$、$F_{SE}(g)$、$F_{SW}(g)$，则

$$F_{NE}(g_{22}) = g_{12}.F + g_{13}.F + g_{22}.F + g_{23}.F$$

定义网格单元的流上界$UB(g)$为：

$$UB(g) = \mathrm{Max}\big(F_{NW}(g), F_{NE}(g), F_{SE}(g), F_{SW}(g)\big)$$

如果$UB(g) < \tau$，那么 g 不是候选单元，即不应该从 g 开始寻找黑洞；根据 STG 索引，可以快速计算出每个网格单元的流上界，快速剪去许多不可能的单元，剩余的单元将作为候选单元。

定理 3：子图$S = (V, E)$的实际流量等于其自身边的实际流量之和，即$S.f_{\mathrm{a}} = \sum_{e \in S.E} e.f_{\mathrm{a}}$。

证明：

$$\sum_{e \in S.E} e.f_{\mathrm{a}}$$

$$= \sum_{e \in S.E} \big(f_{in}(e) - f_{out}(e)\big)$$

$$= f_{in}(S) - f_{out}(S)$$

$$= f_{\mathrm{a}}(S)$$

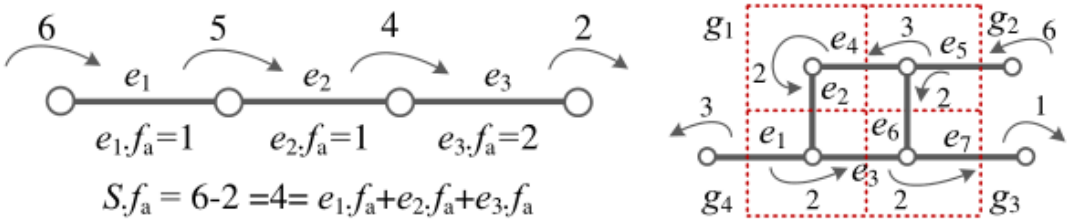


图 5 定理 3 的样例

图 5 是定理 3 的一个样例，如图 4 左侧所示，S 的实际流量等于e_1，e_2，e_3的实际流量之和。同样地，如右图所示，落在四个网格单元中的子图 S 的实际流量$S.f_{\mathrm{a}} = \sum_{j=1}^{7} e_j.f_{\mathrm{a}} = 2$，这实际上比$\sum_{i=1}^{4} g_i.F = (2+1)+(1+1+1)+(0+2+1)+(2+0) = 11$要来的小，因为一条边可能属于两个网格单元，那么这条边就会被计算成两次。因此，在这里证明了，对于每个网格单元 g，$F_{NW}(g) \geq \sum_{e\in NW(g)} e.f_{\mathrm{a}}$，$F_{NE}(g) \geq \sum_{e\in NE(g)} e.f_{\mathrm{a}}$，$F_{SW}(g) \geq \sum_{e\in SW(g)} e.f_{\mathrm{a}}$和$F_{SE}(g) \geq \sum_{e\in SE(g)} e.f_{\mathrm{a}}$。如果$UB(g) < \tau$，那么$\sum_{e\in NW(g)} e.f_{\mathrm{a}} < \tau$，因此由$e \in NW(g)$形成的子网不是一个黑洞。

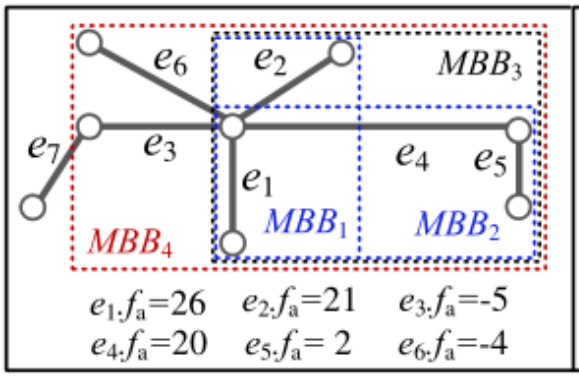
第五章 空间拓展算法

空间选择算法从具有最大$g.F$的候选单元开始，根据以下策略将单元中的初始边拓展为黑洞。在初始时，算法选择具有最大$e.f_\alpha$的边作为初始边，然后逐个添加 e 的相邻边以形成黑洞。为了确定添加顺序，需要计算每个边的优先级分数：

$$R(S,e) = \begin{cases} f_a(e)/\Delta & \Delta \neq 0 \; and \; e.f_a \geq 0 \\ f_a(e) \times \Delta & \Delta \neq 0 \; and \; e.f_a < 0 \\ +\infty & \Delta = 0 \; and \; e.f_a \geq 0 \\ -\infty & \Delta = 0 \; and \; e.f_a < 0 \end{cases}$$

其中 e 是要添加到现有黑洞 S 中的边，$\Delta = \mathrm{MBB}(S+e).\mathrm{diagonal} - \mathrm{MBB}(S).\mathrm{diagonal}$，表示如果将 e 添加到黑洞 S 中，MBB(S)的值将增加。

具有最高优先级分数的边将被添加到黑洞中。该分数为具有高实际流量的边提供了更高的优先级，并导致 MBB 的小幅增长。同时，那些逆流和逆空间阈值的边将不会被添加进来。空间拓展算法会不断迭代筛选，直到没有边可以被添加进黑洞中。



	S	MBB(S)	Δ	R(S,e)
1	$\emptyset$	0		
2	$\{e_1\}$	0		
3	$\{e_1,e_2\}$	MBB_1=3	3	7
4	$\{e_1,e_2,e_4\}$	MBB_2=4	2	10
5	$\{e_1,e_2,e_4,e_5\}$	MBB_3=5	0	$+\infty$
6	$\{e_1,e_2,e_4,e_5,e_6\}$	MBB_4=7	2	-8

图 6 空间拓展算法图解

图 6 给出了一个示例来说明空间拓展算法，假设空间的阈值为 7，流量阈值为 40。算法从e_1开始，因为其具有最大的实际流量。然后拓展其相邻边e_2，e_3，e_4和e_6，分别计算这四条边的优先级分数。因为e_3和e_6的流量为负，算法直接不纳入比较队列，只需比较e_2和e_4。如果将e_2添加入 S 中，$\Delta =$ MBB$_1$.diagonal $-$ 0 $= 3$，则$R(S, e_2) = 7$；同样地，可以计算出$R(S, e_4) = 5$。因为$R(S, e_2) < R(S, e_4)$，e_2会优先被添加进 S 中，MBB(S).diagonal $= 3$。

再添加e_2后，e_4可以被添加进 S 中，此时MBB(S) $=$ MBB$_3$；之后e_5也被添加进 S 中。虽然e_3和e_6的流量为负，但是它们可以作为桥梁帮助算法找到实际流量较高的附近边缘。因为e_6的优先级分数高于e_3，且添加e_6不会打破空间和流量的限制，因此拓展e_6。不过，为了控制黑洞的质量，应该尽可能少地添加这类型地边，因此在添加e_6之后，e_3的优先级被设置为$-\infty$。e_7不能包含在黑洞中，因为它打破了空间阈值的限制。最后经过拓展的子图$S = e_1, e_2, e_4, e_5, e_6$是一个黑洞，因为它的实际流量$S.f_a = 26 + 21 + 20 + 2 - 4 = 65 > 40$。

当基于最高$g.F$的候选单元检测到黑洞，来自其他候选单元的边可能已经包含在该黑洞中，因此需要更新剩余候选单元的流上界UB，并修剪去$UB < \tau$的候选单元。随后，算法继续从具有最大的$g.F$的候选单元进行拓展，直到候选单元集合为空。

第六章 连续检测

由于黑洞会随着时间的推移而变化，因此需要不断对其进行检测，以便通知用户即使做出决策。为了提高检测算法的效率和有效性，在这里引入一种连续检测的算法，该算法利用在过去的时间间隔中检测到的黑洞和火山信息，以及长时间段中挖掘出的黑洞模式，来协助在连续的时间间隔内检测到黑洞信息。

连续检测算法的动机有两个：

第一，虽然黑洞随时间演变，但在实际中，它在地理空间中不会发生巨大的变化。但值得注意的是， t+1 的黑洞可能起源于 t 间隔的火山，反之亦然。如

在体育赛事开始前，体育场周围会产生黑洞；在体育赛事结束后，体育场周围形成火山。因此，可以利用在时间间隔 t 中检测到的黑洞/火山来减少在 t+1 时刻的检测开销。

第二，黑洞的出现遵循一定的模式。比如，在工作日的早晨商业区周围往往会形成黑洞。因此，使用 gSpan 算法从不同天的相同时间间隔内检测到的历史黑洞中挖掘封闭的频繁子图。如果不存在与 S 具有相同支持的超图，则称 S 在图数据库中是封闭的。

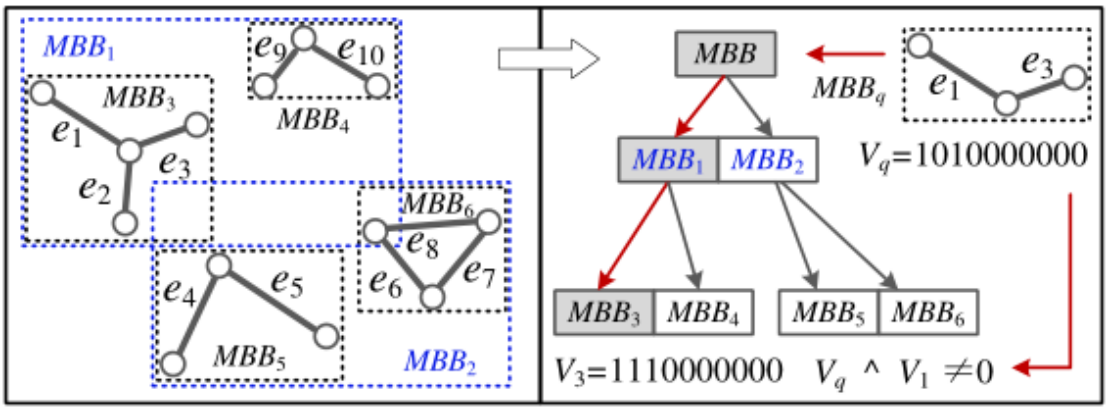


图 7 利用黑洞模式匹配黑洞

连续检测算法由五个步骤组成：

(1) 使用第四章技术选择一组候选单元集合 C。

(2) 检索在时间间隔 t 中检测到的黑洞和火山，这些黑洞和火山在 t+1 中不与任何黑洞模式重叠，然后将它们与黑洞模式放在一个并集$BP_{t+1} \cup BV_t$中。为了加快检索速度，使用 R-Tree 组织每个时间间隔的黑洞模式，如图 6 所示。给定查询的黑洞 q，首先查找那些 MBB 和MBB_q相交的叶节点。然后通过异或操作快速检查黑洞模式是否存在相同的边。例如，$V_q = 1010000000$表示S_q有e_1和e_3。

(3) 用集合中的正实际流量来检查每个黑洞/火山 S。如果$S.f_a \geq \tau$，则 S 被视为 t+1 中的一个新黑洞。如果 $0 < S.f_a < \tau$，尝试通过向 S 添加一些具有正实际流量的相邻边，或通过从 S 中移除一些具有负实际流量的非桥边来寻找黑洞。桥边是那些删除会增加其连接组件数量的边。在图$G = (V, E)$中查找桥边的方法是广泛可用且非常高效，它的复杂度是$O(|V| + |E|)$。此外，将边添加到 S 中是从具有较高优先级分数的边开始。相反，优先级分数较低的边将更早地从 S 中删除。

(4) 一旦发现黑洞，更新每个单元网格地流上限，然后进行剪枝操作。

(5) 如果网格候选单元集合不为空，则继续调用空间拓展算法。

Algorithm 1: ContDetection($G, BH_t, BP_{t+1}, d, \tau$)

Input: BV_t: Black holes and volcanos in t, BP_{t+1}: Black hole patterns of t+1, an STG, G, a spatial threshold d and a flow threshold τ.
Output: BH_{t+1}: Black holes detected in t+1.

```
C ← CandidateSelection (G, d, τ); BH_{t+1} ← ∅;
For each S ∈ BV_t
    If ∃S' ∈ BP_{t+1}, s.t. S ∩ S' ≠ ∅ //non-overlap
        BV_t ← BV_t − S;   //remove a black hole/volcano
For each S ∈ BP_{t+1} ∪ BV_t
    If S.f_a ≥ τ in t + 1
        BH_{t+1} ← BH_{t+1} + S;
    Else if 0 < S.f_a < τ in t
        While MBB(S) < d
            If S has a neighboring edge with a positive actual flow
                Add such edges to S according to Equation 4;
            Else if S has non-bridge edges with negative actual flow
                Remove such edges from S according to Equation 4;
            If S.f_a ≥ τ
                BH_{t+1} ← BH_{t+1} + S;
                C←UpperBoundUpdating(S, G); Continue;
BH_{t+1} ←SpatialExpansion(C, G);
Return BH_{t+1};
```

第七章 实验评估

7.1 数据

本文在验证过程中主要使用自行车出行数据。该数据由曼哈顿 Citibike 共享系统提供；该数据包含 2013 年 10 月中 6376 辆自行车和 330 个站点产生的 1037712 次出行。每次自行车出行都有开始/停止时间和开始/结束站点，它们组成曼哈顿自行车出行的 STG。由于纽约人每天约有 113000 次自行车出行，自行车出行数据部分反映了纽约市的交通流量。

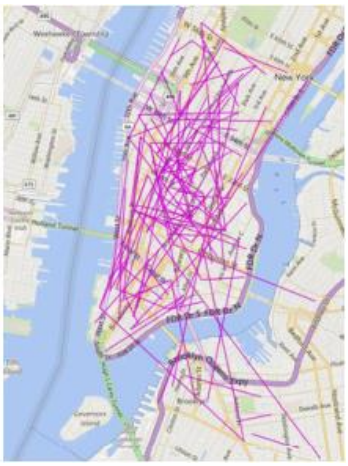


图 8 曼哈顿 STG

7.2 曼哈顿样例分析

由于人们通常将自行车租用或归还到附近的自行车站，因此有必要探测一个区域内几个自行车站所形成的黑洞/火山。图 9 为 2013 年 10 月 8 日和 10 月 17 日 16:00 - 20:00 在纽约曼哈顿探测到的黑洞和火山。例如，8 日 17 时~ 18 时位于华尔街附近的火山(C)和 17 日 17 时~ 18 时位于时代广场附近的黑洞(A)。值得注意的是，在所有时间间隔中，17 时~ 18 时期间黑洞和火山的数量最多，这是因为当前时间间隔为在工作日的高峰时段，交通流量是最高的。

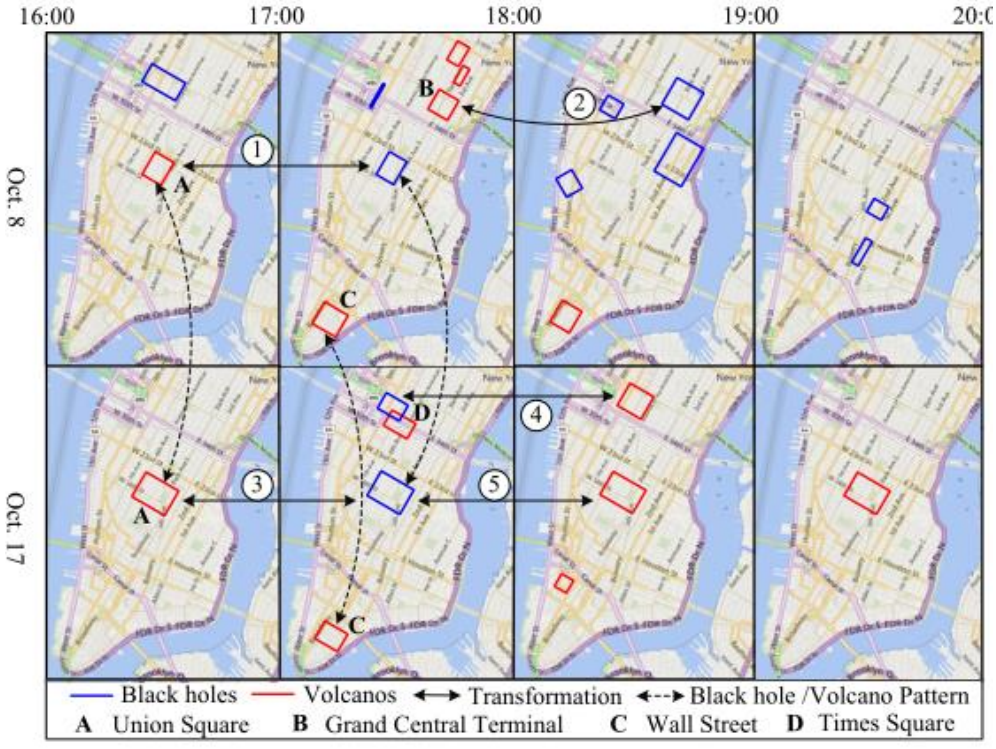


图 9 纽约市内的黑洞/火山

7.2.1 异常事件

如图 9 所示，10 月 17 日 17:00 ~ 18:00，联合广场附近的火山变成了一个黑洞(③)，之后的两个时间间隔变成了两个火山(⑤)。结果与当天 8 时到 11 时，联合广场绿色市场事件一致。在绿色市场形成两座火山后，人们骑着自行车离开联合广场。

表 1 联合广场(A)的入流和出流

Time Interval	Station ID	Inflow	Outflow
16:00-17:00 Oct. 17 Volcano	382	24	42
	253	15	20
	497	24	40
	285	12	0
	Total:	**75**	**102**
17:00-18:00 Oct. 17 Black Hole	253	14	9
	382	37	22
	497	56	50
	285	19	23
	Total:	**126**	**104**
18:00-19:00 Oct. 17 Volcano	382	45	63
	253	22	25
	497	53	59
	285	19	23
	Total:	**139**	**170**
19:00-20:00 Oct. 17 Volcano	382	30	36
	497	28	32
	253	8	9
	285	22	21
	Total:	**88**	**98**

表 1 记录了 10 月 17 日 16 时到 20 时， A 附近自行车站点的进出情况。如，在 16 时到 17 时期间，382 站和 497 站的实际出车人数分别为 18 人和 16 人。

根据黑洞和火山的实时情况，Citibike 可以在 16:00 ~ 17:00 和 18:00 ~ 20:00 期间临时使用卡车运送自行车到 382 站和 497 站，在 17:00 ~ 18:00 期间从 253 站、382 站和 497 站运送自行车。

检测此类异常事件在交通控制、事件检测中有许多广泛的应用。例如，在图 9 的S_1中有 56 条边，如果α设置为 0.7 时，其道路网络容量为 998，可用于确定其交通是否存在拥堵。如果发现S_1的实际流量接近道路网络容量 998，运输当局可启动交通控制以调节交通。同时，有关黑洞的相关信息也可以显示在屏幕上，推送到司机的手机上，已通知司机在即将到来的交通堵塞之前选择其他线路。

如图 9 中，S_2和S_3有 24 条重叠边，因此在检测过程中，通过将S_2拓展到S_3，本文所设计的连续检测算法至少可以节省 24 个边的访问。

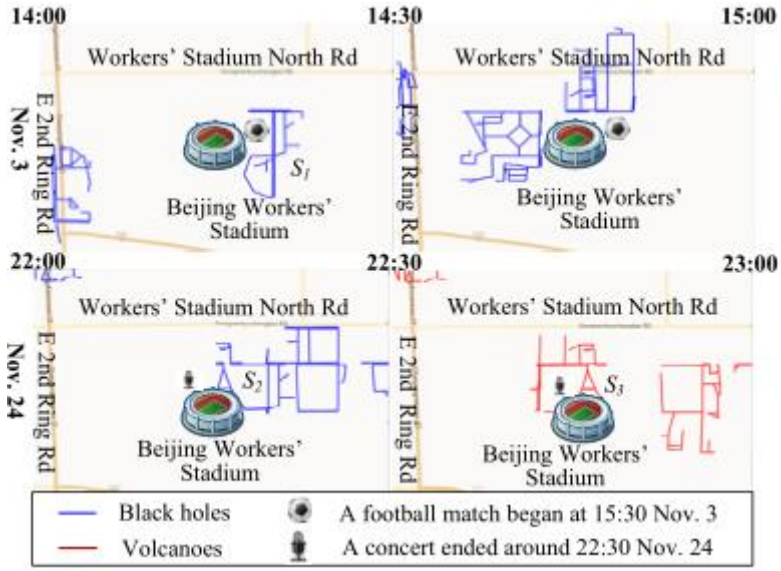


图 10 工人体育场周边的黑洞/火山

7.2.2 黑洞/火山的常规模式

如图 9 所示，2013 年 10 月 8 日和 10 月 17 日，分别在 16:00 ~ 17:00 和 17:00 ~ 18:00，在联合广场(A)附近有规律地出现了火山和黑洞。

火山可能是由来自著名购物街联合广场附近第五大道的购物者形成的，而黑洞则是由附近商业办公室的下班人员形成的。8 日和 17 日下午 17:00 ~ 18:00，从华尔街下班的员工(C)也形成了黑洞。值得注意的是，这些模式通常超出了本地人的常识范围。

基于上述模式，Citibike 可以定期在 16:00 ~ 17:00 将自行车送到联合广场附近的站点，并在 17:00 ~ 18:00 从这些站点移走自行车。Citibike 甚至可以优化这些站点的设计，以缓解自行车停靠位或自行车的短缺。例如，253 站、285 站、382 站和 497 站应该同时增加自行车停靠位和自行车，因为这些站在不同时间间隔的实际流入、流出量都很大，如表 1 所示。但是，华尔街周围的车站只需要增加自行车，因为这里只有在高峰时段才会形成了火山。从长远来看，这些黑洞/火山模式有助于 Citibike 选择自行车站点的选址，从而提高共享单车系统的运营效率。直观地说，Citibike 应该在黑洞和火山频繁发生的地方设置更多的站点。

7.2.3 黑洞和火山之间的联系

如图 9 所示，2013 年 10 月 8 日和 10 月 17 日，共有 3 个火山转变为黑洞，2 个黑洞转变为火山。例如，10 月 8 日 17:00 ~ 18:00 在中央车站(B)附近的火山在下一个时间间隔转变为黑洞(②)。这种转变很有可能是骑自行车的人进出火车站造成的。

我们还发现，许多通勤者在高峰时段离开华尔街前往中央车站(Grand Central Terminal)，这是一条热门的自行车路线。另外，17 日 17 时~ 18 时，时代广场附近的黑洞(D)在 18 时~ 19 时变成了火山。

黑洞/火山之间的关系有助于更好地了解城市动态，从而帮助纽约市规划部门设计自行车道(例如在中央车站和华尔街之间)，更好地缓解纽约市的交通拥堵。

7.3 Storm 平台搭建

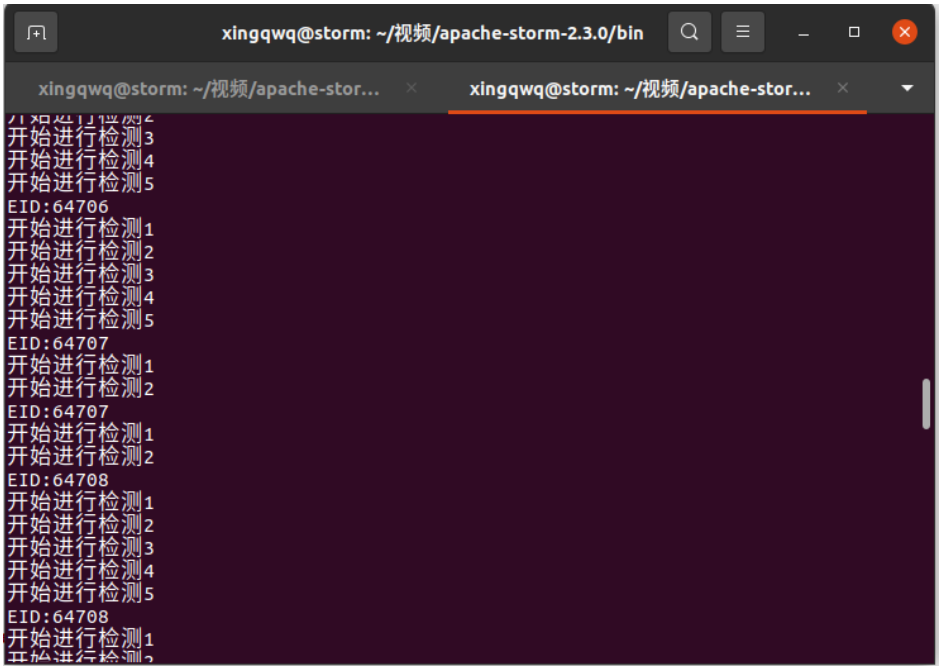


图 11 storm 本地模式测试

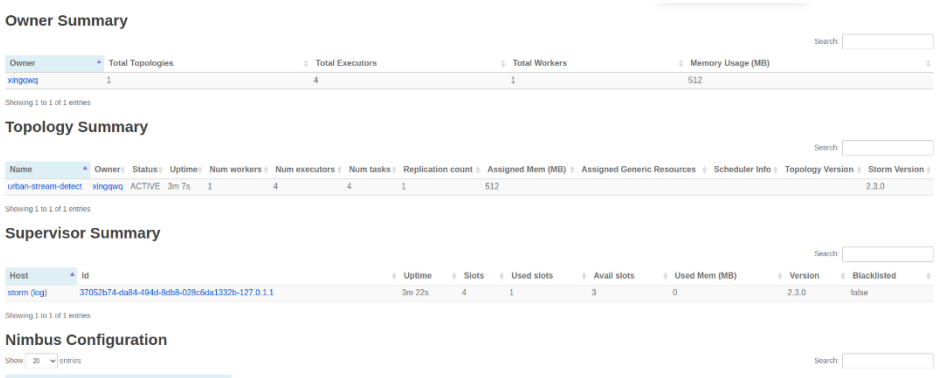


图 12 storm 集群模式测试

第九章 结论

本文在验证过程中主要使用自行车出行数据，该数据由曼哈顿 Citibike 共享系统提供；本文所构建的城市流量黑洞检测系统基于 Storm 的分布式计算架构，将城市 STG 图划分为多个网格单元，将相邻近的网格单元汇聚成处理组，每个处理组对应单独的 Bolt 进行实时分析检测。系统采用 TCP 与数据提供方进行通信，在经过对数据的处理后分发到对应的 Bolt 中。该架构有利于控制算力的具体投入，更精准地监测某些具有阶段性流量特征的区域。

参考文献

[1] Liang H , Zheng Y , Yung D , et al. Detecting Urban Black Holes Based on Human Mobility Data[C]// Sigspatial International Conference on Advances in Geographic Information Systems. ACM, 2015.

[2]Deng, Liu, Luo. Detecting Urban Polycentric Structure from POI Data[J]. International Journal of Geo-Information, 2019, 8(6):283.

[3]U. Brandes, D. Delling, M. Gaertler, R. Gorke, M. Hoefer, Z. Nikoloski, and D. Wagner. On modularity clustering. IEEE Trans. on Knowledge and Data Engineering, 20(2):172–188, 2008.

[4] X. Chen, Y. Liu, H. Liu, and J. G. Carbonell. Learning spatial -temporal varying graphs with applications to climate data analysis. In Proc. of AAAI, 2010.

[5] M. Hadjieleftheriou, G. Kollios, D. Gunopulos, and V. J. Tsotras. On-line discovery of dense areas in spatio-temporal databses. In Proc.of SSTD, 2003.

[6] C. S. Jensen, D. Lin, B. C. Ooi, and R. Zhang. Effective density queries on continuously moving objects. In Proc. of ICDE, 2006.

[7] H. Jeung, M. L. Yiu, X. Zhou, C. S. Jensen, and H. T. Shen. Discovery of convoys in trajectory databases. PVLDB, 1(1):1068–1080, 2008.

[8] M. Lahiri and T. Y. Berger-Wolf. Periodic subgraph mining in dynamic networks. Knowledge and Information Systems, 24(3), 2010.

[9] T. Lappas, M. R. Vieira, D. Gunopulos, and V. J. Tsotras. On the spatiotemporal burstiness of terms. PVLDB, 5(9):836–847, 2012.

[10] Z. Li, B. Ding, J. Han, and R. Kays. Swarm: Mining relaxed temporal moving

object clusters. PVLDB, 3(1-2):723–734, 2010.
[11] Z. Li, H. Xiong, and Y. Liu. Mining blackhole and volcano patterns in directed graphs: A general approach. Data Mining and Knowledge Discovery, 2012(25), 2012.
[12] Z. Li, H. Xiong, Y. Liu, and A. Zhou. Detecting blackhole and volcano patterns in directed networks. In Proc. of ICDM, 2010. [11] Z. Liu, J. X. Yu, Y. Ke, and X. Lin. Spotting significant changing subgraphs in evolving graphs. In Proc. of ICDM, 2008.
[13] M. Mathioudakis, N. Bansal, and N. Koudas. Identifying, attributing and describing spatial bursts. PVLDB, 3(1-2):1091–1102, 2010.
[14] B. Pan, Y. Zheng, D. Wilkie, and C. Shahabi. Crowd sensing of traffic anomalies based on human mobility and social media. In Proc.of GIS, 2013.
[15] C. Ren, E. Lo, B. Kao, X. Zhu, and R. Cheng. On querying historical evolving graph sequences. In PVLDB, 2011.
[16] C. Robardet. Constraint-based pattern mining in dynamic graphs. In Proc. of ICDM, 2009