Flow Computation in Temporal Interaction Network

摘要

时间交互网络记录实体间的活动历史，每一次交互的时候，会有一定量的数据从网络的一个顶点流向另一个顶点，基于流的分析可以揭示重要的信息。

本文介绍了交互网络或子图中的流量计算问题。

提出并研究了两种流量计算模型

- **基于贪婪的流量传递假设**
- **找到最大可行流量**

对于第一个模型，可以简单的按照时间顺序对交互进行扫描即可

对于第二个最大流量问题，使用了图形预处理和简化计算的方法，大大降低其复杂度。

本文目标是在使用技术大大降低在*大型交互网络中查找其实例及其流*的成本。

基本概念

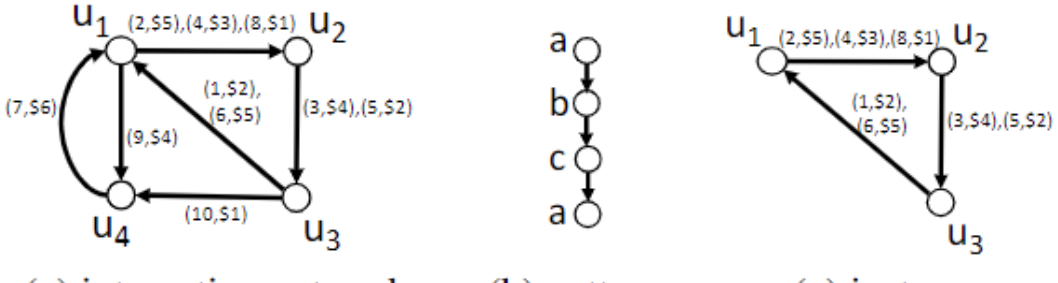
交互网络： 一个有向图 $G(V,E)$，对于每一条边 (v,u)，其存在一个序列 $S=\{(t1,q1),(t2,q2),\dots\}$，每一个交互 (ti,qi) 有 qi 的,量从 v 到 u 在时间戳 ti.

网络模式： 一个有向无环图，$G_p(V_P,E_P)$，每一个点 $v\in V_P$

实例： 与模式相对应，

DEFINITION 3 (INSTANCE). *An instance of pattern G_P in graph G is a subgraph $G_M(V_M,E_M)$ of G_T, such that*

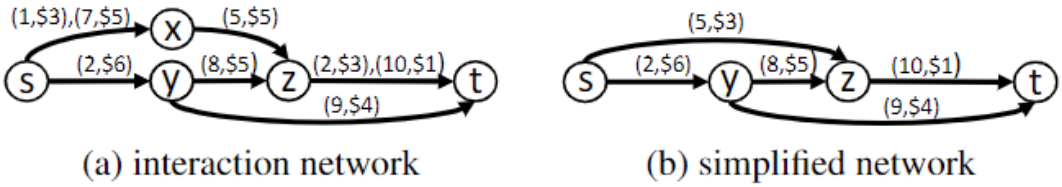
- *there is a surjection $\mu: V_P \to V_M$ from the vertex set V_P of the pattern G_P to the vertex set V_M of G_M;*
- *for two vertices v,u of G_P, $\mu(v)=\mu(u)$ iff $\ell(v)=\ell(u)$;*
- *$(v,u)\in E_P$ iff $(\mu(v),\mu(u))\in E_M$.*



(a) interaction network (b) pattern (c) instance

介绍

问题



(a) interaction network (b) simplified network

Figure 1: A toy interaction network

指定观察两个顶点之间流量，S、T，对于每一个点有一个 $Buffer$，表示当前的点有的流量大小

第一种模型：每条边的流量，$min\{q_i, B_v\}$

第二种模型：每条边的流量，$[0, min\{q_i, B_v\}]$

贪心有时候得不到最大流

应用

金融资金流向，交通人口流动，通信网络异常流量

贡献

贪婪流量计算较为简单，重点为最大流量的计算

使用线性规划来解决最大流量问题

提出方法来解决：

- **判断某些类网络，使用贪婪算法可以得到最大流量（验证需要遍历一遍顶点）**
- **使用预处理算法，消除不会影响最大流量的边和顶点，有可能极大的减少问题的复杂性**
- **设计算法，对图的一部分进行贪婪最大流计算，从而简化了必须使用线性规划的图**

在计算中还制定和研究大型交互网络中的流模式搜索问题，问题是找到实例并未每个实例计算流量。提出**图形预处理**的方法。

使用了3个真实的时间交互网络，来评估实验技术

对于静态图和时间图的流计算、模式枚举在大型网络中

静态图最大流：FF、EK、Dinic、ISAP、HLPP（分为增广路算法、预留推进）

时间图最大流：不考虑边存在流量，而是考虑边具有与流的瞬时相互作用的序列，这些序列发生在特定的时间戳记上。考虑交互是临时的，边也是临时的

本文流模式的枚举特点：

- 流量计算模型于以前的工作不同，因为考虑了最大流量计算，并且还允许时间交错的交互序列
- 研究的模式不限于简单路径
- 提出了用于模式枚举的预计算方法

最大流计算模型和算法

- **贪婪流计算：** 作为交互作用的结果，节点通过交互作用从其缓冲量中转移得尽可能多。时间复杂度分析：只需要遍历图的点和边

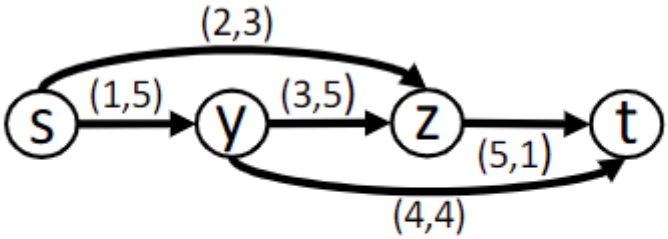


Figure 3: Example of a graph

Table 2: Example of greedy flow computation

(t_i, q_i)	(v, u)	B_s	B_y	B_z	B_t
$(1, 5)$	(s, y)	∞	5	0	0
$(2, 3)$	(s, z)	∞	5	3	0
$(3, 5)$	(y, z)	∞	0	8	0
$(4, 4)$	(y, t)	∞	0	8	0
$(5, 1)$	(z, t)	∞	0	7	1

- **最大流计算：**
  - **LP（线性规划）**

$$0 \leq x_i \leq q_i \tag{1}$$

$$x_i \leq \sum_{dest_j = src_i \wedge t_j < t_i} x_j - \sum_{src_j = src_i \wedge t_j < t_i} x_j \tag{2}$$

$$\text{maximize} \sum_{dest_i = sink(G)} x_i \tag{3}$$

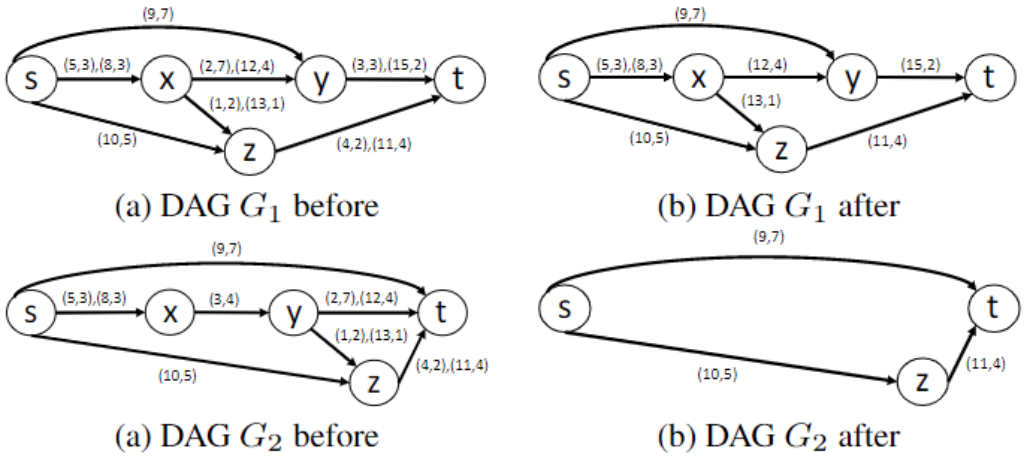
有一个约束，即在边（src_i，$dest_i$）上的交互作用（t_i，q_i）不能将总传入单元转移到从 src_i 减去总传出单元，直到时间戳记 t_i

LP问题的目的是找到所有变量 x_i 的值，这将流最大化

计算时间交互网络上的最大流量（即我们的问题）对边上的交互数具有二次成本

  - **充分使用贪婪解决最大流问题：**对于两张图可以使用：1、对于一个链，是可以使用贪婪算法的；2、除了源和汇点之外的所有顶点都只有一个出边，那么也是可以使用贪婪算法的

- **图的预处理：**

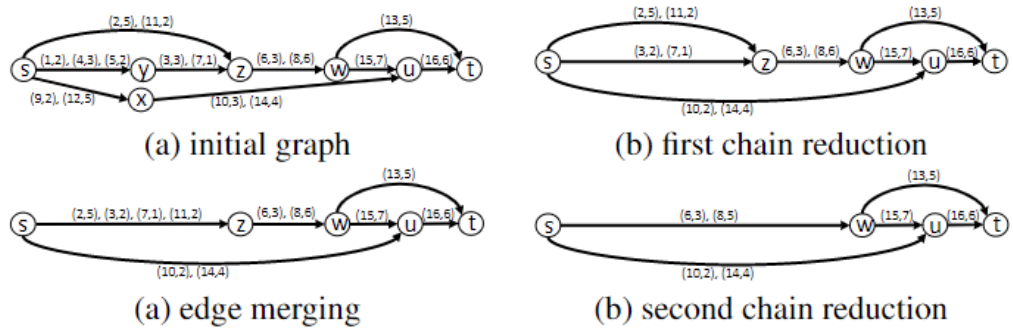


(a) DAG G_1 before　(b) DAG G_1 after

(a) DAG G_2 before　(b) DAG G_2 after

Figure 6: DAG preprocessing examples

- **图的简化：**



(a) initial graph　(b) first chain reduction

(a) edge merging　(b) second chain reduction

Figure 7: Example of graph simplification

流模式研究

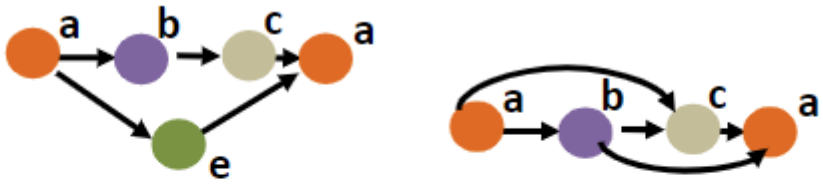
graph browsing （GB 图浏览）

以模式的拓扑序进行构建实例，对于一个一个顶点进行逐步匹配，直到匹配失败。

优点：一个通用的方法，不需要预先计算的信息

preprocessing-based（PB 图预处理）

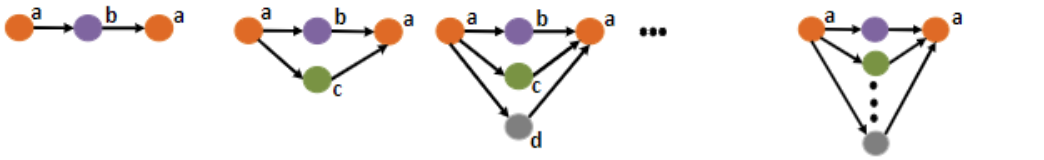
- **路径预计算：** 预先计算的子图是达到一定长度的路径（即最多k跳）。 每个长度形成一张表，其中包含该长度的所有路径。 也就是说，对于每条路径存储：
  - 形成该路径的顶点的序列，
  - 应用贪婪算法后，进入路径接收器缓冲区B的交互序列
- **模式实例的枚举：** 在此过程中，尽可能使用表中的任何预先计算出的流来避免进行流计算，例如，考虑图 8（a） 中所示的流模式，已经过预处理并且可以使用两跳和三跳的所有实例。 从两个表 $L2$ 和 $L3$ 的同一节点开始和结束的循环路径。 在这种情况下，我们只需访问和使用预处理数据就可以轻松计算 G_P 的所有实例。最后，为了计算生成的模式实例的总流量，我们将所有预先计算的传入流量汇总到两条路径的汇点。另一方面，例如图8（b）中的实例时，预先计算的数据可能无法完全利用。 在这种情况下，无法使用L3中的路径的预先计算的流，因为路径在模式实例中不是孤立的。通常，沿着路径的预先计算的流仅对那些路径是独立的且可以使用在本文档中提出的技术逐步简化的模式实例才有用。



(a) easy pattern (b) hard pattern

Figure 8: Examples of flow patterns

- **非刚性模式：** 刚性模式为DAG，非刚性存在换，如图9（a）所示。 然后，我们可以聚合对应于同一节点的不同模式的所有实例的流量，以计算来自其他节点的总流量。这种方法有几个缺点。 首先，我们必须计算并合并多个模式查询的结果。 第二，我们应该使用多少个模式没有限制。 第三，最终结果可能不正确，因为子模式的流程可能包含在超级模式的流程中（例如，图9（a）中第二模式的实例包括第一模式的两个实例），使用我们的预计算方法，找到此模式的实例并测量其流量非常容易，因为我们只需要扫描2跳周期表L2。



(a) 2-hop rigid patterns (b) relaxed pattern

Figure 9: 2-hop nonrigid pattern

实验结果

Table 4: Characteristics of Datasets

Dataset	#nodes	#edges	#interactions	avg. flow
Bitcoin	12M	27.7M	45.5M	34.4฿
CTU-13	607K	697K	2.8M	19.2KB
Prosper Loans	88K	3M	3.04M	$76

三个数据集（比特币交易，网络流量、贷款服务资金流动）

Table 5: Statistics of subgraphs

Dataset	#subgraphs	avg #vertices	avg #edges	avg #interactions
Bitcoin	48.7K	5.16	6.42	448.4
CTU-13	9235	3.24	2.49	15.9
Prosper Loans	137	6.1	8	611.5

A类：包含最简单的子图，B类：包含一般的子图，C类：包含最难的图

Table 6: Runtime (msec) for Bitcoin subgraphs

	Greedy	LP	Pre	PreSim
All (48.7K)	0.0491	5775	838.8	524.5
Class A (35.4K)	0.0074	2667.18	0.0078	0.0078
Class B (7891)	0.295	7179.39	0.575	0.575
Class C (5366)	0.353	24248	7615.8	4762.43

Table 7: Runtime (msec) for CTU-13 subgraphs

	Greedy	LP	Pre	PreSim
All (9235)	0.0035	10.313	6.314	0.7902
Class A (9199)	0.0032	3.835	0.0033	0.0033
Class B (3)	0.0037	71.07	0.0074	0.0074
Class C (33)	0.0757	1810.38	1767.5	220.2

Table 8: Runtime (msec) for Prosper Loans subgraphs

	Greedy	LP	Pre	PreSim
All (137)	0.0027	0.5105	0.0352	0.0157
Class A (94)	0.0015	0.5072	0.0016	0.0016
Class B (25)	0.004	0.5646	0.008	0.008
Class C (18)	0.0067	0.4527	0.2373	0.0889

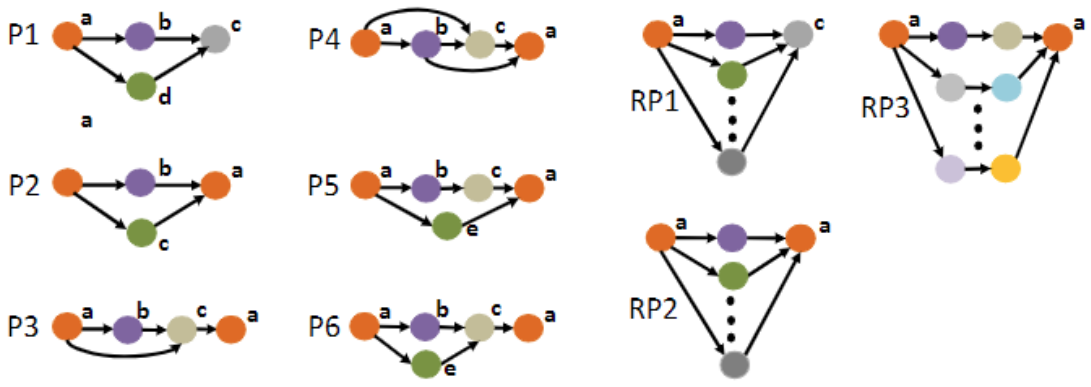


Figure 12: Set of tested patterns

Table 9: Pattern Search on Bitcoin

Pattern	Instances	Average flow	GB	PB
P2	22.3G	56.15	23.2 hours	30.59 sec
P3	2.8M	4786.18	3155.96 sec	179.70 sec
P4*	3000	697.04	446.73 sec	421.85 sec
P5	577.5M	8069.2	15 days (est.)	179.74 sec
P6*	2.04T	2.81	1445 sec	1059 sec
RP2	655K	39.86	422.79 sec	53.273 msec
RP3	1.2M	1.86	306 min	13.53 msec

（使用P4和P6对于GB算法和PB算法的效率区别不大：无法预处理流，并且实例的最大流必须使用线性规划计算，实例包含大量的交互，最大流计算主导总体成本）

Table 10: Pattern Search on CTU-13

Pattern	Instances	Average flow	GB	PB
P2	709M	2888.90	1952.61 sec	762.65 msec
P3	182	528.5K	55.71 sec	8.61 msec
P4	91	1.56M	58.564 sec	2.518 sec
P5	208K	13116.5	443.97 sec	4.73 msec
P6	586	52892	410.4 sec	14.87 msec
RP2	51266	11942.65	24.15 sec	0.63 msec
RP3	91	61485.58	375.39 sec	0.035 msec

Table 11: Pattern Search on Prosper Loans

Pattern	Instances	Average flow	GB	PB
P1	5.12M	45.89	119.08 sec	2.80 sec
P2	201	223.23	88.66 msec	0.004 msec
P3	268	100.44	3.57 sec	1.3 msec
P4	98	299.55	3.54 sec	0.723 msec
P5	1833	121.47	605.67 msec	0.021 msec
P6	1296	43.55	474.61 msec	11.13 msec
RP1	25.5M	25.12	133.37 sec	3.01 sec
RP2	260	58.061	0.016 msec	0.004 msec
RP3	532	10.94	503.89 msec	0.040 msec

总结和未来研究方向

该论文聚焦于时间窗口的交互

论文研究在时间交互网络中的流计算问题

提出了两个模型（目标：计算最大可行流）：

- 基于贪心的流量传递
- 任意的流量传递

第一种模型更有效率。

论文采用一系列的技术，将最大流计算问题的复杂度下降一个数量级

还研究了巨大图的模式枚举问题

对于每一个子图模式，都进行计算最大流

使用预计算简单子图实体和它们的流。

通过这些预计算加速加快发现以这些子图为组成部分的更复杂模式

未来工作的方向

- 研究额外的技术减少计算最大流量问题的成本；
- 研究类似的简化技术用于其他计算流问题；
- 自动识别感兴趣的模式和一些有显著超过预期流量的子图