EpCom: A Parallel Community Detection Approach for Epidemic Diffusion over Social Networks

Heng Zhang*‡, Libo Zhang*, Da Cheng†‡, Yanjun Wu*, Chen Zhao*
*Institute of Software, Chinese Academy of Sciences, No. 4 Zhongguancun South 4th Street, Beijing 100190, PR China
†Institute of Botany, Chinese Academy of Sciences, 20 Nanxincun, Xiangshan, Beijing, 100093, PR China
‡University of Chinese Academy of Sciences, No. 19 Yuquanlu, Beijing 100049, PR China
zhangheng@nfs.iscas.ac.cn, zmsj@hotmail.com, chengda@ibcas.ac.cn, {yanjun, zhaochen}@iscas.ac.cn

***Abstract*—Detecting community structure in epidemics networks is crucial for the assessment of epidemic dynamics and effective control of disease spread by targeting at the individuals bridging communities. Common community detection models (e.g., cut-criteria and modularity-criteria based model) are efficient in optimal quality of network partitions. However, most of the approaches fail to consider the dynamic infected possibility in person-to-person interactions. In addition, they present high computational complexity, which was limited by the scale of networks and the performance of hardware platform. In this paper, we propose a Jaccard distance based community detection model by considering both the quality of network partitions and the dynamics of infected interacts (i.e., edges) between two individuals in epidemic diffusion. Then, we design a novel parallel approach based on the high parallism of GPU, called `EpCom`, for boosting the performance and scalability of parallel community detection over large-scale epidemic networks. From the evaluation results, the proposed GPU-based implementation `EpCom` exhibits great performance and achieves maximum 604 million TEPS (traversed edges per second), which corresponds to up to 54.2 times and 15.6 times than CPU-based *NCut* and *Louvain* approaches separately.**

I. Introduction

With an exponentially growing number of urbanized and moving population, the likelihood of a worldwide pandemic is increasing sharply. Since the epidemics (e.g., influenza, HIV/AIDS, SARS, Ebola, etc.) are occurrences via person-to-person transmission with characteristics of spreading fast and involving widely, understanding their diffusion behavior in populations is key to controlling them. Traditionally, mathematical and computational simulations of dynamics of epidemics [1]–[3] have provided a valuable equation tool for completely mixing populations, which are represented as *networks* or *graphs*. In such epidemic networks, persons and their interactions among each other are represented by nodes and edges. The effective methodologies to discover the propagation rules and to mitigate or prevent the spreading of infectious disease in the epidemic networks have been a hot topic in the complex network literature.

In these methodologies, most works [4], [5] have been done towards the directions to identify top-ranked influential spreaders for targeted immunization via *vertex-centric paradigms*, such as degree centrality or betweenness centrality. Although degree distributions and centrality are studied extensively in effects on epidemic dynamics, they would not be suitable for growing scale of networks, due to require a global knowledge of the whole network. On the other hand, detecting community structure in the spread of disease in epidemics networks (e.g., CBF algorithm [6], community level influence measures [7], [8]) have shown great potential in understanding the epidemic dynamics and controlling the spreading by *cutting inter-community edges*. They are based on a fact that *human contact networks exhibit strong community structure* [9]–[11], and eventually demonstrated community structure strongly affects disease dynamics. Compared to the targeting of high-centrality individuals in epidemic networks using a global strategy, the community-based methods are more effective for immunization interventions targeted at the individuals bridging communities. For example, when the community structures are cohesive, the epidemic can be trapped in community by cutting only a few inter-community links. Meanwhile, the effort to identify which communities have been contracted can be avoided by monitoring the bridging individuals instead of all members in these communities, and early community detection is important for limiting new infections and taking early care of the exposed individuals.

Within the state-of-the-art community detection approaches, most common community detections introduce *cut-criteria* based model in epidemic networks [12], [13]. Communities are partitioned as disjointed subgraphs with minimized number of cut edges. Another mainstream model is based on *modularity-criteria* [14], in which incremental optimal quality of network partitions are obtained via iteratively adjusting expected cut edges. However, the challenges of applying these models in epidemic networks lie in both models' limitations and high computational complexity on large-scale datasets: 1) fails to consider the dynamic infected possibility in person-to-person interactions and ignores to detect small-sized communities due to resolution limit; 2) presents high computational complexity, which was limited by the scale of networks. Moreover, enhancing performance in the growing large-scale networks is posing parallel detection technical challenge using limited size of resources, for example, analyzing epidemic network dataset which exceeds the GPU-resident memory size.

In this paper, we propose a novel parallel community detection approach for the simulated epidemic diffusion. First, based on the introduction of *Jaccard distance* between individuals [15], we present a dynamic distance view in epidemic network, in which a weight of interaction is associated with the dynamic interacts between two persons for the infected chance of transmitting a disease. Instead of the global fashions in *cut* and *modularity* based methods, we apply three local interaction patterns to determine the affects among individuals, and identify the communities via local dynamic interactions. Second, we aim to boost the performance and scalability of parallel community detection over large-scale epidemic networks. Consider that GPU (Graphics Processing Unit) platform provides large network analytic not only massive parallelism (approximately 10Ks threads) but also efficient memory I/O (up to 100GB/s memory bandwidth) [16]. We design and implement a novel GPU-based approach, namely `EpCom`, by utilizing the high parallelism of GPU platform. Specifically, due to realistic networks tend to be skewed and highly compressible, `EpCom` designs an optimal "shard" graph partition for epidemic network representation which provides a similar compression ratio and also omits the overhead from decoding edge set. Then, `EpCom` presents parallel asynchronized processing ability via exploiting *CUDA Stream Object* feature to enhance performance further. To the best of our knowledge, `EpCom` is the first approach to exploit I/O efficiency in limited GPU-resident memory to support GPU-accelerated community detection over out-of-core networks, the size of which has exceeded the GPU-resident memory.

The main contributions of this paper is as follows:

- A community-based early detection model based on Jaccard distance definition is presented by considering three affect pattern between two individuals. In this regrad, the problem of community detection in epidemic diffusion is formulated.
- An efficient parallel GPU-based approach, called `EpCom`, is proposed for community detection. Two key optimization strategies designed in `EpCom` are outlined: 1) design a shard-based graph representation to support community detection in billion-scale networks using limited GPU-resident memory size; 2) exploit GPU-based asynchronized data movement and computation to enhance parallel procedure performance.
- Experiments are conducted on the realistic social network datasets to evaluate the effectiveness and efficiency of `EpCom`. The results show that distance-based model allows effective community detection in epidemic networks and `EpiCom` significantly outperforms other competitors as well as representative parallel methods.

The rest of paper is orgnized as follows. Section II introduces the background and challenges of community detection in epidemic diffusion. Section III presents the Jaccard distance based community detection model. The detailed design and implementation of our proposed approach `EpCom` is discussed in Section IV-C. We report our experimental results in Section V. Section VI concludes the paper.

II. Challenges and Motivation

Early detection for the epidemic diffusion is important for taking early care of the individuals in exposed and limiting new infections. To reduce the unaffordable running time, simulation-based techniques have been employed to accelerate the visualization of SER (susceptible, infectious, removed) based spreading of contagious disease [2], [8]. Whereas the approaches generally lack a comprehensive consideration on community-based methodology for achieving the epidemic early detection. By introducing constraining policies in communities, the objective is to give more early targeted care to infected communities and to limit disease in much smaller areas progressively. Along with community structure based epidemic diffusion, We mainly focus on addressing the following two challenges of the state-of-art approaches.

1) Challenge of flexible community detection. The two most common computational methodologies for community detection introduce *cut-criteria and modularity-criteria* based model in epidemic networks. They seek to partition a graph into disjoint subgraphs with minimized number of cut edges across subgraphs and measure the quality of network divisions [12]–[14], [17]. The methodologies yield network partitions using a global fashion following the assumption that each node has the equal chance to link any other node of the network. However, applying such methodologies in epidemic diffusion analysis fails to consider the dynamic infected possibility in person-to-person interactions. Moreover, since inherent "resolution limit" in such models, the above two models cannot detect many small-sized communities. And enhancing performance in the growing large-scale networks is posing parallel detection challenge, in which communities need to be detected in local.

2) Challenge of realistic social networks. Community detection is a NP-hard problem, which was confirmed in [18]. The non-uniform nature of realistic epidemic networks represents a rapidly growing large scale and an irregular distribution of heavy-tailed degree. The quality of community detection in large networks depends on many factors such as the amount of input graph data, the time complexity of algorithm and parallel scalability. As the size and complexity of networks increase, faster community detection implies larger amount of graph data in a given runtime. Therefore, an efficient parallel processing strategy needs to be designed by considering both the quality and the performance of scaling methodology to parallel computing environment. Moreover, considering the limited memory resource for residing datasets, when scaling community detection to parallel environment over these networks, designing an optimal data partition and distribution scheme is key to the scalability of approaches.

III. Distance-based Community Detection Model for Epidemic Diffusion

Epidemic diffussion have been performed on complex networks [2], which consist of spartially-explicit agents (individuals) with interactions on household and community scale. The most common epidemic diffusion model is introduced as a *susceptible*, *infectious*, *removed* (SIR) model. As shown in Figure 1(a), with periodic boundary conditions, indiduals in SIR model can be in one of the following states:

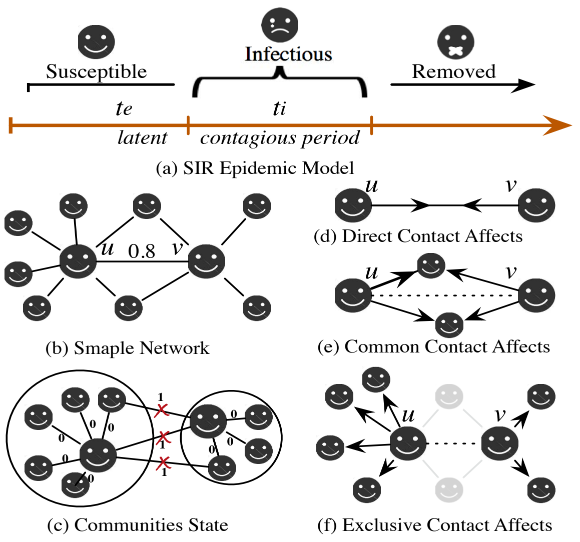


Fig. 1. SIR model, community structures in a sample network and the three contact patterns in `EpCom`.

- *Susceptible:* disease-free and never previously infected.
- *Infectious:* infected with symptoms.
- *Removed:* recovered or dead.

We format SIR model for spreading of contagious disease in complex networks as following: each individual in the complex networks represents a node, and the interactions between two nodes represent an edges. Considering an infected possible rate γ during a given interaction, whose value depends on several factors (e.g., interaction number, spreading rate, etc.), we give each edge a weight value to define a synthetical infected rate between two individuals. For example, in Figure 1(b), there are several interactions between two individuals u and v, the edge (u, v) is assigned and associated with a weight $w(u, v) = 0.8$.

For the sake of efficient community-based early detection in epidemic diffusion analysis, we introduce a dynamic distance-based model over realistic social networks [15].

First, we format several necessary definitions for the presentation of our community detection methodology.

Definition 3.1 **Epidemic Network (Graph)**: Let $\widehat{G} = (\widehat{V}, \widehat{E}, \widehat{W})$ be an undirected weighted epidemic network, where $\widehat{V}$ and $\widehat{E}$ is the set of nodes (individuals) and edges (interactions) and the $\widehat{W}$ is the corresponding set of infected possibility (weights) in $\widehat{G}$. Each edge $\hat{e} \in \widehat{E}$ is expressed as (n_s, n_e, w), where $n_s, n_e \in \widehat{V}$ and $w(n_s, n_e)$ is a weight assigned to $\hat{e}$ (e.g., an infected possibility after n_s interacts n_e). We define the number of nodes in $\widehat{G}$ as $n = |\widehat{V}|$, the number of edges in $\widehat{G}$ as $m = |\widehat{E}|$.

Definition 3.2 **Contacts (Interactions or Edges)**: Given a node n_s, the contact set of node n_s is defined as $\Gamma(n_s, \widehat{G}) = \{n_e | (n_s, n_e, w) \in \widehat{E}\} \cup n_s$. The number of contacts of one node n_s is defined as the degree of n_s, i.e., $deg(n_s, \widehat{G}) = |\Gamma(n_s, \widehat{G})|$.

Then, we introduce a *Jaccard distance dynamics* based community property for detection, which is proposed in [19], The definition of Jaccard dis shown in the following definition:

Definition 3.3 **Jaccard Distance**: Given an undirected network $\widehat{G} = (\widehat{V}, \widehat{E}, \widehat{W})$ and two nodes n_s, n_e. The distance value between two individuals is defined as $d(n_s, n_e) = 1 - \frac{\Gamma(n_s) \cap \Gamma(n_e)}{\Gamma(n_s) \cup \Gamma(n_e)}$. Given the weights to whole edge set in $\widehat{G}$, the Jaccard distance for n_s, n_e is further defined as:

$$d(n_s, n_e) = 1 - \frac{\sum_{x \in \Gamma(n_s) \cap \Gamma(n_e)} (w(n_s, x) + w(n_e, x))}{\sum_{\{x,y\} \in \widehat{E}; x,y \in \Gamma(n_s) \cup \Gamma(n_e)} w(x, y)} \quad (1)$$

Three diffusion patterns for person-to-person interactions are considered as following:

Pattern I **(Affects from Directed Contacts.)** Through mutable interactions, the direct linked node n_s and n_e is obviously all influenced (see Figure 1(d)). Thus, their cohesiveness tends to be increased gradually. Based on the definition of Jaccard distance between n_s and n_e, the affects from interactions of directed linked nodes is defined as: $I_{DC} = -(\frac{1}{deg(n_s)} \cdot sin(1 - d(n_s, n_e)) + \frac{1}{deg(n_e)} \cdot sin(1 - d(n_s, n_e)))$. Here the $deg(\cdot)$ term indicates the degree of nodes and $sin(\cdot)$ math function is used for coupling operator.

Pattern II **(Affects from Common Contacts.)** The second affect is from the common contacts of the two nodes n_s and n_e (see Figure 1(e)), where the common contacts is defined $\Gamma(n_s) \cap \Gamma(n_e)$. Considering the common contacts have both directed links to n_s and n_e, they interact with n_s and n_e and thus result change their cohesiveness. We define the affects from common contacted individuals as $I_{CC} = -\sum_{x \in \Gamma(n_s) \cap \Gamma(n_e)} (\frac{1}{deg(n_s)} \cdot sin(1 - d(x, n_s)) \cdot (1 - d(x, n_e)) + \frac{1}{deg(n_e)} \cdot sin(1 - d(x, n_e)) \cdot (1 - d(x, n_s)))$. Here we quantify the affects from each common contact with its relative distance (e.g., $(1 - d(x, n_s))$).

Pattern III **(Affects from Exclusive Contacts.)** The third affect (see Figure 1(f)) happens when the two nodes n_s and n_e have their exclusive contacts, i.e., $(\Gamma(n_s) - \Gamma(n_s)) \cap \Gamma(n_e)$ and $(\Gamma(n_e) - \Gamma(n_s)) \cap \Gamma(n_e)$. To determine the negative or positive affect of the exclusive contacts, a similarity of each exclusive contacts for n_s and n_e needs to be obtained. Thus, we note a $\rho(\cdot)$ function indicates the similarity of a node and its exclusive node (e.g, x and n_s) to determine the positive or negative affect from exclusive nodes. Given a cohesion parameter δ, if $(1 - d(x, n_s)) \geq \delta$ then $\rho(x, n_s) = (1 - d(x, n_s))$; others $\rho(x, n_s) = (1 - d(x, n_s)) - \delta$. Furthermore, similar to common contacts affect determination, we define the affects from exclusive contacts for n_s and n_e as $I_{EC} = -(\sum_{x \in (\Gamma(n_s) - \Gamma(n_s) \cap \Gamma(n_e))} \frac{1}{deg(n_s)} \cdot sin(1 - d(x, n_s)) \cdot \rho(x, n_s) + \sum_{y \in (\Gamma(n_e) - \Gamma(n_s) \cap \Gamma(n_e))} \frac{1}{deg(n_e)} \cdot sin(1 - d(y, n_e)) \cdot \rho(y, n_e))$.

Based on the above three patterns, the community detection can be obtained by the dynamic distance between each two nodes in network. Given two node u and v and iteration step τ, the iterative distance between u and v is calculated by:

$$d(u, v, \tau + 1) = d(u, v, \tau) + I_{DC}(\tau) + I_{CC}(\tau) + I_{EC}(\tau) \quad (2)$$

Finally, Equation 2 is applied to update distances of whole nodes in network iteratively until converge. The distance range of two node is [0,1], the lower distances equate to the closer two nodes are. During each iteration, we remove edges whose distance equal (or larger than) 1, and obtain communities from the connected component.

Discussion: We illustrate the steady state for incremental changes of edges' distances after iterations in Figure 1(c). The advantages for distance based community dection model suitable for epidemic diffusion can be concluded as the following three-folds. First, epidemic diffusion is a dynamic spreading activity for traceable interactions, in which the states of nodes are monitored and changed temporally. The changes of network are revealed adaptively in distance based model and updated simultaneously of distance dynamics. Second, compared to modularity gradually increment models, the dynamic distance with local interaction model gives richer variation descriptions for communities. For example, with adjustment of cohesive parameter δ value, distance-based model allows the detection of arbitrary-size communities. Third, when scaling to large-scale networks, the distance-based model provide community detection methodology an ability to reveal the distance of nodes within adaptive subgraph partitions locally.

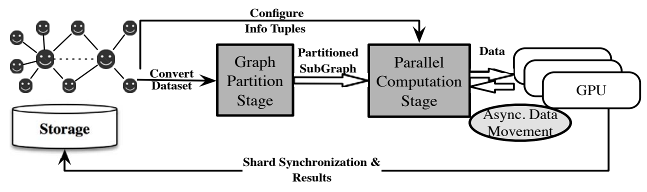
IV. Parallel Community-Based Early Detection Approach for Epidemic Networks

Following the paradigm of Jaccard distance based community detection, we design our `EpCom` approach in the analysis of epidemic diffusion. As the size and complexity of epidemic networks increase (*Challenge 2* in Section II), pursuing faster community detection in parallel computing environment is emerging to be considered. To achieve optimal performance of community detection, we choose the *high parallelism of GPU* to build `EpCom` due to high computing and power efficiency offered by GPU that promise to provide an excellent hardware platform. The strategy we designed considers both the quality and the performance of scaling methodology to parallel computing environment. Moreover, considering the limited GPU-resident memory resource for residing datasets, we design an efficient data partition and distribution scheme to schedule local kernel execution in GPU threads.

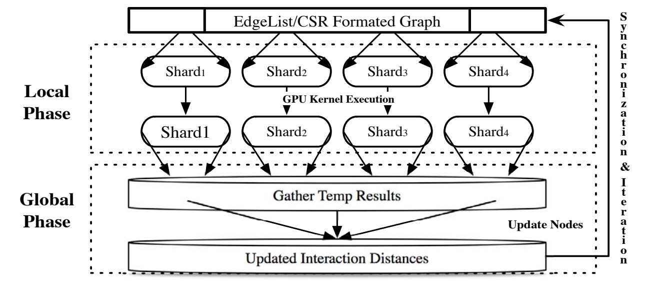
In this prototype, we support an out-of-core strategy for large-scale realistic social network, in which the large-scale epidemic contact networks have larger sizes than GPU's device memory. More specifically, these large-scale networks can be processed efficiently on one single GPU-based machine by employing compact partitions and effective data movement and computation. Meanwhile, a pipeline of community detection for epidemic diffusion is implemented, which consists of two stages, graph partition and parallel processing (shown in Figure 2(a)).

A. Approach Overview

The overview of `EpCom` is shown in Figure 2(a), in which the two stages in gray blocks represent our main contribution for optimization. Within `EpCom` approach, the workflow is transformed to data partition stage and parallel computation stage, along with a preliminary step to read the raw graph data into host memory.



(a) WorkFlow of Spreador Method



(b) Sub-phases of Parallel Computation Stage

Fig. 2. `EpCom` Method Overview.

1) Graph Partition Stage: This stage is to partition the input raw epidemic network graph reads to subgraph partitions (i.e., shards) to make them accessible to hold in GPU's limited device memory. In `EpCom`, we provide two choosable formats for raw graph as edge list or Compressed Sparse Row (CSR). For the paired reads and workload balance, the reads are sequentially partitioned into shards (edge size fit into device memory) in this stage to guarantee that each shard will be resident in GPU and processed correctly (detailed in Sec. IV-B).

2) Parallel Computation Stage: This stage covers whole parallel procedures in CPU and GPU, in which GPU independently processes each individual partitioned shard one by one. After partition stage, each shard is then treated as a single chunk with an assigned ID, loaded and processed into GPU device memory as a whole. It should be noted that the chunk-based data movement over PCIe is more efficient than pageable transfer. When the processing on shards finishes, the computation results are transformed to a collection of key-value (i.e., <*(Shard ID, Contact ID), Affects*>) pairs. In each collection, the key is generated from local computation, which specifies the exactly mapped nodes in the local shards and combined same key pairs into one. The result collections are copied to host memory as data chunks. Meanwhile, we leverage an data movement optimization in this stage by introducing Stream Object provided by CUDA to overlap data movement and GPU kernel execution [20], in which the operations from multiple Streams are executed concurrently and interleaved. As shown in Figure 2(b), we separate the distance-based affect computation phase implemented in `EpCom` to two sub-phases: *Local phase* and *Global phase*. Within local phase, the shards and their corresponding node value vectors are sequentially executed by GPU's kernel functions. By simulating the dynamic affects on edges (i.e., *DC, CC, EC* contacts among individuals) in shards (cf. Eq. (2)), it yields the transmission distances among nodes. Along with introducing the three pattern affects in network topology, the transmission distances among individuals in different communities increase gradually while those sharing a same community tend to decrease. After iterative loading and processing shards in GPU, the distance result collections among individual are copied to host memory. Then in global phase of CPU, the final distances will be gathered and cohered to one set. Finally, all distances for interactions will converge. After removing edges with maximum distance (i.e., $d(n_s, n_e) \geq 1$), communities can be easily obtained from the connected component. Sec. IV-C details the design for parallel computation stage.

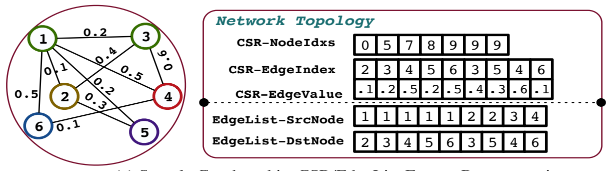
B. Realistic Network Representation and Partition Stage

The input raw epidemic network formats in EpCom are edge list or CSR matrix. In EpCom, we mainly consider the realistic social network analytics via GPU is limited by the fact that raw dataset cannot fit into GPUs' limited device memories. Thus, the partitioning or chunking not only needs to fit into GPU memory but also has to deal with the irregular nature of networks and I/O manner of input data. We designed *"shard"* data structure to present epidemic network in EpCom, inspired by the traditional CPU-based work [21]. As shown in Figure 3(c), network partitioning is performed by segmenting the node set $\widehat{V}$ of network $\widehat{G}$ into disjoint intervals and for each interval, allocating a shard, in which all the edges that have a source node in that interval are maintained in the corresponding shards.

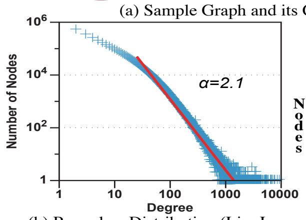
Considering the non-uniform nature of realistic epidemic networks (e.g., power-law degree distribution (shown in Figure 3(b))), EpCom focus on two key technical points about the network partition: (1) the determination of intervals and (2) the construction of shards. For (1), the number of interval $|P|$ is determined such that one entire shard (or multiple shards), at least, can be resident completely into GPU device memory. Meanwhile, the intervals are chosen in a load-balance fashion. Thus each shard for processing contains an approximately equal number of interactions (contacts) among nodes, then the intervals consist of the whole source nodes from shards. For (2), the edge list are maintained in CSR compressed sparse row format (shown in Fig. 3(a)). In CSR format, *NodeIdxs* array ($|\widehat{V}|$) represents the starting index of a correspoding edge sub-array in *EdgeIndex*. Then *EdgeIndex* array ($|\widehat{E}|$) represents the destination node id of edges and the weight of edges are stored in *EdgeValue* array ($|\widehat{E}|$) . The data-format transposition can be benefit from the sequential disk I/O for partitioning edge list. Therefore, EpiCom can enjoy the compressed storage space and also greatly reduce the preprocessing overhead.

C. Parallel Processing Stage

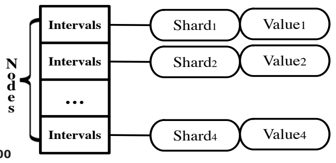
In this section, we present the simplified procedure pseudocode of our EpCom approach. The illustration of parallel computation stage is shown in Fig. 2(b). Algorithm 1 presents the pseudo code of GPU-based parallel procedure of EpCom. It performs four phases for the specific community detection for epidemic diffusion, where the *local phase* is performed using the high parallelism of GPU and other three phases are in CPU. Before computation, EpCom allocates two buffers in GPU device EDBuf, UVBuf, and loads network data set $\widehat{G}$ into memory (MMBuf), creates GPU Streams for GPU


0.2
0.4
0.5
0.6
0.1
0.2
0.3
0.5
0.1
1
2
3
4
5
6
Network Topology
CSR-NodeIdxs 0 5 7 8 9 9 9
CSR-EdgeIndex 2 3 4 5 6 3 5 4 6
CSR-EdgeValue .1 .2 .5 .2 .5 .4 .3 .6 .1
EdgeList-SrcNode 1 1 1 1 1 2 2 3 4
EdgeList-DstNode 2 3 4 5 6 3 5 4 6

(a) Sample Graph and its CSR/EdgeList Format Representation



(b) Powerlaw Distribution (LiveJournal)



(c) Shard Partition Representation

Fig. 3. Network Representation in EpCom.

asynchronized I/O (Line 1-5). Then, the dynamic distance of each interaction is initialized using the Equation (1) (Line 6-10). We note that distance values of interactions are stored in MMBuf and synchronized in UVBuf of GPU device.

After data and device initialization, based upon the proposed interaction model (cf. Eq. (2)), the GPU-based community detection in epidemic diffusion can be parallel simulated. The *while* loop (Lines 11-34) takes charge of phase-by-phase processing. It mainly involves the two following phases:

a) Local Phase (GPU): After loading one shard into GPU EDBuf, the kernel function of GPU is triggered to process the interactions' affects. The value range of distance for two nodes need to be in $0 < d_e < 1$. Lines 13-26 are performed in GPU for updating the distances of two nodes by accumulating the affects from there patterns I_{DC}, I_{CC} and I_{EC} (discussed in Sec. III). The step number is noted by t and the dynamic distances would converge until there are no changes for distance values (i.e., d_e equals to 0 or 1). At the end of each iteration, the updated distance values *UVBuf* and states of interactions are copied back to *MMBuf* in main memory, to be next processed.

b) Global Phase (CPU): The global phase performs in Lines 29-33, in which we obtain the updated distances for whole interactions. Finally, by removing the interactions with maximal distances ($d_e = 1$), the distances will converge and the community can be easily obtained from the connected components (Line 35).

V. Experimental Results

In this section, we present experimental results in three categories. First, we evaluate the effectiveness of Jaccard distance-based community detection model in epidemic diffusion, by comparing with the most popular models, *NCut* [12] and *Louvain* [22]. Second, we evaluate the performance of EpCom compared with the state-of-art parallel community detection processing methods, GPU-based *Lourvain* implementation [17], to show the superiority of our approach. For reference, we also evaluate the performance of parallel methods using CPUs, including multi-core CPU-based *Louvian* [22] and *NCut* implementation [12]. Third, we evaluate the performance of EpCom while varying optimal strategies (graph partition, asynchronized computation and I/O), the densities of graphs to show the characteristics of EpCom.

Algorithm 1: Parallel Procedure of `EpCom`

```
Input: $\widehat{G} = (\widehat{V}, \widehat{E}, \widehat{W})$: Input network dataset
Input: $M_h, M_d$: the size of host and device memory
Input: PLAN: Structure for stream objects and data in $GPU$
1  Create $PLAN.stream$ for $GPU$;
2  Allocate EDBuf, UVBuf in $GPU$ device memory;
3  Allocate MMBuf in main memory;
4  $P \leftarrow \frac{|\widehat{E}|}{M_d}$, obtain individual intervals;
5  $ShardSet$, $ValueSet \leftarrow \cup_{1 \leq i \leq P} TopoData$ from $\widehat{G}$;
   /* Initialization Phase                                  */
6  for shard $s \in ShardSet$ of $\widehat{G}$ do
7      for edge $e = (n_s, n_e) \in s$ do
8          Compute initial distance $d^0_{n_s n_e}$ using Eq. (1);
9      end
10 end
11 while all distance $d_e$ are 0 do
12     Async-copy $nextShard\ S_i$ to EDBuf in $GPU$;
       /* Local Phase: GPU Kernel function                  */
13     for edge $e = (n_s, n_e) \in nextShard$ do
14         if $0 < d^t_{(n_s,n_e)} < 1$ then
15             Compute $I^t_{DC(n_s,n_e)}$, $I^t_{CC(n_s,n_e)}$, $I^t_{EC(n_s,n_e)}$
               affects from interactions;
16             Compute Eq. (2) obtain $d(n_s, n_e, t+1) =$
               $d^t_{(n_s,n_e)} + I^t_{DC(n_s,n_e)} + I^t_{CC(n_s,n_e)} + I^t_{EC(n_s,n_e)}$;
17             if $d(n_s, n_e, t+1) > 1$ then
18                 $d(n_s, n_e, t+1) = 1$;
19             end
20             else if $d(n_s, n_e, t+1) < 0$ then
21                 $d(n_s, n_e, t+1) = 0$;
22             end
23             Update $d(n_s, n_e, t+1)$ value in UVBuf;
24         end
25         Async-copy UVBuf of $GPU$ to MMBuf;
26     end
27     $next\_shard \leftarrow ShardSet$;
28     Thread Synchronizatin;
       /* Global Phase: Updating Distance                   */
29     for edge $e = (n_s, n_e) \in \widehat{E}$ do
30         if $d(n_s, n_e, t+1)$=1 then
31             Remove edge $e$ from the network $\widehat{E}$;
32         end
33     end
34 end
35 Obtaining the resulting components (communities) $C$;
```

A. Experimental Setup

Setup. The evaluations are conducted on five realistic networks with a broad range of sized and features from dierent origins, including two small social networks and three large-scale epidemic network datasets, which are from Stanford Large Network Dataset Collection (http://snap.stanford.edu/data/). We illustrate the evaluation datasets in Table I. More specifically, the *KarateClub*, *Friendship* and *LiveJournal* are three social network graph data set, where users actively interact with other members in the networks. *ER16MV* is generated by a Erdős-Rényi (ER) random graph model, which contains 16 million agents or humans and 134 million person-person social interactions or edges. Then *US41MV* is a realistic social contact network spanning the continental United States (US dataset), synthesized using census and relevant data [23], which contains 41.7 million agents and 1.4 billion interactions. The interactions or edges are initially assigned weight values to represent the factor of infected possible rate γ. The experiments are performed on a system with NVIDIA GeForce GTX980 each having 16 Maxwell Streaming Multiprocessors (128 Cores/MP) and 4GB GDDR5 RAM. The host side of the node is consist of two 10-core Intel Xeon E5-2650 v3, and 64GB DDR4 main memory, running with Ubuntu 16.04 (kernel v4.4.0-38) with CUDA 7.5.

TABLE I
Dataset of Epidemic Networks for Evaluation.

Dataset	**Nodes** $\lvert V \rvert$	**Edges** $\lvert E \rvert$	**Avg. Degree** deg_{avg}	**RAW Data** size
KarateClub	34	78	4.58	3.39KB
Friendship	58,228	214,078	14.71	10.2MB
LiveJournal	4.8 million	68.9 million	28.46	2.6GB
ER16MV	16 million	134 million	16	12GB
US41MV	41.7 million	1.4 billion	47.68	54GB

Comparisons. Among competitions of our model and approach, we selected two representative methods, including *NCut* [12] and *Louvain* [22]. Specifically, *NCut* method is a well-known algorithm for graph clustering by achieving the optimized *cut criterion*. *Louvain* is a popular community detection algorithm based on modularity measure, which hierarchical community detection. We specify the cluster number in *NCut* to $|C|$, which is the true number of classes with the available ground truth, and configure default value for parameters in *Louvain*.

Metrics. We set the cohesion parameter $\delta = 0.5$ for `EpiCom` as default parameter. And three network metrics are introduced for the community detection model comparison, as following.

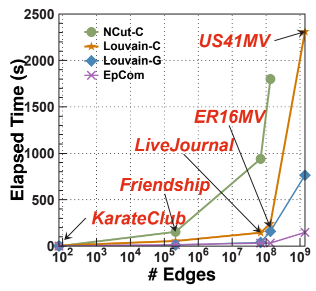
- *Modularity Measurement* [14]: measure the quality of the partition of epidemic networks, defined as a score $Q = \sum_{c_i \in C}[\frac{|E^{in}_{c_i}|}{|E|} - (\frac{2|E^{in}_{c_i}|+|E^{out}_{c_i}|}{2|E|})^2]$, where C is the set of all communities, $|E^{in}_{c_i}|$ and $|E^{out}_{c_i}|$ are the number of in-edges and out-edges of c_i community.
- *Normalized Cut (ncut) Measurement* [12]: measure the minimization of cut edges w.r.t. disease control.
- *Similarity Measurement* [24]: measure the similarity between two communities for subnetworks, *F-measure*$= \frac{1}{|V|}\sum_{c_i \in C}|c_i| \cdot \max_{c'_j \in C'} \frac{2|c_i \cap c_j|}{|c_i|+|c'_j|}$.

With the respect to performance comparison, we choose to a GPU-based *Louvain* implementation (named *Louvain-G*) and two parallel methods using CPUs, including multi-core based *Louvain* (*louvain-C* and *NCut* implementation (NCut-C).

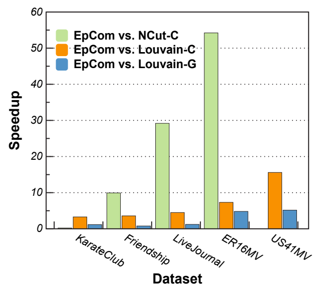
B. Effectiveness of Distance-Based Community Detection in Epidemic Diffusion

The statistics of above realistic networks are summarized in Table II. We report the external metrics *Modularity*, *NCut* and *Similarity* for the comparison of *NCut* and *Louvian* models with our proposed *Jccard distance based* model in `EpCom`.

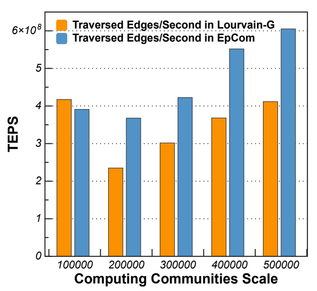
Zachary karate club network: The network is frequently used as a pedagogical illustration of community detection,



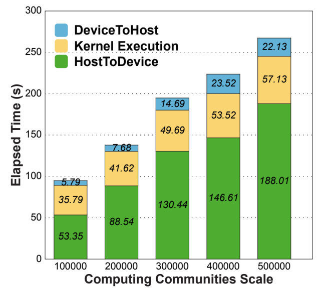
(a) Elapsed Time in Various Networks



(b) Normalized Speedup Comparison



(c) TEPS Comparison to *Louvain-G*



(d) GPU Internal Processing Time

Fig. 4. The comparison results for elapsed time and detailed processing time of related methods and `EpCom` on different network inputs. Note that *NCut-C* method cannot process US41MV due to running "out-of-memory".

TABLE II
COMMUNITY DETECTION MODEL COMPARISON IN NETWORKS.

Dataset	Metrics	`EpCom`	NCut	Louvain
KarateClub	*#community*	3	3	4
	modularity	0.40	0.427	0.415
	ncut	0.92	2.41	2.03
	similarity	**0.9266**	0.3521	0.7356
Friendship	*#community*	8045	7863	746
	modularity	0.44	0.24	0.684
	ncut	7325	41724	38.6
	similarity	**0.8961**	0.6243	0.6839
LiveJournal	*#community*	976474	890482	56961
	modularity	0.68	0.45	0.740
	ncut	32408	165286	492.9
	similarity	**0.9145**	0.4659	0.8067

and previously divided into two smaller clubs following a conflict between its members. We plot the community structure comparison of the network between ground truth communities and detected by `EpCom` as shown in Figure 5. Specifically, two communities are successfully detected in `EpCom`, and the node 30 is found as anomaly. Although *NCut* and `Louvain` also achieve the similar *modularity* and performance, many individuals are wrongly clustered, which results in relatively high values of *ncut*.

Brightkite friendship network: `EpCom` detects the maximum quantity of communities (8045) and shows an obvious advantage over the other models. For *NCut* (7863 communities), many individuals are incorrectly clustered, leading to a low value of *modularity* 0.24 and much large value of *ncut*. *Louvain* tend to produce a small quantity of communities, due to neglecting many small-sized communities.

Social Livejournal network: The network represents the power law degree distribution and a data sparsity, as we illustrated in Figure 3(b). `EpCom` identifies 976474 communities with *modulartiy* 0.68 and 32408 *ncut*. While *NCut* achieves a comparable community number and a better *modularity*, it entails a much larger number 165286 of *ncut*. *Louvain* only find small number of 56961 equal-size communities. Here, `EpCom` also achieve the best performance in similarity between subnetworks among communities.

C. Performance Comparison with Other Approaches

We first show the performance effects of our `EpCom` approach on GPU-based platform. The three comparison of core decomposition algorithms are coming from the state-of-art approaches, including a GPU-based Louvain implementation

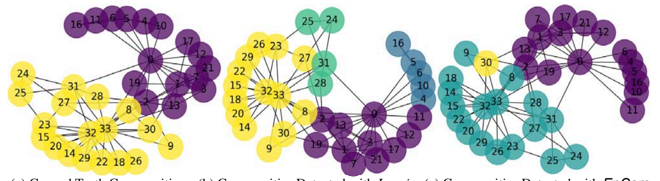

(a) Ground Truth Communities (b) Communities Detected with *Louvian*(c) Communities Detected with *EpCom*

Fig. 5. Ground truth community and detected community structure on KarateClub network. Colors of nodes indicate different communities.

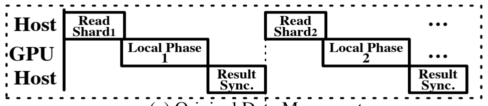
(*Lourvain-G*), two multi-core CPU based methods for Lourvain and NCut implementation (*Lourvain-C* and *NCut-C*). The *Lourvain-C* and *NCut-C* implementations are configured to use 16 threads in evaluation. Figure 4(a) shows us the elapsed time for `EpCom` with GPU can achieve the best optimal performance compared to other executions, since `EpCom` has a linear time complex against to $|E|$ and also utilizes the high parallelism of GPU.

Then we illustrate the speedup ratio for `EpCom`. Compared to CPU-based *NCut-C* and *Lourvain-C* methods, `EpCom` achieves 10-54.2 times and 3.2-15.6 times of speedup separately. With the size of temporal graph data increasing, `EpCom` with GPU represent much better performance enhancement. Meanwhile, with respect to *Louvain-G* algorithm, `EpCom` also achieves 1.2-5.2 times of speedup over the same configuration of GPU environment. The reason that GPU-based `EpCom` represents excellent performance accelerating was that the optimized scalable algorithms benefit from efficient massive parallelism and effective data movement among large scale networks.

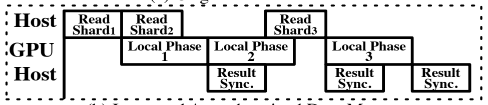
Figure 4(c) illustrates TEPS (traversed edges per second) metric comparison for GPU-based methods, *Louvain-G* and `EpCom`. Results show `EpCom` achieves maximum 604 million TEPS in wiki-talk graph and almost enhances 1.5× TEPS than *Louvain-G*. Figure 4(d) summaries the memory copy time and kernel exection time during GPU internal processing. It's obviously, the data copy overhead occupies a major proportion in the total GPU processing time, and showing data movement between CPU and GPU is worth to be optimized.

D. Optimization Characteristic of `EpCom`

We applied *CUDA stream object* to reorganize the execution flow of data movement and kernel execution, expecting that the data movement via PCI-E may increase in case storage IO is not fully utilized. As shown in Figure 6, the original execution (a) implies that a GPU is not fully utilized due to the idle

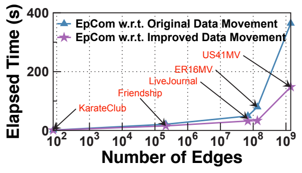


(a) Original Data Movement

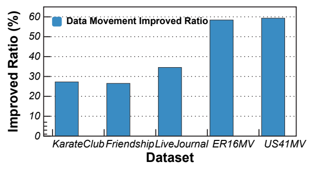


(b) Improved Asynchronized Data Movement

Fig. 6. Overlapping Data Movement and Kernel Execution in `EpCom`.



(a) Improved Comparison



(b) Optimized Speedup

Fig. 7. The comparison results for optimized data movement in `EpCom`.

waiting for *shard read* and result synchronization. Thus, by introducing CUDA stream object to manage and organize the execution flow, the performance gained from asynchronized data movement have presented a sharply improved. And with one stream, the idleness of GPU can be eliminated with a little fraction. Figure 7 illustrates the evaluation for improved data movement in `EpCom`, and shows that imporved `EpCom` achieved almost 59.4% performance enhancement and shows better improved ratio with larger dataset.

VI. Conclusion

In this paper, we propose a novel parallel community detection approach for the simulated epidemic diffusion. First, based on the introduction of Jaccard distance between individuals, we present a dynamic distance view in epidemic network and apply three local interaction patterns to determine the affects among individuals. Second, we design a GPU-based approach for parallel community detection in large-scale epidemic networks. The utilization of linear time complex Jaccard distance based model and the high parallelism of GPU make `EpCom` approach a great improvement for community detection in large-scale social epidemic networks. Extensive experiments demonstrate our Jaccard distance based community detection model allows effectiveness finding communities in epidemic networks with high quality, and also shows benefits in detecting small communities and community similarity. Our GPU-based `EpCom` implementation exhibits remarkable improvements in runtime performance. The maximum speedup is achieved 54.2 times and 15.6 times than CPU-based *NCut* and *Louvain* approaches separately.

References

[1] B. Shekh, E. De Doncker, and D. Prieto, "Hybrid multi-threaded simulation of agent-based pandemic modeling using multiple gpus," in *Bioinformatics and Biomedicine (BIBM), 2015 IEEE International Conference on*. IEEE, 2015, pp. 1478–1485.

[2] C. L. Barrett, K. R. Bisset, S. G. Eubank, X. Feng, and M. V. Marathe, "Episimdemics: an efficient algorithm for simulating the spread of infectious disease over large realistic social networks," in *Proceedings of the 2008 ACM/IEEE conference on Supercomputing*. IEEE Press, 2008, p. 37.

[3] J.-S. Yeom, A. Bhatele, K. Bisset, E. Bohm, A. Gupta, L. V. Kale, M. Marathe, D. S. Nikolopoulos, M. Schulz, and L. Wesolowski, "Overcoming the scalability challenges of epidemic simulations on blue waters," in *Parallel and Distributed Processing Symposium, 2014 IEEE 28th International*. IEEE, 2014, pp. 755–764.

[4] P. C. Pinto, P. Thiran, and M. Vetterli, "Locating the source of diffusion in large-scale networks," *Physical review letters*, vol. 109, no. 6, p. 068702, 2012.

[5] K. Zhu, Z. Chen, and L. Ying, "Catch'em all: Locating multiple diffusion sources in networks with partial observations." in *AAAI*, 2017, pp. 1676–1683.

[6] M. Salath and J. H. Jones, "Dynamics and control of diseases in networks with community structure," *PLOS Computational Biology*, vol. 6, no. 4, pp. 1–11, 04 2010.

[7] N. Gupta, A. Singh, and H. Cherifi, "Community-based immunization strategies for epidemic control," in *2015 7th International Conference on Communication Systems and Networks (COMSNETS)*, Jan 2015, pp. 1–6.

[8] V. Wong, D. Cooney, and Y. Bar-Yam, "Beyond contact tracing: community-based early detection for ebola response," *PLoS currents*, vol. 8, 2016.

[9] C. Stegehuis, R. van der Hofstad, and J. S. van Leeuwaarden, "Epidemic spreading on complex networks with community structures," *Scientific reports*, vol. 6, 2016.

[10] M. Salathé and J. H. Jones, "Dynamics and control of diseases in networks with community structure," *PLoS computational biology*, vol. 6, no. 4, p. e1000736, 2010.

[11] E. Aramaki, S. Maskawa, and M. Morita, "Twitter catches the flu: detecting influenza epidemics using twitter," in *Proceedings of the conference on empirical methods in natural language processing*. Association for Computational Linguistics, 2011, pp. 1568–1576.

[12] J. Shi and J. Malik, "Normalized cuts and image segmentation," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 22, no. 8, pp. 888–905, 2000.

[13] S. Fortunato, "Community detection in graphs," *Physics reports*, vol. 486, no. 3, pp. 75–174, 2010.

[14] M. Chen, K. Kuzmin, and B. K. Szymanski, "Community detection via maximization of modularity and its variants," *IEEE Transactions on Computational Social Systems*, vol. 1, no. 1, pp. 46–65, 2014.

[15] C. Hennig and B. Hausdorf, "Design of dissimilarity measures: A new dissimilarity between species distribution areas," in *Data Science and Classification*. Springer, 2006, pp. 29–37.

[16] Y. Wang, A. Davidson, Y. Pan, Y. Wu, A. Riffel, and J. D. Owens, "Gunrock: A high-performance graph processing library on the gpu," in *Proceedings of the 21st ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming*. ACM, 2016, p. 11.

[17] G. Li, D. Zhang, K. Xie, T. Huang, and Y. Li, "A gpu based fast community detection implementation for social network," in *International Conference on Algorithms and Architectures for Parallel Processing*. Springer, 2015, pp. 688–701.

[18] T. N. Dinh, N. P. Nguyen, and M. T. Thai, "An adaptive approximation algorithm for community detection in dynamic scale-free networks," in *INFOCOM, 2013 Proceedings IEEE*. IEEE, 2013, pp. 55–59.

[19] J. Shao, Z. Han, Q. Yang, and T. Zhou, "Community detection based on distance dynamics," in *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2015, pp. 1075–1084.

[20] N. Wilt, *The cuda handbook: A comprehensive guide to gpu programming*. Pearson Education, 2013.

[21] A. Kyrola, G. Blelloch, and C. Guestrin, "Graphchi: Large-scale graph computation on just a PC," in *10th USENIX Symposium on Operating Systems Design and Implementation (OSDI 12)*. Hollywood, CA: USENIX, 2012, pp. 31–46.

[22] X. Liu and T. Murata, "Advanced modularity-specialized label propagation algorithm for detecting communities in networks," *Physica A: Statistical Mechanics and its Applications*, vol. 389, no. 7, pp. 1493–1500, 2010.

[23] C. L. Barrett, R. J. Beckman, M. Khan, V. A. Kumar, M. V. Marathe, P. E. Stretz, T. Dutta, and B. Lewis, "Generation and analysis of large synthetic social contact networks," in *Simulation Conference (WSC), Proceedings of the 2009 Winter*. IEEE, 2009, pp. 1003–1014.

[24] S. Wagner and D. Wagner, *Comparing clusterings: an overview*. Universität Karlsruhe, Fakultät für Informatik Karlsruhe, 2007.