Graph-enabled Intelligent Vehicular Network Data Processing

Zhigao Zheng, *Member, IEEE*, Ali Kashif Bashir, *Senior Member, IEEE*

Abstract—Intelligent vehicular network (IVN) is the underlying support for the connected vehicles and smart city, but there are several challenges for IVN data processing due to the dynamic structure of the vehicular network. Graph processing, as one of the essential machine learning and big data processing paradigm, which provide a set of big data processing scheme, is well-designed to processing the connected data. In this paper, we discussed the research challenges of IVN data processing and motivated us to address these challenges by using graph processing technologies. We explored the characteristics of the widely used graph algorithms and graph processing frameworks on GPU. Furthermore, we proposed several graph-based optimization technologies for IVN data processing. The experimental results show the graph processing technologies on GPU can archive excellent performance on IVN data.

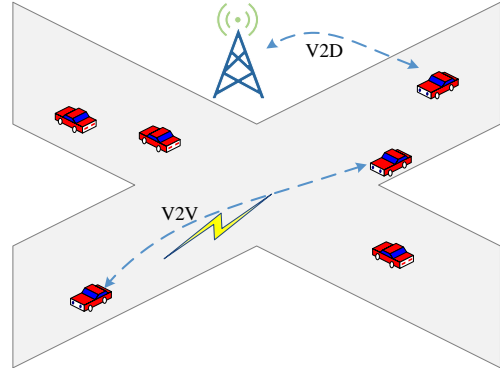*Index Terms*—Graph processing, vehicular networks, Internet of intelligent vehicles.

Fig. 1: An example structure of vehicular networks. V2D is the communication of vehicle-to-device, while the V2V is the communication of vehicle-to-vehicle.

I. INTRODUCTION

WIRELESS network is the basis of many applications that can support mobile broadband access. In recent years, the wireless network attracted more and more attention from both industry and academics [1], [2]. Vehicular network is a smart-vehicles network with complex intra-vehicle systems [3], [4], [5]. Various vehicular infrastructures (such as cameras, traffic lights) and kinds of sensors, which used to collect vehicle and driving status, are included in the vehicular network. The general architecture of the vehicular network shown in Fig. 1, there are three layers (vehicles, connections, and servers/clouds) that included in this architecture. The vehicles communicate with each other through the V2V (vehicle to vehicle) network, at the same time, the vehicles can communicate with remote mobile phones and some other devices through V2I (vehicle to infrastructure) network. Promoted by the new onboard computing and sensing technologies, the vehicular networks have become one of the most important aspect of the intelligent transportation systems (ITS) and smart cities [6], [7], [8]. This new trend proposed some new challenges for current vehicular communication systems with higher security, reliability, and effectiveness requirements. Along with recent advances in some new technologies such as computational intelligence, artificial intelligence (AI),

Zhigao Zheng is with School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan 430074, China E-mail: (zhengzhigao@hust.edu.cn).

Ali Kashif Bashir is with the Department of Computing and Mathematics, Manchester Metropolitan University, Manchester, United Kingdom and also with the National University of Science and Technology, Islamabad, Pakistan. E-mail:(dr.alikashif.b@ieee.org)

blockchain, and 5G will help the ITS move forward into a smarter level with autonomous driving in near further.

In recent years, there are many academic and industry organizations, and even some government departments also proposed a set of communication standards for vehicular Ad Hoc networks (VANETs), including the Dedicated Short-Range Communication (DSRC) [9], and ITS-G5 [10], [11] in the United States and Europe respectively, both of them are proposed based on IEEE 802.11p [12]. However, some recent researches [13], [14] show the communication data processing technologies suffered from several issues, such as lack of quality of service (QoS) guarantees, channel access delay, different communication standard transfer, and short-lived vehicle-to-infrastructure (V2I) connection. In order to solve all these problems, the 3rd Generation Partnership Project (3GPP) started several investigations to find the vehicle-to-everything (V2X) supporting services for the long term evolution (LTE) network and the future 5G cellular system [15]. With the development of device banded communications, some researchers try to employ the device-to-device (D2D) communications to support the vehicle-to-vehicle (V2V) transmission in cellular systems [6], [16], [17]. In addition, recent research have extended the graph algorithms and some graph enabled communication data processing framework for vehicular networks resource allocation [16], [17], [18], [19]. However, there are still some challenges in designing graph enabled vehicular network communication data processing, such as how to provide QoS guarantees for the heterogeneous network, and how to ensure the computation results in the dynamic vehicular environment.

Meanwhile, with the development of sensing technologies

and 5G, the further vehicles will be equipped with more and more smart sensors, such as the radar, engine monitor and control unit, light control unit and some other device such as remote emergency alarm device as well as some networks based applications such as social networking applications. Most of these sensors need to be work in real-time, together with the high-performance computing and storage unit onboard, and these sensing technologies will promote the intelligent driving progress into autonomous driving. All these sensors and onboard devices keep collecting, generating, storing, and communicating with each other, which generates large scale connected data. There are complex connections between all these communicating and controlling data, how to explore the relations between these data to improve the control and communication performance for all these devices is a great challenge.

The graph is a fundamental mathematical structure used to model pairwise relations between objects, and it is widely used in machine learning [20], [21], and deep learning [22] technologies to express the connections between different objects. The context in a graph is called vertices (also called nodes), while the connections are called edges. Graph theory has been widely used in vehicle communications [23], [24], a graph-based metrics is proposed to gauge the redundancy of dissemination protocols in [25], some open issues of the metrics such as attackers colluding and eviction are also included in this work. The authors of [26] proposed a VANET communication model based on evolving graph theory to determines the regular routes preemptively. The authors of [27] formulated the problem of cooperative communications scheduling in vehicular networks by using graph theory, and the authors proposed a bipartite-graph-based (BG) scheduling scheme to allocate the vehicle-to-infrastructure (V2I) and V2V links for both single-hop and dual-hop communications. The experimental result shows that the proposed method can archive an excellent overall performance than state-of-the-art works. Graph theory and algorithms have provided essential theory support for vehicular networks in resource allocation and communication modeling [28], and graph applications also help the network more informed and data-driven decisions. However, how to use graph theory and algorithms to support the distinctive characteristics of intelligent vehicular networks and provide high performance and real-time decision making policy remains challenging and represents a promising research direction.

In this paper, we discuss some major challenges in supporting intelligent vehicular networks with high performance, such as real-time decision making and network topologies updating in dynamic changing environments, path planning, and QoS in high dimensional communication links. To address these challenges, we introduce some high-performance graph processing frameworks into intelligent vehicular network data processing. Furthermore, we proposed a fundamental shift of the graph processing approach for intelligent vehicular network data processing framework, Hooker. We also designed a set of experiments to verify that the idea of applying the graph processing approach into intelligent vehicular network communication data processing is extremely appealing for several reasons.

The rest of the paper is organized as follows. We introduce the characteristics and research challenges of intelligent vehicular network data processing and the motive of using the graph processing method to process the large scale intelligent vehicular network data in section II. We introduce the major graph processing frameworks and algorithms in section III and IV. Then, we introduce how we can use graph algorithms to solve the challenges of intelligent vehicular network data processing, and also the design methodologies and principles of Hooker in section V. The experimental evaluation of the existed graph processing frameworks introduced in section VI. At last, we conclude the paper in section VII.

II. CHALLENGES OF IVN COMMUNICATION DATA PROCESSING

High dimensional intelligent vehicular network data exhibit distinctive characteristics, which have created significant challenges to intelligent vehicular network design. In this section, we discuss these challenges and try to figure out the potential solutions by using graph processing technologies.

A. Highly Dynamic Topology

Unlike typical mobile network graphs, the vertices represent vehicles are moving at quite high speed, which leads the topology of vehicular network changes frequently. This characteristic will affects system design and computing in multiple aspects of vehicular networks. On the one hand, the network connection performance is much worse than a common mobile network, and the link failures and message loss rate will also be increased, under this scenario. How to elongate the life of communication connections and how to manage the connection links between the moving vehicles is an excellent challenge for intelligent vehicular network. On the other hand, the network density also changes with moving vehicles. The network density may be very high when traffic jam occurred, while network density will be relatively low in suburban traffic. This changing density may lead to a high communication delay. How to reduce the delay for a changing density network is another challenge. At last, the distance between the vehicles is also challenging with the vehicle moves. How to update the computation results, such as the shortest path between two vehicle is also a challenge for existed vehicular networks.

B. High Reliability and Scalability Requirements

With the development of vehicle technologies and big data processing technologies, there are more and more smart transportation and aided driving applications have been used in our daily life. While all transportation and driving-related applications are safety-sensitive, hence, how to design a highly reliable intelligent vehicular communication networks is necessary. However, due to the complex network topology with poor stability and large network scale, how to design a highly reliable vehicular network is a great challenge.

As mentioned earlier, there are more and more vehicles and some other devices added into the intelligent vehicular

network, which make the network proliferates. How to keep the high scalability to keep the high quality of services (QoS) of the network is another challenge.

C. The Potential of Graph Processing

Graph processing has been widely used in kinds of applications to deal with the complex relations, such as molecule structures model in chemistry, cognitive processes in computational neuroscience, atoms structure analysis in physics, code structures analysis in computer science, and so on. To facilitate the development of large scale graph processing, there are kinds of generic graph programming, and both academic and industry searchers have implemented computing frameworks. Both single machine graph processing frameworks, such as GraphChi [29], X-Stream [30], and GridGraph [31], and distributed graph processing systems, such as Pregel [32] and PowerGraph [33] are included.

Recently, the technical advance of kinds of the new hardware and accelerator, such as the General-Purpose Graphics Processing Units (GPGPUs) [34] and FPGA, has attracted many researchers from both academic and industry to investigate how to use these kinds of new hardware and accelerators to accelerate the computational and memory-intensive applications, including graph processing [35], [36]. Benefit for the massive parallelism and high memory bandwidth, the GPGPU has been widely used for graph processing frameworks. With the efforts from both academics and industry, a set of general graph processing frameworks have been developed, such as Totem [37], Medusa [38], CuSha [39], and GunRock [40]. In particular, graph processing, one of the domain-specific big data processing systems, can interact with a dynamic environment and develop satisfactory policies to meet diverse data format and QoS requirements to of vehicular network in a dynamic and varying wireless environment. For example, a vehicle adds or leaves the network will make the network structure changes.

III. GRAPH ALGORITHMS FOR IVN APPLICATIONS

There are kinds of graph processing algorithms that can be used to accelerate IVN applications. For example, Breadth-First Search (BFS) is widely used in kinds of graph searching applications, Betweenness Centrality (BC) is used to find important paths and vertices in IVN, some other algorithms such as Connected Component (CC) is used to find the component of vehicles which with frequent communications, Single Source Shortest Path (SSSP) is used find the suitable paths in root/road planning, and PageRank (PR) is used to find the important vehicle in the components of intelligent vehicular network applications. This section try to discuss some existed efforts to accelerate specific graph algorithms on GPU. We will introduce some typical algorithms in the following sections.

A. Path Planning

Traversal algorithms can be leveraged to find the paths in dynamical vehicular networks, including the path planning and vehicle trajectory prediction. The path planning can be further used towards navigation planning and data pre-processing for system performance improvement. For example, the single source shortest path (SSSP) algorithm is widely used for path planning, and betweenness centrality (BC) is used to solve the traffic assignment problem [41]. Rami et al. [41], shows the applicability of betweenness centrality and certain augmented types of it for obtaining traffic flows through links of a transportation network. In Rami's work, the authors achieved a strong positive correlation with the traffic flows in transportation networks by using the betweenness centrality measurement and then proposed a betweenness-driven traffic assignment model to optimize the position of the transport network. The proposed method can archive an excellent performance due to betweenness centrality simultaneously considers all shortest paths between an origin and a destination. Ademar et al. [42], proposed a distributed approach to compute egocentric betweenness scores over VANETs by only use the local knowledge of the network topology, which can release the computation capacity of vehicles.

B. Vehicle Scheduling

Vehicle scheduling is one of the most typical applications of intelligent vehicles and smart cities, which is widely used in intersection management and intelligent drive control. The most classic application of intersection management is to decide the passing order of the vehicles passing through the intersection without traditional traffic signals. The naïve implementation of vehicle scheduling is the greedy method, which called as First-Come-First-Serve (FCFS) approach. In this method, the manager schedules the vehicles according to the arrival time. The earlier come ones passing through the intersection eariler. The FCFS is easy to implement, however, this method ignores some essential information, such as the interactions between vehicles and conflict zones, and thus leads to extra delay in many cases. On the other hand, the FCFS method do not take the priority into consideration, which can delay some important vehicles and also will lead to some serious problem. Based on improved visibility graphs, Wei et al. [43] proposed a local path planning algorithm for an intelligent vehicle on a structured road. This paper introduced the graph processing model for vehicle scheduling, which can improve the scheduling efficiency.

In intersection management, our objective is to minimize the total time needed for all vehicles to go through the intersection, equivalent to the leaving time of the last vehicle. To remove cycles while considering the edge costs, finding a minimum spanning tree (MST) of the graph can be a potential solution, and one approach is Kruskal's algorithm [44]. Kruskal's algorithm repeatedly chooses a minimum-cost edge, which does not form any cycle with those already-chosen edges. Kruskal also proposed the backward version of the original one, and it repeatedly removes a maximum-cost edge whose removal does not disconnect the graph. Inspired by this method, we do intend to remove the edge, which results in the most significant delay to the objective. This can remove cycles and benefit the objective minimization at the same time. Based

on the graph model, Lin develops a centralized cycle removal algorithm for the graph-based model to schedule vehicles to go through the intersection safely (without collisions) and efficiently without deadlocks [45]. The algorithm is sufficiently efficient to consider more conflict zones and more vehicles in real-time.

IV. GRAPH PROCESSING FRAMEWORKS ON GPU

Many graph processing frameworks are proposed in recent years, including some single machine framework [29] and distributed frameworks [33], and also some frameworks on new hardware devices, such as GPUs [40] and FPGA [46], [47]. In this paper, we consider the character of the IVN data, and try to conclude some optimization strategies for graph processing framework for IVN data processing. In this section, we discuss the programming models and system implementation and optimization technologies.

A. Graph Programming Models

The GAS (Gather-Apply-Scatter) and BSP (Bulk Synchronous Parallel) programming models are mostly used, BSP is firstly used in Pregel [32] and GAS is proposed in PowerGraph [33].

BSP programming model is widely used in many graph processing frameworks on GPU, including TOTEM, Medusa and GunRock. In BSP programming model, the program executed as a so-called super-steps. The threads run asynchronously in parallel within a super-step, while all the threads need to be synchronized at the end of the super-step. Hence, the threads only can communicate with each other at the end of the super-steps, which is called a barrier. This execution fashion is easy to implement, but it is easy to lead the straggler problem since the computation task on the vertices are varies.

There are also many graph processing frameworks on GPU are adopt the GAS model, such as MapGraph [48] and CuSha [39]. In the GAS programming model, the program on each vertex is divided into three phases, which are named *Gather*, *Scatter*, and *Apply*. In *Gather* phase, the vertices collect information from their neighbors, and the updates from the neighbors will be added in the *Apply* phase, then the *Scatter* phase broadcast the updates to the neighbors to promotes the program into the next iteration. In this fashion, the GAS can be implemented in an asynchronous fashion, which can avoid the synchronization overhead and the straggler problem.

B. Memory Access Pattern

The random memory access manner of graphs will limit the performance of the GPU, how to unleash the computation ability of GPU is a great challenge. In this section, we discuss some existed technologies on memory access optimization.

In GraphReduce [49], the vertices are sorted according to the source vertices ID to make sure all the edge from the same vertex can be visited at the same time. A Unified Virtual Addressing (UVA) technology is also used to allocate the memory space for GraphReduce, combined with the DMA memory loading/storing technology, GraphReduce can make the memory access in a sequential manner, and the communication overhead can be overlapped with GPU computations through pre-fetching.

In GTS [50], the graph data have been distinguished according to the attribute of the data, the *attribute data* was copied into GPU on broad memory with a long lifetime, while the *topology data* are copied to the GPU device from the host memory by using CUDA streaming. In this processing method, the GPU device can quickly visit the whole graph, while the extraordinary computation data can be loaded when the computation moves on. GTS also introduced the SSDs to accelerate the data transfer operation.

C. Workload Mapping

The workload imbalance problem is caused by the irregular vertex degree. How to mapping the uneven workload onto GPU is another challenge. The mostly used method is evenly assign the thread to the edges. In this method, every thread just processes an edge, the computation task of every thread is the same. However, there are much more threads are needed than assign threads to vertices, which will slow down the overall performance. In order to solve this problem, *Dynamic scheduling* strategy is proposed in MapGraph [48]. In MapGraph the threads are assigned according to the vertex's degree, a warp is assigned to the vertices with lower degree while the CTA is assigned the vertices with higher degree. In this thread assignment model, it is easy to distinguish different workloads, and then it can dynamically schedule the workload.

Gunrock [40] implemented a similar thread assignment fashion with MapGraph [48], but Gunrock classified the tasks into fine- and coarse-grained workload, then assign the thread warp/block to the different workloads. A virtual warp include multiple warps is also implemented in Gunrock, which is assigned to the vertex with tremendous degree.

D. Miscellaneous

There are some other research challenges in graph-enabled IVN data processing, for example, how to provide a programmer-friendly interface and how to improve the degree of parallelism.

Medusa [38] and MapGraph [48] both provides a set of programmer-friendly APIs, which is easy for the programmers to implement the user-defined functions. In addition, Medusa also provides a set of configuration parameters to enhance the flexibility of the framework. Unlike the previous implementation, GunRock provides a data-centric abstraction to enhance the flexibility of the library.

V. GRAPH-ENABLED IVN FRAMEWORK

Graph processing represents a useful tool for a variety of applications. In this paper, we proposed a graph-enabled IVN data processing framework, Hooker. In Hooker, we designed a new programming model to fit multi-architectures, a new communication paradigm to maintain the data consistency, integrality and security.

A. The JCS programming model

Vertex-centric programming model is easy to express most of the graph algorithms, and it can provide high scalability by partition the graphs. However, it is easy to lead random memory access and load imbalance problem, due to the skewed degree distribution of real-word graphs. Edge-centric programming model can provide a continuous memory access fashion, but it will introduce lots of redundant computation since there are many more edges than the vertices in real-world graphs. The Gather-Apply-Scatter (GAS) is a fine-grained vertex-centric programming model, which proposed in PowerGraph [33]. Previous research shows that the number of active vertices in each iteration is far less than the total number of vertices in a graph [33]. Hence, this paper proposes a queue-based vertex-centric Join-Compute-Scatter (JCS) programming model, which is shown in figure 2. In JCS programming model, the operation on the vertex is divided into join, compute and scatter three steps. The join operation adds the active vertices into the worklist, and the compute operation updates the vertex's value according to the user-defined function, while the scatter operation scatter the vertex's value to its neighbours, which similar like the scatter operation in GAS model. In JCS model, each iteration cares about the vertices which need to be updated. This execution fashion can provide a unified and concise implementation for different algorithms, and it can provide high scalability for the vertex-centric method.
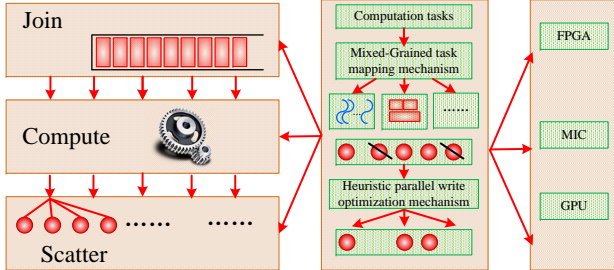

Fig. 2: The JCS programming model and how it matches with multiple hardware.

In order to make the JCS model matches with different kinds of hardware architecture, this paper provides a mixed granularity task mapping mechanism and a heuristic parallel write optimization mechanism. We introduce the two optimization mechanisms as follow:

- **Mixed-granularity task mapping mechanism.** Here, we take GPU as an example to introduce the mixed-granularity task mapping mechanism. We assign the vertices to thread, warp, CTA, and kernel according to the number of the neighbour of the active vertex, and the virtual-warp is used to solve the load imbalance problem. While, on KNL, the proposed mechanism not only supports regular round-level parallelism for different task size, including the *for-all* parallelism, reduction parallelism, and scan parallelism, it also supports the irregular parallelism, which can achieves excellent load balancing through reasonable task stealing scheduling.

- **Heuristic parallel writes optimization mechanism.** There are many duplicate vertices in the worklist since there are more than one vertices connected with the same vertex. Atomic operations and locks are introduced to ensure data consistency, but both atomic operations and lock operation will lead lost of conflicts with will slow down the overall performance. However, recent researches show that some updating operations are idempotent (i.e. the updating order does not affect data consistency). Hence, there is no need to design a specific algorithm by using atomic operation or lock to remove the duplicate vertices from the worklist, design a lightweight runtime heuristic policy is enough to remove the redundant computation, which will be more efficient.

B. Feature-aware data partitioning and placement strategy

In order to meet the architecture's feature to unleash the device performance, this paper propose a three-dimensional graph partition scheme. The proposed graph partition scheme will load the graph blocks into the device on broad memory to make sure the locality of data access and hence to reduce the I/O overhead by considering the communication overhead. Figure 3 show the essential operation of the proposed graph partition scheme.
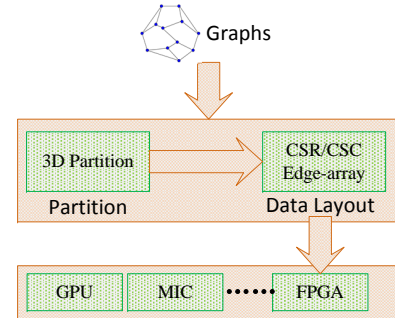

Fig. 3: Data partitioning and placement strategies for data-aware and structure-aware.

The existed graph processing system applied the one-dimensional or two-dimensional partition strategy, which is the vertex-centric and edge-centric partition method, respectively. The vertex-centric partition strategy will lead to a load imbalance problem, since the vertex degree distribution of most real-world graphs is power-law. While the edge-centric partition strategy will lead to a large amount of communication between the master node and the replicas. Recent research proposed a 3D partition strategy, which partition the vertex attribution as the third-dimensional partition object, this partition can archive an excellent system performance in machine learning applications, but do not suitable for full broad applications [31]. The traversal tree-based partition method can maintain good locality, but the partition operation executed after traverse the whole graph. The overhead of this partitioning method is enormous. In order to solve the problems of the existed partition methods, this paper proposes a hybrid partition method. The proposed method first partition the vertices with similar degrees together by using a two-dimensional

partition method and then partition the sub-graphs again by using a one-dimensional partition method. The hybrid partition strategy will take the third-dimensional partition as the first round partition for some specific graph algorithms, and then partition the results again by using a two-dimensional partition method. This method can archive load balance and very low communication computation ratio for different algorithms and architectures.

The data placement is closely related to the representation of the graph, and it has a deep effect on system performance. The upper-level framework requires the graph representation with high memory bandwidth utilization and locality of memory access. While, the lower-level storage requires high space utilization, avoid the space-wasting for sparsity graph. The storage level also requires the graph can be loaded into the memory during the I/O operation in graph processing. In order to meet both the upper-level and lower-level's requirements, this paper provides a hybrid CSR/CSC graph representation, and the edge-list representation is also provided according to the characteristics of the graph. The mixed CSR/CSC graph representation is benefit for the Scatter/Gather operation. For example, some implementation of the BFS algorithm will change the traversal direction from bottom-up to top-down (or top-down to bottom-up). This hybrid representation will improve memory access efficiency by changing CSR to CSC (or change CSC to CSR). Community is another essential characteristic of the real world graphs, i.e. the vertices in a community are connected but few vertices connect with the vertices outside the community. In this kind of graph, the community can be processed by some SIMD devices, such as GPU, by using the edge-list representation will archive an excellent performance. Hence, edge-list representation can be an optional method for users.

C. Comunnications

In order to processing large scale IVN graphs, we proposed a layer centric communication model for Hooker, which is shown in figure 4. In this communication model, the threads in each layer passing messages to each other asynchronous, while the messages need to be synchronized between layers. This communication model can accelerate the task in each layer and keep the data consistency, integrality and security between layers.
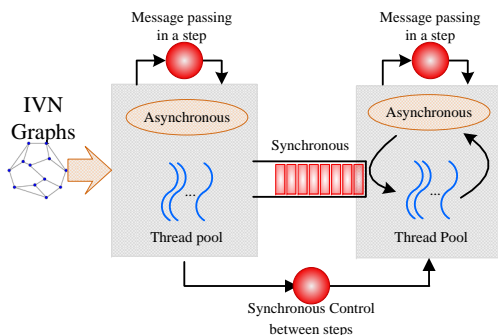

Fig. 4: Layer centric communication model.

VI. EXPERIMENTS

In this paper, we conducted a set of experiments to try to find some insights for graph enabled IVN data processing framework. On the one hand, we compare the performance of graph processing frameworks with different types of graphs, which widely used in an intelligent vehicular network. On the other hand, we compare the performance of different graph processing frameworks, which implemented on different devices. Hence, we can conclude that when we need to use some accelerator device to accelerate the algorithm on GPU and what kind of framework we can use to process the different types of intelligent vehicular network data.

A. Experimental Datasets and Algorithms

In this section, we introduce some typical datasets and algorithms which chosen to demonstrate the performance of the graph-enabled IVN data processing frameworks.

All the datasets of the experiments conducted in this paper are follow the classic graph formalism [51]. We use V and E to present the vertices and edges of the graph, respectively. $G = (V, E)$ present the graph. The edge presented as e, where $e = (u, v)$ and $e = <u, v>$ are the undirected and directed edges. In this paper, both directed and undirected graphs used in our experiments.

In order to include both power-law and large diameter graphs, we select six graphs from different real-world applications, such as e-business, social network and some other source networks. All these graphs are with different structures and a varying number of vertices and edges. The graphs are shown as table I. All these six graphs can be downloaded from the Stanford Network Analysis Project (SNAP) [52]. As the power-law graph follow a distribution, shown as formula 1 [53], [54], we list the exponent and the fitness in table I to compare the power-law attribution. In this paper, the graphs stored in a plain text file with edgelist format, which is easy for us to locate the edges.

$$\mathbb{P}(x) \propto x^{-\alpha} \qquad (1)$$

In order to compare the performance of different algorithms on all these frameworks, we implemented four most frequently used graph algorithms, include two traversal algorithms (BFS and SSSP) and two iterative algorithms (PageRank and Connected Component).

All the experiments are conducted on an NVIDIA GeForce GTX 1060 GPU, which is a Maxwell architecture device with 1280 CUDA cores and 6GB onboard memory. The programs compiled by using CUDA 8.0 with "-arch=sm 35" flag on Ubuntu 16.04. We also compared the graph frameworks on GPU with some frameworks which run on CPU, such as GraphChi [29], X-Stream [30], and GridGraph [31]. All the CPU enabled graph frameworks are run on a machine with 8GB memory and 4 Intel Core(TM) i5-5200 CPUs at 2.2GHz. All the source codes are provided by the authors.

B. Graph-processing frameworks

Both graph processing frameworks on GPU and CPU, which are widely used in research and industrial communities, are

TABLE I: Datasets used in the experiments

Datasets	Vertices	Edges	Avg. Degree	Max Degree	Diameter	Exponent(α)	x_{min}	Fitness (p)
Amazon	735,323	5,158,388	7.01	1,076				
dblp-2011	986,208	6,707,236	6.80	979	23	3.9736	119	0.4
RoadNet-CA	1,971,282	5,533,214	2.81	12	8,440	15.5587	4	0
Wiki-Talk	2,394,386	5,021,410	4.19	100,032	11	2.4610	1	0.787
soc-LiveJournal	4,847,571	86,220,856	28.25	22,887	20	2.6510	59	0.930
twitter-2010	41,652,229	1,468,365,167	70.51	3,081,112	23	1.54	12	0.96

compared. The fameworks on GPU are TOTEM [37], CuSha [39], Medusa [38], Gunrock [40], and MapGraph [48].

TOTEM is the first GPU and CPU hybrid system [37], which partition the graphs into two parts, one partition is processed on GPU and the other one is processed on CPU. The vertex-centric processing pattern is used in TOTEM, and there are three different partition strategies for TOTEM, named HIGH-degree, LOW-degree and RANDOM-degree, according to the vertices' degree. The HIGH-degree partition strategy assigns the high degree vertices to the CPU and assigns the low degree vertices to the GPU. LOW-degree partition strategy is opposite with the HIGH-degree strategy, and the RANDOM-degree strategy the vertices to CPU and GPU device in a random manner. Medusa and CuSha adopt the edge-centric processing model, G-Shards and Concatenated Windows (CW) technology like GraphChi are implemented in CuSha, while Medusa provides a set of simplified programming interface. MapGraph implemented the GAS strategy, which proposed in PowerGraph, on GPU. Gunrock is the most recent high-performance library on GPUs. All the parameters in this paper are the same with the authors' original work list in their publication, which can achieve the best system performance.

C. Experiment Results

1) Perspective of dataset: We first try to conclude that types of datasets and algorithms which can executed on GPU. In this experiment, the BFS is implemented as a textbook version, while PR and SSSP are implemented according to the best-reported performance in [37], and CC is implemented according to Wu's [55] version. Figure 5 shows the experimental performance of different algorithms executed different datasets for both GPU and CPU.

Figure 5 shows that the algorithms achieve better performance on GPU than on CPU except for PageRank. As we discussed before, GPU has higher computing power than CPU when the memory access is regular. However, a graph is of the irregular data structure, and the GPU memory accessed irregularly in graph algorithms. Figure 5 also indicts that GPU is hard to process the large scale graph, particularly when the graph size reaches the memory size. The experiment shows the performance of PageRank on GPU is even worse than CPU, this is because PageRank needs to send the updates of the vertices to their neighbours in every iteration, there are many communication operations. As the GPU connected with the host through PCI-E bus, while the bandwidth of PCI-e is limited. Hence, the compunction cost is the main bottleneck of PageRank on GPU, in particular for the large scale graphs which with the graph size larger than the GPU

(a) Amazon   (b) DBLP

(c) RoadNet   (d) Wiki-Talk
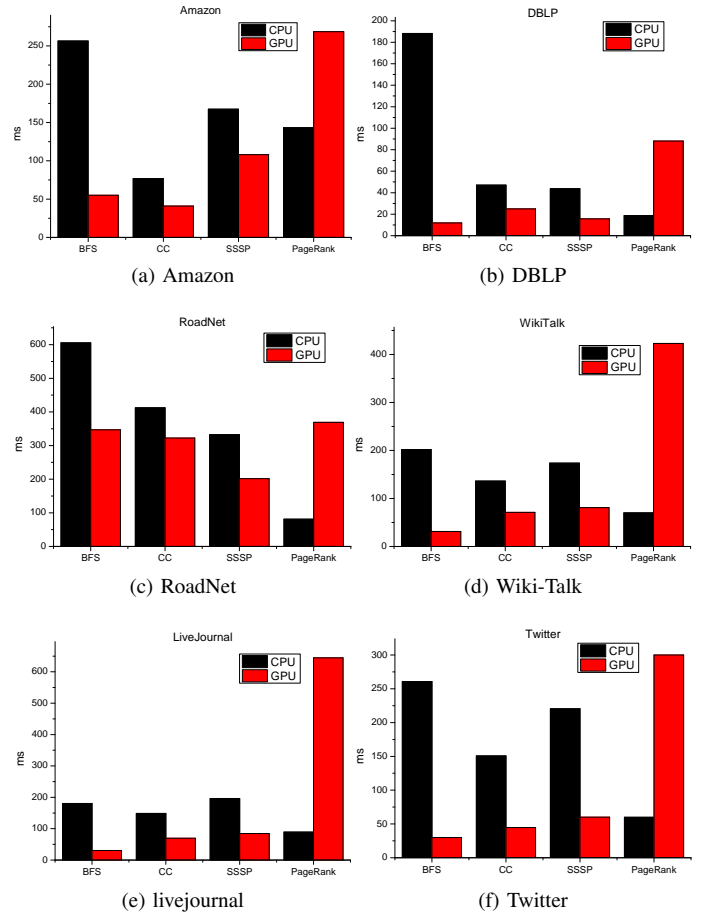
(e) livejournal   (f) Twitter

Fig. 5: System performance for different graphs on CPU and GPU

on broad memory. The experiment also shows that the power-law datasets, such as Amazon and Twitter can achieve a good speed up when the dataset can be loaded into the GPU on broad memory, while the large diameter graphs such as RoadNet can not achieve such an excellent performance.

Comparing Figure 5(a), Figure 5(c) and Figure 5(f), we can conclude that the algorithms performance on Amazon and Twitter outperforms on RoadNet. This result indicates that GPU is more suitable to process power-law graphs than CPU.

2) Perspective of algorithms: In order to investigate the performance of different algorithms, we run the selected algorithms on different GPU graph processing systems. For each combination of algorithm and graph processing framework, we run the algorithm 5 times and obtain the average performance. The experimental results are shown in Tables II–V.

We can conclude from the experimental results that Gunrock

TABLE II: Execution Time of BFS (ms)

Datasets	Amazon	DBLP	LiveJournal	RoadNet-CA	WikiTalk	Twitter
TOTEM	23.46	6.48	58.38	439.96	79.96	25.68
CuSha	100.087	31.949	68.833	2632.5	6.983	5.074
Medusa	4.5169	3.024	13.29	85.884	2.009	3.253
MapGraph	4.48	4.83	4.0868	35.142	7.702	1.683
Gunrock	6.0179	3.813	30.699	37.575	0.2222	0.1662

TABLE III: Execution Time of PageRank (ms)

Datasets	Amazon	DBLP	LiveJournal	RoadNet-CA	WikiTalk	Twitter
TOTEM	55.3	44.26	446.16	179.64	264.27	117.34
CuSha	5.507	3.61	10.383	10.113	9.795	23.191
Medusa	14.265	29.798	47.89	15.77	394.299	69.616
MapGraph	17.198	28.357	27.827	15.987	83.366	23.95
Gunrock	22.6998	16.3391	101.9559	35.6269	74.024	74.2691

outperforms all other GPU-based graph processing systems with BFS. This is because Gunrock makes use of the frontier to store its graph data, creating a 'queue' for the data structure. This works well with BFS-like algorithms. TOTEM follows Gunrock as the second most competitive system.

Table II shows a comparison of the performance of all six GPU based graph processing systems on the BFS algorithm. MapGraph has a clear performance advantage on Amazon, however, Medusa outperforms other systems on DBLP. RoadNet-CA is the dataset with the biggest diameter, which makes the processing more difficult for BFS. However, MapGraph and Gunrock achieve good performance with this dataset. This is because they use the queue-like data structure.

The damping factor and precision of PageRank in our experiments set as 0.85 and 0.005, respectively. Table III shows that CuSha performs better than any other system with PageRank, which mainly benefits from the strategy of G-shards partition.

We also looked at three CPU-based graph processing frameworks, GraphChi, X-Stream, and GridGraph. GraphChi chops large graphs into small parts. It designs a Parallel Sliding Windows (PSW) method in order to gain a better efficiency for random access to vertices. We can see that CPU-based graph processing systems achieve higher performance than the GPU-based systems when processing the PageRank algorithm. This is because PageRank uses a considerable number of random accesses, which will slow down the GPU performance.

From Table II–V we can conclude that not all the graph algorithms are suitable for the GPU graph processing system. BFS-like algorithms, which access the memory regularly by using some data layout techniques, are suitable to be run by the GPU-based graph processing system, so are the power-law graphs.

TABLE IV: Execution Time of CC (ms)

Datasets	Amazon	DBLP	LiveJournal	RoadNet-CA	WikiTalk	Twitter
TOTEM	135.117	120.636	93.729	2880.238	185.123	1690.31
CuSha	135.719	126.619	91.41	1509.08	81.637	962.49
MapGraph	104.08	118.6	70.09	121.86	50.71	230.34
Gunrock	33.87	35.77	24.01	963.01	42.95	544.3

TABLE V: Execution Time of SSSP (ms)

Datasets	Amazon	DBLP	LiveJournal	RoadNet-CA	WikiTalk	Twitter
TOTEM	30.39	3.79	33.91	492.51	140.07	45.83
CuSha	107.285	34.6	80.208	2948.24	6.918	5.054
MapGraph	38.26	25.547	14.56	606.64	26.023	605.91
Gunrock	224.9389	104.5270	251.8849	1253.0050	1.6890	0.3371

(a) BFS on different systems

(b) CC on different systems

(c) PageRank on different systems
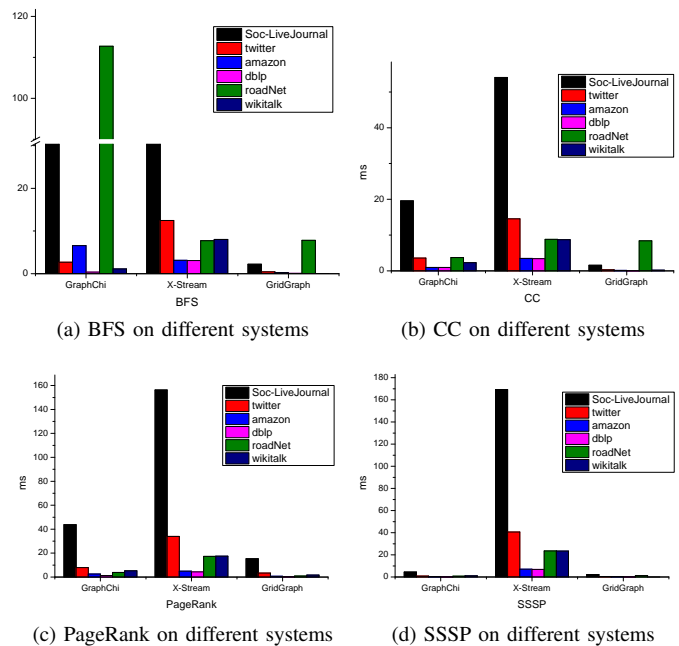

(d) SSSP on different systems

Fig. 6: The performance of different algorithms on CPU graph processing system.

VII. CONCLUSION AND FUTURE OPPORTUNITIES

The IVN is the basis of smart cities, how to processing the dynamic IVN data is a significant challenge. Nowadays, most of the big data processing systems are under a batch processing model, which is hard to process the dynamic data. The graph processing frameworks, with a set of data processing paradise for the highly connected data, may provide a chance for IVN data processing. This paper first discussed the research challenges of IVN data processing and then proposed a new paradise for IVN data processing.

In the future, we plan to focus on some other aspects of IVN data processing, such as the IVN processing framework on some other new devices like Directed Self Assembly (DSA) and Field Programmable Gate Array (FPGA), the real-time dynamic data processing for IVN or some other new devices.

REFERENCES

[1] L. Liang, H. Peng, G. Y. Li, and X. Shen, "Vehicular communications: A physical layer perspective," *IEEE Transactions on Vehicular Technology*, vol. 66, no. 12, pp. 10 647–10 659, Dec 2017.

[2] G. Araniti, C. Campolo, M. Condoluci, A. Iera, and A. Molinaro, "Lte for vehicular networking: a survey," *IEEE Communications Magazine*, vol. 51, no. 5, pp. 148–157, May 2013.

[3] W. Wu, Z. Yang, and K. Li, "Chapter 16 - internet of vehicles and applications," in *Internet of Things*, R. Buyya and A. V. Dastjerdi, Eds. Morgan Kaufmann, 2016, pp. 299 – 317. [Online]. Available: http://www.sciencedirect.com/science/article/pii/B9780128053959000162

[4] E. Ahmed and H. Gharavi, "Cooperative vehicular networking: A survey," *IEEE Transactions on Intelligent Transportation Systems*, vol. 19, no. 3, pp. 996–1014, March 2018.

[5] B. Menezes, C. Canales, T. Zimmerman, and M. Toussaint, "Gartner 2019 magic quadrant for the wired and wireless lan access infrastructure," https://www.fortinet.com/solutions/gartner-wired-wireless-lan.html.

[6] X. Cheng, L. Yang, and X. Shen, "D2d for intelligent transportation systems: A feasibility study," *IEEE Transactions on Intelligent Transportation Systems*, vol. 16, no. 4, pp. 1784–1793, Aug 2015.

[7] L. Zhu, F. R. Yu, Y. Wang, B. Ning, and T. Tang, "Big data analytics in intelligent transportation systems: A survey," *IEEE Transactions on Intelligent Transportation Systems*, vol. 20, no. 1, pp. 383–398, Jan 2019.

[8] J. Seo and T. Walter, "Future dual-frequency gps navigation system for intelligent air transportation under strong ionospheric scintillation," *IEEE Transactions on Intelligent Transportation Systems*, vol. 15, no. 5, pp. 2224–2236, Oct 2014.

[9] J. B. Kenney, "Dedicated short-range communications (dsrc) standards in the united states," *Proceedings of the IEEE*, vol. 99, no. 7, pp. 1162–1182, July 2011.

[10] M. Jutila, J. Scholliers, M. Valta, and K. Kujanpää, "Its-g5 performance improvement and evaluation for vulnerable road user safety services," *IET Intelligent Transport Systems*, vol. 11, no. 3, pp. 126–133, 2017.

[11] M. Wegner, T. Schwarz, and L. Wolf, "Connectivity maps for v2i communication via etsi its-g5," in *2018 IEEE Vehicular Networking Conference (VNC)*, Dec 2018, pp. 1–8.

[12] "Ieee standard for information technology– local and metropolitan area networks– specific requirements– part 11: Wireless lan medium access control (mac) and physical layer (phy) specifications amendment 6: Wireless access in vehicular environments," *IEEE Std 802.11p-2010 (Amendment to IEEE Std 802.11-2007 as amended by IEEE Std 802.11k-2008, IEEE Std 802.11r-2008, IEEE Std 802.11y-2008, IEEE Std 802.11n-2009, and IEEE Std 802.11w-2009)*, pp. 1–51, July 2010.

[13] G. Araniti, C. Campolo, M. Condoluci, A. Iera, and A. Molinaro, "Lte for vehicular networking: a survey," *IEEE Communications Magazine*, vol. 51, no. 5, pp. 148–157, May 2013.

[14] A. Vinel, "3gpp lte versus ieee 802.11p/wave: Which technology is able to support cooperative vehicular safety applications?" *IEEE Wireless Communications Letters*, vol. 1, no. 2, pp. 125–128, April 2012.

[15] A. Sultan and M. Pope, "3rd generation partnership project; technical specification group services and system aspects; release 14 description; summary of rel-14 work items(release 14)," 3GPP, Valbonne, FRANCE, Technical Report 21.914, 2018.

[16] L. Liang, S. Xie, G. Y. Li, Z. Ding, and X. Yu, "Graph-based radio resource management for vehicular networks," in *2018 IEEE International Conference on Communications (ICC)*, May 2018, pp. 1–6.

[17] Z. Li, J. Liu, and Y. Chen, "An energy-efficient resource allocation strategy for vehicular networks," in *Proceedings of the ACM Turing Celebration Conference - China*, ser. ACM TURC '19. New York, NY, USA: Association for Computing Machinery, 2019. [Online]. Available: https://doi.org/10.1145/3321408.3326657

[18] D. Liu, M.-C. Lee, C.-M. Pun, and H. Liu, "Analysis of wireless localization in nonline-of-sight conditions," *IEEE Transactions on Vehicular Technology*, vol. 62, no. 4, pp. 1484–1492, May 2013.

[19] Y. Huang, M. Chen, Z. Cai, X. Guan, T. Ohtsuki, and Y. Zhang, "Graph theory based capacity analysis for vehicular ad hoc networks," in *2015 IEEE Global Communications Conference (GLOBECOM)*, Dec 2015, pp. 1–5.

[20] K. Zhang, L. Lan, J. T. Kwok, S. Vucetic, and B. Parvin, "Scaling up graph-based semisupervised learning via prototype vector machines," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 26, no. 3, pp. 444–457, March 2015.

[21] M. Nickel, K. Murphy, V. Tresp, and E. Gabrilovich, "A review of relational machine learning for knowledge graphs," *Proceedings of the IEEE*, vol. 104, no. 1, pp. 11–33, Jan 2016.

[22] Z. Zhang, P. Cui, and W. Zhu, "Deep learning on graphs: A survey," 2018.

[23] C. L. Azevedo, J. L. Cardoso, and M. Ben-Akiva, "Applying graph theory to automatic vehicle tracking by remote sensing," in *Proceedings of the 93rd Annual Meeting on Transportation Research Board*, ser. TRB 2014. Washington, DC, USA: TRB, 2014, pp. 1–12. [Online]. Available: https://trid.trb.org/view/1288375

[24] ——, "Vehicle tracking using the k-shortest paths algorithm and dual graphs," *Transportation Research Procedia*, vol. 1, no. 1, pp. 3 – 11, 2014, planning for the future of transport: challenges, methods, analysis and impacts - 41st European Transport Conference Selected Proceedings. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S2352146514000039

[25] S. Dietzel, J. Petit, G. Heijenk, and F. Kargl, "Graph-based metrics for insider attack detection in vanet multihop data dissemination protocols," *IEEE Transactions on Vehicular Technology*, vol. 62, no. 4, pp. 1505–1518, May 2013.

[26] M. H. Eiza and Q. Ni, "An evolving graph-based reliable routing scheme for vanets," *IEEE Transactions on Vehicular Technology*, vol. 62, no. 4, pp. 1493–1504, May 2013.

[27] K. Zheng, F. Liu, Q. Zheng, W. Xiang, and W. Wang, "A graph-based cooperative scheduling scheme for vehicular networks," *IEEE Transactions on Vehicular Technology*, vol. 62, no. 4, pp. 1450–1458, May 2013.

[28] H. Kröner, "Radio resource allocation for data services in umts networks," *AEU - International Journal of Electronics and Communications*, vol. 55, no. 1, pp. 55 – 62, 2001. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1434841104702472

[29] A. Kyrola, G. Blelloch, and C. Guestrin, "Graphchi: Large-scale graph computation on just a pc," in *Presented as part of the 10th USENIX Symposium on Operating Systems Design and Implementation*, ser. OSDI '12. Hollywood, CA: USENIX, 2012, pp. 31–46. [Online]. Available: https://www.usenix.org/conference/osdi12/technical-sessions/presentation/kyrola

[30] A. Roy, I. Mihailovic, and W. Zwaenepoel, "X-stream: Edge-centric graph processing using streaming partitions," in *Proceedings of the Twenty-Fourth ACM Symposium on Operating Systems Principles*, ser. SOSP '13. New York, NY, USA: ACM, 2013, pp. 472–488. [Online]. Available: http://doi.acm.org/10.1145/2517349.2522740

[31] X. Zhu, W. Han, and W. Chen, "Gridgraph: Large-scale graph processing on a single machine using 2-level hierarchical partitioning," in *2015 USENIX Annual Technical Conference*, ser. USENIX ATC '15. Santa Clara, CA: USENIX Association, Jul. 2015, pp. 375–386. [Online]. Available: https://www.usenix.org/conference/atc15/technical-session/presentation/zhu

[32] G. Malewicz, M. H. Austern, A. J. Bik, J. C. Dehnert, I. Horn, N. Leiser, and G. Czajkowski, "Pregel: A system for large-scale graph processing," in *Proceedings of the 2010 ACM SIGMOD International Conference on Management of Data*, ser. SIGMOD '10. New York, NY, USA: ACM, 2010, pp. 135–146. [Online]. Available: http://doi.acm.org/10.1145/1807167.1807184

[33] J. E. Gonzalez, Y. Low, H. Gu, D. Bickson, and C. Guestrin, "Powergraph: Distributed graph-parallel computation on natural graphs," in *Presented as part of the 10th USENIX Symposium on Operating Systems Design and Implementation*, ser. OSDI 12. Hollywood, CA: USENIX, 2012, pp. 17–30. [Online]. Available: https://www.usenix.org/conference/osdi12/technical-sessions/presentation/gonzalez

[34] J. D. Owens, D. Luebke, N. Govindaraju, M. Harris, J. Krüger, A. E. Lefohn, and T. J. Purcell, "A survey of general-purpose computation on graphics hardware," *Computer Graphics Forum*, vol. 26, no. 1, pp. 80–113, 2007. [Online]. Available: http://dx.doi.org/10.1111/j.1467-8659.2007.01012.x

[35] D. Merrill, M. Garland, and A. Grimshaw, "Scalable gpu graph traversal," in *Proceedings of the 17th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming*, ser. PPoPP '12. New York, NY, USA: ACM, 2012, pp. 117–128. [Online]. Available: http://doi.acm.org/10.1145/2145816.2145832

[36] A. Ashari, N. Sedaghati, J. Eisenlohr, S. Parthasarath, and P. Sadayappan, "Fast sparse matrix-vector multiplication on gpus for graph applications," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, ser. SC '14, Nov 2014, pp. 781–792.

[37] A. Gharaibeh, L. Beltrão Costa, E. Santos-Neto, and M. Ripeanu, "A yoke of oxen and a thousand chickens for heavy lifting graph processing," in *Proceedings of the 21st International Conference on Parallel Architectures and Compilation Techniques*, ser. PACT '12. New York, NY, USA: ACM, 2012, pp. 345–354. [Online]. Available: http://doi.acm.org/10.1145/2370816.2370866

[38] J. Zhong and B. He, "Medusa: Simplified graph processing on gpus," *IEEE Transactions on Parallel and Distributed Systems*, vol. 25, no. 6, pp. 1543–1552, June 2014.

[39] F. Khorasani, K. Vora, R. Gupta, and L. N. Bhuyan, "Cusha: Vertex-centric graph processing on gpus," in *Proceedings of the 23rd International Symposium on High-performance Parallel and Distributed Computing*, ser. HPDC '14. New York, NY, USA: ACM, 2014, pp. 239–252. [Online]. Available: http://doi.acm.org/10.1145/2600212.2600227

[40] 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*, ser. PPoPP '16. New York, NY, USA: ACM, 2016, pp. 11:1–11:12. [Online]. Available: http://doi.acm.org/10.1145/2851141.2851145

[41] R. Puzis, Y. Altshuler, Y. Elovici, S. Bekhor, Y. Shiftan, and A. S. Pentland, "Augmented betweenness centrality for environmentally aware traffic monitoring in transportation networks," *Journal of Intelligent Transportation Systems*, vol. 17, no. 1, pp. 91–105, 2013. [Online]. Available: https://doi.org/10.1080/15472450.2012.716663

[42] A. T. Akabane, R. Immich, R. W. Pazzi, E. R. M. Madeira, and L. A. Villas, "Distributed egocentric betweenness measure as a vehicle selec-

tion mechanism in vanets: A performance evaluation study," *Sensors (Basel)*, vol. 18, no. 8, pp. 1–27, 2018.

[43] H. Wei, "Path planning algorithm of intelligent vehicle based on improved visibility graphs," 2018.

[44] J. B. Kruskal, "On the shortest spanning subtree of a graph and the traveling salesman problem," *Proceedings of the American Mathematical Society*, vol. 7, no. 1, pp. 48–50, 1956. [Online]. Available: http://www.jstor.org/stable/2033241

[45] Y.-T. Lin, H. Hsu, S.-C. Lin, C.-W. Lin, I. H.-R. Jiang, and C. Liu, "Graph-based modeling, scheduling, and verification for intersection management of intelligent vehicles," *ACM Trans. Embed. Comput. Syst.*, vol. 18, no. 5s, Oct. 2019. [Online]. Available: https://doi.org/10.1145/3358221

[46] G. Dai, Y. Chi, Y. Wang, and H. Yang, "Fpgp: Graph processing framework on fpga a case study of breadth-first search," in *Proceedings of the 2016 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, ser. FPGA '16. New York, NY, USA: Association for Computing Machinery, 2016, p. 105–110. [Online]. Available: https://doi.org/10.1145/2847263.2847339

[47] G. Dai, T. Huang, Y. Chi, N. Xu, Y. Wang, and H. Yang, "Foregraph: Exploring large-scale graph processing on multi-fpga architecture," in *Proceedings of the 2017 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, ser. FPGA '17. New York, NY, USA: Association for Computing Machinery, 2017, p. 217–226. [Online]. Available: https://doi.org/10.1145/3020078.3021739

[48] Z. Fu, M. Personick, and B. Thompson, "Mapgraph: A high level api for fast development of high performance graph analytics on gpus," in *Proceedings of Workshop on GRAph Data Management Experiences and Systems*, ser. GRADES'14. New York, NY, USA: ACM, 2014, pp. 2:1–2:6. [Online]. Available: http://doi.acm.org/10.1145/2621934.2621936

[49] D. Sengupta, S. L. Song, K. Agarwal, and K. Schwan, "Graphreduce: processing large-scale graphs on accelerator-based systems," in *SC '15: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, 2015, pp. 1–12.

[50] M.-S. Kim, K. An, H. Park, H. Seo, and J. Kim, "Gts: A fast and scalable graph processing method based on streaming topology to gpus," in *Proceedings of the 2016 International Conference on Management of Data*, ser. SIGMOD '16. New York, NY, USA: Association for Computing Machinery, 2016, p. 447–461. [Online]. Available: https://doi.org/10.1145/2882903.2915204

[51] D. B. West, *Introduction to graph theory*, ser. Math Classics. New Jersey, USA: Prentice Hall, 2001.

[52] J. Leskovec, "Stanford network analysis project," 2009. [Online]. Available: http://snap.stanford.edu/index.html

[53] A. Clauset, C. R. Shalizi, and M. E. J. Newman, "Power-law distributions in empirical data," *SIAM Review*, vol. 51, no. 4, pp. 661–703, Nov. 2009. [Online]. Available: http://dx.doi.org/10.1137/070710111

[54] Y. Virkar and A. Clauset, "Power-law distributions in binned empirical data," *The Annals of Applied Statistics*, vol. 8, no. 1, pp. 89–119, 03 2014. [Online]. Available: https://doi.org/10.1214/13-AOAS710

[55] B. Wu and Y. Du, "Cloud-based connected component algorithm," in *2010 International Conference on Artificial Intelligence and Computational Intelligence*, vol. 3, Oct 2010, pp. 122–126.