2023 IEEE International Performance, Computing, and Communications Conference (IPCCC) | 979-8-3503-0293-6/23/$31.00 ©2023 IEEE | DOI: 10.1109/IPCCC59175.2023.10253885

GPU-based Parallel Computing for VANETs: Current State and Future Prospects

Abinash Borah
Department of Computer Science
Oklahoma State University
Stillwater, Oklahoma, USA
aborah@okstate.edu

Anirudh Paranjothi
Department of Computer Science
Oklahoma State University
Stillwater, Oklahoma, USA
anirudh.paranjothi@okstate.edu

***Abstract*—Vehicular ad-hoc networks (VANETs) demand fast processing of a high volume of data transmitted by the vehicular nodes to meet the requirements of low processing delay and high throughput. To meet such requirements through computational speed-up, parallel computing techniques are adopted in diverse areas. Such an approach is the use of Graphics Processing Unit (GPU)-based parallel computing. To effectively use GPUs, the challenges of this architecture need to be addressed. In this paper, we review the state-of-the-art GPU-based techniques for VANETs, present a taxonomy of them, and identify their limitations. We also recommend future directions toward further usage and optimization of GPU-based parallel computing for VANETs. To the best of our knowledge, this is the first review on GPU-based parallel computing for VANETs.**

Keywords*—*VANET, ITS, parallel computing, GPU

I. Introduction

Vehicular ad-hoc networks (VANETs) serve as the basis of intelligent transportation systems (ITS). VANETs are a special type of mobile ad-hoc network. VANETs facilitate vehicle-to-vehicle (V2V) and vehicle-to-infrastructure (V2I) communication to offer road safety and enrich driving conditions by offering different services ranging from real-time traffic condition monitoring to infotainment services. In VANETs, communication devices called Road-Side Units (RSUs) are deployed at selected critical points alongside the roads, and the vehicles are mounted with devices known as On-Board Units (OBUs). The vehicles, using the OBUs, communicate with each other (V2V communication) and with the RSUs (V2I communication) [1].

In VANETs, the vehicles broadcast periodic beacon messages with information such as their position, speed, identity, etc. In addition, the vehicles broadcast safety messages when an event such as a collision occurs to alert other vehicles. All these messages essentially lead to a high volume of information getting disseminated in the network. Parallelly, the computational techniques for solving different problems in VANETs vastly rely on the messages transmitted by the vehicles, and consequently, the amount of data that needs to be processed is high, especially when the density of the nodes in the network increases. Moreover, high throughput and low processing delay are desired in various VANET applications considering their time-critical nature [2].

The many-core parallelism of Graphics Processing Units (GPUs) is widely used in various domains to accelerate computations. GPUs are parallel co-processors that can support massive parallel computation with high instruction throughput. Several solutions are proposed in the literature for different problems in VANETs where computational speed-up is achieved using GPUs. The characteristics of the GPU architecture bring certain challenges (discussed in section II) in GPU-based parallelization. To achieve optimal performance gain from GPUs, efficiently handling these challenges is important. At the hardware level, proposals for RSUs equipped with GPUs and using devices with GPUs as OBUs have also been explored in the literature. It is noteworthy that the performance of the computational techniques for VANETs is commonly evaluated with simulations, and some studies emphasize accelerating simulations for VANETs using GPUs. Another perspective considering the importance of simulations is that GPU-based parallel computing solutions for VANETs should be easily integrable into commonly used network simulators.

In this paper, we review the state-of-the-art GPU-based parallel computing solutions for VANETs classifying them into four different categories, viz., 1) Techniques for optimizing travel time, 2) Techniques for security enhancement, 3) Techniques for accelerating simulation, and 4) Techniques for other problems. We further identify the limitations of these existing works and recommend future directions toward further usage and optimization of GPU-based computing for VANETs. To the best of our knowledge, this is the first review on GPU-based computing for VANETs, and we believe it will serve as a motivation towards further exploration of GPU-based parallel computing for VANETs. Figure 1 presents the scope and taxonomy of this paper.

The rest of this paper is organized as follows: Section II reviews the key concepts of GPUs along with their challenges and optimization strategies; Section III discusses the GPU-based techniques for VANETs according to the above taxonomy; Section IV provides the directions for future research, and finally, Section V offers the conclusions.

II. GPU and its Challenges

In this section, we present the required background material on GPUs to follow the rest of the paper and the challenges to address for achieving optimal performance in GPU-based parallel computing.

GPUs are highly parallel co-processors available in most devices, ranging from smartphones to supercomputers, for graphics rendering. It is possible to utilize the large-scale

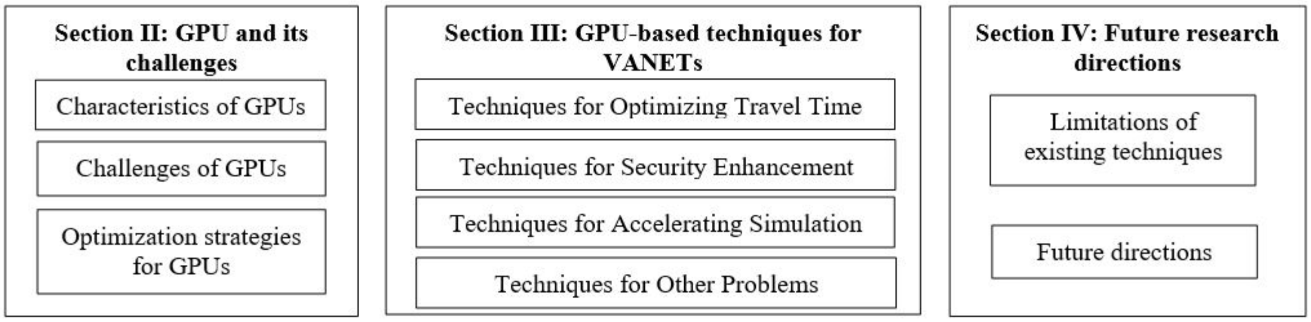
Section II: GPU and its challenges
Characteristics of GPUs
Challenges of GPUs
Optimization strategies for GPUs
Section III: GPU-based techniques for VANETs
Techniques for Optimizing Travel Time
Techniques for Security Enhancement
Techniques for Accelerating Simulation
Techniques for Other Problems
Section IV: Future research directions
Limitations of existing techniques
Future directions

Fig. 1. The above figure shows the scope and taxonomy of this paper.

parallelism of GPUs for general-purpose computing. GPUs work on the Single Instruction Multiple Data model for parallel processing that exploits data-level parallelism by performing a single operation on multiple data elements concurrently. GPUs are connected to the CPU through a relatively low throughput interface like a PCIe bus. In this paper, we use the terminologies of CUDA, an extension to C/C++ for GPU programming [3]. Another commonly used programming framework for GPUs is OpenCL (Open Computing Language) [4]. Functions called kernel [3] are called from the CPU to execute on the GPUs. Calling a kernel launches a grid of concurrent GPU threads, which are sub-grouped into blocks. GPUs and CPUs have different memory spaces. So, it is required to copy the input data from CPU memory to GPU memory for processing on GPUs. Similarly, the results of computation need to be copied back from GPU memory to CPU memory. The memory of GPUs has a hierarchical organization. Each thread has its private registers, the threads in a block cooperate by using a block-wide shared memory, and all the threads across different blocks have access to a slower but much larger global memory.

To optimally exploit the parallelism of GPUs, it is imperative to address the challenges of this architecture. As the threads in a block share a faster shared memory, to exploit this feature, a common approach is to copy portions of data from global memory to shared memory in a phased manner and process them. It reduces access to slower global memory. Every time there is global memory access, a huge number of threads perform a memory read. To deal with the slowness of global memory, GPUs have caches that can exploit the spatial locality of global memory accesses to reduce traffic through the memory controller. To take advantage of such caches, GPU threads need to access the global memory following patterns that respect the spatial locality, i.e., consecutive threads access adjacent memory locations. This is called global memory coalescing [5].

Another challenge is the low throughput CPU-GPU interface. To address this, it is required to optimize the amount of communication through the CPU-GPU interface. This can be achieved by reorganizing the input data in CPU memory if necessary and transferring only the necessary data to the GPU. Balancing the number and size of chunks for data transfer is required as transferring data in a small number of large chunks may not be enough to keep the GPU busy, while a large number of small chunks is not optimal due to the overhead associated.

Load balancing is another critical issue, which refers to ensuring that all the threads of GPU perform a similar amount of work so that the execution time of a single thread doing more work does not end up slowing the execution time of an entire kernel function. Overlapping data transfer between CPU and GPU with computations on GPU is another optimization strategy. In this, one chunk of data can be transferred from CPU to GPU and a kernel can be launched to work on that; while the GPU performs the computations on that chunk, another chunk can be transferred from the CPU. In the same manner, GPU to CPU data transfer can also be overlapped with computations on GPU. The degree of parallelism is another challenge in obtaining the optimal performance of GPUs as launching of threads incurs some overhead. Launching too many threads than necessary may increase the overhead and thus nullify a portion of the performance gain obtained. On the other hand, launching an insufficient number of threads may not provide the ideal feasible speedup.

III. GPU-BASED TECHNIQUES FOR VANETs

In this section, we discuss the GPU-based parallel computing solutions for VANETs, according to the taxonomy shown in Figure 1, in the following subsections: A. Techniques for optimizing travel time, B. Techniques for security enhancement, C. Techniques for accelerating simulation, and D. Techniques for other problems. Finally, we provide a summary of the reviewed techniques in Table I.

A. Techniques for Optimizing Travel Time

Efficient routing of vehicles is important in VANETs to enhance driving experiences by reducing travel time. Finding the shortest and congestion-free routes and minimizing the wait time at intersections are some common approaches to this. Several works consider the GPU-based parallelization of certain serial algorithms to reduce travel time. However, all these techniques discussed in this subsection are parallelization of some existing serial algorithms; none of them is designed to specifically target the parallel architecture of GPU, which is desired to obtain the full benefits of GPU-based parallelization.

Among the approaches that focus on shortest and congestion-free routes, a GPU-based parallel implementation of the Modified Ant Colony Optimization (MACO) algorithm [6], CUDA_MACO is one. MACO algorithm reduces travel time by avoiding congested routes during vehicle routing. However, when the number of vehicles in the network increases, the processing time of MACO also increases resulting in an increase in travel time. To address this, CUDA_MACO is proposed where all the phases of MACO are parallelized on GPU.

Another algorithm for travel time reduction by avoiding congested routes, GHAPO-P (GPU accelerated Preemptive Hybrid Ant Particle Optimization) is presented in [8]. GHAPO-P is again a GPU-based parallelization of the algorithm HAPO-P [9] which reduces travel time by avoiding enroute congestion combined with preemptive traffic control. GHAPO-P reduces processing time with parallel processing compared to HAPO-P.

IHAPO-G [10] is another algorithm for travel time reduction through congestion avoidance which parallelizes the IHAPO algorithm. IHAPO combines MACO and particle swarm optimization [11] for travel time reduction, and IHAPO-G reduces the travel time further by using parallel processing on GPU.

The algorithm proposed in [12] deals with concurrently finding the shortest path for all vehicles to their destinations. This is achieved through a parallel implementation of Hedetniemi's algorithm for solving the all-pairs shortest path problem. Hedetniemi's algorithm repeatedly computes the powers of an adjacency matrix until two consecutive matrix powers match. The parallel algorithm speeds up the matrix multiplication process using GPU to reduce the processing time.

A GPU-based parallel genetic algorithm for the Travelling Salesman Problem (TSP) is proposed in [13]. The algorithm can be integrated into a vehicular cloud computing (VCC) model and be used for making routing decisions in a traffic management system. The authors first propose a methodology for parallelizing the main operations of a genetic algorithm, then apply it to solve the TSP, and finally suggest integration with VCC for traffic management.

Another algorithm based on MACO is MACO-P (Preemptive MACO), which also considers the traffic signals in addition to finding an optimal route and thus improving the travel time. A parallel version of this algorithm GMACO-P (GPU-assisted Preemptive MACO) is proposed in [14] parallelizing the various phases of MACO-P on a GPU.

The parallel preemptive algorithm in [15] and the parallel adaptive traffic control algorithm in [16] reduce the average queue length at intersections, thus minimizing the overall waiting time. These algorithms parallelize their serial counterparts that use the current vehicle density at a given time for selecting the next phase for the allocation of green light time. The parallel versions run all the subparts of the serial algorithms on GPUs, thus reducing the computation time to make quicker decisions for green light time allocation.

A graph-enabled Intelligent Vehicular Network (IVN) data processing framework, Hooker is proposed in [17] which explores using several graph algorithms for addressing path planning and intersection management for IVN. Various graph-based optimizations are used in this work for optimizing GPU performance. The optimizations facilitate coalesced GPU global memory access and overlapping of CPU-GPU communication with computation on GPU.

Except for the work in [17], none of the techniques discussed in this subsection adopt any strategy for performance optimization of GPUs, thus missing opportunities for possible additional speed-up. Moreover, none of them suggest the degree of parallelism to consider, i.e., how many parallel threads should be launched in GPUs for the optimal performance of the algorithm. This is critical in any parallel programming model as launching more threads than necessary degrades the performance due to the associated overhead. Also, none of these techniques are integrated with the commonly used network simulators; such integration can be beneficial for newer applications of these techniques in VANET research.

B. Techniques for Security Enhancement

Security is one of the major challenges in the deployment of VANETs. Security is the state of keeping the network free from threats or attacks. It is vital to ensure the security of VANETs adequately, as VANETs are highly mobile and dynamic networks where the topology changes frequently. In this subsection, we discuss a group of works that address security enhancement in VANETs using GPUs. Nevertheless, these techniques make minimal or no use of optimization strategies for performance enhancement of GPUs by addressing their challenges.

Among the GPU-based security enhancement solutions for VANETs, in [18], a cryptographic hardware-acceleration framework HAA is designed for Rapid Authentication (RA), a delay-aware offline-online signature scheme for command and control systems. HAA uses the Chinese Remainder Theorem and Montgomery Multiplication algorithm to enable parallelism on GPUs. In the proposed GPU-based parallelization, coalesced memory access for GPU global memory is ensured to optimize the performance.

A hybrid approach that simultaneously uses GPU-based and CPU-based parallelization for Elliptic Curve Digital Signature Algorithm (ECDSA) for vehicle signature verification in RSU is presented in [19]. At the CPU end, OpenSSL-based ECDSA verification is performed in each thread, and one signature is verified at a time; on the GPU end, multiple signatures are verified simultaneously.

The work in [20], proposes a hybrid trust-based system MobileTrust for secure resource sharing in VANETs. Its goal is to evaluate an entity’s behavior after establishing cryptographically secure communication. The system is a centralized trust computing system that uses cloud and 5G technologies for trust establishment. GPU-based parallelization is adopted in this system for parallel data collection for different road segments.

Amongst the above three techniques discussed in this subsection, only HAA uses some optimization strategy for GPU by using coalesced global memory access. None of the techniques analyzes the impact of varying the degree of parallelism, nor provides any guidelines on that. Further, these techniques do not provide any integration with the network simulator tools.

C. Techniques for Accelerating Simulation

Evaluating the performance of a network is important to investigate any issues that may exist in the network and VANETs are no exception. The most suitable way for this task is to deploy simulations that provide the closest results to real-world observations. Various simulation tools are used to simulate and evaluate the performance of VANETs. Parallel processing can be useful in reducing the time required for carrying out simulation studies. Although there exists minimal support for multicore parallelization in some commonly used simulators, there is no support for GPU-based parallelization. Hence, some approaches using GPU-based parallelization are proposed to accelerate VANET simulations, which are discussed in this subsection. The issue however is that these approaches do not address the challenges of the GPU architecture.

A GPU-accelerated VANET simulation tool using the NS-3 simulator is proposed in [21] where the mobility module of NS-3 is modified to integrate with GPU using CUDA. Specifically, all pairwise Euclidean distance computation between the nodes on a highway is parallelized on GPUs, where one GPU thread is allocated for computing the distance between a pair of nodes.

A GPU-based radio propagation simulation method, based on ray-launching simulation is introduced in [21] which deterministically models V2V communication channels. In this two-phase method, the first phase uses parallel ray launching to accelerate the visibility computations configuring a scene. The computed propagation paths between transmitters and receivers are then passed to the second phase where electromagnetic properties are computed for each propagation path. In both phases, parallel processing on GPU is used to accelerate the computations.

A vehicular network simulation framework, Veneris, is proposed in [23] which is implemented on top of the Unity game engine that includes a realistic vehicle model and a set of driving and lane change behaviors replicating traffic dynamics. A GPU-based ray-launching propagation simulator, Opal is used in Veneris for reducing the time for ray launching.

None of the techniques reviewed in this subsection discusses the degree of parallelism to be adopted for optimal performance, nor do they use any optimization strategy for GPUs. While the GPU-accelerated VANET simulator proposed in [21] is limited to only a single module, the remaining two proposals focus on accelerating very specific simulation scenarios rather than a more generic accelerated simulation tool that can be used in diverse simulations.

D. Techniques for Other Problems in VANETs

Some other GPU-based parallel computing techniques are addressing different discrete problems for VANETs. Those techniques are discussed in this subsection. Among these techniques, the techniques proposed in [24] and [25] use some form of optimization strategy for GPUs by using the shared memory of GPUs. Beyond this, the work in [26] has explored the use of multiple GPUs to achieve higher speed-up through parallelization.

In [24], a greedy heuristics algorithm is proposed to schedule vehicular tasks in various vehicular clouds forming a VCC system, with opportunistically available V2I communication. A vehicular task placement scheme is formulated as a Markov Decision Process (MDP) in this work, wherein the iteration value algorithm used to compute the optimal decisions of the MDP is accelerated on GPUs with a parallel algorithm, Blocks of State Divided Iteration (BSDI).

GPU-based parallelization of packet classification for SDN (Software-Defined Networking)-based Internet of Vehicles (IoV) is proposed in [25]. The Tuple Space Search (TSS) and Pruned Tuple Space Search (TPS) algorithms for packet classification are parallelized in this work considering their highly parallelizable structure.

The work in [26] presents a three-dimensional universal cooperative localizer (3D UCL) for VANETs in 3D space under varied types of ranging measurements. A parallel version of 3D UCL using GPUs is implemented to reduce the processing time of the serial version.

A metaheuristic optimization model, based on particle swarm optimization, addressing the problem of low service quality due to vehicle movements and limited edge coverage in a vehicular edge computing setting is discussed in [27]. The proposed GPUPSO algorithm uses GPU-based parallelism to accelerate certain computations of the optimization model.

None of the above four techniques discusses the degree of parallelism to be adopted for optimal performance. Although the techniques in [24] and [25] have used shared memory of GPUs to optimize the performance, the techniques in [26] and [27] do not use any optimization strategy for GPUs. Moreover, none of these techniques are integrated with network simulator tools.

To conclude this section, in Table I below, we present a summary of the techniques discussed in this paper in terms of the problem they address, optimization strategies adopted for GPUs, and implementation tools used (this also specifies whether any integration is made with a network simulator).

TABLE I. SUMMARY OF THE GPU-BASED PARALLEL COMPUTING TECHNIQUES FOR VANETS

Classification	**Proposal**	**Problem Addressed**	**Optimization for GPUs**	**Implementation Tools**
Optimizing Travel Time	CUDA_MACO [7]	Travel time reduction by avoiding congested route	-	CUDA toolkit 7.0, C/ C++ on Visual Studio 2010
	GHAPO-P [8]	Travel time reduction by avoiding enroute congestion combined with preemptive traffic control	-	CUDA toolkit 7.5, C on Visual Studio 2013
	IHAPO-G [10]	Travel time reduction by avoiding congested route	-	CUDA toolkit 7.5, C on Visual Studio 2013
	GPU-based Hedetniemi's algorithm [12]	Travel time reduction by finding all pairs shortest path	-	OpenCL
	Parallel genetic algorithm for TSP [13]	Making vehicle routing decisions for traffic management	-	CUDA toolkit 8.0 (V8.0.61), C++
	GMACO-P [14]	Travel time reduction by	-	CUDA toolkit 7.5,

Classification	Proposal	Problem Addressed	Optimization for GPUs	Implementation Tools
		avoiding congested routes and reducing traffic signal wait time		C on Visual Studio 2013
	Parallel preemptive traffic control algorithm [15]	Wait-time minimization by reducing the queue length at intersections	-	CUDA toolkit 7.5, C on Visual Studio 2013
	Parallel adaptive traffic control algorithm [16]	Wait-time minimization by reducing the queue length at intersections	-	CUDA toolkit 7.5, C on Visual Studio 2013
	Hooker [17]	Vehicle path planning, intersection management	Coalesced memory access, CPU-GPU Communication, and computation overlap	CUDA 8.0
Security Enhancement	HAA [18]	Accelerated rapid authentication	Coalesced memory access	CUDA
	Parallel ECDSA [19]	Vehicle signature verification in RSU	-	OpenCL
	MobileTrust [20]	Trust-based resource sharing	-	SUMO, C++, CUDA
Accelerating Simulation	Accelerated NS-3 [21]	Accelerating VANET simulation in NS-3 simulator	-	NS-3, CUDA
	GPU-based ray-launching simulation [22]	Simulation of V2V communication channel	-	CUDA 6.5, OptiX 3.7
	Veneris [23]	Simulation of cooperative automated driving	-	Unity game engine, Opal, OMNET++
Other Techniques	BSDI [24]	Vehicular task placement in distributed vehicular clouds	Use of GPU's shared memory	CUDA
	Parallel TSS and TPS [25]	Packet classification for SDN-based IoV	Use of GPU's shared memory	CUDA 9, C
	Parallel 3D UCL [26]	Cooperative positioning to recover vehicle positions	-	CUDA Toolkit v.8.0
	GPUPSO [27]	Low service quality in vehicular edge computing	-	CUDA

IV. Directions for Future Research

Although there is a proliferation of works using GPU-based parallel computing in diverse fields, as we find in section III, only a limited number of such works exist in the literature for VANETs. Besides, there are other limitations in these works, such as not adopting the strategies for gaining optimal performance of GPUs, lack of integration of GPUs with network simulators, etc. We believe that effectively addressing these issues will enable the research community to realize the true potential of parallel computing powers of GPUs in VANET research. It is also noteworthy that at the hardware level, several proposals [28-30] consider using devices such as smartphones as OBUs. As smartphones are already equipped with GPUs, the computing capabilities of the GPUs can be used for parallel computing at the OBU. Some other works [31-33] explore the use of GPU-based RSUs. Based on our observations in this review, we next recommend some directions for future research to meet the prospects of using GPUs for VANETs.

A. Optimization Strategies for GPUs

As seen from Table I, the optimization strategies for achieving the maximum potential of GPU-based parallelization are sparingly used in the existing methods for VANETs, although such optimizations can lead to significant performance benefits. The experimental evaluations in [25] are a confirmation of this. Effectively adopting these optimization strategies to address the unique characteristics of this architecture will enable realizing the true capabilities of GPU-based parallel computing in VANET research.

B. Appropriate Degree of Parallelism

The existing GPU-based methods for VANETs do not specify the degree of parallelism to be adopted, nor do they experiment with the varying numbers of GPU threads in the experimental evaluations for the proposed approaches. Appropriate evaluations of a parallel technique with varying degrees of parallelism will enable determining the correct degree of parallelism for different problem sizes. Thus, a GPU-based technique will be more amenable to real-life applications based on the amount of data an application needs to process.

C. Integration of GPUs with Network Simulators

There are very limited attempts toward integrating GPUs with commonly used network simulator tools. We believe that such integrations will motivate the researchers to explore GPU-based solutions for VANETs further. Moreover, it can encourage exploring GPU-based parallelization for various other problems in VANETs by providing support for simulation-based evaluations. For instance, there is no attempt at GPU-based parallelization for some extensively studied problems in VANETs, such as packet routing, clustering, etc.

D. GPU-based Machine Learning Solutions for VANETs

Numerous solutions for different problems in VANETs use machine learning (ML) techniques. GPUs have been widely used by the ML community and different ML libraries have been developed targeting the GPU architecture. The ML techniques reliant solutions for VANETs can make use of the existing GPU-based ML libraries towards developing GPU-based solutions for VANETs. Additionally, new solutions for various problems in VANETs can be designed based on the available GPU-based ML techniques.

E. Exploration of Multi-GPU Solutions

Another potential area to investigate further is the use of multiple GPUs together for additional performance improvement through parallelization. As shown by the experimental evaluations in [26], using two GPU cards provides better performance for their proposed technique. However, in doing so, the balance between the amount of data to be processed in GPUs and the number of GPUs to be used still needs to be maintained.

V. Conclusions

To conclude, in this paper, we reviewed the state-of-the-art GPU-based parallel computing techniques that address different problems in VANETs presenting a taxonomy for them. We further identified the limitations in these approaches and based on that recommended several future directions for exploring and exploiting GPU-based parallel computing for VANETs including the possibility of addressing previously untouched problems with GPUs.

References

[1] M. Bayat, M. Pournaghi, M. Rahimi, and M. Barmshoory, "NERA: a new and efficient RSU based authentication scheme for VANETs," Wireless Networks, 26, 2020, pp. 3083–3098.

[2] H. Shahwani, S. A. Shah, M. Ashraf, M. Akram, J. P. Jeong, and J. Shin, "A comprehensive survey on data dissemination in Vehicular Ad Hoc Networks," Vehicular Communications 34, 2022, 100420.

[3] "CUDA C++ Programming Guide" NVIDIA Corporation, 2023. [Online]. Available at: https://docs.nvidia.com/cuda/cuda-c-programming-guide/. [Accessed July 5, 2023]

[4] https://www.khronos.org/api/opencl [Accessed July 5, 2023]

[5] "CUDA C++ Best Practices Guide," NVIDIA Corporation, 2023. [Online]. Available at: https://docs.nvidia.com/cuda/cuda-c-best-practices-guide/. [Accessed July 5, 2023]

[6] V. Jindal, H. Dhankani, R. Garg, and P. Bedi, "MACO: Modified ACO for reducing travel time in VANETs," in Proceedings of the Third International Symposium on Women in Computing and Informatics, 2015, pp. 97-102.

[7] V. Jindal and P. Bedi, "Reducing travel time in VANETs with parallel implementation of MACO (modified ACO)," in Innovations in Bio-Inspired Computing and Applications: Proceedings of the 6th International Conference on Innovations in Bio-Inspired Computing and Applications (IBICA 2015) held in Kochi, India during December 16-18, 2015, pp. 383-392. Springer International Publishing, 2016.

[8] V. Jindal, and P. Bedi, "GPU accelerated preemptive HAPO (GHAPO-P) for reducing trip time in VANETS," in International Conference on Advances in Computing, Communications and Informatics (ICACCI), pp. 1569-1575. IEEE, 2017.

[9] V. Jindal, V. R. Sharma, and P. Bedi, "A preemptive hybrid ant particle optimization (HAPO-P) algorithm for smart transportation." In International Conference on Advances in Computing, Communications and Informatics (ICACCI), pp. 1357-1363. IEEE, 2016.

[10] V. Jindal, and P. Bedi, "An Improved HAPO algorithm using GPU harness (IHAPO-G) for rapid responses in VANETs." in International Conference on Recent Innovations in Electrical, Electronics & Communication Engineering (ICRIEECE), pp. 1150-1155. IEEE, 2018.

[11] J. Kennedy and R. C. Eberhart, "Particle Swarm Optimization," in International Conference on Neural Networks, IEEE, Piscataway, NJ, 1995, pp. 1942-1948.

[12] S.A. M. Ali, and E. H. Al-Hemiary, "A Parallel Implementation Method for Solving the Shortest Path Problem for Vehicular Networks," in 1st International Conference of Information Technology to Enhance e-learning and Other Application (IT-ELA), pp. 121-126. IEEE, 2020.

[13] M. Abbasi, M. Rafiee, M. R. Khosravi, A. Jolfaei, V. G. Menon, and J. M. Koushyar, "An efficient parallel genetic algorithm solution for vehicle routing problem in cloud implementation of the intelligent transportation systems," Journal of Cloud Computing 9, 2020, pp. 1-14.

[14] V. Jindal, and P. Bedi. "GMACO-P: GPU assisted Preemptive MACO algorithm for enabling Smart Transportation," arXiv preprint arXiv:2010.14244, 2020.

[15] V. Jindal, and P. Bedi, "Reducing waiting time with parallel preemptive algorithm in VANETs," Vehicular Communications 7, 2017, pp. 58-65.

[16] V. Jindal, and P. Bedi, "High performance adaptive traffic control for efficient response in vehicular ad hoc networks," International Journal of Computational Science and Engineering 16, no. 4, 2018, pp: 390-400.

[17] Z. Zheng, and A. K. Bashir, "Graph-enabled intelligent vehicular network data processing," IEEE Transactions on Intelligent Transportation Systems 23, no. 5, 2022, pp. 4726-4735.

[18] A. Singla, A. Mudgerikar, I. Papapanagiotou, and A. A. Yavuz, "Haa: Hardware-accelerated authentication for internet of things in mission critical vehicular networks," in IEEE Military Communications Conference, pp. 1298-1304. IEEE, 2015.

[19] S. Lee, H. Seo, H. Kwon, and H. Yoon, "Hybrid approach of parallel implementation on CPU–GPU for high-speed ECDSA verification," The Journal of Supercomputing 75, 2019, pp. 4329-4349.

[20] G. Hatzivasilis, O. Soultatos, S. Ioannidis, G. Spanoudakis, V. Katos, and G. Demetriou, "MobileTrust: Secure knowledge integration in VANETs," ACM Transactions on Cyber-Physical Systems 4, no. 3, 2020, pp. 1-25.

[21] S. Yip, M. Chok, and A. Asaduzzaman, "A promising CUDA-accelerated vehicular area network simulator using NS-3," in IEEE 33rd International Performance Computing and Communications Conf., 2014, pp. 1-2.

[22] M. Schiller, A. Knoll, M. Mocker, and T. Eibert, "GPU accelerated ray launching for high-fidelity virtual test drives of VANET applications," in 2015 International Conference on High-Performance Computing & Simulation (HPCS), pp. 262-268. IEEE, 2015.

[23] E. Egea-Lopez, F. Losilla, J. Pascual-Garcia, and J. M. Molina-Garcia-Pardo, "Vehicular networks simulation with realistic physics," IEEE Access 7, 2019, pp. 44021-44036.

[24] Y. Maalej, A. Abderrahim, M. Guizani, and B. Hamdaoui," CUDA-accelerated task scheduling in vehicular clouds with opportunistically available V2I," in IEEE International Conference on Communications (ICC), 2017, pp. 1-6.

[25] M. Abbasi, A. Najafi, M. Rafiee, M. R. Khosravi, V. G. Menon, and G. Muhammad," Efficient flow processing in 5G-envisioned SDN-based Internet of Vehicles using GPUs," IEEE Transactions on Intelligent Transportation Systems 22, no. 8, 2020, pp. 5283-5292.

[26] S. Wang, and X. Jiang, "Three-dimensional cooperative positioning in vehicular ad-hoc networks," IEEE Transactions on Intelligent Transportation Systems 22, no. 2, 2020, pp. 937-950.

[27] M.A. Alqarni, M. H. Mousa, and M. K. Hussein, "Task offloading using GPU-based particle swarm optimization for high-performance vehicular edge computing", Journal of King Saud University-Computer and Information Sciences 34, no. 10, 2022, pp. 10356-10364.

[28] M.A. Abdel-Aty, Q. Cai, S. Agarwal, Z. Islam, P. Li, S. Zhang, D. Hasan, and J. Huang, "Using Smartphone as On-board Unit (OBU) Emulator Implementation Study," 2020.

[29] X. Li, Y. Feng, F. Wang, and Q. Qian. "When smart phone meets vehicle: A new on-board unit scheme for VANETs," in IEEE International Conference on Computer and Information Technology; Ubiquitous Computing and Communications; Dependable, Autonomic and Secure Computing; Pervasive Intelligence and Computing, 2015, pp. 1095-1100.

[30] K. Su, H. Wu, W. Chang, and Y. Chou, "Vehicle-to-vehicle communication system through wi-fi network using Android smartphone," in International conference on connected vehicles and expo (ICCVE), 2012, pp. 191-196.

[31] G.A.G. Castillo, E. Bonetto, D. Brevi, F. Scappatura, A. Sheikh, and R. Scopigno, "Latency assessment of an ITS safety application prototype for protecting crossing pedestrians," in 91st Vehicular Technology Conference, 2020, pp. 1-5.

[32] R. Ojala, J. Vepsäläinen, J. Hanhirova, V. Hirvisalo, and K. Tammi, "Novel convolutional neural network-based roadside unit for accurate pedestrian localization," IEEE Transactions on Intelligent Transportation Systems 21, no. 9, 2019, pp. 3756-3765.

[33] D. Weimer, S. Köhler, C. Hellert, K. Doll, U. Brunsmann, and R. Krzikalla, "GPU architecture for stationary multisensor pedestrian detection at smart intersections," in IEEE Intelligent Vehicles Symposium, 2011, pp. 89-94.