International Journal of Electrical and Computer Engineering (IJECE)
Vol. 12, No. 1, February 2022, pp. 748~755
ISSN: 2088-8708, DOI: 10.11591/ijece.v12i1.pp748-755

Parallel genetic approach for routing optimization in large ad hoc networks

Hala Khankhour[1], Otman Abdoun[2], Jâafar Abouchabaka[1]
[1]Department of Computer Science, Faculty of Sciences, Ibn Tofail University, Kenitra, Morocco
[2]Department of Computer Science, Faculty of Polydisciplinary, Abdelmalek Essaadi University, Larache, Morocco

Article Info

Article history:

Received Oct 23, 2020
Revised Jul 31, 2021
Accepted Aug 19, 2021

Keywords:

Ad hoc
Artificial intelligence
Genetic algorithm
Np-complete
Parallel computer

ABSTRACT

This article presents a new approach of integrating parallelism into the genetic algorithm (GA), to solve the problem of routing in a large ad hoc network, the goal is to find the shortest path routing. Firstly, we fix the source and destination, and we use the variable-length chromosomes (routes) and their genes (nodes), in our work we have answered the following question: what is the better solution to find the shortest path: the sequential or parallel method? All modern systems support simultaneous processes and threads, processes are instances of programs that generally run independently, for example, if you start a program, the operating system spawns a new process that runs parallel elements to other programs, within these processes, we can use threads to execute code simultaneously. Therefore, we can make the most of the available central processing unit (CPU) cores. Furthermore, the obtained results showed that our algorithm gives a much better quality of solutions. Thereafter, we propose an example of a network with 40 nodes, to study the difference between the sequential and parallel methods, then we increased the number of sensors to 100 nodes, to solve the problem of the shortest path in a large ad hoc network.

Corresponding Author:

Hala Khankhour
Department of Computer Science, Faculty of Science, Ibn Tofail University
B.P 859, Main Post 92000, Larache, Morocco
Email: hala.khankhour@uit.ac.ma

1. INTRODUCTION

In ad-hoc networks, one of the most problems is routing in quick time, which has a significant impact on the network's performance, so the best routing should find an optimum path in a specified time to satisfy the quality of service from the source node to destination node [1], the field use extensively the mathematical reasoning (logic, probabilities, data analysis) and process modeling, but it is very difficult to solve the complex problems by the exact methods [2]. For that reason, we use the meta-heuristic [3], among these methods are the genetic algorithm (GA) that will give good results, we will explain them in detail in the following sections. In general, GA is very reliable and pushes to solve very high complexity problems, but they take a long time to find the best solution, therefore the global search method can obtain accurate approximate results. However, its computational cost is fairly high because it carries out an exhaustive search of solutions [4]. Consequently, in recent years, researchers want to speed up the operation of GA, so they tried to combine two or more independent processors in the same computer; they can create a multicore processor. Today, all modern operating systems support concurrency across processes and threads, processes are instances of programs that generally run independently of one another, e.g: if you start a java program, the operating system spawns a new process that runs in parallel with other programs; within these processes, we can use threads to execute code simultaneously, so that we can make the most of the available cores of the

central processing unit (CPU), for this reason, we propose the adequate recommendations is to use parallel implementations. In that matter, the concurrency application programming interface (API) was first introduced with java 5, and gradually improved with each new version of Java, the majority of concepts also work for highly complex problems of the NP-complete class using thread class can cause a lot of errors, for this reason, new version of java 5 introduced concurrency API [5]. The API is in the java *package.util.concurrent* and contains many powerful classes to manage simultaneous programming, since then, the concurrency API has been improved with each new version of Java, and even Java 8 provides new classes and methods for dealing with concurrency. So, the main current existing alternative is to use parallel architectures, specifically for highly complex problems of the NP-Complete class. Graphics processing units (GPUs) offer attractive performance to energy consumption and the cost of purchase ratio and allow performing many types of computations more quickly while maintaining the same cost concerning the CPUs. On the other hand, the utility of GPUs [6] is evidenced by the fact that they are used in approx. 10% of the fastest supercomputers in the world [7]. This paper focuses on GA in the ad hoc network, GA is efficient for our problem, based on principles of natural selection [8], they are being applied successfully to find acceptable solutions to problems in business, engineering, and science [4]. Our challenge is to change the design of meta-heuristics GA with parallelism, to take advantage of GPU for solving large-scale complex problems in ad hoc network with a view to high effectiveness and efficiency in a short time. The remainder of this paper is planned according to this plan. Section 2 presents the problem studied: "routing problems in ad hoc networks". In sections 3 and 4, the parallelism with a GA, and its implementation details are presented in section 4, in section 5, the simulated result is discussed. The paper is concluding in section 6.

2. PROBLEM STUDIED "AD HOC NETWORKS ROUTING OPTIMIZATION PROBLEM"

For several years, mobile multi-user wireless ad-hoc networks have attracted the attention of scientists, network performance depends on the routing protocol, delay, energy consumption, quality of service, and the path chosen which has a vital role in delivering the message. However, there are problems for the network of large maps [9], the simulations used are limited to 50 nodes at most [10]-[13]. Therefore, our problem is to find the shortest path from the source to the destination by visiting the neighboring city (n) only once, for example, in the traveling salesman problem (TSP), the traveler wants to sell a product, and wants to know the best path to minimize time and supplies, how should he plan his way for a minimum total cost of n cities? It is impossible to solve a big map ad hoc network problem by the exact traditional methods because it is considered as an NP-complete problem [14]; meta-heuristics have proven their performance in solving NP-complete problems, such as GA initiated by Charles Darwin [2]. The principle of GA is directly inspired by the laws of natural selection, it is defined by chromosome/sequence, population, and fitness function. Our work is the complement of these previous works [15], [16], last year, researchers are studying the stopping criterion of the GA because the search time in the space of solutions is one of the very important factors to find the global optimal solution [17], hence, the multi-hop network topology can be illustrated by a graph F=(N, K), where N is the number of nodes and K the link between the nodes, there is a cost C_{ij} associated with each link (i,j), the cost of the path between the source and destination is specified by the cost matrix C=$[C_{ij}]$ by the (2), and every link noted by the (1) I_{ij}.

$$I_{ij} = \begin{cases} 1, & if\ the\ link\ between\ i\ and\ j\ exists\ in\ the\ routing\ path \\ 0, & otherwise \end{cases} \quad (1)$$

$$C = \sum_{j=1}^{i \neq j} C_{gi(j).gi(j+1)} \quad (2)$$

The cost calculation takes a long time if we increase the number of nodes, which means the size of the network has become larger. For this reason, we propose a hybridization between GA and parallel architectures to end the search process quickly, especially for large map network, our results showed that the use of the parallel method, is powerful in this case of our problem [18].

3. IMPROVING OPTIMIZATION PERFORMANCE WITH PARALLEL COMPUTING TECHNIQUES

The advantage of paralleling GA is to gain computing time, there are at least two conventionally used methods for this (we can refer to this article [19] for more details). Firstly, Data parallelism: occurs when many data items can be activated at the same time, it focuses on the distribution of data across multiple

cores, it can use to speed up calculations. Secondly, the parallelism of tasks: occurs when there are many tasks or functions which can be operated independently and largely in parallel, it focuses on the distribution of functions between several hearts, this is used to decrease the latency [20]. We will focus on the second, since the given problem is more suitable to be implemented using the method of parallelism of the tasks. However, the multiplication of the number of calculation units does not spontaneously reduce the execution time of programs. On the other hand, parallel programming makes it possible mainly to reduce the execution time, in theory, if we divide a task into N processors, the time will be reduced N times, but in reality, there are other factors that can determine the speed of execution of a parallel program, the time of transmission of messages between the processes, and also, the parts in all programs that cannot be parallelized [4]; we add that parallel programming is used and applied in almost all areas as an example we find: treatment of scientific problems constituting major challenges, commercial applications parallel databases, web search engines, dealing with larger and more complex problems and exploit current architectures.

3.1. Parallel computing with openMP and with MPI

OpenMP is the distribution of computational loads to several light processes called threads, the purpose of parallel calculation is to decrease the execution time of the program, a program is executed by a single task (process), by entering a parallel region, this task activates several "subtasks", each thread is executed on a processor core, a sequential task is executed by the master thread [21]. The message passing interface (MPI) is a distribution of computational loads on several processor cores, called processes, in order to reduce the execution time of the program, these processors can belong to different machines linked by a communication network which allows the exchange of data by "message passing". So MPI and openMP embody two different approaches to parallel programming, by their different material abstractions, and each model is suitable for a type of architecture. Indeed, the message passing model is suitable to multi-computer type architectures (processors each having their memory and connected by a network) and the shared memory model is appropriate to multiprocessor type architectures (processors accessing the same memory via a bus). In addition, in the message passing model the parallel processes remain active throughout the execution of the program while in the shared memory model the thread is activated at the start of the program and this number can change dynamically throughout of execution [22]. In view of the advantages of the multiprocessor parallel, we use this technologie with GA to optimize our problem in ad hoc.

4. PROPOSED APPROACH: PARALLELISM GENETIC ALGORITHMS IN AD HOC NETWORK (PGA)

GA is an effective research method based on the principles of natural selection and genetics. They are successfully applied to find acceptable solutions to problems in business, engineering, and science; in general, GA is powerful specifically for highly complex problems of the NP-Complete, but each time we increase the size of the network, we use more the CPU resources. For this reason, we thought of the use of parallel programs, the basic idea of most parallel programs is to split a task into several pieces and to solve the pieces simultaneously using several processors, this rule approach can be applied to GA in different ways. In the literature we found many examples of successful implementation, some methods use a single population, while others divide the population into several relatively parallel subpopulations, also there are more suitable for multi-computers with fewer and more powerful processing elements [19]. There are three main types of parallel GA: master salves single-world GA, fine-grained single-population GA, and multi-population GA. In a master-slave GA, there is only one population, but the evaluation of physical fitness is distributed among several processors, given that in this type of parallel GA, selection and crossing takes into account the whole population. Parallel GA is suitable for massively parallel computers and consists of a structuring the population, selection and mutation are limited to a small neighborhood, but the neighborhoods overlap, allowing some interaction between all individuals, so the ideal case is to have one individual for each available treatment item, and the multi-population GA is more sophisticated because they are consisted of several sub-populations that exchange individuals occasionally [2], [23], [24]. So, in this article, firstly we propose an algorithm to find the best routing path as shown in Figure 1.

In this sequence, to select the best path we have to repeat the steps previously several times, and it takes a lot of time. That's why we took advantage of multithreading java and each thread does the same steps mentioned above starting at the same time. After a certain number of iterations, all the threads finish their tasks and give the best path found, as shown in Figure 2.

One of the most steps in measuring the performance of each individual is to calculate the Fitness, to be able to judge the quality of an individual and compare it to others, which are defined in (3):

$$fi = \frac{1}{\sum_{j=1}^{li-1} C_{gi(j),gi(j+1)}} \quad (3)$$

where $f\,i$ represents the fitness cost of the xi chromosome, li is the length of the xi chromosome, $gi(j)$ describes the gene of the xj position in the xi chromosome, and C is the link value between two genes.

The goal is to build an application that solves the problem of routing quickest and safest for a large map network, we tried to study the problem as multiple travelling salesman problem (mTSP) by using traveling salesman problem library (TSPLIB) benchmark instances after we have applied parallelism genetic algorithms (PGA) on ad hoc networks. Indeed, we have fixed the source and the destination, and considered variable-length chromosomes or individual as a route, and their genes as nodes, such that the distance between nodes must be non-zero. We start with a randomly selected initial population that contains several routes. The path selection is done by the roulette system, and inspired by lottery wheels, for each pair of routes chosen. The crossover operator chose one-point crossover randomly, except for the source and destination node. If there is a loop in the path after the crossing, they avoid it, after crossing, so there is a low possibility of producing illegal offspring, herein that's why we're going to introduce the mutation operator from the last population, the illegal offspring will be forced to mutate, after the mutation operator. The repair function deletes the node included twice in route and check if the next node is valid, each route is assessed by fitness function (3), repeat according to the number of iterations. Finally, with the fitness function, we select the best route traveled from the source node to the destination node, the procedure is illustrated by Figure 3.

```
Repeat for i = 0 to i = length of database
    do chromosome (ci); calculate fitness (fci); choose the best (bi);
Repeat for i = 0 to i = population size; do population (P);
Repeat for i = 0 to i = number of iterations; do for i = o to i<= population size, i1 and  i2 ∈  P
    if parent ((i1).random) < crossover rate),
        do select parent  (i2) randomly from P, i3= do crossover (i1, i2); repair(i3);
    if parent ((i3).random) < mutation rate ,
        do select (i4) randomly from P; i5= do mutation (i3, i4); repair (i5); calculate fitness (i5);
        compare fitness (i5, bi), choose the best (di);
until converged;
calculate the distance (di);
```

Figure 1. Algorithm of the best routing path

```
Step 1: Begin with the constraint limits is set for the sp route
  GenerationSize = 10000; PopulationSize=80; Number of iterations = 1000;
  CrossoverRate =0.8; MutationRate = 0.05; NodeSource = 5; NodeDistination = 40; K = 8;
Step 2: create a new generation;
Step 3: for i=o to K= size of number of threads; ExecutorService; Repeat GetBestChromosome
  GenerationSize, PopulationSize, NodeSource, NodeDestination, CrossoverRate, mutationRate; end for;
Step 4: ShutdownExecutor;
Step 5: Display the best distance with best fitness;
```

Figure 2. Proposed GA for solving the sensor network in parallel method (PGA)

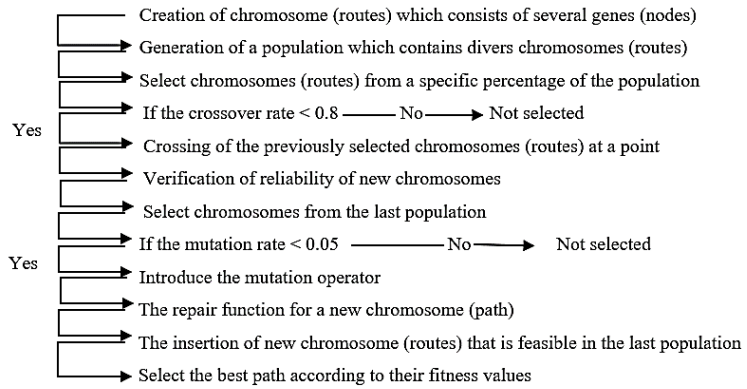

Figure 3. Genetic algorithm in sensor network

5. NUMERICAL RESULT AND DISCUSSION

5.1. Implementation

The challenge is to improve our approach of meta-heuristics GA to takes advantage of CPU for solving large-scale complex problem networks by using PGA. The parameters and their description chosen in this study are as shown in Table 1. Their configuration is summarized in Table 2.

Table 1. Parameters used in PGA

Parameters	Values	Description
Generation Size	10000	To have a great diversity
Number of Iterations	1000	After 1000 iterations we noticed the repetition of the same path
Population Size	80	The quality of the solution can be adjusted according to the population 80
Node Source	1	We fixe the source node
Node Destination	40	We fixe the destination node
Mutation Rate	0.05	After several tests, we see that with 0.05 we get better results
Crossover Rate	0.8	After several tests, we see that with 0.8 we get better results
Number of Threads	8	According to the computer used

Table 2. Configuration machine used in PGA

Configuration	Value
RAM	8 Gb
Technology	LPDDR3 SDRAM
Vitesse	1866 MHz
Cache	4 Mb
CPU	Intel Core i7, 7600U/2.8 GHz
Maximum Speed in Turbo Mode	3.9 GHz
Number of Hearts	Double Heart
Technology Platform	Technology Intel vPro

5.2. Results and discussion

In the first step, we tried to study the problem as mTSP to evaluate our new approach PGA. We compared it with another article Latah (KAG) [25]. Latah [25] proposed to combines the advantage of k-means clustering and PGA to resolve the mTSP problem, the comparison seen in Table 3.

Table 3. The computational results of benchmark problems between KAG and PGA

Method	TSPLIB Instance	K Based Best Distance		
		K=2	K=4	K=6
KAG	Att48	50725.81	74083.53	-
PGA	Att48	34932	38547	36656
KAG	Berlin52	11066.69	11736.74	-
PGA	Berlin52	8965	8841	6877
KAG	Rat99	2487.64	1970.48	-
PGA	Rat99	2264	1962	1878
KAG	Bier127	282343.86	233708.3	-
PGA	Bier127	243238	235824	277325

We notice from Figure 4, that for number of thread equal at 2, we found Berlin52 as a small town, the approximate solution found is 8965 betters than the algorithm of KAG is 11066, with regard to large-scale cities, we tested Bier127, the approximate solution found is 243238 its is much better than the KAG is 282343, for number of thread equal to 4, with our PGA we found 38547 so good result compared by KAG is 74083, for att48, there is a small different between PGA and KAG. In general, for the four instances used, our proposed algorithm gave fairly good results. Thus, guaranteeing a good quality of the solution proposed in a short time, more than result KAG. However, the simulation of PGA on TSPLIB has shown an excellent result for small instances, but there is not a big difference between big instances; that means, the multiplication of the number of calculation units does not spontaneously reduce the execution time of programs. For this reason, we will use the parallelism, and we propose an algorithm that combines the benefits of K-means clustering and GA with parallelism (PGA). In theory, we divide a task to N processors, so the time will be reduced N times. But in reality, this is not the case, because other factors can determine the speed of execution of a parallel program. Firstly, the time of transmission of messages between the processes, and also, parts in all programs that cannot be parallelized.

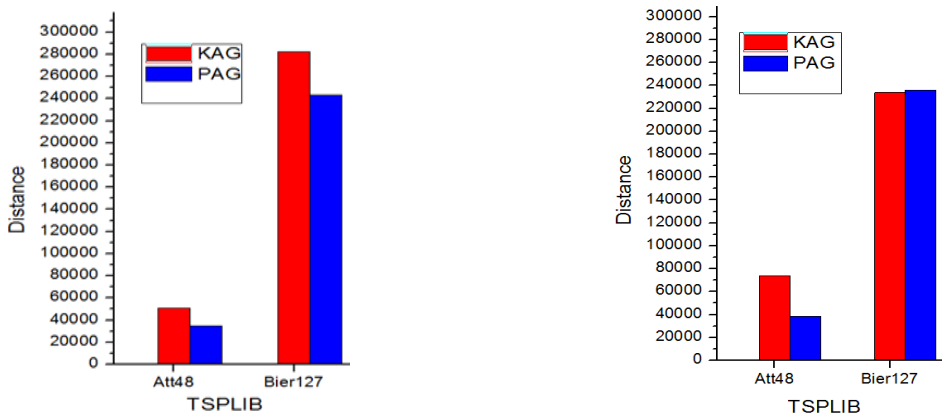

Figure 4. mTSP comparison our new approach PGA with KAG: (a) k=2, (b) k=4

5.3. Solve the sensor network using GA with parallel programming (PGA)

Ad hoc network is presented as a connected graph with N nodes, the chromosomes are considered as the path and each gene represents a node ID that is selected randomly from the set of nodes connected, but the length of the path should not exceed the maximum of a number of the node in the network. The goal is to find the minimal cost path from the source to the destination [4].

5.4. Comparison of parallel and sequential programming

5.4.1. Simulation results for a fixed network with 40 nodes

In our research, we don't find a real topology on the ad hoc network to apply our approach in a real way, even we contacted the large companies of the networks they refused to give us a topology because the data is confidential, that's why we opted to generate our topology with 40 nodes, depicted in Figure 5. In sequential, repeating the previous steps several times help to select the best path, and it takes a lot of time that's why we took advantage of multithreading java and each thread does the same steps starting at the same time, after 1000 of iterations the cost of the best path found is 146 in but in 6415 ms. On the other hand, in parallel, we find the same shot path 146 but just in 1535 ms, that means, the parallelism reduces the time execution.

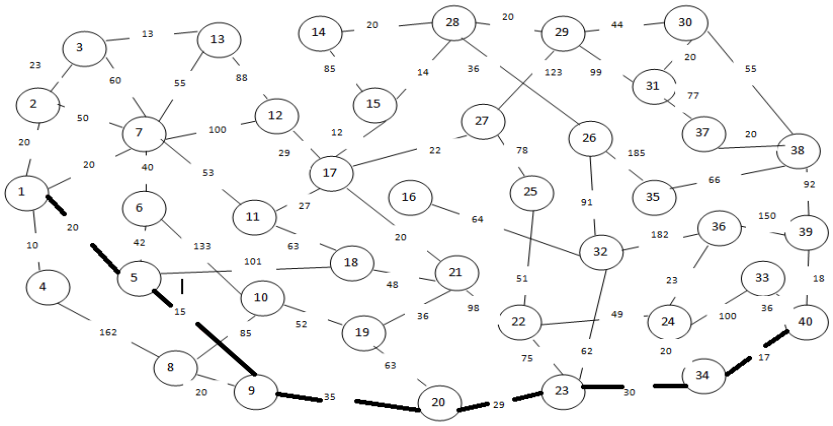

Figure 5. Example network with the optimal path in bold line (optimal path costs is 146)

5.4.2. Simulation results for random network topologies

To evaluate the quality of the proposed PGA approach and the convergence speed time, we generate some network topologies with 40 nodes to 100 nodes randomly. We considered the number of cities as the number of sensors, as shown in Table 4, we have translated the data from Table 4. In the form of a graph as shown in Figure 6 to see the big difference between the two sequential and parallel methods.

We notice from Table 4 and Figure 6, that with the topology of 40 nodes, our PGA approach finds the best path in 6415 ms in sequential, compared to 1535 in parallel, with a time delay of 4880 ms. As for to topology with 70 nodes, the difference between sequential and parallel has become bigger with 10323 ms, and about the topology of 100 nodes, the difference is remarkable, and it has become 16914 ms. The new simulation proposed PGA, show that the algorithm presents a much better quality of solution, and a much higher rate of convergence between two methods, sequential and parallel. The performance of the GA in the parallel method is better than the GA in sequential method. Above all, PGA with parallelism is insensitive to variations of network topologies, or the numbers of nodes. Concerning the choice of short path and speed, the proposed GA in parallel was shorter than GA in sequential method. Therefore, a hardware implementation is required for applications involving real-time services in a dynamic topology network, especially in our algorithm we did not need to know the network signal or the variance of the bandwidth; the proposed PGA algorithm with parallelism can find the solution efficiently and quickly especially for the size of the network of 100 sensors.

Table 4. Sensor network, sequential GA vs parallel GA

Number of Sensors	Execution Time of GA	
	Time Sequential (ms)	Parallel (ms)
40	6415	1535
50	8495	1911
60	10229	2255
70	12947	2624
80	15320	3137
90	17757	3487
100	20939	4035

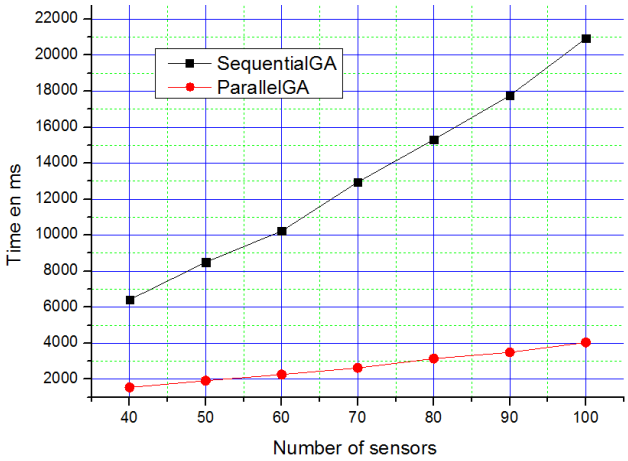

Figure 6. Comparison of the sequential GA vs parallel GA in ad hoc network

In Figure 6, for the topology of 40 nodes, the difference in time between the sequential approach and the parallel approach is smaller. On the other hand, for a topology of 100 nodes we see a big difference in the time calculated between the source node and the destination node, this is due to the communication between the processors. For the topology of 100 nodes, the processor clock speed at which it can process tasks does not increase with hyperthreading. Therefore, one can get a better battery life of the nodes especially for large networks. We summarize that our approach PGA in a parallel way, can search the results speedily and relatively independent of the size of the network ad hoc that means we can find better paths even if we expand the large network.

6. CONCLUSION

This work presented a new approach of GA with parallelism to solve the large network routing problem. Since crossbreeding and mutation operations work on chromosomes of variable length, which means, we have studied in a dynamic topology, for this reason, we have expanded the network up to 100 sensors. Therefore, the results show that the PGA algorithm can search for the best solution in a very efficient

way. We introduced the mutation, which maintains the diversity of the population. On the other hand, we chose the powerful programming of parallelism, which is the way to find the short path from the source to destination in quick time, with the different methods sequential and parallel. The compared results demonstrated that parallel programming in GA found the shortest path solution from the source node to the destination in a short time. In the future, it is proposed to evaluate the performance of the network using more advantage of GPU for solving the problem of the large network ad hoc.

REFERENCES

[1] I. Souissi, N. B. Azzouna, and T. Berradia, "Trust management in vehicular ad hoc networks: a survey," *International Journal of Ad Hoc and Ubiquitous Computing*, vol. 31, no. 4, pp. 230-243, 2019, doi: 10.1504/IJAHUC.2019.101210.

[2] H. Khankhour, J. Abouchabaka, and O. Abdoun, "Genetic algorithm for shortest path in ad hoc networks," *Advanced Intelligent Systems for Sustainable Development*, vol 92, pp. 145-154, 2019, doi: 10.1007/978-3-030-33103-0_15.

[3] P. Nuno, J. C. Granda, and F. J. Suárez, "Assessment of heuristics for self-stabilisation in real-time interactive communication overlays," *International Journal of Ad Hoc and Ubiquitous Computing (IJAHUC)*, vol. 28, no. 2, 2018, doi: 10.1504/IJAHUC.2018.092652.

[4] Z. Wu, C. Zhao, and B. Liu, "Polygonal approximation based on coarse-grained parallel genetic algorithm," vol. 71, 2020, doi: 10.1016/j.jvcir.2019.102717.

[5] C. G. Calderona and C. B. Castañonb, "Isula: A java framework for ant colony algorithms," *SoftwareX*, vol 11, 2020, Art. no. 100400, doi: 10.1016/j.softx.2020.100400.

[6] N. Wilt, "The cuda handbook: A comprehensive guide to GPU programming," *Pearson Education*, 2013.

[7] B. C. Vermeire, F. D. Witherden, and P. E. Vincent, "On the utility of GPU accelerated high-order methods for unsteady flow simulations: A comparison with industry-standard tools," *Journal of Computational Physics*, vol. 334, pp. 497-521, 2017, doi: 10.1016/j.jcp.2016.12.049.

[8] M. Tuberquia and C. Hernandez, "New approaches in cognitive radios using evolutionary algorithms," *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 8, no. 3, pp. 1636-1646, Jun. 2018, doi: 10.11591/ijece.v8i3.pp1636-1646.

[9] A. K. Giri, D. K. Lobiyal, and C. P. Katti, "Optimization of value of parameters in ad-hoc on demand multipath distance vector routing using teaching-learning based optimization," *Procedia Computer Science*, vol 57, pp 1332-1341, 2015, doi: 10.1016/j.procs.2015.07.445.

[10] S. Sharma, D. Jindal, and R. Agarwal, "Analysing mobile random early detection for congestion control in mobile ad-hoc network," *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 8, no. 3, pp. 1305-1314, 2018, doi: 10.11591/ijece.v8i3.pp1305-1314.

[11] N. Khalil and A. Najid, "Performance analysis of 802.11ac with frame aggregation using NS3," *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 10, no. 5, pp. 5368-5376, 2020, doi: 10.11591/ijece.v10i5.pp5368-5376.

[12] A. Taha, R. Alsaqour, M. Uddin, M. Abdelhaq and T. Saba, "Energy efficient multipath routing protocol for mobile ad-hoc network using the fitness function," in *IEEE Access*, vol. 5, pp. 10369-10381, 2017, doi: 10.1109/ACCESS.2017.2707537.

[13] M. Abdelhaq *et al.*, "The resistance of routing protocols against DDOS attack in MANET," *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 10, no. 5, pp. 4844-4852, 2020, doi: 10.11591/ijece.v10i5.pp4844-4852.

[14] S. Khurana, G. Tejpal, and S. Sharma, "Efficient data dissemination approach for QoS enhancement in VANETs," *International Journal of Ad Hoc and Ubiquitous Computing*, vol. 31, no. 4, 2019, doi: 10.1504/IJAHUC.2019.101214.

[15] H. Khankhour, J. Abouchabaka, and O. Abdoun, "Optimization of the ad hoc network by using hybridization of genetic algorithm with a two-optimization algorithm," *International Conference on Digital Technologies and Applications (ICDTA): Digital Technologies and Applications*, vol. 211, 2021, pp. 1073-1080.

[16] Chang Wook Ahn and R. S. Ramakrishna, "A genetic algorithm for shortest path routing problem and the sizing of populations," in *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 6, pp. 566-579, Dec. 2002, doi: 10.1109/TEVC.2002.804323.

[17] F. F.Yeng, S. K. Yoke, and A. Suhaimi, "The saturation of population fitness as a stopping criterion in genetic algorithm," *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 9, no. 5, pp. 4130-4137, 2019, doi: 10.11591/ijece.v9i5.pp4130-4137.

[18] A. Zeni *et al.*, "LOGAN: high-performance GPU-based X-Drop long-read alignment," *IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, pp. 462-471, 2020, doi: 10.1109/IPDPS47924.2020.00055.

[19] P. Bajpa and M. Kumar, "Genetic algorithm – an approach to solve global optimization problems," *Indian Journal of Computer Science and Engineering*, vol. 1, no. 3, pp. 199-206, 2010.

[20] J. Donggara, I. Foster, G. Fox, W. Gropp, K. Kennedy, L. Torczon, and A. White, "Sourcebook of parallel computing," *Morgan Kaufmann*, San Francisco, CA, 2003.

[21] M. N. F. Jamaluddin, A. Ismail, A. A. Rashid, and T. T. B. O. Takhleh, "Performance comparison ofjava based parallel programming models," *Indonesian Journal of Electrical Engineering and Computer Science (IJEECS)*, vol. 16, no. 3, pp. 1577-1583, 2019, doi: 10.11591/ijeecs.v16.i3.pp1577-1583.

[22] R. Baños, J. Ortega, C. Gil, F. de Toro, and M. G. Montoya, "Analysis of OpenMP and MPI implementations of meta-heuristics for vehicle routing problems," *Applied Soft Computing*, vol. 43, pp. 262-275, 2016, doi: 10.1016/j.asoc.2016.02.035.

[23] C. Guerrero, I. Lera, and C. Juiz, "Genetic algorithm for multi-objective optimization of container allocation in cloud architecture," *Journal of Grid Computing*, vol. 16, no. 7, pp. 113–135, 2018, doi: 10.1007/s10723-017-9419-x.

[24] P. Kitjacharoenchai, M. Ventresca, M. M. Javadi, S. Lee, J. M. A. Tanchoco, and P. A. Brunese, "Multiple traveling salesman problem with drones: Mathematical model and heuristic approach," *Computers & Industrial Engineering*, vol. 129, 2019, doi: 10.1016/j.cie.2019.01.020.

[25] M. Latah, "Solving multiple TSP problem by k-means and crossover based modified ACO algorithm," *International Journal of Engineering and Technical Research*, vol. 5, no. 2, pp. 430-434, 2016.