EAI Endorsed Transactions
on Energy Web

Research Article EAI.EU

GPGPU based Multi-hive ABC Algorithm for Constrained Global Optimization Problems

Sandeep U. Mane[1,*], Amol C. Adamuthe[2] and Aprupa S. Pawar[3]

[1]Dept. of CSE, Rajarambapu Institute of Technology (affiliated to Shivaji University Kolhapur), Rajaramnagar, Sangli Dist., MH, India.
[2]Dept. of CS&IT, Rajarambapu Institute of Technology (affiliated to Shivaji University Kolhapur), Rajaramnagar, Sangli Dist., MH, India.
[3]Dept. of CSE, Walchand College of Engineering, Sangli, (affiliated to Shivaji University Kolhapur), MH, India.

Abstract

INTRODUCTION: The artificial bee colony (ABC) algorithm is a nature-inspired technique used for solving different optimization problems. This paper presents a multi-hive ABC algorithm for solving constrained benchmark functions of CEC2006. The CEC2006 data set contains the global benchmark functions with different design variables, number and type of constraints.
OBJECTIVES: The objective of the proposed work is to design and apply the GPGPU based multi-hive ABC algorithm to solve constrained optimization problems.
METHODS: The proposed approach is a multi-population coarse-grained system implemented using General Purpose Graphics Processing Unit (GPGPU). The performance of the proposed approach is compared with the serial ABC algorithm for eleven benchmark functions and results in the literature. The multi-hive ABC algorithm has multiple hives, each running separate ABC algorithm on different cores of GPGPU.
RESULTS: The proposed approach provides global best solutions in significantly reduced time for all benchmark functions. The speed-up obtained is approximately 7X to 9X. The GPGPU device utilization is approximately 57% to 91%.
CONCLUSION: The GPGPU based multi-hive ABC algorithm is a found good with respect to best results, speed up factor and GPU utilization to solve constrained optimization problems.

Keywords: Multi-hive ABC algorithm, GPGPU, Coarse-grained model, Constrained benchmark functions, Optimization problems.

Received on 30 September 2019, accepted on 31 January 2020, published on 14 February 2020

doi: 10.4108/eai.13-7-2018.163156

1. Introduction

There exist various optimization techniques broadly classified into traditional and non-traditional techniques. Each type of technique has its own merits and demerits. The widely used non-traditional techniques are nature-inspired. These techniques include swarm intelligence based optimization algorithms like particle swarm optimization, ant colony optimization, artificial bee colony algorithm, etc. The global collective behaviour of social insects that arises from self-organization and division of tasks is the root of all swarm-based optimization algorithms. Ant colony optimization imitates the behaviour of ants, Particle swarm intelligence imitates the behaviour of birds and Artificial Bee Colony algorithm imitates the behaviour of honey bees [1-5]. The artificial Bee Colony (ABC) algorithm was proposed by Karaboga in 2005, for multidimensional, multimodal optimization problems. Artificial bee colony algorithm is inspired by the honey bees' behaviour of searching for food. The colony of honey bees contains three types of bees namely, employed bees, onlooker bees and scout bees. The

*Corresponding author. Email: manesandip82@gmail.com

entire colony is divided into two parts - the employed bees and the onlooker bees. The number of food sources is equal to the number of employed bees. Each bee tries to improve its solution by neighbour search method [4].

The benchmark functions are used to test the performance of heuristic algorithms. Benchmarking is required for an effective evaluation of the working of newly proposed algorithms. There are different benchmark functions used for the evaluation of the algorithms. The benchmark functions are designed in such a way that, it possesses different properties of the real-world optimization problems. The different properties of the objective function are continuous, discontinuous, separable, non-separable, linear, nonlinear, unimodal, multimodal, constrained and unconstrained [6].

Parallel processing is the ability to process or perform multiple operations simultaneously. It is the simultaneous use of more than one CPU or processor core to execute a program or multiple computational tasks. Parallel programming is useful for getting better and fast performance from an algorithm which gives us results with a short period of time. The parallel processing is useful for getting results of highly non-linear, high dimensional real-world problems [7]. Due to the development of high throughput oriented hardware as well as advancement in parallel programming framework, nowadays multi-core or many-core based software's performance is enriched enormously. The General Purpose Graphic Processing Unit (GPGPU) is a device with huge computational power due to the presence of hundreds of cores as compare to multi-core CPUs. The Nvidia developed a Compute Unified Device Architecture (CUDA) framework to program the GPGPU, developed by them. The development of the CUDA framework made GPGPU programming easy. The GPGPU based development of a parallel algorithm or application is most preferable when selected algorithm or application supports data parallelism. The compute-intensive applications when implemented using GPGPU, the speed-up achieved is more than 100 times as compared to its sequential version. The GPGPUs have developed throughput oriented rather than latency oriented [8].

In the literature, it is found that many algorithms have been successfully implemented on GPGPU and have yielded better performance [5, 9, 10]. The parallel ABC versions were also developed to address optimization problems. Parpinelli et al. [11] developed three parallel models of the ABC algorithm: multi-hive with migrations, master-slave, and hybrid hierarchical. The proposed approach tested unconstrained benchmark optimization problems. Anan Banhamsakun et al. proposed an artificial bee colony algorithm for the distributed environment using the manager-worker model [12]. Ruhai Luo et al. [13] proposed ABC optimization with ripple communication strategy (PABC-RC), implemented for three benchmark problems. Milos Subotic et al. proposed three different approaches for the parallelization of a standard artificial bee colony (ABC) algorithm. These approaches are independent parallel runs and two variations of multiple swarm parallelizations [14]. Zhang et al. proposed a "multi-hive artificial bee colony" to solve constrained multi-objective optimization problems (MHABC-CMO). The experiments were performed on a set of five benchmark test functions [15]. Basturk and Akay in [16] presented a parallel version of the ABC algorithm based on a coarse-grained parallel computing model. In [17] Kun Xu et al. have presented Parallel Artificial Bee Colony Algorithm (PABCA) using Linux based Message Passing Interface for solving the travelling salesman problem.

The parallelization strategy used for the ABC algorithm are Parallel Master-Slave Model, Parallel Multi-Hive Model, Parallel Hybrid Hierarchical Model, Manager-Worker Model, Independent parallel runs approach, and Multiple swarms-one best solution and Multiple swarms-best solutions from all swarms [11, 14, 15]. Each strategy has its own merits and demerits.

The proposed work presents the design and implementation of GPGPU based multi-hive ABC algorithm to solve constrained optimization problems. The main motive of implementing GPGPU based multi-hive ABC algorithm is to reduce execution time and improve the performance. The eleven single-objective constrained benchmark functions are used to test the performance of the proposed approach, which possesses different properties. The multi-hive model is a multiple-population coarse-grained system that uses two or more hives that are initialized at the same time with different random seeds. Each population can be seen as an island. The hives work independently from each other, and each one runs an ABC algorithm, with a migration process occurring periodically.

The constrained optimization problems were addressed by researchers using various traditional and non-traditional techniques. When such problems are solved with a sequential (CPU based) approach it takes significant time to get results. The importance of the proposed work is to implement the multi-hive ABC algorithm on GPGPU so as to obtain the result in less time and utilize the available computational resources. As the proposed approach succeeds in reducing execution time significantly, it can be used to solve real-time single-objective constrained optimization problems.

The remaining paper is organized as follows: Section 2 discusses the proposed methodology. The benchmark functions used to evaluate the performance of the proposed approach are presented in section 3. The results and discussion with experimental setup and parameter settings are presented in section 4. The conclusion of the proposed work and future research direction are discussed in section 5.

2. Methodology

This section briefs about the Artificial Bee Colony algorithm, proposed GPGPU based multi-hive ABC algorithm. The steps of the proposed approach and execution flowchart are presented.

2.1. Artificial Bee Colony Algorithm

The Artificial Bee Colony (ABC) algorithm was proposed by Karaboga in 2005, to solve multidimensional, multimodal optimization problems. Artificial bee colony algorithm is

inspired by the honey bees' behaviour of searching for food. A colony of artificial bees has three types of bees namely employed bees, onlooker bees and scout bees. The entire colony is divided into two parts - the employed bees and the onlooker bees. The number of food sources is equal to the number of employed bees. Each bee tries to improve its solution by neighbour search method [3-4]. The detailed pseudo-code of the ABC algorithm to solve constrained and unconstrained optimization problems and its explanation can be found in [4].

2.2 Proposed GPGPU based Multi-hive ABC Algorithm

The proposed GPGPU based multi-hive ABC is described in this section. This section presents and describes the flowchart of the proposed approach as well as the execution of a multi-hive model on GPGPU. The execution steps of the proposed approach and strengths of the selected parallel model are presented.

The proposed GPGPU based multi-hive ABC algorithm is implemented using GPGPU for solving constrained benchmark functions. The major steps of the ABC algorithm are executed on GPGPU using multi-hive with a migration strategy. This strategy consists of multiple colonies, each colony executes the ABC algorithm. At the end of certain iterations, each colony shares its best solution with other colonies and replaces its own worst solution with the migrated best solution from other colonies. By following this procedure, all the colonies move towards the best solution in a co-evolution fashion. For migration, unidirectional ring topology is used. In this topology, each colony migrates (sends) its best solution to the colony on its right and replaces its worst solution by the migrated solution of the colony to its left. This strategy adds the migration gap (number of iterations between successive migrations) as one more parameter to the ABC algorithm. Fig. 1 shows the flowchart of the proposed GPGPU based multi-hive ABC algorithm.

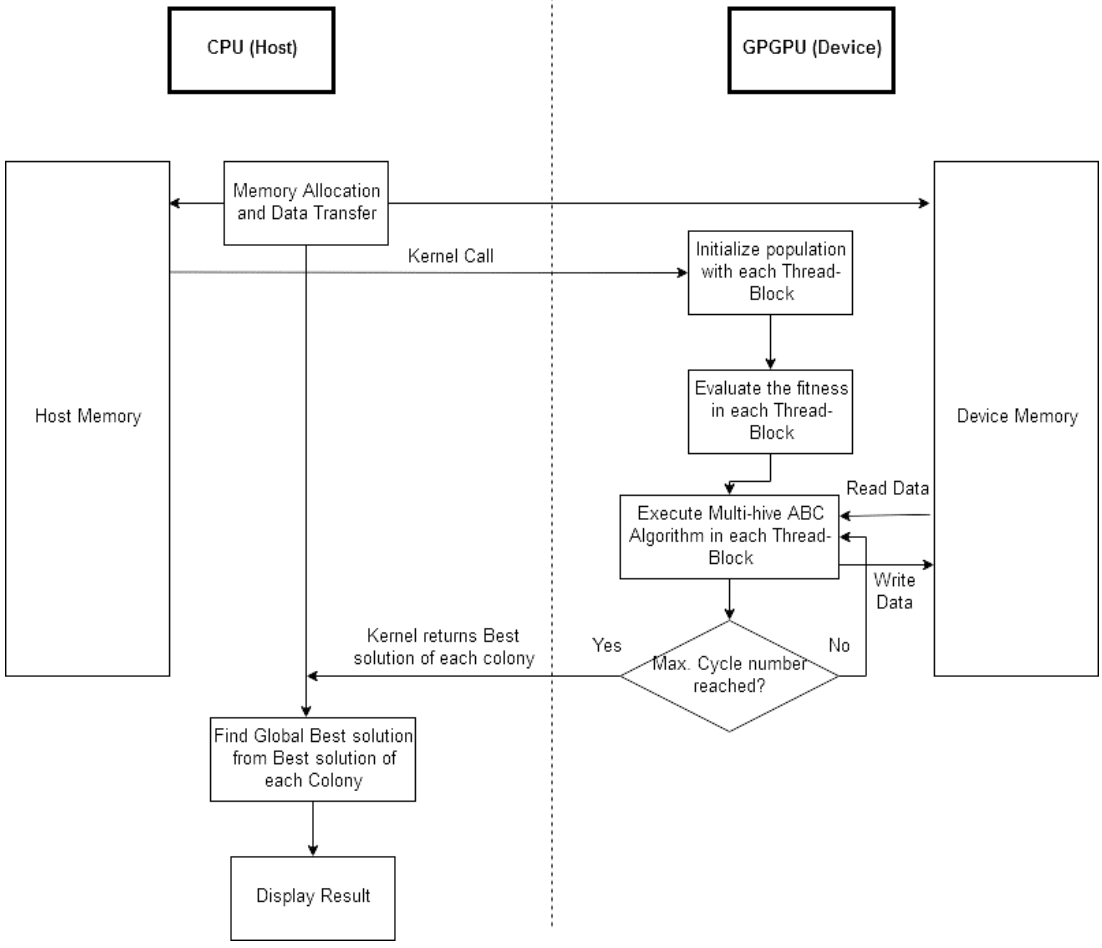

Figure 1. Flow of Proposed GPGPU based Multi-hive ABC algorithm

The CPU initially allocates memory on GPGPU's global memory for migrating the best solution parameters and saving the best solution after the execution. When the kernel is launched, it creates the instances of function which are equivalent to the total number of threads. In the end, when the Maximum Cycle number is reached the global best solutions from each hive are copied into a global array that is sent to CPU with their number of constraints violated. From this

array one best solution is returned as the final solution. Odd-even sorting is performed to select the best solution.

For constrained benchmark problems, the ABC algorithm given in [3] is implemented. It has two more parameters called the Modification rate (MR) and the Scout generation period (SPP). Here for finding best solution between two candidate solutions, the solution with minimum constraint violation is preferred than the solution with more constraint violation and if the number of constraints violated by both solutions is same or zero the solution with minimum objective function value is preferred over another in case of minimization function and the solution with maximum problem value is preferred than the other solution in case of maximization problem. For finding this minimum Odd-Even sort [18] is used as atomic minimum CUDA functions are available only for GPGPU with compute capability above 3.5.

Fig. 2 presents the execution of kernel (function) on GPGPU. The thread organization in CUDA is two-level hierarchical. The grid consists of the number of blocks and each block consists of the number of threads. The blocks and threads are multi-dimensional. Here, when the kernel is launched, the number of blocks denotes the number of colonies and the number threads denote the number of employee bees. The grid size is a product of a number of blocks and number of threads per block. Each thread-block runs a separate copy of multi-hive ABC algorithm and migrates the solution after every migration gap with the use of global memory. The total instances of multi-hive ABC algorithm are equivalent to grid size.

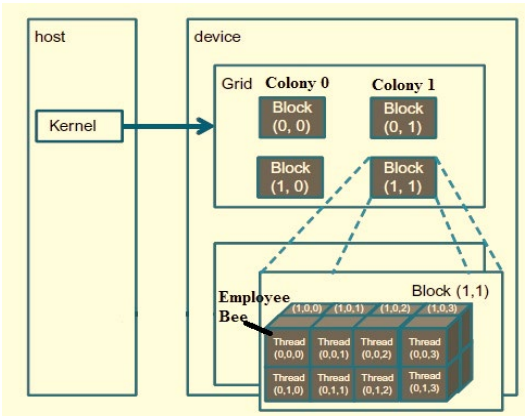

Figure 2. Execution of multi-hive ABC on GPGPU's Thread-Block

The steps of the proposed GPGPU based multi-hive ABC algorithm are presented below.

(i) Input: colony size, number of food sources equal to half of the colony size, scout, production period Maximum cycle number, Number of colonies (hives), migration interval, modification rate for a new solution and Problem specific variables.
(ii) Output: f(x) function's optimal value.
(iii) Allocate memory on the host (CPU) and device (GPGPU) for solution variables and optimal solution. Transfer data from the host to the device.
(iv) Algorithm:
    a. Launch kernel with the number of blocks equal to the number of colonies and the number of threads equal to the number of employed bees.
    b. Each block independently runs the ABC algorithm on the device.
    c. The best solution from each colony is migrated to another colony by using a unidirectional ring topology, depending on the value of the migration rate parameter.
    d. According to ring topology, each colony (block) replaces its worst solution with the best solution of another colony.
    e. The kernel exits when it reaches to maximum cycle number.
(v) Copy the best solution of each colony to host.
(vi) Sort the copied best solutions on the host and find the optimal solution to the problem.

The parallel multi-hive model is a multiple-population coarse-grained system that uses two or more hives that are initialized at the same time with different random seeds [11]. Each population can be seen as an island. The hives work independently from each other, and each one runs an ABC algorithm, with a migration process occurring periodically. For example, one processing core can be used for each hive. Migrations occur between hives by using a migration policy that includes some parameters defined by the user. Two important parameters used are the migration gap, which is the number of generations between successive migrations; and the migration rate at which the number of individuals will migrate per migration event. It also defines some criteria for the selection of immigrants and the substitution of emigrants. For example, unidirectional ring topology. The merits of the coarse-grain approach are presented in [16]. The coarse-grained method is preferred due to its ease of implementation. Other parallelization models provide profit in the evaluation of expensive fitness functions, while the coarse-grained model is very useful when the fitness function is computationally inexpensive. The serial algorithm uses only one random seed at the initial step, the coarse-grained model uses a number of random seeds equal to the number of subpopulations (or swarms). This method increases the diversity of the population and thus the enhancement of the exploration process on search space increases the probability to hit the optimal solution as well as improves the convergence rate. Another advantage of this model is that it can be used with many topologies (mesh, ring, hypercube, etc.) and also it needs less communication between nodes, so its efficiency may be better than those of other models.

3. Tested Benchmark Functions

This section describes the properties of CEC2006 benchmark functions used for experimentation. The CEC 2006 benchmark functions are single-objective linear, nonlinear, cubic, and polynomial types. Each function has a different number of design variables as well as the range of each design variable varies from function to function. The real-time single-objective optimization problems exist in engineering and scientific domains possess similar characteristics to selected benchmark functions. Hence, instead of solving real-time single-objective optimization problems, the proposed work considers CEC 2006 standard benchmark functions. The performance of the proposed GPGPU based Multi-Hive ABC algorithm evaluated using constrained benchmark functions (CEC2006). The definition and mathematical model of the constrained benchmark functions can be found in [19].

Table 1 presents the constrained benchmark functions with the number of decision variables, type of function, number and type of constraints, and global best solution. G1 is a quadratic function with 9 linear inequality constraints. G2 is a non-linear function with two non-linear inequality constraints. G3 is a polynomial function with 1 non-linear equality function. G4 is a quadratic function with 6 non-linear inequality constraints. G5 is a cubic function with two linear inequality and three non-linear equality constraints. G6 is a cubic function with two non-linear inequality constraints. G8 is a non-linear function with two non-linear inequality constraints. G10 is a linear function with three linear inequality constraints and three non-linear inequality constraints. G11 is a quadratic function with one non-linear equality constraint. G12 is a quadratic function with 1 non-linear inequality constraint. G13 is a non-linear function with three non-linear equality constraints.

Table 1. Benchmark functions with its properties

Function	No. of Decision Variables	Type of function	Number and Type of constraints	Global Best Solution
G1	13	Quadratic	9 Linear Inequality	-15
G2	20	Nonlinear	2 Nonlinear Inequality	-0.803619
G3	10	Polynomial	1 Nonlinear Equality	-1.0
G4	5	Quadratic	6 Nonlinear Inequality	-30665.539
G5	4	Cubic	2 Linear Inequality and 3 Nonlinear Equality	5126.484
G6	2	Cubic	2 Nonlinear Inequality	-6961.814
G8	2	Nonlinear	2 Nonlinear Inequality	-0.095825
G10	8	Linear	3 Linear Inequality and 3 Nonlinear Inequality	7049.28
G11	2	Quadratic	1 Nonlinear Equality	0.75
G12	3	Quadratic	1 Nonlinear Inequality	-1
G13	5	Nonlinear	3 Nonlinear Equality	0.05394

4. Results and Discussion

This section describes the parameter settings, experimental setup, results obtained and its comparison with results from the literature.

The experiments performed using GeForce GTX 680 Nvidia's GPGPU. It has 8 streaming multiprocessors with 1536 CUDA cores. The device has 2GB global memory with a speed of 6.0 Gbps. The compute capability of the device is 3.0. The proposed approach is implemented in CUDA-C using CUDA Toolkit 7.0 with the Thrust library. The performance of the proposed GPGPU based multi-hive ABC algorithm is evaluated using the CEC2006 constrained benchmark dataset. The proposed approach is implemented for G01 to G06, G08, and G10 to G13 benchmark functions.

The artificial bee colony size set as 64, the maximum number of cycles used as 1000, Modification Rate (MR) set as 0.8, SPP set as a product of half of the colony size and the number of problem dimensions. The Migration gap used as 250. The maximum number of cycles used for the G1 benchmark function is 5000 [3]. The number of CUDA blocks used is equal to the number of hives and the number of CUDA threads used is equal to the number of employed bees. The GPGPU based multi-hive ABC algorithm is executed for 30 times for each function. The mean and standard deviation statistical tests are used to measure the performance. Table 2 to Table 7 presents the results obtained by GPGPU based multi-hive ABC algorithm for selected benchmark functions.

Table 2. Results obtained using GPGPU based multi-hive ABC algorithm for 11 benchmark functions

Function	No. of Decision Variables	Global Best Solution	Obtained Best Solution	Mean	Std. Deviation	Time (in ms)
G1	13	-15	**-15**	-15	0.00E+00	86.72
G2	20	-0.803619	**-0.803619**	-0.7924	5.56E-03	120.07
G3	10	-1.0	**-1.0**	-0.99999	8.43E-07	17.93
G4	5	-30665.539	**-30665.53**	-30365.35	7.35E+01	21.49
G5	4	5126.484	**5126.48**	5184.603	4.37E+01	135.12
G6	2	-6961.814	**-6961.814**	-6961.810	4.99E-02	18.76
G8	2	-0.095825	**-0.095825**	-0.09582	4.83E-06	20.65
G10	8	**7049.28**	7051.43	7156.287	5.78E+01	19.66
G11	2	0.75	**0.75**	0.75	3.89E-04	15.12
G12	3	-1	**-1**	-1.00	0.00E+00	47.92
G13	5	0.05394	**0.05394**	0.0539	4.97E-06	19.027

The values in boldface indicate the best results

Table 2 presents the results obtained in terms of best, mean, standard deviation, and execution time. The GPGPU based multi-hive algorithm obtains global best results for all the selected functions. The number of hives used is 4 with 64 as population size. The function evaluations performed for function G1 are 12,80,000 (as the maximum cycle number used is 5000). The function evaluations performed for the remaining functions are 2,56,000. The parallel execution time is from 15.12 milliseconds to 135.12 milliseconds. The proposed approach succeeds to obtain the global best solution for all the selected benchmark function except G10 function. The best value obtained for G10 function using the proposed approach is close to the global best solution. From the results, it can be said that irrespective of the platform used to implement the multi-hive ABC algorithm, it gives better results to selected benchmark problems. The GPGPU based multi-hive ABC algorithm is a promising approach to solve the single objective constrained optimization problems.

Table 3. Comparison of Sequential ABC and Proposed GPGPU based multi-hive ABC Algorithm

Function	Global solution	Sequential ABC Results taken from [3]			Proposed GPGPU based Multi-hive ABC Algorithm		
		Best	Mean	Std. Deviation	Best	Mean	Std. Deviation
G1	-15	-15	-15	0.00E+00	**-15**	-15	0.00E+00
G2	-0.803619	-0.803598	-0.7924	1.20E-02	**-0.803619**	-0.7924	5.56E-03
G3	-1.0	-1.0	-1.0	0.00E+00	**-1.0**	-0.99999	8.43E-07
G4	-30665.539	-30665.539	-30665.539	0.00E+00	**-30665.53**	-30365.35	7.35E+01
G5	5126.484	5126.484	5185.714	7.54E+01	**5126.48**	5184.603	4.37E+01
G6	-6961.814	-6961.814	-6961.813	2.00E-03	**-6961.814**	-6961.810	4.99E-02
G8	-0.095825	-0.095825	-0.09582	0.00E+00	**-0.095825**	-0.09582	4.83E-06
G10	7049.28	7053.904	7224.407	1.34E+02	**7051.43**	7156.287	5.78E+01
G11	0.75	0.75	0.75	0.00E+00	**0.75**	0.75	3.89E-04
G12	-1	-1	-1.00	0.00E+00	**-1**	-1.00	0.00E+00
G13	0.05394	0.760	0.968	5.50E-02	**0.05394**	0.0539	4.97E-06

The values in boldface indicate the best results.

Table 3 presents a comparison of results given in [3] of sequential ABC and proposed GPGPU based multi-hive ABC algorithm. The results are compared based on best, mean and standard deviation. The proposed approach obtains equally best results for G1, G3, G4, G5, G6, G8, G11, and G12. The results obtained for function G2, G10, and G13 are better than the results given in [3]. The mean results obtained for G4, G5, G10, and G13 are better than the results presented in [3] for

the ABC algorithm. From the results presented in Table 3, it is observed that the proposed approach is competitive with the existing ABC algorithm.

Table 4. Mean solutions comparison of proposed approach with results from [3]

Function	Global Best Solution	PSO [3]	DE [3]	ABC [3]	Proposed GPGPU based multi-hive ABC algorithm
G1	-15	-14.710	-14.555	-15.000	**-15**
G2	-0.803619	-0.419960	-0.6657	-0.792412	**-0.7924**
G3	-1.0	-0.764813	**-1.000**	-0.99999	-0.99999
G4	-30665.539	-30665.539	-30665.539	**-30665.539**	-30365.35
G5	5126.484	**5135.973**	5264.270	5185.714	5184.603
G6	-6961.814	**-6961.814**	--	-6961.813	-6961.810
G8	-0.095825	-0.095825	-0.095825	**-0.095825**	-0.09582
G10	7049.28	7205.5	**7147.334**	7224.407	7156.287
G11	0.75	0.749	0.901	0.750	**0.75**
G12	-1	-0.998875	-1.000	-1.000	**-1.00**
G13	0.05394	0.569358	0.872	0.968	**0.0539**

The values in boldface indicate the best results

Table 4 presents the comparison of mean results obtained using the proposed GPGPU based multi-hive ABC algorithm and results presented in [3] for ABC, PSO and DE algorithm. Form the comparison, it is observed that, the proposed approach succeed to obtain better mean results for G1, G2, G11, G12, and G13 functions than the other approaches used for comparison. The proposed approach gives better mean results than the ABC algorithm for all the selected benchmark functions except G4 and G8.

Table 5. Execution time and Device utilization of GPGPU based multi-hive ABC algorithm

Function	No. of Decision Variables	Population Size	No. of Iterations	Time (in ms)		Speed-up
				Sequential ABC Algorithm	Proposed GPGPU based multi-hive ABC Algorithm	
G1	13	64	5000	679.21	86.72	7.832
G2	20	64	1000	1083.41	120.07	9.023

Table 5 presents the comparison of results obtained using the ABC algorithm and proposed GPGPU based multi-hive ABC algorithm and the speed-up of the proposed approach. The proposed approach gives 7X to 9X faster results than the sequential ABC algorithm. The parameter settings were kept similar to both the approaches to obtain better results.

Table 6. Execution time and Device utilization of proposed GPGPU based multi-hive ABC algorithm

Function	No. of Decision Variables	Population Size	No. of Iterations	Time (in ms)	Device Utilization
G1	13	64	5000	86.72	86.9%
G2	20	64	1000	120.07	77.2%
G3	10	64	1000	17.93	57.9%
G4	5	64	1000	21.49	63.5%
G5	4	64	1000	135.12	61.8%
G6	2	64	1000	18.76	65.7%
G8	2	64	1000	20.65	59.8%
G10	8	64	1000	19.66	63.0%
G11	2	64	1000	15.12	57.0%
G12	3	64	1000	47.92	91.2%
G13	5	64	1000	19.027	68.0%

Table 6 presents the execution time required for proposed GPGPU based multi-hive ABC algorithm and GPGPU utilization for selected benchmark functions. The device utilization varies with the number of decision variables, number and type of constraints as well as the type of function. The device utilization is more than 55% for all the selected benchmark functions. The maximum device utilization recorded is about 91% for G12 function, which is quadratic with one nonlinear inequality constraint. The minimum device utilization recorded is about 57% for the G11 function which is quadratic in nature with one nonlinear equality constraint. The device utilization is varying due to different reasons, it depends on the number of ideal threads during execution, the thread divergence, as well as the complexity of the selected problem.

Table 7. Convergence of 4 hives vs. 8 hives

Function	No. of Iterations for Convergence	
	4 – Hives	8 – Hives
G1	920	906
G2	951	853
G3	946	920
G4	961	954
G5	391	307
G6	993	987
G8	916	850
G10	159	141
G11	832	750
G12	993	981
G13	980	972

Table 7 presents the convergence rate of 4 hives and 8 hives GPGPU based multi-hive ABC algorithm. As the number of hives increased from 4 to 8, it is observed that the best solutions are obtained early due to co-evolution effect.

5. Conclusion and Future Work

This paper presents a GPGPU based multi-hive ABC algorithm to solve constrained optimization problems. The proposed approach is implemented on GPGPU using a coarse-grain model. The multi-hive ABC algorithm is executed on GPGPU using CUDA-C. The number of blocks and the number of threads used is equivalent to the number of hives (colonies) and the number of employed bees, respectively. The performance of the proposed approach is evaluated using 11 constrained benchmark functions of CEC2006. The proposed approach obtains global best solutions for all selected benchmark functions. The speed-up achieved is approximately 7X to 9X faster

than the sequential ABC algorithm. The parallel execution time to obtain the best result varies from 15.12 milliseconds to 135.12 milliseconds as per the properties of selected benchmark functions. The GPGPU device is utilized from approximately 57% to 91%. The parallel execution time required and device utilization depends on the type of problem, the number of decision variables, number and type of constraints. The device utilization according to the number of active threads, the thread divergence, number of design variables and type of constraints of the problem. The performance of the proposed approach is also compared with the results found in the literature using various statistical tests. From obtained results and comparison with literature, the GPGPU based multi-hive ABC algorithm is one of the promising approaches to solve constrained optimization problems on GPGPU.

The performance of the proposed approach can be evaluated using complex constrained global optimization problems as future work. The device utilization can be improved by using various CUDA related features. The parallel execution time can be improved by implementing the proposed approach using GPGPU with compute capability more than 3.5, it will facilitate to use various libraries developed by Nvidia for CUDA-C. The GPGPU based multi-hive ABC algorithm can be used to solve real-world constrained optimization problems. Also, the ABC algorithm can be implemented on GPGPU using different parallelization strategy that has not been implemented yet.

References

[1] Rao SS. Engineering optimization: theory and practice. John Wiley & Sons; 2019.

[2] Karaboga D. An idea based on honey bee swarm for numerical optimization. Technical report-tr06, Erciyes University, engineering faculty, computer engineering department; 2005.

[3] Karaboga D, Basturk B. Artificial bee colony (ABC) optimization algorithm for solving constrained optimization problems. In: Melin P., Castillo O., Aguilar L.T., Kacprzyk J., Pedrycz W. (eds) Foundations of Fuzzy Logic and Soft Computing. IFSA 2007. Lecture Notes in Computer Science, vol 4529. Springer, Berlin, Heidelberg; 2007. PP. 789-798.

[4] Karaboga D, Basturk B. A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm. Journal of global optimization. 2007 Nov 1; 39 (3):459-71.

[5] Luo GH, Huang SK, Chang YS, Yuan SM. A parallel Bees Algorithm implementation on GPU. Journal of Systems Architecture. 2014 Mar 1; 60 (3):271-9.

[6] Opara K, Arabas J. Benchmarking procedures for continuous optimization algorithms. Journal of Telecommunications and Information Technology. 2011:73-80.

[7] Mane S.U., Narsinga Rao M.R. Large-Scale Compute-Intensive Constrained Optimization Problems: GPGPU-Based Approach. In: Ray K., Sharma T., Rawat S., Saini R., Bandyopadhyay A. (eds) Soft Computing: Theories and Applications. Advances in Intelligent Systems and Computing, vol 742. Springer, Singapore; 2019. PP. 579-589.

[8] Mane SU, Omane R, Pawar A. GPGPU based teaching learning based optimization and artificial bee colony algorithm for unconstrained optimization problems. In 2015 IEEE International Advance Computing Conference (IACC). IEEE; 2015. PP. 1056-1061.

[9] Hofmann J, Limmer S, Fey D. Performance investigations of genetic algorithms on graphics cards. Swarm and Evolutionary Computation. 2013 Oct 1; 12:33-47.

[10] Mussi L, Daolio F, Cagnoni S. Evaluation of parallel particle swarm optimization algorithms within the CUDA™ architecture. Information Sciences. 2011 Oct 15; 181 (20):4642-57.

[11] Parpinelli RS, Benitez CM, Lopes HS. Handbook of Swarm Intelligence. Berlin, Heidelberg: Springer; 2011. Parallel approaches for the artificial bee colony algorithm; PP. 329-345.

[12] Banharnsakun A, Achalakul T, Sirinaovakul B. Artificial bee colony algorithm on distributed environments. In 2010 second world congress on nature and biologically inspired computing (NaBIC). IEEE; 2010. PP. 13-18.

[13] Luo R, Pan TS, Tsai PW, Pan JS. Parallelized artificial bee colony with ripple-communication strategy. In 2010 Fourth International Conference on Genetic and Evolutionary Computing. IEEE; 2010. PP. 350-353.

[14] Subotic M, Tuba M, Stanarevic N. Different approaches in parallelization of the artificial bee colony algorithm. International Journal of mathematical models and methods in applied sciences. 2011 Mar; 5 (4): 755-62.

[15] Zhang H, Zhu Y, Yan X. Multi-hive artificial bee colony algorithm for constrained multi-objective optimization. In 2012 IEEE Congress on Evolutionary Computation. IEEE; 2012 PP. 1-8

[16] Basturk A, Akay R. Performance analysis of the coarse-grained parallel model of the artificial bee colony algorithm. Information Sciences. 2013 Dec 20; 253:34-55.

[17] Xu K, Jiang MY, Yuan DF. Parallel artificial bee colony algorithm for the traveling salesman problem. In Advanced Materials Research 2013 (Vol. 756, pp. 3254-3259). Trans Tech Publications Ltd.

[18] Jan B, Montrucchio B, Ragusa C, Khan FG, Khan O. Fast parallel sorting algorithms on GPUs. International Journal of Distributed and Parallel Systems. 2012 Nov 1; 3(6):107.

[19] Liang JJ, Runarsson TP, Mezura-Montes E, Clerc M, Suganthan PN, Coello CC, Deb K. Problem definitions and evaluation criteria for the CEC 2006 special session on constrained real-parameter optimization. Journal of Applied Mechanics. 2006 Sep 18; 41(8):8-31.