DOI: https://doi.org/10.46632/jeae/1/2/8

Journal on Electronic and Automation Engineering
Vol: 1(2), 2022
REST Publisher; ISSN: 2583-6951
Website: http://restpublisher.com/journals/jeae/

A Heuristic Framework to Organize Heterogenous Network Embedded Systems

***R. Kathirvel, K. Kalaimaamani, V. Senthil Kumaran, A. Malarvizhi,**
Mahendra Engineering College, Namakkal, Tamil Nadu, India.
*Corresponding author email: kathirvelr@mahendra.info

Abstract: This paper introduces a heuristic method to process sequencing in Heterogeneous Distributed Embedded Systems (HDES). This suggested study discusses the Genetic Algorithm (GA) and Ant Colony Optimization (ACO) enhanced using the greedy algorithm added to the system, which consists of numerous Heterogeneous Embedded Machines (HEMs) operating as a group. Customers have the ability to make use of their computing capacity remotely. The connectivity options on various kinds of buses are considered to identify the best option. Novel meta-heuristic data based on transporting reliant is proposed to increase the chance of ACO generating task prioritization. Furthermore, a greedy approach for machine assignment is used to carry out task planning. Simulations using arbitrary task graphs in the HEM collection show that the Enhanced Greedy Ant Colony Optimization (EGACO) process exceeds each of the others by 33% in terms of outcome eminence.

Keywords: HDES, Network cluster, Meta-heuristic task scheduling, GA, ACO, Greedy process.

1. Introduction

For many years, the development and expansion of diverse outstanding performance computer architectures has been remarkable. Typically, diverse systems [1] for distributed and parallel IT include system software, numerous processors, memory, and a means of interaction network. Currently, over sixty percent of the computerized control elements in both motor vehicle microchip technology and avionics systems use the HDES [2] due to its advantages in utilizing resources, volume, size, electrical ingesting, and high reliability. In this design, specific dispensation units are utilized to expedite complicated operations, while coprocessors for general-purpose applications collaborate with dual or multiple processors to do jobs that demand flexibility. There are actually numerous varieties of coprocessors, including specific to the application combined circuit chips namely, Graphics Processing Units (GPUs) and Field-Programmable Gate Arrays (FPGAs) building. Comparable to Intel Corporation's GPGPUs (general purpose GPUs), Multiple Integrated Cores (MICs) are another type of coprocessor designed for extremely concurrent tasks with massive amounts of statistical data.

A Directed Acyclic Graph (DAG) describes data-driven applications in several domains, including mathematical, the field of engineering, and science. Each of the nodes provide operations, whereas connections indicate messages of communication between activities in the DAG. Development work on HDES machines [3] to reduce the timetable duration of a DAG-based identical technological request is a popular Nondeterministic Polynomial-hard (NP-hard) optimization issue, which has multiple meta-heuristic list schedulers proposed for obtaining optimal solutions. Utilizing decentralized clusters to their greatest potential and provide large capacity to scale, HDES jobs are often implemented and orchestrated via a Message Passing Interface (MPI). It [3] is supported by a variety of Application Programming Interfaces (APIs) or a advanced programming dialects, including Brown Deer Technology's SDK for OpenMP, coprocessing threads (COPRTHR) and OpenCL.
In order to enhance the efficiency of existing algorithms, [4] proposes a hybrid heuristic-genetic method with parameter modifications for dynamic task development in a distributed computation environment. In [5,] a Greedy Ant method is employed for heterogeneous computing process scheduling. This demonstrates a revolutionary scheduling workflow approach called Greedy-Ant that minimizes an application's total time to execution in heterogeneous contexts.

The renowned GA is used for effective concurrent processing on Epiphany multi-core computers. The study in [6] provides an evaluation of Parallella, a tiny board with Epiphany coprocessors made up of 16 MIMD components coupled via a network based-on-a-chip. The outcomes are based on traditional GA. Preparation

operations on HDES processing companies [7] to reduce the schedule length of a DAG-based parallel technological request is a well-known NP-hard optimizing issue, that has multiple heuristic list algorithms for scheduling proposed for producing solutions that are close to optimal.

The goal for this research is to use the meta-heuristic approach to schedule processes in a cluster-based HDES system with HEM. We evaluate and implement the GA, the ACO, the Enhanced Greedy Genetic Algorithm (EGGA), Genetic Simulated Annealing (GSA) and EGACO on the projected HDES design. Relatively than traditional algorithms like GA and ACO, updated heuristic algorithms like GSA are also encompassed. The presumptions of HDES, namely task implementation and architecture-based message models, are presented. Comparisons are made based on the quality of the results, such as scheduling time or duration. This research's contribution can be characterized as follows: developing a system for data-driven HDES, the assessment and implementation of GA-based HDES, and a comparison of HDES with similar efforts.

2. HDES Building Design

Several individuals have access to the N-layer HDES, featuring N HEMs. These devices are linked and communicating via a wireless network device (such as a router, toggle switch, or gateway).
An overall, developing programs on diverse systems involves four layers: the application layer, the API layer, the OS layer, and the computer hardware layer. The suggested diverse group is depicted in Figure 1. In this study, the application layer denotes data-intensive requests. The API interface is a collection of methods or packages that enable clients to handle systems assets and execute programs. For instance, OpenMP has been developed for shared storage and may run on a HEM cluster, while MPI is meant for memory distribution [8], including local recall, external storage, and bus systems. In this framework, multiple buses serve as communication pathways between processing units, including the processor bus (GPP bus), hardware bus (HW bus), network bus (NET bus), coprocessor bus (CO bus) and external bus (EX bus) as illustrated in Figure 1. Every interaction time via various types of busses is taken consideration when programming is displayed later. The OpenCL is an open industry standard for programming a heterogeneous processor which include general-purpose processing units, CO bus, and computer hardware acceleration devices [9]. It enables quick utilization of the influence of the HEM group. Another noteworthy API is the COPRTHR SDK, offering programs and collections [10] that streamline the utilization of the HEM group for builders addressing coprocessors and equipment accelerate. The OS software layer, including integrated Linux, delivers shared functions to programs and serves as a bridge between the API and the computer hardware. Finally, the hardware layer is a system using only one board with various CPUs, busses, memories, and memory. This study focuses on the Parallella board from Adapteva Inc [11]. The hardware layer comprises of a gigabit high-speed Ethernet, 28MB QSPI Flash, coprocessor, 1GB DDR3 RAM, 48GPIOs and micro-HDMI.

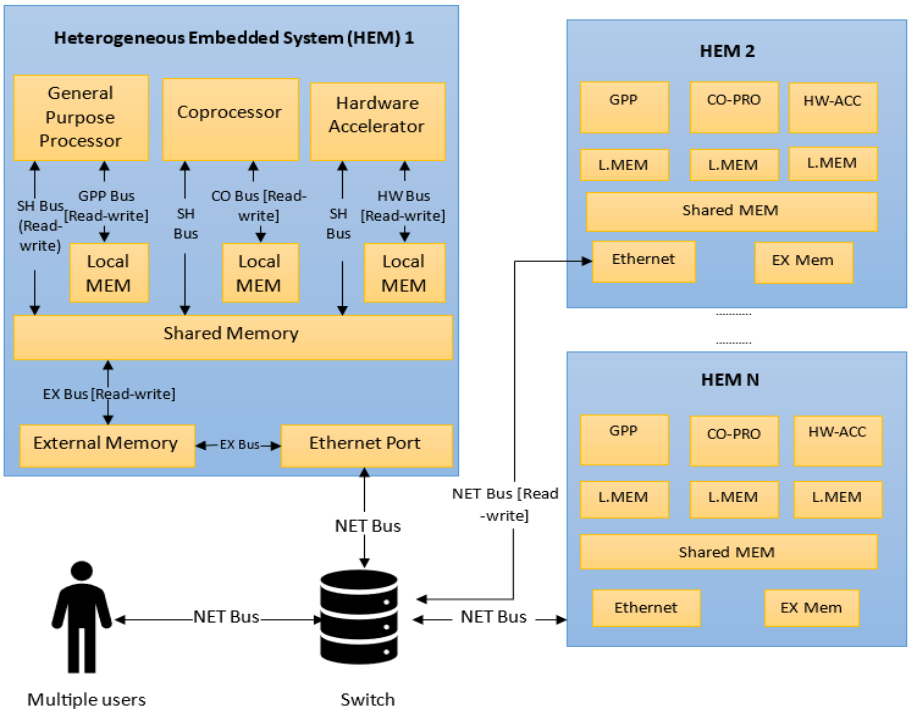

FIGURE 1. A cluster of HEM

3. Proposed Heuristic Algorithms Based On the HDES Framework

This paper provides four meta-heuristic methods for development challenges depending on the suggested HDES framework. The processes are EGGA and EGACO.

3.1 Enhanced Greedy Genetic Algorithm (EGGA) implementation on HDES

To apply EGGA to the pairing challenge, multiple variables must be defined and some adaptations made. EGGA is recommended in six steps, as follows.

The initial step in chromosomal decoding comprises a succession of tasks x = {x_0, x_1,…,x_n} of N layers of HEM group. Every HEM gets three successive DNA, partially within the chromosome, that indicate the arrangement of findings in processing components, namely HW-ACC, GPP and CO-PRO. For instance, a HEM has a succession of activities R = {R_0,R_1,R_2} implemented in GPP, CO-PRO and HW-ACC respectively. The task R_α can be expressed in eq. (1):

$$R_\alpha = \alpha + (n+2), \alpha = 0,1,2; n = 1,2,\dots,N: \quad (1)$$

The second step is to produce an initial population at random as an order of integers within a single chromosome matching the group indexes for a single of the matching DAG.
The third stage involves determining if every suggested remedy signifies a succession of errands in the DAG. The following sequence is designated as GA in order to discover the equivalent topical complement. The goal function for this topical complement is the total of the resemblances between the unmatched components; the bigger the aggregate, the better the fit.

The fourth phase involves selecting organisms from the HDES population (S < n) for reproductive. We use the wheel of fortune to excellent parents from the population. Every DNA k corresponds to the same parental populace with proportion P_k. The roulette wheel is produced by scheming an overall likelihood for all chromosomal as shown in equation (2):

$$P_i = \sum_{K=1}^{i} Q_k,\ i = 0,1,2,\dots,n: \quad (2)$$

where P_i is a increasing likelihood of the i^{th} solution.
For choosing parents, spin the steering wheel times. Each spin generates a random integer between 0 and 1.

The fifth step involves an interaction among the two grandparents. The research work employs a basic technique called randomized one-point crossing. An arbitrary rate is 25 percent. The border function generates a child by randomly selecting cut-points in the two grandparents, duplicating every sequence between the cut opinions in both parents into two new children, one for apiece, and then filling the empty directories, positional astute, from each additional parent.

The single point crossing is a set of two offspring from a 1-D parents
$Parent1$ ($\alpha_0, \alpha_1, \dots, \alpha_N$)and $Parent2$ ($\beta_0, \beta_1, \dots, \beta_N$), where β_N and α_N represent each assignment N. Let ξ represent the border point. βN and αN represent each assignment N. Let ξ represent the border point. Thus, the children of parent develop as follows in equations (3) and (4):

$$Child_0^{new} = (\alpha_0, \alpha_1 \dots, \alpha_\xi, \beta_{\xi+1}, \beta_{\xi+2}, \dots, \beta_N), \quad (3)$$

$$Child_0^{new} = (\beta_0, \beta_1 \dots, \beta_\xi, \alpha_{\xi+1}, \alpha_{\xi+2}, \dots, \alpha_N) \quad (4)$$

The last step is to accomplish the mutation by randomly selecting a place in the genome and probabilistically changing its task via a nonuniform alteration procedure from the pool of genomes. Nonuniform mutation results in a progressively localized search for the best possible solutions, with the groups of genes selected for mutation determined by borders.

Let Sol ($\eta_0, \eta_1, \dots, \eta_i, \eta_N$) be an integrated solution to a problem of optimization and its i choice mutable (η_i) be specific for transformation. Inhomogeneous evolution leads to a modified solution Sol ($\gamma_0, \gamma_1, \dots, \gamma_i, \dots, \gamma_N$), whereby γ_i is intended as follows in eq.(5) and eq.(6):

$$\gamma_i = random\ (\eta_i - h, \eta_i + h), \quad (5)$$

$$H = H_0 \times \frac{iter_{max} - iter_{now}}{iter_{max}}, \quad (6)$$

where H_0 is an original value of $\hbar$, $iter_{now}$ describes as present repetition, and $iter_{max}$ is a maximum iteration.

3.2 Enhanced Greedy ACO (EGACO) implementation on HDES

To use the ACO, this research work needs to establish up a few settings and make some adjustments. The first phase, basic variables, comprises selecting variable values (ρ_1, ρ_2, ρ_3,…, ρ_N) which are selected from a list of predetermined values. Individually choice adjustable i receive a charge from a predetermined list of numbers F_i as follows in eq. (7):

$$F_i = \{\rho_{i,1}, \rho_{i,2}, \dots, \rho_{i,\zeta}, \dots \rho_{i,Y_i}\}, \text{ i = 1,2,… N:} \quad (7)$$

In which set of F_i established values for the decision variable, $\rho_{i,\zeta}$ is a various values n for the choice parameter, and Y_i is a overall amount of alternative values for the choice variable.

The second step is to create a collection of $1 \times N$ that represents the journey of each ant. This set is specified by equation (8):

$$A = \{\chi_1, \chi_2, \chi_3, \dots, \chi_i, .., \chi_N\} \quad (8)$$

where A is a key of the optimization problematic, χ_i the result adjustable of key A, and N is the amount of choice variable quantity.

Assigning a pheromone to the function's objective is the third stage; a pheromone is a collection of values that causes superior solutions to have a greater pheromone intensity than worse solutions. N collections of extent $1 \times h_i$ are attempting to assign pheromones to the choice field so that everyone of them gets allocated to one choice mutable as follows in eq.(9):

$$\rho_i = \{\rho_{i,1}, \rho_{i,2}, \dots, \rho_{i,k}, \dots \rho_{i,k_i}\} \quad (9)$$

where i=1,2,…,N.

In which ρ_i is pheromones vector for the choice mutable. and $\rho_{i,k}$ is pheromone intensity in the k potential values of the i choice variable. The basics of the medium ρ_i are equivalent to zero at the start of the procedure development. The pheromone assignment is accomplished by boosting the pheromone concentrations related to a predetermined selection of effective remedies. The pheromone attentiveness for the conceivable value of the choice mutable changes as in eq. (10).

$$\rho_{i,h}^{new}(1-\rho) \times \rho_{i,k} + \sum_{j=1}^{n} \Delta\rho_{i,k}(j), \quad (10)$$

where $\rho_{i,h}^{new}$ is the new pheromone concentration of the decision variable quantity probable value, ρ is a vanishing rate, and $\Delta\rho_{i,h}(j)$ The number of pheromones put on the probable value of the h choice variable by the jth A.

The fourth stage assigns a probability to the formation of new ants for each choice variable i, based on the pheromone's intensity. A increasing likelihood for all the potential values of all decision variables is as follows in equation (11):

$$\chi_{i,j} = \sum_{q=1}^{g} Q_{i,q}, \quad (11)$$

where i=0,1,2,…,N and j=1,2,…,θ_i.

In which $\chi_{i,}$ describes as increasing likelihood of the j likely value of the i^{th} choice variable.

4. Experimental Results

In the following section, tests are performed to determine ideal HDES solutions by comparing the results utilizing heuristic algorithms. The study's HDES architecture incorporates a 9-layer HEM linked by a system switch with a star topology. This HEM simulates processes using a variety of assignment graphs.

4.1 HDES hardware setup

The HDES equipment ecosystem comprises of nine Parallella embarkations, referred to as HEM, and a 16-port D-Link DGS-1016D broadband switch connected by CAT5e LAN cable. Customers may observe the processer over radiocommunication LAN. The OS is Parabuntu 2016.11.1, entrenched Linux, which is the primary Ubuntu distribution for Parallella, which is open source and available through the www.parallella.org community. The APIs are handled utilizing a combination of Brown Deer Technology's COPRTHR SDK and Adapteva's Epiphany SDK.

4.2 Simulation Results

Utilizing the master-slave approach, MPI allows meta-heuristic algorithms to run on specific HEMs. The HEM master collects outcomes from different tasks with graph sizes in this HES kind of algorithm. The most effective solutions are combined and evaluated based on the basis of makespan, accelerate, and SLR criteria.

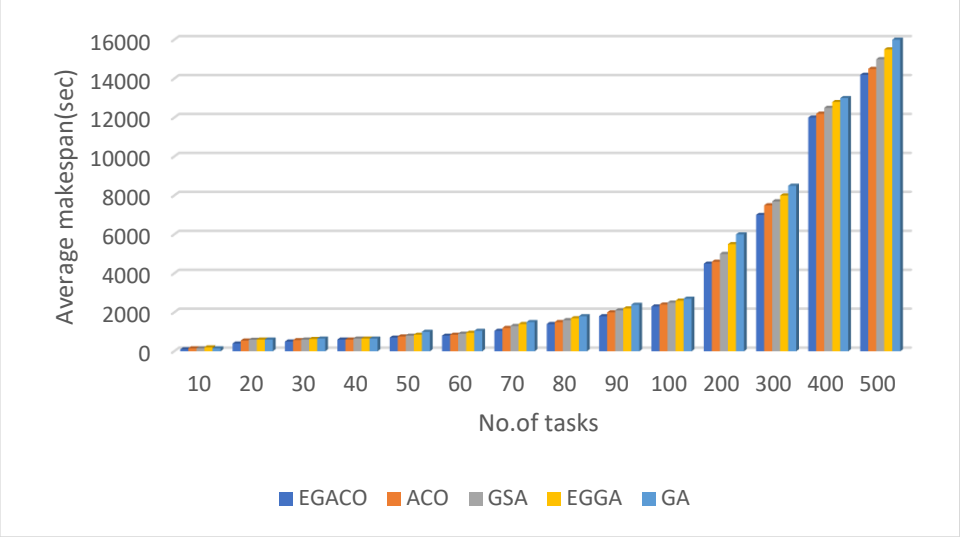

FIGURE 2. The median makespan has been compared to several algorithms on randomized task graphs.

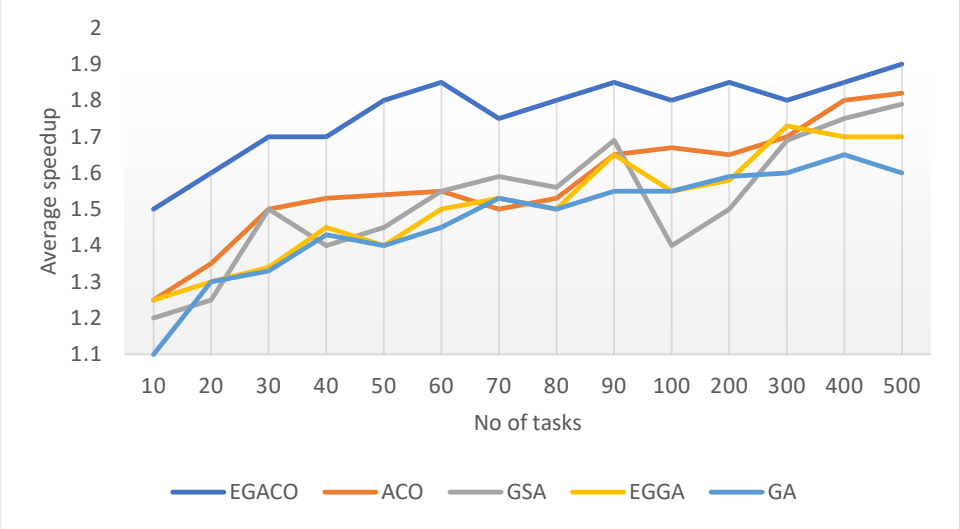

FIGURE 3. The median speedup compared with several algorithms on randomized job graphs

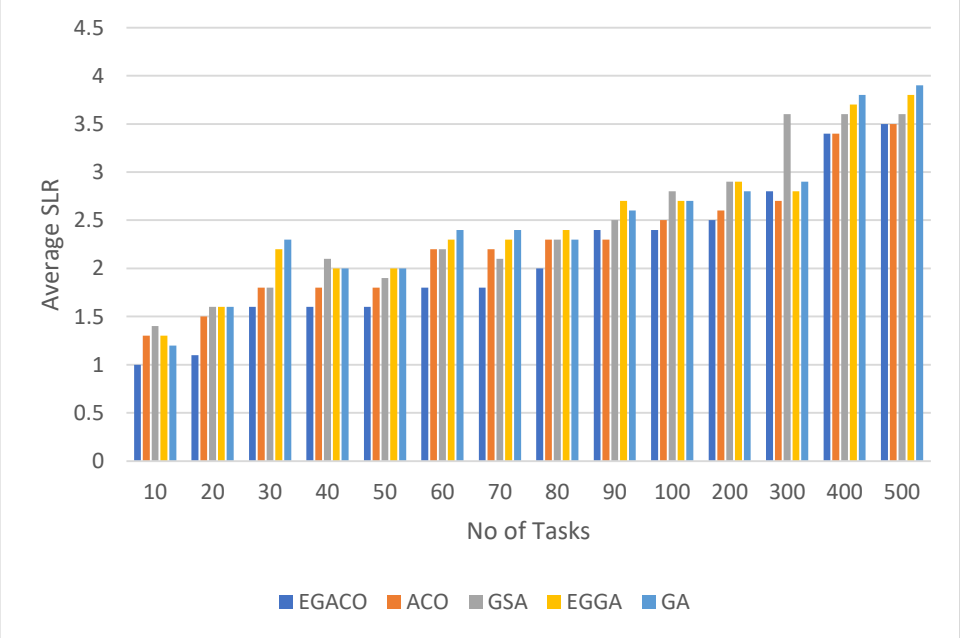

FIGURE 4. The median SLR is evaluated with various techniques on randomized task graphs

Figure 2 depicts the normal the duration of all DAG chart sizes calculated using ACO, GA, EGGA, EGACO and GSA. The GSA method [12] use the SA process in the component of the change process based on GA that determines the optimum value. EGACO has the best overall average the duration at 240 minutes, followed by ACO (247 minutes), GSA (251 minutes), EGGA (259 minutes), and GA (266 minutes).
In Figure 3, the suggested EGACO outperforms the previous algorithms. When compared to GA and GSA, EGACO progresses speed by 30% for minor chart sizes. When the number of tasks upsurges, EGACO exhibits a significant advantage. When the number of tasks exceeds 30, EGACO's performance converges swiftly at the steady state. Although ACO outperforms GSA, EGGA, and GA, its performance declines once the task count approaches 50. When the number of tasks is equivalent to 80, EGACO boosts speed by approximately 33%.

Figure 4 depicts the SLR's evaluations on random task graphs. It's evident that EGACO outperforms the other algorithms. When n = 60, 70, or 80, the bottom boundary of SLR is achieved, and all algorithms are about equivalent. In terms of speeding up and SLR, the overall findings show that EGACO beats ACO, GSA, EGGA, and GA in determining optimal task scheduling in the suggested HDES design.
All algorithms used in the research imitate an early mission order that is created at random. GA, GSA, and EGGA are genetic algorithms, whereas EGACO and ACO use the ant colony approach to generate lists. Figures 2, 3, and 4 clearly show that EGACO surpasses ACO, GSA, EGGA, and GA on criteria such as duration, speedup, and SLR.

As can be observed, inserting a basic greedy algorithm into the choosing stage of standard heuristic algorithms improves the efficiency of the algorithm searching for a higher quality solution. It typically finds an almost-optimal answer in polynomial time. In theory, [13] elaborates on the presentation examination of employing greedy algorithms to tackle basic acyclic network optimization problems. For the majority of problems, the proportion of a viable solution to a best resolution is never greater than two.
This study introduces a modest greedy method in meta-heuristic penetrating for a through make span, which allows EGACO to produce improved development consequences than GA and EGGA since an ant colony can assist in determining the optimum trail. As the difficulty of the commission chart increases, it develops more difficult for them to give reliable outcomes across a range of graphs. The cost function addresses resource limits since the number of programmable cells and coprocessors is restricted.

5. Conclusion

The EGGA is examined in this study, and EGACO suggests a method for based workflow scheduling. It is suggested to use a new meta-heuristic information based on forward dependency to make it more likely that ACO will establish task priorities. Additionally, a greedy machine allocation strategy is employed to finish scheduling. Experimental results in the HEM cluster indicate that the modified greedy ant colony optimization algorithm outperforms the others by 33% in terms of result quality, according to random task graphs. The actual implementation of data-dominated applications in this HDES utilizing the best scheduling outcomes would be the subject of future research.

References

[1]. M. Zahran, *Heterogeneous Computing: Hardware and Software Perspectives*, Vol.1, ACM Books, New York, N.Y.2019.
[2]. J. Huang, R. Li, J. An, D. Ntalasha, F. Yang and K. Li, “Energy-Efficient Resource Utilization for Heterogeneous Embedded Computing Systems”, *IEEE Trans. on Computers*, Vol.66, No.9, pp.1518-1531, 2017.
[3]. S. Prongnuch and T. Wiangtong, “Heterogeneous Computing Platform for data processing”, In: *Proc. ISPACS*, pp.1-4, 2016.
[4]. S. Ding, J. Wu, G. Xie, and G. Zeng, “A Hybrid Heuristic-Genetic Algorithm with Adaptive Parameters for Static Task Scheduling in Heterogeneous Computing System”, In: *Proc. IEEE Trustcom/ BigDataSE/ICESS*, pp.761-766, 2017.
[5]. B. Xiang, B. Zhang, and L. Zhang, “Greedy-Ant: Ant Colony System-Inspired Workflow Scheduling for Heterogeneous Computing”, *IEEE Access*, Vol. 5, pp.11404-11412, 2017.
[6]. L. Faber and K. Boryczko, “Efficient Parallel Execution of Genetic Algorithms on Epiphany Manycore Processor”, In: *Proc. FedCSIS*, pp.865-872, 2016.
[7]. G. Xie, G. Zeng, X. Xiao, R. Li and K. Li, “Energy-Efficient Scheduling Algorithms for Real-Time Parallel Applications on Heterogeneous Distributed Embedded Systems”, *IEEE Trans. on Parallel and Distributed Systems*, Vol.28, No.12, pp.3426-3442, 2017.

[8]. P. Czarnul, *Parallel Programming for Modern High Performance Computing Systems*, Vol.1, Chapman and Hall/CRC, New York, N.Y.2018.
[9]. L. Howes, *The OpenCL Specification, Version: 2.1, Khronos OpenCL Working Group*, [Online] Available: https://www.khronos.org/registry/ OpenCL/specs/opencl-2.1.pdf. Accessed on: Sep. 2, 2019.
[10]. Brown Deer Technology, *COPRTHR-2 Overview*, [Online]. Available: http://www. browndeertechnology.com/docs/COPRTHR-2 Overview rev-b-20160629.pdf. Accessed on: Sep. 2, 2019.
[11]. Adapteva, *Parallella-1.x Reference Manual*, [Online]. Available: https://www.parallella.org /docs/parallella manual.pdf. Accessed on: Sep.2, 2019.
[12]. M. Zhao, X. Yin, and H. Yue, "Genetic Simulated Annealing-Based Kernel Vector Quantization Algorithm", *International Journal of Pattern Recognition and Artificial Intelligence*, Vol.31, No.5, pp. 1-28, 2017.
[13]. L. C. Thanh, "Performance Analysis of Greedy Algorithms for Max-IS and Min-Maxl-Match", *Vietnam Journal of Mathematics*, Vol.36, No.3, pp.327-336, 2008.