A NEW PARALLEL MATRIX MULTIPLICATION ALGORITHM ON HEX-CELL NETWORK (PMMHC) USING IMAN1 SUPERCOMPUTER

Enas Rawashdeh[1], Mohammad Qatawneh[1] and Hussein A. Al Ofeishat[2]

[1] Department of Computer Science, King Abdullah II School for Information
Technology, The University of Jordan,
[2]Al-Balqa Applied University-Jordan.

ABSTRACT

A widespread attention has been paid in parallelizing algorithms for computationally intensive applications. In this paper, we propose a new parallel Matrix multiplication on the Hex-cell interconnection network. The proposed algorithm has been evaluated and compared with sequential algorithm in terms of speedup, and efficiency using IMAN1, where a set of simulation runs, carried out on different input data distributions with different sizes. Thus, simulation results supported the theoretical analysis and meet the expectations in which they show good performance in terms of speedup and efficiency.

KEYWORDS

Parallel processing, matrix multiplication, Interconnection Network, Hex-Cell.

1. INTRODUCTION

Matrix multiplication is commonly used in many areas like graph theory, residue-level protein folding [4], numerical algorithms, digital image processing and others. Working with matrix multiplication algorithm of huge matrices requires a lot of computation time where the complexity time for sequential matrix multiplication algorithm is O (n^3), where n is the dimension of the matrix. Because higher computational throughputs are required with the applications, many parallel algorithms based on sequential algorithms are developed to improve the performance of matrix multiplication algorithm. There a lot of improvement [7, 8] done on sequential algorithms to follow the big requirements but still has shown a limitation in performance. For that, parallel approaches have been examined and enhanced for decades.

In common parallel matrix multiplication algorithms used decomposition of matrices depends on the number of processors available in the interconnection network [10, 9]. Each algorithms use the matrices that decomposed into sub matrices (blocks). During execution process of matrix multiplication, each processor calculates a partial multiplication result using the sub matrices that are currently accessed by it. When the multiplication is completed, the coordinator processor assembles and generates the complete matrix multiplication result.

The interconnection networks are the core of a parallel processing system which the system's processors are linked. Due to the big role played by the networks topology to improve the parallel system's performance, Several interconnection network topologies have been proposed for that purpose; such as the tree, hypercube, mesh, ring, and Hex-Cell (HC) [1, 2, 5, 6, 11, 12, 14, 15, 18].

DOI:10.5121/ijcsit.2017.9503

Among the wide variety of interconnection networks structures proposed for parallel computing systems is Hex-Cell network which received much attention due to the attractive properties inherited in their topology [1, 16, 17].

The proposed parallel matrix multiplication on the Hex-cell network is implemented by the library Message Passing Interface MPI, where MPI processes are assigned to the cores. If the MPI process is assigned to a core, then it will be parallel computation; but if more than one MPI process is assigned to the same core, then it will be concurrent computation. Experimentation of the proposed algorithm was conducted using IMAN1 supercomputer which is Jordan's first supercomputer. The IMAN1 is available for use by academia and industry in Jordan and the region.

The rest of the paper is organized as follows. Section 2 describes the definition of Hex-Cell network. Section 3 presents the proposed algorithm. Section 4 provides an Analytical Evaluation. Section 5 provides the performance results, and Section 6 summarizes and concludes the paper.

2.DEFINITION OF HEX-CELL NETWORK TOPOLOGY

Hex-Cell network is one of interconnection networks structures proposed for parallel computing systems where the nodes are connected with each other in hexagonal topology. A Hex-Cell network with depth d is denoted by HC(d) and can be constructed by using units of hexagon cells, each of six nodes. A Hex-Cell network with depth d has d levels numbered from 1 to d, as shown in Figure 1:

- **Level 1** states the innermost level corresponding to one hexagon cell.
- **Level 2** correlate with the six hexagon cells surrounding the hexagon at level 1.
- **Level 3** correlate with the 12 hexagon cells surrounding the six hexagons at level 2.

The levels of Hex-Cell network with depth d are labeled from 1 to d. Each level i has N_i nodes, representing processing elements and interconnected in a ring structure [1].

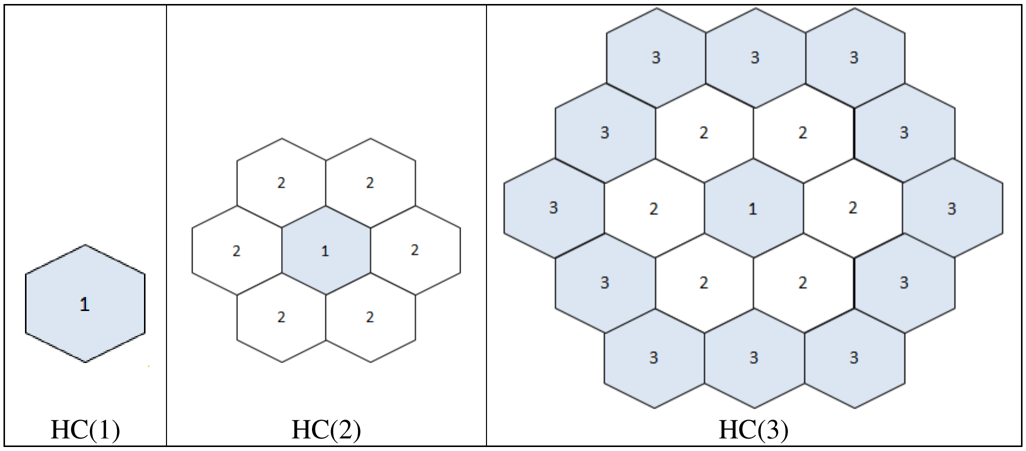

Figure 1. Hex-cell network in different level one, two and three [1].

The address of each node in the Hex-Cell topology is identified by (S,L,Y) where S denotes the section number, L denotes the level number, and Y denotes the node number on that level labeled from $Y_1,\ldots, Y_n$; where n = ((2×L) - 1) [1].

A node with the address 1.1.1 is the first node that exists at the section number 1 and level number 1, and address 6.1.1 is first node that exists at the section number 6 and level number 1, as shown in Figure 2.

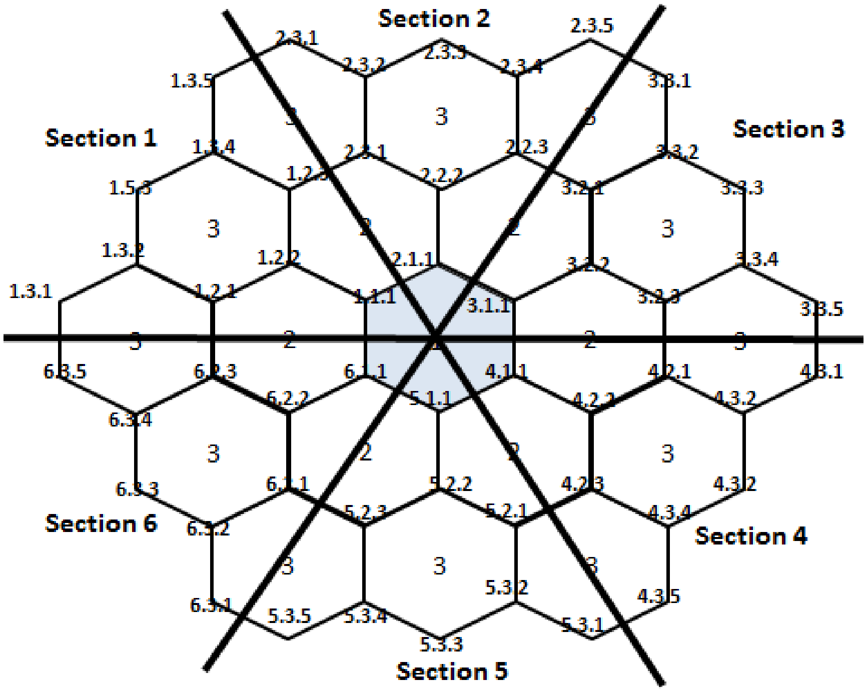

Figure 2. Hex-Cell addressing scheme by section [5].

3.PMMHC Algorithm

In this section, we propose a new Parallel Matrix Multiplication Algorithm on Hex-Cell Network (PMMHC) as shown in Figure 4. The aim behind the parallelism of the matrix multiplication is to make the algorithm runs faster and more efficient in comparison with the sequential one for very large data matrices. It depends on partitioning matrices of size n into a set of partitions; each partition is assigned to a separate processor to multiply sequentially using sequential matrix multiplication. Thus, the number of partitions depends on the number of the available processors.

In this paper, we apply matrix multiplication on the Hex-cell interconnection network topology. The hex-cell network [1] is divided into six sections as shown in Figure 2. The proposed algorithm uses each section as ring topology and the root nodes of level 0 depend on one to all personalized broadcast for child's nodes. As shown in figure 4, the proposed work is assumed that a matrices data is stored in the main coordinator processor (MC), which it will be partitioned, multiply, and then combined at the main coordinator processor. And L0-HC nodes are level 0 ring nodes of Hex-Cell network; L1-Ring coordinators are the root nodes of each ring section correspond each one with the nodes of level 0.

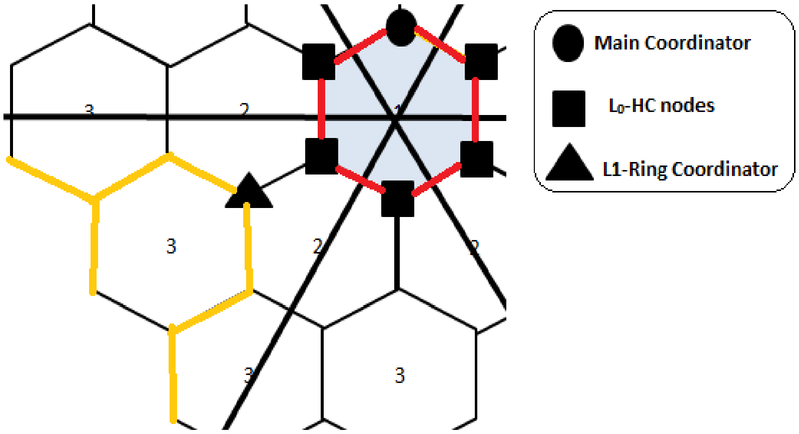

Figure 3. PMMHC algorithm Hex-Cell section.

```
Input: Matrix A and B
Output: Matrix C on Hex-Cell using parallel Matrix Multiplication
Phase 1: Data Distribution Phase
Data Distribution in the Hex-Cell root nodes at L0.
1.  MC (Main Coordinator) generates a set of blocks of Matrix A.
2.  MC generates a set of blocks according to the Matrix B.
3.  MC routes the $A_{ik}$ and $B_{kj}$ values internally to all $L_0$-HC nodes on $L_0$-Ring.
4.  For all processors in $L_0$ (that received blocks of matrices in the previous steps), do the
    following in parallel: Send the blocks of matrices A and B to the $L_1$-RCs ($L_1$-Ring
    Coordinator) of the connected ring.
5.      Wait until the coordinator who received the data will send an acknowledgment message.
6.      Send a message for the MC informing that the process completed.
7.      MC stops the process of distribution and announces the beginning of the next step.
L1-Ring Distribution of Data
8.  For all ring coordinators $L_1$-RCs, do the following in parallel:
9.    Blocks of matrix A is partitioned into a number of horizontal stripes.
10.   Blocks of matrix B is presented as a set of vertical stripes.
11.   Send stripes for all processors in each Ring in $L_1$.
12.   Stop the process of distribution and announce the beginning of the next step.
Phase 2: Data Multiplication Phase
13. For all processors in each $L_1$-Rings, do the following in parallel:
14. Multiply the stripes of matrix A with stripes of Matrix B (for each block) of data using
    sequential matrix Multiplication. Where all processors perform $C_{ij} = \sum_{K=0}^{K-1} A_{ik} B_{Kj}$.
Phase 3: Data Combining Phase
L1-Ring Data Combining
15. For all $L_1$-RCs, do in parallel:
16.   Combine the collected multiplication in one matrix.
Global Data Combining
17. For all level 1-Ring coordinators ($L_1$-RCs) in the Hex-Cell interconnection, do the following
    in parallel:
18.   Send the multiplication matrix to the Hex-Cell root nodes at L0.
Combining Data in the Hex-Cell root nodes
19. MC combines the collected matrices correctly from $L_0$-HC roots nodes in matrix C.
```

Figure 4. The PMMHC algorithm

The parallel matrix multiplication on Hex-Cell interconnection network in Figure 4 is illustrated in more details as follows:

Phase 1*:* Data Distribution Phase.

Assume I×K matrix A and a K×J matrix B, and the whole matrices A_{ik} and B_{kj} stored on MC (main coordinator). The distribution phase is composed of three steps as follows (see Figure 4):

- **Data Distribution in the Main Ring** (Lines 1-4 in Figure 4). The MC starts the process of data decomposition the initial matrices *A* and *B*. We assume all the matrices are square of n×n size, the number of vertical blocks and the number of horizontal blocks are the same and are equal to q and the size of all block is equal to v×v, v=n/q.

- **Global Distribution of Data** (Lines 5-7 in Figure 4). The nodes (in parallel) start sending the partitions through the optical links. As shown in Figure 5, the nodes $\{N_0, N_1, N_2, N_3, N_4, N_5\}$ will send their partitions to their directly connected neighbors; rings $\{R0, R_1, R_2, R_3, R_4, R_5\}$, respectively. It is important to note that each node in the main group (L_0-Ring) receives an acknowledgement message from its neighbor in the other ring after the process is completed. Consequently, each node in the main ring (L_0-Ring) sends a message to MC telling that the process was completed. When MC receives messages from all the processors, who participated in the global distribution steps, it announces the beginning of the next step, which is the ring distribution.

- **Ring Distribution of Data** (Lines 8-12 in Figure 3). In this step each L_1-RC makes blocks of matrix A as a number of horizontal stripes, and matrix B is presented as a set of vertical stripes. The stripe size should be equal to v=n/p (assuming that n is divisible by p), as it will make possible to provide equal distribution of the computational load among the processors.

Phase 2*:* Multiplication Phase (Lines 13-14 in Figure 3).

- All the elementary processors (nodes) in the interconnection network apply sequential matrix multiplies a stripes of A by stripes of B .The processor computes it's part of the product to produce a block of rows of C, as

```
For $i$ from 1 to $n$:
    For $j$ from 1 to $p$:
        Let sum = 0
        For $k$ from 1 to $m$:
            Set sum ← sum + $A_{ik} \times B_{kj}$
        Set $C_{ij}$ ← sum
```

Phase 3*:* Data Combining Phase.

Combining phase is parallelized by reversing the order of steps in the distribution phase as follows:

- **Level 1-Ring Data Combining** (Lines 15-16 in Figure.3). The aim of this step is to combine all the result of multiplication for the Hex-Cell sections via electronic links. This is done by first collecting the multiplication from the elementary processors of L_1-Ring for each section on hex-cell network and stores the first combined partitions in the RCs of each section.
- **Global Data Combining (Lines 17-18 in Figure 3).** All RCs in the whole section of interconnection network will send their chunks of multiplication data via optical links to their corresponding processors in the main Ring (L_0-Ring).

- **Combining Data in the Main Ring (Lines 19 in Figure 3).** MC collects the whole set of data multiplication by combines all partitions in one matrix called *C*.

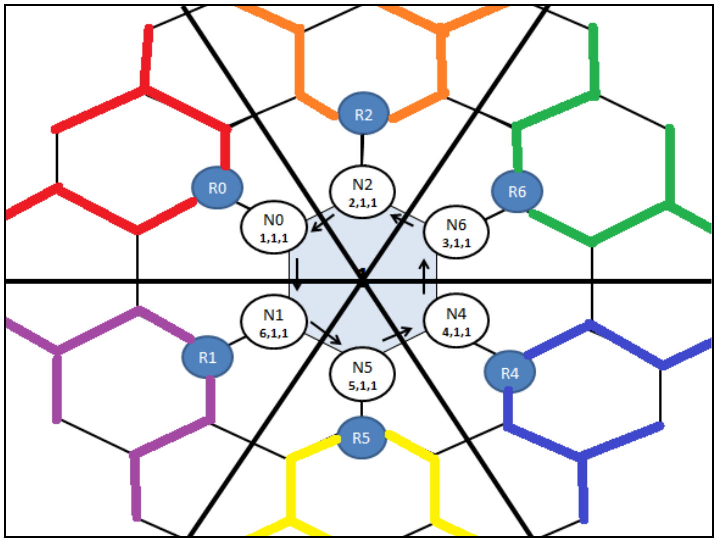

Figure 5. Nodes on ring topology in proposed algorithm

4.ANALYTICAL EVALUATION

This section provides the analytical evaluation of the proposed (PMMHC) parallel Matrix multiplication on Hex-Cell interconnection network. Three performance metrics are used to evaluate the algorithm, namely: Run time complexity, speedup and efficiency.

4.1 Run time complexity

Time complexities of distribution phase in PMMHC is the same as complexity of One to all personalized in L_0-Ring and L_1-Ring with the different ($\frac{N}{p}$) chunk for each processor, and the time complexities of combining phase in PMMHC the same as complexity of All- to one personalized for L_1-Ring and L_0-Ring, So, the total Time communication in the matrix multiplication on Hex-Cell network is: $\frac{N}{p}(p-1)$

Time Complexity of Computation for each processor will multiply $\frac{N}{p}$ elements using sequential Matrix multiplication= ($\frac{N^3}{p}$). So, the total Time complexity of the proposed algorithm is:

$$\frac{N^3}{p}+\frac{N}{p}(p-1)$$

4.2 Speedup

Speedup is one of the performance metrics used in the evaluation of parallel algorithms in general. It evaluates the performance of a parallel algorithm in comparison with its sequential counterpart [3]. The speedup of the PMMHC network is shown in Equation 1.

$$Speedup=\frac{N^3}{\frac{N^3}{p}+\frac{N}{p}(p-1)} \qquad (1)$$

4.3 Efficiency

The efficiency is another performance metric that is widely used to assess the performance improvement in parallel algorithms in general. Its value represents an indicator on how much do the processors being utilized [3]. The efficiency of the PMMHC network is shown in Equation 2.

$$\text{Efficiency} = \frac{N^3}{N^3+N(p-1)} \qquad (2)$$

5.EXPERIMENTAL RESULTS AND PERFORMANCE EVALUATION

In this section, the results of different simulation runs over different data distributions are presented. Table 1 show the results of speedup for different datasets in which you can observe that in general, the result of speedup is better with large matrices to multiply. IMAN1 Zaina cluster is used to conduct our experiments and open MPI library is used in our implementation of the following parallel matrix multiplication algorithms; and the experimental runs on a dual quad core intel xeon Cpu with smp, 16 gb ram, where the software specification is conducted on scientific linux 6.4 with open mpi 1.5.4, C and C++ compiler.

Table 1 shows architectural information about the Hex-Cell interconnection network. Also, it shows information about the expected size of the input data that can be assigned for each group in a lucky-case partitioning, when applying the parallel matrix multiplication on the Hex-Cell interconnection network.

Table 1. Experimental results of proposed algorithm

Matrix Size	2 processors		4 processors		8 processors		16 processors		32 processors	
	Time	Speed up	Time	Speed Up	Time	Speed Up	Time	Speed Up	Time	Speed Up
500	2.7654	0.312829	1.4121	0.612633	0.9814	0.881495	0.824	1.049878	1.021	0.8473065
1000	7.9574	1.404818	6.0367	1.851789	2.5272	4.423353	2.628	4.253691	2.769	4.0370892
2000	77.629	1.44340	27.47	4.078998	22.543	4.970505	17.92	6.252795	5.256	21.318512
3000	189.21	1.733187	67.23	4.877932	54.698	5.995528	69.32	4.730862	33.78	9.7070625
4000	531.09	1.169383	182.54	3.402255	83.826	7.408772	81.82	7.590415	49.25	12.610107
5000	622.36	1.815913	198.63	5.689735	108.32	10.43345	88.17	12.81787	79.14	14.280417

Figure 5 shows the speedup for the proposed algorithm according to different matrices sizes. All results are performed on a different number of processors. Where with the data size increases, the run time increases due to the increased number of multiplication and the increased time required for data combining. The size of data assigned to each processor plays a primary role in obtaining the highest speedup values. This means that the ratio between the data size and the number of processors can be considered as an indicator of whether we can obtain a high speedup value or not.

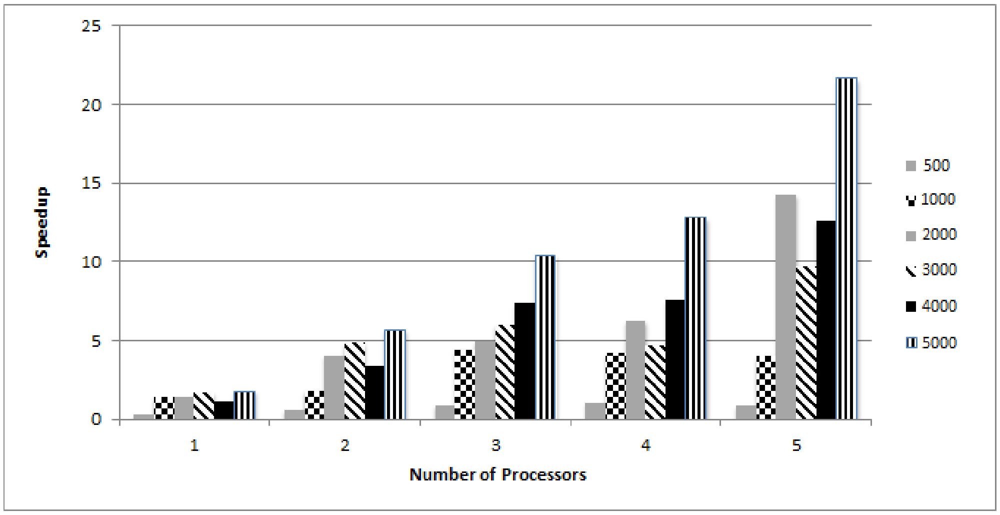

Figure 5. Number of processors versus Speed Up

6. CONCLUSIONS

In this paper, we present a parallel matrix multiplication on Hex-Cell interconnection network. The proposed parallel matrix multiplication algorithm was simulated over different number of processors, with different sizes of matrices, where the algorithm comprises three phases to be applied on the Hex-Cell interconnection network. These phases are the distribution phase, the multiplication phase using the sequential matrix multiplication, and finally the combining phase. Actually, these phases can be easily modified to suit other application that requires massive data to be manipulated. However, the parallel matrix multiplication on Hex-Cell interconnection network shows higher performance in comparison with its sequential version on a single processor.

As a part of our future work, we aim to conduct a comparative study by applying the matrix multiplication over different interconnection networks. We also aim to extend this study by applying sorting algorithms on Hex-Cell interconnection such as merge sort and quick sort.

ACKNOWLEDGEMENTS

We thank Eng. Zaid Abudayyeh for assistance with to accomplish this research.

REFERENCES

[1] LEE, S.HYUN. & KIM MI NA, (2008) "THIS IS MY PAPER", ABC TRANSACTIONS ON ECE, VOL. 10, NO. 5, PP120-122.

[2] SHARIEH, M. QATAWNEH, W. ALMOBAIDEEN, AND A. SLEIT, (2008)"HEX-CELL: MODELING, TOPOLOGICAL PROPERTIES AND ROUTING ALGORITHM", EUROPEAN JOURNAL OF SCIENTIFIC RESEARCH, VOL. 22, NO. 2.

[3] MOHAMMAD, Q. AND KHATTAB, H. (2015) NEW ROUTING ALGORITHM FOR HEX-CELL NETWORK. INTERNATIONAL JOURNAL OF FUTURE GENERATION COMMUNICATION AND NETWORKING, 8, 295-306.

[4] ANANTH GRAMA, GEORGE KARYPIS, VIPIN KUMAR, ANSHUL GUPTA" INTRODUCTION TO PARALLEL COMPUTING", 2ND ED, THE MIT PRESS.

[5] SERGEY V. VENEV , KONSTANTIN B. ZELDOVICH.(2015) "MASSIVELY PARALLEL SAMPLING OF LATTICE PROTEINS REVEALS FOUNDATIONS OF THERMAL ADAPTATION",THE JOURNAL OF CHEMICAL PHYSICS 143, 055101.

[6] Qatawneh, M., Alamoush, A. and Alqatawna, J. (2015) Section Based Hex-Cell Routing Algorithm (SBHCR). International Journal of Computer Networks and Communications, 7, 167-177.

[7] Mohammad, Qatawneh. (2006) "Adaptive fault tolerant routing algorithm for Tree-Hypercube multicomputer." Journal of Computer Science 2, no. 2.

[8] Ziad Alqadi and Amjad Abu-Jazzar, (2005). “Analysis of program methods used for optimizing matrix multiplication”, J. Eng., Vol. 15, NO. 1: 73-78.

[9] Dongarra, J.J., R.A. Van de Geijn and D.W. Walker,(1994). “Scalability Issues Affecting the Design of a Dense Linear Algebra Library”, J. Parallel and Distributed Computing, Vol. 22, No. 3, Sept., pp:523-537.

[10]Chtchelkanova, A., J. Gunnels, G. Morrow, J. Overfelt, R. van de Geijn,)1995(. "Parallel Implementation of BLAS: General Techniques for Level 3 BLAS", TR-95-40, Department of Computer Sciences, University of Texas, Oct.

[11]Choi, J., J.J. Dongarr,(1992), “BLAS for distributed memory concurrent computers” ,a and D.W. Walker, Level 3. CNRS-NSF Workshop on Environments and Tools for Parallel Scientific Computing, Saint Hilaire du Touvet, France, Sept. 7-8, Elsevier Sci. Publishers.

[11]Mahafzah, B., Sleit, A., Hamad, N., Ahmad, E., and Abu-Kabeer, T. (2012). “The OTIS hyper hexa-cell optoelectronic architecture”. Computing, 94(5), 411-432.

[12]Grama, Ananth, Ed.(2003).” Introduction to parallel computing”. Pearson Education.

[13]Maha Saadeh, Huda Saadeh, and Mohammad Qatawneh. (2016). “Performance Evaluation of Parallel Sorting Algorithms on IMAN1 Supercomputer.”, International Journal of Advanced Science and Technology, 95, pp. 57-72.

[14]Qatawneh Mohammed.(2005). “Embedding Linear Array Network into the tree-hypercube Network.”, European Journal of Scientific Research, 10(2), pp. 72-76.

[15]Mohammad Qatawneh. (2011). “Multilayer Hex-Cells: A New Class of Hex-Cell Interconnection Networks for Massively Parallel Systems”, International journal of Communications, Network and System Sciences, 4(11).

[16]Mohammad Qatawneh. (2011).“Embedding Binary Tree and Bus into Hex-Cell Interconnection Network.”, Journal of American Science, 7(12).

[17]Mohammad Qatawneh. (2016). “New Efficient Algorithm for Mapping Linear Array into Hex-Cell Network”, International Journal of Advanced Science and Technology, 90.

[18]Anna Syberfeldt and Tom Ekblom (2017). “A Comparative Evaluation Of The GPU Vs. The CPU For Parallelization Of Evolutionary Algorithms Through Multiple Independent Runs”, IJCSIT, Vol 9, No 3, June 2017.