Biological and Combinatorial Problems Exploration Using Parallel and Evolutionary Computing

Fahmida Hossain
Department of Computer Science & Engineering, University of Dhaka
Dhaka-1000, Bangladesh
Email: fahmidacsedu@gmail.com

Md. Shamsujjoha
Department of Computer Science & Engineering, East West University
Dhaka-1212, Bangladesh
Email: dishacse@yahoo.com

Md. Nawab Yousuf Ali
Department of Computer Science & Engineering, East West University
Dhaka-1212, Bangladesh
Email: nawab@ewubd.edu

***Abstract*—J. Conway's renowned evolutionary problem Game of Life attracts researchers for its capability to transfer the deputized patterns from a generation to its next. In this paper, we propose an efficient solution to this Game problem. In the implementation of the proposed solution a higher abstraction has been used to partition the program into serial and parallel regions. This abstraction improves the performances of the proposed scheme over the existing conventional schemes for all possible design approaches. The paper also illustrates the possible enhancements of the proposed solution to solve the complex combinatorial and biological problems of the real world.**

***Keywords*—Bio-informatics, Game of Life, Gene Evocation, Open-MP, Parallel Computing, Pattern Recognition.**

I. Introduction

The Game of Life is a cellular automaton invented by a British mathematician John Horton Conway in 1970 [1]. The Game is a study of elaboration of patterns and behaviors that emerged from few rules. The rules from the Game can be modified to see the evaluation and evocation during the lifespan of the genes *i.e.*, changes in the behavior of the genes from the generation to generation for different patterns [2]. It can even help us to understand the diversity of chromosomes life that has evolved. Actually, this game problem lets us observe a simple system of life where we know all the rules in advance. These rules can be modified on demand for the enhancement related to the problems. The universe of the Game is an infinite, two-dimensional rectangular grid. However, for the computational convention, $n \times n$ grid is considered where $n \epsilon N$, the set of natural numbers. The population of the Game is a collection of grid cells that are marked as dead or alive. The value of the dead and live cell are considered as Zero and One, respectively. The population evolves at discrete time steps known as generations [3]. The value of a cell in the next generation is determined by the value of its eight nearest neighbors[1]and the following rule:

"*A live cell with two alive neighbors, or any cell with three alive neighbors remains alive at the next step. Every other cell will be dead or remains dead at the next step*".

[1]Let the location of cell be [*i*, *j*] in a two dimensional space. Then, its eight nearest neighbors are [*i*-1, *j*+1], [*i*, *j*+1], [*i*+1, *j*+1], [*i*-1, *j*], [*i*+1, *j*], [*i*-1, *j*-1], [*i*, *j*-1], [*i*+1, *j*-1], where both i, j are positive integers. In a finite $n \times n$ grid, there may have few unavailable cells which are considered as the dead cells.

More Specifically:

- Any live cell with fewer than two alive neighbors died in the next generation because of loneliness.
- Any live cell with two or three alive neighbors remains live on to the next generation.
- Any live cell with more than three alive neighbors died because of overpopulation or congestion.
- Any dead cell with exactly three alive neighbors turns to a live cell because of reproduction.

Because of these simple rules and its capability to find the pattern, the game problem can be used to introduced power and elegance of design patterns [4], hardware modeling of cellular automata [5], eye-tracking interface [6], feedback networks [7], polynomial neural networks [8], hyperbolic domain [9], explorations in ecology [10] etc. In these consequences, this paper presents an efficient method to solve the Game of Life using Open-MP. Open-MP is a standard de-facto parallel programming interface. It provides a thread level abstraction to partition a program into serial and parallel regions. In a multi core machine, it feeds the cores on each node. On the other hand, it deploys a level of threads for compaction in single core machine [11]. Moreover, in Open-MP any number of parallel constructs can be specified in a single program. Parallel constructs can also be nested but it depends on the implementation. Thus, it is especially important where multiple threads attempt to read or write from the same file and is up to the programmer to ensure that input-output is conducted correctly within the context of a multi-threaded program [12]. More detail on Open-MP and the necessary background required to understand the proposed work are detailed in Sec. 2. Then, the paper illustrates the design methodology of the proposed algorithms in Sec. 3. The enhancements of the proposed method are depicted in Sec 4 that includes management of power streams [13], identification of network traffic delay [14], ground water flow [15], and abnormal/uncontrolled cell division [16] etc. Performance of the proposed schemes is evaluated in Sec. 5. Finally, a conclusion is drawn in Sec. 6.

978-1-5090-4417-7/16/$31.00 ©2016 IEEE

II. Background Study and Literature Review

This section illustrates the necessary basic definitions. The detail on working procedure and implementation issues of Open-MP are discussed in Section A. The, Section B shows the exemplification of the game of life.

A. Open-MP

Generally Open-MP is used to specify shared memory parallelism for C/C++ and Fortran programs. However, it can also operate on distributed memory systems with additional layer of software. The additional layer is needed to manage the memory coherency. It uses the fork-join model of parallel execution. Every Open-MP program begins execution with a single thread referred as master thread. When the master thread encounters a parallel region, it forks additional worker threads as shown in Fig. 1 [17], [18], [19], [20], [21].

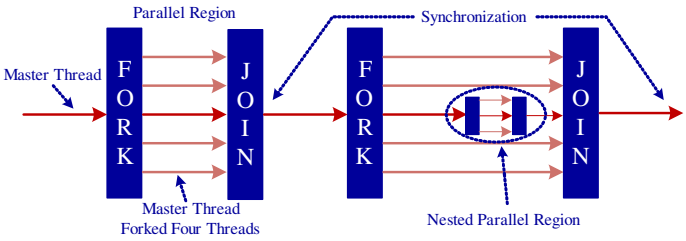

Figure 1. Fork Join Model of Open-MP [22].

The recent release of the Open-MP API can be found in the specifications page [23]. Two tutorial on Open-MP is presented in [12], [24]. The tutorials broadly discuss monitoring, debugging and performance analysis on Open-MP. A set of test suites and benchmarks on Open-MP can be found in [25], [26]. A survey on performance tools for Open-MP is discussed in [27], which evaluates the performance of Open-MP constructs, kernels and application program on multi-core systems. The runtime performance monitoring interface for Open-MP is presented in [20]. Another performance evaluation of a multi-zone application in different Open-MP approaches is presented in [28]. Performance analysis of hybrid Open-MP/MPI n-body application is presented in [29]. These three articles used time and memory as the major performance evaluation criteria. The shared memory parallel programming performance analysis in Open-MP is shown in [11], [30]. Efficient implementation of Open-MP for clusters with implicit data distribution can be found in [31].

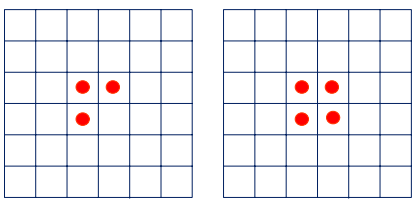

Figure 2. Three live cells in an inverse L-shape and four live cells as block [32].

B. Game of Life

The fascination of Conways Game is that, the simple rules leads to an unbelievable variety of mathematical problems, puzzles and patterns, yet at the same time it appears to exemplify emergent and self-organized behavior. For example let, the initial population consists of only one or two live cells, then it loses validity only one step next. On the other hand, if the initial population consists of three live cells then, because of rotational and reflexive symmetries, there are only two different possibilities which are shown in left of Fig. 2 [32]. Referring to rules of Game of Life as discussed in Sec. 1, all three cells have two live neighbors, so they survive. The dead cell that they all touch has three live neighbors, so it springs to life. None of the other dead cells have enough live neighbors to come to life. Thus, after one step, the population becomes as shown in the right grid Fig. 2. As discussed in [32], another three-cell initial population is I-shaped. The two possible orientations are shown in two steps of Fig. 3(a). At each step, two end cells die, the middle cell stays alive, and two new cells are born to give the orientation shown in the next step, thus generally known as blinker. One possible four-cell initial population is the block which is stationary because each of the live cells has three live neighbors and so lives on. The most important five-cell initial population, known as the glider, is shown in Fig. 3(b). At each step two cells die and two new ones are born. More discussion on it is beyond the scope of this paper.

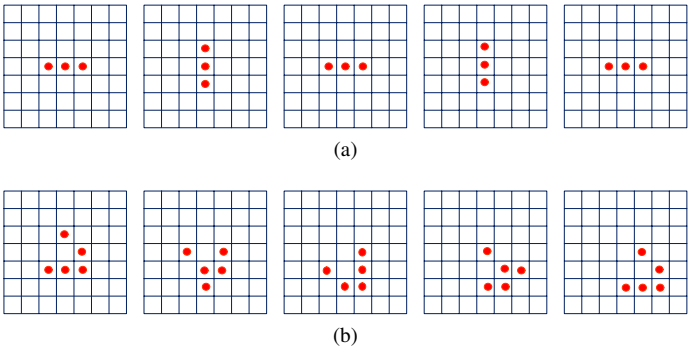

Figure 3. (a) Three live cells as a blinker. (b) Five live cells as a glider [32].

An excellent Java based simulation on Game of Life is presented in [33]. The performance analysis of the Game of Life based on parallel algorithm is shown in [34]. In [35], the combinations of Conway's new solitaire is presented. Open-MP implementation of Game is shown in [22]. Here, a 2-D $N \times N$ array is initialized 1 and 0 (for live and dead cell respectively) and memory requirement remains constant for entire computation. A different automaton of the game is presented in [36] with a $M \times N$ matrix, where each cell indicates the living status. In order to store the calculated next round state, it used second matrix. By the end of each round a swap takes place between the two pointers of the matrixes to keep the next round within current state.

Algorithm 1: Proposed conventional read procedure **read-grid(FILE* infile, twoD-array-t *grid).**

Input : A pointer to input file and to the structure.
Output: One or zero.

```
begin
    for i ← 0 to n − 1 do
        for j ← 0 to n − 1 do
            if (file.size ≠ n || build array = NULL ||
            A[i][j] ≠ 0 || A[i][j] ≠ 1) then
                return 0
            else
                A [i][j]→ temp.
                return 1
```

III. Proposed Solution to the Game of Life

This section proposes Open-MP and sparse matrix based implementation for the Game problem. Processor used for the simulations is intel core i5 3.10 GHz. The physical memory of the machine is 4 GB in addition with 3.14 GB page file space. The operating system is Ubuntu 12.04 with upgraded linux kernel. The conventional procedure that implements the game problem with respect to Open-MP is shown in Algorithm 2. Algorithm 2 assumes that the entire universe has already been read through the read grid procedure, which is shown in Algorithm 1. Here it has also been assumed that an array structure named as ***build array*** is available that either returns pointer to the structure or NULL if unable to allocate space for the elements. The read procedure reads initial configuration from the input file, returns 1(one) if all is well, otherwise prints error message and returns a 0(zero) value.

Lemma 1: Let, a $n \times n$ two dimensional array works as universe to the game with at most l live cells and d dead cells, where n, l, d are positive intergers. Let, $P(A_{al})$ and $P(A_{de})$ be the probability of being alive or dead for an arbitrary cell A. Then,

$$l, d \leq n$$
$$l + d \leq n^2$$
$$P(A_{al}) + P(A_{de}) = 1$$

□

As shown in [37], the universe is infinite for the game but the number of alive population is finite and usually very small compare to the entire universe. Thus, we can store the alive population in a finite matrix based on list, and increase the size if necessary. The procedure is known as sparse matrix representation. It leads to several advantages. For example, the Fig. 3 is represented by 7-by-7 matrix. The conventional full matrix storage (Algorithm 2) would require 7 × 7 = 49 elements. However, the sparse matrix storage requires 5+7=12 elements *i.e.*, for general l+n elements not n^2 elements.

Algorithm 2: Proposed conventional algorithm for the game problem with respect to Open-MP

Input : An inter input n, an $n \times n$ grid A $[n][n]$ and number of generation to compute G.
Output: $A[n][n]$ with corresponding cells values.

```
begin
    0 = Dead,  1 = Alive
    if (read-grid(file-name, n) → 0 then
        Print Error: Unable to Read
    else
        for i ← 1 to G do
            #pragma omp parallel for private(k, nbr)
            for j ← 0 to n − 1 do
                for k ← 0 to n − 1 do
                    nbr = A[i-1, j+1]+ ... +A[i+1, j-1]
                    if A [j][k] = 1 then
                        if (nbr = 2) || (nbr = 3) then
                            A [j][k] = 1
                        else
                            A [j][k] = 0
                    else
                        if ((nbr = 3) then
                            A [j][k] = 1
                        else
                            A [j][k] = 0
        return Resulted universe A[0][0] to A[n−1][n−1]
```

This memory minimization adds several advantages as more gliders are created *i.e.*, when the population expands and also for large n. It also add more advantages to the primary strength in shared data of parallel computing. The sparse matrix representation for Game of life has been addressed with Matlab, but to the best of our knowledge, it is yet to be implemented in the literature in addition with any parallel construct. In the remainder of the section, we provide a more implementation oriented modification of the presented algorithms and detail of our experiences in implementation and evaluating the performance monitoring on it. Algorithm 3 presents proposed modification to the Algorithm 2. Most of the modification is perform with respect to the list and computation of nbr. Here, rather than reserving memory for the entire universe, memory is dynamically allocated.

Lemma 2: Let, N be the set of non-negative integer and $n \times n$ grid work as the universe to the game with at most l live cells and d dead cells, where $n, l, d \, \epsilon \, N$. Then, sparse matrix representation to the game problem requires at most three vectors, one with length l, one with length n and an additional integer. It will require $n + l$ unit of memory initially but the conventional required n^2 unit of memory and $l + n \ll n^2$.

□

Algorithm 3: Proposed modified algorithm for the game problem with respect to Open-MP

Input : Integer inputs n, l and G.
Initialized list with $n \times l$ nodes
Output: A list $n \times m$ with corresponding cells values.

```
begin
    if (modified read-grid(file-name, n) → 0 then
        Print Error: Unable to Read
    else
        for i ← 1 to G do
            #pragma omp parallel for private(k, nbr)
            for j ← 0 to n − 1 do
                for k ← 0 to l − 1 do
                    compute nbr from the nodes of the
                    list.
                    if (Any nbr pre→0, cur→1) then
                        call malloc() function and set to 1.
                    else
                        if (Any nbr pre→1, cur→0) then
                            call free() function.
                    else
                        Set node value as per rule.
        return Print resulted list of universe
```

The modified algorithm (Algorithm 3) requires an extra integer input corresponding to the number of alive cell(s). It is needed to initialize the memory allocation for the universe. The working procedure of the algorithm is as follows: Let, the initial population is surrounded by the dead cells. Considering the dead cells as border cells, we are able to provide a viewing window [37]. If the population expands during the computations and cells travel beyond this viewing window, they continue to live and participate in the computation because of the dynamic behavior of the memory allocation. In Algorithm 3, all the live calls contribute to the computation through the modified read-grid procedure. Modified read-grid procedure is identical to the read-grid procedure presented in Algorithm 1 with the exception that the inner loop runs constant O(1) times rather than linear O(n) times here. We also find that, when all the initial population are alive, worse case complexity of Algorithm 3 is identical to the Algorithm 2 . However, the complexity decreases as i increases towards G as many cells died because of congestion.

Lemma 3: Let, n^2 be the total number of population with at most l live cells and the remaining are dead cells, where n and l are positive integers. Also let G is the number of generations to simulate. Then, average complexity of Algorithm 3 and 2 are $O(nlG)$ and $O(n^2G)$, respectively.

□

IV. Enhancement of the Proposed Scheme

This section briefly illustrates the enhancement of the proposed ideas to solve the biological problems and complex combinatorial problems.

A. Biological Enhancements

There are historical connections among the pattern recognition, the Game of Life and sequence alignment. Analysis of biological sequence can briefly be defined as the similar sequence discovery in the primary structure of related proteins or genes. The similarity generally fits during evolution. In addition, alignment of gene sequences and searching biological databases are the two most common methodologies in the field of bio-informatics [38]. There are two main types of biological sequence alignments *i.e.*, pair-wise sequence alignment and multiple sequence alignment. The first one compares two sequences at a time, whereas the second one compares many sequences in step [39]. In pair-wise sequence alignment, the proposed ideas can be enhanced with respect to dead and alive cells comparison[2] possibly in two consecutive generations. The snapshot of the entire universe can work as an input in such case. The similar sequence discovery in proteins or genes can also be handled by this snapshot. For example, identical snapshots or dissimilarity between alive cell in snapshots is less than a pre-specified verge. Besides, Comparing the new sequences with known one and recognition of changes in the gene pattern with respect to generations will facilitate understanding of the source biota of an organism. Generally the matches should be ordered to show the most closely related sequences first followed by sequences with diminishing similarity *i.e.*, a ranking mechanism based on alive cells and their similarity. These matches can usually be reported with a measure of statistical significance.

Moreover, the repeated pattern searching in DNA, RNA or peptide sequences are bit pertinent in the field of physiology [40]. Thus, researchers try to develop automatic sensitive methods to identify these repeated pattern. The proposed method can be used to find periodic patterns in biological sequences based on evolutionary distance and phase-shifts corresponding to alive (insertions) and dead (deletions) cell's distance and shifting (changes). For example, a given sequence is aligned to itself in a certain sense defining how many dead/alive cells in between and is trying to minimize a distance to periodicity. Relationships between different such periodicity measures can also be obtained with the proposed idea. In that case, we need to calculate the changes among two or more different snapshots. Interestingly, the proposed methods are iterative, and the running time is nearly proportional to the alive sequence length. Thus, the alignment will produce a periodic consensus pattern. Number of alive cells can work as a phase score, which can be used to indicate a statistical significance as well, and can further be enhanced to explore interesting pattern in gene sequences *i.e.* the universe.

[2]With respect to no. of cells alive and their corresponding positions

The proposed idea can be exploited to identify the anarchical blood cell division. Generally when a cell divides, it is suppose to produce two identical replicas of itself, but a cell is triggered to divide differently in case of anarchical division (cancerous cell). More specifically, the replicas are different than the parent cell(s) and other cells of the same type. When those cells divide they may further mutate. At some point, the cells that started of as a certain type and no longer resemble the type of cell they are suppose to be, then the differences among them determines whether they are pre-cancerous or cancerous [16], [41]. As discussed earlier, the proposed ideas can be enhanced to identify these difference between cell division among generation. For example, the snapshots of the universe for any two consecutive value of G can be used. Then, the difference is to be measured, and if it is more then a pre-specific threshold, it can be mark as an anarchical division.

B. Enhancement in the Complex Combinatorial Problem

Now a days, monitoring the operation of network communication has become indispensable. In a large network system, monitoring activity generally depends on analyzing the system logs to detect anomalies. The proposed ideas can be introduced to the iterative network mapping with respect to the snapshot discussed earlier. It will results a deeper insight into the network flow behavior by means of universe's similarity over a period. Intriguing the clustering with proposed methods can be used to identify relevant patterns in network traffic flows and to reduce traffic reports as well, for example with respect to density and threshold mechanism as discussed earlier.

Proposed ideas can also be enhanced to identify the flow of ground water and management of power stream. Generally, underground aquifer gets recharged by rain water, canal seepage and irrigation. If the **pf** rate abstraction of ground water is higher than the recharge rate, the water level of the aquifer recedes. The technology related to various areas of ground water management namely exploration, assessment, abstraction, recharge, pollution and salinity control has advanced in the past decade [42]. The proposed idea can be enhanced for ground water development with respect to potential aquifer locating. For example, particular sequence of cells in the universe with respect to our proposed algorithm, if there are more alive cells in the universe or a part of the universe, which are previously dead entails that the aquifer gets recharged. On the other hand, cell previous alive, if become dead will entails the the shortage of aquifer or there are recharging mechanism is some how defective. Moreover, the proposed ideas can also be used to manage the power streams. Considering time as generation will generate data arriving rapidly and continuously. In such case the date should consists of power consumption up to every second in advance in terms of present or absent (dead or alive). All this enhancement will inheritably gets the advantage of parallel computation and memory reduction of the proposed method. The implementing discussion of these ideas are beyond the scope of these paper and are considered for the future work.

V. Performance of the Proposed Method

Three variables namely n, l and G set the structures of overall computational process in the proposed method. As G always varies, overall comparative results are shown in contrast with three possible design approaches: (i) Varying l while keeping n constant, (ii) Varying n while keeping l constant, and (iii) Varying both n and l.

A. Memory Comparison

Figs. 4 and 5 present the performance of the proposed method in terms of memory required. Form these figures we find that the proposed scheme requires much lesser memory than the existing designs [33,39] for all possible design approaches (sub figures in these two figures can respectively be used for the performance evaluation of case iii). From these figures we also find that the performance of the proposed design is much better for the larger data set *i.e.*, performance of the proposed design improves when n, l, and G increase.

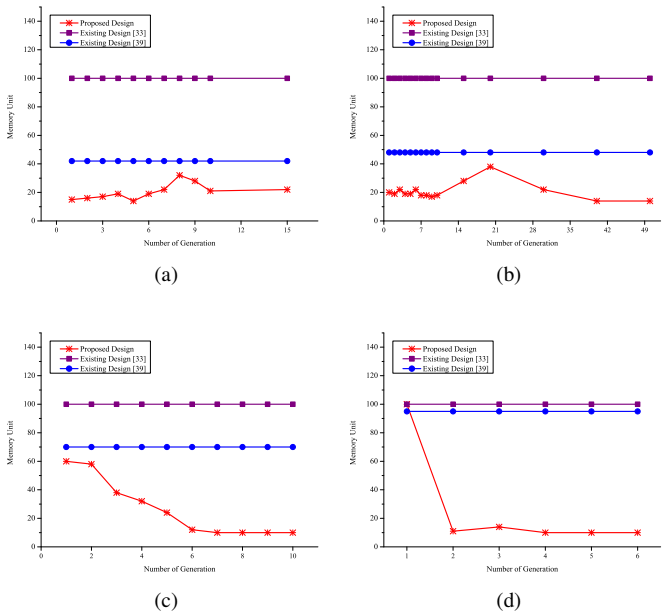

Figure 4. Performance of the proposed work and existing works with respect to the memory uses while varying l and keeping the constant n (a) n=10, l=5 (b)n=10, l=10 (c)n=10, l=50 (d) n=10, l=90.

B. Time Comparison

Fig. 6 and Fig. 7 present the performance of the proposed simulation with respect to the time consumption for case i and case ii, respectively (sub figures in these two figures can respectively be used for the performance evaluation of case iii). From these figures we find that initially proposed and the existing methods [33, 39] perform equally well. However, performance of the proposed method improves as G Grows for all possible design approaches.

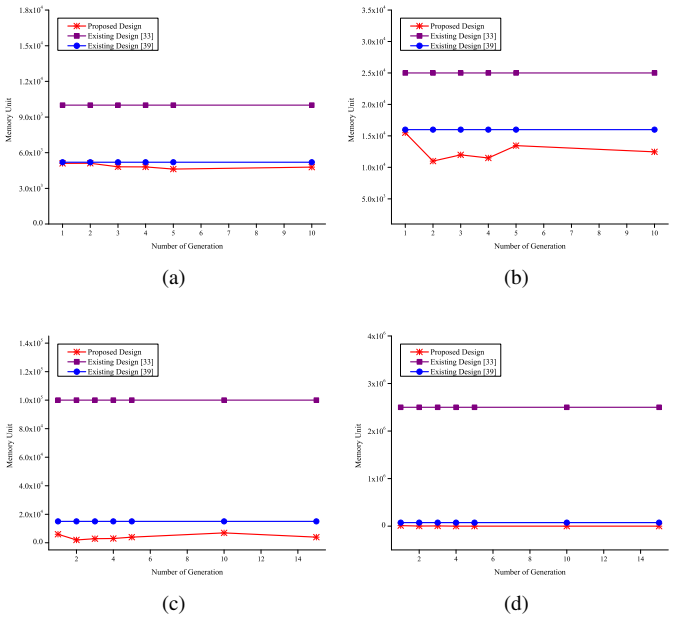

Figure 5. Performance of the proposed work and existing works with respect to the memory uses while varying n and keeping the constant l (a) n=100, l=5000 (b)n=500, l=5000 (c) n=1000, l=5000 (d) n=5000, l=5000.

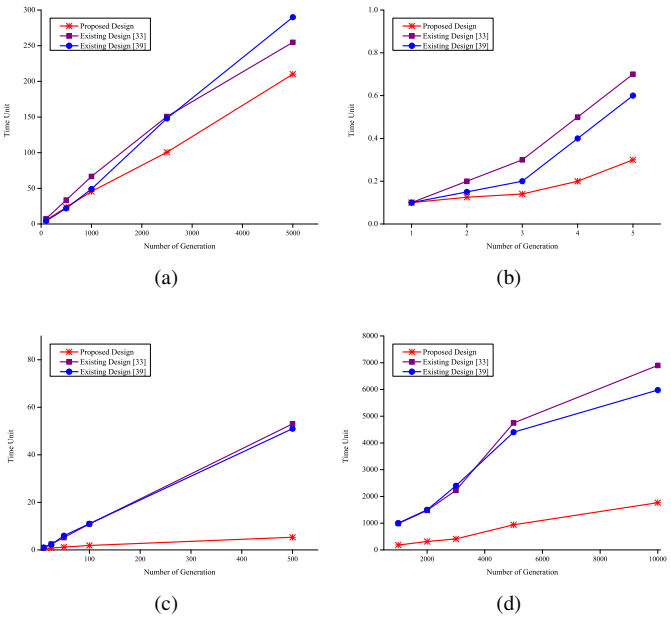

Figure 7. Performance of the proposed work and existing works with respect to the computational time required while varying n and keeping the constant l (a) n=100, l=5000 (b)n=500, l=5000 (c) n=1000, l=5000 (d) n=5000, l=5000.

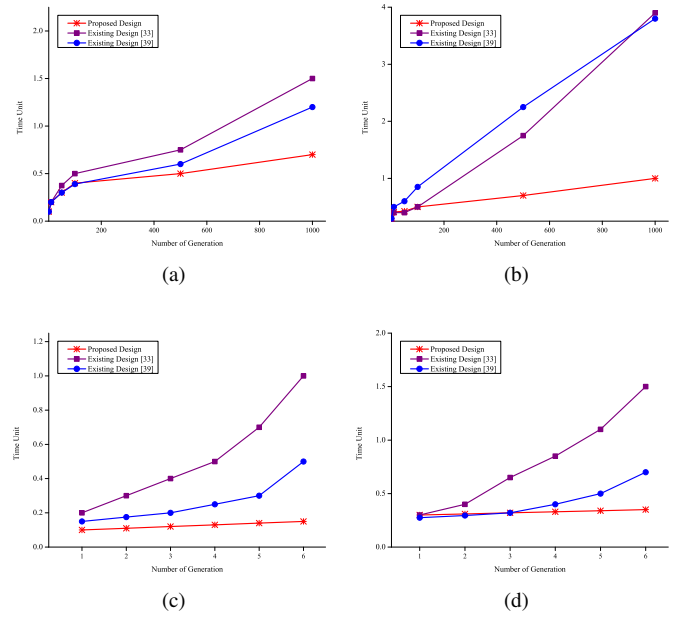

Figure 6. Performance of the proposed work and existing works with respect to the computational time required while varying l and keeping the constant n (a) n=10, l=10 (b)n=10, l=25 (c) n=10, l=50 (d) n=10, l=90.

From the Figs. 3-to-6 we find the following[3]:

- The computation time is more dependent on G rather than l for the proposed design but existing designs [33, 39] are more dependent to n and G rather than l.
- Independent on G, for the initial $l \simeq n^2/2$, there are less than 13.64% changes in the number of live cells and their positions after G/l generation of computation and 79% cases the universe remain stationary[4].
- The initial $l < n^2/2$ returns less than $(l+n)$ live cells after n/l generations of computation. On the other hand, the initial $l > n^2/2$ returns less than $|l - n/2|$ live cells after n/l^2 generation of computation.
- If $n > l^2$, the overall time complexity for all the designs is almost equal until G becomes $\lceil l^{2n/3} \rceil$ and then performance of the existing design is much worse than the proposed one. All the other cases, proposed design performs either equally or better than the existing designs.

VI. Conclusion

In this paper, we have proposed an efficient parallel computational method (Open-MP) based solution for the Game of Life problem. The inherent flexibility of the Open-MP such as higher thread level abstraction and breakdown of serial parallel regions make it comfortable for the proposed solution to fit in the field of bio-informatics specially when dynamic allocation and deallocation techniques are used. Here, a finite domain constraints has been used to represent an infinite domain associate problem without any changes in its behavior. This is key to the many biological and combinatorial problems, because these problems generally exhibit a search space that have the potentially to speed-up through parallelization.

[3]All the graphs are drawn using the normalized data rather than actual one. The normalization is performed in terms of corresponding smallest unit.

[4]with at least one and not more that two dead cell in between each alive cell for the initial configuration and $G > l$

Several other enhancements of the proposed method in addition with the applications are detailed in Sec IV. In Sec III, we presents the generalized algorithms to solve the Game problem in addition with the several theoretical explanations and examples. As shown here, the proposed solution frees the programmer from the burden of initialization and tracking of all the variables during computation. This adds several advantages, such as linear speed-up trend both in terms of time and memory as reflected in the performance study section (Sec. V).

REFERENCES

[1] A. Adamatzky, *Game of Life Cellular Automata*, 1st ed. Springer Publishing Company, Incorporated, 2010.

[2] P. Callahan, "What is the game of life?" Last visited, Jun 1 2014. [Online]. Available: Availableat,http://www.math.com/students/wonders/life/life.html

[3] A. H. Dekker, "The game of life: a clean programming tutorial and case study," *SIGPLAN Not.*, vol. 29, no. 9, pp. 91–114, 1994.

[4] M. R. Wick, "Using the game of life to introduce freshman students to the power and elegance of design patterns," in *Companion to the 19th annual ACM SIGPLAN conference on Object-oriented programming systems, languages, and applications*. ACM, 2004, pp. 103–105.

[5] J. A. Gómez-Pulido, J. M. Matas-Santiago, F. Pérez-Rodríguez, M. A. Vega-Rodríguez, J. M. Sánchez-Pérez, and F. F. De Vega, "Hardware modelling of cellular automata: the game of life case," ser. EUROCAST'07. Springer-Verlag, 2007, pp. 589–595.

[6] M. Satomi and C. Sommerer, "game_of_life: interactive art installation using eye-tracking interface," in *Proceedings of the international conference on Advances in computer entertainment technology*, ser. ACE '07. ACM, 2007, pp. 246–247.

[7] N. M. Gotts, "Ramifying feedback networks, cross-scale interactions, and emergent quasi individuals in conway's game of life," *Artif. Life*, vol. 15, no. 3, pp. 351–375, Jul. 2009.

[8] G. E. Pazienza, E. Gomez-Ramirez, and X. Vilasís-Cardona, "Polynomial cellular neural networks for implementing the game of life," in *Proceedings of the 17th international conference on Artificial neural networks*. Springer-Verlag, 2007, pp. 914–923.

[9] C. A. Reiter, "The game of life on a hyperbolic domain," *Comput. Graph.*, vol. 21, no. 5, pp. 673–683, Sep. 1997.

[10] K. Sigmund, *Games of Life: Explorations in Ecology, Evolution and Behaviour*. Oxford University Press, Inc., 1993.

[11] B. Chapman, G. Jost, and R. v. d. Pas, *Using OpenMP: Portable Shared Memory Parallel Programming (Scientific and Engineering Computation)*. The MIT Press, 2007.

[12] B. Barney, "Open-mp," Available at, https://computing.llnl.gov/tutorials/openMP/#Exercise1, Last visited, Dec 1 2015.

[13] H. Shen, Y. Tan, J. Lu, Q. Wu, and Q. Qiu, "Achieving autonomous power management using reinforcement learning," *ACM Trans. Des. Autom. Electron. Syst.*, vol. 18, no. 2, pp. 24:1–24:32, Apr. 2013.

[14] I.-H. Hou and P. R. Kumar, "Broadcasting delay-constrained traffic over unreliable wireless links with network coding," ser. MobiHoc '11. ACM, 2011, pp. 4:1–4:10.

[15] I. V. Schevtschenko, "A parallel adi method for a nonlinear equation describing gravitational flow of ground water," ser. ICCS '01. Springer-Verlag, 2001, pp. 904–910.

[16] S. Kawano, T. Shimamura, A. Niida, S. Imoto, R. Yamaguchi, M. Nagasaki, R. Yoshida, C. Print, and S. Miyano, "Identifying gene pathways associated with cancer characteristics via sparse statistical methods," *IEEE/ACM Trans. Comput. Biol. Bioinformatics*, vol. 9, no. 4, pp. 966–972, Jul. 2012.

[17] Y. Meng, O.-K. Ha, and Y.-K. Jun, "Dynamic instrumentation for nested fork-join parallelism in openmp programs," in *Proceedings of the 4th international conference on Future Generation Information Technology*, ser. FGIT'12. Berlin, Heidelberg: Springer-Verlag, 2012, pp. 154–158. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-35585-1_21

[18] M. Voss, E. Chiu, P. M. Y. Chow, C. Wong, and K. Yuen, "An evaluation of auto-scoping in openmp," in *Proceedings of the 5th International Conference on OpenMP Applications and Tools: Shared Memory Parallel Programming with OpenMP*, ser. WOMPAT'04. Berlin, Heidelberg: Springer-Verlag, 2005, pp. 98–109.

[19] L. Dagum and R. Menon, "Openmp: An industry-standard api for shared-memory programming," *IEEE Comput. Sci. Eng.*, vol. 5, no. 1, pp. 46–55, Jan. 1998. [Online]. Available: http://dx.doi.org/10.1109/99.660313

[20] V. Bui, "Perfomp: A runtime performance/event monitoring interface for openmp," Master's thesis, Faculty of the Department of Computer Science, University of Houston, May 2007.

[21] C. Liao, O. Hernandez, B. Chapman, W. Chen, and W. Zheng, "Openuh: an optimizing, portable openmp compiler: Research articles," *Concurr. Comput. : Pract. Exper.*, vol. 19, no. 18, pp. 2317–2332, Dec. 2007. [Online]. Available: http://dx.doi.org/10.1002/cpe.v19:18

[22] D. Joiner, "Game of life with openmp," http://www.shodor.org/ petascale/materials/sharedMemory/code/Life_omp/, Last visited, Dec 5 2015.

[23] Open-MP, "The open-mp api specification for parallel programming," Available at, http://openmp.org/wp/, Last visited, Jun 8 2015.

[24] S. Pannala, E. D'Azevedo, and M. Syamlal, "Hybrid (openmp and mpi) parallelization of mfix: A multiphase cfd code for modeling fluidized beds," in *Proceedings of the 2003 ACM Symposium on Applied Computing*, ser. SAC '03. ACM, 2003, pp. 199–206.

[25] X. Teruel, "The barcelona openmp task suite (bots) project," Available at, https://pm.bsc.es/projects/bots, Last visited, Dec 1 2013.

[26] B. R. de Supinski, "Llnl openmp performance suite description," Available at, http://computation.llnl.gov/casc/RTS_Report/openmp_perf.html, Last visited, Dec 1 2015.

[27] M. S. Mohsen, R. Abdullah, and Y. M. Teo, "A survey on performance tools for openmp," pp. 754–765, 2009.

[28] H. Jin, B. Chapman, and L. Huang, "Performance evaluation of a multi-zone application in different openmp approaches," in *Proceedings of the 3rd International Workshop on OpenMP: A Practical Programming Model for the Multi-Core Era*, ser. IWOMP '07. Springer-Verlag, 2008, pp. 25–36.

[29] R. Aversa, B. Di Martino, N. Mazzocca, and S. Venticinque, "Performance analysis of hybrid openmp/mpi n-body application," in *Proceedings of the 5th International Conference on OpenMP Applications and Tools: Shared Memory Parallel Programming with OpenMP*, ser. WOMPAT'04. Berlin, Heidelberg: Springer-Verlag, 2005, pp. 12–18.

[30] R. Chandra, L. Dagum, D. Kohr, D. Maydan, J. McDonald, and R. Menon, *Parallel Programming in OpenMP*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2001.

[31] Z. Liu, L. Huang, B. Chapman, and T.-H. Weng, "Efficient implementation of openmp for clusters with implicit data distribution," in *Proceedings of the 5th International Conference on OpenMP Applications and Tools: Shared Memory Parallel Programming with OpenMP*, ser. WOMPAT'04. Berlin, Heidelberg: Springer-Verlag, 2005, pp. 121–136.

[32] C. B. Moler, *Experiments with MATLAB*. MathWorks, Inc, 2011.

[33] E. Martin, "Game of life: Simulation/java source code," http://www.bitstorm.org/gameoflife/, Last visited, Jun 1 2015.

[34] L. Ma, X. Chen, and Z. Meng, "A performance analysis of the game of life based on parallel algorithm," *CoRR*, vol. abs/1209.4408, 2012.

[35] M. Gardner, "The fantastic combinations of John Conway's new solitaire game "life"," *Scientific American*, vol. 223, pp. 120–123, Oct. 1970.

[36] A. D. Bechtsoudis, "Game of life c parallel implementation," https://bechtsoudis.com/work-stuff/projects/game-of-life-c-parallel-implementation-pthreads-openmp/, Last visited, Dec 5, 2015.

[37] A. Weeden, "Parallelization: Conway's game of life," Available at, http://www.shodor.org/petascale/materials/UPModules/GameOfLife/, Last visited, Jun 1 2013.

[38] K. Laurio, F. Linåker, and A. Narayanan, "Regular biosequence pattern matching with cellular automata," *Inf. Sci. Appl.*, vol. 146, no. 1-4, pp. 89–101, Oct. 2002.

[39] G. S. Sadasivam and G. Baktavatchalam, "A novel approach to multiple sequence alignment using hadoop data grids," in *Proceedings of the 2010 Workshop on Massive Data Analytics on the Cloud*, ser. MDAC '10. ACM, 2010, pp. 2:1–2:7.

[40] Z. Zhu, J. Zhou, Z. Ji, and Y.-H. Shi, "Dna sequence compression using adaptive particle swarm optimization-based memetic algorithm," *Trans. Evol. Comp*, vol. 15, no. 5, pp. 643–658, Oct. 2011.

[41] I. Park, K. H. Lee, and D. Lee, "Mining cancer genes with running-sum statistics," in *Proceedings of the third international workshop on Data and text mining in bioinformatics*. ACM, 2009, pp. 35–42.

[42] "New technologies for ground water," Available at, http://tifac.org.in, Last visited, Jun 1 2015.