Higher Dimensional Games of Life

Research · August 2015

DOI: 10.13140/RG.2.1.4679.3440

6 authors, including:

Deepak Karunakaran
Environmental Science & Research
11 PUBLICATIONS **30** CITATIONS
SEE PROFILE

Shailesh Kumar Jagadeesan
International Institute of Information Technology Bangalore
3 PUBLICATIONS **1** CITATION
SEE PROFILE

Shrisha Rao
International Institute of Information Technology Bangalore
132 PUBLICATIONS **990** CITATIONS
SEE PROFILE

Higher Dimensional Games of Life

Abhishek Srivastava, Deepak Karunakaran, Shailesh Kumar,
Vikram B Narayana, Veeresh Halakatti, and Shrisha Rao*

International Institute of Information Technology - Bangalore,
Bangalore 560 100, India
http://www.iiitb.ac.in

Abstract. The basic configurations of the Game of Life, viz., gliders, oscillators and invariants, are presented in higher dimensions. Visualization of these configurations in higher dimensions is challenging, and we present two novel and simple techniques, the simple projection technique and the pulse of life technique for the visual representation of the configurations. The former is specifically suited to the 4-dimensional Game of Life and the latter applicable to all dimensions. These methods also verify the characteristics of the configurations. We also present a heuristic based brute-force (PR-CF) algorithm which is used to discover these configurations by pruning the rule space to obtain potential candidate rules. We present a glider in the 3 dimensional GOL, and oscillators and invariants in 3, 4 and 5 dimensional GOLs, the details of which are also presented here.

Key words: cellular automata, game of life, gliders, oscillators, invariant, higher dimensional visualization

1 Introduction

The Game of Life (GOL) [1] has yielded very interesting and complex patterns since its inception in 1960s by John H. Conway. The GOL, which initially started off with two dimensions, has been extended to higher dimensions. A significant amount of work has been done in the three-dimensional GOL [2], [3], [4], [5]. Some of the basic GOL configurations include gliders, oscillators, invariants, etc. [1].

In this paper, we explore the existence of these basic configurations in higher dimensions. We present the configurations of invariants and oscillators in $3^{rd}, 4^{th}$ and 5^{th} dimensions, and of a glider in 3^{rd} dimension. Higher dimensions pose two challenges:

- Computational complexity
- Visualization

* Corresponding Author: shrao@ieee.org

A basic configuration is represented by an "initial state" and a "rule." An initial state is a set of cells which are set alive before the GOL begins. A *seed* is an initial state which gives rise to the desired configuration on application of a *rule*. Examining all combinations of rules and candidate seeds for discovering a configuration is a computationally exhaustive task, as the total number of such combinations increases exponentially (2^N, where N is the number of cells) with the increase in the number of cells. We consider the reflections and rotations of a particular combination as a different candidate seed.

For example, in the two-dimensional GOL, determining the seed for a glider in a 3×3 grid by brute force would require testing for the glider on 2^9 possible combinations. One such test that has been formulated is the Bays' Test [2]. Numerous gliders have been discovered so far in 2D and a considerable amount of work has been done in 3D as well [2], [3], [4], [5]. A similar attempt to discover the glider in four-dimensional space with just $3 \times 3 \times 3 \times 3$ grid would require conducting a test on 2^{81} combinations which poses an impossible computational challenge. This in combination with the huge rule space of the GOL makes the computation even further complex. In order to tackle this complexity, we developed a heuristic based brute-force algorithm called *Pruned Rules based Configuration Finder* (PR-CF) algorithm. This algorithm prunes the rule space so that only a subset of rules that may potentially yield a desired configuration are considered. Specifically, we restrict the rule space to only those rules which maintain constant number of live cells in different generations of the GOL.

The GOL is visualized in a rectangular array of cells in 2D. In 3D, a cubical arrangement of cells gives a good representation. Hence visualization of the basic configurations for these dimensions is simple. But, as we proceed to higher dimensions, there is no apt method for visualizing and also for verifying the configurations. In addition to the computational challenge in determining the configurations exhibiting the desired patterns, visualization of these patterns in higher dimensions is a challenging task. Our work addresses this problem with two simple techniques to analyze the patterns for the multi-dimensional GOL. We call these techniques the *simple projection* technique and the *pulse of life* technique. Using the PR-CF algorithm we successfully discovered some configurations in dimensions 3, 4, and 5, and verified them using these visualization techniques.

Section 2 talks about the related work done in the area of multi-dimensional GOL. In Section 3, the two proposed visualization techniques are explained in detail. Section 4 discusses at length the algorithm for the discovery of the configurations and the heuristic used in the algorithm to tackle the computational complexity. Section 5 shows the various configurations discovered for different dimensions using the algorithm. The two visualization techniques described in Section 3 have been used in Section 5.

2 Related Work

The GOL has been extended and generalized to different domains of cells. One such extension involving a circular hyperbolic domain has been investigated and symmetry-breaking patterns have been observed [6]. Time evolution of the GOL has been visualized in 3D in which the color of the cell is based on the number of its neighbors [7].

Various projection techniques have been used to visualize higher dimensional polytopes [8]. Different projection techniques used to project 3D objects to 2D-plane [9] has been extended to project 4D polytopes onto 3D-hyperplane [10]. Our Simple Projection Technique for visualization is similar to the parallel projection technique mentioned in [10].

The two-dimensional GOL has been explored extensively and many configurations have been discovered. Evolutionary algorithms have been used for the discovery of patterns in 2D successfully. The search for Glider Guns [11] in [12] follows a genetic-algorithm-based approach to discover glider guns in the two-dimensional GOL. The highlight of the work in [12] is to develop a suitable fitness function to stabilize the mutating "primordial soup" to get the desired patterns [12].

A very interesting method of formulating Conway's Game of Life using Integer Programming has been used to discover interesting patterns [13]. Two such formulations for finding still-life (invariants) have been provided in [13].

3 Visualization Techniques

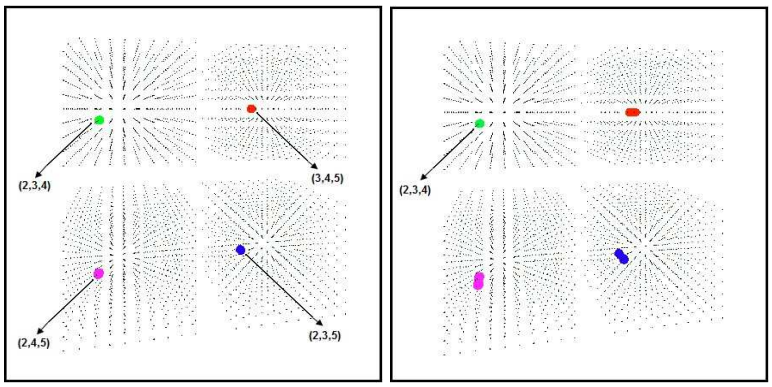
Here we present two visualization techniques which can be used to analyze the qualitative properties of configurations in dimensions two and greater.

3.1 Simple Projection Technique

The *simple projection* technique uses the idea of parallel (orthogonal) projection of higher dimensions on to 3D subspaces [10], [8], [14], [15], specifically that of 4D space to 3D subspaces. Consider a 4D coordinate say (2, 3, 4, 5), the corresponding projections on to 3D subspaces are (2, 3, 4), (3, 4, 5), (2, 4, 5) and (2 , 3 , 5). These four 3D subspaces are plotted and interpreted as shown in Figure 1(a).

Suppose we have another 4D coordinate (2, 3, 4, 6), then the corresponding orthogonally projected vectors in 3D subspaces will be (2, 3, 4), (3, 4, 6), (2, 4, 6), (2, 3, 6). The orthogonal projections of both these 4D coordinates coincide to the same coordinate (2, 3, 4) in one of the 3D subspaces. The coordinate (2, 3, 4) projected from both (2, 3, 4, 5) and (2, 3, 4, 6) is shown in the upper left 3D subspace in the Figure 1(b)). Since two 4D coordinates project to the same 3D coordinate, it is represented by a darker shade.

Therefore, as the number of coordinates that project to a particular point in the 3D subspace increases, the intensity of the color of the projected coordinate



(a) (b)

Fig. 1. Simple Projection Technique

in the 3D subspace also increases. Figure 2 shows a more prominent difference in the intensity of this point as there are four 4D coordinates projected onto it. The 4D coordinates are, viz., (2, 3, 4, 5), (2, 3, 4, 6), (2, 3, 4, 7), (2, 3, 4, 8) and (7, 7, 7, 7). The color intensity of this point (2, 3, 4) can be compared with that of the point (7, 7, 7) which has only one 4D coordinate projecting onto it.

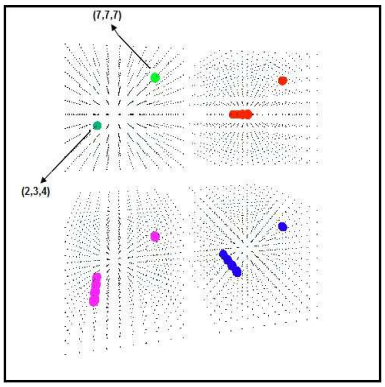


Fig. 2. Five points in 4D projected to 3D subspaces

A configuration in the four-dimensional GOL is represented as a collection of 4D coordinates. To visualize the configurations, these coordinates are projected onto the 3D subspaces. For every new generation of the GOL, as the cell states vary on application of the rule, different set of 4D coordinates are obtained. These new coordinates are projected again to the 3D subspaces. This process is repeated for every generation of the GOL.

The orthogonal projection technique is best suited for the four-dimensional GOL. For higher dimensions, the number of 3D subspaces increases and the configurations cannot be visually appreciated. In the next section, we present a simple visualization technique which can be used for any higher dimension.

3.2 Pulse of Life Technique

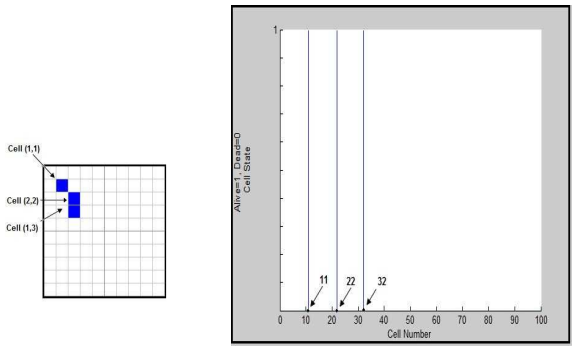
The *pulse of life* technique uses a 2D-graph based approach to observe the behavior of different configurations. In this graph, the X-axis represents cell number and the Y-axis represents the cell's state.

Consider a two-dimensional grid of size $N \times N$. Each cell is identified by a number which is calculated using the relation

$$number(i, j) = i \times N + j, \tag{1}$$

where i is the *row index* and j is the *column index*, with $0 \leq i, j \leq N - 1$.

For a two-dimensional grid of size 10×10 in Figure 3(a), the numbering of the cells is as shown in Figure 3(b). The numbering starts with 0 at the top left corner and moves in row-major order. The filled cells represent cells that are alive, and the others represent dead cells. We now map this onto a 2D-graph as shown in the Figure 3(b). The X-axis represents the cell numbers from 0 to 99. On the Y-axis, the state of a cell that is alive is shown with a pulse. The filled cells (1, 1), (2, 2) and (2, 3) in the two-dimensional grid which correspond to the cell numbers 11, 22 and 32 respectively in the graph, are represented by pulses.

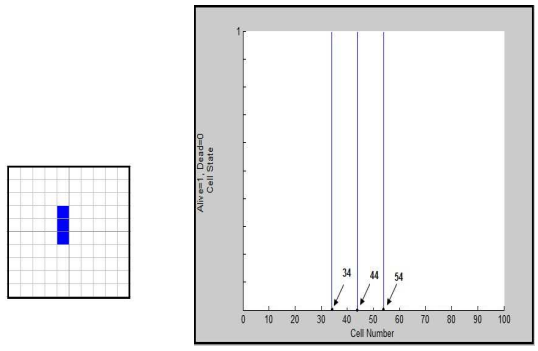


(a) (b)

Fig. 3. Cells in two-dimensional grid and its corresponding graphical representation

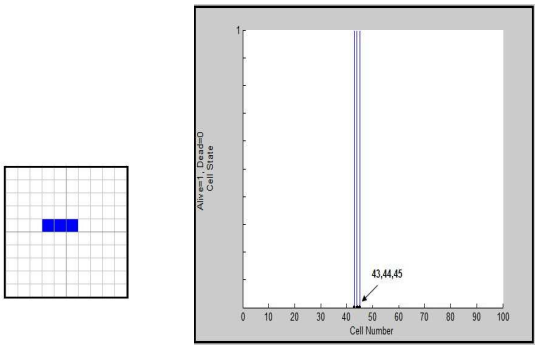
Figure 4(a) represents State A of a 2D oscillator. The mapping of this state onto the graph using the pulse of life technique is shown in Figure 4(b).

Figure 5(a) represents State B of the oscillator. The mapping of this state onto the graph is shown in Figure 5(b).



(a) (b)

Fig. 4. State A of an oscillator using graphical representation



(a) (b)

Fig. 5. State B of the oscillator using graphical representation

The same technique can be extended to higher dimensions. In this case, the number of a cell $(u_{d-1}, u_{d-2}, \ldots, u_0)$ is calculated as

$$N^{d-1} \times u_{d-1} + N^{d-2} \times u_{d-2} + \ldots + u_0, \tag{2}$$

where d represents the number of dimensions, N represents the size of the grid in each dimension, and $u_{d-1}, u_{d-2}, u_{d-3}, \ldots, u_0$ are the indices which represent a cell in d^{th} dimension.

Suppose we have a four-dimensional grid of size $5 \times 5 \times 5 \times 5$ (i.e., $N = 5$ and $d = 4$). If the coordinate of a point in it is (2, 3, 5, 3), then the corresponding cell number will be $5^{4-1} \times 2 + 5^{4-2} \times 3 + 5^{4-3} \times 5 + 5^{4-4} \times 3 = 250 + 75 + 25 + 3 = 353$, i.e., it is the 353^{rd} cell in the 2D-graph, out of a total of $5 \times 5 \times 5 \times 5 = 625$ cells.

4 The Pruned Rules based Configuration Finder (PR-CF) Algorithm

The PR-CF Algorithm uses a heuristic based brute-force approach for finding the basic configurations, viz., glider, oscillator and invariant. The idea behind the algorithm is to test for properties of the basic configurations on a particular combination of a rule and a candidate seed. The heuristic is devised such that it prunes the large rule space to obtain a list of possible candidate rules. These rules have the property that the number of live cells is maintained constant in the successive generations. The algorithm is generalized for any of the multi-dimensional GOL.

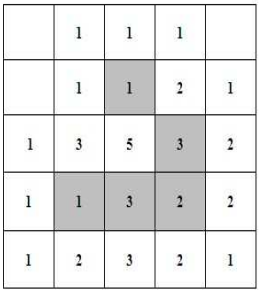
Notations used in the Algorithm

- The grid is a matrix in d-dimensions that is used to hold the candidate seed, and N represents the size of the matrix.
- The candidate seed is a set of elements in the grid which are set to 1.
- A rule is represented by a 3-tuple, $(newLife, overPopulation, underPopulation)$, interpreted as follows. If a dead cell has $newLife$ number of neighbors, then it comes to life in the next generation. If a living cell has less than $underPopulation$ number of neighbors or more than $overPopulation$ number of neighbors, then it dies in the next generation.
- $rulesList$ contains the list of rules obtained from the procedure $prepareRuleList$.
- k is used to represent the number of live neighbors for each cell which varies from 0 to $3^d - 1$.
- $cellCountLive$[k] holds the count of the live cells which have k live neighbors.
- $cellCountDead$[k] holds the count of the dead cells which have k live neighbors.

Algorithm 1 and Algorithm 2 together constitute the PR-CF algorithm. Algorithm 1 iterates through all candidate seeds for finding a configuration. The total number of candidate seeds for any grid is 2^N. In line 3 of Algorithm 1, the rules pruning heuristic (Algorithm 2) is invoked.

Algorithm 2 applies the heuristic to determine the set of rules for a given candidate seed. In lines 1–8, for each cell in the grid except the non-neighbors of the candidate seed, the count of the live neighbors is calculated. An example is shown in Figure 6. The columns in the table represent the values of k, $cellCountDead$[k] and $cellCountLive$[k] for the candidate seed shown in the grid adjacent to it. The illustrations in the Figure 6 and the corresponding notations are similar to the signature used in [3].

$cellCountDead[k]$ represents the number of cells that come alive in the next generation. The lines 20–21 check if $cellCountDead[k]$ can be neutralized by the value of $(sum1+sum2)$. Now, if $(sum1+sum2)$ becomes equal to $cellCountDead[k]$, then this could be a candidate rule with $(newLife = k, overPopulation = k-1, underPopulation = j)$.



	1	1	1	
	1	1	2	1
1	3	5	3	2
1	1	3	2	2
1	2	3	2	1

No of live neighbors	No of dead cells	No of live cells
1	9	2
2	5	1
3	2	2
5	1	0

Fig. 6. A candidate seed and the number of live neighbors for the cells

For value of $k = 2$ and $cellCountDead[2] = 5$, $(sum1 + sum2)$ becomes equal to 5 for values of $j = 5$ and hence the rule (2 , 1 , 5) is obtained. This is interpreted as follows. Since $newLife = 2$, five cells become alive in the next generation, and this is neutralized by the values of

- $overPopulation = 1$, due to which 3 cells die in the next generation.
- $underPopulation = 5$, due to which 2 cells die in the next generation. i.e., 5 - 3 - 2 = 0, which keeps the same live cell count as that of the candidate seed.

Input: N, d
Output: Rule and Seed of a Configuration

```
1  while ( there exists a candidate seed) do
2      initialize grid with candidate seed;
3      prepareRuleList(candidate seed);
4      while (rulesList not empty) do
5          initialize grid with candidate seed;
6          for ( i = 1 to 10) do
7              apply rule to produce next state;
8              if (no of live cells in next state != no of live cells in previous
               state) then
9                  break;
10             else
11                 check for oscillator/glider/invariant;
12             end
13         end
14     end
15 end
```

Algorithm 1: Find Configuration

The rules list so obtained from Algorithm 2 is then used by Algorithm 1 to find out if there exists an oscillator or a glider or an invariant. We test for configurations which can have a maximum period of 10 (line 6). This can be extended to higher periods.

In Algorithm 1, a rule selected from the rules list is applied to the candidate seed and a test (line 11) is conducted to verify the configuration. This test is similar to Bays' test [2] and is described as follows. The check for glider involves a test to verify whether the candidate seed has translated one cell distance in any of the hyperplanes in the multi-dimensional grid in subsequent generations. The check for an oscillator involves a test to verify whether the candidate seed reappears for alternate generations. The check for invariant involves a test to verify whether the candidate seed remains unchanged for all generations.

Input: Candidate Seed
Output: Set of rules that maintains the number of live cells constant for atleast one generation

```
1  for (each cell in the grid except the non-neighbors of the candidate seed) do
2  |  x = calculate number of live neighbors;
3  |  if (cell is alive) then
4  |  |  cellCountLive[x]++;
5  |  end
6  |  cellCountDead[x]++;
7  end
8  for (k = 1 to (3^d − 1) ) do
9  |  if (cellCountDead[k] == 0) then
10 |  |  continue;
11 |  end
12 |  sum1 = 0;
13 |  for (j = (3^d − 1) to 0) do
14 |  |  sum1 = sum1 + cellCountLive[j];
15 |  |  if (sum1 > cellCountDead[k]) then
16 |  |  |  break;
17 |  |  end
18 |  |  sum2 = 0;
19 |  |  for (l = 1 to j and (sum1 + sum2) < cellCountDead[k]) do
20 |  |  |  sum2 = sum2 + cellCountLive[l];
21 |  |  end
22 |  |  if (sum1 + sum2 == cellCountDead[k]) then
23 |  |  |  overPopulation = k - 1;
24 |  |  |  underPopulation = j;
25 |  |  |  newLife = k;
26 |  |  |  update rulesList with newLife, overPopulation,
   |  |  |  underPopulation;
27 |  |  end
28 |  end
29 end
```

Algorithm 2: Prepare Rules List

5 Results

In this section, we present the basic higher-dimensional configurations, viz., invariants, oscillators and gliders, that we have discovered using the PR-CF algorithm. The algorithm was implemented on a Cell Broadband Engine (CBE) [16]. We leveraged the multi-core architecture of the CBE to speed up the process of discovering configurations. Two different variations of the algorithm were used. One implementation used pseudo-random number generators to generate candidate seeds for discovering configurations. The other used a sequential seed generator for the candidate seeds. The configurations so discovered are presented here for different dimensions using the two visualization techniques discussed. They are summarized in Table 1. The same convention ($NewLife$, $OverPopulation$, $UnderPopulation$) as mentioned in Section 4 is used to represent the rules for the configurations.

5.1 Basic 3D Configurations

As far as we know the following 3D configurations were yet undiscovered, and we were able to discover them using the PR-CF algorithm. Figure 7 shows an invariant in 3D with rule (3, 3, 2).

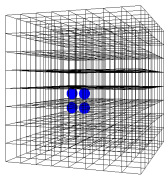
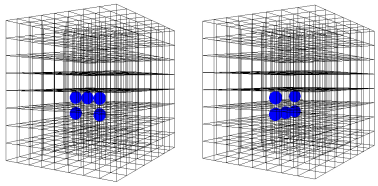


Fig. 7. 3D Invariant

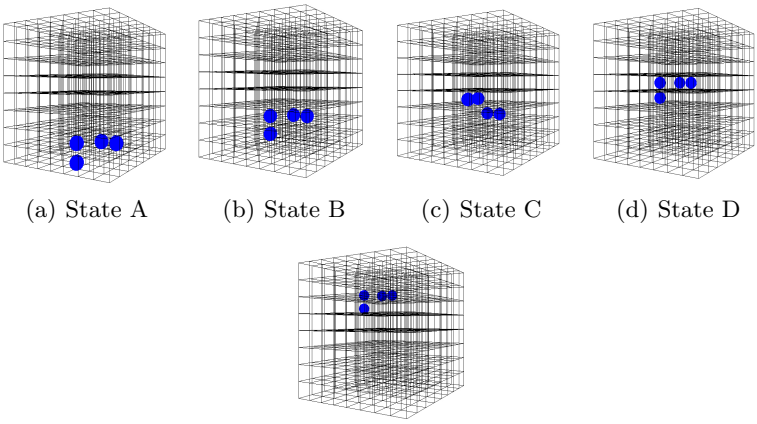
Figure 8 shows a period-2 oscillator in 3D with rule (5, 4, 5).



(a) State A (b) State B

Fig. 8. 3D Oscillator

Figure 9 shows a glider in 3D with rule (3, 0, 0).



(a) State A (b) State B (c) State C (d) State D

(e) State E

Fig. 9. 3D Glider

5.2 Basic 4D Configurations

Figure 10 shows an invariant in 4D with rule (3, 3, 2). Figure 11 shows a period-2 oscillator in 4D with rule (10, -1, -1).

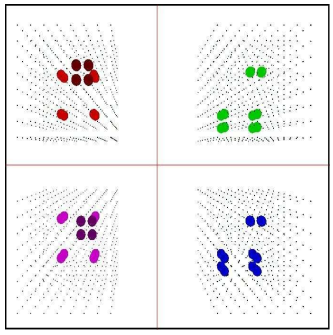
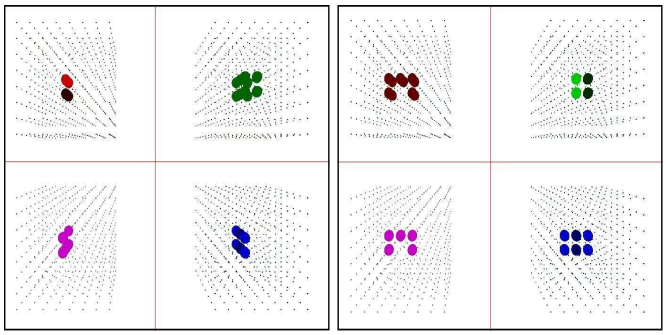


Fig. 10. 4D Invariant

5.3 Basic 5D Configurations

Figure 12 shows an invariant in 5D with rule (11, 11, -1). Figure 13 shows a period-2 oscillator in 5D with rule (17, -1, -1).



(a) State A (b) State B

Fig. 11. 4D Oscillator

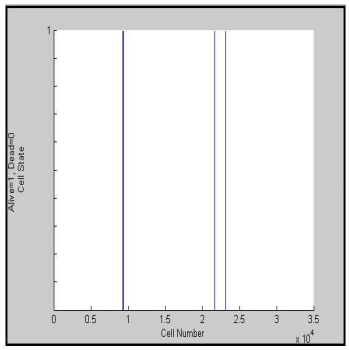
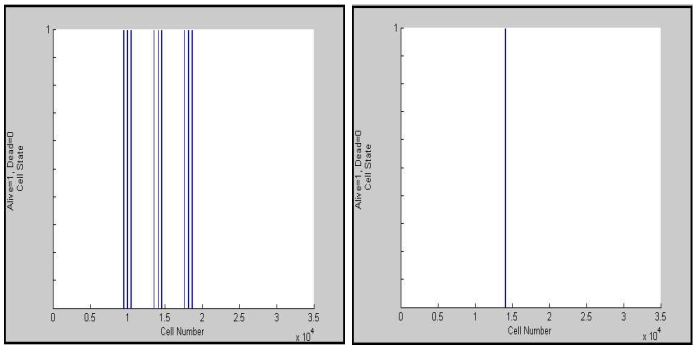


Fig. 12. 5D Invariant



(a) State A (b) State B

Fig. 13. 5D Oscillator

We can formulate a rule for the configuration of an invariant in any dimension as $newLife = overpop = (numOfLiveCellsInSeed) + 1$ and $underpop = -1$. A

value of $newlife$ as the number of live cells + 1, does not allow for any new life to be created and the same overpopulation value does not result in the death of the cell. Thus the total number of cells remains invariant. Table 1 summarizes the results obtained.

Table 1. Summary of Results

Configuration	Seed	Rule
3D-Glider	(3,3,3),(3,4,4),(3,4,5),(4,3,3)	(3,0,0)
3D-Oscillator	(3,2,3),(3,2,4),(3,2,5),(3,3,3),(3,3,5)	(5,4,5)
3D-Invariant	(1,2,2),(1,3,2),(2,2,2),(2,3,2)	(3,3,2)
4D-Oscillator	(3,3,3,3),(3,3,3,4),(3,3,3,5),(3,3,4,3),(3,3,4,4) (3,3,4,5),(3,4,3,3),(3,4,3,5),(3,4,4,3),(3,4,4,5)	(10,-1,-1)
4D-Invariant	(2,2,2,1),(2,2,2,2),(2,2,1,2),(2,2,1,1),(5,5,2,2) (5,5,2,1),(5,5,1,2),(5,5,1,1),(5,2,2,2),(5,2,2,1) (5,2,1,2),(5,2,1,1),(2,5,2,2),(2,5,2,1),(2,5,1,2) (2,5,1,1),(5,5,5,3),(5,4,5,3),(4,5,5,3),(4,4,5,3)	(3,3,2)
5D-Oscillator	(3,3,3,3,3),(3,3,3,3,4),(3,3,3,3,5),(3,3,3,4,3) (3,3,3,4,4),(3,3,3,4,5),(3,3,3,5,5),(3,3,4,3,3) (3,3,4,3,4),(3,3,4,3,5),(3,3,4,4,3),(3,3,4,4,5) (3,3,4,5,3),(3,3,4,5,4),(3,3,3,5,3),(3,3,3,5,4) (3,3,4,5,5)	(17,-1,-1)
5D-Invariant	(2,2,2,2,4),(2,2,2,1,4),(2,2,1,2,4),(2,2,1,1,4) (5,5,2,2,4),(5,5,2,1,4),(5,5,1,2,4),(5,5,1,1,4) (5,2,2,2,4),(5,2,2,1,4)	(11,11,-1)

6 Conclusion

In this paper, we have presented the findings of the basic configurations of the GOL in higher dimensions. We proposed two techniques, the *simple projection* technique to visualize the configurations in the four-dimensional GOL, and the *pulse of life* technique for dimensions greater than four. The pulse of life technique can also be used to verify the properties of basic configurations in lower dimensions as well. Also, this technique can be used to verify whether a complex pattern attains a stabilized state after a certain number generations. We have also proposed the *PR-CF* algorithm that applies a heuristic to reduce the number of rules to be tested for finding a configuration. This algorithm was successful in finding invariants and oscillators in 3D, 4D and 5D and a glider in 3D. All these configurations discovered were visualized using the visualization techniques mentioned.

Further enhancements in the PR-CF algorithm can be considered for dealing with the computational complexity in the discovery of configurations like gliders in 4D and higher dimensions in future work.

References

1. Berlekamp, E.R., Conway, J.H., Guy, R.K.: Winning Ways for your Mathematical Plays. Volume 2. Academic Press, ISBN 0-12-091152-3 (1982) chapter 25.
2. Bays, C.: Candidates for the game of life in three dimensions. Complex Systems **1**(2) (April 1987) 373–400
3. Bays, C.: The discovery of a new glider in the game of three-dimensional life. Complex Systems **4**(6) (December 1990) 599–602
4. Bays, C.: A new game of three-dimensional life. Complex Systems **5**(1) (February 1991) 15–18
5. Bays, C.: A new candidate rule for the game of three-dimensional life. Complex Systems **6**(5) (1992) 433–442
6. Reiter, C.A.: The game of life on a hyperbolic domain. In: Computers and Graphics. Volume 21. (September-October 1997) 673–683
7. Pulsifera, J.E., Reiter, C.A.: One tub, eight blocks, twelve blinkers and other views of life. In: Computers and Graphics. Volume 20. (May-June 1996) 457–462
8. Arenas, Y., Pérez-Aguila, R.: Visualizing 3d projections of higher dimensional polytopes: An approach linking art and computers. In: Memorias del Cuarto Congreso Nacional de Ciencias de la Computacion, Facultad de Ciencias de la Computacion, BIJAP (November 2006)
9. Foley, J.D., van Dam, A., Feiner, S.K., Hughes, J.F.: Computer Graphics: Principles and Practice in C. Second edn. Addison-Wesley Professional (August 1995)
10. Banks, D.: Interactive manipulation and display of surfaces in four dimensions. In: SI3D '92: Proceedings of the 1992 symposium on Interactive 3D graphics, New York, NY, USA, ACM (1992) 197–207
11. Gardner, M.: Wheels, Life, and Other Mathematical Amusements. W. H. Freeman and Company, New York (1983) ISBN 0-7167-1589-9.
12. E. Sapin, O.B., Chabrier, J.: Research of complex forms in cellular automata by evolutionary algorithms. In: Book Series Lecture Notes in Computer Science. Volume Volume 2936/2004., Springer Berlin / Heidelberg 357–367
13. Bosch, R.A.: Integer programming and Conway's Game of Life. SIAM Rev. **41**(3) (1999) 594–604
14. Aguila, R.P.: The extreme vertices model in the 4d space and its applications in the visualization and analysis of multidimensional data under the context of a geographical information system. M.Sc thesis. Master's thesis, Universidad de las Americas, Puebla, Puebla, México (May 2003)
15. Banchoff, T.: Beyond the Third Dimension. Scientific American Library (1996)
16. Kahle, J.A., Day, M.N., Hofstee, H.P., Johns, C.R., Maeurer, T.R., Shippy, D.: Introduction to the cell multiprocessor. IBM J. Res. Dev. **49**(4/5) (2005) 589–604