# Generalized Engel's Algorithm for Minimizing Playing Time to Stabilize the Initial Configuration and for finding Absorbing Probability

Namrata Kaushal
Department of Engineering Mathematics,Indore Institute of Science and Techanology,Indore-453331,MP, India

Madhu Tiwari
Department of Mathematics,Govt. Girls Post graduate College,Ujjain-456001,MP, India

Virendra Singh
Department of Computer Science,Indore Institute of Science and Techanology-II,Indore-453331,MP, India

C.L.Parihar
Indian Academy of Mathematics,
500-Pushparatan Park(Devguradiya) ,
Indore-452016 MP, India

## ABSTRACT
In this paper a generalized Engel's algorithm based on known Engel's algorithm has been introduced. Using this algorithm playing time of chip-firing game which is defined on directed graph, can be minimized for evaluation of absorbing probability of an absorbing Markov chain. Here proposed algorithm has been compared empirically in terms of timings, for playing game as well as for determining absorbing probability. As MATLAB is a high-performance language for technical computing, hence hare performance of generalized algorithm will be analyze by MATLAB language.

## Keywords
Chip-Firing Game, Configuration space, Critical loading. (AMS 2010)Subject Classification: 03E20, 03G10, 06D75, And 05D 40

## 1. INTRODUCTION
Chip-firing game is discrete dynamic model (G, L (μ),and ≤) which consists of a directed multi-graph G, the set of configurations on G i.e. L (μ) and an evolution rule ≤ called firing rule on set of configurations. Here, a configuration μ on G is a map from the set V (G) of vertices of G to non-negative integers associate a weight to each vertex, which can be considered as a number of chips stored in the vertex. In a configuration μ, vertex v is firable if v has at least one outgoing edge and μ (v) is at least the out-degree of v called evolution rule [1, 2]. When v is firable in μ, μ can be transformed into another configuration $\mu'$ by moving chips stored in v along each outgoing edge of v. We call this process firing v, and write $\mu \to \mu'$. An execution is a sequence of firing and is often written in the form $\mu_1 \xrightarrow{v_1} \mu_2 \to \mu_3 \to \mu_4 \to \dots\dots\dots\dots \xrightarrow{v_{k-1}} \mu_k$. The set of configurations which can be obtained from CFG by a sequence of firing is called configuration space, and denoted by L ( $\mu$ ). Chip firing games were introduced independently, many time but this is not counting the special cases of general Petri nets, or the obvious similarity with neural nets, which remains unexplored.

Engel [4, 5] considered a chip-firing as a procedure called the "probabilistic abacus" to determine the absorption probabilities and should be access time of certain Markov chains by combinatorial means. In [5] Arthur Engel provides a chip-moving algorithm on directed graph to describe the basic qualities of an absorbing Markov chain, this algorithm depends on recurrence of the initial distribution of chips.

### 1.1 Engels algorithm for Chip-firing Game
CFG defined on directed graph are classified on the basis of transition vertex called transition states and absorbing vertex called absorbing states.Initially game starts by placing $d^+(k) - 1$ chips on each transition states this is called "critical loading" .At this stage no node can be fired hence now place an additional chip on transition states and play the chip firing game until it terminates. If it terminates with the critical position on the transition states, stop, otherwise, feed a new chip and play the game until it terminates. Process will be continued as long as the critical position does not reappear on the transition states .If it does then we stop.

Algorithmic aspects of a chip-firing game on a digraph have the properties that every starting configuration leads to a so-called critical configuration. In [8] it has been proved that the number of steps needed to reach a critical configuration is polynomial in the number of edges of the graph and the number of chips in the starting configuration, but not necessarily in the size of the input. Critical loading reoccurs in Chip-firing game has been proved in [9] by using properties of upper locally distributive lattice, in this we found that manually to perform the entire procedure by Engel algorithm playing and terminating game takes long time, so here in this paper we have proposed an generalized Engel algorithm by which time of playing and getting the final configuration minimizes. So this result is very useful for various areas of mathematics, and the sciences at large.

## 2. BASIC DEFINITIONS AND NOTATIONS
### 2.1 Markov Chains
A sequence of stochastic events is completely based on probabilities rather than certainties, where the current state of a variable or system is independent of all past states, except the current state, this process is called Markov chains.

### 2.2 Absorbing Markov Chain
A state $s_i$ of a Markov chain is called absorbing if it is impossible to leave it (i.e., $p_{ij} = 1$). A Markov chain is absorbing if it has at least one absorbing state, and if from every state it is possible to go to an absorbing state.

## 2.3 Properties of Transient State and Absorbing State

Transient States may only Progress forward until reaching an Absorbing State.

Transient States may skip other transient states and go directly into an Absorbing State.

Transient states may revert back to previous transient states before reaching Absorbing.

Transient States have the chance to stay the same, however, the chance of this being true is not equal to 1, hence the following:

If $t_a$ goes to $t_a$ in element <a,a> and $P \neq 1$ then Transient

If $t_a$ goes to $t_a$ in element <a,a> and $P = 1$ then Absorbing.

In absorbing Markov chain at least one absorbing state should be

In absorbing Markov chain, it is possible to go from any transition state to an absorbing state in one or more then by one stapes.

## 2.4 Critical Loading

Critical loading is one in which each node has one less chip that it needs to fire, i.e. $c_i = r_i - 1$.

## 2.5 Absorbing Probability

A state Si of a Markov chain is called an absorbing state if, once the Markov chains enter the state, it is impossible to leave that state. Therefore the probability of leaving that state would be zero and it is shown as $p_{ii} = 1$ . And probability of absorbing the Markov chain is called absorbing probability.

## 2.6 Absorbing Probability Matrix

A Markov transition matrix is a square matrix describing the probabilities of moving from one state to another in a dynamic system. In each row are the probabilities of moving from the state represented by that row, to the other states. In transition matrix if probability are Absorbing Probability then transition matrix is called Absorbing Probability Matrix.

## 2.7 Criteria for input Matrix

1. Since programming is done for a particular strategy, so only those matrices which follow same strategy can be examined otherwise it shows that matrix is invalid.
2. Input matrix should be strictly absorbing probability matrix i.e $p_{ij} = 1$ where i=j.
3. The entire element of any columns of input matrix should not be zero.
4. If there is only one absorbing state then absorbing probability on that state is unity.

## 3. MAIN RESULT:

Generalized Engel's algorithm

**Generalized Engel's algorithm**

***Description:*** *Input-absorbing probability matrix.*

*Output: absorbing probability*.

**Global variable used:**

```
int   MD[n], D[n][n], N[n] [n] ;
int   x, y, a, j=0, c, count=0 ;
int   new[n], temp[n], sum=0, A[n]
;
```

**Functions used in Algorithm**:

```
void read_matrix(void) ;
void fire_max_D(void) ;
void count_nzero_D(void) ;
void fire_chip (void) ;
int fire_nzero_P(int) ;
void fire_node ( int, int[ ]) ;
```

**Algorithmic procedure**:

```
int main( )
{
        int  k,L ;
        bool x ;
        read_mat ( );
        find_max_D( );
        k=find_nzero_P( 0 ) ;
        y=k;
        for all i such that 0≤i<n,
                set new[i]=MD[i];
        fire_node ( k,new [ ]) ;
        k= find_nzero_P(k+1);
        x= true;
        do
        {
                if   temp[i] < MD[i] && MD[i] != 0
&& k<j ; then
                {
                        L= find_nzero_P(k+1) ;
                        fire_node (L,new [ ]) ;
                }
                else if temp[i] = = MD[i] && MD[i]
!=0 && k<j , then
                        fire_node (k,new[ ]) ;
                else
                        k++ ;
                for  i=0 , i<n   i++ ,
```

```
                    if temp[i] == MD[i]  then
                            count ++ ;
                if   count = c-1, then
                    x = false ;
        } while (x);
        for  i=0  to  n-1,
                if   temp[i] != MD[i] && MD[i] != 0
        then
                    temp[i] = MD[i];
        for i=0 to n-1
                if  MD[i]= =0  then
                    sum=sum + temp[i];
        // print output ;
        for  i=0 to n-1
                if   MD[i]= = 0 then
                    print temp[i]/sum= y+1,i+1;
        return 0;
}
```

**Function definition:**

```
 void  read_mat ( )
{
        read numerator in N[n][n] and denominator in
D[n][n] of input matrix;
}
void  find_max_D ( )
{
        for all i such that  0≤i<n
                set MD[i]=0 ;
        for all i, such that 0≤i<n;
                for all j such that 0≤j<n;
                    if MD[i][j] then
                        MD[i]=D[i][j]-1 ;
}
int   find_nzero_P (int a)
{
        for all i; such that a≤i<n ;
                if MD[i]!=0 then break ;
        return i ;
}
void   fire_node(int k, int new [ ])
{
         int i, j ;
         new [k]=MD[k]+1 ;
         temp[k]=new[k] ;
        for all j such that 0≤j<n
                if  j !=k then
                    temp[j]=MD[j]+new[k]*N[k
                ][j]/D[k][j]
                else
                    temp[j]=new[k]*N[k][j]/D[k
                ][j] ;
}
void   count_nzero_D( )
{
        c=0 , A[i]= 0 ;
        for all i=0 to n-1
                if  MD[i] !=0 then
                    c++ ;
         return c ;
        }
```

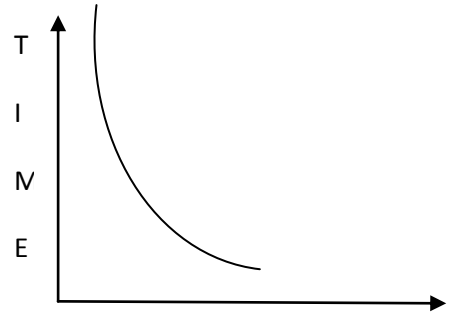


**Fig: 3.1**

## 4. ANALYSIS OF ALGORITHM

On analyzing all the results presented in [6,8] we found that, the bounds on the number of chip movements are polynomial in number of vertices (n) and number of edges (m), but not necessarily in the size of the input. In particular, if norm of the configuration is large, then the number of chip movements also is large as well and for each movement of chip in game we require a firing, which takes time by Engel algorithm that's why a generalized Engle algorithm presented in this paper can be used to explain the results of [6, 8] by some different approaches.

## 5. CONCLUSION

[3, 10, 11] Availability of technical computing environment such as MATLAB provides an opportunity to easily conduct numerical experiments and to tackle realistic and more complicated problems. [7] Performance of generalized algorithm for absorbing probability matrix has been examined by MATLAB; during examination we found that time to stabilize the critical configuration is inversely proportional to number of absorbing stats in critical configuration, while it is directly proportional to order of matrix. Moreover all results have been calculated in micro seconds, by fixing any one strategy. Graphical representation **(3.1 & 3.2)** is self explanatory for the

## 6. ACKNOWLEDGEMENT

The authors are grateful to the anonymous referee for a careful checking of the details and for helpful comments that improved this paper.

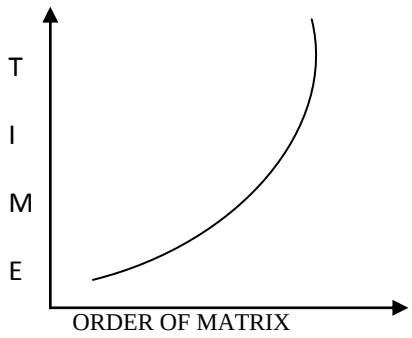


**Fig: 3.2**

## 7. REFERENCES

[1] Bjorner. A, Lovasz .L and Schor. P. W. **1991**. "Chip-firing games on graphs". Europ. J. Combinatory. Vol.**12** pp-283-291

[2] Bjorner. A and Lovasz .L. **1 992**. "Chip-firing games on directed graphs". J. Algebraic Combinatory. Vol.1 pp-304- 328

[3] Dhar.D, Ruelle,P, Sen.S, and Verma.D.**1995** . "Algebraic aspects of abelian sandpile models"J. Phys.A Vol.28 pp-805-831

[4] Engel. A. **1975**. "The probabilistic abacus". Educ.Stud. In Math. Vol.**6** pp-1-22

[5] Engel.A.**1976**. "Why does the probabilistic abacus work". Educ. Stud. In Math. Vol.**7** pp-59-69

[6] Eriksson.K.**1991. "**No polynomial bound for the chip-firing game on directed graphs". Proc. Amer. Math. Soc. Vol.**112 ,** pp-1203-1205

[7] Gilat.A.**2004.** "MATLAB: An introduction with Applications". John Wiley and Sons

[8] Heuvel .J. D. **2001**. "Algorithmic Aspect of Chip-Firing Game" .Combinatory Probability and computing

[9] Kaushal.N, Tiwari.M and Parihar.C.L. **2014**."Chip-firing Game as Probability Abacus by Characterization of ULD Lattices". Asian Journal of Mathematics and Applications (Science Asia Pub.) Vol.**0617** pp-1-8

[10] Moler. C.B. **2004**. "Numerical Computing with" MATLAB. Siam

[11] Polking .J.C and Arnold.D.**2004**. "ODE using MATLAB" Prentice Hall