Deadlock Detection and Recovery in P Systems

Samar H. Hassan
Faculty of Computers and information Cairo University
Egypt

Lamiaa H. Ahmed
Faculty of Computers and information Cairo University
Egypt

Amr A. Badr
Faculty of Computers and information Cairo University
Egypt

ABSTRACT

A P system is a computational model in computer science abstracted from the structure of real living cell that preforms some calculations under a certain conditions. They are distributed, parallel and run in a non-deterministic manner. This paper presents a model for solving the deadlock that could be happen between P system membranes. The P system model of degree n is used to simulate the deadlock that could be happen and the procedure of detecting the deadlock and removing it by triggering set of rules to interfere and solve the deadlock instead of killing the rules that cause the deadlock.

Keywords

Membrane computing; P systems; Deadlock problem; Deadlock recovery.

1. INTRODUCTION

In the last few decades the researchers have noticed that there was a great explosion in biological knowledge due to the spawning of various genome projects that has created new fields and industries, and a need for trained computational biologists who are familiar with Biology, Mathematics, and Computer Sciences. This kind of field is called Bioinformatics that can be seen as a computer science model or application that uses biological data and algorithms to solve certain problems that faced computer science field but there is an important question to be asked, Dose the structure and the functioning of a living cell can provide any solutions or any suggestions to computer science problems?. Membrane computing will be the possible answer to the previous question. P system is a computational model that was introduced by Gheorge Paun as a class of parallel, distributed and nondeterministic computing devices inspired by some biological features of a real cell [1]. The essential ingredient of P systems is a membrane structure which consists of several numbers of membranes in a hierarchical manner all these membranes embedded in a main membrane which called skin membrane. The membrane structure consists of several membrane-delimited compartments; each membrane can contain objects which can be strings or symbols over a given alphabet [2]. These objects can evolve according to specified evolution rules, which also determine the communication of objects between membranes. The evolution rules are applied in maximally parallel nondeterministic manner [2].Consider the following P system [1]:

$\Pi = (O, \mu, w_3, R_1, R_2, R_3, i_o)$

Where:

$O = \{a, b, c, d, e, f\}$;

$\mu = [_1[_2[_3]_3]_2]_1$;

$w_3 = a\ f\ c$;

$R_1 = \{e \rightarrow (e, out), f \rightarrow f\}$,

$R_2 = \{b \rightarrow d,\ d \rightarrow de,\ ff \rightarrow f,\ cf \rightarrow cd\delta\}$,

$R_3 = \{a \rightarrow ab,\ a \rightarrow b\delta,\ f \rightarrow ff\}$

The structure of this example is shown in figure 1 from [1]. In membrane 3, there are three objects *a, f and c.* There are no other objects in the other membranes; therefore, the only membrane that can start applying its rules is membrane 3. The rule r_1 can be repeated in parallel with r_3, resulting in the growth of the number of copies of *b* and the doubling of copies of *f* in each step. In case of applying r_2 in parallel with r_3 then the system will not achieve a halting status. Thus, in order to ever halt, membrane 3 must dissolve. When membrane 3 is dissolved, its contents (n + 1 copies of *b*, 2^{n+1} copies of *f* and one copy of the catalyst *c*) are left free [3].

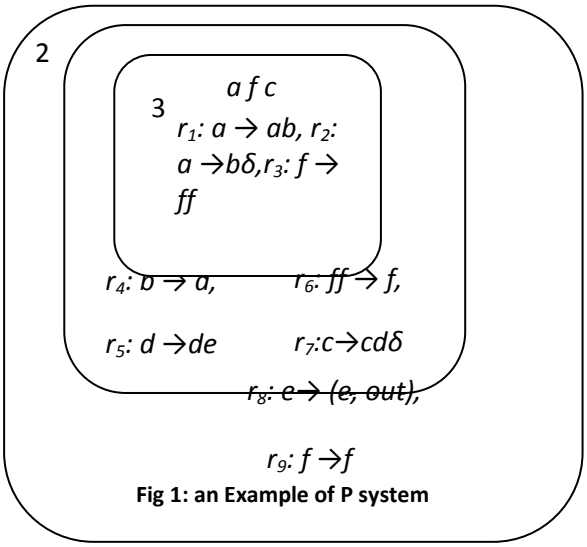

Fig 1: an Example of P system

in membrane 2, which now can start using its rules .In the next step, all objects *b* become *d*. When r_7 dissolves membrane 2, the contents of this membrane are passed to membrane 1. When the objects of membrane 2 arrive to membrane 1, and there is at least one copy of *f* then the rule r_9 from region 1 can be used forever and the computation never stops. Also, if the rule r_6 is used at least once in parallel with the rule r_7, then at least one copy of *f* is present. Therefore, the rule r_7 should be used only if region 2 contains only one copy of *f* [3]. This means that the rule r_6 was always used for all the available pairs of *f*, that is, in each step the number of copies of *f* is divided by 2. This is already done once in the step when all copies of *b* become *d*, and will be done from now on as long as at least two copies of *f* are

present. Also, in each step each *d* causes the production one copy of *e*. This process can continue until there is a configuration with only one copy of *f* present; it will be the time to use the rule r_7 because rule r_6 is no longer applicable, and membrane 2 is also dissolved. The parallel application of rule r_5 for all copies of *d* (there are n + 1 such copies), during n + 1 steps, the system will have (n + 1) (n + 1) copies of *e*, n + 2 copies of *d* (one of them was produced by the rule r_7), and one copy of *c* present in the skin membrane of the system [3]. The object *e* is sent out, and the computation halts. Therefore, the computation in this way the number $(n + 1)^2$, for some $n \geq 0$, that is, N (II) = $\{n^2 \mid n \geq 1\}$ [3].

1.1 Transportation in P Systems

Transportation in P systems can be defined as a set of mechanisms that regulate the passage of substances such small molecules through biological membranes, under a certain conditions [4][5]. There are several mechanisms such as using transportation rules, absorption, P systems with symport/antiport rules that uses channels to transport molecules between membranes [6] and P system with carriers that uses special rules called carrier rules to carry a certain object to and from the membrane [7]. But in this paper transportation rules will be used.

1.2 Deadlock in Membrane Systems

If the applying rules in the same string have mixed target indications then there will be consistency problems which called a Deadlock state [8] as shown in figure 2 from [8]. For instance consider that there are two evolution rules in different membranes that share the same objects (strings or symbols), Rule1 and Rule2, where Rule1 holds Object1 and is waiting for Object2 that held by Rule2 to be applied, and Rule2 holds Object2 and is waiting for Object1 that is also held by Rule1 to be applied too. In this case these two rules will wait forever and the P system will be in a deadlock state. In other words, the deadlock occurs in P systems when the rules with mixed target indications are applied in the same time on the common objects (strings or symbols) and the objective is to create a model to solve the deadlock [8].

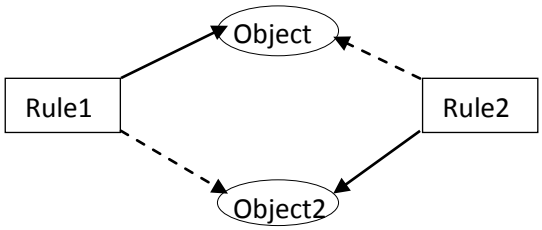

Fig 2: Resource allocation Graph with deadlock

2. SOLUTION APPROACH

There are several methods to handle deadlock such as deadlock prevention, detection and avoidance. This paper is concerned with deadlock detection and system recovery. This model simulates the deadlock that may be occurring in any living cell .Suppose that there are n rules in a living cell.

The construction of membrane system with deadlock detection will be as follows:

$II= (O, \mu, w_1, w_2 \ldots, wn, R_1, R_2 \ldots Rn, D_{[i,j][k,l]}, S_{[a,b][c,d]}, i_o)$

Where:

O: The alphabet of objects, i.e cellular molecules.

μ: The membrane structure .It consists of n membranes labeled with 1,2,3,..n.

w_1, w_2, w_n : Represent the multi set of objects initially present region 1,2,..n. these strings will be formed by the user's input and output of some rules [1].

R_1, R_2, R_n : The set of evolution rules associated with the region of the system.

$D_{[i,j][k,l]}$: Refers to the set of deadlocks that may occur in the system where:

- i,k refers to membrane number
- j,l refers to rules number
- $D_{[I,j][k,l]}=\{D_{[m1,r1][m2,r2]}, \ldots, D_{[m3,r3][m4,r4]}\}$ i.e., $D_{[1,2][2,5]}$ refers that there is a deadlock occurs between membrane 1 rule 2 and membrane 2 rule 5.

$S_{[a,b][c,d]}$: Represents system's deadlock recovery where:

- a,b refers to membrane number
- c,d refers to rules number
- $S_{[a,b][d,c]}=\{Object\_1_{[m,r][k,l]},..,Object\_2_{[e,f][h,l]}\}$ i.e., $Object\_1_{[1,3][3,4]}$ means that the system recovers from the deadlock by triggering rule 3 in membrane 1 to produce object_1 and transport it to membrane 3 ,rule 4 to solve the deadlock.

i_o: The output region, that will take one of the labels 1,2,…n.

The proposed system is for handling a deadlock state will follow the following steps:

1. Initialization step: in this step the system generates random number of objects that simulates the objects in real living cell, a number of membranes and their associated rules.
2. Monitoring step: in this step the system monitor the behaviour of a cell during its execution if it detects a deadlock sate it will identify which rules causes the deadlock and its corresponding membrane.
3. Deadlock resolving: in this step the system relax the causes of the deadlock and return back to step 2.

The following section provides a complete example of the above producers will be given.

2.1 Illustrative Example

Here is an example about how the P system solves a deadlock problem in mitochondria cell which has three membranes as shown in figure 3 adapted from [9]

The construct of P system in this case will be as follows [9]:

$$\mathbf{II_{sim} = (O, \mu, w_1, w_2, w_3, R_{e1}, R_{e2}, R_{e3}, D_{[i,j][k,l]}, S_{[a,b][c,d]}, i_o)}$$

Where:

- O: The alphabet of objects.

O= {Glucose, NAD, ADP, P, Fatty-Acid, CoA, ATP, Acyl-CoA, H_2O, FAD, GDP, pyruvate, Carnitine, Acyl-CoA,

Acylcarnitine, Acetyl-CoA, H^+, GTP, 2 e^- , O_2 }

- **μ**: The membrane structure [2]. The plasma membrane (Skin Membrane) will take label 1, the outer mitochondrial membrane will take label 2 and the inner mitochondrial membrane will take label 3. The structure is $\mu = [_1 [_2 [_3]_3]_2]_1$. This means that membrane 3 is inside membrane 2 which is inside membrane 1[1].
- **w_1, w_2, w_3**: representing the multi sets of objects initially present region 1,2 and 3 respectively. These strings will be formed by the user's input and the output of some rules.

w_1={#Glucose, #Fatty_acid, #ADP, # H^+, # NADH, #Pyruvate, # NAD, #CoA, #P, # H_2O, #ATP, # Acyl-CoA}
w_2={#Acyl_CoA, #Carnitine, #Acyl_carnitine }
w_3={#Pyruvate, #NAD, #CoA, #P, # H_2O, #FAD, #GDP,# Acetyl_CoA, #Acyl_CoA, #Acyl_carnitine, #Aarnitine, # $FADH_2$, #NADH, # H^+, ADP, ATP**}**

- **R_{e1}, R_{e2}, R_{e3}**: The set of evolution rules associated with the three regions of the system.

Where:

- R_{e1}= {r_1 = *Glucose + 2NAD + 2ADP + 2P → (2Pyruvate,3) + 2ATP + (2NADH, 3) + ($2H^+$,3) + ($2H_2O$,3)* ,

  r_2 = *Fatty acid +2ATP + CoA → (Acyl_CoA, 2),*
  r3=Acyl-CoA+Carnitine → CoA+ (Acyl-Carnitine,2)[10]
- R_{e2} = {r_4 = *Acyl_CoA + Carnitine → (Acylcarnitine, 3),*
  r5= (Acyl-Carnitine,3)[10]
  }
- R_{e3} = {

  r_6 = *2Pyruvate +2 NAD + 2CoA →2Acetyl_CoA +2 NADH +2 H^++2Co_2,*
  r_7= *2Acetyl_CoA + 6NAD+2FAD + 3NAD + 2GDP + 2P + H_2O → 6NADH + 2$FADH_2$ + 2GTP +2 CoA + 3H^+ ,*

  r_8= *Acyl_CoA + 7FAD + 7NAD + 7CoA + 7H_2O → 8Acetyl_CoA + 7$FADH_2$ + 7NADH + 7H^+ ,*
  r_9=*Acylcarnitine + CoA → Acyl_CoA + (Carnitine,2)}*

- **$D_{[I,j][k,l]}$**: Represents the set of the deadlocks that could be happen in the system

Where:

- D= {$D_{[2,2][3,9]}$} which means that the deadlock happens between membrane 2 , r_4 and membrane 3, r_9. In this case the deadlock occurs because of r_4 in membrane 2 and r_9 in membrane 3, where r_4 holds *Acyl-Carnitine* and is waiting for *Carnitine* that held by r_9 to be applied, and r_9 holds *Carnitine* and is waiting for *Acyl-Carnitine* that is also held by r_4 to be applied too. In this case these two rules will wait forever and the P system will be in a deadlock state.

- **$S_{[a,b][d,c]}$**: Represents system's deadlock recovery

Where:

- S={*$Acyl\text{-}Carnitine_{[1,3][2,5]}$,$AcylCarnitine_{[2,5],[3,9]}$*} , recover from system's deadlock problem by triggering r_3 in membrane 1 to produce *Acyl-Carnitine* and transport it to (*membrane 2* ,r_5)which in turns will transport Acyl-Carnitine to (*membrane 3*,r_9) using transportation rule.

- i_o: The output region, in this case the output region is *ATP* in membrane 1.

3. OBSERVATIONS and RESULTS

A simulation model of mitochondria cell have been created, with its rules and objects in order to obtain the occurrence if the deadlock according to different aspects of the number of initial values of *Acyl-Carnitine* and *Carnitine* , number of iterations and number of runs. Table 1 shows the results obtained after running the system 5 times with different values with a different number of experiments. i.e. if the number of experiments is 100 this means that he system run 100 times in a one run and number of runs means that the system is being run the 100 experiments 5 times to get the average value.

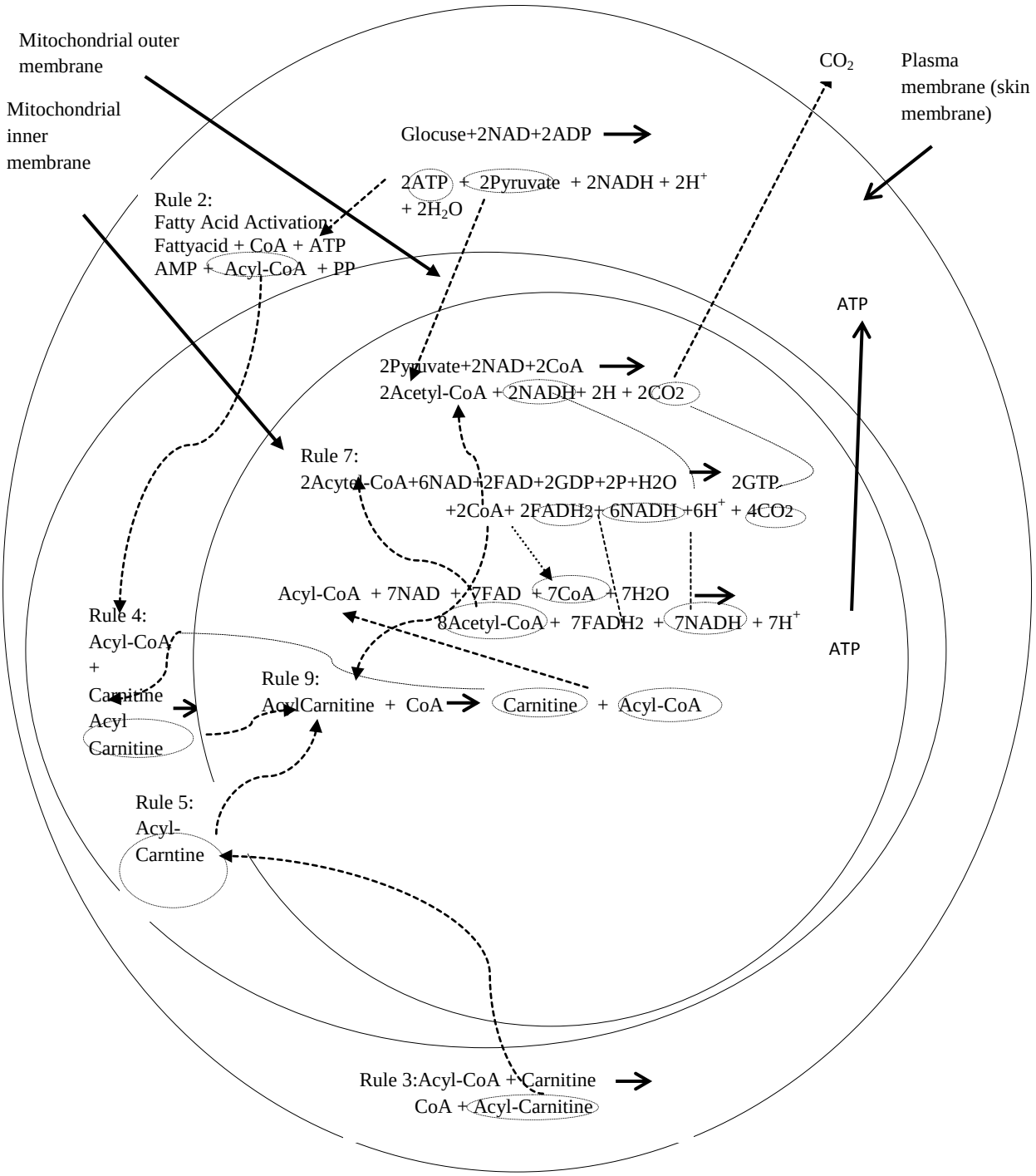

Fig 3: the Mitochondrial energy Production Cycle adapted from [9]

Table 1. Results from simulation model

A	B	C	D	E	F
				27	26
				28	22
				38	29
				29	28
1-10	1-10	100	5	27	23
				17	15
				21	17
				19	17
				25	21
1-50	1-50	100	5	19	17
				4	4
				10	9
				7	7
				15	13
1-100	1-100	100	5	8	9
				128	118
				125	112
				130	116
				135	116
1-10	1-10	500	5	123	112
				97	87
				105	87
1-50	1-50	500	5	106	97
				119	106
				105	90
				60	54
				43	38
				50	45
				51	41
1-100	1-100	500	5	43	40
				260	239
				268	241
				266	239
				229	210
1-10	1-10	1000	5	273	250
				207	190
				212	192
1-50	1-50	1000	5	215	99
				204	179
				202	180
				97	87
				105	93
				101	91
				95	86
1-100	1-100	1000	5	98	82

Where: column:

A: initial number of Acyl-Carnitine

B: Initial value of Carnitine

C: No. of Experiments in each run

D: No. of runs

E: Deadlock Occurrence (No .of Deadlocks occurred in a whole system)

F: Deadlock in Experiments (No. of experimenters that deadlock occurred in it).

Table 1 represents some experiments with different initial value of *Acyl-Carnitine* and `will be at deadlock state and how many times the proposed model can recover this deadlock. The noticeable mark is that every deadlock state happened in the model can recover the deadlock and it also noticeable that while the initial values of *Acyl-Carnitine* and *Carnitine* is from 1-10 the average percentage of deadlock occurred in the system is from 25.64% up to 29.8% and the deadlock occurs in non-deterministic manner across the iterations, while the initial values is from 1-50 the average percentage of deadlock occurred in the system is from 20.2% up to 20.8% and the deadlock also occurs in non-deterministic manner across the iterations, and while the initial values is from 1-100 the average percentage of deadlock occurred in the system is only from 8.8% up to 9.92% and the deadlock also occurs in non-deterministic manner across the iterations. the conclusion is that while there are an enough numbers of objects the probability to have a deadlock is reduced.

5. COMPARATIVE STUDY

In this section the aim is to provide a comparative study between the proposed system and Kang's system [11].In The proposed idea of Kang's system is to remove the dead lock by determine which rule should be killed according to its killing time to ensure the minimum cost .This solution is suffering from two main problems which causes an extra delay time:

- Killing a rule that causes the deadlock necessitate recreating the killed rule which causes a delay time (waiting time).
- The situation of having an equal killing time (cost), to prevent the system from failing, the system

should be reinitialized which causes another delay time (waiting time). Having this situation the system is considered to be failed to recover from the deadlock.

To have a more comprehensive comparison study, the implementation of both Kang's system and the proposed system are done and experimented with equal parameters and the same case data mentioned in section 2.1, with an initial value of *Acyl-Carnitine* and *Carnitine* (1-50), and tested with different number of iterations (1000,500,100).

Table 2a, 2b and 2c contains the results of the preformed comparative study. Whereas figure 4 compares between the execution time between the proposed systems and Kang's system.

Table 2a. Comparative study results with 1000 iterations

Number of Iterations =1000		
	Proposed System	**Kang's System**
Avg.No for Successful recovery	208	125
Avg.No for Failed recovery	0	137.4
Time taken	7.154	14.2

Table 2b. Comparative study results with 500 iterations

Number of Iterations =500		
	Proposed System	**Kang's System**
Avg.No for Successful recovery	106.4	67
Avg.No for Failed recovery	0	65.8
Time taken	3.68	7.58

Table 2c. Comparative study results with 100 iterations

Number of Iterations =100		
	Proposed System	**Kang's System**
Avg.No for Successful recovery	20.2	12.4
Avg.No for Failed recovery	0	13.6
Time taken	0.66	1.52

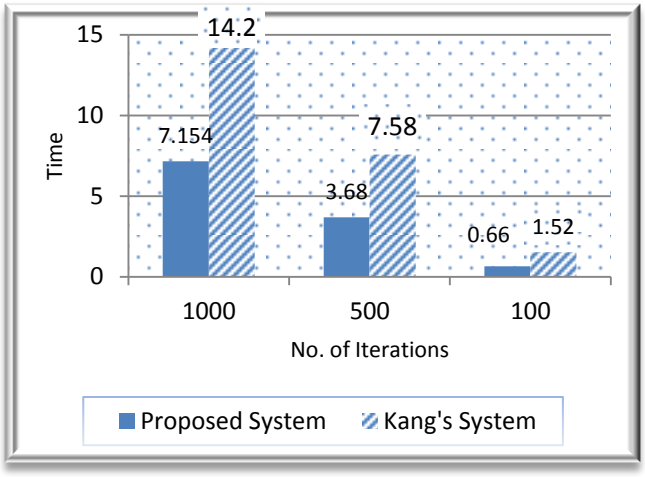

Fig 4. Execution time comparison

As shown in the above figure the proposed system is almost needs half execution time of Kang's system, the main reason behind this is the delay time mentioned above.

6. CONCLUSIONS

To further improve the efficiency of removing dead lock, this paper proposes a P system model to solve a deadlock without a need to kill any rule, only solve this problem by halting the rules that cause the dead lock and triggering a set of another rules to interfere to help this rules to resume its function again. There are many solutions to solve deadlock in membrane systems one of these solutions was introduced according to [11]. The solution proposed in mentioned paper is to remove the dead lock by determine which rule should be killed according to its killing time to ensure the minimum cost without any motioning of what the system should do if the killing time of more than one rules is equal to each other [11], this solution is not applicable in a real cell, killing a rule could leads fatal error in the cell which can lead to fatal error in the whole origin and the origin will fail.

7. REFERENCES

[1]Lamiaa.H.Abd El-Naby,"A Fuzzy Membrane System",Ph.D. thesis, Faculty of computers and information Cairo university,2012.

[2]Reid,D.Oddie, A.Hazlewood,P.Dept.of Computer Scinece,Liverpool hope University., Liverpool UK 2010 ,"Parallel Numerical P systems using MIMD based architecture",IEEE International Conference, pp.1646-1653.

[3]Păun,Gh..2002,"Membrane computing. An introduction", Springer, pp. 2-15.

[4]Gabrial Ciobanu 2003,"Distributed algorithms over communicating membrane systems", National university of Singapore, School of computing department of Computer science, ElSevier, pp 123-133.

[5]Sergey Verlan,Francesco Bernadini,Marian Gheorghe,Maurice Margenstern 2008,"Generalized communicating P systems", ElSevier ,Vol 4.4,pp 170-184.

[6] Rudolf Freund, Andrei Paun 2003. "Membrane systems

with Symport/Antiport rules: Universality Results", Springer, pp.134-145.

[7]Yan Gao, Hendrik J.Hoogeboom 2007,"P systems with single passenger carriers", International Journal of Foundation of Computer Science", Vol 18,Issue 6.

[8] Silberschatz, Galvin and Gagne 2013,"Operating System Concepts", Ninth Edition, Wiely,pp.322-337.

[9] Lamiaa Hassaan Ahmed, Amr Ahmed Badr, and Ibrahim Farag Abd El-Rahaman 2011, " A P-Simulator with Carriers of Cellular Respiration and Mitochondrial Oxidative Metabolism", International Journal of Computer Theory and Engineering, Vol. 3, No. 3,pp.448-465.

[10]Willam W.Christie "Carnitune and AcylCarnitine Structure Occurrence, Biology and Analysis" *lipidlibrary.aocs.org* (accessed: December 12 2013).

[11] Kang Wang, Hong Peng, Chenliang Zhu ,Yuhong Fan and Hao Wang 2011, "The P System Model for Solving Deadlock"Springer,pp.111-118.