Enzymatic Numerical P System based Workflow Scheduling in Cloud

S Raghavan[1] and K. Chandrasekaran[2]

[1] Indian Institute of Information Technology, Design and Manufacturing, Kancheepuram
raghavans@iiitdm.ac.in
[2] National Institute of Technology Karnataka Surathkal
kchnitk@ieee.org

Abstract. Workflow is an integral part of cloud infrastructure. Several algorithms have been proposed for scheduling them but primarily the proposed solutions are sequential. As Workflows get bigger with multiple tasks, they consume more time, and it becomes difficult to schedule them. Thus, an inherently parallel workflow scheduling algorithm using Membrane computing paradigm is developed. Membrane Computing is a computing paradigm inspired by living cells. Membrane computing is realized using devices called as P Systems and these follow a maximally parallel execution method. Our proposed approach for Workflow Scheduling is realized using Enzymatic Numerical P System (ENPS). ENPS is a numerical variant of P System. The ENPS model is the base with aspects of the Heterogeneous Early Finish Time (HEFT) model added, giving it a complete membrane-based parallel solution for workflow scheduling. The proposed algorithm has been implemented using GPUPeP (Simulator for ENPS) and a membrane generator is developed for automatic creation of complex membrane structure proposed here. The proposed method is compared with other standard methods with multiple number of tasks and workflow structures.

Keywords: Enzymatic Numerical P System (ENPS) · Cloud Workflow Scheduling · Membrane Computing · P System

1 Introduction

Cloud Computing has become as an inevitable part of today's technological industry. Cloud computing is a business model where everything is given as service to the user, primarily in pay as you go model [2, 13]. Workflows are collections of interconnected actions that typically have a single entrance point and a single exit point. A workflow aims to complete one major activity separated into several smaller, interconnected tasks. Workflow scheduling in the cloud is the process of allocating these workflow tasks to various processing units. As part of the virtualization process, virtual machines (VMs) are generated, and many of these VMs can be a component of a physical machine [16].

A simple workflow structure is given in Figure 1. Workflow Scheduling in general is a method of mapping tasks to computational resources satisfying the objective functions imposed by users. A workflow contains many tasks which are dependent on each other. It is a NP-Complete problem, and there is no polynomial time algorithm to accurately perform workflow scheduling.

Cloud Workflow Scheduling is a process of assigning the limited stock of VMs to the tasks of the workflows satisfying certain criteria. A scheduling criteria can be anything. The algorithm can be designed either to minimize or maximize the scheduling criteria. In this case the criteria is minimizing the makespan [12]. Makespan is the time taken between the starting of the first task and completion of the last task of the workflow. It is one of the prime objectives of workflow scheduling.

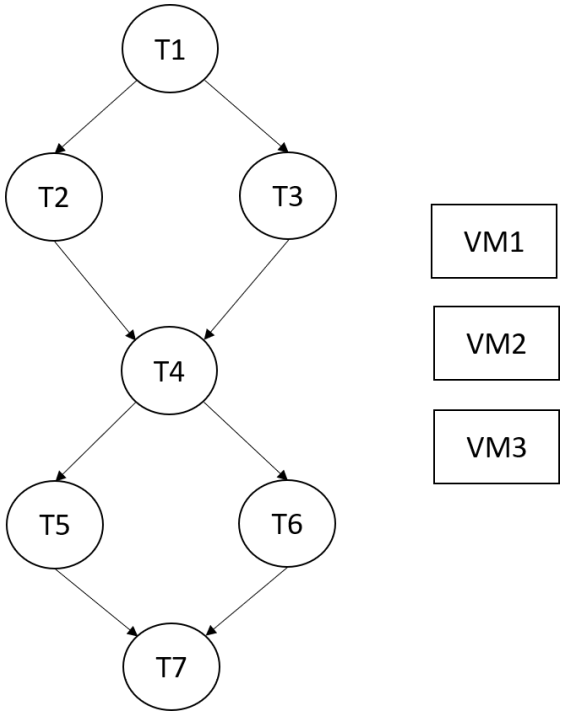

Fig. 1: Cloud Workflow Basic Structure

Figure 1 represents a simple workflow in cloud as a directed acyclic graph. In cloud computing, Infrastructure as a Service (IaaS) offers a suitable option for workflow administration with computational components. In particular, they give access to VMs of various kinds. These VMs can be used on-request, they can be rented whenever they are required and terminated when they are not.

As a rule, clients are charged just for what they utilize, as a rule in addition to charging period characterized by the supplier. This flexibility makes environment of IaaS perfect for the execution of logical workflows.

The aim of this paper is to find an inherently parallel method for solving this problem of workflow scheduling. The reason of selecting a parallel model is based on the fact that the complexity of workflow scheduling increases as the number of nodes of a workflow increases. A normal workflow algorithm usually takes about n^3 time for execution, where n is number of tasks. Hence, by using a parallel mechanism, this time complexity could be reduced.

There are several approaches that have been used for workflow scheduling in cloud which are discussed in the literature review, where it is inferred that there are very few parallel methods for workflow scheduling. Thus, an inherently parallel membrane computing model based on a heuristic approach is proposed for workflow scheduling in cloud. Membrane computing is a Natural Computing Model proposed by Gheorghe Paun [17]. It is a maximally parallel computing paradigm and is realized using devices called P Systems.

There are a total of six sections in this paper; the next section elaborates literature on workflow scheduling in Cloud and Membrane Computing Applications. The third section deals with the enzyme numerical P system (ENPS). The fourth section discusses the ENPS-based methodology for Workflow Scheduling. The prefinal section analyses the results. The final section concludes the work.

2 Literature Review

Workflows are an integral part of the cloud ecosystem. Of several important processes associated with workflows, scheduling tends to be an important one. A proper scheduling approach is an important component of a good workflow management system, as it ensures optimal utilization of resources. Many researchers have proposed several heuristic and meta-heuristic algorithms for scheduling cloud workflows [12]. Often the objectives considered are makespan, cost or energy.

Heuristics is a class of techniques in which, according to the nature of the problem, several specific approaches are applied to solve the problem more efficiently and quickly [38]. Usually, these techniques are used for problems where exhaustive solutions are a time consuming process (Many real-world problems). The other class of solutions, called meta-heuristic approaches, which are similar to heuristic since they search for the solution to optimization problems in an nonexhaustive and often in a probabilistic manner. There is a primary difference between heuristics and meta-heuristics, the former being problem specific and the latter not being so.Meta-heuristic approaches are primarily used when exhaustive methods are not available or are too expensive, and we can accept an approximation to an optimal solution. The problem of workflow scheduling considered in this study is a similar problem where obtaining exhaustive solutions is not

possible. There are several heuristic and metaheuristic approaches for solving this, out of which many are bioinspired approaches.

The objective of the work is to propose a workflow scheduler based on an inherently parallel computing paradigm. Workflow scheduling is a NP−Complete problem. There are many heuristic and meta-heuristic algorithms that have been proposed for this problem in literature [8, 16, 38, 47]. Usually, heuristics find out solution faster than meta-heuristic where as the latter consumes more time due to exploration of solution space. The advantage of meta-heuristic, over heuristic is its generality, thereby, allowing us to apply the algorithm anywhere, wherever it is suitable. There are many works in this area and few important works are discussed here.

Calheiros and Buyya [3] have proposed Enhanced IC-PCP with Replication (EIPR) algorithm which completes user tasks within given deadline in public cloud. Zhang et al. [40] have proposed a method, Iterative Ordinal Optimization (IOO) method. This method uses iterative approach. Similarly, Wu et al. [38] have proposed a heuristic algorithm named Critical Greedy Algorithm. This algorithm's main goal is to reduce the workflow's end to end delay under financial constraint. Zhu et al. [47] have proposed an Evolutionary Multi-objective workflow scheduling algorithm which optimizes both cost and makespan on IaaS.

Other than these algorithms discussed there are several other meta-heuristic algorithms available. Wu et al. [39] have proposed a meta-heuristic algorithm L-Ant Colony Optimization (L-ACO). The main objective of the algorithm is to minimize execution cost of workflow by meeting deadline constraint. Chen et al. [4] worked on a method for workflow scheduling called as Multi-Objective Ant Colony System (MOACS) approach for cloud workflow scheduling. Pandey et al. [16] have proposed a Particle Swarm Optimization (PSO) based algorithm. This algorithm shows that it is three times better than Best Resource Selection (BRS). Another work related to workflow scheduling is presented by Rodriguez and Buyya [30] using a meta-heuristic algorithm, PSO is used for resource provisioning and task scheduling on scientific workflows on IaaS, thereby minimizing execution time and meeting deadline constraint.

Nasonov et al. [14] have proposed a hybrid algorithm which combines both HEFT and genetic algorithm. The main objective is to improve the makespan of workflow, which the algorithm achieves. Mansouri et al. [11] have proposed hybrid task scheduling strategy Fuzzy System and Modified PSO (FMPSO) for improving makespan. Su et al. [33] have proposed cost efficient task scheduling algorithm using two heuristic strategies. The main objective is to reduce the makespan and this algorithms successfully does that.

There is an important method for the heterogeneous machines, that is one of the first heuristic methods for workflow scheduling [34]. The parameter considered for optimization is makespan of the workflow. The heuristic method proposed is one of the best heuristics for workflow scheduling over heterogeneous components and it has given better results than other standard methods of that time and is

considered as one of the best methods till date. This method's logic is considered in the present work to develop an approach that uses strict membrane computing structure for workflow scheduling in cloud.

Apart from these works there is an important work which has come in late 2014, where, for the first time Membrane computing model was used for Workflow Scheduling [1]. The developed model though based on membrane computing is quite different from the model that is proposed in this paper. Ahmed et al. [1] proposed a membrane inspired model that imbibes the structure of P System but it doesn't follow all the membrane-based execution rules. It is a workflow scheduling approach that uses only the hierarchical structure of membrane instead of using P System as a computing paradigm. Thus, the mentioned approach by [1] can be called as a membrane inspired method which solves the workflow in an hierarchical fashion and it does not use strict rule based model of P System.

Considering the literature for scheduling workflows, heuristic solutions would be a good choice for scheduling workflows and at the same time the time complexity should be reduced. Hence, a parallel model can be used to solve this problem of complexity on an exhaustive method. Thus, a suitable parallel model is to be analysed and selected. There are several parallel models available, many of them are directly based on hardware or software that is being used at the base. So, to have a generic model which can be inherently parallel at its core, the focus moved to natural parallel computing models. There are several Natural and Unconventional computing models available like DNA Computing, Amorphous computing [7]. Among all these models, Membrane computing is selected owing to its lucid and rich structure that assures maximal parallelism. P System, which are the devices used to realize membrane computing are found suitable for this problem, owing to its maximal parallelism property.

Membrane computing paradigm can be effectively applied to multiple problems, as membrane-based algorithms. Membrane-based algorithms are either loosely or strictly based on the membrane computing paradigm. Several problems have been solved using membrane-based algorithms and some of them have been mentioned here. P System based optimization algorithm have been developed by Huang and Wang [6] which was one of the first applications and later membranes were also applied for combinatorial optimization problem [42, 46]. Further in the area of image processing a method for image segmentation [35] and methods for image thresholding [22, 23] have been proposed. Membrane-based algorithms have been heavily applied for clustering, primarily to enhance their efficiency [24, 25]. The algorithms have also found its application to the area Evolutionary Computing, forming a completely new area called as Evolutionary membrane algorithm, which are discussed in detail by Zhang et al. [41]. Apart from this P System has also been used for enhancing Particle Swarm Optimization [32, 36].

P System is a vast area with several models being a part of its helm. An appropriate model to suit the needs had to be selected.

There are several variants of P System currently available for use. Primarily, they can divided into two types; Symbolic P System and Numerical P System. Symbolic P System uses symbols and rewriting rules which form as the base unit of this model [17, 19]. These were the first to be proposed and studied as part of P System models. Because of its symbolic nature, there were several problems that could not be modelled using these type of models; mainly it was difficult to model numeric based problems. Hence, there was a requirement of some variant which will allow numerical values to be used as part of the model. Later in 2006, Gheorghe Paun proposed Numerical P System (NPS) [18], designed for Economics related problems. This model holds a significant distinction of using numerical values, constants and variables as part of the core model. As with other P System variants, this also has properties of maximal parallelism but with numerical values. The problem that is considered in this paper is numerical problem so NPS is suitable. Though this model is good, there are two primary shortcomings of this model which restrict its efficient use to the numerical problems. The issues are; only one rule per membrane is allowed and there is no method specified to control the execution of rules in runtime. These restrictions limit its use for several standard numerical problems. Albeit both these problems were considered by Pavel et al. [21] and a solution was proposed in the form of a new variant called as Enzymatic Numerical P System (ENPS). ENPS allows simultaneous execution of multiple rules in a single membrane and additionally uses enzyme with each rule, that allows controlling rules based on the enzyme values.

There are several works which apply ENPS as the solution and some of them are mentioned here. One of the earliest application include a solution for robot localization problem by Pavel and Buiu [20]. Later it was used by Llorente Rivera and Gutiérrez Naranjo [10] for solving pole balancing problem. There are also a few important applications of ENPS for robot controllers [37, 43]. Further, there are a few latest works by Pérez-Hurtado et al. [26] where they use Rapidly-exploring Random Tree (RRT) with ENPS for robot path planning. Further, they propose an enhanced model for RRT and RRT* algorithm [27]. They also propose a simulator that has been used as part of the work. For realizing ENPS, there are a few dedicated Python based simulators available [5, 29]. ENPS has also been used for cloud service selection by Raghavan and Chandrasekaran [28]. The latest ENPS applications include implementation using FPGA [31].

Further, there are several new variants of NPS available [15, 44, 45] but ENPS model is found to be perfectly suitable for the problem of Workflow Scheduling and hence has been used here.

3 Enzymatic Numerical P System

There are a few classification of P System based on the structure of the model, like Cell-based P System, Tissue-based P System and Neural P Systems [19]. Each of them is based on different type of cells. The cell based P system struc-

ture is one of the firsts to be proposed and this structure is considered as the base for NPS. The cell based P system's general structure consists of a skin membrane, inside which there can be any number of membranes (child membranes). Each membrane further can have any number of membranes, inside it. The child membranes inside each membrane can communicate among each other, i.e., each of them can pass information to one another. Each membrane has rules and these are the basic building blocks of the model [17, 19]. This is the basic structure in general and based on this cell structure there are several variants that have been proposed. One such variant is Enzymatic Numerical P System (ENPS), which is a part of Numerical P System family [18, 21].

There are two important components of NPS: programs (rules) and enzymes (variables). The NPS model has a little different structure from normal rule based structure. A formal definition of NPS is given as in the equation 1 [18].

$$\prod = (m, H, \mu, (Var_1, Pr_1, Var_1(0)), \ldots, (Var_m, Pr_m, Var_m(0))) \quad (1)$$

H is an alphabet with m symbols (used as labels of membranes), where m is the number of membranes used in the system, m $\geq$ 1,

μ is the membrane structure with m membranes.

Var_1 to Var_m are the set of variables that are available.

$Var_1(0)$ to $Var_m(0)$ are the initial values of first to last variable, that have been defined.

Pr_1 to Pr_m denote the available programs in the whole membrane system.

A rule in NPS is called as a program [18]. There are two components of a program; the production function and re-partition protocol. Production function is like any other function with some expression that can be represented with basic arithmetic operations. The result of the function is passed into the next component called as re-partition protocol. The re-partition protocol consists of a set of dual values (one variable and it corresponding constant). The values passed from the production function is proportionally divided into the variables available according to the constant value of each variable.

In an NPS model, there is a skin membrane and the program/ membranes inside the skin membrane can be executed any number of times (cycles). Inside skin membrane, there can be any number of membranes but here each membrane can have only one executable program at a point of time (though many programs can be defined in membrane, only one can be selected and executed). To solve this problem, there is another variant of NPS called Enzymatic NPS that was proposed by Pavel et al. [21]. This model adds enzyme, a numerical variable over NPS and further relaxes the single program per membrane rule.

There are two types of programs available for ENPS, one is normal program (as for NPS as in equation 2) which is defined as follows, according to [9, 20, 21],

for a given region i, $x_{l,i}, .., x_{k_i,i}$ be some variables from Var_i and

let $F_{l,i}(x_{l,i}, ..., x_{k_i,i})$ of a given program $P_{l,i} \in Pr_i$ and $c_{l,1}, .., c_{l,n}$ are considered to be positive integers. [9]

$$F_{l,i}(x_{l,i}, ..., x_{k_i,i}) \;\; \rightarrow \;\; c_{l,1} \mid v_1 + c_{l,2} \mid v_2 + \ldots + c_{l,n_i} \mid v_{n_i} \tag{2}$$

and the next program, mentioned in 3 is for ENPS,

$$F_{l,i}(x_{l,i}, ..., x_{k_i,i}) \mid_{e_{j,i}} \;\; \rightarrow \;\; c_{l,1} \mid v_1 + c_{l,2} \mid v_2 + \ldots + c_{l,n_i} \mid v_{n_i} \tag{3}$$

With addition of enzyme, the execution method of a program slightly differs in ENPS. The enzyme here is the primary component which will control the execution of each program. A program is executed only if the value of the enzyme is greater than the smallest value of the variables used in the production function [21]. Thus, if the condition is not satisfied the program is not executed for that particular cycle. Further, there can be any number of programs in each membrane with each program having same or different enzyme values. This paper uses ENPS as the base model for scheduling workflows in cloud.

4 Scheduling in Cloud using Enzymatic Numerical P System

Workflow Scheduling in Cloud is a complex process and a parallel, P System based model is proposed as a solution. As per the literature, there are several P System variants to choose. As the workflows involve numerical value comparison, ENPS is deemed suitable for the problem. For sequential scheduling logic, Heterogeneous Earliest Finish Time (HEFT) is considered as the base [34].

The P System based method developed in this work primarily involves generating an order with which the tasks are scheduled on heterogeneous components. This process is being done for a cloud based workflow and thus the heterogeneous components are VMs that are of different sizes. To generate rank, two important data about the workflow is required:

- The execution time of each task on the available VM
- The communication cost between each and every VM $t_{i,j}$, which transfers control to other VM in the current workflow

It is represented by using an adjacency matrix. The whole process is logically similar to HEFT algorithm proposed by [34], but it is structurally and methodically different. The sequential logic of base algorithm (HEFT [34]) is given in 1 for a better understanding.

There are a set of tasks available, which have to be assigned processors such that the aggregated time taken by all the tasks for execution is minimum (Earliest

Finish Time). To elaborate on this part, each and every task is assigned a VM (Processor) such that the execution time of that particular VM is least (Earliest Finish Time). The process of scheduling is achieved through parameter called as rankup value which is calculated for each and every node based on the task execution time and data transfer time between the current node and succeeding node. The rank up value calculation is done from the bottom of the workflow and proceeds towards up, as mentioned in equation 4 [34].

$$Rankup(n_i) = w_i + max(c_{i,j} + Rankup(n_j)) \ \ where n_j \in succ(n_i) \qquad (4)$$

Based on the decreasing rankup value, a sequence of tasks is generated. This sequence of task is the crux and processors are assigned to each task in this sequence such that each task executes in minimum time (EFT). Thus, the overall time taken for completion of this sequence of task is called as makespan, which is the final value that is calculated 1.

Algorithm 1 Sequential Heterogeneous Earliest Finish Time [34]

```
Computation cost of tasks (considered as nodes)
Communication cost between the tasks (considered as edges) is obtained
Calculate the rankup value for all tasks starting from the last value
Sort the tasks based on rankup value
while All the unscheduled tasks (T_i) are not scheduled do
    for each processor available in the set P_j do
        Calculate EFT for T_i and P_j
    end for
Assign task such that it is EFT of T_i is minimum
end while
```

Based on the logical layout as given in algorithm 1 [34], a ENPS based algorithm is designed, which is elaborated as follows. As mentioned the ENPS based algorithm is structurally and operationally different.

There are primarily two sequential steps that are involved in scheduling a workflow.

- Creating a sequence of tasks before scheduling
- Scheduling and calculating the makespan of the actual schedule generated.

The components of the method is as given in Figure 2.

The first part of the process involves sequence calculation and the second part is mapping the sequences generated. The first component of the sequence calculation, which is strictly based on ENPS and consists of two independent membrane systems. These independent membrane systems are executed in sequence as in Figure 2. A membrane generator is developed, which is used to automatically

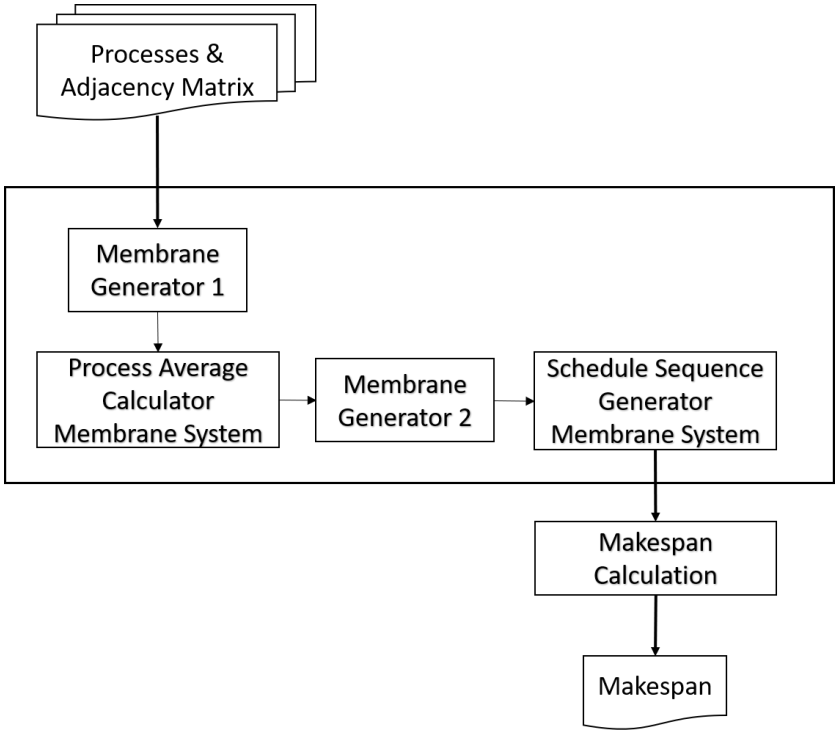

Fig. 2: P System based Workflow Model

generate membrane system as part of the solution. The membrane systems for sequence calculation is considered and elaborated.

$$\underline{Sub-problem\,1\;-\;M_1}$$

$$a_{1,1}[value],\ \ldots\ ,a_{n,1}[value]\,,a_{n+1,1}[value]$$
$$\ldots$$
$$\ldots$$
$$z_{1,1}[value],\ \ldots\ ,z_{n,1}[value],z_{n+1,1}[value]enz_{1,1}[value]$$

$$Pr_{1,1}:\ (a_{1,1}+\ldots+a_{n,1})/n\ \ \langle enz_{1,1}\rightarrow\rangle\ \ 1\mid a_{n+1,1}$$
$$\ldots$$
$$Pr_{n,1}:\ (z_{1,1}+\ldots+z_{n,1})/n\ \ \langle enz_{1,1}\rightarrow\rangle\ \ 1\mid z_{n+1,1}$$

Fig. 3: Membrane System for Process Average Calculator

Membrane system 1 (Process Average Calculator):

The first membrane system consists of a single membrane which is designed to calculate the average of the processes available. This average calculated is sent to the next membrane system 2 where it is used for finding the values based on which the sequence is scheduled. Here, the basic property of ENPS of passing values to multiple variables is used. The second membrane system is the most important and complex, it involves several programs and is elaborated as follows.

Membrane System 2 (Schedule Sequence Generator):

There are several steps for calculating the scheduling sequence. This membrane system is a single membrane system and for any number of tasks the structure remains the same. Figure 4 and Figure 5 represent the upper portion and lower portion of a single membrane respectively. The first step is to calculate the rank-up function recursively but there is no provision to perform recursion using ENPS. Thus, the whole process is done without using recursion. The whole process of execution involves calculating the values in the reverse order of the nodes present i.e. the adjacency matrix is accessed in reverse order from the last row. The enzyme n controls the maximum value operation which involves important programs. Maximum value calculation is a three step process.

There are a total of $4n^2 + 6n$ programs where, n refers to number of tasks considered. There are a total of $3n$ cycles (steps) used for this process. The total number of enzymes are $n^2 + 2n$. The sequential equivalent requires implementation as recursion, which cannot be directly used in this case. Rather, in this study the positive use of adjacency matrix is considered and the recursion is realised. The execution starts by activating the last enzyme and subsequently they execute in reverse order. Initially, enzyme number $2n$ is activated. This activates n programs which do pre-processing, related to finding maximum value among the children of current node. After pre-processing, the next step involves calculating the actual maximum value. This process involves n programs with the resultant being passed on to the next programs. The next most important steps, involves $2n$ programs, but it is also a one-step cycle. Two objectives are simultaneously achieved, one is updating the adjacency matrix values and the next is getting the final values of each task. These set of tasks values are calculated at the end of execution of all programs ($k_{1,1}$ to k_{n-1} and $w_{n,1}$) as the final set of values required.

$\underline{Sub-problem\,2\ -\ M_1(Part\,1)}$

$$a_{1,1}[value],\ \ldots\ ,a_{n,1}[value]$$
$$\ldots$$
$$\ldots$$
$$z_{1,1}[value],\ \ldots\ ,z_{n,1}[value]$$
$$k_{1,1}[value],\ \ldots\ ,k_{n,1}[value],w_{1,1}[value],\ \ldots\ ,w_{n,1}[value]$$
$$min_{1,1}[value],max_{1,1}[value],fin_{1,1}[value],enz_{1,1}[value],\ \ldots\ ,enz_{2n,1}[value]$$
$$e1_{1,1}[value],\ \ldots\ ,e1_{n,1}[value]$$
$$\ldots$$
$$\ldots$$
$$en_{1,1}[0],\ \ldots\ ,en_{n,1}[0]$$

$$Pr_{1,1}:\ 2\times(max_{1,1}-a_{1,1})\ \ \langle enz_{2,1}\rightarrow\rangle\ \ 1\mid e1_{2,1}+1\mid k_{1,1}$$
$$\ldots$$
$$Pr_{n,1}:\ 2\times(max_{1,1}-a_{n,1})\ \ \langle enz_{2,1}\rightarrow\rangle\ \ 1\mid enz_{n,1}+1\mid k_{1,1}$$
$$Pr_{n+1,1}:\ a_{1,1}\times enz_{2,1}\ \ \langle enz_{2,1}\rightarrow\rangle\ \ 1\mid enz_{2,1}$$
$$Pr_{n+2,1}:\ max_{1,1}\ \ \langle enz_{2,1}\rightarrow\rangle\ \ 1\mid max_{1,1}$$
$$Pr_{n+3,1}:\ e1_{2,1}*0+e1_{3,1}*0+e1_{4,1}*0-e1_{1,1}\ \ \langle enz_{2,1}\rightarrow\rangle\ \ 1\mid k_{1,1}$$
$$Pr_{n+4,1}:\ e1_{1,1}*0-e1_{2,1}+\ldots+e1_{4,1}*0\ \ \langle enz_{2,1}\rightarrow\rangle\ \ 1\mid k_{1,1}$$
$$\ldots$$
$$Pr_{2n+2,1}:\ e1_{1,1}*0+e1_{2,1}*0+\ldots-e1_{n,1}\ \ \langle enz_{2,1}\rightarrow\rangle\ \ 1\mid k_{1,1}$$
$$Pr_{2n+3,1}:\ a_{1,1}*(e1_{1,1}+\ldots+e1_{n,1})\ \ \langle enz_{2,1}\rightarrow\rangle\ \ 1\mid e1_{1,1}+\ldots+1\mid e1_{n,1}$$
$$Pr_{2n+4,1}:\ a_{1,1}+20000\ \ \langle e1_{1,1}\rightarrow\rangle\ \ 1\mid enz_{1,1}$$
$$Pr_{2n+5,1}:\ (max_{1,1}-k_{1,1}+w_{1,1})*fin_{1,1}/fin_{1,1}\ \ \langle e1_{1,1}\rightarrow\rangle\ \ 1\mid 1\mid k_{1,1}$$
$$Pr_{2n+6,1}:\ (max_{1,1}-k_{1,1}+w_{1,1}+a_{1,1})*fin_{1,1}/fin_{1,1}\ \ \langle e1_{1,1}\rightarrow\rangle\ \ 1\mid a_{1,1}$$
$$Pr_{2n+7,1}:\ (max_{1,1}-k_{2,1}+w_{1,1})*b_{1,1}/b_{1,1}\ \ \langle e1_{1,1}\rightarrow\rangle\ \ 1\mid 1\mid k_{1,1}$$
$$Pr_{2n+8,1}:\ (max_{1,1}-k_{1,1}+w_{1,1}+b_{1,1})*b_{1,1}/b_{1,1}\ \ \langle e1_{1,1}\rightarrow\rangle\ \ 1\mid b_{1,1}$$

$$\ldots$$
$$Pr_{4n+3,1}:\ (max_{1,1}-k_{1,1}+w_{1,1})*z_{1,1}/z_{1,1}\ \ \langle e1_{1,1}\rightarrow\rangle\ \ 1\mid k_{1,1}$$
$$Pr_{4n+4,1}:\ (max_{1,1}-k_{1,1}+w_{1,1}+z_{1,1})*z_{1,1}/z_{1,1}\ \ \langle e1_{1,1}\rightarrow\rangle\ \ 1\mid z_{1,1}$$
$$Pr_{4n+5,1}:\ (max_{1,1}\ \ \langle e1_{1,1}\rightarrow\rangle\ \ 1\mid max_{1,1}$$
$$\ldots$$
$$\ldots$$
$$Pr_{4n^2+2n-4,1}:\ 2\times(max_{1,1}-z_{1,1})\ \ \langle enz_{n,1}\rightarrow\rangle\ \ 1\mid en_{1,1}+1\mid k_{n,1}$$
$$\ldots$$
$$Pr_{4n^2+3n-5,1}:\ 2\times(max_{1,1}-z_{n,1})\ \ \langle enz_{n,1}\rightarrow\rangle\ \ 1\mid en_{n,1}+1\mid k_{n,1}$$

Fig. 4: Membrane System for Sub-problem 2 (Schedule Sequence Generator) - Part 1

$\underline{Sub-problem\,2\ -\ M_1(Part\,2)}$

$$Pr_{4n^2+3n-4,1}:\ a_{1,1}\times enz_{n,1}\ \ \langle enz_{n,1}\rightarrow\rangle\ \ 1\mid enz_{n,1}+1\mid k_{n,1}$$
$$Pr_{4n^+3n-3,1}:\ max_{1,1}\ \ \langle enz_{n,1}\rightarrow\rangle\ \ 1\mid max_{1,1}$$
$$\mathrm{Pr}_{4n^+3n-2,1}:\ en_{2,1}*0+en_{3,1}*0+\ldots+en_{n,1}*0-\ en_{1,1}\ \ \langle en_{1,1}\rightarrow\rangle\ \ 1\mid k_{n,1}$$
$$Pr_{4n^+3n-1,1}:\ e1_{1,1}*0-e1_{2,1}+\ldots+e1_{n,1}*0\ \ \langle en_{2,1}\rightarrow\rangle\ \ 1\mid k_{1,1}$$
$$\ldots$$
$$Pr_{4n^2+4n-3,1}:\ en_{1,1}*0+en_{2,1}*0+\ldots-en_{n,1}\ \langle en_{n,1}\rightarrow\rangle\ \ 1\mid k_{1,1}$$
$$Pr_{4n^2+4n-2,1}:\ a_{1,1}*(en_{1,1}+\ldots+\ en_{n,1})\ \ \langle en_{1,1}\rightarrow\rangle\ \ 1\mid en_{1,1}+\ldots+1\mid en_{n,1}$$
$$Pr_{4n^2+4n-1,1}:\ a_{1,1}+20000\ \ \langle en_{1,1}\rightarrow\rangle\ \ 1\mid enz_{n,1}$$
$$Pr_{4n^2+4n,1}:\ (max_{1,1}-k_{n,1}+w_{n,1})\ *a_{n,1}/a_{n,1}\ \ \langle enz_{n-1,1}\rightarrow\rangle\ \ 1\mid 1\mid k_{n,1}$$
$$Pr_{4n^2+4n+1,1}:\ (max_{1,1}-k_{n,1}+w_{n,1}+\ a_{n,1})*a_{n,1}/a_{n,1}\ \ \langle enz_{n-1,1}\rightarrow\rangle\ \ 1\mid a_{n,1}$$
$$Pr_{4n^2+4n+2,1}:\ (max_{1,1}-k_{n,1}+\ w_{n,1})*b_{n,1}/b_{n,1}\ \ \langle enz_{n-1,1}\rightarrow\rangle\ \ 1\mid 1\mid k_{n,1}$$
$$Pr_{4n^2+4n+3,1}:\ (max_{1,1}-k_{n,1}+w_{n,1}+\ b_{n,1})*b_{n,1}/b_{n,1}\ \ \langle enz_{n-1,1}\rightarrow\rangle\ \ 1\mid b_{n,1}$$
$$\ldots$$
$$Pr_{4n^2+6n-2,1}:\ (max_{1,1}-k_{n,1}+\ w_{n,1})*z_{n,1}/z_{n,1}\ \ \langle enz_{n-1,1}\rightarrow\rangle\ \ 1\mid k_{1,1}$$
$$Pr_{4n^2+6n-1,1}:\ (max_{1,1}-k_{n,1}+w_{n,1}+\ b_{n,1})*z_{n,1}/z_{n,1}\ \ \langle enz_{n-1,1}\rightarrow\rangle\ \ 1\mid b_{n,1}$$
$$Pr_{4n^2+6n,1}:\ (max_{1,1}\ \ \langle enz_{n-1,1}\rightarrow\rangle\ \ 1\mid max_{1,1}$$

Fig. 5: Membrane System for Sub-problem 2 (Schedule Sequence Generator) - Part 2

After generating the schedule sequence, as the final step, the schedule sequence is mapped and the final makespan is obtained. The final makespan is the parameter that is considered in the work, based on which further comparison with other algorithms are done.

5 Results and Analysis

For implementing membrane-based model ENPS-based simulator (GPUPeP) [29] is used as the base. A membrane generator is developed for generating the

the main membrane-system for workflow sequence calculation. This membrane generator is developed using Python 3.0 and can generate the membrane system for any number of tasks. These membrane systems are then executed over ENPS-based simulator, GPUPeP and the results are passed to the trailing steps (Membranes).

After the sequence of execution is calculated, the makespan is obtained. The proposed method has been compared with two other standard methods, namely, Min-min and Max-min, for scheduling workflow based task. These methods have been chosen to compare, because they are accepted as standard methods.

Five type of workflows are considered and compared; Five cases each for, five tasks, ten tasks, 15 tasks, 20 tasks and 25 tasks. For each workflow type, five sample workflows of different structures are considered. Figure 6 shows results of five different workflows with different workflow structures and different virtual machine configurations; considered with five tasks. The results show that the proposed model is better than other methods on three occasions and it gives same makespan value for the considered cases. This is because the input considered is not so complex and it doesn't involve much parallelism (task parallelism). Thus, all the algorithms give similar sequence for few cases.

For next comparison, five different workflows of size 10 are considered. As from the Figure 7 it is visible that, in all the cases the proposed algorithm tends to be better. Similarly when 15 tasks are considered the proposed algorithm fares

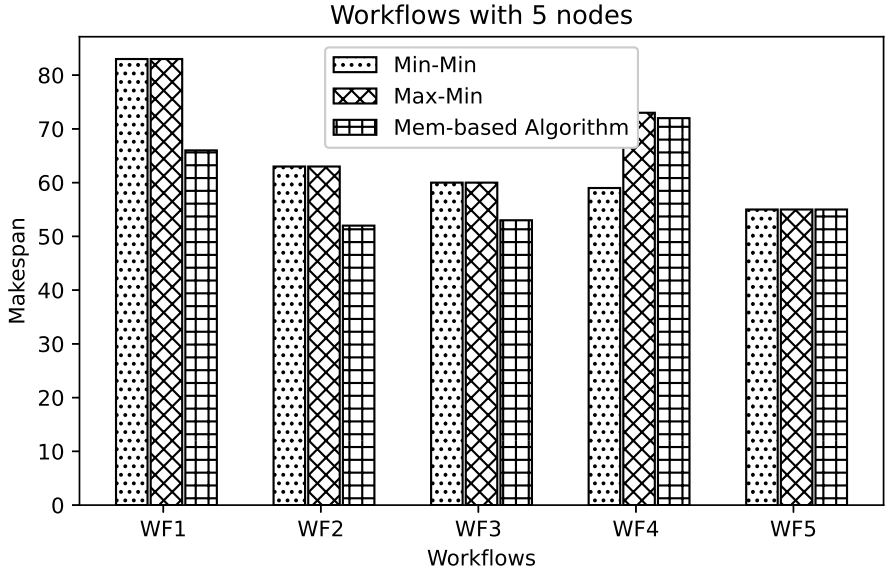

Fig. 6: Workflow Scheduling for 5 Tasks

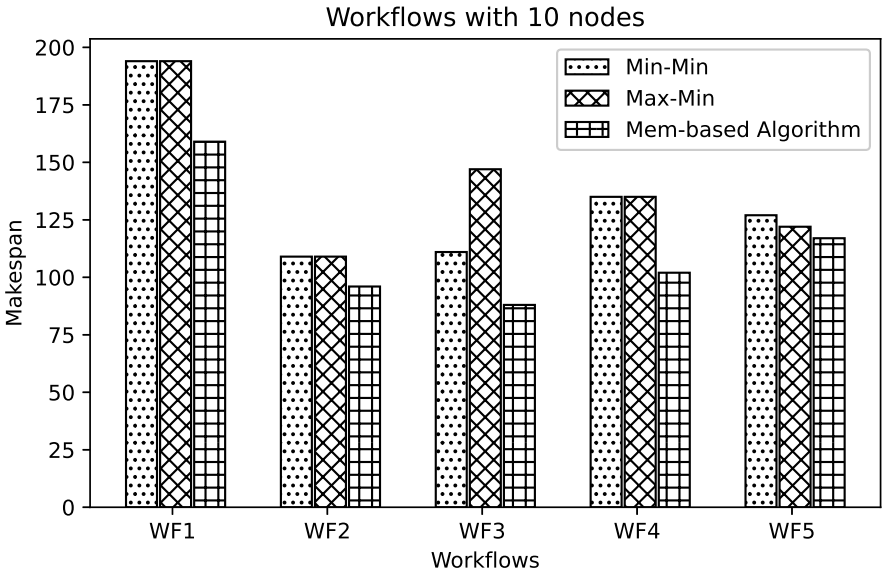

Fig. 7: Workflow Scheduling for 10 Tasks

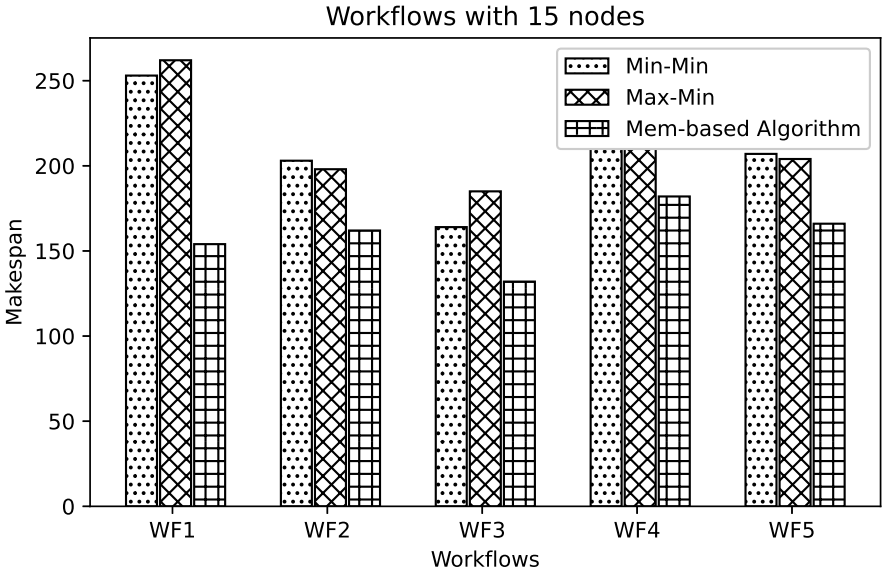

Fig. 8: Workflow Scheduling for 15 Tasks

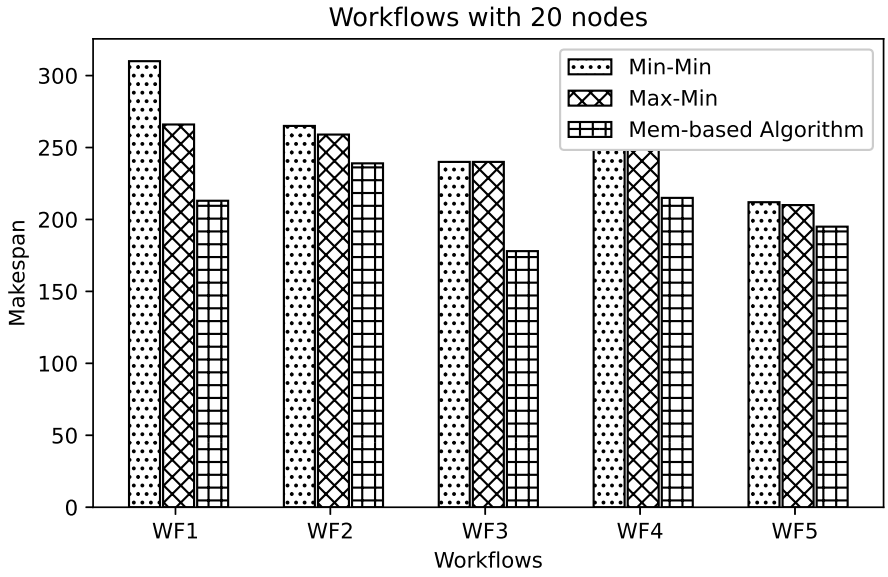

Fig. 9: Workflow Scheduling for 20 Tasks

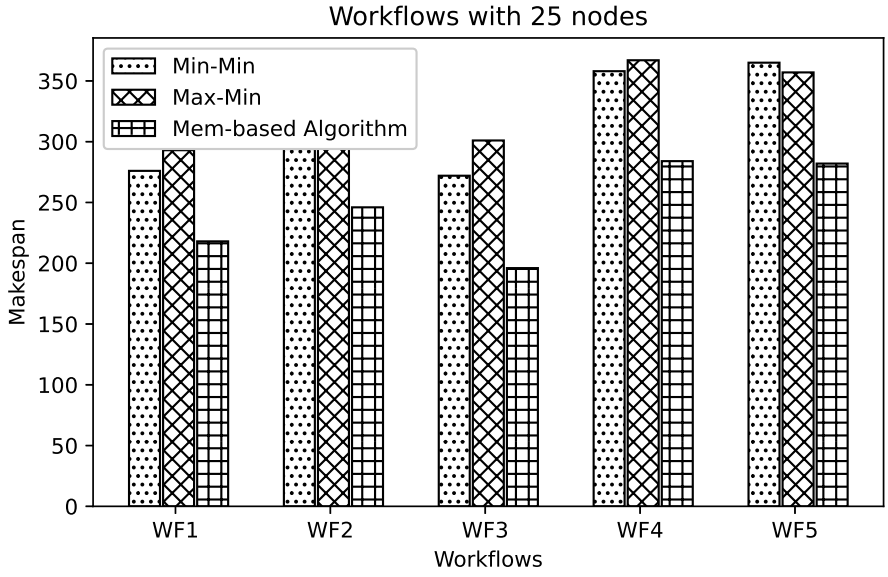

Fig. 10: Workflow Scheduling for 25 Tasks

better than the others, as in Figure 8. Thus, few of the cases are significantly better than the others and similar is the case for 20 tasks (Figure 9).
When 25 tasks are considered to be scheduled for a workflow there are many cases for which the proposed method is better than other algorithms as in Figure 10. Based on the results it can be inferred that the algorithm performs better when the number of tasks increase. Though the improvement is not only based on the number of tasks, it is one of the factor which judges the effectiveness of the algorithm. The backbone of the work is the membrane model, which theoretically has very less time complexity.
To summarize, a ENPS-based Workflow scheduling algorithm for cloud has been developed whichi is inturn inspired by HEFT. The ENPS-based algorithm has been designed and implemented. The implemented algorithm is compared with other algorithms for makespan and it is found that the proposed algorithm performs better for makespan. It is one of the first membrane based solutions for workflow scheduling in cloud, which strictly follows the rules of the membrane computing paradigm.

6 Conclusion

The proposed algorithm is for scheduling workflows in cloud. The membrane-based algorithm is particularly designed based on ENPS and an efficient heuristic-based solution. Thus, the developed algorithm is ENPS structure-based heuristic algorithm for workflow scheduling in cloud. The proposed multi-membrane system module is designed to be parallel, based on the natural property of ENPS. As part of the work, a workflow membrane generator is also created to automatically create the membranes that represent the method. The membranes generated are tested and the results obtained are compared with other standard workflow scheduling methods. Based on the results, the proposed algorithm is found to be better than two standard methods.

Acknowledgement

This publication is an outcome of the R & D work undertaken under the Visvesvaraya Ph.D. Scheme of Ministry of Electronics & Information Technology, Government of India, being implemented by Digital India Corporation.

References

[1] T. Ahmed, R. Verma, M. Bakshi, and A. Srivastava. Membrane Computing Inspired Approach for Executing Scientific Workflow in the Cloud. In *International Conference on Membrane Computing*, pages 51–65. Springer, 2014.

[2] R. Buyya, C. S. Yeo, S. Venugopal, J. Broberg, and I. Brandic. Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering

computing as the 5th utility. *Future Generation computer Systems*, 25(6): 599–616, 2009.

[3] R. N. Calheiros and R. Buyya. Meeting deadlines of scientific workflows in public clouds with tasks replication. *IEEE Transactions on Parallel and Distributed Systems*, 25(7):1787–1796, 2013.

[4] H. Chen, X. Zhu, D. Qiu, L. Liu, and Z. Du. Scheduling for workflows with security-sensitive intermediate data by selective tasks duplication in clouds. *IEEE Transactions on Parallel and distributed Systems*, 28(9):2674–2688, 2017.

[5] A. G. Florea and C. Buiu. Modelling multi-robot interactions using a generic controller based on numerical P Systems and ROS. In *2017 9th International Conference on Electronics, Computers and Artificial Intelligence (ECAI)*, pages 1–6. IEEE, 2017.

[6] L. Huang and N. Wang. An optimization algorithm inspired by membrane computing. In *International Conference on Natural Computation*, pages 49–52. Springer, 2006.

[7] L. Kari and G. Rozenberg. The many facets of natural computing. *Communications of the ACM*, 51(10):72–83, 2008.

[8] M. Kumar and S. Sharma. PSO-COGENT: Cost and energy efficient scheduling in cloud environment with deadline constraint. *Sustainable Computing: Informatics and Systems*, 19:147–164, 2018.

[9] A. Leporati, A. E. Porreca, C. Zandron, and G. Mauri. Improving universality results on parallel Enzymatic Numerical P Systems. *Proceedings of the Eleventh Brainstorming Week on Membrane Computing, 177-200. Sevilla, ETS de Ingeniería Informática, 4-8 de Febrero, 2013*, 2013.

[10] D. Llorente Rivera and M. Á. Gutiérrez Naranjo. The Pole Balancing Problem with Enzymatic Numerical P Systems. In *Proceedings of the Thirteenth Brainstorming Week on Membrane Computing, 195-206. Sevilla, ETS de Ingeniería Informática, 2-6 de Febrero, 2015*. Fénix Editora, 2015.

[11] N. Mansouri, B. M. H. Zade, and M. M. Javidi. Hybrid Task Scheduling Strategy for Cloud Computing by Modified Particle Swarm Optimization and Fuzzy Theory. *Computers & Industrial Engineering*, 2019.

[12] M. Masdari, S. ValiKardan, Z. Shahi, and S. I. Azar. Towards workflow scheduling in cloud computing: A comprehensive analysis. *Journal of Network and Computer Applications*, 66:64–82, 2016. ISSN 1084-8045. https://doi.org/https://doi.org/10.1016/j.jnca.2016.01.018.

[13] P. Mell and T. Grance. The NIST definition of Cloud Computing. 2011.

[14] D. Nasonov, N. Butakov, M. Balakhontseva, K. Knyazkov, and A. V. Boukhanovsky. Hybrid evolutionary workflow scheduling algorithm for dy-

namic heterogeneous distributed computational environment. In *International Joint Conference SOCO'14-CISIS'14-ICEUTE'14*, pages 83–92. Springer, 2014.

[15] L. Pan, Z. Zhang, T. Wu, and J. Xu. Numerical P Systems with Production Thresholds. *Theoretical Computer Science*, 673:30–41, 2017. ISSN 03043975. `https://doi.org/10.1016/j.tcs.2017.02.026`. URL `http://dx.doi.org/10.1016/j.tcs.2017.02.026`.

[16] S. Pandey, L. Wu, S. M. Guru, and R. Buyya. A particle swarm optimization-based heuristic for scheduling workflow applications in cloud computing environments. In *2010 24th IEEE international conference on advanced information networking and applications*, pages 400–407. IEEE, 2010.

[17] G. Păun. Computing with membranes. *Journal of Computer and System Sciences*, 61(1):108–143, 2000.

[18] G. Păun and R. Păun. Membrane computing and economics: Numerical P Systems. *Fundamenta Informaticae*, 73(1, 2):213–227, 2006.

[19] G. Paun, G. Rozenberg, and A. Salomaa. *The Oxford Handbook of Membrane Computing*. IOS Press, 2010. ISBN 9780199556670.

[20] A. B. Pavel and C. Buiu. Using Enzymatic Numerical P Systems for modeling mobile robot controllers. *Natural Computing*, 11(3):387–393, 2012.

[21] A. B. Pavel, O. Arsene, and C. Buiu. Enzymatic Numerical P Systems-A new class of membrane computing Systems. In *Bio-Inspired Computing: Theories and Applications (BIC-TA), 2010 IEEE Fifth International Conference on*, pages 1331–1336. IEEE, 2010.

[22] H. Peng, J. Shao, B. Li, J. Wang, M. d. J. Pérez Jiménez, Y. Jiang, and Y. Yang. Image thresholding with cell-like P Systems. *Proceedings of the Tenth Brainstorming Week on Membrane Computing,(2) 75-88. Sevilla, ETS de Ingeniería Informática, 30 de Enero-3 de Febrero, 2012,*, 2012.

[23] H. Peng, J. Wang, M. J. Pérez-Jiménez, and P. Shi. A novel image thresholding method based on membrane computing and fuzzy entropy. *Journal of Intelligent & Fuzzy Systems*, 24(2):229–237, 2013.

[24] H. Peng, Y. Jiang, J. Wang, and M. Pérez-Jiménez. Membrane clustering algorithm with hybrid evolutionary mechanisms. *Journal Software*, 26(5): 1001–1012, 2015.

[25] H. Peng, P. Shi, J. Wang, A. Riscos-Núñez, and M. J. Pérez-Jiménez. Multiobjective fuzzy clustering approach based on tissue-like membrane Systems. *Knowledge-Based Systems*, 125:74–82, 2017.

[26] I. Pérez-Hurtado, M. J. Pérez-Jiménez, G. Zhang, and D. Orellana-Martín. Robot path planning using rapidly-exploring random trees: A membrane

computing approach. In *2018 7th International Conference on Computers Communications and Control (ICCCC)*, pages 37–46. IEEE, 2018.

[27] I. Pérez Hurtado de Mendoza, M. Á. Martínez del Amor, G. Zhang, F. Neri, and M. d. J. Pérez Jiménez. A membrane parallel rapidly-exploring random tree algorithm for robotic motion planning. *Integrated Computer-Aided Engineering, 27 (2), 121-138.*, 2020.

[28] S. Raghavan and K. Chandrasekaran. Membrane-based models for service selection in cloud. *Information Sciences*, 558:103–123, 2021.

[29] S. Raghavan, S. S. Rai, M. Rohit, and K. Chandrasekaran. GPUPeP: Parallel Enzymatic Numerical P System simulator with a Python-based interface. *Biosystems*, 196:104186, 2020. ISSN 0303-2647.

[30] M. A. Rodriguez and R. Buyya. Deadline based resource provisioningand scheduling algorithm for scientific workflows on clouds. *IEEE transactions on cloud computing*, 2(2):222–235, 2014.

[31] Z. Shang, S. Verlan, G. Zhang, and H. Rong. Fpga implementation of numerical p systems. *International Journal of Unconventional Computing*, 16, 2021.

[32] G. Singh, K. Deep, and A. K. Nagar. Cell-like P-Systems based on rules of Particle Swarm Optimization. *Applied Mathematics and Computation*, 246: 546–560, 2014.

[33] S. Su, J. Li, Q. Huang, X. Huang, K. Shuang, and J. Wang. Cost-efficient task scheduling for executing large programs in the cloud. *Parallel Computing*, 39(4-5):177–188, 2013.

[34] H. Topcuoglu, S. Hariri, and M.-y. Wu. Performance-effective and low-complexity task scheduling for heterogeneous computing. *IEEE transactions on parallel and distributed Systems*, 13(3):260–274, 2002.

[35] H. Wang, H. Peng, J. Shao, and T. Wang. A thresholding method based on P Systems for image segmentation. *ICIC Express Letters*, 6(1):221–227, 2012.

[36] X. Wang, G. Zhang, J. Zhao, H. Rong, F. Ipate, and R. Lefticaru. A modified membrane-inspired algorithm based on particle swarm optimization for mobile robot path planning. *International Journal of Computers Communications & Control*, 10(5):732–745, 2015.

[37] X. Wang, G. Zhang, X. Gou, P. Paul, F. Neri, H. Rong, Q. Yang, and H. Zhang. Multi-behaviors coordination controller design with enzymatic numerical p systems for robots. *Integrated Computer-Aided Engineering*, 28 (2):119–140, 2021.

[38] C. Q. Wu, X. Lin, D. Yu, W. Xu, and L. Li. End-to-end delay minimization for scientific workflows in clouds under budget constraint. *IEEE Transactions on Cloud Computing*, 3(2):169–181, 2014.

[39] Q. Wu, F. Ishikawa, Q. Zhu, Y. Xia, and J. Wen. Deadline-constrained cost optimization approaches for workflow scheduling in clouds. *IEEE Transactions on Parallel and Distributed Systems*, 28(12):3401–3412, 2017.

[40] F. Zhang, J. Cao, K. Hwang, K. Li, and S. U. Khan. Adaptive workflow scheduling on cloud computing platforms with iterativeordinal optimization. *IEEE Transactions on Cloud Computing*, 3(2):156–168, 2014.

[41] G. Zhang, M. Gheorghe, L. Pan, and M. J. Perez-Jimenez. Evolutionary membrane computing: a comprehensive survey and new results. *Information Sciences*, 279:528–551, 2014.

[42] G. Zhang, H. Rong, F. Neri, and M. J. Pérez-Jiménez. An optimization spiking neural p system for approximately solving combinatorial optimization problems. *International Journal of Neural Systems*, 24(05):1440006, 2014.

[43] G. Zhang, M. J. Pérez-Jiménez, and M. Gheorghe. Robot Control with Membrane Systems. In *Real-life Applications with Membrane Computing*, pages 213–258. Springer, 2017.

[44] Z. Zhang and L. Pan. Numerical P Systems with Thresholds. *International Journal of Computers Communications & Control*, 11(2):292–304, 2016.

[45] Z. Zhang, T. Wu, A. Păun, and L. Pan. Numerical P Systems with migrating variables. *Theoretical Computer Science*, 641:85–108, 2016.

[46] M. Zhu, Q. Yang, J. Dong, G. Zhang, X. Gou, H. Rong, P. Paul, and F. Neri. An adaptive optimization spiking neural p system for binary problems. *International Journal of Neural Systems*, 31(01):2050054, 2021.

[47] Z. Zhu, G. Zhang, M. Li, and X. Liu. Evolutionary multi-objective workflow scheduling in cloud. *IEEE Transactions on parallel and distributed Systems*, 27(5):1344–1357, 2015.