Scheduling Jobs in Multi-Grid Environment

A Modeling Approach based on Petri Nets

Albana Roci
Electrical and Computer Engineering
University of Stavanger
Stavanger, Norway
e-mail: Albana.Roci@uis.no

Reggie Davidrajuh
Electrical and Computer Engineering
University of Stavanger
Stavanger, Norway
e-mail: Reggie.Davidrajuh@uis.no

***Abstract*— Scheduling is a process that allocates resources to the competing tasks. The distribution of the resources to the various tasks forms a job. The aggregation of the tasks within the job, determine the optimal utilization of the resources and minimal completion time and costs. This paper demonstrates the scheduling problem in the grid applications, considering single-grid and multi-grid environments. Petri Net is used for modeling the system, and the models are implemented with the GPenSIM tool. Also, this paper presents an algorithm for scheduling in multi-grid, in which the modules give the right to each other to use one others resources without causing delays. The novelty of this algorithm is that while it is easy to implement, it is also efficient as it minimizes the total processing time of the jobs.**

Keywords: Multi-Grid Scheduling, Petri Net, GPenSIM.

I. INTRODUCTION

Grid computing facilitates several resources that are geographically distributed in different parts of the world to operate as a single integrated system. This arrangement enables computers to collaborate and achieve at least same or better processing time compared to processing on a standalone computer. Grid computing also enables the possibility of using specialized processing that is not available on the local host [1, 2].

A grid job is organized as a flow of different activities, named tasks or sub-jobs. These tasks can be executed in parallel or sequence. If the tasks have dependencies between them, meaning one task needs the output of one or more of the other tasks, then these tasks are to be executed in sequence. Otherwise, the execution can be done in parallel. To have an efficient performance, it is important to obtain good coordination between the tasks and the delivery of resources to perform the tasks. To obtain good coordination, grid systems employee 'job schedulers' [3].

Grid jobs are usually composed of a large number of tasks. Because of this, a single grid may not be able to address all the requests coming from these tasks. In this case, the grid should be interconnected with some other grids to enable the executions of the tasks in a timely manner; this is known as the multi-grid [4]. There are several benefits in the use of the multi-grid, but the main reason is that there will not be any deadlocked job [4].

In this paper: Section-II presents a short literature study on the related works. Section-III presents a new algorithm for scheduling jobs in a multi-grid environment. Section-IV presents a simulation study as a proof-of-concept. The results of the simulation study are analyzed in Section-V.

II. RELATED WORK

Literature review reveals some works on modeling and scheduling of jobs in grid computing. There exists some general purpose modeling software that can be used for modeling grid applications too (such as Simscript [5], network simulator OMNet++ [6], and JGPSS [7]). Simulation models in Simscript are programmed using FORTRAN language, whereas C++ is used in OMNet++. JGPSS is a Java implementation of the General Purpose Simulation Software (GPSS) [8]. Thus, Java is used for programming the models.

Focusing on the dedicated simulators for the distributed computing environment, such as peer-to-peer (P2P) and grid, literature study reveals some simulators developed at the universities. The Bricks simulator - developed at the Tokyo Institute of Technology - is a simulation and benchmarking environment for distributed computing [9]. The MicroGrid emulator is developed at the University of California at San Diego [10]. The MicroGrid is built on top of an earlier toolkit known as the Globus toolkit. GridSim toolkit is for modeling and simulation of distributed resource allocation and scheduling in the grid and P2P environment [1]. GridSim was developed at the Monash University in Australia. All these three toolkits are Java-based meaning the modeler need to program the modules for simulation using Java programming language.

The Simgrid toolkit also developed at the University of California at San Diego (UCSD) is a C language based toolkit [11]. The Simgrid toolkit is also for the simulation of application scheduling, and its supports modeling and simulation of grid environment where a large number of tasks and resources are involved.

0-7695-6405-4/18/$31.00 ©2018 IEEE
DOI 10.1109/UKSim.2018.00024

In this paper, we use a new MATLAB based simulation software known as General Purpose Petri Net Simulator (GPenSIM) [12, 13]. As the name declares, GPenSIM is based on Petri net, and it is designed to work together with the other MATLAB toolboxes such as Control Systems Toolbox and Fuzzy Logic Toolbox. Thus, hybrid Petri net models can be developed by combining the GPenSIM with the other MATLAB toolboxes. Also, the tools like Advanced Statistical toolbox can be used to analyze the Petri net models [13, 14].

This paper uses Petri Net for modelling the system, as Petri Net has significant performance in the modeling of concurrent systems. Plenty of research is done in modeling grid computing using Petri Nets, but in many of these cases, the Petri net models become huge due to a large number of resources [2]. Therefore, the main purpose of this paper is to propose a simple yet efficient algorithm for scheduling and a simplified way of modeling grid with Petri Nets.

III. PROPOSED ALGORITHM

Each grid application has a job scheduler or broker. The broker coordinates the local tasks request with the local resources. Also, the broker can accept and send requests from/to other external brokers. As [2] suggest, the environments for scheduling of resources are classified into two categories:

1. Single Grid: scheduling of local resources to local tasks by the (local) broker, and
2. Multi-Grid: collaboration between brokers of different modules that schedule the external and local requests (Multi-Grid).

It is important to mention that this paper assumes that the resources are heterogeneous, this means that they are specialized to perform different types of jobs.

A. Single-Grid

The paper [2] introduces a new approach for modelling grid application with Petri Nets, and the models are implemented in GPenSIM. The scope of [2] is limited to the single-grid environment. In this paper, there can be several grid modules equipped with local resources, and these resources may be available to perform tasks from external grids (e.g., due to lack of local jobs). When executing a task, the broker does not give priority to local jobs over the external jobs. Thus, local and external jobs are assumed to have the same level of preference, which is not a realistic situation.

B. Multi-Grid

The grid systems have a limited number of resources. Therefore, some of them may not be able to fulfill all the requests that come from the local tasks. To solve this problem, the broker directs the local task request to an external resource in another grid. Thus, the local resources of one grid can be required by several local and external tasks. In this case, the local tasks possess higher priority to use the local resources than the external tasks.

How do we allocate a local resource to an external task, when the local resource is already sought by local tasks? The following subsections present two solutions: 1) Naïve solution that is not an efficient solution, and 2) A new algorithm that is based on graph algorithms.

1) Naïve Solution

The easiest way to solve this problem is to let the local tasks use the resource first. When the local tasks are complete, and the resource is available, the external task can be allowed to use the resource. The method of keeping the external task wait until the local tasks are complete is not an efficient method because of the following reasons:

1) The external grid that requested the local resource endures long delay.
2) The local resource is not scheduled optimally. This is because the local resource may have pockets of availability between the execution of the local tasks. These pockets of free availability could have been used to execute the external tasks.

2) A new algorithm for scheduling in Multi-Grid

The algorithm proposed in this section is a new optimistic method, which addresses the problem raised above. To allow the external task to perform between the internal tasks, the broker should check for the pockets of availability between the executions of the local tasks. Also, even when a resource is available, the broker should check if the time consumed by the external task will not cause a delay in the completion time of the local jobs. The algorithm checks if permission can be given by all the local tasks that need the required resource. There are three cases:

a. If the task has been completed, it gives the permission.
b. There are some cases where the task has the opportunity to choose between two or more resources. If this task is in the execution mode and the required resource is not contributing in its execution, then it gives the permission.
c. If the resource is available, but the task has not started the execution because it is waiting for another task under execution to complete. In this case, the path that takes maximum execution time is found from this task to the ones that are under execution. The algorithm returns the time gap between the task under the execution and the start of this task. If the time gap is larger than the processing time of the external task, then it gives the permission.

Fig.1 shows the algorithm. Firstly, it checks if the local resources needed for the task are available. If the required local resources are not available, the algorithm halts and will come back after some time. If the local resources are available, then the algorithm checks whether

the task needs any external resources. If there is a need for external resources, then it will proceed with rules mention above.

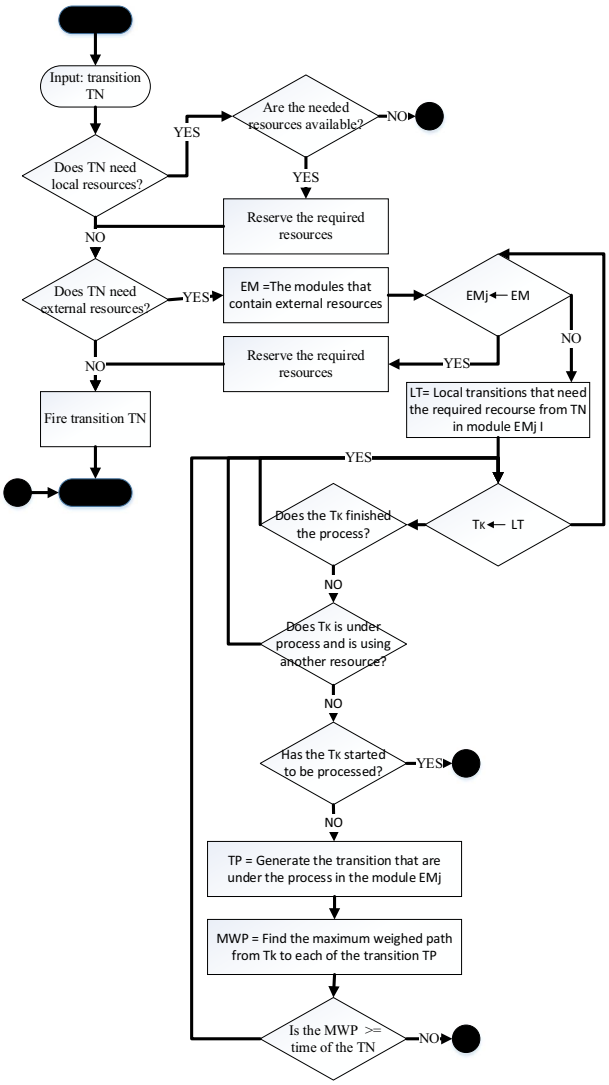

Figure 1. Flowchart explaining the algorithm

Different graph algorithms can be used for finding the maximum path between one source node and several terminal nodes. In this paper, only the Depth-First-Search (DFS) algorithm and the Directed Acyclic Graph (DAG/Topological sort) algorithm are used. To apply these algorithms, the Petri net model is transformed into a weighted and directed graph. During the transformation, the transitions are converted in nodes and the firing times of the transition as added as the weights of the corresponding arcs. When the DFS is used, the root of the tree is the task that needs the required resource. The leaves are the tasks that are currently executing. This means the weighted graph is effectively a transpose of the Petri net.

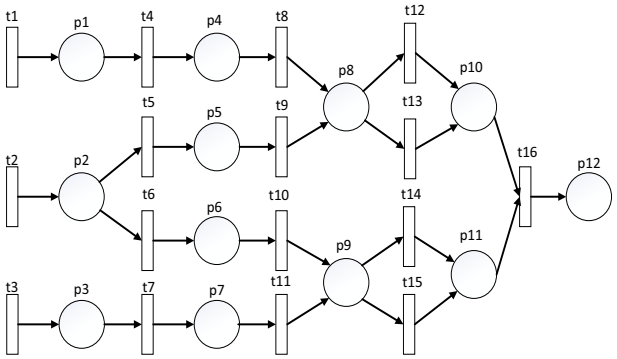

Figure 2. The Petri Net model of the first module

Starting from that task or node and going through its successors until the leaves are hit, the DFS will calculate the total weight of each path. In the other hand when DAG is used, the nodes that are under execution are considered as the sources of the graph. Starting from the source(s), it assigns the weight to each node and relaxes the weight of the node if there exists a larger value. Both DFS and DAG can be executed in ***O(E+V)***. However, the DAG takes less time because the DFS executes N times, where N is the number of the tasks that are under the execution, and the algorithm finds the path for each of these nodes to the task. The time complexity when the DFS is performing is ***O(N(E+V))***.

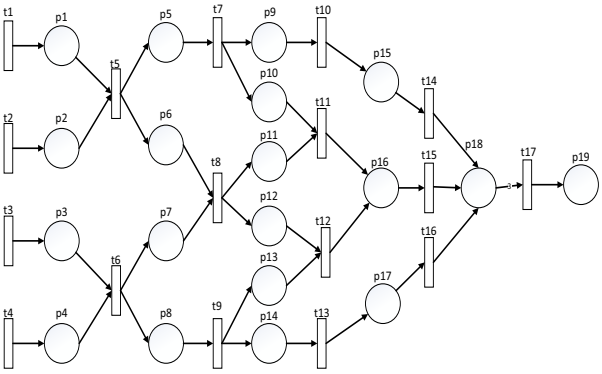

Figure 3. The Petri Net model of the second module

As a result, the algorithms output the maximum weighted path which represents the time gap between the ongoing execution and the waiting execution. This resulting time gap is compared with the processing time of the external task. If the processing time of the external task is less than or equal to the resulting time gap, then it will permit the external task.

IV. SIMULATION

The proposed algorithm was applied to solve several examples. Due to brevity, only one of these examples is given in this section.

Table I. Processing Time and Required Resources for the first module shown in Fig.2.

Task Name	Processing Time	Required Resources
T1	2	R11
T2	6	R12
T3	1	R14
T4	5	R13 & R22
T5	1	R14
T6	4	R11 & R12
T7	3	R14 & R23
T8	2	R14 & R24
T9	1	R12
T10	3	R21
T11	4	R13
T12	2	R11
T13	4	R14
T14	3	R11
T15	3	R13
T16	2	R12 or R14

Table II. Processing Time and Required Resources for the second module shown in Fig.3.

Task Name	Processing Time	Required Resources
T1	2	R22
T2	2	R21 & R24 & R14
T3	3	R23
T4	1	R21 & R24
T5	3	R21 & R11
T6	4	R24
T7	4	R21
T8	5	R22
T9	4	R23
T10	5	R23
T11	3	R14
T12	6	R22
T13	1	R24 & R12
T14	3	R24
T15	2	R23
T16	2	R13
T17	2	R22

In this example, two different modules (grids) were constructed. The first module has 16 tasks and 4 local resources while the second module has 17 tasks and 4 local resources. Fig.2 and Fig.3 show the Petri Net models of these modules. Table 1 presents the processing times and requirement of resources of the task in the modules.

To explain the running of the algorithm, let us suppose that the system clock is at 3 TU. The fifth task from the second module requires the first resource from the first module. In Fig.4, we show the directed and weighted graph of the first module in the specific moment.

At this moment, the required resource is available. However, as we mentioned before, firstly, we should check for priorities of the local tasks. The local task of the first module that can use the first resource at times 1, 6, 12 and 14. As can be seen even in Fig.4, the first resource has finished processing. Therefore, it gives the permission to the external task.

While the remaining tasks have not started yet, there two tasks that are under execution, the second and fourth tasks. From the fourth task to the sixth task, there is no path. The path from the second task to the sixth task normally weights 6 TU. As the system is in the 3 TU, therefore 3 TU remains. As the processing time of the external resource is also 3 TU, the sixth task gives the permission. The twelfth task is connected either from the second or fourth task. The weight from the second to the twelfth task is 5 TU, and the weight from the fourth to the twelfth tasks is 4 TU. The maximum weight is 5 TU, which is larger than the processing time of the fifth task of the second module. There is no path connecting the fourteenth task to the fourth task. The weight of the path from the second task to the fourteenth is 10 TU. Hence, the fourteenth task too permits the external task. At this moment, the broker delegate this external task of the second module to the first resource. There are cases where the local tasks will not give the permit the external one. In this case, the external one will continue requesting permission until it gets it.

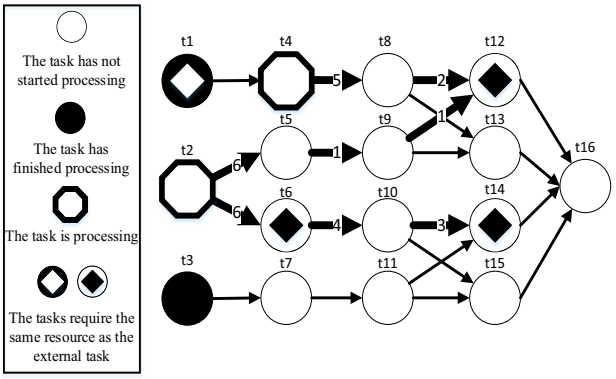

Figure 4. The graph model of the first module

A deadlock can occur if both of the modules require resources from each other at the same time. None of the modules can give the permission because they are waiting for the execution of their local tasks. To address this problem, we decided to give the permission to the task that requires less processing time.

Implementing the Petri net model with GPenSIM results in four M-files:

1. Petri Net Definition File (PDF), which is declares the static Petri net (the structure of the Petri net defined by the sets of places, transitions, and arcs).
2. Main Simulation File (MSF), which declares the initial dynamics (e.g., initial tokens in the places and the firing times of transitions).
3. COMMON_PRE, which defines the conditions for the enabled transitions to satisfy before start firing. One of the main parts of the file is the reservation of resources by the transitions (tasks).
4. COMMON_POST, which defines the post-firing actions of the transitions. In this file, releasing of resources are coded here.

Due to brevity, the contents of the four files are not shown in this paper. However, the interested reader is referred to the webpage [15] from where the complete code can be downloaded.

A. Simulation Results

Fig.5 and Fig.6 show the Gantt chart resulting from the Naïve Solution and the proposed algorithm, respectively.

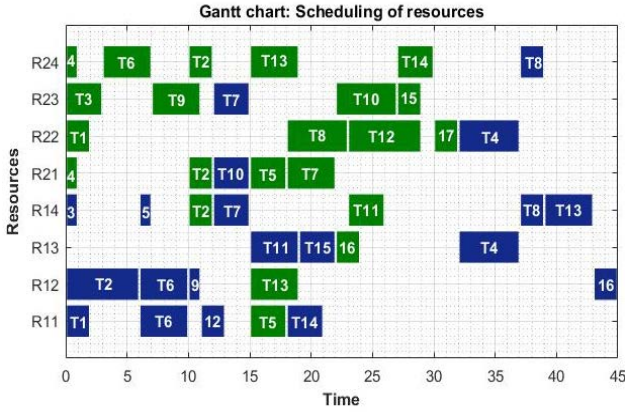

Figure 5. The Gantt chart resulting from the Naïve Solution

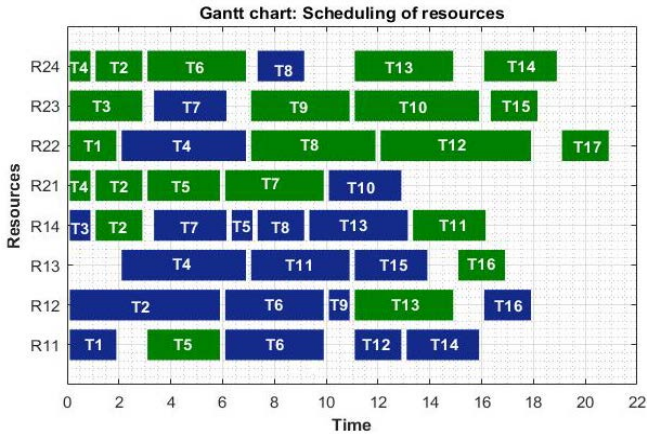

Figure 6. The Gantt chart resulting from our proposed algorithm

The tasks that are with blue color are from the first module, and the tasks with the green color are from the second module. The length of each rectangle is the processing time of the task. The results from naïve algorithms show that the completion time of the first module is 45 TU and 32 TU for the second module. Whereas, the results from the proposed algorithm show that the first module is completed in 18 TU and the second module in 21 TU.

V. DISCUSSION

The algorithms presented in this paper are extended versions of the basic Depth-First-Search (DFS) graph algorithm. Using DFS based graph algorithms for scheduling jobs in grid computing is not a new concept. For example, [16] presents modelling and scheduling of Flexible Manufacturing Systems using Buffer-nets and Artificial Intelligence (AI) based on heuristic search methods. In this paper, scheduling is performed as DFS-based heuristic search. [17] proposes a scheduling algorithm that minimizes execution cost of scheduling while retaining the deadline for completion of the jobs. For scheduling, the algorithm uses Markov Decision Process, and Breadth-First Search (BFS) and DFS algorithms with critical path analysis is used for respecting the deadlines. iGrid2005 [18] is a system for scheduling computing and network resources through Web services interfaces. iGrid2005 uses dedicated schedulers (e.g., grid resource scheduler and network resource scheduler) for scheduling different type of entities. The scheduling employs a DFS technique to find required resources and the earliest timeframe to meet the user-specified deadline. Also, there are many other works (e.g., [19-22]) that use DFS-based techniques for scheduling of jobs in grid environments.

What is new is this paper is the simplicity of the algorithm and that the correctness and validity of the algorithm can be verified by simple simulation runs. Whereas, the so-called advanced algorithms that combine graph-based algorithms with AI, evolutionary algorithms, etc., are either fail to discuss the validity of the algorithms or present complex procedures for verifications that cannot be easily verified. Since all scheduling algorithms are NP-hard and thus gives optimistic solutions instead of optimal solutions, verification of the proposed algorithm for their optimistic results are essential. The proposed algorithm in this paper is Petri net based (thus easy to understand and extend) and DFS-based, hence provides a simple verification process.

VI. CONCLUSION

In this paper, we proposed an algorithm to coordinate the resources and the tasks from different grid modules. The main purpose was to have an understandable and significant algorithm that can be implemented with the GPenSIM tool as a Petr net model on the MATLAB

platform. The simulation results show that the proposed algorithm minimizes the processing times of the modules. Furthermore, resources are well organized and that there is no deadlock during the processing.

Further Work: As future work, we intend to extend the algorithm in two directions: 1) priority-based, and 2) cost-based. In the former case, all the tasks (both internal and external) will be assigned priorities. In this case, an external task with a higher priority can win a local resource, had it compete with a local resource with lower priority. The algorithm discussed in this paper is free from priorities; thus local task always wins if it competes with an external one. In the second case, all the tasks will be assigned with costs, in addition to processing times. In this case, optimal scheduling does not necessarily mean minimization of total time for completion of jobs, as it can mean minimization of total costs of completion too.

REFERENCES

[1] R. Buyya & M. Murshed (2002). Gridsim: A toolkit for the modeling and simulation of distributed resource management and scheduling for grid computing. Concurrency and computation: practice and experience, 14(13‐15), 1175-1220.

[2] R. Davidrajuh, A New Two-Phase Approach for Petri Net Based Modeling of Scheduling Problems. In Industrial Engineering, Management Science and Applications 2015 (pp. 125-134). Springer, Berlin, Heidelberg. 2015.

[3] V. Hamscher, U. Schwiegelshohn, A. Streit, & R. Yahyapour, Evaluation of job-scheduling strategies for grid computing. In International Workshop on Grid Computing (pp. 191-202). Springer, Berlin, Heidelberg. December 2000.

[4] M. Silberstein, D. Geiger, A. Schuster, & M. Livny, Scheduling mixed workloads in multi-grids: the grid execution hierarchy. In High Performance Distributed Computing, 2006 15th IEEE International Symposium on (pp. 291-302). IEEE. June 2006.

[5] Simscript II. Programming Language. Los Angeles, CA: CACI, 22, 72-80. 1987.

[6] A. Varga, Omnet++ user manual. OMNeT++ Discrete Event Simulation System. Available at: http://www. omnetpp. org/doc/manual/usman. html.

[7] F. P. Casas & J. Casanovas, JGPSS, an Open Source GPSS Framework to Teach Simulation. In Procediings of the Winter Simulation Conference 2009.

[8] J. O. Henriksen, & R. C. Crain, GPSS/H reference manual. Wolverine Software Corporation. 1989.

[9] A. Takefusa, Bricks: A performance evaluation system for scheduling algorithms on the grids. In JSPS Workshop on Applied Information Technology for Science (JWAITS 2001). January 2001.

[10] H. J. Song, X. Liu, D. Jakobsen, R. Bhagwan, X. Zhang, K. Taura, & A. Chien, The microgrid: a scientific tool for modeling computational grids. In Supercomputing, ACM/IEEE 2000 Conference (pp. 53-53). IEEE. November, 2000.

[11] H. Casanova, Simgrid: A toolkit for the simulation of application scheduling. In Cluster computing and the grid, 2001. proceedings. first ieee/acm international symposium on (pp. 430-437). IEEE. 2001.

[12] General Purpose Petri Net Simulator (GPenSIM): http://davidrajuh.net/gpensim/

[13] R. Davidrajuh, "Developing a new Petri net tool for simulation of discrete event systems." 2008 Second Asia International Conference on Modelling & Simulation (AMS). IEEE, 2008.

[14] R. Davidrajuh, Modeling Discrete-Event Systems with GPenSIM: An Introduction. Springer. 2018.

[15] Complete Code for the Example. Available online: http://www.davidrajuh.net/gpensim/Pub/2018/UKSim2018/

[16] A.Reyes, H.Yu, G.Kelleher, and S.Lloyd, "Integrating Petri Nets and hybrid heuristic search for the scheduling of FMS," Computers in Industry, 47 (1), 123-138, January 2002.

[17] J. Yu, R. Buyya, and C.K.Tham, "Cost-based Scheduling of Scientific Workflow Applications on Utility Grids," In e-Science and Grid Computing, 2005. First International Conference on (pp. 8-pp). IEEE.

[18] A. Takefusa, m. Hayashi, N. Nagatsu, H. Nakada, T. Kudoh, T. Miyamoto, T. Otani, H. Tanaka, M. Suzuki, Y. Sameshima, W. Imajuku, M. Jinno, Y. Takigawa, S. Okamoto, Y. Tanaka, S. Sekiguchi, "G-lambda: Coordination of a Grid scheduler and lambda path service over GMPLS," Future Generation Computer Systems, 22(2006), 868–875

[19] M. Cafaro, I. Epicoco, M. Mirto, D. Lezzi, & G. Aloisio, "The grid resource broker workflow engine," Concurrency and Computation: Practice and Experience, 20(15), 1725-1739, 2008.

[20] Krenczyk, D., Davidrajuh, R., & Skolud, B. "An Activity-Oriented Petri Net Simulation Approach for Optimization of Dispatching Rules for Job Shop Transient Scheduling," In International Joint Conference SOCO'17-CISIS'17-ICEUTE'17 León, Spain, , Proceeding (pp. 299-309). Springer, September 6–8, 2017.

[21] S. Merz, M. Quinson, & C. Rosa, "Simgrid mc: Verification support for a multi-api simulation platform". In Formal Techniques for Distributed Systems (pp. 274-288). Springer, Berlin, Heidelberg. 2011.

[22] J. Yu, Qos-based scheduling of workflows on global grids, Doctoral dissertation. 2007.