International Conference on Computational Science and Information Management (ICoCSIM)
Vol. 1 (2012) 259-265

Combinatorial Rules Approach to Improve Priority Rules Scheduler in Grid Computing Environment

Zafril Rizal M Azmi[a*], Kamalrulnizam Abu Bakar[b], Abdul Hanan Abdullah[b], Mohd Shahir Shamsir[c], Tutut Herawan[a]

[a]*Faculty of Cumputer System and Software Engineering, Universiti Malaysia Pahang, 26300 Kuantan, Pahang, Malaysia*
[b]*Faculty of Computer Science and Information System, Universiti Teknologi Malaysia, 81310 Skudai, Johor, Malaysia*
[c]*Faculty of Biosciences and Bioengineering, Universiti Teknologi Malaysia, 81310 Skudai, Johor, Malaysia*

Abstract

This paper proposed a set of Combinatorial Rules (CR) based on five traditional Priority Rules (PR) algorithms. The proposed CR aims to improve the PR by balancing the tradeoff between five performance metrics. Experimental results shows that the proposed algorithms are much better compared to the PR and well known SJFR-LJFR.

Keywords: Priority Rule, Combinatorial Rule, Grid Scheduling, performance.

1. Introduction

Priority Rule (PR) algorithms are based on the Giffler and Thompson algorithms [14]. The problem of determining powerful PR has produced and is still producing an abundance of literature. A very large number of rules or combination of rules have been defined and tested in many field of study. Traditionally, in the Grids scheduling environment, the PR algorithms such as First Come First Serve (FCFS) have been widely applied to solve the queuing problems but recently several researchers have combined the PR algorithms with meta-heuristics technique to schedule the jobs and resources effectively. Most of these works have used PR as initial scheduler for their meta-heuristics scheduler[5]. Among them, Abraham in his work [2] used Longest Job First (LJF) and FCFS as an initial schedule for the Fuzzy Particle Swarm based scheduling. [23] used Shortest Job First (SJF) as initial scheduler for their Swift Scheduler algorithm. [9] used SJF and LJF for initial schedule for the Genetic Algorithms based scheduler. Based on these previous works, it is very important to investigate the performance of these PR algorithms. Several works have been carried out to study performance of some PR algorithms and the results shows that no single PR is capable of performing equally well across the entire common matrics of performance[4, 15]. In order to overcome this limitation, researchers introduced combinatorial rule [25] by combining more than one PR in a predetermined manner. A combinatorial rule begins with a specific PR and switches to another PR when it reaches a predetermined level. The most well known combinatorial rule is the combination of Shortest Job First with Longest Job First [1], which design to balance the performance of only two important performance measures: flowtime and makespan. However it is not clear what the effects are this algorithm bring to the others metrics of performance.

This paper will introduce a new combinatorial rule that not only balanced and minimize one or two performance metrics but five commonly used metrics: flowtime, makespan, total number of delayed jobs, total tardiness and utilization. For the sake of comparison, a few other combinatorial rule algorithms will also be introduced. These algorithms are combination of two from five well known PR: First Come First Serve (FCFS), Longest Job First (LJF), Shortest Job First (SJF), Earliest Deadline First (EDF) and Minimum Time To Deadline (MTTD).

2. Priority Rules

Priority rules also referred as Queue-based [18]. Instead of guaranteeing optimal solution, these techniques aim at finding reasonably good solutions in a relatively short time. Although it is a suboptimal algorithms, it is yet the most frequently used for solving scheduling problem in real world because of the easiness to implement and their low time complexity. The most basic priority rules algorithm is First Come First Serve (FCFS). The rules presented in Table 1 are the basic rules. To design more complex rules, a set of elementary simple rules must be defined. The following definition and notation are introduced [15, 21]:

p_i: processing time for job i.
d_i: due date of job i.
r_i: arrival or release time of job i.

Table 1: Formalization of Basic Priority Rules

Rule	Definition	Description
FCFS	r_j	Arrived at queue first
SJF	p_j	The shortest processing time
LJF	$-p_j$	The longest processing time
MTTD	d_i-r_j	The smallest allowance
EDF	d_j	The earliest due date

FCFS or also known as First In First Out (FIFO) is the simplest and the basic of scheduling. It is known to be used worldwide in many fields that involve client-server interaction. In grid scheduling, FCFS policy manage the jobs based on their arriving time which mean first job will be process first without other biases or preferences. This concept has been used by several well-known enterprise scheduler such as MAUI [17] and PBS [16]. As can be seen in Algorithm 1, FCFS do not require any specific computation or process to schedule the jobs. The only thing this algorithm concern is the sequence of the jobs. Job entering the system on r_i time will be processed earlier then job on r_{i+1}.

Algorithm 1 FCFS

0 Add job$_i$ last in the main queue

SJF also known as Shortest Job Next (SJN) or Shortest Process Next (SPN) is a scheduling technique that selects the job with the smallest execution time to execute next. It also means that job with the longest execution time will receive the lowest priority and will be put last in the queue. Abraham in [1] said that the theoretical rule to minimize the flowtime is to schedule the shortest job on the fastest resource. Since this policy gives preference to some groups of jobs over other group of jobs, this policy is not fair compared to FCFS policy. In extreme cases, when jobs with shorter execution time continue arriving, the jobs with longer execution period may not get a chance to execute and may have to wait forever. This situation is known as starvation and could be a serious problem and shows the low degree of fairness for this policy [13]. In addition SJF also is believed to have the maximum makespan time compared to other algorithms in this paper because of its characteristics mentioned.

Algorithm 2 SJF

0 **for** i=0 to i<main queue.size
1 **if** job$_{i+1}$ length <job$_i$length **then**
2 add job$_{i+1}$ in front of job$_i$ in the queue
3 **end if**
4 **if** main queue.size = 0 **then**
5 add job$_i$last in the main queue
6 **end if**

LJF have the contradiction behavior of SJF. While shortest job is believe will reduce the flowtime, processing longest job first on the fastest resource according to Abraham in [1] will minimize the makespan time. However, LJF will be suffering due to slightly increase in the flowtime.

Algorithm 3 LJF

0 **for** i=0 to i<main queue.size
1 **if** job$_{i+1}$ length >job$_i$length **then**
2 add job$_{i+1}$ in front of job$_i$ in the queue
3 **end if**
4 **if** main queue.size = 0 **then**
5 add job$_i$last in the main queue
6 **end if**

MTTD is a scheduling algorithm which put the priority on the jobs according to the time that can be considered for the job to be executed without tardiness [3]. In the simplest words MTTD aims for producing a scheduler that has minimum tardiness. To achieve this objective, MTTD define the time as follow:

(Deadline-Release Date) (1)

Algorithm 4 MTTD

```
0  for i=0 to i<main queue.size
1    if job_{i+1} due date - job_{i+1} release date <job_i due date -
job_i release date then
2     add job_{i+1} in front of job_i in the queue
3    end if
4  if main queue.size = 0 then
5    add job_ilast in the main queue
6  end if
```

EDF is a policy that checks all the incoming jobs due date or deadline. Jobs will be process or put in the queue based on the time indicate by the dateline. First job to meet the deadline will be put first in the queue.

Algorithm 5 EDF

```
0  for i=0 to i<main queue.size
1    if job_{i+1} due date <job_idue date then
2      add job_{i+1} in front of job_i in the queue
3    end if
4  if main queue.size = 0 then
5    add job_ilast in the main queue
6  end if
```

3. Performance Metrics

Total Tardiness: One of the main objectives of the scheduling procedure is the completion of all jobs before their agreed due dates. Failure to keep that promise has negative effects on the credibility of the service provider. The lateness of job j can be defined as the difference between its completion time C_j and the corresponding due date d_j. This metric known as the tardiness of job is calculated using the following expression:

$$T_j = \max(0, C_j - d_j) \tag{2.1}$$

On the other hand, total tardiness involving a set of n jobs, which are to be processed each in a single machine, the formula can be described as follows:

$$T = \sum_{j=1}^{n} \max(0, C_j - d_j) \tag{2.2}$$

$$C_j = S_j + p_j \tag{2.3}$$

Where C_j is the completion time of job j and S_j is the start time of job j in machine m. Each job j has processing time p_j, and due date d_j. The total tardiness problem is not easy to solve, especially for large values of n[10].

Makespan: Small values of makespan mean that the scheduler is providing good and efficient planning of jobs to resources. The makespan of a schedule can be defined as the time it takes from the instant the first task begins execution to the instant at which the last task completes execution [26]. Makespan can be representing by the following equation:

$$C_{\max} = \max_{1 \le j \le n} C_j \tag{2.4}$$

In simplest words, makespan is the time when finishes the latest job.

Flowtime: Flowtime also known as response time. Flowtime is the sum of completion times C_j of all the jobs [26]. Mathematically, flowtime can be formulated as:

$$F = \sum_{j=1}^{n} C_j \tag{2.5}$$

Flowtime and makespan always become two major objectives to be minimized in research involving scheduling. However,

minimization of makespan always results in the maximization of flowtime[24].

Delayed jobs: Delayed Jobs means jobs that failed to meet their deadline or due date is a period of time a job must be completed [8]. The goal typically in such problems is to complete the maximum number of jobs by their deadlines [6]. A higher machine usage fulfills resource owner's expectations, while a higher number of non delayed jobs guarantees a higher Quality of Service (QoS) provided to the users [18, 19]. By reducing the number of delayed jobs, QoS for the system that using the proposed scheduling technique also will be improved. Delayed jobs D is measured by:

$$D = \sum_{j=1}^{n} D_j \tag{2.6}$$

Where

$$D_j = \begin{cases} 1 & if\ (C_j > d_j) \\ 0 & otherwise \end{cases} \tag{2.7}$$

D_j equal to 1 if job j is late ($C_j > d_j$). Otherwise D_j is equal to 0.

Resource Utilization: Maximizing resource utilization or machine usage of the grid system is another important performance metrics. Utilization is the percentage of time that a resource is actually occupied, as compared with the total time that the resource is available for use. Low utilization means a resource is idle and wasted. Throughout this paper resource utilization will be referred as machine usage. According to [7, 22], Machine usage (MU) is computed as:

$$MU = \frac{CPUs_{active}}{\min\left(CPUs_{available}, CPUs_{required}\right)} \tag{2.8}$$

Where $CPUs_{active}$ denotes the numbers of non idle CPUs, $CPUs_{available}$ denotes the number of available CPUs, and $CPUs_{required}$ represent the number of required CPUs.

4. Proposed Combinatorial Rules Algorithms

In order to minimize one performance criteria, traditional PR will have to sacrifice the others criteria. Combinatorial Rule (CR) intends to overcome this limitation. However, to our knowledge there is still no study carried out to examine the effect of CR to other performance metrics other than the one they try to tackle. For example, in [1] SJF and LJF were implement together in order to balance the tradeoff between flowtime and makespan without mentioning the effect to the other criteria. This paper try to search for the best CR that can minimize all five metrics covered in this study. To achieve this, SJF, LJF, EDF and MTTD were paired together to form CR algorithms as shown in Table 2.

Table 2: Proposed Combinatorial Rules

Combinatorial Rule
MTTDFR-SJFR
EDFR-SJFR
MTTDFR-LJFR
EDFR-LJFR
EDFR-MTTDFR

Note that the CR algorithms in Table 2 have –FR in them which means Fastest Resource. In order to fully utilize the PR algorithms, the conceptual idea of PR-FR have been introduced by [1]. Through their work, the Shortest Job on Fastest Resource (SJFR) and Longest Job on Fastest Resource (LJFR) schedules have been proposed to minimize both flowtime and makespan. The combination of SJFR-LJFR algorithms is believed to minimize both makespan and flowtime at the same time. The concept of SJFR-LJFR later on have been applied as initial schedule for Genetic Algorithm based schedulers [9, 11, 12, 20]. LJFR-SJFR initially assigns the longest jobs, increasingly sorted by their workload, to the available resources, increasingly sorted by their computing capacity (application of LJFR heuristic). Then, for the remaining jobs, in every step, an unassigned job j is assigned to the first available machine, being j the longest job (LJFR) or the shortest job (SJFR) alternatively. This is done until all jobs have been allocated. Basically, the difference between SJF with SJFR and LJF with LJFR is the resource selection for SJFR and LJFR have been fixed to fastest resource available in the Grid system, while for SJF and LJF, there are no specific rule to determine which resources should be selected as a $resource_{candidate}$. The interaction between PR algorithms and Resource Selection (RS)

algorithms can be seen at Figure 1. Similarly, the proposed PR-FR firstly considers which job *j* will be sorted first in the main queue based on a specific rule. Job *j* then is send to the fastest resource ($resource_{candidate}$) available in the Grid system. By using this concept, MTTDFR have been introduced along with two other PRFR algorithms; First Come First Serve-Fastest Resource (FCFSFR) and Earliest Deadline First-Fastest Resource (EDFFR). Table 3 shows the list of PRFR used in this study.

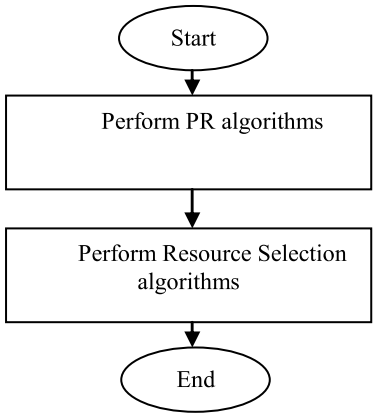

Fig.1: Interaction between PR and RS

Table 3: Priority Rule - Fastest Resource (PRFR) Algorithms

Rule	Description
FCFSFR	Execute job arrived at queue first on the fastest resource
SJFR	Execute job with the shortest processing time date on the fastest resource
LJFR	Execute job with the longest processing time date on the fastest resource
MTTDFR	Execute job with the smallest allowance on the fastest resource
EDFFR	Execute job with the earliest due date on the fastest resource

Structured of CR introduced in this study was adapted from LJFR-SJFR introduced by [1]. LJFR-SJFR algorithms consist of LJFR and SJFR combined together. However, it is separately execute in turn. For example, job J_1will be execute using LJFR, J_2 by SJFR, J_3 back to LJFR and so on. To make sure jobs continuously dispatch based on two different rules for example LJFR and SJFR, CR implemented in this study used a counter named *icount*. *icount* will be reset to the value that represent specific algorithm. For example *icount* = 0 is meant to be for LJFR and *icount* = 1 is for SJFR. This study introduces new CR algorithm as can be seen in Table 2. In order to do these, Algorithm 8 have to be modified. For example, if MTTDFR-SJFR were used, line 1 has to be change to Algorithm 4.

Algorithm 8 LJFR-SJFR

```
0  ificount=0
1      perform Algorithm 3
2      icount=1
3  else if icount=1
4      perform Algorithm 2
5      icount=0
6  end if
7  schedule job using Algorithm 7
```

4.1 Experimental Results for Proposed CR Algorithms.

In order to examine the impact of each CR used in this research, the algorithms were evaluated using grid simulator named Alea. The datasets used consists of five folders. Each folder contains 20 different jobs file and each file has 3000 different jobs with specific inter-arrival time generated from negative exponential distribution. DataSets-1 contained jobs with inter-arrival 1, DataSets-2 contained jobs with inter-arrival 2 and so on. Jobs with inter-arrival 1 will have higher system contention while inter-arrival 5 is the lowest. There are total of 150 heterogeneous machines used in this experiment. Total numbers of experiments are 20 x 5 = 100 for each algorithm. Results from the experiments (Figure 2-4) proven that LJFR-SJFR algorithm when apply together will balanced the tradeoff between makespan and flowtime which obviously much better compared to stand alone SJF and LJF. However, the side effect of this algorithm can be seen on the dropdown of performance for total number of delayed jobs, total tardiness and machine utilization. The proposed MTTDFR-SJFR on the other hand, compared to other CR algorithms, recorded the best performance in term of number of delayed jobs, utilization and total tardiness. It is also shown the makespan of MTTDFR-SJFR is 2.6% better than LJFR-SJFR. Unfortunately, LJFR-SJFR still has better flowtime however it is just 0.98% lower than MTTDFR-SJFR (Ta=1).

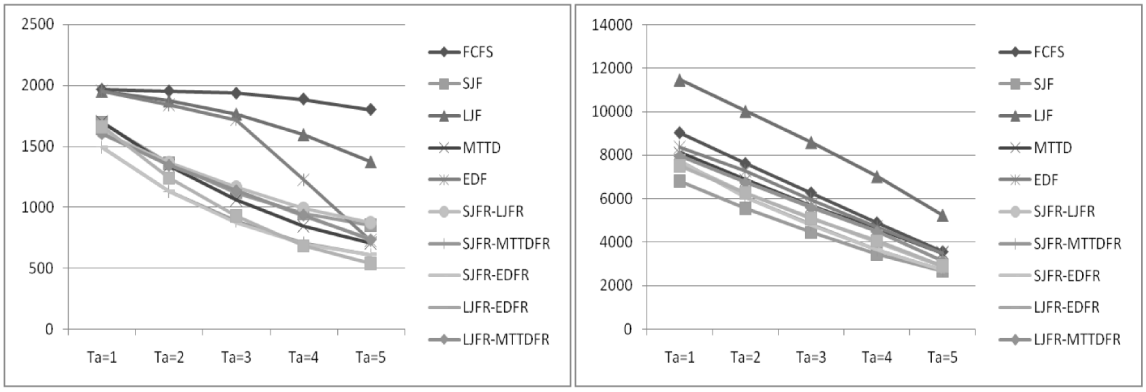

Fig. 2. (a) CR Delayed Jobs (b) CR Flowtime

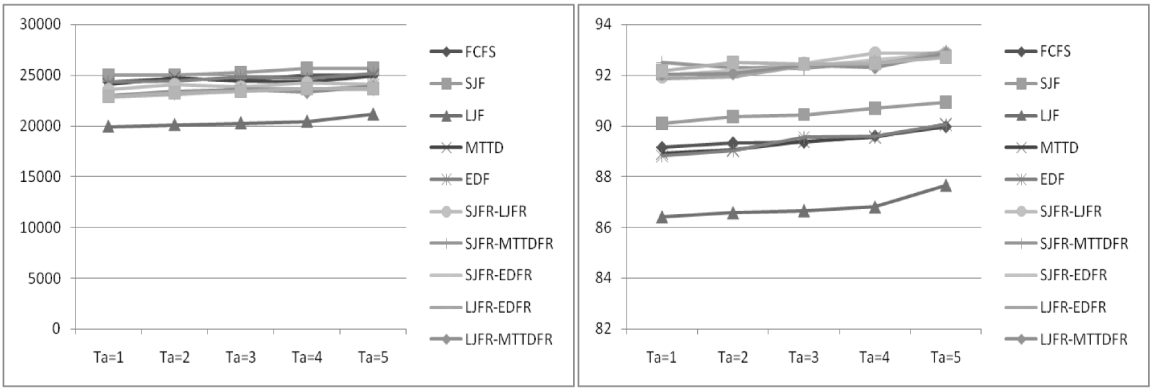

Fig. 3. (a) CR Makespan (b) CR Utilization

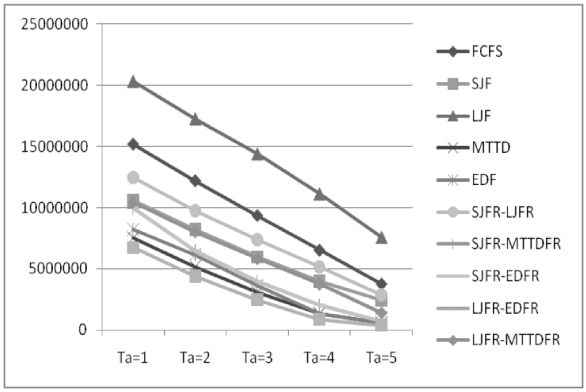

Fig. 4. CR Total Tardiness

5. Conclusion

Prior scheduling work shows that no single priority rule is capable of performing well on all five performance metrics; flowtime, makespan, number of delayed jobs, total tardiness and utilization. Several research have introduced combinatorial rule to solve this problem. However, there are still no works that focus on these five metrics. This paper investigates this issue and using combinatorial rule concept to produce new scheduling algorithm from the combination of MTTD and SJF. From the results, this algorithm performance is the best compared to other combinatorial rule algorithms tested. In order to achieve this, this paper also introduced new combination of Priority Rule – Fastest Resource algorithm that further improves the performance of Priority Rule.

References

[1] Abraham, A., R. Buyya, and B. Nath, 2000. Nature's Heuristics for Scheduling Jobs on Computational Grids, in The 8th IEEE International Conference on Advanced Computing and Communications (ADCOM 2000): Cochin, India.

[2] Abraham, A., H. Liu, W. Zhang, and T.-G. Chang, 2006.Scheduling Jobs on Computational Grids Using Fuzzy Particle Swarm Algorithm, in Knowledge-Based Intelligent Information and Engineering Systems, B. Gabrys, R. Howlett, and L. Jain, Editors. Springer Berlin / Heidelberg. p. 500-507.

[3] Aysan Rasooli, O. 2008. Introduction of Novel Rule Based Algorithms for Scheduling in Grid Computing Systems. Proceedings of the 2008 Second Asia International Conference on Modelling & Simulation (AMS): IEEE Computer Society

[4] Azmi, Z. R. M., K. A. Bakar, A. H. Abdullah, M. S. Shamsir, and W. N. W. Manan, 2011. "Performance Comparison of Priority Rule Scheduling Algorithms Using Different Inter Arrival Time Jobs in Grid Environmen". International Journal of Grid and Distributed Computing. Vol. 4 (No. 3): p. 61-70.

[5] Azmi, Z. R. M., K. A. Bakar, A. H. Abdullah, M. S. Shamsir, R. N. Romli, and S. A. M. Sharif, 2011.Grid Jobs Scheduling Improvement Using Priority Rules and Backfilling, in Software Engineering and Computer Systems, J. Mohamad Zain, W.M.b. Wan Mohd, and E. El-Qawasmeh, Editors. Springer Berlin Heidelberg. p. 401-415.

[6] Bansal, N., H.-L. Chan, T.-W. Lam, and L.-K. Lee, 2008.Scheduling for Speed Bounded Processors, in Automata, Languages and Programming, L. Aceto, I. Damgård, L. Goldberg, M. Halldórsson, A. Ingólfsdóttir, and I. Walukiewicz, Editors. Springer Berlin / Heidelberg. p. 409-420.

[7] Baraglia, R., P. Dazzi, G. Capannini, and G. Pagano. 2010. A Multi-criteria Job Scheduling Framework for Large Computing Farms. Computer and Information Technology (CIT), 2010 IEEE 10th International Conference on.

[8] Brucker, P., 2007. Scheduling Algorithms. 5th ed. Berlin Heidelberg: Springer

[9] Carretero, J. and F. Xhafa, 2006. "Using Genetic Algorithms for Scheduling Jobs in Large Scale Grid Applications". Journal of Technological and Economic Development. 12: p. 11-17.

[10] Dimopoulos, C. and A. M. S. Zalzala, 2001. "Investigating the use of genetic programming for a classic one-machine scheduling problem". Advances in Engineering Software. 32(6): p. 489-498.

[11] Duran, B. and F. Xhafa, 2006. The effects of two replacement strategies on a genetic algorithm for scheduling jobs on computational grids, in Proceedings of the 2006 ACM symposium on Applied computing. ACM: Dijon, France. p. 960-961.

[12] Fatos, X. 2008. Tuning Struggle Strategy in Genetic Algorithms for Scheduling in Computational Grids.

[13] Garrido, J. M., 2000. Performance modeling of operating systems using object-oriented simulation: a practical introduction. Norwell, MA: Kluwer Academic Publishers

[14] Giffler, B. and G. L. Thompson, 1960. "Algorithms for Solving Production-Scheduling Problems". Operations Research. 8(4): p. 487-503

[15] Haupt, R., 1989. "A survey of priority rule-based scheduling". OR Spectrum. 11(1): p. 3-16.

[16] Henderson, R., 1995.Job scheduling under the Portable Batch System, in Job Scheduling Strategies for Parallel Processing, D. Feitelson and L. Rudolph, Editors. Springer Berlin / Heidelberg. p. 279-294.

[17] Jackson, D., Q. Snell, and M. Clement, 2001.Core Algorithms of the Maui Scheduler, in Job Scheduling Strategies for Parallel Processing, D. Feitelson and L. Rudolph, Editors. Springer Berlin / Heidelberg. p. 87-102.

[18] Klusacek, D. and H. Rudova, 2008. Improving QoS in computational Grids through schedule-based approach. , in Scheduling and Planning Applications Workshop at the Eighteenth International Conference on Automated Planning and Scheduling (ICAPS 2008): Sydney, Australia

[19] Klusacek, D., H. Rudová, R. Baraglia, M. Pasquali, and G. Capannini, 2008.Comparison Of Multi-Criteria Scheduling Techniques, in Grid Computing, S. Gorlatch, P. Fragopoulou, and T. Priol, Editors. Springer US. p. 173-184.

[20] Kołodziej, J. and F. Xhafa, 2011. "Enhancing the genetic-based scheduling in computational grids by a structured hierarchical population". Future Generation Computer Systems. 27(8): p. 1035-1046.

[21] Lopez, P. and F. Roubellat, eds. 2008. Production Scheduling. John Wiley & Sons: London, UK.

[22] Pasquali, M., R. Baraglia, G. Capannini, L. Ricci, and D. Laforenza, 2011. "A multi-level scheduler for batch jobs on grids". J. Supercomput. 57(1): p. 81-98.

[23] Somasundaram, K. and S. Radhakrishnan, 2009. "Task Resource Allocation in Grid using Swift Scheduler". International Journal of Computers Communications & Control. 4(2): p. 158-166.

[24] Subashini, G. and M. C. Bhuvaneswar, 2011. "Non Dominated Particle Swarm Optimization For Scheduling Independent Tasks On Heterogeneous Distributed Environments". Int. J. Advance. Soft Comput. Appl. vol.3(1).

[25] Tzafestas, S. and A. Triantafyllakis, 1994. "A new adaptively weighted combinatorial dispatching rule for complex scheduling problems". Computer Integrated Manufacturing Systems. 7(1): p. 7-15.

[26] Xhafa, F. and A. Abraham, 2008.Meta-heuristics for Grid Scheduling Problems, in Metaheuristics for Scheduling in Distributed Computing Environments, F. Xhafa and A. Abraham, Editors. Springer Berlin / Heidelberg. p. 1-37.