A Scheduling Policy for Improving Tardiness Performance

Papakostas N.
Laboratory for Manufacturing Systems and Automation, University of Patras, Patras, Greece

Chryssolouris G.
Laboratory for Manufacturing Systems and Automation, University of Patras, Patras, Greece

Abstract
This paper proposes a scheduling policy with a new dispatch rule, named RTSLACK, which is based on maximizing the slack time of the remaining tasks in the manufacturing resources queues. This scheduling policy is tested against three widely used dispatch rules, such as the Shortest Process Time rule (SPT), the Earliest Due Date rule (EDD), and the minimum slack time rule (SLACK). A generic form of the RTSLACK rule is also devised, with the aid of a stochastic hill climbing algorithm. A series of experiments with different manufacturing configurations and workload patterns are conducted. The results show that the proposed policy leads to an improved tardiness performance for specific workload patterns and it can act complementarily to the other rules, namely the SPT, the EDD and the SLACK

Keywords: *Scheduling, manufacturing, planning and dispatch rule*

1 Introduction

Production scheduling has attracted many researchers from both academia and industry during the past decades. Nearly every manufacturing organization has to deal with scheduling tasks on a daily basis. Nevertheless, although the academic research has generated an abundance of theoretical work, on a number of classical scheduling problems, the use of academic results in industry has been rather minimal [1]. One particular reason for this fact is that the actual manufacturing systems are extremely variable and many academic approaches are far from practical when used in an industrial environment. Overall, the complexity of the scheduling problem and the increasingly important need for optimising the performance of a manufacturing system in today's competitive world, dictate the formulation of new, practical, ready to use, methods for addressing the production scheduling challenges. Typically, the entities to be scheduled in a manufacturing system are referred to as jobs. They usually correspond to individual parts. The individual operations, required for the production of a part, thus completing a job, are referred to as tasks. The simplest scheduling case is the so-called single machine scheduling problem, whose objective is to define the dispatching sequence of all available jobs-tasks in a single machine. On the contrary, for the scheduling of a flow shop, there are many machines and the objective is to schedule all jobs, with an identical, however, precedence ordering of the tasks. A hybrid flow shop, in particular, is composed of a series of production stages with several identical parallel machines at each stage.

In this work, the following assumptions have been made:

- Parts (jobs) are not known in advance; they arrive at the manufacturing system in the form of orders, according to specific demand patterns.
- Each machine may process only one task at a time.
- Whenever a machine is available, it immediately processes the task with the highest priority from the queue of pending tasks.
- Precedence constraints, where applicable, between the tasks of a job, constituting the job routing, have to be satisfied: a succeeding task cannot start if the preceding one has not been completed.

- Pre-emption is not allowed: a task cannot be interrupted and be replaced by another one.
- Machine breakdowns or maintenance tasks are not considered.

Traditionally, a significant part of the research carried out in production scheduling has been focused on heuristic policies, called dispatch or dispatching rules. In general, a heuristic solution is often based on intuition, whilst its application on empirical evidence of its success, which, in turn, is considered as its ability to find, by some measure, a solution, close enough to the optimum [1]. In the production scheduling domain, a dispatching rule is the method of ranking a set of tasks, which are waiting to be processed by a machine. The task with the best rank is selected to be processed on the machine [1]. In the last decades, numerous dispatch rules have been proposed. Haupt [2] conducted an extensive literature survey on heuristic priority rule-based job shop scheduling rules and presented a classification, a characterization and evaluation of elementary priority rules. Chiang and Fu [3] emphasised on the usage and performance of dispatching rules for the job-shop scheduling, with due date-based objectives. Apart from the various elementary, though simple and quite often used, dispatch rules, they propose new scheduling policies, which combine different priority rules [3]. In particular, they present a new rule, combining three well-known rules: 'Shortest Process Time', 'Earlier Due Date' and 'Longer Remaining Processing Time'. The simulation study that was conducted, confirmed an advantage of this rule, regarding tardy rate and mean tardiness and this indicates that the proposed rule is a promising alternative for the job shops, in which a due date-based performance is of primary importance. Furthermore, in another work [4], the combination of past dispatching rules has been utilized for the formation of three new scheduling policies. All these policies are combinations of the following dispatch rules: Shortest Process Time – SPT, where the task with the shortest processing time is selected and the SLACK rule, where the job selected has the least slack time, determined by its due date minus the remaining processing time for the job. Chryssolouris et al proposed a new procedure for determining scheduling policies of manufacturing systems, based on neural networks combined with simulation [5]. The neural network determines the appropriate scheduling policy in order for a set of performance criteria to be satisfied. In [6], a simulation-based, real-time scheduling mechanism, originally proposed by Kim and Kim [7], is presented. Eight scheduling strategies were presented and their performance was compared through a series of computational experiments. Finally, in [7] it was proposed that the predictive ability of simulation-based scheduling can be considered as being better than the majority of the look-ahead heuristics. Apart from the great need for improved dispatching rules, there is also a need for a clearer understanding of the dynamics inherent in the dispatching environment [8]. Agliari et al [9] presented an analytical technique, based on the Markov Chain Theory, including the analysis of well known dispatch rules, such as the 'First Come First Served', the 'Shortest Process Time' and the NEP rule, which is a variable priority rule. When using the NEP rule, the highest priority is assigned to the task, being identical to the one just dispatched. The highest priority is assigned to other tasks when similar ones are not part of the buffer's queue. Other dispatching rules, based on principles dominating other scientific disciplines and domains, were also proposed in the past. A representative example is the Time Delay plot Rule (TDR), which takes advantage of concepts underlying the nonlinear dynamics and chaos theory [10].

Special cases, such as the single machine total weighted tardiness scheduling problem or the flow shop scheduling problem with blocking, have been reported [11,12]. The tardiness of each task is equivalent to the difference between the completion time of a task and its due date. In the case of the weighted tardiness problem, each task is associated with a positive tardiness weight or penalty and the objective is to minimise the overall weighted tardiness, taking into consideration the tardiness of all tasks. In [11], a novelty is presented, allowing for the better determination of a look-ahead parameter, used by the Apparent Tardiness Cost (ATC) rule. This is actually a composite dispatch rule that combines the weighted shortest processing time (WSPT) rule and the minimum slack rule. In the flow shop scheduling problem with blocking, there are no buffers among the successive machines and therefore, the production of the preceding machines is blocked when the supervening ones are not available. In [12], heuristic approaches, employing a combination of other dispatch rules and algorithms, have been proposed for minimising the overall tardiness.

Most of the modern, composite dispatch rules are more complex when compared with the simple, well-

known elementary ones, such as the SPT, the SLACK and the EDD rule. Moreover, they usually require the definition of extra parameters, such as the look-ahead parameter, mentioned above, or complex calculations for identifying the priorities of the queued tasks.

In this paper, a new scheduling policy is proposed, based on a simple, comprehensible rationale. The core of this scheduling policy is a new dispatch rule, whose objective is to select the one task, out of the list of tasks waiting to be processed by one or more manufacturing resources, which would maximise the total slack time of the remaining tasks.

2 The new scheduling policy

In this work, it is assumed that every task of each job may be dispatched to every parallel machine of a work centre. The parallel machines of each work centre are identical, and therefore, the processing time of each task is the same, regardless of the machine it is dispatched to.

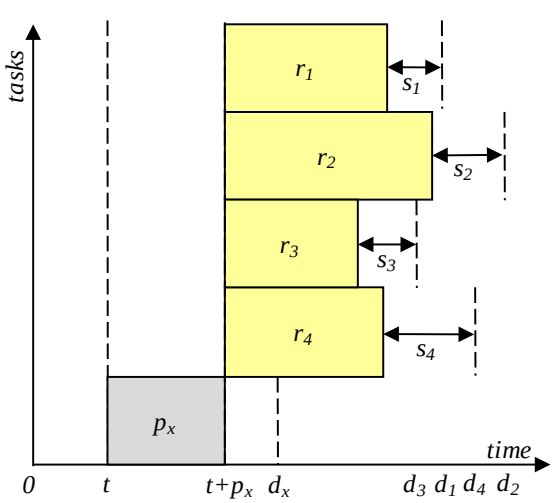

Figure 1: Time variables representation

The variables taken into consideration at every work centre are as in Figure 1:

- n: number of jobs waiting in the queue of the work centre.
- k_i: the index of job i waiting in the queue of the work centre.
- p_i: the processing time of the imminent task of job i in the work centre.
- r_i: the remaining processing time of job i (including p_i).
- d_i: the due date of job i.
- s_i: the slack time of job i; this variable may take negative values when a job is overdue.
- s^{rtot}: the total slack time of the remaining tasks in the queue (the task selected is excluded)
- t: the system time, i.e. the time during which, a machine is available in the work centre and a decision has to be made as to which task to be selected.

Taking into consideration all these definitions, if the task of job x is selected, then the slack time s_i of a job i, with $i \neq x$ is:

$$s_i = d_i - (t + p_x + r_i) \tag{1}$$

The total slack time of all tasks waiting to be dispatched is therefore given by the following equation:

$$s^{rtot} = \sum_{i=1}^{n-1} \left[d_i - (t + p_x + r_i) \right] \tag{2}$$

Apparently:

$$s^{rtot} = \sum_{i=1}^{n-1} d_i - \sum_{i=1}^{n-1} r_i - (n-1)(t + p_x) \tag{3}$$

$$s^{rtot} = \left(\sum_{i=1}^{n} d_i - d_x \right) - \left(\sum_{i=1}^{n} r_i - r_x \right) - (n-1)(t + p_x) \tag{4}$$

$$s^{rtot} = \sum_{i=1}^{n} (d_i - r_i) - (n-1)t - d_x + r_x - (n-1)p_x \tag{5}$$

$$s^{rtot} = \sum_{i=1}^{n} (d_i - r_i) - (n-1)t - \left[(n-1)p_x - r_x + d_x \right] \tag{6}$$

If A is defined according to the following equation:

$$A = \sum_{i=1}^{n} (d_i - r_i) - (n-1)t = ct \tag{8}$$

Then, finally:

$$s^{rtot} = A - \left[(n-1) \cdot p_x - r_x + d_x \right] \tag{8}$$

The basic idea behind the new scheduling policy, denoted from this point onwards as **RTSLACK** (Remaining Tasks SLACK), is to select the job x from the ones waiting to be dispatched at a work

centre, in order for the total slack time of the remaining tasks to be maximised:

$$x : \max\{A - [(n-1)p_x - r_x + d_x]\}, x \in [1, n] \quad (9)$$

Since A is constant, according to equation (7), Equation (9) is equivalent to:

$$x : \min[(n-1)p_x - r_x + d_x], x \in [1, n] \quad (10)$$

It is interesting to note that if the total slack time of all jobs (including job x, which is selected) was to be maximised, then Equation (10) would be:

$$x : \min[p_x], x \in [1, n] \quad (11)$$

Equation (11) is equivalent to the Shortest Processing Time rule and it represents the fact that the use of the SPT rule minimises the total slack time of all jobs, including the job, whose task at a specific work centre, is selected. However, since the job selected will anyway continue its way to the next work centres of its routing, there is a question as to whether the slack time of this particular task should be included in the total slack time, when the slack time of the system's jobs is the only criterion for the selection decision. In case that the due date is too far away from the current decision point, having taken into consideration the total remaining time of the job as well, there is a good chance that the task selected will be dispatched much earlier than its due date. Such cases may be observed when the demand patterns and workloads are not too tight. In these cases, the use of the RTSLACK rule would increase the possibility of the job selected not to be completed too early.

One possible drawback of the rule, when used in an actual production environment, is that the priority of the remaining jobs in the queue of a work centre will have to be evaluated every time the queue changes, either due to the arrival of a new job or due to the dispatching of a job to the work centre. One way of addressing this issue is by transforming the proposed rule to a priority-based one, according to the following equations:

$$PR_{k_i} = -[(n-1)p_{k_i} - r_{k_i} + d_{k_i}], \forall i \quad (12)$$

$$x : \max[PR_x], x \in [1, k_{n'}] \quad (13)$$

Whenever a job i with index k_i arrives at a work centre, its priority is estimated according to Equation (12). From that time onwards, whenever a decision has to be made about the job to be selected in this work centre, the selection is carried out, based on the priorities of the waiting jobs according to equation (13), where n' represents the number of waiting jobs in the queue of the work centre at the new decision point. The job with the maximum priority is selected and moves on to the next work centre of its routing, where a new priority is estimated for this particular job, following the same process.

Compared with the dynamic version of the proposed rule as defined in Equation (10), the priority-based rule offers the advantage of estimating the priority of each job only once. The basic concept, underlying the rationale of the RTSLACK rule, is in principle, maintained, since as time passes by, the new jobs added to the waiting queues have on average later due dates and therefore, their priority is lower than that of the jobs that have entered the queue of a work centre earlier on.

Theoretically, the proposed rule could also be modelled in a more generic form:

$$PR_{k_i} = -[b_1(n)(n-1)p_{k_i} - b_2(n)r_{k_i} + b_3(n)d_{k_i}], \forall i \quad (14)$$

where: b_j are the functions of the waiting queue length.

Observing equations (12-14), the RTSLACK rule seems to embody in a linear manner, a combination of the characteristics of the SPT, EDD and SLACK rules. Therefore, it would be expected that the proposed rule performs well in the cases that the values of the characteristics of the demand and workload profiles lie in an intermediate space between those that the SPT, EDD and SLACK rules perform best.

2.1 *Generic form of the RTSLACK rule*

Identifying the functions b_j of Equation (14) is a challenging task by itself. One relatively straightforward way to demonstrate how well the generic form of the RTSLACK rule could perform is to test different combinations of simple functions b_j in Equation (14). This objective may be pursued by employing a non-demanding optimisation algorithm, advancing Equation (14) in terms of tardiness performance in a step-by-step fashion.

A well-known and widely used optimisation approach is the use of a stochastic hill climbing algorithm, especially in the case that the objective function seems to follow a regular yet unknown pattern. Stochastic hill climbing is a variation of the standard hill climbing algorithm, in which the first closer node of the solution space to the current best solution is chosen and tested against, in terms of performance: it selects a neighbour at random and decides whether to move ahead accepting this

neighbour as the current best solution, or not. Since Equation (14) is actually a function of the number n of the waiting jobs, the functions b_j, in their simplest form, would be equal to constants for specific ranges of n. Assuming three ranges of n, the following equations could be formed:

$$b_{j1}(n) = z_{j1}, z_{j1} \in R, \forall j = 1\ldots3, n > n_1 \quad (15)$$

$$b_{j2}(n) = z_{j2}, z_{j2} \in R, \forall j = 1\ldots3, n_2 < n \le n_1 \quad (16)$$

$$b_{j3}(n) = z_{j3}, z_{j3} \in R, \forall j = 1\ldots3, n \le n_2 \quad (17)$$

z_{j1}, z_{j2}, z_{j3} and n_1, n_2 are eleven constant values that have to be identified in Equation (14), for the three different ranges of n, as per Equations (15-17). The pseudocode of the algorithm is shown in Figure 2.

```
L = {b_j1, b_j2, b_j3, n_1, n_2}
initialise(L)
bestTard = simulate(L)
failures = 0
do
{
        for all v in L
        {
                K = L
        K_v = K_v + [2 · rand(0,1) – 1] · step_v
                newTard = simulate(K)
        if [newTard< bestTard]
        {
                        L = K
                        bestTard = newTard
                }
                else
                        failures = failures + 1
         }
} while (failures < 100)
```

Figure 2: Stochastic hill climbing algorithm

3 Experiments

A series of experiments is conducted, in order to evaluate the performance of the RTSLACK rule, under different demand and workload patterns. Three different parameters are used for modelling these patterns:

- The inter-arrival time of jobs, following an exponential distribution.
- The process time of jobs at the work centres, following a normal distribution.
- The due dates of jobs modelled as:

$$d_i = a_i + p_i^{tot} + m_u \quad (18)$$

Where:

- a_i: the arrival time of job i.
- p_i^{tot}: the total processing time of job i.
- m_u: a parameter following a uniform distribution.

The rule is tested against EDD, SLACK and SPT.

The evaluation of the performance of these rules is carried out with the use of the following performance indicators over the total number N of the jobs completed in each experiment:

- *Mean tardiness:*

$$\overline{td} = \sum_{i=1}^{N} [\max(c_i - d_i, 0)] / N \quad (19)$$

- *Maximum tardiness:*

$$td^{\max} = \max_{i=1\ldots N} [\max(c_i - d_i, 0)] \quad (20)$$

where c_i the completion time of finished job i.

- *Fraction tardy*, which is the fraction of delayed jobs, expressed as percentage.

At first, a considerable part of the experimentation focuses on the single machine problem. A few additional experiments have been conducted, however, in order for the performance of the rule, in a hybrid flow shop, to be demonstrated.

3.1 *Single machine problem instances*

The RTSLACK rule and its generic form (G-RTSLACK, paragraph 3.1.1) are first tested on a series of single machine problem instances. This type of problem offers the opportunity for the basic aspects of the rule to be tested as well as for a large number of experiments to be conducted and their results to be analysed with a lower programming and computational effort.

The development of the simulation model as well as the execution of the experiments were both carried out with the help of an implementation, based on the DESMO-J [13] framework. The analysis of the results was carried out through R, a language and an environment for statistical computing [14]. A challenging part of the experimentation is to model the demand and workload profiles in a coherent manner. For this reason, a series of demand profiles were generated according to Table 1.

Table 1: Single machine scheduling - experiments

Exp. Set #	Mean Process Time	Mean Inter-Arrival Time	Due Date Parameter m_u
1-10	norm(100,30)	expo(110)	0,100-1000
11-20	norm(100,30)	expo(120)	0,100-1000
21-30	norm(100,30)	expo(130)	0,100-1000
31-40	norm(100,30)	expo(140)	0,100-1000
41-50	norm(100,30)	expo(150)	0,100-1000
51-60	norm(100,30)	expo(160)	0,100-1000
61-70	norm(100,30)	expo(170)	0,100-1000
71-80	norm(100,30)	expo(180)	0,100-1000
81-91	norm(100,30)	expo(190)	0,100-1000
91-100	norm(100,30)	expo(200)	0,100-1000

Table 2: Single machine scheduling rules performance – mean values

RULE	Mean Tardiness	Fraction Tardy	Max Tardiness
EDD	63.57	26.2%	640.4
SLACK	65.27	26.7%	649.8
RTSLACK	62.36	26.0%	834.2
SPT	62.34	19.5%	3815.6
G-RTSLACK	57.94	24.0%	1202.1

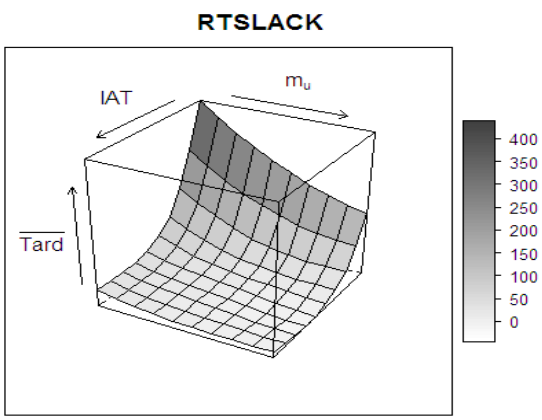

Figure 3: Wireframe showing mean tardiness as a function of jobs' inter-arrival times and due-date parameter m_u.

One hundred (100) different workload and demand configurations were generated with 1000 jobs each. The process times follow a normal distribution with a mean value and a deviation equal to 100 and 30 respectively. Inter-arrival times follow an exponential distribution. Due dates are generated according to Equation (18), following a uniform distribution ranging from 0 to 100, 200, …, 1000. In the second set of experiments, for instance, the mean inter-arrival time is equal to 110 and the due date parameter m_u ranges from 0 to 200. Each one of the 100 sets of experiments is replicated 50 times for each dispatch rule with different seed numbers in each replication for generating new streams of distribution values, in order for better confidence intervals of the results to be achieved. The average values of mean tardiness, fraction tardy and maximum tardiness for all 100 sets of the experiments, have been included in Table 2.

Figure 3 illustrates the way that mean tardiness is affected in the single machine problem instance, when the RTSLACK rule is used, as a function of the jobs' arrival rate and the due date parameter m_u, which indicates how tight the due date policy is.

In figures 4, 5 and 6, the performance of the new rule is compared against the performance of EDD, SLACK and SPT. While the SPT performs better when the due date policies are tight, the RTSLACK seems better in looser cases. As orders are reduced (and so is jobs' arrival rate), the RTSLACK performs equally well with the EDD and the SLACK, in terms of mean tardiness as in Figure 6.

Overall, the proposed rule seems to have a balanced performance, combining the efficiency of the SPT, the EDD and the SLACK rules

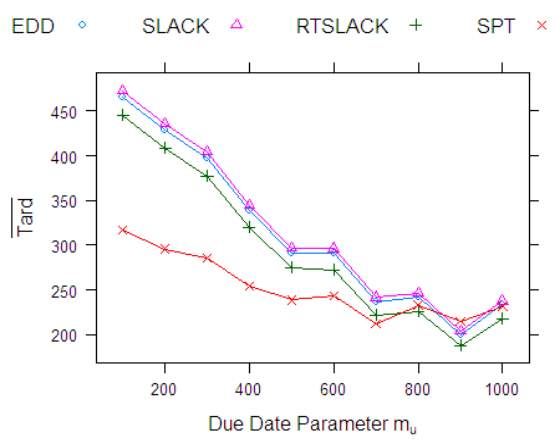

Figure 4: Tardiness for experiments sets 1-10

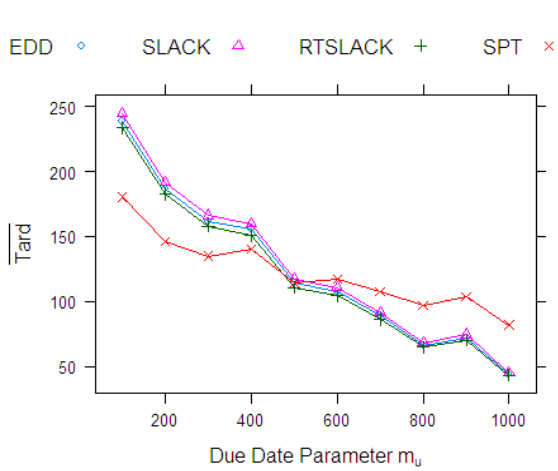

Figure 5: Mean tardiness for experiments sets 11-20

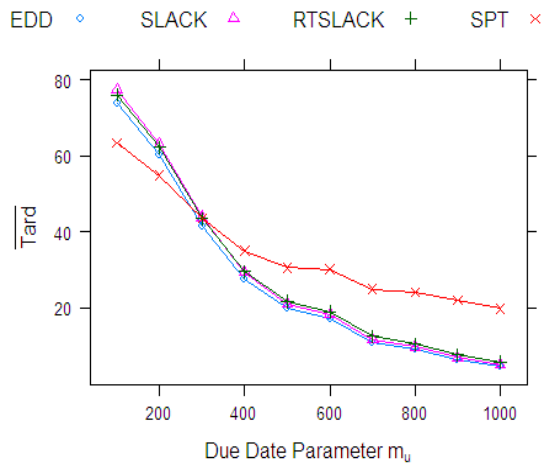

Figure 6: Mean tardiness for experiments sets 41-50

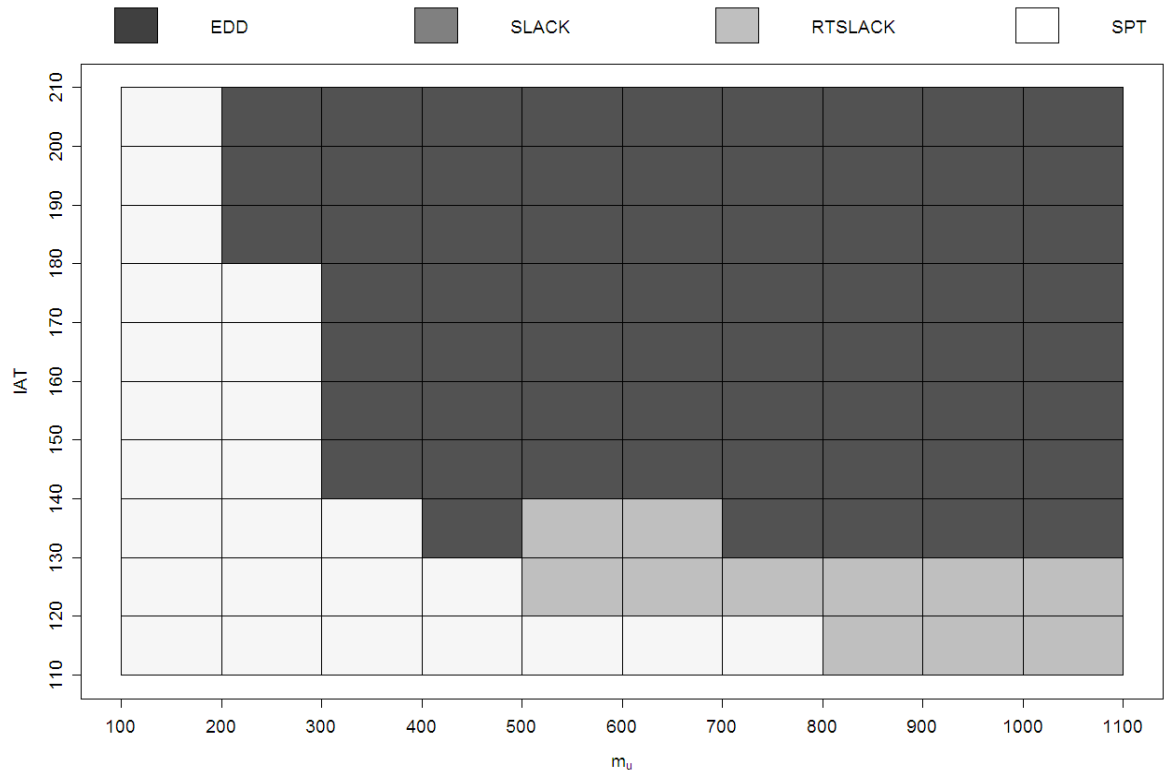

Figure 7: Dispatch rules exhibiting best mean tardiness performance

3.1.1 *Generic form of the RTSLACK rule*

For the same single machine problem configuration, the stochastic hill climbing algorithm (Figure 2) was used for identifying the functions b_j of equation (14) over three ranges of queue length n, as defined in Equations (15-17).

A quite important issue, when using this type of algorithms is the initialisation of the variables in

search, namely the initial values of z_{j1}, z_{j2}, z_{j3} and n_1, n_2 of Equations (15-17). While letting $z_{j1...3}$ be equal to 1.0 is equivalent to the original definition of the RTSLACK rule, identifying the initial values of n_1 and n_2 requires the better understanding of the way that the dispatch rules perform under different workload patterns and as a function of the queue length n. Due date policies play a less significant role in the determination of the queue length.

Figure 7 shows which rule seems to perform best in terms of mean tardiness for every workload pattern tested. While the SPT rule seems to be having the best performance when the workload is heavy (low inter-arrival times) or when the due date policy is tight (low m_u values), the RTSLACK is better when the workload is heavy and the due date policy is less tight; the EDD rule seems to be having a better performance in all other cases.

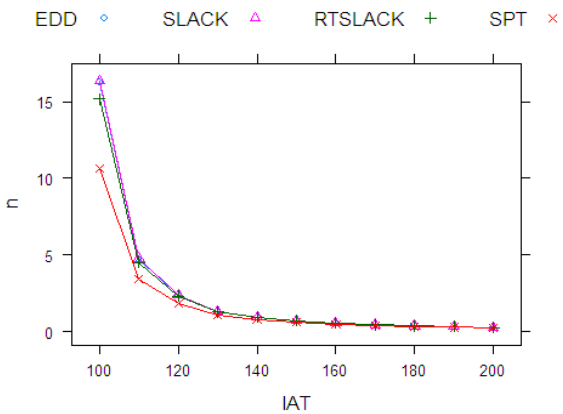

Figure 8: Average queue length per rule

Figure 8 shows the average queue length as a function of the mean inter-arrival time. In conjunction with the results shown in Figure 7, it may be assumed that when the average queue length n is high, which occurs when the inter-arrival time is also high, the SPT rule seems to perform at best. It is therefore anticipated that the variable b_{11} is higher than b_{12} and b_{13}. It is also apparent that $n_1 = 5$ and $n_2 = 3$ could be used as the initial values for dividing the queue length range into 3 categories, as per Equations (15-17), namely: $n > 5$, $3 < n \leq 5$, $n \leq 3$.

Assigning the initial value of all $b_{j1...3}$ equal to 1.0, as the original definition of the RTSLACK rule suggests, the stochastic hill climbing algorithm may be employed for achieving a more effective, in terms of tardiness, Equation (14). The step of the search for variables $b_{j1...3}$ is set equal to 0.2, while for the variables n_1 and n_2 it is set equal to 1.5.

Indeed, after the execution of the algorithm has been completed, having used as a termination criterion the reaching of 100 consecutive iterations, without improving the average tardiness achieved, the generic form of the proposed rule, G-RTSLACK, for every job i, is as follows:

$$PR_{k_i} = -\left[2.28(n-1)p_{k_i} - .27r_{k_i} + 1.00d_{k_i}\right], n > 6 \quad (21a)$$

$$PR_{k_i} = -\left[1.67(n-1)p_{k_i} - .26r_{k_i} + 1.00d_{k_i}\right], 4 < n \leq 6 \quad (b)$$

$$PR_{k_i} = -\left[.14(n-1)p_{k_i} + 2.75r_{k_i} + 1.00d_{k_i}\right], n \leq 4 \quad (c)$$

Figures 9 and 10, demonstrate that the generic form of the RTSLACK rule (G-RTSLACK) performs better compared with the original version of the RTSLACK rule. In particular, the improvement in absolute values and in terms of mean tardiness is larger in heavier workloads and in tighter due-date policies centres, WC_1, WC_2 and WC_3 with 3, 2, 4 identical, parallel resources in Figure 11. There are two different job types. For both job types, the process time of each task follows a normal distribution; the deviation is equal to the 30% of the mean. The inter-arrival time follows an exponential distribution for both job types. The due date of each job is given by Equation (18).

One hundred (100) different workload and demand configurations were generated with 1000 jobs each (Table 3). Every one of the 100 sets of experiments is replicated 50 times for each dispatch rule, applying the same dispatch policy to every work centre. The average values of mean tardiness, fraction tardy and maximum tardiness for all 100 sets of the experiments are included in Table 4.

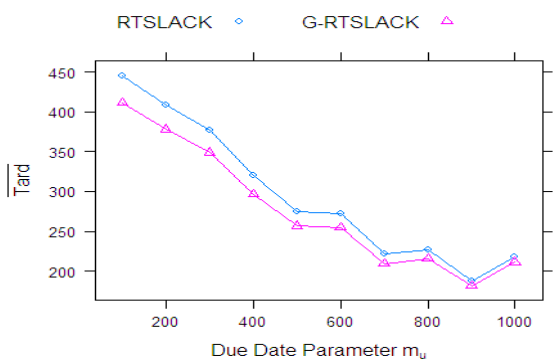

Figure 9: Tardiness for experiments sets 1-10

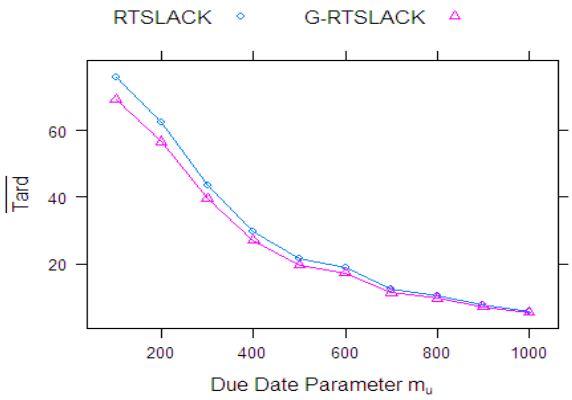

Figure 10: Tardiness for experiments sets 41-50

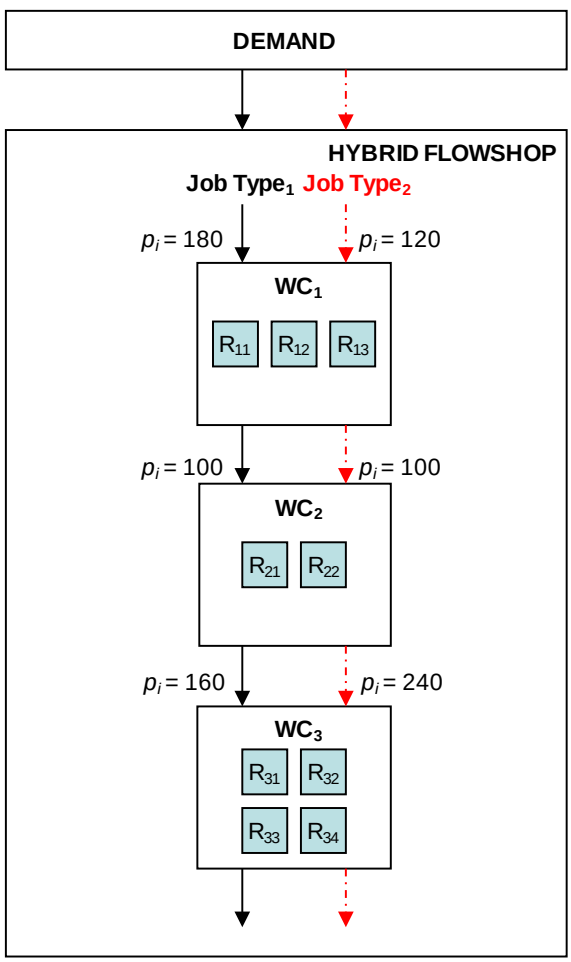

Figure 11: Hybrid flow shop test case

Table 3: Hybrid flow shop scheduling – experiments

Exp. Set #	Process Time as per Figure 6	Mean Inter-Arrival Time	Due Date Parameter m_u
1-10	normal distr.	expo(110)	0,100-1000
11-20	normal distr.	expo(120)	0,100-1000
21-30	normal distr.	expo(130)	0,100-1000
31-40	normal distr.	expo(140)	0,100-1000
41-50	normal distr.	expo(150)	0,100-1000
51-60	normal distr.	expo(160)	0,100-1000
61-70	normal distr.	expo(170)	0,100-1000
71-80	normal distr.	expo(180)	0,100-1000
81-91	normal distr.	expo(190)	0,100-1000
91-100	normal distr.	expo(200)	0,100-1000

The performance of the new rule is similar to that of the rule in the single machine problem instances. It seems that the proposed rule performs equally well, under different workload scenarios and due date policies Figure 12, 13 and 14.

Table 4: Hybrid flow shop scheduling rules performance – mean values

RULE	Mean Tardiness	Fraction Tardy	Max Tardiness
EDD	42.43	21.9%	418.9
SLACK	43.73	22.7%	383.9
RTSLACK	40.59	21.4%	778.1
SPT	42.30	18.8%	2004.2

The best performance for the proposed RTSLACK rule is observed in the cases that demand is higher and the due date policy is less tight.

4 Conclusion

This paper describes a new, simple scheduling policy suggesting a dispatch rule, which is based on maximizing the slack time of the remaining tasks in the manufacturing resources queues. Two variations of the rule, named RTSLACK, are proposed against three widely used dispatch rules, such as the SPT rule, the EDD rule, and the SLACK rule in a series of single machine and hybrid flow shop scheduling problem instances.

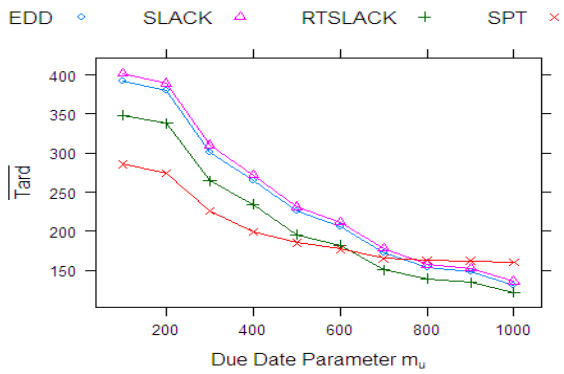

Figure 12: Mean tardiness for experiments sets 1-10

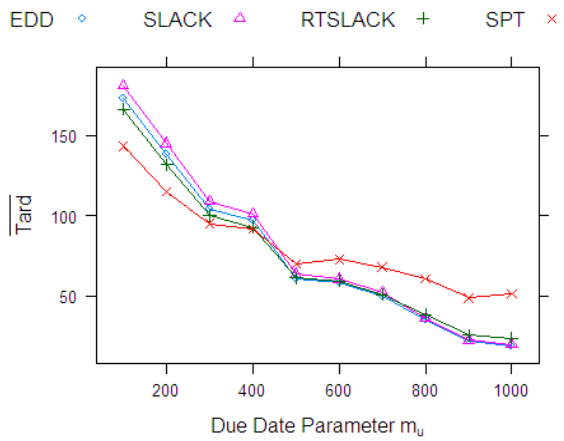

Figure 13: Mean tardiness for experiments sets 11-20

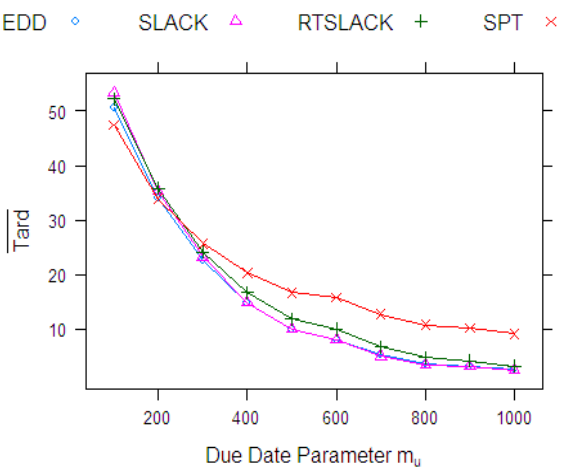

Figure 14: Mean tardiness for experiments sets 41-50

The first version changes the priority of each job, at each work centre, whenever the number of the waiting jobs is changed, whilst the second variation assigns a priority each time that the job enters a work centre's queue and remains the same until the job leaves the work centre.

The proposed rule seems to perform well under different workload scenarios and due date policies. It has the best performance when the demand is high 1and the due dates are not tight. It seems to combine the performance characteristics of the SPT, the EDD and the SLACK rules and it could therefore, constitute an interesting alternative to the use of these rules, especially in the case that different demand patterns and due date policies are applicable and the simultaneous use of more than one rules is difficult. It is interesting to note that the behaviour of the new rule changes, depending on the values of the process time, the due date and the length of the queue with the waiting jobs. The generic form of the proposed rule was also investigated with the aid of a stochastic hill climbing algorithm. Its application to the single machine scheduling problem revealed a promising potential, especially when the information, pertaining to the workload, is accurate enough, so that the functions b_j can be defined with the aid of a search algorithm. The extended application of this rule to other cases and systems as well as the investigation of other similar or more complex ways of combining the efficiency of different dispatch rules, on the basis of Equations (10) and (12-14), with the use of other search or optimisation approaches, are some of the ideas that could be further explored.

5 Acknowledgments

This work has been partially supported by the "K. Karatheodoris" grant from the University of Patras and the research project "MyCar", funded by the CEU.

References

[1] Chryssolouris G., 2006, *Manufacturing Systems - Theory and Practice*, 2nd edition, Springer-Verlag, New York.

[2] Haupt R., 1989, A Survey of Priority Rule-Based Scheduling, *OR Spectgrum*, 11(3): 3-16.

[3] Chiang T.C and Fu L.C., 2007, Using dispatching rules for job shop scheduling with

due date-based objectives, *International Journal of Production Research*, 45(14): 3245-3262.

[4] Kawai, T. and Fujimoto Y., 2005, Efficient Combination of Dispatch Rules for Job-shop Scheduling Problem, *Proceedings of the 3rd IEEE International Conference on Industrial Informatics*, 484-488.

[5] Chryssolouris G., Lee M. and Domroese M., 1991, The Use of Neural Networks in Determining Operational Policies for Manufacturing Systems, *Journal of Manufacturing Systems*, 10(2): 166-175.

[6] Jeong K.-C. and Kim Y.-D., 1998, A real-time scheduling mechanism for a flexible manufacturing system: using simulation and dispatching rules, *International Journal of Production Research*, 36(9): 2609-2626.

[7] Kim, M.-H, Kim, Y.-D., 1994, Simulation-based real-time scheduling mechanism in a flexible manufacturing system, Journal of Manufacturing Systems, 13(2): 85-93.

[8] Graves S.C., 1981, A review of production scheduling, Operations Research, 29(4): 646-675.

[9] Agliari A., Diligenti M. and Zavanella, L., 1995, Variable priority dispatching rules: An analytical approach, *International Journal of Production Economics*, 41: 51-58.

[10] Giannelos N., Papakostas N., Mourtzis D. and Chryssolouris G., 2007, Dispatching policy for manufacturing jobs and time-delay plots, *International Journal of Computer Integrated Manufacturing*, 20(4): 329-337.

[11] Valente, J.-M.S., 2007, Performance of the ATC dispatch rule by using workload data to determine the lookahead parameter value, *International Journal of Production Economics*, 106: 563-573.

[12] Ronconi D.P. and Henriques L.-R.S., 2009, Some heuristic algorithms for total tardiness minimization in a flowshop with blocking, *Omega*, 37: 272-281.

[13] DESMO-J Development Team, 2008, The DESMO-J Simulation Framework, version 2.4.1b, Available online via http://www.desmoj.de.

[14] R Development Core Team, 2008, R: A Language and Environment for Statistical, R Foundation for Statistical Computing, Vienna, Austria, ISBN 3-900051-07-0, http://www.R-project.org.