Approximation algorithms for no-idle time scheduling on a single machine with release times and delivery times

Imed Kacem
LITA, University of Metz, France
kacem@univ-metz.fr

Hans Kellerer
ISOR, University of Graz, Austria
hans.kellerer@uni-graz.at

Abstract

This paper is the first attempt to successfully design efficient approximation algorithms for the single-machine maximum lateness minimization problem when jobs have different release dates and tails (or delivery times) under the no-idle time assumption (i.e., the schedule cannot contain any idle-time between two consecutive jobs on the machine). Our work is motivated by interesting industrial applications to the production area (Chrétienne [3]). Our analysis shows that modifications of the classical algorithms of Potts and Schrage can lead to the same worst-case performance ratios obtained for the relaxed problem without the no-idle time constraint. Then, we extend the result developed by Mastrolilli [13] for such a relaxed problem and we propose a polynomial time approximation scheme with efficient time complexity.

1 Introduction

We have a set J of n jobs $J = \{1, 2, ..., n\}$. Every job j has a processing time p_j, a release date r_j and a tail (delivery time) q_j. The jobs have to be performed on a single machine under the no-idle time scenario, i.e., the schedule should consist of a single block of jobs (non idle time between the jobs). The machine can perform only one job at a given time. Preemption is not allowed. The objective is to minimize the maximum lateness:

$$L_{max} = \max_{1 \leq j \leq n} \{C_j + q_j\} \quad (1)$$

where C_j is the completion time of job j.

The studied problem is denoted by `P` and it can be represented by $1, NI|r_j, q_j|L_{max}$ according to the classical 3-field notation. It consists of a generalization of the well-known problem $1|r_j, q_j|L_{max}$ already widely studied in the literature. In the remainder of this paper, the relaxed problem $1|r_j, q_j|L_{max}$ without no-idle time constraint is denoted by `P'`. According to Lenstra et al [12] problem `P'` is NP-Hard in the strong sense. Since the NP-hardness example in [12] contains no idle time, problem P is also NP-Hard in the strong sense. Given the aim of this paper we give a short review on the two mentioned problems.

The unconstrained version `P'` has been intensively studied. For instance, most of the exact algorithms are based on enumeration techniques. See for instance the papers by Dessouky and Margenthaler [4], Baker and Su [1], McMahon and Florian [14], Carlier et al [2], Larson et al [11] and Grabowski et al [5]. Various approximation algorithms were also proposed. Most of these algorithms are based on variations of the extended Schrage rule [17]. The Schrage rule consists of scheduling ready jobs (available jobs) on the machine by giving priority to one having the greatest tail. It is well-known that the Schrage sequence yields a worst-case performance ratio of 2. This was first observed by Kise et al [10]. Potts [16] improves this result by running the Schrage algorithm at most n times to slightly varied instances. The algorithm of Potts has a worst-case performance ratio of $\frac{3}{2}$ and it runs in $O(n^2 \log n)$ time. Hall and Shmoys [6] showed that by modifying the tails the algorithm of Potts has the same worst-case performance ratio under precedence constraints. Nowicki and Smutnicki [15] proposed a faster $\frac{3}{2}$-approximation algorithm with $O(n \log n)$ running time. By performing the algorithm of Potts for the original and the inverse problem (i.e., in which release dates are replaced by tails, and vice-versa) and taking the best solution Hall and Shmoys [6] established the existence of a $\frac{4}{3}$-approximation. They also proposed two polynomial time approximation schemes (PTAS). The first algorithm is based on a dynamic programming algorithm when there are only a constant number of release dates. The second algorithm distinguishes large and small jobs with a constant number of large jobs. A more effective PTAS has been proposed by Mastrolilli [13] for the single-machine and parallel-machine cases. For more details on lateness problems the reader is invited to consult the survey papers by Hall [7] and Kellerer [9].

Some works exist also on applications of the no-idle time scenario (see for instance the paper by Irani and Pruhs [8] related to the power management policies). In a recent paper, Chrétienne [3] mentioned several practical motivations to consider the no-idle time scenario. In particular, it may be very expensive to stop the machine and restart the production after. Chrétienne

[3] mentioned the applications where we need to use the machine at a high temperature. In such a case, the no-idle time scenario allows to make significant savings by avoiding the setup costs. Note that general useful properties have been also proposed by Chrétienne, who gave an interesting study on some aspects of the impact of the no-idle time constraint on the complexity of a set of single-machine scheduling problems. An exact method has been also proposed by Carlier et al [2] who have elaborated an extended branch-and-bound algorithm for solving the studied problem. Despite the interest to consider such an assumption, there are few papers dealing with our problem. To the best of our knowledge there is no approximation algorithm for problem P. Thus, this paper is a first attempt to successfully design new approximation algorithms for this fundamental no-idle time scheduling problem.

This paper is organized as follows. Section 2 shows that, subject to some adaptations, some classical heuristics (Schrage algorithm and Potts algorithm) keep their worst-case performance ratio under the no-idle time scenario. In Section 3 the existence of a PTAS is proven. Finally, Section 4 concludes the paper.

2 Worst-case analysis of classical rules

2.1 Increasing the release dates

Let us consider the following generalized list scheduling algorithm (GLS): Whenever the machine becomes available, schedule the first available job in the list. Thus, algorithm Schrage is a GLS. Let $\underline{C}$ denote the makespan obtained by using a GLS for P'. Obviously, $\underline{C}$ is a lower bound on the makespan for any feasible schedule for P. Hence, the following relation holds for every $j \in J$:

$$S_j \geq \underline{C} - \sum_{j \in J} p_j, \tag{2}$$

where S_j is the starting time of job j in a feasible solution for problem P.

From (2) it can be deduced that release times can be increased without modifying the optimal solution:

$$r_j := \max \left\{ r_j, \underline{C} - \sum_{j \in J} p_j \right\}. \tag{3}$$

This useful property was also reported in Chrétienne [3] and Carlier et al [2].

In the remainder of this paper TRANSFORM denotes the procedure that consists of calculating $\underline{C}$ and updating the release dates for problem P according to (3). The following lemma is the basis of the modifications of our classical heuristics for problem P' applied to problem P:

Lemma 1 *After applying TRANSFORM the optimal solution for problem P does not change and GLS yields a solution without idle time.*

2.2 Folklore

Now we return our attention to two classical heuristics already proposed for problem P' by Schrage and Potts. For self-consistency we recall the principles of these heuristics and some important results on the relaxed problem problem problem P'.
First, we recall the principle of the Schrage algorithm. It consists of scheduling the job with the greatest tail from the available jobs at each step. At the completion of such a job, the subset of the available jobs is updated and a new job is selected. The procedure is repeated until all jobs are scheduled. Assume that the jobs are reindexed such that Schrage yields the sequence $\sigma = (1, \ldots, n)$. The job c which attains the maximum lateness in the Schrage schedule, is called the *critical job*. Then the maximum lateness of σ can be defined as follows:

$$L^{\sigma}_{max} = \min_{j \in B}\{r_j\} + \sum_{j \in B} p_j + q_c = r_a + \sum_{j=1}^{c} p_j + q_c \tag{4}$$

where job a is the first job so that there is no idle time between the processing of jobs a and c, i.e. either there is idle time before a or a is the first job to be scheduled. The sequence of jobs $a, a+1, \ldots, c$ is called the *critical path* (or the critical block B) in the Schrage schedule. It is obvious that all jobs j in the critical path have release dates $r_j \geq r_a$. If c has the smallest tail in B, then sequence σ is optimal. Otherwise, there exists an *interference job* $b \in B$ such that

$$q_b < q_c \text{ and } q_j \geq q_c \text{ for all } j \in \{b+1, b+2, ..., c-1\}. \tag{5}$$

Moreover, the following relations holds:

$$L^{\sigma}_{max} - L^{*}_{max} < p_b \tag{6}$$

$$L^{\sigma}_{max} - L^{*}_{max} < q_c \tag{7}$$

where L^{*}_{max} is the optimal maximum lateness (see, e.g., Kise [10]).

Finally, we recall the following useful lower bound, valid for every subset $F \subset J$:

$$L^*_{max} \geq \min_{j \in F}\{r_j\} + \sum_{j \in F} p_j + \min_{j \in F}\{q_j\}. \tag{8}$$

2.3 Constant approximations

Let us call `MSchrage` the algorithm defined for problem `P` as follows. First, we apply procedure `TRANSFORM` to increase the release dates as given in Equation (3). Then, we apply the Schrage algorithm to the modified instance. From Lemma 1 we can immediately conclude:

Theorem 1 *Algorithm `MSchrage` has a tight worst-case performance ratio of* 2 *for problem `P`.*

To improve the performance of the Schrage algorithm for the relaxed problem `P'` (without no-idle time constraint) Potts proposed to run this algorithm at most n times to some modified instances (Potts [16]). He starts with the Schrage sequence. If there is an interference job b, then it is forced to be scheduled after the critical job c in the next iteration by setting $r_b := r_c$. Then, another Schrage sequence is computed on the modified instance. The procedure is reiterated until no interference job is found or n sequences have been constructed. Potts proved that his algorithm provides at least a sequence with a worst-case performance ratio of $\frac{3}{2}$ for problem `P'`.

For the original problem `P` we can extend the result obtained by Potts. Let `NI-P` be the extension of the Potts algorithm by combining it with procedure `TRANSFORM`. It can be summarized as follows.

`NI-P` algorithm

(i). $k := 0$; $I := (r_j, p_j, q_j)_{1 \leq j \leq n}$.

(ii). Update instance I by applying procedure `TRANSFORM`.

Apply the Schrage algorithm to I and store the obtained schedule σ^k.

Set $k := k + 1$.

(iii). If $k = n$ or if there is no interference job in σ^{k-1}, then `stop` and `return` the best generated schedule among σ^0, σ^1,..., σ^{k-1}.

Otherwise, identify the interference job b and the critical job c in σ^{k-1}. Set $r_b := r_c$ and `go to` step (ii).

Theorem 2 *Algorithm NI-P has a tight worst-case performance ratio of* $\frac{3}{2}$ *for problem P.*

Proof. The proof is quite similar to the proof for the worst-case performance of the Potts algorithm, but we repeat it for the sake of completeness. Nevertheless, NI-P and Potts may yield different outputs as it is illustrated in the example below.

Let us consider the first schedule σ^0 generated by Algorithm NI-P. If there is no interference job, then σ^0 is optimal. Otherwise, if $q_c \leq \frac{L^*_{max}}{2}$ or $p_b \leq \frac{L^*_{max}}{2}$, then σ^0 is a $\frac{3}{2}$-approximation. Therefore, we can restrict our analysis to the case where

$$q_c > \frac{L^*_{max}}{2} \text{ and } p_b > \frac{L^*_{max}}{2} \tag{9}$$

It follows that job b must be scheduled after c in the optimal schedule. Otherwise,

$$L^*_{max} \geq r_b + p_b + p_c + q_c > r_b + p_c + L^*_{max}, \tag{10}$$

which leads to a contradiction.

By imposing $r_b := r_c$ in the next iteration and after applying procedure TRANSFORM according to Lemma 1 the optimal maximum lateness will not increase and the new obtained schedule has no idle time.

At iteration $k = 1$, we obtain again a new Schrage sequence σ^1 for the modified instance I and the analysis is the same; either we have a $\frac{3}{2}$-approximation or the update of r_b is coherent with the optimal solution.

At the end of the procedure, if we stop because there is no interference job, then we are guaranteed a $\frac{3}{2}$-approximation ratio. Otherwise, we obtain a new interference job $b' \neq b$, since b cannot be an interference job more than $(n-1)$ times. In this case, from (9) we deduce that

$$p_{b'} < \frac{L^*_{max}}{2}. \tag{11}$$

Hence, from Equation (6) the last sequence yields a $\frac{3}{2}$-approximation ratio.

To prove the tightness of the bound and to illustrate the difference between Potts and NI-P algorithms, we prefer to recall the last example in [16]. In this example, given a very large number T, we have three jobs to schedule such that $r_1 = 0$, $r_2 = 1$, $r_3 = \frac{T+1}{2}$, $p_1 = \frac{T-1}{2}$, $p_2 = \frac{T-1}{2}$, $p_3 = 1$, $q_1 = 0$, $q_2 = \frac{T-3}{2}$ and $q_3 = \frac{T-1}{2}$. Algorithm NI-P yields three schedules σ^0, σ^1 and σ^2 as it is depicted in Figure 1. For this example, the optimal maximum lateness $L^*_{max} = T + 1$ can be obtained for sequence $(2, 3, 1)$ whereas the

algorithm gives a maximum lateness of $\frac{3T-1}{2}$. Hence, the worst-case performance ratio can be close to $\frac{3}{2}$ when $T \to +\infty$. Contrary to NI-P, the Potts algorithm yields two sequences $(1,2,3)$ and $(1,3,2)$ for this instance when considering problem P'.

■

PLEASE INSERT FIGURE 1 HERE
FIG 1. Tightness of the bound given by the NI-P algorithm

3 Existence of a PTAS

In this section we show the existence of a PTAS for problem P. We recall that a PTAS is an algorithm that yields for a given $\epsilon > 0$ a $(1+\epsilon)$-approximation such that the time complexity is polynomial when ϵ is fixed. It is well-known that the relaxed problem P' admits a PTAS. As mentioned in Section 1 several algorithms have been published and the most effective of them has been recently developed by Mastrolilli [13], who proposed the clever idea to cluster jobs into subsets of equivalent release dates and tails. He also demonstrated that for each subset one can consider the jobs as large except a small constant number of them. The corresponding transformations have a small impact on the optimal objective function and the best solution of the modified instance can be obtained in a polynomial time in n when ϵ is fixed. The main result of this section is, using some ideas of Mastrolilli, to show how this last result can be extended when the no-idle time constraint is imposed.

First, given an instance I of problem P define $L^*_{max}(I)$ as the optimal maximum lateness for I and $L^H_{max}(I)$ the result of the MSchrage heuristic. By dividing all data by $\frac{L^H_{max}}{2}$ and using the fact that the MSchrage sequence σ' yields a 2-approximation, we may assume w.l.o.g. that

$$1 \leq L^*_{max}(I) \leq 2. \tag{12}$$

This implies that for every $j \in J$ we have $0 \leq r_j \leq 2$, $0 \leq p_j \leq 2$ and $0 \leq q_j \leq 2$.

Theorem 3 *For a given $\epsilon > 0$ and for every instance I there is an instance I_2 with the following properties:*
(i) I_2 can be constructed in polynomial time and has a constant number of jobs when ϵ is fixed.
(ii) The optimal maximum lateness $L^*_{max}(I_2)$ is not too far away from $L^*_{max}(I)$.

Proof. First, construct an instance I_1 by rounding down all release dates and tails to the next multiple of ϵ. It follows that I_1 has at most $(1+\frac{2}{\epsilon})$ different release dates and $(1+\frac{2}{\epsilon})$ different tails. Moreover, by rounding down these data, we have $L^*_{max}(I_1) \leq L^*_{max}(I)$. Second, put all the jobs with processing times less or equal to $\frac{\epsilon}{2}$ and having the same release date and the same tail into classes Ω_1, Ω_2,..., Ω_l where $l \leq \frac{9}{\epsilon^2}$.

Create a new instance I_2 by greedily merging jobs from the same class until the job size is greater than $\frac{\epsilon}{2}$. Therefore, all the new created jobs have processing times between $\frac{\epsilon}{2}$ and ϵ. From each class at most one job with processing time $< \frac{\epsilon}{2}$ remains. It can be easily seen that instance I_2 has only a constant number of jobs and can be constructed in polynomial time. Hence, the first part (i) of this theorem is proven.

Let Ψ be the set of all the jobs having processing times $> \epsilon$ and denote by S^*_j the optimal starting time of $j \in \Psi$ for instance I_1. Now construct instances I'_1 and I'_2, respectively by setting for all $j \in \Psi$:

$$\tilde{r}_j := S^*_j, \tilde{p}_j := p_j, \tilde{q}_j := L^*_{max}(I_1) - p_j - S^*_j.$$

Note that Ψ is the same set in I'_1 and I'_2 and it contains no merged jobs. Obviously, the following relation holds:

$$L^*_{max}(I'_1) = L^*_{max}(I_1) \tag{13}$$

Consider now I_2 and I'_2. For jobs $\notin \Psi$ nothing is changing. For jobs $\in \Psi$ we have $\tilde{r}_j \geq r_j$, $\tilde{p}_j = p_j$, $\tilde{q}_j \geq q_j$. Thus,

$$L^*_{max}(I_2) \leq L^*_{max}(I'_2). \tag{14}$$

Moreover, from (8) it follows that for each $\Omega \subseteq I'_1$ we have

$$L^*_{max}(I'_1) \geq \min_{j\in\Omega}\{r_j\} + \sum_{j\in\Omega} p_j + \min_{j\in\Omega}\{q_j\}. \tag{15}$$

The only difference between instances I'_1 and I'_2 is, that some jobs of I'_1 are merged in I'_2. Thus, for any set $\Omega' \subseteq I'_2$ there exists a set $\Omega \subseteq I'_1$ such that

$$\min_{j\in\Omega'}\{r_j\} + \sum_{j\in\Omega'} p_j + \min_{j\in\Omega'}\{q_j\} = \min_{j\in\Omega}\{r_j\} + \sum_{j\in\Omega} p_j + \min_{j\in\Omega}\{q_j\}. \tag{16}$$

Apply Algorithm `MSchrage` to instance I'_2 and consider the critical sequence. Let c denote the critical job, b the interference job and Λ_b the jobs processed after b until c. Let $L^{\sigma'}_{max}(I'_2)$ denote the maximum lateness for I'_2

using Algorithm `MSchrage`. Hence, $L^{\sigma'}_{max}(I'_2) = S_b + p_b + \sum_{j\in\Lambda_b} p_j + q_c$. By definition, the following two relations hold:

$$S_b < \min_{j\in\Lambda_b}\{r_j\} \tag{17}$$

and

$$q_c = \min_{j\in\Lambda_b}\{q_j\}. \tag{18}$$

We conclude from (17) and (18) that $L^*_{max}(I'_2) \leq L^{\sigma'}_{max}(I'_2) < \min_{j\in\Lambda_b}\{r_j\} + p_b + \sum_{j\in\Lambda_b} p_j + \min_{j\in\Lambda_b}\{q_j\}$. Therefore, by using (15) and (16) it can be deduced that $L^*_{max}(I'_2) \leq L^*_{max}(I'_1) + p_b$. Hence, by (13) and (14) the following relation holds

$$L^*_{max}(I_2) \leq L^*_{max}(I_1) + p_b. \tag{19}$$

Note that the last inequality becomes $L^*_{max}(I_2) \leq L^*_{max}(I_1)$ if there is no interference job.

Assume now that $p_b > \epsilon$ in I'_2. This implies that $b \in \Psi$ and $r_c > r_b = S^*_b$. Thus, either job c or, if c was merged in I'_2, a job with the same release time and tail as c is processed after job b in the optimal solution for I_1. One can deduce that $S^*_c \geq S^*_b + p_b$. By definition, $q_b < q_c$ and we know that $q_b = L^*_{max}(I_1) - p_b - S^*_b$ and $q_c \leq L^*_{max}(I_1) - p_c - S^*_c$. Hence, $q_b - q_c \geq p_c + S^*_c - p_b - S^*_b > p_c > 0$ which leads to a contradiction. In conclusion, if b exists then

$$p_b \leq \epsilon. \tag{20}$$

Recall that we have

$$L^*_{max}(I_1) \leq L^*_{max}(I). \tag{21}$$

As a consequence of relations (19), (20) and (21), the following inequality can be deduced

$$L^*_{max}(I_2) \leq L^*_{max}(I) + \epsilon, \tag{22}$$

and from (22) the second part (ii) of this theorem is verified. ■

Now, we are ready to introduce our PTAS for problem `P`. It can be summarized as follows.

`NI-PTAS` algorithm

(i). Construct instance I_2 according the rounding down and merging procedures mentioned in the proof of Theorem 3.

(ii). Determine all the possible sequences for instance I_2 and choose the best no-idle time schedule produced by `TRANSFORM`.

(iii). Create a feasible schedule for instance I by moving all the jobs to the right by at most ϵ.

The next theorem establishes the existence of a PTAS for problem `P`.

Theorem 4 *Algorithm `NI-PTAS` is a PTAS for problem `P`.*

Proof. The running time is polynomial by part (i) of Theorem 3 and since there are only a constant number of sequences in instance I_2. Indeed, it can be observed that the number of jobs cannot be more than $\frac{4}{\epsilon}+\frac{9}{\epsilon^2}$. Hence, the number of sequences generated in Step (ii) of Algorithm `NI-PTAS` cannot be more than $\lfloor \frac{4}{\epsilon}+\frac{9}{\epsilon^2} \rfloor !$ which can be considered as equivalent to $(\frac{1}{\epsilon})^{O(\frac{1}{\epsilon^2})}$. The time complexity of Step (ii) of Algorithm `NI-PTAS` remains equivalent to $(\frac{1}{\epsilon})^{O(\frac{1}{\epsilon^2})}$ since the construction of any sequence can be done in $O(\frac{1}{\epsilon^2})$ time. It is also obvious to see that Algorithm `MSchrage` can be implemented in $O(n \log n)$ time.
Moreover, the accuracy is good enough and by reconstructing a solution for I we add at most 2ϵ to the maximum lateness (a loss of ϵ by moving jobs to the right and a loss of ϵ by increasing the tails). ■

4 Conclusion

In this paper, we aimed at designing efficient approximation algorithms to minimize maximum lateness on a single machine under the no-idle time scenario. In the studied problem, jobs have different release dates and tails. In a first step, we showed that the Schrage sequence can lead to a worst-case performance bound of 2. Hence, we studied the Potts modified sequence in order to obtain a $\frac{3}{2}$-approximation algorithm. Finally, based on modification of the input we showed the existence of a PTAS for the studied problem.
As a perspective of our work, the extension of our algorithms to minimize other criteria seems to be a very interesting study (for example, the weighted completion time).

Acknowledgements
This work has been supported by the Conseil Régional Champagne-Ardenne and carried out at the Université de Technologie de Troyes (ICD Laboratory, OSI Team).

References

[1] Baker KR, Su ZS (1974) Sequencing with due-dates and early start times to minimize maximum tardiness. *Naval Research Logistics Quarterly* 21: 171-176

[2] Carlier J, Moukrim A, Hermes F, Ghedira K (2010) Exact resolution of the one-machine sequencing problem with no machine idle time. *Computers and Industrial Engineering* 59(2): 193-199

[3] Chrétienne Ph (2008) On single-machine scheduling without intermediate delays. *Discrete Applied Mathematics* 156(13): 2543-2550

[4] Dessouky MI, Margenthaler CR (1972) The one-machine sequencing problem with early starts and due dates. *AIIE Transactions* 4(3): 214-222

[5] Grabowski J, Nowicki E, Zdrzalka S (1986) A block approach for single-machine scheduling with release dates and due dates. *European Journal of Operational Research* 26: 278-285

[6] Hall LA, Shmoys DB (1992) Jacksons rule for single machine scheduling: making a good heuristic better. *Mathematics of Operations Research* 17: 22-35

[7] Hall L (1997) Approximation algorithms for scheduling. In D.Hochbaum, (Ed) *Approximation Algorithms for NP-hard Problems*, PWS Publishing Co. 1-45

[8] Irani S, Pruhs K (2005) Algorithmic Problems in Power Management. *ACM Press*, New York, USA 36: 63-76

[9] Kellerer H (2004) Minimizing the Maximum Lateness. In J Y-T Leung (Ed) *Handbook of Scheduling: Algorithms, Models, and Performance Analysis* 10: 185-196

[10] Kise H, Ibaraki T, Mine H (1979) Performance analysis of six approximation algorithms for the one-machine maximum lateness scheduling problem with ready times. *Journal of the Operational Research Society of Japan* 22: 205-224

[11] Larson RE, Dessouky MI, Devor RE (1985) A forward-backward procedure for the single machine problem to minimize the maximum lateness. *IIE Transactions* 17: 252-260

[12] Lenstra JK, Rinnooy Kan AHJ, Brucker P (1977) Complexity of machine scheduling problems. *Annals of Operations Research* 1: 342-362

[13] Mastrolilli M (2003) Efficient Approximation Schemes for Scheduling Problems with Release Dates and Delivery Times. *Journal of Scheduling* 6(6): 521-531

[14] McMahon GB, Florian M (1975) On scheduling with ready times and due dates to minimize maximum lateness. *Operations Research* 23:475-482

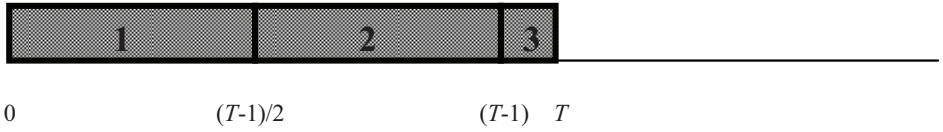
[15] Nowicki E, Smutnicki C (1994) An approximation algorithm for single-machine scheduling with release times and delivery times. *Discrete Applied Mathematics* 48: 69-79

[16] Potts CN (1980) Analysis of a heuristic for one machine sequencing with release dates and delivery times. *Operations Research* 28: 1436-1441

[17] Schrage L (1971) Obtaining optimal solutions to resource constrained network scheduling problems. *Unpublished manuscript*

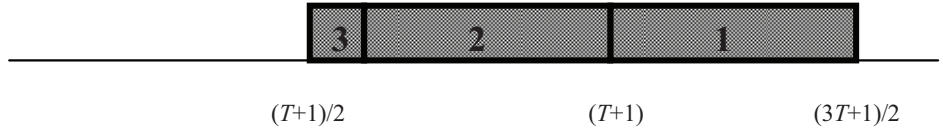
Figure

σ^0 schedule



σ^1 schedule

σ^2 schedule



Optimal schedule