Extending Smith's Rule with Task Mandatory Parts and Release Dates

Camille Bonnin, Margaux Nattaf, Arnaud Malapert, Marie-Laure Espinouse

▶ **To cite this version:**

Camille Bonnin, Margaux Nattaf, Arnaud Malapert, Marie-Laure Espinouse. Extending Smith's Rule with Task Mandatory Parts and Release Dates. PMS 2022 (18th International Workshop on Project Management and Scheduling), Apr 2022, Ghent, Belgium. hal-03641789

HAL Id: hal-03641789

https://hal.archives-ouvertes.fr/hal-03641789

Submitted on 14 Apr 2022

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Extending Smith's Rule with Task Mandatory Parts and Release Dates

Bonnin Camille[1,2], Nattaf Margaux[1], Malapert Arnaud[2] and Espinouse Marie-Laure[1]

[1] Univ. Grenoble Alpes, CNRS, Grenoble INP**, G-SCOP, 38000 Grenoble, France
camille.bonnin@grenoble-inp.fr, marie-laure.espinouse@grenoble-inp.fr, margaux.nattaf@grenoble-inp.fr
[2] Université Côte d'Azur, CNRS, I3S, France
arnaud.malapert@univ-cotedazur.fr

Keywords: 1-Machine Scheduling, Flow Time, Mandatory Parts, Release Dates, Preemption.

1 Context and problem description

Smith's rule is well-known in scheduling for solving the one-machine problem in which the flow time is minimized, i.e. $1||\sum C_j$. A first extension of this rule including release date and allowing preemption has been developed in Brucker P. (2006). This rule is called the Modified Smith Rule (MSR). This paper aims to extend MSR to include the notion of task mandatory part, a key concept in Constraint Programming (CP). This extension aims at computing lower bounds for the more general problem $1|r_j;d_j|\sum C_j$ and to use it to improve the solving of this problem by CP.

CP is, currently, a well established method to solve scheduling problems as shown in Baptiste P. et al. (2001). However, constraint programming techniques are oriented toward feasibility and makespan minimization. Recently, some efforts have been made to integrate other objective functions such as the maximum lateness, or the weighted flowtime. Recently, a lot of efforts have been made to integrate several objective functions such as the maximum lateness, or the weighted flow time like Kovács A. and Beck J. C. (2011). Still, few of them consider the flow time as objective.

One crucial notion in CP is the task mandatory part. A Mandatory Part (MP) is the time interval in which the non-preemptive task is executed in each feasible solution of the problem. It is defined as the intersection between the time interval where the task starts at its release date and the time interval where the tasks ends at its deadline. In this intersection (if it is not empty), the task is sure to be executed in each feasible solution of the problem. Typically, in CP, task time-windows narrow during the solving. Thus, the MP size increases.

A key concept in CP is filtering. Filtering consists in removing value from the domain of a variable that will not lead to feasible solutions. In practice, this process is used very often during the solving procedure. Thus, the algorithm used in the filtering process must be as quick as possible. As shown in Nattaf M. and Malapert A. (2020), it is possible to use lower bounds to design filtering algorithms. Therefore, it is interesting to compute lower bounds for $1|r_j;d_j|\sum C_j$ in polynomial time. Since it is an $\mathcal{NP}$-hard problem, we cannot find a polynomial algorithm to solve it. Thus, to find efficient lower bounds for $1|r_j;d_j|\sum C_j$, we need to consider its polynomial relaxations. Among those relaxations, only two can be solved in polynomial time, $1|r_j;pmtm|\sum C_j$ and $1|sp-graph|\sum w_jC_j$. In this paper, the first relaxation is considered. Then, to improve the lower bounds for $1|r_j;d_j|\sum C_j$ a new constraint is added to the relaxed problem: the mandatory part constraint (MP). This constraint ensures that the MPs of the tasks are respected.

** Institute of Engineering Univ. Grenoble Alpes

2 Modified Smith's Rule with Mandatory Parts (MSRMP)

There already exist algorithms that compute a lower bound for $1|r_j;d_j|\sum C_j$ in polynomial time. One of those algorithms is the modified Smith rule given by Brucker P. (2006) that solves $1|r_j;pmtn|\sum C_j$. This rule is an adaptation of the Smith rule and states that at each release date and at each competition time (C_j) of a task, the available task with the shortest remaining processing time must be scheduled (a task can then be interrupted by the release of a quicker task).

In this section, we are describing a rule to find the optimal solution of a relaxation of $1|r_j;d_j|\sum C_j$, $1|r_j;pmtn;MP|\sum C_j$. This rule is based on the modified Smith rule in which we are adding the execution of the mandatory parts (MP) of the tasks. It is called the Modified Smith's Rule with Mandatory Parts (MSRMP).

Rule 1 (Modified Smith's Rule with Mandatory Parts (MSRMP)) *Schedule the mandatory parts (MPs) in the correct time slots. At each release date or completion time of a task, schedule an unfinished task that is available and has the shortest remaining processing time. In case of equality, schedule the unfinished task without MP or with the earliest MP.*

Example 1.
Table 1 and Figure 1 present an example of the application of the MSRMP with six tasks. The MPs are in gray. To execute this rule, we begin by computing and scheduling the MPs and then, we apply the modified Smith rule on the rest by taking care of the equality cases. We can notice that even if tasks 3 and 5 are finished on time, task 1 is too long and finishes late, a long time after its MP.

Table 1: Example the application of the MSRMP - instance-

j	1	2	3	4	5	6
$\mathbf{p_j}$	9	5	2	3	6	3
$\mathbf{r_j}$	0	0	2	9	9	14
$\mathbf{d_j}$	15	∞	4	∞	17	∞

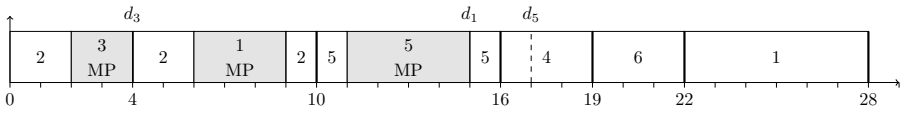


Fig. 1: Example of application of the MSRMP

Proposition 1 (Optimality of the Modified Smith Rule with Mandatory Parts). *The Modified Smith Rule with Mandatory Parts gives an optimal solution to the* $1|r_j;pmtn;MP|\sum C_j$ *problem.*

Proof. Clearly, the rule gives a feasible solution to the $1|r_j;pmtn;MP|\sum C_j$ problem.

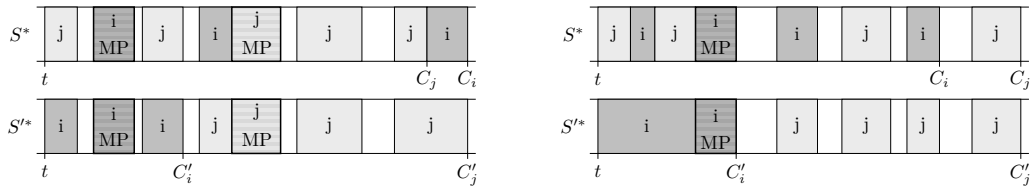
The proof relies on an exchange argument and is similar to the one of Theorem 4.9 in Brucker P. (2006). Therefore, we only present the exchange argument which is different here.

Let us assume that for the same feasible instance of $1|r_j;pmtn;MP|\sum C_j$, S is the schedule obtained with MSRMP and S^*, an optimal one. Those two schedules are identical until time t, where task i is scheduled in S, and task j in S^*. We then construct a new schedule S'^* based on S^* by re-scheduling, after t, the non-mandatory parts of i before those of j. As $r_j \leq t$ and $r_i \leq t$, S'^* is a feasible schedule. Lemma 1 shows that S'^* is still optimum. We then take S'^* as S^*, update t and redo the same reasoning until S'^* and S are the same. We then have proven the optimality of S.

Lemma 1 (Optimality of the exchange argument) *Let assume S^* and S'^* are constructed as in the proof of Proposition 1, the objective value of S'^* is no greater than those of S^*.*

We define $\widehat{p_k}$ as the remaining processing time of a task k, p_k its processing times, C_k and C'_k its completion time respectively in S^* and S'^*, and $[lst_k, eft_k[= [d_k - p_k, r_k + p_k[$ its MP.

Proof. We need to consider three cases based on the presence and the position of the MPs of i and j after t. We can notice that by the definition of an MP, there are executed at simultaneously in S^* and S'^* and so cannot be executed at time t.



(a) Case 2 (b) Case 3

Fig. 2: Examples of the cases 2 and 3 of the proof

Case 1 : neither i nor j has an MP after t. It is the modified Smith rule.

Case 2 : the mandatory part of i and/or j is in the middle of its execution. This case is represented by Figure 2a. By definition, the MPs of i and j are executed simultaneously in S^* and in S'^*. Also, as the MP of i (resp. j) is in the middle of the execution of i (resp. j) in S'^*, it has no influence on C'_i (resp. C'_j). So we can ignore the MP of i (resp. j) and return to case 1 or 3 depending on the situation.

Case 3 : i and/or j ends by its MP in S'^*. If j has an MP located at the end of its execution, as the MP of j is scheduled simultaneously in S^* and S'^*, we can deduce that $C'_j = C_j = eft_j$. By construction, we have that $C'_i \leq C_i$ so the objective value of S'^* is no greater than those of S^*. If, as in Figure 2b, it is i that have its MP at the end of its execution in S'^*, then by definition of the MP, if i finishes at eft_i, then it starts at r_i and is executed without preemption. So in S'^*, i is processed in $[t, eft_i[$. As S^* and S'^* differ after t, at least one part of i is scheduled after its MP in S^*, so $C_i > C'_i = eft_i$. For $C'_j = \max(C_i, C_j)$, if $C'_j = C_j$, then S'^* is strictly better than S^*, which contradict the optimality of S^*. If $C'j = C_i$, then $\widehat{p_i} < \widehat{p_j}$. Indeed, $\widehat{p_i}$ cannot be equal to $\widehat{p_j}$ as otherwise j would have been scheduled before i in S (if j has an MP after t, it is after those of i, else i could not be scheduled in all $[t, eft_i[$). Thus $\widehat{p_j} > \widehat{p_i}$ and then :

$$\begin{aligned} \triangle_{F_T} &= C_i - C'_i + C_j - C'_j \\ &= \cancel{C_i} - C'_i + C_j - \cancel{C_i} \\ &= C_j - C'_i \\ &= C_j - eft_i \end{aligned}$$

and $\triangle_{F_T} \geq 0$ as $C_j > eft_i$ ($\widehat{p_j} > \widehat{p_i}$). It is also impossible to have i and j finishing together by their MP as otherwise, they will be processed simultaneously on $[t, \min(eft_i, eft_j)[$ which is impossible as there is only one resource.

The MSRMP can be implemented in polynomial time. An example of such implementation, whose complexity is $O(n \log n)$, is a sweep line algorithm. Indeed, each task generates one event linked with its release date, and two events linked with the start and end of its mandatory part (if any). Then, the events are sorted in non-decreasing order where ties are broken by the earliest start of the mandatory part. Finally, events are processed sequentially so that a preemptive schedule is built under MSRMP. The worst-case complexity of

this algorithm is coming from the events sort and the selection of the task with the shortest remaining processing time.

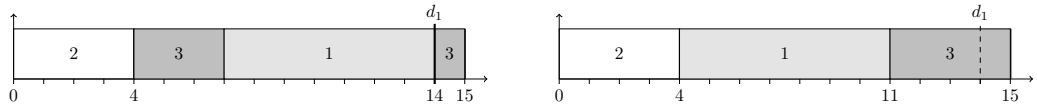
3 Non optimality of MSRMP with deadlines (d_j)

MSRMP computes in polynomial time a better lower bound of $1|r_j;d_j|\sum C_j$ than $1|r_j;pmtn|\sum C_j$ as $1|r_j;pmtn|\sum C_j$ is a relaxation of $1|r_j;pmtn;MP|\sum C_j$. However, its direct adaptation to include the deadlines by prioritazing tasks that just have the time to finish is not optimal, as shown by this example.

Example 2.
Table 2 and Figure 3a present an example of the application of the MSRMP with deadlines and three tasks. This adaptation of MSPMP is trivial: if a task has just the time to finish before being late, it has higher priority than all the other tasks. Figure 3a gives us the objective value of the MSRMP with deadlines on the instance of Table 2: it is 33 (0 + 4 + 14 + 15). However, Figure 3b gives a better solution for the same problem as its objective value is 30 (4 + 11 + 15). As Figure 3b is not following the MSRMP with deadlines, we can deduce that this rule is not optimal.

Table 2: Example the application of the MSRMP with d_j -instance-

j	1	2	3
$\mathbf{p_j}$	0	0	0
$\mathbf{r_j}$	7	4	4
$\mathbf{d_j}$	14	∞	∞



(a) MSRMP with d_j (b) Optimal solution

Fig. 3: Example of the non optimality of MSRMP with d_j

4 Conclusion and prospects

In this paper, we extended Smith rule to $1|r_j;pmtn;MP|\sum C_j$. It can solve this problem in polynomial time and so be used as a lower bound for $1|r_j;d_j|\sum C_j$ to improve its CP filtering algorithms. We also show that the adaptation of the rules to include the deadlines is not trivial. Even though, $1|r_j;d_j|\sum C_j$ and $1|r_j;d_j;pmtn|\sum C_j$ are $\mathcal{NP}$-hard, finding an algorithm which takes the deadline into account in a better way than ours is a challenging research direction.

References

Baptiste P., Le Pape C. and Nuijten W., 2001, "Constraint-Based Scheduling: Applying Constraint Programming to Scheduling Problem", *International Series in Operations Research & Management Science*, Vol. 39, Verlag Springer US, pp. xiii-198, https://doi.org/10.1007/978-1-4615-1479-4.

Brucker P., 2006,"Single Machine Scheduling Problem", *Scheduling Algorithms*, Springer, Berlin, Heidelberg, ch. 04, pp. 61–106, https://doi.org/10.1007/978-3-540-69516-5.

Kovács A. and Beck J. C., 2011, "A global constraint for total weighted completion time for unary resources ", *Constraints*, Vol. 16, No. 1, pp. 100–123, https://doi.org/10.1007/s10601-009-9088-x.

Nattaf M. and Malapert A., 2020, "Filtering Rules for Flow Time Minimization in a ParallelMachine Scheduling Proble", *CP 2020: Principles and Practice of Constraint Programming*, pp. 462-477, https://doi.org/10.1007/978-3-030-58475-7_27, https://hal.archives-ouvertes.fr/hal-3013857.