How good are SPT schedules for fair optimality criteria

Eric Angel[1], Evripidis Bampis[1], Fanny Pascual[2]

[1] IBISC - Université d'Évry Val d'Essonne, 523 Place des Terrasses, 91000 Évry, France.

E-mail: {angel, bampis}@ibisc.univ-evry.fr

[2] LIP6 - Université Paris 6, 104 avenue du président Kennedy, 75016 Paris, France.

E-mail: fanny.pascual@lip6.fr. Corresponding author.

Abstract

We consider the following scheduling setting: a set of n tasks have to be executed on a set of m identical machines. It is well known that *shortest processing time* (SPT) schedules are optimal for the problem of minimizing the total *sum of completion times* of the tasks. In this paper, we measure the quality of SPT schedules, from an approximation point of view, with respect to the following optimality criteria: *sum of completion times per machine*, *global fairness*, and *individual fairness*.

Keywords: Multiprocessor scheduling, SPT, fairness measures, approximation algorithms.

Introduction

We are given a set of n tasks, $T_1, T_2, \cdots, T_n$, with execution times $l_1, l_2, \cdots, l_n$ and a set of m identical machines (or processors) $P_1, P_2, \cdots, P_m$. Given a schedule we denote by C_j the completion time of task T_j, and by X (resp. $\overrightarrow{X}$) the completion time vector whose i^{th} coordinate is the completion time of task T_i (resp. where the coordinates of X have been reordered in non increasing order). For each instance $\mathcal{I}$ we define $V(\mathcal{I})$ to be the set of all vectors that are induced by some feasible schedule. One of the most popular optimality criteria in scheduling theory is the *sum of completion times* criterion defined as $\sum_{i=1}^{n} C_i$. An optimal solution to the problem of minimizing the *sum of completion times* can be constructed by using the *shortest execution time* (or Smith's) rule (cf. Conway, Maxwell and Miller (1967)) in which we sort the tasks in non decreasing processing time order and we schedule the tasks greedily on the machines following this order: when a machine becomes idle it executes the smallest unscheduled task. A broader class of optimal schedules for the *sum of completion times* criterion is the following class of schedules that are defined using the notion of *rank* (cf. Conway, Maxwell and Miller (1967)), that we recall now. Let ξ be the ordered collection $(l_1, \ldots, l_n)$, and let us assume that the indexing is chosen so that $l_1 \leq l_2 \leq \cdots \leq l_n$. Our scheduling instance $\mathcal{I}$ is completely defined by the system (ξ, m), where m is the number of (identical) machines. Let π_i be a collection of processing times defined as follows: $\pi_k = \{l_n, l_{n-1}, \ldots, l_{n-m+1}\}$, $\pi_{k-1} = \{l_{n-m}, \ldots, l_{n-2m+1}\}, \ldots, \pi_1 = \{l_{n-(k-1)m}, \ldots, l_1\}$, where $k = \lceil \frac{n}{m} \rceil$. The collection π_i is called the i^{th} *rank* of tasks with respect to (ξ, m). Consider a schedule obtained by scheduling the tasks *rank-by-rank* in the order $\pi_1, \pi_2, \ldots, \pi_k$ and such that no two tasks of the same rank are scheduled on the same machine: the tasks of π_1 are scheduled in the first position, each one on a different machine, then tasks of π_2 are scheduled in the second position, also each one on a different machine (and so each one after a task of the first rank), and so forth. Thus a task belonging in the i^{th} rank is scheduled at the i^{th} position on its machine, and is the only task of the i^{th} rank on its machine. Notice that any permutation of the tasks of the same rank assures the optimality of the schedule with respect to the *sum of completion times* criterion. This is the family of schedules that we call **SPT schedules** in the sequel.

We are interested to know whether among the **SPT schedules**, which are the optimal schedules for the *sum of completion times* criterion, it is possible to obtain good solutions for the following optimality criteria:

- the *maximum sum of completion times per machine*: $\max\limits_{1 \leq i \leq m} \sum\limits_{T_j \in P_i} C_j$. This measure captures the wish of distributing as much as possible the total sum of completion times among the m machines of the system.

- the *global fairness (cf. Kumar and Kleinberg (2000))*: For two vectors $X, Y \in V(\mathcal{I})$, we write $X \preccurlyeq Y$ if $X_i \leq Y_i$ for all i. The *global approximation ratio* of X, denoted by $c_{gbl}(X)$, is the smallest α such that $\overrightarrow{X} \preccurlyeq \alpha\overrightarrow{Y}$ for all $Y \in V(\mathcal{I})$. Informally $c_{gbl}(X)$ is the smallest α for which $\overrightarrow{X}$ is an α-approximation, in the coordinate-wise sense, to every vector $Y \in V(\mathcal{I})$. The best global approximation ratio achievable on the instance $\mathcal{I}$ is then defined as

$$c^*_{gbl}(\mathcal{I}) = \inf_{X \in V(\mathcal{I})} c_{gbl}(X).$$

- the *individual fairness*: The *individual happiness factor* compares the completion time of a task with the smallest possible completion time of the same task in any feasible schedule. More formally, we define $c_{ind}(X)$ to be the smallest α such that $X \preccurlyeq \alpha Y$ for all $Y \in V(\mathcal{I})$. The best approximation ratio achievable on the instance $\mathcal{I}$ is then defined as

$$c^*_{ind}(\mathcal{I}) = \inf_{X \in V(\mathcal{I})} c_{ind}(X).$$

Our approach is in the same vein as the one of Bruno, Coffman and Sethi (1974) who considered the following question: *among all optimal schedules for the* sum of completion times*, is it possible to compute one that minimizes the* makespan? (They proved that the problem is $\mathcal{NP}$-hard.) For a related problem, see also Stein and Wein (1997).

Organization of the paper and our contribution In Section 2, we consider the problem of *minimizing the maximum sum of completion times per machine* (MIN MAX SCT). We first show that contrarily to the sum of completion times, the problem MIN MAX SCT is $\mathcal{NP}$-hard. Furthermore, we show that an SPT schedule is a $(3 - 3/m + 1/m^2)$-approximation algorithm for the MIN MAX SCT problem and that there are instances for which any SPT schedule cannot achieve an approximation guarantee better than $2 - \frac{2}{m^2+m}$. In Section 3, we consider the *global approximation ratio* and we prove that SPT schedules have an approximation ratio of $2-1/m$ and that no algorithm can have a better approximation ratio. Philips, Stein and Wein (1995) presented a 3-approximation for the same problem when release dates are taken into account. For the *individual happiness factor* however the performance ratio cannot be bounded by any constant but we prove that an SPT schedule obtains the best possible performance guarantee. Finally, we focus on a more restricted version of the individual

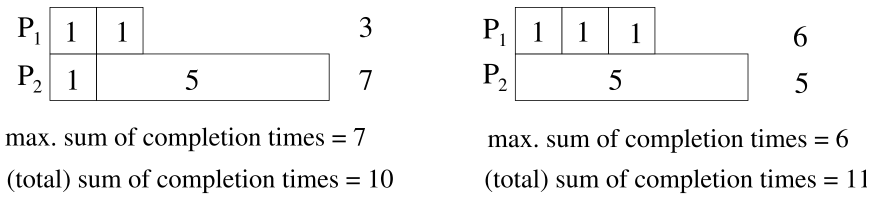


Figure 1: Instance for which the optimal solutions to the problems Minimize $\sum_{j=1}^{n} C_j$ and $\textsc{Min Max SCT}$ are different.

happiness factor where the obtained solutions are compared with respect to the family of SPT schedules and we prove a 2-approximation bound.

1 The $\textsc{Min Max SCT}$ problem

We first remark that the problem of *minimizing the sum of completion times* and the $\textsc{Min Max SCT}$ problem are different. It is easy to verify that by considering the following instance: three tasks of length 1 and a task of length 5 on two identical machines (see Figure 1). The optimal solution for the problem of minimizing the sum of completion times consists in putting two tasks of length 1 on a machine, and the other two tasks on the other machines (total sum of completion times $= 10$, maximum sum of completion times per machine $= 7$). The optimal solution of $\textsc{Min Max SCT}$ consists in putting the three tasks of length 1 on a machine, and the task of length 5 on the other machine (total sum of completion times $= 11$, maximum sum of completion time per machine $= 6$).

1.1 Hardness

We prove in this section that the $\textsc{Min Max SCT}$ problem is $\mathcal{NP}$-hard.

Theorem 1.1 $\textsc{Min Max SCT}$ *is $\mathcal{NP}$-hard.*

Proof: Consider the decision version of the $\textsc{Min Max SCT}$ problem:

$\textsc{Min Max SCT}$ problem: Given a system (ξ, m), and a number k. Does there exist a schedule such that its maximum sum of completion times is smaller than or equal to k?

We will show that the $\textsc{Partition}$ problem which is known to be $\mathcal{NP}$-hard (cf. Garey and Johnson (1979)) can be phrased in terms of problem $\textsc{Min Max SCT}$.

PARTITION problem: Given a collection $C = \{x_1, x_2, \dots, x_n\}$ of integers. Does there exist a partition (A, B) of C, i.e. $A \cup B = C$ and $A \cap B = \emptyset$, such that $\sum_{x \in A} x = \sum_{x \in B} x$?

Given an instance of PARTITION with a collection $C = \{x_1, x_2, \dots, x_n\}$ of non decreasing integers $x_1 \leq x_2 \leq \dots \leq x_n$, we define a system $(\xi, 2)$ and a limit k such that the MIN MAX SCT instance admits a solution if and only the instance of PARTITION admits a solution. Let $k = \frac{3}{2}(\sum_{i=1}^{n} x_i) - \sum_{i=1}^{n} \frac{x_i}{n-i+1}$. We now define the system $(\xi, 2)$: we have $2n$ tasks $T_1, T_2, \dots, T_{2n}$ to schedule on 2 machines P_1 and P_2. Let l_i denote the execution time of task T_i. Let the execution times $\xi = \{l_1, l_2, \dots, l_{2n}\}$ be such that $l_{2j-1} = \sum_{i=1}^{j-1} \frac{x_i}{n-i+1}$ and $l_{2j} = \sum_{i=1}^{j} \frac{x_i}{n-i+1}$, for $j \in \{1, 2, \dots, n\}$.

For example, (if $n \geq 6$), $l_1 = 0,\ l_2 = \frac{x_1}{n}$,

$$l_3 = \frac{x_1}{n},\ l_4 = \frac{x_1}{n} + \frac{x_2}{n-1},$$

$$l_5 = \frac{x_1}{n} + \frac{x_2}{n-1},\ l_6 = \frac{x_1}{n} + \frac{x_2}{n-1} + \frac{x_3}{n-2},$$

$$l_{2n-1} = \frac{x_1}{n} + \frac{x_2}{n-1} + \dots + \frac{x_{n-1}}{2},$$

$$l_{2n} = \frac{x_1}{n} + \frac{x_2}{n-1} + \dots + \frac{x_{n-1}}{2} + x_n$$

Let us now show that there is a solution of the MIN MAX SCT problem if and only if there is a solution of PARTITION.

If a partition (A, B) exists for C then there is a solution of the MIN MAX SCT problem: we can obtain for our system $(\xi, 2)$ a schedule whose maximum sum of completion times is $\frac{3}{2}(\sum_{i=1}^{n} x_i) - \sum_{i=1}^{n} \frac{x_i}{n-i+1}$. This schedule can be obtained by assigning tasks T_{2j-1} and T_{2j} at rank j: assign task T_{2j} to P_1 and T_{2j-1} to P_2 if $x_j \in A$, otherwise assign T_{2j} to P_2 and T_{2j-1} to P_1.

Indeed, the execution time of a task added at the j^{th} rank will be counted $n - j + 1$ times in the sum of the completion times of its machine. For example, the execution time of the last task of a machine will be counted only once, whereas the execution time of the first task of a machine will be counted n times (because there are n tasks on each machine). Thus the contribution to the sum of completion times per machine of the task T_{2j-1} (resp. T_{2j}), assigned at the j^{th} rank, is $(n-j+1) \times l_{2j-1}$ (resp. $(n-j+1) \times l_{2j} = (n-j+1) \times (l_{2j-1} + \frac{x_j}{n-j+1})$). The difference between these two contributions is then x_j: if $x_j \in A$ (i.e. T_{2j} is assigned to P_1) then the contribution of the j^{th} task of P_1 is equal to the contribution of the j^{th} task of P_2, plus x_j. Likewise, if $x_j \in B$

(i.e. T_{2j} is assigned to P_2) then the contribution of the j^{th} task of P_2 is equal to the contribution of the j^{th} task of P_1, plus x_j.

Thus, the sum of the completion times of the tasks of P_1 is $\sum_{j=1}^{n} l_{2j-1} + \sum_{x_j \in A} x_j$, and the sum of the completion times of the tasks of P_2 is $\sum_{j=1}^{n} l_{2j-1} + \sum_{x_j \in B} x_j$. Moreover $\sum_{j=1}^{n} l_{2j-1} = (n-1)\frac{x_1}{n} + (n-2)\frac{x_2}{n-1} + \cdots + \frac{x_{n-1}}{2} = x_1 - \frac{x_1}{n} + x_2 - \frac{x_2}{n-1} + \cdots + x_{n-1} - \frac{x_{n-1}}{2} = \sum_{i=1}^{n} x_i - \sum_{i=1}^{n} \frac{x_i}{n-i+1}$. So the sum of the completion times of the tasks of P_1 is $(\sum_{i=1}^{n} x_i - \sum_{i=1}^{n} \frac{x_i}{n-i+1}) + \sum_{x_j \in A} x_j$, and the sum of the completion times of the tasks of P_2 is $(\sum_{i=1}^{n} x_i - \sum_{i=1}^{n} \frac{x_i}{n-i+1}) + \sum_{x_j \in B} x_j$.

If there is partition of C (i.e. $\sum_{x_j \in A} = \sum_{x_j \in B} = \frac{\sum_{i=1}^{n} x_i}{2}$) then the maximum sum of completion times per machine is equal to $(\sum_{i=1}^{n} x_i - \sum_{i=1}^{n} \frac{x_i}{n-i+1}) + \frac{\sum_{i=1}^{n} x_i}{2} = \frac{3}{2}(\sum_{i=1}^{n} x_i) - \sum_{i=1}^{n} \frac{x_i}{n-i+1} = k$. Thus, if there is a solution of PARTITION, there is a solution of the MIN MAX SCT problem.

Let us now show that if there is a solution of the MIN MAX SCT problem, then there is also a solution of PARTITION. If the maximum sum of the completion times per machine is smaller than or equal to $\frac{3}{2}(\sum_{i=1}^{n} x_i) - \sum_{i=1}^{n} \frac{x_i}{n-i+1}$, then we have an SPT schedule. Indeed, the total sum of the completion times of an SPT schedule is $S = 2 \times \left(\frac{3}{2}(\sum_{i=1}^{n} x_i) - \sum_{i=1}^{n} \frac{x_i}{n-i+1}\right)$, as we saw it above, and a schedule has a minimum total sum of completion times if and only if it is an SPT schedule (cf. Bruno, Coffman, Sethi (1974)) Since the total sum of completion times cannot be greater than twice the maximum sum of completion times per machine, a schedule which is a solution of the MIN MAX SCT problem is an SPT schedule. In any SPT schedule, the tasks of length l_{2i} and l_{2i-1} are at the i^{th} rank, because $l_1 < l_2 \leq l_3 < \cdots \leq l_{2j-1} < l_{2j} \leq l_{2j+1} < l_{2j+2} \leq \cdots < l_n$. So, in any SPT schedule, the sum of completion times of P_1 minus the one of P_2 is equal to $\sum_{T_{2j} \in P_1} x_j - \sum_{T_{2j} \in P_2} x_j$. If there is a solution of the MIN MAX SCT problem with $k = \frac{3}{2}(\sum_{i=1}^{n} x_i) - \sum_{i=1}^{n} \frac{x_i}{n-i+1}$ then the sum of the completion times of P_1 is equal to the sum of the completion times of P_2 (otherwise the sum of completion times of all the tasks would be smaller than S), and then $\sum_{T_{2j} \in P_1} x_j = \sum_{T_{2j} \in P_2} x_j$: there is a partition of C and we can construct this partition (A, B) by placing x_j in A if T_{2j} is assigned to P_1 and x_j in B if T_{2j} is assigned to P_2. $\square$

1.2 Approximation

In this section, we show that an optimal algorithm for the *sum of completion times* criterion gives a 3-approximate solution for the Min Max SCT problem. Consider the following algorithm, denoted by $\text{SPT}_{\text{greedy}}$:

> Order tasks by non decreasing execution times. At each step i, for $1 \leq i \leq n$, schedule the current task on the machine which has the smallest completion time.

This algorithm gives an optimal solution for the problem of minimizing the *sum of completion times of the tasks*. Let us show that this algorithm is a $(3 - \frac{3}{m} + \frac{1}{m^2})$-approximation algorithm for Min Max SCT.

In order to let $\text{SPT}_{\text{greedy}}$ be deterministic when several machines have the smallest execution time, we will refer in the proofs to the following algorithm, which is a greedy SPT algorithm (Lemma 1.1 shows that we add at each step a task on a machine which has the smallest completion time):

> Order tasks by non decreasing execution times. At each step i, for $1 \leq i \leq n$, schedule the current task on the machine $P_{i \bmod m}$.

Lemma 1.1

a) At the beginning of step i of SPT_{greedy}, the machine $P_{i \bmod m}$ is the machine which has the smallest completion time and the smallest sum of completion times.

b) At the end of step i of SPT_{greedy}, the machine $P_{i \bmod m}$ is the machine which has the largest sum of completion times.

Proof:

a) We are at the beginning of step i. Let c be the machine $P_{i \bmod m}$, and let p denote its completion time. Let us show that c has the smallest completion time:

- For each $k > i \bmod m$, machine P_k has a completion time greater than or equal to p. Indeed P_k has the same number of tasks as c and, for each i such that $1 \leq i \leq$ number of tasks on c, the i-th task of P_k is greater than or equal to the i-th task of c, by construction.

– For each $k < i \bmod m$, machine P_k has a completion time greater than or equal to p. Indeed the number of tasks on P_k is equal to the number of tasks on c plus one, and, for each i such that $1 \leq i \leq$ (number of tasks on c), the $(i+1)$-th task of P_k is greater than or equal to the i-th task of c, by construction.

We use the same reasoning to show that c has the smallest sum of completion times.

b) We are at the end of step i. Let c be the machine $P_{i \bmod m}$, and let p denote its sum of completion times at this step. Let us show that c has the largest sum of completion times:

– For each $k > i \bmod m$, machine P_k has a sum of completion times smaller than or equal to p. Indeed P_k has one task less than c and, for each i such that $1 \leq i <$ number of tasks on c, the i-th task of P_k is smaller than or equal to the $(i+1)$-th task of c, by construction.

– For each $k < i \bmod m$, machine P_k has a sum of completion times smaller or equal to p. Indeed P_k has the same number of tasks as c and, for each i such that $1 \leq i \leq$ number of tasks on c, the i-th task of P_k is smaller than or equal to the i-th task of c, by construction.

□

Lemma 1.2 *Let OPT be the maximum sum of completion times of a solution of* MIN MAX SCT. *We have:*

$$OPT \geq \frac{Min(\sum_{j=1}^{n} C_j)}{m},$$

where $Min(\sum_{j=1}^{n} C_j)$ is the optimal value for the problem of minimizing the sum of completion times.

Proof: By definition each machine has a sum of completion times smaller than or equal to OPT: $\forall i \in \{1, \ldots, m\},\ \sum_{T_j \in P_i} C_j \leq OPT$.

So, $\sum_{j=1}^{n} C_j = \sum_{i=1}^{m} \sum_{T_j \in P_i} C_j \leq m \times OPT$, and $OPT \geq \frac{\sum_{j=1}^{n} C_j}{m} \geq \frac{\mathrm{Min}(\sum_{j=1}^{n} C_j)}{m}$. □

Theorem 1.2 *The algorithm SPT$_{greedy}$ achieves an approximation guarantee of $(3 - \frac{3}{m} + \frac{1}{m^2})$ for* MIN MAX SCT.

Proof: Before we add the last task T_n, the machine P_x on which T_n will be scheduled has the smallest sum of completion times, denoted by p, and the smallest completion time, denoted by e (Lemma 1.1 a). Let l_n denote the execution time of task T_n. The machine on which the last task is scheduled has the largest sum of completion times (Lemma 1.1 b), so:

$$\begin{array}{rcl} \max_{1\leq i\leq m}\sum_{T_j\in P_i} C_j & = & \sum_{T_j\in P_x} C_j = p+e+l_n \\ & \leq & \frac{\sum_{j=1}^{n-1} C_j}{m} + e + l_n \end{array}$$

Since $\text{SPT}_{\text{greedy}}$ gives an optimal solution of the minimum sum of completion times problem, and since the completion time of T_n in $\text{SPT}_{\text{greedy}}$ is $e+l_n$, we have:

$$\begin{array}{rcl} \max_{1\leq i\leq m}\sum_{T_j\in P_i} C_j & \leq & \frac{\text{Min}(\sum_{j=1}^{n} C_j)-(e+l_n)}{m} + e + l_n \\ & \leq & \frac{\text{Min}\sum_{j=1}^{n} C_j}{m} + (1-\frac{1}{m})(e+l_n). \end{array}$$

Since $e+\frac{l_n}{m}\leq OPT$ (because e is the minimum completion time at step $n-1$), and $l_n\leq OPT$, we have:

$$\begin{array}{rcl} e+l_n & = & e+\frac{l_n}{m}+\frac{(m-1)\,l_n}{m} \\ & \leq & (1+\frac{m-1}{m})OPT \\ & \leq & (2-\frac{1}{m})OPT \end{array}$$

Moreover, since $\frac{\text{Min}\sum_{j=1}^{n} C_j}{m}\leq OPT$ (Lemma 1.2), we have:

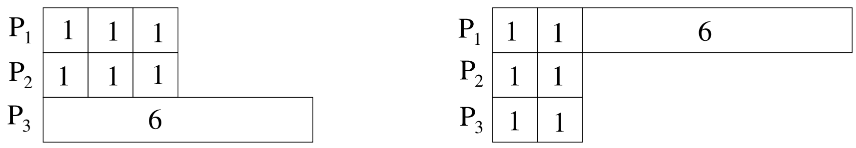
$$\begin{array}{rcl} \max_{1\leq i\leq m}\sum_{T_j\in P_i} C_j & \leq & OPT+(1-\frac{1}{m})(2-\frac{1}{m})OPT \\ & \leq & (3-\frac{3}{m}+\frac{1}{m^2})OPT. \end{array}$$

$\square$

1.2.1 Lower bound for $\text{SPT}_{\text{greedy}}$

We now show that $\text{SPT}_{\text{greedy}}$ does not achieve an approximation guarantee better than $2-\frac{2}{m^2+m}$. Consider the following instance: m machines, $m\times(m-1)$ tasks of length 1 and a task of length $B=\sum_{i=1}^{m} i=\frac{m(m+1)}{2}$.

$\text{SPT}_{\text{greedy}}$ will schedule $m-1$ tasks of length 1 on each machine, and the task of length B will be the last task of the first machine (see Figure 2). The maximum sum of completion times is then $\sum_{i=1}^{m-1} i+(m-1)+B$, which is equal to $2(\sum_{i=1}^{m} i)-1$.



max. sum of completion times = 6 max. sum of completion times = 11

Figure 2: SPT$_{\text{greedy}}$ does not achieve an approximation guarantee better than $(2 - \frac{2}{m^2+m})$: example with m=3.

An optimal solution for MIN MAX SCT would be the following one: schedule m tasks of length 1 on each of the $(m-1)$ first machines, and the task of length B on the last machine. The maximum sum of completion times in this solution is then $\sum_{i=1}^{m} i$.

So the ratio between the maximum sum of completion times of these two schedules is $\frac{2(\sum_{i=1}^{m} i)-1}{\sum_{i=1}^{m} i}$, which is equal to $2(1 - \frac{1}{m^2+m})$, which tends towards 2 when m gets large.

2 Fairness measures

In this section we will consider fairness measures in order to compare a schedule given by the greedy SPT algorithm, SPT$_{\text{greedy}}$, to any other schedule.

2.1 Global fairness measure

We will first use the fairness measure introduced in Kumar and Kleinberg (2000), namely the *global approximation ratio*. We prove that SPT$_{\text{greedy}}$ has a global approximation ratio of $2 - \frac{1}{m}$ for the problem that we consider (and which is denoted by P $\|$ all), and that no algorithm can achieve a better global approximation ratio if $m = 2$.

Theorem 2.1 *One has $c^*_{gbl}(\mathcal{I}) \leq 2 - \frac{1}{m}$ for all instances $\mathcal{I}$ of scheduling on m identical parallel machines (P $\|$ all), for any $m \geq 1$. Moreover the completion times vector X of a schedule returned by SPT$_{\text{greedy}}$ verifies $c_{gbl}(X) \leq 2 - \frac{1}{m}$.*

Proof: Let us consider an instance $\mathcal{I}$ of tasks T_i ($i \in \{1, \ldots, n\}$) ordered by non decreasing lengths. Let MR be the maximal ratio between the schedule X_{SPT} returned by SPT$_{\text{greedy}}$ and any schedule Y: MR $= \max_Y \frac{\overrightarrow{X_{SPT}}}{\overrightarrow{Y}}$, where $\frac{X}{Y}$ means $\max_i \frac{X_i}{Y_i}$. So we have $\forall Y \in V(\mathcal{I}), \frac{\overrightarrow{X_{SPT}}}{\overrightarrow{Y}} \leq$ MR, and then $\overrightarrow{X_{SPT}} \preccurlyeq$ MR $\overrightarrow{Y}$. So MR $= c_{gbl}(X_{SPT})$. We will show that MR cannot be greater than $2 - \frac{1}{m}$. Let us consider that this ratio is the i^{th} coordinate of $\overrightarrow{X_{SPT}}$, divided by the i^{th} coordinate of $\overrightarrow{Y}$ ($i \in \{1, \ldots, n\}$).

The worst ratio can be achieved if the completion time of the i^{th} task is as large as possible in the SPT$_{\text{greedy}}$ schedule. By construction, in a schedule returned by SPT$_{\text{greedy}}$, the i^{th} completion time is the completion time of the task T_i, and T_i is started after the tasks from T_1 to T_{i-1}, and before the tasks from T_{i+1} to T_n. Since SPT$_{\text{greedy}}$ is a greedy algorithm, the worst completion time of the i^{th} task is achieved when the $(i-1)$ first tasks are completed when the i^{th} task starts to be executed: in this case, the completion time of T_i is $\frac{\sum_{j=1}^{i-1} l_j}{m} + l_i$. So the worst completion time of T_i in an SPT$_{\text{greedy}}$ schedule is $\frac{\sum_{j=1}^{i-1} l_j}{m} + l_i$.

Let us now find the minimal value which can be taken by the i^{th} coordinate of $\overrightarrow{Y}$, an ordered completion time vector of a schedule of $\mathcal{I}$. Note that this value cannot be smaller than l_i: indeed $\overrightarrow{Y}_i$ is the i^{th} completion time and is then greater or equal to $(i-1)$ other completion times, and l_i is the length of the i^{th} smallest task. Note also that $\overrightarrow{Y}_i$ cannot be smaller than $\frac{\sum_{j=1}^{i} l_j}{m}$: indeed this is the minimum completion time of the i smallest tasks (when no machine is idle). Therefore, we have:

$$\text{MR} \leq \frac{\frac{\sum_{j=1}^{i-1} l_j}{m} + l_i}{\max\left(\frac{\sum_{j=1}^{i} l_j}{m}\ ,\ l_i\right)}.$$

Let A denote $\frac{\sum_{j=1}^{i} l_j}{m}$. We now consider the two possible cases:

- Suppose that $l_i \geq A$, we have: MR $\leq \frac{\frac{\sum_{j=1}^{i-1} l_j}{m}+l_i}{l_i} \leq \frac{A-\frac{l_i}{m}+l_i}{l_i} \leq \frac{A}{l_i} + 1 - \frac{1}{m} \leq 2 - \frac{1}{m}$

- Suppose that $l_i < A$, we have: MR $\leq \frac{\frac{\sum_{j=1}^{i-1} l_j}{m}+l_i}{\frac{\sum_{j=1}^{i} l_j}{m}} \leq \frac{A-\frac{l_i}{m}+l_i}{A} \leq 1 + \frac{l_i}{A}(1 - \frac{1}{m}) \leq 2 - \frac{1}{m}$

In both cases, we have MR $\leq 2 - \frac{1}{m}$. Since MR $= c_{gbl}(X_{SPT})$, and since X_{SPT} is the completion time vector of the SPT$_{\text{greedy}}$ schedule of $\mathcal{I}$, we proved that a schedule X returned by SPT$_{\text{greedy}}$ verifies $c_{gbl}(X) \leq 2 - \frac{1}{m}$, and so that $c^*_{gbl}(\mathcal{I}) \leq 2 - \frac{1}{m}$. □

completion times vector: X1=(1, 2, 2) completion times vector: X2=(1, 1, 3)

Figure 3: Example where we have $c^*_{gbl}(\mathcal{I}) = \frac{3}{2} = 2 - \frac{1}{m}$ in the case where $m = 2$.

Let us now show that it is not possible to have $c^*_{gbl}(\mathcal{I}) < 2 - \frac{1}{m}$ for all instances $\mathcal{I}$ of (P || all).

Theorem 2.2 *It is not possible to have $c^*_{gbl}(\mathcal{I}) < 2 - \frac{1}{m}$ for all instances $\mathcal{I}$ of scheduling on m identical parallel machines.*

Proof: Let us consider a system with m machines. Let us consider the following instance $\mathcal{I}$: $m\,(m-1)$ tasks of length 1 and at task of length m (see Figure 3 in the case where $m = 2$). Let us consider the completion time vector X_1 obtained with SPT$_{\text{greedy}}$: the m first values are equal to 1; for all $i \in \{2, \ldots, m-1\}$, the values between $(i\,m+1)$ and $(i+1)\,m$ are equal to $(i+1)$; and the value $m\,(m-1)+1$ is equal to $(m-1)+m = 2\,m-1$. In any schedule in which the m first tasks are not tasks of length 1, the task of length m is one of these m first tasks, and this one has a completion time equal to $m \geq 2$, and the global approximation ratio is thus larger than or equal to 2. Likewise, for all $i \in \{2, \ldots, m-1\}$, in any schedule in which the tasks between the positions $(i\,m+1)$ and $(i+1)\,m$ are not tasks of length 1, there is the task of length m whose completion time is larger than or equal to $(i+m)$ and the global approximation ratio is larger than or equal to $(i+m)/(i+1) \geq 2-1/m$ because $i \leq m-1$.

Note that the global approximation ratio of the schedule returned by SPT$_{\text{greedy}}$ is equal to $2 - 1/m$ because the completion time of the last task is $(m-1) + m = 2\,m - 1$ whereas the last completion time of an optimal schedule for the makespan (where the task of length m is alone on its machine) is equal to m. The ratio between these two values is then $(2\,m-1)/m = 2-1/m$. Thus, for each vector $Y \in V(\mathcal{I})$, we have $c_{gbl}(Y) \geq 2 - \frac{1}{m}$, and so $c^*_{gbl}(\mathcal{I}) = 2 - \frac{1}{m}$. □

2.2 Individual fairness measure

In this section, we will compare the completion time of each task in a solution given by SPT$_{\text{greedy}}$ to the best completion time that this task could have in another schedule.

Theorem 2.3 $c^*_{ind}(\mathcal{I}) \leq 1 + \frac{n-1}{m}$ *for all instances* $\mathcal{I}$ *of scheduling on* m *identical parallel machines (P* $\|$ ind_all*), for any* $m \geq 1$*. Moreover the completion times vector* X *of a schedule returned by SPT*$_{greedy}$ *verifies* $c_{ind}(X) \leq 1 + \frac{n-1}{m}$.

Proof: Let us consider an instance $\mathcal{I}$ of tasks T_i ($i \in \{1, \dots, n\}$) ordered by non decreasing lengths. It is easy to see that $c_{ind}(X_{SPT})$ is the maximal ratio between a schedule X_{SPT} returned by SPT$_{\text{greedy}}$ and any schedule Y. We will show that $c_{ind}(X_{SPT})$ cannot be greater than $1 + \frac{n-1}{m}$. Let us consider that this ratio is the i^{th} coordinate of X_{SPT}, divided by the i^{th} coordinate of Y ($i \in \{1, \dots, n\}$).

The worst ratio can be reached if the completion time of T_i is as large as possible in the SPT$_{\text{greedy}}$ schedule. Since SPT$_{\text{greedy}}$ is a greedy algorithm, this is achieved when the $(i-1)$ first tasks are completed when T_i starts to be executed: in this case, the completion time of T_i is $\frac{\sum_{j=1}^{i-1} l_j}{m} + l_i$. So the maximal value which can be taken by the i^{th} coordinate of X_{SPT} is $\frac{\sum_{j=1}^{i-1} l_j}{m} + l_i$. The minimal value which can be taken by the i^{th} coordinate of Y is l_i: this is achieved in a schedule in which T_i is in the first position. We have:

$$c_{ind}(X_{SPT}) \leq \frac{\frac{\sum_{j=1}^{i-1} l_j}{m} + l_i}{l_i} \leq 1 + \frac{\sum_{j=1}^{i-1} l_j}{m\, l_i} \leq 1 + \frac{(i-1)\, l_i}{m\, l_i} \leq 1 + \frac{n-1}{m}.$$

Since $c_{ind}(X_{SPT}) \leq 1 + \frac{n-1}{m}$, we have $c^*_{ind}(\mathcal{I}) \leq 1 + \frac{n-1}{m}$. □

Theorem 2.4 *It is not possible to have* $c^*_{ind}(\mathcal{I}) < 1 + \frac{n-1}{m}$ *for all instances* $\mathcal{I}$ *of scheduling on* m *identical parallel machines (P* $\|$ ind_all*).*

Proof: Let us consider a system with m machines and the following instance $\mathcal{I}$: $m+1$ tasks of length 1. We have $c^*_{ind}(\mathcal{I}) = 2$ because at least one of the tasks will have a completion time of 2. Since $c^*_{ind}(\mathcal{I}) = 2 = 1 + \frac{n-1}{m}$, it is not possible to have $c^*_{ind}(\mathcal{I}) < 1 + \frac{n-1}{m}$ for all instances $\mathcal{I}$. □

2.3 Individual fairness measure among SPT schedules

In this section, we will compare the completion time of each task in an SPT$_{\text{greedy}}$ schedule to the best completion time this task could have in another SPT schedule.

In addition to the notations introduced in the last sections, we define $V_{SPT}(\mathcal{I})$ as the set of all completion times vectors that are induced by some SPT solutions of the instance $\mathcal{I}$. Let $X \in V_{SPT}(\mathcal{I})$. We define $c_{SPT}(X)$ to be the smallest α such that $X \preccurlyeq \alpha Y$ for all $Y \in V_{SPT}(\mathcal{I})$. This can be informally viewed as the global approximation ratio of X: it is the smallest α for which X is an α-approximation to every vector $Y \in V_{SPT}(\mathcal{I})$. The best approximation ratio achievable on the instance $\mathcal{I}$ is then defined as

$$c^*_{SPT}(\mathcal{I}) = \inf_{X \in V_{SPT}(\mathcal{I})} c_{SPT}(X).$$

Theorem 2.5 *$c^*_{SPT}(\mathcal{I}) \leq 2$ for all instances $\mathcal{I}$ of scheduling on m identical parallel machines (P $\|$ ind_all_{SPT}). Moreover any SPT schedule X verifies $c_{SPT}(X) \leq 2$.*

Proof: Let us consider an instance $\mathcal{I}$ of tasks T_i ($i \in \{1, \ldots, n\}$) ordered by increasing lengths.

It is easy to see that $c_{SPT}(X)$ is the maximal ratio between an SPT schedule X and any SPT schedule Y. We will show that $c_{SPT}(X)$ cannot be greater than 2. Let us consider that this ratio is the i^{th} coordinate of X, divided by the i^{th} coordinate of Y ($i \in \{1, \ldots, n\}$). Let P_{iX} (resp. P_{iY}) be the machine on which T_i is scheduled in the solution X (resp Y), and let r_i denote the rank on which the task T_i is scheduled.

The worst ratio is reached if the completion time of T_i is as large as possible in the schedule X. This is achieved when the tasks before T_i on P_{iX} are as large as possible: each largest task on each rank smaller than r_i is on P_{iX} in the solution X. Let B_m denote the execution time of these tasks. B_m is the execution time of the tasks on the machine P_m in an $\text{SPT}_{\text{greedy}}$ schedule. In the same way, the worst ratio is reached if the completion time of T_i is as small as possible in the schedule Y: each smallest task on each rank smaller than r_i is on P_{iY} in the solution Y. Let B_1 denote the execution time of these tasks. B_1 is the execution time of the tasks on the machine P_1 in an $\text{SPT}_{\text{greedy}}$ schedule. So the difference between the completion time of T_i in X and in Y is equal to $\delta = B_m - B_1$. This number is smaller than or equal to the length of each task in the rank r_i, and then is smaller than or equal to l_i, the length of T_i (this property is a direct consequence of Lemma 1.1). We have:

$$c_{SPT}(X) \leq \frac{B_m + l_i}{B_1 + l_i} \leq \frac{B_1 + \delta + l_i}{B_1 + l_i} \leq 2.$$

For any SPT schedule X of an instance $\mathcal{I}$, we have: $c_{SPT}(X) \leq 2$, and so $c^*_{SPT}(\mathcal{I}) \leq 2$. □

Let us now show that this bound is the best possible one.

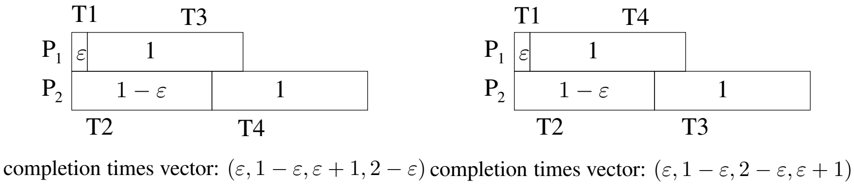


Figure 4: Example where $c^*_{SPT}(\mathcal{I})$ tends towards 2 when ε tends towards 0.

Theorem 2.6 *Let ε be any small number such that $\varepsilon > 0$. It is not possible to have $c^*_{SPT}(\mathcal{I}) < 2-\varepsilon$ for all instances $\mathcal{I}$ of scheduling on m identical parallel machines ($P \parallel ind\_all_{SPT}$).*

Proof: For the ease of presentation we give a proof for a system with two machines. Let us consider the following instance $\mathcal{I}$: a task T_1 of length ε, T_2 of length $1-\varepsilon$, and two tasks T_3 and T_4 of length 1. Assume that $\varepsilon < 1$. The completion times vectors corresponding to the two possible SPT schedules are $X_1 = (\varepsilon, 1-\varepsilon, 1+\varepsilon, 2-\varepsilon)$ and $X_2 = (\varepsilon, 1-\varepsilon, 2-\varepsilon, 1+\varepsilon)$ (see Figure 4). We have: $c^*_{SPT}(\mathcal{I}) = c_{SPT}(X_1) = c_{SPT}(X_2) = \frac{2-\varepsilon}{1+\varepsilon} < 2-\varepsilon$, which tends towards 2 when ε tends towards 0. If $m > 2$, the proof is the same except that we add $m-2$ tasks of length $1-\varepsilon$ and $m-2$ tasks of length 1 to the instance described for the case where $m = 2$. □

References

[1] R.W. Conway, W.L. Maxwell, and L.W. Miller. (1967). *Theory of scheduling*. Addison-Wesley.

[2] M. Garey and D. Johnson. (1979). *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W H Freeman & Co.

[3] J. Bruno, E.G. Coffman Jr., and R. Sethi. (1974). *Algorithms for minimizing mean flow time*. Proceedings of IFIP Congress 74, 504-510, Stockholm, Sweden, August 5-10, 1974, North-Holland.

[4] A. Kumar and J. Kleinberg. (2000). *Fairness Measures for Resource Allocation*. Proceedings of 41st IEEE Symposium on Foundations of Computer Science, 75-85.

[5] C. Phillips, C. Stein and J. Wein. (1995). *Scheduling jobs that arrive over time*. Proceedings of the Fourth Workshop on Algorithms and Data Structures, LNCS 955, Springer, 86-97.

[6] C. Stein and J. Wein. (1997). *On the existence of schedules that are near-optimal for both makespan and total weighted completion time*. Operations Research Letters 21 (3), 115-122.