

Lower and Upper Bounds for the Preemptive Single Machine Scheduling Problem with Equal Processing Times

Mikhail Batsyn, Boris Goldengorin, Pavel Sukhov, and Panos M. Pardalos

Abstract The preemptive single machine scheduling problem of minimizing the total weighted completion time with equal processing times and arbitrary release dates is one of the four single machine scheduling problems with an open computational complexity status. In this chapter we present lower and upper bounds for the exact solution of this problem based on the assignment problem. We also investigate properties of these bounds and worst-case behavior.

Keywords Single machine scheduling · Lower bound · Upper bound · Assignment problem · Weighted completion time · Equal processing times · Release dates

1 Introduction

The complexity status of many scheduling problems is known, but there are four single machine scheduling problems which computational complexity is an open question. In this chapter, we consider one of these four problems—the preemptive single machine scheduling problem of minimizing the total weighted completion time with equal processing times and arbitrary release dates. In the notation

M. Batsyn (✉) · P. Sukhov · P.M. Pardalos

Laboratory of Algorithms and Technologies for Network Analysis, National Research University Higher School of Economics, 136 Rodionova, Nizhny Novgorod, Russian Federation
e-mail: mbatsyn@hse.ru

P. Sukhov

e-mail: pavelandreevith@gmail.com

P.M. Pardalos

e-mail: pardalos@ufl.edu

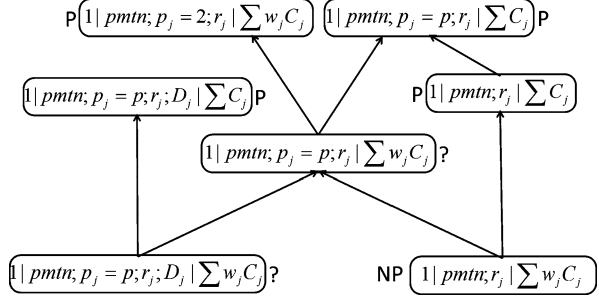
B. Goldengorin · P.M. Pardalos

Center of Applied Optimization, University of Florida, 401 Weil Hall, P.O. Box 116595, Gainesville, FL 32611-6595, USA

B. Goldengorin

e-mail: goldengorin@ufl.edu

Fig. 1 Scheduling problems with completion time as an objective function and allowed preemptions



$\langle \text{machine environment} \mid \text{job characteristics} \mid \text{objective function} \rangle$ introduced by Graham et al. (1979) [6], this problem is written as follows.

$$1|pmtn; p_j = p; r_j | \sum w_j C_j \quad (1)$$

Here “1” means a single machine, “pmtn”—permutations are allowed: processing of a job can be interrupted in favor of another job, $p_j = p$ —jobs have equal processing times p , r_j —jobs have arbitrary release dates r_j , $\sum w_j C_j$ —the objective is to minimize the total weighted completion time, where w_j is the weight of job j and C_j —its completion time. There are no polynomial algorithms developed for this problem. So solving it for a large number of jobs or long processing time might be quite difficult.

Special cases of this problem are polynomially solvable, generalizations are NP-hard or have unknown complexity. Figure 1 shows the graph of existing single machine scheduling problems with completion time as an objective function and allowed preemptions. A good classification of a wide range of static scheduling problems is given by Herrman et al. (1993) [7].

The first of these problems $1|pmtn; r_j | \sum C_j$ was solved by Baker (1974) [1]. He proved that this problem is polynomially solvable and, therefore, a special case of $1|pmtn; p_j = p; r_j | \sum w_j C_j$ problem without weights: $1|pmtn; p_j = p; r_j | \sum C_j$ is also polynomial. The generalization of this problem, $1|pmtn; r_j | \sum w_j C_j$, is strongly NP-hard. It was proved by Labetoulle et al. (1984) [8]. Another related problem having strict deadlines $1|pmtn; p_j = p; r_j; D_j | \sum C_j$ is polynomial with complexity $O(n \log(n))$ (Smith, 1956 [11]). Complexity of a more general $1|pmtn; p_j = p; r_j; D_j | \sum w_j C_j$ problem is still an open question. Bouma and Goldengorin (2009) [3] proved that a special case of $1|pmtn; p_j = p; r_j | \sum w_j C_j$ problem with $p = 2$ is polynomially solvable. Obviously there is one more special case of $1|pmtn; p_j = p; r_j | \sum w_j C_j$ problem: the problem with release dates equal to 0. But preemption has no sense without different release dates. In this case, the problem can be solved exactly using WSPT (Weighted Shortest Processing Time) rule (Pinedo, 2012 [10]). $1 | \sum w_j C_j$ problem was one of the first problems in scheduling theory which complexity status was determined. Smith (1956) [11] proved that the complexity of this problem is $O(n \log(n))$. After that, in 1975 $1|r_j | \sum C_j$ problem was shown to be strongly NP-hard (Lenstra et al., 1975 [9]).

and therefore its weighted generalization $1|r_j|\sum w_j C_j$ is NP-hard too. The special case of this generalization was solved by Baptiste (2000) [2]. He suggested a polynomial algorithm for $1|p_j = p; r_j|\sum w_j C_j$ with complexity $O(n^7)$. Another generalization of $1|p_j = p; r_j|\sum w_j C_j$ problem for parallel machines was shown to be polynomial by Brucker and Kravchenko (2008) [4].

The chapter is organized as follows. In the next section, we formulate the considered single machine problem and give an illustrating example. In the third and fourth sections, we describe the suggested lower and upper bounds based on the assignment problem. We provide the worst-case analysis of these bounds and give estimations of their precision. In the fifth section, we conclude the chapter with a short summary and some remarks on the future research.

2 Problem Formulation

The problem $1|pmtn; p_j = p; r_j|\sum w_j C_j$ can be described as follows. We are given $n \geq 1$ jobs that need to be processed on one machine. Jobs have the same processing time $p_j = p$, arbitrary release dates r_j , and weights w_j . A release date r_j is the time moment at which job j becomes available for processing. A weight w_j can be seen as a priority factor of job j . Release dates r_j are assumed to be non-negative integers, weights w_j and processing time p are assumed to be strictly positive integers. Preemptions are allowed, which means that processing of a job may be interrupted in favor of another job. The objective is to schedule the jobs such that the total weighted completion time $\sum w_j C_j$ is minimized, where C_j denotes the completion time of job j . Also we assume that there are no idle time intervals. In what follows, we will call this scheduling problem shortly as *SP*.

To make the description clearer, let us consider the following example. Let $n = 3$, $p = 4$, $w_j = 1, 2, 3$, $r_j = 1, 4, 7$. The feasible solutions of this problem are shown in Table 1 excluding the solutions in which a job with a greater weight is interrupted by a job with a smaller weight. Such solutions cannot be optimal.

The optimal solution is the first one in this example. Its total weighted completion time is $w_1 \cdot C_1 + w_2 \cdot C_2 + w_3 \cdot C_3 = 1 \cdot 4 + 2 \cdot 8 + 3 \cdot 12 = 56$. This solution can be represented as a vector $(1, 1, 1, 1, 2, 2, 2, 2, 3, 3, 3, 3)$ which has $np = 12$ elements corresponding to moments of time $1, 2, \dots, 12$. An element on position t shows what job (one part of a job) is scheduled to the time moment t .

3 Lower Bound

If we consider the solution of the scheduling problem *SP*, it can be noticed that it is actually an assignment of np job parts (each of n jobs has p parts) to np time moments $1, 2, \dots, np$ such that every part of job j is assigned to a time moment not earlier than its release date r_j . Only the last (p -th) part of every job j is taken into account in the objective function. The cost of its assignment to time t is equal

Table 1 Possible solutions

1	2	3	4	5	6	7	8	9	10	11	12	
1				2				3				
$1 \cdot 4 + 2 \cdot 8 + 3 \cdot 12 = 56$												
1				2				3				2
$1 \cdot 4 + 2 \cdot 12 + 3 \cdot 11 = 61$												
1				2		3				2		
$1 \cdot 4 + 2 \cdot 12 + 3 \cdot 10 = 58$												
1		2				3				1		
$1 \cdot 12 + 2 \cdot 7 + 3 \cdot 11 = 59$												
1		2				3				2	1	
$1 \cdot 12 + 2 \cdot 11 + 3 \cdot 10 = 64$												

Table 2 Assignment problem

	1	2	3	4	5	6	7	8	9	10	11	12
1 (job 1, part 1)	0	0	0	0	0	0	0	0	0	∞	∞	∞
2 (job 1, part 2)	∞	0	0	0	0	0	0	0	0	0	∞	∞
3 (job 1, part 3)	∞	∞	0	0	0	0	0	0	0	0	0	∞
4 (job 1, part 4)	∞	∞	∞	4	5	6	7	8	9	10	11	12
5 (job 2, part 1)	∞	∞	∞	0	0	0	0	0	0	∞	∞	∞
6 (job 2, part 2)	∞	∞	∞	∞	0	0	0	0	0	0	∞	∞
7 (job 2, part 3)	∞	∞	∞	∞	∞	0	0	0	0	0	0	∞
8 (job 2, part 4)	∞	∞	∞	∞	∞	∞	14	16	18	20	22	24
9 (job 3, part 1)	∞	∞	∞	∞	∞	∞	0	0	0	∞	∞	∞
10 (job 3, part 2)	∞	∞	∞	∞	∞	∞	∞	0	0	0	∞	∞
11 (job 3, part 3)	∞	∞	∞	∞	∞	∞	∞	∞	0	0	0	∞
12 (job 3, part 4)	∞	∞	∞	∞	∞	∞	∞	∞	∞	30	33	36

to w_{jt} , and the costs for parts $1, 2, \dots, p-1$ are all equal to 0. The objective is to minimize the total cost over all jobs. Thus, SP problem is equivalent to the linear assignment problem with additional constraints related to release dates r_j and also to the sequence of job parts which requires the last part of a job to be assigned to the latest time moment among the time moments of all its parts. Moreover, release date constraints can be taken into account in the classical assignment problem using infinite costs for the time moments to which job parts cannot be assigned. For the example in Table 1, the assignment problem will have the cost matrix shown in Table 2.

The solution is shown by gray cells. It satisfies all the constraints of the original *SP* problem except the sequence constraints requiring the last part of a job to be scheduled before all other parts. The following theorem shows that the assignment problem provides a lower bound on the optimal solution of *SP*. We will denote this assignment problem as AP_L .

Theorem 1 *The solution of the following assignment problem AP_L gives a lower bound to the solution of the scheduling problem $1|pmtn; p_j = p; r_j|\sum w_j C_j$:*

$$\min \sum_{i=1}^{np} \sum_{t=1}^{np} c_{it} x_{it} \quad (2)$$

$$\sum_{t=1}^{np} x_{it} = 1 \quad \forall i = \overline{1, np} \quad (3)$$

$$\sum_{i=1}^{np} x_{it} = 1 \quad \forall t = \overline{1, np} \quad (4)$$

$$x_{it} \in \{0, 1\} \quad (5)$$

where

$$c_{it} = \begin{cases} 0, & \text{if } t \in [r_j + i', (n-1)p + 1 + i'] \text{ \& } i \bmod p > 0 \\ \infty, & \text{if } t \notin [r_j + i', (n-1)p + 1 + i'] \\ w_{i/p} t, & \text{if } t \in [r_j + p - 1, np] \text{ \& } i \bmod p = 0 \end{cases} \quad (6)$$

$$j = \lceil (i-1)/p \rceil + 1, \quad i' = (i-1) \bmod p$$

Proof We will show that *SP* problem is equivalent to the stated assignment problem AP_L with some additional constraints. By definition, *SP* is the problem of assigning every part $1, 2, \dots, p$ of every job $j = 1, 2, \dots, n$ to position (time moment) $t = 1, 2, \dots, np$ of a schedule so that the total weighted completion time $\sum_{j=1}^n w_j C_j$ is minimized and the following constraints are satisfied:

1. *Assignment constraints*: every part of every job should be assigned to exactly one position of a schedule and every position should be occupied by exactly one job part.
2. *Sequence constraints*: for every job j positions of its parts $1, 2, \dots, p$ in a schedule should stay in the following sequence: $t_1^j, t_2^j, \dots, t_p^j$. Note that $t_p^j = C_j$.
3. *Release date constraints*: the first part of every job j should be assigned to the time moment which is not earlier than the release date of this job: $t_1^j \geq r_j$.

In the assignment problem AP_L , we minimize the following objective function:

$$f_{AP_L} = \sum_{i=1}^{np} \sum_{t=1}^{np} c_{it} x_{it} = \sum_{j=1}^n \sum_{t=1}^{np} c_{pj,t} x_{pj,t}$$

Table 3 SP and AP_L solutions

1	p	2p	3p	np-p	np
2	n	n-1	...	3	1

1	p	p+1p+2				np
	2	n	n-1	...	3	...

because when $i \bmod p > 0$ either $c_{it} = 0$ or $c_{it} = \infty$ and $x_{it} = 0$. Since $x_{pj,t} = 1$ only for $t = t_p^j$, $c_{pj,t} = w_j t$, and $t_p^j = C_j$ then:

$$f_{AP_L} = \sum_{j=1}^n c_{pj,t_p^j} = \sum_{j=1}^n w_j t_p^j = \sum_{j=1}^n w_j C_j$$

So the objective function of AP_L is the same as for SP . *Assignment constraints* of SP are equivalent to constraints (3) and (4) of AP_L . *Release date constraints* of SP are always satisfied in AP_L solution because $c_{p(j-1),t} = \infty$ for $t < r_j$, and thus $t_1^j < r_j$ always gives $f_{AP} = \infty$. *Sequence constraints* of SP are not satisfied by AP_L . But these constraints can be represented by the following linear inequalities:

$$\forall j = \overline{1, n} \quad \forall t_0 = \overline{r_j + p - 1, np - 1} \quad \sum_{t=t_0+1}^{np} \sum_{i=1+p(j-1)}^{pj} x_{it} \leq p(1 - x_{pj,t_0}) \quad (7)$$

So we have got that AP_L is a relaxation of SP in which *sequence constraints* are removed. This proves that the optimal solution of AP_L gives a lower bound to the optimal solution of SP : $f_{AP} \leq f_{SP}$. \square

Our goal is to determine how good is the lower bound given by the AP_L . For this purpose, we are going to find the worst case for this bound. The worst case is the scheduling problem for which the AP_L solution differs from the SP solution as much as possible.

Theorem 2 *The worst case for the AP_L (for which the ratio f_{AP_L}/f_{SP} is minimal) is the scheduling problem with $p < n$, $w_1 \leq w_2 \leq \dots \leq w_n$, $r_1 = r_2 = 1$, $\forall j = \overline{3, n}$ $r_j = n - j + 2$.*

Proof Here we provide only some ideas and a scheme of the proof. The complete and detailed proof will be presented in a full journal paper. For the described scheduling problem, SP and AP_L solutions are shown in Table 3.

The proof is based on the following observation. If in the optimal SP schedule job n with the maximal weight w_n starts from time $(t + 1)$, then it ends at time $(t + p)$, because the job with the maximal weight cannot be preempted in the optimal schedule. It is not difficult to prove that preemption of any job by a job with a smaller weight always increases the value of the objective function. It is also easy to prove

that in this case its release date $r_n \geq t - p + 2$ if $t \geq p$ (if $t < p$, then r_n can have any value and job n in the AP_L solution will be close to its position in the SP solution). So in the optimal AP_L schedule the last part of job n will be placed at the earliest possible time $r_n + p - 1 \geq t + 1$. The relative difference of the objective functions of SP and AP_L solutions only for job n (assuming that all other jobs last parts are placed on the same positions in AP_L and SP solutions) is as follows.

$$\frac{f_{SP} - f_{AP_L}}{f_{SP}} = \frac{((t + p) - (t + 1))w_n}{(t + p)w_n} = \frac{p - 1}{t + p}$$

This difference is maximal when t is minimal. Since $t \geq p$ then for job n the worst case of AP_L solution is realized when $t = p$. In this case release date $r_n = 2$, job n starts at time $p + 1$ and ends at time $2p$ in the optimal SP schedule. Since job n makes the greatest contribution to the objective function it is reasonable to consider it first, then job $n - 1$, and so on. Following this logic after freezing the position of job n in the optimal SP schedule, we find that in the worst case job $n - 1$ starts at time $2p + 1$, ends at time $3p$, and $r_{n-1} = 3$. An so on till job 3 which should start at time $np - 2p + 1$, end at time $np - p$, and have release date $r_3 = n - 1$ (see Table 3). The remaining jobs 2 and 1 can have release dates $r_1 = r_2 = 1$ and can be placed to time intervals $[1, p]$ and $[np - p + 1, np]$ in the optimal SP schedule. \square

The worst case for the AP_L solution immediately gives us the estimation of the AP_L lower bound precision.

Theorem 3 *The AP_L optimal solution provides a value of the objective function f_{AP_L} which is not less than $(\frac{1}{p} + \frac{2}{n+1} - \frac{2}{p(n+1)})f_{SP}$.*

Proof According to Theorem 2, the worst case for AP_L is the scheduling problem with $p < n$, $w_1 \leq w_2 \leq \dots \leq w_n$, $r_1 = r_2 = 1$, $\forall j = 3, n$ $r_j = n - j + 2$. The AP_L and SP solutions for this problem are shown in Table 3. The values of the objective function and the relative difference are as follows.

$$\begin{aligned} f_{SP} &= \sum_{i=1}^{n-2} (i+1)pw_{n+1-i} + pw_2 + npw_1 \\ f_{AP_L} &= \sum_{i=1}^{n-2} (p+i)w_{n+1-i} + pw_2 + npw_1 \\ \frac{f_{SP} - f_{AP_L}}{f_{SP}} &= \frac{\sum_{i=1}^{n-2} i(p-1)w_{n+1-i}}{\sum_{i=1}^{n-2} (i+1)pw_{n+1-i} + pw_2 + npw_1} \end{aligned}$$

The difference is maximal when $w_1 = w_2 = 0$, and so we have:

$$\frac{f_{SP} - f_{AP_L}}{f_{SP}} \leq \frac{\sum_{i=1}^{n-2} i(p-1)w_{n+1-i}}{\sum_{i=1}^{n-2} (i+1)pw_{n+1-i}}$$

Let us consider this ratio only for one job i without others.

$$\frac{i(p-1)w_{n+1-i}}{(i+1)pw_{n+1-i}} = \frac{ip-i}{ip+p} = 1 - \frac{1}{p} - \frac{p-1}{(i+1)p}$$

This value is maximal when i is maximal: $i = n-2$. This means that the $(n-2)$ -th summand makes the greatest contribution to the value of the whole expression. If all the weights except w_3 are zero, then we will get the maximum, but $w_3 \leq w_4 \leq \dots \leq w_n$. So to make the contribution of the last summand as great as possible we should set all the weights except w_3 as small as possible: $w_n = w_{n-1} = \dots = w_3$. In this case, we have:

$$\begin{aligned} \frac{f_{SP} - f_{AP_L}}{f_{SP}} &= \frac{(p-1)w_3 \sum_{i=1}^{n-2} i}{pw_3 \sum_{i=1}^{n-2} (i+1)} = \frac{(p-1)(n-1)}{p(n+1)} \\ &= 1 - \left(\frac{1}{p} + \frac{2}{n+1} - \frac{2}{p(n+1)} \right) \end{aligned}$$

These are only intuitive considerations which lead to the estimation for the lower bound. But this estimation should be carefully proved. First, we need to prove the following auxiliary inequality.

$$\sum_{i=1}^{n-2} (i+1)pw_{n+1-i} \geq \frac{1}{2}p(n-2)(n+1)w_3 \quad (8)$$

Since $\forall i = \overline{1, n-2} \ w_{n+1-i} \geq w_3$, we have:

$$\sum_{i=1}^{n-2} (i+1)pw_{n+1-i} \geq \sum_{i=1}^{n-2} (i+1)pw_3 = \frac{1}{2}p(n-2)(n+1)w_3$$

Let us use the mathematical induction to prove the estimation for the lower bound:

$$\frac{f_{SP} - f_{AP_L}}{f_{SP}} \leq \frac{\sum_{i=1}^{n-2} i(p-1)w_{n+1-i}}{\sum_{i=1}^{n-2} (i+1)pw_{n+1-i}} \leq \frac{(p-1)(n-1)}{p(n+1)}$$

For $n = 3$, this inequality is true:

$$\frac{(p-1)w_3}{2pw_3} \leq \frac{2(p-1)}{4p}$$

Now we assume that the inequality is true for $n = k$ and prove that then it is also true for $n = k+1$.

$$\frac{\sum_{i=1}^{k-1} i(p-1)w_{k+2-i}}{\sum_{i=1}^{k-1} (i+1)pw_{k+2-i}} = \frac{\sum_{i=1}^{k-2} i(p-1)w_{k+2-i} + (k-1)(p-1)w_3}{\sum_{i=1}^{k-2} (i+1)pw_{k+2-i} + kpw_3}$$

Let us make the following denotations.

$$a = \sum_{i=1}^{k-2} i(p-1)w_{k+2-i}$$

$$b = \sum_{i=1}^{k-2} (i+1)pw_{k+2-i}$$

Since our statement is true for $n = k$, we have:

$$\frac{a}{b} \leq \frac{(p-1)(k-1)}{p(k+1)} \implies a \leq \frac{(p-1)(k-1)}{p(k+1)}b$$

We need to prove that:

$$\frac{a + (k-1)(p-1)w_3}{b + kpw_3} \leq \frac{(p-1)k}{p(k+2)}$$

Using the statement for $n = k$, we have

$$\begin{aligned} \frac{a + (k-1)(p-1)w_3}{b + kpw_3} &\leq \frac{\frac{(p-1)(k-1)}{p(k+1)}b + (k-1)(p-1)w_3}{b + kpw_3} \\ &= \frac{p-1}{p} \cdot \frac{(k-1)b + (k-1)(k+1)pw_3}{(k+1)b + k(k+1)pw_3} \end{aligned}$$

So it is enough to prove that:

$$\begin{aligned} \frac{(k-1)b + (k-1)(k+1)pw_3}{(k+1)b + k(k+1)pw_3} &\leq \frac{k}{(k+2)} \\ \iff (k+2)(k-1)b + (k+2)(k-1)(k+1)pw_3 &\leq k(k+1)b + k^2(k+1)pw_3 \\ \iff -2b &\leq (k+1)(k-2)pw_3 \\ \iff b &\geq \frac{1}{2}p(k-2)(k+1)w_3 \end{aligned}$$

Since the auxiliary inequality (8) is true for any n and any weights w_j , we have the inequality which we need to prove.

$$b = \sum_{i=1}^{k-2} (i+1)pw_{k+2-i} \geq \frac{1}{2}p(k-2)(k+1)w_3$$

This proves the inductive step and so our statement is true.

$$\frac{f_{SP} - f_{APL}}{f_{SP}} \leq \frac{(p-1)(n-1)}{p(n+1)} = 1 - \left(\frac{1}{p} + \frac{2}{n+1} - \frac{2}{p(n+1)} \right)$$

Table 4 SP and AP_L solutions for a worst case example

1	4	8	12	392	396
2	99	98	...	3	1

1	4	5	6	396		
	2	99	98	...	3	...

$$\Rightarrow \frac{f_{AP_L}}{f_{SP}} \geq \frac{1}{p} + \frac{2}{n+1} - \frac{2}{p(n+1)}$$

This ends the proof of the estimation for the lower bound f_{AP_L} . \square

Let us show the worst case for the AP_L solution on an example with $n = 99$ jobs, $p = 4$, $r_1 = r_2 = 1$, $r_{99} = 2$, $r_{98} = 3, \dots, r_3 = 98$, and weights $w_1 = \epsilon$, $w_j = 1 + \epsilon j$ for $j = 2, 99$, where ϵ is a very small value. The optimal SP and AP_L solutions for this problem are shown in Table 4.

Neglecting the value of ϵ , we have $f_{SP} = 4 + 8 + \dots + 4 \cdot 98 = 19404$, $f_{AP_L} = 4 + 5 + \dots + 101 = 5145$. The ratio of f_{AP_L} to f_{SP} and its estimation are the following.

$$\frac{f_{AP_L}}{f_{SP}} = \frac{5145}{19404} \approx 0.26515$$

$$\frac{f_{AP_L}}{f_{SP}} \geq \frac{1}{p} + \frac{2}{n+1} - \frac{2}{p(n+1)} = \frac{1}{4} + \frac{2}{100} - \frac{2}{400} = 0.265$$

As one can see in the worst case (when n is big) AP_L lower bound can be p times smaller than the optimal SP solution f_{AP_L} .

4 Upper Bound

The main drawback of the assignment problem AP_L is that it does not include the sequence constraints. However, it is possible to incorporate the sequence constraints to the assignment problem. For this purpose, we set the assignment costs c_{it} to be equal to $w_{[(i-1)/p]+1}t$ not only for the last part of a job (when $i \bmod p = 0$) but for all its parts. For the example from Table 1, the assignment problem will have the cost matrix shown in Table 5.

The optimal assignment problem solution is highlighted with gray color. Now it satisfies all the constraints of the original SP problem. Unfortunately, it is not an optimal solution for the SP problem, because its objective function differs from the scheduling problem SP objective function. The following theorem proves that this assignment problem provides an upper bound for the optimal solution of SP . We will denote this assignment problem as AP_U .

Table 5 Assignment problem

	1	2	3	4	5	6	7	8	9	10	11	12
1 (job 1, part 1)	1	2	3	4	5	6	7	8	9	∞	∞	∞
2 (job 1, part 2)	∞	2	3	4	5	6	7	8	9	10	∞	∞
3 (job 1, part 3)	∞	∞	3	4	5	6	7	8	9	10	11	∞
4 (job 1, part 4)	∞	∞	∞	4	5	6	7	8	9	10	11	12
5 (job 2, part 1)	∞	∞	∞	8	10	12	14	16	18	∞	∞	∞
6 (job 2, part 2)	∞	∞	∞	∞	10	12	14	16	18	20	∞	∞
7 (job 2, part 3)	∞	∞	∞	∞	∞	12	14	16	18	20	22	∞
8 (job 2, part 4)	∞	∞	∞	∞	∞	∞	14	16	18	20	22	24
9 (job 3, part 1)	∞	∞	∞	∞	∞	∞	21	24	27	∞	∞	∞
10 (job 3, part 2)	∞	∞	∞	∞	∞	∞	∞	24	27	30	∞	∞
11 (job 3, part 3)	∞	∞	∞	∞	∞	∞	∞	∞	27	30	33	∞
12 (job 3, part 4)	∞	∞	∞	∞	∞	∞	∞	∞	∞	30	33	36

Theorem 4 *The solution of the following assignment problem AP_U gives an upper bound to the solution of the scheduling problem $1|p_{mtn}; p_j = p; r_j| \sum w_j C_j$.*

$$\min \sum_{i=1}^{np} \sum_{t=1}^{np} c_{it} x_{it} \quad (9)$$

$$\sum_{t=1}^{np} x_{it} = 1 \quad \forall i = \overline{1, np} \quad (10)$$

$$\sum_{i=1}^{np} x_{it} = 1 \quad \forall t = \overline{1, np} \quad (11)$$

$$x_{it} \in \{0, 1\} \quad (12)$$

where

$$c_{it} = \begin{cases} w_j t, & \text{if } t \in [r_j + i', (n-1)p + 1 + i'] \\ \infty, & \text{if } t \notin [r_j + i', (n-1)p + 1 + i'] \end{cases} \quad (13)$$

$$j = \lceil (i-1)/p \rceil + 1, \quad i' = (i-1) \bmod p$$

Proof We provide here only the basic ideas of the proof. The complete detailed proof will be presented in a full journal paper. The ideas are the following.

1. In the AP_U solution, a job with a smaller weight being processed will be interrupted by a job with a greater weight immediately when this job becomes available.
2. In the AP_U solution, a job with a smaller weight never interrupts a job with a greater weight.
3. If in the AP_U solution job i is interrupted by job j , then job i will be completed after job j .

Table 6 SP and AP_U solutions

1	p	2p					np
1	2	...				n	

1	p-1	2p-2			np-n-p+1	np-n+1	np-n+2	np-n+3	np-1	np
1	2	...	n-1	n	n-1	n-2	...	2	1	

4. The AP_U solution can be obtained by assigning at every time moment the available job with the greatest weight.

It follows from statement 4 that the AP_U solution is a feasible solution for the SP problem. So the value of the SP objective function for this solution f_{AP_U} is an upper bound for the optimal value f_{SP} of this objective function. \square

We want to determine how good is the upper bound given by the AP_U . For this purpose, we are going to find the worst case for this bound. The worst case is the scheduling problem for which the AP_U solution differs from the SP solution as much as possible in terms of the SP objective function.

Theorem 5 *The worst case for the AP_U (for which the ratio f_{AP_U}/f_{SP} is maximal) is the scheduling problem with $p > n$, $w_1 \leq w_2 \leq \dots \leq w_n$, $\forall j = \overline{1, n}$ $r_j = (p-1)(j-1) + 1$.*

Proof Here we provide only some ideas and a scheme of the proof. The complete and detailed proof will be presented in a full journal paper. For the described scheduling problem, SP and AP_U solutions are shown in Table 6. The proof is based on the following statements.

1. If in the SP solution job i is interrupted by job j , then job i will be completed after job j .
2. If in the SP solution job i is interrupted by job j , then it is made in time moment r_j .
3. In the SP solution, the job which first part is assigned to some time moment has the greatest weight among all the jobs available at this moment and not started yet.
4. If in the SP solution job i is interrupted by job j , then the same is true for the AP_U solution.
5. If in the SP solution whole job j is scheduled before whole job i with a smaller weight, then the same is true for the AP_U solution.

From statements 4 and 5, it follows that the greatest difference between the AP_U and SP solutions is reached when the SP solution does not have interrupts. \square

The worst case for the AP_U solution immediately gives us the estimation of the AP_U upper bound precision.

Theorem 6 *The AP_U optimal solution provides a value of the SP objective function f_{AP_U} which is not greater than $(2 - \frac{1}{p} - \frac{2}{n+1} + \frac{2}{p(n+1)})f_{SP}$.*

Proof According to Theorem 5, the worst case for AP_U is the scheduling problem with $p > n$, $w_1 \leq w_2 \leq \dots \leq w_n$, $\forall j = \overline{1, n}$ $r_j = (p-1)(j-1) + 1$. The AP_U and SP solutions for this problem are shown in Table 6. The values of the objective function and the ratio are as follows.

$$\begin{aligned} f_{SP} &= \sum_{i=1}^n ipw_i \\ f_{AP_U} &= \sum_{i=1}^n (np+1-i)w_i \\ \frac{f_{AP_U}}{f_{SP}} &= \frac{\sum_{i=1}^n (np+1-i)w_i}{\sum_{i=1}^n ipw_i} \end{aligned}$$

Let us consider this ratio only for one job i without others.

$$\frac{(np+1-i)w_i}{ipw_i} = \frac{np+1-i}{ip} = -\frac{1}{p} + \frac{np+1}{ip}$$

This value is maximal when i is minimal: $i = 1$. This means that the first summand makes the greatest contribution to the value of the whole expression. If all the weights except w_1 are zero, then we will get the maximum, but $w_1 \leq w_2 \leq \dots \leq w_n$. So to make the contribution of the first summand as great as possible we should set all the weights except w_1 as small as possible: $w_n = w_{n-1} = \dots = w_1$. In this case, we have:

$$\frac{f_{AP_U}}{f_{SP}} = \frac{\sum_{i=1}^n (np+1-i)w_1}{\sum_{i=1}^n ipw_1} = \frac{2np-n+1}{p(n+1)} = 2 - \frac{1}{p} - \frac{2}{n+1} + \frac{2}{p(n+1)}$$

These are only intuitive considerations which lead to the estimation for the upper bound. But this estimation should be carefully proved. Let us use the mathematical induction to prove the following inequality.

$$\frac{f_{AP_U}}{f_{SP}} = \frac{\sum_{i=1}^n (np+1-i)w_i}{\sum_{i=1}^n ipw_i} \leq \frac{2np-n+1}{p(n+1)}$$

For $n = 1$, our statement is true:

$$\frac{f_{AP_U}}{f_{SP}} = \frac{(p+1-1)w_1}{pw_1} = 1, \quad \frac{2np-n+1}{p(n+1)} = \frac{2p}{2p} = 1$$

Assuming that the statement is true for $n = k$:

$$\frac{\sum_{i=1}^k (kp+1-i)w_i}{\sum_{i=1}^k ipw_i} \leq \frac{2kp-k+1}{p(k+1)}$$

we need to prove that it is also true for $n = k + 1$:

$$\frac{\sum_{i=1}^{k+1} ((k+1)p + 1 - i)w_i}{\sum_{i=1}^{k+1} ipw_i} \leq \frac{2(k+1)p - k}{p(k+2)}$$

We use the following denotations.

$$a = \sum_{i=1}^k (kp + 1 - i)w_i$$

$$b = \sum_{i=1}^k ipw_i$$

Since we assume that the statement is true for $n = k$, we have:

$$\frac{a}{b} \leq \frac{2kp - k + 1}{p(k+1)} \implies a \leq \frac{2kp - k + 1}{p(k+1)}b$$

For $n = k + 1$, we have:

$$\begin{aligned} & \frac{\sum_{i=1}^{k+1} ((k+1)p + 1 - i)w_i}{\sum_{i=1}^{k+1} ipw_i} \\ &= \frac{\sum_{i=1}^k (kp + 1 - i)w_i + \sum_{i=1}^k pw_i + ((k+1)p - k)w_{k+1}}{\sum_{i=1}^k ipw_i + (k+1)w_{k+1}} \\ &= \frac{a + \sum_{i=1}^k pw_i + ((k+1)p - k)w_{k+1}}{b + (k+1)w_{k+1}} \\ &\leq \frac{\frac{2kp - k + 1}{p(k+1)}b + \sum_{i=1}^k pw_i + ((k+1)p - k)w_{k+1}}{b + (k+1)w_{k+1}} \end{aligned}$$

So we need to prove the following:

$$\begin{aligned} & \frac{(2kp - k + 1)b + p(k+1) \sum_{i=1}^k pw_i + p(k+1)((k+1)p - k)w_{k+1}}{p(k+1)(b + (k+1)w_{k+1})} \\ &\leq \frac{2(k+1)p - k}{p(k+2)} \end{aligned}$$

Multiplying this inequality by the denominators, we get an equivalent one:

$$\begin{aligned} & (p(k+2)(2kp - k + 1) - 2(k+1)^2p^2 + k(k+1)p)b \\ & + p^2(k+1)(k+2) \sum_{i=1}^k pw_i + p^2(k+1)(k+2)((k+1)p - k)w_{k+1} \end{aligned}$$

$$\begin{aligned}
& -2p^3(k+1)^3w_{k+1} + p^2k(k+1)^2w_{k+1} \leq 0 \\
\iff & (2p - 2p^2)b + p^2(k+1) \left((k+2) \sum_{i=1}^k pw_i - (k^2p + kp + k)w_{k+1} \right) \\
& \leq 0
\end{aligned}$$

Substituting the expression for b we come to the following inequality which we need to prove:

$$\begin{aligned}
& k(k+1)(kp + p + 1)w_{k+1} - \left(p(k+1)(k+2) \sum_{i=1}^k w_i - 2(p-1) \sum_{i=1}^k iw_i \right) \geq 0 \\
\iff & k(k+1)(kp + p + 1)w_{k+1} - \left(\sum_{i=1}^k (p(k+1)(k+2) - 2(p-1)i)w_i \right) \\
& \geq 0
\end{aligned}$$

It is clear that $p(k+1)(k+2) - 2(p-1)i \geq 0$ and so the maximum value of the sum in the brackets is reached when $w_i = w_{k+1}$ (because $w_i \leq w_{k+1}$). Thus, we have:

$$\begin{aligned}
& k(k+1)(kp + p + 1)w_{k+1} - \left(\sum_{i=1}^k (p(k+1)(k+2) - 2(p-1)i)w_i \right) \\
& \geq k(k+1)(kp + p + 1)w_{k+1} - \left(\sum_{i=1}^k (p(k+1)(k+2) - 2(p-1)i)w_{k+1} \right) \\
& = k(k+1)(kp + p + 1)w_{k+1} - (p(k+1)(k+2)k - (p-1)k(k+1))w_{k+1} \\
& = k(k+1)(kp + p + 1 - kp - 2p + p - 1)w_{k+1} = 0
\end{aligned}$$

This proves the inductive step and so our statement is true.

$$\frac{f_{AP_U}}{f_{SP}} \leq \frac{2np - n + 1}{p(n+1)} = 2 - \frac{1}{p} - \frac{2}{n+1} + \frac{2}{p(n+1)}$$

This ends the proof of the estimation for the upper bound f_{AP_U} . \square

We show the worst case for the AP_U solution on an example with $n = 99$ jobs, $p = 100$, $r_1 = 1, r_2 = 100, r_3 = 199, \dots, r_{99} = 9703$, and weights $w_j = 1 + \epsilon j$ for $j = \overline{1, 99}$, where ϵ is a very small value. The optimal SP and AP_L solutions for this problem are shown in Table 7.

Neglecting the value of ϵ we have $f_{SP} = 100 + 200 + \dots + 100 \cdot 99 = 495000$, $f_{AP_U} = 9802 + 9803 + \dots + 9900 = 975249$. The ratio of f_{AP_U} to f_{SP} and its esti-

Table 7 SP and AP_L solutions for a worst case example

1	100	200	9900						
1	2	...	99						

1	99	198	9702	9802	9803	9804	9899	9900	
1	2	...	98	99	98	97	...	2	1

mation are the following.

$$\frac{f_{AP_U}}{f_{SP}} = \frac{975249}{495000} = 1.9702$$

$$\frac{f_{AP_U}}{f_{SP}} \leq 2 - \frac{1}{p} - \frac{2}{n+1} + \frac{2}{p(n+1)} = 2 - \frac{1}{100} - \frac{2}{100} + \frac{2}{10000} = 1.9702$$

As one can see in the worst case AP_U upper bound cannot be more than two times greater than the optimal value f_{SP} .

5 Concluding Remarks

In this chapter, we have studied the relationships between the linear assignment problem (AP) and the preemptive single machine scheduling problem $1|pmtn; p_j = p; r_j| \sum w_j C_j$ of minimizing the total weighted completion time with equal processing times and arbitrary release dates (further abbreviated as SP). The relationships between these problems might be expressed as follows. SP is equivalent to the modified AP with adjusted assignment costs (reflecting the release dates of the jobs) and additional sequence constraints. We have derived lower and upper bounds based on the relationships between the AP and SP . Our theoretical analysis of the lower bound shows that in the case with equal processing times $p = 2$ and infinitely many jobs the lower bound is just a half ($1/2$) of the unknown optimal value f_{SP} of the objective function for SP . And it tends to become worse when the processing time p is increased. Meanwhile for the same case with $p = 2$ and infinitely many jobs the AP-based upper bound tends to come as close as possible to $3/2 f_{SP}$. And it tends to reach $2 f_{SP}$ value when the processing time p is increased. Note that the best known approximation ratio for a similar NP-hard problem with arbitrary processing times evaluates by 1.47 (see Goemans et al. (2000) [5]).

The first direction of our future research is to empirically compare the quality of our upper bound and the 1.47 upper bound of Goemans et al. (2000) [5] algorithm for the so called equal binary processing times $p = 2$ reflecting the most popular in practice case of scheduling jobs with two essential units: the start unit and the completion unit. Another direction of our research is to derive similar AP-based upper and lower bounds for other single machine scheduling problems including the NP-hard version of SP with arbitrary processing times (see Lenstra et al. (1997) [9]).

Acknowledgements The authors are partially supported by LATNA Laboratory, National Research University Higher School of Economics (NRU HSE), Russian Federation government grant, ag. 11.G34.31.0057.

Boris Goldengorin's research was partially supported by the Exchange Visiting Program Number P-1-01285 carried out at the Center of Applied Optimization, University of Florida, USA.

References

1. Baker, K.R.: Introduction to Sequencing and Scheduling. Wiley, New York (1974), 305 pp.
2. Baptiste, P.: Scheduling equal-length jobs on identical parallel machines. *Discrete Appl. Math.* **103**(1), 21–32 (2000)
3. Bouma, H.W., Goldengorin, B.: A polytime algorithm based on a primal LP model for the scheduling problem $1|pmtn; p_j = 2; r_j|\sum w_j C_j$. Proceedings of the 2010 American Conference on Applied Mathematics, AMERICAN-MATH'10, Stevens Point, Wisconsin, USA, pp. 415–420. World Scientific and Engineering Academy and Society (WSEAS) (2010)
4. Brucker, P., Kravchenko, S.A.: Scheduling jobs with equal processing times and time windows on identical parallel machines. *J. Sched.* **11**, 229–237 (2008)
5. Goemans, M.X., Wein, J.M., Williamson, D.P.: A 1.47-approximation algorithm for a preemptive single-machine scheduling problem. *Oper. Res. Lett.* **26**, 149–154 (2000)
6. Graham, R.L., Lawler, E.L., Lenstra, J.K., Rinnooy Kan, A.H.G.: Optimization and approximation in deterministic sequencing and scheduling: a survey. *Ann. Discrete Math.*, **5**, 287–326 (1979)
7. Herrman, J., Lee, C., Snowden, J.: A classification of static scheduling problems. In: Pardalos, P.M. (ed.) *Complexity in Numerical Optimization*, pp. 203–253. World Scientific, Singapore (1993)
8. Labetoulle, J., Lawler, E.L., Lenstra, J.K., Rinnooy Kan, A.H.G.: Preemptive scheduling of uniform machines subject to release dates. In: *Progress in Combinatorial Optimization*, pp. 245–261. Academic Press, Toronto (1984)
9. Lenstra, J.K., Rinnooy Kan, A.H.G., Brucker, P.: Complexity of machine scheduling problems. studies in integer programming. In: Hammer, P.L., Johnson, E.L., Korte, B.H., Nemhauser, G.L. (eds.) *Annals of Discrete Mathematics*, vol. 1, pp. 343–362. North-Holland, Amsterdam (1977)
10. Pinedo, M.L.: *Scheduling: Theory, Algorithms, and Systems*, 4th edn. Springer, Berlin (2012), 673 pp.
11. Smith, W.E.: Various optimizers for single-stage production. *Nav. Res. Logist. Q.* **3**, 59–66 (1956)

Models, Algorithms, and Technologies for Network
Analysis

Proceedings of the Second International Conference on
Network Analysis

Goldengorin, B.; Kalyagin, V.A.; Pardalos, P. (Eds.)

2013, XIV, 217 p. 44 illus., 13 illus. in color., Hardcover

ISBN: 978-1-4614-8587-2