A. Aggregation of identical jobs

In the available real life test instances, usually, there are identical jobs. We call two jobs a and b identical if $p_a = p_b$, $l_a = l_b$, $\alpha_a = \alpha_b$, $\beta_a = \beta_b$, and $n(a) = n(b)$. In order to decrease the size of the formulation (TIA) we will aggregate jobs into groups. We partition the set N of jobs into the set $Q = \{1, \ldots, q\}$ of groups $\{G_k\}_{k \in Q}$ of identical jobs. In the following theorem, we show that there exists an optimal schedule, in which the operations of jobs in the same group alternate.

Theorem 1. *There exists an optimal schedule, in which the j^{th} operations of the jobs in the same group are processed in the same order for all j.*

Proof. Consider an optimal schedule S. If it satisfies the condition of the theorem, we are done.

Otherwise, there should exist a triple $\langle i, k; j\rangle$ such that $S_{i,j-1} < S_{k,j-1} < S_{k,j} < S_{ij}$ (we will call such triples *non-alternating*). Consider such a triple $\langle i, k; j\rangle$ with the maximum possible j.

We now transform schedule S into S' by setting

$$S'_{ij'} = S_{kj'} \quad \text{and} \quad S'_{jk'} = S_{ij'} \quad \forall j' : \; j \le j' \le n(i) = n(k).$$

Obviously, S' is a feasible schedule (as $p_i = p_k$).

First, we show that the cost $F(S')$ of schedule S' is not bigger than the cost $F(S)$ of schedule S. Taking into account that $\delta_i(x) = \delta_k(x)$, and setting $\epsilon = S_{ij} - S_{kj} > 0$, we have

$$\begin{aligned} F(S) - F(S') \;=\; & \delta(\underbrace{S_{ij} - S_{i,j-1}}_{=\Delta i}) + \delta(\underbrace{S_{kj} - S_{k,j-1}}_{=\Delta k}) \\ & - \delta(\underbrace{S_{kj} - S_{i,j-1}}_{=\Delta i - \epsilon}) - \delta(\underbrace{S_{ij} - S_{k,j-1}}_{=\Delta k + \epsilon}). \end{aligned}$$

As the function $\delta_i(x)$ is convex, $\Delta k < \Delta i - \epsilon$, and $\Delta k + \epsilon < \Delta i$, we have $\delta_i(\Delta k) + \delta_i(\Delta i) \ge \delta_i(\Delta i - \epsilon) + \delta_i(\Delta k + \epsilon)$, and $F(S) \ge F(S')$.

Secondly, we show that number of alternating triples has been increased. Triple $\langle i, k; j\rangle$ which was non-alternating in S became alternating in S'. It is easy to see that, if an alternating triple $\langle i, k'; j\rangle$, $k' \ne k$, (or $\langle i', k; j\rangle$, $i' \ne i$) in S became non-alternating in S', then triple $\langle k, k'; j\rangle$ (respectively,

triple $\langle i, i'; j\rangle$) was non-alternating in S and become alternating in S'. The alternating status of other triples in S' stayed the same as in S. So, the number of alternating triples has been strictly increased in S'. Therefore, repeating the same transformation a finite number of times, we can obtain a desired schedule without increasing the cost of schedule S. □

Using the fact just proved, in the next theorem, we show that the penalty functions γ_i for the jobs i in group G_k, $k \in Q$, can be "aggregated" into a single penalty function θ_k for the group. Let $G_{kt} = \{i \in G_k : \ t \in \Delta_i\}$. We renumber the jobs such that, for each $k \in Q$, $G_k = \{i'_k, i'_k + 1, \ldots, i''_k\}$, $S_{i'_k 0} < S_{i'_k+1,0} < \ldots < S_{i''_k}$. We set $\bar{H}_k = H_{i'_k}$, and

$$\theta_k(t) = \max\left\{\alpha_i\left(\sum_{i\in G_k} n_i(t) - |G_{kt}|\right)\ ,\ \beta_i\left(|G_{kt}| - \sum_{i\in G_k} n_i(t)\right)\right\}.$$

Theorem 2. *Consider a group G_k, $k \in Q$, of jobs and a schedule $\{S_{i0}, S_{i1}, \ldots, S_{i,n(i)}\}_{i\in G_k}$, in which the operations of jobs in G_k alternate. Then, for all $t \in \bar{H}_k$,*

$$\sum_{i\in G_{kt}} \gamma_i(t) = \theta_k(t).$$

Proof. To prove the proposition, we need to show that, given $t \in \bar{H}_k$,

$$\sum_{i\in G_k} n_i(t) \geq |G_{kt}| \quad \Leftrightarrow \quad n_i(t) \geq 1,\ \forall i \in G_{kt}, \tag{1}$$

$$\sum_{i\in G_k} n_i(t) \leq |G_{kt}| \quad \Leftrightarrow \quad n_i(t) \leq 1,\ \forall i \in G_{kt}, \tag{2}$$

The implication "$\Leftarrow$" of (1) is obvious. We will now prove the implication "$\Rightarrow$" (1) by contradiction. Suppose that $\sum_{i\in G_k} n_i(t) \geq |G_{kt}|$ and there exists a job $i' \in G_{kt}$ such that $n_{i'}(t) = 0$. Then, there is a job $i'' \in G_{kt}$ such that $n_{i''}(t) \geq 2$. This means that, in the schedule, between two operations of job i'', there is no operation of job i' and an operation of job i' starts later than time t (as $i' \in G_{kt}$ and $n_{i'}(t) = 0$). This contradicts the condition of the proposition which states that the operations of jobs i' and i'' alternate.

The implications in (2) can be proved in the same way. □

Finally, to find an optimal solution of the problem, we can limit the search to the set of "alternating" schedules and use the penalty function

$\sum_{k\in Q}\sum_{t\in\bar{H}_k}\theta_k(t)$ instead of $\sum_{i\in N}\sum_{t\in H_i}\gamma_i(t)$. We will now modify the formulation (TIA). Integer variables X_{kt}, $k\in Q$, $t\in H$, are now equal to the number of operations of jobs in G_k started before or at time t. Continuous variable W_{kt}, $k\in Q$, $t\in\bar{H}_k$, now represents the value $\theta_k(t)$. Here we again need to introduce additional variables E in order to compute $|G_{kt}|$, $k\in Q$, $t\in H$. Binary variable E_{kst}, $k\in Q$, $s\in\{1,\ldots,|Q_k|\}$, $t\in H_k$, takes value 1 if and only if $|G_{kt}|\geq s$. Again, we do not need to impose the integrality on these variables. We now present the aggregated alternative time indexed ILP formulation. For each $k\in Q$, we set $\bar{p}_k=p_{i'_k}$, $\bar{l}_k=l_{i'_k}$, $\bar{\alpha}_k=\alpha_{i'_k}$, $\bar{\beta}_k=\beta_{i'_k}$, $\bar{n}(k)=n(i'_k)$.

$$\min \quad \sum_{k\in Q}\sum_{t\in\bar{H}_k} W_{kt} \tag{3}$$

$$s.t. \quad X_{k,S_{i0}} = i''_k - i + 1, \quad \forall k\in Q,\ i\in G_k, \tag{4}$$

$$X_{k,S_{i0}-1} = i''_k - i, \quad \forall k\in Q,\ i\in G_k, \tag{5}$$

$$X_{k,t-1}\leq X_{kt}, \quad \forall k\in Q,\ t\in H\setminus\{0\}, \tag{6}$$

$$X_{k,h-\bar{p}_i} = |Q_k|\cdot(\bar{n}(k)+1), \quad \forall k\in Q, \tag{7}$$

$$\sum_{k\in Q} X_{kt} - \sum_{\substack{k\in Q,\\ t-\bar{p}_k\geq 0}} X_{k,t-\bar{p}_k}\leq 1, \quad \forall t\in H, \tag{8}$$

$$E_{k1t}\geq X_{k,t-\bar{l}_k+\bar{p}_k} - X_{k,t-\bar{l}_k}, \quad \forall k\in Q,\ t\in\bar{H}_k,\ t\geq S_{i''_k,0}+\bar{l}_k, \tag{9}$$

$$E_{kst}\geq X_{k,t-\bar{l}_k+\bar{p}_k} + X_{k,t-\bar{l}_k} + E_{k,s-1,t+\bar{p}_k} - 1, \quad \forall k\in Q,\ s\in\{2,\ldots,|Q_k|\},\ t\in\bar{H}_k,\ t\geq S_{i''_k,0}+\bar{l}_k, \tag{10}$$

$$E_{kst}\leq E_{k,s,t-1}\leq 1, \quad \forall k\in Q,\ s\in\{1,\ldots,|Q_k|\},\ t\in\bar{H}_k,\ t> S_{i''_k,0}, \tag{11}$$

$$W_{kt}\geq\bar{\alpha}_k\left(X_{kt} - X_{k,t-\bar{l}_k} - \sum_{s=1}^{|Q_k|} E_{kst}\right), \quad \forall k\in Q,\ t\in\bar{H}_k, \tag{12}$$

$$W_{kt}\geq\bar{\beta}_k\left(\sum_{s=1}^{|Q_k|} E_{kst} - X_{kt} + X_{k,t-\bar{l}_k}\right), \quad \forall k\in Q,\ t\in\bar{H}_k, \tag{13}$$

$$X_{kt}\in\mathbb{Z}_+, \quad \forall k\in Q,\ t\in H, \tag{14}$$

$$E_{kst}=0,\ \forall k\in Q,\ s\in\{2,\ldots,|Q_k|\},\ t\in\bar{H}_k,\ t< S_{i'_k+s-1,0}. \tag{15}$$

Instance	bib2	bib3	bib5
n	3	5	6
$(n(i); p_i, d_i; \alpha_i, \beta_i)$	(4; 9,35; 10,10) (4; 13,35; 10,10) (12; 5,5; 1,1)	(4; 9,50; 10,10) (4; 9,50; 10,10) (4; 13,50; 10,10) (4; 13,50; 10,10) (6; 5,5; 1,1)	(4; 6,50; 2,2) (4; 6,50; 2,2) (5; 6,40; 2,2) (4; 9,50; 1,1) (4; 9,50; 1,1) (5; 9,40; 1,1)
h	150	208	200

Table 1: Data for small instances

The constraints (4)-(5) fix the position of the "zero" operations. The constraints (6) reflect the nature of the variables X. The constraints (7) state that all operations of jobs in G_k should be processed within the time horizon. The constraints (8) prevent overlapping. The constraints (9)-(11) relate the variables X and E. The constraints (12)-(13) are used to compute the values of the variables W. Here $\sum_{s=1}^{|Q_k|} E_{kst} = |G_{kt}|$, $\forall k \in Q$, $t \in \bar{H}_k$.

The formulation (3)-(15) has roughly $(2q + n)h$ variables, qh which are integer, $(3q + 2n + 1)h$ constraints and $(10q + 9n)h$ non-zero coefficients in the matrix. So, the size of this formulation depends first and foremost on the length of the time horizon h, then on the number of groups of identical jobs q, and finally on the number of jobs n.

B. Testing small instances

We have tested the formulations of three small instances whose data is given in Table 1. These test instances are of the same type as real life instances but of a reduced size. Results are presented in Table 2.

C. Testing random instances

The random instances were generated using the following procedure. The time horizon length h, the density d (the relation between the total processing time of operations and the horizon length), the number n of jobs and the total number of operations $o = \sum_{i=1}^{n} n(i)$ are fixed. Let $a = \lfloor \frac{o}{n} \rfloor$ and $b = \lfloor \frac{dh}{o} \rfloor$. Then, for each job $i \leq n$, $n(i)$ and p_i are uniformly generated over $[1, 2a]$ and $[1, 2b]$ accordingly in such a way that $\sum_{i=1}^{n} p_i n(i) = dh$. Next, for each job i,

Test	(TI)			(TIA)			Branch-and-price		
	Gap_{LB}	Gap_{UB}	Tm	Gap_{LB}	Gap_{UB}	Tm	Gap_{LB}	Gap_{UB}	Tm
bib2(LP)	74.9%		1.3s	45.0%		0.3s	42.3%		9.1s
bib2	59.0%	647%	>10m	0%	**0%**	17s	0%	**0%**	6m
bib3(LP)	56.3%		3.4s	37.6%		1.2s	19.7%		9.0s
bib3	47.5%	—	>10m	0%	**0%**	32s	0%	**0%**	37s
bib5(LP)	90.5%		0.3s	56.2%		1.4s	56.2%		1.4s
bib5	85.7%	25.7%	>30m	0%	**0%**	11m	0%	**0%**	23m

Table 2: Comparison of formulations

we set $l_i = \lfloor \frac{h}{n(i)} \rfloor$. At last, the "zero" operations are scheduled before time 0 one just after another in the reverse order:

$$S_{10} = -p_1, \qquad S_{i0} = S_{i-1,0} - p_i,\ i \in [2, \ldots, n].$$

The instances were generated for the next values of the parameters: $h \in \{250, 500\}$, $d \in \{0.5, 0.75, 1.0\}$, $o \in \{30, 60, 100, 150\}$, $n \in \{3, 6, 15, 30\}$. For each combination of values $\{h, d, o, n\}$ satisfying the conditions $3o < dh$ and $3n < o$, 2 instances were created. Therefore, in total, 234 instances were tested.

The formulation (TI) was not tested. Firstly, because of its poor results for the three small instances. Secondly, because of the quality of the lower bound given by the LP relaxation of the formulation (TI). For the generated random instances, it was always less than the lower bound of the LP relaxation of the formulation (TIA), and on the average the relation between these two bounds was only 25.7%.

In the following, we present results for the following statistics:

- P — percentage of instances solved within 30 minutes.
- Tm — average solution time (if, for some instance, the time limit of 1800 seconds was reached, this time limit is included into statistics).
- Gap — average difference between the best upper bound known and the best lower bound obtained, in percents from the best solution known (this statistics takes into account only instances for which an upper bound is known and the LP relaxation can be solved within the time limit).

Selection	P_{LP}	Tm_{LP}	Gap_{LP}	P	Tm	Gap	Nd
All instances	94.9%	222s	60.0%	26.1%	1458s	40.4%	2902
$h = 250$	100.0%	13s	57.2%	44.4%	1187s	26.4%	4660
$h = 500$	92.6%	315s	61.5%	17.9%	1578s	47.6%	1537
$d = 0.5$	100.0%	13s	47.6%	71.4%	772s	11.9%	1430
$d = 0.75$	100.0%	155s	59.3%	24.1%	1550s	37.9%	4090
$d = 1.0$	88.6%	363s	69.0%	13.3%	1597s	55.3%	4577
$n = 3$	100.0%	5s	68.2%	45.2%	1104s	37.4%	4291
$n = 6$	100.0%	36s	59.9%	22.6%	1568s	41.4%	1618
$n = 15$	100.0%	403s	50.4%	8.3%	1756s	45.5%	309
$n = 30$	33.3%	1614s	28.0%	0.0%	1800s	27.6%	1

Table 3: Results for random instances — the formulation (TIA)

- Nd — average number of nodes in the branch-and-bound or branch-and-price tree (only instances for which the LP relaxation can be solved within the time limit).

The results for the random instances are presented in Tables 3 and 4. The statistics for the whole set of instances show that the efficiency of both approaches is roughly the same. The strength of the lower bound given by the solution of the LP relaxation of the formulation (TIA) is almost the same in comparison to the lower bound obtained by the column generation procedure (in average, the relation is 94.2%). This advantage of the column generation procedure results in slightly smaller gaps on average for the branch-and-price algorithm. Unfortunately, these gaps remain quite large even after 30 minutes of the solution time. So, the random instances happened to be harder to solve than the real life instances.

It can be also seen that instances with longer time horizon or with bigger operations density d are harder to solve. The impact of the number of jobs is not so simple. The results show that instances with small number of jobs are simpler for the formulation (TIA) and harder for the branch-and-price algorithm. This can be better seen in Table 5 where detailed results for the statistics T_{LP}, Gap_{LP} and Gap are presented. From Table 5 it is clear that such a dependence of the approach efficiency from the number of jobs comes from the LP relaxations. The solution of the LP relaxation of the formulation (TIA) takes much more time for instances with a larger number of jobs. The contrary takes place when the column generation procedure is concerned.

Selection	P_{LP}	Tm_{LP}	Gap_{LP}	P	Tm	Gap	Nd
All instances	93.2%	212s	57.2%	26.1%	1385s	32.4%	802
$h = 250$	97.2%	72s	54.6%	37.5%	1202s	20.8%	1443
$h = 500$	91.4%	274s	58.4%	21.0%	1467s	37.9%	499
$d = 0.5$	100.0%	1s	46.5%	73.8%	512s	5.2%	1389
$d = 0.75$	100.0%	2s	57.0%	33.7%	1321s	23.3%	1322
$d = 1.0$	86.7%	422s	63.8%	12.4%	1611s	48.8%	713
$n = 3$	86.9%	294s	67.5%	33.3%	1272s	35.7%	965
$n = 6$	94.1%	231s	57.6%	30.6%	1313s	30.1%	849
$n = 15$	100.0%	96s	47.8%	14.6%	1554s	31.9%	677
$n = 30$	100.0%	50s	38.7%	0.0%	1800s	30.3%	268

Table 4: Results for random instances — the branch-and-price algorithm

The main conclusion of the computational tests on random instances is that the formulation (TIA) should be used when the number of jobs (or the number of groups of identical jobs) is small (6 or less). When the number of jobs is big and each job contains a moderate number of operations, the branch-and-price algorithm should be preferred.

	Branch-and-price								
	Tm_{LP}			Gap_{LP}			Gap		
$d =$	0.5	0.75	1.0	0.5	0.75	1.0	0.5	0.75	1.0
$n = 3$	.2s	3s	682s	56%	70%	77%	8%	29%	67%
$n = 6$	.3s	1s	539s	45%	58%	65%	5%	23%	51%
$n = 15$	.6s	2s	191s	21%	46%	56%	0%	21%	49%
$n = 30$	-	2s	74s	-	26%	45%	-	10%	40%
	Formulation (TIA)								
	Tm_{LP}			Gap_{LP}			Gap		
$d =$	0.5	0.75	1.0	0.5	0.75	1.0	0.5	0.75	1.0
$n = 3$	1s	3s	9s	58%	72%	74%	8%	37%	56%
$n = 6$	8s	23s	61s	46%	61%	69%	17%	40%	57%
$n = 15$	63s	256s	600s	22%	48%	59%	8%	41%	58%
$n = 30$	-	1243s	1800s	-	28%	-	-	28%	-

Table 5: Detailed results for random instances