RAIRO-Oper. Res. 57 (2023) 325–332
https://doi.org/10.1051/ro/2023014

RAIRO Operations Research
www.rairo-ro.org

NEW CHARACTERIZATIONS FOR THE IDENTICAL COUPLED TASKS SCHEDULING PROBLEM

Hatem Hadda* and Mhamed Abdessalem

Abstract. In this note we introduce new characterizations for the optimal solution of the identical coupled tasks scheduling problem. We also develop three lower bounds and formulate useful observations about the choice of the test instances.

Mathematics Subject Classification. 90B35, 90B30 .

Received July 25, 2022. Accepted February 3, 2023.

1. Introduction

First introduced by Orman and Potts [7], the coupled tasks problem is a practical problem that tackles data treatment in radars and torpidos. The authors established the complexity status of several sub-cases except for the Identical Coupled Tasks (ICT) configuration which remained open. Authors also conjectured that this configuration is polynomial and that its makespan may be automatically computed within the solving algorithm. Later, Brauner *et al.* [4] established an interesting link between the ICT problem and a particular class of the one-machine robotic cell scheduling problem. Further interesting generalizations of the ICT problem may be found in [3, 5].

The ICT problem consists in scheduling n identical jobs on a single machine. Each job is composed of two operations: an a-task (of length a) and a b-task (of length b). The execution of the two tasks must be separated by a mandatory (and fixed) delay time L. The machine cannot execute more than one task at a time. For a given schedule, the makespan (*i.e.* the maximum completion time) is given by the completion time of the last b-task. The objective is to find a schedule minimizing the makespan. We will use the following notation:

a	Duration of the a-task
b	Duration of the b-task
L	Mandatory delay time separating the completion of an a-task and the starting of its associated b-task
(a, b, L)	An instance of the ICT problem
$C_{\max}(\sigma)$	The makespan of a given schedule σ
$C^{\star}_{\max}(a, b, L, n)$	The optimal makespan for a set of n jobs of an (a, b, L) instance. We will drop the reference to (a, b, L) and n whenever no confusion can arise

Keywords. Identical coupled tasks, lower bound, scheduling.

Université de Tunis El Manar, Ecole Nationale d'Ingénieurs de Tunis, Oasis, BP. 37, Le belvédère 1002, Tunis, Tunisia.
*Corresponding author: hatem.hadda@esti.rnu.tn; hatem.hadda@gmail.com

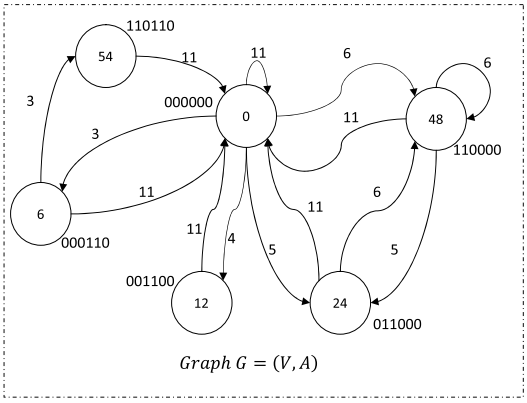

FIGURE 1. Pattern graph for $a = 3$, $b = 2$ and $L = 6$.

Without loss of generality we suppose that $a \geq b$. We will only consider the configuration such that $n > L/a$, as the other case is obvious.

Ahr *et al.* [1] introduced an interesting approach in which a schedule can be seen as a sequence of $P(a, b, L)$ patterns. A $P(a, b, L)$ pattern is a sequence of L 0-1 indicating whether the machine is idle (0) or busy (1) during the gap of the last job. Jobs are scheduled one after the other, and each time a new job is scheduled a $P(a, b, L)$ pattern is associated to it. Based on this representation, a directed graph $G = (V, A)$ is associated to the problem, where the set of vertices V is composed by all possible patterns, and the set A is composed by weighted arcs representing the possible transitions between the patterns and the induced increases of the makespan. We stress out that the weights of the arcs lie between a and $(a+b+L)$ (the latter value corresponds to an arc joining any given pattern to pattern 0^L).

As an illustration let us consider the example used in [1] with $a = 3$, $b = 2$ and $L = 6$. Six patterns are possible (labeled by their decimal representation):

- 0 : 000000
- 6 : 000110
- 12 : 001100
- 24 : 011000
- 48 : 110000
- 54 : 110110

Figure 1 displays the pattern graph associated to these vertices with all possible transitions (arcs). For instances, the arc $(0, 6)$ represents the transition from a schedule in which the last job is not interleaved with none of its predecessors to a schedule in which a new job is added such that its a-task starts directly after the last a-task. This leads to an increase of the makespan value by $a = 3$.

Using the pattern graph, the problem reduces to finding a minimal weight path μ with exactly $(n-1)$ arcs starting from pattern 0^L. The corresponding makespan is the total weight of the path $w(\mu)$ plus $(a+b+L)$ (*i.e.* $C^{\star}_{\max} = a+b+L+w(\mu)$). Figure 2 illustrates an optimal schedule with $n = 4$ jobs for the $(a = 3, b = 2, L = 6)$ instance. The reader is referred to [1] for a more complete presentation.

Based on this approach, Ahr *et al.* [1] proved that the problem can be solved in $O(n)$ for fixed a, b and L. Later Baptiste [2] improved the complexity to $O(\log n)$, however the constant remains highly exponential in L.

In a related paper, Lehoux-Lebacque *et al.* [6] present a remarkable work in which they prove that the ICT problem is polynomial under the cyclic version. They present the formulas that allow computing the minimum weight mean cycles. Given a directed cycle θ in G, its mean weight is defined by $r = \frac{w(\theta)}{|\theta|}$ where $w(\theta)$ and $|\theta|$ designate the weight and the length (number of arcs) of θ. A minimum weight mean cycle $\sigma_{a,b,L}$ is a directed

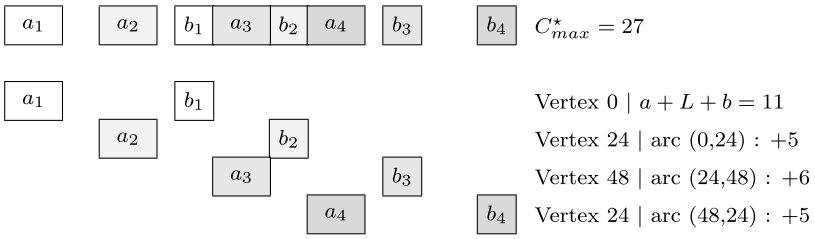

FIGURE 2. Optimal schedule with $n = 4$ tasks for $a = 3$, $b = 2$ and $L = 6$.

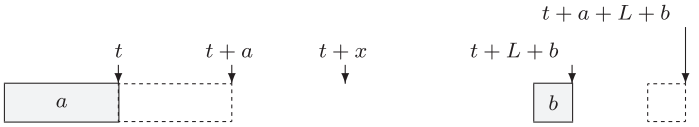

FIGURE 3. An a-task succeeded by an idle period.

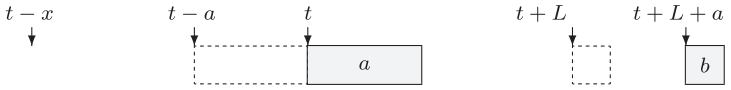

FIGURE 4. An a-task preceded by an idle period.

cycle such that $r_{a,b,L} = \frac{w(\sigma_{a,b,L})}{|\sigma_{a,b,L}|} = \min\{\frac{w(\theta)}{|\theta|} \mid \theta \text{ cycle in } G\}$. Given a minimum weight mean cycle $\sigma_{a,b,L}$, we note by $l_{a,b,L}$ its length and by $w_{a,b,L}$ its weight.

In the rest of this note we introduce new characterizations for the optimal solution and develop three lower bounds.

2. Characterizations of the optimal solution

In this section we introduce new characterizations for the optimal solution of the ICT problem.

Lemma 1. *There exists an optimal schedule in which no a-task is neither preceded nor succeeded (except for the last one) by an idle period larger than $a - 1$.*

Proof. Let us consider an optimal schedule in which an a-task (not the last one) is succeeded by an idle period $x \geq a$ (see Fig. 3). Let t be the finishing time of the a-task. As no a-task appear in the interval $[t, t + x]$, then the interval $[t + L + b, t + L + b + x]$, and particularly the sub-interval $[t + L + b, t + L + b + a]$, is either idle or contains an a-task. In the first case, the last job can be rescheduled at date t and the schedule cannot be optimal. In the second case, let t' be the starting time of the first a-task in the interval $[t + L + b, t + L + b + a]$. Its finishing time is such that $t' + a \geq t + a + L + b$ which means that the interval $[t + L + b, t + L + b + a]$ can only contain a single a-task. In this case it is possible to schedule that job such that its a-task starts at date t without altering the starting times of the other jobs. By applying the same argument for similar situations we obtain an optimal solution in which no a-task (except for the last one) is succeeded by an idle period larger than $a - 1$.

Now let us consider an optimal schedule in which an a-task is preceded by an idle period $x \geq a$, and let t be its starting time (see Fig. 4). As the interval $[t - x, t]$, and particularly the sub-interval $[t - a, t]$ is empty, then the interval $[t + L, t + L + a]$ is either idle or contains only a single a-tasks. In the first case the last job can be rescheduled such that its a-task starts at $t - a$ and the schedule is not optimal. In the second case the single

FIGURE 5. Two a-tasks separated by an idle period.

a-task of the interval $[t+L, t+L+a]$ can be rescheduled such that it starts at date $t-a$ without disrupting the rest of the schedule. By doing the same for all similar situations, we construct an optimal schedule such that no a-task is preceded by an idle period larger than $a-1$. □

Lemma 2. *There exists an optimal schedule in which no consecutive a-tasks are separated by an idle period δ such that $0 < \delta < b$.*

Proof. Let us consider an optimal schedule in which two consecutive a-tasks are separated by an idle period δ such that $0 < \delta < b$. Their corresponding b-tasks are separated by a period $\delta + a - b$ (see Fig. 5). Note that no other b-task can be scheduled between the two b-tasks for otherwise its corresponding a-task would be scheduled between the two a-tasks which are by definition separated by an idle period.

The period separating the two b-tasks is such that $\delta + a - b < a$. This means that no a-task is scheduled between the two b-tasks and that they are separated by an idle period of $\delta + a - b$. Consequently the second job may be rescheduled to start directly after the first (with no idle time between the a-tasks). □

Lemmas 1 and 2 imply the following corollary.

Corollary 1. *There exists an optimal schedule in which consecutive a-tasks are either scheduled directly after each other (with no idle time), or separated by an idle time δ such that $b \leq \delta \leq a-1$.*

In the conclusion of their paper, Ahr *et al.* [1] formulated the following remark:

> "Another question is whether the size of the pattern graph can be decreased by eliminating transitions which can only lead to suboptimal schedule."

Lemmas 1 and 2 contribute directly to the eliminations of a large number of futile transitions. Let us consider again the example $(a = 3, b = 2, L = 6)$. Figure 6 displays the pattern graph $G = (V, A)$ as described [1] with all possible vertices and transitions. Now, by virtue of Lemma 1, the arc $(0, 0)$ should not be considered. Indeed this transition will leave after the a-task of the first job an idle period of length 6 which exceeds $a - 1 = 2$. The same applies for the arc $(0, 48)$ where an idle period of length 3 will appear between the two a-tasks. The arc $(0, 12)$ should also be eliminated by virtue of Lemma 2 as this transition leads to an idle period of length 1 between two a-tasks which is such that $0 < 1 < b = 2$. Consequently node 12 will not be created and we get the reduced graph $G' = (V', A')$.

To assess the curtailing effect of Lemmas 1 and 2, we present in Table 1 the reduction ratios of the number of vertexes ($\text{red}_{\text{vertex}}$) and arcs ($\text{red}_{\text{arc}}$) obtained after the application of the two lemmas. For each instance (a, b, L) we generate two graphs G and G', where $G = (V, A)$ is the initial graph as described in [1], and $G' = (V', A')$ is the graph generated while considering Lemmas 1 and 2. The reduction ratios are computed as follows $\text{red}_{\text{vertex}} = \frac{(|V|-|V'|)\times 100}{|V|}$ and $\text{red}_{\text{arc}} = \frac{(|A|-|A'|)\times 100}{|A|}$. The obtained results show an important gain on the size of the pattern graph. Indeed, for some of the instances, up to 67% of the vertices, and up to 80% of the arcs have been eliminated.

Another observation about the asymptotic behavior of the problem is captured by the Lemma 3.

Lemma 3. *There exists a threshold m such that for any $n' = n + \alpha l_{a,b,L}$ where $n \geq m$ and α a positive integer, we have $C^{\star}_{\max}(n') = C^{\star}_{\max}(n) + \alpha w_{a,b,L}$.*

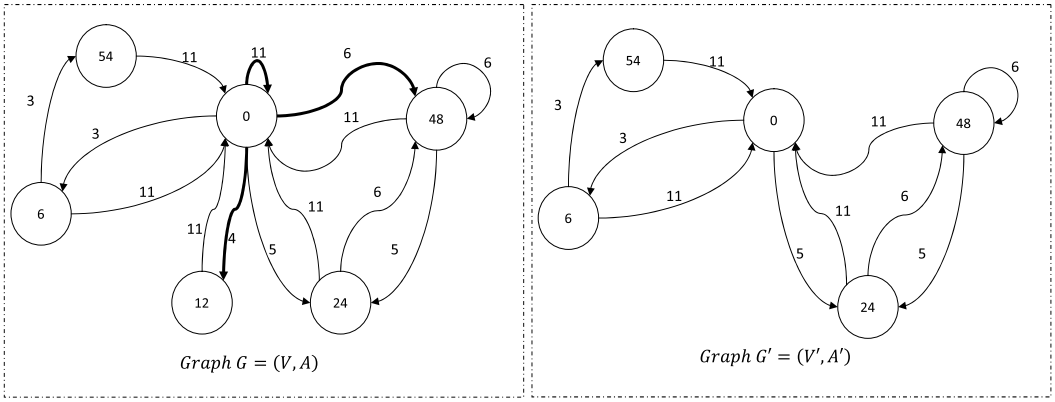

FIGURE 6. Example of pattern graph reduction for $a = 3$, $b = 2$ and $L = 6$.

TABLE 1. Size reduction of the pattern graph.

a	b	L	$\text{red}_{\text{vertex}}(\%)$	$\text{red}_{\text{arc}}(\%)$
3	2	10	39.3	58.3
3	2	20	59.0	75.4
3	2	25	67.0	80.9
5	2	10	12.5	20.0
5	2	20	38.6	52.3
5	2	25	45.6	59.3
5	3	10	25.0	33.3
5	3	20	58.6	65.0
5	3	25	66.7	72.4
7	2	10	20.0	25.0
7	2	20	27.9	38.2
7	2	25	33.1	42.2
7	4	10	60.0	77.8
7	4	20	55.8	50.6
7	4	25	63.9	64.8
9	3	10	33.3	60.0
9	3	20	36.8	24.4
9	3	25	44.4	44.7
9	5	10	33.3	60.0
9	5	20	47.4	35.4
9	5	25	61.1	48.8

Proof. It can be seen that there exists a threshold m such that for any $n \geq m$ the optimal solution includes an optimal cycle at least once. Considering $n' = n + \alpha l_{a,b,L}$ with $n \geq m$, it should be clear that the additional $\alpha l_{a,b,L}$ jobs will from α cycles which leads to $C^{\star}_{\max}(n') = C^{\star}_{\max}(n) + \alpha w_{a,b,L}$. □

Lemma 3 implies that starting from m, all optimal makespans can be derived by a simple translation of the first $l_{a,b,L}$ makespans following m. Consequently the problem reduces to the determination the optimal makespans for the sizes $n \leq m + l_{a,b,L}$.

Finally, we point out the following observation regarding the choice of the test instances. Let (a, b, L) and (a', b', L') be two different instances with the same number of jobs such that $a' = \alpha\, a$, $b' = \alpha\, b$ and $L' = \alpha\, L$,

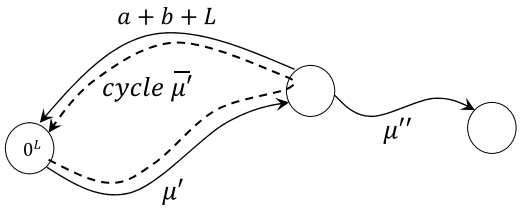

FIGURE 7. Illustration for lb_1.

where $\alpha > 0$ is a positive coefficient. It should be clear that their optimal makespans are such that $C^{\star'}_{\max} = \alpha\, C^{\star}_{\max}$. Furthermore $w_{a',b',L'} = \alpha\, w_{a,b,L}$ and $l_{a',b',L'} = l_{a,b,L}$. In other words, the two problems are equivalent in the sense that we can use an optimal solution of one of them to construct an optimal schedule for the other and *vice versa*.

Given the previous observation, we can affirm that several instances considered in the numerical experimentation in [1] are redundant. Namely, instances $(4, 2, 10)$, $(8, 4, 20)$ and $(10, 5, 25)$ can be replaced by $(2, 1, 5)$. Instance $(8, 4, 30)$ is equivalent to $(4, 2, 15)$, and instances $(4, 2, 20)$ and $(6, 3, 30)$ can both be replaced by $(2, 1, 10)$. We also add that the pattern graph approach is blind to this aspect. Indeed it would treat the problems $(20, 10, 50)$ and $(2, 1, 5)$ as independent with a much bigger graph for $(20, 10, 50)$ which would require much more computational effort. Consequently we think that this observation may help reducing the computational time required by the pattern graph based approach.

3. Lower bounds

We now introduce a lower bound for the optimal makespan.

Lemma 4. *For a given problem (a, b, L), a lower bound for $C^{\star}_{\max}(n)$ is given by*

$$lb = \max\{lb_1; lb_2; lb_3\},$$

where $lb_1 = n' w_{a,b,L} + n'' a$ with $n' = \lfloor \frac{n-1}{l_{a,b,L}} \rfloor$ and $n'' = n - n' l_{a,b,L}$; $lb_2 = n(a + b)$ and $lb_3 = na + L + b$.

Proof. The last two bounds are obvious. For the first bound, let us consider an optimal makespan, and let μ be its associated path in the pattern graph. We have $C^{\star}_{\max} = a + b + L + w(\mu)$. Recall that μ is composed by $n - 1$ arcs starting from pattern 0^L.

If $n' = 0$ (*i.e.* $n'' = n$), then it is easy to see that $C^{\star}_{\max} \geq na$. Otherwise, let μ' be the path composed by the $(n' l_{a,b,L} - 1)$ first arcs of μ, and let μ'' be the path composed by the remaining n'' arcs of μ. Note that $w(\mu) = w(\mu') + w(\mu'')$. Let $\overline{\mu'}$ be the cycle obtained from μ' by adding an arc joining the last pattern in μ' to 0^L (see Fig. 7). We have $w(\overline{\mu'}) = w(\mu') + a + b + L$. By definition we have $\frac{w_{a,b,L}}{l_{a,b,L}} \leq \frac{w(\overline{\mu'})}{n' l_{a,b,L}}$. We then conclude that $C^{\star}_{\max} = w(\overline{\mu'}) + w(\mu'') \geq n' w_{a,b,L} + n'' a$. □

In order to assess the quality of the lower bounds we evaluate in Table 2 the relative gaps Gap_k, $k \in \{1, 2, 3\}$, between the three lower bounds and the optimal makespan. The gaps are computed as follows $Gap_k = \frac{(C^{\star}_{\max} - lb_k) \times 100}{C^{\star}_{\max}}$ for $k \in \{1, 2, 3\}$. The best values are highlighted in bold. The results show the good performance of lb_1 for big sizes. This results are predictable as the optimal schedules are expected to contain optimal cycles for big size instances. We also remark that the performance of lb_1 is directly related to $l_{a,b,L}$. On the other hand lb_2 outperforms the two others for medium and small sizes. Finally, and as expected, we note the low performance of lb_3.

TABLE 2. Lower bounds quality.

(a, b, L)	$(l_{a,b,L}, w_{a,b,L}, r_{a,b,L})$	n	$lb_1(\%)$	$lb_2(\%)$	$lb_3(\%)$
$(3, 2, 10)$	$(15, 75, 5)$	25	22.2	**7.4**	35.6
		50	7.7	**3.8**	37.7
		100	5.9	**2.0**	38.8
		250	2.4	**0.8**	39.5
		500	0.8	**0.4**	39.8
		1000	0.6	**0.2**	39.9
		2000	0.2	**0.1**	39.9
$(3, 2, 20)$	$(45, 225, 5)$	25	48.3	**13.8**	33.1
		50	11.1	**7.4**	36.3
		100	7.7	**3.8**	38.1
		250	5.5	**1.6**	39.2
		500	1.2	**0.8**	39.6
		1000	0.8	**0.4**	39.8
		2000	0.6	**0.2**	39.9
$(5, 2, 10)$	$(6, 51, 8.5)$	25	**5.4**	20.8	38
		50	**3.2**	19	39.4
		100	**2.5**	18.3	40.3
		250	**1.0**	17.9	40.8
		500	**0.3**	17.8	41
		1000	**0.2**	17.7	41.1
		2000	**0.1**	17.7	41.1
$(5, 2, 20)$	$(18, 140, 7.8)$	25	**17.8**	**17.8**	31
		50	**13.6**	**13.6**	32.8
		100	**5.4**	11.7	34.2
		250	**3.1**	10.7	35.1
		500	**1.4**	10.4	35.4
		1000	**0.6**	10.2	35.6
		2000	**0.1**	10.1	35.6

4. CONCLUSION

In this note we developed new characterizations for the optimal solution of the identical coupled tasks problem. Although we did not succeed to solve the problem, we think that the presented results may help identifying dominant structures. We share the conjecture that the problem can be solved in polynomial time, however we think that the proof will require a technical construction as the one presented for the cyclic version [6].

REFERENCES

[1] D. Ahr, J. Bekesi, G. Galambos, M. Oswald and G. Reinelt, An exact algorithm for scheduling identical coupled tasks. *Math. Methods Oper. Res.* **59** (2004) 193–203.

[2] P. Baptiste, A note on scheduling coupled tasks in logarithmic time. *Discrete Appl. Math.* **158** (2010) 583–587.

[3] J. Blazewicz, G. Pawlak, M. Tanas and W. Wojciechowicz, New algorithms for coupled tasks scheduling – a survey. *RAIRO: Oper. Res.* **46** (2012) 335–353.

[4] N. Brauner, G. Finke, V. Lehoux-Lebacque, C. Potts and J. Whitehead, Scheduling of coupled tasks and one-machine no-wait robotic cells. *Comput. Oper. Res.* **36** (2009) 301–307.

[5] B. Darties, R. Giroudeau, J.C. Konig and G. Simonin, Some complexity and approximation results for coupled-tasks scheduling problem according to topology. *RAIRO: Oper. Res.* **50** (2016) 781–795.

[6] V. Lehoux-Lebacque, N. Brauner and G. Finke, Identical coupled task scheduling: polynomial complexity of the cyclic case. *J. Scheduling* **18** (2015) 631–644.

[7] A.J. Orman and C.N. Potts, On the complexity of coupled-task scheduling. *Discrete Appl. Math.* **72** (1997) 141–154.