23rd International Congress on Modelling and Simulation, Canberra, ACT, Australia, 1 to 6 December 2019
mssanz.org.au/modsim2019

A simple heuristic for the coupled task scheduling problem

M. Khatami[a] **and A. Salehipour** [a]

[a]*School of Mathematical and Physical Sciences, University of Technology Sydney, Australia.*

Email: Mostafa.Khatami@student.uts.edu.au

Abstract: The coupled task scheduling problem deals with the problem of scheduling a set of jobs to be processed on a single machine. Each job consists of two separated tasks where the second task of a job must be started after the completion of its first task plus a predefined exact delay ti me. We study the coupled task scheduling problem with the objective of minimising the maximum completion time, i.e., the makespan. The problem is known to be strongly NP-hard. We propose a simple heuristic for the problem. The heuristic sets the midpoint of given lower and upper bounds as a makespan and solves a feasibility problem. Upon solving the feasibility problem the upper bound is updated to the midpoint. Otherwise, the lower bound is updated to the midpoint. The algorithm keeps iterating until the bound interval is zero. Computational experiments indicate that the proposed heuristic outperforms the exact solver.

Keywords: *Heuristic search, feasibility, makespan, scheduling*

1 INTRODUCTION

In this paper we consider the problem of minimising the makespan for scheduling a set J of "coupled-task" jobs on a single machine. Each job $j \in J$ consists of two separated tasks where there is an "exact" time interval, denoted as delay, between them. The second (completion) task must be processed exactly after the completion of the first (initial) task plus the duration of the delay. A job j is shown by a triple (a_j, L_j, b_j) where parameters a_j, L_j and b_j denote the processing time of the initial task, the duration of the delay and the processing time of the completion task. Using the three-field notation of Graham et al. (1979), minimising the makespan for the coupled task scheduling problem is commonly denoted as $1|(a_j, L_j, b_j)|C_{max}$.

The coupled task scheduling problem was first proposed by Shapiro (1980) to model a pulsed radar system, in which the transmission of the pulse and the reception of its reflection after a period of time help to measure the size and/or the shape of the tracking object. The objective is to detect as many objects as possible. As pointed out by Ageev and Baburin (2007), the coupled task problem can also be applied to certain scheduling problems in chemistry manufacturing in which there is an exact technological delay between the consecutive tasks of a job.

Sherali and Smith (2005) proved that minimising the makespan for the coupled task scheduling problem is strongly NP-hard. It is even strongly NP-hard if the sequence for the initial (or completion) tasks is given (Condotta and Shakhlevich 2012). Several special cases were also shown to be strongly NP-hard in Orman and Potts (1997). For instance, minimising the makespan for problems $(a_j = L_j = b_j)$, $(a_j, L_j = L, b_j = b)$ and $(a_j = p, L_j, b_j = p)$, where L, b, and p are positive integers.

There are a few solution methods for the general case of the coupled task scheduling problem. Some notable works include the branch-and-bound of Békési et al. (2014) and tabu search of Li and Zhao (2007) and Condotta and Shakhlevich (2012), though no detailed comparison was reported. These studies utilised instances with different characteristics and did not make them publicly available, making therefore difficult to conduct a comparison study. We refer the interested reader to Khatami et al. (2019) for a comprehensive review of the coupled task studies, applications and models.

In this paper, we propose a heuristic for the coupled task scheduling problem with the objective of minimising the makespan. The remainder of this paper is organised as follows. In Section 2, we discuss the proposed heuristic for the problem. The results of numerical experiments are presented in Section 3. The paper ends with a few conclusions in Section 4.

2 SOLUTION METHOD

We propose a solution method for the following problem. A set $J = \{1, \ldots, n\}$ of coupled task jobs are to be processed on a single machine, where there is an exact delay period between two consecutive tasks of each job. The objective is to minimise the length of the schedule, i.e., the makespan represented as C_{max}. We let $H = \{1, \ldots, 2n\}$ be a set of tasks, where H_{2j-1} and H_{2j} represent the initial and completion tasks of job j. All processing times and delay durations are integral.

We now present a heuristic to solve the coupled task scheduling problem. The general idea is as follows. First, a lower bound (LB) and an upper bound (UB) are derived on the value of makespan and then the midpoint between LB and UB is selected as the "current bound". Next, the heuristic investigates whether it is possible (feasible) to schedule all jobs when makespan is bounded from above by the current bound. For this we utilise an exact solver to solve a feasibility problem. A feasible solution implies that the current bound is a valid makespan. Therefore, UB is updated to the current bound. A no feasible solution status indicates that the current bound is a new (tighter) LB for the makespan, and hence the LB is updated to the current bound. The algorithm iterates until $UB - LB = 1$. In that case the UB is the optimal solution. It should be noted that since the problem is strongly NP-hard it is less likely that we can obtain the optimal solutions for large-sized instances in a reasonable amount of time. Therefore, we stop the algorithm with a time limit and the current UB is regarded as the heuristic solution. Next, we detail the main components of the heuristic including LB, UB and the feasibility model.

2.1 Lower bound

The trivial LB for the coupled task scheduling problem is $LB_0 = \sum_{j \in J} (a_j + b_j)$, that is to schedule all tasks without any idle time between them. However, this LB can be improved if there are some jobs with delays smaller than the minimum task processing times, i.e., if $\exists j' | L'_j < \min_{j \in J}\{a_j, b_j\}$. For such jobs, known as

"singleton" jobs in the literature (Li and Zhao 2007), the delay period cannot be utilised to schedule any other task. Hence, the improved LB is $LB_1 = \sum_{j \in J} (a_j + b_j) + \sum_{j' \in J} L_{j'}, \forall L_{j'} < \min_{j \in J}\{a_j, b_j\}$. We note that $LB_1 \geq LB_0$ for any instance of the problem. Hence, we use LB_1 as LB in the heuristic.

2.2 Upper bound

Shapiro (1980) proposed two heuristics of "interleaving" and "nesting" for the coupled task scheduling problem. The interleaving heuristic sequences the jobs such that the completion tasks arrive for processing in the same order as the initial tasks (Figure 1a), while in the nesting heuristic, the completion tasks arrive in the reverse order of the initial tasks (Figure 1b).

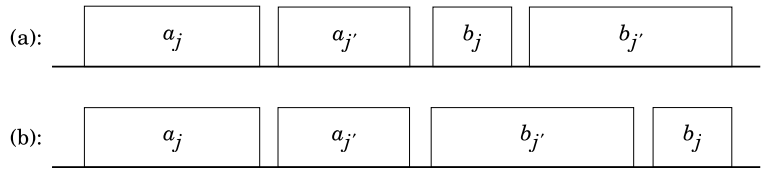

Figure 1. (a): Interleaving jobs j and j', and (b): nesting jobs j and j'.

The trivial UB of the problem is $\sum_{j \in J} (a_j + L_j + b_j)$, that is to schedule jobs one by one without any interleaving or nesting of jobs, called as "appending" in the literature. Since this bound is very loose, we aim at improving it via a prompt local search approach. Our local search algorithm (Algorithm 1) starts with a given sequence $\pi_0 = (1, \ldots, n)$ of jobs, and then iteratively improves it with adjacent pairwise interchanges. The quality of a sequence is evaluated based on the "feasible schedule" generated from a given sequence. For any sequence π' the algorithm generates a feasible schedule with nesting, interleaving and appending operations as following. For a pair of consecutive jobs j and j' in the current sequence π', if nesting of job j' inside job j is possible, i.e., if $L_j \leq a_{j'} + L_{j'} + b_{j'}$, then it is performed. However, if nesting is not possible but interleaving is, i.e., if $L_j \geq a_{j'} \wedge L_{j'} \geq b_j$, then the interleaving is performed where job j is the first job in the pair. When neither nesting nor interleaving of the two jobs is possible, then job j' is adjacently appended after job j. The algorithm performs the same operations for consecutive pairs of jobs till all jobs are scheduled. Additionally, the first improvement criterion is implemented, i.e., once an improving solution is obtained it is accepted and the sequence is updated.

2.3 Feasibility model

A number of mathematical programs are available in the literature for the coupled task scheduling problem. We utilise the model proposed by Békési et al. (2014) that according to Khatami et al. (2019) is among the top performing models. Linear ordering variables are used in this formulation and the sequence is generated by ordering the tasks rather than jobs. For any pair of tasks $h, h' \in H$, a binary variable $x_{hh'}$ is defined that takes a value of 1 if task h' is started after task h in the sequence, and 0 otherwise. The formulation is presented as Problem P below.

Problem P

$$z = \min C_{max} \tag{1}$$

subject to

$$C_{max} \geq s_{2j} + b_j, \quad 1 \leq j \leq n, \tag{2}$$

$$x_{2j-1,2j} = 1, \quad 1 \leq j \leq n, \tag{3}$$

Algorithm 1: The local search algorithm.

Input: $\pi_0 = (1, 2, \ldots, n)$, C_{π_0}, $j = 1$.
Output: A sequence π with makespan C_S.

$\pi = \pi_0$;
$C_\pi = C_{\pi_0}$;
while $j \leq n - 1$ **do**
    $Improve = 0$;
    for $k = j + 1 : n$ **do**
        $\pi' \leftarrow$ swap(j, k);
        Generate feasible schedule();
        $C_{\pi'} \leftarrow$ makespan(π');
        if $C_{\pi'} < C_\pi$ **then**
            $\pi = \pi'$;
            $C_\pi = C_{\pi'}$;
            $Improve = 1$;
        end
    end
    if $Improve = 0$ **then**
        $j = j + 1$;
    end
end
return S;

$$x_{hh'} + x_{h'h} = 1, \quad 1 \leq h < h' \leq 2n, \tag{4}$$

$$x_{hh'} + x_{h'h''} + x_{h''h} \leq 2, \quad \text{for any triple distinct tasks:} \quad (h, h', h'') \in H, \tag{5}$$

$$s_{2j} = s_{2j-1} + a_j + L_j, \quad 1 \leq j \leq n, \tag{6}$$

$$s_{2j} \leq UB - b_j, \quad 1 \leq j \leq n, \tag{7}$$

$$s_{h'} \geq s_h + p_h - UB(1 - x_{hh'}), \quad 1 \leq h, h' \leq 2n, \quad h \neq h', \tag{8}$$

$$s_h \geq 0, \quad 1 \leq h \leq 2n, \tag{9}$$

$$x_{hh'} \in \{0, 1\}, \quad 1 \leq h, h' \leq 2n, \quad h \neq h', \tag{10}$$

where s_h denotes the start time of the task h. Constraints (3) defines that the completion task of a job should be scheduled after its initial task. Constraints (4) shows the relative order of any pair of tasks. Constraints (5) represent the so-called "3-dicycle inequalities" for any triple distinct tasks. The relation between the starting times of the tasks of a job is defined in constraints (6), and an upper bound on the starting times of the completion tasks is set in constraints (7). Denoting the processing time of task h as p_h, constraints (8) relate the starting time variables to the linear ordering variables. Precisely, the constraints ensure that the start time of task h' must be at least as large as the finishing time of task h if task h' is scheduled after task h.

To generate the feasibility model of a problem, we can change the objective function to a constant value. In this way, the solver seeks for a feasible solution during the optimisation process. Finding a single feasible solution implies that the problem is feasible. On the other hand, if it is proved that the model has no solution, it indicates that the original problem is infeasible as well.

3 COMPUTATIONAL EXPERIMENTS

To investigate the performance of the proposed heuristic, we use the benchmark instances proposed in Khatami et al. (2019) for the general case of the coupled task scheduling problem, denoted as the "general set". For

the number of jobs we use the instances with $n \in \{10, 20, 25, 40, 50\}$, and for the choice of processing time of tasks and the duration of delays, three types of small, medium and large jobs are considered as small jobs: $a_j, b_j \sim U(1, 20), L_j \sim U(10, 80)$; medium jobs: $a_j, b_j \sim U(1, 50), L_j \sim U(25, 200)$; and, large jobs: $a_j, b_j \sim U(1, 100), L_j \sim U(50, 400)$.

For each combination of n and the size of jobs there are ten instances in the benchmark. Consequently, we perform the computational experiments on a set of $5 \times 3 \times 10 = 150$ instances. The same instances are also solved by the solver Gurobi version 8.0.0 (Gurobi Optimization 2018) and utilising the model presented as Problem P earlier. As discussed, the model presented in P is shown to outperform all available mixed integer programs (MIPs) for the coupled task scheduling problem (Khatami et al. 2019). The MIP model and algorithms are implemented in the programming language Python version 3.6 and all computational experiments are performed on a PC with Intel® Core™ i5-7500 CPU clocked at 3.40GHz with 8GB of memory under Linux Ubuntu 18.04 operating system. A time limit of 3600 seconds is set for both the solver Gurobi and the heuristic. We utilise all four processors during the computational experiments.

We report the outcomes of the heuristic and MIP in Tables 1 and 2. Four criteria of "Feas", "Best", "Opt" and "Gap (in %)" are used to evaluate the two comparing methods that are as following. The metrics Feas and Opt denote the number of feasible and optimal solutions, respectively, obtained by the heuristic and MIP. The metric Best represents the number of best solutions obtained by each method and the metric Gap is calculated as $\frac{z-z^*}{z^*} \times 100$, where z is the objective function value obtained by the method and z^* is the best objective function value between the two comparing methods. We note that the metric Gap is calculated over the number of feasible solutions for the method.

According to Table 1, the heuristic delivers feasible solution for all 150 instances (the highlighted numbers denote the outperforming values), while MIP is unable to find feasible solution for 13 instance from $n = 40, 50$. It also reports significantly superior solutions indicated by the smaller value of gap of the heuristic compared to that of MIP. According to Table 2 both methods deliver optimal solution for the instances with 10 jobs. With regard to the metric Best, the heuristic approach obtains more best solutions than the MIP, and that in the instances with larger number of jobs. Precisely, the heuristic approach outperforms the MIP for instances with $n = 40, 50$. Similar outcomes can be seen with regard to the metric Gap as the solutions obtained by the heuristic are far better than those obtained by MIP, specifically for the instances with $n = 40, 50$.

In summary, both methods perform closely for the instances with fewer number of jobs, i.e., $n = 10, 20, 25$. However, as the number of jobs becomes larger, MIP is either unable to deliver a feasible solution, or it delivers one, and it is of poor quality. The overall outcomes indicate the effectiveness of the proposed heuristic in comparison to the best available MIP for the coupled task scheduling problem.

As stated in Section 2.2, the local search algorithm (Algorithm 1) was presented to serve as a tight upper bound for the proposed heuristic algorithm. To evaluate the effectiveness of the local search algorithm, we compare it to the trivial upper bound $\sum_{j \in J} (a_j + L_j + b_j)$. For the two upper bounds, we calculate their gap (in %) to the lower bound as $\frac{UB-LB}{UB} \times 100$ where UB is the value of the upper bound and LB is the value of the lower bound obtained by LB_1. In Table 3, we denote the trivial upper bound and the local search as UB_0 and UB_1 respectively. The results show that the bounds obtained by the local search method have an average gap of 45.3% to the lower bound, that is 23.4% tighter than that of the trivial upper bound.

4 CONCLUSION

There are not many solution approaches available in the literature for the strongly NP-hard coupled task scheduling problem. Due to the increasing interest in the coupled task problem in recent years, as well as identifying interesting applications for the problem, there is a need for proposing algorithms to solve the problem effectively and efficiently. In this study, we proposed a heuristic algorithm to solve the coupled task problem with the objective of minimising the makespan. The results of our computational experiments demonstrated that the proposed method reports high quality solutions, particularly, for the instances with large number of jobs.

ACKNOWLEDGMENT

Mostafa Khatami is the recipient of UTS International Research Scholarship (IRS) and UTS President's Scholarship (UTSP). Amir Salehipour is the recipient of an Australian Research Council Discovery Early Career Researcher Award (project number DE170100234) funded by the Australian Government.

Table 1. Brief comparison of the heuristic and MIP.

Metric	Heuristic	MIP
Feas	**150**	137
Opt	**30**	**30**
Best	**98**	83
Gap (in %)	**1.17**	9.92

Table 2. Detailed comparison of the heuristic and MIP.

		Feas		Opt		Best		Gap (in %)	
n	Job size	Heuristic	MIP	Heuristic	MIP	Heuristic	MIP	Heuristic	MIP
10	Small	10	10	10	10	10	10	0.00	0.00
	Medium	10	10	10	10	10	10	0.00	0.00
	Large	10	10	10	10	10	10	0.00	0.00
	Total/Average	30	30	30	30	30	30	0.00	0.00
20	Small	10	10	0	0	0	10	3.24	0.00
	Medium	10	10	0	0	1	10	2.99	0.00
	Large	10	10	0	0	1	9	2.20	0.20
	Total/Average	30	30	0	0	2	29	2.81	0.07
25	Small	10	10	0	0	8	2	0.40	1.75
	Medium	10	10	0	0	4	6	1.35	0.57
	Large	10	10	0	0	5	5	0.98	1.31
	Total/Average	30	30	0	0	17	13	0.91	1.21
40	Small	10	9	0	0	10	0	0.00	34.54
	Medium	10	10	0	0	8	2	0.27	25.72
	Large	10	10	0	0	9	1	1.15	35.82
	Total/Average	30	29	0	0	27	3	0.47	32.03
50	Small	10	4	0	0	8	2	1.45	10.56
	Medium	10	6	0	0	8	2	1.79	34.77
	Large	10	8	0	0	6	4	1.67	17.94
	Total/Average	30	18	0	0	22	8	1.63	21.09

Table 3. Evaluation of the proposed local search algorithm.

n	Job size	Gap (in %) to the lower bound	
		UB_0	UB_1
10	Small	67.8	45.0
	Medium	67.4	44.2
	Large	69.3	46.5
20	Small	68.8	45.6
	Medium	69.7	46.9
	Large	68.4	45.2
25	Small	67.6	43.8
	Medium	68.1	45.1
	Large	69.0	46.1
40	Small	69.6	46.0
	Medium	69.6	46.0
	Large	68.7	44.6
50	Small	68.9	44.9
	Medium	69.0	44.9
	Large	68.9	45.1
Average		68.7	45.3

REFERENCES

Ageev, A. A. and A. E. Baburin (2007). Approximation algorithms for uet scheduling problems with exact delays. *Operations Research Letters 35*(4), 533 – 540.

Békési, J., G. Galambos, M. N. Jung, M. Oswald, and G. Reinelt (2014, Aug). A branch-and-bound algorithm for the coupled task problem. *Mathematical Methods of Operations Research 80*(1), 47–81.

Condotta, A. and N. Shakhlevich (2012). Scheduling coupled-operation jobs with exact time-lags. *Discrete Applied Mathematics 160*(16), 2370 – 2388.

Graham, R., E. Lawler, J. Lenstra, and A. R. Kan (1979). Optimization and approximation in deterministic sequencing and scheduling: A survey. *Annals of Discrete Mathematics 5*, 287 – 326.

Gurobi Optimization, L. (2018). Gurobi optimizer reference manual.

Khatami, M., A. Salehipour, and T. C. E. Cheng (2019). Coupled task scheduling with exact delays: Literature review and models. *European Journal of Operational Research*.

Li, H. and H. Zhao (2007, April). Scheduling coupled-tasks on a single machine. In *IEEE Symposium on Computational Intelligence in Scheduling*, pp. 137–142.

Orman, A. and C. Potts (1997). On the complexity of coupled-task scheduling. *Discrete Applied Mathematics 72*(1), 141 – 154.

Shapiro, R. D. (1980). Scheduling coupled tasks. *Naval Research Logistics Quarterly 27*(3), 489–498.

Sherali, H. D. and J. C. Smith (2005). Interleaving two-phased jobs on a single machine. *Discrete Optimization 2*(4), 348 – 361.