# 70068 Scheduling and Resource Allocation

# Assessed Coursework

Working: Pairs (recommended) or Individual
Submission deadline: 27th November 2024 (19:00)
To submit: PDF + ZIP (see spec)

## 1 Context

The topic of this coursework arises in the context of performance management of serverless workflows. These are workflows where jobs are executions of serverless functions, i.e. lightweight functions running remotely on the cloud. Serverless workflows are increasingly used in industry to implement image, video, and data processing pipelines. The processing times for the functions considered in this coursework are obtained from actual measurements on Microsoft Azure VMs. The functions deal with image processing: each function receives an input image, passes it through a neural network filter that mixes the content of an image with the style of another image, and creates in output a new image. The inputs and outputs of the functions create precedences in their executions, which can be described by means of a directed acyclic graph (DAG). The challenge is to schedule the execution of a serverless workflow DAG on a single machine as close to optimality as possible.

## 2 Assumptions

We take a number of assumptions similar to what was seen in the lectures:

**a)** A *machine* is a VM with a single CPU core

**b)** *Single machine scheduling*: only one function can run at a time.

**c)** Scheduling is *non-preemptive.*

**d)** Processing times should be treated approximately as *deterministic.* In reality, they depend on the processing sequence, since the image sizes get modified along the way, but for the sake of the scheduling model you should treat them as usual (sequence independent).

**e)** There are *no release times*, i.e., all jobs are ready at the earliest time at which their precedences are met.

**f)** A filter (e.g., *blur*) can be invoked multiple times in the same workflow. Each invocation can be treated as a separate job with identical processing time.

### 2.1 What to upload

- A PDF file with the answers to the questions. Length is unrestricted, but we recommend to aim for a couple of pages of text (excluding tables, figures, etc). Please include some text to help the markers understand how you structured your code.
- A ZIP archive with: i) your code, ii) a README file with minimal instructions to compile and use your code, iii) a printout (text file) of the execution of your code, which must print out the current solution considered at each iteration and its cost.

## 3 Questions

### 3.1 Question 1 (40%)

For a schedule $S$, consider an additive cost function $g(S) = \sum g_j(C_j)$ of the completion times of job $j = 1, \ldots, n$ and define $g^*_{max} = \max_j g_j(C_j)$. The *Least Cost Last* (LCL) rule, also called Lawler's algorithm, solves the cost minimization problem $1|prec|g^*_{max}$ to optimality. LCL finds the optimal schedule in backward order, from the last processed job to the first one. At each step, we look at the set of jobs $V$ such that their successors, if they exist, have all been already added to the schedule. Within the set $V$, we choose the job $l$ incurring the minimum cost $g_l(C_l)$. Ties are broken arbitrarily.

Quesiton 1 asks you to:

1. Give a short presentation of the proof at page 68 of the PDF available at the URL: `https://arxiv.org/pdf/2001.06005.pdf` (starting at "Let $N = \{1, 2, ..., n\}$ be the index set of all jobs, ...") that justifies why LCL is optimal (in the PDF notation, $f_j(\cdot)$ corresponds to our $g_j(\cdot)$). Include in your answer a brief discussion of each passage in the proof and an example with a very small DAG of your choice that illustrates the application of LCL.

2. Using a programming language of your choice, implement the LCL rule for a general DAG and apply it to the workflow, processing time and due dates given in **Appendix A**. In your code, use as the cost function, the tardiness $g_j(C_j) = T_j = max(0, C_j - d_j)$ with the due dates given in Appendix A. Report in your PDF answer sheet the first two iterations, the final iteration, and a few selected intermediate iterations obtained during the execution of your code, showing the partial schedule $S$ at each iteration.

### 3.2 Question 2 (60%)

Suppose now that we wish to solve a total tardiness problem for the same workflow studied in Question 1. The problem is now NP-hard and since the measure $\sum T_j$ is no longer expressible as a maximum of cost functions, the LCL rule is no longer optimal. Using a programming language of your choice, write a tabu search algorithm for the $1|prec|\sum T_j$ problem for the workflow given in **Appendix A**.
In your implementation:

- Make sure that your implementation is generic, i.e., it can accept an arbitrary set of processing times, due dates, and precedences (the latter can be assumed to form a directed acyclic graph).

- Include in your implementation code to generate a valid initial solution (which may not be optimal).

- Explore the local neighborhoods using the same rules used in the exercises of Problem Sheet 3, i.e., for a schedule 1234 if at iteration $k$ you last considered adjacent interchange e.g. $(2, 3)$, at iteration $k + 1$ consider first adjacent interchange $(3, 4)$, then $(1, 2)$, etc.

- Compared to the tabu search algorithms seen in class, you will need to introduce in the generation of the neighborhood a strategy to account for job precedences.

To illustrate your implementation, include in your answer the following results:

3.2.1 Execution of the tabu search method using the processing times, due dates, and precedences in **Appendix A**. Assume a tabu list length $L = 20$. Obtain the tabu search schedules with $K = 10$, $K = 100$, and $K = 1000$ iterations. Set the tolerance to $\gamma = 10$. Force the initial solution to be

$$x_0 = [30, 29, 23, 10, 9, 14, 13, 12, 4, 20, 22, 3, 27, 28, 8, 7, 19, 21, 26, 18, 25, 17, 15, 6, 24, 16, 5, 11, 2, 1, 31]$$

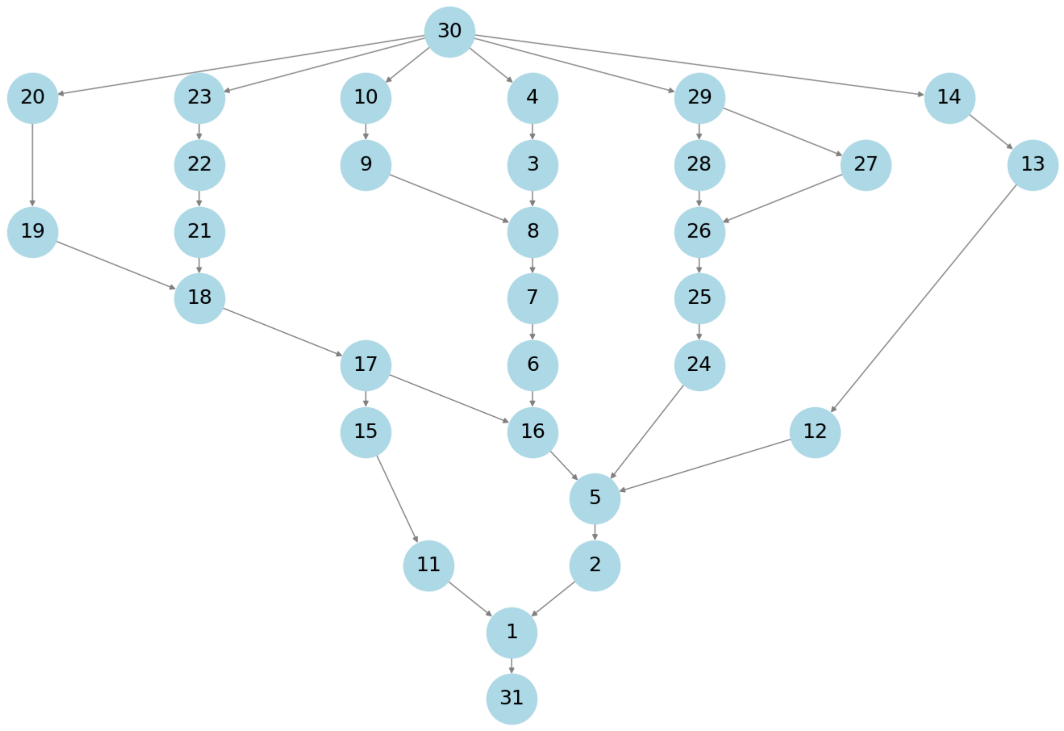
Include in the PDF the first few iterations and notable intermediate solutions (i.e., those where a new optimum is found) as the search progresses. Use a level of detail similar to what seen in the tabu search example in the lecture notes.

3.2.2 In this part, you are free to vary the values of $\gamma$ and $L$ as you wish. Include in your answer the best schedule $x_{TS}$ you find using the tabu search algorithm and its total tardiness. Discuss your findings, commenting on the effects that you observed by varying the parameters of $\gamma$ and $L$.

Suggestion: as you develop your code, you may consider applying your code to some of the exercises solved in the Tutorial Problem Sheet 3 to verify its correctness. You are *not* asked to document this debugging phase in the coursework answer.

## A Workflow

The figure below shows the considered image processing workflow, consisting of 31 nodes. Node 31 is the exit node. Despite its size may come across as large at first, in reality, this is of moderate size for typical workflows used in industry.



## A.1 Incidence matrix of the direct acyclic graph

Element `G(i,j)=1` if and only if there exists an edge from node $i$ to $j$.

MATLAB-like format (indices start at 1):

```
G(1,31)=1;
G(2,1)=1;
G(3,8)=1;
G(4,3)=1;
G(5,2)=1;
G(6,16)=1;
G(7,6)=1;
G(8,7)=1;
G(9,8)=1;
G(10,9)=1;
G(11,1)=1;
G(12,5)=1;
G(13,12)=1;
G(14,13)=1;
G(17,15)=1;
G(15,11)=1;
G(16,5)=1;
G(17,16)=1;
G(18,17)=1;
G(19,18)=1;
G(20,19)=1;
G(21,18)=1;
G(22,21)=1;
G(23,22)=1;
G(24,5)=1;
G(25,24)=1;
G(26,25)=1;
G(27,26)=1;
G(28,26)=1;
G(29,27)=1;
G(29,28)=1;
G(30,4)=1;
G(30,10)=1;
G(30,14)=1;
G(30,20)=1;
G(30,23)=1;
G(30,29)=1;
```

Python-like format (indices start at 0):

```
G[0, 30]=1;
G[1, 0]=1;
G[2, 7]=1;
G[3, 2]=1;
G[4, 1]=1;
G[5, 15]=1;
G[6, 5]=1;
G[7, 6]=1;
G[8, 7]=1;
G[9, 8]=1;
G[10, 0]=1;
G[11, 4]=1;
G[12, 11]=1;
G[13, 12]=1;
G[16, 14]=1;
G[14, 10]=1;
G[15, 4]=1;
G[16, 15]=1;
G[17, 16]=1;
G[18, 17]=1;
G[19, 18]=1;
G[20, 17]=1;
G[21, 20]=1;
G[22, 21]=1;
G[23, 4]=1;
G[24, 23]=1;
G[25, 24]=1;
G[26, 25]=1;
G[27, 25]=1;
G[28, 26]=1;
G[28, 27]=1;
G[29, 3]=1;
G[29, 9]=1;
G[29, 13]=1;
G[29, 19]=1;
G[29, 22]=1;
G[29, 28]=1;
```

## A.2 Nodes

The following table gives the node numerical index used in the previous incidence matrix declarations and the corresponding filter type. For example, nodes 3 and 12 are both of `emboss` type, so they should be treated as having the same processing time.

| Index | Type | Processing time | Due date | Index | Type | Processing time | Due date |
|---|---|---|---|---|---|---|---|
| 1 | onnx | 3 | 172 | 17 | wave | 6 | 233 |
| 2 | muse | 10 | 82 | 18 | wave | 6 | 77 |
| 3 | emboss | 2 | 18 | 19 | emboss | 2 | 88 |
| 4 | emboss | 2 | 61 | 20 | onnx | 3 | 122 |
| 5 | blur | 5 | 93 | 21 | emboss | 2 | 71 |
| 6 | emboss | 2 | 71 | 22 | onnx | 3 | 181 |
| 7 | vii | 14 | 217 | 23 | vii | 14 | 340 |
| 8 | blur | 5 | 295 | 24 | blur | 5 | 141 |
| 9 | wave | 6 | 290 | 25 | night | 18 | 209 |
| 10 | blur | 5 | 287 | 26 | muse | 10 | 217 |
| 11 | blur | 5 | 253 | 27 | emboss | 2 | 256 |
| 12 | emboss | 2 | 307 | 28 | onnx | 3 | 144 |
| 13 | onnx | 3 | 279 | 29 | wave | 6 | 307 |
| 14 | onnx | 3 | 73 | 30 | emboss | 2 | 329 |
| 15 | blur | 5 | 355 | 31 | muse | 10 | 269 |
| 16 | wave | 6 | 34 | | | | |

Processing times:

$$p = [3, 10, 2, 2, 5, 2, 14, 5, 6, 5, 5, 2, 3, 3, 5, 6, 6, 6, 2, 3, 2, 3, 14, 5, 18, 10, 2, 3, 6, 2, 10]$$

Due dates:

$$\begin{aligned} d = [&172, 82, 18, 61, 93, 71, 217, 295, 290, 287, 253, 307, 279, 73, 355, 34,\\ &233, 77, 88, 122, 71, 181, 340, 141, 209, 217, 256, 144, 307, 329, 269] \end{aligned}$$