XJCO3910 – Combinatorial Optimisation

Coursework 2: Modelling; Strong Formulations; B&B

Deadline: 1700 GMT on Wednesday 12 April 2023

Award: This piece of summative coursework is worth 10% of your grade

Submission:
1) Write or type your answers inside the boxes of the current document.
2) Scan **pages 1-4** using the MyPrint portal, including p. 1 (see instructions at https://it.leeds.ac.uk/it?id=kb_article&sysparm_article=KB0012731)
3) Upload one pdf-file containing **pages 1-4** to Gradescope (see instructions at https://help.gradescope.com/article/ccbpppziu9-student-submit-work#submitting_a_pdf)

Student Number	

Name	

(same as in Gradescope)

Question 1. **[12 marks]**

A furniture company needs to make a decision about the mix of products to be manufactured next week. It has seven types of desks, each with a profit (£) per unit and a production time (man-hours) per unit as shown below:

	Desk 1	Desk 2	Desk 3	Desk 4	Desk 5	Desk 6	Desk 7
Profit (£ per unit)	10	22	35	19	55	10	115
Production time (man-hours per unit)	1.5	2.0	3.7	2.4	4.5	0.7	9.5

The company has 720 man-hours available in the coming week.

The following restrictions are also placed on the production.

R1: If any amount of Desk 7 is to be produced, then an additional fixed cost of £2000 is incurred.

R2: If any amount of Desk 6 is to be produced, then 80 man-hours are needed for production line set-up and hence the (effective) number of man-hours available falls to 720-80=640.

R3: If any amount of Desk 1 is to be produced, then Desks 2 and 3 cannot be produced.

R4: If Desk 4 is to be produced, then at least 10 desks of that type should be produced.

Formulate an ILP to determine the quantities of desks to be produced in order to maximise the profit.
Use the following variables:

- x_i ($x_i \geq 0$ and integer) for the number desks of type i to be produced ($i = 1,2,\ldots,7$),
- δ_i ($\delta_i \in \{0,1\}$) to indicate whether desk i is to be produced ($\delta_i = 1$) or not ($\delta_i = 0$).

Present an ILP, without explaining how it is derived. If the model uses some numbers which are calculated based on the problem input, then present the formulae (no need to explain). As a sample answer, consider the final model for Problem 2 from Worksheet 5-6 (p. 11 of the solutions file). Do not include unnecessary constraints and unnecessary variables.

(do not solve the formulated ILP)

Question 2 **[5 marks]**

Consider the Production Planning problem (Example 7) from Chapter 6, and the corresponding MILP:

$$
\begin{array}{llllllllll}
\max \ z = & 5x_1 & +7x_2 & +3x_3 & & & & & & \\
\text{s.t.} & 3x_1 & +4\text{x}_2 & +2x_3 & -36\delta & & & & \leq & 30, \\
 & 4x_1 & +6x_2 & +2x_3 & +36\delta & & & & \leq & 76, \\
 & x_1 & & & & & & & \leq & 7, \\
 & & x_2 & & & & & & \leq & 5, \\
 & & & x_3 & & & & & \leq & 9, \\
 & x_1 & & & & -9\delta_1 & & & \leq & 0, \\
 & & x_2 & & & & -9\delta_2 & & \leq & 0, \\
 & & & x_3 & & & & -9\delta_3 & \leq & 0, \\
 & & & & & \delta_1 & +\delta_2 & +\delta_3 & \leq & 2, \\
 & \multicolumn{9}{l}{x_1, x_2, x_3 \geq 0,} \\
 & \multicolumn{9}{l}{\delta, \delta_1, \delta_2, \delta_3 \in \{0,1\}.}
\end{array}
$$

The disjunctive constraint

$$3x_1 + 4x_2 + 2x_3 \leq 30$$
or
$$4x_1 + 6x_2 + 2x_3 \leq 40$$

are modelled as

$$3x_1 + 4x_2 + 2x_3 \leq 30 + M'\delta, \qquad (1)$$
$$4x_1 + 6x_2 + 2x_3 \leq 40 + M'(1 - \delta). \qquad (2)$$

One possible choice is $M' = 36$ (see Chapter 6). In fact a smaller value of M' can be selected.
Find such a value. Provide a justification.

Question 3 **[23 marks]**

Consider an instance of the scheduling problem given by the table:

Job j	1	2	3	4	5
Release time r_j	0	2	14	27	40
Processing time p_j	6	18	10	17	16
Due date d_j	27	22	23	61	59

Find an optimal solution by the branch-and-bound algorithm. Present the search tree. Use the depth-first strategy, selecting each time the most promising node among **all** child nodes of the same parent node. Number the nodes of the tree in the order they are obtained. State the optimal solution and the optimal objective value.

The full description of the problem is given on pages 5-6, together with the details of the B&B algorithm.

Present the search tree. Number the nodes of the tree in the order they are obtained. Mark the nodes by corresponding partial schedules. For example, the node marked by (*,*,*,*,*) is the root of the tree; the node marked by (1,*,*,*,*) is one of the child nodes. For each node either specify its *LB* or explain why the node is discarded. State the initial *UB* and indicate how it is updated. Mark the node corresponding to the optimal schedule. Output a job permutation which defines an optimal schedule and the optimal objective value.

Problem description and instructions for **Question 3.**

A small garage specialises in car painting. The garage has sufficient room to keep multiple cars, but only one car can be painted at a time. For a job *j*, which corresponds to painting car *j*, three parameters are given:

- release time r_j (the time when car *j* is delivered to the garage for painting);
- processing time p_j (the time it takes to paint car *j*);
- due date d_j (the time by which it is desirable to complete job *j*).

The jobs can be completed in any order. A manager needs to decide in which order the jobs should be performed. Since it might not be possible to observe due dates for all jobs, the manager aims to provide a fair service to the customers, so that the maximum lateness is as small as possible.

Consider an instance with 5 jobs and the data given in the table below.

Job *j*	1	2	3	4	5
Release time r_j	0	2	14	27	40
Processing time p_j	6	18	10	17	16
Due date d_j	27	22	23	61	59

A schedule is defined by a job sequence. The starting time S_j of job *j* should satisfy the release time constraint

$$S_j \geq r_j.$$

If job *j* starts at time S_j, then its completion time is

$$C_j = S_j + p_j$$

and its lateness is

$$L_j = C_j - d_j. \qquad (1)$$

Notice that if job *j* is completed before its due date, then L_j is negative.

For example, if the jobs are processed in the order (1,3,2,4,5), then the Gantt chart of the schedule is shown in Fig. 1. In the schedule, there is an idle interval in-between jobs 1 and 3 caused by the release time $r_3 = 14$ of job 3.

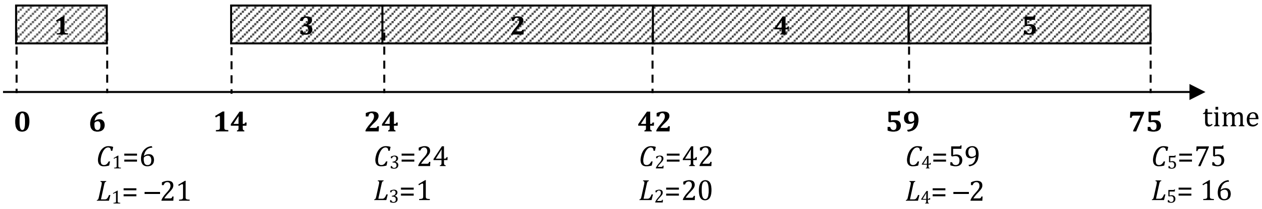


Fig. 1: The Gantt chart of the schedule given by job sequence (1,3,2,4,5)

The maximum lateness $L_{\max}$ for the above schedule is calculated as

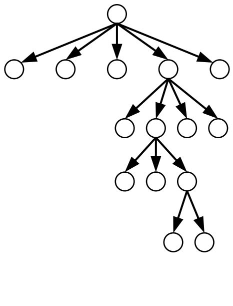
$$L_{\max} = \max\{L_1, L_2, L_3, L_4, L\_5\} = \max\{-21, 20, 1, -2, 16\} = 20.$$

In a schedule with a different job order, the values of L_j will be different, and therefore the value of the maximum lateness $L_{\max}$ might also be different.

Branching

The root of the search tree corresponds to an empty job sequenced (∗,∗,∗,∗,∗). Here ∗ in the job sequence indicates that no job has yet been assigned to the corresponding position in a sequence. We start by generating all child-nodes for (∗,∗,∗,∗,∗). They correspond to allocating one job to the first position. Thus for the 5-job instance, there are 5 nodes in level 1: (1,∗,∗,∗,∗), (2,∗,∗,∗,∗), (3,∗,∗,∗,∗), (4,∗,∗,∗,∗), (5,∗,∗,∗,∗). With the depth-first strategy, we compute *LB*s for all partial solutions in level 1, select the one with the smallest *LB* and branch from there.

If a node (j_1,*,*,*,*) is selected in level 1, then we create 4 child-nodes, which correspond to allocating one of the remaining 4 jobs to position 2, compute *LB*s for all partial solutions in level 2, select the one with the smallest *LB* and branch from there. The process continues until in the last level, 4 jobs are allocated, as the allocation of the 5th job is obvious. The solutions at that level are of the form (j_1, j_2, j_3, j_4,∗).



An outline of one round of branching, which leads to the leaf-level, is presented in the figure. Notice that this is just an outline. You need to label the nodes by indicating the partial solutions they represent, to state an *LB* for each node, to indicate how the *UB* is updated, and to possibly to perform further branching, if the stopping criterion is not reached after reaching the leaf-level.

<u>Upper bounding</u>

An initial upper bound can be based on schedule (1,3,2,4,5) shown in Fig. 1. Since the maximum lateness of that schedule is 20, we set *UB*=20. Whenever a better solution is found with an objective value smaller than *UB*, we replace *UB* by that smaller value.

<u>Lower bounding</u>

Consider a partial solution obtained after *k* jobs have been assigned to the first *k* positions.

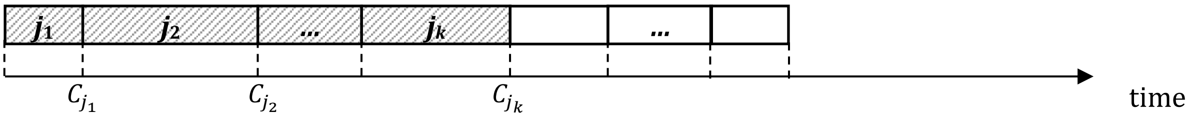


Fig. 2: A partial solution with *k* jobs assigned to the first *k* positions

Clearly, the maximum lateness $L_{\max}$ of any complete solution, that has the initial part of the schedule as in Fig. 2, can be estimated as

$$L_{\max} \geq \max\{L', L''\},$$

where L' is the actual maximum lateness of the jobs which have been assigned to the first *k* positions,

$$L' = \max\{L_{j_1}, L_{j_2}, \dots, L_{j_k}\},$$

and L'' is the lower bound on the maximum lateness of the remaining unscheduled jobs. As we do not know an optimal sequence of the unscheduled jobs and there is no efficient algorithm for finding it, we consider a relaxed problem, which is easier to solve. <u>We allow processing unscheduled jobs with pre-emption:</u>

- any unscheduled job j can be preempted and resumed later on;
- for every job, the total duration of its parts should be equal to the job processing time p_j.

The relaxed subproblem, defined for the unscheduled jobs processed with pre-emption, can be solved efficiently by the following algorithm: *at every decision point t which corresponds to a completion time or a release time of some job, schedule an <u>available</u> job with the smallest due date.* A job j is <u>*available*</u> at time t if it has not been processed in full before t, and $r_j \leq t$ (job j is released at time t or earlier).

Note that the formulated rule finds an optimal solution to the problem with pre-emption allowed. It provides a correct lower bound for scheduling jobs without pre-emption.

To illustrate the rule, consider calculating a lower bound for the partial schedule (1,∗,∗,∗,∗). We assume (temporarily) that unscheduled jobs {2,3,4,5} can be processed with pre-emption.

- The first decision point is time $t = 6$, when job 1 is completed and unscheduled jobs {2,3,4,5} may start. The only available job at $t = 6$ is job 2 ($r_2 \leq t$), and we assign job 2 to start at time $t = 6$.
- The next decision point is time $t = 14$, when job 3 becomes available. There are two available jobs {2,3} at $t = 14$. Since $d_2 < d_3$ $(22 < 23)$, processing of job 2 continues without interruption.
- The next decision point is time $t = 24$, when job 2 is completed. The only available job at $t = 24$ is job 3, and we assign job 3 to start at time $t = 24$.
- The next decision point is time $t = 27$, when job 4 becomes available. There are two available jobs {3,4} at $t = 27$. Since $d_3 < d_4$ $(23 < 61)$, processing of job 3 continues without interruption.
- At time $t = 34$ job 3 is completed. The only available job at $t = 34$ is job 4, and we assign job 4 to start at time $t = 34$.
- The next decision point is time $t = 40$, when job 5 becomes available. Since $d_5 < d_4$ $(59 < 61)$, we pre-empt job 4 and start job 5.
- Finally at time $t = 56$ job 5 is completed. We then allocate the remaining part of job 4.

The resulting scheduling is illustrated by Fig. 3. The lower bound for (1,*,*,*,*) is

$$LB = \max\{L', L''\} = \max\{-21, 2, 11, -3, 6\} = 11.$$

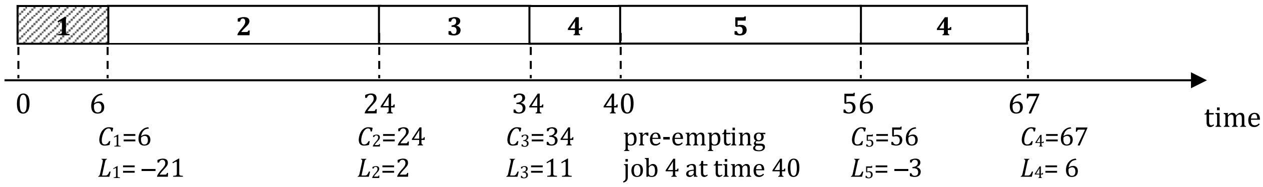


Fig. 3: The Gantt chart for LB-calculation for (1,∗,∗,∗,∗) (job 4 is processed with pre-emption)