Assignment 3

Teun Zwier [redacted], Jelle van der Schoot [redacted], Dragos Pop [redacted]
Group 54

October 2021

Exercise 3.1 - Single-Machine Scheduling

a) **Known information:** Duration, release time and due date
For i = 1, …, 10:
s_i = duration of job i
r_i = release time of job i
d_i = due date of job i
The goal is to minimize the total costs (or tardiness sum).

Decision variables:
x: starting time. For i = 1, …, 10: x_i = starting time job i
y: order of jobs. For i = 1, …, 10: For j = 1, … , 10: y_{ij} = order job i and job j
z: cost. For i = 1, …, 10: z_i = cost of job i

Necessary Constraints:
- x_i has to be greater than or equal to release time (can not start job before release time).
- x_i can not overlap with x_j so either end of job i is before or equal to x_j or j is before i.
- z_i has to be greater than or equal to 0 (can not have negative cost).
- z_i has to be greater than or equal to difference between end of job i and due date d_i.
- y_{ij} is a binary variable (so either 0 or 1, as job i is either before or after job j).
- y_{ij} and y_{ji} sum to 1, as either i is before j or j is before i, so either $y_{ij} = 1$ or $y_{ji} = 1$.

The model can be formally defined as follows

$$\min \sum_i z_i$$

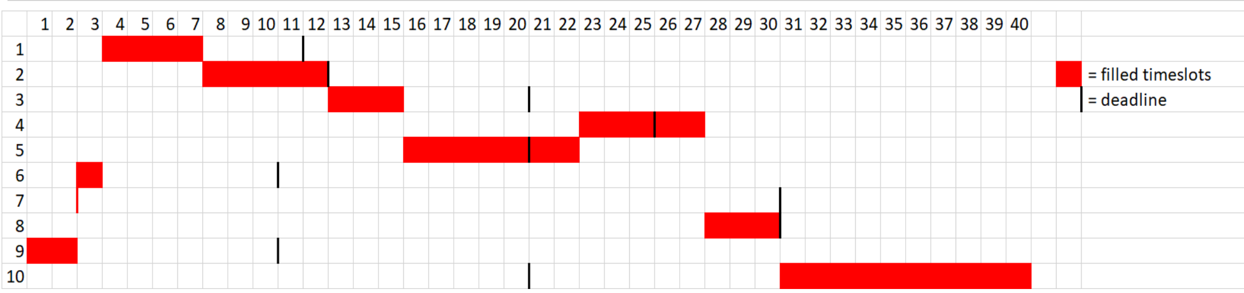
$s.t$	$x_i \geq r_i, \forall i$	(no start before release time)
	$x_i + s_i \leq x_j + M(1 - y_{ij}),\ \forall i, j,\ i \neq j,\ M \gg$	(no overlap)
	$z_i \geq 0,\ \forall i$	(no negative costs)
	$z_i \geq x_i + s_i - d_i,\ \forall i$	(difference end job i and due date)
	$y_{ij} \in \{0, 1\},\ \forall i, j,\ i \neq j$	(y is binary)
	$y_{ij} + y_{ji} = 1,\ \forall i, j,\ i < j$	(either $i \rightarrow j$ or $j \rightarrow i$)

Below, python code of the model is shown.

```python
# Known Variables
jobs = ['Job 1', 'Job 2', 'Job 3', 'Job 4', 'Job 5', 'Job 6', 'Job 7', 'Job 8', 'Job 9', 'Job 10']
s = [4, 5, 3, 5, 7, 1, 0, 3, 2, 10]  # Durations of the jobs
r = [3, 4, 7, 11, 10, 0, 0, 10, 0, 15] # Release times of the jobs
d = [11, 12, 20, 25, 20, 10, 30, 30, 10, 20] # Due dates of the jobs
n = len(jobs)  # Number of jobs

ILO_problem = pulp.LpProblem(name="ILO_problem", sense=pulp.LpMinimize)
# Decision Variables
x = [pulp.LpVariable(name=f'x_{i}', lowBound=0, cat='Continuous') for i in range(n)]
y = [[pulp.LpVariable(name=f'y_{i},{j}', cat='Binary') for j in range(n)]for i in range(n)]
z = [pulp.LpVariable(name=f'z_{i}', cat='Continuous') for i in range(n)]

# Objective
ILO_problem += sum(z), 'tardiness sum'

# Constraints
for i in range(n):
    ILO_problem += x[i] >= r[i], f'job_{i}_should_not_start_before_release_time'
    ILO_problem += z[i] >= x[i] + s[i] - d[i], f'cost_{i}_for_unit_time_after_deadline_{i}'
    ILO_problem += z[i] >= 0, f'cost_{i}_can_not_be_negative'
M = 100 #big_M
for i in range(n):
    for j in range(n):
        if i == j:
            continue  # skip if i = j, because a job can not be before or after itself
        ILO_problem += y[i][j] + y[j][i] == 1, f'job_{i}_is_before_or_after_job_{j}'
        ILO_problem += x[i] + s[i] <= x[j] + M*(1-y[i][j]), f'job_{i}_does_not_overlap_with_job_{j}'
# Solve
ILO_problem.solve()
print("Optimization status:", pulp.LpStatus[ILO_problem.status])
print('with objective value of', ILO_problem.objective.value())
for v in ILO_problem.variables():
    print(v.name, "=", v.varValue)
```

This code gave the following schedule:

job	1	2	3	4	5	6	7	8	9	10
x	3	7	12	22	15	2	2	27	0	30
z	0	0	0	2	2	0	0	0	0	20



The algorithm finds an optimal solution with a total cost of 24. This solution has the following order: 9 - 7 - 6 - 1 - 2 -3 - 5 - 4 - 8 - 10. Only job 4, 5 and 10 can not be finished before their respective due dates. It makes sense that job 10 gives the highest tardiness, as the job is both the longest (10) and has the shortest amount of time to finish (5).

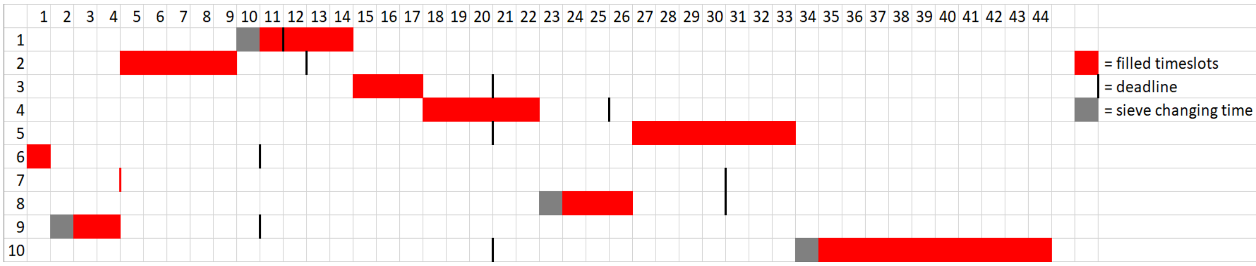
b) To add the sieve change, two additions had to be made. First, the sieves of the jobs were added as an array to known variables. sieve = [1, 2, 1, 1, 2, 1, 2, 2, 2, 1] # Sieve type
Second, an additional rule was added to the constraints that prevents overlap as shown below:

```python
sieve_change = abs(sieve[i] - sieve[j]) # 0 if same sieve type, 1 if different sieve type
ILO_problem += x[i] + s[i] + sieve_change <= x[j] + M*(1-y[i][j]), f'job_{i}_does_not_overlap_with_job_{j}'
```

So using the sieve array, the difference between sieve type of job i and job j is computed. If the sieve type is different for two jobs, 1 timeslot is added to the overlap rule, to give space for a sieve change.

The results from this updated algorithm can be seen below:

job	1	2	3	4	5	6	7	8	9	10
x	10	4	14	17	26	0	4	23	2	34
z	3	0	0	0	13	0	0	0	0	24



This gives fairly similar results to the model in a). Order: 6 - 9 - 7 - 2 - 1 - 3 - 4 - 8 - 5 - 10. Again, the first jobs to be finished are jobs 9, 6 and 7 and the last job is 10. Most changes occur in the middle. The optimal solution gives a tardiness sum of 40, with a delay of 3 for job 1, 13 for job 5 and 24 for job 10. Because of sieve changes between job 6 & 9 and 2 & 1, job 1 had to be pushed ahead over it's due date. Where job 5 was put in between 3 and 4 the last time , this was less desirable this time, as the sieve type would have to be changed before and after job 5. As such it's pushed ahead to a timeslot after job 3 and 4 (and 8).4 sieve changes were made, which makes the total schedule exactly 4 timeslots longer. This causes an increase of 4 in the delay for job 10 - job 10 starts at 34 and ends at 44, instead of starting at 30 and ending at 40.

Exercise 3.2 - Project Planning

a) First, the duration of each activity was sampled randomly from an exponential distribution with the given expectation d_i. As a function for the inverse of the exponential cumulative distribution does not exist in Excel, $GAMMA.INV(RAND(); 1; d_i)$ was used because a random variable from a Gamma distribution with alpha 1 and beta 1/λ is exponentially distributed with lambda λ. When it comes to the beta parameter of the function, the given expected duration was used because d_i = 1/λ =E(X~Exp(λ)) = E(X~Gamma(1,1/λ)). Alternatively, $-\ LN(RAND())\ /(1/d_i)$could have been used instead as shown in slide 19 of Lecture 11.
Next, the finish times for each activity were computed as one can see in the following table, where X_i is the previously generated duration of activity i:

Activity	Formula
1	X_1
2	$X_1 + X_2$
3	$X_1 + X_3$
4	$X_1 + X_4$
5	$MAX(X_1 + X_2, X_1 + X_3) + X_5$

6	$MAX(X_1 + X_3, X_1 + X_4) + X_6$
7 (Finish time)	$MAX[MAX(X_1 + X_2, X_1 + X_3) + X_5, MAX(X_1 + X_3, X_1 + X_4) + X_6] + X_7$

This was then repeated 10000 times in order to simulate the project and the average of all finishing times pointed to an expected project finish time of 14,04 days.

b) At this point, the simulated values previously generated were used to compute the lower and upper bound of the 95% Confidence Interval as follows (where s is the sample standard deviation $STDEV.S$ and n is the number of simulations, namely 10000):

$$\text{CI} = \overline{X} \pm t_{\alpha/2} \frac{s}{\sqrt{n}} = \left[\overline{X} - t_{\alpha/2} \frac{s}{\sqrt{n}}, \overline{X} + t_{\alpha/2} \frac{s}{\sqrt{n}}\right]$$

Since $\alpha = 5\%$ and $\alpha/2 = 2,5\%$, the 97.5 percentile of t-dist with n-1 was computed using $T.INV(0,975; 10000 - 1) = 1,96$, even though 2 could be used instead given the large number of simulations. Accordingly, the 95% Confidence Interval found is [13,92, 14,16] and signifies that we can be 95% confident that the true expected finish time of the project will take between 13,92 and 14,16 days to finish.

c) The probability that a project takes over 12 days was measured taking the average number of simulations with a finish time over 12. Consequently, a probability of 0,575 was found. Regarding the 95% confidence interval, following a similar approach as in the previous point, this was found to be [0,565, 0,584], meaning that we can be 95% confident that the true probability that the project takes over 12 days is between 0,565 and 0,584.

T26 =T19+T23*T24/SQRT(T22)

	A	B	C	D	E	F	G	H
1								
2	Expected duration d_i of activity i							
3	i	1	2	3	4	5	6	7
4	d_i	1	3	2	3	4	5	2
5								
6		Realizations of each activity duration						
7	Simulation run	1	2	3	4	5	6	7
8	1	0,7122733	5,975719	4,0174801	3,4990529	4,6102617	4,9226689	0,4852002
9	2	1,2060343	1,6270421	2,0621155	0,0408044	5,3310649	0,6866119	1,6415772
10	3	0,0526862	5,3384289	2,5581127	6,3556472	1,7079706	1,1626476	0,516558
11	4	0,7756332	3,9797345	3,0275067	1,6122376	0,0526817	7,0201932	0,401627
12	5	0,6609862	5,7964683	4,7870286	0,5197517	0,5537911	6,1715606	3,8227068
13	6	0,0008127	2,5863006	2,1214926	1,3778212	1,7469604	14,070353	1,7954389
14	7	0,1663013	1,7040847	4,7941739	0,2243696	0,9163678	20,647842	2,0389806
15	8	0,0621942	3,8836098	2,367951	2,484754	6,3620428	5,2890236	0,399634
16	9	0,2105412	10,808717	2,5789877	5,1259006	0,2887555	3,1784367	0,7012559
17	10	1,3846026	17,092497	2,33939	0,6542299	2,6390065	7,1087046	1,0269989
18	11	1,8821112	7,8977637	0,3255108	4,4184735	2,824031	3,0300039	0,5494928
19	12	0,5015515	0,0358599	2,831502	11,830879	2,8375107	7,7536852	2,0062996
20	13	2,2648562	1,3337033	0,9645325	1,9303046	4,8586064	3,0124161	1,3711615
21	14	0,5990283	9,0282847	2,0203244	0,4548389	0,7730764	1,654295	0,0112617
22	15	0,4937942	2,9650096	2,3527732	2,8584792	4,4983459	4,5713057	0,6171629
23	16	2,3534524	1,3166692	13,492203	1,3565168	0,6901376	7,2366317	0,7805124
24	17	1,953421	5,4830131	1,1909733	1,0314226	7,2846813	2,6786024	0,0226166
25	18	2,4750905	0,8462678	3,0947496	7,6091547	0,1184301	0,6264292	0,3959024
26	19	0,2965759	2,8571776	0,8204275	3,0357498	5,0834127	5,8324792	1,8036871
27	20	0,4120736	4,0514875	1,533573	2,4286742	4,2669683	18,771388	4,8046214

	I	J	K	L	M	N	O	P
6	Finish Times							
7	1	2	3	4	5	6	7 (Finish time)	larger than 12
8	0,7122733	6,6879924	4,7297534	4,2113262	11,298254	9,6524223	11,7834543	0
9	1,2060343	2,8330763	3,2681498	1,2468386	8,5992147	3,9547617	10,24079189	0
10	0,0526862	5,3911151	2,6107989	6,4083334	7,0990857	7,5709811	8,087539087	0
11	0,7756332	4,7553678	3,8031399	2,3878708	4,8080495	10,823333	11,2249601	0
12	0,6609862	6,4574544	5,4480148	1,1807379	7,0112455	11,619575	15,4422822	1
13	0,0008127	2,5871133	2,1223053	1,3786339	4,3340737	16,192658	17,98809698	1
14	0,1663013	1,870386	4,9604751	0,3906709	5,8768429	25,608317	27,64729764	1
15	0,0621942	3,945804	2,4301452	2,5469482	10,307847	7,8359718	10,70748078	0
16	0,2105412	11,019258	2,7895288	5,3364417	11,308014	8,5148785	12,00926977	1
17	1,3846026	18,477099	3,7239926	2,0388324	21,116106	10,832697	22,14310452	1
18	1,8821112	9,7798749	2,2076221	6,3005847	12,603906	9,3305886	13,15339872	1
19	0,5015515	0,5374114	3,3330535	12,33243	6,1705642	20,086115	22,09241498	1
20	2,2648562	3,5985595	3,2293888	4,1951609	8,457166	7,2075769	9,828327451	0
21	0,5990283	9,627313	2,6193527	1,0538672	10,400389	4,2736477	10,41165112	0
22	0,4937942	3,4588038	2,8465674	3,3522734	7,9571498	7,9235791	8,574312653	0
23	2,3534524	3,6701216	15,845655	3,7099693	16,535793	23,082287	23,86279924	1
24	1,953421	7,4364341	3,1443944	2,9848436	14,721115	5,8229968	14,74373206	1
25	2,4750905	3,3213584	5,5698402	10,084245	5,6882703	10,710674	11,10657676	0
26	0,2965759	3,1537535	1,1170034	3,3323257	8,2371662	9,1648049	10,96849202	0
27	0,4120736	4,4635611	1,9456466	2,8407478	8,7305294	21,612136	26,41675769	1

	R	T
6	a)	
7	sample avg	14,043692
10	b)	
11	n	10000
12	97.5 percentile of t-dist with n-1	1,9602013
13	Sample standard deviation (s)	6,182542
14	Left CI bound	13,922501
15	Right CI bound	14,164882
18	c)	
19	P(finish time >= 12)	0,575
21	CI	
22	n	10000
23	97.5 percentile of t-dist with n-1	1,9602013
24	Sample standard deviation (s)	0,4943677
25	Left CI bound	0,5653094
26	Right CI bound	0,5846906

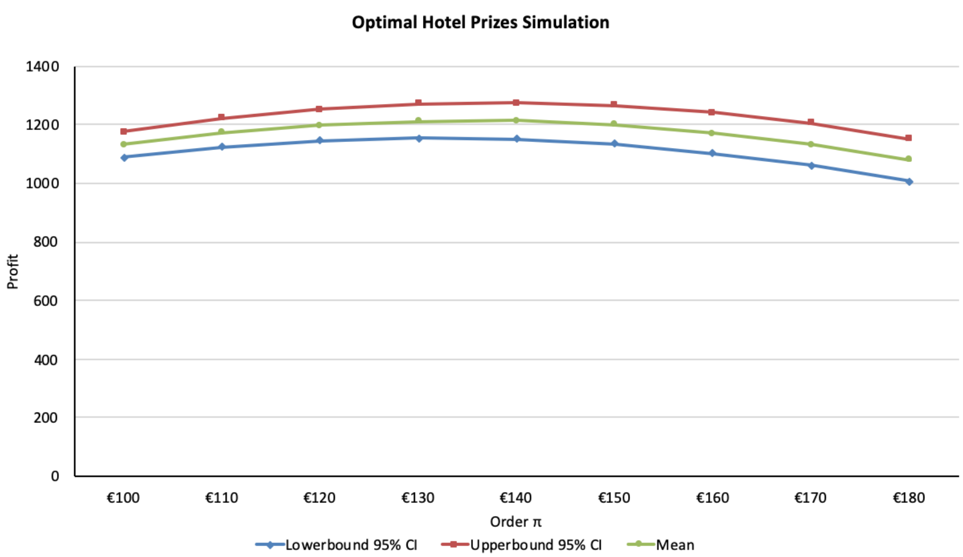
Exercise 3.3 - Optimal Hotel Prizes

a) An Excel simulation can be used to calculate the optimal hotel prizes. We introduce the variable π_k representing the prize for a hotel room: {100, 110, 120, 130, 140, 150, 160, 170, 180}. The demand can be simulated using:
$= BINOM.INV(20; 0,9 - (roomprice/300);\ RAND()) * roomprice$. The profit is calculated by multiplying the demand with the prize of each room.

A setup of the Excel sheet can be found below.

Mean	1132,55628	1172,7877	1199,42876	1212,8224	1213,75756	1201,4896	1172,20525	1133,81447	1079,36096
Standard deviation	221,894348	245,918162	268,194414	289,992656	311,132877	328,511476	344,45803	358,145678	368,633361
Lowerbound 95% CI	1088,17741	1123,60406	1145,78988	1154,82387	1151,53099	1135,78731	1103,31365	1062,18533	1005,63428
Upperbound 95% CI	1176,93515	1221,97133	1253,06764	1270,82093	1275,98414	1267,1919	1241,09686	1205,4436	1153,08763
	Order \pi								
Simulation run	100	110	120	130	140	150	160	170	180
1	*1500*	*1100*	*1200*	*1560*	*980*	*1200*	*960*	*1870*	*900*
2	*1400*	*1320*	*1320*	*910*	*1260*	*1050*	*1120*	*1190*	*1260*
3	*1300*	*1100*	*1320*	*1690*	*1120*	*750*	*1280*	*1020*	*1440*
4	*1300*	*1210*	*1200*	*1300*	*1260*	*1050*	*1280*	*1190*	*1260*
5	*1200*	*1430*	*1440*	*1300*	*1820*	*1500*	*640*	*1360*	*720*
6	*1200*	*1540*	*960*	*1300*	*1260*	*600*	*480*	*1360*	*1080*
7	*1300*	*1320*	*1080*	*1430*	*1120*	*1050*	*480*	*850*	*540*
8	*1200*	*660*	*1080*	*1560*	*980*	*1050*	*1440*	*850*	*1260*
9	*800*	*990*	*1440*	*1430*	*700*	*1050*	*1120*	*1360*	*1080*
10	*1000*	*1650*	*1080*	*1300*	*1260*	*1200*	*800*	*1190*	*900*
11	*1100*	*1760*	*1080*	*910*	*700*	*1800*	*800*	*680*	*720*
12	*1100*	*1540*	*1080*	*910*	*1680*	*1500*	*480*	*1190*	*540*

Looking at the graph, the mean profit (after 100.000 simulations per π_k) is the highest for a hotel room price of € 140.



The t-test ($\alpha = 5\%, df = \infty$) can be used to verify whether this result is significant.

$H_0: \mu_1 = \mu_2$
$H_1: \mu_1 \neq \mu_2$

Reject if bigger than 1.960
Test statistic of $\pi_{140}\ vs.\ \pi_{150}$ = 2.107

The test statistic is in the critical region, so the null hypothesis is rejected: the difference is significant.

b) An alternative method is ranking and selection. 900 simulations were used to generate the following matrix containing the test statistics for each of the prices.

For each π, the following two-sided t-test for equal means is performed with $\alpha = 5\%$.

$H_0: \mu_1 = \mu_2$
$H_1: \mu_1 \neq \mu_2$

Green cells indicate significant differences between the means and can be discarded when those are lower. Because of this, the prices 100, 160, 170 and 180 are discarded.

Cr. Value	1,960								
H0 / H1	***100***	***110***	***120***	***130***	***140***	***150***	***160***	***170***	***180***
100		4,938	5,461	6,409	4,852	5,509	1,424	0,171	3,006
110			0,691	1,952	0,461	1,428	2,555	3,858	6,573
120				1,270	0,176	0,811	3,073	4,319	6,968
130					1,362	0,353	4,045	5,191	7,714
140						0,925	2,756	3,969	6,500
150							3,483	4,607	7,010
160								1,297	3,739
170									2,347
180									

We have 5 prices left for 9100 simulations: {110, 120, 130, 140, 150}, so each price is simulated 1.820 times. This results in the highest mean revenue for π_{140}, so the hotel room price of 140 will yield the highest revenue: € 1.216,81.