Assignment 3 Group 4

Katrin Grunert, Carmen Byrne Salsas, Bono Lardinois

10/23/2022

Statistics, Simulation & Optimization: Assignment 3

3.1 Single-Machine Scheduling

3.1.a

To solve this single-machine scheduling problem, an integer linear optimization model was created by using PULP. With this model, the most optimal solution was found to complete all ten jobs with the least amount of tardiness. The model has the following decision variables.

- jobs: the names of jobs available
- d_i: duration of a certain job
- r_i: release time of a certain job in hours
- t_i: due date of a certain job
- n: number of jobs
- y_i_j: decision variable, with y_i_j = 1 if job i goes before before job j
- z_i: tardiness of each job

The model also made use of the following constraints.

1. No overlap: $x_i + d_i <= x_j + 1000 * y_{i,j}$ (for each i, j. As long i is not j)
2. Job order: $y_{i,j} + y_{j,i} = 1$ (for each i, j. As long i is not j)
3. release date: $x_i >= r_i$ (for each i)
4. Start and duration times not greater than the due date (tardiness): $x_i + d_i - t_i <= z_i$, $z_i >= 0$ (for each i)
5. Binary decision variable: $y_{i,j} \in 0,1$, (for all i and j. As long i is not j)

i and j are the number of jobs with a possible integer value of 1 to 10.

To calculate the optimal solution to this problem:

$$\sum_{n=1}^{10} c_i * z_i$$

Here c_i is 1, and z_i is the tardiness of a job.

After running the model, this gave the following optimal schedule:

Job order	6	7	9	1	2	3	5	4	8	10
Start time (x_i)	0	1	1	3	7	12	15	22	27	30
Tardiness (z_i)	0	0	0	0	0	0	2	2	0	20

In this schedule, we start with job 6 and end with job 10. Tasks 5, 4, and 10 will have tardiness with a total of 24 hours. This means that the optimal schedule the machines need an extra 24 hours in order to complete all the jobs.

3.1.b

Compared to the problem in exercise, a few things need to change to model the optimal solution. Now Jobs 1, 3, 4, 6, and 10 require a sieve of type 1, whereas jobs 2, 5, 7, 8, and 9 require a sieve of type 2. When jobs come after each other with a different sieve, the extra time it costs to change the sieve is 1 hour. The constraints in this problem were modified with an extra binary decision variable: $costs_{i,j}$. This variable will be 1 if the jobs require a different sieve. Moreover, the constraint of preventing overlap was modified to implement the cost variable correctly. The new no-overlap constraint is now:

$$x_i + d_i + cost_i <= xj + 1000 * y_{i.j}$$

With this new constraint, the 1 hour of extra time will be added if jobs x_i and x_j are not in the same groups. This gave a result of a total tardiness of 40 hours. The table with the optimal schedule is presented below:

Job order	9	7	6	1	3	2	5	8	4	10
Start time (xi)	0	2	3	4	8	12	17	24	28	33
Tardiness (zi)	0	0	0	0	0	5	4	0	8	23

Compared to the previous schedule, we see a different order and a delay for the following jobs: 6, 2, 4. This means there will be a change of sieve three times. This has changed the order of the most optimized solution and brings 16 additional hours compared to the first schedule without the extra costs of sieve change.

3.2 Project Planning

Exercise 3.2.a

The project consists out of 7 activities (1,2,3,4,5,6,7) which all have a respective expected duration $d_i = [1,3,2,3,4,5,2], i \in [1,2,3,4,5,6,7]$

Their finish times FT_i are formulated according to the dependencies depicted in the assignment description as follows:

1. $FT_1 = d_1$
2. $FT_2 = d_1 + d_2$
3. $FT_3 = d_1 + d_3$
4. $FT_4 = d_1 + d_4$

5. $FT_5 = max(d_2, d_3) + d_5$
6. $FT_6 = max(d_3 + d_4) + d_6$
7. $FT_7 = max(d_5, d_6) + d_7$

The activity durations follow an exponential distribution $r(x) = e^{-d_i x}$ and therefore the function $x = \frac{-ln(u)}{\frac{1}{d_i}}$ was used to simulate new values for each activity, where u is a random sample from a uniform distribution.

This was done 10,000 times in Excel, and the sample average (expected project finish time) was 14.081.

Exercise 3.2.b

Next, we calculate a 95% confidence interval using the sample standard deviation of 6.167, the z-score of 1.96, and the sample size of 10,000.

We can say with 95% confidence that the expected finish time lies between the lower and upper bounds of [13.85, 14.09]

Exercise 3.2.c

The probability that the project takes more than 12 days can be represented as a sample proportion p. Therefore we count the occurrences where the finishing time is greater than 12 in our 10,000 samples and divide that by the total number of samples.

Doing so, the sample proportion is 0.581.

Knowing the sample proportion, we can construct the 95% confidence interval using the sample standard deviation of 0.494, the z-score of 1.96, and the sample size of 10,000.

With 95% confidence, we can say that the probability for the expected finishing time to be greater than 12 is between the lower and upper bounds of [0.571, 0,591].

3.3 Optimal Hotel Prizes

3.3.a

We calculated the probability parameter of the demand distribution (p_sucess) based on the formula 0.9 - price/300 where the price ranges from 100 to 180 euro. Then, we simulated the revenue for each price 100000 times using the excel formula MIN(10, BINOM.INV(n_trials, p_sucess, RAND())) * price. In this formula, n_trials is equal to 20 according to the parameters of the demand probability distribution. The inverse binomial distribution of the demand returns how many rooms were demanded in that simulation round. We take the minimum between the demand and 10 because only 10 rooms can be occupied. Finally we multiply the number of demanded rooms by their price to calculate the revenue.

We then calculated 95% confidence intervals for the mean revenue for each price using the student t-distribution, the sample average and the sample standard deviation. At a

significance level of 0.05 and with 100000-1 degrees of freedom, the t critical value is approximately equal to 1.96. We obtained the following confidence intervals:

Price	100	110	120	130	140
Mean	963.10	1035.33	1094.06	1135.85	1161.62
95% Confidence interval	[793.71, 1132.48]	[801.12, 1269.54]	[785.09, 1403.03]	[746.09, 1525.61]	[693.97, 1629.28]
Confidence interval width	338.78	468.42	617.94	779.52	935.31

Price	150	160	170	180
Mean	1169.70	1154.40	1124.76	1076.41
95% Confidence interval	[626.64, 1712.76]	[542.94, 1765.86]	[460.80, 1788.73]	[372.61, 1780.21]
Confidence interval width	1086.12	1222.93	1327.94	1407.60

From this table it would seem that the best price is 150€ since it yields the highest mean revenue (1169.70€); however, the confidence intervals grow wider as the prices increase which raises the doubt as to whether 150€ is significantly better than the lower prices.

In order to address this question we performed pairwise t-tests using Sidak's correction for the significance level. There are 9 possible prices; hence, for a given price π, and every other price π' we will test the following hypotheses:

H0: π is better than π'
H1: π is not better than π'

Given an original significance level α of 0.05, the corrected significance level α' is $1 - (1 - \alpha)^{1/(9-1)}$, which is approximately equal to 0.00369.

Let μ and s be the mean revenue and its sample standard deviation for a price π (μ' and s' for price π'). The corrected hypotheses become then:

H0: $\mu > \mu' - \beta \frac{\sqrt{s^2+s'^2}}{\sqrt{n}}$

H1: $\mu \leq \mu' - \beta \frac{\sqrt{s^2+s'^2}}{\sqrt{n}}$,

where n is equal to the number of simulations per price (100000), and beta is equal to the inverse t-distribution at $1 - \alpha'$ with $n - 1$ degrees of freedom (calculated using the following expression in excel: T.INV(1-0.00369, 100000-1)), namely 2.49.

The only price for which the null hypothesis was not rejected was 150€, meaning that 150€ is the optimal price to maximize the revenue in this problem at a 0.05 significance level.

3.3.b

In order to implement the "ranking and selection option 2" algorithm we simulated revenues using the same excel commands as in part 3.3.a. The main difference is that we had a total budget of 900 simulations to split between 9 prices for the first rounds of simulations; hence, n is equal to 100 (i.e., 900/100).

We then performed pairwise t-tests as explained in part 3.3.a; the main difference is that β is equal to the inverse t-distribution at $1 - \alpha'$ with $100 - 1$ degrees of freedom; namely, 2.54.

The hypotheses are similar to those in part 3.3.a.; namely:

H0: $\mu > \mu' - \beta \frac{\sqrt{s^2+s'^2}}{\sqrt{n}}$

H1: $\mu \leq \mu' - \beta \frac{\sqrt{s^2+s'^2}}{\sqrt{n}}$.

For this set of prices {130, 140, 150, 160, 170} the null hypothesis could not be rejected when compared to every other price (from the set {100, 110, …, 180}). Hence, 5 prices moved on to the next round of simulations. The remaining simulation budget was 9100; hence, the revenue was simulated 1820 (i.e., 9100/5) times for each price.

At a significance level of 0.05 and with 1820-1 degrees of freedom, the t critical value is approximately equal to 1.96. We obtained the following confidence intervals:

Price	130	140	150	160	170
Mean	1138.71	1158.85	1163.82	1156.57	1124.06
95% Confidence interval	[755.40, 1522.03]	[686.81, 1630.88]	[614.29, 1713.35]	[549.12, 1764.02]	[459.12, 1788.99]
Confidence interval width	766.62	944.07	1099.06	1214.89	1329.88

Thus, with this method the price that achieves the largest mean revenue (1163.82€) is also 150€; hence, 150€ is the optimal price in order to maximize the revenue which agrees with our answer from part 3.3.a.

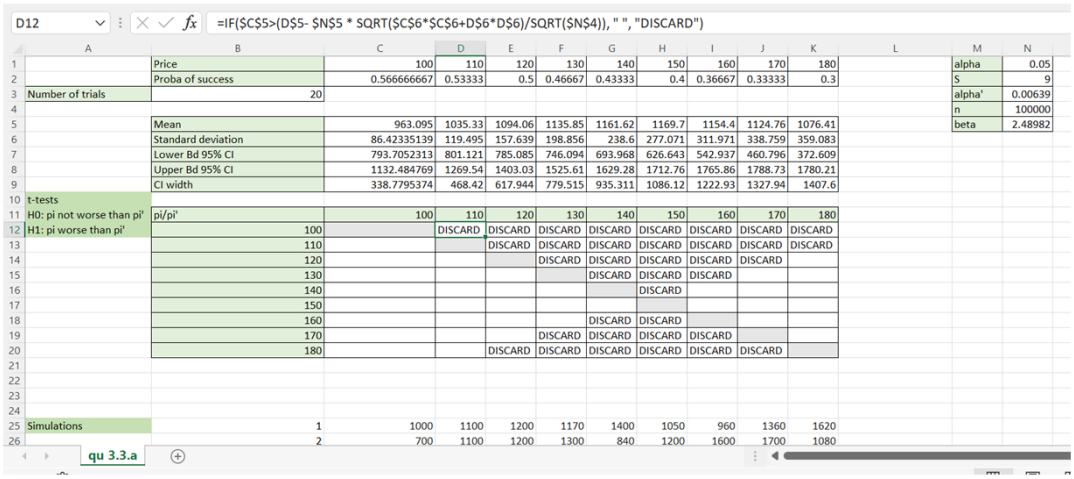
Appendix

```
import pulp

# data

# Jobs
jobs = ['Job 1', 'Job 2', 'Job 3', 'Job 4', 'Job 5', 'Job 6', 'Job 7', 'Job 8', 'Job 9', 'Job 10']

# Duration of jobs
d = [4,5,3,5,7,1,0,3,2,10]

# Releasetime
r = [3,4,7,11,10,0,0,10,0,15]

# Due date
t = [11,12,20,25,20,10,30,30,10,20]

# number of jobs
n = len(jobs)

# constant M for decision variable
M = 1000

# initiate model
ILO_problem = pulp.LpProblem(name="ILO_problem", sense=pulp.LpMinimize)

# define decision variables

# staring time of job i
x = [pulp.LpVariable(name=f'x_{i}', lowBound=0, cat='Continuous') for i in range(n)]

# binary decision variable
y = [[pulp.LpVariable(name=f'y_{i},{j}', cat='Binary') for j in range (n)] for i in range(n)]

# tardiness of job i
z = [pulp.LpVariable(name=f'z_{i}', lowBound=0, cat='Continuous') for i in range(n)]

# objective --> minimize total cost (tardiness)
ILO_problem += sum([z[i] for i in range (n)]), 'total tardiness'

# constraint: no overlap
for i in range (n):
    for j in range (n):
        if i == j:
            continue
        ILO_problem += x[i]+d[i] <= x[j]+M*y[i][j], f'job_{i}_finishes_after_bin_var_{i}{j}'

# constraint: order
for i in range(n):
    for j in range (n):
        if i == j:
            continue
        ILO_problem += y[i][j]+y[j][i]==1, f'bin var {i}{j} vs bin var {j}{i}'

# constraints release
for i in range(n):
    ILO_problem += x[i]>=r[i], f'job_{i}_after_release{i}'

# constraint: start and duration time not over due date
for i in range(n):
    ILO_problem+= x[i]+d[i]-t[i] <= z[i], f'job{i} should be under tardiness {i}'


# Solve problem
print(ILO_problem)
ILO_problem.solve()
print("Status:", pulp.LpStatus[ILO_problem.status])

for v in ILO_problem.variables():
    print(v.name,"=",v.varValue)

print("Optimized obj func =", pulp.value(ILO_problem.objective))
```

Fig 1: Screenshot of Python code (Exercise 3.1 A)

```
# data

# Jobs
jobs = ['Job 1', 'Job 2', 'Job 3', 'Job 4', 'Job 5', 'Job 6', 'Job 7', 'Job 8', 'Job 9', 'Job 10']

# Duration of jobs
d = [4,5,3,9,7,1,8,3,3,10]

# Release time
r = [3,6,7,11,10,5,0,18,0,15]

# Due date
b = [11,12,20,25,20,18,30,38,10,20]

# number of jobs
n = len(jobs)

# constant M for decision variable
M = 1000

list_1 = [1,3,6,8,10]
list_2 = [2,5,7,8,9]

# initiate model
ILO_problem = pulp.LpProblem("ILO_problem", sense=pulp.LpMinimize)

# define decision variables

# starting time of job i
s = [pulp.LpVariable(name=f's_{i}', lowBound=0, cat='Continuous') for i in range(n)]

# binary decision variable
y = [[pulp.LpVariable(name=f'y_{i},{j}', cat='Binary') for j in range (n)] for i in range(n)]

# tardiness of job i
a = [pulp.LpVariable(name=f'a_{i}', lowBound=0, cat='Continuous') for i in range(n)]

cost = [[pulp.LpVariable(name=f'cost_{i},{j}', cat='Binary') for j in range (n)] for i in range(n)]


# objective --> minimize total cost (tardiness)
ILO_problem += sum([a[i] for i in range (n)]), 'total tardiness'

# constraint: no overlap
for i in range (n):
    for j in range (n):
        if i == j:
            continue
        ILO_problem += s[i] + d[i] + cost[i][j] <= s[j]+M*y[j][i], f'job_{i}_finishes_after_bin_var_{i}{j}'

# constraint: order
for i in range(n):
    for j in range (n):
        if i == j:
            continue
        if (i+1 in list_1) and (j+1 in list_2) or (i+1 in list_2) and (j+1 in list_1):
            ILO_problem += cost[i][j]==1, f'bin var cost {i}{j} vs bin var cost{j}{i}'


# constraint: order
for i in range(n):
    for j in range (n):
        if i == j:
            continue
        ILO_problem += y[i][j]+y[j][i]==1, f'bin var {i}{j} vs bin var {j}{i}'

# constraints release
for i in range(n):
    ILO_problem += s[i]>=r[i], f'job_{i}_after_release[i]'

# constraint: start and duration time not over due date
for i in range(n):
    ILO_problem+= s[i]+d[i]-b[i] <= a[i], f'job{i} should be under tardiness {i}'


# Solve problem
print(ILO_problem)
ILO_problem.solve()
print("Status:", pulp.LpStatus[ILO_problem.status])

for v in ILO_problem.variables():
    print(v.name,"=",v.varValue)

print("Optimized obj func =", pulp.value(ILO_problem.objective))
```

Fig 2: Screenshot of Python code (Exercise 3.1 B)

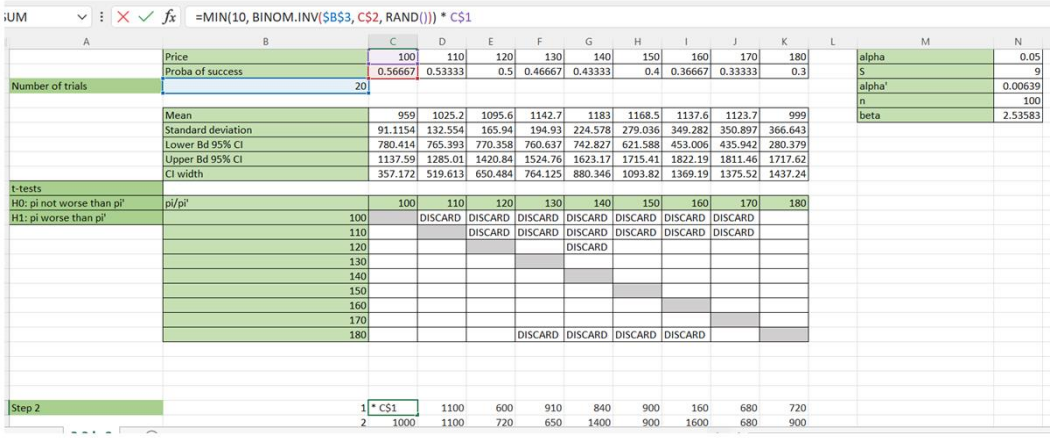


D12 =IF(C5>(D$5- N5 * SQRT(C6*C6+D$6*D$6)/SQRT(N4)), " ", "DISCARD")

	Price	100	110	120	130	140	150	160	170	180	alpha	0.05
	Proba of success	0.566666667	0.53333	0.5	0.46667	0.43333	0.4	0.36667	0.33333	0.3	S	9
Number of trials	20										alpha'	0.00639
											n	100000
	Mean	963.095	1035.33	1094.06	1135.85	1161.62	1169.7	1154.4	1124.76	1076.41	beta	2.48982
	Standard deviation	86.42335139	119.495	157.639	198.856	238.6	277.071	311.971	338.759	359.083		
	Lower Bd 95% CI	793.7052313	801.121	785.085	746.094	693.968	626.643	542.937	460.796	372.609		
	Upper Bd 95% CI	1132.484769	1269.54	1403.03	1525.61	1629.28	1712.76	1765.86	1788.73	1780.21		
	CI width	338.7795374	468.42	617.944	779.515	935.311	1086.12	1222.93	1327.94	1407.6		
t-tests												
H0: pi not worse than pi'	pi/pi'	100	110	120	130	140	150	160	170	180		
H1: pi worse than pi'	100		DISCARD	DISCARD	DISCARD	DISCARD	DISCARD	DISCARD	DISCARD	DISCARD		
	110			DISCARD	DISCARD	DISCARD	DISCARD	DISCARD	DISCARD	DISCARD		
	120				DISCARD	DISCARD	DISCARD	DISCARD	DISCARD			
	130					DISCARD	DISCARD	DISCARD				
	140						DISCARD					
	150											
	160					DISCARD	DISCARD					
	170				DISCARD	DISCARD	DISCARD	DISCARD				
	180			DISCARD	DISCARD	DISCARD	DISCARD	DISCARD	DISCARD			
Simulations	1	1000	1100	1200	1170	1400	1050	960	1360	1620		
	2	700	1100	1200	1300	840	1200	1600	1700	1080		

qu 3.3.a

Fig 3: Screenshot of Excel sheet (Exercise 3.3 A)



SUM =MIN(10, BINOM.INV(B3, C$2, RAND())) * C$1

	Price	100	110	120	130	140	150	160	170	180	alpha	0.05
	Proba of success	0.56667	0.53333	0.5	0.46667	0.43333	0.4	0.36667	0.33333	0.3	S	9
Number of trials	20										alpha'	0.00639
											n	100
	Mean	959	1025.2	1095.6	1142.7	1183	1168.5	1137.6	1123.7	999	beta	2.53583
	Standard deviation	91.1154	132.554	165.94	194.93	224.578	279.036	349.282	350.897	366.643		
	Lower Bd 95% CI	780.414	765.393	770.358	760.637	742.827	621.588	453.006	435.942	280.379		
	Upper Bd 95% CI	1137.59	1285.01	1420.84	1524.76	1623.17	1715.41	1822.19	1811.46	1717.62		
	CI width	357.172	519.613	650.484	764.125	880.346	1093.82	1369.19	1375.52	1437.24		
t-tests												
H0: pi not worse than pi'	pi/pi'	100	110	120	130	140	150	160	170	180		
H1: pi worse than pi'	100		DISCARD	DISCARD	DISCARD	DISCARD	DISCARD	DISCARD	DISCARD			
	110			DISCARD	DISCARD	DISCARD	DISCARD	DISCARD	DISCARD			
	120					DISCARD						
	130											
	140											
	150											
	160											
	170											
	180				DISCARD	DISCARD	DISCARD	DISCARD				
Step 2	1	* C$1	1100	600	910	840	900	160	680	720		
	2	1000	1100	720	650	1400	900	1600	680	900		

Fig 4: Screenshot of Excel sheet (Exercise 3.3 B Step 2)

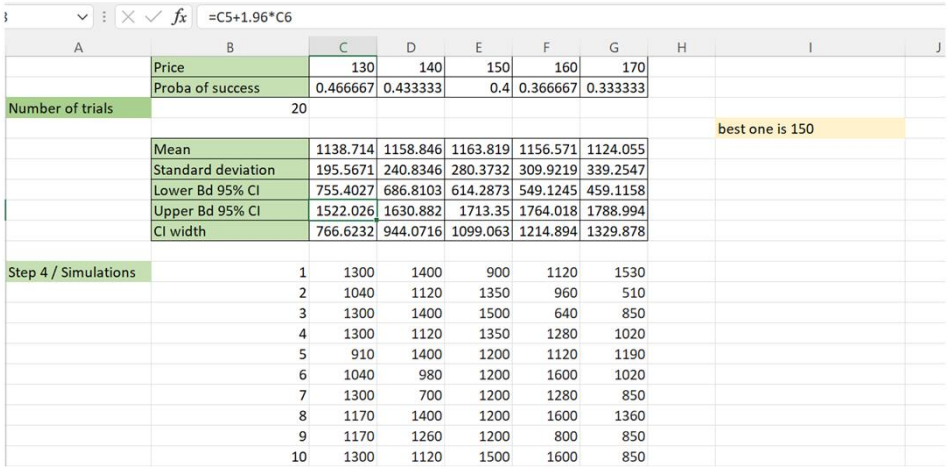


Fig 5: Screenshot of Excel sheet (Exercise 3.3 B Step 4)