Management Science Assignment 1

Angelo Barisano; 508903

November 16th, 2022

Table 1: Payoff table

	Revenues		Total Costs (Invest + Marginal)		Profit		Expected Profits
Type	Favourable	Unfavourable	Favourable	Unfavourable	Favourable	Unfavourable	Pf = 0.6 ; Puf = 0.4
Small	80000	48000	78000	47000	2000	1000	1600
Medium	80000	48000	69000	43000	11000	5000	8600
Large	80000	48000	64000	48000	16000	0	**9600**
No Invest	0	0	0	0	0	0	0
Probability	0.6	0.4	0.6	0.4	0.6	0.4	

1 Part 1: Decision Trees Risk Analysis

1.1 Question 1 A: Expected Payoff the Rational Investor

The findings for the expected value analysis can be found in table 1. This is based on considerations of Al regarding 1) revenue (80€ per sold pair of shoes * quantity sold during market condition), 2) the total costs (initial investment costs depending on the size of the operation + marginal costs depending on the size of the operation * quantity sold), and 3) the resulting profit (total revenue - total costs) per market phase. Based on the provided probabilities of the market phases - P(favorable = 0.6); P(unfavorable = 0.4) - an expected profit was calculated for each investment option (small, medium, large) weighted by the aforementioned probabilities. As can be observed in table 1, the expected profit is supposed to be higher for a large manufacturer, which draws its high potential from the favorable market condition yielding very high profits for this option when compared to the medium and small options.

As such, conditional on Al being a rational investor - expected return maximizing - and 1) given the fact that all expected profits are above 0, 2) that no profit falls below 0 (unfavourable + large manufacturing = 0€ profit), in addition to a large plant yielding the expected return of 9,600€ (vs 8,600€ medium; vs 1,600€ small), the conclusion is that Al should invest (so don't forego opportunity) and that the investment should be made for a large plant in this case which yields a very large profit for favourable market conditions, offsetting the 0€ return for unfavorable conditions.

Note assumptions: Al is a rational investor (maximizing expected payoff regardless of the fact that the large plant might lead to a 0€ return; most people would choice medium here as it pays decently in even the unfavourable case).

1.2 Question 1 B: Decision tree

Based on the assumption of Al being a rational investor (i.e. expected profit-maximizing), he should conduct the study in any case. This is because the path (the upper path in green) yields the largest expected profits of $100,74.4 (10.0744k). These expected profits are based on the potential outcomes for the respective expected profits resulting from the two possible outcomes for the study.

1. Study results in favor of good market conditions (up-most path): In this case, the high probability of favorable market conditions suggests that Al should invest in a Large Manufactory, which yields the highest expected outcome for the positive study results ($14020; 14.02k).

2. Study results do not favor good market conditions (path below): In this case, the choice that maximizes the expected profits is a Medium Manufactory, which in this case yields the highest payoff for the negative study results ($5800; 5.8k).

Figure 1: Decision Tree

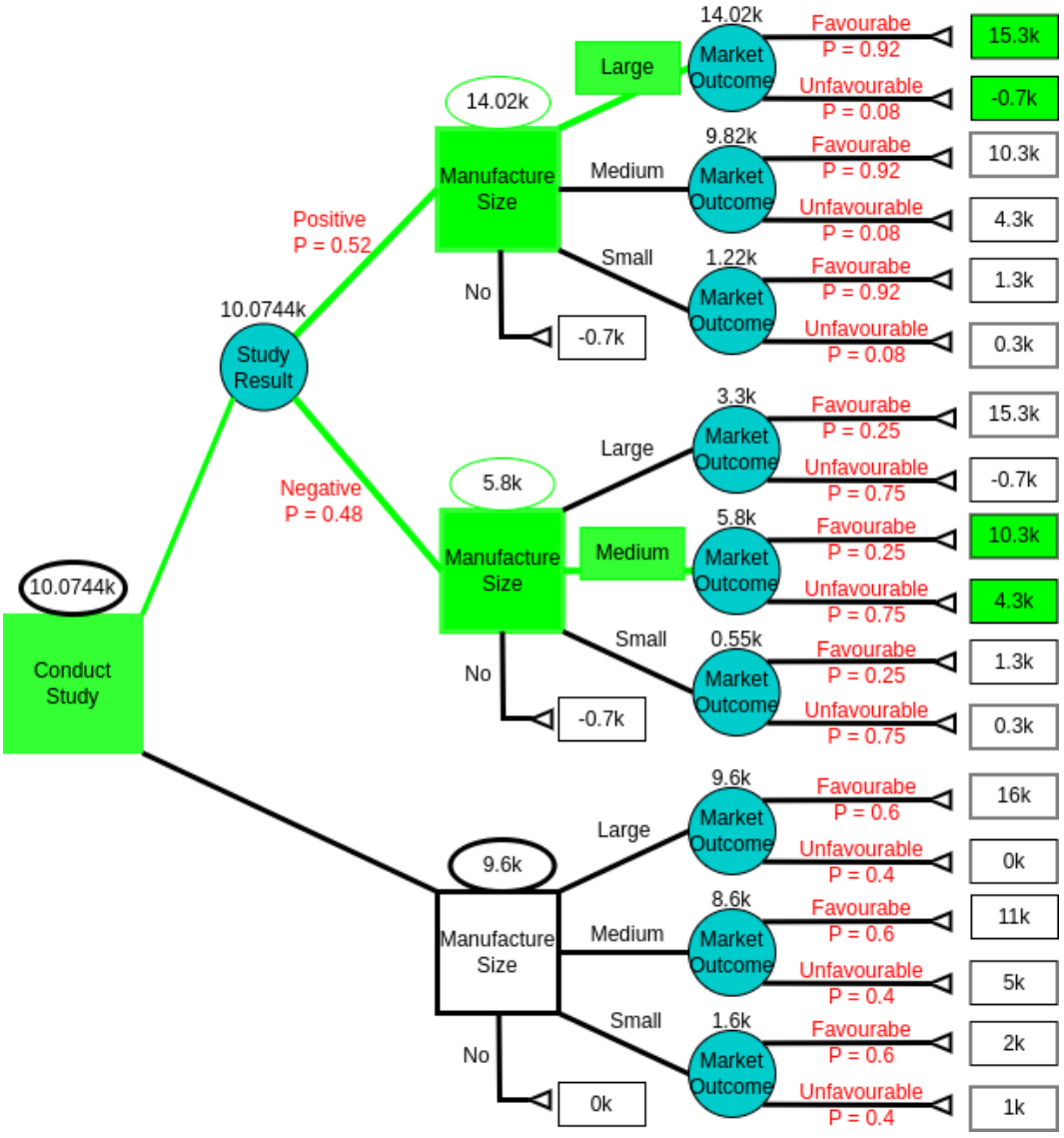


Table 2: Risk Profile

Positive Study Result	**Market Conditions**	**Profit**	**Probability**
(Choose Large)	Favourable	15.3	0.92
	Unfavourable	-0.7	0.08
		ExpValue =	**14.02**
Negative Study Result		**Profit**	**Probability**
(Choose Medium)	Favourable	10.3	0.25
	Unfavourable	4.3	0.75
		ExpValue =	**5.8**
Overall Results		**Profit**	**Probability**
(Choosing Study (default) + Study specific choice)	Favourable	15.3	0.4784
		-0.7	0.0384
	Unfavourable	10.3	0.12
		4.3	0.36
		ExpValue =	**10.0744**

1.3 Question 1 C: Risk Profile

Table 2 reports the risk profile for Al's investment. Given the analysis in Q1b), it was concluded that a study should be conducted in any case as this is the optimal path. Subsequently, table 2 is broken down into 3 parts; the up-most part corresponds to the up-most path in Figure 1 (conduct study + positive outcome + large manufactury). The second (middle part) corresponds to the conduct study, negative outcome, large plant path. The third part corresponds to the combined paths which also depicts the expected values overall. The probabilities in the 3rd part correspond to the multiplied probabilities from the other two table pieces. It may be noted that no Bayesian updating was performed.

Subsequently, even though the "highlighted" path in Figure 1 is the best choice from an expected profit perspective, it is also quite risky, considering the fact that this path contains the only terminal node with a negative payoff (-0.7k for large + study under positive study outcomes).

Overall, Al's probability to obtain the highest possible payoff of 10.0744k is 47.84%. This originates from the probability of a positive study result multiplied by favorable market outcomes (47.84% = 0.52 * 0.92). Thereafter, Al's probability to obtain a 4.3k payoff is 36% (= 0.48*0.75) under negative study results and unfavorable market conditions. Further, negative study results and favorable market conditions have a probability of 12% (= 0.48 * 0.25) for a payoff of 10.3k followed by positive study results and unfavorable market conditions with 4.16% probability and a payoff of -0.7k.

Consequently, while this is path contains the only negative possible payoff, the large upside provided by investments into large and medium manufacturers depending on the study results suggests that Al has a 95.84% to gain a reasonably large positive payoff.

2 Part 2: Oil, refineries, and distribution center

2.1 Question 2 A: mathematical breakdown of the problem

In order to solve the route optimization problem for Texago Corp., two approaches were considered which both yielded exactly the same outcomes; one where the routes are .

Firstly, a matrix with all possible paths was created, summing up the prices for transporting the oil. This approach simplifies the problem to 12 possible paths that exhaust all possibilities

Table 3: All possible paths in shipping costs between oil fields, refineries, and distribution

The Combinations of Refineries (N, C, P) and Distribution (P, A)	Oil Fields (T, CA)	
	California (Ca)	Texas (T)
New Orleans to Pittsburgh (NP)	18	22
Charlton to Pittsburgh (CP)	11	14
Seattle to Pittsburgh (SP)	13	7
New Orleans to Atlanta (NA)	14	18
Charlton to Atlanta (CA)	8	11
Seattle to Atlanta (SA)	11	5

- this can be seen in Table 3. The transport values in this matrix are the total cost of shipping one barrel from a given oilfield to a refinery and then to a distribution center; e.g. Texas to New Orleans costs $11 and from New Orleans to Atlanta the transport then costs another $11 summing to a total of $22 for this path. This has been done for all possible paths as can be observed in Table 3.

Based on this matrix, the quantity of oil shipment was expressed as x, which is the decision variable to drive the costs (constants), for each complete path as described above and in Table 3. Consequently, in x_{ij}, i represents the respective oil field in question (Texas or California), and the subscript j represents the combined path from any refinery to any distribution center. This expression could of course have been created as well in any other combination. Thus, the possible combinations were defined as:

Such that:

$$\forall\ x_{ij},\ i \in I, j \in J \text{ where } I = \{T, CA\} \text{ and } J = \{NA, CA, SA, NP, CP, SP\}$$

The value that has to be minimized here is the cost based on the demanded quantity at the distribution centers. As such, the values in Table 3 are the values based on which the objective function will be set up - the cost is represented by c, which is defined similarly as above as we are again talking about all possible paths exhaustively:

$$\forall\ c_{ij},\ i \in I, j \in J \text{ where } I = \{T, CA\} \text{ and } J = \{NA, CA, SA, NP, CP, SP\}$$

Based on this, the objective function to minimize total shipping costs for Texago is defined in the following way.

$$min \sum_{i \in I} \sum_{j \in J} (x_{ij} * c_{ij})$$

Which will be minimized (optimized in code) on the price c only as the demanded quantity (decision variable x) is fixed for this problem and we are only faced with the problem which route to chose. For simplicity, I also defined the objective function based on the price only for any path as in Table 3 below (this is not part of my mathematical definition of the problem; it just shows how I implemented it in Gurobi later on).

$$\min_{i,j}\ 22 * x_{TNP} + 14 * x_{TCP} + 7 * x_{TSP} + 18 * x_{TNA} + 11 * x_{TCA} + 5 * x_{TSA} + 18 * x_{CANP} + 11 * x_{CACP} + 13 * x_{CASP} + 14 * x_{CANA} + 8 * x_{CACP} + 11 * x_{CASP}$$

Regarding the demand limit:

- $\sum_{i \in I} x_{iNP} + x_{iCP} + x_{iSP} \ = \ 20,000$
- $\sum_{i \in I} x_{iNA} + x_{iCA} + x_{iSA} \ = \ 25,000$

Regarding the supply limit for each oil field respectively:

- $\sum_{j \in J} x_{Tj} \leq 10,000$
- $\sum_{j \in J} x_{CAj} \leq 50,000$

Finally, the quantity shipped can only be positive $0 \leq \forall x_{ij}$ while the cost must be positive as defined in Table 3 ($0 \leq \forall c_{ij}$) - this is somewhat redundant but it is included for completeness. Below you can find a written description of the constraints above as they were implemented in Gurobi:

1. Constraint: The supply limits of the oil fields (though not completely used) were included by selecting all paths that originate from the respective oil field (e.g. Texas) and limiting those paths collectively to a maximum supply (in this example 10000).
2. Constraint: The demand constraint is implemented similarly to the supply constraint; any path that for instance ends with Atlanta will have a combined demand satisfaction of 25000 barrels.
3. Constraint: There cannot be any negative values for the objective or negative flows overall.
4. (Implicit) Constraint: By constructing a matrix of all possible paths, inflows into refineries must yield the same outflows.

The implementation of the model is shown below (VS-Code editor; I used python 3.10.8; not anaconda python)

```python
combinations_oil_refineries_distributioncenters: Dict[tuple, int] = {
    ("T", "N", "P"): 11 + 11,
    ("T", "C", "P"): 7 + 7,
    ("T", "S", "P"): 2 + 5,
    ("T", "N", "A"): 11 + 7,
    ("T", "C", "A"): 7 + 4,
    ("T", "S", "A"): 2 + 3,
    ("CA", "N", "P"): 7 + 11,
    ("CA", "C", "P"): 4 + 7,
    ("CA", "S", "P"): 8 + 5,
    ("CA", "N", "A"): 7 + 7,
    ("CA", "C", "A"): 4 + 4,
    ("CA", "S", "A"): 8 + 3,
}

# quantity constraint

combinations_oil_refineries_distributioncenters_list = list(
    combinations_oil_refineries_distributioncenters.keys()
)


oil_fields: List[str] = ["T", "CA"]
refineries: List[str] = ["NO", "C", "S"]
distribution_centers: List[str] = ["P", "A"]


# oil_field_limits
oil_field_limits: Dict[str, int] = {"CA": 50000, "T": 10000}

# demand limits
demand_limits: Dict[str, int] = {"P": 20000, "A": 25000}
```

```python
demand_limits: Dict[str, int] = {"P": 20000, "A": 25000}


# set the decision function
model = gp.Model()
model.ModelSense = gp.GRB.MINIMIZE

# set the objective function
obj = {}
for combi in combinations_oil_refineries_distributioncenters_list:
    obj[combi] = model.addVar(
        lb=0,
        obj=combinations_oil_refineries_distributioncenters[combi],
        name="{}".format(str(combi)).replace(" ", ""),
    )

# CONSTRAINT FOR SUPPLY: essenetially: for every combintion with T at beginning; this
for oil_field in oil_field_limits.keys():
    lhs = 0
    for objective_r in obj.keys():
        if objective_r[0] == oil_field:
            lhs += obj[objective_r]
    model.addConstr(lhs <= oil_field_limits[oil_field])

# Constraint for demand
for distro in demand_limits.keys():
    lhs = 0
    for objective_r in obj.keys():
        if objective_r[-1] == distro:
            lhs += obj[objective_r]
    model.addConstr(lhs == demand_limits[distro])

model.optimize()
if not model.status == gp.GRB.OPTIMAL:
    print("something went wrong")
print("optimal value", model.objval)
model.printAttr("X")
```

2.2 Question 2 B: implementation

The screenshot below shows the model output. The most optimal value in terms of the lowest possible transportation cost for Texaco Corp. is $380,000 for a total of 45,000 barrels, which can be seen under the optimal value - given the constraints specified beforehand.

Overall, this model optimizes on only three of 12 possible paths. In terms of flows from oil fields to refineries, Texas supplies 10000 barrels to Seattle refineries, which is then further transported to Pittsburgh. The California oilfield supplies only Charleston with a total of 35,000 barrels. This is then further transported to Pittsburgh (10,000) and Atlanta (25,000), thereby fulfilling demand for a total of $380,000 (optimal value).

The practical part of this implementation is that any inflow into one path is automatically how much was supplied from an oilfield and to the final distribution center in one picture.

```python
model.optimize()
if not model.status == gp.GRB.OPTIMAL:
    print("something went wrong")
print("optimal value", model.objval)
model.printAttr("X")
```

✓ 0.4s

```
Gurobi Optimizer version 9.5.2 build v9.5.2rc0 (linux64)
Thread count: 4 physical cores, 8 logical processors, using up to 8 threads
Optimize a model with 4 rows, 12 columns and 24 nonzeros
Model fingerprint: 0x74384aec
Coefficient statistics:
  Matrix range     [1e+00, 1e+00]
  Objective range  [5e+00, 2e+01]
  Bounds range     [0e+00, 0e+00]
  RHS range        [1e+04, 5e+04]
Presolve removed 4 rows and 12 columns
Presolve time: 0.00s
Presolve: All rows and columns removed
Iteration    Objective       Primal Inf.    Dual Inf.      Time
       0    3.8000000e+05   0.000000e+00   0.000000e+00      0s

Solved in 0 iterations and 0.00 seconds (0.00 work units)
Optimal objective  3.800000000e+05
optimal value 380000.0

    Variable            X
------------------------
('T','S','P')        10000
('CA','C','P')        10000
('CA','C','A')        25000
```

2.3 Question 2 C: Sensitivity

```
print("Sensitivity Analysis:")
model.printAttr(["X", "Obj", "SAObjLow", "SAObjUP"])
model.printAttr(["RC", "LB", "SALBLow", "SALBUp", "UB", "SAUBLow", "SAUBUp"])
model.printAttr(["Sense", "Slack", "Pi", "RHS", "SARHSLow", "SARHSUp"])
```

[3] ✓ 0.6s

```
... Output exceeds the size limit. Open the full output data in a text editor
Sensitivity Analysis:
```

```
Sensitivity Analysis:

    Variable            X          Obj     SAObjLow      SAObjUP
-------------------------------------------------------------------
('T','N','P')           0           22            7          inf
('T','C','P')           0           14            7          inf
('T','S','P')       10000            7         -inf            8
('T','N','A')           0           18            4          inf
('T','C','A')           0           11            4          inf
('T','S','A')           0            5            4          inf
('CA','N','P')          0           18           11          inf
('CA','C','P')      10000           11           10           13
('CA','S','P')          0           13           11          inf
('CA','N','A')          0           14            8          inf
('CA','C','A')      25000            8         -inf            9
('CA','S','A')          0           11            8          inf

    Variable           RC           LB      SALBLow       SALBUp          UB      SAUBLow       SAUBUp
-----------------------------------------------------------------------------------------------------------
('T','N','P')          15            0         -inf        10000         inf            0          inf
('T','C','P')           7            0         -inf        10000         inf            0          inf
('T','S','P')           0            0         -inf        10000         inf        10000          inf
('T','N','A')          14            0       -10000        10000         inf            0          inf
('T','C','A')           7            0       -10000        10000         inf            0          inf
('T','S','A')           1            0       -10000        10000         inf            0          inf
('CA','N','P')          7            0         -inf        10000         inf            0          inf
('CA','C','P')          0            0         -inf        10000         inf        10000          inf
('CA','S','P')          2            0         -inf        10000         inf            0          inf
('CA','N','A')          6            0         -inf        25000         inf            0          inf
('CA','C','A')          0            0         -inf        25000         inf        25000          inf
('CA','S','A')          3            0         -inf        25000         inf            0          inf

  Constraint       Sense        Slack           Pi          RHS     SARHSLow      SARHSUp
-----------------------------------------------------------------------------------------
oil_limit_CA           <        15000            0        50000        35000          inf
 oil_limit_T           <            0           -4        10000            0        20000
    demand_P           =            0           11        20000        10000        35000
    demand_A           =            0            8        25000            0        40000
```

2.3.1 Q2 C i)

An increase in supply in Texas by 5,000 to 15,000 would lead to a total reduction of $20,000 in transportation costs; consequently, the new optimal would be at $360,000 instead of the original $380,000. This can be concluded by observing the constraint oil_limit_T, in the lowest table which

specifies that supply from Texas is a binding constraint, based on the negative shadow price of -4 and the corresponding ranges to this constraint of (0; 20,000) where the shadow price will not change and as such a conclusion can be drawn. Consequently, the reduction in transportation cost by $20,000 originates from the -4 * 5,000 barrels = -20,000, which is in the case of this minimization problem simply the net reduction in cost. Thus, yes the optimal solution changes to $360,000.

2.3.2 Q2 C ii)

A reduction in supply for California would not have an effect for the specifications given in this model. This is because the California supply constraint is not binding. This can be observed in the lowest table of the sensitivity analysis, where the oil_limit_S displays a slack of 20,000. Subsequently, the shadow price is 0 and the shadow price can be assumed to stay 0 in the ranges from 30,000 to 50,000 barrels. The reduction of supply to 40,000 would thus fall into the limits of boundaries calculated for this model to hold. As such, the optimal value would not change as the California supply limit would continue being a non-binding constraint.

2.3.3 Q2 C iii)

In case of having to deliver either 10,000 barrels to Atlanta or Pittsburgh, the model suggests that Atlanta would be the better option. This is again based on the shadow price. As this is a minimization problem, we want to select the lowest shadow price for this problem. Subsequently, Pittsburgh has a shadow price (Pi) of 11 while Atlanta only has 8. Thus, if Texaco were to deliver 10,000 additional barrels, this would only be to Atlanta, leading to an increase of the optimal value (transportation cost) of $80,000 to a total of $460,000. It may be noted, that there is no point in splitting this delivery simply because the range in which the model still provides reliable results ranges from 0 to 30000 (SARHSLow; SARHSUp); thus, an increase by 10,000 would not change the model and all constraints still hold.

2.3.4 Q2 C iiii)

If the transportation cost per barrel from California to Seattle would decrease from $8 to $7, the impact could be analyzed with the first table in the sensitivity analysis. There are two paths that go from California to Seattle - one with Pittsburgh at its end (objective value 13) and one with Atlanta at its final destination (objective value 11). Consequently, the total costs of transporting one barrel would decrease the total transport cost for both paths by $1 per barrel:

1. California, Seattle, Pittsburgh = $12; (SAObjLow, SAObjUp) - (11, infinity)
2. California, Seattle, Atlanta = $10;(SAObjLow, SAObjUp) - (8, infinity)

Consequently, in order for the reduction in cost to have an impact the total cost of each path must fall below the lower bound as can be seen above. Both, the reduction to 12 and 10 for either path does not reach below the required 11 and 10 to have an impact on the oil flows.

As such, the optimal value does not change and, thus, the transportation cost also do not change.

3 Part 3

3.1 Question 3 A: mathematical breakdown of the problem

The variable descriptions were copied from the assignment sheet as follows:

- P_a and P_b are the number of sails that are purchased for store A and B
- E^t_{ab} and E^t_{ba} at time t (d in the assignment) are the expensive one day transfers from a to b and b to a respectively
- C^t_{ab} and C^t_{ba} at time t (d in the assignment) are the inexpensive two day transfers from a to b and b to a respectively
- I_a and I_b at time t (d in the assignment) is the inventory at A and B respectively
- s^t_a and s^t_b at time t (d in the assignment) is the demand at store A and B respectively for

Based on these variables, the objective variable was the demand, which was driving the costs in the objective function. Consequently, the objective function was defined as:

$$min\ 200 * (P_a + P_b) + 20 * \sum_{t \in T} (E^t_{ab} + E^t_{ba}) + 5 * \sum_{t \in T} (C^t_{ab} + C^t_{ba})$$

In order to solve this problem, the inventory constraint in both A and B had to be specified. This is because there is to be no decay or loss in sails. Subsequently, based on the aforementioned possibilities of 1) purchasing on day one, 2) expensive one day transfer, and 3) inexpensive two day transfer, the inventory management constraint was defined in the following way:

1. $I^0_a = P_a$ and $I^0_b = P_b$ define the inventory on day one of the operations; this is the only day on which any sails can be purchased an no flows between A and B can happened.
2. $I^1_a = I^0_a - E^0_{ab} + E^0_{ba} - C^0_{ab}$ and $I^1_b = I^0_b - E^0_{ba} + E^0_{ab} - C^0_{ba}$ define the inventory on day two (index 1 in the python implementation) where the inventory is defined by the previous day's inventory plus previous inflows from an expensive transfer minus the inexpensive transfers for in "two days" to the other inventory.
3. $I^t_a = I^{t-1}_a - E^{t-1}_{ab} + E^{t-1}_{ba} - C^{t-1}_{ab} + C^{t-2}_{ba}$ and $I^t_b = I^{t-1}_b - E^{t-1}_{ba} + E^{t-1}_{ab} - C^{t-1}_{ba} + C^{t-2}_{ab}$ this defines the ongoing concern of the inventory.
4. $I^6_a = I^5_a + E^5_{ba} + C^4_{ba}$ and $I^6_b = I^5_b + E^5_{ab} + C^4_{ab}$ which defines the inventory for the 6th (final) day (using python index 0 as the base of this set - see set definition below).

Continuing from the inventory constraint, the demand will be implemented. The demand controls the objective function and the total sails needed. Additionally, constrains regarding the transfers and purchases must be made. Subsequently:

1. $I^t_a \geq s^t_a$ and $I^t_b \geq s^t_b$ where $t \in T$ - meaning that the inventory levels must be at least as big as the demand at time t.
2. $P_a \geq 0$, $P_b \geq 0$ - the purchased amount on day one must be greater equal than 0.
3. $max(s^t_a + s^t_b) \geq (P_a + P_b)$ - the total amount purchased on day one must be greater or equal to the maximum of demand on a given day.
4. $I^t_a \geq 0$, $I^t_b \geq 0$ - the inventory of a or b must be greater equal 0.
5. $E^t_{ab} \geq 0$, $E^t_{ba} \geq 0$ - there cant be any "negative" transfers".

6. $C_{ab}^t \geq 0$, $C_{ba}^t \geq 0$ - there cant be any "negative" transfers".

All these constraints and functions are dependent on the following set of values $J = \{A, B\}$ and $I = \{B, A\}$ to represent the flows - for both outlets:

1. $\forall\ S_j^t, j \in J, t \in T$
2. $\forall\ P_j, j \in J$
3. $\forall\ E_{ij}^t, i \in I, j \in J, t \in T$
4. $\forall\ C_{ij}^t, i \in I, j \in J, t \in T$
5. $\forall\ I_j^t, j \in J, t \in T$
6. $T = \{0, 1, 2, 3, 4, 5, 6\}$

The implementation into gurobi can be seen in the screenshots below

```
stores = ["A", "B"]

demand = {
    ("A", 0): 45,
    ("A", 1): 20,
    ("A", 2): 20,
    ("A", 3): 25,
    ("A", 4): 15,
    ("A", 5): 28,
    ("A", 6): 15,
    ("B", 0): 8,
    ("B", 1): 12,
    ("B", 2): 23,
    ("B", 3): 30,
    ("B", 4): 12,
    ("B", 5): 10,
    ("B", 6): 33,
}

days = 7

model = gp.Model()
model.ModelSense = gp.GRB.MINIMIZE

# purchasing day one
Purc_A = model.addVars(days, name="Purc_A", obj=200)
Purc_B = model.addVars(days, name="Purc_B", obj=200)

# "C allocation (5)" from A to B and B to A
cab = model.addVars(days, name="Cab", obj=5)
cba = model.addVars(days, name="Cba", obj=5)

# "E allocation (20)" from A to B and B to A
eab = model.addVars(days, name="Eab", obj=20)
eba = model.addVars(days, name="Eba", obj=20)

# set the function
model.setObjective(
    gp.quicksum(
        ((Purc_A[t] + Purc_B[t]) * 200 + (eab[t] + eba[t]) * 20 + (cab[t] + cba[t]) * 5)
        for t in range(days)
    )
)

Inventory_A = model.addVars(days, name="Inventory_A")
Inventory_B = model.addVars(days, name="Inventory_B")
```

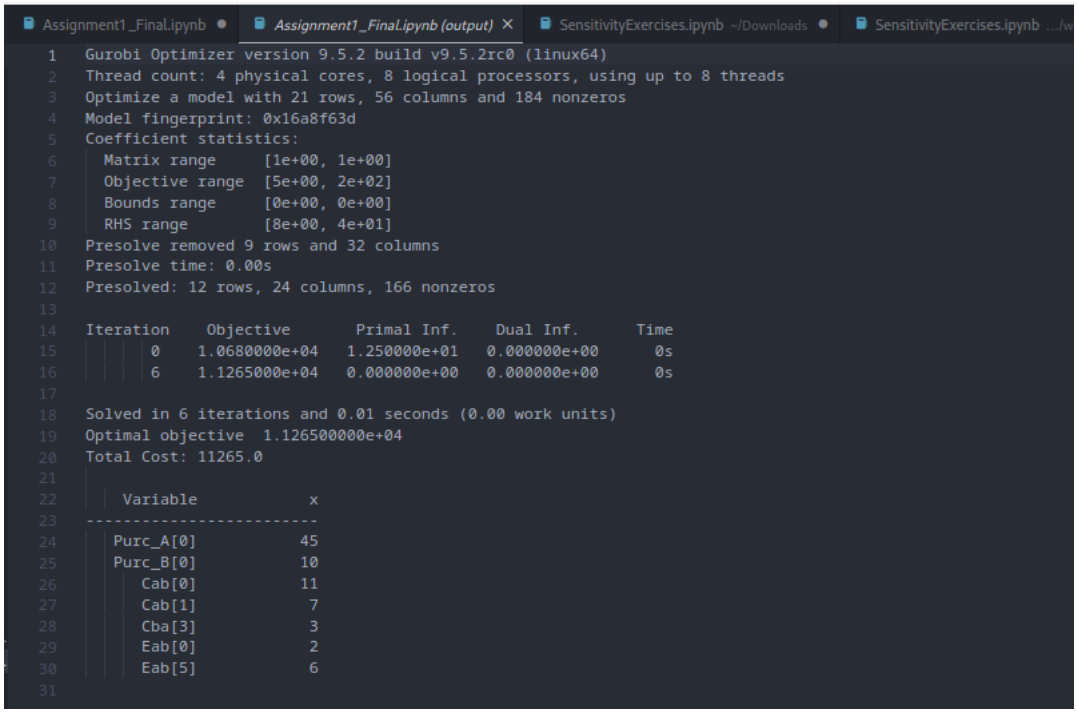
```python
## inventory management A
# day 1 purchase only; index 0
Inventory_A[0] = Purc_A[0]
# day 2 no purcahse but E allocation: inventory day 2 = Inventory day 1 (t - 1 = 0) + transfers from b at
# day t - 1 - minus all transfers from a to b and from a to b in the 5 € concerns
Inventory_A[1] = Inventory_A[0] + eba[0] - eab[0] - cab[0]
# day 3 and beyond; is essentially the same as above but now you can also receive
for t in range(2, days):
    Inventory_A[t] = (
        Inventory_A[t - 1] + eba[t - 1] + cba[t - 2] - eab[t - 1] - cab[t - 1]
    )

## the same for B just the other way around!
Inventory_B[0] = Purc_B[0]
Inventory_B[1] = Inventory_B[0] + eab[0] - eba[0] - cba[0]
for t in range(2, days):
    Inventory_B[t] = (
        Inventory_B[t - 1] + eab[t - 1] + cab[t - 2] - eba[t - 1] - cba[t - 1]
    )


# enforce that inventory is at least as big as demand on date t for both inventory a and b
for t in range(days):
    model.addConstr(Inventory_A[t] >= demand[("A", t)], name="Inventory_A")

for t in range(days):
    model.addConstr(Inventory_B[t] >= demand[("B", t)], name="Inventory_B")


# only day one can be used for purchasesing
for t in range(1, days):
    lhs = Purc_A[t] + Purc_B[t]
    model.addConstr(lhs == 0)

# only the last day does not work in this case as there cant be two day transfers (the C transfers)
model.addConstr((cab[days - 1] + cba[days - 1]) == 0)


model.optimize()

print("Total Cost:", model.OBJVAL)
model.printAttr(["x"])
```

3.2 Question 3 B: Optimal Cost composition

Please note, that I have to export the outputs of the model into the Vs Code text editor. As can be seen in the screenshots below, the optimal (minimal cost) polity yields a total cost of $11,265 for purchasing and reallocation. In this context, purchasing is the largest part of the cost with 55 purchases resulting in $11,000 of the cost (10 at B and 45 at A). Following, there were in total 21 cheap transfers (total = $105) and 8 expensive transfers (total = $160). The cheap transfers were split 11 and 7 from A to B on day one and day two respectively, and a 3 return from B to A on day 3. Following, Expensive transfers were made from A to B on the first and the sixth day (python has zero index) for 2 and 6 respectively.

As such, the makeup of the total costs is 97.65% by the purchasing of the sails, 1.42% via expensive shipping, and 0.93% for the inexpensive shipping.

This is also in line when conducting a sensitivity analysis which yields the inventory levels always being at or above demand, which I used to control whether the model performed as I intended it to (sensitivity analysis not shown for this).



3.3 Question 3 C: One year of perfect demand forecast

The screenshots are shown below. The model definition in Q3A) was generalized in the following way:

1. $D = \{0, 1, 2, ..., 364\}$ - (zero index)
2. $I_a^{364} = I_a^{363} + E_{ba}^{363} + C_{ba}^{362}$ and $I_b^{364} = I_b^{363} + E_{ab}^{363} + C_{ab}^{362}$ which defines the inventory for the 6th (final) day (using python index 0 as the base of this set - see set definition below).

In total the optimal value (minimized) is \$22,850.0 . In total on day one A purchased 58 and B 49 sails. It is assumed that we can only buy on day one again as the exercise was not more precise on this. AS such, 107 * 200 = 21,400; as such, again the majority of the costs are attributed to the purchasing of the sails.

this is constrained by the maximum demand on a given day during the year. Subsequently, when observing the maximum demand of a specific day it is 69 (A: 28 and B 41) on day 65 (python index 64). This is interesting as in the previous exercise based on one week, the model tried to purchase as few as max demand during the week, while in this model over 365, purchasing more sails than max demand becomes more feasible.

The reason for this is simple: in a 7 day model, the costs of moving the sails is considerably lower than buying a new sail. However, in a 365 (or higher) day model, the movement costs are the only costs that contribute to the costs as there is no decay or loss in sails. As such, we would expect the amount of purchased sails to increase with longer demand forecasts while the movements proportionally to a 7 days model decreases; the model will just start to increase the inventory at A and B without moving them in-between anymore.

This is also reflected by the choices that the model made regarding the transfers between A and B. In total there were 10 expensive transfers from a to b throughout the year for \$200,

while there were only 9 expensive transfers from b to a accounting for $180. As such, only 19 expensive transfers were made, accounting for $380.

Regarding the cheap transfers, as expected, they are used more frequently as we use perfect demand forecasts.

In total a moved 104 sails from A to B, accounting for $520, and 110 sails moved from B to A, accounting for $550; for a total of 214 sails moved and $1,070.

$$min\ 200 * (P_a + P_b) + 20 * \sum_{t \in T} (E^t_{ab} + E^t_{ba}) + 5 * \sum_{t \in T} (C^t_{ab} + C^t_{ba})$$

```
df = pd.read_csv(
    "/home/angelo/Documents/Uni/Courses/Management Science/Management_Science_Code/
)
```

✓ 0.3s

```
dict_A = {}
dict_B = {}
for index, row in df.iterrows():
    dict_A[("A", row["day"] - 1)] = row["demandA"]

for index, row in df.iterrows():
    dict_B[("B", row["day"] - 1)] = row["demandB"]

dictionary_ab = dict_A | dict_B
```

✓ 0.5s

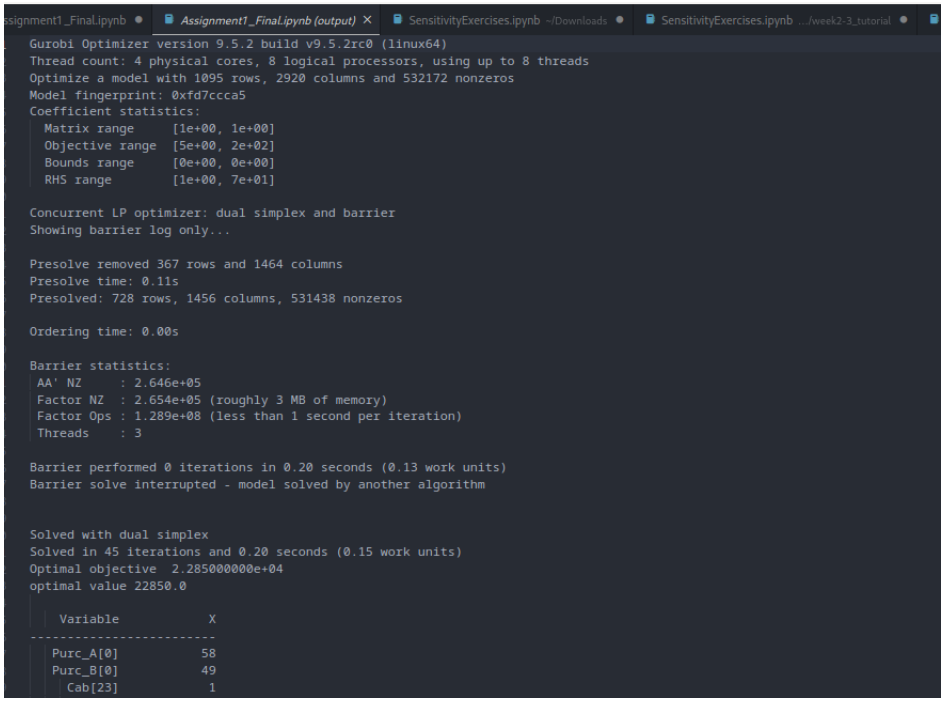
```
stores = ["A", "B"]
demand = dictionary_ab
days = 365
```

✓ 0.3s

```
model = gp.Model()
```

```
model.ModelSense = gp.GRB.MINIMIZE

# purchasing day one
Purc_A = model.addVars(days, name="Purc_A", obj=200)
Purc_B = model.addVars(days, name="Purc_B", obj=200)

# "C allocation (5)" from A to B and B to A
cab = model.addVars(days, name="Cab", obj=5)
cba = model.addVars(days, name="Cba", obj=5)

# "E allocation (20)" from A to B and B to A
eab = model.addVars(days, name="Eab", obj=20)
eba = model.addVars(days, name="Eba", obj=20)

# set the function
model.setObjective(
    gp.quicksum(
        ((Purc_A[t] + Purc_B[t]) * 200 + (eab[t] + eba[t]) * 20 + (cab[t] + cba[t])
        for t in range(days)
    )
)


Inventory_A = model.addVars(days, name="Inventory_A")
Inventory_B = model.addVars(days, name="Inventory_B")

## inventory management A
# day 1 purchase only; index 0
Inventory_A[0] = Purc_A[0]
```

```
    )

## the same for B just the other way around!
Inventory_B[0] = Purc_B[0]
Inventory_B[1] = Inventory_B[0] + eab[0] - eba[0] - cba[0]
for t in range(2, days):
    Inventory_B[t] = (
        Inventory_B[t - 1] + eab[t - 1] + cab[t - 2] - eba[t - 1] - cba[t - 1]
    )


# enforce that inventory is at least as big as demand on date t for both inventory
for t in range(days):
    model.addConstr(Inventory_A[t] >= demand[("A", t)], name="Inventory_A")

for t in range(days):
    model.addConstr(Inventory_B[t] >= demand[("B", t)], name="Inventory_B")

# only day one can be used for purchasesing
for t in range(1, days):
    lhs = Purc_A[t] + Purc_B[t]
    model.addConstr(lhs == 0)

# only the last day does not work in this case as there cant be two day transfers (
model.addConstr((cab[days - 1] + cba[days - 1]) == 0)


model.optimize()
```



```
ssignment1_Final.ipynb ●  Assignment1_Final.ipynb (output) ×  SensitivityExercises.ipynb ~/Downloads ●  SensitivityExercises.ipynb .../week2-3_tutorial ●

Gurobi Optimizer version 9.5.2 build v9.5.2rc0 (linux64)
Thread count: 4 physical cores, 8 logical processors, using up to 8 threads
Optimize a model with 1095 rows, 2920 columns and 532172 nonzeros
Model fingerprint: 0xfd7ccca5
Coefficient statistics:
  Matrix range     [1e+00, 1e+00]
  Objective range  [5e+00, 2e+02]
  Bounds range     [0e+00, 0e+00]
  RHS range        [1e+00, 7e+01]

Concurrent LP optimizer: dual simplex and barrier
Showing barrier log only...

Presolve removed 367 rows and 1464 columns
Presolve time: 0.11s
Presolved: 728 rows, 1456 columns, 531438 nonzeros

Ordering time: 0.00s

Barrier statistics:
 AA' NZ     : 2.646e+05
 Factor NZ  : 2.654e+05 (roughly 3 MB of memory)
 Factor Ops : 1.289e+08 (less than 1 second per iteration)
 Threads    : 3

Barrier performed 0 iterations in 0.20 seconds (0.13 work units)
Barrier solve interrupted - model solved by another algorithm


Solved with dual simplex
Solved in 45 iterations and 0.20 seconds (0.15 work units)
Optimal objective  2.285000000e+04
optimal value 22850.0

    Variable            X
-------------------------
   Purc_A[0]           58
   Purc_B[0]           49
     Cab[23]            1
```

Variable	X
Purc_A[0]	58
Purc_B[0]	49
Cab[23]	1
Cab[26]	5
Cab[32]	1
Cab[43]	10
Cab[62]	3
Cab[72]	6
Cab[81]	3
Cab[94]	4
Cab[108]	11
Cab[150]	3
Cab[173]	6
Cab[200]	15
Cab[237]	12
Cab[247]	7
Cab[256]	2
Cab[265]	1
Cab[293]	8
Cab[329]	6
Cba[50]	9
Cba[54]	12
Cba[86]	8
Cba[96]	2
Cba[117]	8
Cba[163]	13
Cba[222]	3
Cba[233]	13
Cba[242]	3
Cba[243]	5
Cba[252]	8
Cba[304]	5
Cba[305]	2
Cba[350]	4
Cba[360]	15
Eab[73]	10
Eba[74]	5
Eba[110]	4