# DSA/ISE 5113
# Advanced Analytics and Metaheuristics
# Homework #5

**March 23$^{rd}$, 2023**

**Alex Nozka**

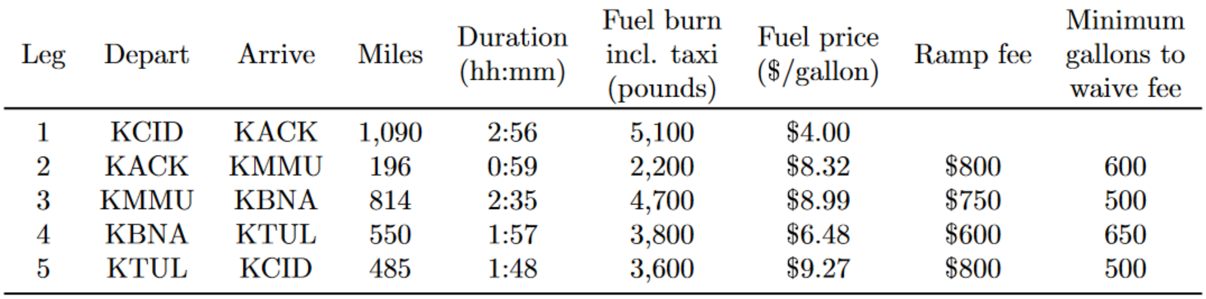
**Marshall Baldwin**

## Question 1: Boomer Sooner Air Services

**(a) Formulate a model and solve for a minimum cost fuel plan for the upcoming trip. What is the optimal fueling plan and minimal cost? Compare your results with a no “tankering” solution.**

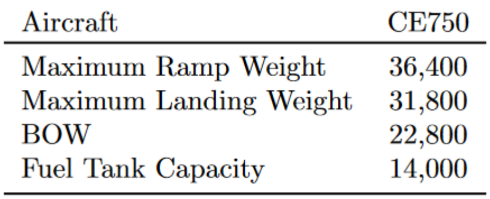
Let’s first examine the “no-tankering” solution. This involves buying only as much fuel as will reach the destination with 2500 pounds remaining in the tank and then refilling to 7000 pounds after the last flight. That is, buying 600 pounds to start, then purchasing the flight burn amount (see the table below) for the remaining legs and then purchasing 4500 pounds after the trip to refill. These purchases total $20,739.25.

Table 1: Flight Details



| Leg | Depart | Arrive | Miles | Duration (hh:mm) | Fuel burn incl. taxi (pounds) | Fuel price ($/gallon) | Ramp fee | Minimum gallons to waive fee |
|---|---|---|---|---|---|---|---|---|
| 1 | KCID | KACK | 1,090 | 2:56 | 5,100 | $4.00 | | |
| 2 | KACK | KMMU | 196 | 0:59 | 2,200 | $8.32 | $800 | 600 |
| 3 | KMMU | KBNA | 814 | 2:35 | 4,700 | $8.99 | $750 | 500 |
| 4 | KBNA | KTUL | 550 | 1:57 | 3,800 | $6.48 | $600 | 650 |
| 5 | KTUL | KCID | 485 | 1:48 | 3,600 | $9.27 | $800 | 500 |

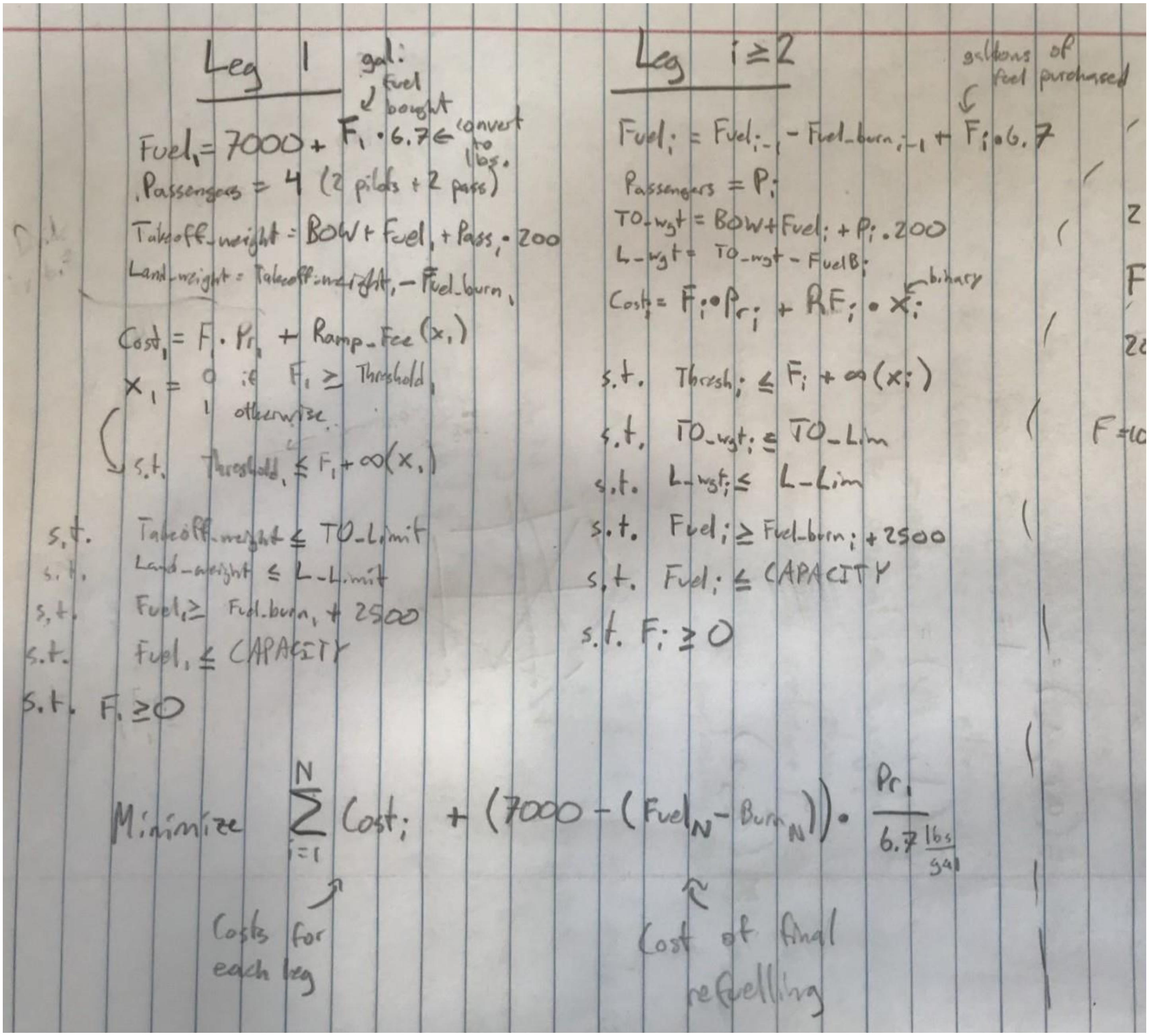
Table 2: Aircraft Limitations (in pounds)



| Aircraft | CE750 |
|---|---|
| Maximum Ramp Weight | 36,400 |
| Maximum Landing Weight | 31,800 |
| BOW | 22,800 |
| Fuel Tank Capacity | 14,000 |

For formulating the model, first note that we wish to minimize total cost, given by the cost of purchased fuel and airport ramp fees. The aircraft also has certain physical restraints which must be followed: takeoff and landing weight capacities as well as a maximum fuel capacity. I also assume that it is not possible to sell fuel after landing and that there are two pilots aboard the plane.

The model can then be formulated as follows:



Leg 1

gal: fuel bought

$Fuel_1 = 7000 + F_1 \cdot 6.7$ ← convert to lbs.

$Passengers = 4$ (2 pilots + 2 pass)

$Takeoff\_weight = BOW + Fuel_1 + Pass_1 \cdot 200$

$Land\_weight = Takeoff\_weight_1 - Fuel\_burn_1$

$Cost_1 = F_1 \cdot Pr_1 + Ramp\_Fee(x_1)$

$x_1 = 0$ if $F_1 \geq Threshold_1$
$\phantom{x_1 =}$ 1 otherwise.

s.t. $Threshold_1 \leq F_1 + \infty(x_1)$

s.t. $Takeoff\_weight \leq TO\_Limit$

s.t. $Land\_weight \leq L\_Limit$

s.t. $Fuel_1 \geq Fuel\_burn_1 + 2500$

s.t. $Fuel_1 \leq CAPACITY$

s.t. $F_1 \geq 0$

Leg $i \geq 2$

gallons of fuel purchased

$Fuel_i = Fuel_{i-1} - Fuel\_burn_{i-1} + F_i \cdot 6.7$

$Passengers = P_i$

$TO\_wgt = BOW + Fuel_i + P_i \cdot 200$

$L\_wgt = TO\_wgt - FuelB_i$

$Cost_i = F_i \cdot Pr_i + RF_i \cdot x_i$ ← binary

s.t. $Thresh_i \leq F_i + \infty(x_i)$

s.t. $TO\_wgt_i \leq TO\_Lim$

s.t. $L\_wgt_i \leq L\_Lim$

s.t. $Fuel_i \geq Fuel\_burn_i + 2500$

s.t. $Fuel_i \leq CAPACITY$

s.t. $F_i \geq 0$

$$\text{Minimize} \ \underbrace{\sum_{i=1}^{N} Cost_i}_{\text{Costs for each leg}} + \underbrace{(7000 - (Fuel_N - Burn_N)) \cdot \frac{Pr_1}{6.7 \frac{lbs}{gal}}}_{\text{Cost of final refuelling}}$$

There are a few features of the model to note. First is that the fuel variable at each leg of the journey is the fuel prior to taxi and takeoff. There are constraints respecting the physical limitations of the plane with regards to weight and fuel capacity. Note the inclusion of a binary variable x which allows the ramp fee to be waived for that leg of the trip if enough gallons of fuel are purchased. The full AMPL model is shown on the next page.

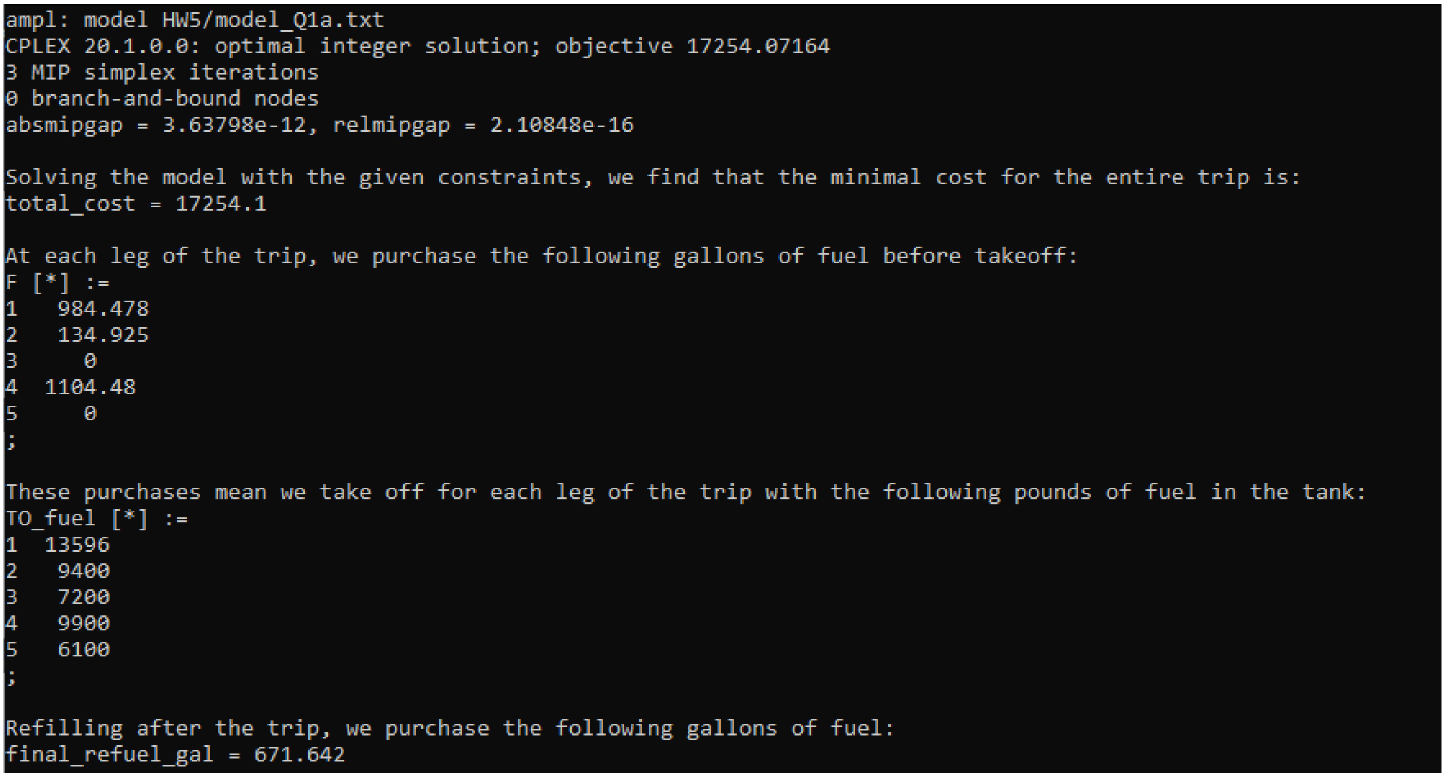
```
#clear memory
reset;

#choose solver;
option solver cplex;

#create sets to define variables within
set Legs = {1,2,3,4,5}; #legs of flight plan

#initialize non-variable parameters
param default_fuel; #pounds of fuel in aircraft by default (must be restored after trip)
param fuel_buffer; #pounds of extra fuel the aircraft must land with
param fuel_lpg; #pounds of fuel contained in one gallon
param f_cap; #maximum fuel (in pounds) the aircraft can carry
param TO_Lim; #upper limit in pounds allowed for takeoff
param L_Lim; #upper limit in pounds allowed for landing
param p_wgt; #weight (in pounds) of a passenger
param BOW; #basic operating weight of aircraft without fuel or passengers

param Ppl {i in Legs}; #number of people on plane during each leg
param Pr {i in Legs}; #price of fuelling before takeoff in dollars per gallon
param RF {i in Legs}; #ramp fee for taking off
param RF_thresh {i in Legs}; #minimum purchased fuel (in gallons) to waive ramp fee
param FB {i in Legs}; #pounds of fuel burned during flight

param huge_num = 1000000; #large number to use for integer programming constraint tricks

#load parameter data
data HW5/data_Q1a.txt;

#initialize variables
var F {i in Legs}; #gallons of fuel purchased before takeoff
var TO_fuel {i in Legs}; #pounds of fuel before takeoff
var TO_wgt {i in Legs}; #the weight of the aircraft (in pounds) at takeoff
var L_wgt {i in Legs}; #the weight of the aircraft (in pounds) when landing
var x {i in Legs} binary; #whether or not the ramp fee gets waived (0=waived, 1=paid)
var cost {i in Legs}; #the cost of purchasing fuel and taking off
var final_refuel_gal = (default_fuel - (TO_fuel[5] - FB[5])) / fuel_lpg; #gallons of
fuel needed to refill to default after trip
var final_refuel_cost = final_refuel_gal  * Pr[1]; #cost of refuelling to the default
level

#set objective function
#sum of the cost at each leg plus the cost of the final refuelling
minimize total_cost: final_refuel_cost + sum{i in Legs} cost[i];

#specify constraints
#constraints to calculate the value of variables dependent on other ones
s.t. TO_fuel_init {i in Legs: i = 1}: TO_fuel[i] = default_fuel + F[i] * fuel_lpg;
s.t. TO_fuel_calc {i in Legs: i > 1}: TO_fuel[i] = TO_fuel[i-1] - FB[i-1] + F[i] *
fuel_lpg;
s.t. TO_wgt_calc {i in Legs}: TO_wgt[i] = BOW + TO_fuel[i] + Ppl[i];
s.t. L_wgt_calc {i in Legs}: L_wgt[i] = TO_wgt[i] - FB[i];
s.t. cost_calc {i in Legs}: cost[i] = F[i] * Pr[i] + RF[i] * x[i];

#limiting constraints
s.t. TO_wgt_constr {i in Legs}: TO_wgt[i] <= TO_Lim; #takeoff weight must be under the
maximum threshold
s.t. L_wgt_constr {i in Legs}: L_wgt[i] <= L_Lim; #landing weight must be under the
maximum threshold
```

```
s.t. min_flight_fuel_constr {i in Legs}: TO_fuel[i] >= FB[i] + fuel_buffer; #fuel must
be enough to finish leg and have extra in tank
s.t. fuel_cap_constr {i in Legs}: TO_fuel[i] <= f_cap; #weight of fuel must be under the
plane's max capacity
s.t. no_selling {i in Legs}: F[i] >= 0; #no selling fuel to an airport!
s.t. ramp_fee_waive_constr {i in Legs}: RF_thresh[i] <= F[i] + huge_num * x[i]; #if F[i]
exceeds the threshold, the ramp fee can be waived

#solve model
solve;

#display results of model nicely

printf "\nSolving the model with the given constraints, we find that the minimal cost
for the entire trip is:\n";
display total_cost;

printf "At each leg of the trip, we purchase the following gallons of fuel before
takeoff:\n";
display F;
printf "These purchases mean we take off for each leg of the trip with the following
pounds of fuel in the tank:\n";
display TO_fuel;
printf "Refilling after the trip, we purchase the following gallons of fuel:\n";
display final_refuel_gal;
```

The data file is given by:

```
param default_fuel := 7000;
param fuel_buffer :=2500;
param fuel_lpg := 6.7;
param f_cap := 14000;
param TO_Lim := 36400;
param L_Lim := 31800;
param p_wgt := 200;
param BOW := 22800;


param:  Ppl Pr   RF  RF_thresh FB:=
1       4   4.00 0   0         5100
2       6   8.32 800 600       2200
3       10  8.99 750 500       4700
4       10  6.48 600 650       3800
5       10  9.27 800 500       3600;
```

Running this model give the following output:



```
ampl: model HW5/model_Q1a.txt
CPLEX 20.1.0.0: optimal integer solution; objective 17254.07164
3 MIP simplex iterations
0 branch-and-bound nodes
absmipgap = 3.63798e-12, relmipgap = 2.10848e-16

Solving the model with the given constraints, we find that the minimal cost for the entire trip is:
total_cost = 17254.1

At each leg of the trip, we purchase the following gallons of fuel before takeoff:
F [*] :=
1   984.478
2   134.925
3     0
4  1104.48
5     0
;

These purchases mean we take off for each leg of the trip with the following pounds of fuel in the tank:
TO_fuel [*] :=
1  13596
2   9400
3   7200
4   9900
5   6100
;

Refilling after the trip, we purchase the following gallons of fuel:
final_refuel_gal = 671.642
```

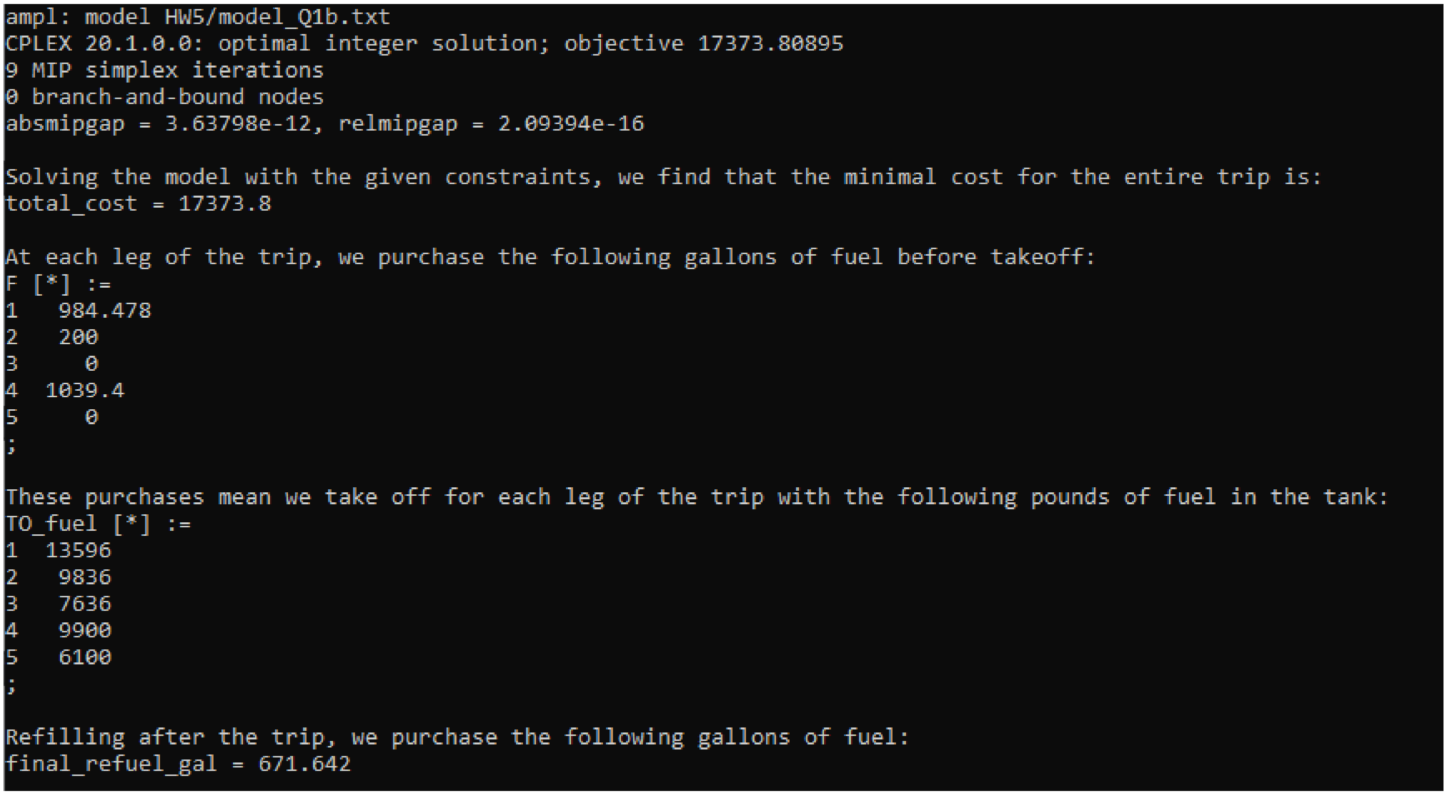
This solution results in a total trip cost of $17,254.10… that’s $3000 saved!

**(b) Suppose the BSAS department manager wished to modify the model to require that "if you buy any gas, you must buy at least 200 gallons". Formulate and solve the problem with this modification. How does this change the solution and cost for Ms. Chase's current optimal plan?**

This requires a small addition to the model and data file to include two disjunctive constraints. This is achieved by adding a new variables $z_i$ for leg "i" of the journey and adding the following constraints:

```
#disjunctive constraints
s.t. no_fuel_bought {i in Legs}: F[i] <= 0 + huge_num * z[i]; #don't buy fuel
s.t. minimum_fuel_purchase {i in Legs}: f_thresh <= F[i] + huge_num * (1 - z[i]); #buy
more gallons of fuel than the threshold
```

Note that f_thresh is 200 gallons of fuel. In our case. This new formulation gives the following output:



```
ampl: model HW5/model_Q1b.txt
CPLEX 20.1.0.0: optimal integer solution; objective 17373.80895
9 MIP simplex iterations
0 branch-and-bound nodes
absmipgap = 3.63798e-12, relmipgap = 2.09394e-16

Solving the model with the given constraints, we find that the minimal cost for the entire trip is:
total_cost = 17373.8

At each leg of the trip, we purchase the following gallons of fuel before takeoff:
F [*] :=
1   984.478
2   200
3     0
4  1039.4
5     0
;

These purchases mean we take off for each leg of the trip with the following pounds of fuel in the tank:
TO_fuel [*] :=
1  13596
2   9836
3   7636
4   9900
5   6100
;

Refilling after the trip, we purchase the following gallons of fuel:
final_refuel_gal = 671.642
```

This solution results in a total trip cost of $17,373.80 which is still an improvement of over $3000.

**Question 2: OKC Zoo**

This question gives the following incompatibility table for animals in the OKC zoo:

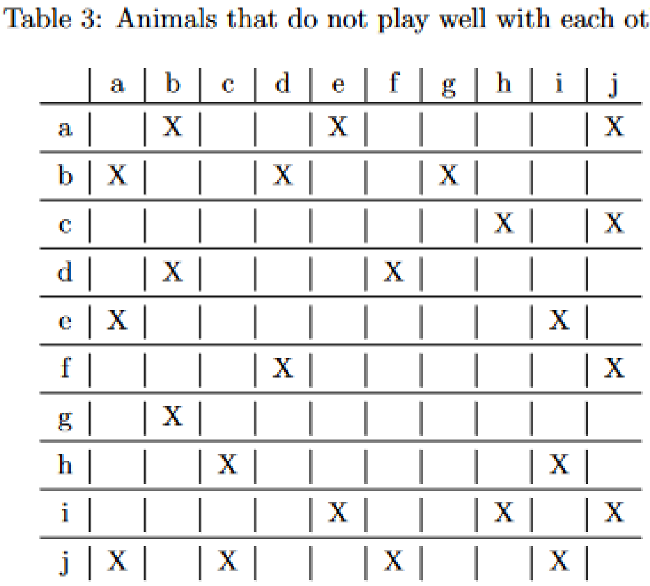
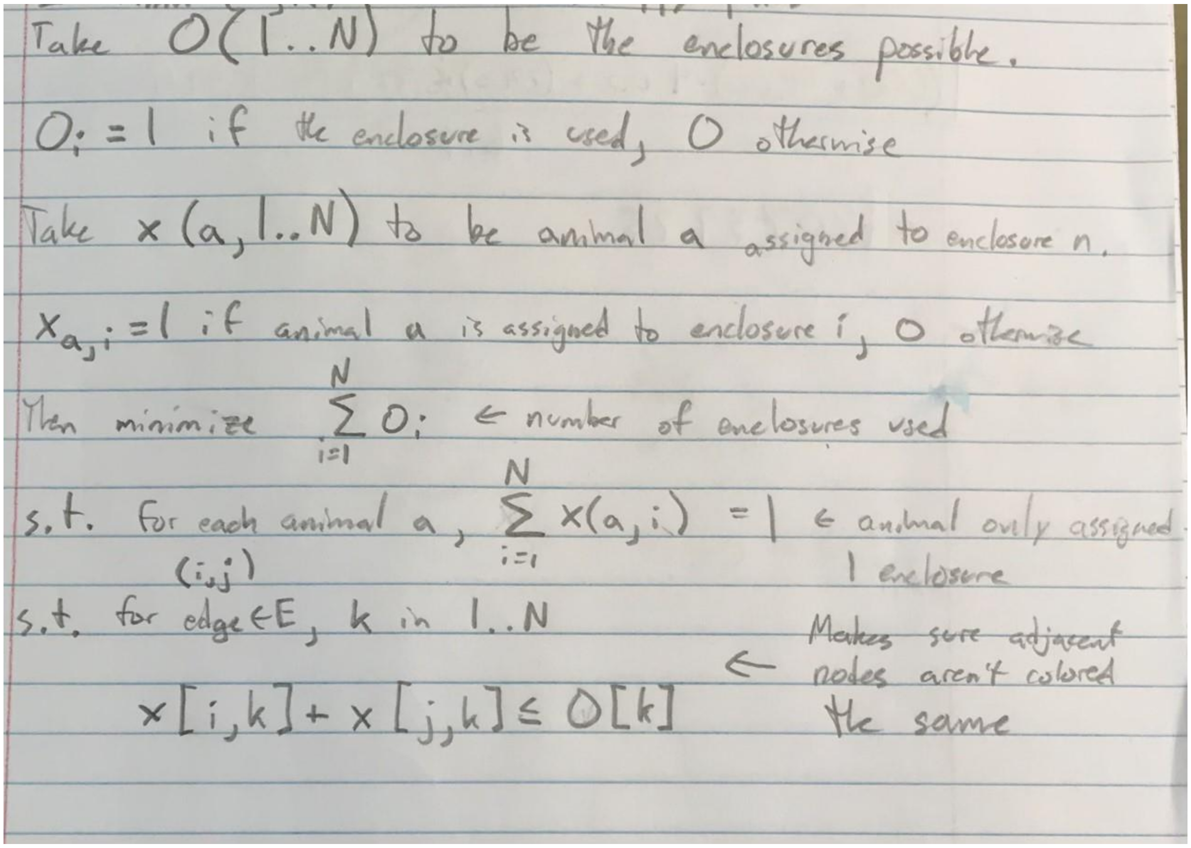


Table 3: Animals that do not play well with each other

| | a | b | c | d | e | f | g | h | i | j |
|---|---|---|---|---|---|---|---|---|---|---|
| a | | X | | | X | | | | | X |
| b | X | | | X | | | X | | | |
| c | | | | | | | | X | | X |
| d | | X | | | | X | | | | |
| e | X | | | | | | | | X | |
| f | | | | X | | | | | | X |
| g | | X | | | | | | | | |
| h | | | X | | | | | | X | |
| i | | | | | X | | | X | | X |
| j | X | | X | | | X | | | X | |

Take the set of animals, A, and consider a graph formed using A as the set of nodes and the subset of AxA denoted by the above table's marked X's as edges which I will denote E. For example, (a,b) is an element of E but (c,d) is not. We can then frame this as a graph-coloring problem, since a correctly colored graph (i.e. one where no nodes with a common edge are identically colored) corresponds to an assignment of enclosures were no animal is housed with an unfriendly species. We can then formulate the following model:



Take $O(1..N)$ to be the enclosures possible.

$O_i = 1$ if the enclosure is used, $0$ otherwise

Take $x(a, 1..N)$ to be animal a assigned to enclosure n.

$x_{a,i} = 1$ if animal a is assigned to enclosure i, 0 otherwise

Then minimize $\sum_{i=1}^{N} O_i$ ← number of enclosures used

s.t. For each animal a, $\sum_{i=1}^{N} x(a,i) = 1$ ← animal only assigned 1 enclosure

s.t. for $\overset{(i,j)}{\text{edge}} \in E$, k in 1..N

$x[i,k] + x[j,k] \leq O[k]$ ← Makes sure adjacent nodes aren't colored the same

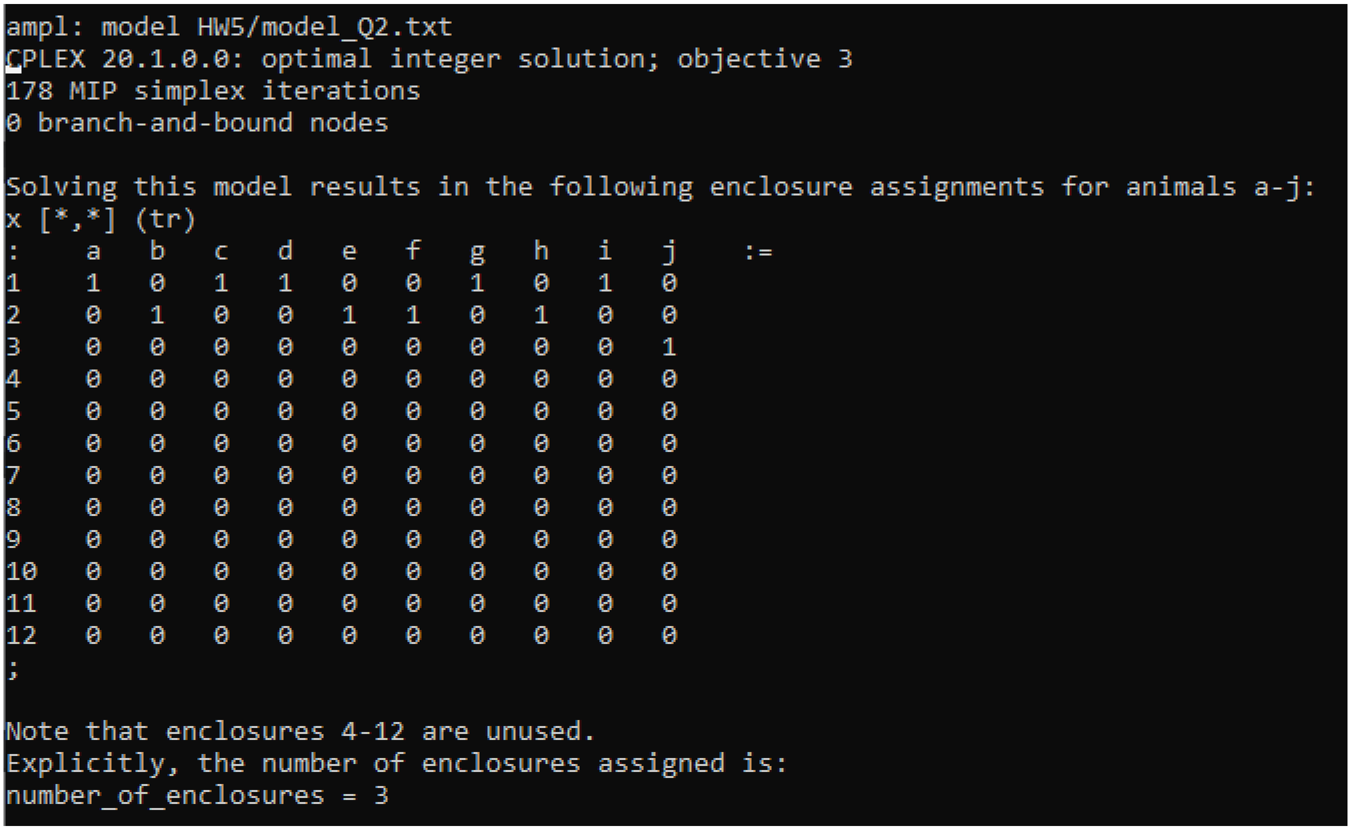
This model was then translated into AMPL as follows:

```
#clear memory
reset;

#choose solver;
option solver cplex;

#set our sets
set Animals; #the animals in our zoo
set Edges within {Animals, Animals}; #pairs on animals that don't get along

#set static parameters
param N; #number of edges (the maximum number of colors)

#define variables
var C{i in 1..N} binary; #whether or not enclosure i is used
var x{a in Animals, i in 1..N} binary; #whether animal a is assigned to enclosure i

#objective function
minimize number_of_enclosures: sum{i in 1..N} C[i];

#constraints
s.t. unique_assignment {a in Animals}: sum{i in 1..N} x[a, i] = 1; #each animal must be
assigned to only one enclosure
s.t. friends_only {(i,j) in Edges, k in 1..N}: x[i, k] + x[j, k] <= C[k]; #no animals
sharing an edge can be in the same enclosure
                                                                       #also if an
animal is in enclosure k, C[k] must be 1

#load data and solve
data HW5/data_Q2.txt;
solve;

#nicely display results
printf "\nSolving this model results in the following enclosure assignments for animals
a-j:\n";
display x;
printf "Note that enclosures 4-12 are unused.\n";
printf "Explicitly, the number of enclosures assigned is:\n";
display number_of_enclosures;
```

The following data file was also used:

```
set Animals := a b c d e f g h i j;
set Edges := (a,b) (a,e) (a,j) (b,d) (b,g) (c,h)
             (c,j) (d,f) (e,i) (f,j) (h,i) (i,j);

param N := 12; #since there are 12 edges
```

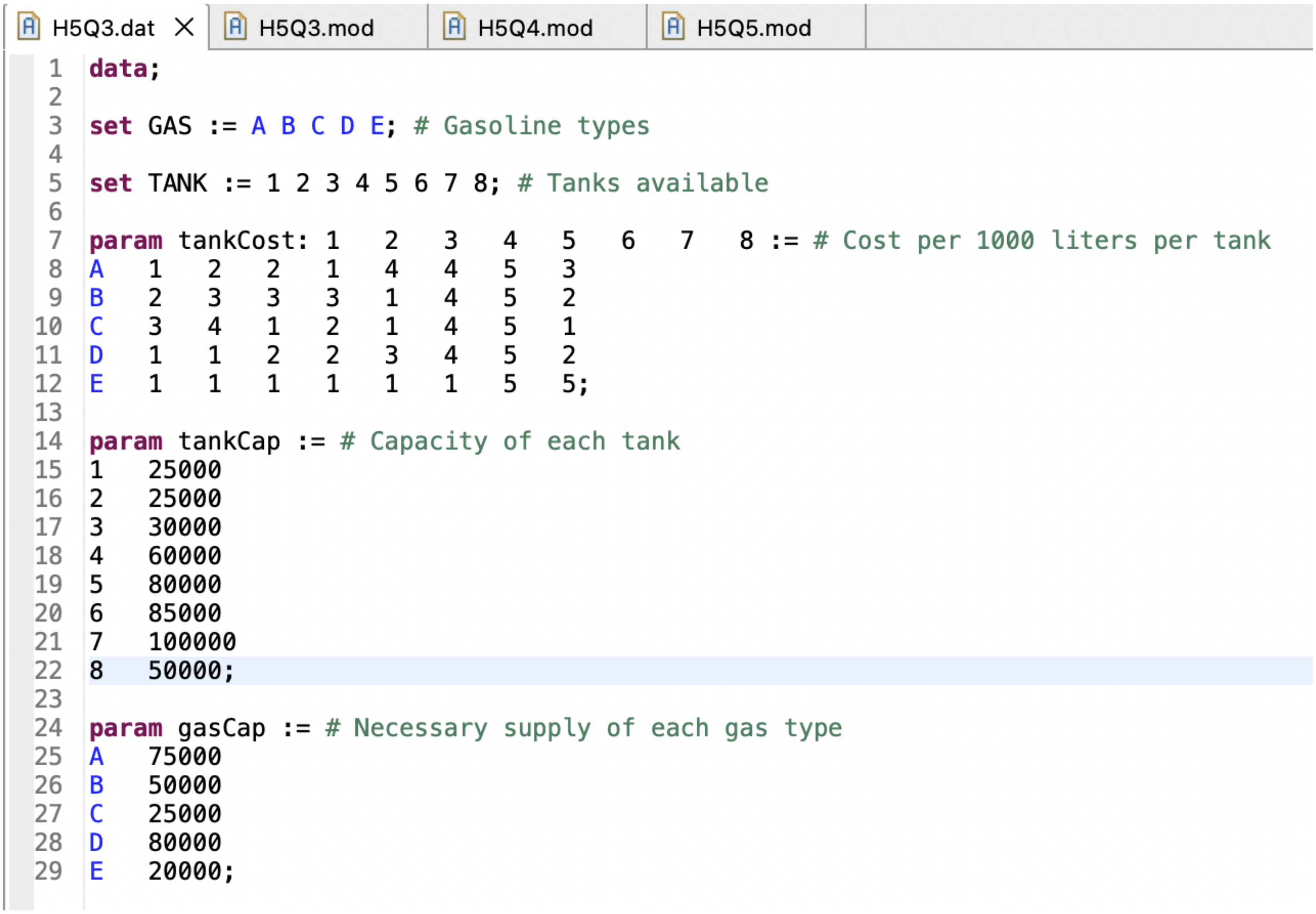
Running the model file produced the following ouput:



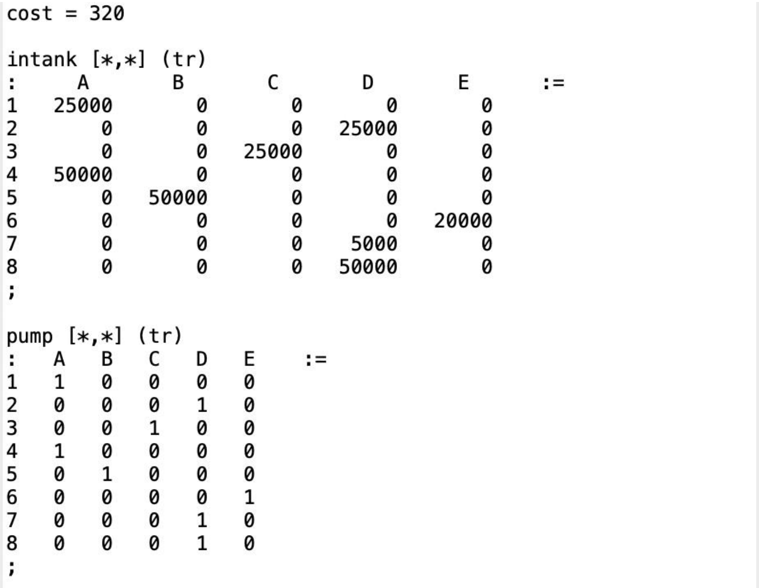
```
ampl: model HW5/model_Q2.txt
CPLEX 20.1.0.0: optimal integer solution; objective 3
178 MIP simplex iterations
0 branch-and-bound nodes

Solving this model results in the following enclosure assignments for animals a-j:
x [*,*] (tr)
:    a   b   c   d   e   f   g   h   i   j    :=
1    1   0   1   1   0   0   1   0   1   0
2    0   1   0   0   1   1   0   1   0   0
3    0   0   0   0   0   0   0   0   0   1
4    0   0   0   0   0   0   0   0   0   0
5    0   0   0   0   0   0   0   0   0   0
6    0   0   0   0   0   0   0   0   0   0
7    0   0   0   0   0   0   0   0   0   0
8    0   0   0   0   0   0   0   0   0   0
9    0   0   0   0   0   0   0   0   0   0
10   0   0   0   0   0   0   0   0   0   0
11   0   0   0   0   0   0   0   0   0   0
12   0   0   0   0   0   0   0   0   0   0
;

Note that enclosures 4-12 are unused.
Explicitly, the number of enclosures assigned is:
number_of_enclosures = 3
```

As can be seen in the output, the minimum number of enclosures that can be built without incurring animal violence is 3.

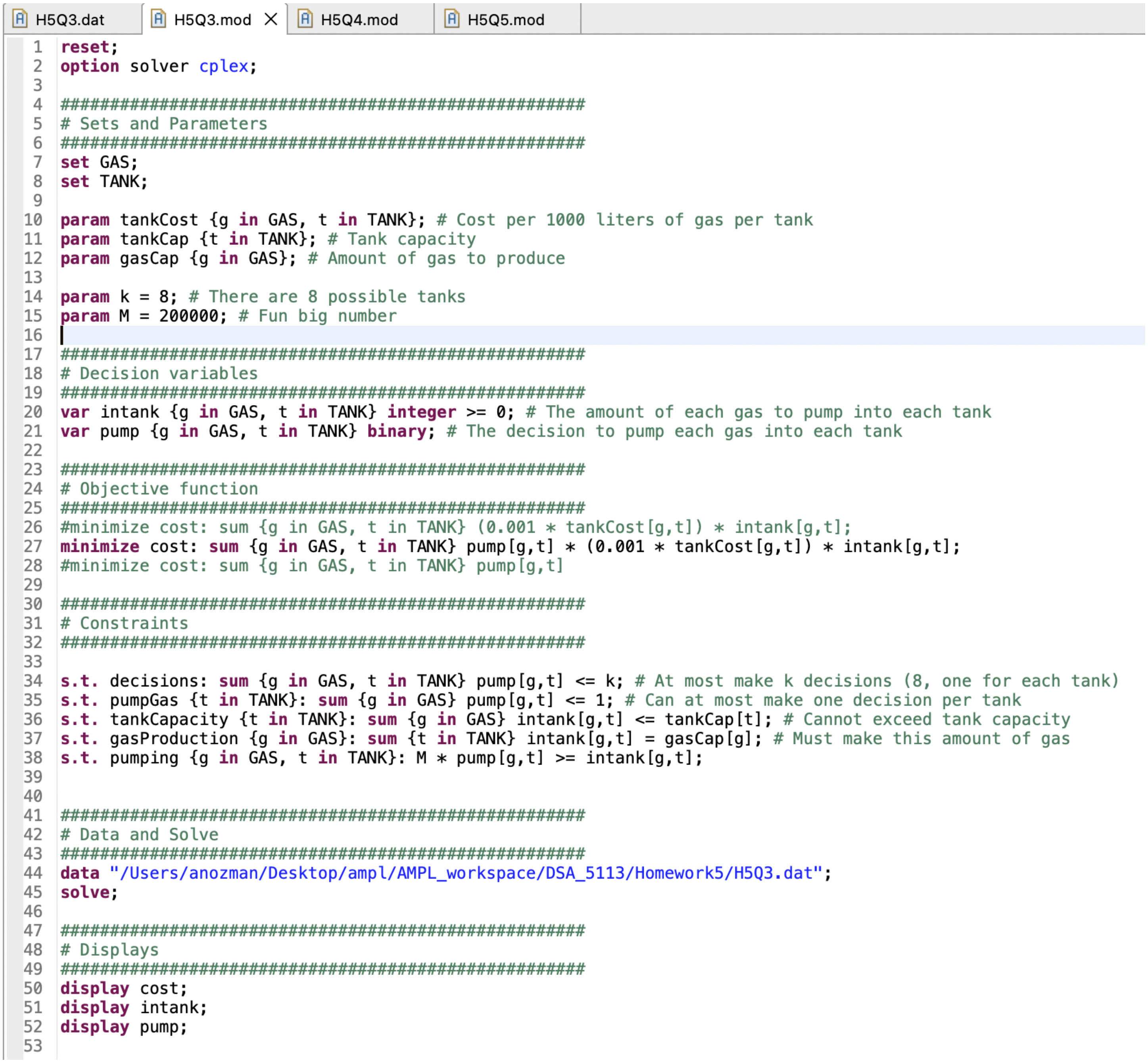
**Question 3: We Got Gas!**



```
data;

set GAS := A B C D E; # Gasoline types

set TANK := 1 2 3 4 5 6 7 8; # Tanks available

param tankCost: 1   2   3   4   5   6   7   8 := # Cost per 1000 liters per tank
A   1   2   2   1   4   4   5   3
B   2   3   3   3   1   4   5   2
C   3   4   1   2   1   4   5   1
D   1   1   2   2   3   4   5   2
E   1   1   1   1   1   1   5   5;

param tankCap := # Capacity of each tank
1   25000
2   25000
3   30000
4   60000
5   80000
6   85000
7   100000
8   50000;

param gasCap := # Necessary supply of each gas type
A   75000
B   50000
C   25000
D   80000
E   20000;
```



```
cost = 320

intank [*,*] (tr)
:     A       B       C       D       E      :=
1   25000       0       0       0       0
2       0       0       0   25000       0
3       0       0   25000       0       0
4   50000       0       0       0       0
5       0   50000       0       0       0
6       0       0       0       0   20000
7       0       0       0    5000       0
8       0       0       0   50000       0
;

pump [*,*] (tr)
:   A   B   C   D   E    :=
1   1   0   0   0   0
2   0   0   0   1   0
3   0   0   1   0   0
4   1   0   0   0   0
5   0   1   0   0   0
6   0   0   0   0   1
7   0   0   0   1   0
8   0   0   0   1   0
;
```



```
reset;
option solver cplex;

##################################################
# Sets and Parameters
##################################################
set GAS;
set TANK;

param tankCost {g in GAS, t in TANK}; # Cost per 1000 liters of gas per tank
param tankCap {t in TANK}; # Tank capacity
param gasCap {g in GAS}; # Amount of gas to produce

param k = 8; # There are 8 possible tanks
param M = 200000; # Fun big number

##################################################
# Decision variables
##################################################
var intank {g in GAS, t in TANK} integer >= 0; # The amount of each gas to pump into each tank
var pump {g in GAS, t in TANK} binary; # The decision to pump each gas into each tank

##################################################
# Objective function
##################################################
#minimize cost: sum {g in GAS, t in TANK} (0.001 * tankCost[g,t]) * intank[g,t];
minimize cost: sum {g in GAS, t in TANK} pump[g,t] * (0.001 * tankCost[g,t]) * intank[g,t];
#minimize cost: sum {g in GAS, t in TANK} pump[g,t]

##################################################
# Constraints
##################################################

s.t. decisions: sum {g in GAS, t in TANK} pump[g,t] <= k; # At most make k decisions (8, one for each tank)
s.t. pumpGas {t in TANK}: sum {g in GAS} pump[g,t] <= 1; # Can at most make one decision per tank
s.t. tankCapacity {t in TANK}: sum {g in GAS} intank[g,t] <= tankCap[t]; # Cannot exceed tank capacity
s.t. gasProduction {g in GAS}: sum {t in TANK} intank[g,t] = gasCap[g]; # Must make this amount of gas
s.t. pumping {g in GAS, t in TANK}: M * pump[g,t] >= intank[g,t];


##################################################
# Data and Solve
##################################################
data "/Users/anozman/Desktop/ampl/AMPL_workspace/DSA_5113/Homework5/H5Q3.dat";
solve;

##################################################
# Displays
##################################################
display cost;
display intank;
display pump;

```

SETS:

$GAS: g \in \{A, B, C, D, E\}$

$TANK: t \in \{1,2,3,4,5,6,7,8\}$

PARAMETERS:

$tankCost: P_{g,t} ; \forall g \in GAS, t \in TANK$

$tankCap: P_t, \forall t \in TANK$

$gasCap: p_g, \forall g \in GAS$

DECISION VARIABLES:

$intank: x_{g,t} \geq 0 ; \forall g \in GAS, t \in TANK$, the amount of each gas to pump into each tank.

$pump: y_{g,t} \in \{0,1\}; \; \forall \; g \in GAS, t \in TANK$, decision variable for each gas to be pumped into each tank. 1 if pumping *g* gas into *t* tank.

OBJECTIVE FUNCTION:

$minimize: cost = \sum pump_{g,t} * \left(0.001 * tankCost_{g,t} * intank_{g,t}\right); \; \forall \; g \in GAS, t \in TANK$

CONSTRAINTS:

$decision: \sum pump_{g,t} \leq k \; ; \; k = |TANK|, \forall \; g \in GAS, \; \forall \; t \in TANK$ , used to make sure no more than 8 decisions are made, one for each tank.

$pumpGas: \sum_{g \in GAS} pump_{g,t} \leq 1, \forall \; t \in TANK$, used to make sure that only one gas product is pumped into each tank.

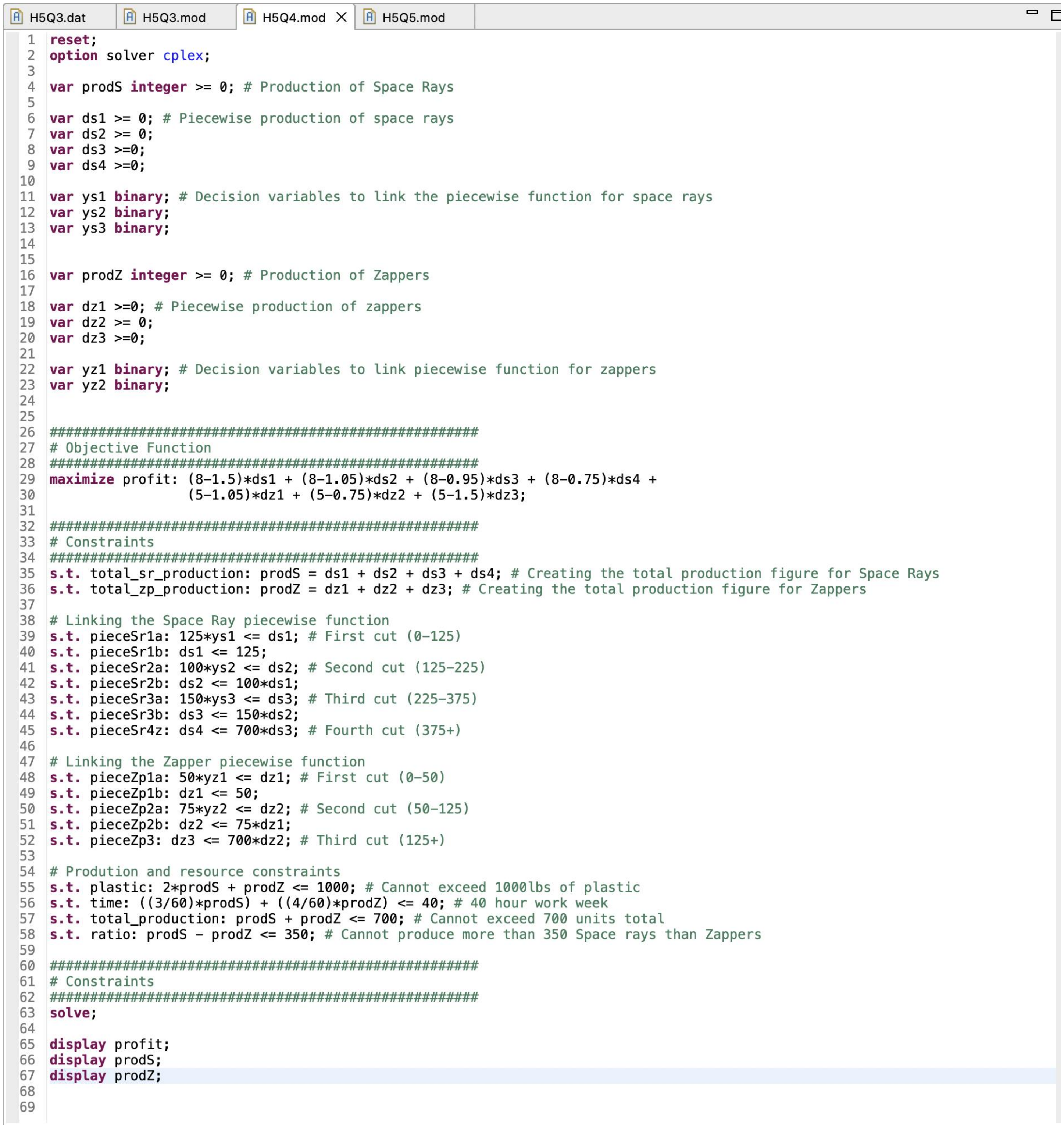
$tankCapacity: \sum_{g \in GAS} intank_{g,t} \leq tankCap_t \; ; \; \forall \; t \in TANK$ , each tank cannot hold beyond its capacity.

$gasProduction: \sum_{t \in TANK} intank_{g,t} = gasCap_g \; ; \; \forall \; g \in GAS$ , the sum of each gas product over all the tanks must be the amount of product needed to be stored.

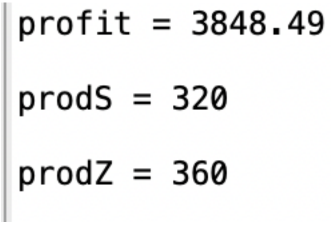
$pumping: M * pump_{g,t} \geq intank_{g,t} \; ; \; \forall \; g \in GAS, t \in TANK$, used to relate the binary decision variable to the storage decision variable. If pump = 0, then there will not be any gas pumped into each tank, and if pump=1, then gas can be pumped into the tank.

The solution realized in the problem utilized all 8 tanks to store the different gas products. The final cost that was incurred by the company was $320.

## Question 4: Galaxy Industries



```
reset;
option solver cplex;

var prodS integer >= 0; # Production of Space Rays

var ds1 >= 0; # Piecewise production of space rays
var ds2 >= 0;
var ds3 >=0;
var ds4 >=0;

var ys1 binary; # Decision variables to link the piecewise function for space rays
var ys2 binary;
var ys3 binary;


var prodZ integer >= 0; # Production of Zappers

var dz1 >=0; # Piecewise production of zappers
var dz2 >= 0;
var dz3 >=0;

var yz1 binary; # Decision variables to link piecewise function for zappers
var yz2 binary;


#####################################################
# Objective Function
#####################################################
maximize profit: (8-1.5)*ds1 + (8-1.05)*ds2 + (8-0.95)*ds3 + (8-0.75)*ds4 +
                 (5-1.05)*dz1 + (5-0.75)*dz2 + (5-1.5)*dz3;

#####################################################
# Constraints
#####################################################
s.t. total_sr_production: prodS = ds1 + ds2 + ds3 + ds4; # Creating the total production figure for Space Rays
s.t. total_zp_production: prodZ = dz1 + dz2 + dz3; # Creating the total production figure for Zappers

# Linking the Space Ray piecewise function
s.t. pieceSr1a: 125*ys1 <= ds1; # First cut (0-125)
s.t. pieceSr1b: ds1 <= 125;
s.t. pieceSr2a: 100*ys2 <= ds2; # Second cut (125-225)
s.t. pieceSr2b: ds2 <= 100*ds1;
s.t. pieceSr3a: 150*ys3 <= ds3; # Third cut (225-375)
s.t. pieceSr3b: ds3 <= 150*ds2;
s.t. pieceSr4z: ds4 <= 700*ds3; # Fourth cut (375+)

# Linking the Zapper piecewise function
s.t. pieceZp1a: 50*yz1 <= dz1; # First cut (0-50)
s.t. pieceZp1b: dz1 <= 50;
s.t. pieceZp2a: 75*yz2 <= dz2; # Second cut (50-125)
s.t. pieceZp2b: dz2 <= 75*dz1;
s.t. pieceZp3: dz3 <= 700*dz2; # Third cut (125+)

# Prodution and resource constraints
s.t. plastic: 2*prodS + prodZ <= 1000; # Cannot exceed 1000lbs of plastic
s.t. time: ((3/60)*prodS) + ((4/60)*prodZ) <= 40; # 40 hour work week
s.t. total_production: prodS + prodZ <= 700; # Cannot exceed 700 units total
s.t. ratio: prodS - prodZ <= 350; # Cannot produce more than 350 Space rays than Zappers

#####################################################
# Constraints
#####################################################
solve;

display profit;
display prodS;
display prodZ;


```



```
profit = 3848.49

prodS = 320

prodZ = 360
```

DECISION VARIABLES:

$prodS\ \geq 0$ , total production of Space Rays

$ds_x\ \geq 0$ , $x\ \in \{1,2,3,4\}$, piecewise production of space rays according to their bins (I have those bins commented out on the ampl code, and were provided in the problem).

$ys_x\ \in \{0,1\}, x\ \in \{1,2,3\}$, binary decision variables to transition along piecewise cost function.

$prodZ\ \geq 0$ , total production of Zappers

$dz_x\ \geq 0$ , $x\ \in \{1,2,3\}$ , piecewise production of space rays according to their bins.

$yz_x\ \in \{0,1\}, x\ \in \{1,2\}$ , decision variable to transition along piecewise cost function.

CONSTRAINTS:

$totalSRProduction\colon \sum_{x=1}^{4} ds_x = prodS$

$totalZPProduction\colon \sum_{x=1}^{3} dz_x = prodZ$

$plastic\colon 2 * prodS + prodZ\ \leq 1000$, cannot exceed the amount of plastic available for production.

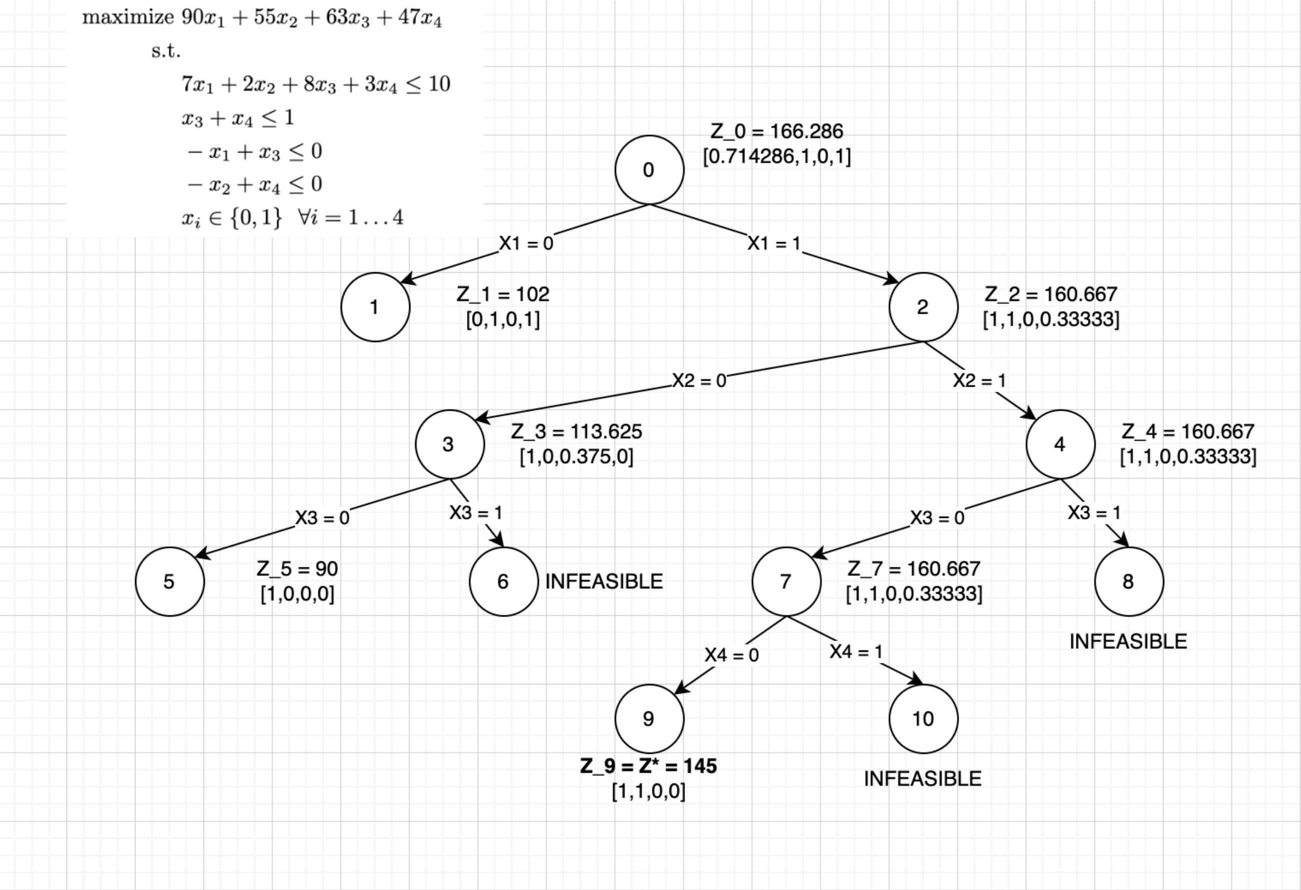
$time\colon \left(\left({}^{3}\!/_{60} * prodS\right) + \left({}^{4}\!/_{60} * prodZ\right)\right) \leq 40$ , cannot exceed the time of production.

$totalProduction\colon prodS + prodZ\ \leq 700$ , cannot produce more than 700 units.

$ratio\colon prodS - prodZ\ \leq 350$ , cannot produce 350 more Space Rays than Zappers.

For this section I will more or less describe how the different constraints work rather than writing them out. Hopefully this will reduce unnecessary repetition. For both the Space Rays and the Zappers we have piecewise cost functions, with each step having a varied cost than the previous. For each step, there are two parts, one that contains the binary decision variable (which is labeled part A for each “joint”) and the part that does not have this binary decision variable (labeled part B for each “joint”). Part A makes the decision to produce more product than the previous segment. Part B ensures that the amount of product produced meets the criteria to enter into the next cost bracket. In other words, it fills the piecewise function in order of the number of units produced.

## Question 5: Binary B&B problem



$$\begin{aligned}
&\text{maximize } 90x_1 + 55x_2 + 63x_3 + 47x_4\\
&\quad\text{s.t.}\\
&\qquad 7x_1 + 2x_2 + 8x_3 + 3x_4 \le 10\\
&\qquad x_3 + x_4 \le 1\\
&\qquad -x_1 + x_3 \le 0\\
&\qquad -x_2 + x_4 \le 0\\
&\qquad x_i \in \{0,1\} \quad \forall i = 1 \ldots 4
\end{aligned}$$

Beside each of the nodes is a Z_x term, which represents the LP solution for each node number. If the solution was impossible due to the constraints displayed in the top left of the diagram, there is an “INFEASIBLE” tag next to this node. Through our manual B&B search we were able to see that the optimal IP solution is 145.