Decision Support Methods - Assignment 1

Hugo Sales Tiago Macedo

Question 1

> *Assuming that you cannot hold the product in inventory (i.e., all that is produced in a given month must be sold in that month), formulate a linear optimization problem for maximizing the revenue.*

Assumptions

We assume that the number of marmalade produced must be a positive integer.

We assume the production capacity for each production operation is shared among all types. That is, if we clean 800 pieces of fruit for type R, that is $800/1000 = 80\%$ of the cleaning capacity; this means we can still clean at most $20\% * 1535 = 307$ of type C or $20\% * 1750 = 350$ of type I.

Method

We solved this question with the following program (the whole program can be found in the annexed `1.mod`):

```
set month;
param venus_price_r{month};
table data IN "CSV" "demand.csv": month <- [Month], venus_price_r ~ VenusR, ...
```

We declare the column data and use `table IN` to read it from a CSV file created with the values in the assignment PDF.

```
set operations;
param production_capacity_r{operations};
table data IN "CSV" "operations.csv": operations <- [Process], production_capacity_r ~ R, ...;
```

Similarly for the second table in the assignment, the production capacity for each operation.

```
param shuttle_capacity := 1000;
var venus_r{month}, >= 0, integer;
...
s.t. venus{m in month}: 0 <= (venus_r[m] + venus_c[m] + venus_i[m]) <= shuttle_capacity;
```

We define `venus_r` as the amount of R marmalade that should be sent to Venus each month, which are integers, and the total amount that is sent to Venus each month, `venus`, as the sum of the amount of each type, which is limited to the shuttle's 1000 unit transport capacity.

```
var prod_r{m in month}, >= 0, integer;
s.t. _prod_r{m in month}: prod_r[m] = (venus_r[m] + mars_r[m] +
mercury_r[m]);
...
s.t. production_capacity{m in month, op in operations}: 0 <= ((prod_r[m] / production_capacity_r[op]) + (p
```

We define `prod_r` as the amount of R that is produced each month and define `production_capacity` with the restrictrion explained in the assumptions, which states that the production capacity of each operation of shared among the production of each type of marmalade.

```
var profit_r{m in month};
s.t. _profit_r{m in month}: profit_r[m] = ((venus_r[m] * venus_price_r[m]) + (mars_r[m] * mars_price_r[m])

var profit{month};
s.t. _profit{m in month}: profit[m] = (profit_r[m] + profit_c[m] + profit_i[m]);

maximize annual_profit: sum{m in month} profit[m];
...
```

Finally we calculate the profit for each type and month. The reason we maximize the annual profit instead of each monthly profit is that doing so with:

```
maximize profit{m in month}: (profit_r[m] + profit_c[m] + profit_i[m]);
```

Lead to the profit being 0 for all but the first month, which we think is a limitation of GLPK.

Question 2

Determine the optimal revenue and the corresponding production plan

Running the model `1.mod` annexed (with `glpsol -m 1.mod -o res1`), we get the following:

Table 1: Monthly profit

Month	Profit
1	23696
2	29750
3	32300
4	35700
5	36836
6	46304
7	72400
8	88000
9	38250
10	36550
11	28080
12	25080

Giving us a total annual profit of 492946 *solarcoins*

Table 2: Amount sent to each planet each month

Month	Venus	Mars	Mercury
1	308	1000	0
2	0	850	0
3	0	850	0
4	0	850	0
5	422	656	0
6	422	656	0
7	200	1000	0
8	1000	200	0
9	0	850	0
10	0	850	0
11	308	1000	0
12	308	1000	0

Question 3

Determine (without solving additional problems) in which of the production lines the company should increase capacity

Table 3: Production line utilization for each operation, each month

Month	Cleaning	Cooking	Packing
1	0.879429	0.99982	0.910667
2	0.553746	1	0.566667
3	0.553746	1	0.566667
4	0.553746	1	0.566667
5	0.849362	0.999873	1
6	0.849362	0.999873	1
7	0.685714	1	0.6
8	0.685714	1	0.6
9	0.553746	1	0.566667
10	0.553746	1	0.566667
11	0.879429	0.99982	0.910667
12	0.879429	0.99982	0.910667

We observe that with these constraints, 0 units get sent to Mercury. This is a surprising result, but it happens simply because the whole production pipeline is limited by the ammount of fruit that can be cooked each month, which we can conclude by the fact that `production_capacity` for `Cooking` is always very close to 1, or 100%; in some months it doesn't quite reach 1 because the produced count is restricted to be an integer.

Therefore we conclude that the company should increase the production capacity of the cooking production line.

Question 4

> *Solve the problem again, now supposing that there is unlimited inventory capacity on Earth, with a unit holding cost of 1 solarcoin per month (i.e., the company must pay that amount for each unit of any of the marmalades left in stock at the end of each month)*

Assumptions

In addition to the assumptions stated for question 1, we assume the following the storage starts out empty and that items can stay in storage for more than a month. In addition, we can only store the finished product, rather than intermediate steps.

We represent storage as the amount available at the beginning of the month, produced in excess the month before. Therefore we need to consider 13 months for storage, as we need to deal with the overflow of the last month. We restrict the amount in storage at the first and thirteenth months to be 0.

Method

This problem can be solved with the following additions and modifications to the model used for problem 1:

```
set storage_month := 1..13;
var storage_r{storage_month}, >= 0, integer;
s.t. _storage_r_start: storage_r[1] = 0;
s.t. _storage_r_end: storage_r[13] = 0;
...
```

Declare the storage amount for each type as greater than 0 and assume it starts at 0 in the first month and 0 zero after the last month.

```
var sent_r{month}, >= 0;
s.t. _sent_r{m in month}: sent_r[m] = venus_r[m] + mars_r[m] + mercury_r[m];
...
var prod_r{month}, >= 0, integer;
s.t. _store_r{m in month}: storage_r[m + 1] = prod_r[m] + storage_r[m] - sent_r[m];
...
```

We replace `_prod_r` with `_store_r`, which declares that the amount in storage the next month is the amount produced plus the amount available in storage this month minus the amount sent to all planets.

`production_capacity` is unchanged from the previous model.

```
var storage_cost{storage_month} >= 0;
s.t. _storage_cost{m in storage_month}: storage_cost[m] = (storage_r[m] + storage_c[m] + storage_i[m]);

var revenue_r{m in month};
s.t. _revenue_r{m in month}: revenue_r[m] = ((venus_r[m] * venus_price_r[m]) + (mars_r[m] * mars_price_r[m]
...

var profit{month};
s.t. _profit{m in month}: profit[m] = (profit_r[m] + profit_c[m] + profit_i[m] - storage_cost[m]);

maximize annual_profit: sum{m in month} profit[m];
```

We calculate the storage cost, with the cost of 1 solarcoin per unit. This calculation doesn't include the "13th" month, but this isn't a problem since there can't be anything in storage that month. `profit_r` is changed to `revenue_r`, as this calculation doesn't include the cost; this calculation is unchanged, but we change the objective to subtract the cost of storage each month from the revenue of sales.

Results

Table 4: Profit each month

Month	Profit
1	0
2	-850
3	-1920
4	39003
5	32588
6	79159
7	111400
8	173200
9	38250
10	36550
11	28005
12	25155

As we can see, we obtain a negative profit in the first 3 months. We justify this by the fact that the price of nearly all prodcuts goes up significantly in the following months, making it more profitable to store the production for later months.

Giving us a total annual profit of 560540 *solarcoins*

Table 5: Amount of type of product sent to each planet each month

	Venus			Mars			Mercury		
Month	R	C	I	R	C	I	R	C	I
1	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0
4	0	0	0	0	1000	0	0	0	0
5	1000	0	0	0	330	0	0	0	0
6	1000	0	0	0	1000	0	20	0	0
7	0	0	1000	0	0	1000	0	0	0
8	0	0	1000	0	0	1000	0	0	1000
9	0	0	0	0	850	0	0	0	0
10	0	0	0	0	850	0	0	0	0
11	296	0	3	0	0	1000	0	0	0
12	316	0	0	0	0	1000	0	0	0

Table 6: Storage of each type by month

Month	R	C	I
1	0	0	0
2	0	850	0
3	407	1513	0
4	827	2170	0

Month	R	C	I
5	1228	1330	714
6	624	1000	1657
7	0	0	2600
8	0	0	1800
9	0	0	0
10	0	0	0
11	0	0	0
12	0	0	5
13	0	0	0

We can see from this table that it becomes optimal to store product in the first half of the year, for sale later on, when the prices increase. Additionally, we note that storage increases each month by at most the prodction capacity of each product.