CS 325 Project 4 Report

Group 3

Date: December 2, 2014

Contributors: Cezary Wojcik, Sean McGlothlin, Matthew Eilertson

Problem 1

Linear Program - Mathematical

Objective:

$$\begin{aligned}\max\{&4 \times \text{Edameme}_{\text{plain}} + 12 \times \text{Edameme}_{\text{regular}} + 7 \times \text{Edameme}_{\text{overtime}} \\ &+8 \times \text{Tofu}_{\text{plain}} + 14 \times \text{Tofu}_{\text{regular}} + 11 \times \text{Tofu}_{\text{overtime}} \\ &+4 \times \text{Tempeh}_{\text{plain}} + 13 \times \text{Tempeh}_{\text{regular}} + 9 \times \text{Tempeh}_{\text{overtime}}\}\end{aligned} \tag{1}$$

Constraints:

$$\begin{gathered}\text{Edameme}_{\text{plain}} + \text{Edameme}_{\text{regular}} + \text{Edameme}_{\text{overtime}} \leq 400 \\ \text{Tofu}_{\text{plain}} + \text{Tofu}_{\text{regular}} + \text{Tofu}_{\text{overtime}} \leq 480 \\ \text{Tempeh}_{\text{plain}} + \text{Tempeh}_{\text{regular}} + \text{Tempeh}_{\text{overtime}} \leq 230 \\ \text{Edameme}_{\text{regular}} + \text{Tofu}_{\text{regular}} + \text{Tempeh}_{\text{regular}} \leq 420 \\ \text{Edameme}_{\text{overtime}} + \text{Tofu}_{\text{overtime}} + \text{Tempeh}_{\text{overtime}} \leq 250\end{gathered}$$

Linear Program - Matrix

Objective:

$$\begin{pmatrix} 4 & 12 & 7 & 8 & 14 & 11 & 4 & 13 & 9 \end{pmatrix}$$

Constraints:

$$\begin{pmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 \end{pmatrix} \leq \begin{pmatrix} 400 \\ 480 \\ 230 \\ 420 \\ 250 \end{pmatrix}$$

Optimal Solution

400 bags of edamame flavored on regular time, 440 blocks of plain tofu, 40 blocks of tofu flavored on overtime, 20 blocks of tempeh flavored on regular time, and 210 blocks of tempeh flavored on overtime.

Total profit: $10,910.

LP Environment

To solve the LP problem, we used Python and a package named "PyLPSolve," which is an object oriented wrapper for the open source Python LP solver "lpsolve."

Code

```
from pylpsolve import LP
lp = LP()
lp.setObjective([4, 12, 7, 8, 14, 11, 4, 13, 9], mode="maximize")
lp.addConstraint([[1, 1, 1, 0, 0, 0, 0, 0, 0],
  [0, 0, 0, 1, 1, 1, 0, 0, 0],
  [0, 0, 0, 0, 0, 0, 1, 1, 1],
  [0, 1, 0, 0, 1, 0, 0, 1, 0],
  [0, 0, 1, 0, 0, 1, 0, 0, 1]],
  "<=",
  [400, 480, 230, 420, 250])
lp.solve()
print lp.getSolution()
```

Problem 2

Linear Program - Mathematical

We want to minimize Φ subject to $\Phi = \max_{1 \leq i \leq n} \{ax_i + by_i - c\}$.

Objective:

$$\min \{\Phi\}$$

Constraints:

$$\Phi \geq ax_0 + by_0 - c$$
$$\Phi \geq -ax_0 - by_0 + c$$
$$\ldots$$
$$\Phi \geq ax_n + by_n - c$$
$$\Phi \geq -ax_n - by_n + c$$

Also, to account for the $a = b = c = 0$ case, we have to run this LP twice, once with a constraint that $a = 1$ and once with a constraint that $b = 1$. The one with a lower Φ value is the better solution.

Specific Problem

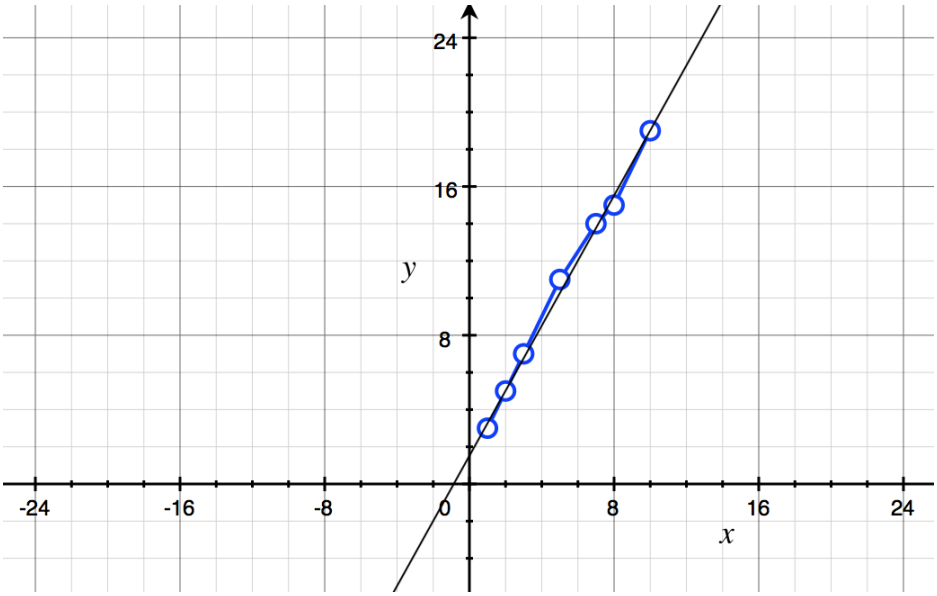
The best solution for:

$$(1, 3), (2, 5), (3, 7), (5, 11), (7, 14), (8, 15), (10, 19)$$

is approximately:

$$x - 0.571429y = -0.857143$$

Here is a graph of the solution:



LP Environment

For this problem, we used AMPL, which allows defining the LP in more abstract terms as opposed to just using a matrix. The code below can be executed on http://ampl.com/cgi-bin/ampl/amplcgi. Output 3 from APL should be set to:

```
solve;
display _varname, _var
```

Code

```
set Points;
param x{Points};
param y{Points};

var a;
var b;
var c;
var z{Points};

minimize obj: sum{i in Points} z[i];

s.t. c1{i in Points}: z[i] >= a*x[i] + b*y[i] - c;
```

```
s.t. c2{i in Points}: z[i] >= -1*a*x[i] - b*y[i] + c;

s.t. c3{i in Points}: a = 1; # switch to b = 1 for second run

data;

set Points := 1 2 3 4 5 6 7;

param: x y   :=
1 1 3
2 2 5
3 3 7
4 5 11
5 7 14
6 8 15
7 10 19;
```