ΑΝΑΦΟΡΑ ΜΟΝΤΕΛΟΥ

Ούτως ώστε η FastVegan να μπορέσει να βρει το βέλτιστο σχέδιο δημιουργίας κουζινών & μεταφοράς στο λεκανοπέδιο, αναπτύχθηκε το ακόλουθο μοντέλο (μεικτού) μαθηματικού προγραμματισμού:

Ξεκινώντας, πρέπει να οριστούν οι μεταβλητές του προβλήματος ως εξής:

(όπου i οι ταχυδρομικοί κώδικες και j οι κουζίνες)

- $\boldsymbol{m_{ij}}$**,** ακέραια μεταβλητή που δηλώνει την ποσότητα που μεταφέρεται από την κουζίνα j στον ΤΚ i
- $\boldsymbol{d_j}$*, δυαδική μεταβλητή* όπου παίρνει την τιμή 1 αν η κουζίνα j ανοίξει και 0 διαφορετικά
- $\boldsymbol{z_{ij}}$*,* δυαδική μεταβλητή όπου παίρνει την τιμή 1 αν το κανάλι μεταξύ κουζίνας j και ΤΚ i ανοίξει και 0 διαφορετικά
- $\boldsymbol{x_j}$*,* ακέραια μεταβλητή που δηλώνει την συνολική ποσότητα που μεταφέρει η κουζίνα j

Αμέσως μετά, ορίζεται η αντικειμενική συνάρτηση η οποία αναπαριστά το συνολικό κόστος της FastVegan, το οποίο θέλουμε να ελαχιστοποίησουμε:

$$\min \quad \underbrace{\sum_i \sum_j (vc_{ij} * m_{ij})}_{\textbf{(α)}} + \underbrace{\sum_j (fc_j * d_j)}_{\textbf{(β)}}$$

όπου:

- $\boldsymbol{vc_{ij}}$ το κόστος που χρειάζεται για να μεταφερθεί μία μερίδα από το κανάλι j-i
- $\boldsymbol{fc_j}$ το κόστος εγκατάστασης της κουζίνας j

στην παραπάνω σχέση, το **(α)** υποδηλώνει το συνολικό μεταβλητό κόστος και το **(β)** το συνολικό σταθερό κόστος που θα υποστεί η επιχείρηση. Τα $\boldsymbol{vc_{ij}}$ & $\boldsymbol{fc_j}$ θεωρούνται σταθερές τιμές για τους δείκτες i & j, δηλαδή είναι γνωστές από το πρόγραμμα πριν την επίλυση.

Στον κώδικα, τα παραπάνω ορίζονται ως:

```
model.d = Var(model.locations, within = Binary) # dj, binary variable so that d[j] is 1 if kitchen j opens
model.m = Var(model.destinations, model.locations, within = NonNegativeIntegers)
# mij, where m[i,j] is the quantity that is being transfered to i from j

#                                                variable cost
model.obj = Objective(expr = sum(t_cost[i][j] * model.m[i,j] for i in model.destinations for j in model.locations)
                  +  sum(model.d[j] * f_cost[j] for j in model.locations), sense = minimize)
#                                fixed cost

model.x = Var(model.locations, within = NonNegativeIntegers) # xj, where x[j] is the quantity that kitchen j moves
model.z = Var(model.destinations, model.locations, within = Binary) # zij, binary variable so that z[i,j] is 1 if TK i is being served from kitchen j
```

Συνεχίζουμε με τον ορισμό των περιορισμών του μοντέλου ως εξής:

$$x_1 = \sum_i m_{i1}$$

.

.

.

$$x_{10} = \sum_i m_{i10}$$

(α)

$$\sum_j z_{1j} = 1$$

.

.

.

$$\sum_j z_{30j} = 1$$

(β)

$$x_1 \leq sp_1 * d_1$$

.

.

.

$$x_{10} \leq sp_{10} * d_{10}$$

(γ)

$$m_{11} = dmnd_1 * z_{11}$$

.

.

.

$$m_{301} = dmnd_{30} * z_{301}$$

.

.

.

$$m_{3010} = dmnd_{30} * z_{3010}$$

(δ)

Όπου:

$\boldsymbol{sp_j}$ η δυναμικότητα παραγωγής *της j κουζίνας*

$\boldsymbol{dmnd_i}$ *η ζήτηση του i ΤΚ*

Οι περιορισμοί **(α)** ορίζουν την μεταβλητή $\boldsymbol{x_j}$ ως την συνολική ποσότητα που μεταφέρει η j κουζίνα.

Οι **(β)** περιορίζουν τους ΤΚ ώστε κάθε i, να μπορεί να εξυπηρετηθεί από μία μόνο κουζίνα και έτσι το άθροισμα των $\boldsymbol{z_{ij}}$ για κάθε j στο συγκεκριμένο i, πρέπει να ισούται με 1.

Οι **(γ)** σιγουρεύουν ότι κάθε κουζίνα j μπορεί να μεταφέρει μέχρι και $\boldsymbol{sp_j}$ μερίδες (πολλαπλασιάζουμε την ποσότητα με $\boldsymbol{d_j}$ ώστε αν η κουζίνα j δεν ανοίξει, να μπορεί να μεταφέρει μέχρι και 0 στο σύνολο).

Οι περιορισμοί **(δ)** είναι για να σιγουρέψουμε ότι στην βέλτιστη λύση, η ζήτηση κάθε ΤΚ έχει εξυπηρετηθεί. Πιο συγκεκριμένα, κάθε $\boldsymbol{m_{ij}}$ για όλους τους συνδυασμούς i & j, πρέπει να ισούται με την ζήτηση του i, μόνο όμως αν το συγκεκριμένο κανάλι μεταξύ i και j έχει ανοίξει, διαφορετικά να ισούται με 0 (για αυτό πολλαπλασιάζουμε την ζήτηση με το $\boldsymbol{z_{ij}}$). Το παραπάνω, σε συνδυασμό με τους **(β)**, σιγουρεύει ότι για έναν συγκεκριμένο ΤΚ, η ποσότητα που θα λάβει από μία μόνο κουζίνα, θα ισούται με την ζήτηση του και οι ποσότητες από τις άλλες κουζίνες να ισούται με 0.

Στην πράξη, τα παραπάνω μορφοποιούνται ως:

```python
# define x[k] as the total quantity kitchen k moves
for j in model.locations :
    model.constraints.add(model.x[j] == sum(model.m[i,j] for i in model.destinations))

# each k kitchen can only produce up to av_space[k] of goods
for j in model.locations :
    model.constraints.add(model.x[j] <= av_space[j] * model.d[j])

# each TK has to be served by one and only kitchen
for i in model.destinations:
    model.constraints.add(sum(model.z[i,j] for j in model.locations) == 1)

# each k TK has to receive exaclty demand[k] worth of goods
for i in model.destinations :
    for j in model.locations :
        model.constraints.add(model.m[i,j] == demand[i] * model.z[i,j])
```

Η βέλτιστη λύση του μοντέλου παρατίθεται ως εξής:

```
------------------------------------------------
                Results
------------------------------------------------

FastVegan has to open kitchens in Location(s): 3,4,7,8
3 will serve 4407.0 portions
4 will serve 4882.0 portions
7 will serve 3477.0 portions
8 will serve 4466.0 portions


TK # 1 is going to eat from Location: 3

TK # 2 is going to eat from Location: 4

TK # 3 is going to eat from Location: 3

TK # 4 is going to eat from Location: 8

TK # 5 is going to eat from Location: 4

TK # 6 is going to eat from Location: 3

TK # 7 is going to eat from Location: 7

TK # 8 is going to eat from Location: 8

TK # 9 is going to eat from Location: 3

TK # 10 is going to eat from Location: 8

TK # 11 is going to eat from Location: 3

TK # 12 is going to eat from Location: 8

TK # 13 is going to eat from Location: 3

TK # 14 is going to eat from Location: 8

TK # 15 is going to eat from Location: 3

TK # 16 is going to eat from Location: 8

TK # 17 is going to eat from Location: 4

TK # 18 is going to eat from Location: 4

TK # 19 is going to eat from Location: 4

TK # 20 is going to eat from Location: 3

TK # 21 is going to eat from Location: 7

TK # 22 is going to eat from Location: 4

TK # 23 is going to eat from Location: 7

TK # 25 is going to eat from Location: 4

TK # 26 is going to eat from Location: 7

TK # 27 is going to eat from Location: 7

TK # 28 is going to eat from Location: 8

TK # 29 is going to eat from Location: 4

TK # 30 is going to eat from Location: 8

------------------------------------------
Total cost of Franchise: 1435619.1
------------------------------------------
```

Με την εισαγωγή των περιορισμών χωροταξίας των κουζινών, πρέπει και το μοντέλο να μεταβληθεί αντίστοιχα. Πιο αναλυτικά, πρέπει να προσθέσουμε 4 νέους περιορισμούς στις μεταβλητές:

$d_1 \leq d_2$
$d_1 \leq d_{10}$
(α)

$d_6 + d_7 \leq 1$ **(β)**

$d_4 + d_7 + d_8 + d_9 \leq 3$ **(γ)**

Οι περιορισμοί **(α)** σιγουρεύουν ότι οι κουζίνες 10 & 2 (Χολαργός και Γλυφάδα) πρέπει αναγκαστικά να ανοίξουν, αν η κουζίνα 1 (Αμπελόκηποι) ανοίξει. Αυτό διότι με την $\boldsymbol{d_1}$ να ισούται με 1, αναγκάζει τις $\boldsymbol{d_2}$ $\boldsymbol{d_{10}}$ επίσης να αυξηθούν στο 1 ώστε να ικανοποιείται ο περιορισμός. Αν η κουζίνα δεν ανοίξει (και έτσι η δυαδική μεταβλητή $\boldsymbol{d_1}$ ισούται με 0), τότε οι αντίστοιχες κουζίνες δεν περιορίζονται με κανέναν τρόπο και έτσι μπορούν είτε να ανοίξουν, είτε όχι.

Ο **(β)** θέτει τον περιορισμό μεταξύ της κουζίνας στο Κουκάκι και Μοσχάτο (κουζίνες 6 και 7), καθώς δεν μπορούν να ανοίξουν και οι δύο μαζί. Αυτό επιτυγχάνεται περιορίζοντας το άθροισμα των δυαδικών τους μεταβλητών με κλειστό άνω όριο το 1, διότι για να ανοίξουν και οι δύο μαζί, θα έπρεπε το άθροισμα να ήταν 2.

Ο **(γ)**, τέλος, δεν επιτρέπει την λειτουργία των κουζινών στις περιοχές της Κυψέλη, Μοσχάτου, Περιστερίου και Χαλανδρίου (κουζίνες 4, 7, 8 και 9 αντίστοιχα) ταυτόχρονα. Αυτό επιτυγχάνεται περιορίζοντας το άθροισμα των δυαδικών τους μεταβλητών με κλειστό άνω όριο το 3. Μία ενέργεια λειτουργίας των τεσσάρων κουζινών ταυτόχρονα, θα απαιτούσε το παραπάνω άθροισα να ήταν 4. Σημειώνεται ότι ο **(γ)** δεν περιορίζει καθόλου την ταυτόχρονη λειτουργία οποιουδήποτε υποσυνόλου κουζινών μικρότερο του 4.

Ο κώδικας που ορίζει τα παραπάνω συντάσσεται ως εξής :

```
# kitchen 10 & 2 (9 & 1 in the model) have to open if kitchen 1 does
model.constraints.add(model.d[0] <= model.d[1])
model.constraints.add(model.d[0] <= model.d[9])
# note: if kitchen 1 does *not* open, the other kitchens are not limited about if they work or not

# if kitchen 6 opens, kitchen 7 has to remain closed
model.constraints.add(model.d[5] + model.d[6] <= 1)

# kitchens 4, 7, 8 & 9 cant open all together
model.constraints.add(model.d[3] + model.d[6] +model.d[7] + model.d[8] <= 3)
```

Η βέλτιστη λύση, με την προσθήκη των παραπάνω περιορισμών, παραμένει αμετάβλητη.

Τέλος, για την υλοποίηση της λογικής του pick-up από τις κουζίνες μας, πρέπει το μοντέλο να μεταβληθεί ως εξής :

Αρχικά, πρέπει να οριστούν οι παρακάτω νέες μεταβλητές :

- $\boldsymbol{y_j}$, *δυαδική μεταβλητή* όπου παίρνει την τιμή 1 αν η κουζίνα j μεταφέρει τις μισές από τις διαθέσιμες μερίδες και 0 διαφορετικά.
- $\boldsymbol{t_{ij}}$, ακέραια μεταβλητή που δηλώνει την ποσότητα όπου παραλαμβάνεται στην κουζίνα j από το ΤΚ i.
- $\boldsymbol{r_j}$, ακέραια μεταβλητή που δηλώνει την συνολική ποσότητα που παραλαμβάνεται στην κουζίνα j.

Μετά, πρέπει να προστεθεί στην αντικειμενική μας συνάρτηση το κόστος που θα υποστεί η FastVegan από τις παραλαβές :

$$\ldots + \sum_i \sum_j t_{ij} * vc_{ij} * 0.2$$

Παρατηρούμε και την εφαρμογή του μειωμένου κόστους της παραλαβής, σε σχέση με την μεταφορά, που ανέρχεται στα 20% του αρχικού κόστους.

Τέλος, πρέπει να αλλάξουμε τους περιορισμούς της δυναμικότητας και ζήτησης, ούτως ώστε να ανταποκρίνονται στα νέα πλαίσια του προβλήματος :

$r_1 = \sum_i t_{i1}$ $\quad x_1 \leq 0.5 * sp_1 * d_1$

. .

. **(α)** . **(β)**

. .

$r_{10} = \sum_i t_{i10}$ $\quad x_{10} \leq 0.5 * sp_{10} * d_{10}$

$r_1 \leq 0.5 * sp_1 * y_1$ $\quad d_1 \geq y_1$

. .

. **(γ)** . **(δ)**

. .

$r_{10} \leq 0.5 * sp_{10} * y_{10}$ $\quad d_{10} \geq y_{10}$

Σημείωση: Για την συνολική υλοποίηση του παραπάνω μοντέλου, θα χρειαστεί να διαγραφούν οι παλιοί περιορισμοί της δυναμικότητας και ζήτησης (βλέπε **(γ)** και **(δ)** σελ. 2).

$$m_{11} + t_{11} = dmnd_1 * z_{11}$$

.

.

.

$$m_{301} + t_{301} = dmnd_{30} * z_{301} \quad \textbf{(ε)}$$

.

.

.

$$m_{3010} + t_{3010} = dmnd_{30} * z_{3010}$$

Οι περιορισμοί **(α)**, ορίζουν την μεταβλητή r_j ως την συνολική ποσότητα που δίνει η κουζίνα j στο κατάστημα (δηλαδή με παραλαβή).

Οι **(β)**, επιτρέπουν κάθε κουζίνα να μεταφέρει μέχρι και τις μισές μερίδες από τις διαθέσιμες, αν όμως ανοίξει (για αυτό και πολλαπλασιάζουμε την δυναμικότητα με $\boldsymbol{d_j}$).

Οι **(γ)** διαδραματίζουν παρόμοιο ρόλο, αλλά για τον περιορισμό της παραλαβής, πάλι των μισών μονάδων, πλέον όμως μόνο αν έχουν μεταφερθεί ήδη οι μισές (για αυτό πολλαπλασιάζουμε με $\boldsymbol{y_j}$).

Οι περιορισμοί **(δ)** είναι απαραίτητοι για να μην επιτραπεί σε καμία κουζίνα να μπορεί να δώσει στο κατάστημα μερίδες, αν παραμείνει κλειστή.

Τέλος, οι **(ε)** χρησιμεύουν για την επιβεβαίωση της κάλυψης της ζήτησης, πλέον και από την παραλαβή, μαζί με την μεταφορά ($\boldsymbol{m_{ij} + t_{ij}}$) (αν όμως το κανάλι i-j δεν ανοίξει, πρέπει να περιοριστεί στο 0).

Τα παραπάνω ορίζονται συνολικά ως ακολούθως :

```
model.y = Var(model.locations, within = Binary)
# yj, binary variable so that y[j] is 1 if kitchen j delivered half of its stock
model.t = Var(model.destinations, model.locations, within = NonNegativeIntegers)
# tij, the quantity that is being served on site j to TK i

model.obj.expr += sum(model.t[i,j] * t_cost[i][j] * 0.2 for j in model.locations for i in model.destinations)

model.r = Var(model.locations, within = NonNegativeIntegers)
# rj, where r[j] is the quantity that is being received from site j

# define r[k] as the total quantity that kitchen k gives on site
for j in model.locations :
    model.constraints.add(model.r[j] == sum(model.t[i,j] for i in model.destinations))

# each k kitchen can only transport up to av_space[k] / 2 of goods
for j in model.locations :
    model.constraints.add(model.x[j] <= 0.5 * av_space[j] * model.d[j])

# each k kitchen can only give on site up to av_space[k] / 2 of goods
for j in model.locations :
    model.constraints.add(model.r[j] <= 0.5 * av_space[j] * model.y[j])

# make sure dj & yj are dependent to eachother (yj can be 1 if dj is 1)
for j in model.locations :
    model.constraints.add(model.d[j] >= model.y[j])

# each k TK has to receive exaclty demand[k] worth of goods, by transportation or from site
for i in model.destinations:
    for j in model.locations:
        model.constraints.add(model.m[i,j] + model.t[i,j] == demand[i] * model.z[i,j])
```

Η βέλτιστη λύση επαναυπολογίζεται ως :

```
------------------------------------------------
               Results
------------------------------------------------

FastVegan has to open kitchens in Location(s): 3,4,7,8
3 will serve 1936.0 portions
4 will serve 2586.0 portions
7 will serve 1626.0 portions
8 will serve 2234.0 portions



-----------
destination 1 is being served from location 3
500.0 portions by delivery
67.0 portions on site
-----------
destination 2 is being served from location 4
0.0 portions by delivery
689.0 portions on site
-----------
destination 3 is being served from location 8
0.0 portions by delivery
432.0 portions on site
-----------
destination 4 is being served from location 8
582.0 portions by delivery
0.0 portions on site
-----------
destination 5 is being served from location 4
312.0 portions by delivery
0.0 portions on site
-----------
destination 6 is being served from location 4
0.0 portions by delivery
304.0 portions on site
-----------
destination 7 is being served from location 7
0.0 portions by delivery
668.0 portions on site
-----------
destination 8 is being served from location 8
595.0 portions by delivery
0.0 portions on site
-----------
destination 9 is being served from location 3
697.0 portions by delivery
0.0 portions on site
-----------
destination 10 is being served from location 8
712.0 portions by delivery
0.0 portions on site
-----------
destination 11 is being served from location 3
0.0 portions by delivery
622.0 portions on site
-----------
destination 12 is being served from location 8
0.0 portions by delivery
446.0 portions on site
-----------
destination 13 is being served from location 3
0.0 portions by delivery
712.0 portions on site
-----------
destination 14 is being served from location 7
411.0 portions by delivery
0.0 portions on site
-----------
destination 15 is being served from location 3
0.0 portions by delivery
334.0 portions on site
-----------
destination 16 is being served from location 8
11.0 portions by delivery
653.0 portions on site
-----------
destination 17 is being served from location 4
754.0 portions by delivery
0.0 portions on site
-----------
destination 18 is being served from location 4
0.0 portions by delivery
573.0 portions on site
-----------
destination 19 is being served from location 4
341.0 portions by delivery
331.0 portions on site
-----------
destination 20 is being served from location 3
739.0 portions by delivery
0.0 portions on site
-----------
```

```
destination 21 is being served from location 7
0.0 portions by delivery
553.0 portions on site
----------
destination 22 is being served from location 4
736.0 portions by delivery
0.0 portions on site
----------
destination 23 is being served from location 7
0.0 portions by delivery
364.0 portions on site
----------
destination 24 is being served from location 8
0.0 portions by delivery
719.0 portions on site
----------
destination 25 is being served from location 4
443.0 portions by delivery
0.0 portions on site
----------
destination 26 is being served from location 3
0.0 portions by delivery
515.0 portions on site
----------
destination 27 is being served from location 7
493.0 portions by delivery
165.0 portions on site
----------
destination 28 is being served from location 7
722.0 portions by delivery
0.0 portions on site
----------
destination 29 is being served from location 4
0.0 portions by delivery
703.0 portions on site
----------
destination 30 is being served from location 8
334.0 portions by delivery
0.0 portions on site
------------------------------------------------
Total cost of Franchise: 1416845.5599999996
------------------------------------------------
```