Πανεπιστήμιο Ιωαννίνων

Τεχνική Αναφορά

Επιχειρησιακή Έρευνα

Αναστάσιος Βήττας (ΑΜ: 197)

19 Μαΐου 2025

Εισαγωγή

Το Burrito Optimization Game[1] είναι ένα εκπαιδευτικό παιχνίδι βελτιστοποίησης που αναπτύχθηκε από τον Larry Snyder και την ομάδα του, με την υποστήριξη της Gurobi. Στόχος του είναι να εισάγει τους παίκτες σε έννοιες της επιχειρησιακής έρευνας και να αναδείξει τις δυνατότητες σύγχρονων λογισμικών επίλυσης μαθηματικών μοντέλων. Το παιχνίδι μπορεί να παιχτεί ατομικά ή σε μορφή πρωταθλήματος. Στην παρούσα εργασία, καλούμαστε να αναπτύξουμε λύσεις με χρήση δύο διαφορετικών εργαλείων βελτιστοποίησης όπως τα OR-Tools και πιο συγκεκριμένα τον επιλυτή CP-SAT αλλά και με τον Gurobi, έναν από τους πιο ισχυρούς και εμπορικά διαθέσιμους επιλυτές. Με τις λύσεις που θα προκύψουν από την επίλυση του μοντέλου με Gurobi και CP-SAT, θα αξιολογήσουμε την απόδοσή τους και θα καταθέσουμε τις αντίστοιχες απαντήσεις απευθείας στο πρωτάθλημα.

BURRITO© OPTIMIZATION GAME

[1] https://www.gurobi.com/burrito-optimization-game/

Ερώτημα1 – Dataset

Κάθε μια από τις πέντε ημέρες του παιχνιδιού συνοδεύεται από τέσσερα αρχεία .csv που περιέχουν όλες τις απαραίτητες πληροφορίες για τη διαμόρφωση του προβλήματος.

1. scenDE4BD4_round1_day(1-5)_demand_node_data.csv:
   Το πρώτο αρχείο αφορά τους κόμβους ζήτησης, δηλαδή τους πελάτες που ζητούν burrito. Για κάθε πελάτη, δίνονται συντεταγμένες στο επίπεδο (x, y), ένα όνομα κτιρίου και το πόσα burrito χρειάζεται. Το σύνολο αυτών των πληροφοριών μάς λέει πού βρίσκεται κάθε πελάτης και βάση του demand πρέπει να παρθεί απόφαση για το αν θα μας προσφέρει μεγάλο κέρδος ή όχι το συγκεκριμένο κτήριο αν χτίσουμε κοντά του.

```
index,name,x,y,demand
demand11,Reinforcement Learning Puppy Training,284.83855125844076,188.79465776293824,50
demand12,KKT Air Conditioning,256.57213014119094,158.0534223706177,35
```

2. scenDE4BD4_round1_day(1-5)_demand_truck_data.csv:
   Το δεύτερο αρχείο άφορα τις πιθανές θέσεις της καντίνας, δηλαδή που μας αφήνει το παιχνίδι την κάθε μέρα ξεχωριστά να χτίσουμε σε σχέση με την ζήτηση. Ουσιαστικά το dataset περιλαμβάνει τέσσερις στήλες, η πρώτη στήλη (demand_node_index) αφορά το αναγνωριστικό του πελάτη δηλαδή του κάθε κτηριού από το προηγούμενο dataset, η δεύτερη στήλη (truck_node_index) το αναγνωριστικό της καντίνας, η τρίτη στήλη (distance) την απόσταση μεταξύ τους και η τέταρτη στήλη (scaled_demand) πόσα burrito από τη ζήτηση του συγκεκριμένου πελάτη μπορεί να καλύψειη καντίνα.

```
demand_node_index,truck_node_index,distance,scaled_demand
demand11,truck9,173.94659592403326,32
demand11,truck14,169.4054166251489,33
```

3. scenDE4BD4_round1_day(1-5)_problem_data.csv:
   Το τρίτο αρχείο αφορά την τιμή πώλησης του burrito, το κόστος παραγωγής του και το κόστος τοποθέτησης κάθε καντίνας ξεχωριστά. Οι τιμές αυτές είναι διαφορετικές σε κάθε μια από τις πέντε μέρες.

```
burrito_price,ingredient_cost,truck_cost
10,5,250
```

4. scenDE4BD4_round1_day(1-5)_truck_node_data.csv
   Το τελεύταιο άρχειο μας δίνει τις συντεταγμένες (x,y) της κάθε καντίνας που μας επιτρέπει το παιχνίδι να στήσουμε την κάθε μέρα ξεχωριστά. Παρολα αυτά δεν χρησιμοποιείται στο Ερώτημα 2 της εργασίας καθώς δεν μας έχει ζητηθεί να υλοποιήσουμε γραφικά το παιχνίδι παρά μόνο να κάνουμε την μοντελοποίηση.

```
index,x,y
truck9,218.6764886433395,145.63272120200335
truck14,150.65070595457337,222.64106844741238
```

Ερώτημα 2 – Επίλυση του προβλήματος με τον CP-SAT και τον Gurobi

Στο δεύτερο ερώτημα της εργασίας υλοποιήθηκαν δυο μοντέλα (Gurobi και CP-SAT) που επιλύουν το Burrito Optimization Game[2].

Τα αρχεία που δημιουργήθηκαν είναι 3 και είναι τα εξής:

1. read_dataset.py: Ανοίγει το dataset, το διαβάζει και κάνει parse τα δεδομένα όταν καλέσουμε την συνάρτηση load_data() που έχει γραφτεί εντός του αρχείου.
2. cpsat_burrito.py: Σε αυτό το αρχείο υπάρχει η μοντελοποίηση του προβλήματος με την χρήση του επιλυτή CP-SAT.
3. gurobi_burrito.py: Σε αυτό το αρχείο υπάρχει η μοντελοποίηση του προβλήματος με την χρήση του Gurobi.

Κατά την εκτέλεση των μοντέλων βελτιστοποίησης, χρησιμοποιούνται δεδομένα που προέρχονται από τρία διαφορετικά αρχεία csv για κάθε ημέρα ξεχωριστά. Αυτά τα δεδομένα διαβάζονται και μετατρέπονται σε δομές τύπου DataFrame μέσω της βιβλιοθήκης pandas της Python. Στην συνάρτηση load_data() δημιουργούμαι δηλαδή τρια DataFrames, πρώτα το demand_nodes, το truck_assignments, και το problem_data, όπως φαίνονται και στα Screenshots παρακάτω.

```
Demand Nodes:
       index                                  name           x           y  demand
0   demand11  Reinforcement Learning Puppy Training  284.838551  188.794658      50
1   demand12                   KKT Air Conditioning  256.572130  158.053422      35
2   demand13  Linear Regression Psychology Services  315.279312  163.642738      30
```

```
Truck Assignments:
     demand_node_index truck_node_index    distance  scaled_demand
0             demand11           truck9  173.946596             32
1             demand11          truck14  169.405417             33
2             demand11          truck17   82.605998             50
```

[2] https://www.gurobi.com/burrito-optimization-game/

```
Problem Data:
    burrito_price  ingredient_cost  truck_cost
0              10                5         250
```

Έπειτα κάνουμε split τα δεδομένα όπως τα θέλουμε εμείς, δηλαδή την κάθε στήλη που χρειαζόμαστε ξεχωριστά.

```
    price = problem_data['burrito_price'][0]
    cost = problem_data['ingredient_cost'][0]
    truck_cost = problem_data['truck_cost'][0]

    demands = demand_nodes['index'].unique()
    trucks = truck_assignments['truck_node_index'].unique()
```

Στην συνέχεια ορίζουμε μεταβλητές απόφασης κάθε τέτοια μεταβλητή είναι δυαδική και παίρνει τιμή 1 μόνο όταν πραγματοποιείται η αντίστοιχη ανάθεση. Δηλαδή δηλώνουμε με 0 ή 1 αν η κάθε καντίνα χρησιμοποιείται στη λύση ή όχι αλλά και αν η σύνδεση με καντίνας με ζήτηση είναι οκ.

```
Truck active check:  {'truck9': truck_truck9(0..1), 'truck14': truck_truck14(0..1),
```

```
Connection active or not:  {('demand11', 'truck9'): assign_demand11_truck9(0..1),
```

Χρειαζόμαστε επίσης και περιορισμούς έτσι ώστε να περιορίσουμε κάθε ζήτηση να αφορά μόνο μια καντίνα και το αντίστροφο.

```
Feasible Assignments:  [assign_demand52_truck9(0..1), assign_demand52_truck14(0..1),
```

Σκοπός μας είναι να πετύχουμε το μεγαλύτερο κέρδος με την τοποθέτηση της κάθε καντίνας. Η αντικειμενική συνάρτηση[3] που φαίνεται και στην εικόνα παρακάτω στοχεύει στην εύρεση του μέγιστου κέρδους.

$$
\begin{aligned}
\text{maximize} \quad & \sum_{i \in I} \sum_{j \in J} (r - k) \alpha_{ij} d_i y_{ij} - \sum_{j \in J} f x_j \\
\text{subject to} \quad & \sum_{j \in J} y_{ij} \leq 1 && \forall i \in I \\
& y_{ij} \leq x_j && \forall i \in I, \forall j \in J \\
& x_j, y_{ij} \in \{0, 1\} && \forall i \in I, \forall j \in J
\end{aligned}
$$

- CP-SAT:

```
model.Maximize((price - cost) * total_units - total_truck_costs)
```

[3] https://www.gurobi.com/burrito-optimization-game-guide/#the_ip_formulation

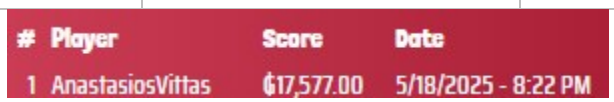
- Gurobi:

```
profit = (burrito_price - ingredient_cost) * total_units - total_truck_costs
model.setObjective(profit, GRB.MAXIMIZE)
```

Ουσιαστικά όπου $r - k$ είναι η τιμή των burrito πλην το κόστος παραγωγής τους, επείτα το πολλαπλασιάζουμε με το total_units δηλαδή $a_{ij}d_i y_{ij}$, όπου y_{ij} αν ο πελάτης i εξυπηρετείται από την καντίνα στη θέση j, πολλαπλασιασμένο με την ζήτηση του κάθε πελάτη d_i και τον συντελεστή a_{ij}, που εξαρτάται από την απόσταση της καντίνας με την ζήτηση, αν δηλαδή ο πελάτης είναι κοντά στην εκάστοτε καντίνα. Τέλος αφαιρούμε από το (price – cost) * total_units το total_truck_costs δηλαδή το $\sum_{jeJ} f x_j$, όπου f είναι η σταθερή τιμή που κοστίζει η κάθε καντίνα για να το τοποθετηθεί πολλαπλασιασμένη με 1 αν η καντίνα j χρησιμοποιείται, αλλιώς 0.[4]

Ερώτημα 3 – Σύγκριση επιλύσεων, αξιολόγηση αποτελεσμάτων

Ημέρα	Κέρδος	Χρόνος Gurobi	Χρόνος CP-SAT
1	945.00	0.0080	0.0701
2	4660.00	0.0091	0.0749
3	1632.00	0.0100	0.0637
4	2415.00	0.0040	0.0223
5	7925.00	0.0211	0.1937
Σύνολο	**17577.00**	**1.6820**	**2.3153**



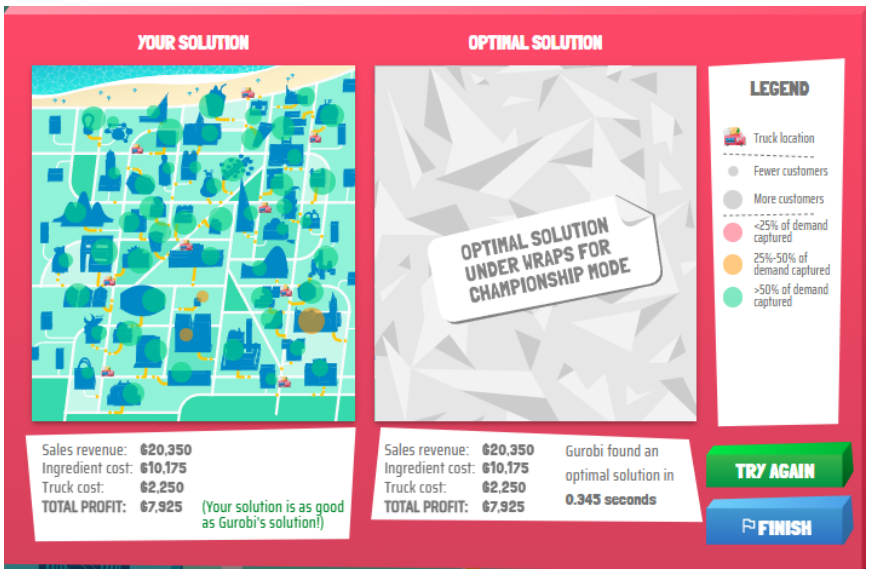
Η βασική διαφορά εντοπίζεται στον χρόνο επίλυσης. Ο Gurobi, είχε καλύτερους χρόνους σε σχέση με τον CP-SAT. Η διαφορά του ήταν στο 27% ταχύτερη από τον CP-SAT, με συνολικό χρόνο 1.682 δευτερόλεπτα έναντι 2.315 δευτερολέπτων για τον OR-Tools CP-SAT.

Αν και οι χρόνοι και των δύο είναι καλοί, η διαφορά επιδόσεων είναι εμφανής, ιδιαίτερα στις ημέρες 1, 2 και 5 όπου ο CP-SAT παρουσίασε μεγαλύτερη καθυστέρηση.

Όσον αφορά την ποιότητας λύσης, και οι δύο επιλυτές παρήγαγαν ίδια αποτελέσματα και για τις 5 ημέρες, με συνολικό κέρδος $17.577. Αυτό σημαίνει ότι η διαμόρφωση του μαθηματικού μοντέλου ήταν ισοδύναμα αποδοτική και για τους δύο επιλύτες. Το αποτελέσματα είναι optimal

[4] https://www.gurobi.com/burrito-optimization-game-guide/#the_ip_notation

συμφώνα και με την δοκιμή μας στο παιχνίδι όπου τα αποτελέσματα είναι ακριβώς ίδια με αυτά της Gurobi όπως φαίνεται και στην εικόνα.




```
Day 1
Profit: €945.00
Total Time (Gurobi): 0.0080 seconds
Active trucks: ['truck14', 'truck17', 'truck36']
Total units sold: 339
Revenue: €3390.00
Ingredient costs: €1695.00
Truck costs: €750.00

Day 2
Profit: €4660.00
Total Time (Gurobi): 0.0091 seconds
Active trucks: ['truck9', 'truck18', 'truck31', 'truck38', 'truck46', 'truck53']
Total units sold: 1232
Revenue: €12320.00
Ingredient costs: €6160.00
Truck costs: €1500.00

Day 3
Profit: €1632.00
Total Time (Gurobi): 0.0100 seconds
Active trucks: ['truck31', 'truck38', 'truck53']
Total units sold: 794
Revenue: €7940.00
Ingredient costs: €5558.00
Truck costs: €750.00

Day 4
Profit: €2415.00
Total Time (Gurobi): 0.0040 seconds
Active trucks: ['truck9', 'truck14', 'truck17', 'truck20', 'truck31', 'truck37', 'truck38', 'truck46', 'truck53']
Total units sold: 933
Revenue: €9330.00
Ingredient costs: €4665.00
Truck costs: €2250.00

Day 5
Profit: €7925.00
Total Time (Gurobi): 0.0211 seconds
Active trucks: ['truck3', 'truck5', 'truck8', 'truck18', 'truck26', 'truck31', 'truck40', 'truck45', 'truck52']
Total units sold: 2035
Revenue: €20350.00
Ingredient costs: €10175.00
Truck costs: €2250.00

============================
Total Score (5 days): €17577.00
Total Time (Gurobi): 1.6820 seconds
```

```
Day 1
Profit: €945.00
Total Time (CP-SAT): 0.0701
Active trucks: ['truck14', 'truck17', 'truck36']
Total units sold: 339
Revenue: €3390.00
Ingredient costs: €1695.00
Truck costs: €750.00

Day 2
Profit: €4660.00
Total Time (CP-SAT): 0.0749
Active trucks: ['truck9', 'truck18', 'truck31', 'truck38', 'truck46', 'truck53']
Total units sold: 1232
Revenue: €12320.00
Ingredient costs: €6160.00
Truck costs: €1500.00

Day 3
Profit: €1632.00
Total Time (CP-SAT): 0.0637
Active trucks: ['truck31', 'truck38', 'truck53']
Total units sold: 794
Revenue: €7940.00
Ingredient costs: €5558.00
Truck costs: €750.00

Day 4
Profit: €2415.00
Total Time (CP-SAT): 0.0223
Active trucks: ['truck9', 'truck14', 'truck17', 'truck20', 'truck31', 'truck37', 'truck38', 'truck46', 'truck53']
Total units sold: 933
Revenue: €9330.00
Ingredient costs: €4665.00
Truck costs: €2250.00

Day 5
Profit: €7925.00
Total Time (CP-SAT): 0.1937
Active trucks: ['truck3', 'truck5', 'truck8', 'truck18', 'truck26', 'truck31', 'truck40', 'truck45', 'truck52']
Total units sold: 2035
Revenue: €20350.00
Ingredient costs: €10175.00
Truck costs: €2250.00

============================
Total Score (5 days): €17577.00
Total Time (CP-SAT): 2.3153 seconds
```

Συμπεράσματα

Οι επιλύτες Gurobi και CP-SAT (OR-Tools) μας έδωσαν ίδιες λύσεις για όλες τις ημέρες, τόσο σε επίπεδο κέρδους όσο και στην επιλογή καντίνων. Αυτό δείχνει ότι τα μαθηματικά μοντέλα υλοποιήθηκαν σωστά. Σε ό,τι αφορά την ταχύτητα, ο Gurobi ήταν πιο γρήγορος, με συνολικό χρόνο επίλυσης 27% μικρότερο. Η διαφορά αυτή οφείλεται στον βελτιστοποιημένο αλγόριθμο επίλυσης που χρησιμοποιεί.

Πόροι Συστήματος

CPU: AMD Ryzen 7 7730U

RAM: 16GB unified DDR4

Οδηγίες Εκτέλεσης Κώδικα

Προαπαιτούμενα:

Python >= 3.8

git clone https://github.com/tasosvittas/Operations-Research.git

Δημιουργία και ενεργοποίηση περιβάλλοντος:

Windows:

cd Operations-Research\Ergasia2_OS

python -m venv erg2env

erg2env\Scripts\activate

Linux / macos:

cd Operations-Research\Ergasia2_OS

python3 -m venv erg2env

source erg2env/bin/activate

Εγκατάσταση βιβλιοθηκών:

pip install -r requirements.txt

Εκτέλεση των ερωτημάτων:

python cpsat_burrito.py

python gurobi_burrito.py