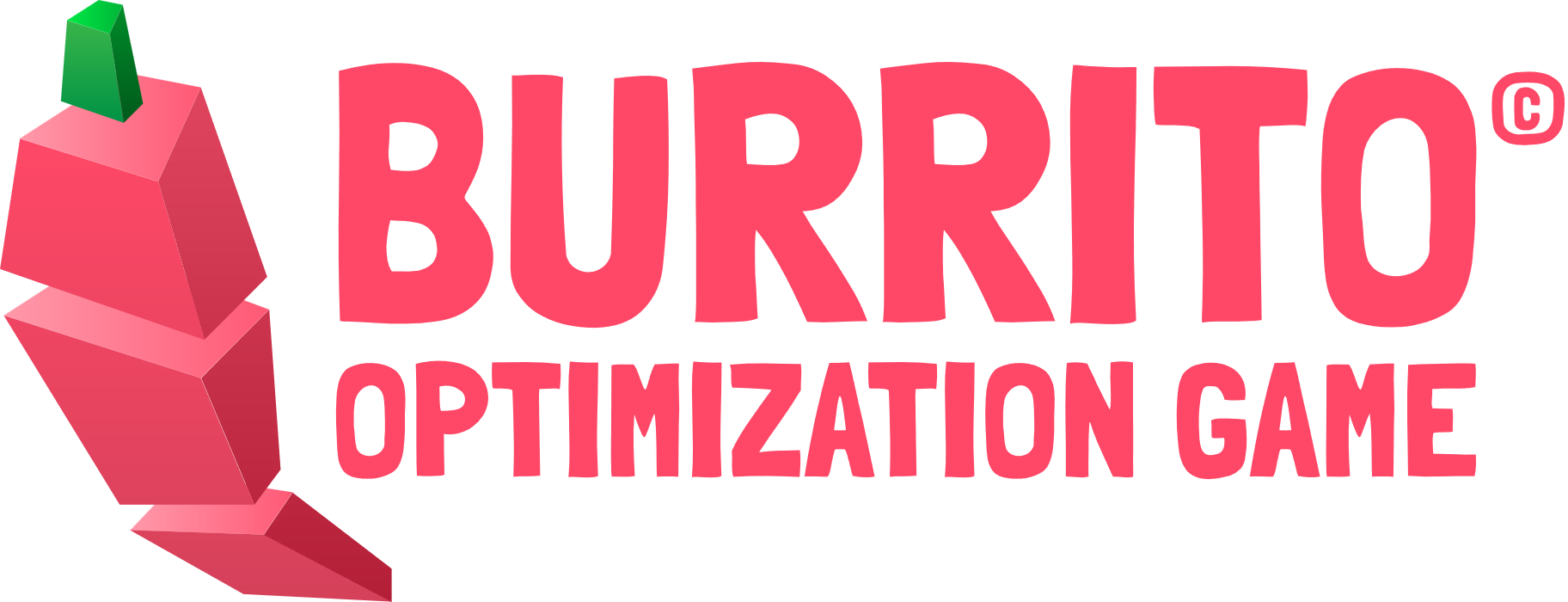
A bird with a head

AI-generated content may be incorrect.

**Τεχνική Αναφορά**  
Επιχειρησιακή Έρευνα  
Αναστάσιος Βήττας (ΑΜ: 197)  
21 Απριλίου 2025

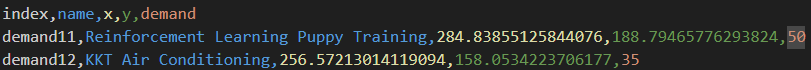
**Εισαγωγή**  
Το Burrito Optimization Game[[1]](#footnote-1) είναι ένα εκπαιδευτικό παιχνίδι βελτιστοποίησης που αναπτύχθηκε από τον Larry Snyder και την ομάδα του, με την υποστήριξη της Gurobi. Στόχος του είναι να εισάγει τους παίκτες σε έννοιες της επιχειρησιακής έρευνας και να αναδείξει τις δυνατότητες 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.



**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.



1. 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 του φορτηγού, η τρίτη στήλη (distance) την απόσταση μεταξύ τους και η τέταρτη στήλη (scaled\_demand) πόσα burrito από τη ζήτηση του συγκεκριμένου πελάτη μπορεί να καλύψει το φορτηγό.

A screenshot of a computer code

AI-generated content may be incorrect.

1. 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.



1. 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.

A screenshot of a computer

AI-generated content may be incorrect.

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

Στο πρώτο ερώτημα (erotima1.py) υλοποιήθηκε ένα μοντέλο που επιλύει το πρόβλημα

|  |  |  |
| --- | --- | --- |
| **Μέγεθος Προβλήματος** | **Συνολικό Κόστος** | **Χρόνος Επίλυσης (δευτ.)** |
| assign100.txt | 305 | 0.19 |
| assign200.txt | 475 | 0.98 |
| assign300.txt | 626 | 2.92 |
| assign400.txt | 804 | 5.99 |
| assign500.txt | 991 | 10.32 |
| assign600.txt | 1176 | 17.44 |
| assign700.txt | 1362 | 24.00 |
| assign800.txt | 1552 | 36.26 |

A screen shot of a computer screen

AI-generated content may be incorrect.

**Ερώτημα 2 – Σύγκριση με τον Ουγγρικό Αλγόριθμο**

Στη συνέχεια, συνέκρινα τα αποτελέσματα του πρώτου ερωτήματος (Μαθηματική Μοντελοποίηση) με τον Ουγγρικό αλγόριθμο, μέσα από τη βιβλιοθήκη NetworkX[[2]](#footnote-2) και πιο συγκεκριμένα τη συνάρτηση minimum\_weight\_full\_matching. Θα μπορούσα να υλοποιήσω επίσης τον αλγόριθμο με την βιβλιοθήκη που βρήκα η οποία ονομάζεται Munkres[[3]](#footnote-3).

Για να αποτυπώσω αυτές τις διαφορές, κατέγραψα τους χρόνους εκτέλεσης και το κόστος της λύσης, δημιουργώντας παράλληλα γραφήματα με την βιβλιοθήκη Matplotlib. Αυτά τα γραφήματα δείχνουν πολύ καλύτερα από τους πίνακες την διαφορά τον δυο αλγορίθμων.

|  |  |  |
| --- | --- | --- |
| **Μέγεθος Προβλήματος** | **OR-Tools (sec)** | **Hungarian (sec)** |
| assign100.txt | 0.19 | 0.20 |
| assign200.txt | 1.00 | 0.02 |
| assign300.txt | 2.80 | 0.06 |
| assign400.txt | 6.88 | 0.13 |
| assign500.txt | 11.66 | 0.18 |
| assign600.txt | 20.05 | 0.28 |
| assign700.txt | 25.78 | 0.37 |
| assign800.txt | 37.88 | 0.47 |

A screen shot of a computer program

AI-generated content may be incorrect.

A graph with a line and a dotted line

AI-generated content may be incorrect.A graph with a line and a point

AI-generated content may be incorrect.

Από τα παραπάνω φαίνεται ότι ο Ουγγρικός αλγόριθμος είναι πολύ ταχύτερος για μεγάλα προβλήματα, ειδικά από 200 εργασίες και πάνω. Και οι δύο μέθοδοι δίνουν το ίδιο κόστος, δηλαδή βρίσκουν την ίδια βέλτιστη λύση, όμως ο Ουγγρικός είναι πιο αποδοτικός χρονικά σε μεγάλα δεδομένα.

**Ερώτημα 3 – Προσθήκη Νέων Περιορισμών**

Στο τρίτο ερώτημα προστέθηκε ένας επιπλέον περιορισμός: για κάθε ομάδα 5 συνεχόμενων εργασιών, τουλάχιστον 2 από αυτές πρέπει να ανατεθούν σε εργαζόμενους της αντίστοιχης ομάδας. Αυτό έκανε το πρόβλημα πιο σύνθετο και το μοντέλο λύθηκε με Integer Programming μέσω SCIP (μέσα από OR-Tools).

Τα αποτελέσματα του Ερώτηματος 3 παρουσιάζονται στον πίνακα που ακολουθεί και αποθηκεύονται ως assign\*\_erotima3\_solution.txt.

|  |  |  |
| --- | --- | --- |
| **Μέγεθος Προβλήματος** | **Συνολικό Κόστος** | **Χρόνος Επίλυσης (δευτ.)** |
| assign100.txt | 551.0 | 0.17 |
| assign200.txt | 1063.0 | 0.82 |
| assign300.txt | 1296.0 | 2.40 |
| assign400.txt | 1872.0 | 5.16 |
| assign500.txt | 2092.0 | 9.05 |
| assign600.txt | 2906.0 | 23.07 |
| assign700.txt | 3163.0 | 22.12 |
| assign800.txt | 3601.0 | 56.92 |

**A screen shot of a computer

AI-generated content may be incorrect.**

Η προσθήκη του περιορισμού αύξησε σημαντικά το συνολικό κόστος ανάθεσης, σχεδόν στο διπλάσιο. Παρ’ όλα αυτά, ο χρόνος επίλυσης παρέμεινε σε αποδεκτά επίπεδα. Αυτό δείχνει ότι το μοντέλο είναι εφαρμόσιμο ακόμα και σε πιο πολύπλοκες εκδοχές του προβλήματος.

**Συμπεράσματα**

Η μαθηματική μοντελοποίηση με την χρήση των εργαλείων OR-Tools λύνει το πρόβλημα με ακρίβεια αλλά είναι πιο αργή σε μεγάλα δεδομένα σε αντίθεση με τον Ουγγρικό αλγόριθμο που είναι πολύ πιο γρήγορος. Το μοντέλο με περιορισμό ανάθεσης σε ομάδες έχει δικαίως πιο αυξημένο κόστος καθώς οι περιορισμοί «δυσκολεύουν» τον επιλυτή.

**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 shdahsdhas.py

1. <https://www.gurobi.com/burrito-optimization-game/> [↑](#footnote-ref-1)
2. <https://networkx.org/documentation/stable/reference/algorithms/generated/networkx.algorithms.bipartite.matching.minimum_weight_full_matching.html> [↑](#footnote-ref-2)
3. https://software.clapper.org/munkres/ [↑](#footnote-ref-3)