ΖΩΓΡΑΦΟΥ ΜΑΡΙΑ-ΝΙΚΗ

ΑΜ: 1096060

Εαρινό Εξάμηνο 2024-2025

ECE\_ΓΚ806

Βελτιστοποίηση Χρονοπρογραμματισμού Παρατηρήσεων Δορυφόρου με χρήση Γραμμικού Και Συνδυαστικού Προγραμματισμού

Τελικό Project – Γραμμική και Συνδυαστική Βελτιστοποίηση

Table of Contents

[Εισαγωγή 2](#_Toc201073688)

[Μοντελοποίηση Προβλήματος 3](#_Toc201073689)

[Σύνολο παραμέτρων 3](#_Toc201073690)

[Μεταβλητές Απόφασης 3](#_Toc201073691)

[Objective Function – Αντικειμενική Συνάρτηση 3](#_Toc201073692)

[Περιορισμοί 4](#_Toc201073693)

[Περιγραφή Υλοποίησης 5](#_Toc201073694)

[Γεννήτρια ευκαιριών παρατήρησης 5](#_Toc201073695)

[Ανάλυση Αποτελεσμάτων 6](#_Toc201073696)

[Ανάλυση Ευαισθησίας 7](#_Toc201073697)

[Βιβλιογραφία 8](#_Toc201073698)

# Εισαγωγή

* Για το τελικό project του μαθήματος θα πρέπει να γράψεται μια αναφορά 10-15 σελίδων, όπου θα περιγράφεται εκτενώς το πρόβλημα σας, η σκοπιμότητα του ή οι εφαρμογές του, θα περιγράφεται αναλυτικά η μοντελοποίηση του (ορισμός μεταβλητών, παραμέτρων, περιορισμών, συναρτήσεων, κ.λ.π), η περιγραφή της επίλυσης, του κώδικα που γράφτηκε και του output με τη λύση, σχολιασμός της λύσης ή των λύσεων. Φροντίστε να «παίξετε» επαρκώς με το μοντέλο σας σε διαφορετικού μεγέθους προβλήματα ή με αλλαγές στις τιμές των παραμέτρων (ανάλυση ευαισθησίας) ή με εναλλακτικές μεθόδους επίλυσης του προβλήματος και σύγκρισή τους. Στο τέλος θα αναγράφεται πλήρως η βιβλιογραφία που χρησιμοποιήσατε. Για την παρουσίαση του project να υπολογίσετε maximum 20 λεπτά, συμπεριλαμβανομένων των ερωτήσεων.
* Περιγραφή του προβλήματος: Τι είναι ο χρονοπρογραμματισμός παρατηρήσεων.
* Πού χρησιμοποιείται (παρακολούθηση δασών, καταστροφές, γεωργία).
* Γιατί είναι δύσκολο (περιορισμένη ορατότητα, ενέργεια, μνήμη, συγκρούσεις).
* Αναφορά στους στόχους του project:
  + Υλοποίηση MILP μοντέλου με ελεύθερους solvers.

Ανάλυση απόδοσης με βάση το μέγεθος προβλήματος και παραμέτρους

# Μοντελοποίηση Προβλήματος

## Σύνολο παραμέτρων

Στόχοι παρατήρησης Τ = {targets}

Δορυφόροι S = {sattelites}

Ευκαιρία Παρατήρησης O = {ευκαιρίες παρατήρησης}, δηλαδή οι πιθανές υποψήφιες παρατήρησεις κάποιου στόχου.

Παράμετροι του συστήματος:

Χρονικός ορίζοντας, διαθέσιμη μνήμη, ισχύς, απαιτούμενη διάρκεια παρατήρησης

## Μεταβλητές Απόφασης

Ορίζουμε ως μεταβλητή απόφασης x την επιλογή αν θα γίνει η παρατήρηση ή όχι:

Αν η παρατήρηση θα εκτελεστεί, αν τότε όχι.

## Objective Function – Αντικειμενική Συνάρτηση

Θέτουμε σαν Αντικειμενική Συνάρτηση το μέγιστο άθροισμα **σταθμισμένων\*** παρατηρήσεων:

*\*Σημείωση:* Σε αντίθεση με το paper[1] που χρησιμοποιεί τρεις δείκτες (i,j,k), εδώ χρησιμοποιήθηκε ένας που αντιστοιχεί και στους 3.

Το βάρος υπολογίζεται ως:

Υπάρχει η επιλογή το βάρος να συνυπολογίζει τον βαθμό σύγκρουσης και να υπάρχει ποινή για τις συγκρούσεις:

Ποινή σύγκρουσης():

*priority(target\_i)*: Προτεραιότητα του στόχου που παρατηρείται

*elevation\_i:* Γωνία ανύψωσης της παρατήρησης (όσο μεγαλύτερη, τόσο καλύτερα)

*data\_volume\_i*: Όγκος δεδομένων που παράγεται από την παρατήρηση (όσο μικρότερος, τόσο πιο αποδοτικά)

**Βαθμός σύγκρουσης (Conflict degree):** Αριθμός άλλων παρατηρήσεων που έρχονται σε σύγκρουση με την παρατήρηση i

## Περιορισμοί

**Σύγκρουση χρόνου:**

Δεν μπορούν να εκτελεστούν ταυτόχρονα παρατηρήσεις στο ίδιο δορυφόρο.

(Προαιρετικό) **Setup time**: Επιπλέον γίνεται έλεγχος για το αν το κενό χρονικό διάστημα μεταξύ των δύο παρατηρήσεων είναι επαρκές για να γίνει το set up του δορυφόρου:

Όπουη στιγμή έναρξης των παρατηρήσεων i και j και το timeGap προκύπτει από τις γωνιακές ταχύτητες του δορυφόρου και τον χρόνο σταθεροποίησης (setup).

**Χωρητικότητα μνήμης:**

Το άθροισμα δεδομένων δεν πρέπει να υπερβαίνει τον χώρο μνήμης.

Όπου ο όγκος δεδομένων και η συνολική μνήμη και το σύνολο των υποψήφιων παρατηρήσεων.

**Κατανάλωση ισχύος ανά παράθυρο**:

Όπου P το όριο μέγιστης κατανάλωσης, η ισχύς που απαιτεί κάθε παρατήρηση και το σύνολο των παρατηρήσεων στο χρονικό παράθυρο w. Το constraint αυτό εφαρμόζεται ανά χρονικό παράθυρο, συνήθως κάθε 4 ώρες, ώστε να προσεγγίζει την κατανάλωση ενέργειας βάσει των δυνατοτήτων των ηλιακών πάνελ ή των αποθεμάτων.

**Περιορισμοί Κάλυψης Στόχων (Προαιρετικό):**

Για κάθε στόχο *t* (το πολύ μία παρατήρηση ανά στόχο):

Όπου οι παρατηρήσεις για τον στόχο *t*.

# Περιγραφή Υλοποίησης

Η υλοποίηση έγινε με χρήση Python + PuLP + CBC solver.

## Δομές

Δημιούργησα τις κλάσεις **Satellite**, **Target** και **Observation** για να περιγράψω τις βασικές παραμέτρους του προβλήματος. Ως Δορυφόρος (Satellite) ορίζεται κάθε δορυφόρος του συστήματος μας και ως Target ο οποιοσδήποτε στόχος παρατήρησής υπάρχει. Τέλος, ως Observation ονομάζεται κάθε υποψήφια ευκαιρία παρατήρησης κάποιου στόχου.

Επιπλέον δημιούργησα την κλάση **SatelliteScheduler** που αναλαμβάνει τη δημιουργία και την επίλυση του προβλήματος με Mixed integer linear programming. Η κλάση αυτή συμπεριλαμβάνει τις εξής λειτουργίες:

## Γεννήτρια ευκαιριών παρατήρησης

Για κάθε ζεύγος δορυφόρου-στόχου θα δημιουργηθεί μια υποψήφια παρατήρηση με την συνάρτηση:

def generate\_observation\_opportunities(self, start\_time: datetime, time\_step: int = 10) -> List[Observation]:

Κάθε ευκαιρία (opportunity) είναι μια τριάδα (tuple) η οποία στη συνέχεια αντιστοιχεί σε μία μεταβλητή απόφασης στο μοντέλο MILP (στο άρθρο αντίστοιχο θα ήταν το ).

Αυτή η συνάρτηση δημιουργεί το πλήρες σύνολο υποψήφιων αποφάσεων για το μοντέλο MILP. Κάθε μία αντιστοιχεί σε ένα παράθυρο παρατήρησης με συσχετισμένους περιορισμούς (διάρκεια, ισχύς, μνήμη κ.λπ.).

Η παραγωγή ευκαιριών γίνεται προσομοιώνοντας την τροχιά των δορυφόρων με στοχαστικό τρόπο. Συγκεκριμένα, για κάθε δύο ώρες στο χρονικό ορίζοντα, κάθε δορυφόρος έχει πιθανότητα 30% να μπορεί να παρατηρήσει έναν στόχο. Αν συμβεί αυτό, τότε επιλέγεται τυχαία:

* **χρόνος έναρξης**: εντός του 2ώρου, με τυχαία μετατόπιση έως 119 λεπτά,
* **διάρκεια παρατήρησης**: μεταξύ 5 και 15 λεπτών,
* **ύψος ανύψωσης (elevation)**: μεταξύ της ελάχιστης επιτρεπτής και 85°.

for hour in range(0, self.time\_horizon, 2): *# κάθε 2 ώρες*

*# Simulate orbital mechanics with some randomness - δορυφόρος σε τροχιά*

*# Assume visibility every 2 hours with some random variation*

    if random.random() < 0.3:  *# 30% chance of visibility*

        start = start\_time + timedelta(hours=hour,

                                        minutes=random.randint(0, 119))

        duration = random.randint(5, 15)  *# 5-15 minutes*

        end = start + timedelta(minutes=duration)

Η χρονική στιγμή ολοκλήρωσης της παρατήρησης προκύπτει προσθέτοντας τη διάρκεια στον χρόνο έναρξης. Η απαιτούμενη μνήμη (data volume) και ισχύς (power) για την παρατήρηση υπολογίζονται βάσει της διάρκειας και των τεχνικών χαρακτηριστικών του δορυφόρου.

Το αποτέλεσμα είναι ένα ρεαλιστικά απλοποιημένο σύνολο από παράθυρα παρατήρησης, τα οποία στη συνέχεια περνούν στον βελτιστοποιητή ως υποψήφιες αποφάσεις. Παράλληλα, εάν έχει ενεργοποιηθεί η επιλογή use\_conflict\_degree, η μέθοδος υπολογίζει και τον βαθμό σύγκρουσης κάθε παρατήρησης με τις υπόλοιπες (μέσω της \_calculate\_conflict\_degrees).

## Έλεγχος χρονικών συγκρούσεων

Για κάθε ζεύγος παρατηρήσεων χρειάζεται να ελέγξουμε αν υπάρχει χρονική σύγκρουση, λαμβάνοντας υπόψη και τον χρόνο προετοιμασίας (setup time) του δορυφόρου. Αυτό γίνεται με τη συνάρτηση:

def check\_temporal\_conflict\_with\_setup(self, obs1: Observation, obs2: Observation)

Η συνάρτηση αυτή εντοπίζει αν δύο παρατηρήσεις (που αφορούν τον ίδιο δορυφόρο) είτε επικαλύπτονται χρονικά είτε δεν διαθέτουν επαρκές χρονικό κενό μεταξύ τους για να πραγματοποιηθεί η απαιτούμενη αλλαγή γωνίας/κατεύθυνσης του δορυφόρου.

Ο έλεγχος περιλαμβάνει τρεις περιπτώσεις:

1. Οι παρατηρήσεις επικαλύπτονται → υπάρχει σύγκρουση,
2. Η πρώτη τελειώνει πριν ξεκινήσει η δεύτερη → αν το χρονικό διάστημα μεταξύ τους είναι μικρότερο από τον setup time → σύγκρουση,
3. Το ίδιο και με την αντίστροφη σειρά.

Η συνάρτηση επιστρέφει True όταν υπάρχει σύγκρουση, ώστε στη συνέχεια να επιβληθεί περιορισμός στο MILP του τύπου:

που αποτρέπει την ταυτόχρονη επιλογή των συγκρουόμενων παρατηρήσεων.

Υπολογισμός βαθμού σύγκρουσης

Για κάθε παρατήρηση, υπολογίζεται ο **βαθμός σύγκρουσης (conflict degree)** με τη συνάρτηση:

def \_calculate\_conflict\_degrees(self):

Ο βαθμός σύγκρουσης μετράει **πόσες άλλες παρατηρήσεις χρονικά συγκρούονται** με τη συγκεκριμένη, λαμβάνοντας υπόψη και τον **χρόνο προετοιμασίας (setup time)** του δορυφόρου. Για κάθε παρατήρηση obs, η συνάρτηση ελέγχει όλες τις υπόλοιπες other\_obs. Αν υπάρχει χρονική σύγκρουση (μέσω check\_temporal\_conflict\_with\_setup), τότε αυξάνεται ο μετρητής συγκρούσεων conflict\_count.

Το αποτέλεσμα αποθηκεύεται ως: self.conflict\_degrees[i] = conflict\_count όπου i είναι το index της παρατήρησης.

## Δημιουργία του MILP μοντέλου

Η συνάρτηση:

def build\_milp\_model(self):

δημιουργεί και διατυπώνει το MILP μοντέλο που χρησιμοποιείται για τον βελτιστοποιημένο προγραμματισμό παρατηρήσεων από δορυφόρους. Κάθε υποψήφια παρατήρηση (Observation) αντιστοιχεί σε μία δυαδική μεταβλητή απόφασης , η οποία δηλώνει αν η παρατήρηση επιλέγεται στο τελικό πρόγραμμα ή όχι.

Προσθέτουμε στο μοντέλο την συνάρτηση στόχου και τους περιορισμούς και το αποθηκεύουμε στην κλάση (self.model = pulp.LpProblem(...))

## Δημιουργία του χρονικών παραθύρων κατανάλωσης ισχύος

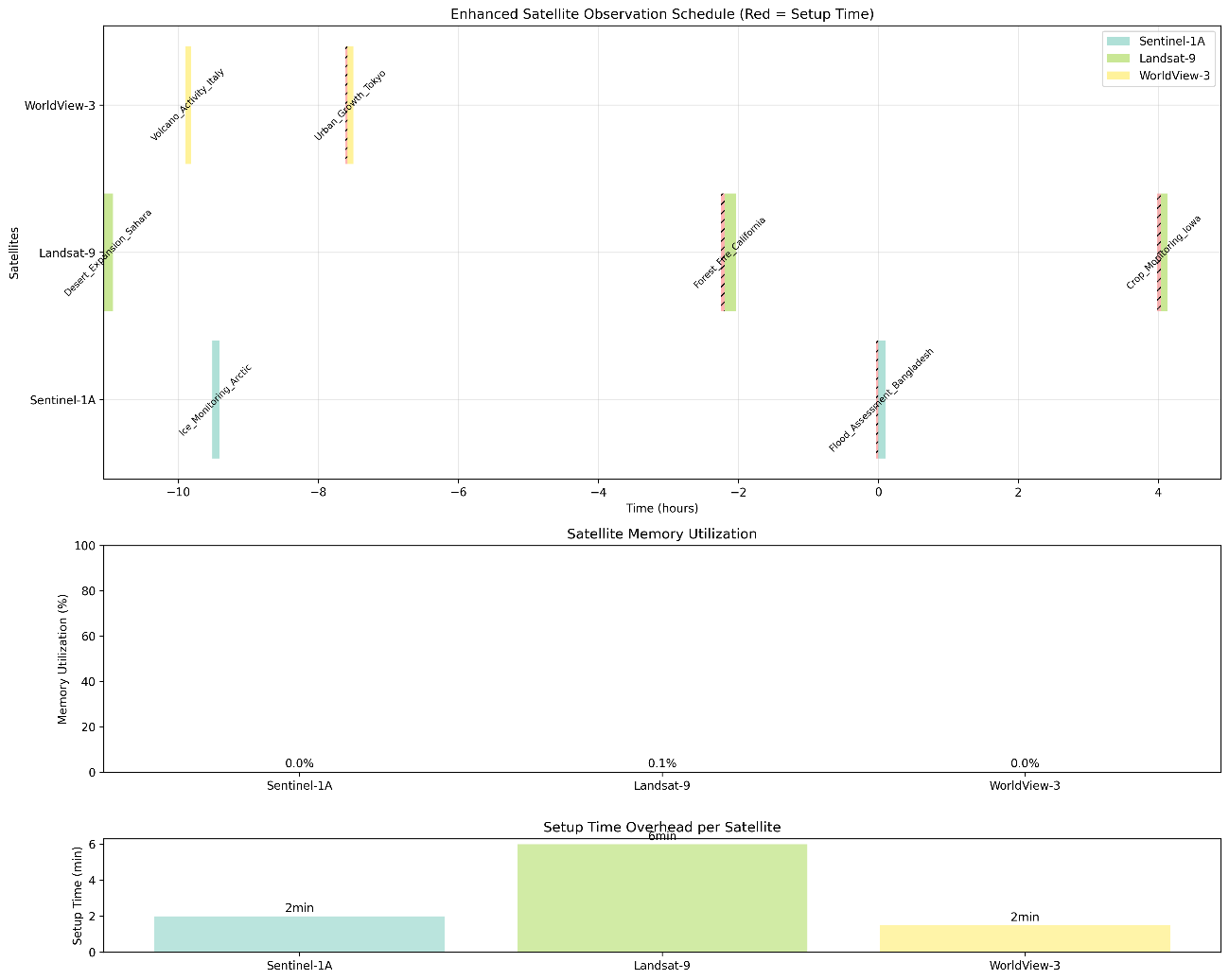
Η συνάρτηση αυτή δημιουργεί διαδοχικά χρονικά παράθυρα (time windows) στα οποία θα εφαρμόζεται ο περιορισμός ισχύος (power capacity constraint) για κάθε δορυφόρο.

Κάθε παράθυρο είναι ένα διάστημα π.χ. 4 ωρών (προεπιλεγμένο), και η συνολική ισχύς των παρατηρήσεων που εκτελούνται μέσα σε αυτό το διάστημα δεν πρέπει να ξεπερνάει την ενεργειακή δυνατότητα του δορυφόρου.

Πώς εφαρμόζονται οι περιορισμοί.

Πώς γίνεται η επίλυση + visual output.

# Ανάλυση Αποτελεσμάτων



**5. Ανάλυση Αποτελεσμάτων (3–4 σελ.)**

**Πειραματικά Σενάρια**

* 3–4 διαφορετικά σενάρια:
  + 3, 5, 7 δορυφόροι
  + 10, 20, 50 στόχοι
  + 24h vs 48h scheduling window

**Δείξε:**

* Πλήθος παρατηρήσεων που προγραμματίστηκαν.
* Ποσοστά κάλυψης στόχων.
* Χρόνοι εκτέλεσης.
* Μνήμη & ενέργεια που χρησιμοποιήθηκε ανά δορυφόρο.
* Οπτικοποίηση με Gantt chart & bar charts.

# Ανάλυση Ευαισθησίας

**Ανάλυση Ευαισθησίας**

* Τι αλλάζει αν αυξήσεις π.χ. την ισχύ ή τη μνήμη;
* Τι αλλάζει αν έχεις πιο «πυκνές» ευκαιρίες παρατήρησης;

# Βιβλιογραφία

[1]. A Mixed Integer Linear Programming Model for Multi-Satellite Scheduling, Chen, X., Reinelt, G., Dai, G., & Spitz, A.

2. Selecting and scheduling observations of agile satellites, Lemaître, M., Verfaillie, G., Jouhaud, F., Lachiver, J.-M., & Bataille, N.