Βελτιστοποίηση Χρονοπρογραμματισμού Παρατηρήσεων Δορυφόρου με χρήση Γραμμικού Και Συνδυαστικού Προγραμματισμού

Τελικό Project – Γραμμική και Συνδυαστική Βελτιστοποίηση

Εαρινό Εξάμηνο 2024-2025

ECE_ΓΚ806

ΖΩΓΡΑΦΟΥ ΜΑΡΙΑ-ΝΙΚΗ
ΑΜ: 1096060

Table of Contents

Εισαγωγή

Η παρατήρηση της Γης μέσω δορυφόρων (Earth-observing satellites – EOS) αποτελεί κρίσιμο εργαλείο που διαδραματίζει ολοένα και σημαντικότερο ρόλο στην εξερεύνηση φυσικών πόρων, στην έγκαιρη προειδοποίηση για φυσικές καταστροφές, στην ανάλυση περιβαλλοντικής υποβάθμισης και σε πολλές άλλες εφαρμογές απεικόνισης[1]. Εφαρμογές όπως η παρακολούθηση δασικών πυρκαγιών, η διαχείριση φυσικών καταστροφών, η γεωργική αξιολόγηση και η παρακολούθηση πολεοδομικής ανάπτυξης εξαρτώνται σε μεγάλο βαθμό από τη δυνατότητα των δορυφόρων να συλλέγουν έγκαιρα και ποιοτικά δεδομένα.

Κάθε δορυφόρος, όμως, υπόκειται σε φυσικούς και τεχνικούς περιορισμούς: έχει περιορισμένο χρόνο ορατότητας προς έναν στόχο (ανάλογα με την τροχιά του), πεπερασμένους ενεργειακούς και αποθηκευτικούς πόρους, καθώς και ανάγκες για προετοιμασία (setup time) μεταξύ διαδοχικών παρατηρήσεων. Επιπλέον, το πλήθος των πιθανών στόχων παρατήρησης είναι συνήθως πολύ μεγαλύτερο από αυτό που μπορεί να εξυπηρετηθεί πλήρως. Αυτά τα χαρακτηριστικά καθιστούν το πρόβλημα χρονοπρογραμματισμού παρατηρήσεων εξαιρετικά σύνθετο, τόσο υπολογιστικά όσο και επιχειρησιακά.

Σκοπός της παρούσας εργασίας είναι η μελέτη και υλοποίηση ενός μοντέλου βελτιστοποίησης για τον χρονοπρογραμματισμό παρατηρήσεων από πολλαπλούς δορυφόρους, με στόχο τη μέγιστη αξιοποίηση των διαθέσιμων πόρων και τη βέλτιστη κάλυψη στόχων. Η εργασία είναι εμπνευσμένη από το επιστημονικό άρθρο των Chen, Reinelt και Spitz [1], όπου προτείνεται ένα εκτεταμένο Mixed Integer Linear Programming (MILP) μοντέλο για τον χρονοπρογραμματισμό πολλαπλών δορυφόρων (Multi-Satellite Scheduling).

Για τις ανάγκες του μαθήματος, υλοποιείται μια απλοποιημένη αλλά ρεαλιστική εκδοχή του μοντέλου, η οποία περιλαμβάνει κρίσιμες πτυχές όπως ο περιορισμός μνήμης, ισχύος, setup time μεταξύ παρατηρήσεων, καθώς και την έννοια του conflict degree (βαθμός σύγκρουσης), η οποία εισάγει ένα ακόμα κριτήριο ταξινόμησης των παρατηρήσεων. Η βελτιστοποίηση πραγματοποιείται μέσω γραμμικού και συνδυαστικού προγραμματισμού με χρήση της βιβλιοθήκης PuLP σε Python και του CBC solver.

Η εργασία οργανώνεται ως εξής: Αρχικά παρουσιάζεται η μαθηματική μοντελοποίηση του προβλήματος και οι σχετικοί περιορισμοί, ακολουθεί η αναλυτική περιγραφή της υλοποίησης σε Python, ενώ στη συνέχεια παρουσιάζονται τα αποτελέσματα για διάφορα σενάρια προσομοίωσης.

Μοντελοποίηση Προβλήματος

Σύνολο παραμέτρων

Στόχοι παρατήρησης T = {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:

$$x_i \in \{0, 1\} \; \forall \; \pi\iota\theta\alpha\nu\acute{\eta} \; \pi\alpha\rho\alpha\tau\acute{\eta}\rho\eta\sigma\eta$$

Αν $x_i = 1$ η παρατήρηση θα εκτελεστεί, αν $x_i = 0$ τότε όχι.

Objective Function – Αντικειμενική Συνάρτηση

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

$$max \sum w_i * x_i$$

*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:

$$w_i = \frac{priority(target_i) * elevation_i}{90.0 * dataVolume_i + 0.1}$$

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

Ποινή σύγκρουσης($conflict\_penalty_i$):

$$cp_i = 1 + conflict\_degree_i$$

$$w_i' = \frac{w_i}{cp_i}$$

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)

cp_i: **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.

$$x_i + x_j \leq 1$$

(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:

$$|\tau_j - \tau_i| \geq \mathrm{timeGap}$$

Όπουτ_i, τ_j η 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.

$$\sum_{i \in O} u_i * x_i \leq M$$

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

Κατανάλωση ισχύος ανά παράθυρο:

$$\sum_{i \in O_w} p_i * x_i \leq P$$

Όπου P το όριο μέγιστης κατανάλωσης, p_i η ισχύς που απαιτεί κάθε παρατήρηση και O_w 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):

$$\sum_{i \in S_t} x_i \leq 1$$

Όπου S_t 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) $\{target, sattelite, time\ window\}$η οποία στη συνέχεια αντιστοιχεί σε μία μεταβλητή απόφασης στο μοντέλο MILP x_i (στο άρθρο αντίστοιχο θα ήταν το x_{ij}^k).

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:

$$x_i + x_j \leq 1$$

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 $x_i \in \{0,1\}$, 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

```
def _create_time_windows(self, window_size_hours: int = 4)
```

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. Ορίζεται ως αρχικής χρονικής στιγμής το μεσάνυχτο της ημέρας της πρώτης παρατήρησης

Επίλυση – Έλεγχος και Παρουσίαση Αποτελεσμάτων

Με την συνάρτηση `solve` επιλύνεται το μοντέλο και ύστερα με την `validate_schedule_constraints` γίνεται δεύτερος έλεγχος ότι η τελική λύση τηρεί όλους τους περιορισμούς. Με την συνάρτηση `analyze_solution` τυπώνονται με ξεκάθαρο τρόπο τα αποτελέσματα και τα στατιστικά επίλυσης. Με την συνάρτηση `visualize_schedule` γίνεται γραφική απεικόνιση της λύσης.

Παραγωγή Σεναρίων

Όπως θα αναλυθεί στην παρουσίαση αποτελεσμάτων δίνεται η επιλογή στον χρήστη να δημιουργήσει σενάριο με custom **αριθμό δορυφόρων**, **στόχων** και **χρονικού ορίζοντα** ή να φορτώσει σενάριο από json αρχείο ή να τρέξει ένα default παράδειγμα.

```
def scenario_picker(x :int):
```

Σημείωση: Για την επίλυση των προβλημάτων υπάρχει χρονικό όριο 300 δευτερολέπτων.

Ανάλυση Αποτελεσμάτων

Σενάριο 1:

Εκτέλεση του προγράμματος `satellite_scheduler.py` για μεγάλο custom σενάριο:

```
=== Enhanced Satellite Observation Scheduling using MILP ===
Features: Setup Time Constraints + Conflict Degree Weighting

Chose mode:
1. Run example scenario
2. Run scenarios from JSON file PREMADE
3. Generate custom scenario
3
Enter scenario name (e.g., 'custom_scenario') or leave empty for default: test1
Enter number of satellites (e.g., 3): 7
Enter number of targets (e.g., 5): 35
Enter scheduling time horizon in hours (default 24): 48
Created scenario with 7 satellites and 35 targets
  Sat-1: Setup time = 1.8 minutes
  Sat-2: Setup time = 3.0 minutes
  Sat-3: Setup time = 2.3 minutes
  Sat-4: Setup time = 2.5 minutes
  Sat-5: Setup time = 2.9 minutes
  Sat-6: Setup time = 2.8 minutes
  Sat-7: Setup time = 3.4 minutes
Calculated conflict degrees - Max: 8, Min: 0, Avg: 2.13
Generated 855 observation opportunities
Enhanced model built with 855 potential observations
Found 912 temporal conflicts (including setup time)
Conflict degree weighting: ENABLED
Solving enhanced MILP model...
Solution status: Optimal
Optimal objective value: 209.455
Selected 35 observations

[**TIMER**] MILP solve time: 0.58 seconds
Schedule validation: All setup time constraints satisfied ✓
```

```
=== Enhanced Solution Analysis ===
Total observations scheduled: 35
Total data volume: 0.09 GB

Conflict Degree Statistics:
  Average conflict degree: 0.77
  Max conflict degree: 2
  Min conflict degree: 0

Enhanced Satellite Statistics:
  Sat-1:
    Observations: 4
    Data volume: 0.01 GB
    Observation time: 32.0 minutes
    Time with setup: 37.4 minutes
```

```
    Setup overhead: 5.4 minutes
    Memory utilization: 0.1%
  Sat-2:
    Observations: 9
    Data volume: 0.02 GB
    Observation time: 78.0 minutes
    Time with setup: 102.0 minutes
    Setup overhead: 24.0 minutes
    Memory utilization: 0.1%
  Sat-3:
    Observations: 2
    Data volume: 0.01 GB
    Observation time: 18.0 minutes
    Time with setup: 20.3 minutes
    Setup overhead: 2.3 minutes
    Memory utilization: 0.0%
  Sat-4:
    Observations: 6
    Data volume: 0.01 GB
    Observation time: 55.0 minutes
    Time with setup: 67.5 minutes
    Setup overhead: 12.5 minutes
    Memory utilization: 0.1%
  Sat-5:
    Observations: 6
    Data volume: 0.02 GB
    Observation time: 55.0 minutes
    Time with setup: 69.5 minutes
    Setup overhead: 14.5 minutes
    Memory utilization: 0.2%
  Sat-6:
    Observations: 5
    Data volume: 0.01 GB
    Observation time: 50.0 minutes
    Time with setup: 61.2 minutes
    Setup overhead: 11.2 minutes
    Memory utilization: 0.1%
  Sat-7:
    Observations: 3
    Data volume: 0.01 GB
    Observation time: 35.0 minutes
    Time with setup: 41.8 minutes
    Setup overhead: 6.8 minutes
    Memory utilization: 0.1%
```

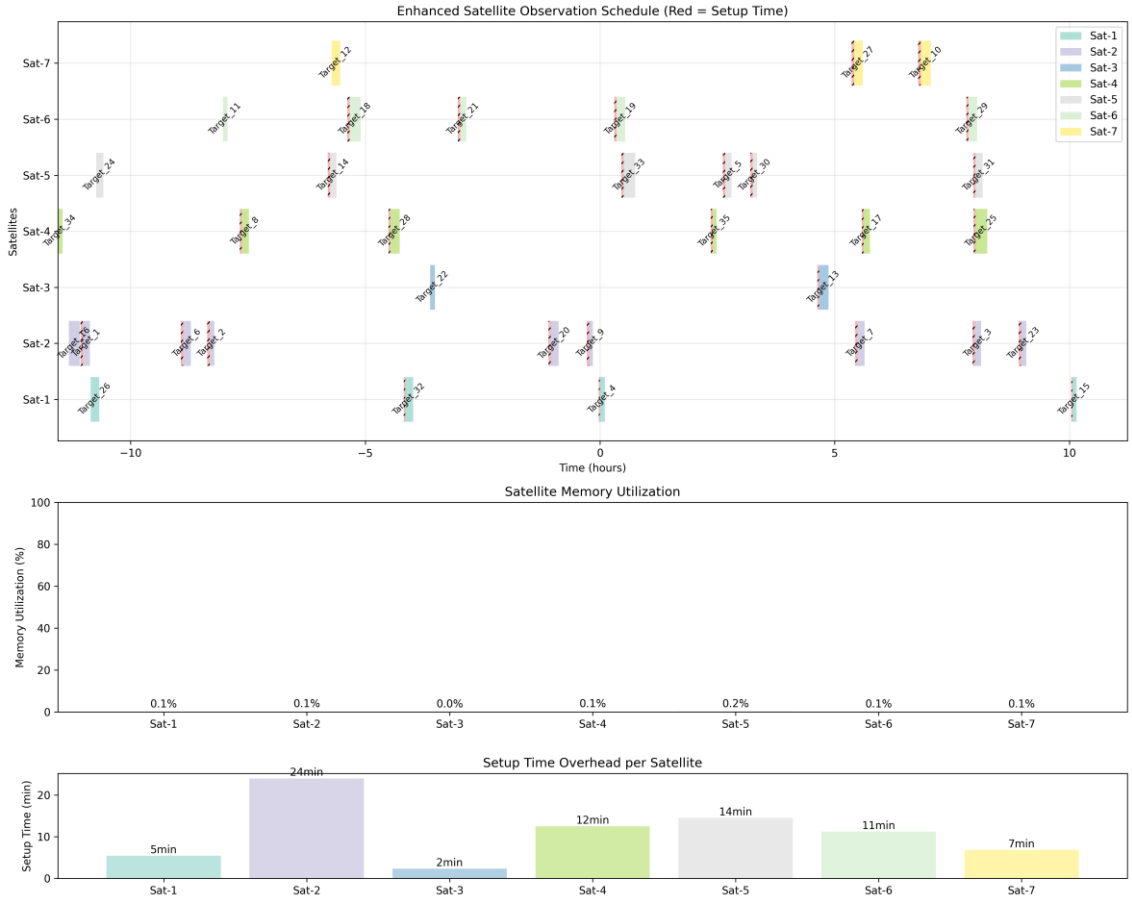
Σχολιασμός:
Για το προσαρμοσμένο σενάριο με **7 δορυφόρους**, **35 στόχους** και **διάρκεια 48 ωρών**, η διαδικασία προγραμματισμού ολοκληρώθηκε επιτυχώς με τα εξής αποτελέσματα:

Παραχθήκαν 855 πιθανά παράθυρα παρατήρησης, **εντοπίστηκαν** 912 χρονικές συγκρούσεις (με συνυπολογισμό του χρόνου προετοιμασίας), η **MILP επίλυση** ολοκληρώθηκε σε **0.58 δευτερόλεπτα** και το μοντέλο επέλεξε **35 παρατηρήσεις** — μία για κάθε στόχο, χωρίς παραβίαση περιορισμών. Ο συνολικός όγκος δεδομένων ήταν **0.09 GB.** Κάθε δορυφόρος έκανε από 2 έως 9 παρατηρήσεις, με πολύ χαμηλή χρήση μνήμης (0.0%–0.2%). Όλοι οι χρονικοί περιορισμοί και οι χρόνοι προετοιμασίας **τηρήθηκαν πλήρως**.

Κάλυψη στόχων: 100% (και οι 35 παρατηρήθηκαν).

Συμπέρασμα:
Το σύστημα διαχειρίστηκε αποτελεσματικά ένα μεσαίας πολυπλοκότητας σενάριο με πλήρη συμμόρφωση σε περιορισμούς και εξαιρετικά χαμηλό υπολογιστικό κόστος.

Γραφική απεικόνιση:



Σενάριο 2

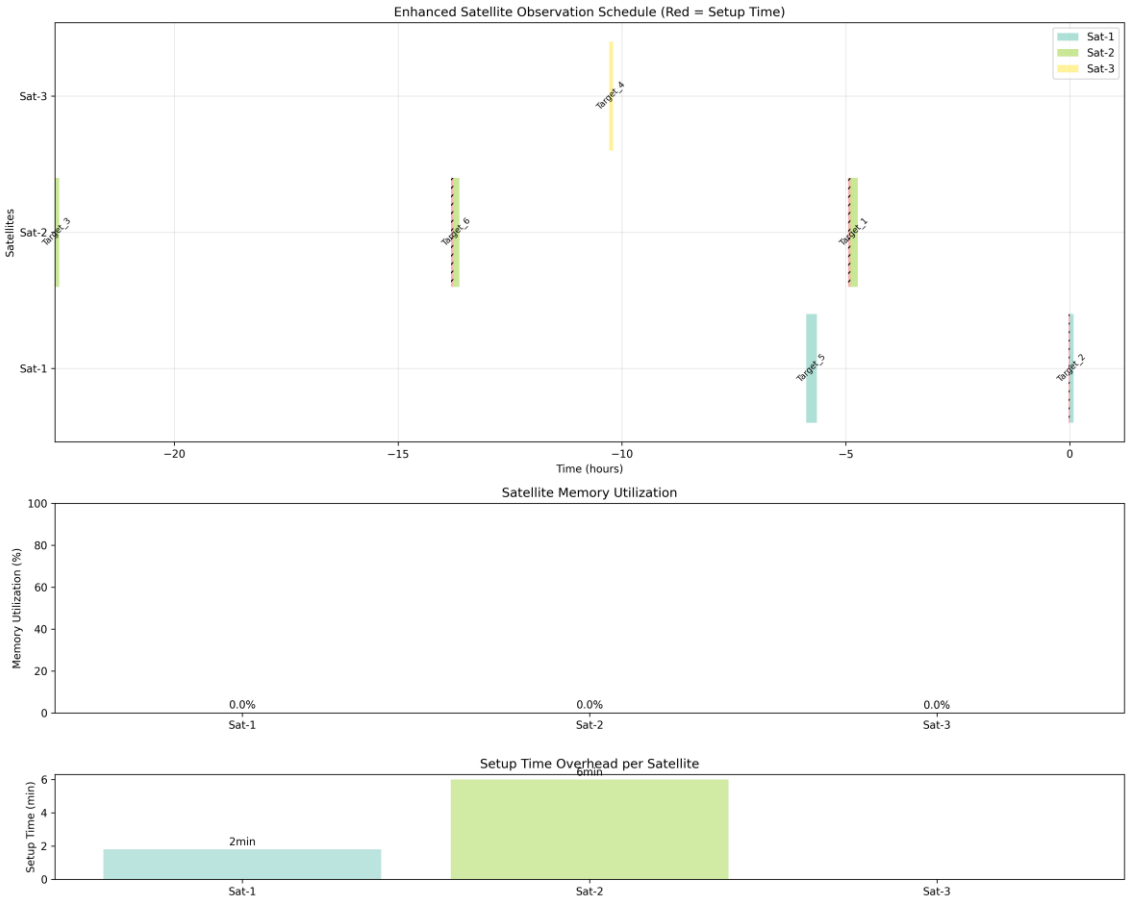
Εκτέλεση του προγράμματος `satellite_scheduler.py` για μικρό custom σενάριο:

```
=== Enhanced Satellite Observation Scheduling using MILP ===
Features: Setup Time Constraints + Conflict Degree Weighting

Chose mode:
1. Run example scenario
2. Run scenarios from JSON file PREMADE
3. Generate custom scenario
3
Enter scenario name (e.g., 'custom_scenario') or leave empty for default: _test2
Enter number of satellites (e.g., 3): 3
Enter number of targets (e.g., 5): 6
Enter scheduling time horizon in hours (default 24): 24
Created scenario with 3 satellites and 6 targets
  Sat-1: Setup time = 1.8 minutes
  Sat-2: Setup time = 3.0 minutes
  Sat-3: Setup time = 2.3 minutes
Calculated conflict degrees - Max: 3, Min: 0, Avg: 0.55
Generated 73 observation opportunities
Enhanced model built with 73 potential observations
Found 20 temporal conflicts (including setup time)
Conflict degree weighting: ENABLED
Solving enhanced MILP model...
Solution status: Optimal
Optimal objective value: 36.393
Selected 6 observations

[**TIMER**] MILP solve time: 0.05 seconds
Schedule validation: All setup time constraints satisfied ✓
```

```
=== Enhanced Solution Analysis ===
Total observations scheduled: 6
Total data volume: 0.01 GB

Conflict Degree Statistics:
  Average conflict degree: 0.17
  Max conflict degree: 1
  Min conflict degree: 0

Enhanced Satellite Statistics:
  Sat-1:
    Observations: 2
    Data volume: 0.01 GB
    Observation time: 19.0 minutes
    Time with setup: 20.8 minutes
    Setup overhead: 1.8 minutes
    Memory utilization: 0.0%
  Sat-2:
    Observations: 3
    Data volume: 0.00 GB
    Observation time: 24.0 minutes
    Time with setup: 30.0 minutes
    Setup overhead: 6.0 minutes
    Memory utilization: 0.0%
  Sat-3:
    Observations: 1
    Data volume: 0.00 GB
```

```
    Observation time: 5.0 minutes
    Time with setup: 5.0 minutes
    Setup overhead: 0.0 minutes
    Memory utilization: 0.0%

Target Coverage:
  ✓ Target_1 (Priority: 0.8)
  ✓ Target_2 (Priority: 0.8)
  ✓ Target_3 (Priority: 0.8)
  ✓ Target_4 (Priority: 0.9)
  ✓ Target_5 (Priority: 0.6)
  ✓ Target_6 (Priority: 0.6)
```

Γραφική απεικόνιση:



Σενάριο 3

Εκτέλεση του προγράμματος `satellite_scheduler.py` για πολύ μεγάλο custom σενάριο:

Enter number of satellites (e.g., 3): 20

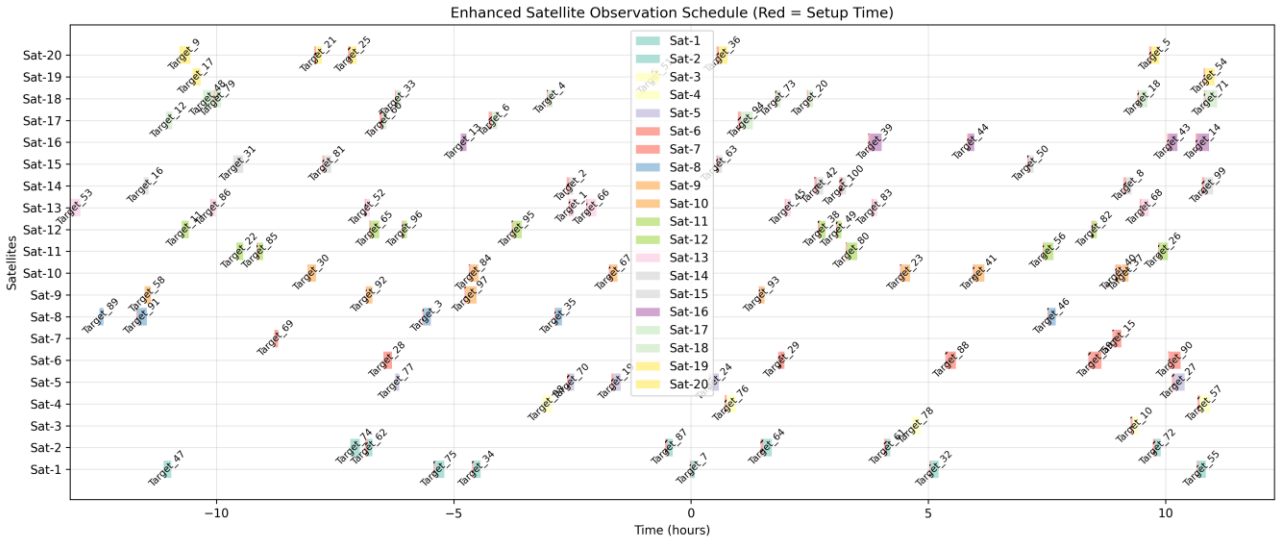
Enter number of targets (e.g., 5): 100

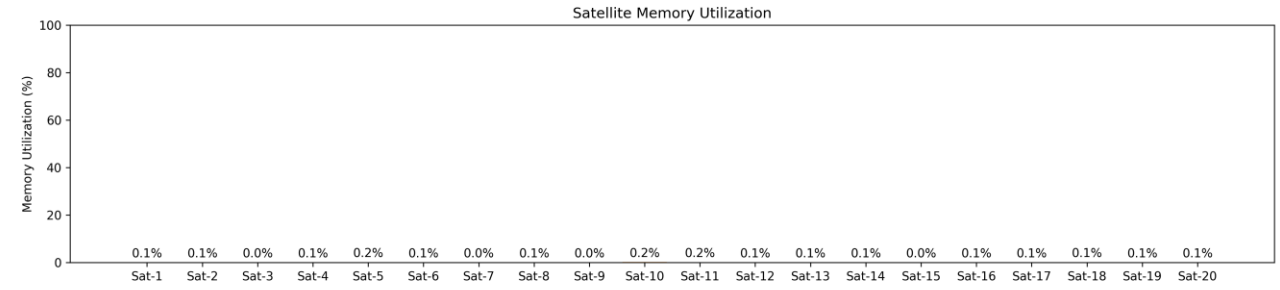
Enter scheduling time horizon in hours (default 24): 48

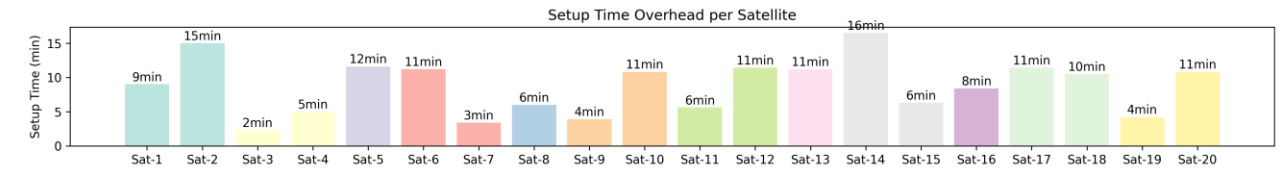
[**TIMER**] MILP solve time: 10.46 seconds

Total observations scheduled: 100

Total data volume: 0.26 GB







Συμπέρασμα:

Παρατηρούμε ότι το μοντέλο είναι εξαιρετικά αποδοτικό και διαχειρίζεται σενάριο διαφόρων μεγεθών με ευκολία.

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.