Πρόβλημα 1

Ερώτημα 1

Σε αυτήν την άσκηση μοντελοποίησα το πρόβλημα χρονοπρογραμματισμού που περιγράφεται στην εκφώνηση σε κατάλληλη μορφή ώστε να λύνεται με το csp module που χρησιμοποιούμε.

Αφού διαβάσω τα στοιχεία από το csv με την βιβλιοθήκη pandas, για κάθε μάθημα που έχει εργαστήριο δημιουργώ ένα καινούριο μάθημα που έχει το ίδιο όνομα + ' LAB'. Όλα τα υπόλοιπα πεδία είναι ίδια.

Επέλεξα αυτήν την προσέγγιση για τον χειρισμό των εργαστηρίων επειδή έδινε πειραματικά καλύτερα αποτελέσματα.

Φυσικά οι περιορισμοί τροποποιήθηκαν για να είναι valid. Για παράδειγμα, ο περιορισμός μαθημάτων του ίδιου καθηγητή σε διαφορετική μέρα δεν θα πρέπει να εφαρμοστεί αν πρόκειται για μαθήματα θεωρία-εργαστήριο, επειδη αυτά προφανώς πρέπει να είναι στην ίδια μέρα. Γενικά ο κώδικας είναι επαρκώς σχολιασμένος για τέτοιες σχεδιαστικές λεπτομέρειες.

Οι γείτονες κάθε μαθήματος ορίζονται ως όλα τα μαθήματα εκτός του εαυτού του (όλα τα μαθήματα συμμετέχουν στο constraint του όχι 2 μαθήματα στο ίδιο slot της ίδιας μέρας).

Το python script χρησιμοποιείται ως εξής:

```
./problem1.py <algorithm> <days> <times>
```

Η λύση του προβλήματος αποθηκεύεται στο αρχείο `solution.txt`

Όπου το algorithm ορίζει τον αλγόριθμο που θα λύσει το πρόβλημα, το days οι μέρες της εξεταστικής και times το πόσες φορές θα τρέξει ο αλγόριθμος (θα τυπωθεί μόνο μία λύση αλλά τα statistics θα περιλαμβάνουν τον μέσο όρο από δεδομένα από 10 εκτελέσεις).

Ως algorithm arguement υπάρχουν οι εξής συνδυασμοί: fc+mrv, mac+mrv, fc+domwdeg, mac+domwdeg, min-conflicts

Υλοποίηση dom/wdeg

Για την υλοποίηση του dom/wdeg όρισα ένα dictionary που ως index έχει ένα tuple (varA, varB) και ως value το weight, προκειμένου να καταγράφω κάθε domain wipe-out από κάποιο constraint που περιλαμβάνει το varA ή το varB.

Το dictionary constraint_weights ενημερώνεται στα εξής σημεία:

```python
def AC3(csp, queue=None, removals=None, arc_heuristic=dom_j_up):
    """[Figure 6.3]"""
    if queue is None:
        queue = {(Xi, Xk) for Xi in csp.variables for Xk in csp.neighbors[Xi]}
    csp.support_pruning()
    queue = arc_heuristic(csp, queue)
    checks = 0
    while queue:
        (Xi, Xj) = queue.pop()
        revised, checks = revise(csp, Xi, Xj, removals, checks)
        if revised:
            if not csp.curr_domains[Xi]:
                csp.constraint_weights[(Xi,Xj)] += 1
                csp.constraint_weights[(Xj,Xi)] += 1
                return False, checks  # CSP is inconsistent
            for Xk in csp.neighbors[Xi]:
                if Xk != Xj:
                    queue.add((Xk, Xi))
    return True, checks  # CSP is satisfiable
```

```python
def forward_checking(csp, var, value, assignment, removals):
    """Prune neighbor values inconsistent with var=value."""
    csp.support_pruning()
    for B in csp.neighbors[var]:
        if B not in assignment:
            for b in csp.curr_domains[B][:]:
                if not csp.constraints(var, value, B, b):
                    csp.prune(B, b, removals)
            if not csp.curr_domains[B]:
                csp.constraint_weights[(B,var)] += 1
                csp.constraint_weights[(var,B)] += 1
                return False
    return True
```

Δηλαδή στα domain wipe-outs. (για τον αλγόριθμο MAC άλλαξα την default συνάρτηση σε AC3 αντί για την βελτιωμένη έκδοση AC3b επειδή μου ήταν πιο ξεκάθαρο το που να ενημερώνω τα weights, σε κάθε περίπτωση όμως είναι ο ίδιος αλγόριθμος).

Τέλος, όρισα το dom_wdeg ακριβώς όπως ορίζεται το mrv:

```python
def dom_wdeg(assignment, csp):
    return argmin_random_tie([v for v in csp.variables if v not in assignment],
                             key=lambda var: dom_wdeg_ratio(var, assignment, csp))


def dom_wdeg_ratio(var, assignment, csp):
    sum = 0
    for neighbor in csp.neighbors[var]:
        if neighbor not in assignment:
            sum += csp.constraint_weights[(var, neighbor)]
    # if we have no neighbors (sum == 0) that means the only unassigned variable is
    # only one unassigned left so it doesnt even matter what we return. just set the
    if sum == 0:
        sum = 1
    # else we update the min_var if we have Found
    # a better ratio dom / wdeg
    if not csp.curr_domains: # if we have no curr_domains that means we didnt prune
        l = len(csp.domains[var])
    else:
        l = len(csp.curr_domains[var])
    return l / sum
```

Δηλαδή επιστρέφουμε την μεταβλητή που ελαχιστοποιεί το dom/wdeg ratio.

Στο dom/wdeg ratio έχουμε δύο ειδικές περιπτώσεις που χρειάζονται συνθήκες (sum == 0 και curr_domains == 0).
Ο κώδικας έχει σχολιαστεί κατάλληλα για να εξηγήσει γιατί χρειάζονται.

Statistics

Ακολουθούν στατιστικά εκτέλεσης των αλγορίθμων για το πρόβλημα χρονοπρογραμματισμού.

Για μετρικές επέλεξα τον χρόνο εκτέλεσης, τα assigns και τα constraint checks.

Ο χρόνος εκτέλεσης είναι προφανές γιατί επιλέχθηκε.

Στα assigns κωδικοποιείται και το πόσες φορές έκανε backtrack (assigns - 43 φορές, όπου 43 είναι τα μαθήματα (38+5 labs)).

Τα constraint checks επιλέχθηκαν καθώς αναφέρονται στις διαλέξεις.

Τα δεδομένα είναι απο εκτελέσεις με 21 μέρες εξεταστικής και αποτελούν το μέσο όρο 10 εκτελέσεων κάθε αλγορίθμου

ALGORITHM	TIME	ASSIGNS	CONSTRAINT CHECKS
FC+MRV	0.048	49	47527
MAC+MRV	0.77	43	917139
FC+DOM/WDEG	0.036	48	47443
MAC+DOM/WDEG	0.73	43	916287
MIN-CONFLICTS	0.18	76	207837

Όπως βλέπουμε, ο FC κάνει outperform τον MAC σημαντικά για το δεδομένο πρόβλημα.

Επιπλέον, ο FC βλέπει μια μικρή βελτίωση σε χρόνο χρησιμοποιώντας την μέθοδο dom/wdeg για την επιλογή επόμενης unassigned μεταβλητής.

Ο min-conflicts έρχεται ανάμεσα τους με επιδόσεις κοντά στον FC.

Ωστόσο διακρίνουμε πως ο min-conflicts είναι ο μοναδικός αλγόριθμος που τερματίζει σε κάτω από ένα δευτερόλεπτο για πολύ λίγες μέρες (όχι όμως λιγότερες από τις θεωρητικά ελάχιστες δυνατές για το πρόβλημα!, βλ ερώτημα 3).

```
PS C:\Users\sakoo\Desktop\stuff\dit\ai\ai-hw3\problem1> python problem1.py min-conflicts 16 10
MIN-CONFLICTS solution saved to solution.txt!

Statistics (average of 10 runs):
Time: 0.586839246749878, Assigns: 210, Checks: 683424
```

Ακόμα και για 16 μέρες, ο αλγόριθμος τερματίζει consistently σε 10 εκτελέσεις στην σειρά. Οι αλγόριθμοι FC και MAC από την άλλη είναι πολύ unreliable για μέρες < 21.

Εξαιτίας του random factor της argmin_random_tie, ο αλγόριθμος FC μπορεί σε μία δεδομένη εκτέλεση να τερματίσει ακαριαία ενώ σε μια άλλη να τρέχει για ώρες χωρίς αποτέλεσμα. Εδώ διακρίνεται η σημασία της διάταξης των μεταβλητών.

Βλέπουμε πως η μη-ντετερμινιστική φύση των αλγορίθμων οδηγεί σε μη προβλέψιμη συμπεριφορά και φανερώνει την εγγενή πολυπλοκότητα των προβλημάτων ικανοποίησης περιορισμών.

Ερώτημα 3

Ας υπολογίσουμε τον ελάχιστο χρόνο διάρκειας της εξεταστικής.

Από το αρχείο csv βλέπουμε ότι υπάρχουν 16 μαθήματα του 7ου εξαμήνου, και εφ'όσον ένα από τα constraints είναι πως τα μαθήματα του ίδιου εξαμήνου θα πρέπει να εξετάζονται σε διαφορετικές ημέρες,

εύκολα συμπαιρένουμε ότι η εξεταστική θα πρέπει να διαρκέσει **κατ' ελάχιστον** 16 μέρες. Αυτό όμως δεν σημαίνει ότι απαντήσαμε στο ερώτημα, έχουμε απλά ένα κάτω όριο.

Ωστόσο, όπως είδαμε στο ερώτημα 2 ο αλγόριθμος min-conflicts παράγει λύση για 16 μέρες. Προφανώς, λύση για 16 μέρες συνεπάγεται λύση για κάθε πλήθος ημερών >= 16.

Αφού η θεωρητικά ελάχιστη λύση θα είναι τουλάχιστον 16 μέρες και έχουμε μία λύση για 16 μέρες, τότε μπορούμε να συνάγουμε πως ο ελάχιστος χρόνος διάρκειας της εξεταστικής για το πρόβλημα είναι ακριβώς **16 μέρες**.

Πρόβλημα 2

Μεταβλητές:

V = {Κρεβάτι,Γραφείο,Καρέκλα γραφείου,Καναπές}

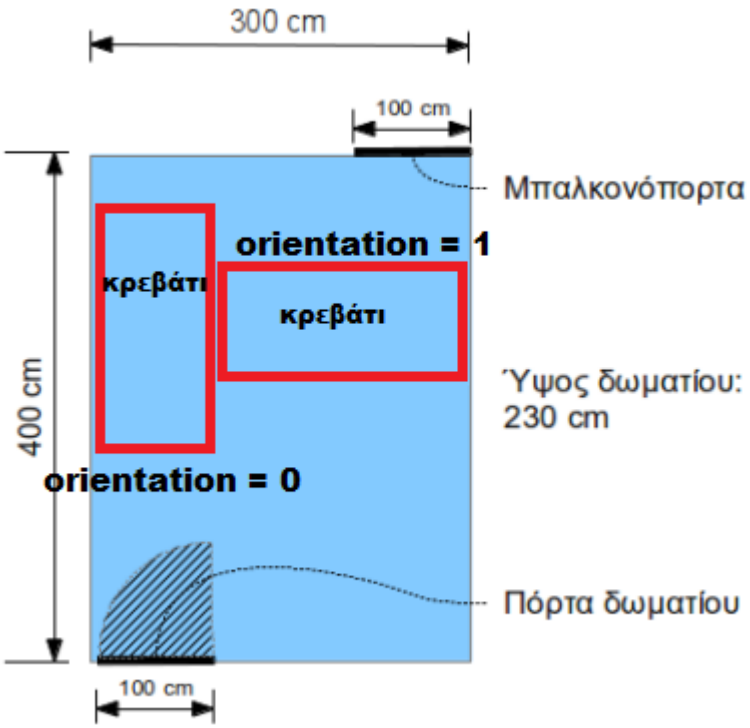
Πεδία:

Ως το πεδίο των μεταβλητών θα ορίσουμε μια πλειάδα (tuple) ως εξής:

D = (x, y, r)

Αγνοούμε το z (ύψος) επειδη δεν χρειάζεται σε κανέναν περιορισμό και το ταβάνι είναι ούτως ή άλλως πιο ψηλό από όλα τα έπιπλα.

Το r δηλώνει το orientation και παίρνει τιμές 0 ή 1 (0 ή 90 μοίρες).
Το orientation θα το ορίσουμε ως με το ποια πλευρά του (πλάτος ή μήκος) είναι παράλληλη στον άξονα των x.



Απο εδώ και πέρα όταν αναφερόμαστε στις διαστάσεις `w, l` θα εξαρτώνται από το orientation.
Το w θα είναι το μήκος της πλευράς παράλληλη στον άξονα x ενώ το l θα είναι το μήκος της πλευράς παράλληλη στον άξονα y.

Περιορισμοί:

Αρχικά πρέπει να μπορεί να ανοίγει η πόρτα, που σημαίνει κανένα έπιπλο στο τετράγωνο που ξεκινάει στο (0,0) και έχει πλευρά 100cm.
Για οποιοδήποτε έπιπλο με συντεταγμένες (x, y):

$$x > 100 \text{ or } y > 100$$

Για οποιαδήποτε δύο έπιπλα θα πρέπει να ισχύει ότι δεν εφάπτονται ή να κάνουν overlap.
Αν θεωρήσουμε (x1, y1), (x2, y2) τις συντεταγμένες των επίπλων και (w1, l1), (w2, l2) οι διαστάσεις του αυτός ο περιορισμός μεταφράζεται ως εξής μαθηματικά:

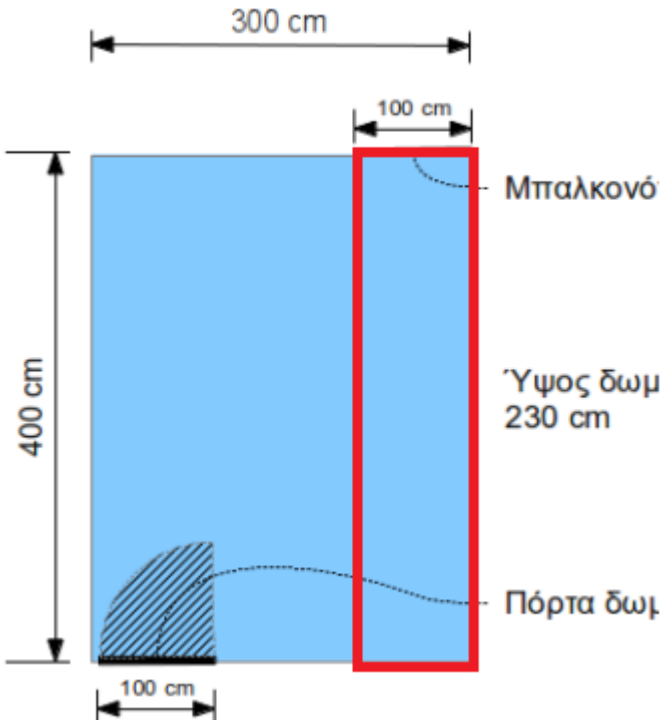
$$x1 + w1 > x2 \text{ or } x2 + w2 > x1 \text{ or } y1 + l1 > y2 \text{ or } y2 + l2 > y1$$

Ουστιαστικά για να μην εφάπτονται, είτε το δεξί άκρο της προβολής του ενός στον άξονα x θα βρίσκεται πριν από το αριστερό άκρο του άλλου,
είτε το πάνω άκρο της προβολής του ενός στον άξονα y θα βρίσκεται πριν από το κάτω άκρο του άλλου.

Επιπλέον, το κρεβάτι θα πρέπει να εφάπτεται με έναν απ'τους τοίχους.
Αυτο σε συντεταγμένες μεταφράζεται ως εξής ((x, y) και (l, w) οι συντεταγμένες και διαστάσεις του κρεβατιού):

$$x = 0 \text{ or } y = 0 \text{ or } x + w = 300 \text{ or } y + l = 300$$

Τέλος, θα πρέπει το γραφείο να είναι κολλημένο σε τοίχο και κάποιο μέρος του να βρίσκεται σε αυτήν την ευθεία του φωτός από το μπαλκόνι:



Με άλλα λόγια, είτε θα εφάπτεται στον δεξιά τοίχο (και σίγουρα κάποιο κομμάτι του θα βρίσκεται στο φως),
είτε θα εφάπτεται στον κάτω τοίχο και ένα μέρος του θα βρίσκεται δεξιά της αριστερής κόκκινης ευθείας.
Φυσικά μπορεί να ισχύουν και τα δύο (και να εφάπτεται στην κάτω δεξιά γωνία)

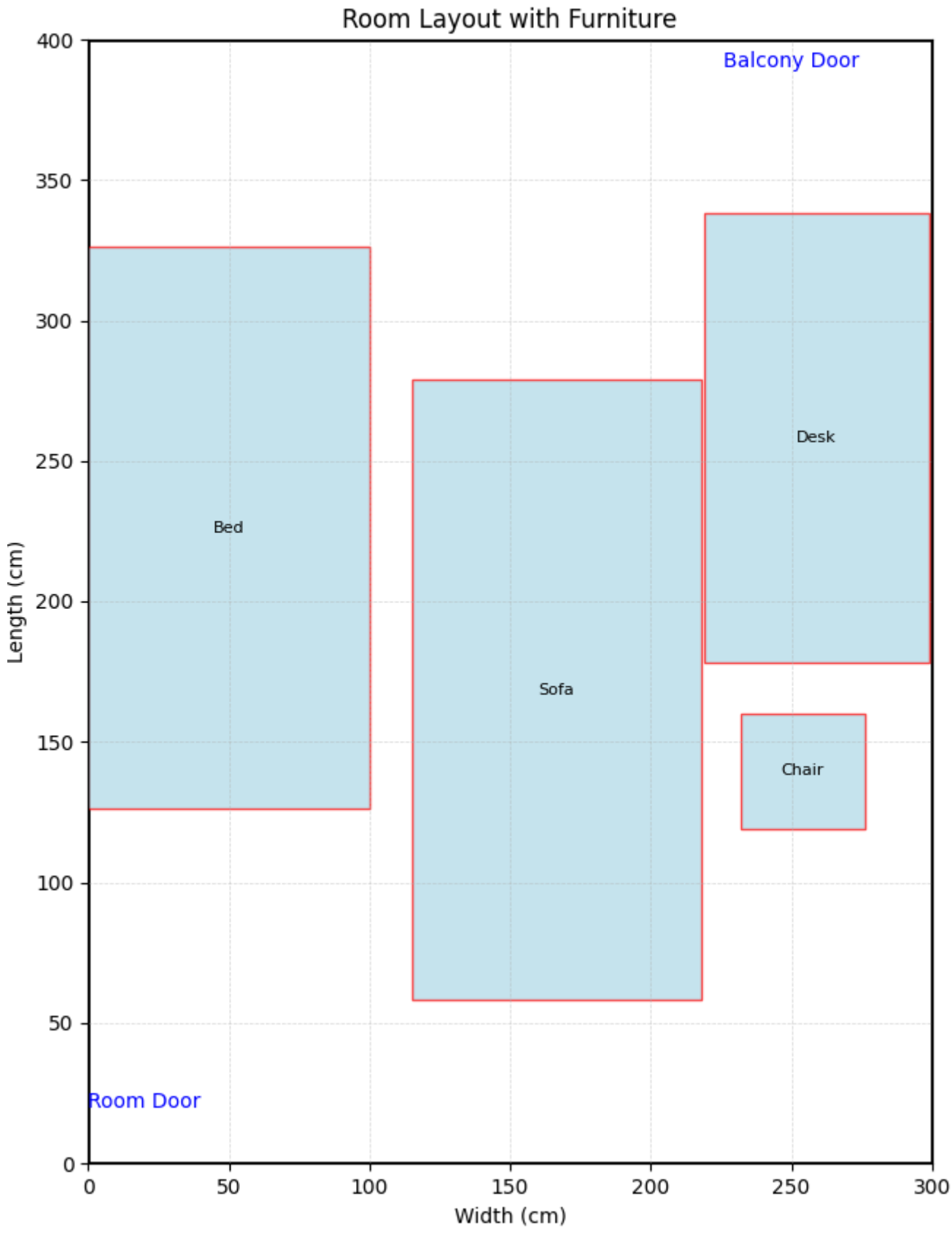
$$x + w = 300 \text{ or } (y = 0 \text{ and } x + w > 200)$$

Λύση

Παραθέτω μία λύση του CSP:

Κρεβάτι: (0, 126, 0)
Γραφείο: (219, 178, 1)
Καρέκλα: (232, 119, 1)
Καναπές: (115, 58, 1)

VISUALIZATION



Όπως βλέπουμε, όλοι οι περιορισμοί ικανοποιούνται.

Αν και δεν ζητήθηκε, στον φάκελο problem2_code έχω παραθέσει και τον κώδικα που χρησιμοποίησα για μοντελοποιήσω το πρόβλημα και να το λύσω με το csp module του πρώτου ερωτήματος.

Πρόβλημα 3

Σε αυτήν την άσκηση θα μοντελοποιήσουμε το πρόβλημα που περιγράφεται στην εκφώνηση ως ένα πρόβλημα ικανοποίησης περιορισμών.

Μεταβλητές:

A1, A2, A3, A4, A5

Πεδίο:

{9:00, 10:00, 11:00}

Περιορισμοί:

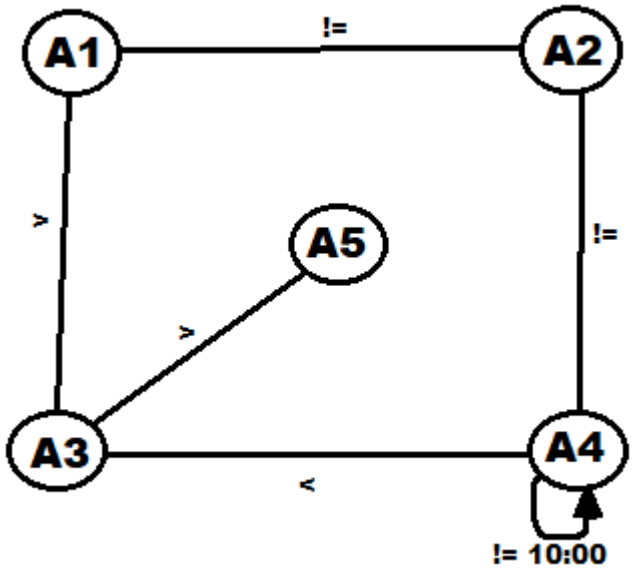
A1 > A3

A3 > A5 and A3 < A4

A2 ≠ A1 and A2 ≠ A4

A4 ≠ 10:00

Γράφος περιορισμών



Αλγόριθμος συνέπειας τόξου AC-3

Ας εφαρμόσουμε τον αλγόριθμο A3 στο πρόβλημα.

```
Starting AC-3 algorithm...
```

```
Initial queue: {('A3', 'A5'), ('A1', 'A2'), ('A3', 'A4'), ('A4', 'A2'), ('A2',
'A1'), ('A3', 'A1'), ('A5', 'A3'), ('A1', 'A3'), ('A4', 'A3'), ('A2', 'A4')}

Removing (A3, A5) from queue
  Removed value 9 from domain of A3, as no value in A5 allows it to satisfy the
constraint between A3 and A5. (A3 > A5 isn't possible if A3 is 9 which is domain
min)
New domain of A3: [10, 11]
Arc (A1, A3) is inserted in the queue.
Arc (A4, A3) is inserted in the queue.

Current queue: {('A1', 'A2'), ('A3', 'A4'), ('A4', 'A2'), ('A2', 'A1'), ('A3',
'A1'), ('A5', 'A3'), ('A1', 'A3'), ('A4', 'A3'), ('A2', 'A4')}

Removing (A1, A2) from queue
No changes to domain of A1.

Current queue: {('A3', 'A4'), ('A4', 'A2'), ('A2', 'A1'), ('A3', 'A1'), ('A5',
'A3'), ('A1', 'A3'), ('A4', 'A3'), ('A2', 'A4')}

Removing (A3, A4) from queue
  Removed value 11 from domain of A3, as no value in A4 allows it to satisfy the
constraint between A3 and A4. (A3 < A4 isn't possible if A3 is 11 which is domain
max)
New domain of A3: [10]
Arc (A1, A3) is inserted in the queue.
Arc (A5, A3) is inserted in the queue.

Current queue: {('A4', 'A2'), ('A2', 'A1'), ('A3', 'A1'), ('A5', 'A3'), ('A1',
'A3'), ('A4', 'A3'), ('A2', 'A4')}

Removing (A4, A2) from queue
  Removed value 10 from domain of A4, as no value in A2 allows it to satisfy the
constraint between A4 and A2. (A4 can't be 10)
New domain of A4: [9, 11]
Arc (A3, A4) is inserted in the queue.

Current queue: {('A3', 'A4'), ('A2', 'A1'), ('A3', 'A1'), ('A5', 'A3'), ('A1',
'A3'), ('A4', 'A3'), ('A2', 'A4')}

Removing (A2, A1) from queue
No changes to domain of A2.

Current queue: {('A3', 'A4'), ('A3', 'A1'), ('A5', 'A3'), ('A1', 'A3'), ('A4',
'A3'), ('A2', 'A4')}

Removing (A3, A1) from queue
No changes to domain of A3.

Current queue: {('A3', 'A4'), ('A5', 'A3'), ('A1', 'A3'), ('A4', 'A3'), ('A2',
'A4')}

Removing (A5, A3) from queue
  Removed value 10 from domain of A5, as no value in A3 allows it to satisfy the
```

```
constraint between A5 and A3. (A3 == 10 at this point, and A3>A5. so A5 = 9)
  Removed value 11 from domain of A5, as no value in A3 allows it to satisfy the
constraint between A5 and A3.
New domain of A5: [9]

Current queue: {('A3', 'A4'), ('A1', 'A3'), ('A4', 'A3'), ('A2', 'A4')}

Removing (A1, A3) from queue
  Removed value 9 from domain of A1, as no value in A3 allows it to satisfy the
constraint between A1 and A3. (A3 == 10 at this point, and A3<A2. so A2 = 11)
  Removed value 10 from domain of A1, as no value in A3 allows it to satisfy the
constraint between A1 and A3.
New domain of A1: [11]

Arc (A2, A1) is inserted in the queue.

Current queue: {('A3', 'A4'), ('A2', 'A1'), ('A4', 'A3'), ('A2', 'A4')}

Removing (A4, A3) from queue
  Removed value 9 from domain of A4, as no value in A3 allows it to satisfy the
constraint between A4 and A3. (A3 == 10 at this point, and A3<A4. so A4 = 11)
New domain of A4: [11]
Arc (A2, A4) is inserted in the queue.

Current queue: {('A3', 'A4'), ('A2', 'A1'), ('A2', 'A4')}

Removing (A2, A4) from queue
  Removed value 11 from domain of A2, as no value in A4 allows it to satisfy the
constraint between A2 and A4. (A4 == 11 at this point, and A2!=A4. so A2 != 11)
New domain of A2: [9, 10]
Arc (A1, A2) is inserted in the queue.

Current queue: {('A1', 'A2'), ('A3', 'A4'), ('A2', 'A1')}

Removing (A1, A2) from queue
No changes to domain of A1.

Current queue: {('A3', 'A4'), ('A2', 'A1')}

Removing (A3, A4) from queue
No changes to domain of A3.

Current queue: {('A2', 'A1')}

Removing (A2, A1) from queue
No changes to domain of A2.

Queue empty!

AC-3 algorithm finished. CSP is arc-consistent.

Final domains (arc-consistent):
  A1: [11]
  A2: [9, 10]
```

```
A3: [10]
A4: [11]
A5: [9]
```

Χρονικό πρόβλημα (BONUS)

Μεταβλητές

X1: Η Μαρία φεύγει απ'το σπίτι

X2: Η Μαρία φτάνει στο γραφείο

X3: Η Ελένη φεύγει απ'το σπίτι

X4: Η Ελένη φτάνει στο γραφείο

Ως X0 (reference time point) θα πάρουμε την νωρίτερη δυνατή ώρα να φύγει η Μαρία απ'το σπίτι, δηλαδή 8:00. Όλες οι μεταβλητές θα παίρνουν τιμές ως την απόσταση από αυτήν την ώρα.

Επομένως:

Περιορισμοί:

- $0 \leq X1 \leq 10$ (Η Μαρία ξεκινάει από το σπίτι της 8 - 8:10 το πρωί)
- $30 \leq X2 - X1 \leq 40$ (το ταξίδι της Μαρίας από το σπίτι στο γραφείο διαρκεί 30-40 λεπτά)
- $X4 = X2 + 15$ (η Ελένη φτάνει ακριβώς 15 λεπτά μετά στο γραφείο)
- $5 \leq X4 - X3 \leq 15$ (το ταξίδι της Ελένης από το σπίτι στο γραφείο διαρκεί 5-15 λεπτά)

Διάδοση περιορισμών

Ας κάνουμε propagate τα constraints. Για παράδειγμα, αν το ταξίδι διαρκεί 30-40 λεπτά και ξεκινάει από το σπίτι της 8-8:10, τότε προφανώς θα φτάσει στις 8:30-8:50. Όμοια για τα υπόλοιπα:

$0 \leq X1 \leq 10$

$30 \leq X2 \leq 50$

$45 \leq X4 \leq 65$

$30 \leq X3 \leq 60$

Εύκολα έχουμε 2 λύσεις:

(X1, X2, X3, X4) = (0, 30, 30, 45)

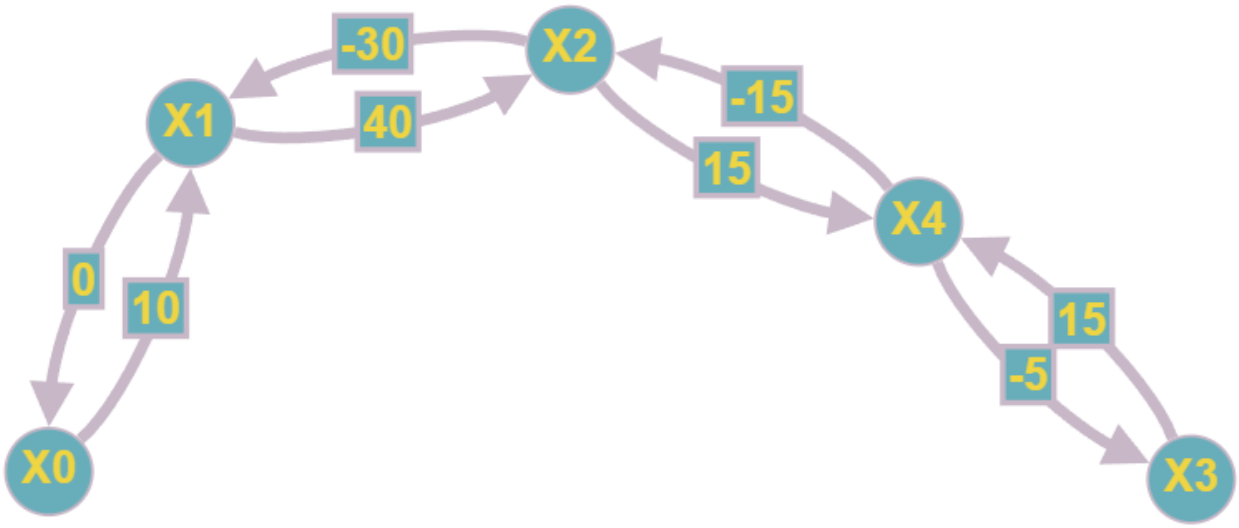
Η Μαρία ξεκίνησε στις 8 και έφτασε στις 8:30, η Ελένη ξεκίνησε στις 8:30 και έφτασε στις 8:45. Όλοι περιορισμοί ικανοποιούνται. Όμοια:

(X1, X2, X3, X4) = (10, 50, 60, 65)

Επομένως το πρόβλημα είναι συνεπές.

Distance graph

Παραθέτω το distance graph όπως περιγράφεται στο paper:



Υλοποίηση αλγορίθμου Floyd Warshall

Στο αρχείο problem4.py είναι υλοποιημένος ο αλγόριθμος Floyd-Warshall όπως περιγράφεται στο paper. Το scipt τυπώνει τα αποτελέσματα 2 εκτελέσεων του αλγορίθμου, μία για το graph του προβλήματος και μία για το graph του προβλήματος στο paper (σελίδα 8).

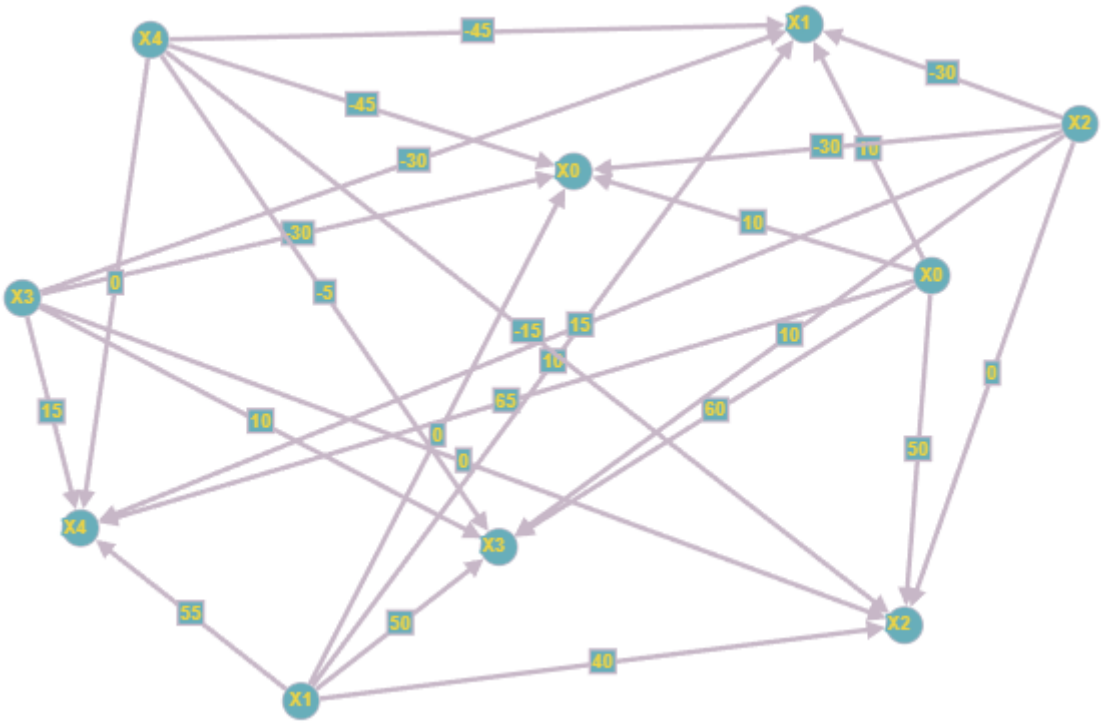
Οι γράφοι κατασκευάζονται και ορίζονται ως tables, όμοια με το paper

Table 1
Lengths of shortest paths in the distance graph of Fig. 3.

	0	1	2	3	4
0	0	20	50	30	70
1	−10	0	40	20	60
2	−40	−30	0	−10	30
3	−20	−10	20	0	50
4	−60	−50	−20	−40	0

```
Shortest path solution on my graph (d-graph):
X0 {'X0': 0, 'X1': 10, 'X2': 50, 'X3': 60, 'X4': 65}
X1 {'X0': 0, 'X1': 0, 'X2': 40, 'X3': 50, 'X4': 55}
X2 {'X0': -30, 'X1': -30, 'X2': 0, 'X3': 10, 'X4': 15}
X3 {'X0': -30, 'X1': -30, 'X2': 0, 'X3': 0, 'X4': 15}
X4 {'X0': -45, 'X1': -45, 'X2': -15, 'X3': -5, 'X4': 0}
```

Η οπτικοποίηση του γράφου θα έμοιζε κάπως έτσι:



Οπότε καλύτερα να το αφήσουμε ως table.

Ως λύση του προβλήματος χρησιμοποιώντας το d-graph τυπώνω την μέγιστη απόσταση κάθε μεταβλητής από το X0. (ή από το 0 στο δεύτερο πρόβλημα)

Αυτό ισχύει σύμφωνα με το paper:

Corollary 3.2. *Let G_d be the distance graph of a consistent STP. Two consistent scenarios are given by:*

$$S_1 = (d_{01}, \ldots, d_{0n}), \tag{3.8}$$

$$S_2 = (-d_{10}, \ldots, -d_{n0}), \tag{3.9}$$

which assign to each variable its latest and earliest possible time, respectively.

n-consistency

Ας εξετάσουμε το παρακάτω θεώρημα του paper:

Theorem 3.3 (Decomposability). *Any consistent STP is decomposable relative to the constraints in its d-graph.*

A network is *decomposable*[2] [36], if every locally consistent assignment[3] to any set of variables, S, can be extended to a solution. The importance of decomposability lies in facilitating the construction of a solution by a *backtrack-free search* [20].

Ουσιαστικά, κάθε consistent STP είναι decomposable σε σχέση με τους περιορισμούς στο d-graph του. Φυσικά δεν έχει νόημα να μιλάμε για n-consistency ή d-graph αν δεν είναι consistent το ίδιο το πρόβλημα,

οπότε υποθέτουμε ότι είναι. Άρα είναι decomposable.

Σε κάθε decomposable network, οποιαδήποτε locally consistent ανάθεση σε οποιοδήποτε πλήθος μεταβλητών μπορεί να επεκταθεί σε πλήρη λύση.
Αυτό όμως είναι ακριβώς ο ορισμός της n-συνέπειας (απλά για κάθε k: 1...n).

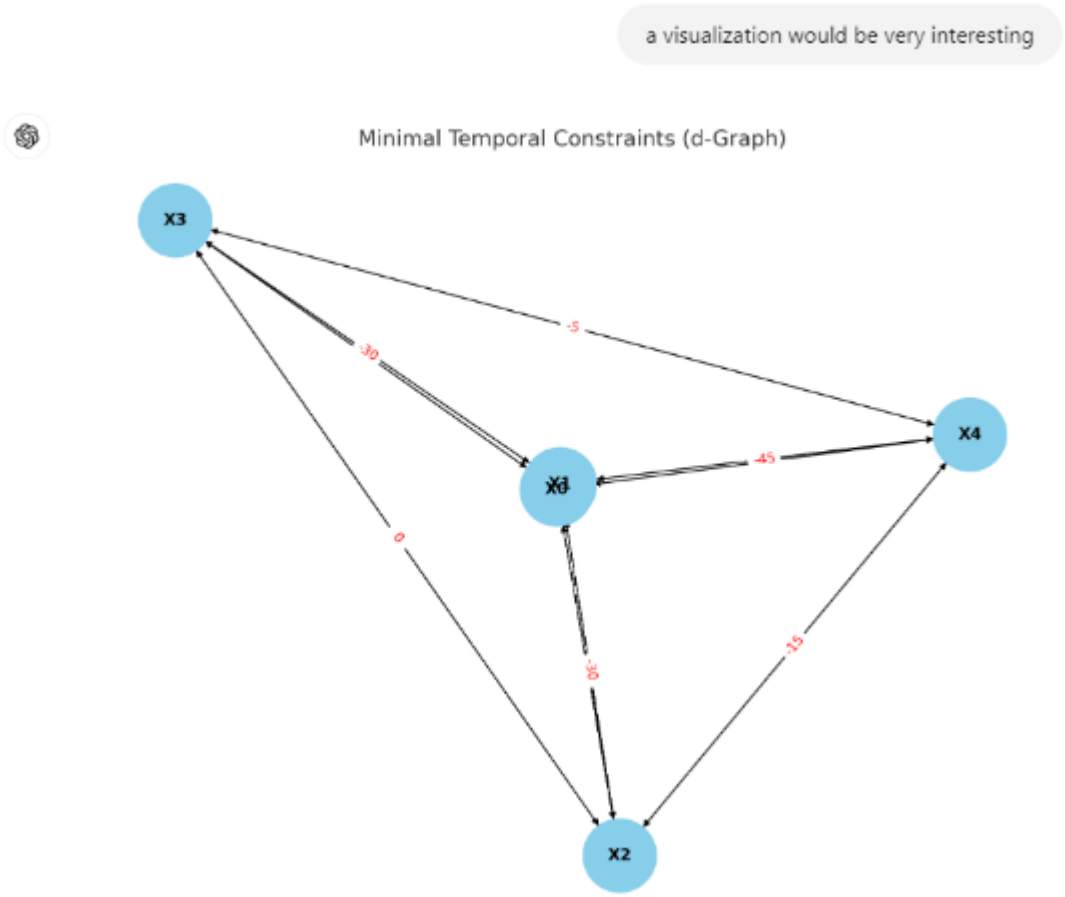
Επομένως, το πρόβλημα που προκύπτει με την εφαρμογή του αλγορίθμου είναι 1-, 2-, ..., n-consistent όπου n είναι το πλήθος των μεταβλητών του προβλήματος.

ChatGPT

https://chatgpt.com/share/67587399-11fc-8005-8701-f86b2c5c426a

Παραπάνω έχω παραθέσει ένα prompt στο οποίο το ChatGPT εντυπωσιακά και σχεδόν τρομακτικά κατασκευάζει python script που
τρέχει τον αλγόριθμο Floyd-Warshall, το τρέχει, και τυπώνει το αποτέλεσμα το οποίο είναι το σωστό d-graph του προβλήματος.

Η νέα δυνατότητα του GPT να τρέχει τον κώδικα που παράγει σε δικό του environment και τα παίρνει τα αποτελέσματα έχει απογειώσει
την απόδοση και το accuracy του σε τέτοιου είδους προβλήματα.



Βέβαια όταν του ζήτησα να κάνει visualization ενός γράφου με 25 ακμές, προφανώς έφτασε στα όριά του 😛.