Τεχνητή Νοημοσύνη Εργασία 3

Μαραντίδης Θεοφάνης

1115201800106

Για την εκτέλεση της εργασίας πρέπει να τρέξουμε τον parser των αρχείων (rlfa.py) ως εξής:

python3.8 rlfa.py <txt problem file> <algorithm>

Πχ. Για το πρόβλημα **6-w2** έχουμε:

- python3.8 rlfa.py **6-w2** FC για τον **FC**
- python3.8 rlfa.py **6-w2** MAC για τον **MAC**
- python3.8 rlfa.py **6-w2** CBJ για τον **FC-CBJ**
- python3.8 rlfa.py **6-w2** MIN για τον **MIN-CONFLICTS**

		2-f24	2-f25	3-f10	3-f11	8-f10	8-f11	14-f27	14-f28
Forward Checking	**Time**	0,627 secs	1m5,055 secs	22,201 secs	5m45,459 secs	-	6m52,544 secs	-	2m35,787 Secs
	Checks	100754	22428615	5904239	95850540	-	61216543	-	5805381
	Visited Nodes	937	135607	68841	563197	-	506070	-	103387
MAC	**Time**	0,582 secs	2m8,146 secs	2,419 secs	2m26,039 secs	1m5,551 secs	2m29,072 Secs	19,592 secs	53,546 Secs
	Checks	158995	66044241	1167548	107331420	33106552	73692412	4103679	13098591
	Visited Nodes	200	28322	776	22089	17487	30663	14185	22043
FC-CBJ	**Time**	0,607 secs	3,059 Secs	33,85 secs	5m28,24 Sec	2m57,21 Secs	2m6,994 secs	2m26,92 Secs	32,454 secs
	Checks	21577	555494	3250161	85785965	8631876	25376777	1018313	412093
	Visited Nodes	265	3525	24838	583427	75311	127232	18974	6665
MIN CONFLICTS	**Time**	4m19,628 secs	-	10m16,078 secs	-	-	-	-	-
	Visited Nodes	100200	-	100400	-	-	-	-	-

		6-w2	7-w1-f4	11
Forward Checking	**Time**	0,552 secs	9,124 secs	10,109secs
	Checks	72220	92337	1203335
	Visited Nodes	642	1685	6316
MAC	**Time**	0,621 secs	0,808 secs	14,416 secs
	Checks	397482	342291	6401738
	Visited Nodes	42	479	2956
FC-CBJ	**Time**	0,524 secs	1,866 secs	18,680 secs
	Checks	72220	92337	1203335
	Visited Nodes	642	1685	6316
MIN CONFLICTS	**Time**	3m12,719 Secs	-	20m34,987 secs
	Visited Nodes	100200	-	100680

<u>Παρατηρούμε:</u>

- Για την αξιολόγηση των αλγορίθμων συγκρίνουμε τον χρόνο εκτέλεσης , το πλήθος των ελέγχων συνέπειας αλλά και το πλήθος κόμβων που επισκέπτονται
- Οι FC , MAC , FC-CBJ δεν εξαρτώνται από κάποιο τυχαίο παράγοντα για αυτό και δεν διαφέρουν τα checks και τα visited nodes από κλήση σε κλήση.
- Για τα στιγμιότυπα του προβλήματος που αλγόριθμος τρέχει για πάνω από 20 λεπτά βάζουμε παύλα στο αντίστοιχο κουτάκι του πίνακα καθώς διακόπτουμε την εκτέλεση.
- Το πλήθος των ελέγχων για την συνέπεια για κάθε αλγόριθμο ακολουθεί την εξής σειρά:

  FC-CBJ ≤ FC

  (Η ισότητα εμφανίζεται στα πιο «απλά» σχετικά προβλήματα.

  Όσο για το πλήθος κόμβων που επισκέπτεται ο κάθε αλγόριθμος ισχύει:

  MAC ≤ FC και FC-CBJ ≤ FC

Οι αλγόριθμοι **FC** και **MAC** έχουν υλοποιηθεί από τον κώδικα που μας υποδείχτηκε στο github του AIMA.

Η αλλαγή που έγινε αφορά την συνάρτηση **revise()** και την **forward_checking()** για την αύξηση του βάρους του περιορισμού κατά 1 (έχουμε ορίσει ένα dictionary στην κλάση csp το οποίο είναι αρχικοποιημένο με 1) σε περίπτωση αποτυχίας έτσι ώστε να μπορούμε να υπολογίσουμε το degree της κάθε μεταβλητής μετέπειτα στην ευρετική μας (dom/wdeg).

Η αλλαγές έγιναν βάση του paper που μας δόθηκε.

Algorithm 1 revise(C : Constraint, X : Variable) : boolean

for each $a \in dom(X)$ **do**
    if $seekSupport(C, X, a) = false$ **then**
        remove a from $dom(X)$
if $dom(X) = \emptyset$ **then**
    $weight[C]$ ++
return $Dom(X) \neq \emptyset$

<u>**FC-CBJ**</u>

Στο πρόγραμμα υλοποιήσαμε τον αλγόριθμο CBJ (Conflicted-Directed Backjumping) αφού το Forward Checking είναι ήδη υλοποιημένο στο github.

Έχουμε ένα σετ για την αποθήκευση των συγκρούσεων της κάθε μεταβλητής το οποίο και ανανεώνουμε μετά από κάθε ανάθεσή μας. Όταν βρισκόμαστε σε dead-end τότε κρατάμε την μεταβλητή στην οποία προκλήθηκε έστω Xj. Ο αλγόριθμός μας υπαναχωρεί προς την μεταβλητή Xκ του συνόλου συγκρούσεων της Xj που βρίσκεται βαθύτερα στο δέντρο αναζήτησης και οι υπόλοιπες μεταβλητές στο σύνολο συγκρούσεων της Xj προστίθενται στο σύνολο συγκρούσεων της Xk με την βοήθεια της merge.

Min-Conflicts

Ο ευρετικός μηχανισμός ελάχιστων συγκρούσεων παρατηρούμε ότι δεν καταφέρνει να λύσει το ίδιο αποδοτικά με τους άλλους αλγορίθμους τα προβλήματα που του δίνονται. Βρίσκει λύση μόνο στα ευκολότερα προβλήματα σε «καλό» χρόνο.

Αυτό συμβαίνει διότι ο αλγόριθμος επιλέγει τυχαία μία μεταβλητή από το σύνολο μεταβλητών που έχει conflicts που παραβιάζουν έναν ή περισσότερους περιορισμούς του προβλήματος μας. Η διαδικασία τυχαίας επιλογής μεταβλητής και εκχώρησης τιμής ελάχιστης σύγκρουσης επαναλαμβάνεται μέχρι να βρεθεί μια επιθυμητή λύση. Γεγονός που επιβαρύνει χρονικά τον αλγόριθμο όταν έχουμε αραιά κατανεμημένες λύσεις.

Ευρετική dom/wdeg

Η υλοποίηση της ευρετικής dom/wdeg έχει υλοποιηθεί βάση του pdf που προτείνεται από την εκφώνηση.

Υπολογίζουμε για κάθε μεταβλητή που δεν της έχουμε κάνει ανάθεση τιμής το weighted degree, προσθέτοντας τα βάρη των περιορισμών στα οποία συμμετέχει η μεταβλητή αυτή.

Ordering Heuristic wdeg:

$$\alpha_{wdeg}(X_i) = \sum_{C \in \mathscr{C}} weight[C] \mid vars(C) \ni X_i \wedge \mid FutVars(C) \mid > 1$$

Στη συνέχεια κάνουμε την διαίρεση του πλήθους των διαθέσιμων τιμών που μπορεί να πάρει η μεταβλητή (domains list size) με το weighted degree της μεταβλητής αυτής. Επιστρέφεται εν τέλη η μεταβλητή με το μικρότερο αποτέλεσμα της παραπάνω διαίρεσης.

Πρόβλημα 2

Το πρόβλημα Ικακοποίησης περιορισμών ορίζεται ως εξής:

Κάθε μεταβλητή καθορίζεται από τις συντεταγμένες 2 σημείων (x1,y1) και (x2,y2) όπου καθορίζουν την κάτω αριστερή και την πάνω δεξία γωνία αντίστοιχα του κάθε επίπλου.Με αυτό τον τρόπο μπορούμε να συγκρίνουμε τις συντεταγμένες του ενός επίπλου με κάθε άλλου και να δούμε αν το ένα πατάει πάνω στο άλλο.

Για το γραφείο λαμβάνουμε υπόψη και την απόσταση από την μπαλκονόπορτα (ευκλείδεια απόσταση) να μην είναι μεγαλύτερη από 5.

Το πάνω δεξία σημείο του γραφείου να μην απέχει περισσότερο από 5 μονάδες από το κάτω αριστερά της μπαλκονόπορτας.

Μόνο το σημείο (x1,y1) είναι αναγκαίο να πάρει τιμή από το Domain του, καθώς το σημείο (x2,y2) είναι άμεσα εξαρτώμενό του και μπορεί να υπολογιστεί με πράξεις.

(Πχ Κρεβάτι **(x1,y1)** και **x2=x1+100**,**y2=y1+200**)

<u>Από ένα σύνολο μεταβλητών:</u>

Καναπές: D1 = {(0 <= x1 <= 300 – 221, 0<= y1 <= 400 -103}

Κρεβάτι : D2 = {(0<=x1<=300-100,0<=y1<=400-200}

Καρέκλα: D3 = {(0<=x1<=300-41,0<=y1<=400-44)}

Γραφείο: D4 = {(0<=x1<=300-160,0<=y1<=400-80)}

<u>Σύνολο περιορισμών:</u>

Καναπές: **C1** = {((Καναπές.x1> Κρεβάτι.x2 OR Καναπές.x2< Κρεβάτι.x1 OR Καναπές.y1> Κρεβάτι.y2 or Καναπές.y2< Κρεβάτι.y1) **AND** (Καναπές.x1> Καρέκλα.x2 OR Καναπές.x2< Καρέκλα.x1 OR Καναπές.y1> Καρέκλα.y2 or Καναπές.y2< Καρέκλα.y1) **AND** (Καναπές.x1> Γραφείο.x2 OR Καναπές.x2<Γραφείο.x1 OR Καναπές.y1> Γραφείο.y2 or Καναπές.y2< Γραφείο.y1) **AND** (Καναπές.x1>100 OR Καναπές.y1>100))}

Κρεβάτι : **C2** = {((Κρεβάτι.x1> Καναπές.x2 OR Κρεβάτι.x2< Καναπές.x1 OR Κρεβάτι.y1> Καναπές.y2 or Κρεβάτι.y2< Καναπές.y1) **AND** (Κρεβάτι.x1> Καρέκλα.x2 OR Κρεβάτι.x2< Καρέκλα.x1 OR Κρεβάτι.y1> Καρέκλα.y2 or Κρεβάτι.y2< Καρέκλα.y1) **AND** (Κρεβάτι.x1> Γραφείο.x2 OR Κρεβάτι.x2<Γραφείο.x1 OR Κρεβάτι.y1> Γραφείο.y2 or Κρεβάτι.y2< Γραφείο.y1) **AND** (Κρεβάτι.x1>100 OR Κρεβάτι.y1>100))}

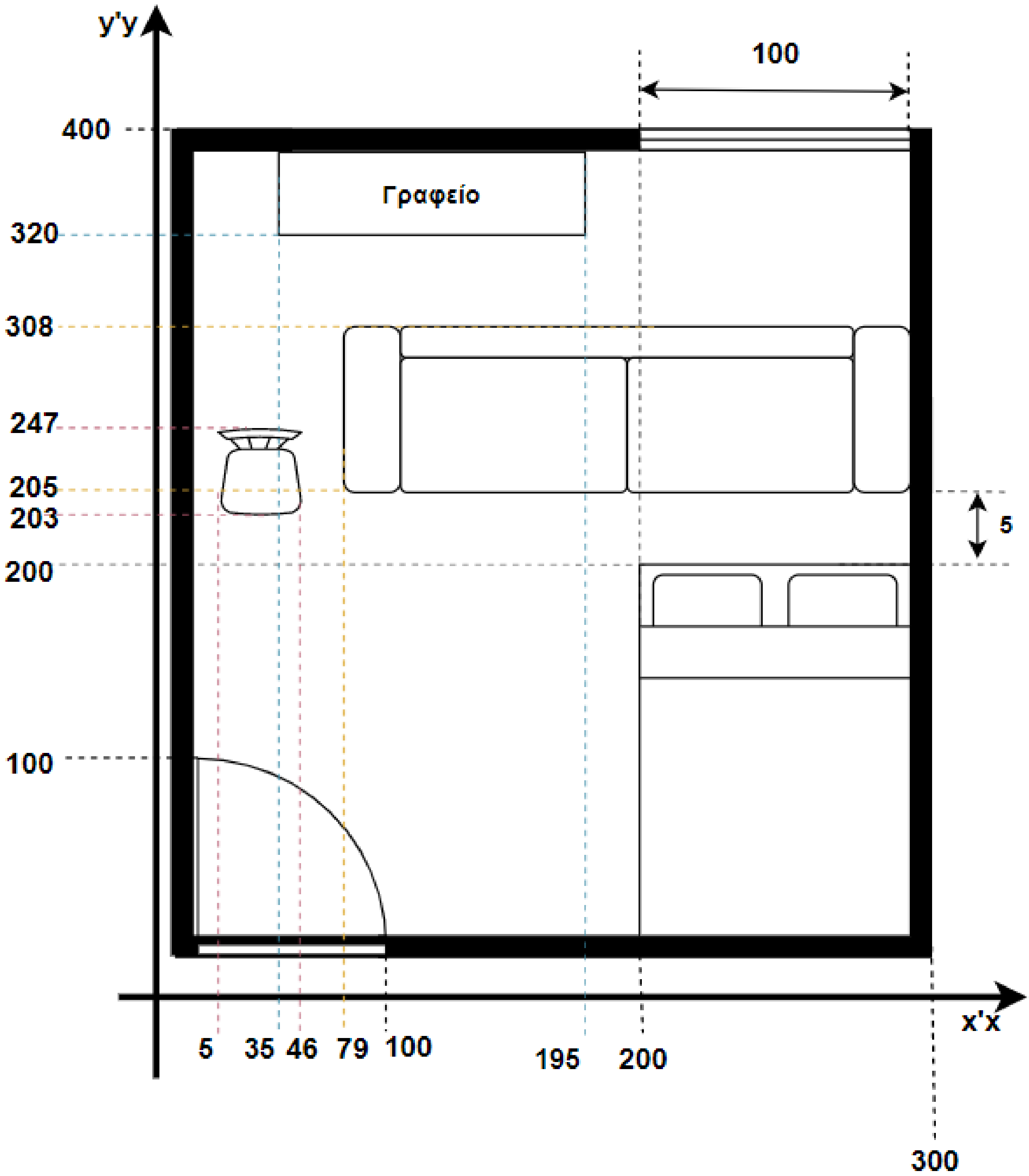
Καρέκλα: **C3** = {(((Καρέκλα.x1>Κρεβάτι.x2 OR Καρέκλα.x2<Κρεβάτι.x1 OR Καρέκλα.y1>Κρεβάτι.y2 or Καρέκλα.y2<Κρεβάτι.y1) **AND** (Καρέκλα.x1> Καναπές.x2 OR Καρέκλα.x2< Καναπές.x1 OR Καρέκλα.y1> Καναπές.y2 or Καρέκλα.y2< Καναπές.y1) **AND** (Καρέκλα.x1> Γραφείο.x2 OR

Καρέκλα.x2<Γραφείο.x1 OR Καρέκλα.y1> Γραφείο.y2 or Καρέκλα.y2< Γραφείο.y1) **AND** (Καρέκλα.x1>100 OR Καρέκλα.y1>100))}

Γραφείο: **C4** = {((Γραφείο.x1>Κρεβάτι.x2 OR Γραφείο.x2<Κρεβάτι.x1 OR Γραφείο.y1>Κρεβάτι.y2 or Γραφείο.y2<Κρεβάτι.y1) **AND** (Γραφείο.x1> Καρέκλα.x2 OR Γραφείο.x2< Καρέκλα.x1 OR Γραφείο.y1> Καρέκλα.y2 or Γραφείο.y2< Καρέκλα.y1) **AND** (Γραφείο.x1>Γραφείο.x2 OR Γραφείο.x2< Καναπές.x1 OR Γραφείο.y1> Καναπές.y2 or Γραφείο.y2< Καναπές.y1) **AND** (Γραφείο.x1>100 OR Γραφείο.y1>100)) **AND** ((Γραφείο.χ2-200)^2 +(Γραφείο.y2-300)^2)^1/2 <= 5}

Για τους περιορισμούς λαμβάνουμε υπόψη αν τα δύο έπιπλα βάση των συντεταγμένων τους κάνουν overlap και το ένα βρίσκεται πάνω στο άλλο.

Μία πιθανή λύση είναι η εξής:



<u>Πρόβλημα 3</u>

1. Το πρόβλημα χρονοπρογραμματισμού ορίζεται από τις μεταβλητές **A1 ,A2 ,A3 ,A4 ,A5.**

Η κάθε μία από τις παραπάνω μεταβλητές έχει ένα σύνολο από τις δυνατές της τιμές το οποίο συμβολίζεται με Di (όπου i = 0,1,2,3,4,5 ανάλογα σε ποια μεταβλητή αναφερόμαστε).

Έχουμε λοιπόν:

D1 = {9:00-10:00,10:00-11:00,11:00-12:00}

D2 = {9:00-10:00,10:00-11:00,11:00-12:00}

D3 = {9:00-10:00,10:00-11:00,11:00-12:00}

D4 = {9:00-10:00,10:00-11:00,11:00-12:00}

D5 = {9:00-10:00,10:00-11:00,11:00-12:00}

Ακόμα έχουμε ένα σύνολο περιορισμών για κάθε μεταβλητή το οποίο ορίζεται ως Ci (όπου i = 0,1,2,3,4,5 ανάλογα σε ποια μεταβλητή αναφερόμαστε).

Έχουμε λοιπόν:

C1 = {A1 > A3}

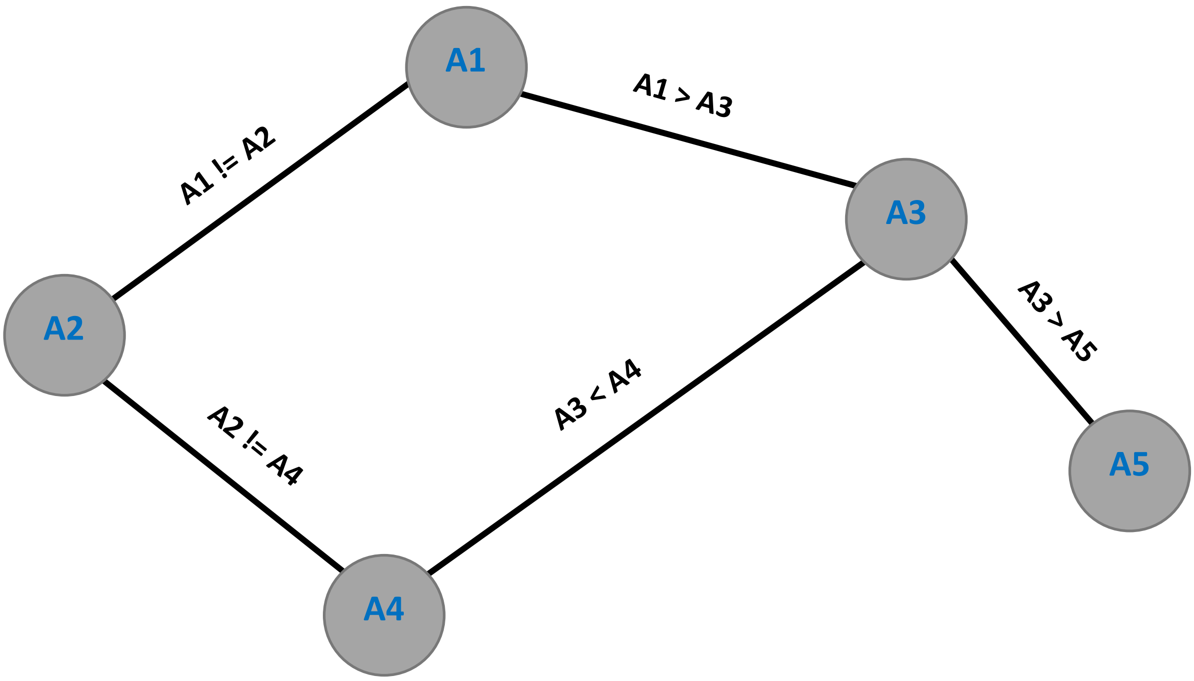
C2 = {A2 != A1}

C3 = { A3 < A4}

C4 = { A4 != 10:00}

C5 = { }

2.



3.

(Ο αλγόριθμος εφαρμόζεται όπως διδάχτηκε και στο φροντιστήριο).

Αρχικά όλες οι ακμές είναι συνεπείς.

Μεταβλητές: A1,A2,A3,A4,A5

Πεδία ορισμού:

- D1 = {9,10,11}
- D2 = {9,10,11}
- D3 = {9,10,11}
- D4 = {9,11}
- D5 = {9,10,11}
  Σειρά ανάθεσης τιμών -> A1,A2,A3,A4,A5
  Σειρά επιλογής τιμών -> 9,10,11

- **A1=9** -> D2 = {10,11} , D3 = {}

Πρέπει να εξετάσουμε (A1,A2) , (A4,A2)

(A1,A2) -> OK

(A4,A2) -> OK

Προχωράμε στην επόμενη τιμή του A1:

- **A1=10** -> D2 = {9,11} , D3 = {9}

Πρέπει να εξετάσουμε (A4,A2) ,(A4,A3),(A5,A3)

(A4,A2) -> OK

(A3,A4) -> OK

(A3,A5) -> ασυνεπής άρα D3 = {} -> προχωράμε στην επόμενη τιμή

Προχωράμε στην επόμενη τιμή του A1:

- **A1 = 11** -> D2 = {9,10} , D3 = {9,10}

Πρέπει να εξετάσουμε (A4,A2) ,(A4,A3),(A5,A3)

(A4,A2) -> OK

(A4,A3) -> OK

(A5,A3) -> OK

- **A2=9** -> D4 = {11}

Πρέπει να εξετάσουμε (A3,A4)

(A3,A4) -> OK

Προχωράμε στην επόμενη τιμή A3:

- **A3=9** -> D5 = {} , D4 = {11}

Προχωράμε στην επόμενη τιμή της **A3 = 10**

- **A3 = 10** -> D5 = {9} , D4 = {11}

Άρα έχουμε A1=11 , A2=9, A3=10, A4=11, A5=9