ΚΩΝΣΤΑΝΤΙΝΟΣ ΝΙΚΟΛΕΤΟΣ

1115201700104

Τεχνητή Νοημοσύνη-Εργασία 3

Πρόβλημα 1

1. **Ορισμός kakuro σαν πρόβλημα ικανοποίησης περιορισμών:**

   Ένα πρόβλημα ικανοποίησης περιορισμών ορίζεται από:

   - Σύνολο από μεταβλητές :

     Έστω $X_{i,j}$ η μεταβλητή με i και j η οριζόντια και η κάθετη συνταταγμένη του πάζλ. Σε αυτές τις μεταβλητές θα γίνει η ανάθεση.

   - Πεδίο μεταβλητών : D = {1, … ,9}
   - Σύνολο από περιορισμούς:

     Για κάθε μεταβλητή $X_{i,j}$ ισχύουν οι εξής περιορισμοί:

     - Οι μεταβλητές X δεν πρέπει να είναι μαύρες θέσεις
     - Το άθροισμα της καθέτου πρέπει να είναι αυτό που αναγράφεται στο αντίστοιχο μαύρο κουτί

       $Sum(X_{i,j})$ = SUM_VERTICAL ,για κάθε j

     - Το άθροισμα της οριζοντίου πρέπει να είναι αυτό που αναγράφεται στο αντίστοιχο μαύρο κουτί

       $Sum(X_{i,j})$ = SUM_HORIZONTAL ,για κάθε i

     - Δεν πρέπει σε μια κάθετο ή οριζόντιο να υπάρχει ο ίδιος αριθμός $X_{i,j} \neq X_{i,j}$ για κάθε i και j

2. **Αλγόριθμοι επίλυσης:**

   Το πρόβλημα του Kakuro λύθηκε και δοκιμάστηκε με πολλούς αλγορίθμους.

   Αρχικά το ζητούμενο πρόβλημα υλοποιήθηκε σε python σύμφωνα με το υλικό που προτάθηκε στο github(aimacode).

- Τεχνικές υλοποίησης
    i. Παρατήρησα ότι υπάρχει μια κλάση Kakuro(Nary) και όπως ζητήθηκε την άλλαξα έτσι ώστε οι περιορισμί να είναι Binary και όχι Nary.Με αυτό τον τρόπο θα μπορούσα να χρησιμοποιήσω τις συναρτήσεις που είναι ήδη υλοποιήμενες στο αρχείο csp.py.
    ii. Δημιούργησα ένα αρχείο kakuro.py στο οποίο υπάρχει η κλάση MyKakuro(CSP) η οποία κληρονομεί από το CSP όλες τις συναρτήσεις - αλγορίθμους που μας ζητήθηκε να εξετάσουμε
    iii. Άλλαξα στην συνάρτηση init της κλάσης Kakuro την διαχείριση των περιορισμών και πρόσθεσα στην δικία μου κλάση ένα επιπλέον στοιχείο : sums , το οποίο είναι μια λίστα στην οποία αποθηκεύω το άθροισμα και ποια κουτάκια πρεπει να αθροίσουν για αυτό το άθροισμα.
    iv. Υλοποιήθηκε μια συνάρτηση KakuroConstraints η οποία ελέγχει αν ισχύουν οι περιορισμοί (επιστέφει true αν ισχύουν και false αν όχι) η οποία είναι όπως προτείνεται στα σχόλια του csp μια συνάρτηση function(A,a,B.b) . Πρώτον ελέγχεται με μια συνάρτηση αν a!=b και ύστερα τα αθροίσματα σε στήλη και κάθετο για τα κουτία που δεν έχει γίνει assignment.
    v. Τέλος , χρησιμοποίω αυτούσια τα παζλ του κακούρο που είναι υλοποίημενα στους αλγορίθμους : Backtracking, Backtracking με FC, Backtracking με FC και MRV, Backtracking με MAC ,AC3, AC4

- Εκτέλεση : python3 kakuro.py

Ύστερα θα ζητηθεί να επιλέξετε την δυσκολία που έλετε να εξετάσουν οι αλγόριθμοι (Δυσκολία 0 ή 1 ή 2 ή 3 ,με 3 το πιο δύσκολο και 0,1 παρόμοιας δυσκολίας)

Για λόγους ευκολίας θα ασχολήθω στην συνέχεια με την σύκριση των αλγόριθμων backtracking με τις προαναφερθείσες μετρικές και δίαφορων βαθμών δυσκολίας.

3. **Σύγκριση αλγορίθμων:**

Χρόνος εκτέλεσης αλγορίθμων

	Backtracking	Backtracking - FC	Backtracking – MAC	Backtracking-MRV-FC	AC3	AC4
Kakuro1	**11.00**	**0.00**	**1.00**	**0.00**	**0.00**	**0.00**
Kakuro2	**1.00**	**0.00**	**1.00**	**0.00**	**1.00**	**0.00**
Kakuro3	**23696.00**	**2.00**	**5.00**	**3.00**	**2.00**	**2.00**
Kakuro4	**807539.00**	**40.00**	**71.00**	**74.00**	**17.00**	**31.00**

***Οι μετρήσεις έγιναν σε μseconds**

Αριθμός αναθέσεων

	Backtracking	Backtracking - FC	Backtracking – MAC	Backtracking-MRV-FC	AC3	AC4
Kakuro1	**97**	**105**	**121**	**113**	**121**	**129**
Kakuro2	**11**	**19**	**35**	**27**	**35**	**43**
Kakuro3	**113376**	**113405**	**113463**	**113434**	**113463**	**113492**
Kakuro4	**4798161**	**4798277**	**4798509**	**4798393**	**4798509**	**4798625**

Παρατηρήσεις:

- Ο αλγόριθμος backtracking χρησιμοποιείται αρχικά χωρίς καμία μετρική και για αυτόν τον λόγο είναι και ο πιο αργός καθώς σε κάθε βήμα εξετάζει όλους τους συνδιασμούς .[Πολ/τα : O(d^ne)]
- Με την προσθήκη του FC,MRV,MAC γίνεται διάδοση περιορισμών κατά την διάρκεια της αναζήτησης ή και νωρίτερα.Είναι ένας αλγόριθμος look-ahead ο οποίος εξετάζει περιορισμούς σε κάθε κατάσταση και μπορεί να κάνεο κλάδεμα και να αποφύγει πολλές συγκρίσεις. Ωστόσο η υλοποίση και εκτέλεση του χρειάζεται μεγαλύτερο χώρο από τον απλό.
- Με την χρήση του MAC ,σε κάθε βήμα διαδίδονται οι περιορισμοί σε κάθε επανάληψη μέχρι να μην υπάρχουν ασυνέπειες ακμής.
- Αλγόριθμος AC-3 (Πολ/τα: O(n^2d^3) ώστοσο όπως παρατηρείται και στις διαφάνειες ο AC-4 είναι ίδιος ή και αργότερος δεσμεύοντας και παραπάνω χώρο
- Τελικά , παρατηρώ ότι ο πιο γρήγορος αλγόριθμος γενικά είναι το backtracking με FC το οποίο κάνει περισσοτερα assignments από τους άλλους αντίστοιχους αλλά έχει μεγάλη απόδοση στον χρόνο.

Πρόβλημα 2

1. **Ορισμός σαν πρόβλημα ικανοποίησης περιορισμών:**

   Ένα πρόβλημα ικανοποίησης περιορισμών ορίζεται από:

   - Μεταβλητές :

     ~ Γ,Μ,Ο : οι ύποπτοι και στις οποίες μεταβλητές θα γίνει ανάθεση αν είναι ένοχοι ή όχι **(Διακριτές τιμές)**

     ~ X_i ,με i μεταξύ του 1 και του 4 , λόγω του αριθμού των συγγραφέων:

     κ. Γιάννη (Χ1) , κ. Μαρία (Χ2) , κ.Όλγα (Χ3) , κ.Μήτσου (Χ4)

     (Συνεχείς τιμές)

     ~ Χρόνος Τ : ο χρόνος που γυρίσαν στο συνέδριο.(D_2) **(Συνεχείς τιμές)**

     *Αυτή η μεταβλητή δηλώνει την ώρα που καθένας από αυτούς έφυγε από το συνέδριο πριν τις 11:00.

   - Πεδίο μεταβλητών : D_1={guilty,no_guilty} , D_2= { 9.00 - 11.00 }
   - Περιορισμοί:
     - Η ανάγνωση γίνεται με σείρα δηλαδή:

       Χ1 , Χ2 , Χ3 , Χ4

     - Η ανάγνωση διαρκεί 30’
     - Από τα δύο παραπάνω προκύπτει ότι οι Χρόνοι εκκίνησης της ανάγνωσης κάθε συγγραφέα είναι:
       a. ΧΕ-Γιάννη = 9.00
       b. ΧΕ-Μαρία = 9.30
       c. ΧΕ-Όλγα = 10.00
       d. ΧΕ-Μήτσος = 10.30
     - Όλοι επέστρεψαν στις 11.00 (Χ-Επιστροφής = 11.00)
     - Χ- Συνέδριο ->Δωμάτιο = [5,10]΄
     - Χ- Συνέδριο ->Θησαυρός = [20,30]΄
     - Χ- Παραβίση = [45-90]΄
     - Ο κ.Μήτσος δεν είναι ύποπτος

2. Λύση προβλήματος:

Ο αστυνόμος Σιέσπης:

Θα πρέπει να σκεφτεί αυτήν την υπόθεση σαν με την βοήθεια ενός αλγορίθμου που χρησιμοποιεί **κλάδεμα μέσω της διάδοσης περιορισμών.** Η ευριστική συνάρτηση θα πρέπει να λειτουργήσει αρχικά επιστρέφοντας τους μικρότερους δυνάτους αριθμούς από τους πιθανούς χρόνους . Αυτό θα διευκολύνει το κλάδεμα .

Άρα:

Αρχή της σκέψης του αστυνόμου(λαμβάνοντας τις minimum τιμές):

α. Ξεκινάει με τον τελευταίο ύποπτο.

Έστω πως έφυγε αμέσως μετά την ανάγνωση του συγγράμματος του δηλαδή:

X3 (ΟΛΓΑ) = 10.30 και κατευθήνθηκε κατευθείαν στο χρηματοκιβώτιο

Οπότε T = 10.30 + 20 + 45 = 11.35 , δεν ανήκει στο domain ,Αδύνατο -> ΚΛΑΔΕΜΑ

Επομένως η όλγα δεν είναι ένοχη και έχει :

X3 = [10.30 , 11.00] και O = no_guilty και T =

Ομοίως και για τους υπόλοιπους:

β. διάστημα = 10.00 + 20 +45= 11.05 , Αδύνατο -> ΚΛΑΔΕΜΑ

X2 = [10.00 , 10.30] και M = no_guilty

γ. διάστημα = 9.30 + 20 +45= 10.35 , Πιθανό

X2 = [10.00 , 10.30] και Γ = guilty και T = 10.35

Επομένως , επείδη οι δύο ένοχοι είναι αδύνατον να προλάβουν να κλέψουν το μήλο για οποιαδήποτε τιμή των T και X , με τη χρήση κλαδέματος συμπεραίνουμε χωρίς αμφιβολία πως ο ένοχος είναι ο κ.Γιάννης.

3. Διάδοση περιορισμών:

Ο αστυνόμος Σιεσπής θα μπορούσε να είχε χρησιμοποιήσει τον αλγόριθμο FC

Σε κάθε βήμα του αλγορίθμου , κάθε φορά που ανατίθεται μια τιμή σε μια μεταβλητή T , ο FC εξετάζει κάθε μεταβλητη X στην οποία δεν έχουν αντεθεί τιμές και διαγράφει όλες τις τιμές της X που είναι ασυνεπείς. Εάν προκύψει κενό σύνολο τότε η τιμή που ανατέθηκε απορρίπτεται.

Πρόβλημα 3

1. Ορισμός σαν πρόβλημα ικανοποίησης περιορισμών:

Ένα πρόβλημα ικανοποίησης περιορισμών ορίζεται από:

- Μεταβλητές : N_0 ,…, N_m : μεταβλητή που δηλώνει την σειρά εκτέλεσης των εργασιών και $T_{i,j}$ όπου είναι ο χρόνος που διήρκησε κάθε εργασία i και κάθε ενέργεια j
- Πεδίο μεταβλητών : Domain του N: D={0, ... , m}

  και domain των χρόνων : D = { d_0, .. , d_m}

- Περιορισμοί:
  - Κάθε ενέργεια m πρέπει να εκτελείται μετά την εκτέλεση των m-1

    T(i,j) : για να εκτελεστεί θα πρέπει στο domain να υπάρχουν τα m-1
  - Κάθε ενέργεια εκτελείται σε ένα μηχάνημα και δεν μπορεί να διακοπεί
  - Κάθε εργασία μπορεί να εκτελέσει μόνο μια ενέργεια την φορά
  - Μια ενέργεια έχει περιορισμένο χρόνο εκτέλεσης d
  - Κάθε εργασία πρέπει να τελειώσει πριν από μια προθεσμία D
  - Κάθε ενέργεια πρέπει να εκτελείται σε ξεχωριστό μηχάνημα

2. Για n=3 και m=4

	m=0	**m=1**	**m=2**	**m=3**	**m=0,...**
n=0	**$E_{0,0}$**			**$E_{0,1}$**	**...**
n=1		**$E_{1,0}$**			**...**
n=2			**$E_{2,0}$**		**...**

3. Παράδειγμα ασυνέπειας

Ασυνέπεια θα υπήρχε αν δεν γινόταν να ολοκληρώθουν οι ενέργειες μέσα στο επιτρεπόμενο χρόνο

4. Αλγόριθμος οπισθοδρόμησης

Θα μπορούσε να χρησιμοποιήθει οπισθοδρόμηση με MRV ,έτσι ώστε να επιλέγεται κάθε φορά η ενέργεια με τις λιγότερες δυνατές νόμιμες ενέργειες και να τελείωσει η εργασία n.

Πρόβλημα 4

Προτασιακή λογική:

*Έστω φ οι λογικές προτάσεις

- **(A & B & C => D) <=> (A => (B => (C => D)))**

Ο πίνακας αλήθειας της παραπάνω πρότασης:

A	B	C	D	(((A ∧ B) ∧ C) → D)	↔	(A → (B → (C → D)
1	1	1	1	1	*1	1
1	1	1	0	0	*1	0
1	1	0	1	1	*1	1
1	1	0	0	1	*1	1
1	0	1	1	1	*1	1
1	0	1	0	1	*1	1
1	0	0	1	1	*1	1
1	0	0	0	1	*1	1
0	1	1	1	1	*1	1
0	1	1	0	1	*1	0
0	1	0	1	1	*1	1
0	1	0	0	1	*1	1
0	0	1	1	1	*1	1
0	0	1	0	1	*1	1
0	0	0	1	1	*1	1
0	0	0	0	1	*1	1

1. Η πρόταση είναι έγκυρη κάθως για κάθε ερμηνεία της πρότασης φ ισχύει η ισοδυναμία(το *1 σε κάθε υπολογισμό στον πίνακα αλήθειας) και επομένως *I(φ) = true*
2. Η πρόταση είναι ικανοποίησιμη καθώς υπάρχει τουλάχιστον μια ερμηνεία για την οποία ισχύει *I(φ) = true*
3. -
4. Η πρόταση αυτή (φ) έχει μια τουλάχιστον ερμηνεία I με I(φ)=true ,που σημαίνει ότι η I ικανοποιεί την φ έχει ένα τουλάχιστον μοντέλο και η *I είναι ένα μοντέλο της φ*
5. Επειδή η πρόταση είναι έγκυρη είναι και ταυτολογία
6. Horn: (Γενικά : Είναι η πρόταση η οποία περιέχει το πολύ ένα θετικό λεκτικό) Με απαλειφή των συνεπαγωγών και χρησιμοποιώντας τους κανόνες: νόμος de morgan , προσεταιριστικότητα του ^ και κάνοντας απαλοιφή αμφίδρομης υποθέτικης πρότασης συμπαιρένω πως μόνο το D είναι ένα θετικό λεκτικό.Η πρόταση αυτή είναι φράση horn.

- **A&(A => B) & (A => -B)**

Ο πίνακας αλήθειας της παραπάνω πρότασης:

A B	(A ∧ (A → B)) ∧ (A → ¬B)
1 1	1 1 *0 0 0
1 0	0 0 *0 1 1
0 1	0 1 *0 1 0
0 0	0 1 *0 1 1

1. Η πρόταση δεν είναι έγκυρη κάθως δεν υπάρχει ερμηνεία της πρότασης φ που να ισχύει η ισοδυναμία(το *0 σε κάθε υπολογισμό στον πίνακα αλήθειας) και επομένως δεν ισχυεί *I(φ) = true*
2. -
3. Είναι μη ικανοποιήσιμη καθώς δεν υπάρχει καμία ερμηνεία που να ισχύει *I(φ)=true*
4. Δεν έχει κανένα μοντέλο
5. Δεν είναι ταυτολογία
6. Είναι φράση horn,καθώς με απαλοιφή των 2 συνεπαγωγών καταλήγω σε πρόταση με μόνο ένα θετικό λεκτικό, το Α.

- **(A | B) & (-A&C) & -B & -C**

Ο πίνακας αλήθειας της παραπάνω πρότασης:

Πρόσθεσα παρενθέσεις για να διακρίνω την προτεραιότητα:

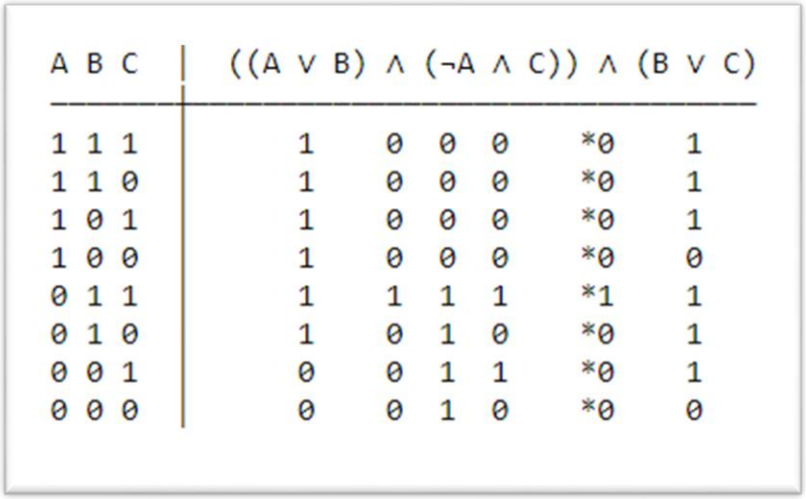
A B C	(((A ∨ B) ∧ (¬A ∨ C)) ∧ ¬B) ∧ ¬C
1 1 1	1 1 0 1 0 0 *0 0
1 1 0	1 0 0 0 0 0 *0 1
1 0 1	1 1 0 1 1 1 *0 0
1 0 0	1 0 0 0 0 1 *0 1
0 1 1	1 1 1 1 0 0 *0 0
0 1 0	1 1 1 1 0 0 *0 1
0 0 1	0 0 1 1 0 1 *0 0
0 0 0	0 0 1 1 0 1 *0 1

Ομοίως με την προηγούμενη πρόταση

Με την διαφορά ότι δεν είναι πρόταση Horn(Παραπάνω από ένα θετικό λεκτικό).

- **(A | B) & (-A|C) & (B | C)**

Ο πίνακας αλήθειας της παραπάνω πρότασης:



A B C	((A ∨ B) ∧ (¬A ∧ C)) ∧ (B ∨ C)					
1 1 1	1	0	0	0	*0	1
1 1 0	1	0	0	0	*0	1
1 0 1	1	0	0	0	*0	1
1 0 0	1	0	0	0	*0	0
0 1 1	1	1	1	1	*1	1
0 1 0	1	0	1	0	*0	1
0 0 1	0	0	1	1	*0	1
0 0 0	0	0	1	0	*0	0

1. Δεν είναι έγκυρη καθώς δεν ισχύει I(ϕ) = true για κάθε ερμηνεία
2. Υπάρχει μια ερμηνεία για την οποία να ισχύει οπότε είναι ικανοποιήσιμη.
3. -
4. Έχει ένα μοντέλο
5. Δεν είναι ταυτολογία
6. Δεν είναι φράση horn.

Πρόβλημα 5

Σύμφωνα με την ανάλυση έχουμε:

1. Μετατροπή των δοσμένων προτάσεων σε συζευκτική κανονική μορφή (CNF):

   Με τα εξής βήματα :

   1. Απαλοιφή του <=> με την (α=>β)^(β=>α)
   2. Απαλοιφή του => με την -α v β
   3. Κανόνας de morgan

   - A ^ (B <=> C)

     A ^ ((B=>C)^(C=>B))

A ^ ((-B v C) ^ (-C v B))

(A ^ (-B v C)) ^ (A^ (-C v B))

(A ^ (-B) v (A^ C))^(A^ (-C)) v (A ^ B))

- (A^B) <=> (A^C)

((A^B)=> (A^C)) ^ ((A^C)=>((A^B))

((- (A^B) v (A^C)) ^ (-(A^C) v ((A^B))

[(-A v -B) v (A^C)] ^[(-A v -C) v (A^B)]

2. Με παραγοντοποίηση των δύο παραπάνω προκύπτει κενή πρόταση,επομένως η πρόταση 1 καλύπτει λογικά την 2

Πρόβλημα 6

1. Καλά ορισμέμη
2. Δεν είναι καλά ορισμένη
3. Καλά ορισμένη
4. Καλά ορισμένη
5. Δεν είναι καλά ορισμένη