Μάθημα Τεχνητά Νοημοσύνη

Εργασία 3

Ονομα : Μαρίνος
Επίθετο : Αγαπίου
AM : 1115201400002

Αρχικα για το προβλημα εχω δημιουργησει τα αρχεια kakuro.py που περιέχει τον κώδικα και το αρχείο input_puzzles.py που περιέχει σε μορφή λίστας ενδεικτικές εισόδους για το πρόβλημα μας.

Στο αρχείο kakuro.py έχω ορίσει class kakuro(CSP) η οποία εχει τα βοηθητικά εξής πεδία:

self.puzzle : μεταβλητή που αποθηκεύω το puzzle

self.condict : μεταβλητή dictionary απο dictionarys που αποθηκεύω για κάθε μεταβλητή τον περιορισμό γραμμής , τον περιορισμό στήλης , τις συντεταγμένες του στοιχείου που προέρχεται ο περιορισμός γραμμής και τις συντεταγμένες του στοιχείου που προέρχεται ο περιορισμός στήλης.

self.constrain_variables : μεταβλητή που για κάθε στοιχείο περιορισμού (πχ (‘’,5)) αποθηκεύω σε dictionary τις μεταβλητές γραμμής και στήλης που ανήκουν στον περιορισμό αυτό καθώς και τις τιμες του περιορισμού αυτού (‘’ , 5)

Συνάρτηση __init_ : η οποία αρχικοποιεί τα κατάλληλα δεδομένα για την συνέχεια.

Συνάρτηση check_if_everything_ok : η οποία ελέγχει στο τέλος αφού βρεθεί λύση στο πρόβλημα αν οι τιμές που πείραν οι μεταβλητές ειναι έγκυρες με βάση τους περιορισμούς.

Συνάρτηση display : η οποία απλά εκτυπώνει το board σε μορφή human readable.

Συνάρτηση Kakuro_constrain(A,α,B,β) : η οποία έχει υλοποιηθεί με βάση δυαδικό περιορισμό και επιστρέφει False αν υπάρχει conflict στον περιορισμό και True αν δεν υπάρχει.
Για την συνάρτηση αυτή έχω τους εξής περιορισμούς :

1) Αν οι τιμές των α, β ειναι ίδιες τότε επιστρέφω False

2) Αν οι μεταβλητές A,B είναι μόνες τους στον περιορισμό είτε γραμμής είτε στήλης τότε πρεπει το άθροισμα τους να είναι ίσο με την τιμή του περιορισμού αυτού.

3) Αν οι μεταβλητές A,B ΔΕΝ είναι μόνες τους στον περιορισμό τοτε:

3α) Αν καμία από τις υπόλοιπες μεταβλητές που εμπλέκεται στον περιορισμό δεν έχει πάρει τιμή τότε πρέπει το άθροισμα (α+β) να είναι μικρότερο από την τιμή του περιορισμού (διαφορετικά οι υπόλοιπες μεταβλητές δεν μπορούν να πάρουν τιμή)

3β) Αν όλες οι υπόλοιπες μεταβλητές που εμπλέκεται στον περιορισμό έχουν πάρει τιμή τότε πρέπει το άθροισμα (α+β) + (το άθροισμα των τιμών των υπόλοιπων μεταβλητών) να ειναι ίσο με την τιμή του περιορισμού

3γ) Αν στις υπόλοιπες μεταβλητές που εμπλέκεται στον περιορισμό υπάρχουν και μεταβλητές που έχουν πάρει τιμή και μεταβλητές που δεν έχουν πάρει τοτε το άθροισμα (α+β) + (το άθροισμα των τιμών των υπόλοιπων μεταβλητών) να ειναι μικρότερο με την τιμή του περιορισμού (διαφορετικά οι υπόλοιπες μεταβλητές που δεν έχουν πάρει ακόμα τιμή δεν μπορούν να πάρουν τιμή)

Έχω επίσης υλοποιήσει τις εξής βοηθητικές συναρτήσεις:

1) get_neighbors(puzzle): η συνάρτηση αυτη παίρνει σαν είσοδο το παζλε και επιστρέφει ένα tuple απο δύο dictionary .Στο πρώτο αποθηκεύω για κάθε μεταβλήτη τους γείτονες της και το δεύτερο απλα το αρχικοποιώ και το χρησιμοποιώ στην συνέχεια.

2) get_constrains(puzzle,condict): η συνάρτηση αυτή αρχικοποιεί το dictionary self.condict (για το οποίο μίλησα παραπάνω) και δημιουργεί το dictionary self.constrain_variables (για το οποίο μίλησα παραπάνω)

Αλγόριθμοι που επέλεξα :

1) MAC (maintaining arc consistency): επέλεξα τον αλγόριθμο αυτό διότι η όπως έχουμε διδαχθεί κιόλας η έννοια της συνέπειας ακμών μας δίνει ένα πολύ ισχυρότερο εργαλείο διάδοσης περιορισμών. Με τον αλγόριθμο αυτό καταφέρνουμε να εντοπίσουμε μια ασυνέπεια πολύ πιο γρήγορα σε σχεση με τους άλλους και αυτό συμβαίνει γιατί ο αλγόριθμος κάθε φορα που αναθέτει τιμή σε μία μεταβλητή ελέγχει, όλες τις ακμές που σχετίζονται με την μεταβλητή αυτή ,για τυχόν ασυνέπειες. Επίσης συμπαιρένουμε και από το πίνακα του αμέσως επόμενου ερωτήματος οτι ο αλγόριθμος MAC είναι ο πιο ισχυρός.

2) FC(forward checking): επέλεξα τον αλγόριθμο αυτό διότι είναι αρκετά ισχυρός και ταιριάζει στο συγκεκριμένο πρόβλημα αρκετά.Αυτό γιατι ο αλγόριθμος αυτός καθε φορά που αναθέτει τιμή σε μια μεταβλητή αφαιρεί απο το πεδίο τιμών ,των μεταβλητών που δεν έχουν πάρει ακόμα τιμή , όσες τιμές δεν είναι συνεπείς με την ανάθεση αυτή . Ετσι μειώνεται αρκετα ο παράγοντας διακλάσωσης και μειώνεται και ο χρόνος επίλυσης του προβλήματος σε σχέση με εναν απλό backtraking αλγόριθμο.

3) FC + MRV : στον προηγούμενο αλγόριθμο επέλεξα να προσθέσω και τον ευρετικό μηχανισμό επιλογής μεταβλητών MRV έτσι ώστε να τον συγκρίνω κυρίως με τον απλό FC . Αυτό γιατι ο MRV επιλέγει κάθε φορά την μεταβλητή με τις λιγότερες νόμιμες επιλογές μειώνοντας ετσι τον παράγοντα διακλάδωσης και πράγματι είναι πιο αποδοτικός απο τον απλό FC όπως καταγραφω και στο επόμενο ερώτημα.

4) BACKTRACKING + MRV: επέλεξα τον αλγόριθμο αυτό διότι ήθελα να πειραματιστώ και με ένα κλασσικό backtraking αλγοριθμο ,ενισχυμένο μεν από τον MRV έτσι ώστε να είναι πιο αποδοτικός σε χρόνο και χώρο, για να τον συγκρίνω με τους παραπάνω καλυτερούς και πιο αποδοτικούς αλγορίθμους.Όπως θα φανεί και στο επόμενο ερώτημα ο αλγόριθμος αυτός είναι χειρότερος από τους όλους τους παραπάνω σε θέματα απόδοσης.