Τεχνητή Νοημοσύνη – Εργασία 3

Αθανασία Τουρνάκη
ΑΜ: 1115201600172

Πρόβλημα 1:

1)

Το CSP του KenKen puzzle μοντελοποιήθηκε με χρήση της μεθόδου Dual Transformation, που συζητήθηκε στο μάθημα και αναλύεται στο κείμενο που δόθηκε ως αναφορά. Πιο αναλυτικά, το πρόβλημα ορίζεται ως εξής:

- **Ordinary μεταβλητές:** Τα κελιά (n x n συνολικά, ανάλογα με το μέγεθος του πίνακα).

- **Dual μεταβλητές:** Οι κλίκες. Πρόκειται για tuples που περιέχουν ζεύγη συντεταγμένων που αντιστοιχούν στα κελιά που συμμετέχουν στις κλίκες, ενώ αποθηκεύεται για την κάθε κλίκα η πράξη που χρησιμοποιείται μέσα σε αυτήν και ο αριθμός-στόχος της. Ουσιαστικά, αυτές οι μεταβλητές χρησιμοποιούνται στη θέση των περιορισμών που θα υπήρχαν μεταξύ των ordinary μεταβλητών-κελιών που ανήκουν στην ίδια κλίκα. Ο αριθμός των dual μεταβλητών εξαρτάται από τον πίνακα που δίνεται.

- **Πεδία:** Το πεδίο μίας dual μεταβλητής περιέχει όλους τους συνδυασμούς τιμών(κάθε τιμή αντιστοιχίζεται σε ένα ζεύγος συντεταγμένων), τέτοιους ώστε να μην παραβιάζεται ο εσωτερικός περιορισμός της κλίκας, δηλαδή στο πεδίο της μεταβλητής περιέχονται μόνο tuples, μεγέθους ανάλογου της κλίκας στην οποία αντιστοιχούν, τιμών για τις οποίες ισχύει ότι η εφαρμογή της πράξης της κλίκας σε αυτές οδηγεί στον αριθμό-στόχο και επιπλέον ότι δε θα υπάρχει παραπάνω από μία φορά ο ίδιος αριθμός σε μία γραμμή ή στήλη(εσωτερικά της κλίκας).

- **Περιορισμοί:** Ανά δύο κλίκες, αν ένα κελί που ανήκει στην πρώτη κλίκα είναι στην ίδια σειρά ή στήλη με ένα κελί που ανήκει στην δεύτερη κλίκα, αυτά δε μπορούν να περιέχουν τον ίδιο αριθμό.

2)

Για την υλοποίηση της μοντελοποίησης που περιγράφηκε παραπάνω, δημιουργήθηκε ένα αρχείο kenken.py. Στο αρχείο αυτό περιέχονται τρεις χάρτες διαφόρων μεγεθών και επιπέδων δυσκολίας που περιγράφονται παρακάτω καθώς επίσης και ο ορισμός μιας κλάσης KenKen και 2 βοηθητικών συναρτήσεων createDomain και createDomainRec, που θα συζητηθούν αργότερα. Μερικές πληροφορίες σχετικά με την υλοποίηση:

- Για την δημιουργία ενός χάρτη, μέσα στην συνάρτηση του εκάστοτε χάρτη ορίζεται μία λίστα cages από tuples που περιέχουν ζεύγη συντεταγμένων που περιγράφουν τη θέση όλων των κελιών που περιέχονται στην κλίκα. Για κάθε εκχώρηση στην λίστα αυτή(δηλαδή για κάθε κλίκα) αποθηκεύονται σε ένα λεξικό info η πράξη που ορίζει η κλίκα και ο αριθμός-στόχος της. Τέλος, σε μία απλή μεταβλητή size αποθηκεύεται το μέγεθος μίας διάστασης του χάρτη. Όλα αυτά επιστρέφονται με ένα tuple μορφής (tuple(cages), info, size).

- Για να ελέγξει τη σωστή λειτουργία του προγράμματος ο χρήστης αφού εκτελέσει το αρχείο kenken.py αρκεί να επιλέξει το puzzle και τον αλγόριθμο που επιθυμεί να τρέξει.

- Κλάση KenKen:

  - __init(self, matrix): Η κλάση κατά την αρχικοποίηση της παίρνει σαν όρισμα έναν πίνακα που δημιουργείται όπως περιγράφηκε παραπάνω. Η μεταβλητή variables αποθηκεύει το tuple που περιέχει τις κλίκες. Αυτές θα είναι οι μεταβλητές του προβλήματος. Η μεταβλητή info κρατάει το λεξικό με τις πληροφορίες για κάθε μεταβλητή-κλίκα. Τέλος, η μεταβλητή n κρατάει το μέγεθος του πίνακα. Επιπλέον, αρχικοποιείται μία μεταβλητή self.constraint_call σε 0, η οποία θα χρησιμοποιηθεί για τις μετρήσεις. Μετά από αυτές τις αναθέσεις γίνονται οι αρχικοποιήσεις των λεξικων domains και neighbors στις οποίες αποθηκεύονται τα πεδία των μεταβλητών και οι γείτονές τους αντίστοιχα. Για την αρχικοποίηση των πεδίων χρησιμοποιείται μία βοηθητική συνάρτηση που θα αναλυθεί παρακάτω. Για την αρχικοποίηση των γειτόνων, απλά προστίθεται στη λίστα των γειτόνων μίας κλίκας μία άλλη κλίκα που περιέχει την ίδια τιμή σε ένα κελί της με αυτή ενος κελιού της πρώτης κλίκας και τα κελιά ανήκουν στην ίδια σειρά ή στήλη. Τέλος καλείται η αρχικοποίηση ενός προβλήματος CSP με τις μεταβλητές, τα πεδία, τους γείτονες και τους περιορισμούς που ορίστηκαν.

  - all_different_constraints(self, A, a, B, b): Πρόκειται για την συνάρτηση περιορισμών. Έχει προστεθεί σαν μέθοδος της κλάσης KenKen, για να μπορεί να επέμβει στην μεταβλητή self.constraint_calls και να την αυξήσει. Δημιουργούνται ζεύγη δεικτών στα περιεχόμενα των μεταβλητών-κλικών A και B. Αν ένα κελί της κλίκας A έχει την ίδια τετμημένη ή τεταγμένη με ένα κελί της κλίκας B και ταυτόχρονα οι τιμές που περιέχονται στα κελιά αυτά είναι ίδιες, η συνάρτηση επιστρέφει False, δηλαδή ο κανονισμός δεν ικανοποιείται. Σε αντίθετη περίπτωση, η συνάρτηση επιστρέφει True.

  - displayMatrix(self, solution): Η συνάρτηση αυτή εκτυπώνει τον αρχικό πίνακα μη συμπληρωμένο. Για να ξεχωρίζει ο χρήστης την τοποθέτηση των κλικών πάνω τον πίνακα και τα χαρακτηριστικά της κάθε κλίκας, στην θέση κάθε κελιού που ανήκει σε μία κλίκα εκτυπώνεται η πράξη και ο αριθμός-στόχος της. Επίσης, παίρνει σαν όρισμα το λεξικό solution που επιστρέφει ο αλγόριθμος που εφαρμόζεται και εκτυπώνει τον τελικό χάρτη τοποθετώντας σε κάθε κελί την τιμή που του έχει ανατεθεί.

- Βοηθητικές συναρτήσεις:

  - createDomain(result, setrange, variable, vlength, vinfo): Πρόκειται για μία μεταβατική συνάρτηση. Η μεταβλητή setrange έχει οριστεί κατά την κλήση της __init__ στην κλάση και, ανάλογα με το μέγεθος του πίνακα δημιουργεί μία λίστα [1.....n]. Η συνάρτηση κρατάει σε μία μεταβλητή n το μήκος αυτής της λίστας και καλεί την αναδρομική συνάρτηση createDomanRec(result, setrange, n, variable, vlength, info).
  - createDomanRec(result, setrange, setlength, variable, vlength, info): Η συνάρτηση αυτή χρησιμοποιείται για την δημιουργία του πεδίου μίας μεταβλητής. Η μεταβλητή result που δίνεται σαν όρισμα αποθηκεύει το πεδίο που προκύπτει. Ουσιαστικά, εντοπίζει αναδρομικά όλους τους δυνατούς συνδυασμούς τιμών που μπορούν να αποθηκευτούν στα κελιά μίας κλίκας αποκλείοντας αυτούς που παραβιάζουν τους περιορισμούς της. Η επιβολή αυτών των περιορισμών γίνεται όταν έχουν δημιουργηθεί όλοι οι συνδυασμοί τιμών. Τότε, εάν δύο κελία που ανήκουν στην ίδια κλίκα έχουν ίδια τετμημένη η τεταγμένη και η τιμή που τους έχει ανατεθεί από την συνάρτηση είναι ίδια η συνάρτηση κάνει return και δεν επιστρέφει τον συνδυασμό. Επιπλέον, ανάλογα με την πράξη και τον αριθμό-στόχο της κλίκας, αν οι τιμές που έχουν ανατεθεί στα κελιά που ανήκουν σε αυτήν δεν επιστρέφουν το σωστό αποτέλεσμα, ο

συνδυασμός δεν αποθηκεύεται. Με αυτόν τον τρόπο γίνεται ένα “κλάδεμα” του πεδίου τιμών των μεταβλητών.

3)

Μετά την μοντελοποίηση του προβλήματος όπως περιγράφηκε παραπάνω, έγιναν κάποια πειράματα για την εξαγωγή συμπερασμάτων σχετικά με την συμπεριφορά και την απόδοση των αλγορίθμων επίλυσης CSPs: BT, BT+MRV, FC, FC+MRV και MAC.

Έγιναν πειράματα σε 4 πίνακες: Έναν μικρής δυσκολίας πίνακα μεγέθους 3x3(easy3x3, όπως ορίζεται στο αρχείο kenken.py), έναν μεγάλης δυσκολίας μεγέθουε 5x5(hard5x5), έναν μέτριας δυσκολίας μεγέθους 6x6 που δόθηκε στην εκφώνηση της άσκησης(given6x6) και έναν μεγάλης δυσκολίας μεγέθους 9x9(expert9x9).

Σαν μέτρα σύγκρισης χρησιμοποιήθηκαν ο χρόνος που χρειάστηκε για την εκτέλεση κάθε αλγορίθμου, η τιμή που επιστρέφει η μεταβλητή nassigns - που ορίζεται στο csp.py και λειτουργεί ως μετρητής των φορών που ανατέθηκε μία τιμή σαν λύση του CSP προβλήματος πριν βρεθεί η πραγματική λύση - και ο αριθμός των κλήσεων της συνάρτησης περιορισμών που έχει οριστεί μέσα στην κλάση Kenken, ο οποίος κρατάται μέσα σε μία μεταβλητή constraint_calls εσωτερική της κλάσης.

Παρακάτω παρουσιάζονται τα αποτελέσματα.

Easy3x3:

Αλγόριθμος	Χρόνος(sec)	nassigns	constraint_calls
BT	~ 0, 0007	4	45
BT + MRV	~ 0, 0008	4	42 ή 45
FC	~ 0, 0007	4	54
FC + MRV	~ 0, 0008	4	54 ή 56
MAC	~ 0, 0009	4	77

Hard5x5:

Αλγόριθμος	Χρόνος(sec)	nassigns	constraint_calls
BT	~ 0, 028	126	4526
BT + MRV	~ 0, 01	20 - 30	900 - 1200
FC	~ 0, 01	36	1555
FC + MRV	~ 0, 007	12 - 16	400 - 600
MAC	~ 0, 03	16	6952

Given6x6:

Αλγόριθμος	Χρόνος(sec)	nassigns	constraint_calls
BT	~ 0, 03	58	2588
BT + MRV	~ 0, 033	Μη σταθερό(συνήθως 20-100)	1000 - 7000
FC	~ 0, 02	39	1571
FC + MRV	~ 0, 022	15	900 - 2500

MAC	~ 0, 025	15	3499

Expert9x9:

Αλγόριθμος	Χρόνος(sec)	nassigns	constraint_calls
BT	~ 55	25187	14850905
BT + MRV	> 80	> 100000	26651547
FC	~ 0, 22	461	47160
FC + MRV	~ 0, 5	200 - 1100	100000 - 200000
MAC	~ 1, 3	52	368485

Τα συμπεράσματα που μπορούν να εξαχθούν από τα παραπάνω πειράματα για κάθε αλγόριθμο που εφαρμόστηκε είναι:

- **BT:** Ο αλγόριθμος απλής ή χρονολογικής υπαναχώρησης φαίνεται να είναι πιο αργός από τους περισσότερους αλγορίθμους. Αυτό είναι προφανές στους πίνακες μεγαλύτερου μεγέθους. Επιπλέον, φαίνεται να κάνει πολλές περιττές αναθέσεις και να καλεί αρκετές φορές την συνάρτηση περιορισμών. Αυτό συμβαίνει επειδή ο αλγόριθμος BT επιλέγει τυχαία μία μεταβλητή από τη λίστα μεταβλητών για να της αναθέσει μια τιμή, χωρίς κάποιο ιδιαίτερο κριτήριο, και υπαναχωρεί μόνο όταν μία τιμή δεν είναι αποδεκτή με βάση τις τιμές που έχουν ήδη ανατεθεί σε άλλες μεταβλητές.
- **BT+MRV:** Ο αλγόριθμος BT+MRV φαίνεται να μην επιφέρει σταθερή βελτίωση του BT. Στους πρώτους τρεις πίνακες η βελτίωση στις κλήσεις της συνάρτησης παριορισμών είναι πιο αισθητή, ενώ η διαφορά χρόνων είναι αμελητέα. Η βελτίωση στις αναθέσεις υπάρχει μεν, αλλά ειδικά στον πίνακα 6x6 δεν είναι πάντα σταθερή. Τέλος, στον πίνακα 9x9 παρατηρώ ότι η εφαρμογή του BT+MRV επιφέρει χειρότερα αποτελέσματα από τον BT σε όλες τις απόψεις. Όλα αυτά είναι αναμενόμενα, διότι ο ευρετικός μηχανισμός MRV, αν και είναι χρήσιμο εργαλείο, δεν αποτελεί πανάκεια. Ο MRV προσθέτει στον BT ένα κριτήριο για την επιλογή της επόμενης μεταβλητής στην οποία θα ανατεθεί τιμή: να είναι εκείνη με τον μικρότερο αριθμό νόμιμων τιμών που μπορεί να λάβει. Σκοπός του είναι συνήθως η ελαχιστοποίηση των διακλαδώσεων του δέντρου αποφάσεων και η εύρεση της λύσης με πιο αποδοτικό τρόπο. Όμως, η επιλογή της μεταβλητής με το μικρότερο πεδίο δεν μας εξασφαλίζει πάντα πιο γρήγορη εύρεση της λύσης, ενώ, όπως σχολιάστηκε παραπάνω, μπορεί να οδηγήσει σε περιττές αναθέσεις και κλήσεις συναρτήσεων. Για αυτόν τον λόγο, ο BT+MRV αλγόριθμος ίσως είναι καλύτερο να εφαρμόζεται για πίνακες μικρότερου μεγέθους.
- **FC:** Ο αλγόριθμος FC απαιτεί για κάθε ανάθεση μιας τιμής x σε μεταβλητή να ελέγχεται για κάθε μεταβλητή στην οποία δεν έχει ακόμα ανατεθεί τιμή αν οι τιμές στο πεδίο της είναι συνεπείς με την x. Αν κάποια από αυτές δεν είναι συνεπής, διαγράφεται. Εαν μετά την διαγραφή των ασυνεπών τιμών το πεδίο μιας μεταβλητής είναι κενό, τότε η x απορρίπτεται. Κατά κύριο λόγο, ο FC βελτιώνει σε μεγάλο βαθμό τα αποτελέσματα του BT, ειδικά στους πίνακες μεγαλύτερου μεγέθους. Σε παρόμοιο χρόνο καταφέρνει να εντοπίσει τη λύση με μικρότερο αριθμό αναθέσεων και κλήσεων της συνάρτησης περιορισμού(πέρα απο το παράδειγμα του πίνακα 3x3). Ειδικά στον πίνακα 9x9 η βελτίωση που επιφέρει είναι εντυπωσιακή, τόσο όσον αφορά τον χρόνο, όσο και τα υπόλοιπα κριτήρια. Ο FC, με τον μηχανισμό του, προβλέπει εαν μία ανάθεση τιμής σε μία μεταβλητή είναι αποδεκτή με βάση τα

πεδία τιμών των υπολοίπων μεταβλητών και την απορρίπτει εγκαίρως, γλιτώνοντας πολλές περιττές κινήσεις.

- **FC+MRV:** Ο αλγόριθμος FC+MRV επιλέγει να αναθέσει τιμή(έστω x) στην μεταβλητή που έχει το μικρότερο πεδίο(MRV) και έπειτα εξετάζει για όλες τις μεταβλητές-γείτονες αυτής στις οποίες δεν έχει ανατεθεί τιμή, εάν κάποια τιμή στο πεδίο τους είναι ασυνεπής με την x. Αν είναι, την αφαιρούν. Αν μετά την αφαίρεση της, το πεδίο μιας μεταβλητής μείνει κενό, τότε η x απορρίπτεται και στην μεταβλητή ανατίθεται η επόμενη τιμή στο πεδίο της. Από τα παραπάνω πειράματα είναι προφανές ότι ο FC+MRV είναι πολύ πιο αποδοτικός από όλους τους προηγούμενους αλγορίθμους που εξετάστηκαν πετυχαίνοντας καλύτερους χρόνους(σχεδόν πάντα) και λιγότερες αναθέσεις τιμών και κλήσεων της συνάρτησης περιορισμών. Για τον πίνακα 9x9, οι αναθέσεις δεν είναι σταθερα μικρότερες ή μεγαλύτερες, αλλά ο χρόνος και οι κλήσεις των περιορισμών είναι μεγαλύτερα από ότι στον FC. Αυτό οφείλεται στο γεγονός ότι η χρήση του MRV δεν εγγυάται πάντα βελτίωση, αν και η πιθανότητα βελτίωσης που προσφέρει σε συνδυασμό με τον FC είναι αρκετά μεγάλη, εφόσον με την επιλογή των μεταβλητών με τα μικρότερα πεδία δεν είναι σίγουρο ότι θα βρεθεί πιο γρήγορα η λύση, ενώ μπορεί να γίνουν και κάποιες περιττές αναθέσεις και κλήσεις της συνάρτησης περιορισμών, όπως είχε αναφερθεί και στον σχολιασμό του BT+MRV. Πάντα υπάρχει περίπτωση, ειδικά όταν οι μεταβλητές είναι περισσότερες και άρα είναι περισσότερες και οι τιμές που μπορούν να τους ανατεθούν, τα αποτελέσματα του FC+MRV να μην είναι καλύτερα από αυτά του FC, δηλαδή η επιλογή της μεταβλητής με το μικρότερο πεδίο να μην επιφέρει καλύτερα αποτελέσματα από την επιλογή μιας τυχαίας μεταβλητής, όπως φαινεται στο παράδειγμα του πίνακα 9x9, ενώ παίζει ρόλο και η δομή του προβλήματος.
- **MAC:** Ο αλγόριθμος MAC μετά την ανάθεση μίας τιμής σε μία μεταβλητή ελέγχει εάν διατηρείται η συνέπεια ακμής μεταξύ της μεταβλητής και των μεταβλητών-γειτόνων της με τη χρήση του αλγορίθμου AC-3. Εάν εντοπίσει κάποια ασυνέπεια, τότε αναθέτει στην μεταβλητή την επόμενη τιμή στο πεδίο της. Από τη στιγμή που γίνονται τόσες πολλές συγκρίσεις για την ικανοποίηση των περιορισμών είναι αναμενόμενο στα αποτελέσματα των πειραμάτων ο MAC να έχει σχεδόν πάντα περισσότερες κλήσεις της συνάρτησης περιορισμών από τους υπόλοιπους αλγορίθμους. Αυτό συνεπάγεται και μεγαλύτερο χρόνο από μερικούς άλλους, συγκεκριμένα τους FC και FC+MRV. Ταυτόχρονα όμως, οι αναθέσεις λύσης που κάνει είναι λιγότερες από ότι άλλων αλγορίθμων, εφόσον πολλές απορρίπτονται κατά τον έλεγχο συνέπειας.

4)

min_conflicts:

Puzzle	Χρόνος(sec)	nassigns	constraint_calls	Solution
Easy3x3	~ 0, 0007	4	45	Yes
Hard5x5	0, 1 - 200	20- 100000	5000-26800000	Yes / No
Given6x6	0, 1 - 200	50 - 100000	50000 - 46000000	Yes / No
Expert9x9	-	-	-	No

Στον παραπάνω πίνακα φαίνονται τα αποτελέσματα που προέκυψαν μετά από πειραματικές δοκιμές του αλγορίθμου min_conflicts στα Puzzles που χρησιμοποιήθηκαν και στο ερώτημα (3). Είναι προφανές ότι ο αλγόριθμος επιστρέφει τη λύση του προβλήματος σίγουρα μόνο για μικρού μεγέθους πίνακες. Για μεγαλύτερα μεγέθη και δυσκολίες ο αλγόριθμος είτε θα εντοπίσει τη λύση σε αποδεκτό χρόνο με εναν σημαντικο αριθμό αναθέσεων και κλήσεων της συνάρτησης για τους περιορισμούς, είτε θα καθυστερήσει πολλή ώρα και δεν θα επιστρέψει κάποια λύση.

Ο ευρετικός μηχανισμός min_conflicts ορίζει εσωτερικά του μία ανάθεση-λύση και έπειτα επαναληπτικά για έναν συγκεκριμένο αριθμό βημάτων επιλέγει μία τυχαία μεταβλητή που συμμετέχει σε συγκρούσεις περιορισμών και της αναθέτει την τιμή που ελαχιστοποιεί τον αριθμό συγκρούσεων ενώ μετά προσθέτει τη μεταβλητή στην λύση.

Από τη στιγμή που ο min_conflicts ξεκινάει με μία τυχαία ανάθεση, δεν είναι εύκολο να προβλέψει κάποιος τη συμπεριφορά του. Αν καταφέρει να εντοπίσει τη λύση του προβλήματος χωρίς να υπερβεί τον αριθμό των βημάτων που του επιτρέπονται, τότε την επιστρέφει και λειτουργεί με έναν τρόπο άξιο σύγκρισης με τους αλγορίθμους του ερωτήματος (3). Αν όμως υπερβεί τον αριθμό βημάτων και δεν έχει καταλήξει ακόμα σε κάποια λύση, ανεξάρτητα από την πρόοδό του, ο αλγόριθμος τερματίζει μετά από κάποιον χρόνο, χωρίς να επιστρέφει απάντηση, έχοντας κάνει περιττές αναθέσεις και κλήσεις της συνάρτησης περιορισμών. Από τη στιγμή που ο αλγόριθμος δεν δίνει πάντα λύση στο πρόβλημα, δεν είναι πλήρης, σε αντίθεση με τους αλγορίθμους του ερωτήματος (3).

Επιπλέον, ο min_conflicts δεν αξιοποιεί τις μερικές λύσεις που ενδεχομένως έχει βρει, όπως οι αλγόριθμοι του ερωτήματος (3), κάτι που ίσως τον απομακρύνει από την τελική λύση του προβλήματος.

Τέλος, οι λύσεις του προβλήματος είναι αραιά κατανεμημένες στον χώρο των καταστάσεων, οπότε ο αλγόριθμος, με τον συγκεκριμένο μηχανισμό, δυσκολεύεται να τις εντοπίσει, ειδικά στους πίνακες μεγαλύτερου μεγέθους, όπως στον 9x9 που αναφέρεται στον πειραματικό πίνακα για τον οποίον ο χρόνος εκτέλεσης του αλγορίθμου ήταν πολύ μεγάλος.

Πέρα άπο την αστάθεια του min_conflicts και την μη εγγύηση λύσης, στις περιπτώσεις που εντοπίζει σωστά τη λύση, εάν αυτός συγκριθεί με έναν από τους αλγορίθμους του ερωτήματος (3), ο χρόνος που απαιτείται για την ολοκλήρωσή του είναι αρκετά μεγαλύτερος.

Με βάση όλα τα παραπάνω, ο αλγόριθμος min_conflicts είναι λιγότερο αποδοτικός από τους αλγορίθμους του παραπάνω ερωτήματος στην επίλυση του συγκεκριμένου CSP και επομένως θα ήταν καλύτερο να μην προτιμάται έναντι αυτών.

Πρόβλημα 2:

Για να εξετασθεί εάν είναι δυνατή η επίπλωση του δωματίου που μας δίνεται με τρόπο τέτοιο ώστε να μην παραβιάζονται οι περιορισμοί αισθητικής που αναφέρονται, ορίστηκε ένα πρόβλημα ικανοποίησης περιορισμών(CSP). Το δωμάτιο θεωρήθηκε σαν ένα grid που χωρίζεται σε τετραγωνικά εκατοστά. Η αρίθμηση τους ξεκινάει από την κάτω αριστερή γωνία του δωματίου. Αυτό σημαίνει ότι το τεραγωνικό εκατοστό που καλύπτει αυτή την γωνία έχει συντεταγμένες (1, 1).

Αναλυτικά, ο ορισμός του CSP:

- **Μεταβλητές:** Σαν μεταβλητές ορίστηκαν δυάδες(tuples) της μορφής (έπιπλο, (xi, yi)), όπου (xi, yi) το ζεύγος συντεταγμένων που αντιστοιχεί στο τετραγωνικό εκατοστό της κάτω αριστερά γωνίας του εκάστοτε επίπλου.
- **Πεδία:** Το πεδίο κάθε μεταβλητής είναι, για κάθε έπιπλο, όλα τα ζεύγη συντεταγμένων που υπάρχουν μέσα στον χώρο που ορίζεται από τις διαστάσεις του δωματίου, δηλαδή η xi μπορεί να πάρει τιμές από 1 έως 300 και η yi μπορεί να πάρει τιμές από 1 έως 400.

- **Περιορισμοί:** Για την συγκεκριμένη μοντελοποίηση θεωρήθηκε απαραίτητο να ικανοποιούνται οι παρακάτω περιορισμοί:

  1. Τα έπιπλα δεν πρέπει να εφάπτονται ή να είναι το ένα πάνω στο άλλο. Για τον λόγο αυτό ορίζονται οι δύο παρακάτω περιορισμοί:

     Έστω δύο έπιπλα i,j με Mi και Πi να είναι το μήκος/βάθος και το πλάτος του επίπλου i και Mj και Pj να είναι το μήκος/βάθος και το πλάτος του επίπλου j. Θα πρέπει:

     α)

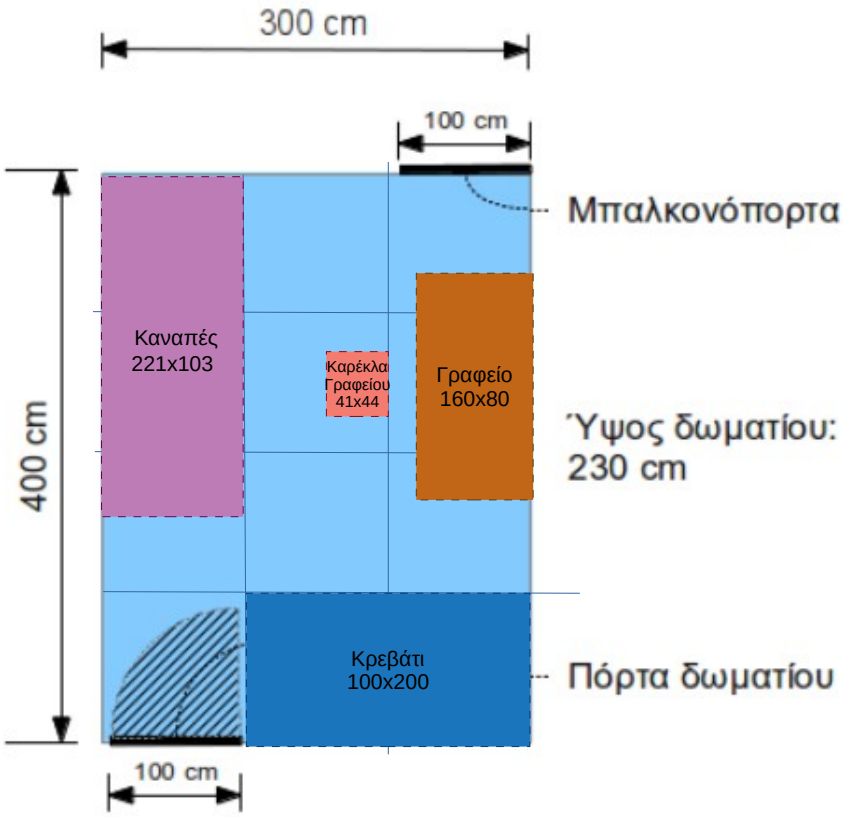
     $$xi + Mi >= xj >= xi \quad \text{ή} \quad xi + Mi >= xj + Mj >= xi \quad \Rightarrow \quad yi > yj + Πj \quad \text{ή} \quad yi + Πi < yj$$

     β)

     $$yi + Πi >= yj >= yi \quad \text{ή} \quad yi + Πi >= yj + Πj >= yi \quad \Rightarrow \quad xi > xj + Mj \quad \text{ή} \quad xi + Mi < xj$$

  2. Το γραφείο πρέπει να είναι κοντά σε πηγή φωτός. Σαν πηγή φωτός θεωρήθηκε μόνο η μπαλκονόπορτα. Για να θεωρείται κοντά στην μπαλκονόπορτα, η κάτω αριστερά γωνία του γραφείου να έχει τετμημένη μεγαλύτερη από 200. Δηλαδή θα πρέπει: xi > 200.
  3. Τα έπιπλα θα πρέπει να βρίσκονται εντός των ορίων που ορίζουν οι διαστάσεις του δωματίου. Δηλαδή, για κάθε έπιπλο θα πρέπει:
     α) xi, yi >= 1
     β) xi + Mi -1 <= 300
     γ) yi + Πi -1 <= 400
  4. Θα πρέπει να μην εμποδίζεται η λειτουργία της πόρτας. Δηλαδή, για κάθε έπιπλο θα πρέπει:
     α) xi > 100 και
     β) yi > 100
  5. (Προαιρετικό, δεν υπήρξε λόγος να συμπεριληφθεί στην συγκεκριμένη μοντελοποίηση του προβλήματος) Θα πρέπει το ύψος ενός επίπλου να μην ξεπερνάει το ύψος του δωματίου. Αν i ένα έπιπλο και Yi το ύψος του, τότε θα πρέπει: Yi =< 230.

Το συμπέρασμα στο οποίο καταλήγει κάποιος μετά τον ορισμό του παραπάνω προβλήματος είναι ότι είναι δυνατή η εύρεση λύσης. Μία από τις λύσεις του είναι η παρακάτω:



Πρόβλημα 3:

α)

Το πρόβλημα που περιγράφεται μπορεί να μοντελοποιηθεί σαν πρόβλημα ικανοποίησης περιορισμών ως εξής:

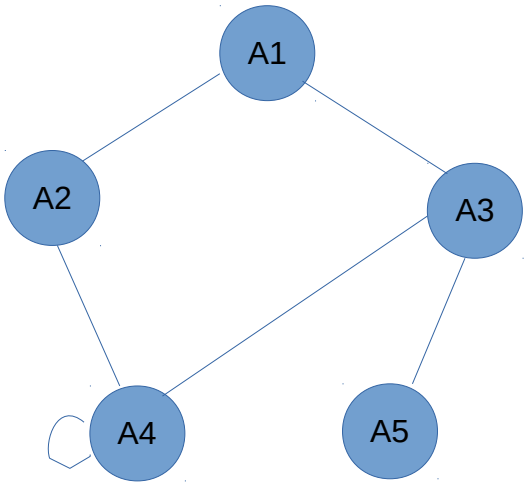
Μεταβλητές: A1, A2, A3, A4, A5

Πεδία: Για κάθε μεταβλητή εκτός απο την A4 το πεδίο είναι {9.00, 10.00, 11.00}. Το πεδίο της A4 θα είναι {9.00, 11.00}.

Περιορισμοί:

- A1 > A3
- A5 < A3
- A3 < A4
- A2 != A1
- A2 != A4

β)



γ)

Πεδία τιμών:
A1: {9.00, 10.00, 11.00}
A2: {9.00, 10.00, 11.00}
A3: {9.00, 10.00, 11.00}
A4: {9.00, 11.00}
A5: {9.00, 10.00, 11.00}

Queue: [(A1, A2), (A2, A1), (A1, A3), (A3, A1), (A2, A4), (A4, A2), (A3, A4), (A4, A3), (A3, A5), (A5, A3)]

Ο αλγόριθμος AC-3 εκτελείται ως εξής:

1) Pop (A1, A2)
2) Remove-Inconsistent-Values(A1, A2) = False
3) Queue: [(A2, A1), (A1, A3), (A3, A1), (A2, A4), (A4, A2), (A3, A4), (A4, A3), (A3, A5), (A5, A3)]

1) Pop (A2, A1)
2) Remove-Inconsistent-Values(A2, A1) = False.
3) Queue: [(A1, A3), (A3, A1), (A2, A4), (A4, A2), (A3, A4), (A4, A3), (A3, A5), (A5, A3)]

1) Pop (A1, A3)
2) Remove-Inconsistent-Values(A1, A3) = True, διότι:
    Για A1 = 9.00 δεν υπάρχει τιμή στο πεδίο της A3 τ.ώ.: A1 > A3.
    A1 = {10.00, 11,00}
3) Queue: [(A3, A1), (A2, A4), (A4, A2), (A3, A4), (A4, A3), (A3, A5), (A5, A3), (A2, A1), (A3, A1)]

1) Pop(A3, A1).
2) Remove-Inconsistent-Values(A3, A1)= True, διότι:
    Για A3 = 11.00, δεν υπάρχει τιμή στο πεδίο της A1 τ.ώ.: A3 < A1.
    A3 = {9.00, 10.00}
3) Queue: [(A2, A4), (A4, A2), (A3, A4), (A4, A3), (A3, A5), (A5, A3), (A2, A1), (A3, A1), (A1, A3), (A4, A3), (A5, A3)]

1) Pop(A2, A4)
2) Remove-Inconsistent-Values(A2, A4) = False
3) Queue: [(A4, A2), (A3, A4), (A4, A3), (A3, A5), (A5, A3), (A2, A1), (A3, A1), (A1, A3), (A4, A3), (A5, A3)]

1) Pop(A4, A2).
2) Remove-Inconsistent-Values(A4, A2) = False.
3) Queue: [(A3, A4), (A4, A3), (A3, A5), (A5, A3), (A2, A1), (A3, A1), (A1, A3), (A4, A3), (A5, A3)]

1) Pop(A3, A4).
2) Remove-Inconsistent-Values(A3, A4) = False.
3) Queue: [(A4, A3), (A3, A5), (A5, A3), (A2, A1), (A3, A1), (A1, A3), (A4, A3), (A5, A3)]

1) Pop(A4, A3).
2) Remove-Inconsistent-Values(A4, A3) = True, διότι:
    Για A4 = 9.00, δεν υπάρχει τιμή στο πεδίο της A3 τ.ώ.: A4 > A3.
    A4 = {11.00}
3) Queue: [(A3, A5), (A5, A3), (A2, A1), (A3, A1), (A1, A3), (A4, A3), (A5, A3), (A2, A4), (A3, A4)]

1) Pop(A3, A5).
2) Remove-Inconsistent-Values(A3, A5) = True, διότι:
    Για A3 = 9.00, δεν υπάρχει τιμή στο πεδίο της A5 τ.ώ.: A3 > A5.
    A3 = {10.00}
3) Queue: [(A5, A3), (A2, A1), (A3, A1), (A1, A3), (A4, A3), (A5, A3), (A2, A4), (A3, A4), (A1, A3), (A4, A3), (A5, A3)]

1) Pop(A5, A3).
2) Remove-Inconsistent-Values(A5, A3) = True, διότι:
    Για A5 = 10.00, δεν υπάρχει τιμή στο πεδίο της A3 τ.ώ: A5 < A3.
    Για A5 = 11.00, δεν υπάρχει τιμή στο πεδίο της A3 τ.ώ.: A5 < A3.
    A5 = {9.00}
3) Queue: [(A2, A1), (A3, A1), (A1, A3), (A4, A3), (A5, A3), (A2, A4), (A3, A4), (A1, A3), (A4, A3), (A5, A3), (A3, A5)]

1) Pop(A2, A1).
2) Remove-Inconsistent-Values(A2, A1) = False.
3) Queue: [(A3, A1), (A1, A3), (A4, A3), (A5, A3), (A2, A4), (A3, A4), (A1, A3), (A4, A3), (A5, A3), (A3, A5)]

1)Pop(A3, A1)
2)Remove-Inconsistent-Values(A1, A3) = False.
3) Queue: [(A1, A3), (A4, A3), (A5, A3), (A2, A4), (A3, A4), (A1, A3), (A4, A3), (A5, A3), (A3, A5)]

1)Pop(A1, A3)
2) Remove-Inconsistent-Values(A1, A3) = True, διότι:
Για A1 = 10.00, δεν υπάρχει τιμή στο πεδίου της A3 τ.ώ.: A1 > A3
A1 = {11.00}
3) Queue: [(A4, A3), (A5, A3), (A2, A4), (A3, A4), (A1, A3), (A4, A3), (A5, A3), (A3, A5), (A2, A1), (A3, A1)]

1) Pop(A4, A3)
2) Remove-Inconsistent-Values(A4, A3) = False
3) Queue: [(A5, A3), (A2, A4), (A3, A4), (A1, A3), (A4, A3), (A5, A3), (A3, A5), (A2, A1), (A3, A1)]

1) Pop(A5, A3)
2) Remove-Incosistent-Values(A5, A3) = False
3) Queue: [(A2, A4), (A3, A4), (A1, A3), (A4, A3), (A5, A3), (A3, A5), (A2, A1), (A3, A1)]

1) Pop(A2, A4)
2) Remove-Incosistent-Values(A2, A4) = True, διότι:
Για A2 = 11.00, δεν υπάρχει τιμή στο πεδίο της A4 τ.ώ.: A2 != A4.
A2 = {9.00, 10.00}
3) Queue: [(A3, A4), (A1, A3), (A4, A3), (A5, A3), (A3, A5), (A2, A1), (A3, A1), (A1, A2), (A4, A2)]

1) Pop(A3, A4)
2) Remove-Incosistent-Values(A3, A4) = False
3) Queue: [(A1, A3), (A4, A3), (A5, A3), (A3, A5), (A2, A1), (A3, A1), (A1, A2), (A4, A2)]

1) Pop(A1, A3)
2) Remove-Inconsistent-Values(A1, A3) = False
3) Queue: [(A4, A3), (A5, A3), (A3, A5), (A2, A1), (A3, A1), (A1, A2), (A4, A2)]

1) Pop(A4, A3)
2) Remove-Inconsistent-Values(A4, A3) = False
3) Queue: [(A5, A3), (A3, A5), (A2, A1), (A3, A1), (A1, A2), (A4, A2)]

1) Pop(A5, A3)
2) Remove-Inconsistent-Values(A5, A3) = False
3) Queue: [(A3, A5), (A2, A1), (A3, A1), (A1, A2), (A4, A2)]

1) Pop(A3, A5)
2) Remove-Inconsistent-Values(A3, A5) = False

3) Queue: [(A2, A1), (A3, A1), (A1, A2), (A4, A2)]

1) Pop(A2, A1)
2) Remove-Inconsistent-Values(A2, A1) = False
3) Queue: [(A3, A1), (A1, A2), (A4, A2)]

1) Pop(A3, A1)
2) Remove-Inconsistent-Values(A3, A1) = False
3) Queue: [(A1, A2), (A4, A2)]

1) Pop(A1, A2)
2) Remove-Inconsistent-Values(A1, A2) = False
3) Queue: [(A4, A2)]

1) Pop(A4, A2)
2) Remove-Inconsistent-Values(A4, A2) = False
3) Queue: [ ]

List is Empty!

Τα πεδία τιμών που προκύπτουν μετά την εφαρμογή του αλγορίθμου AC-3:

A1 = {11.00}
A2 = {9.00, 10.00}
A3 = {10.00}
A4 = {11.00}
A5 = {9.00}