“Sudoku Problem, the Binary Integer Programming Approach”

Αναφορά και επεξήγηση, Ορέστης Νικόλας

Εξήγηση του μοντέλου:

- $\mathbf{1} = \sum_{k=1}^{9} \boldsymbol{x}(\boldsymbol{i},\boldsymbol{j},\boldsymbol{k})$ **i,j,k ∈ {1,2,…,9}** , $\boldsymbol{x}$ **∈ {0,1**} (1) z-axis constraint

  Αν θεωρήσουμε ότι η λύση χ αναπαρίσταται από έναν 9x9x9 binary πίνακα τότε για κάθε στοιχείο στον 2D πίνακα θα πρέπει να υπάρχει αποκλειστικά ένα non-zero element μεταξύ των x(i,j,1),…,x(I,j,9). Π.χ. Αν το x(1,1,5) = 1 τότε τα x(1,1,k!=5) = 0 που σημαίνει ότι στη θέση (1,1) είναι το 5.

- $\mathbf{1} = \sum_{j=1}^{9} \boldsymbol{x}(\boldsymbol{i},\boldsymbol{j},\boldsymbol{k})$ **i,j,k ∈ {1,2,…,9} ,**$\boldsymbol{x}$ **∈ {0,1}** (2) x-axis constraint

  Παρομοίως σε κάθε γραμμή του 2D πίνακα θα πρέπει να υπάρχει ακριβώς ένα non-zero element για το κάθε ένα από τα σύμβολα 1->9.

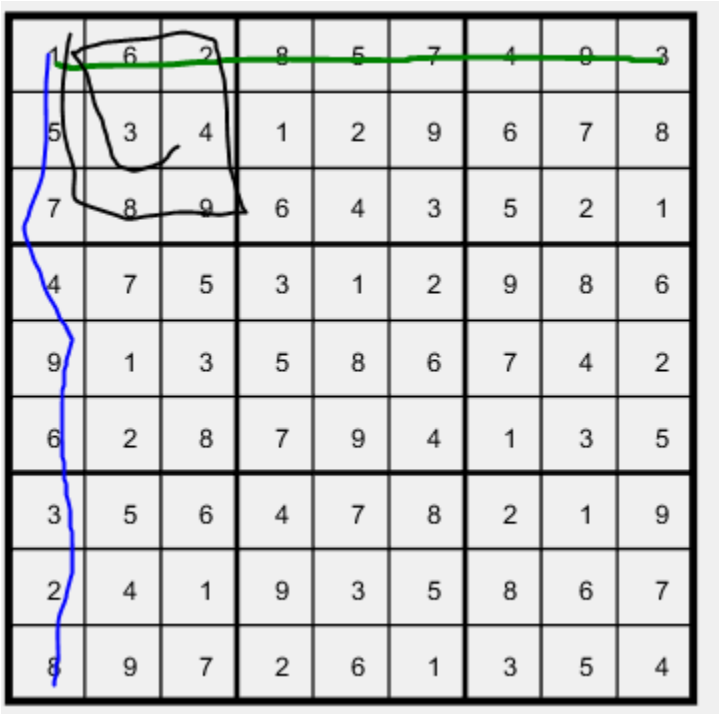
- $\mathbf{1} = \sum_{i=1}^{9} \boldsymbol{x}(\boldsymbol{i},\boldsymbol{j},\boldsymbol{k})$ **i,j,k ∈ {1,2,…,9}** , $\boldsymbol{x}$ **∈ {0,1}** (3) y-axis constraint

  Το ίδιο με το από πάνω απλά για κάθε στήλη.

- $\mathbf{1} = \sum_{i=1}^{3} \sum_{i=1}^{3} \boldsymbol{x}(\boldsymbol{i}+\boldsymbol{U},\boldsymbol{j}+\boldsymbol{V},\boldsymbol{k})$ **i,j,k ∈ {1,2,…,9} ,U,V ∈ {0,3,6} ,**$\boldsymbol{x}$ **∈ {0,1}** (4) x-y-axis constraint. Κάθε 3x3 grid έχει παρόμοιο περιορισμό με τους παραπάνω και εκφράζεται με αυτό τον μαθηματικό τύπο.

Θα δείξω τη γραφική μετάφραση τών περιορισμών(1-RED,2-GREEN,3-BLUE,4-Black):

- Στο κανονικό 2D πρόβλημα:



1	6	2	8	5	7	4	9	3
5	3	4	1	2	9	6	7	8
7	8	9	6	4	3	5	2	1
4	7	5	3	1	2	9	8	6
9	1	3	5	8	6	7	4	2
6	2	8	7	9	4	1	3	5
3	5	6	4	7	8	2	1	9
2	4	1	9	3	5	8	6	7
8	9	7	2	6	1	3	5	4

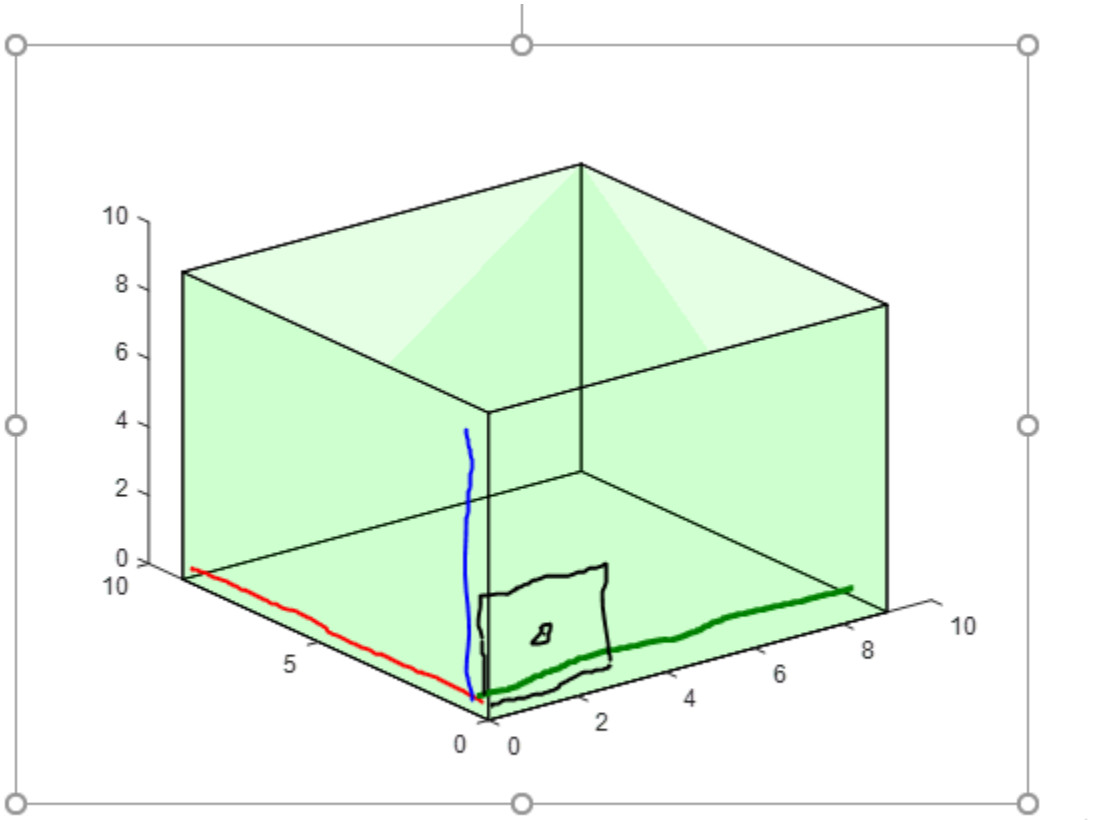
Πράσινο: Περιορισμός σειράς

Μπλέ: Περιορισμός στήλης

Μάυρο: Περιορισμός 3x3 Grid,

Ο κόκκινος περιορισμός δεν υπάρχει στο 2D πρόβλημα αλλά είναι απαραίτητος για να ανταποκρίνεται στην πραγματικότητα το μοντέλο μας.

- Στο BIP μοντέλο:



Πράσινο: Περιορισμός x

Μπλε: Περιορισμός y

Κόκκινος: Περιορισμός z.

Μαύρο: Περιορισμός 3x3 grid στους άξονες x-y που ανοίκει το αρμόδιο στοιχείο.

Π.χ. Όταν βάλουμε τιμή 1 στο x(1,1,1) αυτομάτως μηδενίζονται τα υπόλοιπα στοιχεία στα x,y και z axis που ανήκει το στοιχείο αυτό αλλά και στο x-y grid με διαστάσεις 3x3 που ανήκει, όπως φαίνεται στο σχήμα.

Αυτό το auto completion μεταβλητών είναι η κύρια αιτία γιατί μπήκαμε στη διαδικασία κατάστρωσης του Binary Integer Programming μοντέλου και όχι απλά να βάζαμε σε ένα solver το κανονικό αρχικό πρόβλημα ακέραιου προγραμματισμού.

Οτιδήποτε πληροφορία έχω βρει για sudoku solving και integer programming είναι σχεδόν όλες με BIP προσέγγιση. Φαντάζομαι ότι ο λόγος είναι ότι με απλό IP, η Branch & Bound θα εξερευνάει περισσότερα nodes(περισσότερες υλοποιήσεις του Simplex) αν και δεν έχω βρει κώδικα για να το υποστηρίξω. Στο απλό IP, πρώτη υλοποίηση του simplex θα βρει μια πραγματική λύση που θα είναι

κοντά στην ακέραια αλλά κάθε φορά που θα διαλέγει μία μεταβλητή για να κάνει branch και θα τη βάζει στον άνω/κάτω integer εφόσον οι υπόλοιπες μεταβλητές μπορούν να πάρουν αντί για 2 τιμές(0-1), 9(1-9) τότε θα υπάρχει μεγαλύτερη πιθανότητα να "χαλάσει" κάποια άλλη. Βέβαια αυτή η εξήγηση είναι εντελώς διαισθητική.

Επίσης, αποκλειστικά από προγραμματιστική άποψη, άμα μπορούμε να μοντελοποιήσουμε ένα πρόβλημα ως ένα Boolean satisfiability problem(ή αλλιώς propositional satisfiability problem ή SAT problem) τότε από τη performance πλευρά συμφέρει να καταστρώσουμε τους αλγορίθμους μας βάση αυτού του μοντέλου έναντι άλλων. Και προφανώς το BIP είναι πιο κοντά σε τέτοια μοντελοποίηση, αφού όλες οι συνθήκες μας μπορούν να εκφραστούν με Boolean algebra.

Η υλοποίηση σε matlab της προσέγγισης μου πήρε 0.04 sec για να λύσει ένα medium difficulty sudoku. Οι κώδικες υπάρχουν στο φάκελο codes-BinaryIntegerProgrammingApproach, με το που τρέξετε το SudokuSolver_.m θα δείτε τα αποτελέσματα πιο αναλυτικά. Έχω και αρκετά comments για ευκολότερη κατανόηση του κώδικα και για το πού κάνει τι. Οι έξτρα συναρτήσεις που έκανα είναι η drawSudoku.m και plotcube.m που τις χρησιμοποιώ κυρίως για απεικόνιση και το presentation και η sparse3Dfrom2Dfull.m για να μετατρέπω ένα 2D πίνακα σε μορφή αραιάς μήτρας γιατί σε τέτοια μορφή λειτουργεί ο κώδικας που έχω αναπτύξει.

Άλλα μοντέλα και το performance τους:

1. Backtracking approach:

Για αυτό το approach έχω ένα κώδικα που βρήκα στο:
https://www.mathworks.com/matlabcentral/fileexchange/28168-sudoku-generator
Αυτό που κάνει μπορεί να θεωρηθεί μια μορφή backtracking και μπορείτε να τρέξετε το codes-BruteForceApproach-SudokuSolver.m και κάνει να λύσει ένα medium difficulty sudoku 6.5 sec. Τη σύγκρισή για τα θετικά και τα αρνητικά για αυτό το approach και όλα τα παρακάτω την έχω στο presentation.

2. Stochastic approach:

Εδώ για αυτή τη προσέγγιση δεν έχω βρει κάποιο κώδικα για να δοκιμάσω πάνω στο συγκεκριμένο πρόβλημα. Κυρίως ότι πληροφορία βρήκα είναι για την γενική εφαρμογή αυτής της προσέγγισης σε προβλήματα(π.χ. https://en.wikipedia.org/wiki/Sudoku_solving_algorithms#cite_note-12 , https://en.wikipedia.org/wiki/Stochastic_optimization)
Τώρα το ότι η συγκεκριμένη προσέγγιση είναι λίγο πιο αργή από BIP δεν 100% σίγουρο απλά στη παρουσίαση το είπα γιατί γενικότερα οι stochastic-based αλγόριθμοι είναι γρήγοροι αλλά γενικότερα όχι πιο γρήγοροι από deductive methods.

3. Constraint programming approach:

Αυτή η προσέγγιση ήταν η πιο ενδιαφέρουσα. Την βρήκα από το Paper του Helmut Simonis (Το έχω προσθέσει στο παραδοτέο) Τα αποτελέσματα του paper αυτού δείχνουν ότι για να λύσει με την τεχνική αυτή ένα medium difficulty sudoku θέλει 0.0235 sec κατά μέσο όρο σε πολύ πιο αδύνατο σύστημα από το δικό μου. Δυστυχώς δε βρήκα κώδικα για να το δω στο δικό μου υπολογιστή και να συγκρίνω τα αποτελέσματα αλλά πιστεύω για ένα medium difficulty sudoku θα έκανε γύρω στο 0.01 sec. Παρακάτω φαίνονται τα αποτελέσματα της προσέγγισης αυτής. Πρόκειται για ένα αρκετά δύσκολο paper γιατί αναμιγνύει πολλές τεχνικές, έννοιες και μοντελοποιήσεις όπως το Channeling, Propagation Schemes, Shaving, Row/Column/Block interaction κτλ.

			Preset			Locally	Reduced		
Source	Grade	Inst	Min	Avg	Max	Minimal	Min	Avg	Max
[38]	medium	60	26	35.48	40	0	22	23.50	25
[38]	difficult	50	24	27.84	30	0	22	24.22	26
[38]	super-difficult	40	24	27.68	31	1	22	24.40	29
[42]	none	450	17	17.00	17	450	17	17.00	17
[44]	none	100	17	21.51	31	82	17	21.14	26

Table 7: Reduction Summary

4. Convolutional Neural Networks:

Γενικότερα δε θεωρώ ότι τα CNN μπορούν να λύσουν ικανοποιητικά αυτό το πρόβλημα γιατί πρόκειται για ένα σχετικά απλό και κατανοητό πρόβλημα όπου μπορεί έυκολα να μοντελοποιηθεί και να εξηγηθεί μαθηματικά. Τα CNN χρησιμοποιούνται κυρίως για προβλήματα τα όποια είναι απίστευτα δύσκολα στη μοντελοποίηση η και για προβλήματα τα οποία είναι αδύνατο να μοντελοποιηθούν με αντικειμενικά κριτήρια(π.χ. Αναγνώριση Προσώπου από εικόνα). Παρόλα αυτά λόγω περιέργειας κυρίως βρήκα μια υλοποίηση στο git: https://github.com/Kyubyong/sudoku Τον κώδικα δε τον έχω αναθέσει στο παραδοτέο γιατί υπάρχουν τα results για το CNN training κτλ. Δεν είναι καλή προσέγγιση για το συγκεκριμένο πρόβλημα παρόλα αυτά είναι ενδιαφέρουσα γιατί φαίνεται ότι σε απλά προβλήματα με απλή λογική δε δουλεύουν.