**Μοντελοποίηση του Sudoku με Ακέραιο Γραμμικό Προγραμματισμό (ILP) και με προηγμένες μεθόδους**

Project στην Γραμμική και Συνδυαστική Βελτιστοποίηση

<https://github.com/tsaperlein/Sudoku.git>

**Ονοματεπώνυμο:** Αλέξανδρος Τσαπάρας

**ΑΜ:** 1072824

1. **Εισαγωγή**

Το Sudoku είναι ένα δημοφιλές παζλ τοποθέτησης αριθμών, που βασίζεται στη λογική. Ο στόχος είναι να γεμίσουμε ένα πλέγμα 9x9 με ψηφία έτσι ώστε κάθε στήλη, κάθε σειρά και καθένα από τα εννέα υποπλέγματα 3x3 που συνθέτουν το πλέγμα (γνωστά και ως "κουτιά") να περιέχουν όλα τα ψηφία από το 1 έως το 9. Μολονότι τα Sudoku παζλ μπορούν να επιλυθούν μέσω διαφόρων αλγορίθμων και τεχνικών, εμείς θα διερευνήσουμε τη χρήση του Ακέραιου Γραμμικού Προγραμματισμού (ILP) για την επίλυση παζλ Sudoku χρησιμοποιώντας τη βιβλιοθήκη PuLP.

1. **Ακέραιος Γραμμικός Προγραμματισμός (ILP)**

**A graph on a white board

Description automatically generated**Ο ILP είναι μια μαθηματική τεχνική βελτιστοποίησης που χρησιμοποιείται για την εύρεση της καλύτερης δυνατής λύσης σε ένα πρόβλημα, δεδομένου ενός συνόλου περιορισμών και μιας αντικειμενικής συνάρτησης. Στο πλαίσιο του Sudoku, ο ILP μπορεί να χρησιμοποιηθεί για τη διαμόρφωση του παζλ ως πρόβλημα βελτιστοποίησης με δυαδικές μεταβλητές απόφασης.

1. **Διαμόρφωση του Sudoku παζλ ως πρόβλημα ILP**

* **Διατύπωση προβλήματος**

Στο Sudoku έχουμε έναν πίνακα 9x9, ο οποίος πρέπει να συμπληρωθεί με αριθμούς από το 1 έως το 9 με τέτοιο τρόπο ώστε:

1. Κάθε γραμμή να περιέχει κάθε αριθμό από το 1 έως το 9 ακριβώς μία φορά.
2. Κάθε στήλη να περιέχει κάθε αριθμό από το 1 έως το 9 ακριβώς μία φορά.
3. Κάθε υποπίνακας 3x3 να περιέχει κάθε αριθμό από το 1 έως το 9 ακριβώς μία φορά.

* **Μεταβλητές απόφασης**

Για κάθε κελί στο πλέγμα του Sudoku, δημιουργούμε δυαδικές μεταβλητές απόφασης. Αυτές οι μεταβλητές αντιπροσωπεύουν εάν ένα συγκεκριμένο ψηφίο τοποθετείται σε ένα κελί. Για ένα τυπικό παζλ Sudoku 9x9, υπάρχουν 81 κελιά και κάθε κελί μπορεί να περιέχει ένα από τα εννέα ψηφία. Έτσι, θα έχουμε 729 δυαδικές μεταβλητές απόφασης (9 ψηφία × 81 κελιά).

Θα ορίσουμε δυαδικές μεταβλητές ​ όπου:

* είναι ο δείκτης της γραμμής (1 έως 9)
* είναι ο δείκτης της στήλης (1 έως 9)
* είναι ο αριθμός που τοποθετείται στο κελί (1 έως 9)

Η ​ θα είναι 1 αν ο αριθμός k τοποθετείται στο κελί , και 0 διαφορετικά.

* **Αντικειμενική συνάρτηση:**

Στην περίπτωση του Sudoku, η αντικειμενική συνάρτηση είναι ασήμαντη, καθώς δεν επιδιώκουμε τη βελτιστοποίηση κάποιας ποσότητας αλλά την εύρεση μιας λύσης που ικανοποιεί όλους τους περιορισμούς. Για τη διατύπωση του προβλήματος ως ILP, μπορούμε να ορίσουμε μια αυθαίρετη αντικειμενική συνάρτηση, όπως η ελαχιστοποίηση του μηδενός:

Αυτό απλά σημαίνει ότι ψάχνουμε για οποιαδήποτε εφικτή λύση που ικανοποιεί όλους τους περιορισμούς.

* **Περιορισμοί:**

1. **Περιορισμοί κελιών:**

Βεβαιωθείτε ότι κάθε κελί περιέχει ακριβώς ένα ψηφίο. Αυτό μπορεί να εκφραστεί ως περιορισμός που αθροίζει τις δυαδικές μεταβλητές για κάθε κελί και το άθροισμα πρέπει να ισούται με 1.

1. **Περιορισμοί γραμμής:**

Βεβαιωθείτε ότι κάθε ψηφίο εμφανίζεται ακριβώς μία φορά σε κάθε σειρά. Αυτό μπορεί να εκφραστεί ως περιορισμοί που διασφαλίζουν ότι το άθροισμα των δυαδικών μεταβλητών για κάθε ψηφίο σε κάθε γραμμή ισούται με 1.

1. **Περιορισμοί στήλης:**

Βεβαιωθείτε ότι κάθε ψηφίο εμφανίζεται ακριβώς μία φορά σε κάθε στήλη, παρόμοια με τους περιορισμούς σειρών.

1. **Περιορισμοί πλαισίου:**

Βεβαιωθείτε ότι κάθε ψηφίο εμφανίζεται ακριβώς μία φορά σε κάθε πλαίσιο 3x3 (υποπλέγμα), παρόμοια με τους περιορισμούς σειρών και στηλών.

Για κάθε υποπίνακα που αρχίζει στη θέση με :

* **Δεδομένα Αρχικής Κατάστασης**

Αν έχουμε έναν αριθμό ήδη τοποθετημένο στο κελί :

* **Δυαδικές Μεταβλητές**

Καθορίστε ότι οι μεταβλητές απόφασης είναι δυαδικές (0 ή 1) για να αντιπροσωπεύουν εάν ένα ψηφίο τοποθετείται ή όχι.

Μόλις δημιουργηθεί η διατύπωση ILP, ένας λύτης όπως το PuLP μπορεί να χρησιμοποιηθεί για να βρεθεί μια λύση στο Sudoku παζλ ικανοποιώντας όλους τους περιορισμούς.

A diagram of a decision

Description automatically generated

1. **Υλοποίηση με PULP**

Ο παρεχόμενος κώδικας (sudoku.py) και τα επιμέρους αρχεία (sudoku-solver, killer-sudoku, …) υλοποιούν έναν λύτες Sudoku και παραλλαγών του που βασίζονται σε ILP χρησιμοποιώντας τη βιβλιοθήκη PuLP. Ακολουθεί μια ανάλυση/επισκόπηση του κώδικα:

**Normal Sudoku**

* **Sudoku Class (sudoku.py)**

Η κλάση Sudoku είναι η βασική κλάση για Sudoku παζλ και αξιοποιεί τον ILP ως εξής:

* Αρχικοποιεί το μοντέλο Sudoku χρησιμοποιώντας PuLP.
* Ορίζει δυαδικές μεταβλητές απόφασης για κάθε κελί χρησιμοποιώντας την κλάση LpVariable από το PuLP.
* Προσθέτει περιορισμούς για να διασφαλίσει ότι τηρούνται οι κανόνες του Sudoku, συμπεριλαμβανομένων των περιορισμών κελιών, περιορισμών σειρών, περιορισμών στηλών και περιορισμών πλαισίου.
* Παρέχει μεθόδους για τον ορισμό και λήψη τιμών από τα κελιά του παζλ.
* Προσφέρει μια μέθοδο επίλυσης για την εύρεση λύσης στο παζλ.

A black and white rectangular grid with white numbers

Description automatically generated with medium confidence

Μορφή αρχείου εισόδου:

Αρχικά αναφέρονται οι διαστάσεις του κάθε κουτιού μέσα στο πλέγμα, στη δικιά μας περίπτωση το κάθε κουτί είναι 3x3 , οπότε το πλέγμα είναι 9x9. Έπειτα ακολουθούν m\*n γραμμές και m\*n στήλες, που περιέχουν είτε έναν αριθμό (1..m\*n) είτε ένα από τα εξής σύμβολα (“\_”, “?”, “0”, “.”, “\*“), το οποίο δηλώνει ότι δεν περιέχει κάποιο έγκυρο ψηφίο. Η ανάγνωση της εισόδου σταματά μόλις ο χρήστης συμπληρώσει και την τελευταία γραμμή.

* **sudoku-sovler (ξεχωριστό αρχείο/script)**

Μέσω αυτού του αρχείο γίνεται η ανάγνωση του αρχείο εισόδου και κατηγοριοποιούνται οι διαφορετικές πληροφορίες. Έπειτα δημιουργείται το αντίστοιχο παζλ με ορίσματα τις πληροφορίες από το αρχείο εισόδου και προσθέτουμε τις τιμές του δοσμένου παζλ στο παζλ που δημιουργήσαμε. Τέλος τρέχουμε την συνάρτηση solve() και εκτυπώνουμε την λύση του παζλ και τον χρόνο επίλυσης.

Παρόμοια αρχεία/scripts (killer-sudoku, greater-than, hyper-sudoku, four-pyramids, sandwich-sudoku, x-sudoku) δημιουργήθηκαν για όλες τις παραλλαγές του Sudoku, τα οποία εκμεταλλεύονται τις αντίστοιχες κλάσεις για να επιλύσουν τα δοσμένα παζλ του χρήστη.

**Killer-Sudoku**

* **Killer\_Sudoku Class (sudoku.py)**

A black grid with white lines

Description automatically generated

Η κλάση Killer\_Sudoku προέρχεται από την κλάση Sudoku και την επεκτείνει για να υποστηρίξει Killer Sudoku παζλ. Προσθέτει επιπλέον περιορισμούς ειδικά για το Killer Sudoku, όπου συγκεκριμένα κλουβιά (cages) έχουν καθορισμένες τιμές.

A black and white screen with numbers

Description automatically generatedΌπως παρατηρούμε και από το παράδειγμα, τα κελιά (1,1) και (1,2) περιλαμβάνονται από το κλουβί με τιμή 3, το οποίο σημαίνει ότι το άθροισμα των τιμών τους θα ισούται με 3.

Μορφή αρχείου εισόδου:

Όπως και στο κανονικό sudoku θα δίνουμε τις διαστάσεις και το πλέγμα (board) με τις τιμές, αλλά προσθέτουμε επίσης και τις τιμές των κλουβιών. Είναι σημαντικό να ακολουθούν την μορφή το οποίο σημαίνει ότι οι τιμές των κελιών με συντεταγμένες πρέπει να έχουν άθροισμα 7.

**Greater-Than Sudoku**

* A grid of black and white squares

  Description automatically generated**Greater\_Than\_Sudoku Class (sudoku.py)**

A screenshot of a phone

Description automatically generatedΗ κλάση Greater\_Than\_Sudoku επεκτείνει την κατηγορία Sudoku για να χειριστεί Greater-Than Sudoku παζλ. Εισάγει περιορισμούς που διασφαλίζουν ότι οι τιμές σε ορισμένα κελιά είναι μεγαλύτερες από τις τιμές σε άλλα κελιά.

Παράδειγμα: το κελί (1,1) πρέπει να είναι μεγαλύτερο από το κελί (1,2).

Μορφή αρχείου εισόδου:

Όπως το κανονικό sudoku, απλώς προσθέτουμε τις ανισότητες των κελιών. Δηλαδή η πρώτη γραμμή σημαίνει ότι το κελί (2,1) είναι μεγαλύτερο από το κελί (1,1) κ.λπ.

**X-Sudoku**

**A screenshot of a game

Description automatically generated**

* **X\_Sudoku Class (sudoku.py)**

Η κλάση X\_Sudoku επεκτείνει την κλάση Sudoku για να χειριστεί X-Sudoku παζλ. Επιβάλλει πρόσθετους περιορισμούς όπου οι διαγώνιοι πρέπει να περιέχουν μοναδικές τιμές. Η μορφή του αρχείου εισόδου είναι όπως το κανονικό sudoku.

A grey square with black numbers

Description automatically generated

**Hyper-Sudoku**

* **Hyper\_Sudoku Class (sudoku.py)**

Η κλάση Hyper\_Sudoku επεκτείνει την κλάση Sudoku για να χειριστεί Hyper-Sudoku παζλ. Προσθέτει περιορισμούς ειδικά για το Hyper-Sudoku, όπου ορισμένα μπλοκ 3x3 πρέπει να περιέχουν μοναδικές τιμές. Η μορφή του αρχείου εισόδου είναι όπως το κανονικό sudoku.

A crossword puzzle with numbers

Description automatically generated

**Four-Pyramids Sudoku**

* **Four\_Pyramids\_Sudoku Class (sudoku.py)**

Η κλάση Four\_Pyramids\_Sudoku επεκτείνει την κλάση Sudoku για να χειριστεί Sudoku Four Pyramids παζλ. Εισάγει περιορισμούς όπου οι τιμές σε συγκεκριμένα κελιά σχηματίζουν το σχήμα τεσσάρων πυραμίδων. Η μορφή του αρχείου εισόδου είναι όπως το κανονικό sudoku.

A screenshot of a sudoku game

Description automatically generated**Sandwich-Sudoku**

* **Sandwich\_Sudoku Class (sudoku.py)**

Η κλάση Sandwich\_Sudoku επεκτείνει την κλάση Sudoku για να χειριστεί Sandwich Sudoku παζλ. Προσθέτει περιορισμούς με βάση τους παρεχόμενους περιορισμούς αθροίσματος για γραμμές και στήλες.

Παράδειγμα: το άθροισμα των τιμών των κελιών ανάμεσα στα κελιά με τιμές 1 και 9, πρέπει να ισούται με τον δοσμένο αριθμό από τους περιορισμούς. Δηλαδή στην πρώτη γραμμή ανάμεσα στο 1 και στο 9 πρέπει να πάει το 7.

Μορφή αρχείου εισόδου:

A screenshot of a computer

Description automatically generated

Όπως το κανονικό sudoku, απλώς προσθέτουμε τις τιμές για τις σειρές και τις στήλες, με την σειρά αντίστοιχα. Δηλαδή για την πρώτη γραμμή αντιστοιχεί ο αριθμός 0, ενώ για την πρώτη στήλη αντιστοιχεί ο αριθμός 2 κ.λπ.

Υλοποίηση των Sandwich-Sudoku και Greater-Than Sudoku solvers

Η υλοποίηση των Sandwich-Sudoku και Greater-Than Sudoku solvers δεν ήταν εφικτή, διότι δεν μπορούν οι αντίστοιχοι περιορισμοί να εκφραστούν με Ακέραιο Γραμμικό Προγραμματισμό (ILP).

Προσπάθησα για τον Greater-Than solver να εξαλείψω τις πιθανές τιμές των κελιών, σύμφωνα με τους δοσμένους περιορισμούς.

Π.χ. αν το (1,1) είναι μεγαλύτερο του κελιού (1,2) και είναι τα πρώτα που εξετάζουμε, δηλαδή έχουν και τα δύο τις πιθανές τιμές 1,2,3,4,5,6,7,8,9, τότε μετά από το περιορισμό θα έχουμε ότι οι πιθανοί τιμές του κελιού (1,1) είναι [2,3,4,5,6,7,8,9] και του (1,2) οι [1,2,3,4,5,6,7,8].

Επιπλέον για την υλοποίηση του Sandwich-Sudoku solver προσπάθησα να ορίσω μια γενική έννοια για τα κελιά και το άθροισμα των μεταξύ τους τιμών. Δηλαδή θεωρώ ότι οι τιμές μεταξύ των κελιών με τιμή 1 και με τιμή 9 θα έχουν άθροισμα όσο ο περιορισμός, χωρίς να γνωρίζω τις τιμές των των δύο κελιών.

1. **Υλοποίηση με πολλαπλές μεθόδους και backtracking**

**Προηγμένες μέθοδοι**

1. **Hidden Singles:**

Τα Hidden Singles είναι αριθμοί που μπορούν να υπάρχουν μόνο σε ένα κελί μέσα σε μια γραμμή, στήλη ή πλαίσιο. Ο κώδικας προσδιορίζει τα Hidden Singles εξαλείφοντας τους υποψήφιους αριθμούς με βάση την παρουσία αριθμών σε σχετικά κελιά. Γεμίζει κελιά που έχουν μόνο έναν υποψήφιο, απλοποιώντας το παζλ.

1. **CSP (Πρόβλημα ικανοποίησης περιορισμών):**

Η CSP αντιπροσωπεύει το Sudoku ως ένα μαθηματικό πρόβλημα με περιορισμούς. Εφαρμόζει τεχνικές διάδοσης περιορισμών για τη μείωση των υποψηφίων αριθμών στα κελιά. Το CSP είναι μια συστηματική προσέγγιση που επιβάλλει περιορισμούς για την εύρεση έγκυρων λύσεων.

1. **Intersect (Σημείο τομής):**

Η Intersection αναζητά κελιά όπου δύο πλαίσια μοιράζονται μια γραμμή ή στήλη. Όταν ένας υποψήφιος αριθμός εμφανίζεται μόνο σε αυτήν τη γραμμή ή τη στήλη και στα δύο πλαίσια, μπορεί να εξαλειφθεί από άλλα κελιά στην τομή. Αυτή η τεχνική αξιοποιεί τις αλληλεπιδράσεις κουτιών για την εξάλειψη των υποψηφίων.

1. **X-Wing:**

Η τεχνική X-Wing προσδιορίζει ένα μοτίβο όπου δύο σειρές και δύο στήλες έχουν μόνο δύο υποψηφίους για έναν συγκεκριμένο αριθμό. Περιορίζει την τοποθέτηση αυτού του αριθμού στα τεμνόμενα πλαίσια. Ο κώδικας αναζητά μοτίβα X-Wing και τα χρησιμοποιεί για να εξαλείψει υποψηφίους.

1. **Coloring:**

Το coloring (χρωματισμός) βασίζεται στη θεωρία γραφημάτων, όπου τα κελιά και οι υποψήφιοι αριθμοί είναι κόμβοι σε ένα γράφημα. Προσδιορίζει εναλλασσόμενες αλυσίδες υποψηφίων (χρώματα) και εξαλείφει υποψηφίους που αποτελούν μέρος μιας αντίφασης. Το coloring είναι μια ισχυρή τεχνική για πολύπλοκα παζλ όπου εμφανίζονται μοτίβα αντιφάσεων.

1. **Y-Wing:**

Το Y-Wing βασίζεται σε μια αλυσίδα τριών κελιών, όπου δύο κελιά έχουν δύο υποψήφιους το καθένα και το τρίτο μοιράζεται έναν υποψήφιο και με τα δύο. Αυτή η τεχνική αναζητά διαμορφώσεις Y-Wing για να συναγάγει υποψήφιες εξαλείψεις σε άλλα κελιά. Ο κώδικας προσδιορίζει αποτελεσματικά τις ρυθμίσεις Y-Wing και τις χρησιμοποιεί για στρατηγικές εξαλείψεις.

1. **Nice Chains:**

Η Nice Chains επεκτείνει την έννοια του χρωματισμού εντοπίζοντας μεγαλύτερες αλυσίδες εναλλασσόμενων υποψηφίων. Εκμεταλλεύονται πολύπλοκα πρότυπα υποψηφίων για να εξαλείψουν αριθμούς που οδηγούν σε αντιφάσεις. Οι αλυσίδες Nice είναι ιδιαίτερα χρήσιμες για τα πιο απαιτητικά παζλ Sudoku.

1. **Medusa 3D:**

Το Medusa 3D είναι μια προηγμένη τεχνική που λαμβάνει υπόψη πολλαπλές διαστάσεις υποψηφίων σε κελιά, σειρές, στήλες και πλαίσια. Αναζητά σύνθετα μοτίβα που περιλαμβάνουν τρεις ή περισσότερους υποψηφίους για να κάνουν αφαιρέσεις. Το Medusa 3D είναι μια τεχνική τελευταίας λύσης για τα πιο απαιτητικά παζλ.

1. **Backtracking (Brute Force):**

Το Backtracking είναι μια στρατηγική που χρησιμοποιείται όταν όλες οι τεχνικές που βασίζονται στη λογική αποτυγχάνουν να προχωρήσουν περαιτέρω. Περιλαμβάνει τη συστηματική δοκιμή διαφορετικών υποψηφίων αριθμών για ένα κελί και τη συνέχιση της επίλυσης. Εάν παρουσιαστεί μια αντίφαση (π.χ. δύο αντικρουόμενοι αριθμοί στην ίδια σειρά), ο κωδικός επιστρέφει στην προηγούμενη κατάσταση και δοκιμάζει έναν διαφορετικό υποψήφιο. Αυτή η διαδικασία συνεχίζεται μέχρι να βρεθεί μια έγκυρη λύση ή να εξαντληθούν όλες οι δυνατότητες.

* **Αξιολόγηση της δυσκολίας του παζλ:**

Η υλοποίηση μπορεί να γίνει με μηχανική μάθηση, απλά τροφοδοτώντας το πρόγραμμα με γνωστά αποτελέσματα και προβλέποντας τυχαία δεδομένα, αλλά για την άσκηση αυτή βασιζόμαστε σε πέντε διαφορετικούς παράγοντες:

1. Αριθμός κενών κελιών (30%).
2. Αριθμός υποψηφίων σε κενά κελιά μετά από απλή εξάλειψη (25%).
3. Η διαφορά μεταξύ του συνόλου των υποψηφίων και των υπολοίπων υποψηφίων (20%).
4. Χρόνος που χρειάζεται για να λυθεί το παζλ (15%).
5. Αν χρησιμοποιήθηκε backtracking και πόσες φορές (10%).

Η συγκεκριμένη διαδικασία έχει ως αποτέλεσμα να συμπληρώνουν κάποιους πόντους τα αντίστοιχα επίπεδα. Συνεπώς όποιο επίπεδο έχει τον μεγαλύτερο αριθμό, τότε θεωρητικά καθορίζει και το επίπεδο του παζλ.

* **Μεθοδολογία επίλυσης:**

Ο solver χρησιμοποιεί μια συστηματική προσέγγιση, ξεκινώντας με απλή εξάλειψη και προχωρώντας μέσω προηγμένων μεθόδων. Ελέγχει συνεχώς τον αριθμό των κελιών που έχουν λυθεί και των υποψηφίων που απομένουν. Εάν οι μέθοδοι που βασίζονται στη λογική είναι ανεπαρκείς, καταφεύγει σε οπισθοδρόμηση. Ο λύτης δίνει προτεραιότητα στην εύρεση Hidden Singles και εφαρμόζει πιο σύνθετες μεθόδους όπως απαιτείται. Η διαδικασία συνεχίζεται μέχρι να λυθεί το παζλ ή να μην υπάρξει περαιτέρω πρόοδος.

Αποτέλεσμα λύσης παζλ με προηγμένες μεθόδους:

Από το παραπάνω παράδειγμα παρατηρούμε τις παρακάτω πληροφορίες:

A screenshot of a computer

Description automatically generated

* Πόσο χρόνο χρειάστηκε ο υπολογιστής
* Αν έγινε backtracking πόσο χρόνο χρειάστηκε και πόσες προσπάθειες έγιναν
* Ποιες μέθοδοι χρησιμοποιήθηκαν
* Λύση του προβλήματος
* Εκτίμηση επιπέδου του παζλ

1. **Υλοποίηση με ASP**

Ο ASP (Answer Set Programming) είναι ένα ισχυρό παράδειγμα δηλωτικού προγραμματισμού που χρησιμοποιείται για την επίλυση διαφόρων συνδυαστικών προβλημάτων, συμπεριλαμβανομένων των παζλ Sudoku. Σε αυτό το κεφάλαιο, θα διερευνήσουμε πώς μπορεί να χρησιμοποιηθεί ο ASP για την επίλυση Sudoku παζλ και θα δώσουμε μια σύντομη εξήγηση του παρεχόμενου κώδικα ASP.

* **Εισαγωγή στον ASP Sudoku Solver**

Τα Sudoku παζλ μπορούν να λυθούν αποτελεσματικά χρησιμοποιώντας ASP, ο οποίος μας επιτρέπει να ορίζουμε περιορισμούς και κανόνες για την επίλυση του παζλ. Ο παρεχόμενος κώδικας ASP αντιπροσωπεύει έναν λύτη Sudoku που χρησιμοποιεί τα ακόλουθα στοιχεία:

* **Ορισμός σταθερών**

#const num=3.

#const n=num\*num.

Αυτές οι γραμμές ορίζουν σταθερές για το μέγεθος του παζλ (αριθμός) και τον συνολικό αριθμό των κελιών (n) στο πλέγμα Sudoku.

* **Κωδικοποίηση της αρχικής κατάστασης του παζλ**

fixed(1,2,7). fixed(1,6,6). fixed(1,8,1).

... (άλλες σταθερές τιμές) ...

Τα σταθερά κατηγορήματα κωδικοποιούν τις αρχικές τιμές ορισμένων κελιών στο πλέγμα Sudoku. Αυτοί αντιπροσωπεύουν τους γνωστούς ή δεδομένους αριθμούς στο παζλ.

* **Καθορισμός εκχωρήσεων αξίας**

1{value(I,J,X): X=1..n}1 :- I=1..n, J=1..n.

Αυτός ο κανόνας ASP ορίζει τις πιθανές εκχωρήσεις τιμών (X) σε κάθε κελί (I, J) στο πλέγμα. Επιβάλλει ότι κάθε κελί μπορεί να έχει μόνο μία τιμή μεταξύ 1 και n.

* **Περιορισμός ίδιου πλαισίου**

sameBox(I1,J1,I2,J2) :- I1=1..n, J1=1..n, I2=1..n, J2=1..n, (I1-1)/num=(I2-1)/num, (J1-1)/num=(J2-1)/num.

Το κατηγόρημα sameBox προσδιορίζει κελιά που βρίσκονται στο ίδιο πλαίσιο ή τετράγωνο στο πλέγμα Sudoku. Διασφαλίζει ότι δύο κελιά δεν μπορούν να έχουν την ίδια τιμή εάν βρίσκονται στο ίδιο πλαίσιο.

* **Γενικοί Περιορισμοί**

:- sameBox(I1,J1,I2,J2), value(I1,J1,X), value(I2,J2,X), I1!=I2, J1!=J2.

:- value(I,J1,X), value(I,J2,X), J1!=J2.

:- value(I1,J,X), value(I2,J,X), I1!=I2.

Αυτοί οι κανόνες ορίζουν γενικούς περιορισμούς για τα παζλ Sudoku. Επιβάλλουν ότι κανένα κελί στο ίδιο πλαίσιο, γραμμή ή στήλη δεν μπορεί να έχει την ίδια τιμή (X).

* **Πρόσθετοι περιορισμοί παζλ**

:- value(I,J,X1), fixed(I,J,X2), X1!=X2.

Αυτός ο κανόνας διασφαλίζει ότι η τιμή που έχει εκχωρηθεί σε ένα κελί (I, J) δεν έρχεται σε αντίθεση με τις σταθερές τιμές που παρέχονται στην αρχική κατάσταση του παζλ.

* **Εκτύπωση της λύσης**

#show value/3.

Αυτή η γραμμή καθορίζει τι πρέπει να εμφανίζεται ως λύση. Εκτυπώνει τις τιμές που έχουν εκχωρηθεί σε κελιά στο λυμένο πλέγμα Sudoku.

Συμπέρασμα

Το ASP παρέχει μια κομψή και δηλωτική προσέγγιση για την επίλυση παζλ Sudoku κωδικοποιώντας τους περιορισμούς και τους κανόνες του παζλ. Ο παρεχόμενος κώδικας ASP λύνει αποτελεσματικά παζλ Sudoku εφαρμόζοντας αυτούς τους περιορισμούς και δημιουργώντας έγκυρες λύσεις. Αυτή η προσέγγιση καταδεικνύει την ευελιξία του ASP στην αντιμετώπιση πολύπλοκων συνδυαστικών προβλημάτων.

1. **Οδηγίες χρήσης προγράμματος**

Προϋπόθεση για την χρήση του προγράμματος είναι το λήψη της έκδοσης 2.1 της pulp με την εντολή:

**>>> pip3 install pulp==2.1**

Για την λειτουργία του προγράμματος τρέχουμε την εντολή

**>>> ./sudoku**

για να δούμε τις επιλογές του, αλλά μπορούμε να τρέξουμε την εντολή

**>>> ./ ilp-solvers /όνομα-επιλυτή < ολόκληρο-όνομα-αρχείου**

για να χρησιμοποιήσουμε ξεχωριστά κάθε επιλυτή παζλ. Π.χ. για να χρησιμοποιήσουμε τον Killer-Sudoku solver για το αρχείο test3x3\_1 από τα αρχεία test για Killer-Sudoku εκτελούμε την εντολή

**>>> ./ilp-solvers/killer-sudoku < tests-ks/test3x3\_1.in**

* **Υποσημείωση:** Οι παραπάνω εντολές ισχύουν μόνο για zshell, διότι τα Mac χρησιμοποιούν bash scripts.

A screenshot of a computer program

Description automatically generatedΣυνεχίζοντας με τον πρώτο τρόπο (μέσω της διεπαφής) θα αντικρύσουμε πρώτα το εξής κείμενο, το οποίο θα μας ζητάει input

Επιλέγουμε μια από τις δύο επιλογές, πληκτρολογώντας

**>>> ilp** ή

**>>> method**

A screenshot of a computer program

Description automatically generatedΞεκινάμε με τον Ακέραιο Γραμμικό Προγραμματισμό, οπότε εμφανίζονται οι παρακάτω επιλογές επιλυτών.

Για κάθε επιλογή, θα μας ζητήσει το όνομα του αρχείο εισόδου που θέλουμε να περάσουμε ως input στο πρόγραμμα (μόνο το όνομα του αρχείου, χωρίς τον φάκελο και χωρίς την κατάληξη .in). Μεταβαίνουμε πίσω με το input

**>>> exit**

Για τους επιλυτές Hyper-, X-, Four-Pyramids- Sudoku φαίνονται χρωματισμένες οι νέες περιοχές (των επιπρόσθετων περιορισμών).

A screenshot of a computer program

Description automatically generatedΓια το Killer-Sudoku:

Για το Greater-Than δεν υπάρχει υλοποίηση.

A computer screen shot of a number

Description automatically generatedΓια το Hyper-Sudoku:

A screenshot of a computer

Description automatically generatedΓια το X-Sudoku:

A screenshot of a computer

Description automatically generatedΓια το Four-Pyramids:

Για το Sandwich-Sudoku δεν υπάρχει υλοποίηση.

Για τη χρήση των προηγμένων μεθόδων αρκεί να πληκτρολογήσουμε

**>>> method**

στην αρχική φάση και μόλις πληκτρολογήσουμε το όνομα του αρχείου εισόδου θα διακρίνουμε το παρακάτω αποτέλεσμα:

A screenshot of a computer

Description automatically generated

Για χρήση του προγράμματος σε ASP αρκεί να πληκτρολογήσουμε την εντολή

**>>> clingo test3x3\_1.asp**

A screen shot of a computer

Description automatically generatedστο τερματικό. Προκύπτει το παρακάτω αποτέλεσμα:

Εάν υπάρχουν παραπάνω από μία περιπτώσεις, όλα τα αποτελέσματα μπορούν να εκτυπωθούν με την εντολή

**>>> clingo --models=0 <όνομα\_αρχείου>**

1. **Συμπέρασμα**

Η χρήση Ακέραιου Γραμμικού Προγραμματισμού (ILP) με PuLP παρέχει μια συστηματική και αποτελεσματική προσέγγιση για την επίλυση διαφόρων παραλλαγών παζλ Sudoku. Ο παρεχόμενος κώδικας Python δείχνει πώς να δημιουργήσετε τυποποιήσεις ILP για διαφορετικές παραλλαγές Sudoku και να αξιοποιήσετε το PuLP για να βρείτε λύσεις που ικανοποιούν όλους τους περιορισμούς. Είτε πρόκειται για ένα τυπικό Sudoku, ένα Killer Sudoku ή μια πιο περίπλοκη παραλλαγή, η προσέγγιση ILP με το PuLP προσφέρει έναν ευέλικτο τρόπο αντιμετώπισης αυτών των απαιτητικών παζλ. Επιπλέον δεν υπάρχει κάποιος περιορισμός για τον επιλυτή Sudoku παζλ, συνεπώς μπορεί να λύσει οποιοδήποτε έγκυρο παζλ, ακόμα και 25x25 παζλ με μόνο μειονέκτημα τον χρόνο επίλυσης του.

Τέλος παρατηρούμε από τις τρεις υλοποιήσεις Sudoku solver η καλύτερη χρονικά είναι με χρήση του ASP, αλλά η μορφή του αποτελέσματος είναι αρκετά δυσνόητη για τον χρήστη, διότι δεν βλέπει ξεκάθαρα το πλέγμα με τις τιμές.

1. **Βιβλιογραφία**

* Bartlett A.D., Chartier T.P., Langville A.N., Rankin T. An Integer Programming Model for the Sudoku Problem, J, Online Math. Applicat., vol. 8, 2008
* Simonis H., Sudoku as a Constraint Problem
* killersudokuonline.com, Killer Sudoku Killer Sudoku Online, 29-Sep-2012. Online. Available: http:killersudokuonline.com.Accessed: 30-Sep-2012
* <https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=13c88d2af1dcf28e31b6b88024dcfbde6accdad8>
* <https://towardsdatascience.com/solve-sudoku-using-linear-programming-python-pulp-b41b29f479f3>
* <https://diego.assencio.com/?index=25ea1e49ca59de51b4ef6885dcc3ee3b>
* <https://towardsdatascience.com/using-integer-linear-programming-to-solve-sudoku-puzzles-15e9d2a70baa>
* <https://m.media-amazon.com/images/I/71Ncao-sOlL._AC_UF1000,1000_QL80_.jpg>
* <https://www.aapelivuorinen.com/blog/2023/01/18/killer-sudoku-mip/>
* <http://stuckinthecube.blogspot.com/2007/05/solving-greater-than-sudoku-puzzles.html>
* <https://www.sudokuonline.io/tips/sudoku-x-wing>
* <http://educ.jmu.edu/~taalmala/POW_spring2006/POW4.html>
* <https://dkmgames.com/SandwichSudoku/>