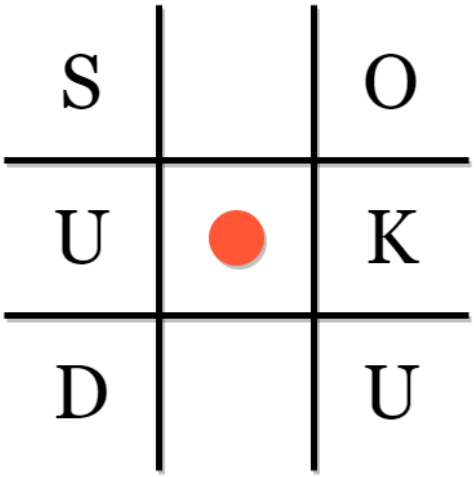
ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
& ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΠΑΝΕΠΙΣΤΗΜΙΟ ΔΥΤΙΚΗΣ ΜΑΚΕΔΟΝΙΑΣ
ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

ΕΠΙΛΥΣΗ ΠΡΟΒΛΗΜΑΤΩΝ SUDOKU ΜΕΣΩ ΔΙΑΔΟΣΗΣ ΠΕΡΙΟΡΙΣΜΩΝ



Ζέλιος Ανδρέας

Υπό την επίβλεψη και καθοδήγηση του καθηγητή Στεργίου Κωνσταντίνου

Κοζάνη

1 Νοεμβρίου 2022

Περίληψη

Ένα από τα βασικά παραδείγματα που χρησιμοποιούνται στη διδασκαλία του προγραμματισμού περιορισμών είναι αυτό του προβλήματος sudoku. Το sudoku μπορεί να λυθεί ως πρόβλημα ικανοποίησης περιορισμών με διάφορους τρόπους, δύο από αυτούς είναι η τεχνική αναζήτησης με οπισθοδρόμηση και η τεχνική διάδοσης περιορισμών.

Η παρούσα εργασία εστιάζει στη μελέτη του δεύτερου τρόπου, δηλαδή στην χρήση τεχνικών διάδοσης περιορισμών για την μείωση των διαθέσιμων τιμών που μπορεί να πάρει κάθε κελί sudoku επιβάλλοντας τοπική συνέπεια περιορισμών (local consistency) στους περιορισμούς του προβλήματος.

Πιο συγκεκριμένα, υλοποιήθηκαν, εξετάστηκαν αλλά και συγκρίθηκαν αλγόριθμοι που εφαρμόζουν τις ιδιότητες Generalized Arc Consistency (GAC) και Singleton Generalized Arc Consistency (SGAC) για την διάδοση περιορισμών καθώς και η χρήση του περιορισμού alldifferent έναντι απλών δυαδικών περιορισμών.

Τέλος, λόγω της αξίας που προσφέρει το sudoku στη διδασκαλία αυτών των ιδιοτήτων, δημιουργήθηκε μια εφαρμογή για την οπτικοποίηση της σύγκρισης των εκτελέσεων των δύο αλγορίθμων ως ένα βοηθητικό εργαλείο στην διαδικασία εκμάθησής τους.

Λέξεις κλειδιά: Προγραμματισμός περιορισμών, διάδοση περιορισμών, πρόβλημα sudoku, αλγόριθμοι ελέγχου συνέπειας

Abstract

One of the basic examples used in teaching constraint programming is that of the sudoku problem. Sudoku can be solved as a constraint satisfaction problem in various ways, two of which are search with backtracking and the constraint propagation technique.

The present work focuses on the study of the second, i.e. the use of constraint propagation techniques to reduce the available values that each sudoku cell can obtain by imposing local consistency on the constraints of the problem.

More specifically, algorithms that apply the Generalized Arc Consistency (GAC) and Singleton Generalized Arc Consistency (SGAC) properties to propagate constraints as well as the use of alldifferent constraint versus simple binary constraints were implemented, studied and compared.

Finally, due to the value that sudoku offers in teaching these properties, an application was created to visualize the comparison of the performances of the two algorithms as an auxiliary tool in the learning process.

Key words: Constraint programming, constraint propagation, sudoku puzzle, consistency check algorithms

Δήλωση Πνευματικών Δικαιωμάτων

Δηλώνω ρητά ότι, σύμφωνα με το άρθρο 8 του Ν. 1599/1986 και τα άρθρα 2,4,6 παρ. 3 του Ν. 1256/1982, η παρούσα Διπλωματική Εργασία με τίτλο

" ΕΠΙΛΥΣΗ ΠΡΟΒΛΗΜΑΤΩΝ SUDOKU ΜΕΣΩ ΔΙΑΔΟΣΗΣ ΠΕΡΙΟΡΙΣΜΩΝ " καθώς και τα ηλεκτρονικά αρχεία και πηγαίοι κώδικες που αναπτύχθηκαν ή τροποποιήθηκαν στα πλαίσια αυτής της εργασίας και αναφέρονται ρητώς μέσα στο κείμενο που συνοδεύουν, και η οποία έχει εκπονηθεί στο Τμήμα Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών του Πανεπιστημίου Δυτικής Μακεδονίας, υπό την επίβλεψη του μέλους του Τμήματος κ. Στεργίου Κωνσταντίνου αποτελεί αποκλειστικά προϊόν προσωπικής εργασίας και δεν προσβάλλει κάθε μορφής πνευματικά δικαιώματα τρίτων και δεν είναι προϊόν μερικής ή ολικής αντιγραφής, οι πηγές δε που χρησιμοποιήθηκαν περιορίζονται στις βιβλιογραφικές αναφορές και μόνον. Τα σημεία όπου έχω χρησιμοποιήσει ιδέες, κείμενο, αρχεία ή / και πηγές άλλων συγγραφέων, αναφέρονται ευδιάκριτα στο κείμενο με την κατάλληλη παραπομπή και η σχετική αναφορά περιλαμβάνεται στο τμήμα των βιβλιογραφικών αναφορών με πλήρη περιγραφή.

Υπογραφή Φοιτητή:

Περιεχόμενα

Κατάλογος σχημάτων - συναρτήσεων - γραφημάτων

Κατάλογος πινάκων

ΕΥΧΑΡΙΣΤΙΕΣ

Αρχικά θα ήθελα να ευχαριστήσω τον επιβλέποντα καθηγητή μου Στεργίου Κωνσταντίνο, που μέσω της διδασκαλίας και της καθοδήγησής του μου κίνησε το ενδιαφέρον στον προγραμματισμό από τα πρώτα έτη της φοίτησής μου, καθώς και με εισήγαγε στον προγραμματισμό περιορισμών που αποτελεί και το θέμα της παρούσας διπλωματικής.

Επιπρόσθετα, αποδίδω θερμά ευχαριστώ στα κοντινά μου πρόσωπα που μου συμπαραστάθηκαν ηθικά, ο καθένας με τον δικό του μοναδικό τρόπο, στην προσπάθεια εκπόνησης της εν λόγω εργασίας.

Κεφάλαιο 1

Εισαγωγή

Σε αυτό το κεφάλαιο αναφέρουμε το κίνητρο για τη συγγραφή της εργασίας και την οργάνωση της σε κεφάλαια.

1.1 Κίνητρο

Κίνητρο για την υλοποίηση της εργασίας είναι η μελέτη του sudoku ως προβλήματος ικανοποίησης περιορισμών. Πιο συγκεκριμένα διερευνώνται τρόποι επίλυσής του με μεθόδους διάδοσης περιορισμών, καθώς και η σύγκριση και ο έλεγχος της απόδοσης αυτών των μεθόδων.

1.2 Περιγραφή προβλήματος sudoku

Το sudoku είναι ένα πρόβλημα που παίζεται σε 9x9 sudoku ταμπλό που χωρίζεται σε εννιά 3x3 πλέγματα. Ένα sudoku ταμπλό δηλαδή αποτελείται από 81 σε σύνολο κελιά, με κάποια από αυτά να έχουν προσυμπληρωμένες τιμές. Σκοπός του παιχνιδιού sudoku είναι, κάθε γραμμή, στήλη και 3x3 πλέγμα στο sudoku ταμπλό να περιέχει τους αριθμούς 1 έως 9 μόνο μία φορά. Υπάρχουν και παραλλαγές sudoku με ταμπλό μεγέθους 4x4, 6x6, 16x16 και άλλες όπως χρωματιστό sudoku ή sudoku με λέξεις, όμως στην παρούσα εργασία επικεντρωνόμαστε στο κλασσικό πρόβλημα sudoku.

1.3 Οργάνωση διπλωματικής

Η διπλωματική εργασία έχει οργανωθεί ως εξής. Στο κεφάλαιο 2 αναφέρονται οι βασικές έννοιες του προγραμματισμού περιορισμών, παραδείγματα προβλημάτων ικανοποίησης περιορισμών καθώς και οι δομές και τεχνικές επεξεργασίας περιορισμών που χρησιμοποιήθηκαν. Στο κεφάλαιο 3 περιγράφονται συνοπτικά οι αλγόριθμοι που υλοποιήθηκαν σε μορφή κώδικα για την εφαρμογή των ιδιοτήτων διάδοσης περιορισμών GAC και SGAC. Στο κεφάλαιο 4 αποτυπώνονται τα αποτελέσματα των πειραμάτων που έγιναν για την σύγκριση των δύο ιδιοτήτων σε μορφή πινάκων καθώς και η ερμηνεία τους ανά κατηγορία σύγκρισης. Στο κεφάλαιο 5 επεξηγείται η εφαρμογή που δημιουργήθηκε για την οπτικοποίηση της σύγκρισης της εκτέλεσης των δύο αλγορίθμων, ενώ τέλος στο κεφάλαιο 6 δίνονται τα συμπεράσματα της μελέτης καθώς και μελλοντικές βελτιώσεις-επεκτάσεις.

Κεφάλαιο 2

Επεξεργασία Περιορισμών

2.1 Προβλήματα ικανοποίησης περιορισμών

Ένα πρόβλημα ικανοποίησης περιορισμών (constraint satisfaction problem – CSP) περιέχει ένα σύνολο από μεταβλητές που πρέπει να πάρουν τιμές υπό την επιβολή ενός συνόλου περιορισμών [**1**]. Ένας τρόπος διάκρισης των περιορισμών είναι βάσει του πλήθους των μεταβλητών που επιβάλλονται, έτσι υπάρχουν:

- Μοναδιαίοι περιορισμοί, περιορισμοί δηλαδή που επιβάλλονται σε μία μόνο μεταβλητή
- Δυαδικοί περιορισμοί, περιορισμοί δηλαδή που επιβάλλονται σε δύο μεταβλητές
- n-αδικοί περιορισμοί, περιορισμοί δηλαδή που επιβάλλονται σε τρείς ή περισσότερες μεταβλητές

Οι μεταβλητές έχουν ένα συγκεκριμένο πεδίο ορισμού (συγκεκριμένες τιμές που είναι επιτρεπτές). Για παράδειγμα εάν μία μεταβλητή αντιπροσωπεύει το πλήθος των πελατών σε ένα σουπερμάρκετ που έχει χωρητικότητα 500 άτομα, τότε οι τιμές που μπορεί να πάρει αυτή η μεταβλητή θα είναι στο εύρος 0 έως 500. Ο κάθε περιορισμός ορίζεται σε ένα υποσύνολο του αρχικού συνόλου μεταβλητών, θέτοντας όρια στις τιμές που μπορούν να ανατεθούν στις συγκεκριμένες μεταβλητές. Στο προηγούμενο παράδειγμα ένας περιορισμός που τέθηκε πρόσφατα, εν μέσω της πανδημίας COVID 19, στην μεταβλητή είναι η είσοδος ενός ατόμου ανά 9 τετραγωνικά μέτρα, αφαιρώντας από το πεδίο ορισμού της μεταβλητής τις τιμές 300 έως 500. Απώτερος σκοπός ενός προβλήματος περιορισμών είναι η εύρεση της ανάθεσης τιμών στις μεταβλητές έτσι ώστε η ανάθεση να ικανοποιεί όλους τους περιορισμούς.

Τα CSPs είναι συνδυαστικά [**2**], δηλαδή υπάρχουν πολλές εξαρτήσεις ανάμεσα στις μεταβλητές μέσω των περιορισμών και έτσι αυτά τα προβλήματα στην γενική τους μορφή ανήκουν στην κατηγορία NP-Complete, προβλήματα δηλαδή που στην χειρότερη περίπτωση επιλύονται σε εκθετικό χρόνο και η λύση μπορεί να επαληθευτεί πολυωνυμικά. Ένα από τα προβλήματα που μπορεί να μοντελοποιηθεί ως CSP είναι και αυτό της επίλυσης ενός sudoku [**3**].

2.1.1 Παραδείγματα CSPs

Τα CSPs εμφανίζονται σε πολλούς τομείς της επιστήμης υπολογιστών καθώς και του κλάδου της Επιχειρησιακής Έρευνας που έχει ως στόχο την εύρεση βέλτιστων λύσεων σε συνδυαστικά προβλήματα, μιας και πολλά συνδυαστικά προβλήματα μπορούν να μοντελοποιηθούν ως CSPs [**4**].

Ένα από τα βασικά προβλήματα που μπορούν να εκφραστούν ως CSPs είναι αυτό της χρονοδρομολόγησης. Το πρόβλημα του Job-Shop Scheduling είναι ένα πολυδιάστατο πρόβλημα χρονοδρομολόγησης που απαιτεί εργασίες να επιλεγούν, ενταχθούν σε σειρά προτεραιότητας και ανατεθούν σε αυτές πόροι και χρόνοι εκτέλεσης [**5**][**6**][**7**]. Όπως αναφέρει ο M. S. Fox, για 10 μόνο παραγγελίες που μετακινούνται μεταξύ 5 επιχειρήσεων, έχοντας μία μηχανή ανά εργασία και την προϋπόθεση πως το χρονοπρόγραμμα που θα δημιουργηθεί θα είναι χωρίς χρονικά κενά, υπάρχουν $(10!)^5$ ή 10^{32} δυνατά χρονοπρογράμματα. Παρ' όλα αυτά ο Fox περιγράφει διάφορους τρόπους προγραμματισμού περιορισμών που μπορούν να δημιουργήσουν λεπτομερή χρονοπρογράμματα δίνοντας λύση στο πρόβλημα [**5**].

Το University Course Timetabling problem αποτελεί ένα παραπλήσιο πρόβλημα που συνεχίζει να απασχολεί την ακαδημαϊκή κοινότητα. Στόχος του προβλήματος είναι η δημιουργία χρονοπρογραμμάτων για τον συντονισμό μαθητών, καθηγητών και τάξεων κατά την περίοδο εξεταστικής σε εκπαιδευτικά ιδρύματα [**8**]. Σε συνέντευξή τους, ο διευθύνων σύμβουλος της ILOG (εταιρία που ενσωματώθηκε στην IBM) Pierre Haren και ο διευθύνων σύμβουλος της COSYTEC, εταιρίας παροχής υπηρεσιών προγραμματισμού περιορισμών, Mehmet Dincbas τονίζουν την σημασία του προγραμματισμού περιορισμών [**9**]. Συγκεκριμένα επισημαίνουν τη συμβολή του στον προγραμματισμό εντός του κλάδου των μεταποιητικών βιομηχανιών, βραχυπρόθεσμα ή σε πραγματικό χρόνο και την διαχείριση του υλικού της παραγωγής. Στην κατηγορία τέτοιων προβλημάτων μπορούμε να εντάξουμε και τα προβλήματα Crew Scheduling problems, δηλαδή τη βέλτιστη κατανομή του ανθρώπινου δυναμικού σε διαφορετικά σενάρια (αστικές συγκοινωνίες, πτήσεις κτλ.) με σκοπό την εξασφάλιση της υγείας του προσωπικού και την ταυτόχρονη αύξηση του κέρδους [**10**][**11**][**12**][**13**][**14**].

Μία άλλη κατηγορία προβλημάτων που επιλύονται ως CSP είναι τα προβλήματα σχεδιασμού, προβλήματα δηλαδή όπου ψάχνουμε τη σωστή ή βέλτιστη διευθέτηση αντικειμένων με βάση αποστάσεις, διαθέσιμο χώρο ή άλλους περιορισμούς. Τέτοια προβλήματα αφορούν ηλεκτρολόγους και αρχιτέκτονες μηχανικούς. Άλλο ένα πρόβλημα αποτελεί το Circuit Design problem δηλαδή ο ορισμός των παραμέτρων (θέση, τρόπος ένωσης) των επιμέρους στοιχείων που θα αποτελέσουν ένα ενιαίο κύκλωμα με συγκεκριμένα δοσμένα επιθυμητά αποτελέσματα. Οι J. Kleers, G. Sussman παρέχουν μια μέθοδο εύρεσης του επιθυμητού σχεδιασμού ενός ηλεκτρικού κυκλώματος μέσω διάδοσης περιορισμών [**17**].

Το Network Design problem είναι ένα άλλο παραπλήσιο πρόβλημα που επιχειρεί την διαστασιολόγηση των audio return channels (arcs) ενός τηλεπικοινωνιακού δικτύου με τέτοιο τρόπο ώστε, να μπορεί να βελτιστοποιηθεί η ταυτόχρονη δρομολόγηση κλήσεων μέσω του δικτύου, χωρίς την υπέρβαση της χωρητικότητας των επιλεγμένων arcs [**18**].

2.1.2 Τρόποι επίλυσης CSPs

Όπως αναφέραμε, τα CSPs είναι προβλήματα που είναι NP-Complete. Λόγω της πολυπλοκότητας τους έχουν προταθεί διάφοροι τρόποι εύρεσης μιας λύσης που γενικώς χωρίζονται σε Inference methods (μέθοδοι διάδοσης περιορισμών) και Search methods (μέθοδοι αναζήτησης) [**19**]. Έχοντας ένα πεπερασμένο πεδίο ορισμού D, με D_i το πεδίο ορισμού κάθε i μεταβλητής στο πρόβλημα, υπάρχει ένας πεπερασμένος χώρος αναζήτησης υποθετικών λύσεων $\Omega = D_0 \times ... \times D_n$ που ορίζεται από το καρτεσιανό γινόμενο των πεδίων τιμών των μεταβλητών, δηλαδή όλους τους συνδυασμούς των τιμών τους. Η Brute Force προσέγγιση αποτελεί μια απλή λύση σε ένα CSP στην οποία όλες οι δυνατές πλειάδες ελέγχονται για το εάν είναι λύσεις στο πρόβλημα.

Στον προγραμματισμό περιορισμών η προσέγγιση αυτή βελτιώνεται με:

1. Search methods, δηλαδή μεθόδους που εξερευνούν τον Ω αναθέτοντας μία τιμή σε μία μεταβλητή και συνεχίζοντας αντίστοιχα για όλες τις υπόλοιπες. Οι αναθέσεις μπορεί να παραβιάζουν κάποιο περιορισμό, έτσι σε κάθε αποτυχία εύρεση λύσης, ο αριθμός των δυνατών πλειάδων που μπορούν να αποτελέσουν λύση μειώνεται και συνεχίζεται η διαδικασία με άλλη ανάθεση τιμής έως ότου τελειώσουν όλοι οι δυνατοί συνδυασμοί αναθέσεων ή βρεθεί λύση. Η μέθοδος οπισθοδρόμησης (Backtracking) συνήθως συνδυάζεται με τη μέθοδο αναζήτησης όντας μια ολοκληρωμένη μέθοδος στα προβλήματα περιορισμών, δηλαδή εγγυάται να βρει λύση εάν υπάρχει [**20**][**21**][**22**]. Η βασική μέθοδος αναζήτησης με οπισθοδρόμηση δημιουργεί μία μερική λύση αναθέτοντας τιμές σε μεταβλητές μέχρι ενός σημείου αποτυχίας (δηλαδή δεν έχουμε οδηγηθεί σε πλήρη λύση). Η μέθοδος τότε επιστρέφει στην τελευταία ανάθεση πριν την αποτυχία και συνεχίζει με μία άλλη ανάθεση. Για να μειωθεί ο χρόνος εκτέλεσης, ευρετικές μέθοδοι (Heuristics) έχουν ενσωματωθεί στον βασικό αλγόριθμο αναζήτησης με οπισθοδρόμηση για την αποδοτική επιλογή μεταβλητών και τιμών κατά την διάρκεια της αναζήτησης.
2. Inference methods, δηλαδή μεθόδους που χρησιμοποιώντας τοπική διάδοση περιορισμών, μειώνουν τον αριθμό των δυνατών πλειάδων που μπορούν να αποτελέσουν λύση από τον Ω με γνώμονα ότι αυτές δεν αποτελούν λύση στο πρόβλημα. Οι μέθοδοι αυτές εντοπίζουν τιμές σε πεδία τιμών που δεν είναι συνεπείς σε σχέση με κάποιους περιορισμούς και τις διαγράφουν. Αυτή η διαδικασία συνεχίζεται μέχρι να μην μπορούν να γίνουν άλλες διαγραφές. Οι κύριες μέθοδοι είναι οι Node Consistency, που αφαιρεί τιμές από τα πεδία ορισμού βάσει μόνο μοναδιαίων περιορισμών του προβλήματος, Arc Consistency που αφαιρεί τιμές από τα πεδία ορισμού βάσει δυαδικών περιορισμών [**23**][**26**], Path Consistency που επιβάλλει μεγαλύτερο βαθμό συνέπειας σε ένα γράφο περιορισμών για δυαδικούς περιορισμούς και Generalized Arc Consistency, Singleton Generalized Arc Consistency για n-αδικούς περιορισμούς [**24**]. Οι τεχνικές αυτές χρησιμοποιούνται είτε για προεπεξεργασία είτε κατά την διάρκεια της αναζήτησης.

Οι μέθοδοι backtracking, όπως αναφέραμε επιλέγουν μία άλλη ανάθεση στην θέση αυτής που προκάλεσε σφάλμα. Τι γίνεται όμως εάν οι αναθέσεις που προκαλούν

σφάλμα επαναλαμβάνονται; Αυτό αποτελεί ένα συχνό φαινόμενο στη μέθοδο αναζήτησης με οπισθοδρόμηση μιας και το δέντρο αναζήτησης θα περιέχει αρκετές φορές το μονοπάτι που οδηγεί σε αποτυχία [**25**]. Οι μέθοδοι inference δημιουργήθηκαν για την εύρεση και εξάλειψη όσο το δυνατόν περισσότερων τέτοιων περιπτώσεων. Έτσι μέθοδοι search μπορούν να συνδυαστούν με μεθόδους inference για καλύτερη απόδοση. Στην συγκεκριμένη εργασία όμως θα ασχοληθούμε αποκλειστικά με μεθόδους inference (διάδοσης περιορισμών).

2.2 Βασικές έννοιες του προγραμματισμού περιορισμών

Ένα πρόβλημα ικανοποίησης περιορισμών, αποτελείται από τρία συστατικά [**19**]: μια πλειάδα n μεταβλητών του προβλήματος X = $\{X0, \dots, Xn\}$ (variables), μια πλειάδα με τα αντίστοιχα πεδία ορισμού κάθε μεταβλητής D = $\{D_1, \dots, D_n\}$ (domains) καθώς και μια πλειάδα περιορισμών C = $\{C_1, \dots, C_t\}$ (constraints). Κάθε περιορισμός C_i που ανήκει στο C, αναφέρεται σε κάποιο υποσύνολο των μεταβλητών και καθορίζει τους επιτρεπτούς συνδυασμούς τιμών για αυτό το υποσύνολο. Για παράδειγμα έχοντας τις μεταβλητές X1, X2, X3, X4, X5, ένας δυνατός περιορισμός πάνω στο υποσύνολο X1, X2, X4, X5 θα μπορούσε να είναι X1 + X2 + X5 > X4. Το άθροισμα των τιμών που έχουν οι μεταβλητές X1, X2 και X5 έτσι, δεν πρέπει να ξεπερνά την τιμή που έχει η μεταβλητή X4. Μια κατάσταση του προβλήματος ορίζεται ως ανάθεση τιμών σε μερικές ή όλες τις μεταβλητές. Μια ανάθεση τιμής που δεν παραβιάζει κανένα περιορισμό ονομάζεται συνεπής (consistent) ή νόμιμη ανάθεση. Πλήρης ανάθεση είναι μια ανάθεση που περιλαμβάνει όλες τις μεταβλητές και λύση ενός CSP είναι μια πλήρης ανάθεση τιμών που ικανοποιεί όλους τους περιορισμούς. Οι λύσεις-αναθέσεις ενός CSP μπορεί να είναι πολλές, μία ή και καμία. Η εμβέλεια (scope) κάθε περιορισμού είναι οι μεταβλητές στις οποίες εφαρμόζεται [**41**].

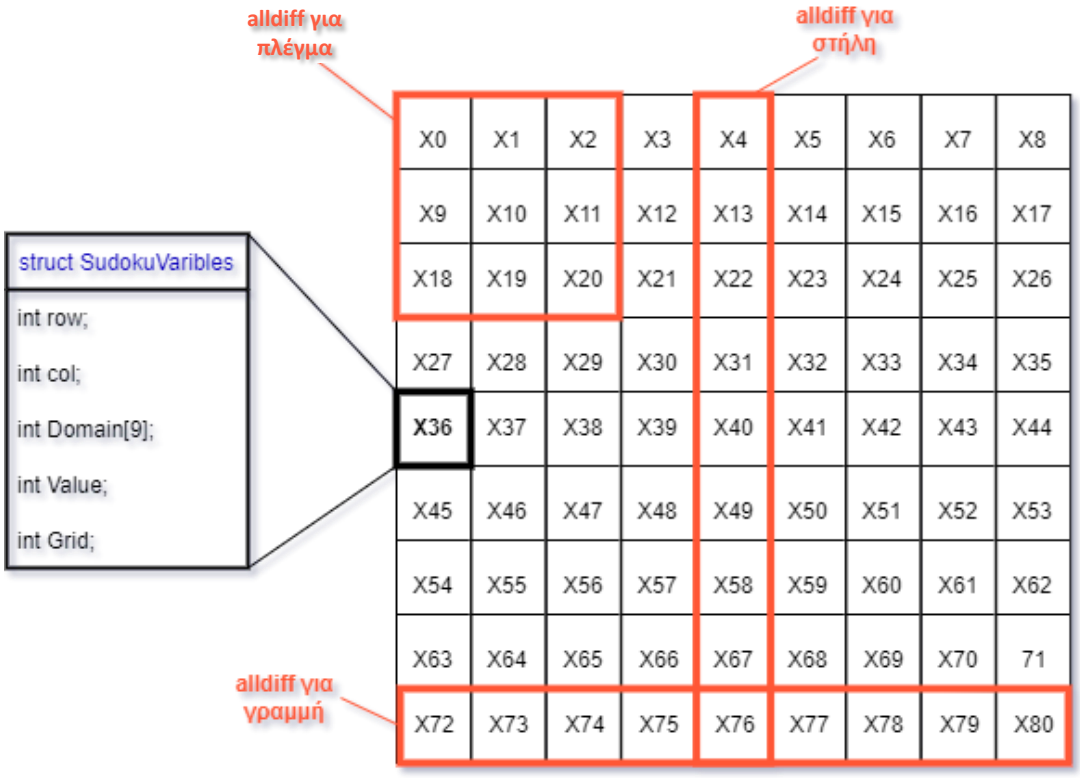
Ένας απλός δυαδικός περιορισμός C_{01} ή περιορισμός ανάμεσα σε δύο μεταβλητές $X0$ και $X1$ (scope οι μεταβλητές X0, X1) είναι συνεπής (consistent) εάν για κάθε α ∈ D_0, υπάρχει β ∈ D_1 τέτοιο ώστε η ανάθεση $X0$ = α και $X1$ = β να ικανοποιεί τον περιορισμό C_{01}. Ένας βασικός δυαδικός περιορισμός είναι ο περιορισμός διάφορου (≠ ή diff) που εξασφαλίζει ότι οι δύο μεταβλητές στις οποίες εφαρμόζεται παίρνουν διαφορετικές τιμές, διαφορετικά ο περιορισμός δεν είναι συνεπής. Ένα άλλο είδος περιορισμών είναι αυτό των καθολικών περιορισμών (Global Constraints). Οι global περιορισμοί, είναι περιορισμοί που εκφράζουν την σχέση μεταξύ ενός μη σταθερού συνόλου μεταβλητών και μερικές φορές μπορούν να εκφραστούν από ένα σύνολο απλούστερων περιορισμών. Με αυτή τους την ιδιότητα μειώνουν το πλήθος των περιορισμών του προβλήματος παρέχοντας έτσι και μία πιο καθαρή οπτική της δομής του προβλήματος. Ένα βασικό παράδειγμα ενός τέτοιου περιορισμού είναι ο alldifferent που εξασφαλίζει ότι σε κάθε μεταβλητή σε ένα σύνολο μεταβλητών ανατίθεται διαφορετική τιμή.

2.3 Μοντελοποίηση του προβλήματος

Για την μοντελοποίηση του sudoku ως προβλήματος ικανοποίησης περιορισμών, κάθε κελί sudoku αντιστοιχίστηκε σε μία μεταβλητή Xi έχοντας ένα πεδίο ορισμού D = {1, …, 9}, όπου i = 0, …, 80 δηλαδή 81 συνολικά μεταβλητές. Κάθε μεταβλητή βρίσκεται αντίστοιχα σε μία στήλη, μία γραμμή και ένα πλέγμα. Μια αναπαράσταση των προαναφερθέντων παρατίθενται στο (Σχ. 1).

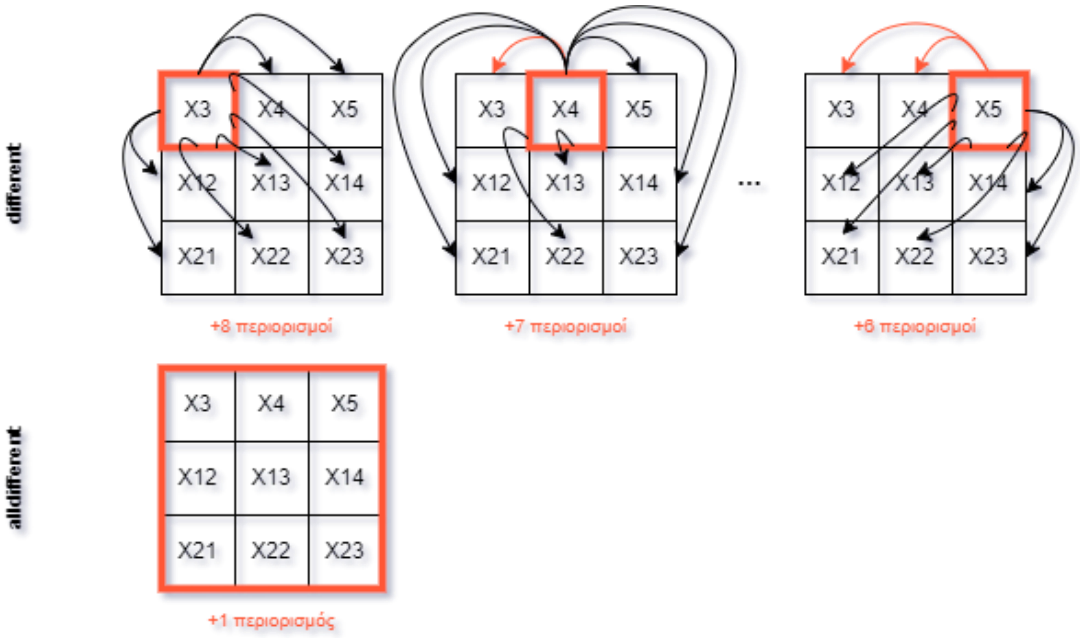
Οι περιορισμοί που εφαρμόστηκαν ήταν αποκλειστικά ο alldifferent περιορισμός σε κάθε πλέγμα, γραμμή και στήλη του sudoku αντίστοιχα. Η διαφορά του με τον diff περιορισμό, είναι πως αντί για δυαδικούς περιορισμούς ανάμεσα στις μεταβλητές, έχουμε n-αδικους περιορισμούς.

Μπορούμε να δείξουμε και την ελαχιστοποίηση του πλήθους των περιορισμών εφαρμόζοντας τον περιορισμό alldifferent έναντι του απλού different με ένα απλό παράδειγμα. Βλέποντας το 2^{ο} πλέγμα του sudoku με τις μεταβλητές X3, X4, X5, X12, X13, X14, X21, X22, X23, εάν είχαμε ένα diff περιορισμό για κάθε δυάδα που μπορεί να δημιουργηθεί, θα είχαμε n(n-1)/2 = 72/2 = 36 διαφορετικούς περιορισμούς όπως φαίνεται στο (Σχ. 2). Αντιθέτως με έναν περιορισμό alldifferent μειώνουμε όλους αυτούς τους περιορισμούς σε έναν.



Σχήμα 1. Δομή του sudoku

Εάν επαναλαμβάναμε την διαδικασία για όλο το πρόβλημα του sudoku θα είχαμε συνολικά 810 diff περιορισμούς ενώ μόλις 27 περιορισμούς alldifferent.



Σχήμα 2. Diff και alldifferent περιορισμός για πλέγμα sudoku

2.4 Διάδοση Περιορισμών

Οι ιδιότητες διάδοσης περιορισμών που μελετήθηκαν ήταν η Generalized Arc Consistency και η Singleton Generalized Arc Consistency που εφαρμόζονται σε n-αδικούς περιορισμούς. Για να περιγράψουμε τις δύο αυτές ιδιότητες θα πρέπει να περιγράψουμε και τις απλές ιδιότητες Arc Consistency καθώς και Singleton Arc Consistency που εφαρμόζονται σε δυαδικούς περιορισμούς.

2.4.1 Arc Consistency

Η ιδιότητα Arc Consistency εφαρμόζεται από όλους τους solvers και κατά την διάρκεια της αναζήτησης. Αυτό επιτυγχάνεται μέσω του ελέγχου συνέπειας (Consistency Check), δηλαδή της διαγραφής από το πεδίο ορισμού κάθε μεταβλητής εκείνων των τιμών που δεν είναι συνεπείς ως προς κάποιον περιορισμό. Ένα τόξο (arc) είναι ένας κατευθυνόμενος περιορισμός. Έτσι για κάθε δυαδικό περιορισμό που συμμετέχουν δύο μεταβλητές X1, X2, θα υπάρχουν δύο arcs (X1 → X2 και X2 → X1). Κάθε arc είναι consistent εάν, για κάθε τιμή της X1 υπάρχει τουλάχιστον μία συμβατή τιμή στο πεδίο ορισμού της X2. Ένα CSP είναι arc consistent εάν όλα τα ζεύγη μεταβλητών είναι arc consistent. Με έναν πιο επίσημο τρόπο [**29**]:

Μια ανάθεση τιμής σε μεταβλητή (Xi, α) είναι node consistent, εάν η ανάθεση αυτή επιτρέπεται από τους μοναδιαίους περιορισμούς που εφαρμόζονται στην μεταβλητή C_i, $\alpha \in C_i$. Η μεταβλητή Xi είναι node consistent, εάν όλες οι τιμές του domain της είναι node consistent. Ένα CSP είναι node consistent εάν κάθε μεταβλητή του είναι node consistent. Μια ανάθεση τιμής σε μεταβλητή (Xi, α) που έχει ένα περιορισμό C_{ij} (περιορισμό ανάμεσα σε αυτή και την μεταβλητή Xj), είναι arc consistent, εάν είναι

node consistent και υπάρχει μια τιμή $\beta \in D_j$ τέτοια ώστε $(\alpha, \beta) \in C_{ij}$, δηλαδή οι αναθέσεις $Xi = \alpha$ και $Xj = \beta$ είναι επιτρεπτές από τον περιορισμό C_{ij}. Η τιμή β λέγεται support (υποστήριξη) της α. Η μεταβλητή Xi είναι arc consistent εάν όλες οι τιμές στο πεδίο ορισμού της, είναι arc consistent για όλους τους δυαδικούς περιορισμού που περιέχουν την μεταβλητή Xi. Ένα CSP είναι arc consistent εάν κάθε μεταβλητή του είναι arc consistent.

2.4.2 Generalized Arc Consistency

Η ιδιότητα Generalized Arc Consistency [**27**] είναι μια επέκταση της ιδιότητας Arc Consistency για n-αδικούς περιορισμούς.

Ορισμός: (Generalized Arc Consistency GAC) Έστω ότι C είναι ένας περιορισμός στις μεταβλητές X1, …, Xn με αντίστοιχα domains D_1, …, D_n στα οποία ισχύει $C \subseteq D_1 \times \ldots \times D_n$. Θα λέμε ότι ο C είναι generalized arc consistent, εάν για κάθε $1 \leq i \leq n$ και $v \in D_i$ υπάρχει μία πλειάδα $(d_1, \ldots, d_n) \in C$ τέτοια ώστε $d_1 = v$. Ένα CSP είναι generalized arc consistent εάν κάθε ένας από τους περιορισμούς του είναι arc consistent [**28**].

Η ιδιότητα GAC εξασφαλίζει ότι κάθε τιμή στο πεδίο ορισμού κάθε μεταβλητής αποτελεί μέρος μιας επιτρεπτής ανάθεσης. Ένας αποδοτικός αλγόριθμος για την εφαρμογή της GAC σε alldiferent περιορισμούς έχει προταχθεί από τον Régin έχοντας πολυπλοκότητα $O(s^2d^2)$ [**31**].Το s εκφράζει το πλήθος των μεταβλητών που συμμετέχουν στο περιορισμό και d το νούμερο των διακριτών τιμών στα πεδία ορισμού των μεταβλητών. Ο αλγόριθμος του Régin χρησιμοποιεί γράφους και γι' αυτό τον λόγο κρίνεται απαραίτητη η παράθεση κάποιων βασικών εννοιών από την θεωρία των γράφων.

Ορισμός: Ένας γράφος G = (V, E) αποτελείται από ένα πεπερασμένο, κενό σύνολο κορυφών V (vertices) και ένα σύνολο από ακμές E (edges). Εάν οι ακμές είναι αριθμημένα κατευθυνόμενα ζεύγη κορυφών της μορφής (v, w), τότε ο γράφος ονομάζεται κατευθυνόμενος, με v την ουρά (tail) και w την κεφαλή (head) της ακμής. Εάν οι ακμές είναι μη-αριθμημένα ζεύγη ξεχωριστών κορυφών που στερούνται διάταξης, τότε ο γράφος ονομάζεται μη κατευθυνόμενος [**39**].

Ο κατευθυνόμενος γράφος που χρησιμοποιείται μάλιστα από τον αλγόριθμο που εφαρμόζει GAC σε alldifferent περιορισμούς είναι διμερής (bipartite) [**40**]. Ένας διμερής κατευθυνόμενος γράφος, είναι ένας γράφος όπου οι κορυφές μπορούν να χωριστούν σε δύο σύνολα τέτοια ώστε κάθε ακμή στον γράφο να καταλήγει σε κάθε ένα από τα δύο αυτά σύνολα, δηλαδή κάθε στοιχείο του ενός συνόλου συνδέεται με κάποιο στοιχείο του άλλου, όμως δύο στοιχεία του ίδιου συνόλου δεν συνδέονται.

Σε ένα κατευθυνόμενο γράφο μπορούμε να εκφράσουμε την έννοια του μονοπατιού, που είναι μία σειρά από κορυφές του γράφου που οδηγούν η μία στην άλλη. Στον διμερή γράφο που χρησιμοποιείται για την εφαρμογή GAC σε έναν alldifferent περιορισμό, κορυφές θεωρούνται οι μεταβλητές (X) και οι τιμές που μπορούν να πάρουν (V), οπότε θα παρουσιάζουμε ένα μονοπάτι ως μία σειρά μεταβλητών

κατευθυνόμενων στις τιμές τους και αντίστροφα, τιμών που οδηγούν στις μεταβλητές που τις περιέχουν (αφού μεταβλητές δεν ενώνονται με μεταβλητές και αντίστοιχα τιμές δεν ενώνονται με τιμές) πχ. $X0 \rightarrow V_0 \rightarrow X2 \rightarrow V_1 \rightarrow X0$.

Ο αλγόριθμος του Régin χρησιμοποιεί επίσης δύο ακόμη έννοιες που ορίζονται σε ένα γράφο, αυτές των Ισχυρά Συνεκτικών Συνιστωσών και του Max Matching.

Ένας κατευθυνόμενος γράφος είναι Ασθενώς Συνεκτικός (weakly connected), εάν υπάρχει ένα μονοπάτι μεταξύ κάθε δύο κορυφών εφόσον αγνοήσουμε τις κατευθύνσεις του γράφου. Ενώ ένας γράφος είναι Ισχυρά Συνεκτικός (strongly connected), εάν υπάρχει ένα μονοπάτι από κάθε κορυφή του γράφου σε κάθε άλλη λαμβάνοντας υπόψιν τις κατευθύνσεις του γράφου [**40**].

Ισχυρά Συνεκτικές Συνιστώσες (Strongly Connected Components - SCC) είναι τα μεγιστοτικά υπογραφήματα του γράφου που είναι ισχυρά συνεκτικά.

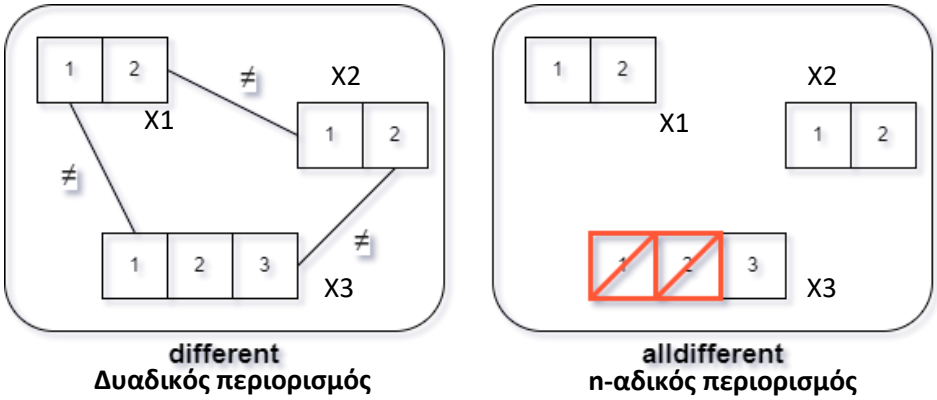
Matching σε ένα γράφο είναι ένα σύνολο ζευγών μη γειτονικών ακμών με καμία από τις δύο ακμές να μοιράζονται κοινές κορυφές. Ένα max matching σε ένα γράφο, είναι ένα matching που περιέχει τον μέγιστο αριθμό ακμών. Μία ακμή που ανήκει σε κάθε max matching ονομάζεται vital.

Ένας κύκλος αναφέρεται σαν ένα κλειστό μονοπάτι που ξεκινά και τελειώνει στην ίδια κορυφή ή στην δικιά μας περίπτωση στην ίδια μεταβλητή.

2.4.3 Περιορισμός diff έναντι alldifferent

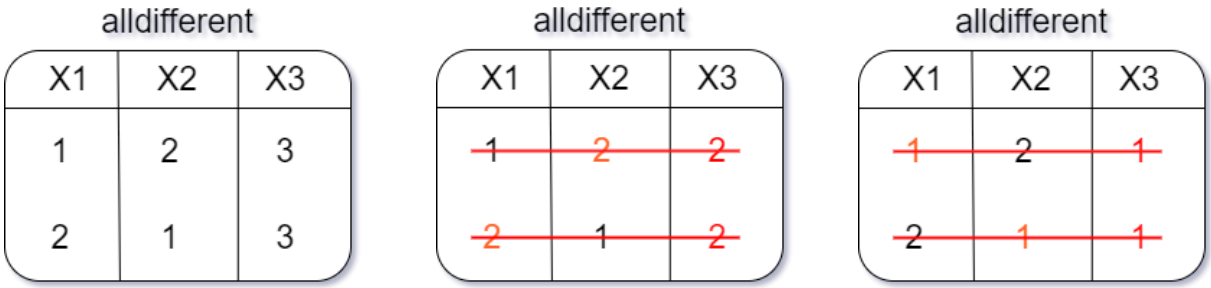
Όπως αναφέρθηκε ο μοναδικός περιορισμός που χρησιμοποιήθηκε είναι αυτός του alldifferent για κάθε γραμμή, στήλη και πλέγμα sudoku, διότι μειώνει κατά μεγάλο βαθμό το πλήθος των περιορισμών. Όμως αυτή δεν είναι η μοναδική χρησιμότητα του alldifferent περιορισμού. Γι' αυτό τον λόγο θα περιγράψουμε αυτόν, καθώς και τον απλό diff περιορισμό και θα δώσουμε ένα παράδειγμα της χρησιμότητας του, συγκρίνοντας τους.

Ο περιορισμός diff εξασφαλίζει ότι οι μεταβλητές στην εμβέλεια του περιορισμού θα πάρουν διαφορετικές τιμές. Ο περιορισμός εφαρμόζεται ανά δύο μεταβλητές την φορά, έτσι για τρείς μεταβλητές X1, X2, X3 θα είχαμε τρείς diff περιορισμούς diff(X1X2), diff(X1X3), diff(X2X3). Ο alldifferent περιορισμός αντίθετα, θα εφαρμοστεί και στις τρείς μεταβλητές (alldifferent(X1X2X3)).



Σχήμα 3. Diff και alldifferent περιορισμός για X1, X2, X3 μεταβλητές

Έχοντας έτσι πεδία ορισμού $D_3 = \{1, 2, 3\}$, $D_1 = D_2 = \{1, 2\}$ όπως φαίνεται στο (Σχ. 3), ο αλγόριθμος που προσπαθεί να εφαρμόσει arc consistency θα ελέγξει κάθε φορά εάν για κάθε τιμή στο domain της μίας μεταβλητής υπάρχει τουλάχιστον μία τιμή στο domain της άλλης που την υποστηρίζει. Στην περίπτωση που συμπεράνει ότι δεν υπάρχει, τότε θα την διαγράψει από το domain της. Κοιτώντας για παράδειγμα τις μεταβλητές X1, X2 από την πλευρά X1, για το 1 υπάρχει η τιμή 2 και για το 2 υπάρχει η τιμή 1, αντίστοιχα ισχύει το ίδιο και από την πλευρά του X2. Έτσι εάν συνεχίσουμε τη διαδικασία δεν θα υπάρξει διαγραφή τιμής. Αντιθέτως, έχοντας έναν alldifferent περιορισμό και εφαρμόζοντας την ιδιότητα GAC, θα υπάρξουν οι διαγραφές 1 και 2 από το πεδίο ορισμού της X3. Αυτό συμβαίνει διότι οι δυνατές πλειάδες του alldifferent περιορισμού είναι δύο (Σχ. 4). Η {1, 2, 3} για την τιμή 1 της X1 και {2, 1, 3} για την τιμή 2, αντίστοιχα {1, 2, 3} για την τιμή 2 της X2 και {2, 1, 3} για την τιμή 1. Δεν υπάρχει καμία δυνατή πλειάδα όμως για τις τιμές 1, 2 της μεταβλητής X3 και έτσι οι τιμές 1, 2 πρέπει να διαγραφούν από το πεδίο ορισμού της. Συνεπώς βλέπουμε πως ο alldifferent περιορισμός με εφαρμογή της GAC ιδιότητας διέγραψε 2 τιμές, ενώ η AC στους απλούς diff περιορισμούς καμία.



Σχήμα 4. Δυνατές και μη δυνατές πλειάδες του alldifferent(X1, X2, X3) περιορισμού

2.4.4 Singleton Arc Consistency

Η ιδιότητα Singleton Arc Consistency (SAC) [**29**] ελέγχει εάν η κατάσταση που θα προέλθει από την επιλογή μίας τιμής στο πεδίο ορισμού κάποιας μεταβλητής είναι arc consistent. Αυτό το πετυχαίνει επιλέγοντας μία τιμή σε μία μεταβλητή την φορά και εκτελώντας διάδοση περιορισμών εφαρμόζοντας AC στο υποπρόβλημα που δημιουργήθηκε. Εάν το πρόβλημα οδηγήσει σε arc inconsistent κατάσταση τότε η τιμή μπορεί να αφαιρεθεί μιας και δεν ανήκει σε κάποια λύση [**31**]. Η διαδικασία επαναλαμβάνεται, επαναφέροντας το πρόβλημα στην αρχική του κατάσταση με την αφαίρεση της τιμής και συνεχίζοντας την διαδικασία για τις υπόλοιπες μεταβλητές του προβλήματος. Πιο επίσημα [**29**]:

Μία τιμή ενός πεδίο ορισμού $\alpha \in D_i$ σε μία κατάσταση ενός CSP είναι singleton arc consistent εάν η κατάσταση που θα επέλθει το CSP από την αφαίρεση όλων των τιμών $\beta \in D_i$, με $\alpha \neq \beta$ μπορεί να γίνει arc consistent χωρίς να μείνει κανένα πεδίο ορισμού

άδειο. Μια κατάσταση ενός CSP είναι singleton arc consistent εάν κάθε τιμή στο πεδίο ορισμού των μεταβλητών είναι singleton arc consistent.

2.4.5 Singleton Generalized Arc Consistency

Η ιδιότητα Singleton Generalized Arc Consistency (SGAC) είναι μία επέκταση τής SAC με την αλλαγή ότι η διαδικασία διάδοσης των περιορισμών επιβάλλει GAC αντί για AC. Η SGAC έτσι, διαβεβαιώνει ότι η κατάσταση που θα προέλθει από την επιλογή μίας τιμής για μία μεταβλητή, για κάθε τιμή σε κάθε πεδίο ορισμού των μεταβλητών θα οδηγήσει σε generalized arc consistent κατάσταση. Δημιουργώντας ένα νέο στιγμιότυπο του προβλήματος με κάθε ανάθεση κάποιας τιμής, ο SGAC αλγόριθμος θα μπορούσε να εκφραστεί (βάσει του 2.4.4) ως:

Ένα CSP πρόβλημα n-αδικών περιορισμών είναι singleton generalized arc consistent εάν δεν έχει άδεια πεδία ορισμού και για οποιαδήποτε ανάθεση τιμής σε κάποια μεταβλητή, το νέο υπό-πρόβλημα είναι generalized arc consistent [**36**].

Κεφάλαιο 3

Αλγόριθμοι για GAC και SGAC

Ο κώδικας που αναπτύχθηκε υλοποιεί δύο αλγορίθμους για την εφαρμογή των ιδιοτήτων GAC και SGAC σε γλώσσα προγραμματισμού C. Παρακάτω παρατίθενται οι συναρτήσεις και οι δομές που απαρτίζουν τους δύο αυτούς αλγόριθμους καθώς και η περιγραφή των βημάτων που ακολουθούν. Το sudoku υλοποιήθηκε ως ένας πίνακας 81 δομών-struct που αντιπροσωπεύουν τις μεταβλητές sudoku. Κάθε struct περιέχει τη γραμμή, τη στήλη και το πλέγμα που βρίσκεται η αντίστοιχη μεταβλητή καθώς και το domain αλλά και την τιμή που έχει (εάν έχει, διαφορετικά ορίζεται ως -1). Το domain των μεταβλητών υλοποιήθηκε ως ένας πίνακας 9 θέσεων που σε κάθε θέση έχει 1 ή -1 ανάλογα με το εάν είναι διαθέσιμη ή έχει διαγραφεί η αντίστοιχη τιμή από το πεδίο ορισμού της συγκεκριμένης μεταβλητής.

3.1 GAC

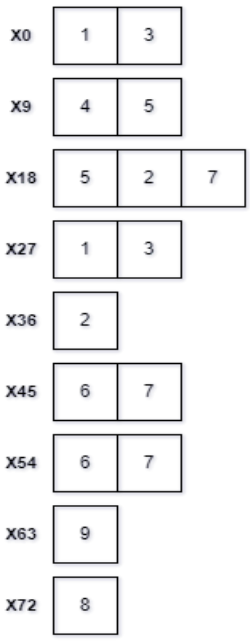
Όπως αναφέραμε έχει ήδη προταθεί ένας αλγόριθμος για την εφαρμογή της GAC σε ένα alldifferent περιορισμό από τον Régin. Στόχος του αλγορίθμου αυτού είναι να διαγράψει τις τιμές που δεν έχουν υποστήριξη στον alldifferent περιορισμό. Αυτό το επιτυγχάνει δημιουργώντας τον γράφο τιμών, δηλαδή ένα διμερή γράφο όπου οι κορυφές του αντιπροσωπεύουν τις μεταβλητές και τις τιμές που βρίσκονται στα πεδία ορισμού των μεταβλητών. Στον κώδικα που υλοποιεί τον αλγόριθμο αυτό, δημιουργήθηκαν 3 πίνακες (ένας για κάθε alldifferent περιορισμό, δηλαδή στηλών, γραμμών και πλεγμάτων), με κάθε ένα από αυτούς να περιέχει 9 γράφους τιμών. Ο αλγόριθμος έτσι προσπαθεί να εντοπίσει ακμές στον γράφο τιμών που δεν είναι συνεπείς.

Για την καλύτερη κατανόηση του αλγόριθμου προτού προχωρήσουμε στην ανάλυση της υλοποίησής μας θα δώσουμε ένα απλό παράδειγμα εκτέλεσης του.

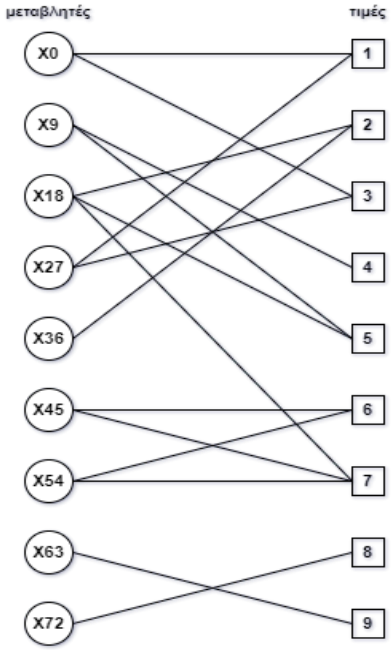
Έχοντας ένα alldifferent περιορισμό 9 μεταβλητών με 9 πεδία ορισμού όπως μίας στήλης sudoku μπορούμε να έχουμε ένα παράδειγμα όπως αυτό του (Σχ. 4). Στο παράδειγμα αυτό, έχουν ήδη διαγραφεί κάποιες τιμές από τα πεδία ορισμού της πρώτης στήλης sudoku.

Η διαδικασία που ακολουθεί ο αλγόριθμος που εφαρμόζει GAC στον alldiferent περιορισμό είναι η εξής:

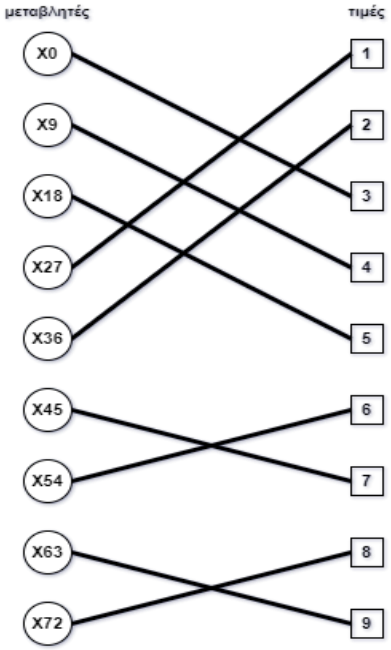
1) Θα δημιουργήσει ένα γράφο τιμών όπως στο (Σχ. 5), οπού κάθε μεταβλητή (αριστερά) θα αντιστοιχιστεί με τις τιμές που βρίσκονται στο πεδίο ορισμού της (δεξιά).
2) Θα βρει ένα max matching (υπό-ενότητα 2.4.2) του γράφου τιμών, δηλαδή κάθε μεταβλητή (αριστερά) πρέπει να αντιστοιχιστεί με μία ακριβώς τιμή (δεξιά) που βρίσκεται στο πεδίο ορισμού της και δεν έχει ήδη αντιστοιχιστεί με άλλη μεταβλητή, όπως φαίνεται στο (Σχ. 6). Εάν το max matching δεν καλύπτει όλες τις μεταβλητές τότε ο περιορισμός χαρακτηρίζεται ως inconsistent.
3) Θα δημιουργεί ένα διμερή κατευθυνόμενο γράφο όπου οι ακμές (edges) στο max matching κατευθύνονται από τις μεταβλητές στις τιμές που βρέθηκαν (από αριστερά προς τα δεξιά), ενώ όλες οι υπόλοιπες ακμές κατευθύνονται από τις τιμές στις μεταβλητές (από δεξιά στα αριστερά) όπως φαίνεται στο (Σχ. 7).



Σχήμα 5. Στήλη 1 sudoku



Σχήμα 6. Γράφος τιμών για τον περιορισμό alldifferent της στήλης 1 sudoku



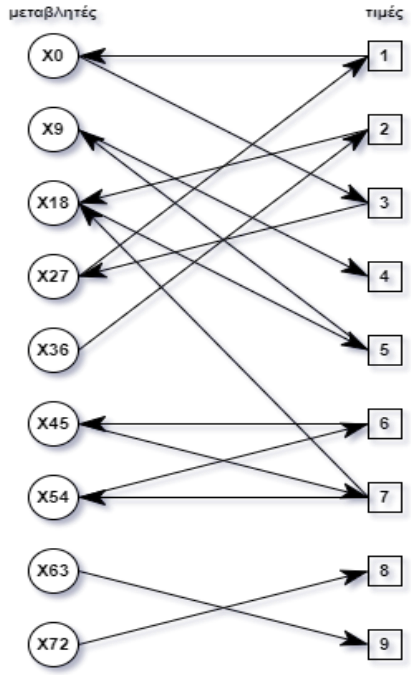
Σχήμα 7. Ένα max matching της στήλης 1 sudoku

4) Θα εντοπίσει με βάση το max matching που βρέθηκε και χρησιμοποιώντας τον κατευθυνόμενο γράφο:
   - Τις Ισχυρά Συνεκτικές Συνιστώσες (Strongly Connected Components-SCC) όπως αναφέραμε στην υπό-ενότητα (2.4.2). Για παράδειγμα $X_0 \rightarrow V_3 \rightarrow X_{27} \rightarrow V_1 \rightarrow X_0$ όπως φαίνεται στο (Σχ. 8) με πορτοκαλί χρώμα. Όλες οι τιμές που αντιστοιχούν σε κορυφές ακμών και βρέθηκαν στα SCC πρέπει να παραμείνουν στο domain των αντίστοιχων μεταβλητών τους.
   - Όλες τις ακμές (edges) που βρίσκονται στο max matching, αλλά δεν είναι στα SCC και που πρέπει να παραμείνουν οι τιμές τους στις αντίστοιχες

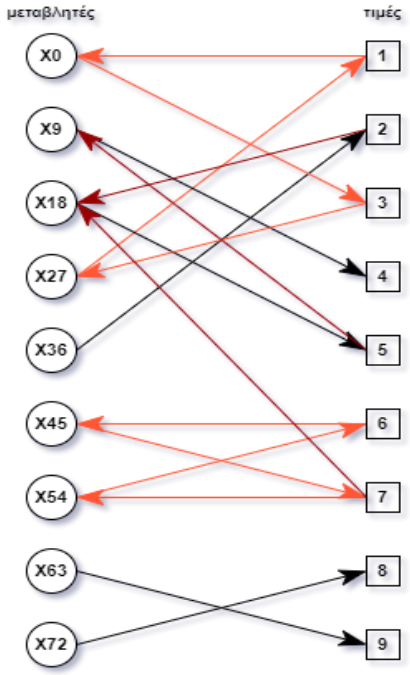
μεταβλητές. Οι ακμές αυτές λέγονται vital (υπό-ενότητα 2.4.2) και παρουσιάζονται με μαύρο χρώμα στο (Σχ. 8).

- Όλες τις ακμές (edges) που δεν ανήκουν σε καμία από τις παραπάνω κατηγορίες και πρέπει να διαγραφούν, όπως φαίνεται στο (Σχ. 8) με κόκκινο χρώμα.

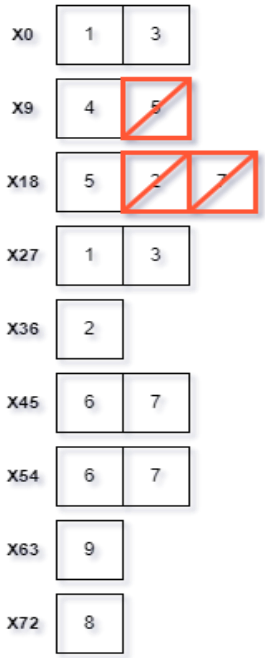
Τα πεδία ορισμού της στήλης 1 του sudoku μετά την εφαρμογή του αλγόριθμου του Régin φαίνονται στο (Σχ. 9). Έπειτα το πέρας της διαδικασίας αυτής οι αλλαγές πρέπει να διαδοθούν στους υπόλοιπους γράφους που περιέχουν τις μεταβλητές στις οποίες έγιναν διαγραφές.



Σχήμα 8. Διμερής κατευθυνόμενος γράφος



Σχήμα 9. SCC, vital και edges που πρέπει να διαγραφούν



Σχήμα 10. Domains έπειτα την εφαρμογή alldifferent περιορισμού

Ο αλγόριθμος για την εφαρμογή του GAC αποτελείται από τις εξής συναρτήσεις μέλη:

1. **void createGraphRCG(**sudoku,ValueRowGraph, ValueColumGraph, ValueGridGraph**)**
2. **bool matchingMax(**ValueGraph, index, maxMatching)
3. **int constraintPropagation(**sudoku, ValueRowGraph, ValueColumGraph, ValueGridGraph, RemovedEdgesList, listToRowVisit, listColToVisit, listGridToVisit**)**
4. **int removeEdges(**sudoku, maxMatching, index, Graph, RemovedEdgesList,, values_removed**)**

Όλες οι συναρτήσεις καλούνται εντός της συνάρτησης:

bool GAC(sudoku, ValueRowGraph, ValueColumGraph, ValueGridGraph, assigned_values, removed_values, SGAC**)**.

Η διαδικασία που ακολουθεί ο αλγόριθμος περιγράφεται πιο αναλυτικά στις παρακάτω υπό-ενότητες.

3.1.1 Δημιουργία γράφου τιμών για κάθε περιορισμό alldifferent

Όπως αναφέραμε στην υπό-ενότητα (3.1) ο αλγόριθμος ξεκινά δημιουργώντας ένα γράφο τιμών για κάθε περιορισμό alldifferent. Η υλοποίηση του γράφου αυτού έγινε σε μορφή πίνακα 9x9x10 για όλες τις γραμμές, στήλες πλέγματα sudoku που έχουν 9 μεταβλητές με 9 δυνατές τιμές στα domains τους. Η τελευταία θέση κάθε γραμμής του πίνακα να είναι η αντίστοιχη μεταβλητή sudoku.

Η διαδικασία αυτή επιτυγχάνεται με την συνάρτηση **void createGraphRCG()** που δέχεται σαν όρισμα τον πίνακα sudoku και τρείς πίνακες (έναν για τις στήλες, έναν για τις γραμμές και έναν για τα πλέγματα) που περιέχουν 9 άδειους γράφους τιμών ο καθένας προκειμένου να τους αρχικοποιήσει βάσει του sudoku.

3.1.2 Δημιουργία ουράς περιορισμών

Ο αλγόριθμος που εφαρμόζει την ιδιότητα GAC χρησιμοποιεί μια ουρά επίσκεψης στην οποία αποθηκεύονται οι περιορισμοί που θα επισκεφτεί σε κάθε πέρασμα του. Με κάθε επίσκεψη αφαιρείται ο εκάστοτε περιορισμός από την ουρά, ενώ κατά την εκτέλεση του μπορεί να υπάρξουν εισαγωγές περιορισμών στην ουρά. Οι alldifferent περιορισμοί επιβάλλονται σε κάθε γραμμή, στήλη και πλέγμα. Έτσι δημιουργούμε 3 διαφορετικές ουρές με την μορφή διπλά συνδεδεμένης λίστας και εισάγουμε τις τιμές 1-9 για κάθε γραμμή, στήλη και πλέγμα sudoku. Έτσι εάν αφαιρέσουμε την τιμή 2 από την ουρά γραμμής, θα αναφερόμαστε στον περιορισμό που επιβάλλεται στην γραμμή 2. Αντίστοιχα εάν αφαιρούσαμε την τιμή αυτή από τις ουρές στηλών, πλεγμάτων θα αναφερόμασταν στους περιορισμούς που επιβάλλονται στην στήλη και πλέγμα 2.

3.1.3 Επαναληπτική διαδικασία αλγόριθμου

Ο αλγόριθμος GAC έπειτα ακολουθεί μια επαναληπτική διαδικασία οπού για τον γράφο που αναφέρεται στον περιορισμό που εξάγεται κάθε φορά από την ουρά επίσκεψης:

1. Βρίσκουμε ένα max matching
2. Βρίσκουμε τις ακμές του γράφου που πρέπει να διαγραφούν (δηλαδή ακμές που συνδέουν μεταβλητές με τιμές). Η διαγραφή μιας ακμής από τον γράφο αντιστοιχεί σε διαγραφή μιας τιμής από το domain μιας μεταβλητής, το οποίο είναι και το ζητούμενο για το GAC
3. Διαδίδουμε τις αλλαγές στους άλλους περιορισμούς (γράφους), δηλαδή διατρέχουμε τους γράφους που περιέχουν τις ακμές αυτές, αφαιρούμε τις

αντίστοιχες τιμές από τα domain των μεταβλητών τους και εισάγουμε τους περιορισμούς που περιέχουν τις ακμές αυτές (όλους εκτός αυτού που εξετάζουμε) στην αντίστοιχη ουρά επίσκεψης της υπό-ενότητας (3.1.2)

4. Εάν βρεθούν οι ακμές που πρέπει να διαγραφούν, επισκεπτόμαστε τους περιορισμούς που εφαρμόζονται στις μεταβλητές των ακμών

Κάθε φορά, προτού βγάλουμε έναν περιορισμό για να τον εξετάσουμε από την εκάστοτε ουρά επίσκεψης, ελέγχουμε πρώτα εάν η αντίστοιχη ουρά που περιέχει τον περιορισμό είναι άδεια.

Η εύρεση του max matching επιτυγχάνεται με την συνάρτηση **bool matchingMax(**ValueGraph, index, maxMatching). Η συνάρτηση δέχεται ως παραμέτρους, ένα γράφο τιμών, για ποια στήλη, γραμμή ή πλέγμα του sudoku θέλουμε να βρούμε το max matching (index) καθώς και ένα πίνακα που επιστρέφει το max matching. Ο πίνακας είναι μεγέθους 2x9 διότι η $2^{η}$ γραμμή περιέχει τις μεταβλητές και η $1^{η}$ γραμμή τις αντίστοιχες τιμές τους.

Η διαδικασία που ακολουθεί η συνάρτηση είναι :

1. Να δημιουργήσει μία λίστα με δυνατές επιλογές τιμής κάθε μεταβλητής του γράφου (πίνακας με λίστες) καθώς και ένα αντίγραφο του γράφου για να μην προκαλέσει αλλαγές στον κύριο γράφο (Συνάρτηση 1: Σειρά 2, 3).
2. Σε μία επαναληπτική διαδικασία:
   a. Εάν η λίστα με τις δυνατές τιμές της μεταβλητής δεν είναι άδεια, επιλέγει την πρώτη δυνατή της τιμή αφαιρώντας την από την λίστα (Συνάρτηση 1: Σειρά 7). Εάν η τιμή αυτή έχει ανατεθεί σε άλλη μεταβλητή, επαναλαμβάνουμε επιλέγοντας την αμέσως επόμενη δυνατή. Εάν η ανάθεση αυτή δεν έχει γίνει και σε άλλη μεταβλητή, επιλέγει την τιμή ως matching της μεταβλητής, αλλάζει τον γράφο (1 μόνο για την τιμή που επιλέχθηκε και 0 για τις άλλες) και συνεχίζει την διαδικασία για την επόμενη μεταβλητή του γράφου.
   b. Εάν όμως η λίστα με τις δυνατές τιμές της μεταβλητής είναι άδεια:
      a. Εάν είναι η πρώτη μεταβλητή, τότε επιστρέφει false
      b. Εάν δεν είναι η πρώτη, επιστρέφει τον γράφο αυτής και της προηγούμενης μεταβλητής στην προηγούμενη κατάσταση (αντιγράφει στον γράφο-αντίγραφο τον κύριο) και την λίστα δυνατών συνδυασμών της προηγούμενης μεταβλητής στην προηγούμενη κατάσταση της και συνεχίζει την διαδικασία για την προηγούμενη μεταβλητή

Σε αυτό το σημείο πρέπει να αναφέρουμε πως αυτή η υλοποίηση εκτελεί DFS (κατά βάθος αναζήτηση) με χρονική πολυπλοκότητα O(bm) για την εύρεση του max matching, οπού b ο παράγοντας διακλάδωσης (9 τιμές) και m το μέγιστο βάθος του δέντρου αναζήτησης (9 μεταβλητές). Ο αλγόριθμος του Régin με πολυπλοκότητα $O(s^2d^2)$ χρησιμοποιεί τον αλγόριθμο Hopcroft–Karp με πολυπλοκότητα $O(|E|\sqrt{|V|})$ και γι' αυτό επιτυγχάνει αυτήν την πολυπλοκότητα. Η διαφορά της χρονικής πολυπλοκότητας ανάμεσα στις δύο εκδοχές του αλγόριθμου όμως δεν αναμένεται να είναι μεγάλη, μιας και στο πρόβλημα sudoku υπάρχουν μόλις 9 μεταβλητές με 9 δυνατές τιμές η κάθε μία, σε μεγαλύτερα προβλήματα όμως, η κύρια πιο αποδοτική έκδοση του αλγορίθμου θα ήταν προτιμότερη.

Συνάρτηση 1 : Εύρεση Max matching

```
1:  matchingMax (ValueGraph, index, maxMatching) {
2:      queue[9] ← Δυνατές τιμές των μεταβλητών του γράφου;
3:      GraphCopy ← ValueGraph;
4:      i = 0;
5:      ΕΝΟΣΩ οι μεταβλητές του γράφου δεν έχουν μοναδικές τιμές
6:          ΕΑΝ queue[i] δεν είναι άδεια
7:              ChoosenValue ← queue[i];
8:              ΓΙΑ κάθε μεταβλητή στον Γράφο
9:                  ΕΑΝ GraphCopy[μεταβλητή][ChoosenValue] == 1
10:                     Επανέλαβε την διαδικασία από την σειρά 7 (επέλεξε άλλη τιμή για την ίδια μεταβλητή)
11:             GraphCopy[i] ← 1 μόνο στην επιλεγμένη τιμή, 0 στις υπόλοιπες;
12:             maxMatching ← ChoosenValue, i (μεταβλητή);
13:             i++;
14:         ΕΙΔΑΛΩΣ
15:             ΕΑΝ η μεταβλητή είναι η αρχική
16:                 return false;
17:             ΕΙΔΑΛΩΣ
18:                 GraphCopy[i] ← ValueGraph[index][i];
19:                 GraphCopy[i - 1] ← ValueGraph[index][i – 1];
20:                 Επανέφερε το queue[i] στην προηγούμενη κατάσταση;
21:                 i = i – 1;
22: return true;
23: }
```

Αφού βρει το max matching, το χρησιμοποιεί για να βρει τις ακμές που μπορούν να διαγραφούν από τον γράφο. Η διαδικασία αυτή γίνεται με την χρήση της συνάρτησης **int removeEdges(**sudoku, maxMatching, index, Graph, RemovedEdgesList**)**. Η συνάρτηση δέχεται ως παραμέτρους, τον πίνακα sudoku, το max matching που βρέθηκε, τον περιορισμό και τον γράφο για τον οποίο βρέθηκε το max matching καθώς και μία λίστα όπου θα εισάγει τις ακμές που πρέπει να διαγραφούν από το γράφο για να τοποθετηθούν στο RemovedEdgesList. Η συνάρτηση επιστέφει επίσης και τον αριθμό των ακμών/τιμών που διαγράφθηκαν για τον υπολογισμό στατιστικών. Στον κώδικα συμβολίζουμε τα SCC με την τιμή 2 ενώ τα vital στοιχεία με 3.

Η διαδικασία που ακολουθεί η συνάρτηση είναι:

1. Δημιουργία του κατευθυνόμενου γράφου από τον γράφο τιμών και το max matching (Συνάρτηση 2: Σειρά 2)
2. Εύρεση των SCC από το max matching (υπό-ενότητα 2.4.2) με την συνάρτηση **computeSCCs(**DirectedGraph, SCC**)** χρησιμοποιώντας τον αλγόριθμο kosaraju [**30**] (Συνάρτηση 3). Η συνάρτηση λαμβάνει ως ορίσματα, τον κατευθυνόμενο γράφο και ένα πίνακα λιστών SCC. Αρχικά εκτελεί DFS στον γράφο κρατώντας τις επισκεπτόμενες ακμές σε μία στοίβα και έπειτα εκτελεί για κάθε μία από αυτές τις ακμές ξανά DFS στον αντίστροφο γράφο παίρνοντας τελικά τα SCC
3. Αρχικοποίηση πίνακα λίστας για τις ακμές RemovedEdgesList **(RE)** (Τιμές από τις μεταβλητές που πρέπει να διαγραφούν) βάσει του αριθμού των SCC (Συνάρτηση 2: Σειρά 4)

4. Για κάθε ένα από τα SCC (Συνάρτηση 2: Σειρές 5 - 17):
    b. Διατρέχει τον γράφο και όταν βρει την μεταβλητή που αντιστοιχεί στην ακμή του SCC, βρίσκει τις αντίστοιχες τιμές της και εάν η μεταβλητή του sudoku δεν έχει ήδη τιμή (ελέγχει εάν το Value του πίνακα sudoku είναι -1) και είναι διαθέσιμη, τότε αλλάζει την τιμή του γράφου σε 2
    c. Στο ίδιο πέρασμα ελέγχει εάν οι μεταβλητές είναι στο max matching και όχι στα SCC και για τις επιλεγμένες τιμές, εάν είναι διαθέσιμες αλλάζει την τιμή του γράφου σε 3
    d. Διαγράφει τις τιμές που δεν είναι 2 ή 3 και είναι διαθέσιμες και τις εισάγει στο RE
5. Εκτέλεση άλλου ενός περάσματος (Συνάρτηση 2: Σειρές 18 - 21) για τις υπόλοιπες μεταβλητές του γράφου, διότι πρέπει να αφαιρεθούν από τις διαθέσιμες τιμές και να προστεθούν στο RE οι υπόλοιπες μεταβλητές και τιμές τους που δεν είναι στα SCC και δεν έχουν τιμή 2 ή 3
6. Αλλαγή των τιμών του γράφου από 2 και 3 σε 1 για να βρίσκονται σε σωστή μορφή για την επόμενη συνάρτηση

Συνάρτηση 2 : Εύρεση των ακμών που πρέπει να διαγραφούν

```
1:  int removeEdges (sudoku, maxMatching, index, Graph, RemovedEdgesList) {
2:      DirectedGraph ← Graph, maxMatching
3:      SCC ← computeSCCs(DirectedGraph, SCC)
4:      RemovedEdgesList[SCC.size]
5:      ΓΙΑ κάθε μεταβλητή στο SCC
6:          ΓΙΑ κάθε τιμή της SCC μεταβλητής
7:              ΓΙΑ κάθε μεταβλητή, τιμή του Γράφου τιμών που έχουν την μεταβλητή SCC
8:                  ΕΑΝ είναι στα SCC ΚΑΙ sudoku[μεταβλητή].Value < 0 ΚΑΙ Graph[index][μεταβλητή][τιμή] != -1
9:                      Graph[index][μεταβλητή][τιμή] = 2;
10:                 ΕΙΔΑΛΩΣ ΕΑΝ είναι στο maxMatching ΚΑΙ Graph[index][μεταβλητή][τιμή] != -1
11:                     Graph[index][μεταβλητή][τιμή] = 3;
12:             ΓΙΑ κάθε μεταβλητή στο SCC
13:                 ΓΙΑ κάθε τιμή της SCC μεταβλητής
14:                     ΓΙΑ κάθε μεταβλητή, τιμή του Γράφου τιμών που έχουν την μεταβλητή SCC
15:                         ΕΑΝ Graph[index][μεταβλητή][τιμή] != 3 ΚΑΙ Graph[index][μεταβλητή][τιμή] != 2 ΚΑΙ
                            Graph[index][μεταβλητή][τιμή] ==1
16:                             Graph[index][μεταβλητή][τιμή] = -1;
17:                             RemovedEdgesList ← μεταβλητή, τιμή;
18:     ΓΙΑ κάθε μεταβλητή, τιμή του Γράφου τιμών
19:         ΕΑΝ Graph[index][μεταβλητή][τιμή] δεν είναι στα SCC ΚΑΙ Graph[index][μεταβλητή][τιμή] != 3 ΚΑΙ
            Graph[index][μεταβλητή][τιμή] != 2
20:             Graph[index][μεταβλητή][τιμή] = -1;
21:             RemovedEdgesList ← μεταβλητή, τιμή;
22:     Graph ← Άλλαξε τις τιμές 2, 3 με 1
23: }
```

Συνάρτηση 3 : Εύρεση Strongly connected components

```
1:  void computeSCCs (graph, SCC) {
2:      Δημιουργία stack
3:      stack ← Vertices εκτελώντας DFS στον graph;
4:      Tgraph ← Αντίστροφος graph;
5:      ΕΝΟΣΩ δεν είναι άδεια η stack
6:          V ← pop(stack);
7:          SCC ← στοιχεία εκτελώντας DFS στον Tgraph για το V, εφόσον
            δεν έχουμε επισκεφθεί το συγκεκριμένο V;
8:  }
```

Μετά την αποθήκευση και διαγραφή των ακμών που πρέπει να διαγραφούν, ακολουθεί η διαδικασία διάδοσης των περιορισμών. Η διαδικασία αυτή γίνεται με την χρήση της συνάρτησης: **int constraintPropagation(**sudoku, ValueRowGraph, ValueColumGraph, ValueGridGraph, RemovedEdgesList, listToRowVisit, listColToVisit, listGridToVisit, current_constraint, RowColGrid**)**. Για κάθε μία από τις ακμές του RemovedEdgesList, αποθηκεύει τους περιορισμούς γραμμών, στηλών και πλέγματατων sudoku που περιέχουν αυτές τις ακμές στις αντίστοιχες λίστες επίσκεψης τους. Αυτό γίνεται αφού εξεταστεί εάν ο εκάστοτε περιορισμός είναι ο περιορισμός που εξετάζεται στην συγκεκριμένη επανάληψη του αλγορίθμου, σε αυτήν την περίπτωση δεν πρέπει να γίνει προσθήκη στην εκάστοτε λίστα επίσκεψης.

Εάν υπάρχει ήδη ένας από αυτούς στις αντίστοιχες λίστες, τους μετακινεί στην αρχή της λίστας δίνοντας τους προτεραιότητα (Συνάρτηση 4: Σειρές 4-6) και έπειτα διαγράφει τις τιμές που έχουν διαγραφεί από αυτή την μεταβλητή και από τους άλλους γράφους.

Συνάρτηση 4 : Αλγόριθμος διάδοσης περιορισμών

```
1:  int constraintPropagation(sudoku, ValueRowGraph, ValueColumGraph, ValueGridGraph,
    RemovedEdgesList, listToRowVisit, listColToVisit, listGridToVisit) {
2:      ΓΙΑ κάθε ένα από τα edges του RemovedEdgesList
3:          variable, values ← RemovedEdgesList
4:          ValueColumGraph ← colum(variable) (εάν δεν είναι ο περιορισμός alldifferent
                                                που εξετάζουμε)
5:          ValueRowGraph ← row(variable) (εάν δεν είναι ο περιορισμός alldifferent
                                            που εξετάζουμε)
6:          ValueGridGraph ← grid(variable) (εάν δεν είναι ο περιορισμός alldifferent
                                             που εξετάζουμε)
7:          ΓΙΑ κάθε γραμμή, στήλη και πλέγμα sudoku
8:              ValueColumGraph[στήλη] ← ValueColumGraph με αφαίρεση των values της
                αντίστοιχης variable
9:              ValueRowGraph[γραμμή] ← ValueRowGraph με αφαίρεση των values της
                αντίστοιχης variable
10:             ValueGridGraph[πλέγμα] ← ValueGridGraph με αφαίρεση των values της
                αντίστοιχης variable
11: }
```

Η ένωση όλων αυτών των διαδικασιών δίνεται στη (Συνάρτηση 5). Ο GAC αλγόριθμος αρχικοποιεί τους γράφους που περιέχονται στους τρείς πίνακες γράφων με την συνάρτηση **createGraphRCG()** και δημιουργεί μια ουρά περιορισμών (Συνάρτηση 5: Σειρά 2, 3). Έπειτα επαναλαμβάνει την διαδικασία της υπό-ενότητας (3.1.3) επαναληπτικά για κάθε γραμμή, στήλη και πλέγμα sudoku. Δηλαδή, όσο οι ουρές περιορισμών δεν είναι άδειες, έχοντας ως παράδειγμα τον alldiferrent περιορισμό στήλης (Συνάρτηση 5: Σειρά 5), θα αφαιρέσει ένα περιορισμό στήλης από την ουρά περιορισμών και για αυτόν τον περιορισμό θα βρει ένα maxMatching με την συνάρτηση **matchingMax**() (Συνάρτηση 5: Σειρές 6,7). Εάν δεν βρεθεί ένα maxMatching ο αλγόριθμος επιστρέφει false, ειδάλλως θα βρει τις ακμές που πρέπει να διαγραφούν με την συνάρτηση **removeEdges()** και θα εκτελέσει διάδοση περιορισμών με την συνάρτηση **constraintPropagation()** (Συνάρτηση 5: Σειρές 8, 9). Εάν βρεθούν ακμές που πρέπει να διαγραφούν, η διαδικασία επαναλαμβάνεται και για τους άλλους δύο περιορισμούς που περιέχουν τις ακμές αυτές (γραμμές και πλέγματα) (Συνάρτηση 5: Σειρές 10, 11). Η διαδικασία που ακολουθήσαμε για τον alldifferent περιορισμό στηλών (Συνάρτηση 5: Σειρές 5-12) θα επαναληφθεί αντίστοιχα και για τους άλλους δύο περιορισμούς (γραμμές και πλέγματα) (Συνάρτηση 5: Σειρές 13-16).

Ο αλγόριθμος έτσι τερματίζει με true εάν δεν αποτύχει η εύρεση ενός maxMatching κάποιου περιορισμού alldiferent και αδειάσουν όλες οι ουρές επίσκεψης περιορισμών ή false εάν αποτύχει.

Συνάρτηση 5 : Αλγόριθμος Generalized arc consistency

```
1:  bool GAC (sudoku, RowGraph, ColumGraph, GridGraph) {
2:      createGraphRCG(sudoku, RowGraph, ColumGraph, GridGraph);
3:      VisitGridList, VisitColList, VisitGridList ← 1-9 γραμμές, στήλες, πλέγματα;
4:      ΕΝΟΣΩ οι VisitGridList, VisitColList, VisitGridList > 0
5:          ΕΑΝ VisitColList > 0
6:              index ← pop(VisitColList);
7:                  ΕΑΝ  matchingMax(ColumGraph, index, maxMatching) == true
8:                      removeEdges (sudoku, maxMatching, index, ColumGraph, RemovedEdgesList);
9:                      changes = constraintPropagation(sudoku, ColumGraph, RowGraph, GridGraph,
                        RemovedEdgesList, listRowToVisit, listColToVisit, listGridToVisit);
10:                     ΕΑΝ changes == 1
11:                         Επανέλαβε την διαδικασία των σειρών 5-9 για τους άλλους δύο alldifferent
                            περιορισμούς (γραμμές, πλέγματα)
12:                 ΕΙΔΑΛΩΣ return false;
13:         ΕΑΝ VisitRowList > 0
14:             Επανέλαβε την διαδικασία των σειρών 5-12 για τον alldifferent περιορισμό γραμμών
15:         ΕΑΝ VisitGridList > 0
16:             Επανέλαβε την διαδικασία των σειρών 5-12 για τον alldifferent περιορισμό πλεγμάτων
17:     return true;
18: }
```

3.2 SGAC

Ο αλγόριθμος που εφαρμόζει την ιδιότητα SGAC όπως αναφέρθηκε στην υπό-ενότητα (2.4.5) είναι μία επέκταση του GAC αλγορίθμου. Συγκεκριμένα, ο αλγόριθμος αυτός χρησιμοποιεί/καλεί τον αλγόριθμο GAC σε μια επαναληπτική διαδικασία.

Η διαδικασία που ακολουθεί ο SGAC αλγόριθμος είναι:

1. Εκτελεί GAC και εάν αποτύχει, ο SGAC αλγόριθμος επιστρέφει false

Συνάρτηση 6 : Αλγόριθμος Singleton generalized arc consistency

```
1:  bool SGAC (sudoku, RowGraph, ColumGraph, GridGraph) {
2:      ΕΑΝ GAC (sudoku, RowGraph, ColumGraph, GridGraph) == false
3:          return false;
4:      changes = false;
5:      ΕΝΟΣΩ changes == true
6:          changes = false;
7:          ΓΙΑ κάθε μεταβλητή του προβλήματος (που δεν έχει ήδη τιμή), επέλεξε μία βάση του βρόγχου
8:              ΓΙΑ κάθε τιμή της μεταβλητής, επέλεξε μία βάση του βρόγχου
9:                  ΓΙΑ κάθε γραμμή, στήλη, πλέγμα των γράφων περιορισμών που έχουν αυτή την μεταβλητή
10:                     RowGraph[γραμμή][μεταβλητή][τιμή] == 1;
11:                     Ανανέωσε το domain της συγκεκριμένης μεταβλητής αυτού του γράφου
12:                     ColumGraph[στήλη][μεταβλητή][τιμή] == 1;
13:                     Ανανέωσε το domain της συγκεκριμένης μεταβλητής αυτού του γράφου
14:                     GridGraph[πλέγμα][μεταβλητή][τιμή] == 1;
15:                     Ανανέωσε το domain της συγκεκριμένης μεταβλητής αυτού του γράφου
16:                 ΕΑΝ GAC (sudoku, RowGraph, ColumGraph, GridGraph) == false
17:                     RowGraph, ColumGraph, GridGraph ← Προηγούμενη κατάσταση των Γράφων πριν τις αλλαγές
18:                     RowGraph, ColumGraph, GridGraph ← Διαγραφή της τιμής που επιλέχθηκε
19:                     changes = true;
20:                 ΕΙΔΑΛΛΩΣ
21:                     RowGraph, ColumGraph, GridGraph ← Προηγούμενη κατάσταση των Γράφων πριν τις αλλαγές
22:     GAC (sudoku, RowGraph, ColumGraph, GridGraph)
23:     return true;
24: }
```

2. Επιλέγει μια μεταβλητή του προβλήματος sudoku που δεν έχει ήδη τιμή

    a. Επιλέγει μία τιμή που μπορεί να πάρει και την ορίζει προσωρινά ως την τιμή της μεταβλητής (Συνάρτηση 6: Σειρές 8-15)

    b. Εφαρμόζει GAC στο νέο στιγμιότυπο του προβλήματος και εάν αποτύχει, αναιρεί την αλλαγή, επαναφέρει τους γράφους στην προηγούμενη κατάσταση τους, διαγράφει αυτήν την τιμή από τα domains της μεταβλητής και επαναλαμβάνει το a για την επόμενη δυνατή τιμή της μεταβλητής (Συνάρτηση 6: Σειρές 16-19). Εάν δεν

αποτύχει, συνεχίζει την διαδικασία αναιρώντας την αλλαγή και επαναφέροντας τους γράφους στην προηγούμενη κατάσταση τους επαναλαμβάνοντας το βήμα 2 έως ότου να έχει κάνει ένα πέρασμα σε όλες τις μεταβλητές (Συνάρτηση 6: Σειρές 20-21)

3. Εάν πραγματοποιήθηκαν διαγραφές τιμών στο βήμα 2, επαναλαμβάνει το βήμα 2 από την αρχή, αλλιώς ο αλγόριθμος συνεχίζει στο επόμενο βήμα

4. Εφόσον έχει τελειώσει ο βρόγχος επανάληψης while (Συνάρτηση 6: Σειρές 5-21) και έχουν διαγραφεί τιμές από τα domain των μεταβλητών εκτελείτε για άλλη μια φορά ο GAC αλγόριθμος για το αρχικό πρόβλημα

Κεφάλαιο 4

Πειράματα

Σκοπός των πειραμάτων που πραγματοποιήθηκαν είναι η σύγκριση της απόδοσης των δύο αλγορίθμων στην επίλυση προβλημάτων sudoku διαφορετικών επιπέδων δυσκολίας για ανθρώπινους λύτες. Τα πειράματα που πραγματοποιήθηκαν έγιναν έτσι σε sudoku βαθμών δυσκολίας easy, medium, hard και fiendish. Ο βαθμός αυτός εκφράζεται συνήθως από τον δημιουργό του προβλήματος και συσχετίζεται με τον αριθμό δοθέντων κελιών sudoku αλλά και την τοποθέτηση τους στο πρόβλημα. Οι πηγές που χρησιμοποιήθηκαν ήταν τα βιβλία “Jumbo Sudoku” του Michael Mepham [**32**], “Advanced Sudoku” των Puzzler Media Ltd [**33**], “New York Post Sudoku 3” του Wayne Gould [**34**] καθώς και η σελίδα websudoku.com που παρέχει μια πληθώρα από προβλήματα κατηγορίας easy, medium, hard [**35**]. Όλες οι πηγές αναγράφονται ανά στιγμιότυπο προβλήματος στην στήλη 1 των πινάκων.

Σε κάθε κατηγορία χρησιμοποιήθηκαν 41 δείγματα προβλημάτων sudoku και καταγράφθηκαν για την εκτέλεση των αλγόριθμων που εφαρμόζουν GAC και SGAC:

1. Ο χρόνος εκτέλεσης των δύο αλγορίθμων (στήλη 5 του πεδίου GAC και SGAC αντίστοιχα).
2. Το ποσοστό ολοκλήρωσης του sudoku (στήλη 4 του πεδίου GAC και SGAC αντίστοιχα).
3. Τα κελιά που έχουν δοθεί στον χρήστη και τα κελιά που πήραν τιμή, δηλαδή βρέθηκαν από τους δύο αλγόριθμους (στήλη 2, 3 του πεδίου GAC και SGAC αντίστοιχα). Εάν το άθροισμα των δύο είναι ίσο με 81, τότε το πρόβλημα έχει επιλυθεί.
4. Οι τιμές που διαγράφθηκαν από τα domains των μεταβλητών (στήλη 1 του πεδίου GAC και SGAC αντίστοιχα).

4.1 Easy στιγμιότυπα sudoku

Στον Πίνακα 1 δίνονται αναλυτικά τα αποτελέσματα από τα easy προβλήματα sudoku. Από τα αποτελέσματα των μετρήσεων βλέπουμε πως ο GAC αλγόριθμος είχε ποσοστό ολοκλήρωσης 100% όπως και ο SGAC. Η διαφορά στο χρόνο εκτέλεσης των δύο αλγορίθμων (1.92×10^{-2} s για τον GAC και 2.19×10^{-2} s για τον SGAC) μπορεί να αποδοθεί στην διαδικασία που ακολουθεί ο SGAC αλγόριθμος. Ακόμη και εάν βρεθεί λύση από τον GAC, ο SGAC αλγόριθμος θα συνεχίσει ελέγχοντας όλες τις δυνατές αναθέσεις τιμών σε όλες τις μεταβλητές (81 μεταβλητές με 9 τιμές η κάθε μία), οι μεταβλητές θα έχουν ήδη πάρει τιμή από την πρώτη εκτέλεση GAC και έτσι μιας και ελέγχεται εάν μια μεταβλητή έχει τιμή, ο SGAC αλγόριθμος θα εκτελέσει βρόγχο 81

επαναλήψεων. Από τα αποτελέσματα αυτά μπορούμε να συμπεράνουμε ότι και οι δύο αλγόριθμοι αποτελούν ένα πολύ αποδοτικό τρόπο επίλυσης easy επιπέδου sudoku, με προτιμότερο τον GAC αλγόριθμο εάν ενδιαφερόμαστε για τον χρόνο εκτέλεσης.

Πηγή	GAC					SGAC				
	Τιμές που διαγράφθηκαν	Κελιά sudoku		% ολοκλήρωσης	Χρόνος Εκτέλεσης (s)	Τιμές που διαγράφθηκαν	Κελιά sudoku		% ολοκλήρωσης	Χρόνος Εκτέλεσης (s)
		Δόθηκαν	Πήραν τιμή				Δόθηκαν	Πήραν τιμή		
[32]	424	28	53	100%	4.30E-02	424	28	53	100%	4.60E-02
[32]	424	28	53	100%	2.70E-02	424	28	53	100%	3.10E-02
[32]	408	30	51	100%	2.30E-02	408	30	51	100%	2.50E-02
[32]	424	28	53	100%	1.80E-02	424	28	53	100%	2.00E-02
[32]	424	28	53	100%	1.90E-02	424	28	53	100%	2.10E-02
[32]	424	28	53	100%	1.10E-02	424	28	53	100%	1.30E-02
[32]	424	28	53	100%	2.60E-02	424	28	53	100%	2.90E-02
[32]	424	28	53	100%	5.90E-02	424	28	53	100%	6.20E-02
[32]	424	28	53	100%	1.80E-02	424	28	53	100%	2.00E-02
[32]	440	26	55	100%	1.20E-02	440	26	55	100%	1.50E-02
[32]	424	28	53	100%	1.20E-02	424	28	53	100%	1.50E-02
[32]	416	29	52	100%	1.20E-02	416	29	52	100%	1.60E-02
[32]	424	28	53	100%	3.70E-02	424	28	53	100%	4.00E-02
[35]	424	28	53	100%	4.10E-02	424	28	53	100%	4.40E-02
[35]	368	35	46	100%	1.10E-02	368	35	46	100%	1.30E-02
[35]	376	34	47	100%	1.40E-02	376	34	47	100%	1.60E-02
[35]	360	36	45	100%	8.00E-03	360	36	45	100%	1.20E-02
[35]	360	36	45	100%	2.20E-02	360	36	45	100%	2.60E-02
[35]	368	35	46	100%	4.20E-02	368	35	46	100%	4.40E-02
[35]	360	36	45	100%	1.00E-02	360	36	45	100%	1.30E-02
[35]	392	32	49	100%	1.10E-02	392	32	49	100%	1.30E-02
[35]	360	36	45	100%	8.00E-03	360	36	45	100%	1.00E-02
[35]	368	35	46	100%	1.10E-02	368	35	46	100%	1.40E-02
[35]	368	35	46	100%	1.90E-02	368	35	46	100%	2.10E-02
[35]	384	33	48	100%	2.20E-02	384	33	48	100%	2.40E-02
[35]	360	36	45	100%	1.40E-02	360	36	45	100%	1.60E-02
[35]	360	36	45	100%	1.40E-02	360	36	45	100%	1.50E-02
[35]	392	32	49	100%	1.10E-02	392	32	49	100%	1.30E-02
[35]	376	34	47	100%	8.00E-03	376	34	47	100%	1.00E-02
[35]	392	32	49	100%	8.00E-03	392	32	49	100%	1.00E-02
[35]	384	33	48	100%	2.10E-02	384	33	48	100%	2.40E-02
[35]	360	36	45	100%	1.20E-02	360	36	45	100%	1.60E-02
[35]	392	32	49	100%	2.30E-02	392	32	49	100%	2.60E-02
[35]	360	36	45	100%	1.10E-02	360	36	45	100%	1.50E-02
[35]	376	34	47	100%	2.20E-02	376	34	47	100%	2.50E-02
[35]	360	36	45	100%	1.10E-02	360	36	45	100%	1.30E-02
[35]	360	36	45	100%	2.60E-02	360	36	45	100%	2.90E-02
[35]	376	34	47	100%	1.40E-02	376	34	47	100%	1.70E-02

[35]	368	35	46	100%	1.30E-02	368	35	46	100%	1.50E-02
[35]	360	36	45	100%	2.40E-02	360	36	45	100%	2.80E-02
[35]	368	35	46	100%	2.10E-02	368	35	46	100%	2.30E-02
M.O	388.68	32.41	48.58	100%	1.92E-02	388.68	32.41	48.58	100%	2.19E-02

Πίνακας 1. Μετρήσεις σε στιγμιότυπα sudoku βαθμού δυσκολίας easy

4.2 Medium στιγμιότυπα sudoku

Στον Πίνακα 2 δίνονται αναλυτικά τα αποτελέσματα από τα medium προβλήματα sudoku. Στα medium προβλήματα έχουμε μία αύξηση του χρόνου εκτέλεσης των αλγόριθμων GAC (από 1.92×10^{-2} s σε $3,06 \times 10^{-2}$ s) και SGAC (από 2.19×10^{-2} s σε $3,34 \times 10^{-2}$ s) έχοντας όμως σταθερό ποσοστό ολοκλήρωσης 100% σε όλα τα στιγμιότυπα medium sudoku. Οι δύο αλγόριθμοι μπορούν έτσι να θεωρηθούν ως ένα αξιόπιστο μέσο επίλυσης και medium προβλημάτων sudoku.

Πηγή	GAC					SGAC				
	Τιμές που διαγράφθηκαν	Κελιά sudoku		% ολοκλήρωσης	Χρόνος Εκτέλεσης (s)	Τιμές που διαγράφθηκαν	Κελιά sudoku		% ολοκλήρωσης	Χρόνος Εκτέλεσης (s)
		Δόθηκαν	Πήραν τιμή				Δόθηκαν	Πήραν τιμή		
[32]	424	28	53	100%	4.00E-02	424	28	53	100%	4.20E-02
[32]	424	28	53	100%	1.60E-02	424	28	53	100%	1.80E-02
[32]	440	26	55	100%	3.80E-02	440	26	55	100%	4.00E-02
[32]	424	28	53	100%	3.10E-02	424	28	53	100%	3.50E-02
[32]	424	28	53	100%	7.30E-02	424	28	53	100%	7.90E-02
[32]	416	29	52	100%	3.40E-02	416	29	52	100%	3.60E-02
[32]	408	30	51	100%	2.00E-02	408	30	51	100%	2.20E-02
[35]	416	29	52	100%	1.50E-02	416	29	52	100%	1.70E-02
[35]	392	32	49	100%	8.00E-03	392	32	49	100%	1.00E-02
[35]	400	31	50	100%	1.50E-02	400	31	50	100%	1.70E-02
[35]	416	29	52	100%	1.20E-02	416	29	52	100%	1.40E-02
[35]	400	31	50	100%	1.10E-02	400	31	50	100%	1.50E-02
[35]	416	29	52	100%	4.00E-02	416	29	52	100%	4.40E-02
[35]	416	29	52	100%	3.00E-02	416	29	52	100%	3.40E-02
[35]	424	28	53	100%	3.60E-02	424	28	53	100%	3.90E-02
[35]	408	30	51	100%	1.80E-02	408	30	51	100%	2.00E-02
[35]	416	29	52	100%	1.10E-02	416	29	52	100%	1.30E-02
[35]	424	28	53	100%	2.10E-02	424	28	53	100%	2.30E-02
[33]	400	31	50	100%	1.10E-02	400	31	50	100%	1.40E-02
[33]	424	28	53	100%	1.50E-02	424	28	53	100%	1.70E-02
[33]	424	28	53	100%	2.10E-02	424	28	53	100%	2.40E-02
[33]	480	21	60	100%	2.70E-02	480	21	60	100%	3.10E-02
[33]	456	24	57	100%	4.30E-02	456	24	57	100%	4.50E-02
[33]	480	21	60	100%	4.40E-02	480	21	60	100%	4.80E-02
[33]	432	27	54	100%	3.40E-02	432	27	54	100%	3.70E-02
[33]	456	24	57	100%	4.50E-02	456	24	57	100%	5.00E-02
[33]	440	26	55	100%	4.20E-02	440	26	55	100%	4.60E-02
[33]	456	24	57	100%	3.00E-02	456	24	57	100%	3.20E-02
[33]	456	24	57	100%	3.30E-02	456	24	57	100%	3.70E-02

[33]	448	25	56	100%	4.00E-02	448	25	56	100%	4.20E-02
[33]	472	22	59	100%	3.30E-02	472	22	59	100%	3.60E-02
[33]	472	22	59	100%	8.30E-02	472	22	59	100%	8.70E-02
[33]	440	26	55	100%	4.40E-02	440	26	55	100%	4.60E-02
[33]	408	30	51	100%	3.30E-02	408	30	51	100%	3.60E-02
[33]	456	24	57	100%	2.90E-02	456	24	57	100%	3.20E-02
[33]	432	27	54	100%	2.10E-02	432	27	54	100%	2.30E-02
[33]	440	26	55	100%	1.60E-02	440	26	55	100%	1.80E-02
[33]	472	22	59	100%	5.90E-02	472	22	59	100%	6.10E-02
[33]	464	23	58	100%	3.40E-02	464	23	58	100%	3.80E-02
[33]	456	24	57	100%	3.70E-02	456	24	57	100%	3.90E-02
[33]	424	28	53	100%	1.10E-02	424	28	53	100%	1.20E-02
M.O	433.56	26.80	54.19	100%	3.06E-02	433.56	26.80	54.19	100%	3.34E-02

Πίνακας 2. Μετρήσεις σε στιγμιότυπα sudoku βαθμού δυσκολίας medium

4.3 Hard στιγμιότυπα sudoku

Στον Πίνακα 3 δίνονται αναλυτικά τα αποτελέσματα από τα hard προβλήματα sudoku. Στα hard προβλήματα παρατηρούμε την πρώτη διαφορά στο ποσοστό επίλυσης των αλγορίθμων GAC και SGAC. Ο SGAC συνεχίζει να έχει ποσοστό ολοκλήρωσης 100% σε όλα τα hard προβλήματα, ενώ ο GAC πέφτει σε μέσο ποσοστό ολοκλήρωσης 98%. Η μείωση αυτή προκλήθηκε από 2 / 41 στιγμιότυπα προβλημάτων και συγκεκριμένα τα:

- 19 οπού βρήκε σύνολο 19 + 27 = 46 από τις 81 τιμές των μεταβλητών
- 25 οπού βρήκε σύνολο 23 + 28 = 51 από τις 81 τιμές των μεταβλητών

Η συμπλήρωση του ποσοστού επίλυσης του GAC αλγόριθμου σε 100% όμως δεν έρχεται χωρίς κόστος, ο SGAC αλγόριθμος δέχεται επιβάρυση στον χρόνο εκτέλεσης από 4.42×10^{-2} s για τον GAC σε 6.61×10^{-2} s για τον SGAC. Επειδή όλα τα προβλήματα προέρχονται από την ίδια πηγή και έχουν τον ίδιο βαθμό δυσκολίας, μπορούμε να πούμε λόγου του μικρού δείγματος αποτυχίας εύρεσης λύσης ότι ο GAC αλγόριθμος αποτελεί έναν αρκετά καλό τρόπο εύρεσης λύσης hard sudoku προβλημάτων. Για εγγυημένη εύρεση λύσης όμως, καλό θα ήταν να χρησιμοποιηθεί ο GAC αλγόριθμος ως προ-επεξεργασία σε μία μέθοδο επίλυσης με αναζήτηση ή ο SGAC αλγόριθμος χωρίς αναζήτηση.

Πηγή	GAC					SGAC				
	Τιμές που διαγράφθηκαν	Κελιά sudoku		% ολοκλήρωσης	Χρόνος Εκτέλεσης (s)	Τιμές που διαγράφθηκαν	Κελιά sudoku		% ολοκλήρωσης	Χρόνος Εκτέλεσης (s)
		Δόθηκαν	Πήραν τιμή				Δόθηκαν	Πήραν τιμή		
[35]	432	27	54	100%	5.60E-02	432	27	54	100%	5.90E-02
[35]	424	28	53	100%	5.30E-02	424	28	53	100%	5.60E-02
[35]	424	28	53	100%	3.20E-02	424	28	53	100%	3.50E-02
[35]	440	26	55	100%	2.40E-02	440	26	55	100%	2.60E-02
[35]	440	26	55	100%	2.00E-02	440	26	55	100%	2.10E-02
[35]	432	27	54	100%	1.30E-02	432	27	54	100%	1.50E-02
[35]	424	28	53	100%	4.90E-02	424	28	53	100%	5.30E-02
[35]	440	26	55	100%	5.60E-02	440	26	55	100%	6.00E-02

[35]	432	27	54	100%	2.50E-02	432	27	54	100%	2.80E-02
[35]	424	28	53	100%	3.20E-02	424	28	53	100%	3.40E-02
[35]	432	27	54	100%	3.30E-02	432	27	54	100%	3.60E-02
[35]	432	27	54	100%	2.90E-02	432	27	54	100%	3.30E-02
[35]	432	27	54	100%	3.20E-02	432	27	54	100%	3.40E-02
[35]	432	27	54	100%	2.40E-02	432	27	54	100%	2.60E-02
[35]	432	27	54	100%	2.60E-02	432	27	54	100%	2.80E-02
[35]	440	26	55	100%	1.20E-02	440	26	55	100%	1.30E-02
[35]	440	26	55	100%	2.20E-02	440	26	55	100%	2.40E-02
[35]	424	28	53	100%	1.50E-02	424	28	53	100%	1.80E-02
[35]	382	27	19	57%	2.10E-02	432	27	54	100%	3.84E-01
[32]	424	28	53	100%	3.20E-02	424	28	53	100%	3.40E-02
[32]	424	28	53	100%	3.80E-02	424	28	53	100%	4.10E-02
[32]	440	26	55	100%	4.60E-02	440	26	55	100%	4.80E-02
[32]	424	28	53	100%	3.40E-02	424	28	53	100%	3.80E-02
[35]	440	26	55	100%	7.20E-02	440	26	55	100%	7.80E-02
[35]	373	28	23	63%	2.80E-02	424	28	53	100%	4.59E-01
[35]	432	27	54	100%	1.90E-02	432	27	54	100%	2.10E-02
[35]	424	28	53	100%	1.30E-02	424	28	53	100%	1.60E-02
[35]	424	28	53	100%	5.80E-02	424	28	53	100%	6.00E-02
[35]	440	26	55	100%	5.30E-02	440	26	55	100%	5.70E-02
[35]	432	27	54	100%	1.03E-01	432	27	54	100%	1.07E-01
[35]	424	28	53	100%	4.80E-02	424	28	53	100%	5.10E-02
[35]	440	26	55	100%	4.10E-02	440	26	55	100%	4.30E-02
[35]	440	26	55	100%	4.50E-02	440	26	55	100%	4.80E-02
[35]	432	27	54	100%	4.60E-02	432	27	54	100%	4.80E-02
[35]	424	28	53	100%	3.72E-01	424	28	53	100%	3.73E-01
[35]	432	27	54	100%	3.60E-02	432	27	54	100%	3.80E-02
[35]	432	27	54	100%	2.00E-02	432	27	54	100%	2.30E-02
[35]	432	27	54	100%	3.30E-02	432	27	54	100%	3.50E-02
[35]	432	27	54	100%	4.20E-02	432	27	54	100%	4.40E-02
[35]	440	26	55	100%	2.60E-02	440	26	55	100%	2.90E-02
[35]	440	26	55	100%	3.20E-02	440	26	55	100%	3.50E-02
M.O	429.34	27.02	52.39	98%	4.42E-02	431.80	27.02	53.97	100%	6.61E-02

Πίνακας 3. Μετρήσεις σε στιγμιότυπα sudoku βαθμού δυσκολίας hard

4.4 Fiendish στιγμιότυπα sudoku

Στον Πίνακα 4 δίνονται αναλυτικά τα αποτελέσματα από τα fiendish προβλήματα sudoku. Τα fiendish προβλήματα θεωρούνται πιο δύσκολα από τα hard προβλήματα sudoku βάσει του New York Post. Αυτό επαληθεύεται, μιας και ο αλγόριθμος που εφαρμόζει GAC μειώνει ακόμη περισσότερο το ποσοστό επιτυχίας του από μέσο όρο 98% σε 95%. Τα στιγμιότυπα κάτω του 90% είναι 4 / 41 σε σχέση με τα 2 / 41 στα hard, με μία μείωση στον χρόνο εκτέλεσης ($4{,}42 \times 10^{-2}$ s σε 3.15×10^{-2} s). Ο SGAC και πάλι έχει ποσοστό ολοκλήρωσης 100%, με την μεγάλη διαφορά στον χρόνο

εκτέλεσης των στιγμιότυπων που οι τιμές που βρέθηκαν από τον GAC ήταν λίγες, όπως και στα στιγμιότυπα hard κάνοντας τον μέσο χρόνο εκτέλεσης του 1.17×10^{-1} s.

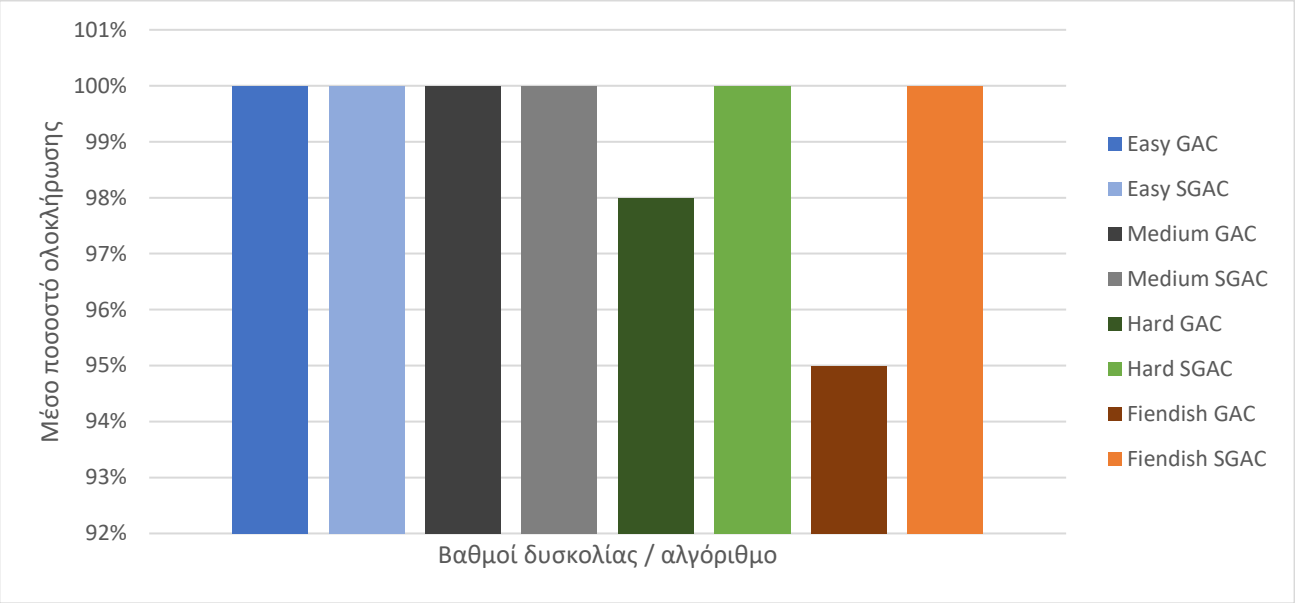
Θεωρητικά, τα αποτελέσματα είναι όπως αναμένονταν, μιας και ο SGAC αλγόριθμος είναι μία επέκταση του GAC αλγόριθμου. Ο SGAC αλγόριθμος έτσι, θα έπρεπε να είχε τόσο χρόνο εκτέλεσης όσο ο GAC, με διαφορά στις περιπτώσεις που δεν βρίσκει λύση ο GAC και έτσι πρέπει να αρχίσει την διαδικασία μείωσης των domains των μεταβλητών από τον SGAC αλγόριθμο, επιβαρύνοντας το χρόνο εκτέλεσης.

Πηγή	GAC					SGAC				
	Τιμές που διαγράφθηκαν	Κελιά sudoku		% ολοκλήρωσης	Χρόνος Εκτέλεσης (s)	Τιμές που διαγράφθηκαν	Κελιά sudoku		% ολοκλήρωσης	Χρόνος Εκτέλεσης (s)
		Δόθηκαν	Πήραν τιμή				Δόθηκαν	Πήραν τιμή		
[34]	472	22	59	100%	5.80E-02	472	22	59	100%	6.10E-02
[34]	456	24	57	100%	5.10E-02	456	24	57	100%	5.60E-02
[34]	397	24	25	60%	5.40E-02	456	24	57	100%	5.49E-01
[34]	456	24	57	100%	4.50E-02	456	24	57	100%	4.80E-02
[34]	464	23	58	100%	4.00E-02	464	23	58	100%	4.20E-02
[34]	472	22	59	100%	3.10E-02	472	22	59	100%	3.30E-02
[34]	456	24	57	100%	3.90E-02	456	24	57	100%	4.10E-02
[34]	362	24	11	43%	1.60E-02	456	24	57	100%	9.27E-01
[34]	456	24	57	100%	4.40E-02	456	24	57	100%	4.70E-02
[34]	440	26	55	100%	2.80E-02	440	26	55	100%	3.10E-02
[34]	440	26	55	100%	4.30E-02	440	26	55	100%	4.60E-02
[34]	464	23	58	100%	5.10E-02	464	23	58	100%	5.30E-02
[34]	464	23	58	100%	2.50E-02	464	23	58	100%	2.70E-02
[34]	448	25	56	100%	1.60E-02	448	25	56	100%	1.80E-02
[34]	363	24	10	42%	2.00E-02	456	24	57	100%	8.94E-01
[34]	456	24	57	100%	5.10E-02	456	24	57	100%	5.40E-02
[34]	464	23	58	100%	3.80E-02	464	23	58	100%	4.30E-02
[34]	456	24	57	100%	6.40E-02	456	24	57	100%	6.70E-02
[34]	324	25	6	38%	4.00E-02	448	25	56	100%	1.16E+00
[34]	456	24	57	100%	5.20E-02	456	24	57	100%	5.40E-02
[34]	440	26	55	100%	1.50E-02	440	26	55	100%	1.60E-02
[34]	456	24	57	100%	1.50E-02	456	24	57	100%	1.70E-02
[34]	456	24	57	100%	4.50E-02	456	24	57	100%	4.80E-02
[34]	456	24	57	100%	6.60E-02	456	24	57	100%	6.90E-02
[34]	424	28	53	100%	2.90E-02	424	28	53	100%	3.10E-02
[34]	424	28	53	100%	2.40E-02	424	28	53	100%	2.60E-02
[34]	416	29	52	100%	1.90E-02	416	29	52	100%	2.20E-02
[34]	392	32	49	100%	2.30E-02	392	32	49	100%	2.40E-02
[34]	408	30	51	100%	1.70E-02	408	30	51	100%	1.90E-02
[34]	424	28	53	100%	1.10E-02	424	28	53	100%	1.20E-02
[34]	392	32	49	100%	9.00E-03	392	32	49	100%	1.00E-02
[34]	408	30	51	100%	8.00E-03	408	30	51	100%	1.00E-02
[34]	392	32	49	100%	1.00E-02	392	32	49	100%	1.20E-02
[34]	424	28	53	100%	1.10E-02	424	28	53	100%	1.40E-02

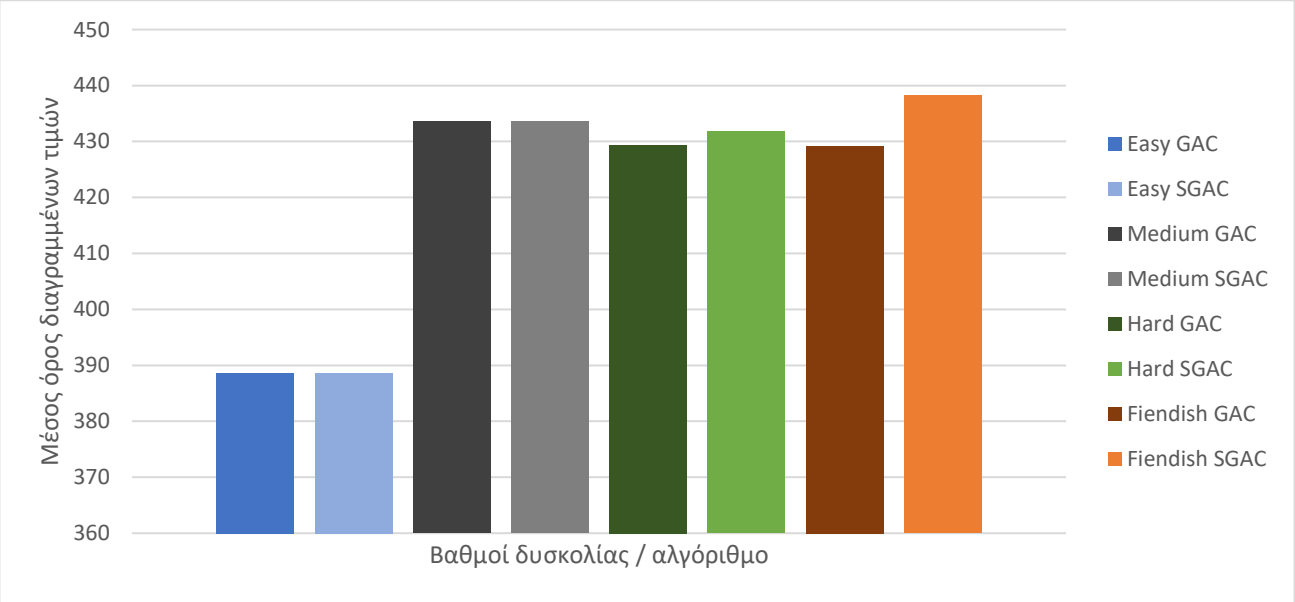
[34]	424	28	53	100%	2.30E-02	424	28	53	100%	2.60E-02
[34]	392	32	49	100%	2.00E-02	392	32	49	100%	2.30E-02
[34]	392	32	49	100%	4.20E-02	392	32	49	100%	4.50E-02
[34]	424	28	53	100%	3.60E-02	424	28	53	100%	3.90E-02
[34]	424	28	53	100%	1.90E-02	424	28	53	100%	2.20E-02
[34]	424	28	53	100%	2.50E-02	424	28	53	100%	2.70E-02
[34]	440	26	55	100%	1.80E-02	440	26	55	100%	2.00E-02
M.O	429.21	26.21	50.51	95%	3.15E-02	438.24	26.21	54.78	100%	1.17E-01

Πίνακας 4. Μετρήσεις σε στιγμιότυπα sudoku βαθμού δυσκολίας fiendish

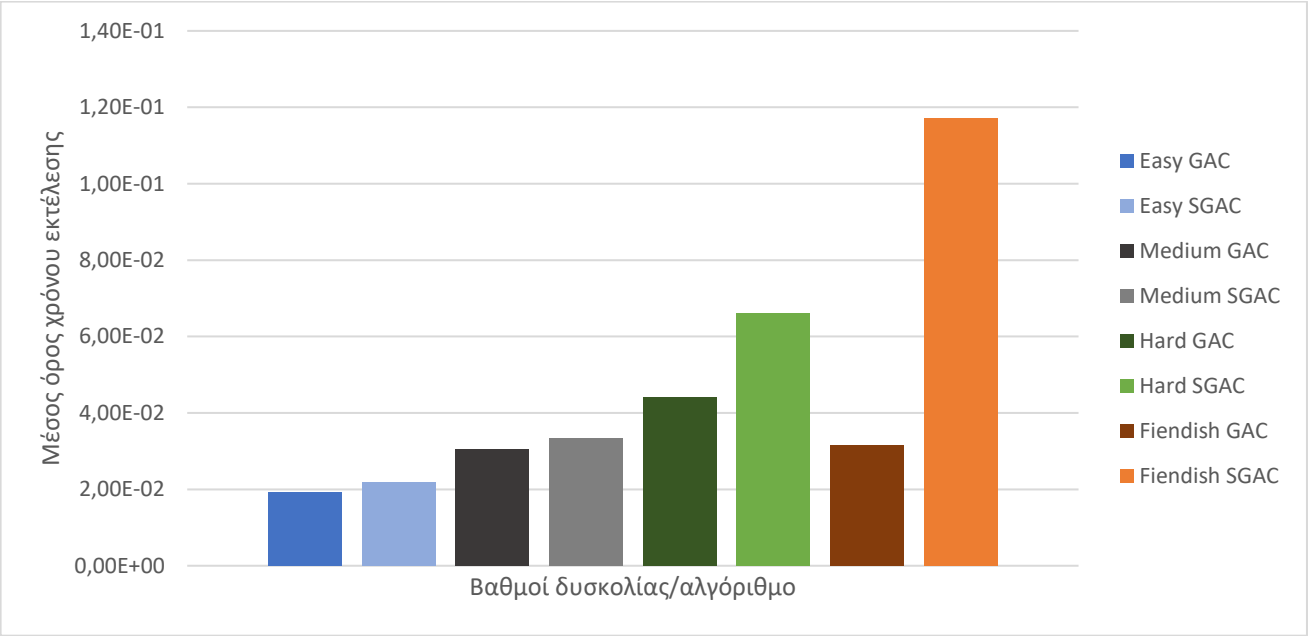
Παρακάτω δίνεται η γραφική απεικόνιση των αποτελεσμάτων του μέσου ποσοστού ολοκλήρωσης, χρόνου εκτέλεσης και διαγραμμένων τιμών από τα πεδία ορισμού των μεταβλητών όλων των πειραμάτων που έγιναν στους δύο αλγόριθμους. Στον x άξονα βρίσκονται οι αλγόριθμοι που χρησιμοποιήθηκαν στον αντίστοιχο βαθμό δυσκολίας και στον y τα αποτελέσματα των πειραμάτων ανά πείραμα.



Γράφημα 1. Μέσο ποσοστό ολοκλήρωσης πειραμάτων σε σχέση με αλγόριθμο που χρησιμοποιήθηκε ανά πείραμα



Γράφημα 2. Μέσος όρος διαγραμμένων τιμών των πειραμάτων σε σχέση με αλγόριθμο που χρησιμοποιήθηκε ανά πείραμα



Γράφημα 3. Μέσος όρος χρόνου εκτέλεσης των πειραμάτων σε σχέση με αλγόριθμο που χρησιμοποιήθηκε ανά πείραμα

4.5 Hardest known Sudoku puzzles

Ακολουθεί πείραμα που έγινε σε 375 στιγμιότυπα των πιο δύσκολων sudoku που παρέχουν οι I. Howell, R. Woodward, B. Choueiry, C. Bessiere [**38**]. Τα προβλήματα αυτά δεν έχουν ταξινομηθεί με βάση κάποια δυσκολία πχ. Hard, Very Hard κτλ. αλλά ως ένα σύνολο πολύ δύσκολων προβλημάτων sudoku. Στον πίνακα 5 δίνονται τα αποτελέσματα των πειραμάτων στα Hardest known Sudoku puzzles. Μπορούμε να δούμε και την διαφορά στην δυσκολία σε σχέση με τα προηγούμενα στιγμιότυπα που χρησιμοποιήθηκαν στα πειράματα μιας και οι δύο αλγόριθμοι έχουν ένα πολύ χαμηλό ποσοστό ολοκλήρωσης με μέσο ποσοστό επίλυσης 28%. Σε αυτό το ποσοστό πρέπει να συμπεριληφθεί το γεγονός ότι οι αλγόριθμοι βρήκαν το περισσότερο 4 επιπλέον τιμές σε αυτές που δόθηκαν από το πρόβλημα. Μοναδικό μέσο σύγκρισης έτσι είναι οι διαγεγραμμένες τιμές από τα domains των μεταβλητών που είναι 286.72 κατά μέσο όρο στον GAC αλγόριθμο και 289.20 στον SGAC αλγόριθμο. Οι χρόνοι εκτέλεσης είναι σχετικά μικροί στον GAC, μιας και δεν βρίσκει λύση, ενώ αρκετά μεγαλύτεροι στον SGAC (3.07 s έναντι του 3.47×10^{-2} s του GAC) μιας και εκτελείται πολλές φορές ο GAC αλγόριθμος για την εύρεση επιπλέον τιμών σε αυτές της απλής εκτέλεσης του GAC αλγόριθμου.

Φυσικά οι αλγόριθμοι αυτοί δεν μπορούν να δώσουν λύση σε τόσο δύσκολα προβλήματα, όμως μπορούν να χρησιμοποιηθούν ως προ-επεξεργασία σε μία μέθοδο επίλυσης με αναζήτηση, μιας και αφαιρούν ένα ικανοποιητικό ποσό τιμών από τα πεδία ορισμού των μεταβλητών.

Πηγή	GAC					SGAC				
	Τιμές που διαγράφθηκαν	Κελιά sudoku		% ολοκλήρωσης	Χρόνος Εκτέλεσης (s)	Τιμές που διαγράφθηκαν	Κελιά sudoku		% ολοκλήρωσης	Χρόνος Εκτέλεσης (s)
		Δόθηκαν	Πήραν τιμή				Δόθηκαν	Πήραν τιμή		
[37]	290	22	1	28%	3.80E-02	290	22	1	28%	2.256000042
[37]	288	22	1	28%	3.10E-02	289	22	1	28%	3.940000057
[37]	292	22	1	28%	3.40E-02	294	22	1	28%	4.102000237
[37]	280	22	0	27%	3.50E-02	281	22	0	27%	4.046999931
[37]	292	22	2	30%	2.00E-02	292	22	2	30%	1.896000028
[37]	292	22	2	30%	5.90E-02	292	22	2	30%	1.378000021

[37]	286	22	1	28%	1.70E-02	288	22	1	28%	3.940999985
[37]	296	21	1	27%	4.00E-02	303	21	1	27%	3.648999929
[37]	284	22	1	28%	2.20E-02	286	22	1	28%	3.686000109
[37]	294	21	0	26%	1.40E-02	298	21	0	26%	3.674999952
[37]	294	22	0	27%	4.80E-02	294	22	0	27%	2.124000072
[37]	289	22	1	28%	3.40E-02	289	22	1	28%	1.674000025
[37]	288	22	1	28%	2.90E-02	288	22	1	28%	1.955000043
[37]	292	21	1	27%	3.10E-02	297	21	1	27%	4.355000019
[37]	290	22	0	27%	2.80E-02	292	22	0	27%	3.822999954
[37]	289	22	1	28%	3.80E-02	289	22	1	28%	2.013000011
[37]	289	22	1	28%	3.90E-02	289	22	1	28%	1.61500001
[37]	281	21	0	26%	1.80E-02	281	21	0	26%	2.198999882
[37]	288	22	0	27%	4.80E-02	288	22	0	27%	2.183000088
[37]	292	21	2	28%	5.10E-02	293	21	2	28%	3.710000038
[37]	281	21	1	27%	5.20E-02	283	21	1	27%	4.205999851
[37]	292	21	1	27%	2.20E-02	294	21	1	27%	4.752999783
[37]	292	21	1	27%	4.80E-02	294	21	1	27%	5.482999802
[37]	283	22	1	28%	6.20E-02	285	22	1	28%	3.497999907
[37]	300	22	3	31%	2.40E-02	301	22	3	31%	3
[37]	286	22	1	28%	3.60E-02	286	22	1	28%	2.053999901
[37]	301	22	3	31%	7.60E-02	302	22	3	31%	3.111000061
[37]	301	22	3	31%	4.40E-02	302	22	3	31%	3.654000044
[37]	301	22	3	31%	2.00E-02	302	22	3	31%	3.391000032
[37]	287	22	1	28%	3.90E-02	287	22	1	28%	1.792999983
[37]	294	22	1	28%	4.90E-02	299	22	1	28%	4.27699995
[37]	305	22	0	27%	5.80E-02	305	22	0	27%	1.697999954
[37]	287	22	1	28%	5.50E-02	287	22	1	28%	1.888000011
[37]	303	22	0	27%	1.80E-02	303	22	0	27%	1.605999947
[37]	293	22	1	28%	2.00E-02	296	22	1	28%	2.943000078
[37]	304	22	1	28%	2.80E-02	304	22	1	28%	1.611999989
[37]	299	21	2	28%	2.10E-02	306	21	2	28%	4.737999916
[37]	279	22	0	27%	1.90E-02	306	22	1	28%	13.81700039
[37]	288	22	1	28%	3.60E-02	289	22	1	28%	3.671000004
[37]	293	21	1	27%	3.50E-02	293	21	1	27%	1.626999974
[37]	294	21	1	27%	2.50E-02	294	21	1	27%	1.789000034
[37]	289	22	1	28%	6.50E-02	289	22	1	28%	1.883999944
[37]	295	21	0	26%	1.60E-02	295	21	0	26%	1.902999997
[37]	297	22	1	28%	3.50E-02	299	22	1	28%	2.795000076
[37]	289	21	0	26%	1.70E-02	289	21	0	26%	1.934999943
[37]	282	21	1	27%	3.80E-02	282	21	1	27%	1.967000008
[37]	285	22	1	28%	1.80E-02	285	22	1	28%	1.967000008
[37]	298	21	0	26%	5.20E-02	298	21	0	26%	1.871999979
[37]	297	21	0	26%	4.20E-02	297	21	0	26%	1.84800005
[37]	287	22	0	27%	4.00E-02	287	22	0	27%	1.728000045
[37]	300	21	0	26%	7.60E-02	300	21	0	26%	1.787999988
[37]	301	21	0	26%	3.80E-02	301	21	0	26%	1.664999962
[37]	299	21	0	26%	6.30E-02	299	21	0	26%	1.628000021
[37]	311	20	0	25%	3.20E-02	311	20	0	25%	1.904999971

[37]	295	21	0	26%	3.00E-02	295	21	0	26%	1.657999992
[37]	296	21	0	26%	1.60E-02	296	21	0	26%	1.985999942
[37]	298	21	0	26%	5.10E-02	298	21	0	26%	1.820000052
[37]	295	21	0	26%	1.50E-02	295	21	0	26%	1.995000005
[37]	299	21	0	26%	2.10E-02	299	21	0	26%	1.83
[37]	295	21	0	26%	2.20E-02	295	21	0	26%	2.055999994
[37]	299	21	0	26%	2.70E-02	299	21	0	26%	1.621000051
[37]	286	21	0	26%	3.00E-02	295	21	0	26%	4.005000114
[37]	285	21	1	27%	6.10E-02	309	21	1	27%	7.58
[37]	308	21	4	31%	1.70E-02	308	21	4	31%	1.809999943
[37]	286	21	0	26%	3.40E-02	286	21	0	26%	1.98300004
[37]	293	21	0	26%	4.00E-02	293	21	0	26%	1.876000047
[37]	297	21	1	27%	4.20E-02	303	21	1	27%	3.540999889
[37]	296	21	0	26%	2.30E-02	296	21	0	26%	2.244999886
[37]	285	21	0	26%	4.60E-02	285	21	0	26%	2.204999924
[37]	288	21	0	26%	1.90E-02	292	21	0	26%	3.796000004
[37]	285	21	0	26%	4.60E-02	287	21	0	26%	4.140999794
[37]	288	21	0	26%	3.70E-02	288	21	0	26%	1.934999943
[37]	303	21	1	27%	2.20E-02	307	21	1	27%	3.40
[37]	309	21	1	27%	4.10E-02	309	21	1	27%	1.830000043
[37]	296	21	0	26%	1.84E-01	296	21	0	26%	2.318000078
[37]	294	21	0	26%	3.60E-02	298	21	0	26%	4.769999981
[37]	288	21	1	27%	6.10E-02	288	21	1	27%	2.387000084
[37]	283	21	0	26%	1.70E-02	287	21	0	26%	4.485000134
[37]	298	21	1	27%	5.20E-02	300	21	1	27%	3.921000004
[37]	294	21	0	26%	6.60E-02	295	21	0	26%	4.535999775
[37]	285	21	0	26%	5.10E-02	289	21	0	26%	4.474999905
[37]	284	21	0	26%	2.70E-02	289	21	0	26%	4.65899992
[37]	283	21	0	26%	3.80E-02	290	21	0	26%	4.705999851
[37]	313	20	0	25%	2.40E-02	313	20	0	25%	1.88
[37]	303	21	0	26%	3.70E-02	304	21	0	26%	3.644999981
[37]	305	21	2	28%	2.30E-02	305	21	2	28%	1.911000013
[37]	285	21	0	26%	2.10E-02	290	21	0	26%	4.30
[37]	285	21	0	26%	1.57E-01	285	21	0	26%	2.349999905
[37]	285	21	0	26%	2.20E-02	290	21	0	26%	4.298999786
[37]	301	21	0	26%	5.00E-02	301	21	0	26%	1.26
[37]	285	23	0	28%	2.70E-02	286	23	0	28%	3.632999897
[37]	296	22	1	28%	9.00E-03	296	22	1	28%	1.881999969
[37]	289	21	1	27%	5.10E-02	297	21	1	27%	3.752000093
[37]	289	21	0	26%	2.00E-02	297	21	0	26%	3.976999998
[37]	287	22	0	27%	2.80E-02	287	22	0	27%	1.60800004
[37]	289	23	0	28%	1.90E-02	290	23	0	28%	2.99
[37]	296	22	1	28%	3.00E-02	296	22	1	28%	1.57
[37]	292	22	0	27%	3.50E-02	292	22	0	27%	1.83
[37]	292	22	0	27%	3.10E-02	292	22	0	27%	1.48
[37]	286	22	0	27%	2.00E-02	288	22	0	27%	3.06
[37]	286	22	0	27%	1.90E-02	286	22	0	27%	1.76
[37]	285	23	0	28%	1.10E-02	287	23	0	28%	2.59

[37]	293	22	0	27%	4.80E-02	297	22	0	27%	2.88
[37]	290	22	1	28%	2.00E-02	290	22	1	28%	1.50
[37]	286	23	0	28%	2.00E-02	286	23	0	28%	1.23
[37]	297	22	1	28%	1.40E-02	297	22	1	28%	1.29
[37]	292	22	1	28%	6.20E-02	292	22	1	28%	1.40
[37]	292	23	1	30%	2.20E-02	292	23	1	30%	1.38
[37]	289	22	0	27%	1.80E-02	291	22	0	27%	2.60
[37]	301	22	1	28%	3.80E-02	301	22	1	28%	1.30
[37]	292	22	0	27%	2.10E-02	292	22	0	27%	1.25
[37]	294	23	0	28%	2.70E-02	295	23	0	28%	2.48
[37]	289	21	0	26%	2.70E-02	289	21	0	26%	1.51
[37]	294	21	0	26%	7.10E-02	294	21	0	26%	1.11500001
[37]	297	21	0	26%	1.06E-01	297	21	0	26%	1.264999986
[37]	294	21	0	26%	8.20E-02	294	21	0	26%	1.402999997
[37]	295	21	0	26%	5.10E-02	295	21	0	26%	1.636000037
[37]	297	21	0	26%	3.50E-02	297	21	0	26%	1.488000035
[37]	295	21	0	26%	8.00E-02	295	21	0	26%	1.412999988
[37]	298	21	0	26%	2.10E-02	298	21	0	26%	1.286000013
[37]	302	21	0	26%	3.80E-02	302	21	0	26%	1.621999979
[37]	295	21	0	26%	3.90E-02	297	21	0	26%	3.295000076
[37]	294	21	0	26%	2.00E-02	296	21	0	26%	2.977999926
[37]	296	21	0	26%	1.09E-01	296	21	0	26%	1.50
[37]	298	21	0	26%	2.10E-02	298	21	0	26%	1.371999979
[37]	293	21	0	26%	1.90E-02	293	21	0	26%	1.674999952
[37]	304	21	0	26%	5.30E-02	304	21	0	26%	1.671000004
[37]	304	21	0	26%	3.70E-02	304	21	0	26%	1.64
[37]	304	21	0	26%	2.40E-02	304	21	0	26%	1.54
[37]	294	21	0	26%	1.60E-02	296	21	0	26%	3.30
[37]	299	22	0	27%	2.30E-02	299	22	0	27%	1.350999951
[37]	286	23	0	28%	2.10E-02	286	23	0	28%	1.720999956
[37]	299	22	2	30%	3.30E-02	300	22	2	30%	2.61
[37]	293	22	0	27%	4.50E-02	293	22	0	27%	1.65
[37]	298	22	0	27%	3.20E-02	299	22	0	27%	2.54
[37]	284	23	0	28%	4.20E-02	284	23	0	28%	2.00
[37]	292	23	0	28%	1.20E-02	295	23	0	28%	2.99
[37]	295	22	1	28%	5.70E-02	301	22	1	28%	2.94
[37]	289	22	0	27%	1.40E-02	289	22	0	27%	1.25
[37]	280	23	0	28%	1.60E-02	282	23	0	28%	2.59
[37]	288	23	0	28%	5.00E-02	288	23	0	28%	1.47
[37]	284	23	0	28%	1.60E-02	284	23	0	28%	1.20
[37]	288	23	0	28%	4.40E-02	288	23	0	28%	1.43
[37]	285	23	0	28%	5.30E-02	286	23	0	28%	2.77
[37]	286	23	1	30%	2.40E-02	286	23	1	30%	1.20
[37]	286	23	0	28%	2.60E-02	287	23	0	28%	3.16
[37]	289	23	0	28%	2.90E-02	294	23	0	28%	2.40
[37]	284	23	0	28%	1.50E-02	287	23	0	28%	2.54
[37]	286	23	0	28%	3.30E-02	288	23	0	28%	3.16
[37]	284	23	0	28%	1.10E-02	287	23	0	28%	3.68

[37]	285	23	0	28%	2.30E-02	285	23	0	28%	1.65
[37]	290	23	0	28%	4.00E-02	290	23	0	28%	1.79
[37]	294	23	1	30%	1.40E-02	296	23	1	30%	2.94
[37]	292	23	1	30%	2.70E-02	293	23	1	30%	3.05
[37]	297	23	2	31%	2.30E-02	297	23	2	31%	1.54
[37]	273	23	0	28%	3.60E-02	278	23	0	28%	4.59
[37]	281	23	0	28%	1.14E-01	282	23	0	28%	3.50
[37]	287	23	0	28%	5.40E-02	287	23	0	28%	1.547999978
[37]	288	23	0	28%	1.30E-02	289	23	0	28%	3.58
[37]	289	23	0	28%	2.20E-02	289	23	0	28%	1.43
[37]	285	23	3	32%	2.80E-02	288	23	3	32%	3.22
[37]	291	23	4	33%	1.30E-02	294	23	4	33%	2.54
[37]	291	23	4	33%	4.30E-02	294	23	4	33%	3.06
[37]	286	23	1	30%	3.40E-02	289	23	1	30%	4.05
[37]	290	23	1	30%	4.60E-02	292	23	1	30%	3.30
[37]	286	23	1	30%	9.00E-03	286	23	1	30%	1.28
[37]	287	23	0	28%	4.50E-02	290	23	0	28%	2.78
[37]	282	23	0	28%	3.60E-02	285	23	0	28%	3.40
[37]	284	23	0	28%	2.40E-02	288	23	0	28%	3.90
[37]	284	23	0	28%	2.00E-02	288	23	0	28%	2.76
[37]	282	23	0	28%	5.80E-02	284	23	0	28%	2.47
[37]	288	23	0	28%	3.00E-02	292	23	0	28%	2.88
[37]	269	23	0	28%	3.80E-02	273	23	0	28%	4.35
[37]	284	23	0	28%	2.20E-02	286	23	0	28%	2.855999947
[37]	283	23	0	28%	2.00E-02	283	23	0	28%	1.65
[37]	292	23	4	33%	6.20E-02	295	23	4	33%	3.19
[37]	292	23	4	33%	4.50E-02	295	23	4	33%	3.11
[37]	288	23	1	30%	2.70E-02	288	23	1	30%	1.25
[37]	280	23	2	31%	2.10E-02	284	23	2	31%	3.99
[37]	279	23	2	31%	4.60E-02	281	23	2	31%	3.95
[37]	279	23	2	31%	2.20E-02	281	23	2	31%	3.80
[37]	285	23	0	28%	1.40E-02	288	23	0	28%	3.21
[37]	279	23	0	28%	1.30E-02	283	23	0	28%	3.47
[37]	283	23	0	28%	4.10E-02	287	23	0	28%	3.23
[37]	283	23	0	28%	2.50E-02	285	23	0	28%	3.21
[37]	279	23	0	28%	1.20E-02	285	23	0	28%	2.96
[37]	281	23	0	28%	3.70E-02	288	23	0	28%	4.04
[37]	282	23	0	28%	3.50E-02	285	23	0	28%	3.316999912
[37]	284	23	0	28%	2.90E-02	288	23	0	28%	2.79
[37]	283	23	0	28%	2.40E-02	283	23	0	28%	1.75
[37]	284	23	0	28%	2.90E-02	284	23	0	28%	1.69
[37]	295	22	1	28%	2.60E-02	295	22	1	28%	1.59
[37]	290	22	1	28%	2.60E-02	290	22	1	28%	1.71
[37]	292	22	0	27%	6.20E-02	292	22	0	27%	1.84
[37]	292	22	1	28%	3.40E-02	294	22	1	28%	3.46
[37]	299	22	1	28%	4.60E-02	301	22	1	28%	2.95
[37]	286	22	0	27%	2.50E-02	288	22	0	27%	3.01
[37]	300	22	0	27%	4.80E-02	300	22	0	27%	1.83

[37]	290	22	1	28%	3.80E-02	290	22	1	28%	1.98
[37]	304	22	3	31%	1.70E-02	304	22	3	31%	1.92
[37]	295	22	1	28%	2.10E-02	295	22	1	28%	1.78
[37]	286	22	0	27%	4.50E-02	287	22	0	27%	3.54
[37]	287	22	0	27%	4.50E-02	288	22	0	27%	3.98
[37]	292	22	1	28%	1.20E-02	294	22	1	28%	3.37
[37]	287	22	1	28%	1.25E-01	287	22	1	28%	1.63
[37]	297	22	1	28%	4.00E-02	298	22	1	28%	3.118000031
[37]	300	22	1	28%	1.80E-02	301	22	1	28%	3.09
[37]	283	22	0	27%	4.20E-02	285	22	0	27%	3.536000013
[37]	287	22	0	27%	1.90E-02	291	22	0	27%	3.23
[37]	282	22	0	27%	1.90E-02	282	22	0	27%	1.71
[37]	291	22	1	28%	3.70E-02	293	22	1	28%	3.34
[37]	286	22	1	28%	6.80E-02	288	22	1	28%	4.14
[37]	291	22	1	28%	5.10E-02	293	22	1	28%	3.08
[37]	288	22	1	28%	4.00E-02	290	22	1	28%	3.55
[37]	293	22	1	28%	3.80E-02	297	22	1	28%	3.27
[37]	289	22	1	28%	1.40E-02	291	22	1	28%	3.29
[37]	290	22	1	28%	3.80E-02	293	22	1	28%	3.46
[37]	304	22	1	28%	1.90E-02	304	22	1	28%	1.42
[37]	283	22	0	27%	4.30E-02	285	22	0	27%	3.35
[37]	285	22	0	27%	2.00E-02	289	22	0	27%	2.96
[37]	283	22	0	27%	5.20E-02	285	22	0	27%	3.43
[37]	282	22	0	27%	2.30E-02	285	22	0	27%	4.16
[37]	283	22	0	27%	1.60E-02	285	22	0	27%	3.25
[37]	285	22	0	27%	4.00E-02	289	22	0	27%	3.51
[37]	288	22	0	27%	3.40E-02	296	22	0	27%	2.89199996
[37]	270	22	0	27%	2.90E-02	275	22	0	27%	4.11
[37]	285	22	0	27%	3.30E-02	290	22	0	27%	3.191999912
[37]	286	22	0	27%	4.40E-02	288	22	0	27%	2.98
[37]	282	22	0	27%	3.70E-02	285	22	0	27%	3.63
[37]	285	22	0	27%	3.10E-02	289	22	0	27%	3.61
[37]	284	22	0	27%	4.10E-02	288	22	0	27%	3.63
[37]	282	22	0	27%	1.90E-02	284	22	0	27%	3.801000118
[37]	284	22	0	27%	3.80E-02	288	22	0	27%	3.45
[37]	281	22	0	27%	2.30E-02	286	22	0	27%	5.11
[37]	286	22	0	27%	2.30E-02	289	22	0	27%	3.13
[37]	288	22	0	27%	2.90E-02	290	22	0	27%	3.20
[37]	282	22	0	27%	2.10E-02	285	22	0	27%	2.68
[37]	294	22	0	27%	4.60E-02	294	22	0	27%	1.71
[37]	289	22	1	28%	2.80E-02	291	22	1	28%	3.87
[37]	289	22	1	28%	5.00E-02	291	22	1	28%	3.92
[37]	279	22	1	28%	1.20E-02	284	22	1	28%	4.34
[37]	283	22	2	30%	2.10E-02	285	22	2	30%	3.88
[37]	278	22	1	28%	4.00E-02	283	22	1	28%	3.75
[37]	281	22	2	30%	2.80E-02	285	22	2	30%	4.190000057
[37]	281	22	2	30%	3.30E-02	285	22	2	30%	3.59
[37]	294	22	1	28%	3.30E-02	295	22	1	28%	2.64

[37]	289	22	1	28%	2.70E-02	292	22	1	28%	2.75
[37]	291	22	1	28%	1.80E-02	293	22	1	28%	3.59
[37]	303	22	1	28%	2.50E-02	304	22	1	28%	3.14
[37]	291	22	1	28%	1.50E-02	296	22	1	28%	3.83
[37]	301	22	0	27%	3.80E-02	302	22	0	27%	3.16
[37]	286	22	0	27%	3.40E-02	290	22	0	27%	3.06
[37]	284	22	0	27%	2.60E-02	291	22	0	27%	3.72
[37]	287	22	0	27%	2.80E-02	293	22	0	27%	3.26
[37]	281	22	0	27%	4.00E-02	283	22	0	27%	3.16
[37]	263	22	0	27%	4.30E-02	267	22	0	27%	4.09
[37]	271	22	0	27%	2.20E-02	276	22	0	27%	4.104000092
[37]	264	22	0	27%	4.70E-02	269	22	0	27%	4.47
[37]	288	22	0	27%	3.90E-02	290	22	0	27%	4.01
[37]	271	22	0	27%	1.40E-02	276	22	0	27%	4.57
[37]	279	22	0	27%	4.30E-02	283	22	0	27%	3.44
[37]	286	22	0	27%	6.00E-02	288	22	0	27%	3.23
[37]	287	22	0	27%	3.20E-02	289	22	0	27%	3.16
[37]	271	22	0	27%	8.50E-02	281	22	0	27%	4.137000084
[37]	280	22	0	27%	1.90E-02	284	22	0	27%	3.572999954
[37]	285	22	0	27%	2.30E-02	287	22	0	27%	3.18
[37]	285	22	0	27%	4.20E-02	286	22	0	27%	3.66
[37]	289	22	0	27%	4.60E-02	293	22	0	27%	3.69
[37]	285	22	0	27%	2.90E-02	289	22	0	27%	3.22
[37]	281	22	0	27%	6.80E-02	281	22	0	27%	1.91
[37]	299	21	0	26%	1.80E-02	299	21	0	26%	1.89
[37]	292	21	1	27%	2.00E-02	298	21	1	27%	3.35
[37]	304	21	1	27%	4.60E-02	305	21	1	27%	3.13
[37]	287	21	0	26%	3.30E-02	296	21	0	26%	3.21
[37]	294	21	1	27%	4.60E-02	301	21	1	27%	3.70
[37]	285	21	0	26%	3.00E-02	292	21	0	26%	3.40
[37]	286	21	0	26%	2.60E-02	290	21	0	26%	3.267999887
[37]	288	22	1	28%	3.50E-02	289	22	1	28%	2.90
[37]	294	22	1	28%	3.50E-02	298	22	1	28%	3.06
[37]	286	23	1	30%	1.90E-02	289	23	1	30%	2.53
[37]	275	23	2	31%	2.10E-02	277	23	2	31%	3.57
[37]	290	22	1	28%	1.60E-02	292	22	1	28%	2.97
[37]	290	22	1	28%	4.30E-02	292	22	1	28%	3.41
[37]	294	23	1	30%	3.40E-02	296	23	1	30%	2.84
[37]	290	23	1	30%	1.50E-02	293	23	1	30%	3.18
[37]	293	21	1	27%	1.50E-02	297	21	1	27%	4.05
[37]	292	21	1	27%	3.50E-02	296	21	1	27%	3.26
[37]	286	23	1	30%	1.60E-02	289	23	1	30%	3.29
[37]	288	22	1	28%	3.40E-02	290	22	1	28%	3.40
[37]	287	22	1	28%	1.40E-02	288	22	1	28%	3.06
[37]	293	21	1	27%	3.30E-02	300	21	1	27%	3.44
[37]	281	23	0	28%	1.10E-02	287	23	0	28%	3.16
[37]	284	22	0	27%	2.80E-02	288	22	0	27%	3.44
[37]	284	22	0	27%	1.40E-02	288	22	0	27%	3.426000118

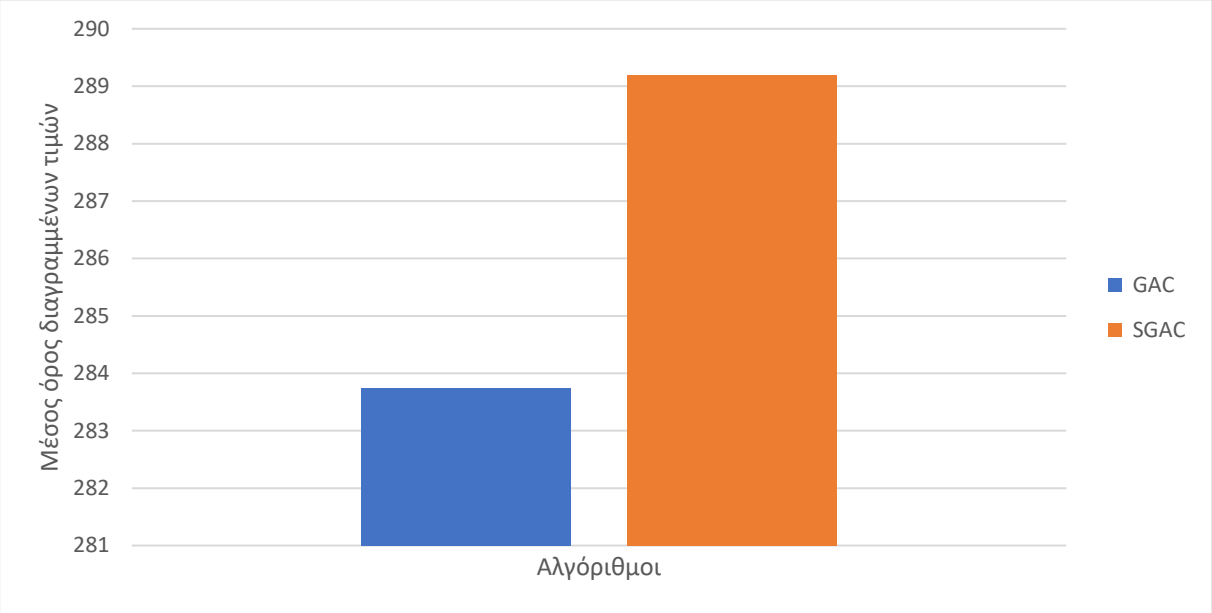
[37]	271	23	0	28%	8.40E-02	276	23	0	28%	3.48
[37]	285	22	0	27%	1.40E-02	287	22	0	27%	3.34
[37]	285	21	0	26%	6.30E-02	289	21	0	26%	3.71
[37]	288	22	0	27%	4.10E-02	292	22	0	27%	2.84
[37]	266	22	0	27%	1.90E-02	270	22	0	27%	3.75
[37]	287	22	0	27%	3.00E-02	292	22	0	27%	3.255000114
[37]	285	22	0	27%	2.50E-02	289	22	0	27%	3.332000017
[37]	282	23	0	28%	4.30E-02	286	23	0	28%	3.82
[37]	282	23	0	28%	2.20E-02	284	23	0	28%	3.18
[37]	290	21	1	27%	2.90E-02	295	21	1	27%	5.07
[37]	289	21	1	27%	3.10E-02	295	21	1	27%	3.40
[37]	285	23	0	28%	3.20E-02	286	23	0	28%	2.83
[37]	284	22	0	27%	3.30E-02	286	22	0	27%	3.59800005
[37]	289	22	0	27%	2.80E-02	293	22	0	27%	4.35
[37]	289	22	0	27%	3.00E-02	296	22	0	27%	3.31
[37]	293	22	1	28%	3.40E-02	297	22	1	28%	2.95
[37]	288	22	1	28%	2.50E-02	295	22	1	28%	3.96
[37]	288	22	1	28%	3.20E-02	294	22	1	28%	3.63
[37]	282	23	2	31%	2.20E-02	284	23	2	31%	3.44
[37]	282	23	2	31%	2.50E-02	284	23	2	31%	3.68
[37]	289	23	1	30%	4.00E-02	289	23	1	30%	1.52
[37]	273	23	0	28%	5.00E-02	279	23	0	28%	4.19
[37]	286	23	0	28%	2.80E-02	290	23	0	28%	3.30
[37]	283	23	0	28%	1.60E-02	287	23	0	28%	3.532999992
[37]	281	21	0	26%	1.70E-02	288	21	0	26%	3.79
[37]	288	23	0	28%	3.50E-02	288	23	0	28%	1.506999969
[37]	293	22	0	27%	3.80E-02	293	22	0	27%	2.12
[37]	271	22	0	27%	3.70E-02	276	22	0	27%	3.67
[37]	269	24	0	30%	1.40E-02	271	24	0	30%	4.16
[37]	268	22	0	27%	3.30E-02	274	22	0	27%	4.87
[37]	270	22	0	27%	2.10E-02	274	22	0	27%	3.86
[37]	267	22	0	27%	2.70E-02	272	22	0	27%	4.55
[37]	275	23	0	28%	1.22E-01	277	23	0	28%	4.37
[37]	270	22	0	27%	4.30E-02	274	22	0	27%	4.34
[37]	265	22	0	27%	3.30E-02	270	22	0	27%	4.44
[37]	275	23	0	28%	2.10E-02	277	23	0	28%	3.44
[37]	268	24	0	30%	3.60E-02	271	24	0	30%	3.89
[37]	265	22	0	27%	4.40E-02	269	22	0	27%	4.58
[37]	273	22	0	27%	3.60E-02	277	22	0	27%	4.48
[37]	270	23	0	28%	3.00E-02	276	23	0	28%	4.51
[37]	280	23	2	31%	2.90E-02	282	23	2	31%	3.68
[37]	280	23	2	31%	2.50E-02	282	23	2	31%	3.795000076
[37]	280	23	2	31%	3.50E-02	282	23	2	31%	3.84800005
[37]	270	23	0	28%	5.70E-02	271	23	0	28%	4.294000149
[37]	281	22	3	31%	1.33E-01	283	22	3	31%	3.98
[37]	273	23	2	31%	6.30E-02	275	23	2	31%	3.305999994
[37]	266	23	0	28%	3.50E-02	271	23	0	28%	6.36
[37]	268	23	0	28%	5.10E-02	277	23	0	28%	5.862999916

[37]	269	23	0	28%	4.30E-02	274	23	0	28%	4.40
[37]	267	24	0	30%	3.20E-02	269	24	0	30%	3.99
[37]	279	23	1	30%	3.00E-02	283	23	1	30%	5.00
[37]	272	23	1	30%	6.80E-02	275	23	1	30%	3.85
[37]	275	24	1	31%	2.40E-02	275	24	1	31%	1.59
[37]	278	23	0	28%	1.30E-02	283	23	0	28%	3.34
[37]	272	23	1	30%	8.90E-02	273	23	1	30%	4.91
[37]	276	23	1	30%	3.20E-02	280	23	1	30%	3.41
[37]	271	22	0	27%	1.20E-02	276	22	0	27%	4.62
[37]	269	22	0	27%	2.90E-02	272	22	0	27%	4.18
[37]	265	22	0	27%	2.70E-02	270	22	0	27%	5.06
[37]	272	22	0	27%	2.20E-02	277	22	0	27%	4.337999821
[37]	268	22	0	27%	2.20E-02	273	22	0	27%	4.980000019
[37]	265	22	0	27%	3.00E-02	271	22	0	27%	4.487999916
[37]	269	22	1	28%	3.30E-02	271	22	1	28%	4.47
[37]	272	22	0	27%	2.20E-02	277	22	0	27%	4.89
[37]	273	22	0	27%	3.70E-02	278	22	0	27%	4.49
[37]	272	22	0	27%	3.20E-02	276	22	0	27%	4.48
[37]	274	22	0	27%	3.00E-02	278	22	0	27%	4.26
[37]	266	22	0	27%	1.80E-02	271	22	0	27%	4.04
[37]	273	22	0	27%	1.30E-02	279	22	0	27%	4.11
[37]	265	22	0	27%	3.40E-02	270	22	0	27%	4.09
[37]	285	23	0	28%	2.00E-02	287	23	0	28%	3.039000034
[37]	273	22	0	27%	1.60E-02	281	22	1	28%	3.72
[37]	271	22	0	27%	1.90E-02	271	22	0	27%	2.05
[37]	278	22	1	28%	2.70E-02	280	22	1	28%	4.58
[37]	269	23	0	28%	2.20E-02	269	23	0	28%	1.69
[37]	273	23	0	28%	2.10E-02	275	23	0	28%	3.56
[37]	275	22	1	28%	2.20E-02	275	22	1	28%	1.94
[37]	278	22	1	28%	1.30E-02	282	22	1	28%	4.22
[37]	278	22	1	28%	1.70E-02	282	22	1	28%	4.29
[37]	293	22	0	27%	3.60E-02	300	22	0	27%	2.83
M.O	286.72	22.03	0.51	28%	3.47E-02	289.20	22.03	0.51	28%	3.07

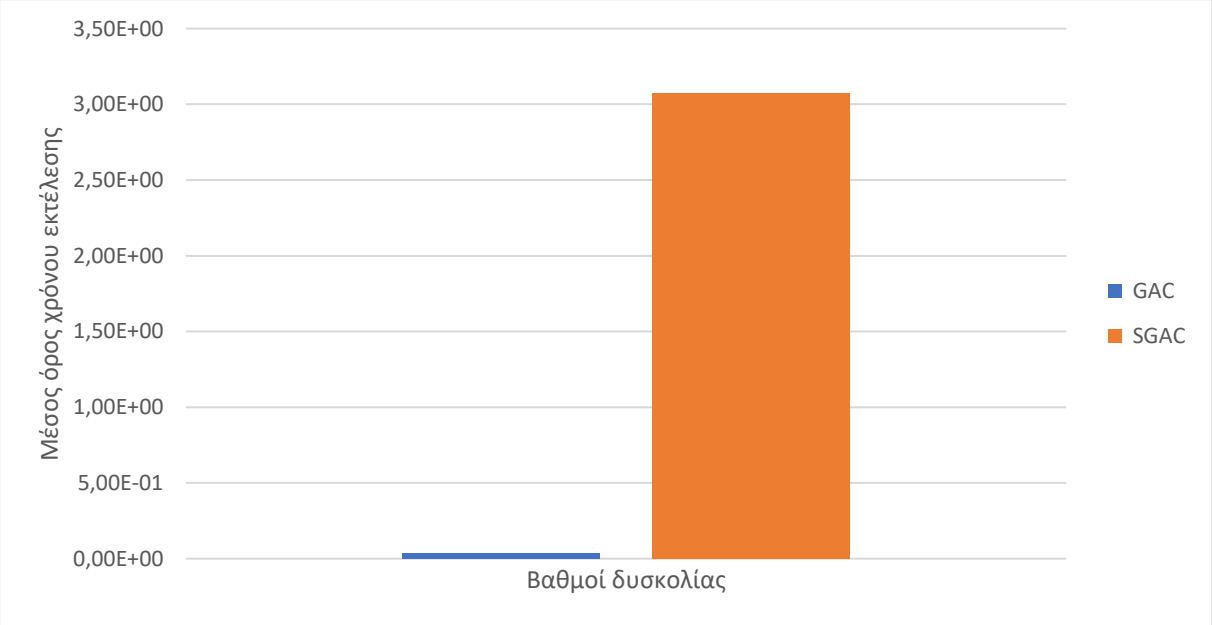
Πίνακας 5. Μετρήσεις σε Hardest known Sudoku puzzles

Παρακάτω δίνεται η γραφική απεικόνιση των αποτελεσμάτων του μέσου χρόνου εκτέλεσης και των διαγραμμένων τιμών από τα πεδία ορισμού των μεταβλητών των πειραμάτων που έγιναν στους δύο αλγόριθμους για τα Hardest known Sudoku puzzles. Στον x άξονα βρίσκονται οι αλγόριθμοι που χρησιμοποιήθηκαν και στον y τα αποτελέσματα των πειραμάτων ανά πείραμα. Όπως αναφέραμε οι δυο αλγόριθμοι βρήκαν το περισσότερο 4 επιπλέον τιμές σε αυτές που δόθηκαν από το πρόβλημα και έτσι το ποσοστό ολοκλήρωσης δεν θεωρείται ως αντιπροσωπευτικό της ικανότητας επίλυσης των προβλημάτων αυτής της δυσκολίας. Μοναδικό μέσο σύγκρισης έτσι είναι οι διαγεγραμμένες τιμές από τα domains διότι οι δύο αλγόριθμοι θα μπορούσαν να χρησιμοποιηθούν ως προ-επεξεργασία σε μία μέθοδο επίλυσης με αναζήτηση, μιας και

αφαιρούν ένα ικανοποιητικό ποσό από τιμές από τα πεδία ορισμού των μεταβλητών. Γι' αυτό τον λόγο συμπεριλήφθηκαν μόνο τα γραφήματα που απεικονίζουν τις διαγραμμένες τιμές από τα domain των μεταβλητών καθώς και ο χρόνος εκτέλεσης.



Γράφημα 4. Μέσος όρος διαγραμμένων τιμών των πειραμάτων στα Hardest known Sudoku puzzles σε σχέση με αλγόριθμο που χρησιμοποιήθηκε



Γράφημα 5. Μέσος όρος χρόνου εκτέλεσης των πειραμάτων στα Hardest known Sudoku puzzles σε σχέση με αλγόριθμο που χρησιμοποιήθηκε

Κεφάλαιο 5

Εφαρμογή

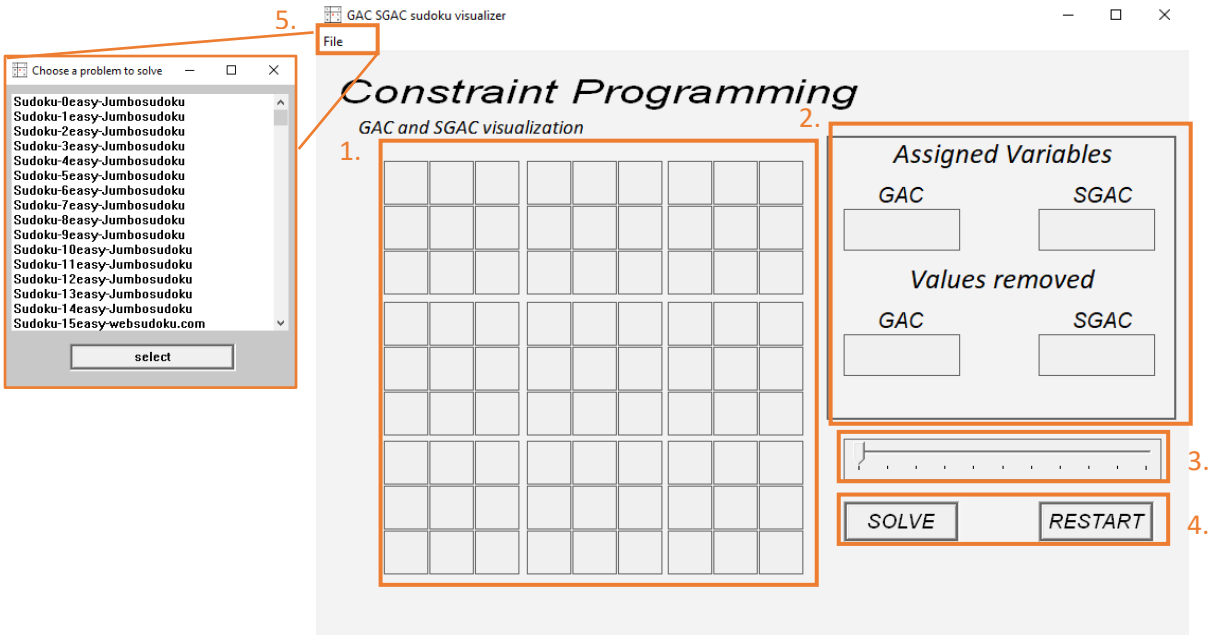
Το πρόβλημα του sudoku, όπως αναφέραμε είναι ένα από τα βασικά προβλήματα που χρησιμοποιούνται ως βοήθημα για την κατανόηση του προγραμματισμού περιορισμών. Οι I. Howell, R. Woodward, B. Choueiry, C. Bessiere δημιούργησαν ένα online solver για sudoku ως αναβάθμιση μιας προηγούμενης έκδοσης που είχε δημιουργηθεί, με σκοπό να δείξουν στους μαθητές του εργαστηρίου του πανεπιστημίου τους οπτικά, διάφορες μεθόδους επιβολής συνέπειας [**38**]. Με αφορμή αυτήν την υλοποίηση, αλλά και άλλες παρόμοιου σκοπού [**37**], δημιουργήσαμε μια εφαρμογή σε τοπικό σύστημα υπολογιστή με χρήση της του windows API έχοντας σκοπό την επέκταση της διδασκαλίας του προγραμματισμού περιορισμών στο δικό μας πανεπιστήμιο.

Η επιλογή του windows API έγινε διότι η δική μας υλοποίηση είναι γραμμένη στην γλώσσα προγραμματισμού C, που με κάποιες μικρές αλλαγές μπορεί να αναγνωριστεί εύκολα από την C++, την γλώσσα προγραμματισμού δηλαδή που χρησιμοποιεί το windows API. Οι αλλαγές που πρέπει να γίνουν προκειμένου να δημιουργηθεί μια τέτοια εφαρμογή έτσι είναι πολύ λιγότερες από αυτές που θα χρειαζόντουσαν εάν αξιοποιούσαμε οποιοδήποτε άλλο εργαλείο, μιας και μπορεί να χρησιμοποιηθεί ο ήδη υλοποιημένος κώδικας που εφαρμόζει τις ιδιότητες GAC και SGAC του Κεφαλαίου 3.

5.1 Περιγραφή του γραφικού περιβάλλοντος

Το γραφικό περιβάλλον της εφαρμογής απαρτίζεται από τα εξής στοιχεία:

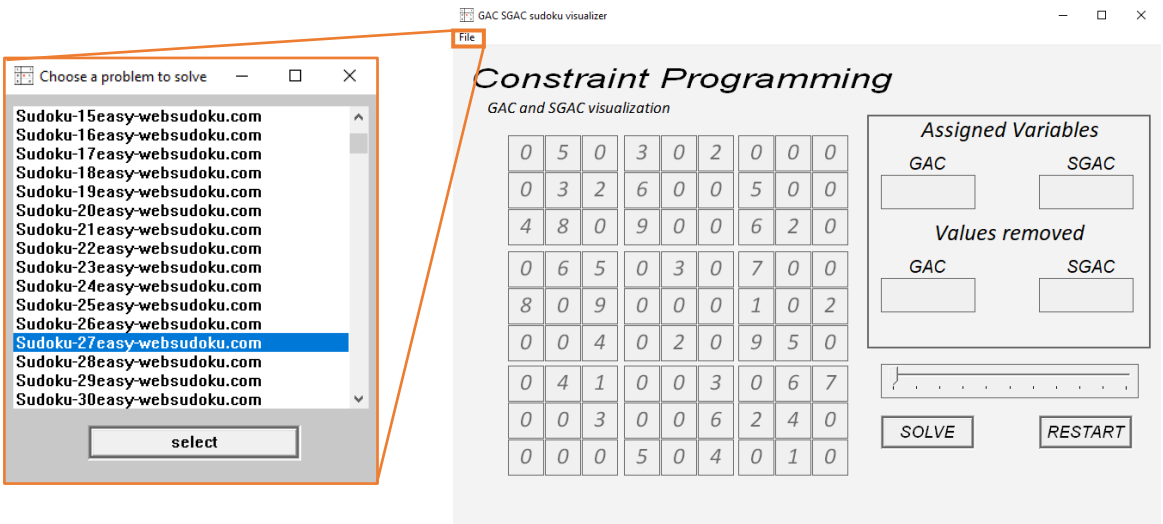
1. Την κύρια περιοχή sudoku που περιέχει το πρόβλημα του έχει επιλέξει ο χρήστης. Ο τρόπος αναπαράστασης είναι, 0 για τα κελιά sudoku που δεν έχουν τιμή και τους αριθμούς 1 έως 9 για τα κελιά που έχουν ή πήραν τιμή μετά την εκτέλεση των αλγορίθμων
2. Την περιοχή αποτελεσμάτων που περιέχει τις μεταβλητές που πήραν τιμή (σύνολο μαζί με τις προσυμπληρωμένες) καθώς και τις τιμές που διαγράφθηκαν από τα πεδία ορισμού των μεταβλητών για κάθε ένα αλγόριθμο
3. Μία μπάρα ταχύτητας για να ελέγχει ο χρήστης την ταχύτητα εμφάνισης της επίλυσης των δύο μεθόδων
4. Δύο κουμπιά για επίλυση και επανεκκίνηση του προβλήματος που βρίσκεται στην κύρια περιοχή sudoku
5. Ένα μενού όπου μπορεί να επιλέξει ο χρήστης ποιο πρόβλημα επιθυμεί να λύσει. Τα προβλήματα που περιέχονται είναι αυτά που χρησιμοποιήθηκαν και για τις μετρήσεις του προηγούμενου κεφαλαίου



Σχήμα 11. Γραφικό περιβάλλον της εφαρμογής οπτικοποίησης των GAC/SGAC αλγόριθμων

5.2 Περιγραφή τρόπου χρήσης της εφαρμογής

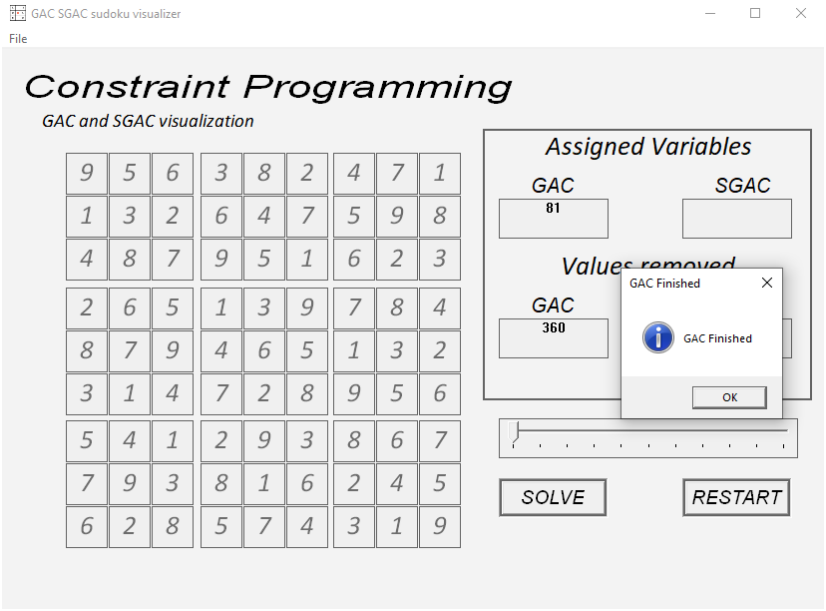
Ο τρόπος χειρισμού της εφαρμογής είναι απλός, για την περιγραφή του θα χρησιμοποιηθεί ένα παράδειγμα που παρουσιάζεται παρακάτω. Ο χρήστης αρχικά επιλέγει ένα από τα διαθέσιμα προβλήματα και η κύρια περιοχή sudoku ανανεώνεται αυτόματα με το πρόβλημα που επιλέχθηκε.



Σχήμα 12. Παράδειγμα περιγραφής χρήσης εφαρμογής: επιλογή προβλήματος

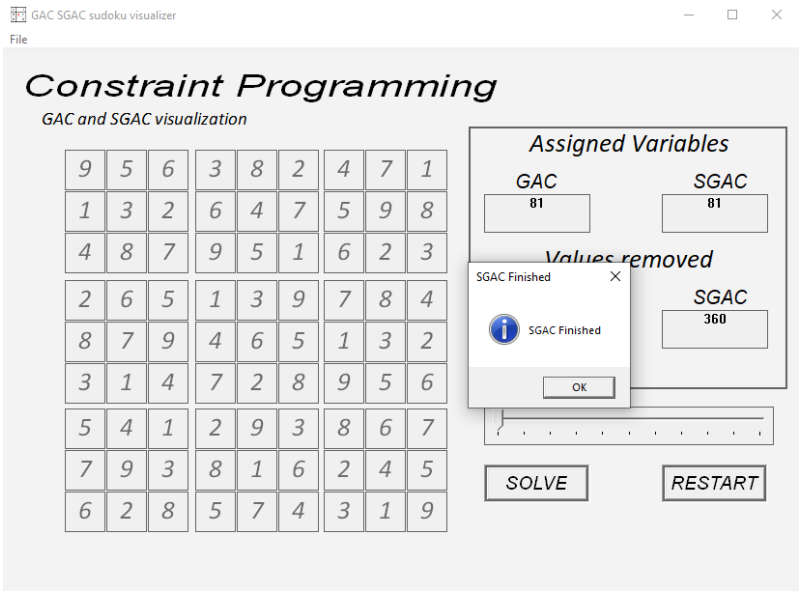
Ο χρήστης έπειτα επιλέγει την ταχύτητα εμφάνισης της επίλυσης (αριστερά πιο γρήγορα, δεξιά πιο αργά) και επιλέγει το κουμπί SOLVE για να επιλυθεί το πρόβλημα.

Εάν δεν έχει επιλεχθεί κάποιο πρόβλημα στο βήμα 1, τότε επιλέγεται το πρώτο πρόβλημα στην λίστα προβλημάτων.



Σχήμα 13. Παράδειγμα περιγραφής χρήσης εφαρμογής: επίλυση με GAC

Το πρόγραμμα εμφανίζει δυναμικά την λύση με βάση την ταχύτητα που ορίστηκε στην μπάρα ταχύτητας για τον αλγόριθμο που εφαρμόζει GAC και μόλις τελειώσει εμφανίζει ένα μήνυμα ότι ο αλγόριθμος τερμάτισε, τον αριθμό των μεταβλητών που πήραν τιμή καθώς και το πλήθος των διαγραμμένων τιμών από τα domains των μεταβλητών. Όταν ο χρήστης επιλέξει το OK, γίνεται επαναφορά της κύριας περιοχής sudoku στο πρόβλημα που είχε επιλεχθεί και ξεκινάει η επίλυση του προβλήματος με τον αλγόριθμο που εφαρμόζει SGAC εκτελώντας την ίδια διαδικασία. Ο χρήστης έπειτα μπορεί να επιλέξει απευθείας ένα νέο πρόβλημα προς επίλυση.



Σχήμα 14. Παράδειγμα περιγραφής χρήσης εφαρμογής: επίλυση με SGAC

Κεφάλαιο 6

Συμπεράσματα – Επεκτάσεις

Ο προγραμματισμός περιορισμών είναι ένα πολύ δυνατό εργαλείο στην επίλυση προβλημάτων sudoku. Ο αλγόριθμος που εφαρμόζει GAC μπόρεσε με πολύ λιγότερο πλήθος περιορισμών έναντι του απλού AC να επιλύσει το μεγαλύτερο ποσοστό προβλημάτων sudoku με μέσο όρο ποσοστού ολοκλήρωσης άνω του 90%. Βάσει θεωρίας, μιας και τα καλά διαμορφωμένα προβλήματα sudoku εγγυώνται λύση, ο αλγόριθμος που εφαρμόζει SGAC μπόρεσε να επιλύσει όλα τα προβλήματα σε όλες τις κατηγορίες δυσκολίας sudoku σε ποσοστό ολοκλήρωσης 100% με εξαίρεση τα Hardest known Sudoku problems.

Στα προβλήματα easy, medium, hard και fiendish, η επιτυχία του αλγόριθμου που επιβάλλει SGAC στην επίλυση των προβλημάτων (Μ.Ο. ποσοστού ολοκλήρωσης 100% σε όλα τα στιγμιότυπα προβλημάτων), οφείλεται στην επανάληψη του αλγόριθμου GAC για διάφορα νέα στιγμιότυπα του προβλήματος με ανάθεση μιας τιμής σε μία μεταβλητή. Και οι δύο αλγόριθμοι είχαν χρόνο εκτέλεσης πολύ μικρό, με μεγαλύτερο χρόνο στα fiendish στιγμιότυπα προβλημάτων, έχοντας 3.15×10^{-2} s για τον αλγόριθμο που εφαρμόζει GAC και 1.17×10^{-1} s για τον αλγόριθμο που εφαρμόζει SGAC.

Σε πολύ δύσκολα προβλήματα sudoku (Hardest known Sudoku problems), από τον αριθμό των τιμών που αποδόθηκαν σε μεταβλητές, που ήταν το πολύ 4, συμπεραίνουμε πως οι αλγόριθμοι που εφαρμόζουν GAC και SGAC δεν επαρκούν για την επίλυση τέτοιου επιπέδου δυσκολίας προβλημάτων. Παρ' όλα αυτά, από τον αριθμό των διαγραμμένων τιμών από τα πεδία ορισμού των μεταβλητών, φαίνεται πως οι δύο αλγόριθμοι μπορούν να χρησιμοποιηθούν ως βοηθήματα στην προ-επεξεργασία ενός αλγόριθμου αναζήτησης. Βλέποντας μάλιστα και την μεγάλη διαφορά χρόνου, με σχετικά ίδιο ποσό διαγραμμένων τιμών από τα domains των μεταβλητών ανάμεσα στους δύο αλγόριθμους (3.47×10^{-2} s και 286.72 διαγραμμένες τιμές κατά μέσο όρο για τον αλγόριθμο που εφαρμόζει GAC και 3.07 s και 289.20 διαγραμμένες τιμές κατά μέσο όρο για τον αλγόριθμο που εφαρμόζει SGAC), μπορούμε να συμπεράνουμε πως στις περισσότερες περιπτώσεις είναι πιο αποδοτική η χρήση του απλού αλγόριθμου που εφαρμόζει GAC λόγω της εξοικονόμησης χρόνου.

Η υλοποίηση σε κώδικα έγινε σε γλώσσα C για λόγους ταχύτητας. Η τροποποίηση του κώδικα ώστε να περιλαμβάνει την δυνατότητα επίλυσης και άλλων ειδών προβλημάτων sudoku όπως Str8ts, διαγώνια, jigsaw ή και διαφορετικού μεγέθους προβλημάτων (16x16 ή 32x32) αποτελεί μια πιθανή μελλοντική προσθήκη. Η ενσωμάτωση των GAC και SGAC σε έναν αλγόριθμο αναζήτησης θα ήταν ακόμη μία ενδιαφέρουσα προσθήκη, μιας και αναμένεται να επιτυγχάνεται η επίλυση όλων των

προβλημάτων sudoku. Ακόμα και των πλέον δύσκολων που οι τεχνικές διάδοσης περιορισμών δεν μπορούν να επιλύσουν από μόνες τους.

Βιβλιογραφία

[1] V. Kumar, “Algorithms for Constraint-Satisfaction Problems: A Survey”, *AI Magazine*, issue 1, vol. 13, p. 32, Mar. 1992

[2] S. Brailsford, C. Potts, B. Smith., “Constraint satisfaction problems: Algorithms and applications”, *European Journal of Operational Research*, vol. 119, Issue 3, pp. 557-581, Dec 1999

[3] Y. Takayuki, “Complexity and Completeness of Finding Another Solution and its Application to Puzzles”, Master’s thesis dissertation, Dep. of Information Science, Univ. of Tokyo, Japan, Bunkyo-ku, 2003

[4] L. Nadel, D. Stein, *Lectures In Complex Systems*, 1st ed., Boca Raton: CRC Press, 1991

[5] M. S. Fox, “Constraint-guided scheduling—A short history of research at CMU”, *Computers in Industry*, vol. 14, Issues 1-3, pp. 79-88, May 1990

[6] M. S. Fox, N. Sadeh, C. Baykan, “Constrained heuristic search”, in Proceedings of the 11th international joint conference on Artificial intelligence, San Francisco, CA, USA, 1989, pp. 309-315

[7] J. F. Rit, “Propagating temporal constraints for scheduling”, in Proceedings of the Fifth AAAI National Conference on Artificial Intelligence, Philadelphia, Pennsylvania, 1986, pp. 383-388

[8] V. Dhar, N. Ranganathan, “Integer programming vs. expert systems: an experimental comparison”, *Communications of the ACM*, vol. 33, Issue 3, pp. 323-336, Mar. 1990

[9] G. Freuder and M. Wallace, "Constraint technology and the commercial world [Interview] ", *IEEE Intelligent Systems and their Applications*, vol. 15, Issue 1, pp. 20-23, Jan.-Feb. 2000

[10] A. Chabrier, “A cooperative CP and LP optimizer approach for the pairing generationproblem”, in Proceedings of the International Workshop on Integration of ArtificialIntelligence and Operations Research Techniques in Constraint Programming forCombintaorial Optimization Problems, Ferrara, Italy, 2000

[11] T. Fahle, U. Junker, S. Karish, N. Kohn, M. Sellmann, B. Vaaben, “Constraint Programming Based Column Generation for Crew Assignment”, *Journal of Heuristics*, vol. 8, pp. 59–81, Jan. 2002

[12] U. Junker, S. Karish, N. Kohl, B. Vaaben, T. Fahle, M. Sellmann, “A Framework for Constraint Programming Based Column Generation”, in Proceedings of the 5th International Conference on Principles and Practice of Constraint Programming, Berlin, Heidelberg, 1999, pp. 261–275

[13] N. Kohl, “Application of or and cp techniques in a real world crew scheduling system”, in Proceedings of the International Workshop on Integration of Artificial Intelligence and Operations Research Techniques in Constraint Programming for Combinatorial Optimization Problems, Paderborn, Germany, 2000, pp. 105-108

[14] T. Yunes, A. Moura, C. Souza, "Hybrid column generation approaches for urban transit crew management problems", *Transportation Science*, vol. 39, Issue 2, pp. 273-288, May 2005
[15] S. Creff, J. Le Noir, E. Lenormand, S. Madelénat, "Towards Facilities for Modeling and Synthesis of Architectures for Resource Allocation Problem in Systems Engineering", in Proceedings of the 24th ACM International Systems and Software Product Line Conference, Montreal, Canada, 2022, pp. 1-11
[16] C. Eastman, "Preliminary report on a system for general space planning", *Communications of the ACM*, Issue 2, vol. 15, pp. 76-87, Feb. 1972
[17] J. Kleers, G. J. Sussman, "Propagation of constraints applied to circuit synthesis", *Circuit Theory and Applications*, vol. 8, Issue 2, pp. 127-144, April 1980
[18] A. Chabrier, "Heuristic branch-and-price-and-cut to solve a network design problem", in Proceedings of CPAIOR 2003, Montr´eal Canada, May 2003
[19] F. Rossi, P. V. Beek, T. Walsh, *Handbook of Constraint Programming*, 1st ed., Amsterdam: Elsevier, 2006
[20] S. Golomb, L. Baumert, "Backtrack Programming", *Journal of the ACM*, Issue 3, vol. 12, pp. 516-524, Oct. 1965
[21] J. R. Bitner, E. M. Reingold, "Backtrack programming techniques", *Communications of the ACM*, Issue 1, vol. 18, pp. 651-656, Nov. 1975
[22] P. Fillmore, S. Williamson, "On Backtracking: A Combinatorial Description of the Algorithm", *SIAM Journal on Computing*, Issue 1, vol. 3, 1974
[23] A. K. Mackworth, "Consistency in networks of relations", *Artificial Intelligence*, vol. 8, Issue 1, pp. 99-118, 1977
[24] U. Montanari, "Networks of constraints: Fundamental properties and applications to picture processing", *Information Sciences*, vol. 7, pp. 95–132, 1974
[25] R. Backofen, S. Will, "Optimally Compact Finite Sphere Packings — Hydrophobic Cores in the FCC", Lecture Notes in Computer Science, pp. 257–271, 2001
[26] Ι. Βλαχάβας, Π. Κεφαλάς, Ν. Βασιλειάδης, Φ. Κόκκορας, Η. Σακελλαρίου, *Τεχνητή Νοημοσύνη*, 4th ed., Κοζάνη: Εκδόσεις Πανεπιστημίου Δυτικής Μακεδονίας, 2011
[27] R. Mohr, G. Masini, "Good old discrete relaxation", in Proceedings of the 8th European Conference on Artificial Intelligence, Munich, Germany, 1988, pp. 651–656
[28] W. J. V. Hoeve, I. Katriel, "Chapter 6 - Global Constraints" in *Handbook of Constraint Programming*, 1st ed., Amsterdam: Elsevier, 2006
[29] R. Debruyne, C. Bessiere, "Some practicable filtering techniques for the Constraint Satisfaction Problem", in Proceedings of IJCAI'97, Nagoya, Japan, 1997, pp. 412–417
[30] M. Sharir, "A strange sorting method inspired by formal differentiation", *Computers & Mathematics with Applications*, Issue 4, vol. 7, pp. 67-72, 1981
[31] J. C. Régin, "A Filtering Algorithm for Constraints of Difference in CSPs", in Proceedings of 12th National Conference on Artificial Intelligence American Association for Artificial Intelligence, Seattle, Washington, 1994, pp. 362-367

[32] M. Mepham, *Jumbo Sudoku Challenge*, Slp ed., Liberty Street, 2006
[33] Puzzler Media Ltd, *Advanced Sudoku*, 1st ed., United Kingdom: Carlton Books, 2005
[34] W. Gould, *New York Post Sudoku 3: The Official Utterly Addictive Number-Placing Puzzle*, 1st ed., Collins, 2005
[35] G. Greenspan, R. Lee. (2005, June). *Web Sudoku*[online]. Available: https://www.websudoku.com/
[36] P. Prosser, K. Stergiou, T. Walsh, "Singleton consistencies", in Proceedings of the 6th International Conference on Principles and Practice of Constraint Programming, Singapore, 2000, pp. 353-368
[37] C. Reeson, K. Huang, K. Bayer, B. Choueiry, "An Interactive Constraint-Based Approach to Sudoku", in Proceedings of the 22th AAAI Conference on Artificial Intelligence, Vancouver, British Columbia, Canada, 2007, pp. 1976-1977
[38] I. Howell, R. Woodward, B. Choueiry, C. Bessiere, "Solving Sudoku with Consistency: A Visual and Interactive Approach", in Proceedings of the 27th International Joint Conference on Artificial Intelligence, Stockholm, Sweden, 2018, pp. 5829-5831
[39] A. Aho, *The Design and Analysis of Computer Algorithms*, 1st ed., Addison-Wesley, 1722
[40] R. Wilson, *Introduction to Graph Theory 4th Edition*, 4th ed., Addison Wesley, 1996
[41] R. Stuart, N. Peter, *Τεχνίτη Νοημοσύνη Μια σύγχρονη προσέγγιση*, 2nd American ed., Εκδόσεις Κλειδάριθμος, 2011