Ονοματεπώνυμο Α.Μ.	Ψαλλίδας Φώτης 1115200600170
Μάθημα	Τεχνητή Νοημοσύνη
Αντικείμενο Εργασίας	**2ο set Ασκήσεων**
Καθηγητής	κ. Μανώλης Κουμπαράκης
Ακαδημαϊκό Εξάμηνο	Χειμερινό 2008-2009

Ερώτηση 1

Να μοντελοποιήστε το Sudoku σαν ένα πρόβλημα ικανοποίησης περιορισμών.

Απάντηση

Ένα πρόβλημα ικανοποίησης περιορισμών ορίζεται αν ορίσουμε τις μεταβλητές το domain set της κάθε μεταβλητής καθώς και τα constraints που πρέπει να υπακούουν οι τιμές των μεταβλητών.

Σύνολο Μεταβλητών

Είναι το σύνολο των κελιών του αρχικού Sudoku που δεν έχουν πάρει τιμή ακόμη. Θα μπορούσαμε να θωρήσουμε ως μεταβλητές και όλες τι θέσεις του πίνακα αρκεί το domain set των μεταβλητών-κελιών που έχουν πάρει ήδη τιμή να αποτελείται μόνο από την τιμή που της έχει ήδη δοθεί.

Domain Set(σύνολο τιμών) της κάθε μεταβλητής.

Κάθε μεταβλητή μπορεί να πάρει τιμές από 1 έως και size*size(όπου size το μέγεθος του Sudoku). Θα μπορούσαμε εδώ να πούμε ότι το domain σετ μπορεί να μειωθεί αρχικά διαγράφοντας τις τιμές εκείνες που δεν είναι consistent με βάση τις constant τιμές που έχουν πάρει κελιά μέσα στο Sudoku.Ωστόσο μπορούμε να το αφήσουμε με αυτό το σκεπτικό ώστε να έχουμε έναν πιο γενικό ορισμό του csp.

Constraints

Σε κάθε γραμμή σε κάθε στήλη και σε κάθε υποτετράγωνο (όπως αυτό ορίζεται από τον ορισμό του προβλήματος του Sudoku) δεν επιτρέπεται δύο ή παραπάνω μεταβλητές να έχουν τις ίδιες τιμές.

Λύση του προβλήματος

Όλες οι μεταβλητές έχουν πάρει τιμή και ικανοποιείται ο παραπάνω περιορισμός.

Ερώτηση 2

Να λύσετε το πρόβλημα ικανοποίησης περιορισμών που ορίσατε με τους αλγόριθμους BT, BT+MRV και FC+MRV.
Μπορείτε να χρησιμοποιήσετε τον C++ κώδικα που δίνεται στην ιστοσελίδα του μαθήματος, να τον τροποποιήσετε ή να γράψετε το δικό σας κώδικα από την αρχή. Στις δύο τελευταίες περιπτώσεις έχετε τη δυνατότητα να πάρετε βαθμούς bonus ανάλογα με την ποιότητα της υλοποίησης σας.

Απάντηση

Τα αρχεία bt.cpp mrv_bt και fc_mrv.cpp(που αντιστοιχούν στις υλοποιήσεις των bt ,bt_mrv και fc_mrv αντίστοιχα) μπορείτε να τα βρείτε στον τρέχοντα κατάλογο (./sudoku).
Όλος ο σχολιασμός είναι εσωτερικός στα αρχεία αυτά.

Ερώτηση 3

Να συγκρίνετε πειραματικά τους αλγόριθμους που υλοποιήσατε χρησιμοποιώντας ικανό αριθμό προβλημάτων Sudoku με διάφορους βαθμούς δυσκολίας και ορίζοντας κατάλληλες μετρικές. Να παρουσιάσετε τα αποτελέσματα σας με ευκρίνεια χρησιμοποιώντας πίνακες και να τα σχολιάσετε.

Απάντηση

Παράθεση αποτελεσμάτων σε μορφή πινάκων για τη σύγκριση των τριών αλγορίθμων.
Ως μετρικές θεωρήθηκαν ο χρόνος και οι κλήσεις της αναδρομικής συνάρτησης που υλοποιεί το backtracking.Τα σχετικά inputs μπορείτε να τα βρείτε στον κατάλογο ./sudoku/input .
Επίσης στον κατάλογο ./sudoku/big-sudoku μπορείτε να βρείτε μεγάλα 16x16 sudoku(just in case: Ο bt_mrv και ο fc_mrv λυνουν τα kid-easy-medium-hard σε λίγοτερο από ένα λεπτό ενώ ο fc_mrv λύνει το evil μετά από αρκετή ώρα.Τα αποτελεσματα του big-sudoku βρίσκονται στο φάκελο output στο path /output/big-sudoku_results)

mine.txt			
	BT	BT_MRV	FC_MRV
Real time	0m0.004s	0m0.003s	0m0.002s
RecursiveBacktracking() Calls	9	9	9

easy.txt			
	BT	BT_MRV	FC_MRV
Real time	0m0.006s	0m0.003s	0m0.004s
RecursiveBacktracking() Calls	45	45	45

medium.txt			
	BT	BT_MRV	FC_MRV
Real time	0m0.003s	0m0.003s	0m0.003s
RecursiveBacktracking() Calls	1396	53	52

hard.txt			
	BT	BT_MRV	FC_MRV
Real time	0m0.113s	0m0.003s	0m0.007s
RecursiveBacktracking() Calls	82406	67	66

evil.txt			
	BT	BT_MRV	FC_MRV
Real time	0m0.008s	0m0.031s	0m0.012s
RecursiveBacktracking() Calls	199	577	530

Sudoku_ekfwnisis.txt			
	BT	BT_MRV	FC_MRV
Real time	0m0.005s	0m0.004s	0m0.005s
RecursiveBacktracking() Calls	218	99	96

ex1.txt			
	BT	BT_MRV	FC_MRV
Real time	0m0.023s	0m0.021s	0m0.007s
RecursiveBacktracking() Calls	10273	378	346

ex2.txt			
	BT	BT_MRV	FC_MRV
Real time	0m0.007s	0m0.004s	0m0.008s
RecursiveBacktracking() Calls	5036	116	109

16x16.txt			
	BT	BT_MRV	FC_MRV
Real time	0m18.639s	0m0.020s	0m0.014s
RecursiveBacktracking() Calls	9696503	372	360

Παρατηρούμε λοιπόν ότι ο αλγόριθμος bt είναι ο σχετικά λιγότερο έξυπνος.Έπεται ο bt_mrv που προσεγγίζει τον fc_mrv αλλά ποτέ δεν τον φτάνει ούτε χρονικά ούτε σε επίπεδο κλήσεις της αναδρομικής συνάρτησης υλοποίησης του backtracking.Παρατηρώντας λίγο καλύτερα μάλιστα στο πιο δύσκολο από τα Sudoku (16x16.txt) φαίνεται η πραγματική διαφορά μεταξύ του bt και των fc_mrv και bt_mrv.Με λίγα λόγια δηλαδή ο bt είναι καλύτερος από τους δύο υπόλοιπους όταν η είσοδος τον «ευνοεί» αλλά δεν έχει την ίδια σταθερότητα όταν η είσοδος είναι το ίδιο δύσκολη και για τους 3 αλγορίθμους. Τον bt έπεται ο mrv ο οποίος καταφέρνει να λύσει το σύνολο των δοκιμαστικών inputs με σχετική ευκολία αλλά όχι τόσο γρήγορα και όχι τόσο έξυπνα όσο τα καταφέρνει ο fc_mrv που φαίνεται να είναι ο καλύτερος όταν η δυσκολία επίλυσης είναι το ίδιο και για τους τρείς. Κάτι τέτοιο είναι απολύτως λογικό βέβαια

από τη στιγμή που ο mrv χρησιμοποιεί τον bt και προσπαθεί να τον βελτιώσει με τη χρήση της ευριστικής για την εύρεση της βέλτιστης μεταβλητής προς ανάθεση(assign) τιμής.Και ο fc χρησιμοποιεί την ευρυστική του mrv για την επιλογή της μεταβλητης και βελτιώνει τον bt τον οποίο χρησιμοποιεί ώστε να κάνει διάδοση περιορισμών στο δέντρο αναζήτησης να κόψει τις inconsistent τιμές από τα πεδία των αντίστοιχων μεταβλητών και να βρίσκει άμεσα από πού προήλθε το Inconsistency στο γράφο περιορισμών. Έτσι καταφέρνει και να μειώσει το χώρο αναζήτησης και να βρίσκει σχετικά γρήγορα από πού προήλθε το inconsistency.

Ερώτηση 4

Υποθέστε ότι υλοποιείτε και τον αλγόριθμο MAC και τον χρησιμοποιείτε για να λύσετε το ίδιο πρόβλημα. Να εξηγήσετε αναλυτικά (και με παραδείγματα) τι παραπάνω μπορεί να κάνει ο MAC από τους αλγόριθμους που υλοποιήσατε για το δοσμένο πρόβλημα αναζήτησης.

Απάντηση

Ο αλγόριθμος MAC είναι ένας αλγόριθμος διάδοσης περιορισμών και ελέγχου του consistency του γράφου.Ο αλγόριθμος αυτός ξεκιναέι από μία μεαβλητή (την μεταβλητή που γίνεται assign και διαδίδει τους περιορισμούς σε όλο το γράφο.
Στο συγκεκριμένο πρόβλημα ο αλγόριθμος MAC θα μπορούσε να χρησιμοποιηθεί ως εξής.Κάθε φορά που γίνεται ανάθεση μιας τιμής σε μια μεταβλητή πηγαίνει σε όλες τις μεταβλητές που συνδέονται με αυτήν την μεταβλητή και κόβει τις inconsistent τιμές από το πεδίο τιμών τους. Έπειτα αν κάποια από τις μεταβλητές αυτές έχει empty domain set τότε ανακάλυψε το inconsistency και γίνεται οπισθοδρόμηση.Αν μετά το κόψιμο κάποια μεταβλητή έχει μόνο μια τιμή μέσα στο πεδίο τιμών της τότε συνεχίζει και ελέγχει για το inconsistency μεταξύ αυτής της τιμής και των μεταβλητών που συνορεύουν με την μεταβλητή που έμεινε με ένα value στο domain set.Αυτό το πράγμα συνεχίζεται μέχρι να γίνει οριστικά consistent ο γράφος ή να βρεθεί inconsistency.Στην ουσία δηλαδή κάνει ότι και ο fc απλά συνεχίζει τη διάδοση και στις υπόλοιπες μεταβλητές.

Παράδειγμα όπου βοηθάει η χρήση του mac

X1	9	2	3	4	5	6	7	8
		9						
		4						
		5						
		6						
		7						
8	4	X3						
9	5	6						
X2	7	3						

Θεωρήστε το παραπάνω Sudoku μεγέθους 3 .Την συγκεκριμένη στιγμή
τα πεδία τιμών των μεταβλητών X1,X2,X3 είναι:
D(X1)={1}
D(X2)={1,2}
D(X3)={1}

Η X3 συνδέεται με τη X2 λόγω του υπο-τετραγώνου
ενώ η X2 συνδέεται με την X1 λόγω της στήλης.
Και οι 4 αλγόριθμοι θα τώρα δώσουν στην μεταβλητή X1 τη τιμή 1.
(ή για να είμαι πιο σαφής με βάση τη δική μου υλοποίηση και οι 4 αλγόριθμοι θα δώσουν στην μεταβλητή X1 την τιμή 1.
Τότε η μεταβλητή X2 θα έχει domain set {1}.Οι τρείς αλγόριθμοι(που εξετάσαμε) σε αυτό το σημείο θα προχωρούσαν για την εύρεση επόμενης μεταβλητής προς ανάθεση τιμής. Ο Mac όμως θέλοντας να κάνει το γράφο consistent θα έβλεπε διαδίδοντας τον περιορισμό του domain set της μεταβλητής X2,ότι η X3 έχει πλέον empty set.Άρα ο αλγόριθμος αυτός είναι στιγμιαία πιο έξυπνος από τους υπόλοιπους. Από τους bt_mrv και bt είναι όντως
πιο γρήγορος σε δύσκολες περιπτώσεις αφού μπορεί να διαδώσει περιορισμούς. Από τον fc_mrv όμως δεν είναι πιο γρήγορος.
Αυτό οφείλεται στο ότι ο fc_mrv και θα διαδώσει τον περιορισμό και θα βρει στην ακριβώς επόμενη κίνηση το inconsistency λόγω της χρήσης του mrv και θα τον βρει πιο γρήγορα αφού ο Mac για να διαδώσει περαιτέρω τον περιορισμό και να βρει το inconsistency πρέπει να βρίσκει μεταβλητές με μέγεθος πεδίου τιμών 1,το οποίο θα το κάνει και ο fc λόγω του mrv. Καθώς λοιπόν ο Mac θα σπαταλήσει αρκετή χρονική πολυπλοκότητα για να κάνει το γράφο συνεκτικό ο fc_mrv θα έχει βρει αυτό που ο Mac ψάχνει..
Σε σχέση με τον mrv τώρα ,υπό συνθήκες ο Mrv μπορεί
να γίνει πιο γρήγορος αλλά στη γενική περίπτωση δεν έχει το προτέρημα που προσφέρει η διάδοση περιορισμών(ελαχιστοποίηση του χώρου του προβλήματος και άμεση εύρεση inconsistency) συν την μεγάλη χρονική πολυπλοκότητα που θα χρειαστεί ο mrv για να βρει την μεταβλητή αυτή

Υποσημείωση: Η ευρετική mrv στον fc_mrv είναι πολύ πιο γρήγορη (O(n)) από αυτή στον bt_mrv αφού στον fc_mrv τα πεδία τιμών μειώνονται ενώ στον bt_mrv κάθε φορά ο mrv πρέπει και να υπολογίζει το μέγεθος των πεδίων τιμών των μεταβλητών).