# A

Αυτοματοποίησα όλη σχεδόν τη διαδικασία με ένα script σε μορφή αρχείου batch. Το script δίνει στο Μ ως αρχική τιμή το 5 και χρησιμοποιεί το εκτελέσιμο που παράγεται από το bcsp\_generate.c, για να παράγει 10 προβλήματα. Μετά εκτελεί τον hill, μέσω του εκτελέσιμου που παράγεται από το bcsp.c, για το κάθε πρόβλημα ξεχωριστά, ελέγχει το αποτέλεσμα με το εκτελέσιμο που παράγεται από το bcsp\_validate.c, γράφει για αρχή, το τρέχων Μ σε ένα αρχείο, και αν το αποτέλεσμα είναι σωστό, γράφει: σε ένα δεύτερο αρχείο, πάλι το Μ, τους χρόνους εκτέλεσης πέντε εκτελέσεων του hill, και μιας εκτέλεσης των depth, walksat & dpll (οι δυο τελευταίοι υλοποιήθηκαν για το ζητούμενο Β), και στο πρώτο αρχείο, 1, αλλιώς 0. Τα αρχεία bcsp\_generate.c, bcsp.c και bcsp\_validate.c έχουν τροποποιηθεί προσωρινά, για να δέχονται παραπάνω παραμέτρους, να επιστρέφουν συγκεκριμένα αποτελέσματα ή και τα 2, για να διευκολύνουν την διαδικασία αυτοματοποίησης. Μετά το Μ αυξάνεται κατά περίπου 40%. Αυτό επαναλαμβάνεται 16 φορές, δηλαδή, στο σύνολο, παράγονται τυχαία, 160 προβλήματα για 16 διαφορετικά Μ. Τέλος, τα 2 αρχεία που προέκυψαν, δίνονται σε ένα δικό μου εκτελέσιμο, το οποίο διαβάζει, τροποποιεί τα περιεχόμενά τους, και τα γράφει σε 3 διαφορετικά αρχεία. Μετά, χειροκίνητα, δίνονται στο εργαλείο gnuplot για να παραχθούν τα ζητούμενα διαγράμματα.

## Αποτελέσματα:

Όταν το Μ έφτασε να ισούται με 207, οι αλγόριθμοι σταμάτησαν να βρίσκουν πάντα λύσεις, με το ποσοστό πληρότητα να πέφτει στα 80%. Από όταν το Μ έφτασε να ισούται με 345 και μετά, σταμάτησαν να βρίσκουν ολοκληρωτικά λύσεις.

# Β

Ο επιπρόσθετος κώδικας εμπεριέχεται στο αρχείο bcsp.c, και αποτελείται από τις συναρτήσεις getUnSat, WalkSat, unit\_propagation & dpll

## WalkSAT

Αρχικοποιεί όλες τις μεταβλητές με τυχαίες τιμές και μετά, μέσω της getUnSat διαλέγει τυχαία μια μη ικανοποιημένη clause, δοκιμάζει να δώσει τιμές μία προς μία στις μεταβλητές της, και στο τέλος διαλέγει την αλλαγή που θα αυξήσει τον αριθμό των μη ικανοποιημένη clauses το λιγότερο. Αυτό επαναλαμβάνεται μέχρι να βρει λύση ή να λήξει το δοθέν χρονικό όριο.

## DPLL

Αρχικά αποθηκεύει έναν πίνακα, που αναπαριστά το σύνολο των μεταβλητών, σε ένα stack. Μέχρι το stack να αδειάσει, αφαιρεί από αυτό τον πίνακα αποθηκευμένο στην κορυφή του, τον αντιγράφει στον τρέχον πίνακα, αποθηκεύει τον πίνακα στο stack, προσπαθεί να εφαρμόσει **unit propagation** σε αυτόν, αν βρεθεί η λύση, τελειώνει, αλλιώς, ξανα-αφαιρεί από το stack τον πίνακα αποθηκευμένο στην κορυφή του, και τον αντιγράφει στον τρέχον πίνακα. Τώρα, δοκιμάζει να δώσει τις τιμές 1 & -1 στην πρώτη μεταβλητή που θα βρει που δεν έχει τιμή, και αν αυτό μπορεί να γίνει, τότε αποθηκεύει τον πίνακα που προέκυψε, από κάθε δοκιμή ξεχωριστά στην κορυφή του stack.

### Unit propagation

Θα διατρέξει κάθε clause, και για όσα δεν έχουν ικανοποιηθεί, αλλά ταυτόχρονα έχουν μόνο μια μεταβλητή που δεν της έχει δοθεί τιμή, θα της δώσει την κατάλληλη έτσι ώστε να ικανοποιηθεί το clause. Αυτό θα επαναλαμβάνεται μέχρι να μην υπάρχει clause που να τηρεί τις προαναφερθείσες συνθήκες.