Ατομική Διπλωματική Εργασία

ΔΥΝΑΜΙΚΗ ΚΑΙ ΔΟΜΗ ΣΥΣΤΗΜΑΤΩΝ ΣΥΜΠΕΡΑΣΜΑΤΟΛΟΓΙΑΣ

Ελένη Πρίγκηπα

ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΥΠΡΟΥ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Μάιος 2020

ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΥΠΡΟΥ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Δυναμική και Δομή Συστημάτων Συμπερασματολογίας

Ελένη Πρίγκηπα

Επιβλέπων Καθηγητής
Γιάννης Δημόπουλος

Η Ατομική Διπλωματική Εργασία υποβλήθηκε προς μερική εκπλήρωση των απαιτήσεων απόκτησης του πτυχίου Πληροφορικής του Τμήματος Πληροφορικής του Πανεπιστημίου Κύπρου

Μάιος 2020

Ευχαριστίες

Αρχικά, οφείλω να πω ένα μεγάλο ευχαριστώ στον επιβλέποντα καθηγητή της διπλωματικής μου κ. Γιάννη Δημόπουλο για την καθοδήγηση μου, την στήριξη του και την βοήθεια που έλαβα καθόλη τη διάρκεια της υλοποίησης της διπλωματικής εργασίας.

Περίληψη

Πλέον η επιχειρηματολογία αποτελεί κυρίαρχο στοιχείο όχι μόνο του κλάδου της Πληροφορικής αλλά και πολλών άλλων. Η Επιχειρηματολογία είναι η διαδικασία σκέψης και παραγωγής επιχειρημάτων για την υπεράσπιση κάποιων απόψεων, συμπεριφορών και καταστάσεων.

Η παρούσα διπλωματική έχει γενικότερο έχει στόχο την κατανόηση της δομής των μεγάλων και σύνθετων επιχειρημάτων. Για να γίνει αυτό θα ασχοληθούμε με το γενικότερο κομμάτι της επιχειρηματολογίας, δηλαδή τα επιχειρήματα που την αποτελούν και την συσχέτιση μεταξύ τους. Κάποια επιχειρήματα μπορεί να είναι αληθή και κάποια όχι. Εάν κάποιο επιχείρημα δεν έχει σύγκρουση με ένα άλλο τότε θεωρούμε ότι το επιχείρημα είναι αληθές, εάν όμως έχει τότε πρέπει αν γίνει μία ανάλυση για να δούμε ποια επιχειρήματα είναι αληθή και ποια όχι.

Θα ασχοληθούμε πιο πολύ με την Ευσταθή Επέκταση, όπου με κάποια κριτήρια βρίσκει τα ευσταθείς σύνολα, δηλαδή τα επιχειρήματα όπου είναι αληθής μπορούν να επιβιώσουν κάθε σύγκρουση καταρρίπτοντας αντίπαλα επιχειρήματα. Θα κατανοήσουμε το πως λειτουργεί και έτσι θα μπορέσουμε να ελέγξουμε το δίκτυο μας έτσι ώστε να κάνουμε τις απαραίτητες αλλαγές για να μας βγάλει τα επιθυμητά αποτελέσματα.

Αυτό θα γίνει μέσω πειραμάτων όπου θα χρησιμοποιηθούν διαφορετικά μοντέλα δημιουργίας τυχαίων γράφων, αλλάζοντας τις παραμέτρους με σκοπό να δούμε πως επηρεάζει το σύνολο της Ευσταθής Επέκτασης. Επιπλέον θα μελετήσουμε την σημασία και τον ρόλο του κάθε κόμβου/επιχειρήματος και πως συσχετίζεται με το σύνολο.

Περιεχόμενα

Κεφάλαιο 1

Εισαγωγή

1.1 Σκοπός Διπλωματικής

Σκοπός της παρούσας διπλωματικής είναι μπορέσουμε να κατανοήσουμε την δομή μεγάλων και σύνθετων επιχειρημάτων, την σημασία και τον ρόλο του κάθε κόμβου. Εάν φτάσουμε στο σημείο όπου θα μπορέσουμε να κατανοήσουμε τον ανθρώπινο εγκέφαλο και το πώς ακριβώς δουλεύει, θα μας ωφελήσει πολύ. Για παράδειγμα, στον τομέα της Υγείας, ή κατανόηση των σύνθετων δικτύων θα είχε μεγάλη αλλαγή, αφού αντιπροσωπεύουν βιομοριακά συστήματα όπου στόχο έχουν την κατανόηση της νόσους και πως αυτή εξαπλώνεται. Έτσι η πιθανότητα να βρεθούν θεραπείες για πολλές ασθένειες αυξάνεται, αφού θα γνωρίζουν που είναι το πρόβλημα και πως θα μπορούν να το πολεμήσουν.

Για να μπορέσουμε να κατανοήσουμε την δομή τους, θα ασχοληθούμε με την ανάλυση των θεωριών της επιχειρηματολογία. Η επιχειρηματολογία πλέον ανήκει σε πολλούς κλάδους όπως είναι η Πληροφορική, η Ιατρική, η Νομική και πολλοί ακόμα κλάδοι όπου σχετίζονται με ένα πρόβλημα όπου απαιτείται να γίνει η χρήση των επιχειρημάτων για την επίλυση του.

Στην συνέχεια, θα δοθούν πολλοί ορισμοί όπου θα βοηθήσουν στην κατανόηση του όρου επιχειρηματολογία. Μια προσέγγιση της επιχειρηματολογίας είναι, όπου με βάση κάποιον κριτηρίων γίνεται η επιλογή των επικρατέστερων επιχειρημάτων, αυτή η προσέγγιση είναι η Ευσταθής Επέκταση. Μέσω της ανάλυσης αυτής της θεωρίας, θα

κατανοήσουμε το πως λειτουργεί και πως μπορούμε να την ελέγξουμε για να μας βγάλει τα επιθυμητά αποτελέσματα.

1.2 Δομή Διπλωματικής

Στο δεύτερο κεφάλαιο, θα γίνει αναφορά στην θεωρία της επιχειρηματολογίας, όπου θα δοθούν πολλοί ορισμοί, παραδείγματα κατανόησης και προσεγγίσεις για την εύρεση των ισχυρών επιχειρημάτων. Επίσης με τον ορισμό και τα παραδείγματα κατανόησης θα βοηθήσουν στην κατανόηση της θεωρίας της Ευσταθούς Επέκτασης όπου η υλοποίηση της διπλωματικής στηρίζεται εκεί.

Στο τρίτο κεφάλαιο, θα οριστεί το τί είναι ένα Πρόβλημα Ικανοποίησης Περιορισμών, το τι είναι Δυναμικός Προγραμματισμός και πως μπορεί να λύσει τέτοια προγράμματα. Έπειτα με πολλά παραδείγματα θα βοηθήσουν στην κατανόηση της γλώσσας και της δομής ενός τέτοιου προγράμματος. Τέλος θα γίνει μία περιγραφή του επιλυτή Clingo όπου ασχολείται για την επίλυση τέτοιων προβλημάτων.

Στο τέταρτο κεφάλαιο, θα αναφερθεί στον τρόπο υλοποίησης της διπλωματικής όπου χωρίζεται σε τέσσερεις φάσεις. Στο πρώτο υποκεφάλαιο θα γίνει η περιγραφή της κάθε φάση και στο δεύτερο γίνει μια εξήγηση των στατιστικών που δημιουργήθηκαν και θα παρουσιαστούν τα αποτελέσαντα της κάθε φάσης σε μορφή πινάκων.

Στο πέμπτο κεφάλαιο, είναι η περιγραφή των αποτελεσμάτων του προηγούμενου κεφαλαίου και το γενικό πόρισμα που καταλήξαμε. Όπως επίσης περιέχει και κάποιες μελλοντικές επεκτάσεις των ιδεών που παρουσιάστηκαν στην παρούσα διπλωματική.

Κεφάλαιο 2

Επιχειρηματολογία

2.1 Τι είναι επιχειρηματολογία

Με τον όρο επιχειρηματολογία αναφερόμαστε στη διαδικασία σκέψης όπου χρησιμοποιείται για την ανάπτυξη και παρουσίαση επιχειρημάτων, όπου στόχο έχει να πείσουμε κάποιον για την ορθότητα των απόψεων μας. Επιπρόσθετα είναι η διεπιστημονική μελέτη του πως οι άνθρωποι μπορούν να φθάσουν σε συμπεράσματα μέσα από τη λογική σκέψη. Το να στηρίξει κάποιος την άποψη και την γνώμη του, πλέον κάποιοι το βλέπουν με κακό μάτι και αμέσως το συνδέουν με την διαμάχη μεταξύ ανθρώπων. Αλλά στην πραγματικότητα η επιχειρηματολογία αποτελεί ένα βασικό συστατικό της ανθρώπινης νοημοσύνης αφού έχουμε την ικανότητα να αναζητήσουμε λογικές αιτίες για μία κατάσταση, έτσι μπορούμε να απαλλαγούμε από προκαταλήψεις και από προκατασκευασμένες ιδέες.

Το κύριο συστατικό της επιχειρηματολογίας είναι το επιχείρημα. Ένα επιχείρημα είναι μια λίστα από δηλώσεις όπου μια από τις δηλώσεις είναι το συμπέρασμα και οι άλλες είναι οι προκείμενες του επιχειρήματος. Συλλογισμός είναι η λογική διαδικασία με την οποία καταλήγει κάποιος σε ένα συμπέρασμα με βάση τις προκείμενες τους επιχειρήματος του. Ένα επιχείρημα διακρίνεται σε τριών ειδών συλλογισμών ως προς το περιεχόμενο του.

Υπάρχουν τρείς κατηγορίες συλλογισμών ως προς το περιεχόμενο του επιχειρήματος:

1. Παραγωγικός Συλλογισμός
2. Επαγωγικός Συλλογισμός
3. Αναλογικός Συλλογισμός

1. Παραγωγικός Συλλογισμός

   Προκείμενη 1: Όλοι οι άνθρωποι είναι θνητοί.
   Προκείμενη 2: Ο Σωκράτης είναι άνθρωπος.
   Συμπέρασμα: Ο Σωκράτης είναι θνητός.

   Στον παραγωγικό συλλογισμό ξεκινούμε από κάτι γενικό και αφηρημένο και καταλήγουμε σε κάτι ειδικό και συγκεκριμένο.

2. Επαγωγικός Συλλογισμός

   Προκείμενη 1: Το σπουργίτι πετάει στον ουρανό.
   Προκείμενη 2: Το περιστέρι πετάει στον ουρανό.
   Προκείμενη 3: Ο γλάρος πετάει στον ουρανό.
   Προκείμενη 4: Το σπουργίτι, το περιστέρι, ο γλάρος είναι πουλιά.
   Συμπέρασμα: (όλα) τα πουλιά πετούν στον ουρανό.

   Στον επαγωγικό συλλογισμό ακολουθούμε πορεία αντίστροφη προς τον παραγωγικό: ξεκινούμε από το ειδικό και το συγκεκριμένο και καταλήγουμε στο γενικό και το αφηρημένο.

3. Αναλογικός Συλλογισμός

   Προκείμενη 1: Κάποιά καλή μαθήτρια βραβεύτηκε.
   Προκείμενη 2: Η Ελπίδα είναι καλή μαθήτρια.
   Συμπέρασμα: Η Ελπίδα είναι πιθανόν να βραβευτεί.

   Στον αναλογικό συλλογισμό από τα επιμέρους συμπεραίνουμε πάλι για τα επιμέρους. Ο αναλογικός συλλογισμός στην περίπτωση αυτή είναι συχνά πολύ ασθενής ως προς το βαθμό πιθανότητας, για το λόγο αυτό στην αναλογία πρέπει να καταλήγουμε με προσοχή σε ένα συμπέρασμα.

Τόσο το συμπέρασμα όσο και οι προκείμενες είναι προτάσεις που μπορούν να είναι αληθείς ή ψευδείς, ενώ ολόκληρο το επιχείρημα χαρακτηρίζεται ως έγκυρο ή άκυρο. Ένα επιχείρημα θεωρείται έγκυρο εάν το συμπέρασμα προκύπτει λογικά από τις προκείμενες. Οι προκείμενες και το συμπέρασμα θεωρούνται αληθή εάν ανταποκρίνονται στα δεδομένα της αντικειμενικής πραγματικότητας. Για να θεωρηθεί ένα επιχείρημα λογικά ορθό πρέπει να είναι συγχρόνως έγκυρο και οι προκείμενές του και το συμπέρασμα να είναι αληθείς.

Η επιχειρηματολογία πλέον αποτελεί ένα βασικό συστατικό και σε πολλά επαγγέλματα. Όπως για παράδειγμα είναι το επάγγελμα του δικηγόρου όπου συλλέγει όλες τις πληροφορίες που αφορούν την υπόθεση για να σχηματίσει τη δικογραφία και να διατυπώσει επιχειρήματα για την υπεράσπιση του πελάτη του. Για να αποδείξει την αθωότητα του πελάτη του, τα επιχειρήματα αυτά πρέπει να είναι ορθά. Ένα ακόμα παράδειγμα είναι και ο πωλητής. Για να θεωρηθεί κάποιος καλός πωλητής πρέπει να έχει την ικανότητα της πειθώ, δηλαδή με έγκυρα επιχειρήματα να μπορεί να προωθήσει το προϊόν σωστά στους πελάτες με σκοπό να τους το πωλήσει.

2.2 Αφηρημένο σύστημα επιχειρηματολογίας αλά Dung

Το αφηρημένο σύστημα επιχειρηματολογίας (argumentation frameworks, AF) προτάθηκε από τον επιστήμονα και καθηγητή της Πληροφορικής στο Τεχνολογικό Ινστιτούτο της Ασίας Phan Minh Dung [1]. Σύμφωνα με των Dung, ένα σύστημα επιχειρηματολογίας ορίζεται ως ένα ζεύγος <A,R>, όπου A είναι ένα σύνολο από επιχειρήματα και R είναι η δυαδική σχέση επίθεσης πάνω στο σύνολο A.

Το σύστημα επιχειρηματολογίας μπορεί να αναπαρασταθεί με την μορφή ενός κατευθυνόμενου γράφου, όπου οι κόμβοι θα αναπαριστούν τα επιχειρήματα και οι ακμές τις σχέσεις επίθεσης μεταξύ των επιχειρημάτων. Η ακμή κατευθύνεται με την φορά της επίθεσης δηλαδή, εξέρχεται από τον κόμβο/επιχείρημα που κάνει την επίθεση και εισέρχεται στον κόμβου/επιχείρημα που δέχεται την επίθεση.

Ένα από παράδειγμα αναπαράστασης ενός συστήματος επιχειρηματολογίας φαίνεται πιο κάτω.

Παράδειγμα Κατανόησης 2.1

Επιχείρημα Α: Η Έλενα θέλει να βγει έξω με τους φίλους της.

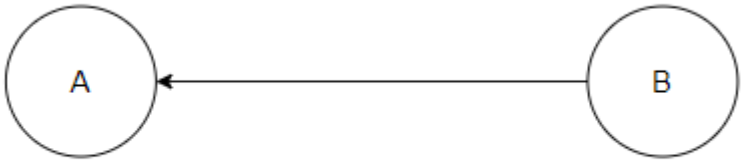
Επιχείρημα Β: Λόγω καραντίνας, απαγορεύονται οι συναθροίσεις.

$AF_{2.1<AR,>}$ = {A, B}, {B, A}

A = Σύνολο επιχειρημάτων = { A, B }

B = Δυαδικές σχέσεις επίθεσης = { (B, A) }

Στο πιο πάνω έχουμε δύο επιχειρήματα, όπου το επιχείρημα Β κάνει επίθεση στο Α, αφού λόγω της καραντίνας η Έλενα δεν θα μπορέσει να βγει με τους φίλους της. Άρα θα έχουμε δύο κόμβους και μία ακμή/επίθεση που γίνεται από το επιχείρημα Β στο επιχείρημα Α. (Σχήμα 2.1)



Σχήμα 2.1: Η αναπαράσταση του παραδείγματος κατανόησης 2.1 σε γράφο

Όπως μπορούμε να καταλάβουμε, εάν σε ένα σύστημα επιχειρηματολογίας υπάρχουν δυαδικές επιθέσεις μεταξύ των επιχειρημάτων σημαίνει ότι δεν γίνεται όλα τα επιχειρήματα να είναι αποδεκτά. Έτσι, υπάρχουν δύο τρόποι όπου μπορούμε να καθορίσουμε ποια επιχειρήματα είναι αποδεκτά και ποια όχι.

Οι δύο τρόποι είναι:

1. Η σημασιολογία με βάση τις ετικέτες (labeling-based)
2. Η σημασιολογία με βάση τις επεκτάσεις (extension-based)

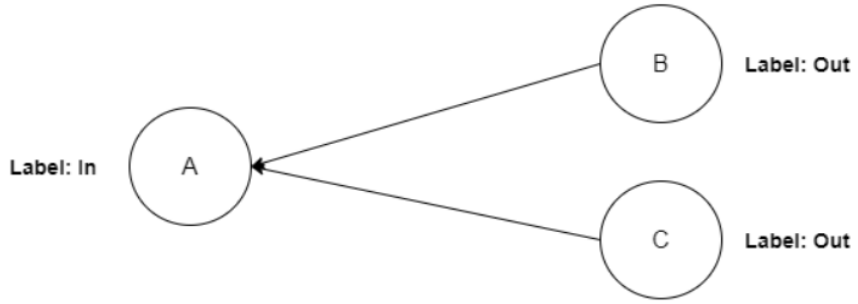
Τους δύο αυτούς τρόπους θα τους αναλύσουμε με λεπτομέρεια και παραδείγματα στα ακριβώς επόμενα δύο υποκεφάλαια.

2.3 Σημασιολογία με βάση τις ετικέτες

Σε αυτή την προσέγγιση, το κάθε επιχείρημα παίρνει μία ετικέτα. Υπάρχουν τρείς πιθανές ετικέτες οι οποίες είναι: in, out, undecided. Η ετικέτα in δηλώνει ότι το επιχείρημα είναι αποδεκτό, η ετικέτα out δηλώνει ότι το επιχείρημα δεν είναι αποδεκτό και η ετικέτα undecided δηλώνει ότι δεν ξέρουμε εάν το επιχείρημα είναι αποδεκτό ή όχι.

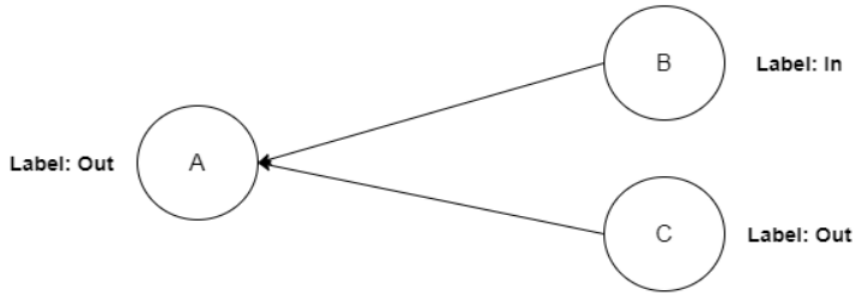
Η ετικέτα του κάθε επιχειρήματος μπορεί να καθοριστεί με τους εξής κανόνες:

- Ένα επιχείρημα παίρνει την ετικέτα in αν και μόνο αν όλα τα επιχείρημα που του κάνουν επίθεση έχουν την ετικέτα out.(Σχήμα 2.2)



Σχήμα 2.2: Το επιχείρημα A παίρνει την ετικέτα In

- Ένα επιχείρημα παίρνει την ετικέτα out αν και μόνο υπάρχει τουλάχιστον ένα επιχείρημα που του κάνει επίθεση έχει την ετικέτα in.(Σχήμα 2.3)



Σχήμα 2.3: Το επιχείρημα A παίρνει την ετικέτα Out

Αυστηρός ορισμός:

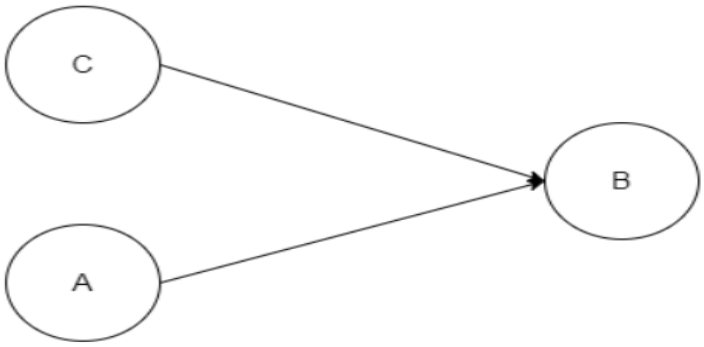
Έστω AF=(A,R) ένα αφηρημένο σύστημα επιχειρηματολογίας, όπου A είναι το σύνολο των επιχειρημάτων, R οι δυαδικές σχέσεις επίθεσης και Lab → {in, out, undecided}η συνάρτηση ανάθεσης ετικετών. Η ετικέτα του κάθε επιχειρήματος μπορεί να καθοριστεί με τους εξής κανόνες:[2]

- ∀a ∈ A: (Lab(a) = out ≡ ∃b ∈ A: (b def a ∧ Lab(b) = in))

- ∀a ∈ A: (Lab(a) = out ≡ ∃b ∈ A: (b def a ∧ Lab(b) = in))

Παράδειγμα Κατανόησης 2.2

Έστω ότι μας δίνεται η θεωρία AF=<{A, B, C},{(A, B), (C, B)}> της οποίας η αναπαράσταση φαίνεται στο πιο κάτω σχήμα. (Σχήμα 2.4)



Σχήμα 2.4: Υπολογισμός ετικετοποιήσεων της θεωρίας

Ετικετοποίηση:

L(A)= In, L(B)= Out, L(C)=In

Εξήγηση Ετικετοποίηση:

Αρχικά το επιχείρημα A έχει την ετικέτα In γιατί δεν του επιτίθεται κανένα άλλο επιχείρημα άρα αυτομάτως παίρνει την ετικέτα In . Για τον ίδιο λόγο και το επιχείρημα C παίρνει την ετικέτα In. Τέλος το επιχείρημα B παίρνει την ετικέτα Out γιατί υπάρχει τουλάχιστον ένα επιχείρημα που του επιτίθεται που έχει την ετικέτα In.

Παράδειγμα Κατανόησης 2.3

Έστω ότι μας δίνεται η θεωρία AF=<{A, B},{(A, B), (B, A)}> της οποίας η αναπαράσταση φαίνεται στο πιο κάτω σχήμα. (Σχήμα 2.5)

Σχήμα 2.5: Υπολογισμός ετικετοποιήσεων της θεωρίας

Ετικετοποίηση:

Σε αυτή την περίπτωση υπάρχουν τρείς διαφορετικές ετικετοποιήσεις:

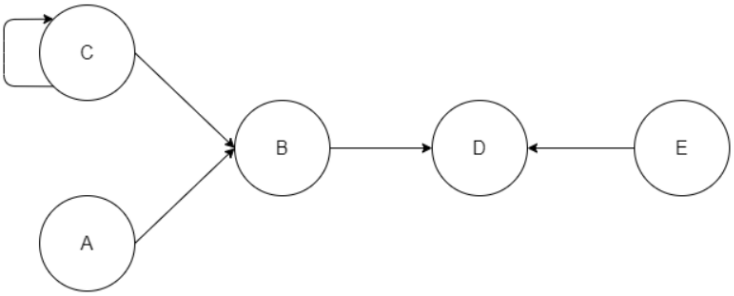
1. L(A)= In, L(B)= Out
2. L(A)= Out, L(B)= In
3. L(A)= Undecided, L(B)= Undecided

Εξήγηση Ετικετοποίηση:

Εδώ παρατηρούμε ότι το πιο πάνω παράδειγμα έχει περισσότερες από μία ανάθεση ετικέτας. Αυτό οφείλετε γιατί στο τι ετικέτα θα δώσουμε στο επιχείρημα A εξαρτάται από το τι ετικέτα θα δώσουμε στο επιχείρημα B και το αντίθετο. Έτσι έχουμε όλες τις πιθανές αναθέσεις.

Παράδειγμα Κατανόησης 2.4

Έστω ότι μας δίνεται η θεωρία AF=<{A, B, C, D, E},{(A, B), (B, D), (C, C), (C, B), (E,D)}> της οποίας η αναπαράσταση φαίνεται στο πιο κάτω σχήμα. (Σχήμα 2.6)



Σχήμα 2.6: Υπολογισμός ετικετοποιήσεων της θεωρίας

Ετικετοποίηση:

L(A)= In, L(B)= Out, L(C)=Undecided, L(D)= Out, L(E)= In.

Εξήγηση Ετικετοποίηση:

Αρχικά το επιχείρημα A και E έχει την ετικέτα In γιατί δεν του επιτίθεται κανένα άλλο επιχείρημα άρα αυτομάτως παίρνει την ετικέτα In . Έτσι το επιχείρημα B παίρνει την ετικέτα Out γιατί υπάρχει τουλάχιστον ένα επιχείρημα που του επιτίθεται που έχει την ετικέτα In (το επιχείρημα A). Για τον ίδιο λόγο και το επιχείρημα D παίρνει την ετικέτα Out λόγω του επιχειρήματος E που έχει την ετικέτα In. Για το επιχείρημα C δεν γνωρίζουμε εάν έχει την ετικέτα In ή Out έτσι θα πάρει την ετικέτα Undecided.

2.4 Σημασιολογία με βάση τις επεκτάσεις

Σε αυτή την προσέγγιση, η σημασιολογία με βάση τις επεκτάσεις ορίζει το σύνολο των επεκτάσεων από το σύστημα της επιχειρηματολογίας <A,R>, όπου A είναι τα επιχειρήματα και R οι δυαδικές επιθέσεις, το οποίο αντιπροσωπεύει ένα σύνολο επιχειρημάτων που ανήκουν στο A και μπορούν να επιβιώσουν από τις επιθέσεις άλλων επιχειρημάτων.

Σε αυτό το σημείο θα δώσουμε κάποιους ορισμούς που σχετίζονται με την σημασιολογία με βάση τις επεκτάσεις[1]

1. Απουσία Συγκρούσεων (Conflict-Free)
2. Αποδεκτό Επιχείρημα (Acceptable Argument)
3. Παραδεκτό Σύνολο (Admissible Set)

1. Απουσία Συγκρούσεων (Conflict-Free)

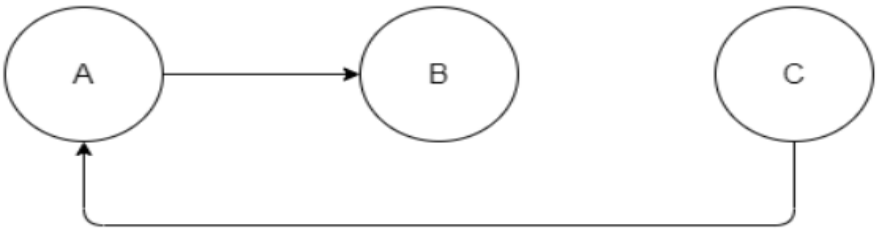
   Δεδομένου ενός συστήματος επιχειρηματολογίας AF=<A,R> , ένα σύνολο επιχειρημάτων S έχει την ιδιότητα απουσίας συγκρούσεων αν και μόνο δεν υπάρχει οποιαδήποτε σχέση επίθεσης μεταξύ των επιχειρημάτων που ανήκουν στο σύνολο αυτό.

   Αυστηρός Ορισμός:

   Δεδομένου ενός συστήματος επιχειρηματολογίας AF=<A,R>, ένα σύνολο S, όπου $S \subseteq A$ έχει την ιδιότητα απουσίας συγκρούσεων αν και μόνο αν $\nexists$ a, b $\in$ S τέτοιο ώστε $(a,b) \in R$. Ένα σύνολο ενός AF που έχει αυτή την ιδιότητα συμβολίζεται με cf.

Παράδειγμα Κατανόησης 2.5

Έστω ότι μας δίνεται η θεωρία AF=<{A, B, C},{(A, B), (C, A)}> της οποίας η αναπαράσταση φαίνεται στο πιο κάτω σχήμα. (Σχήμα 2.6)



Σχήμα 2.6: Η αναπαράσταση του παραδείγματος κατανόησης 2.5 σε γράφο

Εξέταση των πιθανών συνόλων S:

- **S = {A, B, C}**

  Το πιο πάνω σύνολο δεν έχει την ιδιότητα απουσίας συγκρούσεων αφού υπάρχει επίθεση από το επιχείρημα A στο B και από το C στο A.

- **S = {A, B}**

  Το πιο πάνω σύνολο δεν έχει την ιδιότητα απουσίας συγκρούσεων αφού υπάρχει επίθεση από το επιχείρημα A στο B.

- **S = {A, C}**

  Το πιο πάνω σύνολο δεν έχει την ιδιότητα απουσίας συγκρούσεων αφού υπάρχει επίθεση από το επιχείρημα C στο A.

- **S = {B, C}**

  Το πιο πάνω σύνολο έχει την ιδιότητα απουσίας συγκρούσεων αφού δεν υπάρχει επίθεση από το επιχείρημα B στο C ή το αντίθετο.

- **S = {A}**

  Το πιο πάνω σύνολο έχει την ιδιότητα απουσίας συγκρούσεων αφού δεν υπάρχει επίθεση από το επιχείρημα A στον εαυτό του.

- **S = {B}**

  Το πιο πάνω σύνολο έχει την ιδιότητα απουσίας συγκρούσεων αφού δεν υπάρχει επίθεση από το επιχείρημα B στον εαυτό του.

- **S = {C}**

  Το πιο πάνω σύνολο έχει την ιδιότητα απουσίας συγκρούσεων αφού δεν υπάρχει επίθεση από το επιχείρημα C στον εαυτό του.

2. Αποδεκτό Επιχείρημα (Acceptable Argument)

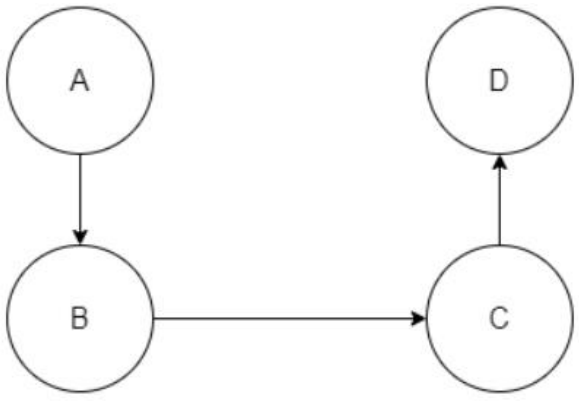
Δεδομένου ενός συστήματος επιχειρηματολογίας AF=<A,R> και ένα σύνολο επιχειρημάτων S, ένα επιχείρημα α, το οποίο ανήκει στο σύνολο A είναι αποδεκτό αν και μόνο αν για το κάθε επιχειρήμα που επιτίθεται στο α, υπάρχει κάποιο επιχείρημα από το σύνολο S που το επιτίθεται. Δηλαδή κάποιο επιχείρημα από το σύνολο S υπερασπίζεται το επιχείρημα α.

Αυστηρός Ορισμός:

Δεδομένου ενός συστήματος επιχειρηματολογίας AF=<A,R> και ένα σύνολο S, όπου $S \subseteq A$, ένα επιχείρημα a, όπου $a \in A$ είναι αποδεκτό σε σχέση με το S αν και μόνο αν $\forall\ b \in A$ όπου $(b,a) \in R$ τότε, $\exists\ c \in S$ όπου $(c,b) \in R$. Η συνάρτηση που με δεδομένο εισόδου ένα σύνολο S, όπου $S \subseteq A$ επιστρέφει το σύνολο των αποδεκτών επιχειρημάτων, λέγεται χαρακτηριστική συνάρτηση του AF και συμβολίζεται F_{AF}.

Παράδειγμα Κατανόησης 2.6

Έστω ότι μας δίνεται η θεωρία AF=<{A, B, C, D},{(A, B), (B, C), (C,D)}> και S=(B,D) της οποίας η αναπαράσταση φαίνεται στο πιο κάτω σχήμα. (Σχήμα 2.7)



Σχήμα 2.7: Η αναπαράσταση του παραδείγματος κατανόησης 2.6 σε γράφο

Εξέταση των πιθανών επιχειρημάτων:

- **B**

  Το επιχείρημα B δεν είναι αποδεκτό γιατί του επιτίθεται το A και δεν υπάρχει επιχείρημα στο σύνολο S που να επιτίθεται στο A.

- **D**

  Το επιχείρημα D είναι αποδεκτό γιατί του επιτίθεται το C και υπάρχει στο σύνολο S το επιχείρημα B που να επιτίθεται στο C.

3. Παραδεκτό Σύνολο (Admissible Set)

   Δεδομένου ενός συστήματος επιχειρηματολογίας AF=<A,R>, ένα σύνολο επιχειρημάτων S είναι παραδεκτό αν και μόνο αν το σύνολο έχει την ιδιότητα απουσίας συγκρούσεων και όλα τα επιχειρήματα του είναι αποδεκτά.

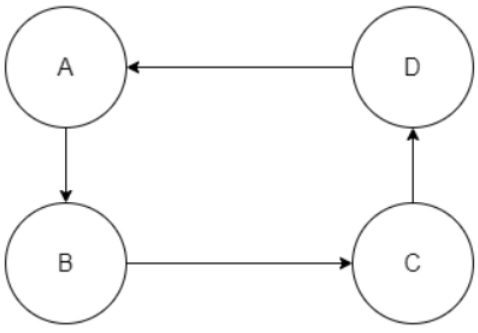
   Αυστηρός Ορισμός:

   Δεδομένου ενός συστήματος επιχειρηματολογίας AF=<A,R>, ένα σύνολο S, όπου $S \subseteq A$ είναι παραδεκτό αν και μόνο αν έχει την ιδιότητα απουσίας συγκρούσεων και $\forall$ a$\in$S το a είναι αποδεκτό όπου (b,a)$\in$R τότε, $\exists$ c$\in$S όπου (c,b)$\in$R.

   Το σύνολο με όλα τα παραδεκτά σύνολα του AF συμβολίζεται με AS(AF).

   Παράδειγμα Κατανόησης 2.7

   Έστω ότι μας δίνεται η θεωρία AF=<{A, B, C, D},{(A, B), (B, C), (C,D), (D,A)}> και S=(B,D) της οποίας η αναπαράσταση φαίνεται στο πιο κάτω σχήμα. (Σχήμα 2.8)



Σχήμα 2.8: Η αναπαράσταση του παραδείγματος κατανόησης 2.7 σε γράφο

Εξέταση αν το σύνολο S:

- **S = {B,D}**

  Το σύνολο S έχει την ιδιότητα απουσίας συγκρούσεων αφού δεν υπάρχει επίθεση από το επιχείρημα B στο D ή το αντίθετο.

Εξέταση επιχειρημάτων:

- **B**

  Το επιχείρημα B είναι αποδεκτό αφού το επιτίθεται το επιχείρημα A και υπάρχει στο σύνολο S το επιχείρημα D που να επιτίθεται στο A

- **D**

  Το επιχείρημα D είναι αποδεκτό αφού το επιτίθεται το επιχείρημα C και υπάρχει στο σύνολο S το επιχείρημα B που να επιτίθεται στο C

Άρα το σύνολο S είναι παραδεκτό.

<u>Σε αυτό το σημείο θα δούμε τέσσερεις βασικές επεκτάσεις:</u>

1. Πλήρης Επέκταση (Complete Extension)
2. Στηριγμένη Επέκταση (Grounded Extension)
3. Ευσταθής Επέκταση (Stable Extension)
4. Προτιμώμενη Επέκταση (Preferred Extension)

1. <u>Πλήρης Επέκταση (Complete Extension)</u>

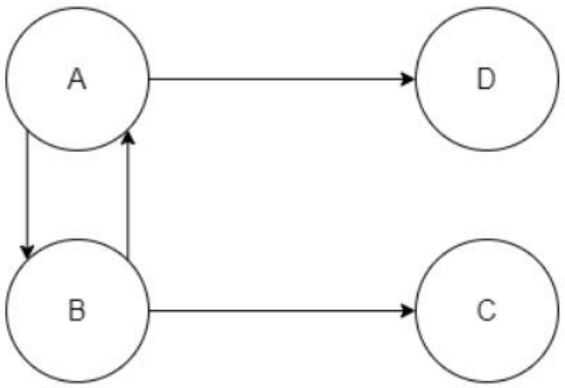
   Δεδομένου ενός συστήματος επιχειρηματολογίας AF=<A,R>, ένα σύνολο S, όπου $S \subseteq A$ είναι πλήρης επέκταση αν και μόνο αν το σύνολο S είναι παραδεκτό και κάθε επιχείρημα από το σύνολο A που είναι αποδεκτό ανήκει στο S.

   <u>Αυστηρός Ορισμός:</u>

   Δεδομένου ενός συστήματος επιχειρηματολογίας AF=<A,R>, ένα σύνολο S, όπου $S \subseteq A$ λέγεται πλήρης επέκταση αν και μόνο αν είναι παραδεκτό και $\forall a \in A$ όπου a είναι αποδεκτό επιχείρημα τότε, το $a \in S$. Το σύνολο των συνόλων που είναι πλήρης επέκταση συμβολίζεται με Eco.

Παράδειγμα Κατανόησης 2.8

Έστω ότι μας δίνεται η θεωρία AF=<{A, B, C, D},{(A, B), (A, D), (B, A), (B, C)}> και S=(B,D) της οποίας η αναπαράσταση φαίνεται στο πιο κάτω σχήμα. (Σχήμα 2.9)



Σχήμα 2.9: Η αναπαράσταση του παραδείγματος κατανόησης 2.8 σε γράφο

Οι πλήρης επεκτάσεις του πιο πάνω παραδείγματος είναι:

- **Eco= {Ø, {A, C}, {B, D}}**

  Το κενό σύνολο είναι παραδεκτό αφού δεν υπάρχει μέσα επιχείρημα που να υπερασπίζεται κάποιο άλλο. Το σύνολο {A, C} είναι παραδεκτό αφού και τα επιχειρήματα A και C είναι αποδεκτά και δεν υπάρχει σχέση επίθεσης μεταξύ τους. Επίσης είναι τα μόνα αποδεκτά επιχειρήματα στην θεωρία άρα το σύνολο είναι πλήρης επέκταση. Το ίδιο ισχύει και για το σύνολο {B, D}αντίστοιχα.

2. Στηριγμένη Επέκταση (Grounded Extension)

   Δεδομένου ενός συστήματος επιχειρηματολογίας AF=<A,R>, ένα σύνολο S, όπου $S \subseteq A$ είναι στηριγμένη επέκταση αν είναι το ελάχιστο σύνολο που είναι πλήρης επέκταση.

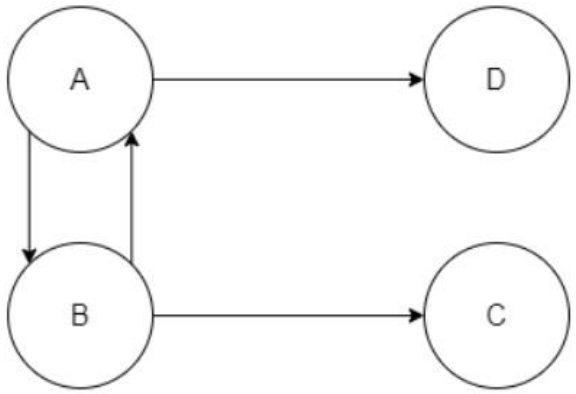
   Αυστηρός Ορισμός:

   Δεδομένου ενός συστήματος επιχειρηματολογίας AF=<A,R>, ένα σύνολο S, όπου $S \subseteq A$ λέγεται στηριγμένη επέκταση αν και μόνο αν είναι το ελάχιστο

παραδεκτό και ∀ a∈A όπου a είναι αποδεκτό επιχείρημα τότε, το a∈S. Το σύνολο που είναι στηριγμένη επέκταση συμβολίζεται με EGR.

Παράδειγμα Κατανόησης 2.9

Έστω ότι μας δίνεται η θεωρία AF=<{A, B, C, D},{(A, B), (A, D), (B, A), (B, C)}> και S=(B,D) της οποίας η αναπαράσταση φαίνεται στο πιο κάτω σχήμα. (Σχήμα 2.10)



Σχήμα 2.10: Η αναπαράσταση του παραδείγματος κατανόησης 2.9 σε γράφο

Η στηριγμένη επέκταση του πιο πάνω παραδείγματος είναι:

- **EGR= {∅}**

  Το κενό σύνολο είναι πλήρης επέκταση και περιέχει τα ελάχιστο επιχειρήματα.

3. Ευσταθής Επέκταση (Stable Extension)

   Δεδομένου ενός συστήματος επιχειρηματολογίας AF=<A,R>, ένα σύνολο S, όπου $S \subseteq A$ είναι ευσταθής επέκταση αν και μόνο αν έχει την ιδιότητα απουσίας συγκρούσεων και κάθε επιχείρημα που ανήκει στο A και δεν ανήκει στο σύνολο S , υπάρχει κάποιο επιχείρημα στο σύνολο S που επιτίθεται στο επιχείρημα αυτό.

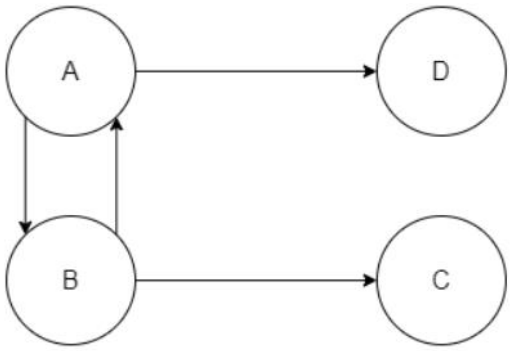
   Αυστηρός Ορισμός:

   Δεδομένου ενός συστήματος επιχειρηματολογίας AF=<A,R>, ένα σύνολο S, όπου $S \subseteq A$ είναι ευσταθής επέκταση αν και μόνο αν έχει την ιδιότητα απουσίας

συγκρούσεων και ∀ a∈A , a∉E τότε ∃ b∈S όπου (b,a)∈R. Το σύνολο των συνόλων που είναι ευσταθής επέκταση συμβολίζεται με Est.

Παράδειγμα Κατανόησης 2.10

Έστω ότι μας δίνεται η θεωρία AF=<{A, B, C, D},{(A, B), (A, D), (B, A), (B, C)}> και S=(B,D) της οποίας η αναπαράσταση φαίνεται στο πιο κάτω σχήμα. (Σχήμα 2.11)



Σχήμα 2.11: Η αναπαράσταση του παραδείγματος κατανόησης 2.10 σε γράφο

Οι ευσταθής επεκτάσεις του πιο πάνω παραδείγματος είναι:

- **Est = {{B, D}, {A, C}}**

  Το σύνολο {B, D} έχει την ιδιότητα απουσίας συγκρούσεων αφού δεν υπάρχει επίθεση από το επιχείρημα B στο D ή το αντίθετο. Τα επιχειρήματα A και C δεν ανήκουν στο σύνολο αλλά υπάρχει επίθεση από το B στο A και από το B στο C αντίστοιχα, έτσι ικανοποιούνται όλα τα επιχειρήματα.

  Το σύνολο {A, C} έχει την ιδιότητα απουσίας συγκρούσεων αφού δεν υπάρχει επίθεση από το επιχείρημα A στο C ή το αντίθετο. Τα επιχειρήματα B και D δεν ανήκουν στο σύνολο αλλά υπάρχει επίθεση από το A στο B και από το A στο D αντίστοιχα, έτσι ικανοποιούνται όλα τα επιχειρήματα.

4. Προτιμώμενη Επέκταση (Preferred Extension)

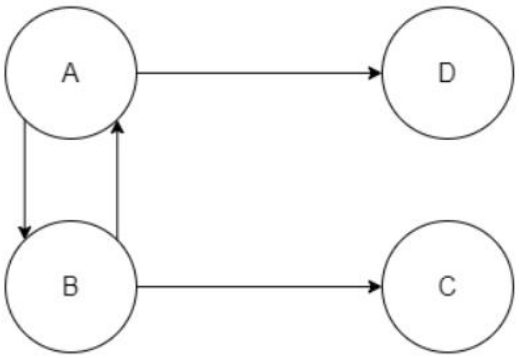
   Δεδομένου ενός συστήματος επιχειρηματολογίας AF=<A,R>, ένα σύνολο S, όπου $S \subseteq A$ είναι προτιμώμενη επέκταση αν και μόνο αν το σύνολο S είναι το μέγιστο παραδεκτό σύνολο.

Αυστηρός Ορισμός:

Δεδομένου ενός συστήματος επιχειρηματολογίας AF=<A,R>, ένα σύνολο S, όπου $S \subseteq A$ είναι προτιμώμενη επέκταση αν και μόνο αν είναι το μέγιστο στοιχείο του AS(AF). Το σύνολο που είναι προτιμώμενη επέκταση συμβολίζεται με E_{PR}.

Παράδειγμα Κατανόησης 2.11

Έστω ότι μας δίνεται η θεωρία AF=<{A, B, C, D},{(A, B), (A, D), (B, A), (B, C)}> και S=(B,D) της οποίας η αναπαράσταση φαίνεται στο πιο κάτω σχήμα. (Σχήμα 2.12)



Σχήμα 2.11: Η αναπαράσταση του παραδείγματος κατανόησης 2.12 σε γράφο

Η προτιμώμενη επέκταση του πιο πάνω παραδείγματος είναι:

- **E_{PR} = {B, D} ή το {A, C}**

  Το σύνολο {B, D} είναι παραδεκτό σύνολο αφού δεν υπάρχει επίθεση από το επιχείρημα B στο D ή το αντίθετο αρά έχει την ιδιότητα απουσίας συγκρούσεων. Επίσης το επιχειρήμα D είναι αποδεκτό γιατί του επιτίθεται το A και το B που ανήκει στο σύνολο επιτίθεται του A. Το επιχείρημα B είναι αποδεκτό γιατί του επιτίθεται το A αλλά το B επιτίθεται του A. Για τους αντίστοιχους λόγους και το σύνολο {A, C} είναι προτιμώμενη επέκταση.

Η σχέση μεταξύ των τεσσάρων βασικών επεκτάσεων που ορίστηκαν πιο πάνω είναι η εξής:

- Κάθε ευσταθής επέκταση είναι και προτιμώμενη.
- Κάθε προτιμώμενη επέκταση είναι και πλήρης.
- Κάθε στηριγμένη επέκταση είναι και πλήρης.

Κεφάλαιο 3

Προβλήματα Ικανοποίησης Περιορισμών

3.1 Προβλήματα Ικανοποίησης Περιορισμών

Ένας Από τους τομείς που ασχολείται η Τεχνητή Νοημοσύνη είναι η επίλυση προβλημάτων ικανοποίησης περιορισμών. Τα προβλήματα ικανοποίησης περιορισμών είναι μαθηματικά προβλήματα όπου μπορούμε να τα μετατρέψουμε σε ένα πρόγραμμα με μεταβλητές και περιορισμούς.

Ένα πρόβλημα ικανοποίησης περιορισμών ορίζεται ως μία τριπλέτα <X,D,C> όπου:

- Ένα σύνολο μεταβλητών X_1, $X_{2,}$….., $X_{n.}$
- Για κάθε μεταβλητή X_i υπάρχει ένα πεδίο ορισμού Di με τις πιθανές τιμές που μπορεί να πάρει.
- Ένα σύνολο από περιορισμούς C_1, $C_{2,}$….., $C_{m.}$ Ένας περιορισμός καθορίζει τους επιτρεπτούς συνδυασμούς τιμών που μπορούν να πάρουν οι μεταβλητές που εμφανίζονται σε αυτόν τον περιορισμό.

Υπάρχουν τρία είδη περιορισμών με βάση την πληθικότητα των μεταβλητών στον περιορισμό:

- <u>Μοναδιαίοι Περιορισμοί:</u> Οι περιορισμοί που περιλαμβάνουν μόνο μία μεταβλητή. Πχ. X>5
- <u>Δυαδικοί Περιορισμοί:</u> Οι περιορισμοί που περιλαμβάνουν δύο μεταβλητές. Πχ. X+Y>10

- Ανώτερης Τάξης Περιορισμοί: Οι περιορισμοί που περιλαμβάνουν τρείς ή περισσότερες μεταβλητές.
  Πχ. X+Y+2*Z+12*W=4

Υπάρχουν δύο είδη περιορισμών με βάση την σημαντικότητα τους:

- Απόλυτοι Περιορισμοί: Οι περιορισμοί όπου η ικανοποίηση τους είναι υποχρεωτική.

- Ελαστικοί Περιορισμοί: Οι περιορισμοί όπου η ικανοποίηση τους είναι προαιρετική.

Αφού γίνει αυτή η μετατροπή, δίνουμε το πρόγραμμα σε ένα επιλυτή όπου θα γίνει η ανάθεση τιμών στις μεταβλητές με βάση τους περιορισμούς που δώσαμε. Μια κατάσταση του προβλήματος ορίζεται ως η ανάθεση τιμών σε κάποιες από τις μεταβλητές. Εάν η ανάθεση τιμών στις μεταβλητές δεν παραβιάζει κανένα περιορισμό τότε λέγεται συνεπής και όταν περιλαμβάνει όλες τις μεταβλητές του προβλήματος τότε λέγεται πλήρης. Έτσι για να είναι ένα πρόβλημα ικανοποιήσιμο πρέπει να είναι και συνεπής και πλήρης.

Επιπρόσθετα το κάθε πρόβλημα ικανοποίησης περιορισμών μπορεί να αναπαρασταθεί και με ένα γράφο, όπου οι κόμβοι αναπαριστούν τις μεταβλητές και οι ακμές τους περιορισμούς μεταξύ των μεταβλητών.

Παράδειγμα Κατανόησης 3.1:

Ένα γνωστό πρόβλημα ικανοποίησης περιορισμών είναι το πρόβλημα χρωματισμού γράφου. Δίνετε ένας προκαθορισμένος χάρτης όπου σκοπός είναι ο χρωματισμός των διάφορων περιοχών με κάποια συγκεκριμένα χρώματα και ο μόνος περιορισμός είναι οι γειτονικές περιοχές να μην έχουν ίδιο χρώμα. Για αυτό το παράδειγμα θα χρησιμοποιήσουμε τον πιο κάτω χάρτη (Σχήμα3.1).

Σχήμα 3.1 Το πρόβλημα χρωματισμού χάρτη

Μετατροπή σε Πρόβλημα Ικανοποίησης Περιορισμών:

Για κάθε περιοχή ορίζουμε μία μεταβλητη:

Western Australia: ***WA****, Northern Territory:* ***NT****, Queensland:* ***Q***

South Australia: ***SA****, New South Wales:* ***NSW****, Victoria:* ***V****, Tasmania:* ***T***

Κάθε μεταβλητή έχει ως πεδίο τιμών το σύνολο ***{Κίτρινο, Κόκκινο, Μπλε}***

Ο μόνος περιορισμός που έχουμε είναι οι γειτονικές χώρες να έχουν διαφορετικό χρώμα. Αυτό ορίζεται με δυαδικούς περιορισμούς ως εξής:

$$WA \neq NT \wedge WA \neq SA \wedge NT \neq Q \wedge NT \neq SA \wedge Q \neq NSW \wedge$$

$$Q \neq SA \wedge NSW \neq V \wedge NSW \neq SA \wedge V \neq SA$$

3.2 Λογικός Προγραμματισμός

Υπάρχουν δύο μεθοδολογίες αντιμετώπισης προβλημάτων, ο διαδικαστικός προγραμματισμός και ο δηλωτικός προγραμματισμός. Με τον διαδικαστικό προγραμματισμό η επίλυση του προβλήματος γίνεται με το να διατυπώσουμε τα βήματα του αλγορίθμου, δηλαδή να περιγράψουμε το πώς θα λυθεί το πρόβλημα. Ενώ με τον δηλωτικό προγραμματισμό η επίλυση του προβλήματος γίνεται με το να διατυπώσουμε τα αξιώματα, όπου περιγράφουν το τι ισχύει στον κόσμο του προβλήματος, ενώ διάφοροι επιλυτές αναλαμβάνουν την επίλυση του. Μία μορφή δηλωτικού προγραμματισμού είναι ο λογικός προγραμματισμός όπου χρησιμοποιείται για την επίλυση Προβλήματα Ικανοποίησης Περιορισμών. Η παρούσα διπλωματική ασχολήθηκε με τον Προγραμματισμός Συνόλου Απαντήσεων (Answer Set

Programming (ASP))[3], όπου είναι βασισμένος στον λογικό προγραμματισμό και στην σημασιολογία με βάση την ευσταθούς επέκταση όπου ορίστηκε στο κεφάλαιο 2. Χρησιμοποιείτε συνήθως σε προβλήματα που αποτελούνται από πολλές αποφάσεις, περιορισμούς και σε προβλήματα βελτιστοποίησης.

3.3 Σύνταξη Λογικού Προγραμματισμού

Ένα λογικό πρόγραμμα P πάνω σε ένα σύνολο A από άτομα είναι ένα πεπερασμένο σύνολο από κανόνες. Ένας κανόνας r ορίζεται ως [3]:
$a_0 \leftarrow a_1, \ldots, a_m, \sim a_{m+1}, \ldots, \sim a_n$ όπου $0 \leq m \leq n$ και τα $a_i \in A$, $0 \leq i \leq n$.

- Όλα τα a είναι άτομα από το σύνολο A
- Το a_0 ορίζεται ως η κεφαλή του κανόνα και το σύνολο $\{ a_1, \ldots, a_m, \sim a_{m+1}, \ldots, \sim a_n \}$ ως το σώμα του κανόνα.
- Αυτοί οι κανόνες που υπάρχει μόνο η κεφαλή και λείπει το σώμα λέγονται γεγονότα που αντιπροσωπεύουν τα δεδομένα μας.

Για να κατανοήσουμε πλήρως το νόημα του κανόνα θα πρέπει πρώτα να δούμε κάποιους βασικούς κανόνες σύνταξης:

- Σύζευξη: “,” (∧)
- Διάζευξη: “;” (∨)
- Συνεπαγωγή: “:-” (←)
- Απλή Άρνηση: “not” (~)
- Κλασική Άρνηση: “-” (¬)
- Οι σταθερές, τα ονόματα συναρτησιακών συμβόλων και των κατηγορημάτων ξεκινούν πάντα με πεζό γράμμα.
- Οι μεταβλητές ξεκινούν πάντα με κεφαλαίο γράμμα.
- Η κάθε λογική πρόταση τελειώνει με τελεία.

- Περιορισμοί Ακεραιότητας

  Σκοπό έχει να εξαλείψει ανεπιθύμητες λύσεις που μπορεί ο επιλυτής να δώσει.

  Αυστηρός Ορισμός:

  $\leftarrow a_0, \ldots, a_m, \sim a_{m+1}, \ldots, \sim a_n$

  όπου $0 \leq m \leq n$ και κάθε a_i είναι ένα άτομο όπου $0 \leq i \leq n$

  Παράδειγμα: :- *at_ supermarket(X), at_ bakery(X)* .

  Εξήγηση: Δεν γίνεται ο X να βρίσκεται και στην υπεραγορά και στον φούρνο ταυτόχρονα.

- Κανόνας Επιλογής

  Σκοπός είναι η επιλογή από ένα υποσύνολο.

  Αυστηρός Ορισμός:

  $\{a_0, \ldots, a_m\} \leftarrow a_{m+1}, \ldots, a_n, \sim a_{n+1}, \ldots, \sim a_o$

  όπου $0 \leq m \leq n \leq o$ και κάθε a_i είναι ένα άτομο όπου $0 \leq i \leq o$

  Παράδειγμα: *{ buy(Wine); buy (Water); buy (Juice)} :- at_ supermarket (X).*

  Εξήγηση: Εάν είναι στην υπεραγορά τότε μπορεί να επιλέξει να πάρει από το σύνολο (κρασί, νερό, χυμό) από 0 μέχρι 3 αντικείμενα.

- Κανόνας Πληθικότητας με Κατώτατο Όριο:

  Σκοπός έχει να ελέγχει την πληθικότητα ενός υποσυνόλου με το να δίνει ένα κατώτατο όριο επιλογής.

  Αυστηρός Ορισμός:

  $a_0 \leftarrow l\{a_{1+1}, \ldots, a_m, \sim a_{m+1}, \ldots, \sim a_n\}$

  όπου $0 \leq m \leq n$ και κάθε a_i είναι ένα άτομο όπου $0 \leq i \leq n$,k ένας θετικός ακέραιος αριθμός.

Παράδειγμα: *final_exam(X):-1{submit(as1); submit(as2); submit(as2)} ,student(X).*

Εξήγηση: Εάν ο Χ είναι μαθητής και παρέδωσε τουλάχιστον 1 εργασία από τις 3 τότε μπορεί να δώσει τελική εξέταση.

- Κανόνας Πληθικότητας με Ανώτατο Όριο:

  Σκοπός έχει να ελέγχει την πληθικότητα ενός υποσυνόλου με το να δίνει ένα ανώτατο όριο επιλογής. Το κατώτατο και το ανώτατο όριο είναι προαιρετικά.

  Αυστηρός Ορισμός:

  $a_0 \leftarrow l\{a_{1+1}, \ldots, a_m, \sim a_{m+1}, \ldots, \sim a_n\}\ u$

  όπου $0 \leq m \leq n$ και κάθε a_i είναι ένα άτομο όπου $0 \leq i \leq n$, l και u θετικοί ακέραιοι αριθμοί.

  Παράδειγμα: *final_exam(X):-1{submit(as1);submit(as2); submit(as3)} 2 ,student(X).*

  Εξήγηση: Εάν ο Χ είναι μαθητής και παρέδωσε τουλάχιστον 1 και το πολύ 2 εργασίες τότε μπορεί να δώσει τελική εξέταση.

- Κανόνας Πληθικότητας στην Κεφαλή:

  Σκοπός έχει να ελέγχει την πληθικότητα ενός υποσυνόλου με το να δίνει ένα κατώτατο ή και ανώτατο όριο επιλογής στην κεφαλή τους κανόνα. Το κατώτατο και το ανώτατο όριο είναι προαιρετικά.

  Αυστηρός Ορισμός:

  $l\{a_0, \ldots, a_m, \sim a_{m+1}, \ldots, \sim a_n\}\ u \leftarrow a_{n+1}, \ldots, a_o, \sim a_{o+1}, \ldots, \sim a_p$

  όπου $0 \leq m \leq n \leq o \leq p$ και κάθε a_i είναι ένα άτομο όπου $0 \leq i \leq p$, l και u θετικοί ακέραιοι αριθμοί.

  Παράδειγμα: *1{ buy(wine); buy (water); buy (juice)}2 :- at_ supermarket (X).*

  Εξήγηση: Εάν είναι στην υπεραγορά τότε μπορεί να επιλέξει να πάρει από το σύνολο (κρασί, νερό, χυμό) 1 ή 2 αντικείμενα.

- Περιορισμοί με Βάρη:

  Σκοπός έχει να ελέγχει την πληθικότητα ενός υποσυνόλου με το να δίνει ένα κατώτατο ή και ανώτατο όριο επιλογής στην κεφαλή τους κανόνα. Το κατώτατο και το ανώτατο όριο είναι προαιρετικά.

  Αυστηρός Ορισμός:

  $l \{w_1 : a_0, \ldots, a_m, w_{m+1} : \sim a_{m+1}, \ldots, w_n : \sim a_n\} u$

  όπου $0 \leq m \leq n$ και κάθε a_i είναι ένα άτομο όπου $0 \leq i \leq n$, l και u θετικοί ακέραιοι αριθμοί.

  Παράδειγμα: *4{ 4:buy(wine); 1:buy (water); 3:buy (juice)} 7*
  Εξήγηση: Μπορεί να επιλέξει να πάρει από το σύνολο (κρασί, νερό, χυμό) τα αντικείμενα όπου τα βάρη στο σύνολο είναι τουλάχιστον 4 και το πολύ 7.

- Άρνηση:

  Υπάρχουν δύο τρόποι άρνησης, η απλή άρνηση "not" και η κλασσική άρνηση "-" . Η απλή άρνηση δηλαδή το *not a* είναι αληθές εάν δεν υπάρχει κάποιος κανόνας που να κάνει το *a* αληθές. Επίσης η απλή άρνηση αντιστοιχεί στο σύμβολο "~", όπου βρίσκεται στους πιο πάνω ορισμούς. Στην κλασσική άρνηση το *-α* είναι αληθές εάν υπάρχει κανόνας που να κάνει το *-α* αληθές. Σημασιολογικά, το *-α* είναι ένα καινούργιο άτομο όπου δεν γίνεται το *-α* και το *α* να είναι αληθές.

  Παράδειγμα: *cross :- not train.*
  Stable Models: {cross}

  Εξήγηση: Για να είναι το άτομο *cross* αληθές πρέπει το σώμα του κανόνα αυτού δηλαδή το *not train* να είναι αληθές. Άρα αφού δεν υπάρχει κάποιος κανόνας που να κάνει το *train* αληθές τότε υπερισχύει το *not train* και έτσι το *cross* είναι αληθές.

  Παράδειγμα: *cross :- -train.*
  Stable Models: {∅}

Εξήγηση: Για να είναι το άτομο *cross* αληθές πρέπει το σώμα του κανόνα αυτού δηλαδή το *-train* να είναι αληθές. Όμως δεν υπάρχει κάποιος κανόνας που να κάνει το *-train* αληθές. Άρα το cross δεν είναι αληθές.

3.4 Επιλυτής Clingo

Το Πανεπιστήμιο του Potsdam ανάπτυξε ένα σύνολο από εργαλεία για τον Προγραμματισμός Συνόλου Απαντήσεων (ASP), το Potassco (Potsdam Answer Set Solving Collection). Τα εργαλεία που ανάπτυξαν είναι gringo, clasp, clingo, iclingo. Το πρώτο εργαλείο είναι το gringo όπου μεταφράζει τα λογικά προγράμματα όπου δίνει ο χρήστης σε προτασιακά λογικά προβλήματα.Το clasp είναι ο επιλυτής που βρίσκει λύση στα προγράμμα που μεταφράζει το gringo. Το εργαλείο clingo ενσωματώνει τις λειτουργίες των gringo και clasp. Τελευταίο είναι το εργαλείο iclingo όπου είναι επέκταση του clingo.Το εργαλείο που θα χρησιμοποιηθεί στην παρούσα διπλωματική είναι το clingo.[4]

Γλωσσική Δομή

Η σύνταξη είναι αρκετά παρόμοια με αυτή του λογικού προγραμματισμού δηλαδή
Πάλι έχουμε τα γεγονότα $(A_0.)$, τους κανόνες $(A_0 :- L_1, \ldots, L_n.)$ και τους περιορισμούς ακεραιότητας $(:- L_1, \ldots, L_n.)$ όπου A_0 είναι η κεφαλή του κανόνα και το σύνολο $L_1, \ldots, L_n$ είναι το σώμα του κανόνα. Η κεφαλή του κανόνα είναι αληθής εάν το σώμα είναι αληθές, όπου το άτομο Li για να είναι αληθές πρέπει να υπάρχει ένας κάποιος κανόνας που παράγεται ότι το Li είναι αληθής.

Παράδειγμα Κατανόησης 3.2:

Δίνεται το πιο κάτω πρόγραμμα:

a :- b.

b :- a.

Το πιο πάνω πρόγραμμα με απλά λόγια λέει ότι το a είναι αληθές εάν είναι το b και το b θα είναι αληθές εάν είναι το a. Άρα το σύνολο των απαντήσεων θα είναι κενό γιατί δεν υπάρχει κανόνας που να παράγεται ότι το a ή το b είναι αληθή. Εάν προσθέσουμε τον κανόνα (a.) ή και το (b.) τότε το σύνολο των απαντήσεων θα είναι {a, b}.

Παράδειγμα Κατανόησης 3.3

Δίνεται το πιο κάτω πρόγραμμα:

a :- not b.

b :- not a.

Το πιο πάνω πρόγραμμα με απλά λόγια λέει ότι το a είναι αληθές εάν είναι το b δεν είναι και το b θα είναι αληθές εάν είναι το a δεν είναι. Άρα το σύνολο των απαντήσεων θα είναι το {{a}, {b}}, όπου έχει 2 λύσεις γιατί μόνο ένα από τα 2 άτομα μπορεί να είναι αληθείς.

Σε αυτό το σημείο θα αναφέρουμε πιο συγκεκριμένες συντάξεις της γλώσσας εισόδου του εργαλείου Clingo.

- Αριθμητικές Συναρτήσεις

  Υπάρχουν διάφορες αριθμητικές πράξεις που μπορούμε να κάνουμε όπως πρόσθεση (+), αφαίρεση (-), πολλαπλασιασμός (*), ακέραια διαίρεση (/), υπόλοιπο διαίρεσης (\), απόλυτη τιμή (| |), δύναμη (**) και δυαδικό AND (&).

  Παράδειγμα Κατανόησης 3.4

  Δεδομένου το πιο κάτω πρόγραμμα:

```
1  left  (7).
2  right (2).
3  plus  (L + R )  :- left(L), right(R). %Πρόσθεση
4  minus (L - R )  :- left(L), right(R). %Αφαίρεση
5  uminus(- R)     :- right(R).          %Unary Minus
6  times (L * R )  :- left(L), right(R). %Πολλαπλασιασμός
7  divide(L / R )  :- left(L), right(R). %Ακέραια Αφαίρεση
8  modulo(L \ R )  :- left(L), right(R). %Υπόλοιπο Διαίρεσης
9  abs   (|-R|)    :- right(R).          %Απόλυτη Τιμή
10 power (L ** R ) :- left(L), right(R). %Δύναμη
11 bitand(L & R )  :- left(L), right(R). %Δυαδικό AND
```

  Το σύνολο απαντήσεων του πιο πάνω προγράμματος περιέχει:

```
right(2) left(7) plus(9)
minus(5) unary_minus(-2)
times(14) divide(3) modulo(1)
absolute(2) power(49) bitand(2)
```

- Συναρτήσεις Συγκρίσεις

  Υπάρχουν διάφορες συγκρίσεις που μπορούμε να κάνουμε όπως ισότητα (==), ανισότητα (!=), μικρότερο (<), μικρότερο ή ίσο (<=), μεγαλύτερο (>) και μεγαλύτερο ή ίσο (>=).

  Παράδειγμα Κατανόησης 3.5

  Δεδομένου το πιο κάτω πρόγραμμα:

```
1  num(1).
2  num(2).
3  eq (X,Y) :- X == Y,    num(X), num(Y). %Ισότητα
4  neq(X,Y) :- X != Y,    num(X), num(Y). %Ανισότητα
5  lt (X,Y) :- X < Y,     num(X), num(Y). %Μικρότερο
6  leq(X,Y) :- X <= Y,    num(X), num(Y). %Μικρότερο ή ίσο
7  gt (X,Y) :- X > Y,     num(X), num(Y). %Μεγαλύτερο
8  geq(X,Y) :- X >= Y,    num(X), num(Y). %Μεγαλύτερο ή ίσο
```

  Το σύνολο απαντήσεων του πιο πάνω προγράμματος περιέχει:

```
num(1) num(2)
eq(1,1) eq(2,2) neq(2,1) neq(1,2)
lt(1,2) leq(1,1) leq(1,2) leq(2,2)
gt(2,1) geq(1,1) geq(2,1) geq(2,2)
```

- Διαστήματα

  Εάν για παράδειγμα έχουμε N γεγονότα *fact(k)* όπου είναι συνεχόμενοι k ακέραια αριθμοί από το i στο j τότε μπορούμε να το απλοποιήσουμε με το διάστημα fact(i..j).

  Παράδειγμα Κατανόησης 3.6

  Εάν έχουμε τα πιο κάτω γεγονότα:

  num(1). num(2). num(3). num(4). num(5). num(6).

  Τότε μπορούμε να τα αντικαταστήσουμε με το *num(1..6)*

  Παράδειγμα Κατανόησης 3.7

  Εάν έχουμε τα πιο κάτω γεγονότα:

  pairs(1,4) pairs(2,4) pairs(3,4) pairs(1,5) pairs(2,5) pairs(3,5)

  Τότε μπορούμε να τα αντικαταστήσουμε με *pairs(1..3,4..5)*

- Συνθήκη

  Οι συνθήκες επιτρέπουν την παρουσίαση μεταβλητών σε συλλογές όρων με ένα μόνο κανόνα. Χρησιμοποιείται συνήθως για την κωδικοποίηση διάζευξης. Το σύμβολο για την συνθήκη είναι το ":".

  Παράδειγμα Κατανόησης 3.7

  Δεδομένου το πιο κάτω πρόγραμμα:

```
1  person(jane). person(john).
2  day(mon). day(tue). day(wed). day(thu). day(fri).
3  available(jane) :- not on(fri).
4  available(john) :- not on(mon), not on(wed).
5  meet :- available(X) : person(X).
6  on(X) : day(X) :- meet.
```

  Το σύνολο απαντήσεων του πιο πάνω προγράμματος περιέχει:

```
Answer: 1
on(thu)
Answer: 2
on(tue)
```

  Ο περιορισμοί στην γραμμή 5 και 6 είναι ισοδύναμοι με τους:

```
5  meet :- available(jane), available(john).
6  on(mon) ; on(tue) ; on(wed) ; on(thu) ; on(fri) :- meet.
```

  Εξήγηση:

  Δεδομένου ότι έχουμε 2 άτομα την Jane και τον John και οι μέρες από την Δευτέρα μέχρι την Παρασκευή. Η Jane και ο John προσπαθούν να βρουν μια μέρα που να μπορούν και οι δύο έτσι ώστε να συναντηθούν. Το άτομο *on* αντιπροσωπεύει την μέρα που θα βρεθούν. Στην γραμμή 3 μας λέει ότι εάν η ημέρα που θα βρεθούν δεν είναι η Παρασκευή τότε η Jane είναι διαθέσιμη ενώ στην γραμμή 3 μας λέει ότι εάν η ημέρα που θα βρεθούν δεν είναι η Δευτέρα ή η Τετάρτη τότε ο John είναι διαθέσιμος. Στην γραμμή 5 μας λέει ότι για κάθε άτομο X που είναι διαθέσιμος τότε μπορούν να βρεθούν. Τέλος στην γραμμή 6 λέει ότι εάν μπορούν να βρεθούν τότε πρέπει να είναι μια μέρα από την Δευτέρα μέχρι την Παρασκευή έτσι ώστε να μην παραβιάζει κανένα από τους πιο πάνω κανόνες.

- Ομαδοποίηση

  Εάν έχουμε περισσότερα από ένα άτομα σε ένα όρο μπορούμε να τα ομαδοποιήσουμε σε ένα με το σύμβολο “;” . Η ομαδοποίηση μπορεί να χρησιμοποιηθεί σε οποιοδήποτε είδος κανόνα όπως γεγονότα, απλούς περιορισμούς και περιορισμούς ακεραιότητας .

  Παράδειγμα Κατανόησης 3.8

  Εάν έχουμε τα πιο κάτω γεγονότα:

  person(alice,id1). person(caroline,id2). person(maggie,id3).

  Τότε μπορούμε να τα αντικαταστήσουμε με το

  person(alice,id1; caroline,id2; maggie,id3).

- Σύνολα

  Υπάρχουν κάποια καθορισμένα σύνολα που έχουν κάποιες συγκεκριμένες λειτουργίες όπως είναι το #sum (άθροισμα), #count (πληθικότητα), #min (μικρότερο) και #max (μεγαλύτερο).Τα σύνολα #sum και #count αφαιρούν τα διπλότυπα (εξήγηση στο παράδειγμα κατανόησης 3.9).

  Παράδειγμα Κατανόησης 3.9

  Δεδομένου το πιο κάτω πρόγραμμα:

```
1  vert(1;2;3;4).
2  edge(1,2,5; 2,1,10; 3,1,2; 4,1,7; 4,3,2).
3
4  vert_count(N):- N =#count{X : edge(X,1,Z)}.
5  weight_count(N):- N =#sum{Z,X,Y : edge(X,Y,Z)}.
6  min_weight(N):- N =#min{Z,X,Y : edge(X,Y,Z)}.
7  max_weight(N):- N =#max{Z,X,Y : edge(X,Y,Z)}.
8
9  min_edge_weight(X,Y):-min_weight(N),edge(X,Y,N).
10 max_edge_weight(X,Y):-max_weight(N),edge(X,Y,N).
```

  Το σύνολο απαντήσεων του πιο πάνω προγράμματος περιέχει:

```
vert(1) vert(2) vert(3) vert(4)
edge(1,2,5) edge(2,1,10)
edge(3,1,2) edge(4,1,7) edge(4,3,1)
vert_count(3) weight_count(26)
min_weight(1) min_edge_weight(4,3)
max_weight(10) max_edge_weight(2,1)
```

Εξήγηση:

Αρχικά υπάρχει ένας γράφος όπου έχει 4 κόμβους και 5 ακμές. Συγκεκριμένα έχει την ακμή από τον κόμβο 1 -> 2 με βάρος 5, από τον κόμβο 2 -> 1 με βάρος 10, από τον κόμβο 3 -> 1 με βάρος 2, από τον κόμβο 4 -> 1 με βάρος 7 και από τον κόμβο 4-> 3 με βάρος 2. Με την βοήθεια των συνόλων, αρχικά υπολογίζει πόσοι κόμβοι δείχνουν στον κόμβο 1, με απλά λόγια μετράει την πληθικότητα του συνόλου των κόμβων X, όπου X είναι κόμβος που δείχνει στον κόμβο 1 δηλαδή υπάρχει edge(X, 1, Z). Έπειτα υπολογίζει το άθροισμα των βαρών όλων των ακμών, όπου για κάθε ακμή edge(X, Y, Z) βρίσκει το άθροισμα των Z. Εάν είχαμε αυτόν τον κανόνα *weight_count(N):- N =#sum{Z : edge(X,Y,Z)}.* όπου αντί Z,X,Y υπάρχει μόνο το Z τότε το weight_count θα ήταν 24 γιατί η ακμή 1->2 και 4->3 έχουν το ίδιο βάρος θα τα θεωρήσει διπλότυπα και θα διαγράψει το ένα από τα 2 ενώ εάν έχουμε Z,X,Y δεν θα τα θεωρήσει διπλότυπα αφού έχουν διαφορετικούς κόμβους. Τέλος, βρίσκει τον κόμβο που έχει το μικρότερο βάρος και τον κόμβο που έχει το μεγαλύτερο βάρος. Για να γίνει αυτό πρώτα, για κάθε ακμή edge(X, Y, Z) βρίσκει το μικρότερο και το μεγαλύτερο βάρος στον γράφο, μετά βρίσκει την ακμή όπου το βάρος είναι το μικρότερο βάρος και την ακμή όπου το βάρος είναι το μεγαλύτερο βάρος που βρήκε πιο πάνω.

- Βελτιστοποίηση

  Μέχρι τώρα είχαμε τους περιορισμούς όπου καθορίζουν εάν ένα σύνολο ατόμων είναι μέσα στο σύνολο απαντήσεων ή όχι. Με τις δηλώσεις βελτιστοποίησης επεκτείνουμε αυτή την θεωρεία στο ότι πρέπει το σύνολο απαντήσεων είναι βέλτιστο. Έχουμε τις δηλώσεις #maximize and #maximize.

  Παράδειγμα Κατανόησης 3.10

  Δεδομένου το πιο κάτω πρόγραμμα:

```
1  vert(1;2;3;4).
2  edge(1,2; 2,1; 3,1; 4,1; 4,3; 4,3).
3
4  {in(X)}:-vert(X).
5  :- in(X),in(Y),edge(X,Y).
6
7  #maximize{1,X:in(X)}.
8  #show in/1.
```

Το σύνολο απαντήσεων του πιο πάνω προγράμματος περιέχει:

```
Answer: 1


Optimization: 0
Answer: 2
in(2)
Optimization: -1
Answer: 3
in(2) in(3)
Optimization: -2
OPTIMUM FOUND
```

Εξήγηση:

Αρχικά εάν το πρόβλημα μας έχει δήλωση βελτιστοποίησης τότε το Clingo βρίσκει πιθανές λύσεις και σταματάει όταν βρει την βέλτιστη. Στο πιο πάνω πρόγραμμα είναι ένας γράφος με 4 κόμβους και 5 ακμές . Έπειτα βρίσκει το μεγαλύτερο ανεξάρτητο σύνολο κόμβων δηλαδή το σύνολο κόμβων όπου δεν έχουν ακμή μεταξύ τους. Πιο αναλυτικά, στη αρχή λέει ότι κάθε κόμβος έχει την επιλογή να βρίσκεται στο σύνολο in ή όχι. Στην συνέχεια, έχει τον περιορισμό ότι δεν γίνεται 2 κόμβοι X και Y βρίσκονται στο σύνολο και να έχουν ακμή που τους συνδέει. Τέλος κάνει maximize την πληθικότητα του σύνολο in έτσι ώστε το βέλτιστο σύνολο να είναι αυτό με τους περισσότερους κόμβους. Σε αυτό το παράδειγμα υπάρχουν πολλές βέλτιστές και μια από αυτές είναι {in(2), in(3)}

Κεφάλαιο 4

Σύστημα Υλοποίησης και Πειραματική Αξιολόγηση

4.1 Εργαλείο IGraph

Το εργαλείο που χρησιμοποιήθηκε για την δημιουργία και επεξεργασία των γράφων είναι το IGraph. Το IGraph είναι μια συλλογή βιβλιοθηκών όπου χρησιμοποιείται για την δημιουργία, επεξεργασία γράφων καθώς επίσης και για την ανάλυση δικτύων. Αναπτύχθηκε από τους Gábor Csárdi και Tamás Nepusz το 2006. Το εργαλείο αυτό είναι γραμμένο στην γλώσσα C και υπάρχει επίσης ως πακέτα στις γλώσσες Python, R και επιπρόσθετα υπάρχει μια διεπαφή για το Mathematica. Σε παρούσα διπλωματική χρησιμοποιήθηκε το πακέτο στην γλώσσα Python.

Το IGraph κυρίως παρέχει χαρακτηριστικά του γράφου, για παράδειγμα υπολογίζει την κεντρικότητα, τον συντελεστή ομαδοποίησης, τις ιδιότητες με βάση το μήκος της διαδρομής, την διάμετρο του δικτύου, την πυκνότητα όπως και πολλά άλλα. Επίσης

χρησιμοποιείτε για την δημιουργία τυχαίων γραφημάτων με βάση πολλών μοντέλων, όπως είναι το μοντέλο Erdős-Rényi, το Barabási–Albert, το Forest Fire κ.ά.

Επιπρόσθετα χρησιμοποιείται για την απεικόνιση γράφων είτε στην οθόνη είτε σε ένα PDF, PNG ή SVG. Τέλος μπορείς να αποθηκεύσεις τον γράφο σε ένα αρχείο έως μια λίστα γειτνίασης, πίνακας γειτνίασης, λίστα ακμών κ.ά. Υπάρχουν πολλά εναλλακτικά εργαλεία όμως με βάση τον χρόνο εκτέλεσης και το ότι μπορεί να διαχειριστεί γράφους με περισσότερο από ένα εκατομμύριο κόμβους το καθιστά πάρα πολύ χρήσιμο.

4.2 Επεξήγηση Συστημάτων Υλοποίησης

Σε αυτό το υποκεφάλαιο θα γίνει μια λεπτομερής επεξήγηση των συστημάτων που υλοποιηθήκαν. Για την εύρεση μέγιστων ευσταθών επεκτάσεων σε ένα γράφο χρησιμοποιήθηκε ο επιλυτής Clingo όπου αναφέρθηκε πιο πάνω. Για την δημιουργία των γράφων και για τον υπολογισμό των αποτελεσμάτων χρησιμοποιήθηκε η γλώσσα Python .

Χρησιμοποιήθηκαν τέσσερα μοντέλα για την δημιουργία τυχαίων γραφημάτων, όπου είναι:

1. Μοντέλο Erdős-Rényi: Το μοντέλο Erdős-Rényi με δεδομένα εισόδου (n,d) θα δημιουργήσει ένα γράφο με n κόμβους με d πυκνότητα. Τα γραφήματα αυτά είναι ακολουθούν στοχαστική διαδικασία δημιουργίας του γράφου. Δηλαδή αρχικά δημιουργεί n κόμβους όπου κανένας κόμβος δεν συνδέεται και έχει και ένα σύνολο όπου είναι όλες οι πιθανές ακμές όπου μπορεί να δημιουργήσει ανάλογα εάν ο γράφος θα είναι κατευθυνόμενος ή όχι. Σε κάθε βήμα προσθέτει μία ακμή όπου επιλέγετε ομοιόμορφα από το σύνολο των πιθανών ακμών όπου και αφαιρείται. Ο αριθμός των ακμών που θα προσθέσει στον γράφο εξαρτάται από την πυκνότητα d
   που θα δοθεί στον γράφο. Έτσι τα γραφήματα που έχει τα ίδια δεδομένα είσοδού (n,d), η πιθανότητα να είναι ακριβώς οι ίδιοι είναι μικρή.[5]

2. Μοντέλο k-Regular:Το μοντέλο k-Regular με δεδομένα εισόδου (n,k) θα δημιουργήσει ένα γράφο με n κόμβους όπου κάθε κόμβος θα έχει βαθμό k. Σε περίπτωση ενός κατευθυνόμενου γράφου, ο βαθμός του κόμβου είναι το άθροισμα των ακμών που εξέρχονται από τον κόμβο και των ακμών που εισέρχονται από τον κόμβο. Δυνατά Κανονικά Γραφήματα (Strongly Regular Graphs) είναι τα γραφήματα με δεδομένα εισόδου (n, d, l, m) θα δημιουργήσει ένα γράφο με n κόμβους με d πυκνότητα, όπου κάθε δύο γειτονικοί κόμβοι έχουν l κοινούς γείτονες και κάθε δύο μη γειτονικοί κόμβοι έχουν m κοινούς γείτονες.[6]

3. Μοντέλο Scale Free: Ένα γράφημα χωρίς κλίμακα (Scale Free) είναι ένα γράφημα οπού η κατανομή βαθμών (degree) των κόμβος ακολουθεί έναν νόμο δύναμης. Δηλαδή το κλάσμα P(k) των κόμβων στο δίκτυο όπου συνδέονται με k άλλους κόμβους, για μεγάλες τιμές του k έχει την μορφή $P(k) \sim k^{-\gamma}$, όπου γ είναι μία παράμετρος όπου η τιμής της συνήθως κυμαίνεται στην περιοχή $2<\gamma<3$. Ένας γράφος για να έχει την ιδιότητα χωρίς κλίμακα υπάρχουν δύο βασικά χαρακτηρίστηκα, την ανάπτυξη και την προτιμησιακή προσκόλλησης. Με τον όρο ανάπτυξη ονομάζεται η διαδικασία ανάπτυξης γράφου όπου για μεγάλο χρονικό διάστημα νέοι κόμβοι συνδέονται με ένα ήδη υπάρχων δίκτυο. Με τον όρο προτιμησιακή προσκόλληση ονομάζεται, όταν ένας νέος κόμβος εισέρχεται στο δίκτυο προτιμάει να συνδεθεί με κόμβους που έχουν ήδη συγκεκριμένο αριθμό συνδέσεων. Έτσι υπάρχει μεγαλύτερη πιθανότητα ότι όλο και περισσότεροι κόμβοι θα συνδέονται με τον κόμβο που συνδέεται με τους περισσότερους κόμβους. Έτσι σε αυτούς τους γράφους υπάρχει η ιδιότητα του κεντρικού σημείου (hub), όπου είναι οι κόμβοι με βαθμό υψηλότερο από τον μέσο όρο. Μερικά παραδείγματα δικτύων που θεωρούνται ότι είναι χωρίς κλίμακα είναι τα κοινωνικά δίκτυα, σημασιολογικά δίκτυα, δίκτυα αεροπορικών εταιριών και πολλά είδη υπολογιστικών δικτύων όπως είναι το διαδίκτυο και ο παγκόσμιος ιστός (World Wide Web). Το μοντέλο Barabási–Albert δημιουργεί τυχαία γραφήματα βασισμένα στα γραφήματα χωρίς κλίμακα ακολουθώντας μια συγκεκριμένη διαδικασία. Δηλαδή, το γράφημα ξεκινά με ένα αρχικό συνδεδεμένο γράφο με m_0 κόμβους. Κάθε νέος κόμβος που προστίθεται στον

γράφο όπου συνδέεται με m υπάρχων κόμβους, όπου m $m \leq m_0$, με πιθανότητα p(i) ανάλογη τον αριθμό των ακμών που έχουν ήδη οι υπάρχοντες κόμβοι .$P(i)=k_i/\sum k_j$ όπου k_i είναι ο βαθμός του κόμβου i και j οι υπάρχων κόμβοι. Έτσι ένας νέος κόμβος είναι πολύ πιθανό να συνδεθεί με ένα βαριά συνδεδεμένο κόμβο.[5]

4. Μοντέλο Symmetric Graph: Το γράφημα αυτό είναι ένας γράφος βασισμένος στο μοντέλο Erdős-Rényi με δεδομένα εισόδου (n,d) όπου n είναι ο αριθμός των κόμβων και d η αρχική πυκνότητα. Έπειτα ελέγχει πόσες συμμετρικές ακμές έχει ο γράφος. Συμμετρικές ακμές είναι ζευγάρια ακμών όπου η μία δείχνει από τον A → B και η άλλη από τον B → A. Δοθέντος ενός αριθμού λόγου (ratio) όπου είναι η αναλογία των συμμετρικών ακμών που θα πρέπει να έχει ο γράφος και r τα ζεύγη συμμετρικών ακμών του γράφου, εάν r>ratio τότε παίρνουμε r-ratio ζεύγη και για κάθε ζεύγος διαγράφουμε τυχαία μια από τις δύο ακμές έτσι ώστε να μειώσουμε τα ζεύγη, αλλιώς
εάν r<ratio τότε παίρνουμε ratio-r ακμές και για κάθε ακμή προσθέτουμε μια ακμή που να έχει αντίθετη κατεύθυνση έτσι ώστε να αυξήσουμε τα ζεύγη.

Η παρούσα διπλωματική χωρίζεται σε 5 φάσεις όπου θα αναπτυχθούν με λεπτομέρεια στα πιο κάτω υποκεφάλαια.

4.2.1 Επεξήγηση Πρώτης Φάσης Υλοποίησης

Σε πρώτη φάση, θέλαμε να δούμε πως επηρεάζετε το σύνολο ευσταθών επεκτάσεων στα διαφορετικά μοντέλα δημιουργίας τυχαίων γράφων με διαφορετικές παραμέτρους όπως είναι η πυκνότητα. Δημιουργήθηκαν τα τέσσερα είδη κατευθυνόμενων γράφων που αναφέρθηκαν πιο πάνω όπου όλα έχουν 200 κόμβους. Για το μοντέλο Erdos Renyi δημιουργήθηκαν 100 διαφορετικά τυχαία γραφήματα για πυκνότητα 0.1, 0.2, 0.3 και 0.4. Για το μοντέλο k-Regular δημιουργήθηκαν 100 διαφορετικά γραφήματα για k=10 και k=15. Για το μοντέλο Scale Free δημιουργήθηκαν 100 διαφορετικά τυχαία γραφήματα με σταθερά ισχύος 0.3 και οι συμμετρικές ακμές είναι το 0%, 0.5% και 0.10% όλων των ακμών. Για το μοντέλο Symmetric

δημιουργήθηκαν 100 διαφορετικά τυχαία γραφήματα όπου οι συμμετρικές ακμές είναι το 10%, 20% και 30% όλων των ακμών με πυκνότητα 0.1, 0.2. Στο σύνολο παράχθηκαν 1500 γραφήματα. Για το κάθε γράφημα δημιουργήθηκε το αντίστοιχο πρόγραμμα στην Clingo όπου βρίσκει όλα τα σύνολα ευσταθών επεκτάσεων που υπάρχουν στον γράφο.

Για υπενθύμιση, για να είναι ένα σύνολο κόμβων S ευσταθής επέκταση πρέπει:

1. Να έχει την ιδιότητα απουσίας συγκρούσεων και
2. Κάθε κόμβος που δεν ανήκει στο σύνολο S, υπάρχει κάποιος κόμβος στο σύνολο S που κατευθύνεται στο κόμβο αυτό.

Παράδειγμα Κατανόησης 4.1:

Πιο κάτω είναι ένα παράδειγμα του αρχείου ErdosGraphD0.1-0.lp, όπου είναι η ονομασία του αρχείο σημαίνει ότι είναι ο πρώτος γράφος από τους 100 που παράχθηκαν από το μοντέλο Erdos Renyi με πυκνότητα 0.1

(1) vertex(0;1;..;199;).

(2) edge(0,21;0,30;…; 199,153;).

(3) in(X) :- not out(X), vertex(X).

(4) out(X) :- not in(X), vertex(X).

(5) :- in(X), in(Y), edge(X,Y).

(6) defeated(X) :- in(Y), edge(Y,X).

(7):- out(X), not defeated(X).

(8) #show in/1.

Εξήγηση:

Στην γραμμή 1 και 2 γίνεται η αρχικοποίηση του γράφου ,επειδή υπάρχουν πολλοί κόμβοι και ακμές για σκοπό της παρουσίασης τοποθετήθηκαν "…". Υπάρχουν 3 σύνολα, το σύνολο in που μέσα ανήκουν οι κόμβοι που ανήκουν στο σύνολο ευσταθών επεκτάσεων, το σύνολο out είναι οι

υπόλοιποι κόμβοι που δεν ανήκουν στο σύνολο και το σύνολο defeated όπου είναι οι κόμβοι που δεν ανήκουν στο σύνολο in αλλά υπάρχει κάποιος κόμβος στο σύνολο in που κατευθύνεται στον κόμβο αυτόν. Έπειτα έχουμε τους περιορισμούς ότι: (1) εάν ένας κόμβος X δεν ανήκει στο σύνολο out τότε ανήκει στο σύνολο in (γραμμή 3), (2) εάν ένας κόμβος X δεν ανήκει στο σύνολο in τότε ανήκει στο σύνολο out (γραμμή 4), (3) δεν γίνεται να υπάρχει ακμή X→Y και οι κόμβος X και Y να ανήκουν στο σύνολο in (γραμμή 5), (4) εάν υπάρχει ακμή Y→X και ο κόμβος Y τότε ο X ανήκει στο σύνολο defeated (γραμμή 6), (4) δεν γίνεται ένας κόμβος να ανήκει στο σύνολο out και να μην είναι στο σύνολο defeated (γραμμή 7). Η γραμμή 8 σημαίνει ότι στο σύνολο απαντήσεων θα εμφανίσεις μόνο τον σύνολο in.

Πιο κάτω φαίνεται το αποτέλεσμα του επιλυτή Clingo στον πρόγραμμα ErdosGraphD0.1-0.lp.

clingo version 5.4.0

Reading from GraphD0.1-0.lp

Solving...

Answer: 1

in(0) in(7) in(9) in(10) in(12) in(20) in(29) in(51) in(53) in(68) in(70) in(84) in(102) in(111) in(132) in(159) in(160) in(163) in(175) in(179) in(185) in(197)

Answer: 2

in(19) in(27) in(51) in(59) in(62) in(78) in(84) in(109) in(127) in(141) in(143) in(146) in(158) in(160) in(162) in(168) in(172) in(177)

Answer: 3

in(0) in(7) in(10) in(12) in(20) in(28) in(29) in(51) in(53) in(67) in(68) in(70) in(89) in(102) in(121) in(131) in(132) in(133) in(159) in(172) in(179) in(185) in(197)

SATISFIABLE

Models : 3

Calls : 1

Time : 5.574s (Solving: 5.53s 1st Model: 0.11s Unsat: 4.52s)

CPU Time : 5.266s

Από το πιο πάνω αποτέλεσμα βλέπουμε ότι υπάρχουν 3 σύνολα ευσταθών επεκτάσεων:

Est={{ 0, 7, 9, 10, 12, 20, 29, 51, 53, 68, 70, 84, 102, 111, 132, 159, 160, 163, 175, 179, 185, 197}, {19, 27, 51, 59, 62, 78, 84, 109, 127, 141, 143, 146, 158, 160, 162, 168, 172, 177}, { 0, 7, 10, 12, 20, 28, 29, 51, 53, 67, 68, 70, 89, 102, 121, 131, 132, 133, 159, 172, 179, 185, 197}}

4.2.2 Επεξήγηση Δεύτερης Φάσης Υλοποίησης

Σε δεύτερη φάση, έχουμε δύο γράφους, ο κύριος γράφος και ο γράφος ελέγχου (controller). Αυτό που θέλαμε να δούμε ήταν εάν μπορούμε να δημιουργήσουμε κάποιον άλλον γράφο ελέγχου όπου θα ελέγχει το πότε ένας συγκεκριμένος κόμβος από τον κύριο γράφο θα είναι μέσα στο σύνολο ευσταθών επεκτάσεων.

Έτσι δημιουργήθηκε ο κύριος γράφος με το μοντέλο Erdos Renyi με 200 κόμβους και ένας άλλος γράφος ελέγχου με αριθμό κόμβων n και με πυκνότητα 0, όπου κάθε του κόμβος συνδέετε με ένα κόμβο από τον κύριο γράφο. Όλοι οι κόμβοι είναι συνδεδεμένου με διαφορετικούς κόμβους. Για να μπορέσουμε να βγάλουμε ένα συμπέρασμα, δημιουργήθηκαν πολλά παραδείγματα γράφων με διαφορετικό αριθμό κόμβων στον γράφο ελέγχου και διαφορετική πυκνότητα στον κύριο γράφο. Στην πυκνότητα δόθηκαν οι τιμές 0.05, 0.10 και 0.15 και για τον αριθμό των κόμβων δόθηκαν οι τιμές 10, 20, 30, 40, 50 και 60. Για κάθε συνδυασμό πυκνότητας και αριθμό κόμβων δημιουργήθηκαν 100 διαφορετικά τυχαία γραφήματα. Στο σύνολο παράχθηκαν 1 800 γραφήματα. Σε αυτή την φάση το πρόγραμμα Clingo που δημιουργήθηκε θα επιλέγει ένα σύνολο κόμβων από τον

γράφο ελέγχου τέτοιο ώστε να υπάρχει τουλάχιστο ένα σύνολο ευσταθών επεκτάσεων όπου ένας συγκεκριμένος κόμβος X είναι μέσα. Για κάθε ένα από τα 1800 γραφήματα, δημιουργήθηκαν 200 προγράμματα της Clingo. Το πώς ακριβώς επηρεάζει το τι σύνολο θα κόμβων θα επιλέξουμε το σύνολο ευσταθών επεκτάσεων θα γίνει κατανοητό στο πιο κάτω παράδειγμα.

Παράδειγμα Κατανόησης 4.2:

Πιο κάτω είναι το αρχείου GraphV10-Density0.05-In0-0.lp, όπου είναι ο πρώτος γράφος (0) από τους 100 που παράχθηκαν με πυκνότητα 0.05, αριθμό κόμβων 10 και να πρέπει να είναι μέσα στο σύνολο ο κόμβος 0 από το κύριο γράφο.

(1) controller(200;201;202;203;204;205;206;207;208;209;).

(2) connections(200,197;201,187;202,138;203,14;204,182;205,11;206,
170;207,13;208,19;209,43;).

(3) vertex(0;1;...199;).

(4) edge(0,2;0,7;...199,198;).

(5) :- not in(0).

(6) {in_sub(X):controller(X)}.

(7) :-in_sub(Y),in(X),connections(Y,X).

(8) defeated(X) :- in_sub(Y), connections(Y,X).

(9) in(X) :- not out(X), vertex(X).

(10) out(X) :- not in(X), vertex(X).

(11) :- in(X), in(Y), edge(X,Y).

(12) defeated(X) :- in(Y), edge(Y,X).

(13) :- out(X), not defeated(X).

(14) #show .

Εξήγηση:

Στην γραμμή 1 και 2 γίνεται η αρχικοποίηση του γράφου ελέγχου και στην γραμμή 3 και 4 η αρχικοποίηση του κύριου γράφου. Επειδή υπάρχουν πολλοί κόμβοι και ακμές για σκοπό της παρουσίασης τοποθετήθηκαν "…". Το άτομο connectios(i,j) αντιπροσωπεύει ότι ο κόμβος i από τον γράφο ελέγχου είναι συνδεδεμένος με τον j κόμβου του κύριου γράφου, όπου $0 \leq i < 200$ και $200 \leq j < n+200$, n ο αριθμός των κόμβων στον γράφο ελέγχου. Υπάρχουν 4 σύνολα, το σύνολο in που μέσα ανήκουν οι κόμβοι που ανήκουν στο σύνολο ευσταθών επεκτάσεων, το σύνολο out είναι οι υπόλοιποι κόμβοι που δεν ανήκουν στο σύνολο, το σύνολο defeated όπου είναι οι κόμβοι που δεν ανήκουν στο σύνολο in αλλά υπάρχει κάποιος κόμβος στο σύνολο in που κατευθύνεται στον κόμβο αυτόν και το σύνολο in_sub όπου είναι οι κόμβοι από τον γράφο ελέγχου που επέλεξε. Έπειτα έχουμε τους περιορισμούς ότι: (1) ο κόμβος 0 δεν γίνεται να μην ανήκει στο σύνολο in(γραμμή 5), (2) θα επιλεχθεί ένα σύνολο in_sub από 0 μέχρι N κόμβους από τον γράφο ελέγχου (γραμμή 6), (3) δεν γίνεται να υπάρχει ακμή $Y \rightarrow X$, όπου Y είναι κόμβος που ανήκει στο σύνολο in_sub και X κόμβος από τον κύριο γράφο τότε ο κόμβος X ανήκει στο σύνολο defeated (γραμμή 7). Οι υπόλοιποι περιορισμοί είναι οι ίδιοι με την φάση 1 όπου βρίσκει το σύνολο των ευσταθών επεκτάσεων. Η γραμμή 14 σημαίνει ότι θα τυπωθεί μόνο εάν το πιο πάνω πρόγραμμα είναι SATISFIABLE ή UNSATISFIABLE. Εάν έχει αποτέλεσμα SATISFIABLE σημαίνει ότι το πρόγραμμα είναι ικανοποιήσιμο δηλαδή βρέθηκε τέτοιος συνδυασμός κόμβων όπου ο κόμβος 0 ανήκει στο σύνολο, αλλιώς UNSATISFIABLE ότι δεν βρέθηκε λύση.

Πιο κάτω φαίνεται το αποτέλεσμα του επιλυτή Clingo στον πρόγραμμα GraphV10-Density0.05-In0-0.lp.

```
clingo version 5.4.0
Reading from GraphV10-Ratio5-In0-0.lp
Solving...
UNSATISFIABLE
```

Models : 0

Calls : 1

Time : 0.156s (Solving: 0.03s 1st Model: 0.00s Unsat: 0.03s)

CPU Time : 0.125s

Όπως φαίνεται πιο πάνω το συγκεκριμένο πρόγραμμα είναι UNSATISFIABLE , δηλαδή δεν βρέθηκε τέτοιος συνδυασμός συνόλου από τον γράφο ελέγχου έτσι ώστε ο κόμβος 0 να ανήκεις στο σύνολο ευσταθών επεκτάσεων.

4.2.3 Επεξήγηση Τρίτης Φάσης Υλοποίησης

Σε τρίτη φάση, θέλαμε να βρούμε εάν υπάρχει κάποια συσχέτιση μεταξύ του πόσες φορές εμφανίζεται ένας κόμβος στα σύνολα ευσταθών επεκτάσεων και με τα χαρακτηριστικά αυτού του κόμβου. Για να καταλάβουμε την δομή ενός γράφου θα πρέπει να εκτιμήσουμε την σημαντικότητα του κάθε κόμβου και αυτό μπορούμε να το βρούμε μέσω της κεντρικότητας τους, η οποία καθορίζεται με διάφορους τρόπους.

Οι βασικοί χαρακτηρισμοί του κόμβου που καθορίζουν της σημαντικότητα του κόμβου σε ένα γράφο είναι:

1. Η Κεντρικότητα Ενδιαμεσότητας (Betweenness Centrality) ενός κόμβου x όπου υπολογίζει το πλήθος των συντομότερων μονοπατιών (shortest path), όπου εμφανίζεται ο κόμβος x. Η ενδιαμεσότητα ενός κόμβου x, ορίζεται ως εξής:

$$betweenness(x) = \sum_{s \neq x \neq t} \frac{\sigma st(x)}{\sigma st}$$

Όπου σst(x) είναι ο αριθμός των συντομότερων διαδρομών (shortest paths) από τον κόμβο s στον κόμβο t όπου παίρνουν από τον κόμβο x.

2. Η Κεντρικότητα Εγγύτητας (Closeness Centrality) ενός κόμβου x έχει υψηλή τιμή εάν ο κόμβος βρίσκεται σε κοντινές αποστάσεις με τους υπόλοιπους κόμβους του γράφου. Η εγγύτητα ενός κόμβου x, ορίζεται ως εξής:

$$closure(x) = \frac{1}{\sum_y d(x,y)}$$

Όπου d(x,y) είναι η απόσταση από τον κόμβο x στον κόμβο y.

3. Η Κεντρικότητα Βαθμού (Degree Centrality) ενός κόμβου x είναι ο αριθμός των ακμών που φεύγουν από τον κόμβο συν τον αριθμό των ακμών που καταλήγουν στον κόμβο.

Για την ανάλυση χρησιμοποιηθήκαν τα γραφήματά από την φάση 1 και τα αποτελέσματα των αντίστοιχων προγραμμάτων της Clingo.Δηλαδή χρησιμοποιήθηκαν τα μοντέλα Erdos Renyi, k-Regular, Scale Free και Symmetric. Για το μοντέλο Erdos Renyi δημιουργηθήκαν 100 τυχαία γραφήματα για πυκνότητα 0.1, 0.2, 0.3 και 0.4. Για το μοντέλο k-Regular δημιουργηθήκαν 100 τυχαία γραφήματα για k 10 και 15. Για το μοντέλο Scale Free δημιουργηθήκαν 100 τυχαία γραφήματα όπου οι συμμετρικές ακμές είναι το 0%, 0.5% και 0.10% όλων των ακμών. Για το μοντέλο Symmetric δημιουργηθήκαν 100 τυχαία γραφήματα όπου οι συμμετρικές ακμές είναι το 10%, 20% και 30% όλων των ακμών και με πυκνότητα 0.1 και 0.2. Για τον κάθε ένα από τα 1500 γραφήματα δημιουργήθηκε πρόγραμμα στην Clingo όπου βρίσκει όλα τα σύνολα ευσταθών επεκτάσεων που υπάρχουν στον γράφο. Σε αυτή την φάση της υλοποίησης χρησιμοποιήθηκαν οι 1500 γράφοι όπου ο κάθε γράφος είναι αποθηκευμένος σε ένα αρχείο .txt και διαβάζεται από το πρόγραμμα με την βοήθεια του εργαλείου IGraph, επίσης χρησιμοποιήθηκαν και τα αποτελέσματα του επιλυτή clingo για κάθε γράφο.

Σε πρώτη προσέγγιση, δημιουργήθηκαν κάποια στατιστικά για κάθε συνδυασμό γράφου (πχ Symmetric Graph με πυκνότητα 0.2 και λόγο 0.5%). Για κάθε συνδυασμό γράφου υπάρχουν 100 τυχαία γραφήματα, έτσι για κάθε ένα γράφο από τους 100 βρέθηκαν οι 20 κόμβοι από τους 200 με τις

πιο πολλές και πιο λίγες εμφανίσεις στο σύνολο ευσταθών επεκτάσεων. Επίσης με την βοήθεια του εργαλείου IGraph βρέθηκαν οι 20 κόμβοι με την μεγαλύτερη και οι 20 με την μικρότερη ενδιαμεσότητα (betweenness), εγγύτητα (closeness) και βαθμό (degree). Έπειτα για κάθε συνδυασμό των διανυσμάτων με τους 20 κόμβους με τις λιγότερες και τους 20 κόμβους με τις περισσότερες εμφανίσεις στο σύνολο ευσταθών επεκτάσεων και των έξι διανυσμάτων με τις μετρήσεις κεντρικότητας (20 κόμβοι με την μεγαλύτερη και 20 κόμβοι με την μικρότερη ενδιαμεσότητα, εγγύτητα και βαθμό), βρέθηκε η πληθικότητα τομή τους, δηλαδή ο αριθμός των κοινών κόμβων μεταξύ των δύο διανυσμάτων. Άρα για κάθε συνδυασμό γράφου θα παραχθούν 12 αριθμοί οι οποίοι είναι:

(|Min Count Λ Min Betweenness|, |Min Count Λ Max Betweenness|, |Max Count Λ Min Betweenness|, |Max Count Λ Max Betweenness|, |Min Count Λ Min Closeness|, |Min Count Λ Max Closeness|, |Max Count Λ Min Closeness|, |Max Count Λ Max Closeness|, |Min Count Λ Min Degree|, |Min Count Λ Max Degree|, |Max Count Λ Min Degree|, |Max Count Λ Max Degree|). Το Max Count αντιπροσωπεύει το διάνυσμα με τους 20 κόμβους με τις περισσότερες εμφανίσεις στο σύνολο ευσταθών επεκτάσεων και το Min Count το διάνυσμα με τους 20 κόμβους με τις λιγότερες εμφανίσεις. Το Max Betweenness αντιπροσωπεύει το διάνυσμα με τους 20 κόμβους με την μεγαλύτερη ενδιαμεσότητα και το Min Betweenness το διάνυσμα με τους 20 κόμβους με την λιγότερη. Με την ίδια λογική ισχύουν για τα διανύσματα Max Closeness, Min Closeness, Max Degree, Min Degree. Για παράδειγμα ο αριθμός |Max Count Λ Min Closeness| αντιπροσωπεύει το πόσους κοινούς κόμβους έχουν τα διανύσματα Max Count και Min Closeness. Με την ίδια λογική ορίζονται και οι υπόλοιποι αριθμοί. Τέλος βρέθηκε ο μέσος όρος αυτών των 12 αριθμών για τους 100 γράφους από κάθε συνδυασμό, με σκοπό να δούμε εάν υπάρχει κάποια πιθανή συσχέτιση.

Σε δεύτερη προσέγγιση, για τον κάθε συνδυασμό να βρέθηκαν 5 αριθμοί για κάθε κεντρικότητα (Ενδιαμεσότητα, Εγγύτητα και Βαθμό):

1. Για κάθε ένα από τους 100 γράφους, αρχικά βρέθηκε το πόσες φορές εμφανίζονται οι 20 κόμβοι με την μεγαλύτερη ενδιαμεσότητα, εγγύτητα και βαθμό στο σύνολο ευσταθών επεκτάσεων του γράφου και διαιρέθηκε αυτός ο αριθμός με το 20. Τέλος βρέθηκε ο μέσος όρος αυτού του αριθμού για τους 100 γράφους από κάθε συνδυασμό.
2. Για κάθε ένα από τους 100 γράφους, βρέθηκε ο μέσος δηλαδή προστεθήκαν όλα τα μεγέθη όλων το συνόλων ευσταθών επεκτάσεων του γράφου και διαιρέθηκαν με το 200. Τέλος βρέθηκε ο μέσος όρος αυτού του αριθμού για τους 100 γράφους από κάθε συνδυασμό.
3. Για κάθε ένα από τους 100 γράφους, αρχικά βρέθηκε το πόσες φορές εμφανίζονται οι 20 κόμβοι με την μικρότερη ενδιαμεσότητα, εγγύτητα και βαθμό στο σύνολο ευσταθών επεκτάσεων του γράφου και διαιρέθηκε αυτός ο αριθμός με το 20. Τέλος βρέθηκε ο μέσος όρος αυτού του αριθμού για τους 100 γράφους από κάθε συνδυασμό.
4. Για κάθε ένα από τους 100 γράφους βρέθηκε ο μέσος όρος της ενδιαμεσότητα, εγγύτητα και βαθμό των 20 κόμβων με την μεγαλύτερη ενδιαμεσότητα, εγγύτητα και βαθμό αντίστοιχα. Τέλος βρέθηκε ο μέσος όρος αυτού του αριθμού για τους 100 γράφους από κάθε συνδυασμό.
5. Για κάθε ένα από τους 100 γράφους βρέθηκε ο μέσος όρος της ενδιαμεσότητα, εγγύτητα και βαθμό των 20 κόμβων με την μικρότερη ενδιαμεσότητα, εγγύτητα και βαθμό αντίστοιχα. Τέλος βρέθηκε ο μέσος όρος αυτού του αριθμού για τους 100 γράφους από κάθε συνδυασμό.

4.2.4 Επεξήγηση Τέταρτης Φάσης Υλοποίησης

Σε τέταρτη φάση, θέλαμε να κάνουμε κάτι παρόμοιο με την φάση 2 αλλά αυτή την φορά επιλέγουμε μόνο ένα κόμβο από τον ελέγχου και θέλαμε να δούμε πως επηρεάζεται το σύνολο ευσταθών επεκτάσεων εάν για παράδειγμα επιλέξουμε τον 1ο κόμβο του ελέγχου ή τον 2ο.

Έτσι δημιουργήθηκε ο κύριος γράφο με το μοντέλο Erdos Renyi με 200 κόμβους και ένα άλλος γράφος ελέγχου με 20 κόμβους και πυκνότητα 0, όπου κάθε του κόμβος συνδέετε με ένα κόμβο από τον κύριο γράφο. Όλοι οι κόμβοι είναι συνδεδεμένου με διαφορετικούς κόμβους. Για να μπορέσουμε να βγάλουμε ένα συμπέρασμα, δημιουργήθηκαν πολλά παραδείγματα γράφων με διαφορετική πυκνότητα στον κύριο γράφο. Στην πυκνότητα δόθηκαν οι τιμές 0.05, 0.10 και 0.15 όπου για κάθε πυκνότητα δημιουργήθηκαν 100 διαφορετικά τυχαία γραφήματα, άρα παράχθηκαν στο σύνολο 300 γραφήματα. Σε αυτή την φάση το πρόγραμμα Clingo που δημιουργήθηκε επιλέγει ένα συγκεκριμένο κόμβο Χ από τον γράφο ελέγχου και να βρεθούν όλα σύνολα ευσταθών επεκτάσεων , έτσι ώστε στην συνέχεια να γίνει δυνατό να υπολογιστεί πως επηρεάζεται το σύνολο ευσταθών επεκτάσεων για διαφορετικό κόμβο Χ.

Παράδειγμα Κατανόησης 4.3:

Πιο κάτω είναι το αρχείου GraphDensity5-In201-1.lp, όπου είναι ο δεύτερος γράφος (1) από τους 100 που παράχθηκαν με πυκνότητα 0.05 όπου πρέπει ο κόμβος που πρέπει να επιλεχθεί από τον γράφο ελέγχου είναι ο 1 (201-200).

(1) controller(200;201;202;203;204;205;206;207;208;209;210;211;212;213;214;215;216;217;218;219;).

(2) connections(200,168;201,130;202,67;203,169;204,15;205,110;206,136;207,88;208,38;209,135;210,70;211,192;212,148;213,121;214,195;215,31;216,99;217,103;218,7;219,131;).

(3) vertex(0;1; … ,199;).

(4) edge(0,5;0,40; … ;199,181;).

(5) in_sub(201).

(6) :-in_sub(Y),in(X),connections(Y,X).

(7) defeated(X) :- in_sub(Y), connections(Y,X).

(8) in(X) :- not out(X), vertex(X).

(9) out(X) :- not in(X), vertex(X).

(10) :- in(X), in(Y), edge(X,Y).

(11) defeated(X) :- in(Y), edge(Y,X).

(12) :- out(X), not defeated(X).

(13) #show in/1.

Εξήγηση:

Στην γραμμή 1 και 2 γίνεται η αρχικοποίηση του γράφου ελέγχου και στην γραμμή 3 και 4 η αρχικοποίηση του κύριου γράφου. Όλοι οι περιορισμοί είναι οι ίδιοι με της φάσης 2 εκτός από την γραμμή 5 και 13. Στην φάση 2 είχαμε ένα περιορισμό που επέλεγε ένα σύνολο κόμβων από τον γράφο ελέγχου, ενώ τώρα ένας κόμβος συγκεκριμένα πρέπει αν είναι στο σύνολο in_sub και ο κόμβος αυτός στο παράδειγμα είναι ο 201 δηλαδή ο κόμβος 1 του ελέγχου, Στην γραμμή 13 αναφέρει ότι θα τυπωθεί το περιεχόμενο του συνόλου των ευσταθών επεκτάσεων.

Πιο κάτω φαίνεται το αποτέλεσμα του επιλυτή Clingo στον πρόγραμμα GraphDensity5-In201-1.lp.

clingo version 5.4.0

Reading from GraphDensity5-In201-1.lp

Solving...

Answer: 1

in(4) in(5) in(7) in(15) in(21) in(22) in(27) in(28) in(37) in(46) in(60) in(61) in(69) in(75) in(77) in(79) in(82) in(84) in(89) in(103) in(105) in(109) in(117) in(124) in(131) in(134) in(140) in(158) in(176) in(187) in(196) in(197)

Answer: 2

in(6) in(8) in(14) in(22) in(25) in(39) in(40) in(41) in(47) in(48) in(50) in(58) in(60) in(64) in(66) in(76) in(79) in(88) in(99) in(101) in(102) in(103) in(105) in(109) in(110) in(121) in(131) in(132) in(140) in(153) in(155) in(162) in(167) in(173) in(182)

Answer: 3

in(7) in(8) in(12) in(13) in(19) in(22) in(27) in(32) in(38) in(39) in(45) in(46) in(55) in(63) in(66) in(86) in(90) in(91) in(112) in(121) in(123) in(124) in(131) in(134) in(135) in(140) in(145) in(158) in(161) in(162) in(184) in(197) in(199)

Answer: 4

in(1) in(8) in(12) in(13) in(27) in(38) in(39) in(41) in(45) in(46) in(55) in(63) in(66) in(86) in(90) in(112) in(121) in(124) in(131) in(134) in(135) in(145) in(151) in(158) in(161) in(162) in(170) in(182) in(184) in(197) in(199)

Answer: 5

in(5) in(11) in(14) in(22) in(33) in(36) in(37) in(41) in(47) in(52) in(54) in(58) in(59) in(60) in(64) in(66) in(79) in(88) in(93) in(95) in(99) in(102) in(103) in(105) in(117) in(118) in(121) in(153) in(171) in(173) in(182) in(183) in(197)

Answer: 6

in(11) in(14) in(22) in(33) in(36) in(37) in(41) in(47) in(52) in(54) in(58) in(59) in(60) in(64) in(66) in(79) in(88) in(93) in(95) in(99) in(102) in(103) in(105) in(118) in(121) in(153) in(162) in(171) in(173) in(179) in(182) in(183) in(197)

Answer: 7

in(3) in(8) in(11) in(13) in(15) in(20) in(28) in(41) in(53) in(57) in(60) in(63) in(66) in(67) in(78) in(81) in(90) in(100) in(102) in(105) in(112) in(121) in(122) in(124) in(134) in(138) in(145) in(147) in(156) in(158) in(169) in(170) in(172) in(182) in(197)

Answer: 8

in(7) in(18) in(19) in(22) in(30) in(33) in(35) in(36) in(47) in(52) in(59) in(64) in(71) in(74) in(79) in(89) in(93) in(94) in(99) in(105) in(111) in(118) in(121) in(148) in(162) in(167) in(169) in(171) in(173) in(182) in(184) in(194)

Answer: 9

in(7) in(18) in(19) in(22) in(30) in(33) in(35) in(36) in(47) in(52) in(59) in(71) in(74) in(78) in(79) in(89) in(93) in(94) in(99) in(102) in(105) in(111) in(118) in(121) in(162) in(167) in(169) in(171) in(173) in(182) in(184) in(187) in(194)

SATISFIABLE

Models : 9

Calls : 1

Time : 1.065s (Solving: 1.02s 1st Model: 0.19s Unsat: 0.23s)

CPU Time : 0.938s

Από το πιο πάνω αποτέλεσμα βλέπουμε ότι υπάρχουν 9 σύνολα ευσταθών επεκτάσεων:

EST={{4, 5, 7, 15, 21, 22, 27, 28, 37, 46, 60, 61, 69, 75, 77, 79, 82, 84, 89, 103, 105, 109, 117, 124, 131, 134, 140, 158, 176, 187, 196, 197},

{6, 8, 14, 22, 25, 39, 40, 41, 47, 48, 50, 58, 60, 64, 66, 76, 79, 88, 99, 101, 102, 103, 105, 109, 110, 121, 131, 132, 140, 153, 155, 162, 167, 173, 182},

…….,

{7, 18, 19, 22, 30, 33, 35, 36, 47, 52, 59, 71, 74, 78, 79, 89, 93, 94, 99, 102, 105, 111, 118, 121, 162, 167, 169, 171, 173, 182, 184, 187, 194}}

Αυτή η διαδικασία έγινε και για επιλογή κάθε συνδυασμού δύο κόμβων Πχ In200-In203 από τον γράφο ελέγχου.

4.3 Στατιστική Ανάλυση

Σε κάθε φάση της διπλωματική με την βοήθεια της Python έγινε μια επεξεργασία των αποτελεσμάτων της Clingo και δημιουργήθηκαν κάποια στατιστικά. Σε αυτό το σημείο θα γίνει μια λεπτομερείς επεξήγηση για το τί ακριβώς στατιστικά δημιουργήθηκαν και το πως, σε κάθε μία από τις 4 φάσεις.

4.2.1 Στατιστικά Πρώτης Φάσης Υλοποίησης

Στη πρώτη φάση, για κάθε μοντέλο δημιουργίας τυχαίων γραφημάτων και για κάθε συνδυασμό δημιουργήθηκαν τρείς αριθμοί: (1) ο μέσος όρος των ευσταθών επεκτάσεων που έχει ο κάθε γράφος (Average of Stable Extensions), (2) ο μέσος όρος των επιχειρημάτων που υπάρχουν στο σύνολο ευσταθών επεκτάσεων(Average of Arguments), (3) ο αριθμός των γράφων που είναι μη ικανοποιήσιμοι (UNSATISFIABLE) (Num of Zero Stable Extensions). Στους πιο κάτω πίνακες αποτελεσμάτων παρουσιάζονται τα αποτελέσματα που δημιουργήθηκαν για κάθε συνδυασμό.

Πίνακας Αποτελεσμάτων 4.1:

Για το μοντέλο Erdős-Rényi δημιουργήθηκαν τα πιο κάτω στατιστικά.

Density / Description	0.1	0.2	0.3	0.4
Average of Stable Extensions	3.21	4.01	5.54	7.15
Average of Arguments	68.68	50.78	49.97	49.27
Num of Zero Stable Extensions	9	2	0	0

Πίνακας Αποτελεσμάτων 4.2:

Για το μοντέλο k-Regular δημιουργήθηκαν τα πιο κάτω στατιστικά.

K / Description	10	15
Average of Stable Extensions	2.35	2.64
Average of Arguments	77.06	66.16
Num of Zero Stable Extensions	14	10

Πίνακας Αποτελεσμάτων 4.3:

Για το μοντέλο Scale Free δημιουργήθηκαν τα πιο κάτω στατιστικά.

Ratio of Symmetric Edges (%) / Description	0	5	10
Average of Stable Extensions	0.58	0.73	1.22
Average of Arguments	26.48	33.20	55.13
Num of Zero Stable Extensions	60	54	39

Πίνακας Αποτελεσμάτων 4.4:

Για το μοντέλο Symmetric δημιουργήθηκαν τα πιο κάτω στατιστικά.

Density	**0,1**			**0.2**		
Ratio of Symmetric Edges(%) / Description	**10**	**20**	**30**	**10**	**20**	**30**
Average of Stable Extensions	8.11	75.42	1162.64	4.13	19.73	123.47
Average of Arguments	173.36	1576.57	23636.12	52.32	243.41	1488.57
Num of Zero Stable Extensions	0	0	0	6	0	0

Συμπεράσματα με βάση τους πίνακες αποτελεσμάτων 4.1, 4.2, 4.2, 4.3, 4.4.

Για τους γράφους με το μοντέλο Erdos Renyi και k-Regular παρατηρήθηκε ότι καθώς αυξάνετε η πυκνότητα, ο μέσος όρος των συνόλων ευσταθών επεκτάσεων αυξάνετε και ο μέσος όρος των κόμβων που είναι μέσα μειώνετε, το ίδιο και ο αριθμός των γράφων που ήταν μη ικανοποιήσιμοι. Για τους γράφους με το μοντέλο Scale Free και Symmetric παρατηρήθηκε ότι καθώς αυξάνετε το Ratio, δηλαδή η αναλογία συμμετρικών ακμών σε σχέση με τον αριθμό των ακμών των συμμετρικών ακμών ο μέσος όρος των συνόλων ευσταθών επεκτάσεων και ο μέσος όρος των κόμβων που είναι μέσα αυξάνετε, και ο αριθμός των γράφων που ήταν μη ικανοποιήσιμοι μειώνετε. Τα αποτελέσματα αυτά είναι αναμενόμενα, αφού καθώς αυξάνετε η πυκνότητα ενός γράφου, αυξάνετε και η συνδεσιμότητα μεταξύ των ακμών και η

πιθανότητα να βρεθεί ένα σύνολο ευσταθών επεκτάσεων αυξάνεται. Όμως στους γράφους με το μοντέλο Symmetric εάν αυξήσουμε την πυκνότητα του γράφου τότε, τα σύνολα ευσταθών επεκτάσεων μειώνονται καθώς και ο μέσος όρος των κόμβων που ανήκουν στο σύνολο.

4.2.2 Στατιστικά Δεύτερης Φάσης Υλοποίησης

Στη δεύτερη φάση, για κάθε συνδυασμό n,d, όπου n είναι ο αριθμός των κόμβων στον γράφο ελέγχου και d η πυκνότητα στον κύριο γράφο, δημιουργήθηκαν δύο αριθμοί: (1) ο μέσος όρος των γράφων όπου ήταν ικανοποιήσιμοι (SATISFIABLE)(Πχ εάν num1=140, σημαίνει ότι από τα 200 προγράμματα που έχει ο κάθε γράφος, ένα για κάθε κόμβου που επιλέγουμε να είναι μέσα στο σύνολο, τα 140 είναι ικανοποιήσιμα) (Average of Satisfiable (out of 200)) (2) ο μέσος όρος των γράφων όπου ήταν μη ικανοποιήσιμοι (UNSATISFIABLE) (Πχ εάν num2=60, σημαίνει ότι από τα 200 προγράμματα που έχει ο κάθε γράφος, ένα για κάθε κόμβου που επιλέγουμε να είναι μέσα στο σύνολο, τα 60 είναι μη ικανοποιήσιμα) (Average of Unsatisfiable (out of 200)). Στους πιο κάτω πίνακες αποτελεσμάτων παρουσιάζονται τα αποτελέσματα που πάρθηκαν.

Πίνακας Αποτελεσμάτων 4.5:

Για τον κάθε συνδυασμό δημιουργήθηκαν τα πιο κάτω στατιστικά.

NumVertex of Controller	**10**			**20**		
Density / **Description**	**0.05**	**0.10**	**0.15**	**0.5**	**0.10**	**0.15**
Average of Satisfiable (out of 200)	90.86	80.60	68.94	142.31	125.89	107.88
Average of Unsatisfiable (out of 200)	109.41	119.40	131.06	57.69	74.11	92.12

NumVertex of Controller	30			40		
Density / Description	0.05	0.10	0.15	0.5	0.10	0.15
Average of Satisfiable (out of 200)	181.30	168.45	145.49	193.77	192.02	177.99
Average of Unsatisfiable (out of 200)	92.12	35.55	54.51	6.23	7.98	22.01

NumVertex of Controller	50			60		
Density / Description	0.05	0.10	0.15	0.5	0.10	0.15
Average of Satisfiable (out of 200)	198.49	197.80	192.89	199.34	199.64	198.40
Average of Unsatisfiable (out of 200)	1.51	2.20	7.11	0.66	0.36	1.60

Συμπεράσματα με βάση τον πίνακα αποτελεσμάτων 4.5.

Από τα αποτελέσματα που δημιουργήθηκαν παρατηρήθηκε ότι καθώς αυξάνετε ο αριθμός των κόμβων στον γράφο ελέγχου και καθώς μειώνετε η πυκνότητα στον κύριο γράφο τόσο αυξάνεται ο αριθμός των κόμβων που μπορούν να «ελεγχτούν». Ο αριθμός αυτός για τον αριθμό των κόμβων 50, φτάνει το 199.64 από τα 200 αλλά για τον αριθμό 60 παρατηρώ ότι μειώνεται αυτός ο αριθμός. Έτσι εγίνε μια δοκιμή και για τον αριθμό 70 και παρατηρήθηκε ότι μειώνετε και άλλο. Άρα για τον αριθμό των κόμβων στον γράφο ελέγχου 50 και με πυκνότητα 5% έχουμε το μέγιστο επιθυμητό αποτέλεσμα. Τα αποτελέσματα αυτά είναι λογικά αφού καθώς ο αριθμός των κόμβων στον ελέγχου αυξάνεται και ο γράφος με χαμηλή πυκνότητα δεν έχει τόσο συνδεσιμότητα και έτσι το πρόγραμμα θα μπορεί να επιλέξει πιο μεγάλο σύνολο και έτσι να αποκλείσει τους κόμβους που περιορίζουν τον κόμβο που θέλουμε να ανήκει στο σύνολο. Όμως αυτό δεν σημαίνει ότι δώσουμε ένα πολύ μεγάλο αριθμό στην πληθικότητα των κόμβων, θα έχουμε καλύτερα αποτελέσματα, όπως στην περίπτωση με τον αριθμό 50 και 60, όπου το 50 είναι ο ιδανικός αριθμός.

4.2.3 Στατιστικά Τρίτης Φάσης Υλοποίησης

Στην τρίτη φάση, τα στατιστικά που δημιουργήθηκαν από την πρώτη προσέγγιση είναι στους πίνακες αποτελεσμάτων 4.6, 4.7, 4.8, 4.9. Όπως αναφέρθηκε και στο Κεφάλαιο 4.13 ο αριθμός Avg |Min Count Λ Max Between.| για παράδειγμα εάν έχουμε τον γράφο Erdős-Rényi με πυκνότητα 0.2 είναι ο μέσος όρος των κοινών κόμβων που έχουν τα διανύσματα Min Count και Max Betweenness από τους 100 γράφους που δημιουργήθηκαν για αυτό τον συνδυασμό. Όπου Min Count είναι το διάνυσμα που αποτελείται από τους 20 κόμβους με τις λιγότερες εμφανίσεις στα σύνολα ευσταθών επεκτάσεων ενός συγκεκριμένου γράφου και Max Betweenness αποτελείται από τους 20 κόμβους με την μεγαλύτερη ενδιαμεσότητα του συγκεκριμένου γράφου. Το Max Count είναι το διάνυσμα που αποτελείται από τους 20 κόμβους με τις περισσότερες εμφανίσεις στα σύνολα ευσταθών επεκτάσεων και το Min Betweenness αποτελείται από τους 20 κόμβους με την μικρότερη ενδιαμεσότητα. Με την ίδια λογική ορίζονται και τα διανύσματα Min Closeness, Max Closeness, Min Degree, Max Degree.

Πίνακας Αποτελεσμάτων 4.6:

Για το μοντέλο Erdős-Rényi δημιουργήθηκαν τα πιο κάτω στατιστικά.

Density / Description	0.1	0.2	0.3	0.4
Avg |Min Count Λ Min Betweenness|	2.12	2.08	2.07	2.03
Avg |Min Count Λ Max Betweenness|	1.51	1.92	1.88	1.92
Avg |Max Count Λ Min Betweenness|	0.9	1.05	1.03	1.27
Avg |Max Count Λ Max Betweenness|	3.34	3.19	2.95	3.22

Density / Description	0.1	0.2	0.3	0.4
Avg |Min Count Λ Min Closeness|	2.49	2.22	2.56	2.45
Avg |Min Count Λ Max Closeness|	1.81	1.85	1.82	2.21
Avg |Max Count Λ Min Closeness|	0.85	0.91	0.93	1.01
Avg |Max Count Λ Max Closeness|	3.61	3.46	3.74	3.60

Description \ Density	0.1	0.2	0.3	0.4
Avg |Min Count Λ Min Degree|	2.52	2.24	2.29	2.21
Avg |Min Count Λ Max Degree|	1.67	1.84	1.97	2.11
Avg |Max Count Λ Min Degree|	0.88	1.09	1.13	1.35
Avg |Max Count Λ Max Degree|	3.52	3.33	3.05	3.20

Πίνακας Αποτελεσμάτων 4.7:

Για το μοντέλο k-Regular δημιουργήθηκαν τα πιο κάτω στατιστικά.

Description \ K	10	15
Avg |Min Count Λ Min Betweenness|	1.84	1.95
Avg |Min Count Λ Max Betweenness|	2.19	2.05
Avg |Max Count Λ Min Betweenness|	1.61	1.86
Avg |Max Count Λ Max Betweenness|	1.79	1.82

Description \ K	10	15
Avg |Min Count Λ Min Closeness|	2.19	3.67
Avg |Min Count Λ Max Closeness|	2.21	2.55
Avg |Max Count Λ Min Closeness|	1.64	1.90
Avg |Max Count Λ Max Closeness|	2.07	2.33

Description \ K	10	15
Avg |Min Count Λ Min Degree|	14.42	14.69
Avg |Min Count Λ Max Degree|	14.42	14.69
Avg |Max Count Λ Min Degree|	3.83	3.82
Avg |Max Count Λ Max Degree|	3.83	3.82

Πίνακας Αποτελεσμάτων 4.8:

Για το μοντέλο Scale Free δημιουργήθηκαν τα πιο κάτω στατιστικά.

Ratio of Symmetric Edges (%) / Description	0	5	10
Avg |Min Count Λ Min Betweenness|	14.01	14.5	14.48
Avg |Min Count Λ Max Betweenness|	0	0	0.01
Avg |Max Count Λ Min Betweenness|	0.82	0.81	0.84
Avg |Max Count Λ Max Betweenness|	0.54	0.79	0.6

Ratio of Symmetric Edges (%) / Description	0	5	10
Avg |Min Count Λ Min Closeness|	16.16	16.17	16.17
Avg |Min Count Λ Max Closeness|	0	0	0
Avg |Max Count Λ Min Closeness|	0.91	0.8	0.67
Avg |Max Count Λ Max Closeness|	0.3	0.84	0.55

Ratio of Symmetric Edges (%) / Description	0	5	10
Avg |Min Count Λ Min Degree|	16.14	16.49	16.87
Avg |Min Count Λ Max Degree|	0	0	0
Avg |Max Count Λ Min Degree|	0.92	0.84	0.7
Avg |Max Count Λ Max Degree|	0.45	0.60	1.0

Πίνακας Αποτελεσμάτων 4.9:

Για το μοντέλο Symmetric δημιουργήθηκαν τα πιο κάτω στατιστικά.

Density	0,1			0.2		
Ratio of Symmetric Edges(%) / Description	10	20	30	10	20	30
Avg |Min Count Λ Min Between.|	5.86	4.37	4.57	2.42	8.43	4.10
Avg |Min Count Λ Max Between.|	0.94	0.29	0.21	1.89	0.51	0.47
Avg |Max Count Λ Min Between.|	1.46	0.89	0.55	0.85	1.97	1.20
Avg|Max Count Λ Max Between.|	4.07	5.56	6.38	3.21	4.03	5.30

Density	0,1			0.2		
Ratio of Symmetric Edges(%) / Description	10	20	30	10	20	30
Avg |Min Count Λ Min Closeness|	3.17	5.26	7.46	2.36	3.34	5.47
Avg |Min Count Λ Max Closeness|	1.42	0.22	0.09	2.02	1.18	0.30
Avg |Max Count Λ Min Closeness|	0.7	0.26	0.15	0.74	0.65	0.28
Avg |Max Count Λ Max Closeness|	4.69	5.96	6.88	3.38	4.60	5.97

Density	0,1			0.2		
Ratio of Symmetric Edges(%) / Description	10	20	30	10	20	30
Avg |Min Count Λ Min Degree|	6.54	4.62	4.81	2.68	9.43	4.55
Avg |Min Count Λ Max Degree|	1.02	0.20	0.16	2.05	0.50	0.36
Avg |Max Count Λ Min Degree|	1.59	0.86	0.40	0.87	2.01	0.91
Avg|Max Count Λ Max Degree|	4.40	5.76	6.90	3.30	4.18	5.56

Στους πίνακες αποτελεσμάτων 4.10, 4.11, 4.12 και 4.13 που ακολουθούν είναι τα στατιστικά που δημιουργήθηκαν από την δεύτερη προσέγγιση. Για κάθε συνδυασμό γράφου δημιουργήθηκαν 5 αριθμοί για την ενδιαμεσότητα, 5 αριθμοί για την εγγύτητα και 5 αριθμοί για τον βαθμό. Για παράδειγμα εάν έχουμε τον γράφο με πυκνότητα 0.2 ο αριθμός (1) Average occurrences of 20 max betweenness nodes που αναγράφεται στους πίνακες σημαίνει ότι για τους 100 γράφους που δημιουργήθηκαν για αυτό τον συνδυασμό είναι ο μέσος όρος του πόσες φορές εμφανίζονται οι 20 κόμβοι με την μεγαλύτερη ενδιαμεσότητα στο σύνολο ευσταθών επεκτάσεων του συγκεκριμένου γράφου δια του 20. Ο αριθμός (3) Average occurrences of 20 min betweenness nodes ορίζεται με την ίδια λογική αλλά για του 20 κόμβους με την λιγότερη ενδιαμεσότητα. Με την ίδια λογική είναι και οι αριθμοί Average occurrences of 20 max closeness nodes, Average occurrences of 20 max closeness nodes όπου αναφέρεται για την εγγύτητα και οι αριθμοί Average occurrences of 20 max degree nodes, Average occurrences of 20 max degree nodes όπου αναφέρεται για τον βαθμό. Επιπρόσθετα, ο αριθμός (2) Average occurrences of each node είναι ο μέσος όρος των εμφανίσεων του κάθε κόμβου στο σύνολο ευσταθών επεκτάσεων. Ο αριθμός (4) Average betweenness of 20 max betweenness nodes ορίζεται ως ο μέσος όρος την ενδιαμεσότητας των 20 κόμβων με την μεγαλύτερη ενδιαμεσότητα και ο αριθμός (5) Average betweenness of 20 min betweenness nodes ορίζεται ως ο μέσος όρος την ενδιαμεσότητας των 20 κόμβων με την μικρότερη ενδιαμεσότητα. Με την ίδια λογική ορίζονται και οι αριθμοί Average betweenness of 20 max closeness nodes, Average betweenness of 20 min closeness nodes όπου αναφέρονται στην εγγύτητα του κόμβου και οι αριθμοί Average betweenness of 20 max degree nodes, Average betweenness of 20 min degree nodes όπου αναφέρονται στον βαθμό του κόμβου.

Πίνακας Αποτελεσμάτων 4.10:

Για το μοντέλο Erdős-Rényi δημιουργήθηκαν τα πιο κάτω στατιστικά.

Density / Description	0.1	0.2	0.3	0.4
Average occurrences of 20 max betweenness nodes	0.2015	0.1525	0.1530	0.1580
Average occurrences of each node	0.3434	0.254	0.249	0.246
Average occurrences of 20 min betweenness nodes	0.5980	0.6165	0.3535	0.3705
Average betweenness of 20 max betweenness nodes	327.64	549.35	728.8	875.7
Average betweenness of 20 min betweenness nodes	108.24	214.9	318.7	413.1
Density / Description	**0.1**	**0.2**	**0.3**	**0.4**
Average occurrences of 20 max closeness nodes	0.1830	0.1405	0.1390	0.4410
Average occurrences of each node	0.3434	0.254	0.249	0.246
Average occurrences of 20 min closeness nodes	0.6165	0.4310	0.4410	0.4160
Average closeness of 20 max closeness nodes	0.568	1.20	1.90	2.67
Average closeness of 20 min closeness nodes	0.5377	1.126	1.77	0.3720
Density / Description	**0.1**	**0.2**	**0.3**	**0.4**
Average occurrences of 20 max degree nodes	0.1945	0.1530	0.1595	0.1635
Average occurrences of each node	0.3434	0.254	0.249	0.246
Average occurrences of 20 min degree nodes	0.5975	0.4080	0.3850	0.3720
Average degree of 20 max degree nodes	50.52	144.24	279.66	455.94
Average degree of 20 min degree nodes	29.74	95.73	199.42	341.67

Πίνακας Αποτελεσμάτων 4.11:

Για το μοντέλο k-Regular δημιουργήθηκαν τα πιο κάτω στατιστικά.

K / Description	10	15
Average occurrences of 20 max betweenness nodes	0.3720	0.3295
Average occurrences of each node	0.385	0.33
Average occurrences of 20 min betweenness nodes	0.4060	0.3515
Average betweenness of 20 max betweenness nodes	317.06	565.52
Average betweenness of 20 min betweenness nodes	283.15	510.24
K / Description	**10**	**15**
Average occurrences of 20 max closeness nodes	0.3630	0.2960
Average occurrences of each node	0.385	0.33
Average occurrences of 20 min closeness nodes	0.4335	0.3845
Average closeness of 20 max closeness nodes	0.504	1.044
Average closeness of 20 min closeness nodes	0.483	1.015
K / Description	**10**	**15**
Average occurrences of 20 max degree nodes	0.4020	0.3355
Average occurrences of each node	0.385	0.33
Average occurrences of 20 min degree nodes	0.4020	0.3355
Average degree of 20 max degree nodes	20.0	50.00
Average degree of 20 min degree nodes	20.0	50.00

Πίνακας Αποτελεσμάτων 4.12:

Για το μοντέλο Scale Free δημιουργήθηκαν τα πιο κάτω στατιστικά.

Ratio of Symmetric Edges (%) / Description	0	5	10
Average occurrences of 20 max betweenness nodes	0.0265	0.0370	0.0590
Average occurrences of each node	0.132	0.17	0.276
Average occurrences of 20 min betweenness nodes	0.2155	0.2775	0.2757
Average betweenness of 20 max betweenness nodes	1159.8	2469.78	3779.80
Average betweenness of 20 min betweenness nodes	62.3	114.2	166.14
Ratio of Symmetric Edges (%) / Description	**0**	**5**	**10**
Average occurrences of 20 max closeness nodes	0.0305	0.0370	0.0535
Average occurrences of each node	0.132	0.17	0.276
Average occurrences of 20 min closeness nodes	0.2005	0.250	0.4130
Average closeness of 20 max closeness nodes	0.53	1.06	1.59
Average closeness of 20 min closeness nodes	0.40	0.81	1.21
Ratio of Symmetric Edges (%) / Description	**0**	**5**	**10**
Average occurrences of 20 max degree nodes	0.300	0.0395	0.0565
Average occurrences of each node	0.132	0.17	0.276
Average occurrences of 20 min degree nodes	0.2420	0.2985	0.4990
Average degree of 20 max degree nodes	33.66	77.11	120.56
Average degree of 20 min degree nodes	8.01	16.12	24.24

Πίνακας Αποτελεσμάτων 4.13:

Για το μοντέλο Symmetric δημιουργήθηκαν τα πιο κάτω στατιστικά.

Density	**0,1**			**0.2**		
Ratio of Symmetric Edges(%) / Description	**10**	**20**	**30**	**10**	**20**	**30**
Average of occurrences of 20 max nodes	0.5455	5.578	80.89	0.138	0.957	7.278
Average of occurrences of each node	0.86	7.88	118.18	0.261	1.22	7.44
Average of occurrences of 20 min nodes	1.4610	13.22	198.2	0.411	1.994	7.431
Average betweenness of 20 max betweenness nodes	353.19	721.01	1066.4	1287.9	1553.0	353.2
Average betweenness of 20 min betweenness nodes	102.02	192.7	278.2	384.8	481.0	102.0

Density	**0,1**			**0.2**		
Ratio of Symmetric Edges(%) / Description	**10**	**20**	**30**	**10**	**20**	**30**
Average occurrences of 20 max closeness nodes	0.408	3.801	54.41	0.118	0.670	7.379
Average occurrences of each node	0.86	7.88	118.18	0.261	1.22	7.44
Average occurrences of 20 min closeness nodes	1.5725	13.3985	206.72	0.4335	2.0625	7.3210
Average closeness of 20 max closeness nodes	0.568	1.13	1.70	2.33	2.97	0.56
Average closeness of 20 min closeness nodes	0.53	13.69	1.61	2.21	2.79	0.54

Density	0,1			0.2		
Ratio of Symmetric Edges(%) / Description	10	20	30	10	20	30
Average occurrences of 20 max degree nodes	0.573	5.394	76.57	0.130	0.354	7.165
Average occurrences of each node	0.86	7.88	118.18	0.261	1.22	7.44
Average occurrences of 20 min degree nodes	1.517	13.5935	205.93	0.4175	2.0055	7.3530
Average degree of 20 max degree nodes	56.74	121.3	189.6	283.4	399.6	56.74
Average degree of 20 min degree nodes	30.63	63.0	97.9	163.8	233.7	30.64

Συμπεράσματα με βάση τους πίνακες αποτελεσμάτων 4.6 --- 4.14.

Από τα αποτελέσματα που δημιουργήθηκαν στην πρώτη προσέγγιση παρατηρήθηκε ότι στους γράφους με το μοντέλο Erdos Renyi οι κόμβοι με υψηλή ενδιαμεσότητας (betweenness), εγγύτητα (closeness) και βαθμό (degree) έχουν μεγαλύτερη πιθανότητα να βρίσκονται στο σύνολο ευσταθών επεκτάσεων. Σε αντίθεση με στους γράφους με το μοντέλο k-Regular όπου οι κόμβοι με υψηλή ενδιαμεσότητας και κλειστότητα τότε έχουν μικρότερη πιθανότητα να βρίσκονται στο σύνολο ευσταθών επεκτάσεων. Για τους γράφους με το μοντέλο Scale Free, το μεγαλύτερο ποσοστό των κόμβων που εμφανίζονται λιγότερο στο σύνολο φαίνεται να έχουν και χαμηλή ενδιαμεσότητας, κλειστότητα και βαθμό. Για τους γράφους με το μοντέλο Symmetric φαίνεται ότι οι κόμβοι που εμφανίζονται περισσότερες φορές στο σύνολο ευσταθών επεκτάσεων ο μέσος όρος της ενδιαμεσότητας, της κλειστότητα και του βαθμό τους είναι πολύ υψηλός. Αρά από αυτά τα αποτελέσματα καταλαβαίνουμε ότι η σημαντικότητα του κόμβου παίζει σημαντικό ρόλο εάν ο κόμβος θα είναι στο σύνολο ευσταθών επεκτάσεων.

Στην δεύτερη προσέγγιση, όπως βλέπουμε από τους πιο πάνω πίνακες, για το μοντέλο Erdos Renyi για τις πυκνότητες 0.1, 0.2, 0.3 και 0.4 , ο μέσος όρος της

ενδιαμεσότητα των 20 κόμβων με την μεγαλύτερη ενδιαμεσότητα κυμαίνεται από την τιμή 327.64 μέχρι 875.7 και ο μέσος όρος της ενδιαμεσότητα των 20 κόμβων με την μικρότερη ενδιαμεσότητα κυμαίνεται από την τιμή 108,24 μέχρι 413,1. Ο μέσος όρος της εγγύτητας των 20 κόμβων με την μεγαλύτερη εγγύτητας κυμαίνεται από την τιμή 0.568 μέχρι 2,67 και ο μέσος όρος της εγγύτητας των 20 κόμβων με την μικρότερη εγγύτητα κυμαίνεται από την τιμή 0,5377 μέχρι 2,4753. Ο μέσος όρος του βαθμού των 20 κόμβων με τον μεγαλύτερο βαθμό κυμαίνεται από την τιμή 50,52 μέχρι 455,94 και ο μέσος όρος του βαθμού των 20 κόμβων με τον μικρότερο βαθμό κυμαίνεται από την τιμή 29,74 μέχρι 341,67.

Για το μοντέλο k-Regular για k=10 και k=15 , ο μέσος όρος της ενδιαμεσότητα των 20 κόμβων με την μεγαλύτερη ενδιαμεσότητα είναι 317,06 για κ=10 και 565,52 για κ=15 και ο μέσος όρος της ενδιαμεσότητα των 20 κόμβων με την μικρότερη ενδιαμεσότητα είναι 283,15 για κ=10 και 510,24 για κ=15. Ο μέσος όρος της εγγύτητας των 20 κόμβων με την μεγαλύτερη εγγύτητας κυμαίνεται από την τιμή 0.504 μέχρι 1,044 και ο μέσος όρος της εγγύτητας των 20 κόμβων με την μικρότερη εγγύτητα είναι 0,483 για κ=10 και 1,015. Ο μέσος όρος του βαθμού των 20 κόμβων με τον μεγαλύτερο βαθμό είναι 20 για κ=10 και 50 το ίδιο και για τους κόμβους με τον μικρότερο βαθμό. Αυτό οφείλεται γιατί στον γράφο k-Regular όλοι οι κόμβοι έχουν ίδιο βαθμό.

Για το μοντέλο Scale Free για αναλογία 0%, 5% και 10% , ο μέσος όρος της ενδιαμεσότητα των 20 κόμβων με την μεγαλύτερη ενδιαμεσότητα κυμαίνεται από την τιμή 1159,8 μέχρι 3779,80 και ο μέσος όρος της ενδιαμεσότητα των 20 κόμβων με την μικρότερη ενδιαμεσότητα κυμαίνεται από την τιμή 62,3 μέχρι 166,14. Ο μέσος όρος της εγγύτητας των 20 κόμβων με την μεγαλύτερη εγγύτητας κυμαίνεται από την τιμή 0,53 μέχρι 1,59 και ο μέσος όρος της εγγύτητας των 20 κόμβων με την μικρότερη εγγύτητα κυμαίνεται από την τιμή 0,40 μέχρι 1,21. Ο μέσος όρος του βαθμού των 20 κόμβων με τον μεγαλύτερο βαθμό κυμαίνεται από την τιμή 33,66 μέχρι 120,56 και ο μέσος όρος του

βαθμού των 20 κόμβων με τον μικρότερο βαθμό κυμαίνεται από την τιμή 8,01 μέχρι 24,24.

Για το μοντέλο Symmetric για αναλογία 10%, 20% και 30% και πυκνότητα 0.1 και 0.2, ο μέσος όρος της ενδιαμεσότητα των 20 κόμβων με την μεγαλύτερη ενδιαμεσότητα κυμαίνεται από την τιμή 353,19 μέχρι 1553 και ο μέσος όρος της ενδιαμεσότητα των 20 κόμβων με την μικρότερη ενδιαμεσότητα κυμαίνεται από την τιμή 102,02 μέχρι 481. Ο μέσος όρος της εγγύτητας των 20 κόμβων με την μεγαλύτερη εγγύτητας κυμαίνεται από την τιμή 0,538 μέχρι 2,97 και ο μέσος όρος της εγγύτητας των 20 κόμβων με την μικρότερη εγγύτητα κυμαίνεται από την τιμή 0,53 μέχρι 2,79. Ο μέσος όρος του βαθμού των 20 κόμβων με τον μεγαλύτερο βαθμό κυμαίνεται από την τιμή 56,74 μέχρι 399,6 και ο μέσος όρος του βαθμού των 20 κόμβων με τον μικρότερο βαθμό κυμαίνεται από την 30,63 μέχρι 233,7.

Σε όλα τα μοντέλα παρατηρήθηκε ότι καθώς αυξάνεται η πυκνότητα, το k ή η αναλογία μεταξύ ακμών και συμμετρικών ακμών, αυξάνεται και η ενδιαμεσότητα, εγγύτητα και βαθμός του κάθε κόμβου. Εκτός από το μοντέλο Symmetric όπου για πυκνότητα 0.2 και αναλογία 30% μειώνονται οι τιμές. Επίσης από τις μετρήσεις φαίνεται ότι ο μέσος όρος εμφανίσεων στα σύνολο ευσταθών επεκτάσεων των 20 κόμβων με τη μικρότερη ενδιαμεσότητα, εγγύτητα και βαθμό, είναι μεγαλύτερος αποτι ο μέσο όρο των 20 κόμβων με τη μεγαλύτερη ενδιαμεσότητα, εγγύτητα και βαθμό. Ένας λόγος που μπορεί να συμβαίνει αυτό είναι επειδή ο μέσος όρος των εμφανίσεων του κάθε κόμβου είναι πολύ κοντά σε αυτές τι τιμές. Επίσης ο μέσος όρος μερικές φορές δεν είναι τόσο κατατοπιστικός αφού μπορεί να υπάρχουν κάποιες ακραίες τιμές, δηλαδή ένα κόμβος να εμφανίζεται πάρα πολλές φορές και ένας άλλος καθόλου και όταν γίνει ο μέσος όρος να φαίνεται ότι ο κάθε κόμβος έχει ίσες εμφανίσεις.

4.2.4 Στατιστικά Τέταρτης Φάσης Υλοποίησης

Στη τέταρτη φάση, για κάθε συνδυασμό d,x, όπου d είναι η πυκνότητα στον κύριο γράφο και x είναι ο κόμβος που επιλέγετε από τον γράφο ελέγχου, δημιουργήθηκαν οι εξής μετρήσεις. Αρχικά μετρήθηκαν οι εμφανίσεις του κάθε κόμβου στο σύνολο ευσταθών επεκτάσεων. Άρα για κάθε πυκνότητα d, έχω 20 διανύσματα, ένα για κάθε κόμβο που επιλέγεται. Έπειτα για κάθε πιθανό συνδυασμό 2 διανυσμάτων υπολογίστηκε ο μέσος όρος (από τους 100 γράφους που δημιουργήθηκαν για κάθε συνδυασμό) του Cosine Similarity όπως επίσης και της Ευκλείδεια Απόστασης τους, με σκοπό να δούμε την ευαισθησία του κυρίου γράφου δηλαδή, το πως επηρεάζεται το σύνολο ευσταθών επεκτάσεων εάν αποκλείσουμε ένα κόμβο από το να βρίσκεται μέσα.

Οι δύο βασικοί αλγόριθμοι για τον υπολογισμό ομοιότητας μεταξύ δύο διανυσμάτων είναι:

1. Το Cosine Similarity υπολογίζει την ομοιότητα μεταξύ δύο θετικών διανυσμάτων και δίνει αποτέλεσμα ένα δεκαδικό αριθμό x , 0<=x<=1 όπου όσο πιο κοντά στο 0 είναι αυτός ο αριθμός σημαίνει ότι τα δύο διανύσματα είναι πολύ διαφορετικά και όσο πιο κοντά είναι στο 1 σημαίνει ότι τα δύο διανύσματα είναι πολύ όμοια. Ο υπολογισμός της ομοιότητας μεταξύ των διανυσμάτων p και q γίνεται με την πράξη:

$$similarity(p,q) = \frac{p \cdot q}{\|p\| * \|q\|} = \frac{\sum_{i=1}^{n} pi * qi}{\sum_{i=1}^{n} pi * \sum_{i=1}^{n} qi}$$

2. Η Ευκλείδεια Απόσταση υπολογίζει την απόσταση μεταξύ δύο διανυσμάτων και δίνει αποτέλεσμα ένα δεκαδικό αριθμό x , x>=0 όπου όσο πιο κοντά στο 0 είναι αυτός ο αριθμός σημαίνει ότι τα δύο διανύσματα είναι πολύ όμοια και όσο πιο μεγάλος είναι αυτός ο αριθμός τόσο διαφορετικά είναι. Ο υπολογισμός της ομοιότητας μεταξύ των διανυσμάτων p και q γίνεται με την πράξη:

$$distance(p,q) = \sqrt{\sum_{i=1}^{n} (qi - pi)^2}$$

Για κάθε συνδυασμό υπολογίστηκε ο μέσος όρος των 20 διανυσμάτων. Οι μετρήσεως αυτές υπολογίστηκαν και για την επιλογή δύο κόμβων από τον γράφο ελέγχου. Στους πιο κάτω πίνακες αποτελεσμάτων παρουσιάζονται τα αποτελέσματα που πάρθηκαν. Ο αριθμός Average Cosine Similarity είναι ο μέσος όρος του Cosine Similarity των 20 διανυσμάτων και ο αριθμός Average Euclidean Distance είναι ο μέσος όρος της Ευκλείδειας Απόστασης των 20 διανυσμάτων.

Πίνακας Αποτελεσμάτων 4.14:

Για την επιλογή ενός κόμβου από τον γράφο ελέγχου, δημιουργήθηκαν τα πιο κάτω στατιστικά.

Density / Description	0.05	0.10	0.15
Average Cosine Similarity	0.582	0.785	0.827
Average Euclidean Distance	3.973	3.660	3.012

Πίνακας Αποτελεσμάτων 4.15:

Για την επιλογή δύο κόμβων από τον γράφο ελέγχου, δημιουργήθηκαν τα πιο κάτω στατιστικά.

Density / Description	0.05	0.10	0.15
Average Cosine Similarity	0.515	0.704	0.758
Average Euclidean Distance	5.862	5.354	4.446

Συμπεράσματα με βάση τους πίνακες αποτελεσμάτων 4.14, 4.15.

Από τα αποτελέσματα που υπολογίστηκαν παρατηρήθηκε ότι καθώς αυξάνετε η πυκνότητα του γράφου αυξάνεται και η ομοιότητα. Το αποτέλεσμα αυτό είναι αναμενόμενο αφού καθώς αυξάνεται η πυκνότητα του γράφου, αυξάνεται και η συνδεσιμότητα μεταξύ των κόμβων, έτσι όποιον κόμβο και να αποκλείσουμε από το να βρίσκεται στο σύνολο ευσταθών επεκτάσεων, φαίνεται να μην επηρεάζει τόσο την επιλογή των κόμβων που θα ανήκουν. Όσων αφορά για την

Ευκλείδεια Απόσταση ξέρουμε ότι αντιπροσωπεύει το πόσο «κοντά» ή «μακριά» είναι δύο αντικείμενα ή δύο διανύσματα. Χαμηλή απόσταση συνεπάγεται στην υψηλή ομοιότητα αυτών των δύο αντικειμένων. Ίδιες παρατηρήσεις υπολογίστηκαν και από την ευκλείδεια απόσταση δηλαδή, σε κάθε πυκνότητα η απόσταση μεταξύ των κόμβων που επιλέγονται είναι πολύ μικρή και καθώς αυξάνετε η πυκνότητα του κόμβου μειώνεται η απόσταση. Στην περίπτωση που επιλέχθηκαν δύο κόμβοι τα αποτελέσματα και στο Cosine Similarity και στην Ευκλείδεια Απόσταση ήταν τα ίδια, δηλαδή καθώς αυξάνεται η πυκνότητα, αυξάνεται η ομοιότητα και μειώνεται η απόσταση μεταξύ των διανυσμάτων. Όμως οι τιμές του cosine similarity σε αυτή την περίπτωση είναι μικρότερες και οι τιμές της ευκλείδειας απόστασης είναι μεγαλύτερες από την περίπτωση που επιλέξαμε μόνο έναν κόμβο από τον γράφο ελέγχου.

Κεφάλαιο 5

Συμπεράσματα

5.1 Συμπεράσματα Μελέτης

Τελικό συμπέρασμα είναι ότι το σύνολο μπορεί να ελεγχθεί και να μας βγάλει τα αποτελέσματα που θέλουμε αλλά θα πρέπει να γίνει η σωστή διαχείριση της πυκνότητας και των αριθμό των κόμβων που θα χρησιμοποιηθούν ανάλογα με το πρόβλημα. Επίσης το ποιοι κόμβοι θα βρίσκονται μέσα στο σύνολο, στα πλείστα μοντέλα δημιουργίας τυχαίων γραφημάτων που χρησιμοποιήθηκαν φάνηκε ότι εξαρτάται από το ενδιαμεσότητας και την εγγύτητα , όμως δεν σημαίνει ότι εάν ένα κόμβος έχει υψηλή ενδιαμεσότητα ή εγγύτητα, τότε θα βρίσκεται στο σύνολο ευσταθών επεκτάσεων πιο πολλές φορές αλλά είναι πολύ πιθανό να βρίσκεται τουλάχιστο μία φορά. Σε κάποια μοντέλα όπως είναι το k-Regular και το Symmetric, επειδή ο βαθμός του κάθε κόμβου τροποποιείται, έτσι ο βαθμός δεν είναι τόσο αντικειμενικός παράγοντας για να καθοριστεί η σημαντικότητα τους κόμβου στον γράφο. Δηλαδή στο μοντέλο k-Regular όλοι οι κόμβοι έχουν τον ίδιο βαθμό και στο μοντέλο Symmetric επειδή τροποποιήθηκε με τέτοιο τρόπο ο βαθμός των κόμβων.

5.2 Μελλοντική Επέκταση

Σε αυτό το υποκεφάλαιο θα αναφερθούν κάποιες βελτιώσεις και εισηγήσεις που θα μπορούσαν να υλοποιηθούν, με σκοπό την καλύτερη κατανόηση της δομής των σύνθετων γράφων.

Αρχικά θα μπορούσαμε να χρησιμοποιήσουμε και άλλα μοντέλα για την δημιουργία τυχαίων γραφημάτων έτσι ώστε να έχουμε πιο πολλές συγκρίσεις και επιλογές.

Επίσης, γενικά υπάρχουν πολλά κριτήρια για την μέτρηση της σημαντικότητα ενός κόμβου σε ένα σύνθετο δίκτυο, αλλά το κάθε ένα εστιάζει μόνο σε ορισμένες πτυχές και έτσι εάν επικεντρωθούμε μόνο σε αυτές σίγουρα θα υπάρχει μια απώλεια πληροφοριών. Έτσι θα ήταν προτιμότερο εάν χρησιμοποιούσαμε κάποια μέθοδό, χρησιμοποιεί πολλαπλά κριτήρια.

Τέλος στην φάση 2 όπου θέλαμε να ελέγξουμε το γράφο και το ευσταθές σύνολο με το να αναγκάζαμε να είναι ένας συγκεκριμένος κόμβος μέσα στο σύνολο, θα μπορούσαμε να το κάνουμε για περισσότερους συνδυασμούς κόμβων.

Βιβλιογραφία

[1] Pietro Baroni, Massimiliano Giacomin, "Semantics of Abstract Argument Systems", *Argumentation in Artificial Intelligence*, G. Simari, I. Rahwan editors, Springer, 2009

[2] Martin Caminada, "A gentle Introduction to argumentation semantic", University of Luxembourg, Summer 2008.

[3] Torsten Schaub, «Answer Set Solving in Practice», University of Potsdam, http://www.cs.uni-potsdam.de/~torsten/Potassco/Slides/asp.pdf

[4] Martin Gebser, Roland Kaminski, Benjamin Kaufmann, Max Ostrowski, Torsten Schaub, Sven Thiele, «Using gringo, clingo and iclingo», 4 October 2010

[5] Prettejohn Brenton J, Berryman Matthew J and McDonnell Mark D, «Methods for generating complex networks with selected structural properties for simulations: a review and tutorial for neuroscientists », *Frontiers in Computational Neuroscience*, 5, 2011.

[6] Wikipedia, «k-Regular graph», https://en.wikipedia.org/wiki/Regular_graph, Last Updated: 13 March 2020

Παράρτημα Α

Σε αυτό το παράρτημα παρατίθεται το πρόγραμμα για την δημιουργία τυχαίων γράφων και των προγραμμάτων της Clingo για τις Φάσεις 1,2 και 4.

```
from igraph import *
import random
import os
from itertools import combinations

def writeInFile(graph,filename):
   f= open(filename,"w+")
   f.write("vertex(")
   for v in graph.vs:
      f.write("%d;" % (v.index))
   f.write(").\n")
   f.write("edge(")
   for e in graph.es:
      f.write("%d,%d;" % (e.source,e.target))
   f.write(").\n")

   f.write("in(X) :- not out(X), vertex(X).\n")
   f.write("out(X) :- not in(X), vertex(X).\n")
   f.write(":- in(X), in(Y), edge(X,Y).\n")
   f.write("defeated(X) :- in(Y), edge(Y,X).\n")
   f.write(":- out(X), not defeated(X).\n")
   f.write("#show in/1.")
   f.close()

def writeInFileController(controller,graph,edges,v_in,filename):
   f= open(filename,"w+")
   f.write("controller(")
   for v in controller.vs:
      f.write("%d;" % (v.index+len(graph.vs)))
   f.write(").\n")
   f.write("connections(")
   for e in range(len(edges)):
      f.write("%d,%d;" % (edges[e][0],edges[e][1]))
   f.write(").\n")

   f.write("vertex(")
   for v in graph.vs:
      f.write("%d;" % (v.index))
   f.write(").\n")
   f.write("edge(")
```

```
   for e in graph.es:
      f.write("%d,%d;" % (e.source,e.target))
   f.write(").\n")

   f.write("\n:- not in(%d).\n" %(v_in))
   f.write("{in_sub(X):controller(X)}.\n")
   f.write(":-in_sub(Y),in(X),connections(Y,X).\n")
   f.write("defeated(X) :- in_sub(Y), connections(Y,X).\n\n")
   f.write("in(X) :- not out(X), vertex(X).\n")
   f.write("out(X) :- not in(X), vertex(X).\n")
   f.write(":- in(X), in(Y), edge(X,Y).\n")
   f.write("defeated(X) :- in(Y), edge(Y,X).\n")
   f.write(":- out(X), not defeated(X).\n")
   f.write("#show .")
   f.close()

def writeInFileControllerCombs(controller,graph,edges,combs_num,density,g
raph_num):
   comb = combinations(controller, combs_num)
   combs=list(comb)
   for combination in combs:
      filename="GraphDensity%d-"%(int(density*100))
      for i in range(combs_num):
         filename+="In%d"%(combination[i])
         if (i+1!=combs_num):
            filename+="-"
      filename+="-%d.lp"%(graph_num)

      f= open(filename,"w+")
      f.write("controller(")
      for v in controller:
         f.write("%d;" % (v))
      f.write(").\n")
      f.write("connections(")
      for e in range(len(edges)):
         f.write("%d,%d;" % (edges[e][0],edges[e][1]))
      f.write(").\n")

      f.write("vertex(")
      for v in graph.vs:
         f.write("%d;" % (v.index))
      f.write(").\n")
      f.write("edge(")
      for e in graph.es:
         f.write("%d,%d;" % (e.source,e.target))
      f.write(").\n")
      for i in range(combs_num):
         f.write("in_sub(%d).\n" %(combination[i]))
```

```
      f.write("\n:-in_sub(Y),in(X),connections(Y,X).\n")
      f.write("defeated(X) :- in_sub(Y), connections(Y,X).\n\n")
      f.write("in(X) :- not out(X), vertex(X).\n")
      f.write("out(X) :- not in(X), vertex(X).\n")
      f.write(":- in(X), in(Y), edge(X,Y).\n")
      f.write("defeated(X) :- in(Y), edge(Y,X).\n")
      f.write(":- out(X), not defeated(X).\n")
      f.write("#show in/1.")
      f.close()

def erdosGraph():
   d=0.1
   for j in list(range(3)):
      for i in list(range(100)):
         v=200
         e=int(round(d*v*(v-1)))
         #create a random directed graph with
         graph = Graph.Erdos_Renyi(v,m=e,directed=True)
         #path=r"C:\\Users\\Elena\\graphs\\erdosGraphs\\clingo"
         #os.chdir(path)
         writeInFile(graph,"ErdosGraphD%.1f-%d.lp"% (d,i))
         #path=r"C:\\Users\\Elena\\graphs\\erdosGraphs\\graphs"
         #os.chdir(path)
         graph.write_edgelist("ErdosGraphD%.1f-%d.txt"% (d,i))
      d+=0.1

def changeRatioOfSymmetricEdges(graph,symmetric):
   symmetricEdges=0
   #count symmetric edges
   for vStart in graph.vs:
      for vEnd in graph.vs:
         if(vStart.index<vEnd.index):
            if(graph.get_eid(vStart,vEnd, directed=True, error=False)!=-
1 and
            graph.get_eid(vEnd,vStart, directed=True, error=False)!=-1):
               symmetricEdges+=1
   #add symmetric edges for symmetric graph
   if symmetricEdges<symmetric:
      for vStart in graph.vs:
         for vEnd in graph.vs:
            if(vStart.index<vEnd.index):
               if(graph.get_eid(vStart,vEnd, directed=True, error=False)!
=-1 and
               graph.get_eid(vEnd,vStart, directed=True, error=False)==-
1):
                  symmetricEdges+=1
                  graph.add_edge(vEnd,vStart)
```

```
                if(symmetricEdges==symmetric):
                    break
            if(symmetricEdges==symmetric):
                break
    else:
        #delete symmetric edges for scale-free graph
        for vStart in graph.vs:
            for vEnd in graph.vs:
                if(vStart.index<vEnd.index):
                    if(graph.get_eid(vStart,vEnd, directed=True, error=False)!
=-1 and
                    graph.get_eid(vEnd,vStart, directed=True, error=False)!=-
1):
                        symmetricEdges-=1
                        if(random.random()>0.5):
                            graph.delete_edges(graph.get_eid(vEnd,vStart, direct
ed=True, error=False))
                        else:
                            graph.delete_edges(graph.get_eid(vStart,vEnd, direct
ed=True, error=False))
                        if(symmetricEdges==symmetric):
                            break
            if(symmetricEdges==symmetric):
                break

def SymmetricEdgesGraph():
    d=0.1
    for k in list(range(2)):
        ratio=10
        for i in list(range(3)):
            for j in list(range(100)):
                v=200
                e=int(round(d*v*(v-1)))
                graph = Graph.Erdos_Renyi(v,m=e,directed=True)
                SymmetricEdges=int(round((ratio*e)/100))
                changeRatioOfSymmetricEdges(graph,SymmetricEdges)
                #path=r"C:\\Users\\Elena\\graphs\\symmetricGraphs\\clingo"
                #os.chdir(path)
                writeInFile(graph,"SymmetricGraphD%.1f-Ratio%d-
%d.lp"% (d,ratio,j))
                #path=r"C:\\Users\\Elena\\graphs\\symmetricGraphs\\graphs"
                #os.chdir(path)
                graph.write_edgelist("SymmetricGraphD%.1f-Ratio%d-
%d.txt"% (d,ratio,j))
            ratio+=10
        d+=0.1

def kRegularGraph():
```

```
   kReg=10
   for i in list(range(2)):
      for j in list(range(100)):
         v=200
         graph = Graph.K_Regular(v,kReg,directed=True,multiple=False)
         #path=r"C:\\Users\\Elena\\graphs\\kRegularGraph\\clingo"
         #os.chdir(path)
         writeInFile(graph,"kRegularGraphK%d-%d.lp"% (kReg,j))
         #path=r"C:\\Users\\Elena\\graphs\\kRegularGraph\\graph"
         #os.chdir(path)
         graph.write_edgelist("kRegularGraphK%d-%d.txt"% (kReg,j))
      kReg+=5


def scaleFreeGraph():
   ratio=0
   for i in list(range(1)):
      for j in list(range(1)):
         v=200
         graph = Graph.Barabasi(v, 8, outpref=0, directed=0, power=0.3, z
ero_appeal=1, implementation="psumtree", start_from=None)

         summary(graph)
         print(graph.get_edgelist())
         print(graph.degree(vertices=0, mode=OUT, loops=True))
         print(graph.degree(vertices=1, mode=OUT, loops=True))
         print(graph.degree(vertices=2, mode=OUT, loops=True))
         graph.to_directed()
         SymmetricEdges=int(round((0*len(graph.es))/100))
         '''
         changeRatioOfSymmetricEdges(graph,SymmetricEdges)
         path=r"C:\\Users\\Elena\\graphs\\scaleFreeGraphs\\new\\clingo"
         os.chdir(path)
         writeInFile(graph,"scaleFreeGraphRatio0-%d.lp"% (j))
         path=r"C:\\Users\\Elena\\graphs\\scaleFreeGraphs\\new\\graph"
         os.chdir(path)
         graph.write_edgelist("scaleFreeGraphRatio0-%d.txt"% (j))'''


         SymmetricEdges=int(round((0.5*len(graph.es))/100))
         changeRatioOfSymmetricEdges(graph,SymmetricEdges)
         path=r"C:\\Users\\Elena\\graphs\\scaleFreeGraphs\\new\\clingo"
         os.chdir(path)
         writeInFile(graph,"scaleFreeGraphRatio05-%d.lp"% (j))
         path=r"C:\\Users\\Elena\\graphs\\scaleFreeGraphs\\new\\graph"
         os.chdir(path)
         graph.write_edgelist("scaleFreeGraphRatio05-%d.txt"% (j))
```

```
        SymmetricEdges=int(round((5*len(graph.es))/100))
        changeRatioOfSymmetricEdges(graph,SymmetricEdges)
        path=r"C:\\Users\\Elena\\graphs\\scaleFreeGraphs\\new\\clingo"
        os.chdir(path)
        writeInFile(graph,"scaleFreeGraphRatio5-%d.lp"% (j))
        path=r"C:\\Users\\Elena\\graphs\\scaleFreeGraphs\\new\\graph"
        os.chdir(path)
        graph.write_edgelist("scaleFreeGraphRatio5-%d.txt"% (j))

        SymmetricEdges=int(round((0.10*len(graph.es))/100))
        changeRatioOfSymmetricEdges(graph,SymmetricEdges)
        path=r"C:\\Users\\Elena\\graphs\\scaleFreeGraphs\\new\\clingo"
        os.chdir(path)
        writeInFile(graph,"scaleFreeGraphRatio010-%d.lp"% (j))
        path=r"C:\\Users\\Elena\\graphs\\scaleFreeGraphs\\new\\graph"
        os.chdir(path)
        graph.write_edgelist("scaleFreeGraphRatio010-%d.txt"% (j))


        SymmetricEdges=int(round((10*len(graph.es))/100))
        changeRatioOfSymmetricEdges(graph,SymmetricEdges)
        path=r"C:\\Users\\Elena\\graphs\\scaleFreeGraphs\\new\\clingo"
        os.chdir(path)
        writeInFile(graph,"scaleFreeGraphRatio10-%d.lp"% (j))
        path=r"C:\\Users\\Elena\\graphs\\scaleFreeGraphs\\new\\graph"
        os.chdir(path)
        graph.write_edgelist("scaleFreeGraphRatio10-%d.txt"% (j))
      ratio+=0.5

def controller():
   v=200
   cont_v=10
   for i in range(6):
      d=0.05
      for k in range(3):
         #os.environ['dir_name'] =("C:\\Users\\Elena\\graphs\\controllerG
raphs\\experiment6\\numVert%d\\ratio%d"%(cont_v,int(d*100)))
         #os.chdir(os.environ['dir_name'])
         for l in range(100):
            e=int(round(d*v*(v-1)))
            graph = Graph.Erdos_Renyi(v,m=e,directed=True)
            controller = Graph.Erdos_Renyi(cont_v,m=0,directed=True)
            vert=[]
            connections = [[0 for x in range(2)] for y in range(len(contr
oller.vs))]
            union_arg=Graph.disjoint_union(graph,controller)
            count=0
```

```
            for v1 in controller.vs:
                  v2=random.randint(1,len(graph.vs)-1)
                  while(v2 in vert):
                     v2=random.randint(1,len(graph.vs)-1)
                  vert.append(v2)
                  connections[count][0]=(v1.index+len(graph.vs))
                  connections[count][1]=v2
                  union_arg.add_edge((v1.index+len(graph.vs)),v2)
                  count+=1
            for in_v in range(200):
               writeInFileController(controller,graph,connections,in_v,"G
raphV%d-D%.1f-In%d-%d.lp"%(cont_v,d,in_v,l))
         d+=0.05
      cont_v+=10

def controller_combs():
   v=200
   d=0.05
   for k in range(3):
      for l in range(100):
         e=int(round(d*v*(v-1)))
         graph = Graph.Erdos_Renyi(v,m=e,directed=True)
         controller_vs=[i for i in range(200,220)]
         vert=[]
         connections = [[0 for x in range(2)] for y in range(len(controll
er_vs))]
         count=0
         for v1 in controller_vs:
               v2=random.randint(1,len(graph.vs)-1)
               while(v2 in vert):
                  v2=random.randint(1,len(graph.vs)-1)
               vert.append(v2)
               connections[count][0]=v1
               connections[count][1]=v2
               count+=1
         #os.environ['dir_name'] =("C:\\Users\\Elena\\graphs\\sensitivity
2\\ccombs1\\density%d"%(int(d*100)))
         #os.chdir(os.environ['dir_name'])
         writeInFileControllerCombs(controller_vs,graph,connections,1,d,l
)
         #os.environ['dir_name'] =("C:\\Users\\Elena\\graphs\\sensitivity
2\\ccombs2\\density%d"%(int(d*100)))
         #os.chdir(os.environ['dir_name'])
         writeInFileControllerCombs(controller_vs,graph,connections,2,d,l
)
         #os.environ['dir_name'] =("C:\\Users\\Elena\\graphs\\sensitivity
2\\ccombs3\\density%d"%(int(d*100)))
         #os.chdir(os.environ['dir_name'])
```

```
        writeInFileControllerCombs(controller_vs,graph,connections,3,d,l
)
      d+=0.05

print("1: Erdos Graph\t\t2: k-Regular Graph\t3: Symmetric Graph")
print("4: Scale-Free Graph\t5: Controller Graph\t6: Controller-
Combinations Graph")
num = int(input("Choose the graph which you want to create: "))
if(num==1):
   erdosGraph()
elif(num==2):
   kRegularGraph()
elif(num==3):
   SymmetricEdgesGraph()
elif(num==4):
   scaleFreeGraph()
elif(num==5):
   controller()
elif(num==6):
   controller_combs()
```

Παράρτημα Β

Σε αυτό το παράρτημα παρατίθεται τα προγράμματα που αφορούν την εξαγωγή των στατιστικών από τα αποτελέσματα της Clingo.

Β.1 Πρόγραμμα για την Φάση 1 και 2

```
import re
import os
def writeInFile(filename,extensions,arguments,zero,text):
    f= open(filename,"a+")
    f.write("\n-------------------------------------\n\n")
    f.write("%s\n"%(text))
    f.write("Number of stable extensions: %d\n" % (extensions))
    f.write("Number of stable extensions/100: %.2f\n" % (extensions/100))
    f.write("Number of arguments: %d\n" % (arguments))
    f.write("Number of arguments/100: %.2f\n" % (arguments/100))
    f.write("Number of zero stable extensions: %d\n" % (zero))
    f.close()

def writeInFileController(filename,sum_sat,sum_unsat,density,numVertex):
    f= open(filename,"a+")
    f.write("\n---------------\n\n")
    f.write("NumVertex of controller:%d\n"%(numVertex))
    f.write("Density of the main graph:%d\n\n"%(density))
    f.write("Number of Satisfiable: %.d\n" % (sum_sat))
    f.write("Average of Satisfiable (out of 200): %.2f\n" % (sum_sat/100)
)
    f.write("Number of Unsatisfiable: %d\n" % (sum_unsat))
    f.write("Average of Unatisfiable (out of 200): %.2f\n" % (sum_unsat/1
00))
    f.close()

def getResults(graph_name):
    if(graph_name=="ErdosGraph"):
        readFile="outputErdos.txt"
        writeFile="StatisticErdos.txt"
        path=r"C:\\Users\\Elena\\graphs\\erdosGraphs\\clingo"
        os.chdir(path)
        k=0.1
        k_add=0.1
        text="Density: "
```

```
    elif(graph_name=="kRegularGraph"):
        readFile="OutputKRegular.txt"
        writeFile="StatistickRegular.txt"
        path=r"C:\\Users\\Elena\\graphs\\KRegularGraphs\\clingo"
        os.chdir(path)
        k=10
        k_add=5
        text="K: "

    elif(graph_name=="SymmetricGraph"):
        readFile="OutputSymmetric.txt"
        writeFile="StatisticSymmetric.txt"
        path=r"C:\\Users\\Elena\\graphs\\symmetricGraphs\\clingo"
        os.chdir(path)
        k=0.1
        k_add=0.1
        l=10
        l_add=10
        text="Density: "
        textRatio="Ratio: "

    elif(graph_name=="ScaleFreeGraph"):
        readFile="OutputScaleFreeRatio0510.txt"
        writeFile="StatisticsFreeRatio0510.txt"
        path=r"C:\\Users\\Elena\\graphs\\scaleFreeGraphs\\new\\clingo"
        os.chdir(path)
        k=0
        k_add=5
        text="Ratio: "

    f= open(readFile,"r")
    array=[]
    i=zero=extensions=arguments=0
    for x in f:
        array.append(x)
        if(array[i].find("Models       :")==0):
            num=int(re.search(r'\d+', array[i]).group())
            if(num==0):
                zero+=1
            else:
                extensions+=num
        elif(array[i].find("in(")==0):
            arguments+=array[i].count("in(")
        elif(array[i].find("-------------")==0):
            text1=text+"%.1f"%(k)
            if(graph_name=="SymmetricGraph"):
                text1=text+"%.1f"%(k)+"\n"+textRatio+"%d"%(l)
            writeInFile(writeFile,extensions,arguments,zero,text1)
```

```
            zero=extensions=arguments=0
            if(graph_name=="SymmetricGraph"):
                l+=l_add
                if(l>20):
                    l=10
                    k+=k_add
            else:
                k+=k_add
        i+=1
    f.close

def getResultsController(readFile,writeFile,density,numVertex):
    f= open(readFile,"r")
    array=[]
    i=sum_sat=sum_unsat=0
    for x in f:
        array.append(x)
        if(array[i].find("Answer: 1")==0):
            sum_sat+=1
        if(array[i].find("UNSATISFIABLE")==0):
            sum_unsat+=1
        i+=1
    f.close
    path=r"C:\\Users\\Elena\\graphs\\controllerGraphs\\experiment6"
    os.chdir(path)
    writeInFileController(writeFile,sum_sat,sum_unsat,density,numVertex)

'''
path_str="C:\\Users\\Elena\\graphs\\\scaleFreeGraphs"
os.environ['dir_name'] = path_str
os.chdir(os.environ['dir_name'])   '''

phase = int(input("Do you want to get results for Phase 1 or Phase 2 (1)
or (2): "))
if(phase==1):
    print("1: Erdos Graph\t2: k-
Regular Graph\t3: Symmetric Graph\t4: Scale-Free Graph")
    num = int(input("Choose the graph which you want to get results: "))
    if(num==1):
        getResults("ErdosGraph")
    elif(num==2):
        getResults("kRegularGraph")
    elif(num==3):
        getResults("SymmetricGraph")
    elif(num==4):
        getResults("ScaleFreeGraph")
    numVertex=10
elif(phase==2):
```

```
    for i in range(6):
        density=5
        for j in range(3):
            #path_str="C:\\Users\\Elena\\graphs\\controllerGraphs\\experi
ment6\\numVert"+str(numVertex)+"\\ratio"+str(density)
            #os.environ['dir_name'] = path_str
            readFile="Output_GraphV"+str(numVertex)+"-
Ratio"+str(density)+".txt"
            writeFile="Statistics_Experiment6.txt"
            os.chdir(os.environ['dir_name'])
            getResultsController(readFile,writeFile,density,numVertex)
            density+=5
        numVertex+=10
```

B.2.1 Πρόγραμμα για την Φάση 3 (για την πρώτη προσέγγιση)

```
from igraph import *
import re
import os

#find the N max elements
def NmaxElements(list, N):
    list1=list.copy()
    max_list = []

    for i in range(N):
        max = 0
        max_ind=0

        for j in range(len(list1)):
            if list1[j] > max:
                max = list1[j];
                max_ind=j

        list1[int(max_ind)]=0;
        max_list.append(max_ind)
    return max_list

#find the N min elements
def NminElements(list, N):
    min_list = []
    list2=list.copy()
```

```
    for i in range(N):
        min = 1000000
        min_ind=0

        for j in range(len(list2)):
            if list2[j] < min:
                min = list2[j];
                min_ind=j

        list2[int(min_ind)]=1000000;
        min_list.append(min_ind)
    return min_list

#write the 20 max nodes and 20 min nodes in file
def writeInFileMaxMin(filename,write_path,array,description,count,text):
    os.environ['dir_name'] =write_path
    os.chdir(os.environ['dir_name'])
    f= open(filename,"a+")
    f.write(description)
    f.write("\n-------------------------------------------------\n")

    f.write("\nAvg Inner Join Min Occurences with Min "+text+" ("+str(arr
ay[0])+") ")
    f.write("\nAvg nner Join Min Occurences with Max "+text+" ("+str(arra
y[1])+") ")
    f.write("\nAvg Inner Join Max Occurences with Min "+text+" ("+str(arr
ay[2])+") ")
    f.write("\nAvg Inner Join Max Occurences with Max "+text+" ("+str(arr
ay[3])+") ")
    f.close()

#find the occurence of each vertex in stable extensions
# (reads the output of clingo in erdosGraph with density 0.2)
def findOccurences(filename,readFileClingo,read_path,write_path):
    os.environ['dir_name'] =read_path
    os.chdir(os.environ['dir_name'])
    f= open(readFileClingo,"r")
    array=[]
    i=0
    k=10
    dens1=0.2
    dens=30
    count=[0 for i in range(200)]
    graph_num=0
    #[0]Avg count inner Join Min Occurences with Min Betweenness/closenes
s/degree
    #[1]Avg count inner Join Min Occurences with Max Betweenness/closenes
s/degree
```

```
    #[2]Avg count inner Join Max Occurences with Min Betweenness/closenes
s/degree
    #[3]Avg count inner Join Max Occurences with Max Betweenness/closenes
s/degree
    closeness_avg=[0 for i in range(4)]
    betweenes_avg=[0 for i in range(4)]
    degree_avg=[0 for i in range(4)]
    for x in f:
        array.append(x)
        #take the number X of the string in(X)
        index=array[i].find("in(")
        indexlast=array[i].find(")")
        if(array[i].find("in(")==0 ):
            indEnd=indBeg=j=0
            while(indBeg!=-1):
                indEnd=array[i].find(")",indBeg)
                a=array[i][indBeg+3:indEnd]
                #increase the occurence of this node
                count[int(a)]+=1
                indBeg=array[i].find("in(",indEnd)
        if(array[i].find("Models")==0):
            #get the betweenness,closeness and degree of graphs with grap
hs
            betweenes_avg,closeness_avg,degree_avg=getCharact1("Symmetric
GraphD"+str(round(dens1, 1))+"-Ratio"+str(round(dens, 1))+"-
",read_path,graph_num,count,betweenes_avg,closeness_avg,degree_avg)
            graph_num+=1
            count=[0 for j in range(200)]
        #if i found the string "-------------
" means that the density or ratio is changed
        #so if the density of ratio change im saving the results that i f
ound until now
        #and write the results in files
        #also i will find the betweenness,closeness and degree of graphs
with this ratio or density
        if(array[i].find("-------------")==0 ):
            print(str(round(dens1, 1)))
            print(str(round(dens, 1)))
            filenameMinMax="MinMax"+filename+"2.txt"
            filenameChar="Characteristics"+filename+"1.txt"

            str_betw="\n\nBetweenness of each node "
            str_clos="\n\nCloseness of each node "
            str_deg="\n\nDegree of each node "

            for j in range(len(betweenes_avg)):
                betweenes_avg[j]/=100
                closeness_avg[j]/=100
```

```
                degree_avg[j]/=100
            # writeInFileMaxMin(filenameMinMax,write_path,count,str_occur,
count,"Occurences")
            writeInFileMaxMin(filenameMinMax,write_path,betweenes_avg,str
_betw,count,"Betweenness",)
            writeInFileMaxMin(filenameMinMax,write_path,closeness_avg,str
_clos,count,"Closeness")
            writeInFileMaxMin(filenameMinMax,write_path,degree_avg,str_de
g,count,"Degree")
            closeness_avg=[0 for i in range(4)]
            betweenes_avg=[0 for i in range(4)]
            degree_avg=[0 for i in range(4)]
            dens+=10
            if(dens==40):
                dens=10
                dens1+=0.1
            graph_num=0
        i+=1
    f.close
    return count

def getCharact1(filename,read_path,graph_num,count,betweenes_avg,closenes
s_avg,degree_avg):
    os.environ['dir_name'] =read_path
    os.chdir(os.environ['dir_name'])
    graph=Graph.Read_Edgelist(f=filename+str(graph_num)+".txt", directed=
True)
    graph.add_vertices(200-len(graph.vs))
    betweeness=graph.betweenness(vertices=None, directed=True, cutoff=Non
e, weights=None, nobigint=True)
    closeness=graph.closeness(vertices=None, mode=ALL, cutoff=None, weigh
ts=None, normalized=True)
    degree=graph.degree(vertices=None, mode=ALL, loops=True)

    #write the join graphs
    max_count=NmaxElements(count, 20)
    min_count=NminElements(count, 20)
    min=NmaxElements(betweeness, 20)
    max=NminElements(betweeness, 20)
    inner_join_minmin=list(set(min_count) & set(min))
    inner_join_minmax=list(set(min_count) & set(max))
    inner_join_maxmin=list(set(max_count) & set(min))
    inner_join_maxmax=list(set(max_count) & set(max))
    betweenes_avg[0]+=len(inner_join_minmin)
    betweenes_avg[1]+=len(inner_join_minmax)
    betweenes_avg[2]+=len(inner_join_maxmin)
    betweenes_avg[3]+=len(inner_join_maxmax)
```

```
    min=NmaxElements(closeness, 20)
    max=NminElements(closeness, 20)
    inner_join_minmin=list(set(min_count) & set(min))
    inner_join_minmax=list(set(min_count) & set(max))
    inner_join_maxmin=list(set(max_count) & set(min))
    inner_join_maxmax=list(set(max_count) & set(max))
    closeness_avg[0]+=len(inner_join_minmin)
    closeness_avg[1]+=len(inner_join_minmax)
    closeness_avg[2]+=len(inner_join_maxmin)
    closeness_avg[3]+=len(inner_join_maxmax)

    min=NmaxElements(degree, 20)
    max=NminElements(degree, 20)
    inner_join_minmin=list(set(min_count) & set(min))
    inner_join_minmax=list(set(min_count) & set(max))
    inner_join_maxmin=list(set(max_count) & set(min))
    inner_join_maxmax=list(set(max_count) & set(max))
    degree_avg[0]+=len(inner_join_minmin)
    degree_avg[1]+=len(inner_join_minmax)
    degree_avg[2]+=len(inner_join_maxmin)
    degree_avg[3]+=len(inner_join_maxmax)

    return betweenes_avg,closeness_avg,degree_avg

#filename="Erdos" for Erdos Graphs "Symmetric" for Symmetric Graphs
#filename="KRegular" for k Regular Graphs "ScaleFree" for Scale Free Grap
hs
filename="Symmetric"
#readFileClingo is the file name of the output of clingo
readFileClingo="outputSymmetric.txt"
#read_path is the path which i have all the graphs and output file of cli
ngo
read_path="C:\\Users\\Elena\\graphs\\getCharacteristics\\symmetric"
#write_path is the path which i want to save the result
write_path="C:\\Users\\Elena\\graphs\\getCharacteristics"
occur=findOccurences(filename,readFileClingo,read_path,write_path)
```

B.2.2 Πρόγραμμα για την Φάση 3 (για την δεύτερη προσέγγιση)

```
from igraph import *
import re
import os

#find the N max elements
def NmaxElements(list, N):
    list1=list.copy()
    max_list = []

    for i in range(N):
        max = 0
        max_ind=0

        for j in range(len(list1)):
            if list1[j] > max:
                max = list1[j];
                max_ind=j

        list1[int(max_ind)]=0;
        max_list.append(max_ind)
    return max_list

#find the N min elements
def NminElements(list, N):
    min_list = []
    list2=list.copy()

    for i in range(N):
        min = 1000000
        min_ind=0

        for j in range(len(list2)):
            if list2[j] < min:
                min = list2[j];
                min_ind=j

        list2[int(min_ind)]=1000000;
        min_list.append(min_ind)
    return min_list

def writeInFile(filename,description,betw_nums,clos_nums,deg_nums,betw_av
g,clos_avg,deg_avg):
    f= open(filename,"a+")
    f.write(description)
    f.write("-----------------------------------\n\n")
```

```
    f.write("Max betweenneess/100: %.4f\nAverage betweenneess of 20 max n
odes: %.4f\n" % (betw_nums[0]/100,betw_avg[0]/100))
    f.write("Average of occurences of each node: %.4f\n" % (betw_nums[1]/
100))
    f.write("Min betweenneess/100: %.4f\nAverage betweenneess of 20 min n
odes: %.4f\n\n" % (betw_nums[2]/100,betw_avg[1]/100))

    f.write("Max closeness/100: %.4f\nAverage closeness of 20 max nodes:
%.4f\n" % (clos_nums[0]/100,clos_avg[0]/100))
    f.write("Average of occurences of each node: %.4f\n" % (clos_nums[1]/
100))
    f.write("Min closeness/100: %.4f\nAverage closeness of 20 min nodes:
%.4f\n\n" % (clos_nums[2]/100,clos_avg[1]/100))

    f.write("Max degree/100: %.4f\nAverage degree of 20 max nodes: %.4f\n
" % (deg_nums[0]/100,deg_avg[0]/100))
    f.write("Average of occurences of each node: %.4f\n" % (deg_nums[1]/1
00))
    f.write("Min degree/100: %.4f\nAverage degree of 20 min nodes: %.4f\n
\n\n" % (deg_nums[2]/100,deg_avg[1]/100))
    f.close()

def findAvg(array,array_avg):
    max_array=NmaxElements(array, 20)
    min_array=NminElements(array, 20)
    sum=0
    #find the average betweeness,closeness and degree of the 20 max nodes
    for item in max_array:
        sum=sum+array[int(item)]
    array_avg[0]=array_avg[0]+(sum/len(max_array))
    sum=0
    #find the average betweeness,closeness and degree of the 20 min nodes
    for item in min_array:
        sum=sum+array[int(item)]
    array_avg[1]=array_avg[1]+(sum/len(min_array))

def find_perc(stable_extensions,array,array_numbers):
    max_array=NmaxElements(array, 20)
    min_array=NminElements(array, 20)
    sum_occur=0
    #num occurence of the 20 max/20
    for item in max_array:
        sum_occur+=stable_extensions.count(item)
    num1=sum_occur/len(max_array)

    #sum occurences of all nodes/200
    num2=len(stable_extensions)/200
```

```
    #num occurence of the 20 min/20
    sum_occur=0
    for item in min_array:
        sum_occur+=stable_extensions.count(item)
    num3=sum_occur/len(min_array)

    array_numbers[0]+=num1
    array_numbers[1]+=num2
    array_numbers[2]+=num3


#find the occurence of each vertex in stable extensions
# (reads the output of clingo in erdosGraph with density 0.2)
def characteristics(graph_type,readFileClingo,read_path):
    os.environ['dir_name'] =read_path
    os.chdir(os.environ['dir_name'])
    f= open(readFileClingo,"r")
    array=[]
    i=0
    ratio=0
    k=0
    dens=0
    graph_num=0
    count=[0 for i in range(200)]
    betw_nums=[0 for i in range(3)]
    clos_nums=[0 for i in range(3)]
    deg_nums=[0 for i in range(3)]

    betw_avg=[0 for i in range(2)]
    clos_avg=[0 for i in range(2)]
    deg_avg=[0 for i in range(2)]

    stable_extensions=[]

    if(graph_type=="Erdos"):
        dens=0.1
    elif(graph_type=="KRegular"):
        k=10
    elif(graph_type=="ScaleFree"):
        ratio=0
    elif(graph_type=="Symmetric"):
        dens=0.1
        ratio=10

    for x in f:
        array.append(x)
        #take the number X of the string in(X)
        index=array[i].find("in(")
```

```
        indexlast=array[i].find(")")
        if(array[i].find("in(")==0 ):
            indEnd=indBeg=j=0
            while(indBeg!=-1):
                indEnd=array[i].find(")",indBeg)
                a=array[i][indBeg+3:indEnd]
                #increase the occurence of this node
                count[int(a)]+=1
                stable_extensions.append(int(a))
                indBeg=array[i].find("in(",indEnd)

        #CPU Time is the last line of each graph clingo output
        if(array[i].find("CPU")==0):
            #read the graph
            if(graph_type=="Erdos"):
                graph=Graph.Read_Edgelist(f="ErdosGraphD"+str(round(dens,
 1))+"-"+str(graph_num)+".txt", directed=True)
            elif(graph_type=="KRegular"):
                graph=Graph.Read_Edgelist(f="kRegularGraphK"+str(k)+"-
"+str(graph_num)+".txt", directed=True)
            elif(graph_type=="ScaleFree"):
                graph=Graph.Read_Edgelist(f="scaleFreeGraphRatio"+str(rou
nd(ratio, 1))+"-"+str(graph_num)+".txt", directed=True)
            elif(graph_type=="Symmetric"):
                graph=Graph.Read_Edgelist(f="SymmetricGraphD"+str(round(d
ens, 1))+"-Ratio"+str(ratio)+"-"+str(graph_num)+".txt", directed=True)

            graph.add_vertices(200-len(graph.vs))
            #calculate the betweenneess,closeness and degree
            betweeness=graph.betweenness(vertices=None, directed=True, cu
toff=None, weights=None, nobigint=True)
            closeness=graph.closeness(vertices=None, mode=ALL, cutoff=Non
e, weights=None, normalized=True)
            degree=graph.degree(vertices=None, mode=ALL, loops=True)

            #calculate the average betweeness,closeness and degree of the
 20 max nodes and 20 min nodes
            findAvg(betweeness,betw_avg)
            findAvg(closeness,clos_avg)
            findAvg(degree,deg_avg)

            #calculate the 3 numbers for betweeness, closeness and degree
            find_perc(stable_extensions,betweeness,betw_nums)
            find_perc(stable_extensions,closeness,clos_nums)
            find_perc(stable_extensions,degree,deg_nums)

            #zero the occurences of the nodes and empty the stable extens
ions
```

```
            count=[0 for i in range(200)]
            stable_extensions=[]
            graph_num+=1

        #if i found the string "------------
" means that the density or ratio is changed
        #so if the density of ratio change im saving the results that i f
ound until now
        #and write the results in files
        if(array[i].find("-------------")==0 ):
            graph_num=0
            if(graph_type=="Erdos"):
                description="\nErdos Graphs\nDensity:"+str(round(dens, 1)
)+"\n"
                writeInFile("CharacteristicErdos1.txt",description,betw_n
ums,clos_nums,deg_nums,betw_avg,clos_avg,deg_avg)
                dens+=0.1
            elif(graph_type=="KRegular"):
                description="\nKRegular Graphs\nK:"+str(k)+"\n"
                writeInFile("CharacteristicKRegular.txt",description,betw
_nums,clos_nums,deg_nums,betw_avg,clos_avg,deg_avg)
                k+=5
            elif(graph_type=="ScaleFree"):
                description="\nScale Free Graphs\nRatio:"+str(ratio)+"\n"
                writeInFile("CharacteristicScaleFree.txt",description,bet
w_nums,clos_nums,deg_nums,betw_avg,clos_avg,deg_avg)
                ratio+=5
            elif(graph_type=="Symmetric"):
                description="\nSymmetric Graphs\nDensity:"+str(round(dens
, 1))+"\nRatio:"+str(ratio)+"\n"
                writeInFile("CharacteristicSymmetric.txt",description,bet
w_nums,clos_nums,deg_nums,betw_avg,clos_avg,deg_avg)
                ratio+=10
                if(ratio==40):
                    ratio=10
                    dens+=0.1
            betw_nums=[0 for i in range(3)]
            clos_nums=[0 for i in range(3)]
            deg_nums=[0 for i in range(3)]
        i+=1
    f.close()
    return count

#read_path is the path which i have all the graphs and output file of cli
ngo saved
'''
read_path="C:\\Users\\Elena\\graphs\\getCharacteristics\\kRegular"
characteristics("KRegular","outputKRegular.txt",read_path)
```

```
'''
read_path="C:\\Users\\Elena\\graphs\\getCharacteristics\\scaleFree"
characteristics("ScaleFree","outputScaleFree.txt",read_path)
'''
read_path="C:\\Users\\Elena\\graphs\\getCharacteristics\\symmetric"
characteristics("Symmetric","outputSymmetric1.txt",read_path)

read_path="C:\\Users\\Elena\\graphs\\getCharacteristics\\erdos"
characteristics("Erdos","outputErdos.txt",read_path)
'''
```

B.3.1 Πρόγραμμα για την Φάση 4 (Για την επιλογή ενός κόμβου)

```
import random
import copy
import os
import numpy as np
from sklearn.metrics.pairwise import cosine_similarity
from scipy import sparse
def writeInFile(filename,pairSimilarity,average,density):
   f= open(filename,"w")
   f.write("Density:"+str(density/100)+"\n")
   f.write("----------------------------\n")
   f.write("Max 20: ")
   max_array=NmaxElements(pairSimilarity, 20)
   for i in range(len(max_array)):
      if(i==0):
         f.write(max_array[i])
      else:
         f.write(", "+max_array[i])
   f.write("\n----------------------------\n")
   f.write("Min 20: ")
   min_array=NminElements(pairSimilarity, 20)
   for i in range(len(min_array)):
      if(i==0):
         f.write(min_array[i])
      else:
         f.write(", "+min_array[i])
   f.write("\n----------------------------\n")
   f.write("Average: "+str(average)+"\n")
   f.write("----------------------------\n")
   for i in range(len(pairSimilarity)):
      for j in range(i+1,len(pairSimilarity)):
         f.write(str(200+i)+"-
"+str(200+j)+": "+str(pairSimilarity[i][j])+"\n")
      f.write("\n")
```

```
   f.close()

#find the N max elements of a 2d array
def NmaxElements(list, N):
    list1 = copy.deepcopy(list)
    max_list = []
    for i in range(N):
        max = 0
        max_row=0
        max_col=0

        for j in range(len(list1)):
            for k in range(len(list1)):
               if list1[j][k] > max:
                  max = list1[j][k];
                  max_row=j
                  max_col=k
        list1[int(max_row)][int(max_col)]=0;
        max_list.append(str(max_row+200)+"-"+str(max_col+200))
    return max_list

#find the N min elements
def NminElements(list, N):
    min_list = []
    list2 = copy.deepcopy(list)
    for i in range(N):
        min = 1000000
        min_row=0
        min_col=0
        for j in range(len(list2)):
            for k in range(j+1,len(list2)):
               if list2[j][k] < min:
                  min = list2[j][k];
                  min_row=j
                  min_col=k
        list2[int(min_row)][int(min_col)]=1000000;
        min_list.append(str(min_row+200)+"-"+str(min_col+200))
    return min_list

def cosineSimilarity(array1,array2):
   a = np.array(array1)
   b = np.array(array2)
   # manually compute cosine similarity
   #cos=a*b/(||A||*||B||)
   dot = np.dot(a, b)
   norma = np.linalg.norm(a)
   normb = np.linalg.norm(b)
   if(norma * normb==0):
```

```
      return 0
   cos = dot / (norma * normb)
   return cos

def findSimilarities(count):
   #pairSimilarity[i][j]=Cosine similarity between two vectors a and b
   #a=the occurences of 200 nodes when the i node is selected from the co
ntroller graph
   #b=the occurences of 200 nodes when the j node is selected from the co
ntroller graph
   #calculate the cosine similarity of each pair of node which is selecte
d from the controller graph
   #sum tha similarity and find the average similarity for each pair
   pairSimilarity=[[0 for i in range(20)] for j in range(20)]
   count_i=0
   similarity_avg=0
   for i in range(100):
      for j in range(20):
         for k in range(j+1,20):
            pairSimilarity[j][k]+=cosineSimilarity(count[i][j],count[i][k
])
   #find the average of the cosine similarity
   for j in range(20):
      for k in range(j+1,20):
         pairSimilarity[j][k]=pairSimilarity[j][k]/100.0
         similarity_avg+=pairSimilarity[j][k]
         count_i+=1
   similarity_avg=similarity_avg/count_i
   return pairSimilarity,similarity_avg

#calculate the euclidean distance
def euclideanDistance(array1,array2):
   a = np.array(array1)
   b = np.array(array2)
   dist = np.linalg.norm(a-b)
   return dist

def calculateEuclideanDistance(count):
   #pairDistance[i][j]=Euclidean Distance between two vectors a and b
   #a=the occurences of 200 nodes when the i node is selected from the co
ntroller graph
   #b=the occurences of 200 nodes when the j node is selected from the co
ntroller graph
   #calculate the euclidean distance of each pair of node which is select
ed from the controller graph
   #sum tha euclidean distance
   pairDistance=[[0 for i in range(20)] for j in range(20)]
   count_i=0
```

```
   distance_avg=0
   for i in range(100):
      for j in range(20):
         for k in range(j+1,20):
            pairDistance[j][k]+=euclideanDistance(count[i][j],count[i][k]
)
   #find the average of the euclidean distance
   for j in range(20):
      for k in range(j+1,20):
         pairDistance[j][k]=pairDistance[j][k]/100.0
         distance_avg+=pairDistance[j][k]
         count_i+=1
   distance_avg=distance_avg/count_i
   return pairDistance,distance_avg

def getResults():
   for density in range(5,20,5):
      os.environ['dir_name'] =("C:\\Users\\Elena\\graphs\\sensitivity\\de
nsity%d"%(density))
      os.chdir(os.environ['dir_name'])
      f= open("OutputSensitivityDensity"+str(density)+".txt","r")
      array=[]
      i=0
      j=0
      h=0
      #count[i][j][k] the occurence of the node k when the j node is sele
cted for the i-th graph
      #from the controller graph (i:0-99)(j:200-219)(k:0-199)
      count=np.zeros((100, 20, 200))
      for x in f:
         array.append(x)
         #take the number X of the string in(X)
         index=array[h].find("in(")
         indexlast=array[h].find(")")
         if(array[h].find("in(")==0 ):
            indEnd=0
            indBeg=0
            while(indBeg!=-1):
                  indEnd=array[h].find(")",indBeg)
                  k=array[h][indBeg+3:indEnd]
                  #increase the occurence of this node
                  count[int(i)][int(j)][int(k)]+=1
                  #stable_extensions.append(int(a))
                  indBeg=array[h].find("in(",indEnd)
         if(array[h].find("Models")==0):
            i=i+1
         if(array[h].find("-------------")==0):
            j=j+1
```

```
            i=0
         h+=1
      f.close
      pairSimilarity,similarity_avg=findSimilarities(count)
      pairDistance,distance_avg=calculateEuclideanDistance(count)
      writeInFile("DistanceDensity"+str(density)+".txt",pairDistance,dist
ance_avg,density)
      writeInFile("SimilarityDensity"+str(density)+".txt",pairSimilarity,
similarity_avg,density)
getResults()
```

B.3.2 Πρόγραμμα για την Φάση 4 (Για την επιλογή δύο κόμβων)

```
import random
import copy
import os
import numpy as np
from sklearn.metrics.pairwise import cosine_similarity
from scipy import sparse
from scipy.special import comb
import re

def writeInFile(filename,array,average,combs2_labels,density):
   f= open(filename,"w")
   f.write("Density:"+str(density/100)+"\n")
   f.write("----------------------------\n")

   f.write("Max 20:")
   max_array=NmaxElements(array, 20)
   for i in range(len(max_array)):
      if(i==0):
         f.write(combs2_labels[int(max_array[i])])
      else:
         f.write(", "+combs2_labels[int(max_array[i])])

   f.write("\n----------------------------\n")
   f.write("Min 20:")
   min_array=NminElements(array, 20)
   for i in range(len(min_array)):
      if(i==0):
         f.write(combs2_labels[int(min_array[i])])
      else:
         f.write(", "+combs2_labels[int(min_array[i])])

   f.write("\n----------------------------\n")
   f.write("Average: "+str(average)+"\n")
```

```
   f.write("----------------------------\n")

   for i in range(len(combs2_labels)):
      f.write(str(combs2_labels[i])+": "+str(array[i])+"\n")
   f.close()

#find the N max elements
def NmaxElements(list, N):
    list1 = copy.deepcopy(list)
    max_list = []

    for i in range(N):
        max = 0
        max_index=0

        for i in range(len(list1)):
            if list1[i] > max:
               max = list1[i];
               max_index=i

        list1[int(max_index)]=0;
        max_list.append(max_index)
    return max_list

#find the N min elements
def NminElements(list, N):
    min_list = []
    list2 = copy.deepcopy(list)

    for i in range(N):
      min = 1000000
      min_index=0

      for i in range(len(list2)):
         if list2[i] < min and list2[i]>0:
            min = list2[i];
            min_index=i

      list2[int(min_index)]=9999999999;
      min_list.append(min_index)
    return min_list

#calculate the euclidean distance
def euclideanDistance(array1,array2):
   a = np.array(array1)
   b = np.array(array2)

   dist = np.linalg.norm(a-b)
```

```
   return dist

#calculate the cosine similarity
def cosineSimilarity(array1,array2):
   a = np.array(array1)
   b = np.array(array2)

   # manually compute cosine similarity
   #cos=a*b/(||A||*||B||)
   dot = np.dot(a, b)
   norma = np.linalg.norm(a)
   normb = np.linalg.norm(b)

   if(norma * normb==0):
      return 0
   cos = dot / (norma * normb)

   return cos
def calculateEuclideanDistance(count,pairDistance,combs2_labels):
   #there is a connection between the list combs2_labels and pairDistance
   #pairDistance[i] is the euclidean distance of the label combs2_labels[
i]
   #e.g. combs2_labels[0]="200,201-200,202" -
> pairDistance[0]=the euclidean distance between vector a and b
   #a=count[0][1]=the occurences of 200 nodes when the 200 and 201 node i
s selected from the controller graph
   #b=count[0][2]=the occurences of 200 nodes when the 200 and 202 node i
s selected from the controller graph
   #calculate the euclidean distance of each pair of node which is select
ed from the controller graph
   for i in range(len(combs2_labels)):
      #extract all the numbers from the string
      #e.g. str=200,201-200,202 -> tokens=[200,201,200,202]
      tokens=re.split(r'\W+',combs2_labels[i])
      #e.g. count[0][1]
      a=count[int(tokens[0])-200][int(tokens[1])-200]
      #e.g. count[0][2]
      b=count[int(tokens[2])-200][int(tokens[3])-200]
      pairDistance[i]+=euclideanDistance(a,b)
   return pairDistance

def calculateCosineSimilarity(count,pairSimilarity,combs2_labels):
   #there is a connection between the list combs2_labels and pairSimilari
ty
   #pairSimilarity[i] is the cosine similarity of the label combs2_labels
[i]
```

```
   #e.g. combs2_labels[0]="200,201-200,202" -
> pairSimilarity[0]=the cosine similarity between vector a and b
   #a=the occurences of 200 nodes when the 200 and 201 node is selected f
rom the controller graph
   #b=the occurences of 200 nodes when the 200 and 202 node is selected f
rom the controller graph

   #calculate the cosine similarity of each pair of node which is selecte
d from the controller graph
   for i in range(len(combs2_labels)):
      #extract all the numbers from the string
      #e.g. str=200,201-200,202 -> tokens=[200,201,200,202]
      tokens=re.split(r'\W+',combs2_labels[i])
      a=count[int(tokens[0])-200][int(tokens[1])-200]
      b=count[int(tokens[2])-200][int(tokens[3])-200]
      pairSimilarity[i]+=cosineSimilarity(a,b)
   return pairSimilarity

#calculate and find the averages of cosine similarity and euclidean dista
nce
def calculateAverages(pairSimilarity,pairDistance):
   similarity_avg=0
   distance_avg=0
   for i in range(len(pairSimilarity)):
      pairSimilarity[i]=pairSimilarity[i]/100.0
      similarity_avg+=pairSimilarity[i]
   similarity_avg=similarity_avg/len(pairSimilarity)
   for i in range(len(pairDistance)):
      pairDistance[i]=pairDistance[i]/100.0
      distance_avg+=pairDistance[i]
   distance_avg=distance_avg/len(pairDistance)
   return pairSimilarity,pairDistance,similarity_avg,distance_avg

def getResults():
   for density in range(5,20,5):
      os.environ['dir_name'] =("C:\\Users\\Elena\\graphs\\sensitivity2\\c
ombs2")
      os.chdir(os.environ['dir_name'])
      f= open("OutputSensitivityDensity"+str(density)+".txt","r")
      array=[]
      i=0
      j=0
      h=0
      #if comb=2 means that 2 nodes are selected from controller graph
      #count[i][j][k] the occurence of the node k when the i and j node a
re selected
      #from the controller graph (i:0-19)(j:0-19)(k:0-199)
      pairDistance=[0 for i in range(int(comb(int(comb(20,2)),2)))]
```

```
      pairSimilarity=[0 for i in range(int(comb(int(comb(20,2)),2)))]
      count=np.zeros((20, 20, 200))
      in_labels=[]
      combs2_labels=[]
      for x in f:
         array.append(x)
         #take the number X of the string in(X)
         index_in=array[h].find("in=")
         #in txt file which is the output of clingo file
         #if 'in=200,201' means that the 0 and 1 node are selected from c
ontroller graph
         #so im trying to extract these numbers
         #im saving the label 200,201 to list in_labels
         if(index_in==0):
            text=array[h][3:]
            tokens=text.split(',')
            if(i==0):
               in_labels.append(text[:len(text)-1])
         index=array[h].find("in(")
         indexlast=array[h].find(")")
         if(array[h].find("in(")==0 ):
            indEnd=0
            indBeg=0
            while(indBeg!=-1):
                  indEnd=array[h].find(")",indBeg)
                  k=array[h][indBeg+3:indEnd]
                  #increase the occurence of this node
                  count[int(tokens[0])-200][int(tokens[1])-
200][int(k)]+=1
                  indBeg=array[h].find("in(",indEnd)
         #if -------------
 found means that all the combinations of the ith graph found
         #so i find the similarity and the euclidean distance of all the
combinations
         #e.g. the similarity of 200,201 and 200,202
         #and after i zero the array count and i count again the occurenc
es of each node of the i+1th graph
         if(array[h].find("-------------")==0):
            #i produce all the possible combination of these labels
            #e.g if i have the labels 200,201 and 200,202 i produse the l
abel 200,201-200,202
            #i do this only one time because the labels dont change
            if(i==0):
               for pair1 in range(len(in_labels)):
                  for pair2 in range(pair1+1,len(in_labels)):
                     combs2_labels.append("%s-
%s"%(in_labels[pair1],in_labels[pair2]))
```

```
            pairDistance=calculateEuclideanDistance(count,pairDistance,co
mbs2_labels)
            pairSimilarity=calculateCosineSimilarity(count,pairSimilarity
,combs2_labels)
            count=np.zeros((20, 20, 200))
            i+=1
         h+=1
      f.close
      #calculate the averages of cosine similarity and pair distance
      pairSimilarity,pairDistance,similarity_avg,distance_avg=calculateAv
erages(pairSimilarity,pairDistance)
      print("End of processing, density:",density)
      writeInFile("DistanceDensity"+str(density)+".txt",pairDistance,dist
ance_avg,combs2_labels,density)
      writeInFile("SimilarityDensity"+str(density)+".txt",pairSimilarity,
similarity_avg,combs2_labels,density)
getResults()
```