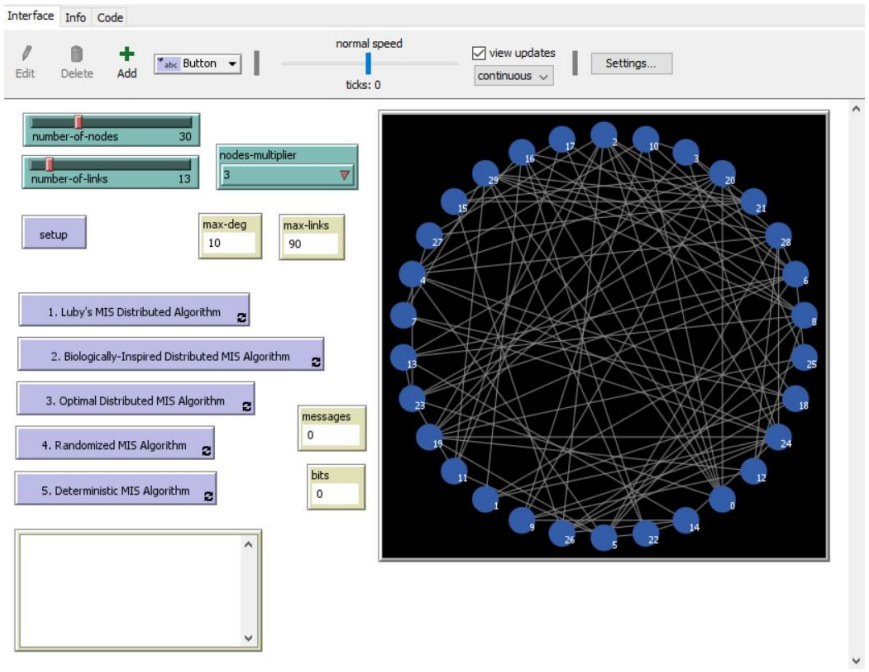
Υλοποίηση MIS Αλγορίθμων σε NetLogo

Αρχικά, εντός της διαδικασίας «**to setup**», δημιουργούμε ένα τυχαίο γράφημα, του οποίου ο αριθμός των κόμβων δίνεται από τον χρήστη μέσω του αντίστοιχου slider "number-of-nodes". Όσον αφορά τον αριθμό των ακμών, ο χρήστης θα πρέπει να μεταβεί στον chooser "nodes-multiplier", στο Interface, και να επιλέξει μία από τις 4 επιλογές (0, 3, 7 και 15). Αν επιλέξει 0, ο αριθμός των ακμών του γραφήματος αντιστοιχεί στον αριθμό που επιλέγεται μέσω του slider "number-of-links", ενώ για οποιαδήποτε από τις υπόλοιπες τρεις επιλογές, ο αριθμός των ακμών ορίζεται ως ο αριθμός των κόμβων επί τον πολλαπλασιαστικό παράγοντα 3, 7, ή 15 αντίστοιχα. Επιπλέον, όπως και στην Άσκηση 1, αν ο αριθμός των ακμών υπερβεί τη τιμή $n \cdot (n-1)/2$, τότε ορίζουμε τη τιμή που προκύπτει απ' αυτό το τύπο ως max-links. Τέλος, εντός του setup υπολογίζουμε και τον μέγιστο βαθμό του γραφήματος (max-deg).

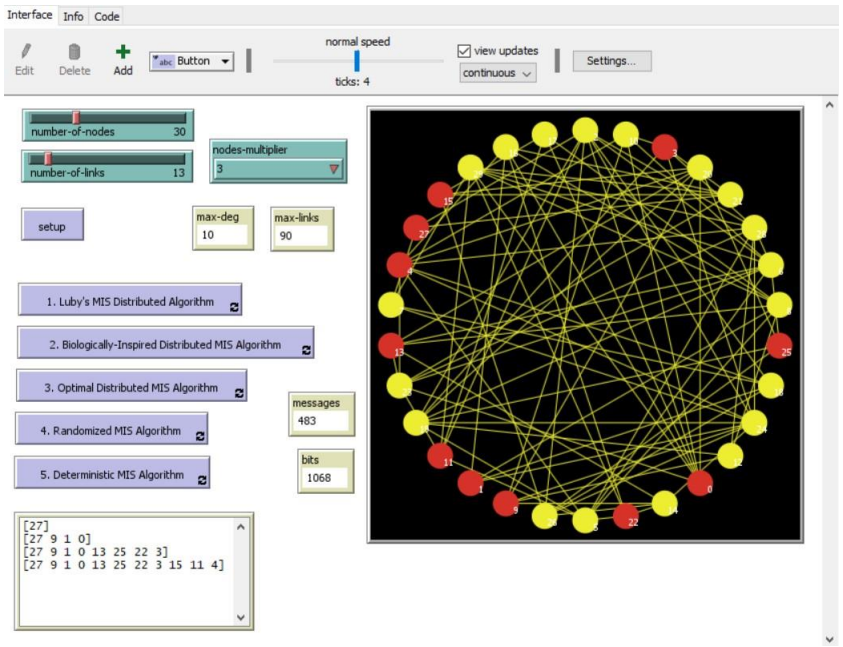


Εικόνα 2.1: Το Interface και η οθόνη προσομοίωσης.

Στο Interface, υπάρχουν αριθμημένα 5 buttons, εκτός του setup, που αντιστοιχούν σε καθένα από τους 5 αλγορίθμους που υλοποιήσαμε, για δημιουργία ενός μέγιστου ανεξάρτητου συνόλου (MIS). Επίσης, υπάρχει και ένα output section, όπου εκεί τυπώνεται, σε κάθε βήμα του κάθε αλγορίθμου, ποιοι κόμβοι βρίσκονται στο MIS, κάτι το οποίο είναι και εμφανές στην οθόνη προσομοίωσης λόγω του κόκκινου χρώματος που έχουν οι κόμβοι στο MIS. Παρακάτω, παρουσιάζουμε για καθέναν από τους 5 αλγορίθμους διευκρινιστικές πληροφορίες, καθώς και τις ζητούμενες γραφικές παραστάσεις.

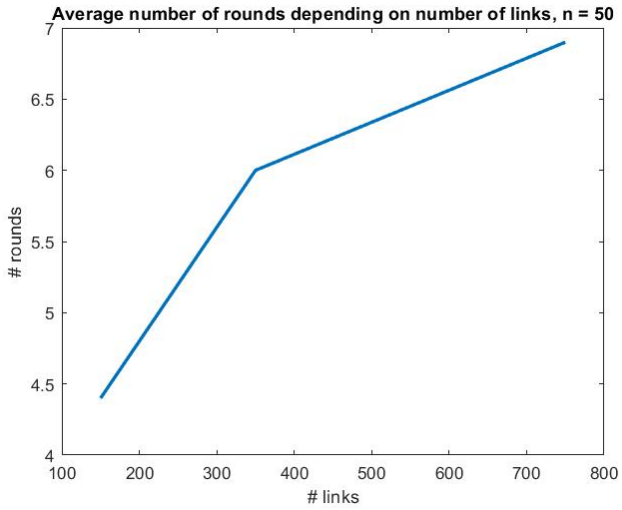
1. **Luby's MIS Distributed Algorithm**

   Υλοποιήσαμε σε κώδικα NetLogo τον αλγόριθμο, όπως περιγράφεται από τον ψευδοκώδικα της σελίδας 17 των διαφανειών. Κάθε κόμβος διαθέτει μια μεταβλητή curr-degree που έχει τον τρέχοντα βαθμό του και δύο λίστες neighbor-degs και neighbor-ids που έχει τους βαθμούς και τα IDs αντίστοιχα που έλαβε από τους γείτονές του. Επίσης, χρησιμοποιούμε δύο boolean μεταβλητές stopped και elected, όπου η πρώτη δείχνει αν ο κόμβος είναι σταματημένος και η δεύτερη αν ο κόμβος είναι υποψήφιος για το MIS. Αναλυτικότερη περιγραφή της διαδικασίας παρέχεται μέσω σχολίων στο αρχείο του κώδικα.

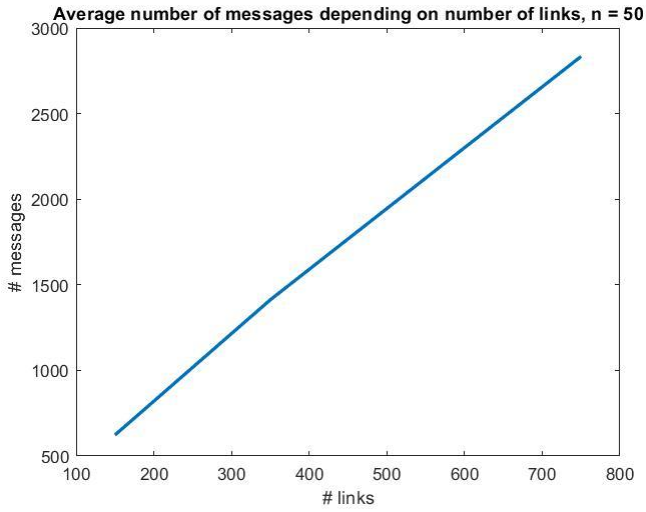


Εικόνα 2.2: Στιγμιότυπο εκτέλεσης του "Luby's MIS Distributed Algorithm".

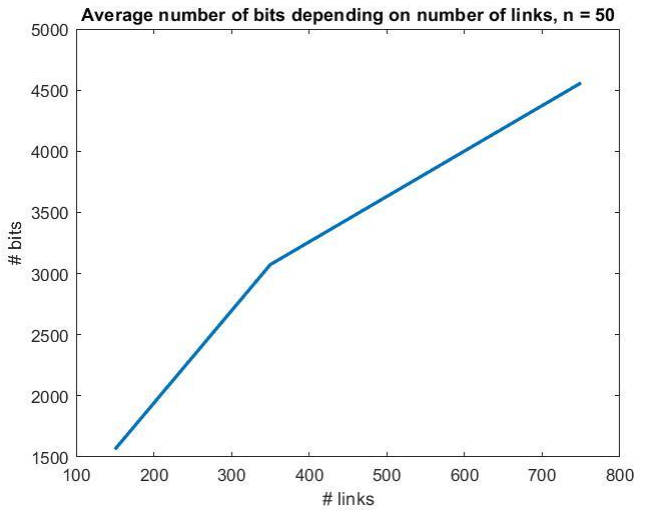
α) Ως πλήθος γύρων του αλγορίθμου, θεωρούμε τον αριθμό των ticks που χρειάζονται μέχρι να τερματίσει ο αλγόριθμος.



β) Θεωρούμε πως κάθε κόμβος στέλνει μηνύματα στους γείτονές του: 1) για να τους γνωστοποιήσει αν είναι elected ή όχι, και 2) όταν εισέρχεται στο MIS, για να τους ειδοποιήσει να μεταβούν σε σταματημένη κατάσταση.

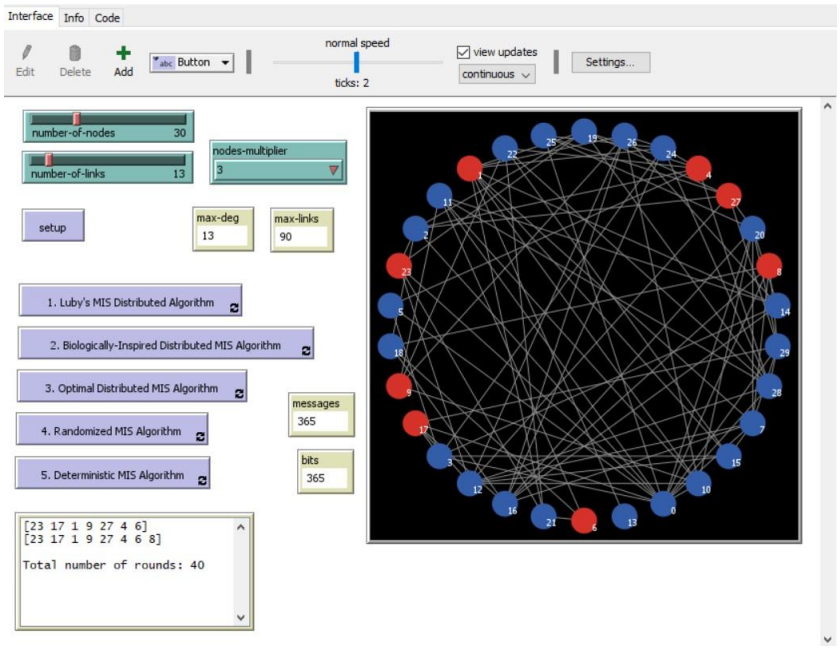


γ) Στις παραπάνω περιπτώσεις που ανταλλάζονται μηνύματα: 1) οι κόμβοι που είναι elected, στα μηνύματα που στέλνουν στους γείτονές τους, επισυνάπτουν τον τρέχοντα βαθμό τους και το μοναδικό αναγνωριστικό τους ID (ο τρόπος που υπολογίζεται το πλήθος των bits είναι εμφανές στον κώδικα) και 2) τα μηνύματα είναι του ενός bit.



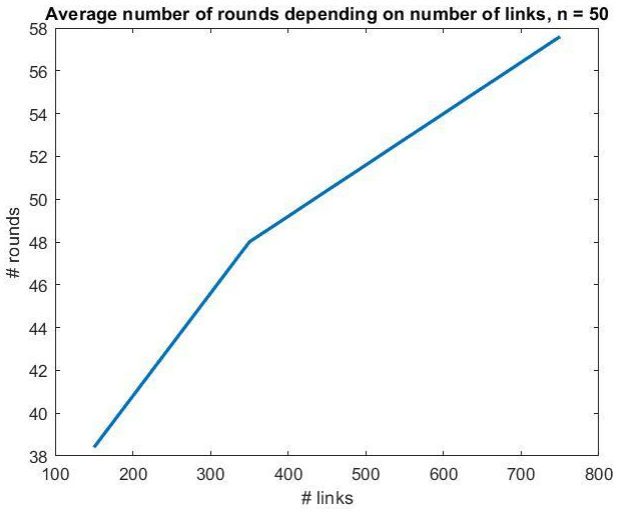
2. Biologically-Inspired Distributed MIS Algorithm

Υλοποιήσαμε σε κώδικα NetLogo τον αλγόριθμο, όπως περιγράφεται από τον ψευδοκώδικα της σελίδας 28 των διαφανειών. Όπως και προηγουμένως, χρησιμοποιούνται οι boolean μεταβλητές, stopped και elected, για κάθε κόμβο. Ο αλγόριθμος τερματίζει όταν όλοι οι unelected κόμβοι έχουν τουλάχιστον έναν γειτονικό κόμβο που να είναι elected, να ανήκει δηλαδή στο MIS. Τότε, ο αλγόριθμος τερματίζει και εκτυπώνει στο output section τον συνολικό χρόνο εκτέλεσης. Αναλυτικότερη περιγραφή της διαδικασίας παρέχεται μέσω σχολίων στο αρχείο του κώδικα. Στο διπλανό στιγμιότυπο (Εικόνα 2.3), οι κόμβοι που βρίσκονται εντός του MIS είναι με κόκκινο χρώμα ενώ όλοι οι υπόλοιποι είναι μπλε (default χρώμα).

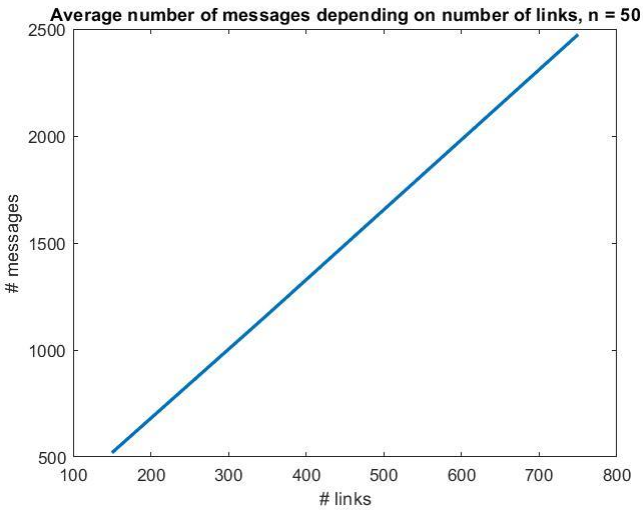


Εικόνα 2.1: Στιγμιότυπο εκτέλεσης του "Biologically-Inspired Distributed MIS Algorithm".

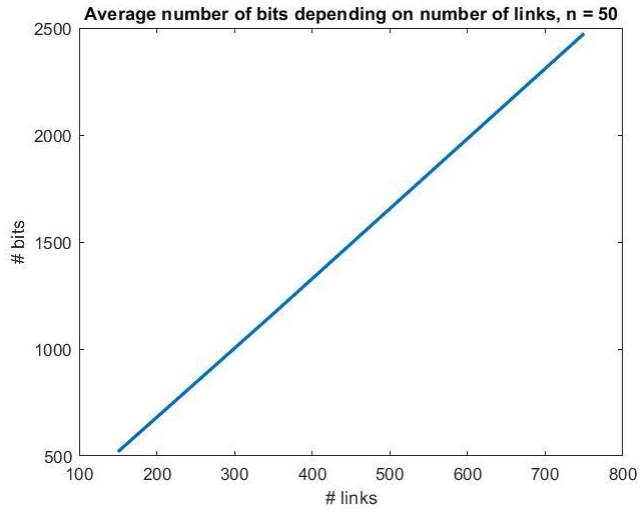
α) Το συνολικό πλήθος γύρων του αλγορίθμου προκύπτει από το γινόμενο των χρόνων των δύο while loops, επί τον αριθμό των ticks που χρειάστηκε να τρέξει ο αλγόριθμος μέχρι να φτάσει στην τερματική κατάσταση. Επομένως, ο συνολικός χρόνος, σ' αυτή την περίπτωση, εκτυπώνεται στο output section όταν τερματίζει ο αλγόριθμος.



β) Θεωρούμε πως κάθε κόμβος στέλνει μηνύματα στους γείτονές του: 1) όταν είναι elected και κάνει broadcast το B, και 2) όταν εισέρχεται στο MIS, όπου και πάλι κάνει broadcast το B.

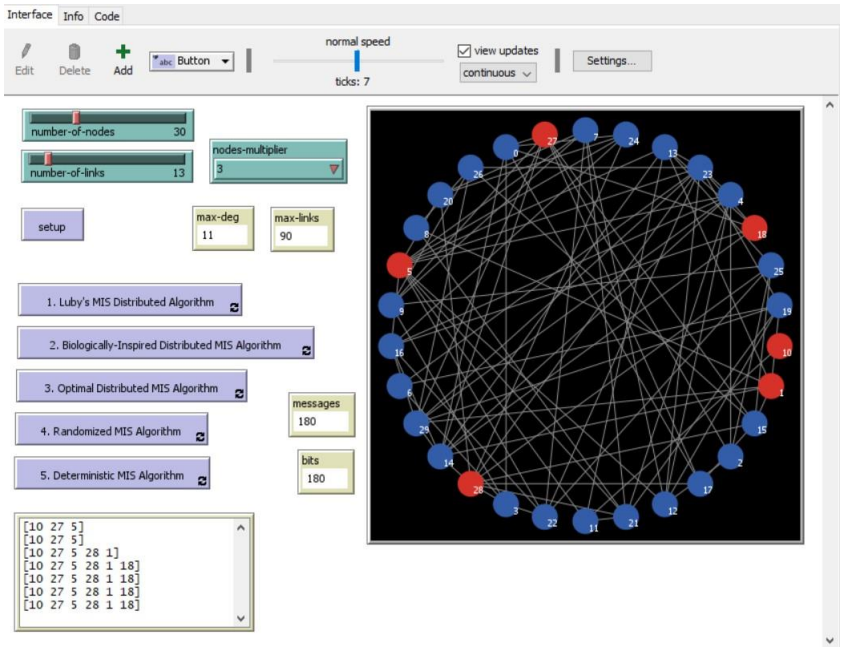


γ) Οι κόμβοι ανταλλάζουν single bit μηνύματα, οπότε το συνολικό πλήθος bits είναι ίδιο με το συνολικό πλήθος των μηνυμάτων. Άρα, οι γραφικές παραστάσεις είναι ίδιες.



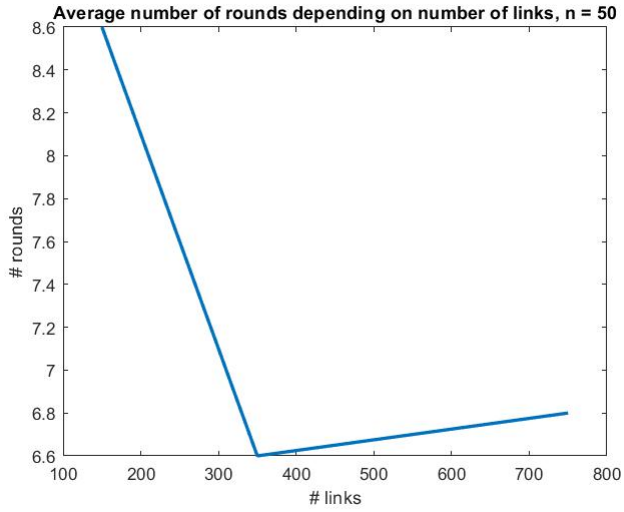
3. Optimal Distributed MIS Algorithm

Υλοποιήσαμε σε κώδικα NetLogo τον αλγόριθμο, όπως περιγράφεται από τον ψευδοκώδικα της σελίδας 31 των διαφανειών. Όπως και προηγουμένως, χρησιμοποιούνται οι boolean μεταβλητές, stopped και elected, για κάθε κόμβο. Αναλυτικότερη περιγραφή της διαδικασίας παρέχεται μέσω σχολίων στο αρχείο του κώδικα. Στο διπλανό στιγμιότυπο (Εικόνα 2.4), οι κόμβοι που βρίσκονται εντός του MIS είναι με κόκκινο χρώμα ενώ όλοι οι υπόλοιποι είναι μπλε (default χρώμα).

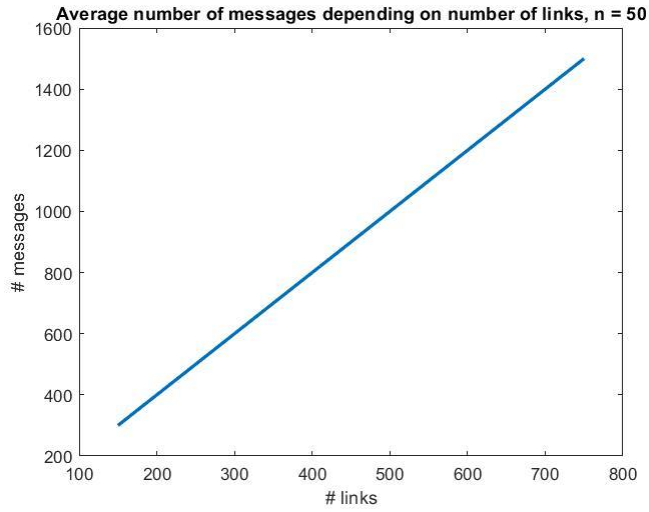


Εικόνα 2.2: Στιγμιότυπο εκτέλεσης του "Optimal Distributed MIS Algorithm".

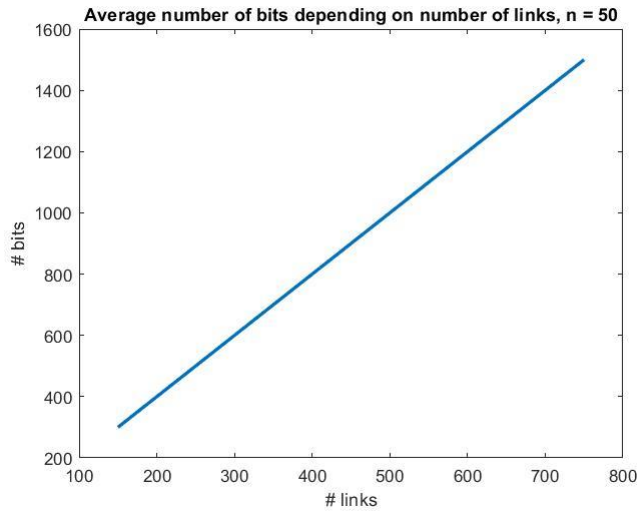
α) Ως πλήθος γύρων του αλγορίθμου, θεωρούμε τον αριθμό των ticks που χρειάζονται μέχρι να τερματίσει ο αλγόριθμος.



β) Σύμφωνα με τον ψευδοκώδικα, θεωρούμε πως κάθε κόμβος στέλνει μηνύματα μόνο όταν είναι elected και κάνει signal σε όλους τους γειτονικούς του κόμβους.

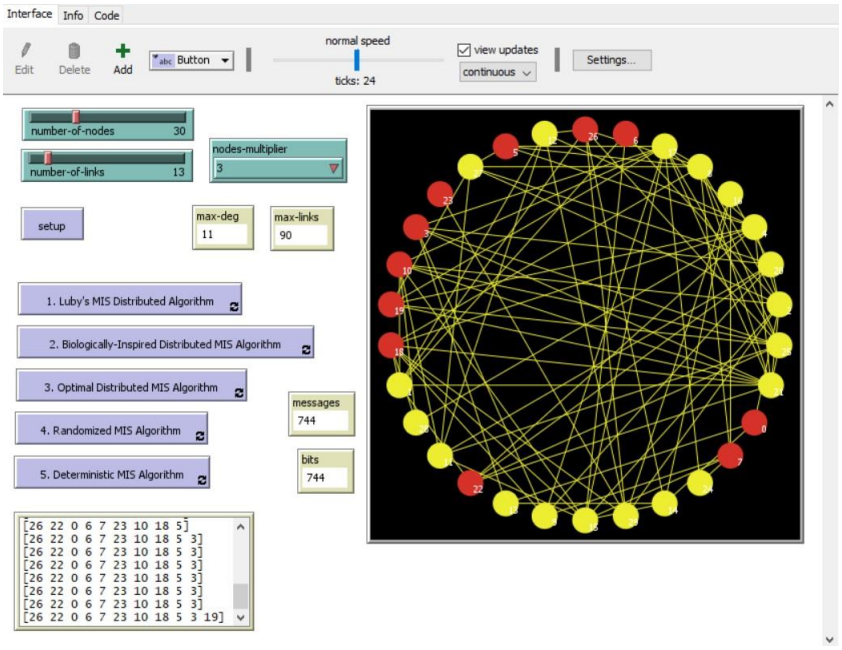


γ) Όπως και στον προηγούμενο αλγόριθμο, τα μηνύματα που ανταλλάζουν οι κόμβοι είναι single bit, επομένως το συνολικό πλήθος bits ισούται με το συνολικό πλήθος μηνυμάτων. Άρα, οι γραφικές παραστάσεις είναι ίδιες.



4. Randomized MIS Algorithm

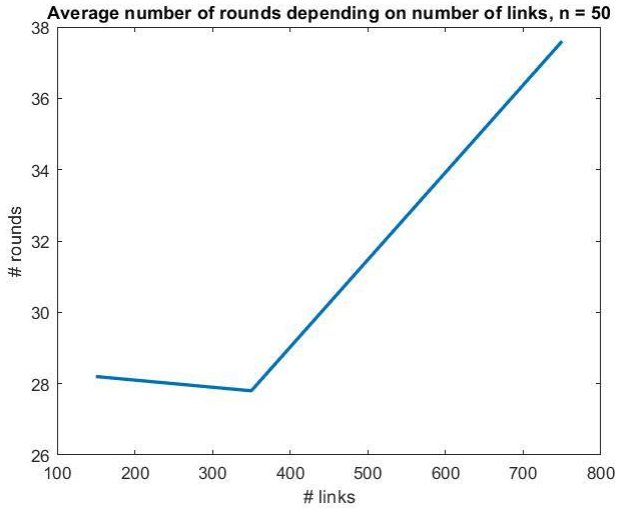
Υλοποιήσαμε σε κώδικα NetLogo τον αλγόριθμο, όπως περιγράφεται από τον ψευδοκώδικα της σελίδας 13 των διαφανειών. Όπως και προηγουμένως, χρησιμοποιούνται οι boolean μεταβλητές, stopped και elected, για κάθε κόμβο. Αναλυτικότερη περιγραφή της διαδικασίας παρέχεται μέσω σχολίων στο αρχείο του κώδικα. Στο διπλανό στιγμιότυπο (Εικόνα 2.5), οι κόμβοι που βρίσκονται εντός του MIS είναι με κόκκινο χρώμα ενώ όλοι οι υπόλοιποι είναι κίτρινοι. Μετά την εισαγωγή ενός κόμβου στο MIS, γίνεται ανανέωση του γραφήματος «σβήνοντας» τους γειτονικούς



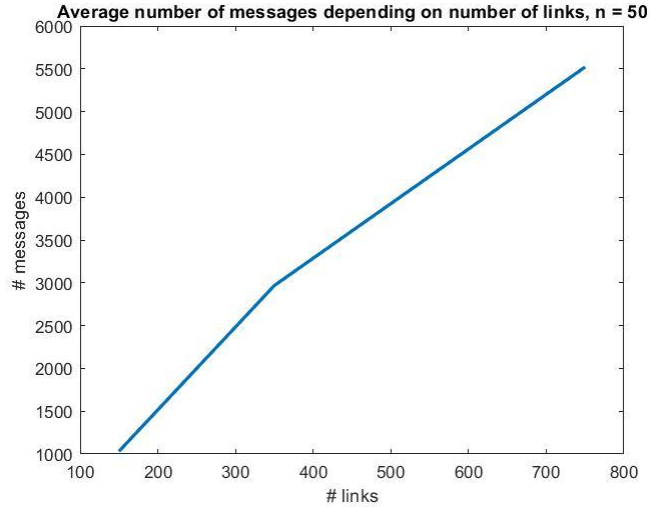
Εικόνα 2.3: Στιγμιότυπο εκτέλεσης του "Randomized MIS Algorithm".

κόμβους του καθώς και τις ακμές που συνδέονται μ' αυτούς τους κόμβους. Στην προσομοίωση, αυτό γίνεται αλλάζοντας το χρώμα των σβησμένων κόμβων και ακμών σε κίτρινο.

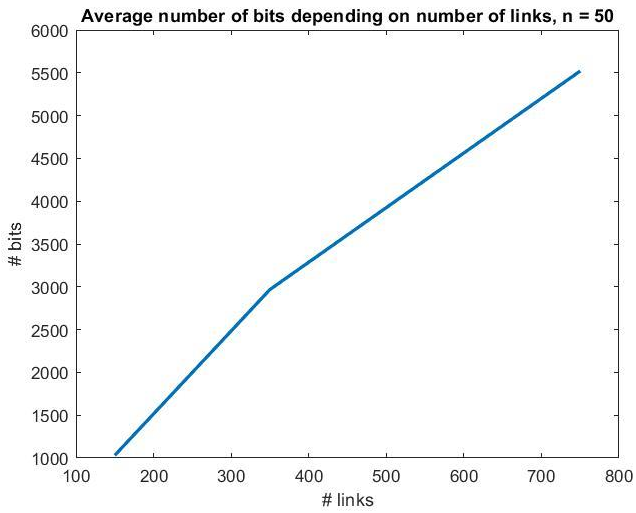
α) Ως πλήθος γύρων του αλγορίθμου, θεωρούμε τον αριθμό των ticks που χρειάζονται μέχρι να τερματίσει ο αλγόριθμος.



β) Θεωρούμε πως κάθε κόμβος στέλνει μηνύματα στους γείτονές του: 1) για να τους γνωστοποιήσει αν είναι elected ή όχι, και 2) όταν εισέρχεται στο MIS, για να τους ειδοποιήσει να μεταβούν σε σταματημένη κατάσταση.

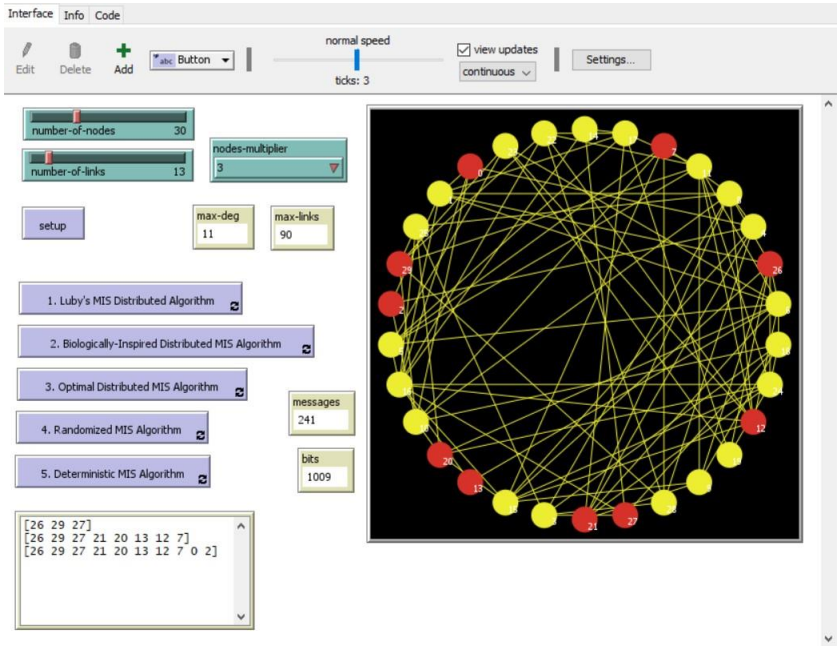


γ) Τα παραπάνω μηνύματα που ανταλλάζουν οι κόμβοι είναι όλα του ενός bit, επομένως το συνολικό πλήθος bits του αλγορίθμου ισούται με το συνολικό πλήθος των μηνυμάτων. Άρα, οι γραφικές παραστάσεις είναι ίδιες.



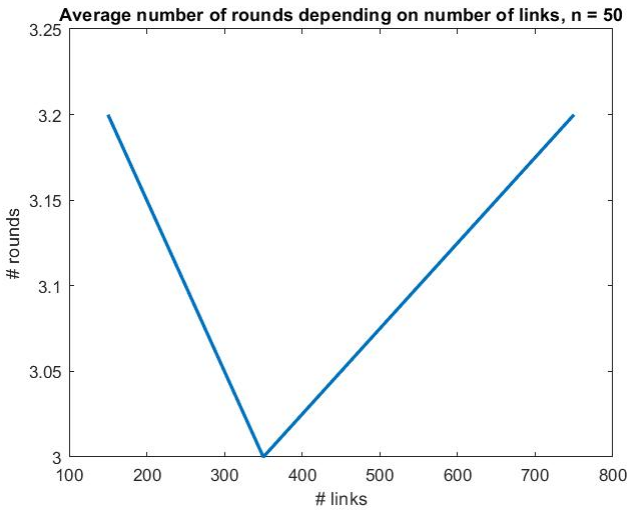
5. Deterministic MIS Algorithm

Υλοποιήσαμε σε κώδικα NetLogo τον αλγόριθμο, όπως περιγράφεται στη σελίδα 8 των διαφανειών. Όπως και προηγουμένως, χρησιμοποιούνται οι boolean μεταβλητές, stopped και elected, για κάθε κόμβο. Αναλυτικότερη περιγραφή της διαδικασίας παρέχεται μέσω σχολίων στο αρχείο του κώδικα. Στο διπλανό στιγμιότυπο (Εικόνα 2.6), οι κόμβοι που βρίσκονται εντός του MIS είναι με κόκκινο χρώμα ενώ όλοι οι υπόλοιποι είναι κίτρινοι.

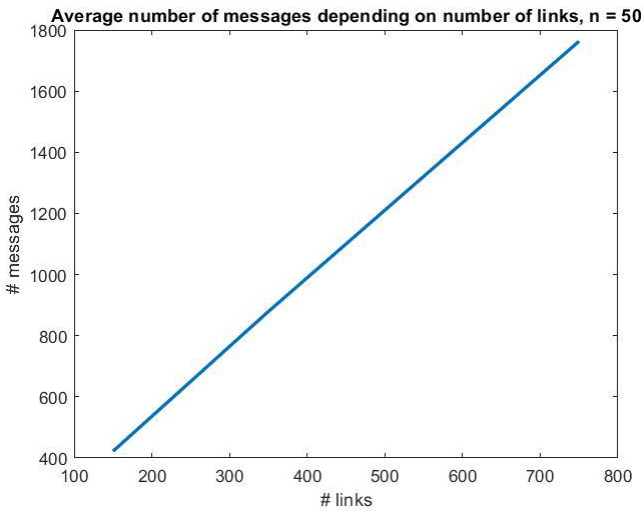


Εικόνα 2.4: Στιγμιότυπο εκτέλεσης του "Deterministic MIS Algorithm".

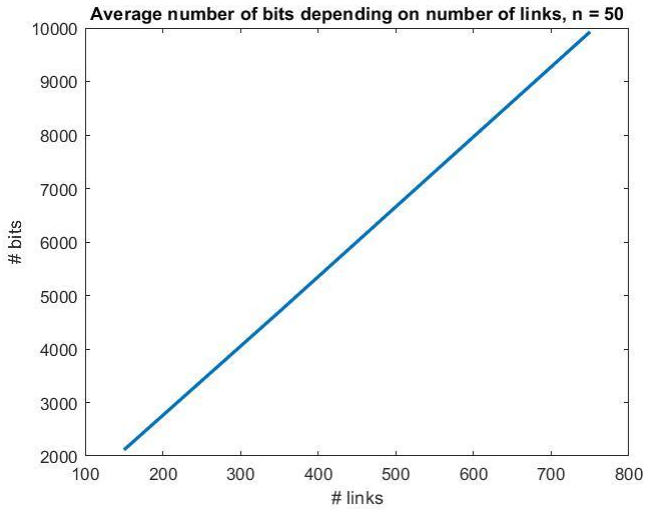
α) Ως πλήθος γύρων του αλγορίθμου, θεωρούμε τον αριθμό των ticks που χρειάζονται μέχρι να τερματίσει ο αλγόριθμος.



β) Θεωρούμε πως κάθε κόμβος στέλνει μηνύματα στους γείτονές του: 1) όταν στέλνει το ID του στους γειτονικούς του κόμβους, και 2) όταν εισέρχεται στο MIS για να ειδοποιήσει τους γείτονές του να μεταβούν σε σταματημένη κατάσταση.



γ) Στις παραπάνω περιπτώσεις που ανταλλάζονται μηνύματα: 1) το μήνυμα που περιέχει το ID του κόμβου είναι μεγέθους $\log_2$(#κόμβων) bits (ο τρόπος που υπολογίζεται το πλήθος των bits είναι εμφανές στον κώδικα) και 2) τα μηνύματα είναι του ενός bit.



Πειραματικά συμπεράσματα:

- Παρατηρούμε πως οι **Deterministic**, **Optimal** και **Luby's MIS Algorithms** είναι αρκετά γρήγοροι και για τα τρία μεγέθη γραφημάτων, με τον **Deterministic** να προηγείται οριακά. Ωστόσο, οι **Deterministic** και **Luby's** έχουν αρκετά μεγάλη πολυπλοκότητα σε πλήθος μηνυμάτων και bits, σε αντίθεση με τον **Optimal** που χρησιμοποιεί μόνο single bit μηνύματα. Επομένως, ο πιο βέλτιστος συνολικά θεωρούμε πως είναι ο **Optimal Distributed MIS Algorithm**.
- Πιο αργός συνολικά, από τους 5 αλγορίθμους, είναι ο **Biologically-Inspired MIS Algorithm**.
- Οι αλγόριθμοι που στα μηνύματά τους ανταλλάζουν πληροφορίες, όπως τον τρέχοντα βαθμό τους ή το μοναδικό αναγνωριστικό τους (ID) έχουν τη μεγαλύτερη πολυπλοκότητα σε bits μηνυμάτων (**Luby's Algorithm**, **Deterministic MIS Algorithm**).
- Ο **Randomized Algorithm**, επειδή είναι τυχαιοκρατικός, μπορεί να παράξει εκτελέσεις με πολύ μικρό αριθμό γύρων, ή και αντίθετα εκτελέσεις με μεγάλο αριθμό γύρων, οπότε δεν χαρακτηρίζεται τόσο για την αξιοπιστία του.