ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΣΧΟΛΗ ΜΗΧΑΝΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ

ΤΟΜΕΑΣ ΜΗΧΑΝΟΛΟΓΙΚΩΝ ΚΑΤΑΣΚΕΥΩΝ & ΑΥΤΟΜΑΤΟΥ ΕΛΕΓΧΟΥ

ΟΙ ΣΥΝΑΡΤΗΣΕΙΣ ΠΛΟΗΓΗΣΗΣ ΣΤΗΝ ΕΝΑΕΡΙΑ ΚΥΚΛΟΦΟΡΙΑ: ΕΦΑΡΜΟΓΗ ΣΤΟΝ ΕΝΑΕΡΙΟ ΧΩΡΟ ΤΗΣ ΚΕΝΤΡΙΚΗΣ ΕΥΡΩΠΗΣ

NAVIGATION FUNCTIONS IN AIR TRAFFIC MANAGEMENT: A CASE STUDY ON CENTRAL EUROPE

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

ΚΩΝΣΤΑΝΤΙΝΟΣ ΒΑΣΙΛΑΚΗΣ

Επιβλέπων: ΚΩΝΣΤΑΝΤΙΝΟΣ ΚΥΡΙΑΚΟΠΟΥΛΟΣ

Καθηγητής Ε.Μ.Π

Αθήνα, Οκτώβριος 2011

ΕΥΧΑΡΙΣΤΙΕΣ

Στο σημείο αυτό θέλω να εκφράσω τις ευχαριστίες μου, στον κ.Κων/νο Κυριακόπουλο και κ. Γιάννη Ρούσσο για την συμπαράσταση και καθοδήγηση τους κατά την εκπόνηση αυτής της διπλωματικής εργασίας. Θα ήθελα να ευχαριστήσω επίσης όλα τα μέλη του εργαστηρίου αυτομάτου ελέγχου του ΕΜΠ για την συνεργασία και τις πολύτιμες συμβουλές τους όπου αυτό ήταν απαραίτητο.

Περιεχόμενα

Εισαγωγή

Η εργασία αυτή έχει σαν σκοπό την εφαρμογή της μεθόδου των συναρτήσεων πλοήγησης για την μελέτη ενός δείγματος πτήσεων τα οποία πετούν στον χώρο της Κεντρικής Ευρώπης σε μία χρονική διάρκεια δύο ημερών. Η πρώτη ημέρα αποτελεί ένα τυπικό σημερινό σενάριο πτήσεων και η δεύτερη τροποποίηση αυτού του σεναρίου έτσι ώστε να προσεγγίζει ένα πιθανό μελλοντικό σενάριο αυξημένης κίνησης. Ο βασικός σκοπός είναι η εφαρμογή της παραπάνω μεθόδου έτσι ώστε να μελετήσουμε πως θα συμπεριφερθούν τα αεροσκάφη πετώντας τελείως αυτόνομα χωρίς ανθρώπινη βοήθεια και η δημιουργία ενός κώδικα σε Matlab που μπορεί να χειριστεί αυτό αλλά και παρόμοια σενάρια μεγάλου αριθμού αεροσκαφών με μικρές αλλαγές. Επίσης στο δεύτερο μέρος αυτής της εργασίας παρουσιάζουμε μία εφαρμογή σε Java η οποία μπορεί να χρησιμοποιηθεί για την λύση, με την βοήθεια της μεθόδου των συναρτήσεων πλοήγησης, απλών δισδιάστατων σεναρίων η οποία μπορεί να χρησιμοποιηθεί για εκπαιδευτικού λόγους ή για μία εισαγωγική επίδειξη της λειτουργίας της μεθόδου.

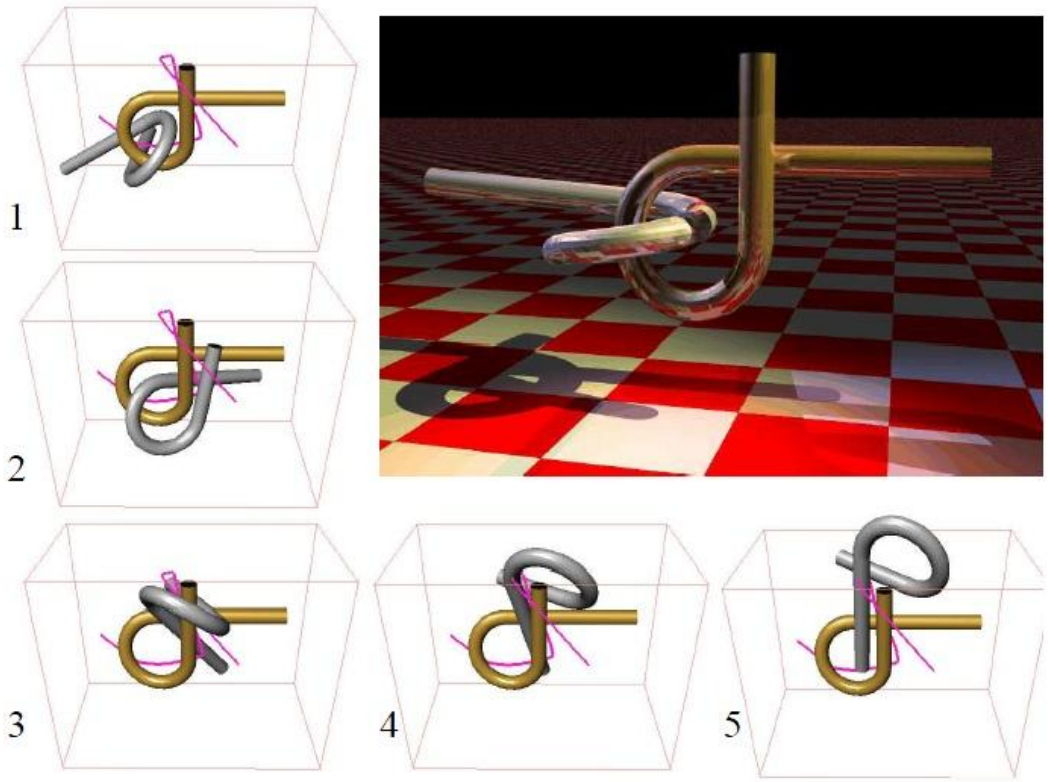
ΚΕΦΑΛΑΙΟ 1

1.1Προγραμματισμός Πορείας

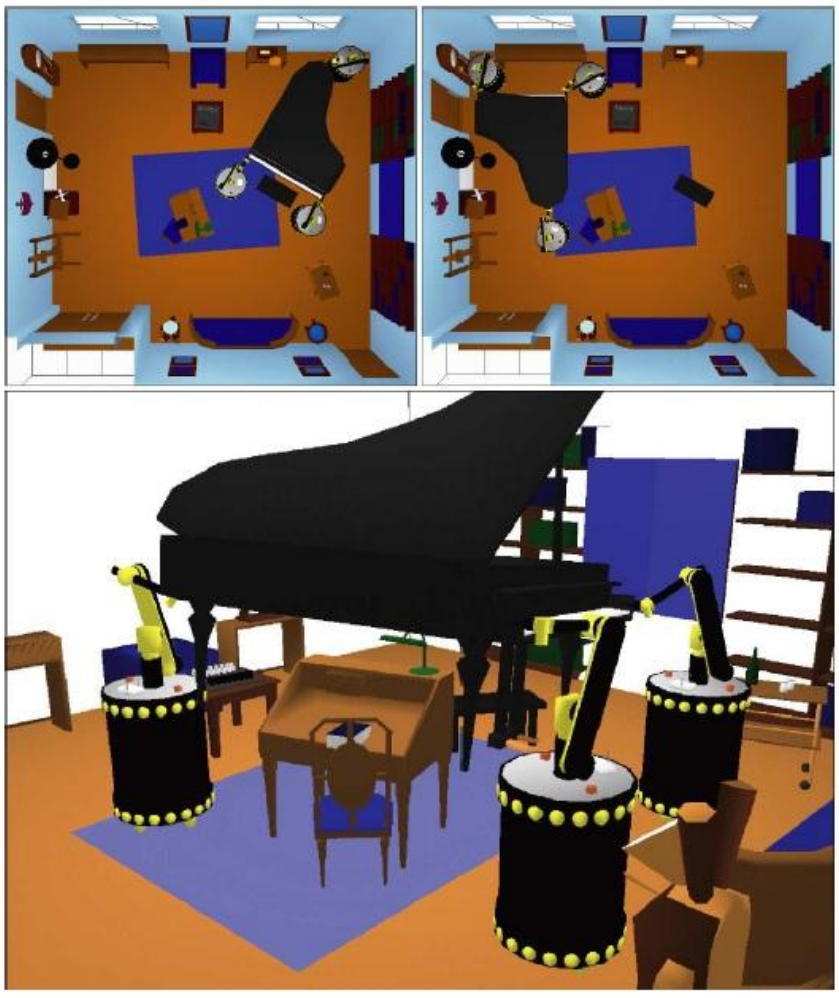
Στη Ρομποτική βασικό ζητούμενο είναι η αυτοματοποίηση ηλεκτρομηχανικών συστημάτων ,για την πραγματοποίηση μίας σειράς επιθυμητών ενεργειών, τα οποία έχουν την δυνατότητα να λαμβάνουν πληροφορίες από το περιβάλλον τους(με τη βοήθεια αισθητήρων) , να αλληλεπιδρούν με αυτό(με τη βοήθεια των επενεργητών) και τα οποία διαθέτουν διάφορες υπολογιστικές δυνατότητες. Ζωτικής σημασίας είναι η ύπαρξη αλγορίθμων και μεθόδων έτσι ώστε υψηλού επιπέδου εντολές που αφορούν το "τι" θέλουμε να γίνει να μπορούν να δίνονται χωρίς να μας αφορά το " πως". Ένα κλασικό πρόβλημα προγραμματισμού πορείας είναι το " Piano Mover's Problem" το οποίο ουσιαστικά είναι το πρόβλημα του πως μπορούμε να μετακινήσουμε ένα πιάνο(έχοντας στο μυαλό μας κάποιους περιορισμούς όπως το αν μπορούμε ή όχι να σηκώσουμε το αντικείμενο από το πάτωμα) σε ένα σπίτι. Βεβαίως παρόμοια προβλήματα όλοι έχουμε συναντήσει κάποιες στιγμές στη ζωή μας και παρόλο που ο ανθρώπινος νους είναι " προγραμματισμένος" να μπορεί εύκολα να λύνει τέτοια προβλήματα η διατύπωση αυτών με συστηματικό τρόπο έτσι ώστε να μπορεί να ενσωματωθεί σε ένα πχ αυτόνομο όχημα παρουσιάζει πολλές φορές δυσκολίες και αδιέξοδα. Συνήθως με τον όρο *προγραμματισμός κινήσεων(motion planning)* αναφερόμαστε στο πρόβλημα της εύρεσης ουσιαστικά ενός "δρόμου" ο οποίος ενώνει δύο σημεία και ενδιάμεσα παρακάμπτει διάφορα εμπόδια που πιθανώς υπάρχουν. Με τον όρο *προγραμματισμός τροχιών (trajectory planning)* συνήθως αναφερόμαστε στην διαδικασία κατά την οποία παίρνουμε την λύση που μας δόθηκε από το πρόβλημα του προγραμματισμού κινήσεων και προσδιορίζουμε πως θα κινηθεί το συγκεκριμένο σύστημα που μας αφορά(ρομπότ) με τρόπο που να επιτρέπει η δυναμική και οι περιορισμοί του.

Βεβαίως προβλήματα τα οποία άρχισαν σαν προβλήματα προγραμματισμού πορείας ενός ρομπότ ή αντικειμένου μέσα σε ένα χώρο με εμπόδια εξελίχθηκαν σε προβλήματα που περιλαμβάνουν βελτιστοποιημένες πορείες ως προς κάποια παράμετρο, πλοήγηση μέσα σε άγνωστο περιβάλλον, πλοήγηση μέσα σε περιβάλλοντα τα οποία εισάγουν δυσεπίλυτες διαταραχές κατά τον έλεγχο καθώς και σε αβέβαια περιβάλλοντα όπου οι αισθητήρες δεν μπορούν να παρατηρήσουν με επιθυμητή ακρίβεια τον χώρο. Επίσης πλέον έχουν υπεισέρθει αρκετά ζητήματα που επηρεάζουν και επηρεάζονται από όλους τους παραπάνω παράγοντες αλλά κοιτούν το πρόβλημα από μία υψηλότερου επιπέδου σκοπιά(decision making, planning).

Παρόλα αυτά το βασικό πρόβλημα του προγραμματισμού πορείας παραμένει και ενώ έχει σημειωθεί πρόοδος στον τομέα αυτόν υπάρχουν αρκετά ζητήματα που μπορούν να επιλυθούν αφού η δυνατότητα εφαρμογής μία εκ των διαφόρων μεθόδων που έχουν προταθεί εξαρτάται από το σύστημα που υπάρχει και την πολυπλοκότητα του.



Alpha 1.0 Puzzle



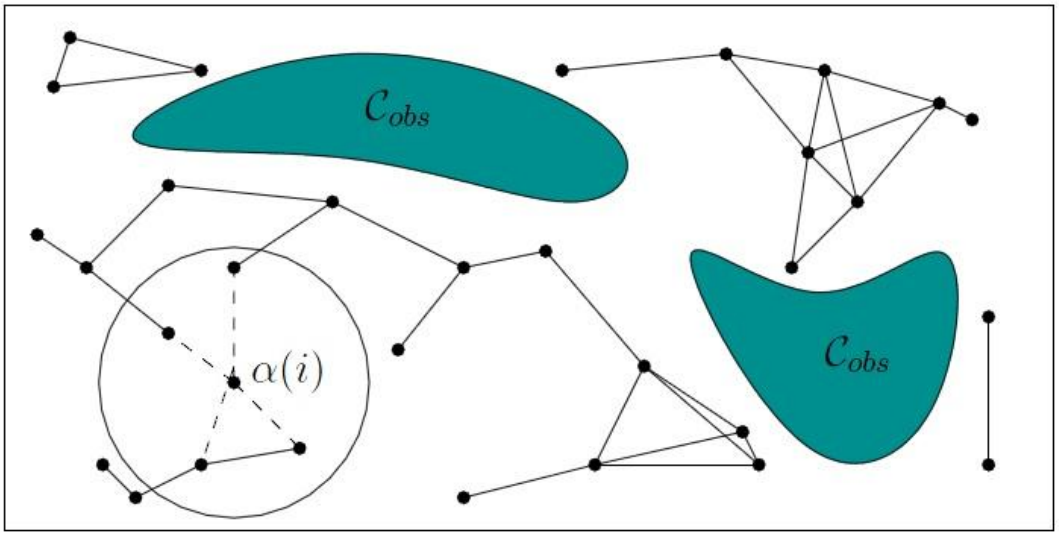
Piano mover's problem

1.2 Βασικές Προσεγγίσεις

Στα παραπάνω προβλήματα υπάρχει μία πληθώρα μαθηματικών και αλγοριθμικών μεθόδων που έχουν προταθεί οι περισσότερες από τις οποίες αφορούν την επίλυση ειδικών περιπτώσεων συγκεκριμένων προβλημάτων. Οι περισσότερες από αυτές είναι ουσιαστικά το αποτέλεσμα λίγων διαφορετικών διατυπώσεων του βασικού προβλήματος. Όπως ίσως και σε άλλα σύνθετα προβλήματα(όπως πχ αυτό της βελτιστοποίησης γενικά) δεν υπάρχει κάποια μέθοδος που να έχει καθολική αποδοχή αλλά μάλλον διαφορετικές διατυπώσεις του ίδιου προβλήματος οι οποίες λειτουργούν καλύτερα σε συγκεκριμένες περιπτώσεις και οι ερευνητές έχουν χτίσει πάνω σε αυτές. Αν και το πεδίο είναι αρκετά μεγάλο κάποιες βασικές προσεγγίσεις είναι οι παρακάτω.

1.2.1Roadmap

Αυτή η προσέγγιση είναι ουσιαστικά η κατασκευή διάφορων "δρόμων" μεταξύ κάποιων εμποδίων (γραφήματα βασισμένα στην τοπολογία του χώρου). Η ιδέα εδώ είναι ότι μπορούμε να αφιερώσουμε αρχικά ένα χρονικό διάστημα εύρεσης αυτών των δρόμων και μετά οποιαδήποτε στιγμή αυτοί χρειαστούν να είναι εύκολα προσπελάσιμοι από το ρομπότ. Για ένα δεδομένο αρχικό και τελικό σημείο πρέπει εύκολα να μπορεί να γίνει η ένωση αρχικού σημείου – roadmap – τελικού σημείου. Ουσιαστικά είναι μία μέθοδος γραφικής περιγραφής της τοπολογίας ενός χώρου. Διάφορες μέθοδοι έχουν προταθεί βασισμένες σε αυτή την προσέγγιση όπως τα γραφήματα ορατότητας, διαγράμματα Voronoi, δίκτυο freeway.



1.2.2 Cell Decomposition

Αυτές οι μέθοδοι "αποσυνθέτουν" τον ελεύθερο χώρο του ρομπότ σε απλούστερες περιοχές που αποκαλούνται κύτταρα έτσι ώστε η πλοήγηση του ρομπότ σε αυτές να μπορεί να γίνει εύκολα. Σε αυτά δημιουργείται το γράφημα συνδετικότητας το οποίο συνδέει γειτονικά κύτταρα μεταξύ τους. Με αυτόν τον τρόπο μπορεί να κατασκευαστεί μία πορεία χρησιμοποιώντας μία ακολουθία κυττάρων (κανάλι).

Οι μέθοδοι αυτές μπορούν να διαχωριστούν σε ακριβείς κατά τις οποίες ο ελεύθερος χώρος αποσυντίθεται σε κύτταρα τα όρια των οποίων συσχετίζονται με τους περιορισμούς της κίνησης του ρομπότ και προσεγγιστικές στις οποίες ο χώρος μπορεί να χωριστεί σε κύτταρα προκαθορισμένης μορφής των οποίων τα όρια δεν έχουν κάποια φυσική σημασία.

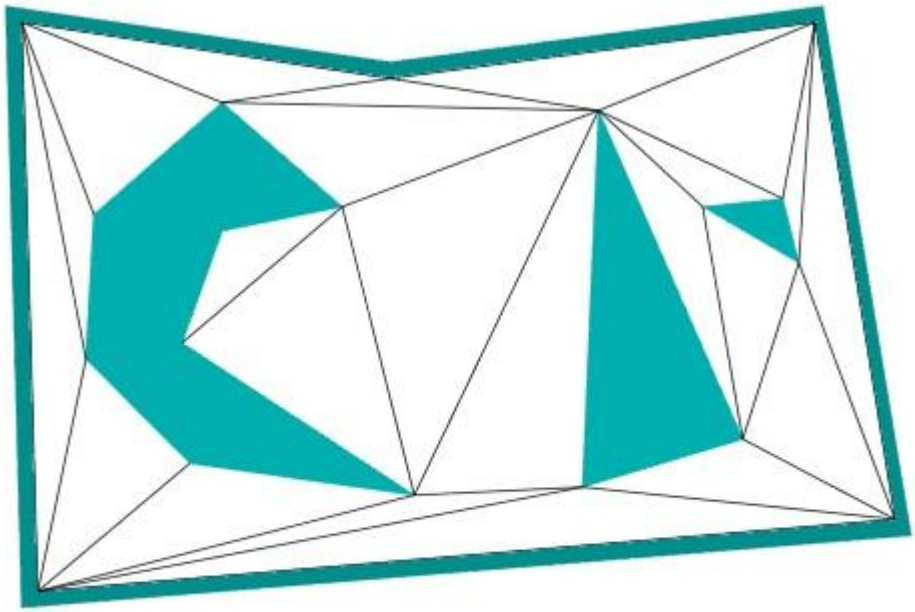


Figure 6.16: A triangulation of C_{free}.

1.2.3 Δυναμικά Πεδία (Potential Fields)

Η βασική ιδέα σε αυτή την προσέγγιση είναι η μοντελοποίηση του ρομπότ ως ένα σημείο το οποίο βρίσκεται υπό την επίδραση του ελκτικού δυναμικού του σημείου προορισμού και του απωθητικού δυναμικού των εμποδίων. Συνήθως σε αυτές της μεθόδους ορίζουμε σαφώς τα όρια του χώρου εργασία του ρομπότ και των εμποδίων και χρησιμοποιούμε μαθηματικές εκφράσεις για τη μοντελοποίηση του δυναμικού του χώρου. Η κίνηση του ρομπότ γίνεται ακολουθώντας την αρνητική κλίση του δυναμικό στο σημείο του χώρου που βρισκόμαστε. Η προσέγγιση αυτή είναι ιδιαίτερα αποτελεσματική. Το βασικότερο μειονέκτημα βρίσκεται στο γεγονός ότι τελικά η μέθοδος βασίζεται στην ελαχιστοποίηση μία συνάρτησης η οποία είναι

αρκετά πιθανόν και λόγω της μορφής της συνάρτησης που περιγράφει το δυναμικό και λόγω της τοπολογικής πολυπλοκότητας του χώρου εργασίας του ρομπότ να εγκλωβιστεί σε τοπικά ελάχιστα. Για τον λόγο αυτόν επιδιώκεται να δημιουργούνται συναρτήσεις που δημιουργούν δυναμικά πεδία με ένα μόνο ολικό ελάχιστο στο στόχο ή δημιουργούνται διάφορες μέθοδοι αποφυγής των τοπικών ελαχίστων. Οι συναρτήσεις πλοήγησης (navigation functions) οι οποίες θα αναλυθούν εκτενέστερα παρακάτω ανήκουν σε αυτή την προσέγγιση.

1.3 Συναρτήσεις Πλοήγησης – Navigation Functions

1.3.1 Εισαγωγή

Οι συναρτήσεις πλοήγησης εισήχθησαν από τους Elon Rimon και Daniel E. Koditschek και ενοποιούν το πρόβλημα προγραμματισμού κίνησης, προγραμματισμού τροχιών(υλοποίηση ενός ελεγκτή που λειτουργεί σύμφωνα με την δυναμική του ρομπότ). Η συναρτήσεις αυτές παρέχουν άμεση λύση του παραπάνω προβλήματος η οποία εγγυάται σύγκλιση στον προορισμό από κάθε,(σχεδόν), αρχικό σημείο του χώρου εργασίας του ρομπότ. Στη βασική τους μορφή οι συναρτήσεις πλοήγησης προϋποθέτουν πλήρη γνώση του χώρου εργασίας του ρομπότ, το οποίο υποθέτουμε σημειακό και του οποίου την θέση γνωρίζουμε κάθε στιγμή. Τα όρια του χώρου εργασίας και τα εμπόδια τα οποία περιέχονται στον χώρο αυτό μοντελοποιούνται σαν δίσκοι ή σφαίρες(για το δισδιάστατο και το τρισδιάστατο πρόβλημα αντίστοιχα). Τα εμπόδια υποτίθενται ακίνητα και οι θέσεις τον κέντρων τους γνωστές. Το κέντρο του χώρου εργασίας τοποθετείται στην αρχή των αξόνων. Σε αυτήν την κατασκευή οι συναρτήσεις πλοήγησης μπορούν να εφαρμοστούν και αποτελούν χάρτες δυναμικού το οποίο λαμβάνει μέγιστη τιμή στα σύνορα των εμποδίων και του χώρου εργασίας και ελάχιστη στον προορισμό. Η σύγκλιση του ρομπότ σε αυτό το σημείο επιτυγχάνεται ακολουθώντας την αρνητική κλίση των συναρτήσεων αυτών στο σημείο που βρίσκεται το ρομπότ. Εδώ οφείλουμε να συμπληρώσουμε πως όπως αναφέραμε και παραπάνω οι μέθοδοι path planning οι οποίες βασίζονται στην ιδέα της δημιουργίας δυναμικού πεδίου έχουν το μειονέκτημα της δημιουργίας τοπικών ελαχίστων και επειδή ουσιαστική η σύγκλιση γίνεται ελαχιστοποιώντας τις συναρτήσεις αυτές το ρομπότ μπορεί να εγκλωβιστεί σε κάποιο τοπικό ελάχιστο(το ολικό ελάχιστο βέβαια βρίσκεται πάντα στο στόχο), κα να σταματήσει εκεί. Οι συναρτήσεις πλοήγησης μπορούν να εξαλείψουν τα τοπικά ελάχιστα που υπάρχουν με κατάλληλη επιλογή κάποιας παραμέτρου των συναρτήσεων αυτών(η επιλογή αυτή βέβαια παρουσιάζει κάποιες πρακτικές δυσκολίες όμως υπάρχει ερευνητικό ενδιαφέρον σε αυτήν την κατεύθυνση με αξιοσημείωτα αποτελέσματα). Παρόλα αυτά λόγω της φύσης της κατασκευής των

συναρτήσεων αυτών πάντα θα υπάρχει ένας αριθμός saddle points στα οποία το ρομπότ μπορεί να εγκλωβιστεί. Από την μαθηματική σκοπιά αυτό είναι δυνατόν αλλά πρακτικά είναι εξαιρετικά απίθανο να συμβεί λόγω των διαταραχών που υπάρχουν στα πραγματικά συστήματα (λόγω αισθητήρων, θορύβου, υπολογιστικής ακρίβειας κτλ). Το βασικό πλεονέκτημα των συναρτήσεων πλοήγησης είναι ότι παρόλο που ορίζονται σε σφαιρικούς κόσμους(sphere worlds) παραμένουν αμετάβλητες στους διφεομορφισμούς το οποίο έχει σαν αποτέλεσμα(θεωρητικά τουλάχιστον αφού και η εύρεση διφεομορφισμών είναι αρκετές φορές ένα ιδιαίτερα πολύπλοκο πρόβλημα) να μπορούν να εφαρμοστούν σε οποιοδήποτε χώρο διφεομορφικό σε ένα sphere world. Παρακάτω ακολουθεί μία αναλυτικότερη περιγραφή των συναρτήσεων αυτών.

1.3.2 Βασικές Έννοιες

Σαν σφαιρικό κόσμο(sphere world) ορίζουμε ένα συμπαγές και συνεκτικό υποσύνολο (compact connected subject) του E^n.

$$W \triangleq \{q \in E^n : \| q \|^2 \leq \rho_o^2\}$$

$$O_\iota \triangleq \{q \in E^n : \| q - q_j \|^2 < \rho_o^2\}, j = 1,2, \dots M$$

Όπου W είναι ο χώρος εργασίας (Workspace) και O_ι τα εμπόδια.

Ισχύει ότι

$$F \triangleq W - \bigcup_{\{j=1\}}^{M} O_j, j = 1,2, \dots M$$

Που αποτελεί το Free Space και

$$\rho_o > 0 \; και \; \| q_i \| + \rho_i < \rho_o, 1 \leq i \leq M$$

$$\| q_i - q_j \| > \rho_i + \rho_j, 1 \leq i, j \leq M$$

Δηλαδή τα εμπόδια περιέχονται μέσα στο Workspace και επίσης δεν τέμνονται ούτε ακουμπούν μεταξύ τους.

Μπορούν να αποδειχθούν οι παρακάτω προτάσεις

Πόρισμα 1.3.2.α Δεν υπάρχει λείο (μη εκφυλισμένο) non degenerate διανυσματικό πεδίο ,f , στο F με M>0 εμπόδια , το οποίο είναι κάθετο στο ∂F, έτσι ώστε η ροή του $\dot{x} = -f$ να οδηγεί σε μία καθολική ασυμπτωτικά ευσταθή κατάσταση ισορροπίας.

Πόρισμα 1.3.2.β Έστω f είναι ένα λείο μη εκφυλισμένο διανυσματικό πεδίο στο F , με M>0 εμπόδια, το οποίο είναι κάθετο στο ∂F. Έστω ότι $-f$ τείνει προς ένα μοναδικό σημείο ισορροπίας. Τότε κάθε εμπόδιο δημιουργεί τουλάχιστον ένα σημείο σέλας(saddle point) στο f.

Αυτός είναι και ο λόγος που γράψαμε παραπάνω ότι το ρομπότ συγκλίνει από σχεδόν κάθε αρχικό σημείο στον προορισμό του. Αν υποθέσουμε ότι ξεκινάμε από κάποιο σημείο σέλας τότε ακριβώς πάνω στο σημείο αυτό η κλίση των συναρτήσεων που θα περιγράψουμε παρακάτω. Αυτό όμως συμβαίνει ακριβώς πάνω σε αυτό το σημείο. Πρακτικά και μία πολύ μικρή διαταραχή σε κάποιο πραγματικό πρόβλημα ή και κάποια υπολογιστική διαταραχή θα μας "κουνήσει" από αυτό το σημείο και το ρομπότ θα φύγει.(Παρόμοια περίπτωση είναι η περίπτωση που μία ράβδος με μία μάζα σε μία άρθρωση ισορροπεί στο πάνω μέρος της)

1.3.3 Ορισμοί - Απόδειξη Ύπαρξης

Ορισμός: Έστω $F \subset E^n$ είναι μία συμπαγής, συνεκτική αναλυτική πολλαπλότητα με κάποιο σύνορο. Μία απεικόνιση $\phi: F \to [0,1]$ είναι μία συνάρτηση πλοήγησης (navigation function) αν είναι:

1. Αναλυτική στο F
2. Polar στο F
3. Morse στο F
4. Admissible στο F

Εδώ διευκρινίζουμε ότι αναλυτική στο F είναι η συνάρτηση αν και μόνο αν ισούται με την σειρά Taylor της σε μία γειτονιά κάθε σημείου του F. Πρακτικά αυτή η ιδιότητα εξασφαλίζει ότι η συνάρτηση είναι λεία ή C^∞. Polar σημαίνει ότι η συνάρτηση έχει ένα μοναδικό ολικό ελάχιστο στο q_d που είναι το σημείο προορισμού στο εσωτερικό του F. Αν $\phi \in C^2[E^n, E]$ και συμβολίσουμε με β_ϕ το σύνολο των κρίσιμων σημείων της και στα κρίσιμα αυτά σημεία της φ η Εσιανή της συνάρτησης $D^2\phi \triangleq D\nabla\varphi$ έχει γραμμικά ανεξάρτητες σειρές (ή στήλες αντίστοιχα) η συνάρτηση καλείται Morse. Τέλος είναι admissible στο F αν $\varphi(\partial F) = 1$ και σε κάθε άλλο σημείο του F $\quad 0 \leq \phi < 1$.

Σημείωση: Όπως αναφέρεται και από τους συγγραφεί από μαθηματική σκοπιά οι συναρτήσεις αρκεί να είναι C^2 παρόλα αυτά οι αναλυτικές συναρτήσεις οδηγούν σε εκφράσεις που είναι αποδεδειγμένα συγκλίνουσες στον στόχο που μας ενδιαφέρει. Στην πραγματικότητα βέβαια για να μπορέσουμε να εφαρμόσουμε την μέθοδο σε πιο πρακτικά προβλήματα εκτός από σφαιρικούς κόσμους χρειάζεται μία ανοχή στον παραπάνω ορισμό.

Ορισμός: Ένα κρίσιμο σημείο μία συνάρτησης ϕ είναι μη εκφυλισμένο αν η εσιανή του $D^2\phi$ περιέχει γραμμικά ανεξάρτητα διανύσματα στο σημείο αυτό. Η βαθμωτή αυτή συνάρτηση καλείται Morse αν όλα τα κρίσιμα σημεία της είναι μη εκφυλισμένα.

Ορισμός: Δείκτης Morse (Morse Index) της ϕ σε κάποιο κρίσιμο σημείο της ,έστω x, είναι η διάσταση του υποχώρου το E^n που δημιουργείται από τα ιδιοδιανύσματα της εσιανής που αντιστοιχούν σε αρνητικές ιδιοτιμές

$$\lambda(x) \triangleq \dim\{\, y \in E^n : y^T [D^2\phi](x)\, y < 0 \,\}$$

Θεώρημα 1.3.3.1: (Smale,1961 [7,Θεώρημα C]). Έστω M συμπαγής ν-διάστατη C^∞ πολλαπλότητα(manifold) με το ∂M ίση με την ένωση(disjoint union) των V_1 και V_2 όπου το κάθε V_i κλειστό στο ∂M. Τότε υπάρχει λεία συνάρτηση φ στο M με μη-εκφυλισμένα κρίσιμα σημεία ,ομαλή στο ∂M, με $\phi(V_1) = -\frac{1}{2}, \phi(V_2) = n + \frac{1}{2}$ και σε κάποιο κρίσιμο σημείο p της ϕ , $\phi(p) = index\ p$

Ο Smale χαρακτήρισε αυτές τις συναρτήσεις “nice functions”.

Θεώρημα 1.3.3.2: ([8] Θεώρημα 8.1) Έστω M συμπαγής ν-διάστατη C^∞ πολλαπλότητα με το ∂M ίση με την ένωση(disjoint union) των V_1 και V_2 δύο συμπαγείς πολλαπλότητες χωρίς σύνορο. Έστω φ που ικανοποιεί το θεώρημα 1.3.3.1 με $\phi^{-1}\left(-\frac{1}{2}\right) = V_1$. Αν $H_o(M, V_1) = 0$, τότε τα κρίσιμα σημεία με δείκτη 0 μπορούν να ακυρωθούν από έναν ίσο αριθμό κρίσιμων σημείων με δείκτη 1.

Πρόταση 1.3.3.3: [9]Εστω η φ είναι μία συνάρτηση που ικανοποιεί το θεώρημα 1.3.3.1 (nice function) στην M πολλαπλότητα του προηγούμενου θεωρήματος. Αν η πολλαπλότητα είναι συνεκτική τότε υπάρχουν κρίσιμα σημεία δείκτη 1 τουλάχιστον όσα κρίσιμα σημεία δείκτη 0 της φ στο M.

Η παραπάνω πρόταση μας εξασφαλίζει ότι υπάρχουν αρκετά κρίσιμα σημεία δείκτη 1 για να ακυρώσουν όλα τα ελάχιστα όταν η πολλαπλότητα είναι συνεκτική.

Το παρακάτω θεώρημα δίνεται μαζί με την απόδειξη του λόγω της σημαντικότητας του.

Θεώρημα 1.3.3.4 (Προτάθηκε από τον M.Hirsch) Για κάθε λεία συμπαγής και συνεκτική πολλαπλότητα με σύνορο , M, για κάθε $x_o \in int(M)$ υπάρχει μία συνάρτηση πλοήγησης

Απόδειξη: Έστω N_o ένας ανοιχτός δίσκος γύρω από το x_o στο εσωτερικό του M. Έτσι το $V_o \triangleq \partial N_o$ είναι ένα σύνορο του $M' \triangleq M - N_o$. Επίσης ορίζουμε $V_1 \triangleq \partial M$,οπότε το σύνορο του M' είναι η ένωση των V_0και V_1. Έστω φ συνάρτηση "nice" που η ύπαρξη της ικανοποιείται από το θεώρημα 1.3.3.1. Τα M' και V_o είναι συνεκτικά οπότε $H_o(M', V_0) = 0$. Από το θεώρημα 1.3.3.2 η φ μπορεί να αντικατασταθεί από μία νέα συνάρτηση, ϕ' που είναι ίση με την φ στο V_0 αλλά δεν έχει κρίσιμα σημεία με δείκτη 0(ελάχιστα). Τελικά επεκτείνουμε την ϕ' στο M, ορίζοντας συνάρτηση κόστους ,π, σε μία ανοιχτή γειτονιά του $\overline{N_o}$ που συμφωνεί με την ϕ' στο σύνορο, V_o, και έχει ένα μοναδικό κρίσιμο σημείο, ένα ελάχιστο στο x_o. Αυτό μπορεί να γίνει αφού το N_oείναι διφεομορφικό στο D^n. □

1.3.4 Οι ιδιότητες παραμένουν αναλλοίωτες στους διφεομορφισμούς

Μία από τις σημαντικότερες ιδιότητες των συναρτήσεων πλοήγησης είναι ότι οι ιδιότητες τους παραμένουν αμετάβλητες στους διφεομορφισμούς. Αυτό είναι μία πολύ σημαντική ιδιότητα αφού μας επιτρέπει να εφαρμόσουμε τις συναρτήσεις αυτές σε οποιοδήποτε χώρο διφεομορφικό σε ένα σφαιρικό κόσμο.

Πρόταση 1.3.4.1: Έστω $\phi: M \to [0,1]$ μία συνάρτηση πλοήγησης στο M, και $h: F \to M$ ένας διφεομορφισμός. Τότε

$$\tilde{\phi} \triangleq \phi \circ h$$

είναι μία συνάρτηση πλοήγησης στο F.

Πρόταση 1.3.4.2: Έστω $J_1, J_2 \subseteq \Re$ διαστήματα, $\hat{\phi}: F \to J_1$ και $\sigma: J_1 \to J_2$ αναλυτικές συναρτήσεις. Ορίζουμε την σύνθεση

$$\phi \triangleq \sigma \circ \hat{\phi}$$

Αν η σ είναι γνησίως μονότονη στο J_1, τότε το σύνολο των κρίσιμων σημείων της $\hat{\phi}$ και ϕ συμπίπτουν

$$\beta_\phi = \beta_{\hat{\phi}}$$

και οι δείκτες του κάθε σημείου είναι ίδιοι.

Ουσιαστικά η σύνθεση με την συνάρτηση σ της συνάρτησης $\hat{\phi}$ δεν αλλάζει το σύνολο των κρίσιμων σημείων της αλλά ούτε και τον τύπο τους (ελάχιστο, μέγιστο, σημείο σέλας) ή το αν είναι ή όχι εκφυλισμένα.

Έτσι λοιπόν για οποιοδήποτε χώρο και σχήματα εμποδίων που περιέχονται σε αυτόν αν μπορούμε να βρούμε ένα διφεομορφισμό του χώρου αυτού σε ένα σφαιρικό κόσμο τότε μπορούμε να εφαρμόσουμε τις συναρτήσεις πλοήγησης όπως αυτές ορίζονται για σφαιρικούς κόσμους.

1.3.5 Κατασκευή

Η συνάρτηση που προτάθηκε είναι η $\phi: F \to [0,1]$ και είναι σύνθεση τριών συναρτήσεων

$$\phi \triangleq \sigma_d \circ \sigma \circ \hat{\phi}$$

Η $\hat{\phi}$ είναι πολική , σχεδόν παντού Morse ,αναλυτική και μεγιστοποιείται στο ∂F

Το σ είναι ένας διφεομορφισμός από το $[0, \infty)$ στο $[0,1]$ και

$$\sigma(x) \triangleq \frac{x}{1+x}$$

και η σύνθεση τους είναι μία πολική,admissible και αναλυτική συνάρτησηη οποία είναι μη εκφυλισμένη στο F εκτός από ένα σημείο αυτό του προορισμού.

Ορίζουμε την $\hat{\phi}$ ως

$$\hat{\phi} \triangleq \frac{\gamma}{\beta}$$

όπου $\gamma: F \to [0, \infty)$ με

$$\gamma \triangleq \gamma_d^k \ , k \in \aleph, \qquad \gamma_d \triangleq \|q - q_d\|^2$$

και $\beta: F \to [0. \infty)$ με

$$\beta \triangleq \prod_{j=0}^{M} \beta_j$$

$$\beta_o \triangleq \rho_o^2 - \|q\|^2, \qquad \beta_j \triangleq \left\|q - q_j\right\|^2 - \rho_j^2, \quad j = 1 \dots M$$

Επίσης ορίζουμε

$$\overline{\beta_i} \triangleq \prod_{j=0, j \neq i}^{M} \beta_j$$

Λόγω της παραμέτρου k στον ορισμό του γ τα σημείο του προορισμού είναι εκφυλισμένο κρίσιμο σημείο.. Σαν αντιστάθμιση στην επίδραση αυτή του k ορίζουμε την

$$\sigma_d(x) \triangleq (x)^{\frac{1}{k}}, \quad k \in \aleph, \qquad \sigma_d: [0,1] \to [0,1]$$

Οπότε τελικά προκύπτει

$$\phi = \sigma_d \circ \sigma \circ \hat{\phi} = \left(\frac{\gamma_d^k}{\gamma_d^k + \beta}\right)^{\frac{1}{k}}$$

ή ισοδύναμα

$$\phi = \frac{\gamma_d}{\left(\gamma_d^k + \beta\right)^{\frac{1}{k}}}$$

Θεώρημα 1.3.5.1: Αν ο F είναι ένας σφαιρικός κόσμος (όπως ορίστηκε προηγούμενα), τότε υπάρχει θετικός ακέραιος N τέτοιος ώστε για κάθε $k \geq N$,για

κάποιο πεπερασμένο αριθμό εμποδίων, και για κάθε σημείο προορισμού στο εσωτερικό του F, η

$$\phi = \sigma_d \circ \sigma \circ \hat{\phi} = \left(\frac{\gamma_d^k}{\gamma_d^k + \beta}\right)^{\frac{1}{k}}$$

Είναι συνάρτηση πλοήγησης στο F.

Το παραπάνω θεώρημα παρουσιάζεται ως η βασική συνεισφορά της [1]. Πρακτικά το παραπάνω θεώρημα μας δίνει την δυνατότητα να εφαρμόσουμε την παραπάνω κατασκευή των συναρτήσεων πλοήγησης για κάθε διφεομορφικό χώρο σε έναν σφαιρικό κόσμο και εξασφαλίζει την ύπαρξη σταθεράς k τέτοιας ώστε δεν υπάρχουν τοπικά ελάχιστα στον χώρο λειτουργίας του ρομπότ.

Σημείωση: Τα παραπάνω μας δίνουν ένα σημαντικό εργαλείο στα προβλήματα προγραμματισμού πορείας. Παρόλα αυτά οφείλουμε να αναφέρουμε ότι από μόνα τους τα προβλήματα της εύρεσης κατάλληλου διφεομορφισμού κάποιου χώρου ή η εύρεση κατάλληλου k είναι από μόνα τους πολλές φορές ιδιαίτερα πολύπλοκα προβλήματα και πολλές φορές αντί για λύσεις από καθαρά μαθηματική σκοπιά δίνονται λύσεις από την σκοπιά του engineering. Όμως ερευνητική δραστηριότητα υπάρχει η οποία είναι προσανατολισμένη στα παραπάνω προβλήματα και σημαντικές βελτιώσεις έχουν προταθεί αλλά και προτείνονται ακόμα.

Η απόδειξη του θεωρήματος 1.3.5.1 δίνεται αναλυτικά στην [1]. Παρακάτω δίνουμε συνοπτικά ένα σχεδιάγραμμα της απόδειξης αυτής.

Σχεδιάγραμμα Απόδειξης

Έστω $\epsilon > 0$. Ορίζουμε $B_i(\epsilon) \triangleq \{q \in E^n : 0 < \beta_i < \epsilon\}$ (δηλαδή μία μικρή περιοχή γύρω από τα εμπόδια του χώρου λειτουργίας).

Στην απόδειξη χωρίζουμε το F σε πέντε υποσύνολα

1. Το σημείο προορισμού

$$\{q_d\}$$

2. Το σύνορο του F

$$\partial F = \beta^{-1}(0)$$

3. Τις περιοχές κοντά στα εμπόδια

$$F_o(\epsilon) \triangleq \bigcup_{i=1}^{M} B_i(\epsilon) - \{q_d\}$$

4. Την περιοχή κοντά στο όριο του χώρου λειτουργίας

$$F_1(\epsilon) \triangleq B_o(\epsilon) - (\{q_d\} \cup F_o(\epsilon))$$

5. Την περιοχή μακριά από τα εμπόδια

$$F_2 \triangleq F - (\{q_d\} \cup \partial F \cup F_o(\epsilon) \cup F_1(\epsilon))$$

Υποθέτουμε ότι

$$F_o(\epsilon) \subset F$$

Ή ισοδύναμα ότι

$$\epsilon < \left(\left\|q_i - q_j\right\| - \rho_j\right)^2 - \rho_i^2 , \qquad i,j \in \{1, \dots M\}, i \neq j$$

και

$$\epsilon < (\rho_o - \|q_i\|)^2 - \rho_i^2, \qquad i \in \{1, \dots, M\}$$

Πρόταση 1.3.5.2: Αν ο χώρος λειτουργίας είναι σωστά ορισμένος (όπως τον ορίσαμε παραπάνω) το σημείο προορισμού $\{q_d\}$ είναι ένα μη εκφυλισμένο τοπικό ελάχιστο της φ.

Πρόταση 1.3.5.2: Αν ο χώρος λειτουργίας είναι σωστά ορισμένος, όλα τα κρίσιμα σημεία της φ είναι στο εσωτερικό του F.

Από τις παραπάνω δύο προτάσεις και εφαρμόζοντας την Πρόταση 1.3.4.2 στο $F - \partial F - \{q_d\}$ αποδεικνύεται το θεώρημα 1.3.5.1 για τα σύνολα $F_o(\epsilon), F_1(\epsilon), F_2(\epsilon)$ χρησιμοποιώντας την χρησιμοποιώντας την $\hat{\phi}$ αντί της ϕ που είναι απλούστερο.

Η απόδειξη συνεχίζει σε δύο βήματα. Πρώτα δείχνεται ότι όλα τα κρίσιμα σημεία μπορούν να "τραβηχτούν" οσοδήποτε κοντά στο σύνορο του F και μετά βρίσκεται μία κατεύθυνση κατά μήκος της οποίας η $D^2\hat{\phi}$ έχει μία αρνητική ιδιοτιμή σε κάθε κρίσιμο σημείο. Έτσι αποδεικνύεται ότι το q_d μοναδικό ελάχιστο της $\hat{\phi}$

Πρόταση 1.3.5.3: Για κάθε $\epsilon > 0$ για την οποία υπάρχει θετική σταθερά $N(\epsilon)$ τέτοια ώστε αν $k \geq N(\epsilon)$ τότε η $\hat{\phi}$ δεν έχει κρίσιμα σημεία στο $F_2(\epsilon)$

Πρόταση 1.3.5.4: Αν ο χώρος λειτουργίας είναι σωστά ορισμένος, υπάρχει $\epsilon_o > 0$ έτσι ώστε η $\hat{\phi}$ δεν έχει τοπικά ελάχιστα στο $F_o(\epsilon)$ όταν $\epsilon < \epsilon_o$

Στην απόδειξη της παραπάνω πρότασης το ϵ φράζεται από δύο σταθερές τις $\epsilon_{oi}^{'}, \epsilon_{oi}^{''}$ και θεωρούμε ότι $\epsilon_o \triangleq \min\{\epsilon_{oi}^{'}, \epsilon_{oi}^{''}\}, i = 1, \dots, M$

Πρόταση 1.3.5.5: Αν το $k \geq N(\epsilon)$, τότε υπάρχει $\epsilon_1 > 0$ για το οποίο $\hat{\phi}$ δεν έχει κρίσιμα σημεία στο $F_1(\epsilon)$, όταν $\epsilon < \epsilon_1$

Η πρόταση αυτή καταλήγει πως $\epsilon_1 \triangleq (\rho_o)^2 - \|q_d\|^2$

Οι παραπάνω προτάσεις αρκούν για να αποδείξουν ότι η $\hat{\phi}$ είναι πολική. Τέλος αποδεικνύεται ότι η $\hat{\phi}$ είναι Morse

Πρόταση 1.3.5.6: Υπάρχει $\epsilon_2 > 0$ τέτοιο ώστε για κάθε $\epsilon < \epsilon_2$σε κάθε κρίσιμο σημείο της $\hat{\phi}$ στο στο $F_o(\epsilon), q \in \beta_{\hat{\phi}} \cap F_o(\epsilon)$ υπάρχει $T_q F_o(\epsilon) = P_q \oplus N_q$ όπου

$\dim(P_q) = 1$ για το οποίο $\xi_q|_{P_q}$ είναι θετικά ορισμένο και $\xi|_{N_q}$ είναι αρνητικά ορισμένο.

Λήμμα 1.3.5.7: Έστω $E^n = P \oplus N$, και έστω συμμετρικός πίνακας $Q \in \Re^{n\times n}$ ορίζει μία τετραγωνική μορφή στο E^n

$$\xi(v) \triangleq v^T Q\, v$$

Αν το $\xi_q|_{P_q}$ είναι θετικά ορισμένο και $\xi|_{N_q}$ είναι αρνητικά ορισμένο τότε υπάρχει ο Q^{-1} και επίσης

$$index(Q) = \dim(N)$$

Η απόδειξη του παραπάνω λήμματος δίνεται στην [6]

Σύμφωνα με το παραπάνω λήμμα και την πρόταση 1.3.5.6 όλα τα κρίσιμα σημεία της $\hat{\phi}$ είναι μη εκφυλισμένα.

Η πρόταση 1.3.5.6 καταλήγει ότι

$$\epsilon < \frac{1}{2}\rho_i^2 \triangleq \epsilon_{2i}^{'}$$

$$\epsilon < \frac{1}{4}\frac{\min_{\overline{B_i(\epsilon_{2i}^{'})}}\left\{\sqrt{\overline{\beta}_i}\ \ \|\nabla\beta_i\|\right\}}{\min_{\overline{B_i(\epsilon_{2i}^{'})}}\left\{\sqrt{|\hat{v}^T\nabla^2\overline{\beta}_i\hat{v}|}\right\}} \triangleq \epsilon_{2i}^{''}$$

και

$$\varepsilon_2 \triangleq \min(\epsilon_{2i}^{'}, \epsilon_{2i}^{''})$$

Τελικά λοιπόν

$$\epsilon_{\min} \triangleq \frac{1}{2}\min\{\epsilon_o, \epsilon_1, \epsilon_2\}$$

Παραπάνω δώσαμε συνοπτικά ένα σχεδιάγραμμα της απόδειξης. Αναλυτικά η απόδειξη δίνεται στο [1]

1.4Επέκταση των συναρτήσεων πλοήγησης στο multiagent πρόβλημα- Εφαρμογές στην εναέρια κυκλοφορία

1.4.1Εισαγωγικά

Στην πλοήγηση των αεροσκαφών(ή ρομπότ) θέλουμε όσο είναι δυνατό να γίνονται οι υπολογισμοί σε κάθε αεροσκάφος ξεχωριστά χωρίς να χρειάζεται επικοινωνία μεταξύ τους ή επικοινωνία όλων με κάποιο πύργο ελέγχου. Αυτό από μόνο του δίνει μία ανεξαρτησία στο αεροσκάφος αλλά μοιράζει και το υπολογιστικό κόστος με την έννοια ότι αντί να λύνεται ένα γενικό πρόβλημα χωρίζεται στα επί μέρους κομμάτια του και λύνεται από το κάθε αεροσκάφος η δική του τροχιά. Αυτό εκτός των άλλων δίνει και μία σθεναρότητα σε απρόοπτα που μπορεί να συμβούν κατά την πτήση και δεν υπάρχουν στο γενικό σχέδιο πτήσης που έχει προκύψει από πριν.

Στον έλεγχο εναέριας κυκλοφορίας ο Εντοπισμός Συγκρούσεων και Επίλυση του (Conflict Detection & Resolution-CD&R) χωρίζεται σε τρία βασικά μέρη.

- Μακροπρόθεσμου CD&R με ένα χρονικό ορίζονται ωρών
- Μεσοπρόθεσμου CD&R για ένα χρονικό ορίζονται δεκάδων λεπτών
- Βραχυπρόθεσμου CD&R για ένα χρονικό ορίζοντα 5-10 λεπτών

Οι πρώτες δύο περιπτώσεις είναι πιο πολύ ζητήματα βελτιστοποίησης,(πχ ελάχιστη κατανάλωση καυσίμου, άνεση των επιβατών, χρόνος πτήσης κτλ). Στον Βραχυπρόθεσμου CD&R το βασικό πρόβλημα είναι η αποφυγή σύγκρουσης. Είναι εκεί που ίσως κάτι έχει πάει στραβά και δύο αεροπλάνα έχουν βρεθεί αρκετά κοντά. Εδώ δεν λαμβάνονται υπόψη οι παραπάνω παράγοντες αλλά ο μόνος στόχος είναι η αποφυγή της σύγκρουσης χρησιμοποιώντας ότι μας επιτρέπεται από την δυναμική του αεροσκάφους. Εδώ βρίσκουν κύρια εφαρμογή οι συναρτήσεις πλοήγησης. Παρακάτω παρουσιάζουμε μία περιγραφή του προβλήματος βασιζόμενοι στην [10],[11].

1.4.2 Ορισμός του προβλήματος

Έστω μία περίπτωση N κινούμενων ρομπότ όπου

$$\dot{q}_i = \begin{bmatrix} \dot{x}_i \\ \dot{y}_i \end{bmatrix} = \boldsymbol{J}_i u_i$$

$$\dot{\phi}_i = \omega_i$$

Όπου με $q_i = \begin{bmatrix} x_i \\ y_i \end{bmatrix}$ συμβολίζουμε την θέση του i ρομπότ ως προς ένα γεώδετο σύστημα συντεταγμένων E, και με φ την γωνία μεταξύ του διαμήκους άξονα του ρομπότ και του άξονα x του E, και $J_i = \begin{bmatrix} \cos(\phi_i) \\ \sin(\phi_i) \end{bmatrix}$.

Κάθε ρομπότ i έχει ακτίνα r_i και οδηγείται από μία γραμμική και γωνιακή ταχύτητα u_i και ω_i αντίστοιχα. Επίσης ορίζουμε μία επιθυμητή ταχύτητα u_{di}. Ο χώρος εργασίας έχει μία ακτίνα έστω R_w.

Η μορφή της συνάρτησης πλοήγησης είναι

$$\Phi_i = \frac{\gamma_i + f_i}{((\gamma_i + f_i)^k + G_i)^{\frac{1}{k}}}$$

1.4.3 Αισθητήρες περιορισμένης εμβέλειας(limited sensing)

Η εμβέλεια των αισθητήρων του κάθε αεροσκάφους είναι η εξής. Θεωρούμε το σύστημα αξόνων που κινείται μαζί με το αεροσκάφος και το (0,0,0) βρίσκεται στο κέντρο του όπου ο άξονας x είναι ο διαμήκης άξονας του αεροσκάφους. Θεωρούμε τα δύο παρακάτω ελλειψοειδή

$$\left(\frac{x}{R_{sx}}\right)^2 + \left(\frac{y}{R_{sy}}\right)^2 + \left(\frac{z}{R_{sz}}\right)^2 = 1, \qquad x \geq 0$$

$$\left(\frac{x}{R_{sy}}\right)^2 + \left(\frac{y}{R_{sy}}\right)^2 + \left(\frac{z}{R_{sz}}\right)^2 = 1, \qquad x < 0$$

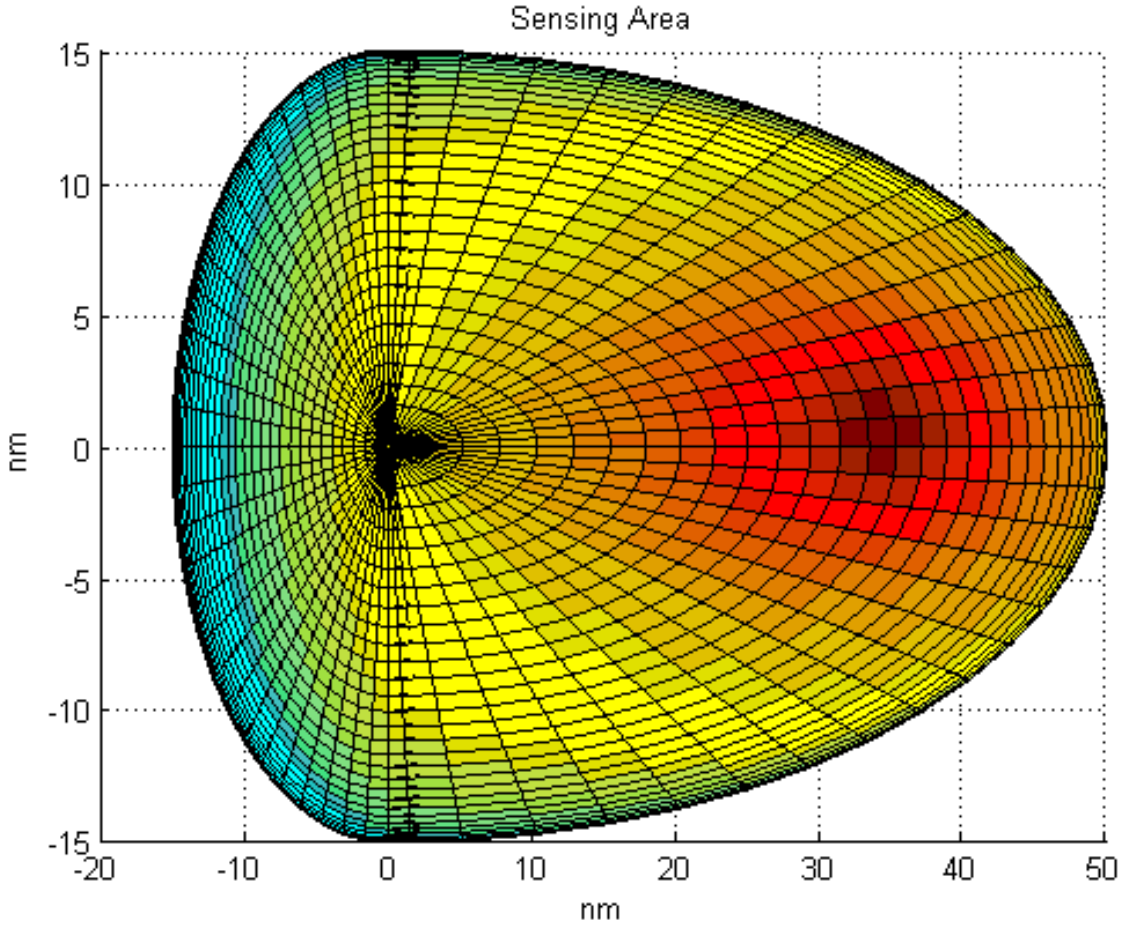
Όπου

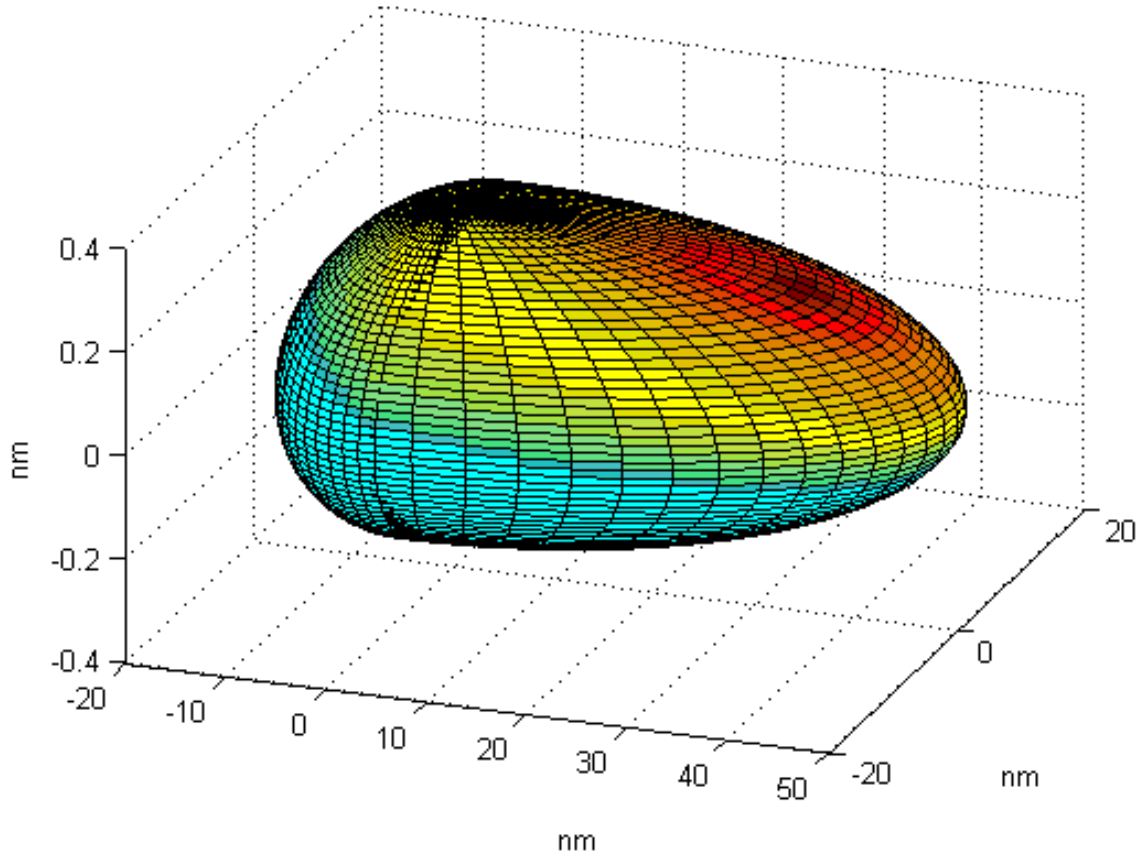
$$R_{sx} = 50nm$$

$$R_{sy} = 15nm$$

$$R_{sz} = 1800ft = 0.2962nm$$

Αυτό περιβάλει το αεροσκάφος και μας δίνει τα σημεία που κάθε στιγμή μπορεί το κάθε αεροσκάφος να δει αεροσκάφη που υπάρχουν στον χώρο που το περιβάλλει. Τα παραπάνω μεγέθη έχουν επιλεγεί έτσι ώστε να προσομοιώσουν μία τυπική πραγματική περίπτωση αεροσκάφους.





Εικόνες 1.4.3.1-2

1.4.4 Συνάρτηση πλοήγησης

Ο τυπικός ορισμός της συνάρτησης εμποδίου στις συναρτήσεις πλοήγησης δόθηκε στην 1.3.5. Σε αυτήν την εργασία εκτός από το case study των αεροσκαφών στην Κεντρική Ευρώπη παρουσιάζουμε μία διαφορετική κατασκευή της συνάρτησης εμποδίου έτσι ώστε να συμπεριληφθεί στην συνάρτηση πλοήγησης και η περιορισμένη εμβέλεια των αισθητήρων. Όπως αναφέρθηκε και παραπάνω η συνάρτηση πλοήγησης είναι της μορφής

$$\Phi_i = \frac{\gamma_i + f_i}{((\gamma_i + f_i)^k + G_i)^{\frac{1}{k}}}$$

Η παραπάνω εξίσωση είναι η συνάρτηση πλοήγησης που αφορά ένα αεροσκάφος i του προβλήματος. Ορίζουμε [10]

$$\gamma_i = \frac{\|q_i - q_{di}\|^2}{4 * R_w^2}$$

Όπου q_i, q_{di} είναι η θέση του αεροσκάφος και ο προορισμός του και R_w η ακτίνα του χώρου λειτουργίας. Στην συγκεκριμένη εφαρμογή πρέπει να αναφέρουμε ότι ο χώρος λειτουργίας είναι κάτι το υποκειμενικό αφού δεν έχει σαφή τρόπο ορισμού. Γενικά λαμβάνεται αρκετά μεγάλος έτσι ώστε να μην επηρεάζει υπερβολικά την κίνηση των αεροσκαφών. Βεβαίως πρέπει να περιλαμβάνει το τετράγωνο που ορίζει τον εναέριο χώρο που γίνονται οι εξομοιώσεις. Θεωρούμε $R_w = 1000nm$

Επίσης [12] ορίζουμε την συνάρτηση συνεργασίας

$$f_i(G_i) = \begin{cases} \alpha_o + \sum_{l=1}^{3} \alpha_l G_i^l \ , G_i \leq X \\ 0, \quad G_i > X \end{cases}$$

Όπου $\alpha_o = Y, a_1 = 0, a_2 = \frac{-3Y}{X^2}, \alpha_3 = \frac{2Y}{X^3}$ είναι θετικές σταθερές. Η συνάρτηση αυτή ενεργοποιείται όταν $G_i \leq X$. Το Y ορίζει μία μέγιστη τιμή για το f_i όταν $G_i = 0$.

1.4.5 Συνάρτηση εμποδίου

Για κάθε αεροσκάφος σύμφωνα με τους κανονισμούς ορίζεται χώρος γύρο από αυτό

$$\left(\frac{q_{ix}}{5}\right)^2 + \left(\frac{q_{iy}}{5}\right)^2 + \left(\frac{q_{iz}}{0.1646}\right)^2 = 1$$

Όπου q_x, q_y, q_z είναι συντεταγμένες ως προς το σύστημα αξόνων του αεροσκάφους.

Έστω αεροσκάφος i όπου μέσα στο εύρος των αισθητήρων του υπάρχουν αεροσκάφη N=1,2…,j,…M.

Το αεροσκάφος λοιπόν περιβάλλεται με ένα ελλειψοειδές το οποίο αποτελεί την συνάρτηση εμποδίου δηλαδή δεν μπορεί κάτι να περάσει μέσα από αυτό το ελλειψοειδές.

Έστω

$$b_{ij} = \left(\frac{q_x^{ij}}{5}\right)^2 + \left(\frac{q_y^{ij}}{5}\right)^2 + \left(\frac{q_z^{ij}}{0.1646}\right)^2 - 1$$

$$r_{ij}^1 = \left(\frac{q_x^{ij}}{50}\right)^2 + \left(\frac{q_y^{ij}}{15}\right)^2 + \left(\frac{q_z^{ij}}{0.2962}\right)^2 - 1\,, q_x^{ij} \geq 0$$

$$r_{ij}^2 = \left(\frac{q_x^{ij}}{50}\right)^2 + \left(\frac{q_y^{ij}}{15}\right)^2 + \left(\frac{q_z^{ij}}{0.2962}\right)^2 - 1, q_x^{ij} < 0$$

Τότε το b_{ij} ισούται με μηδέν πάνω στο εμπόδιο (τον χώρο που θεωρείται ότι καταλαμβάνει το αεροσκάφος), και τα r_{ij}^1, r_{ij}^2 ισούνται με μηδέν πάνω στον χώρο που μπορούν να δουν οι αισθητήρες κατά τα θετικά και αρνητικά του διαμήκους άξονα του αεροσκάφους και $q^{ij} = (q_x^{ij}, q_y^{ij}, q_z^{ij})$ η σχετική θέση των υπόλοιπων αεροσκαφών ως προς το συγκεκριμένο αεροσκάφος. Επίσης αν $r_{ij}^1 < 0$ ή $r_{ij}^2 < 0$ τότε γνωρίζουμε ότι j αεροσκάφος βρίσκεται στον χώρο των αισθητήρων του αεροσκάφους i.

Ορίζουμε

$$\hat{g}_{ij} = \frac{b_{ij}}{b_{ij} - r_{ij}}$$

Οπότε και

$$\hat{g}_{ij} = \begin{cases} 0, & b_{ij} = 0 \\ 1, & r_{ij} = 0 \end{cases}$$

Δηλαδή το μέγεθος $\hat{g}_{ij}$ ισούται με μηδέν αν βρισκόμαστε πάνω στο εσωτερικό ελλειψοειδές το οποίο περιβάλει το αεροσκάφος και με ένα όταν βρισκόμαστε στο εξωτερικό ελλειψοειδές όταν βρισκόμαστε επάνω στα όρια της περιοχής των αισθητήρων. Θεωρούμε επίσης ότι το $\hat{g}_{ij}$ ισούται με ένα και όταν κάποιο αεροσκάφος βρίσκεται εκτός της εμβέλειας των αισθητήρων οπότε τελικά

$$\hat{g}_{ij} = \begin{cases} 0, & b_{ij} = 0 \\ 1, & r_{ij} \geq 0 \end{cases}$$

Τέλος

$$g_{ij} = \begin{cases} L(\hat{g}_{ij}), & r_{ij} \leq 0 \\ 1, & r_{ij} > 0 \end{cases}$$

Όπου $L(x) = x^3 - 3x^2 + 3x$ η οποία ικανοποιεί

$$L(0) = 0$$

$$L(1) = 1$$

$$L'(x) > 0 \; \forall x \in [0,1)$$

$$L'(1) = L''(1) = 0$$

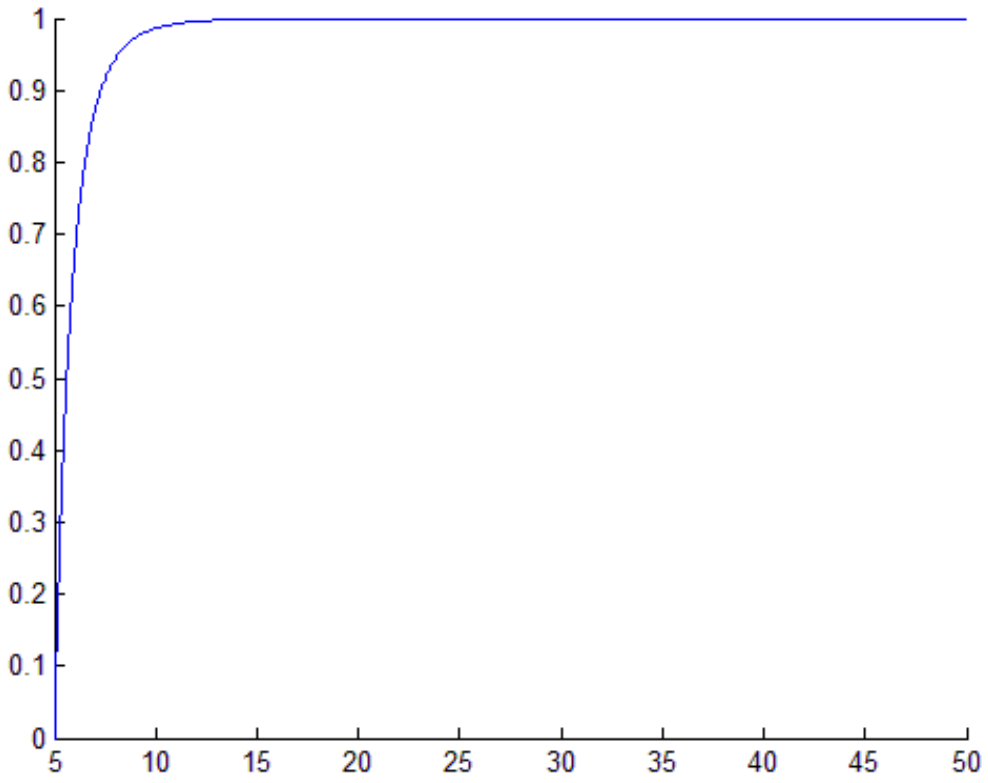
Η τελευταία αυτή συνάρτηση εισέρχεται έτσι ώστε να γίνει η g_{ij} από την κατασκευή της C^2

Πριν από την συνάρτηση L(x) μπορούμε επίσης να εισάγουμε μία συνάρτηση P(x) έτσι ώστε

$$g_{ij} = \begin{cases} L\big(P(\hat{g}_{ij})\big), & r_{ij} \leq 0 \\ 1 & , r_{ij} > 0 \end{cases}$$

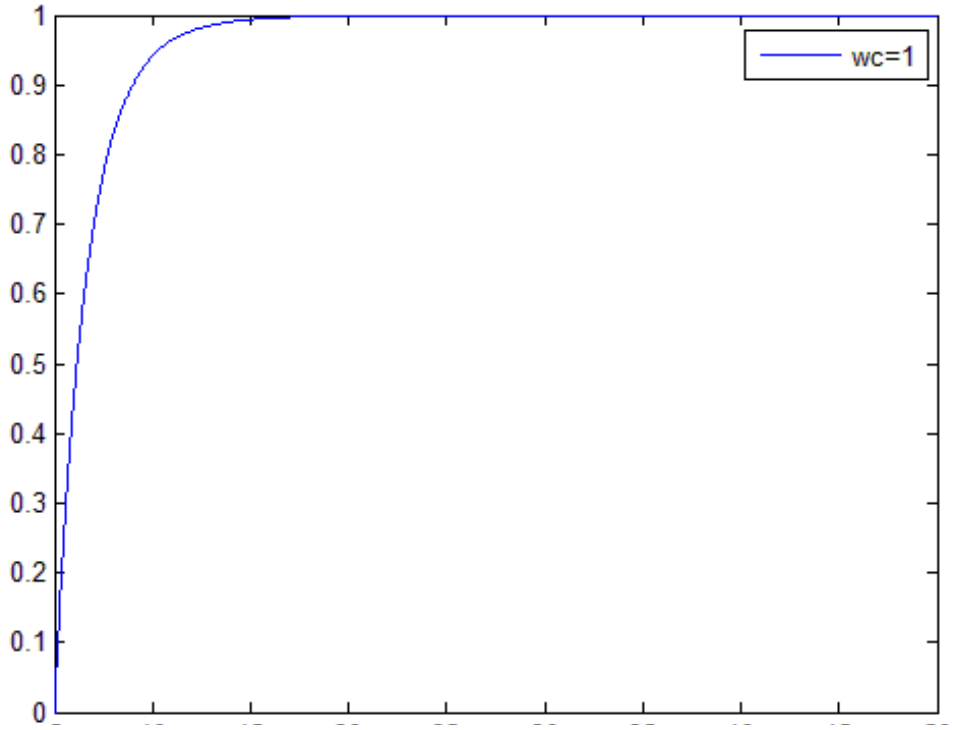
Όπου $P(x) = \frac{w_c g_{ij}}{wc + 1 - g_{ij}}$ με σκοπό να επιδράσουμε πάνω στην συνάρτηση εμποδίου.

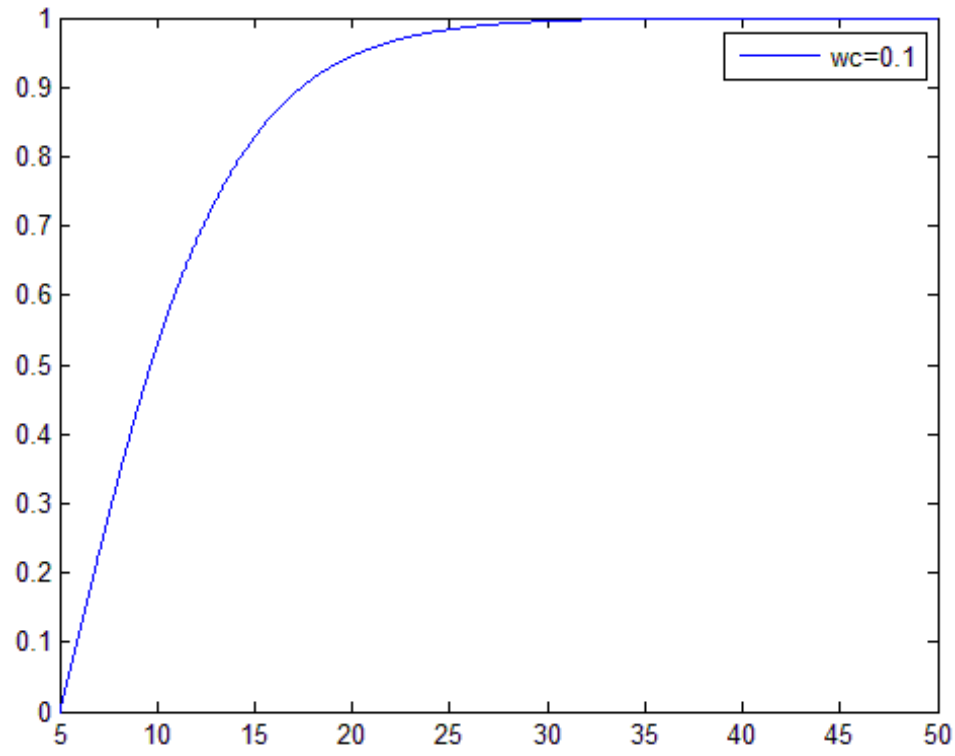
Παρακάτω δίνουμε την μεταβολή της συνάρτησης g_{ij} ως προς τον θετικό διαμήκη άξονα του αεροσκάφους.

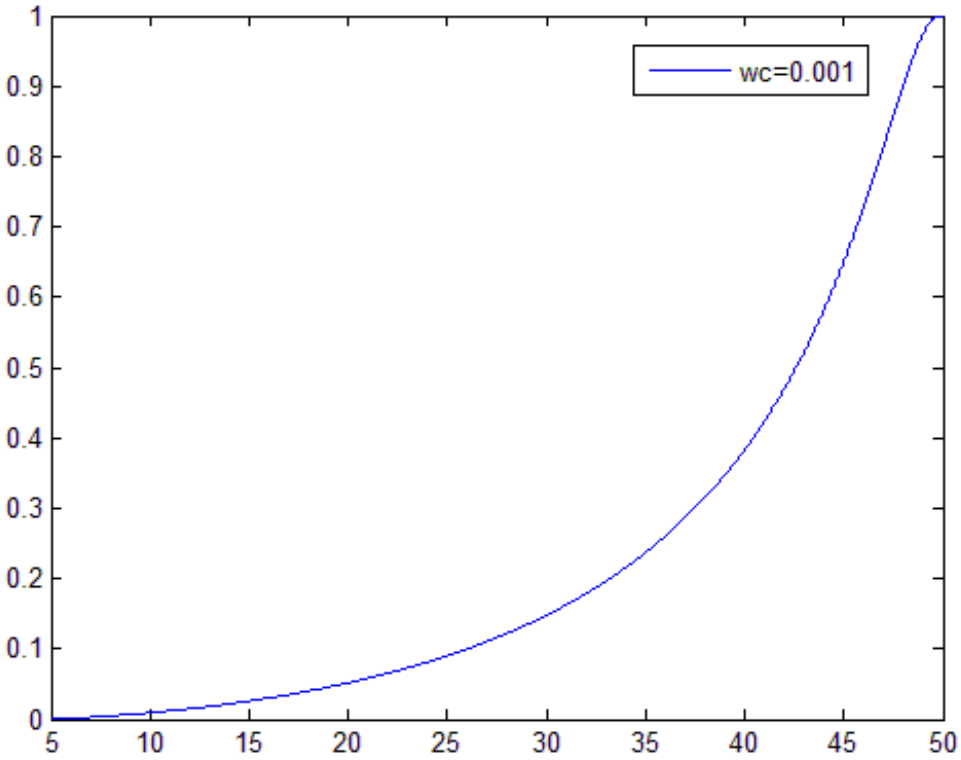


Εικόνα 1.4.4.1

Παρατηρούμε ότι η συνάρτηση εμποδίου παραμένει κοντά στο ένα που ισοδυναμεί με απώλεια αίσθησης του αεροσκάφους που πλησιάζει μετωπικά μέχρι περίπου τα 10-15nm όπου υπάρχει και μία απότομη πτώση της g_{ij} με αποτέλεσμα ο έλεγχος να επιδράσει έντονα πάνω στο αεροσκάφος για να αποτρέψει την σύγκρουση. Με την εφαρμογή και της P(x) έχουμε







Εικόνες 1.4.4.2-4

Δηλαδή παρατηρούμε ότι η συνάρτηση g_{ij} μεταβάλετε πιο ομαλά οπότε και οι αισθητήρες καταλαβαίνουν τα αεροσκάφη που εισέρχονται στον χώρο τους σχεδόν από την στιγμή εισόδου τους

Επίσης ορίζουμε την συνάρτηση εμποδίου του ορίου του χώρου λειτουργίας ως:

$$\hat{g}_{io} = \frac{b_{iw}}{b_{iw} - r_{iw}}$$

$$r_{iw} = \left(\frac{q_x^i}{R_{swx}}\right)^2 + \left(\frac{q_y^i}{R_{swy}}\right)^2 + \left(\frac{q_z^i}{R_{swz}}\right)^2 - 1$$

$$b_{iw} = \left(\frac{q_x^i}{R_w}\right)^2 + \left(\frac{q_y^i}{R_w}\right)^2 + \left(\frac{q_z^i}{R_w}\right)^2 - 1$$

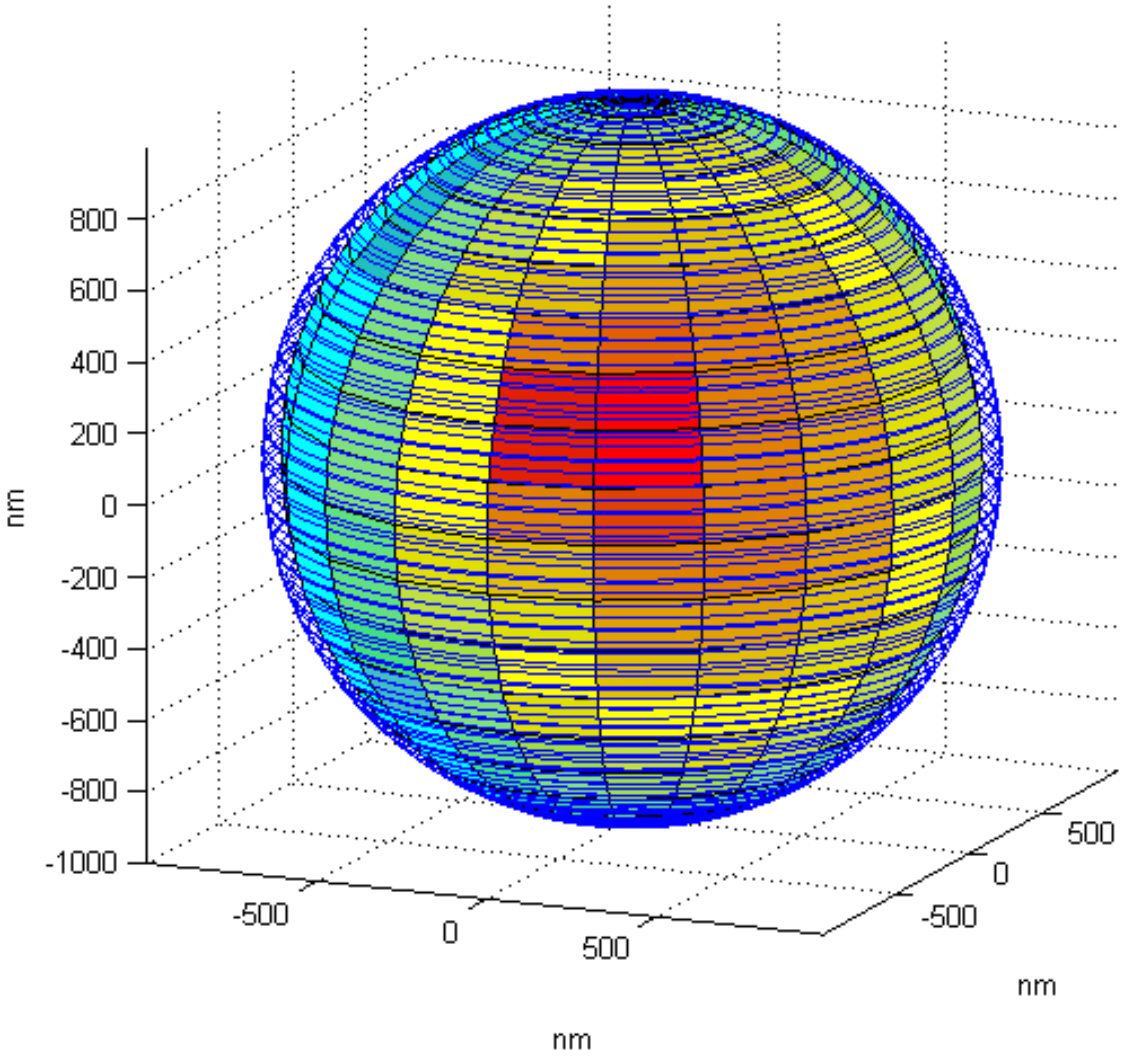
Όπου R_w είναι η ακτίνα του χώρου εργασίας και $\begin{bmatrix} R_{swx} \\ R_{swy} \\ R_{swz} \end{bmatrix} = \begin{bmatrix} R_w \\ R_w \\ R_w \end{bmatrix} - \begin{bmatrix} R_{sx} \\ R_{sy} \\ R_{sz} \end{bmatrix}$

Με $\begin{bmatrix} R_{sx} \\ R_{sy} \\ R_{sz} \end{bmatrix}$ τις τιμές των ημιαξόνων της περιοχής των αισθητήρων.

Έτσι το b_{iw} ισούται με μηδέν στο όριο του χώρου εργασίας και το r_{iw} ισούται με μηδέν στα όρια του ελλειψοειδούς με ημιάξονες $\begin{bmatrix} R_{swx} \\ R_{swy} \\ R_{swz} \end{bmatrix}$. Έτσι έχουμε

$$g_{io} = \begin{cases} L(\hat{g}_{io}), & r_{iw} \geq 0 \\ 1 & , r_{iw} < 0 \end{cases}$$

Πάντως σε αυτού του είδους τα προβλήματα το όριο του χώρου λειτουργίας δεν είναι ιδιαίτερα σημαντικό με την έννοια ότι είναι ένα τεχνητό όριο και το λαμβάνουμε αυθαίρετα και αρκετά μεγάλο έτσι ώστε να μην έχει ιδιαίτερη επίδραση στα αεροσκάφη. Παρακάτω παρουσιάζουμε μία εικόνα του πως περίπου φαίνεται ο χώρος στον οποίο ενεργοποιείται η συνάρτηση του ορίου του χώρου εργασίας. Εξωτερικά έχουμε τον κύκλο που δίνει τα όρια του χώρου εργασίας και εσωτερικά το ελλειψοειδές έξω από το οποίο ενεργοποιείται η συνάρτηση εμποδίου του ορίου του χώρου εργασίας.



Εικόνες 1.4.4.5

Τελικά παίρνουμε την συνάρτηση εμποδίου ως για το αεροσκάφος i ως

$$G_i = g_{io} \prod_{j=1}^{M} g_{ij}$$

Όπου M είναι ο αριθμός των αεροσκαφών μέσα στον χώρο τον αισθητήρων του.

1.4.6 Νόμος Ελέγχου[11]

Έστω

$$\dot{n}_i = \begin{bmatrix} \dot{x}_i \\ \dot{y}_i \end{bmatrix} = J_i u_i$$

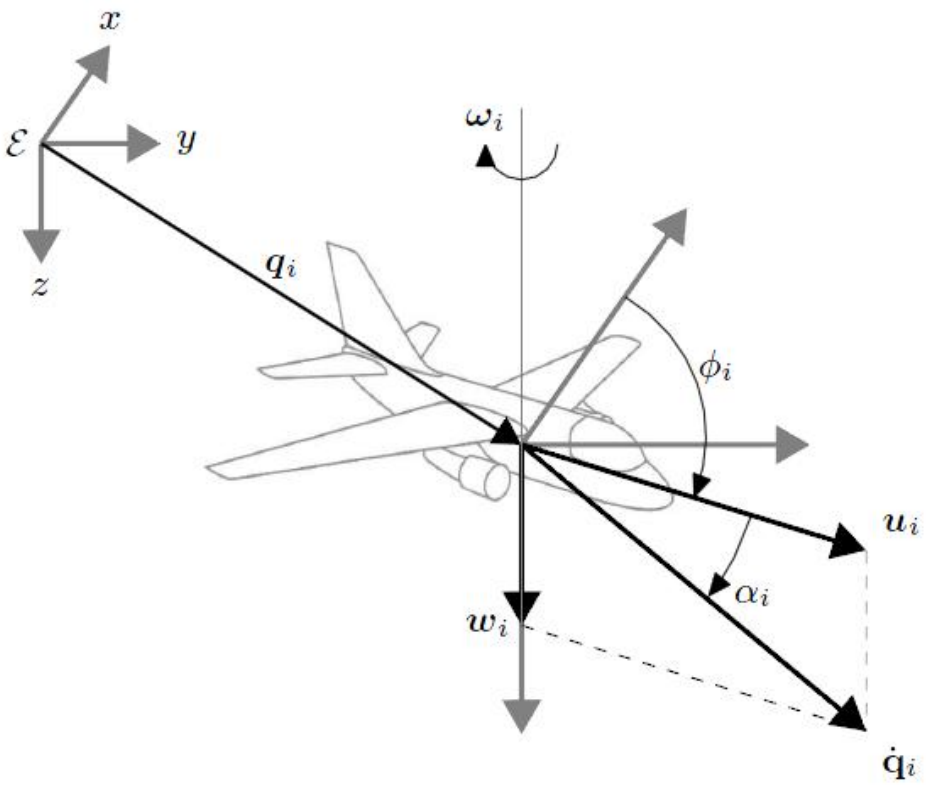
$$\dot{z}_i = w_i$$

$$\dot{\phi}_i = \omega_i$$

Όπου $q_i = \begin{bmatrix} x_i \\ y_i \\ z_i \end{bmatrix}$ η θέση του αεροσκάφους ως προς ένα απόλυτο σύστημα συντεταγμένων και $J_i = \begin{bmatrix} cos\phi_i \\ sin\phi_i \end{bmatrix}$. Ορίζουμε την γωνία ανόδου/καθόδου ως την γωνία μεταξύ των $\dot{q}_i$ και του επιπέδου x-y

$$\alpha_i = \tan^{-1}\left(\frac{w_i}{|u_i|}\right)$$

Έστω επίσης u_{id} μία επιθυμητή βέλτιστη ταχύτητα πλεύσης και α_{iC}, α_{iD} μία μέγιστη γωνία ανόδου και καθόδου αντίστοιχα.



Έχουμε επίσης

$$\dot{\Phi}_i = \sum_{j=1}^{M} \nabla_j \Phi_i^T \, q_j = P_i \dot{u}_i + \Phi_{iz} w_i + \frac{\partial \Phi_i}{\partial t}$$

$$a_{nhi} = -\tan^{-1}\left(\frac{\Phi_{iz}}{\sqrt{\Phi_{ix}^2 + \Phi_{iy}^2}}\right)$$

$$\tilde{\alpha}_i = \begin{cases} \alpha_{iD}, a_{nhi} < a_{iD} \\ a_{nhi}, a_{iD} \le a_{nhi} \le a_{iC} \\ a_{iC}, a_{nhi} > a_{iC} \end{cases}$$

$$\tilde{t}_i = tan\tilde{\alpha}_i$$

Και ορίζουμε τα δύο παρακάτω switches

$$\sigma_{\Phi i} = sat\left(\frac{|u_i|(\tilde{t}_i \Phi_{iz} - |P_i| + \epsilon) + \frac{\partial \Phi_i}{\partial t}}{|u_i| \tilde{t}_i \Phi_{iz}}\right)$$

$$\sigma_{\alpha i} = sat\left(\frac{\hat{\theta}_i - |a_{nhi}|}{\hat{\theta}_i - \theta_i^0}\right)$$

όπου

$$sat(x) = \begin{cases} 0, x < 0 \\ x, 0 \le x \le 1 \\ 1, x > 1 \end{cases}$$

Και θ_i^0 , $\hat{\theta}_i$ ένα άνω και κάτω όριο για την a_{nhi} που που επιτρέπεται να εμφανιστεί κάθετη ταχύτητα στο αεροσκάφος.

Η λογική του ελέγχου είναι η εξής

- Διατηρούμε u_{id} όταν $\frac{\partial \Phi_i}{\partial t} \le u_{id}(|P_i| - \tilde{t}_i \Phi_{\mathrm{iz}} - \epsilon)$ όπου ϵ μικρή θετική σταθερά
- Διατηρούμε $w_i = 0$ όταν για την οριζόντια βέλτιστη ταχύτητα $\sigma_{\Phi i} = \sigma_{ai} = 1$
- Η κλίση ανόδου γίνεται ίση με $\tilde{t}_i$ αν $\sigma_{\Phi i} = 0$ ή $\sigma_{ai} = 0$

- Η ω_i χρησιμοποιεί τον νόμο ελέγχου της [13] έτσι ώστε ν προσανατολίζει το αεροσκάφος προς τον προορισμό του

Έτσι χρησιμοποιούμε τις παρακάτω σχέσεις

$$u_i = \begin{cases} -s_i u_{id}, \dfrac{\partial \Phi_i}{\partial t} \leq u_{id}(|P_i| - \tilde{t}_i \Phi_{iz} - \epsilon) \\ -s_i \dfrac{u_{id}\epsilon + \dfrac{\partial \Phi_i}{\partial t}}{|P_i| - \tilde{t}_i \Phi_{iz}}, \dfrac{\partial \Phi_i}{\partial t} > u_{id}(|P_i| - \tilde{t}_i \Phi_{iz} - \epsilon) \end{cases}$$

Όπου

$$s_i = sgn(P_i)$$

$$w_i = (1 - \min(\sigma_{\Phi i}, \sigma_{\alpha i}))\tilde{t}_i |u_{id}|$$

$$\omega_i = \begin{cases} 0, & M_i \geq \epsilon_\phi \\ \Omega_i \left(1 - \dfrac{M_i}{\epsilon_\phi}\right), & 0 < M_i < \epsilon_\phi \\ \Omega_i, & M_i \leq 0 \end{cases}$$

Όπου

$$M_\iota \triangleq \dot{\phi}_{nhi}(\phi_i - \phi_{nhi})$$

$$\Omega_i \triangleq -k_\phi(\phi_i - \phi_{nhi}) + \dot{\phi}_{nhi}$$

$$\phi_{nhi} \triangleq atan2(sgn(pi)\Phi_{it}, s_i \Phi_{ix})$$

$$p_i = J_{id}^T(n_{i1} - n_{i1d})$$

Με τα ϵ_ϕ, k_ϕ μία μικρή θετική σταθερά και ένα θετικό κέρδος αντίστοιχα

ΚΕΦΑΛΑΙΟ 2 Εξομοίωση και αποτελέσματα

2.1 Εισαγωγή

Θα χρησιμοποιήσουμε τις συναρτήσεις πλοήγησης όπως περιγράφηκαν παραπάνω για να εξομοιώσουμε την εναέρια κυκλοφορία πάνω από την σε ένα νοητό τετράγωνο που εκτείνεται $\sim \pm 200nm$ στους άξονες x,y με κέντρο την Ζυρίχη. Αυτό περιλαμβάνει την περιοχή της Ελβετίας και δεδομένου ότι από την Ανατολή στην Δύση και από τον Βορρά στον νότο οι μεγαλύτερες αποστάσεις με βάση τα σύνορα της είναι περίπου 188nm και 119nm αντίστοιχα εμβαδόν αυτό που εξετάζουμε περιλαμβάνει και τμήματα των γύρω χωρών(Αυστρία, Γαλλία, Γερμανία, Ιταλία). Ο χώρος αυτός βρίσκεται στην κεντρική Ευρώπη κάτι που κάνει μεγάλο αριθμό πτήσεων να περνάνε καθημερινά πάνω από αυτών και ας μην έχουν προορισμό την Ελβετία. Η εξομοίωση αυτή θα γίνει χρησιμοποιώντας σαν βασικό εργαλείο τις συναρτήσεις πλοήγησης όπως περιγράφηκαν παραπάνω. Βασικός στόχος είναι να ελέγξουμε πως θα συμπεριφερθούν τα αεροσκάφη τα οποία εκτελούν πτήσεις που έχουν σαν βάση πραγματικά δεδομένα. Παρόλο που γενικά οι συναρτήσεις πλοήγησης είναι ένα εργαλείο που χρησιμοποιείται κυρίως στο short term collision avoidance εμείς το εφαρμόζουμε σαν έναν "αυτόματο πιλότο" ο οποίος θα πάρει σαν είσοδο από τον χρήση *μόνο* τον προορισμό του αεροσκάφους και θα το οδηγήσει εκεί. Τα αεροσκάφη υποθέτουμε ότι χρησιμοποιούν όλα την ίδια μέθοδο και ότι έχουν μία περιορισμένη γνώση του χώρου γύρω από αυτά με βάση τυποποιημένους κανονισμούς. Τα αεροσκάφη δεν επικοινωνούν μεταξύ τους έτσι ώστε να υπάρχει κάποια συνεργασία στις κινήσεις που εκτελούν αλλά ακολουθείται ένα αυστηρώς decentralized μοτίβο στο οποίο οι απαιτούμενοι υπολογισμοί γίνονται πάνω στο αεροπλάνο χωρίς να υποτίθεται καμιά περαιτέρω επικοινωνία με τα διάφορα αεροσκάφη ή ακόμα και με κάποιο αεροδρόμιο. Βέβαια υποθέτουμε επίσης ότι για τα αεροσκάφη τα οποία βρίσκονται μέσα στην εμβέλεια των αισθητήρων κάποιου αεροσκάφους αυτό γνωρίζει ακριβώς την θέση τους χωρίς ύπαρξη αβεβαιότητας(κάτι το οποίο δεν είναι παράλογη παραδοχή αφού το κάθε αεροσκάφος μοντελοποιείται σαν ελλειψοειδές με ημιάξονες $R_x = 2.5nm\,, R_y = 2.5nm, R_z = 1000ft \approx 0.165nm$(στις συναρτήσεις πλοήγησης μοντελοποιούμε τους διπλάσιους ημιάξονες έτσι ώστε να λάβουμε υπόψη μας τους χώρους και των δύο αεροσκαφών) όπου ο άξονα R_z είναι ο κάθετος στο επίπεδο πτήσης. Το πείραμα αυτό λοιπόν θα προσπαθήσει να δείξει αν μία τέτοια υλοποίηση είναι εφικτή καθώς και να παρουσιάσει κάποια στατιστικά αποτελέσματα των σημαντικότερων μεγεθών που υπολογίστηκαν κατά την διαδικασία αυτή. Επίσης στόχος της εργασίας αυτής είναι ο κώδικας που θα προκύψει στο τέλος να είναι μπορεί να χρησιμοποιηθεί εύκολα και αποδοτικά και για άλλες μελλοντικές εξομοιώσεις στις οποίες μπορούν να αλλάζουν κάποιες παράμετροι του προβλήματος(όπως sensing range, χώρος πάνω από τον οποίο γίνεται η εξομοίωση, τύποι αεροσκαφών, ύψη πτήσης κ.ά.) ή το αρχικά δεδομένα των πτήσεων (διαφορετικές ημέρες, άλλα πραγματικά ή φανταστικά

σενάρια) ή ακόμα και ο "πυρήνας" δηλαδή η παρούσα υλοποίηση των συναρτήσεων πλοήγησης έτσι ώστε να μπορεί να γίνει σύγκριση διαφορετικών και αξιολόγηση διαφορετικών μεθόδων. Η εξομοίωση γίνεται με χρήση κώδικα που γράφτηκε σε Matlab. Παρακάτω περιγράφουμε την εξομοίωση και τα αποτελέσματα.

2.2Δεδομένα

Το δεδομένα αφορούν την εναέρια κυκλοφορία σε ολόκληρη την Ευρώπη για περίπου δύο ημέρες.(20 Ιουλίου 2006 00:34 έως 21 Ιουλίου 2006 23:19). Τα δεδομένα αυτά είναι βασισμένα σε πραγματικές πτήσεις που πραγματοποιήθηκαν την πρώτη ημέρα και για την δεύτερη μέρα και έχουν προστεθεί πτήσεις έτσι ώστε να είναι μία εκτίμηση της εναέριας κυκλοφορίες που θα υπάρχει έως το 2050. Εκείνη την περίοδο υπολογίζεται ότι θα υπάρχει τριπλασιασμός των πτήσεων. Στην αρχή της εκτέλεσης των πειραμάτων δημιουργείται ένα αρχείο flights.mat το οποίο περιέχει τις συντεταγμένες X,Y των διαφόρων αεροδρομίων από τα οποία ξεκίνησαν πτήσεις ,και την ώρα την οποία έγινε η απογείωση, οι οποίες βρέθηκαν κάποια στιγμή μέσα στο εμβαδόν που εξετάζουμε. Περιέχεται επίσης η ταχύτητα κάθε αεροσκάφους στα διάφορα ύψη πτήσης που επιτρέπεται να πετάξει. Γενικά αυτά είναι τα βασικά δεδομένα πάνω στα οποία χτίζουμε το πείραμα. Όλα τα υπόλοιπα προκύπτουν κατά την διάρκεια της εξομοίωσης. Έχουμε στην διάθεσή μας και δεδομένα όπως τύπος των διαφόρων αεροσκαφών, διάφοροι αεροδυναμικοί συντελεστές τους, τύποι των μηχανών κτλ. Το πείραμα αυτό αφορά όμως κάποιες τροχιές οι οποίες θα δώσουν μία γενική εικόνα της εναέριας κυκλοφορίας στο μέλλον. Στην μεθοδολογία αυτή υποθέτουμε ότι οι κατευθύνσεις του δυναμικού που υπολογίζονται μπορούν να ακολουθηθούν από το αεροσκάφος πχ από ένα βασικότερο σύστημα ελέγχου που λαμβάνει υπόψη συγκεκριμένα την δυναμική του. Αυτό είναι πρακτικά αποδεκτό αφού υπάρχουν περιορισμοί έτσι ώστε οι τροχιές που ακολουθούνται να είναι ρεαλιστικές (χωρίς απότομες αλλαγές κατεύθυνσης).

2.3 Αρχική επεξεργασία

Από τα παραπάνω δεδομένα αρχικά χρειαζόμαστε να εξάγουμε τα εξής βασικά στοιχεία κάθε αεροσκάφους που θα χρησιμοποιήσουμε στην συνέχεια.

1)Ποια από τα αεροσκάφη απογειώνονται ή προσγειώνονται στον χώρο που εξετάζουμε.

2)Για τα αεροσκάφη για τα οποία ένα μέρος της πτήσης τους βρίσκεται εκτός του τετραγώνου να βρούμε το/α σημείο/α στα οποία η πορεία του αεροσκάφους το τέμνει. Για την εύρεση των σημείων αυτών η πορεία υποτίθεται ευθεία δηλαδή αυτή που θα ακολουθούσε το αεροσκάφος αν δεν συναντούσε άλλα στον δρόμο του.

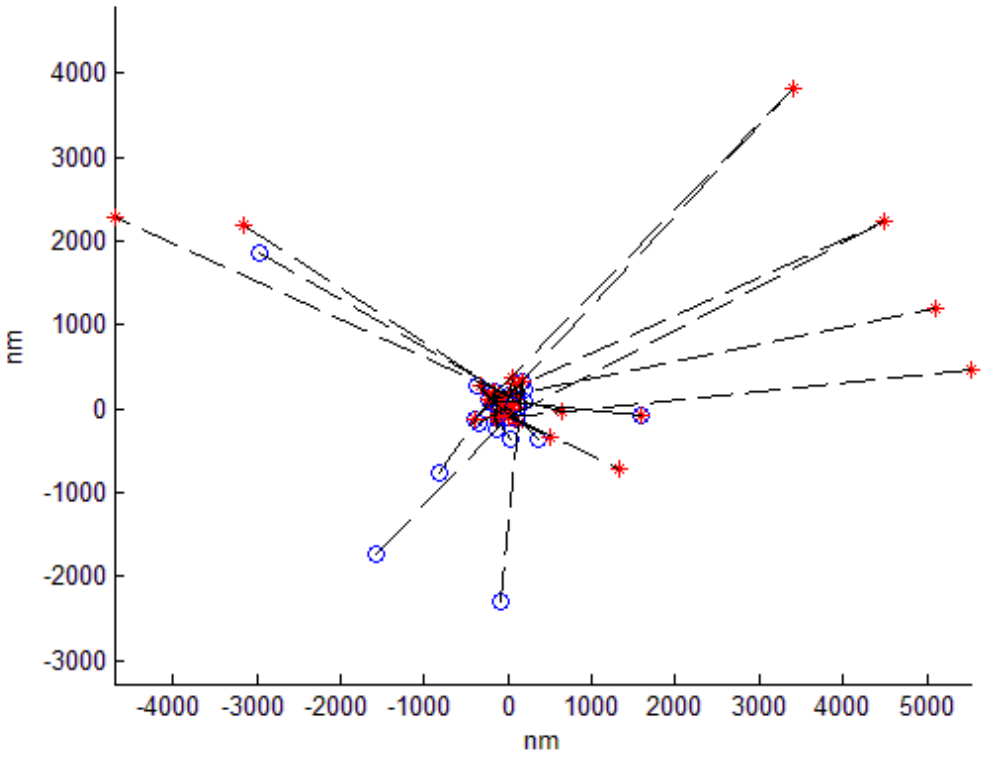
3)Αν το αεροσκάφος εκκινήσει από αεροδρόμιο εκτός του εξεταζόμενου χώρου μας ενδιαφέρει και ο χρόνος την στιγμή που αυτό εισέρχεται σε αυτόν. Αυτός

υπολογίζεται υποθέτοντας ότι το αεροσκάφος είχε μία ευθεία πορεία από την στιγμή που απογειώθηκε μέχρι την στιγμή που πρωτομπήκε στον χώρο που εξετάζουμε.

Σημείωση: Γενικά πρέπει να έχουμε στο νου μας ότι διάφοροι τέτοιου τύπου υπολογισμοί δεν επηρεάζουν το αν η μέθοδος τελικά λειτουργήσει η όχι αφού πχ αν κάποια αεροσκάφη είχαν κάποια καθυστέρηση απλά θα εισέλθουν στον χώρο λίγο αργότερα. Θα αλληλεπιδράσουν βέβαια με λίγο διαφορετικό τρόπο με το περιβάλλον τους αφού τα αεροσκάφη που ίσως βρουν δεν θα είναι ακριβώς τα ίδια αλλά αυτή είναι η αξία της μεθόδου αυτής. Ότι μπορεί να επιλύσει το πρόβλημα του εναέριας κυκλοφορίας χωρίς κανένα δεδομένο εκτός από αυτά που περιγράψαμε παραπάνω και κάνοντας υπολογισμούς μόνο κατά την διάρκεια της πτήσης και μεμονωμένα στο κάθε αεροσκάφος. Εδώ να αναφέρουμε βέβαια ότι στην περίπτωση κατά την οποία κάποιο αεροσκάφος δεν συναντήσει κανένα άλλο στον δρόμο του τότε αυτό θα ακολουθήσει την ευθεία γραμμή που συνδέει το σημείο εκκίνησης με το σημείο προορισμού του.

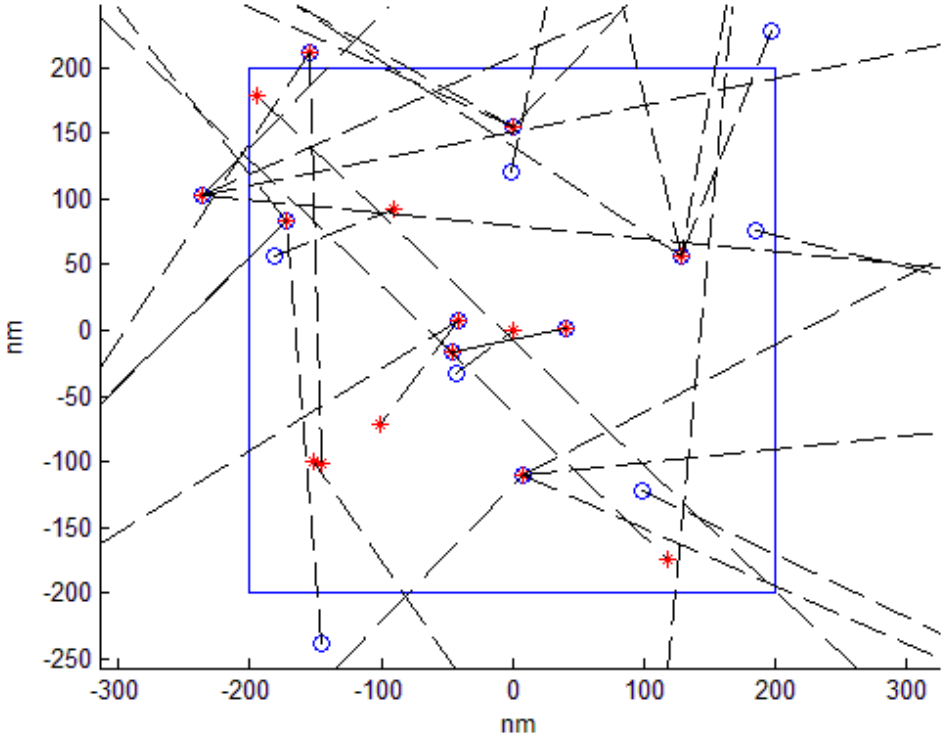
Αυτού του είδους οι υπολογισμοί καθώς και διάφοροι άλλοι που αφορούν την δομή διαφόρων δεδομένων και τον τρόπο που αυτά θα εισαχθούν στο πρόγραμμα παρακάτω γίνονται στο αρχείο *updateflights.m.* Τα δεδομένα αποθηκεύονται σε ένα mat αρχείο για να χρησιμοποιηθούν αργότερα από το πρόγραμμα. Αυτό έχει ως αποτέλεσμα αν θέλουμε να μπορούμε να εισάγουμε ένα δείγμα στο πρόγραμμα (κάποιων ημερών ή και μεγαλύτερων χρονικών περιόδων) στο οποίο θα τρέξουμε μία φορά την υπορουτίνα αυτή και αργότερα να κάνουμε όσες εξομοιώσεις χρειαζόμαστε επιλέγοντας συγκεκριμένα χρονικά διαστήματα μόνο ή συγκεκριμένα αεροσκάφη ή ομάδες αεροσκαφών που διαθέτουν ένα χαρακτηριστικό που θέλουμε να μελετήσουμε.(πχ απογειώνονται *και* προσγειώνονται στον χώρο που εξετάζουμε κτλ).

Τελικά παίρνουμε τα επιθυμητά δεδομένα. Παρακάτω παρουσιάζονται μερικές εικόνες. Στις πρώτες δύο δείχνουμε τα αεροδρόμια απογείωσης(μπλε χρώμα) και τα αεροδρόμια προσγείωσης(κόκκινο χρώμα) και στην τρίτη τα σημεία που εισάγουμε στον κώδικα (δηλαδή τα σημεία αν υπάρχουν τα οποία κόβει η διαδρομή του αεροσκάφους τα σύνορα του τετραγώνου που ελέγχουμε).



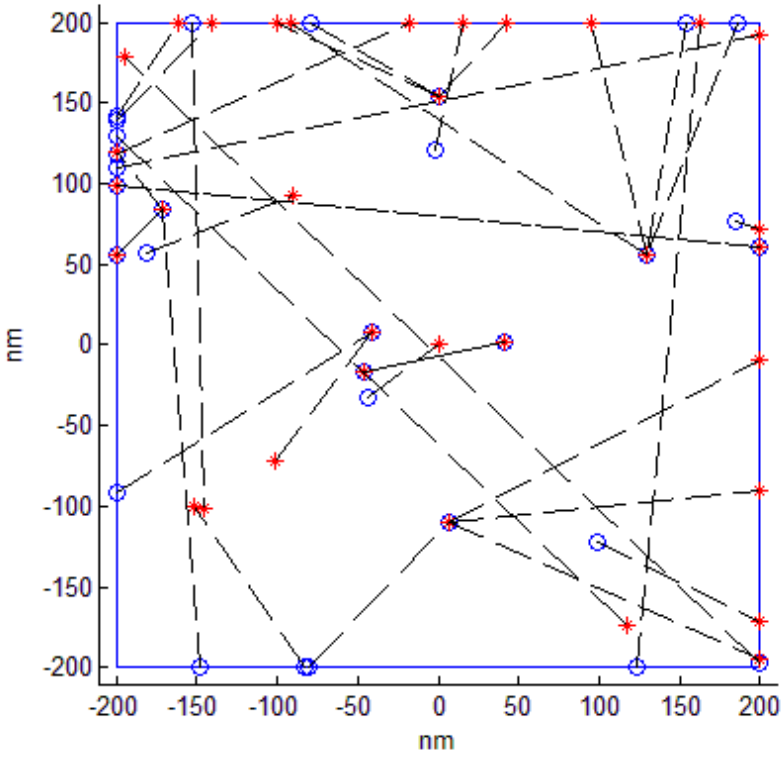
Εικόνα 2.3.1

Τα σημεία απογείωσης(μπλε) και προσγείωσης (κόκκινο) των αεροσκαφών



Εικόνα 2.3.2

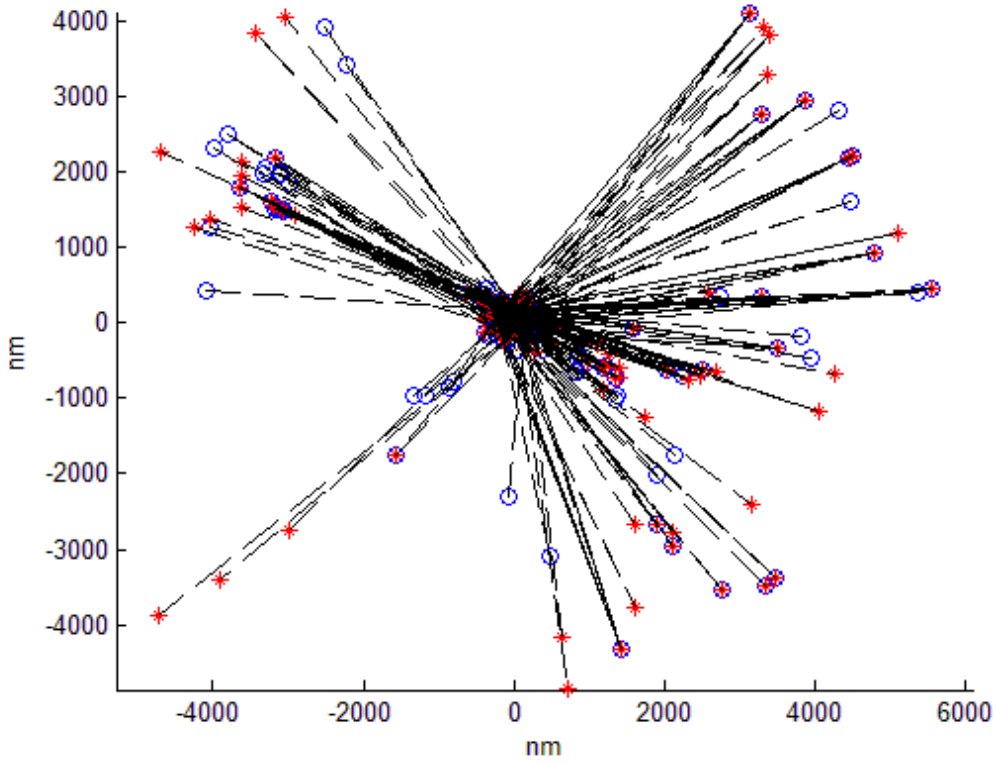
Εστίαση στον χώρο που μας ενδιαφέρει



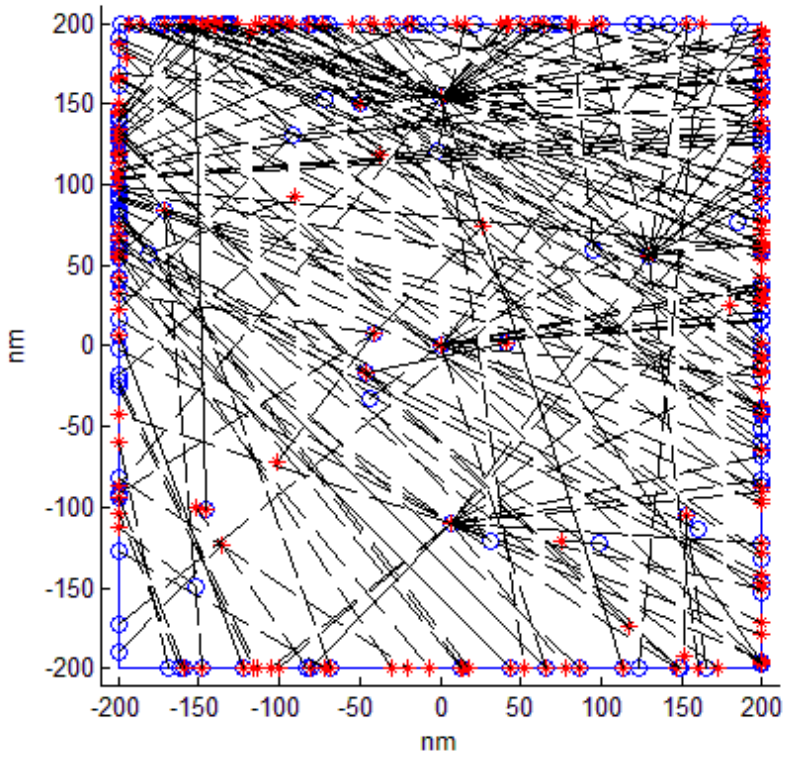
Εικόνα 2.3.3

Τα σημεία που λαμβάνουμε σαν αρχή και τέλος μίας διαδρομής στον κώδικα

Οι παραπάνω εικόνες αφορούν τα πρώτα 100 αεροσκάφη. Συγκρίνοντας την εικόνα 2.3.2 με την εικόνα 2.3.3 βλέπουμε ότι οι εικόνες συμφωνούν. Σε διάφορα αεροδρόμια παρατηρούμε ότι προσγειώνονται και απογειώνονται αεροσκάφη. Φυσικά οι γραμμές είναι λιγότερες από 100 αφού και ίδιες διαδρομές μπορούν να γίνονται από διαφορετικά αεροσκάφη σε διαφορετικές χρονικές στιγμές. Βεβαίως και η ώρα η οποία εισέρχεται κάθε αεροσκάφος είναι διαφορετική αλλά βλέπουμε ότι αν πάνω σε αυτές τις γραμμές πετάνε συνεχώς αεροπλάνα και δεδομένου ότι κάποιες έρχονται αρκετά κοντά δεν είναι προφανές πως με ένα decentralized τρόπο μπορούμε να πετύχουμε αποφυγή συγκρούσεων. Επίσης αλγοριθμικές ή decision based μέθοδοι πάντα αρχίζουν και γίνονται πιο "δυσκίνητες" όσο αυξάνει η πολυπλοκότητα ενός προβλήματος. Έτσι εδώ εφαρμογή των συναρτήσεων πλοήγησης παρουσιάζουν ενδιαφέρον. Εδώ επειδή αναφερόμαστε σε κεντρική Ευρώπη περιμένουμε να διασταυρώνονται πολλά αεροσκάφη. Επίσης παρόλο που τα αεροσκάφη αυξάνονται κατά τις ώρες τις ημέρας τότε περιμένουμε η εναέρια κυκλοφορία να αυξάνεται αρκετά κατά την διάρκεια της μέρα. Για να δείξουμε πόσο πολύπλοκο μπορεί να γίνει το πρόβλημα δίνουμε κάτω τα ίδια διαγράμματα για τους 1000 πρώτους agents.



Εικόνα 2.3.4



Εικόνα 2.3.5

Τρέχουμε το πρόγραμμα καλώντας την ρουτίνα *main_traffic.m*. Αυτή περιλαμβάνει τρία βασικά calls, τις υπορουτίνες init_NF_tr, TrafficData,navigate_tr. Στην υπορουτίνα *init_NF_tr* δηλώνουμε όλες τις βασικές μεταβλητές που θα χρησιμοποιηθούν κατά την διάρκεια του προγράμματος. Εδώ δηλώνουμε το βήμα της εξομοίωσης, τις βασικές παραμέτρους των συναρτήσεων πλοήγησης, τα όρια του κάθε αεροσκάφους, το βελησνεκές των αισθητήρων του κτλ. Γενικά οποιεσδήποτε βασικές μεταβλητές χρειάζονται για την εκτέλεση του προγράμματος. Από εδώ πολύ εύκολα μπορούμε να αλλάξουμε κάποια δεδομένα και να παρατηρήσουμε τις αλλαγές σε κάποια εξομοίωση.

Με την υπορουτίνα TrafficData επιλέγουμε τα αεροσκάφη που θα λάβουν μέρος στο πρόγραμμα. Επιλέγουμε αριθμό αεροσκαφών, ποιον αεροσκάφος θα θεωρήσουμε πρώτο από το δείγμα μας, αν διαθέτουμε μία ομαδοποίηση των αεροσκαφών κατά κατεύθυνση(βορράς, νότος, ανατολή, δύση) μπορούμε να επιλέξουμε να χρησιμοποιήσουμε αεροσκάφη μόνο της συγκεκριμένης κατεύθυνσης ή μπορούμε κατά την επιλογή να αγνοήσουμε κάποια(επιλογή ανά ένα κτλ).

Τέλος στην υπορουτίνα *navigate_tr* τρέχει η μέθοδος και γίνεται η εξομοίωση. Σε αυτήν περιλαμβάνεται η επιθυμητή υλοποίηση των συναρτήσεων πλοήγησης.

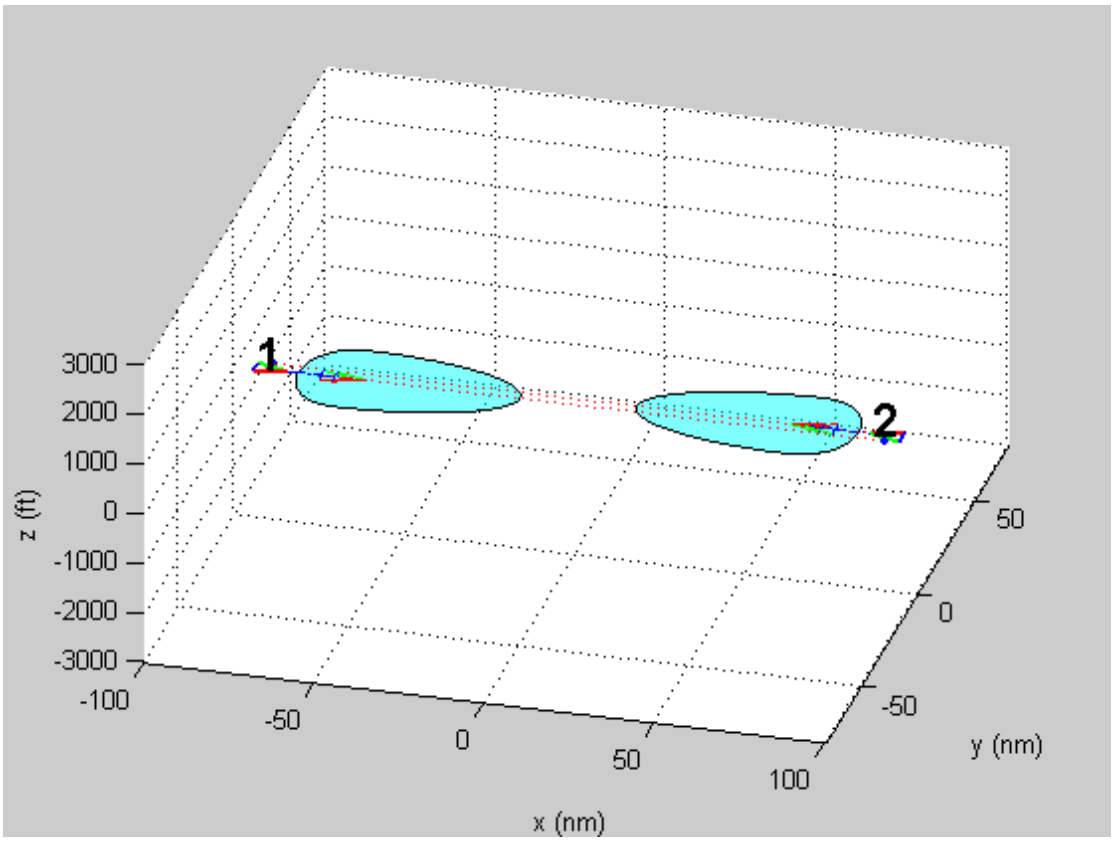
2.4Παραδείγματα Αποφυγής Σύγκρουσης

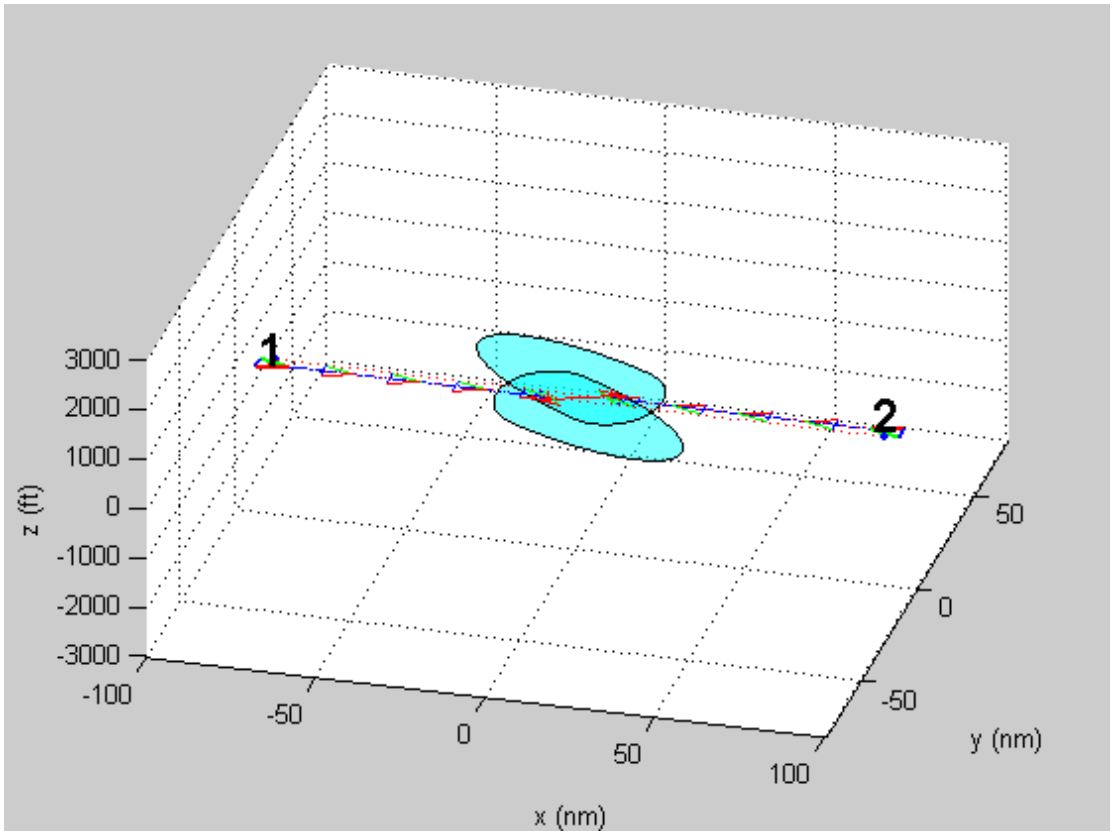
Παρακάνω δείχνουμε δύο απλά παραδείγματα αποφυγής συγκρούσεων χρησιμοποιώντας τον αλγόριθμο. Στην εξομοίωση πολλών αεροσκαφών είναι δυσκολότερο να δείξουμε τέτοιου είδους εικόνες και λόγω του πλήθους τους αλλά και λόγω του μεγέθους του συνολικού χώρο σε σχέση με την ακτίνα του αεροσκάφους που κάνει τις αποκλίσεις από τις πορείες τους δυσδιάκριτες. Επιλέξαμε να κάνουμε την εξομοίωση στις δύο διαστάσεις έτσι ώστε να δούμε πως αυτά κινούνται χωρίς να αλλάζουν ύψος πτήσης.

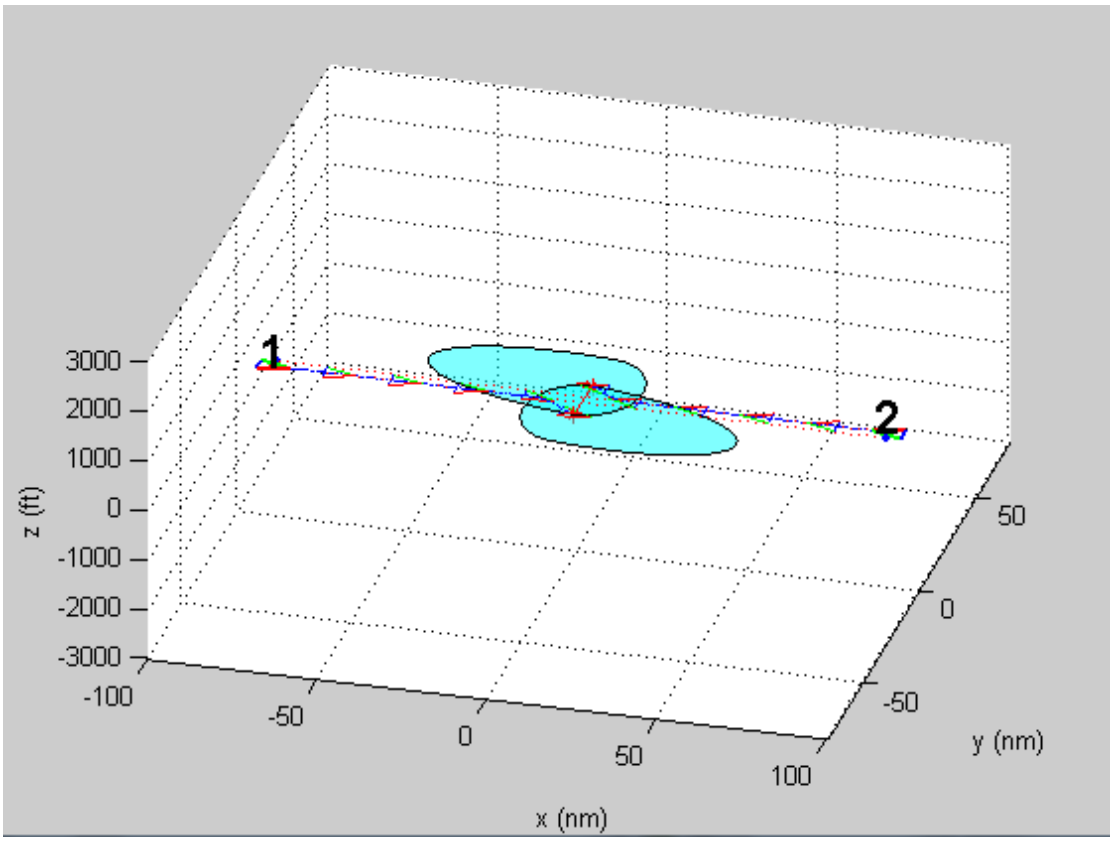
*Σενάριο 1:*Εδώ παρουσιάζουμε δύο αεροσκάφη τα οποία κινούνται σε αντίθετες κατευθύνσεις.

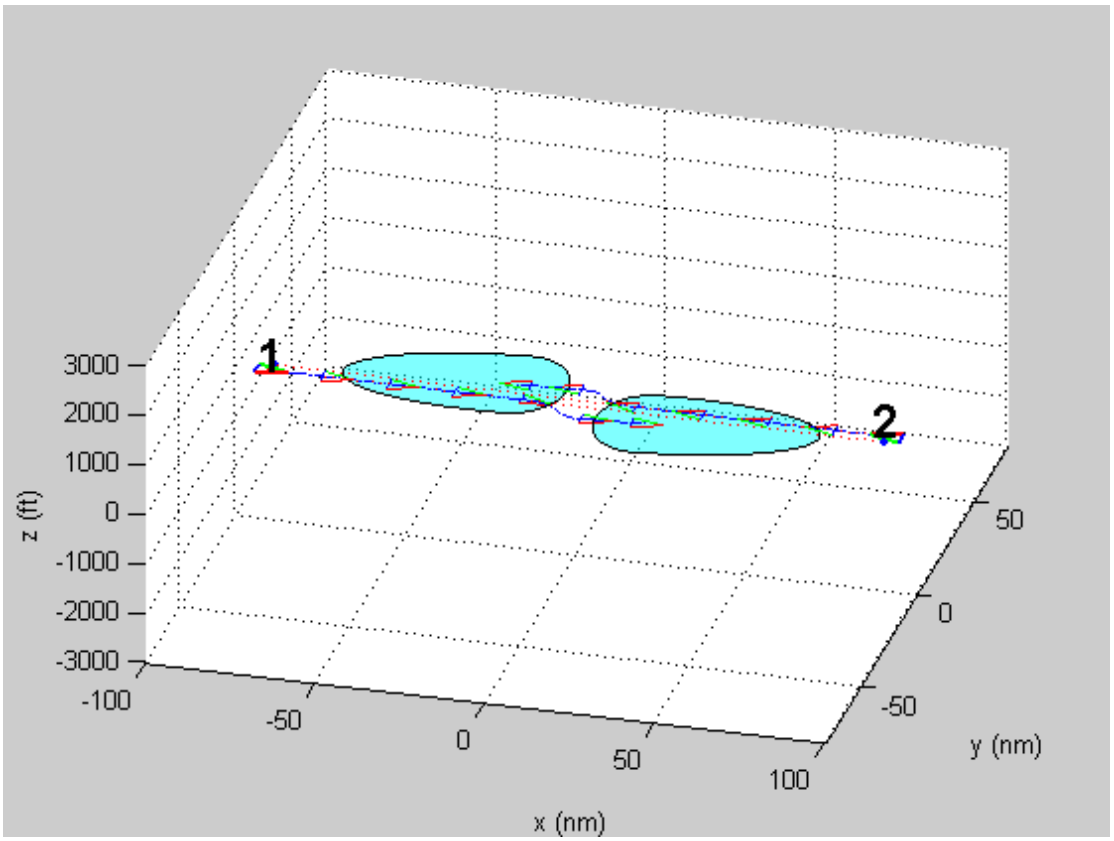
Οι συντεταγμένες των αεροσκαφών είναι

$$(x_{start\,1}, y_{start\,1}) = (-90,0)\,, \qquad (x_{target\,1}, y_{target\,1}) = (90,0)$$
$$(x_{start\,2}, y_{start\,2}) = (90,3)\,, \qquad (x_{target\,2}, y_{target\,2}) = (-90,0)$$


1
2
3000
2000
1000
0
-1000
-2000
-3000
z (ft)
-100
-50
0
50
100
x (nm)
50
0
-50
y (nm)


1
2
3000
2000
1000
0
-1000
-2000
-3000
z (ft)
-100
-50
0
50
100
x (nm)
50
0
-50
y (nm)

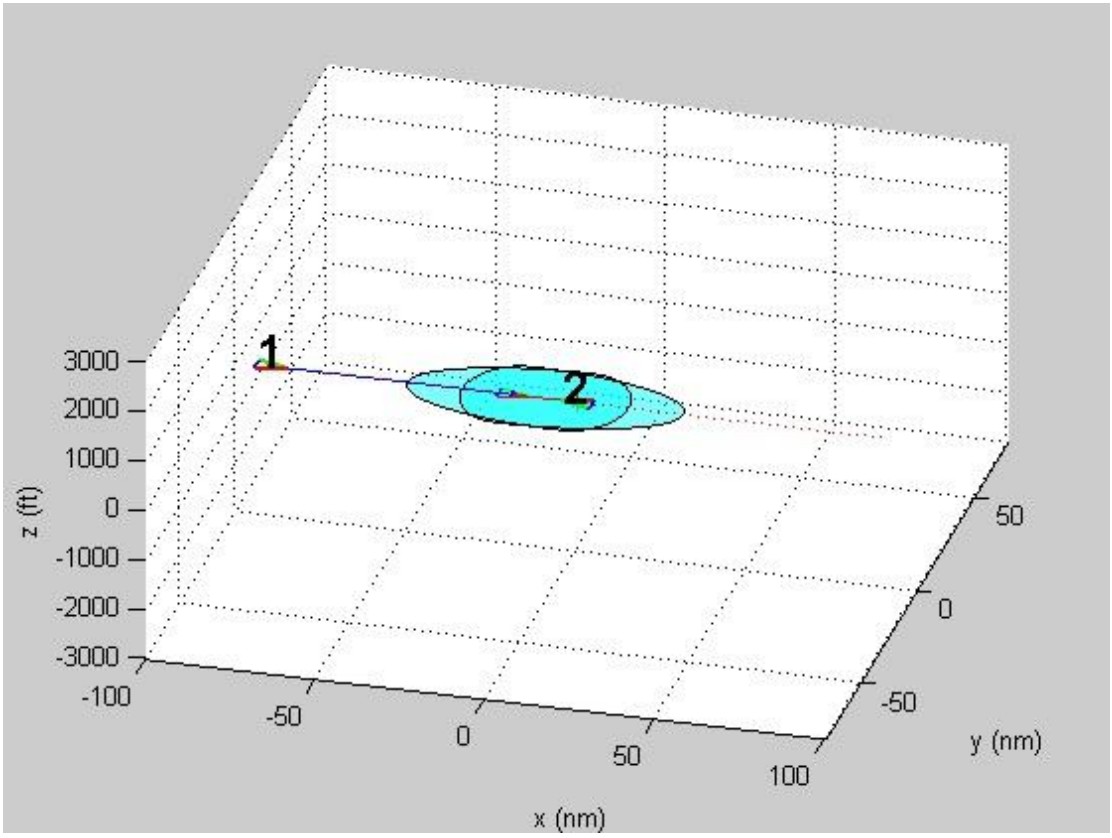




Εικόνες 2.4.1-4

Σημείωση: Στις εικόνες απεικονίζεται ο χώρος που "βλέπουν" οι αισθητήρες. Οι πορείες αρχίζουν και αλλάζουν μόλις το ένα αεροσκάφος εισέλθει στον χώρο του άλλου. Αν δεν υπήρχε sensing range αλλά γνώση όλου του χώρου από το κάθε αεροσκάφος η απόκλιση των αεροσκαφών από τις ευθείες πορείες τους θα συνέβαινε γρηγορότερα(βέβαια αυτό εξαρτάται και από την επιλογή του k).

Σημείωση: Παρατηρούμε ότι τα δύο παραπάνω αεροσκάφη δεν κινούνται πάνω στην ίδια ευθεία αλλά σε δύο παράλληλες ευθείες την y=0 και y=3. Εδώ είναι μία καλή ευκαιρία να αναφέρουμε ένα τυπικό παράδειγμα στο οποίο εμφανίζεται αυτό που οι Rimon και Koditschek χαρακτήρισαν(και αναφέρουμε στο Κεφάλαιο 1) σαν "almost global navigation". Αν θεωρήσουμε το ένα αεροσκάφος ακίνητο στο (0,0) και το άλλο να κινείται πάνω στην y=0 πχ όπως και στο παραπάνω παράδειγμα παρατηρούμε ότι το αεροσκάφος προχωράει μέχρι ένα σημείο και εκεί σταματάει όπως βλέπουμε στην παρακάτω εικόνα.

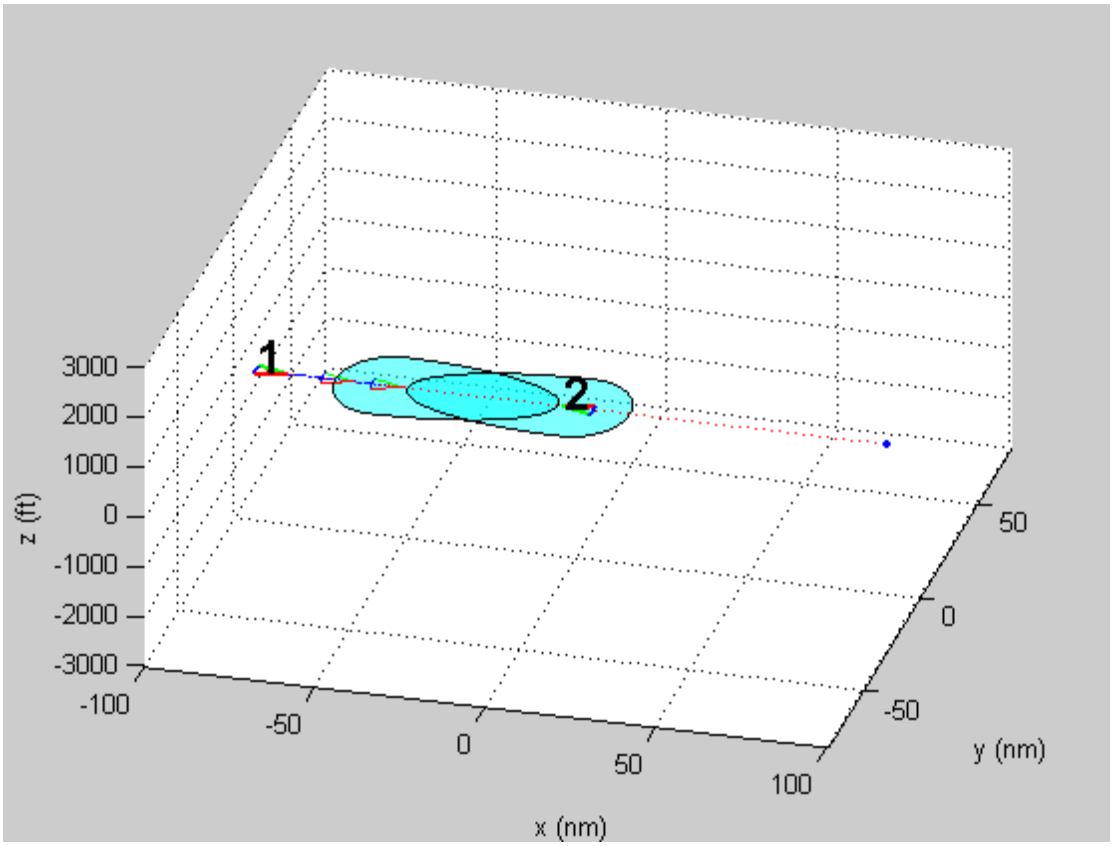


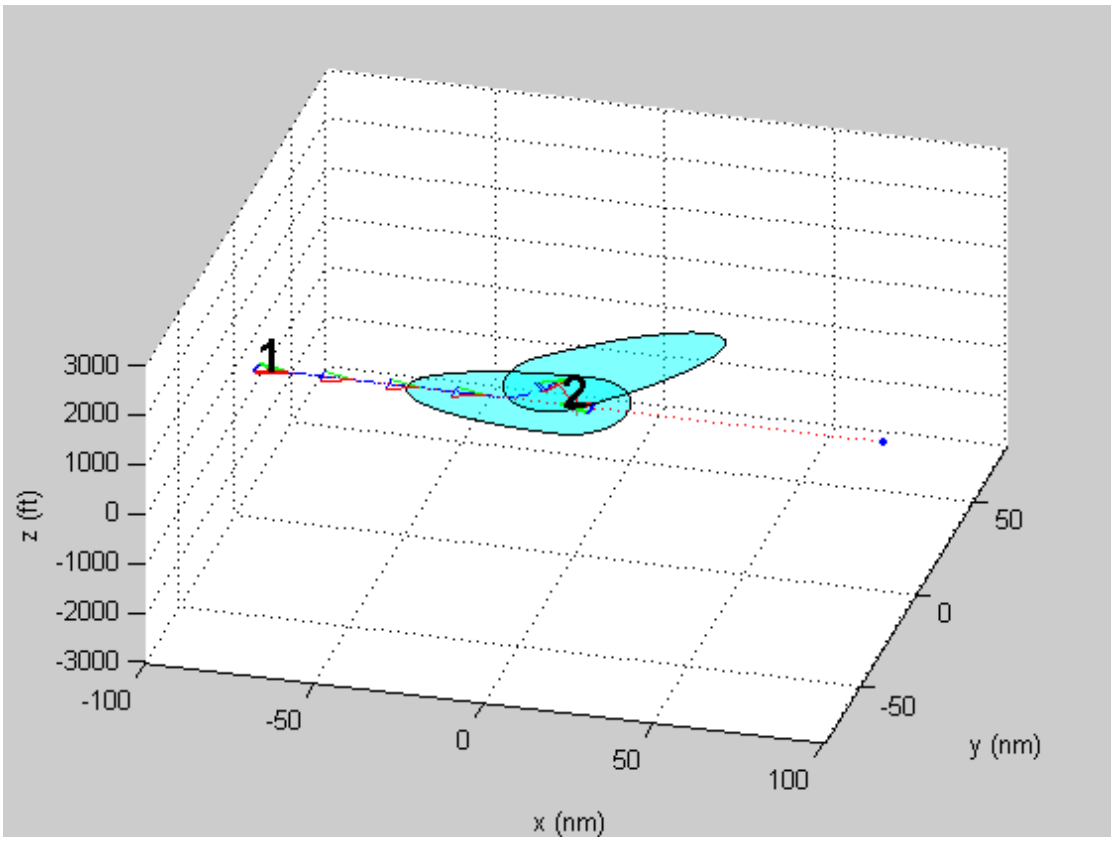
Εικόνες 2.4.5

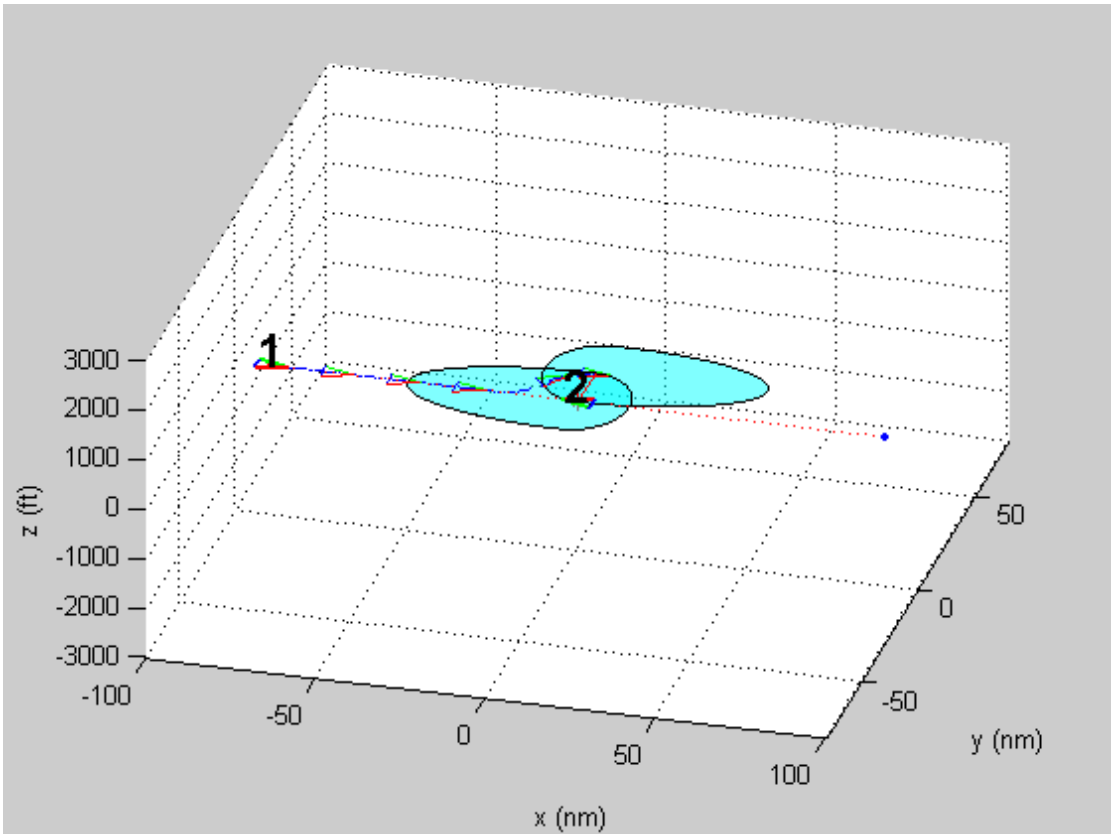
Σε αυτό το σημείο το αεροσκάφος έχει εγκλωβιστεί σε ένα σημείο σέλας(saddle point). Στην πραγματικότητα υπάρχει ένα τοπικό ελάχιστο της συνάρτησης πλοήγησης κατά την κατεύθυνση του άξονα x. Κατά τον άξονα y βρισκόμαστε σε τοπικό μέγιστο. Το σημείο που βρισκόμαστε λοιπόν δεν είναι ένα τοπικό ελάχιστο αλλά ένα σημείο σέλας. Το αεροσκάφος δεν κινείται κατά τον άξονα y γιατί πάνω στον άξονα x έχουμε $\frac{\partial \phi}{\partial y} = 0$. Παρόλα αυτά αν δώσουμε τις εξής συντεταγμένες στο πρόγραμμα

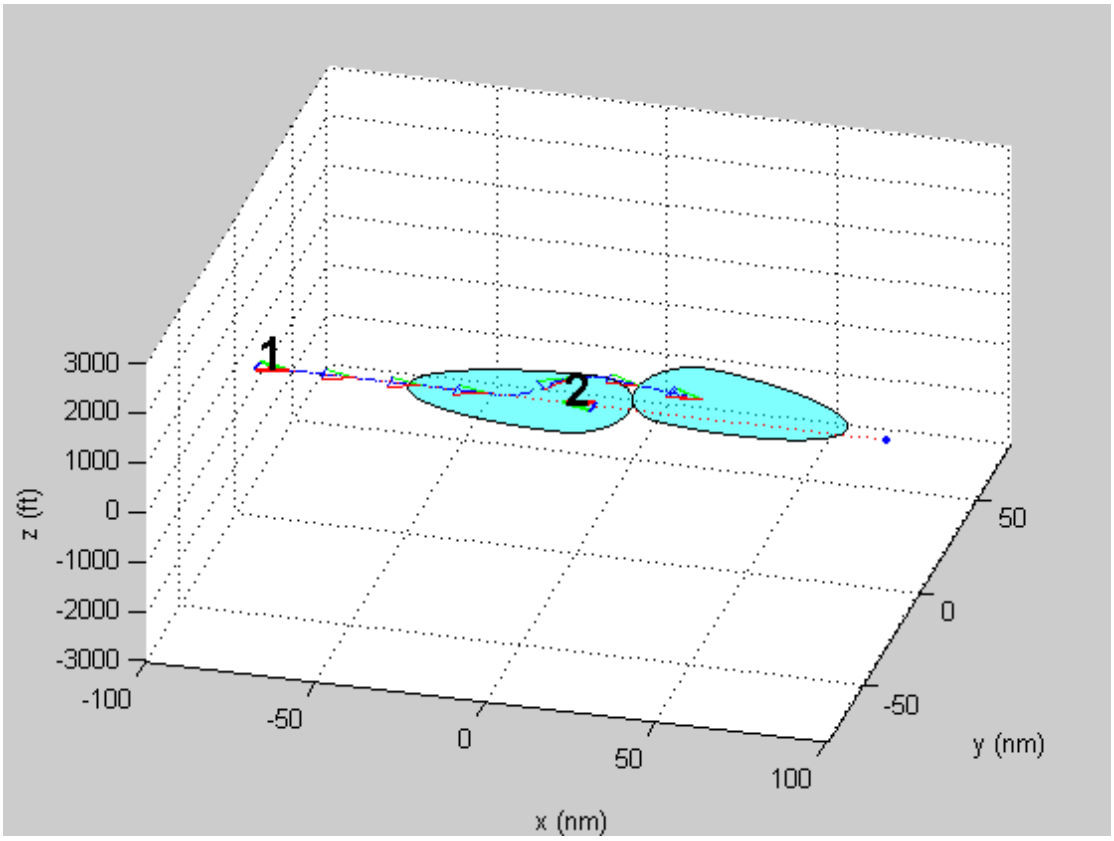
$$(x_{start\,1}, y_{start\,1}) = (-90, 0.001)\,, \qquad \left(x_{target\,1}, y_{target\,1}\right) = (90, 0.001)$$

δηλαδή αφήσουμε το αεροσκάφος να κινηθεί πάνω στην y=0.001 παίρνουμε τα παρακάτω αποτελέσματα




1
2
3000
2000
1000
0
-1000
-2000
-3000
z (ft)
-100
-50
0
50
100
x (nm)
50
0
-50
y (nm)


1
2
3000
2000
1000
0
-1000
-2000
-3000
z (ft)
-100
-50
0
50
100
x (nm)
50
0
-50
y (nm)



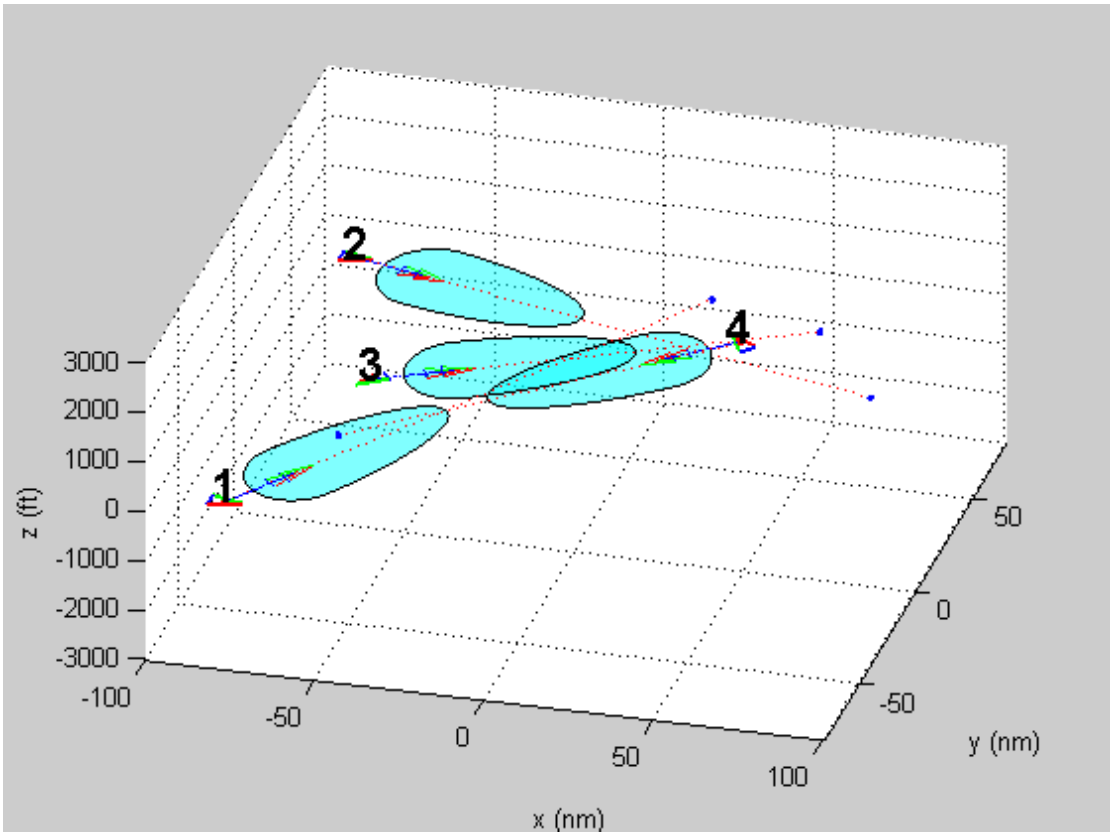
Εικόνες 2.4.6-8

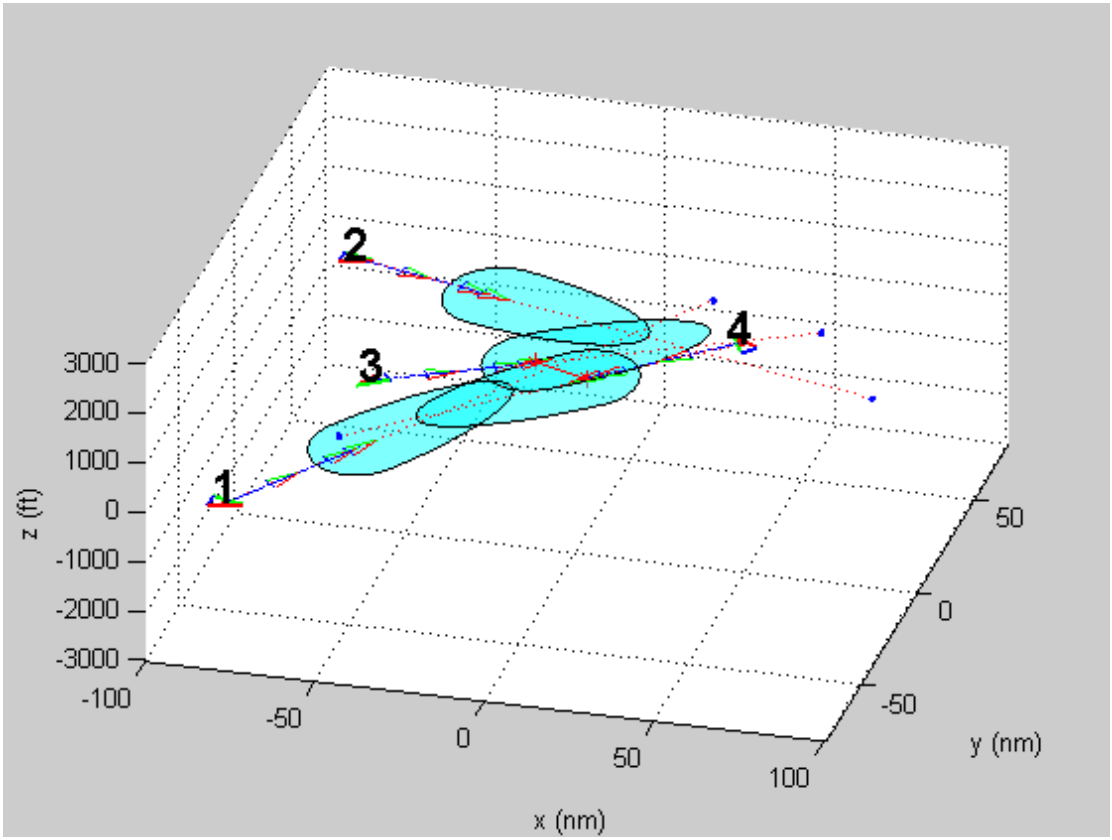
Όπου παραπάνω χρησιμοποιήσαμε το δεύτερο αεροσκάφος σαν ακίνητο εμπόδιο στο (0,0). Έτσι το πρόβλημα του almost global navigation από καθαρά μαθηματική σκοπιά μπορεί να υπάρχει παρόλα αυτά στις εφαρμογές είναι ιδιαίτερα δύσκολο να συμβεί. (Ένα αεροπλάνο δεν μπορεί να κινείται ακριβώς, από μαθηματική σκοπιά, σε μία ευθεία διότι εκτός των άλλων υπάρχουν διαταράξεις θόρυβος κτλ). Παραπάνω βλέπουμε ότι με μία μικρή απόκλιση από το καθαρά συμμετρικό πρόβλημα ο αλγόριθμος τρέχει κανονικά.

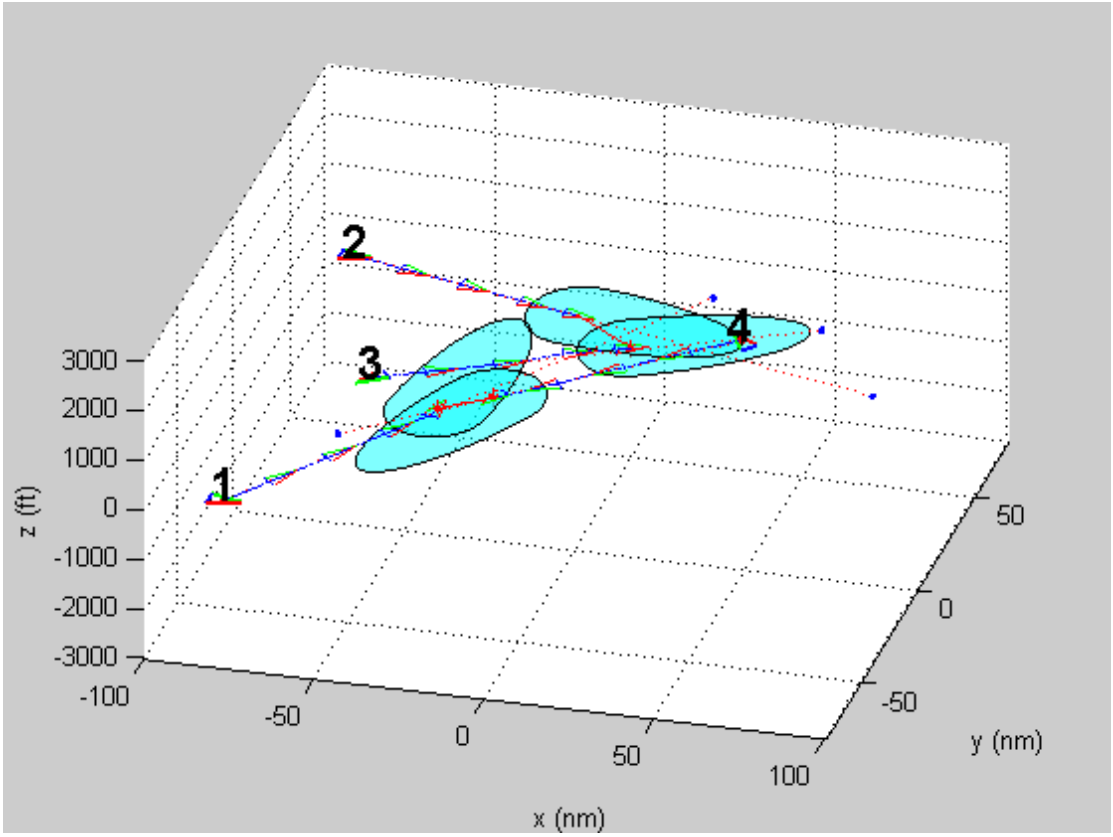
*Σενάριο 2:*Εδώ παρουσιάζουμε τέσσερα αεροσκάφη σε διαφορετικές κατευθύνσεις

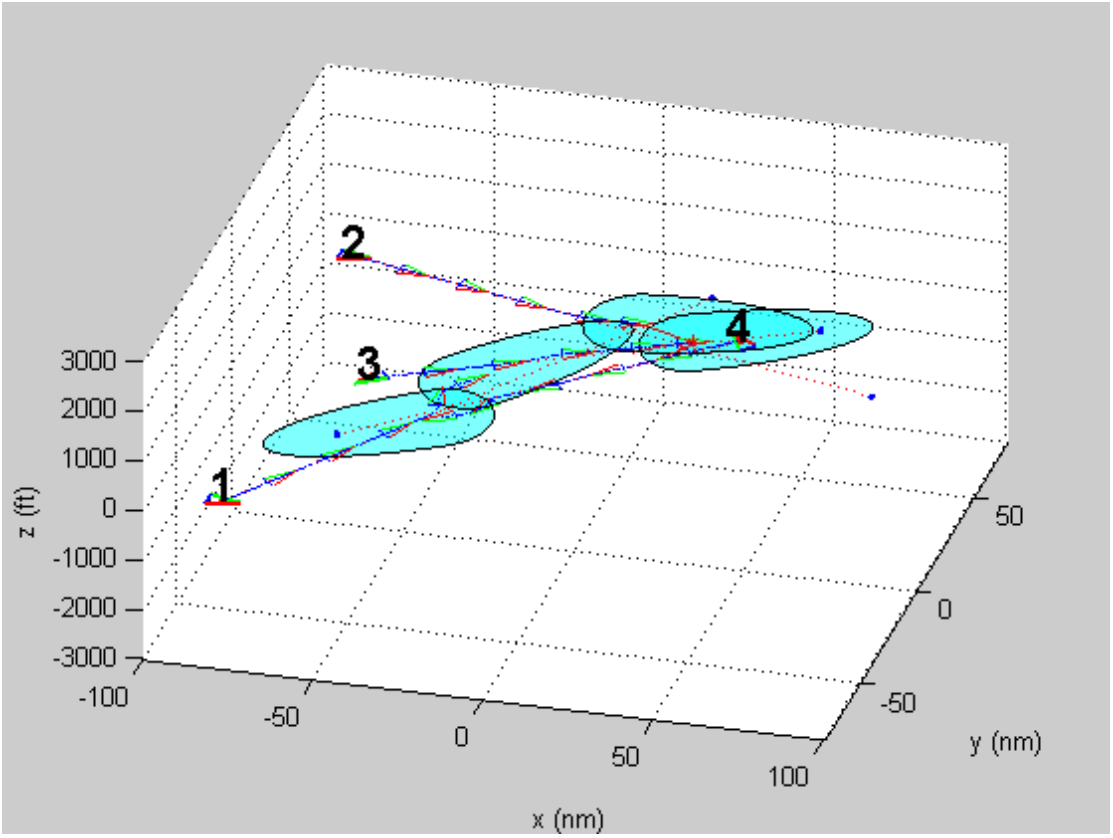
Οι συντεταγμένες των αεροσκαφών είναι

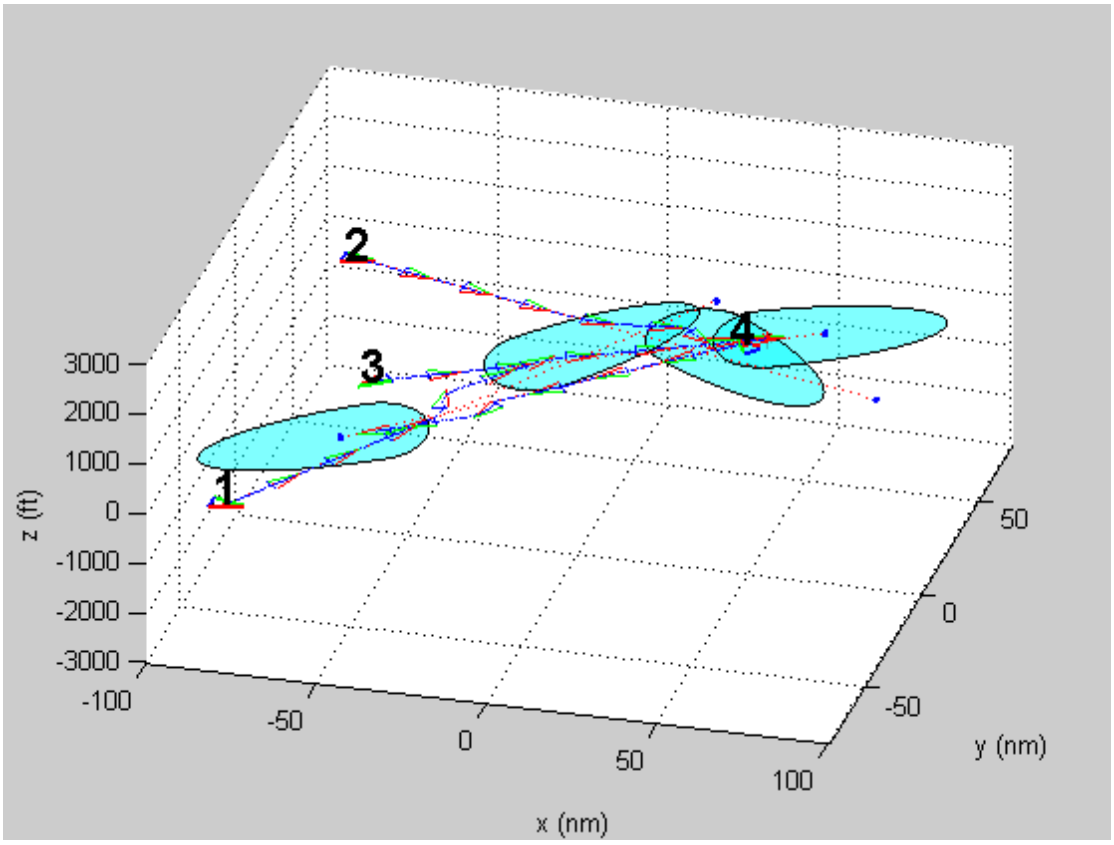
$$(x_{start\,1}, y_{start\,1}) = (-80 - 70)\,, \qquad (x_{target\,1}, y_{target\,1}) = (20,60)$$
$$(x_{start\,2}, y_{start\,2}) = (-85,60)\,, \qquad (x_{target\,2}, y_{target\,2}) = (80,20)$$
$$(x_{start\,3}, y_{start\,3}) = (-60,0)\,, \qquad (x_{target\,3}, y_{target\,3}) = (55,50)$$
$$(x_{start\,4}, y_{start\,4}) = (35,40)\,, \qquad (x_{target\,4}, y_{target\,4}) = (-60,30)$$

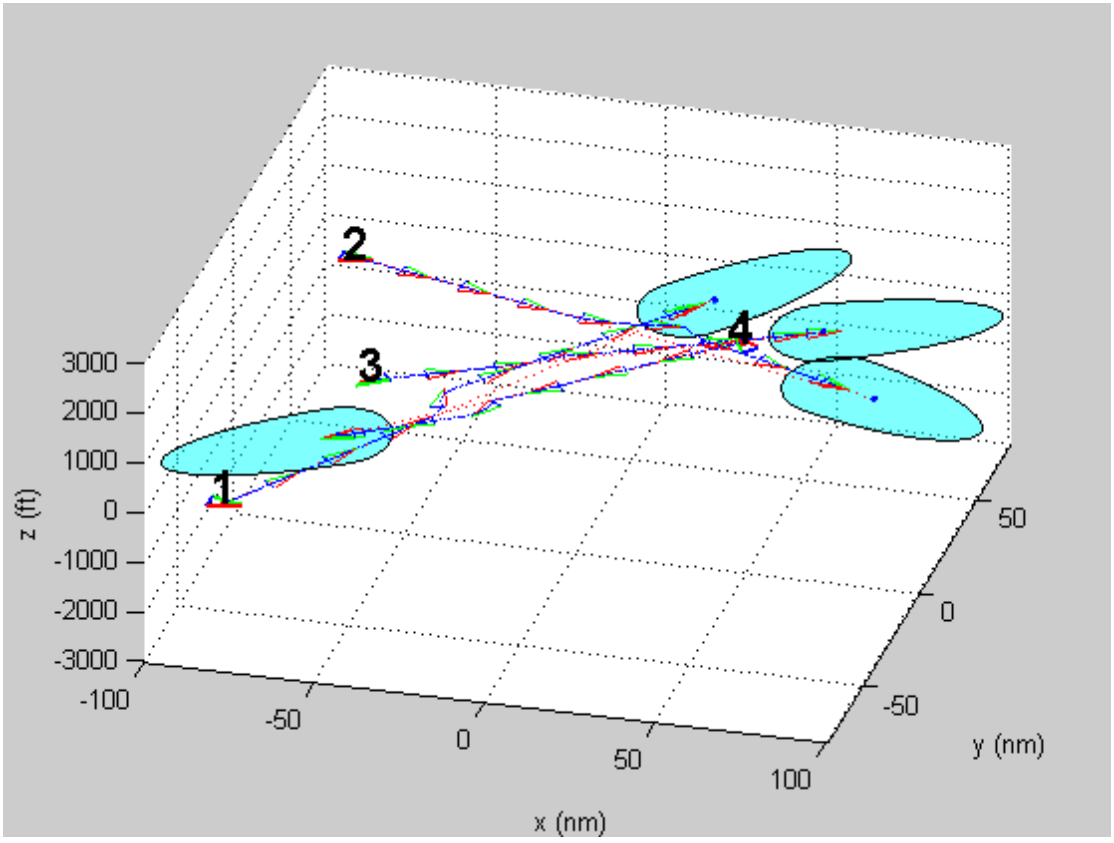

2
3
4
1
3000
2000
1000
0
-1000
-2000
-3000
z (ft)
-100
-50
0
50
100
x (nm)
50
0
-50
y (nm)


2
3
4
1
3000
2000
1000
0
-1000
-2000
-3000
z (ft)
-100
-50
0
50
100
x (nm)
50
0
-50
y (nm)


1
2
3
4
3000
2000
1000
0
-1000
-2000
-3000
z (ft)
-100
-50
0
50
100
x (nm)
50
0
-50
y (nm)


1
2
3
4
3000
2000
1000
0
-1000
-2000
-3000
z (ft)
-100
-50
0
50
100
x (nm)
50
0
-50
y (nm)





Εικόνες 2.4.9-14

2.5 Αποτελέσματα.

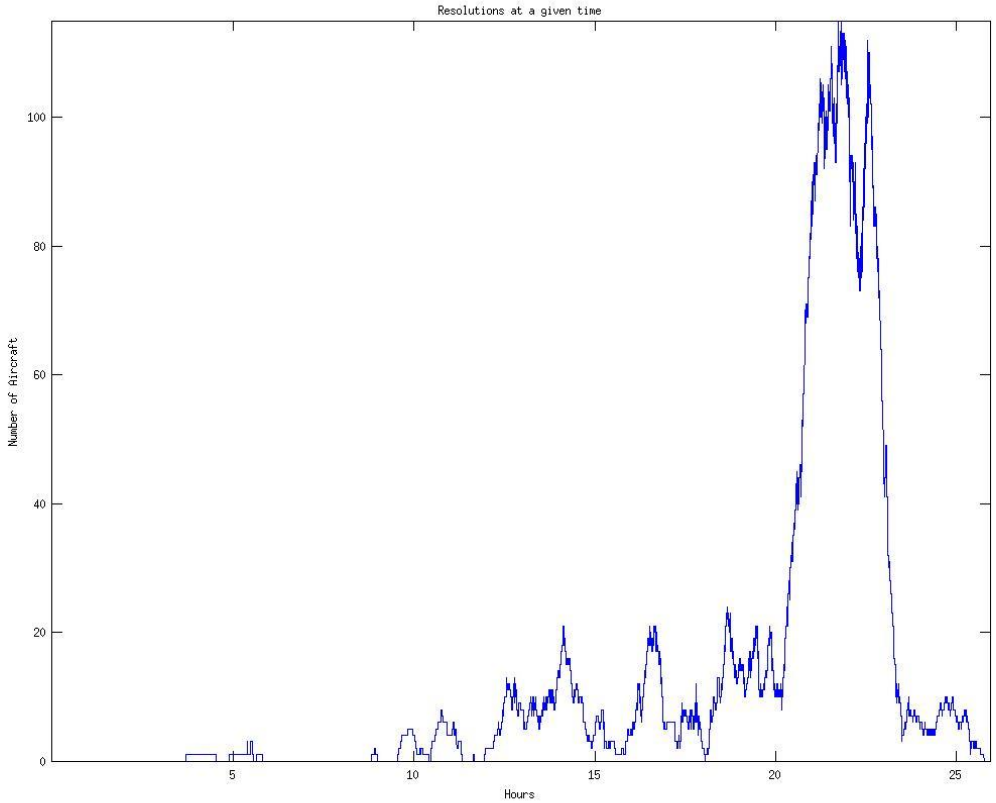
2.5.1 Εισαγωγικά

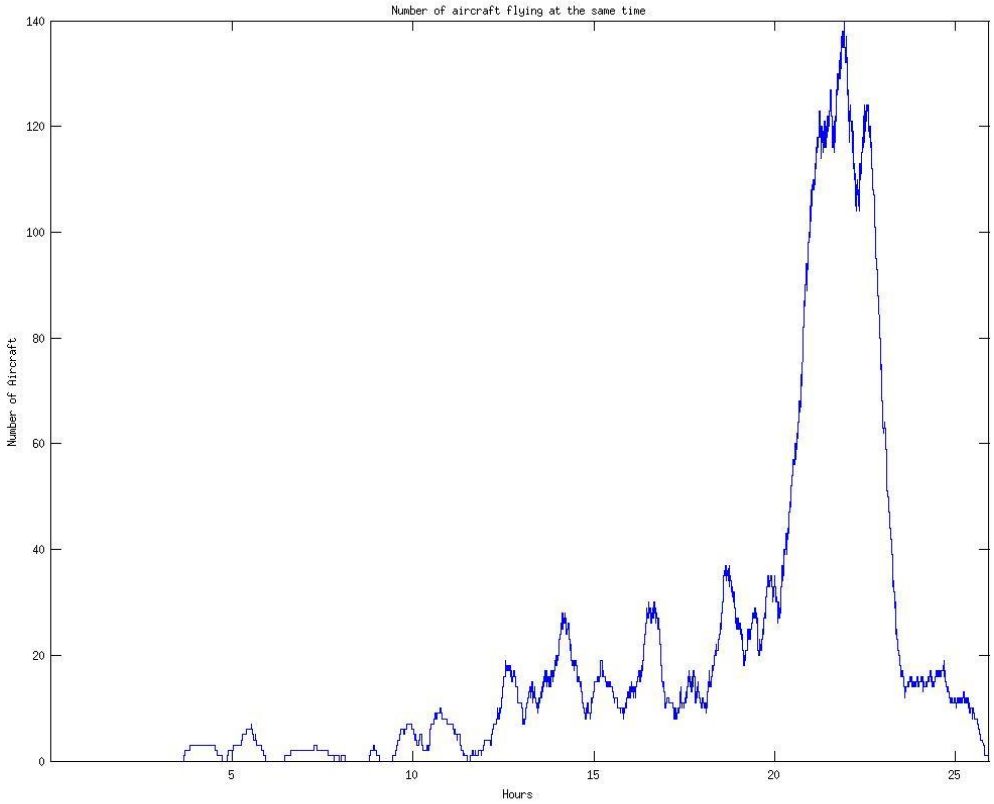
Παρακάτω θα δώσουμε αποτελέσματα από διάφορα cases που τρέξαμε με βάση το δείγμα των αεροσκαφών. Το δείγμα αυτό περιλαμβάνει όπως αναφέραμε πτήσης από δύο ημέρες στην Ευρώπη. Η πρώτη μέρα είναι μία τυπική ημέρα σύμφωνα με τα σημερινά δεδομένα και στην δεύτερη έχουμε ένα παρόμοιο με το προηγούμενο δείγμα στο οποίο έχουν προστεθεί πτήσεις έτσι ώστε να αυξηθεί η εναέρια κυκλοφορία. Οι πτήσεις αυτές δεν έχουν προστεθεί τυχαία αλλά μάλλον κατά τρόπο παρόμοιο με το δείγμα της πρώτης ημέρας απλά προσθέτοντας στις πτήσεις ίδιες πτήσεις κάποια λεπτά πριν και μετά. Στις εξομοιώσεις που ακολουθούν περιγράφουμε αναλυτικά των τρόπο που επιλέχθηκαν τα δεδομένα και παρουσιάζουμε διαγράμματα και δεδομένα τα οποία μας βοηθούν να έχουμε μία εικόνα τις εναέρια κυκλοφορίας. Στην αρχή τρέχουμε το πρόγραμμα εισάγοντας του ποια αεροσκάφη θέλουμε να εξομοιώσουμε. Εξομοιώνει τις τροχιές όλων των αεροσκαφών. Το βήμα είναι στο 1 sec. Στην έξοδο του μας δίνει ένα αρχείο το οποίο περιέχει τους χρόνους απογείωσης – προσγείωσης, διάφορα δεδομένα που χρειάζονται στο επόμενο βήμα και επίσης τις τροχιές των αεροσκαφών. Οι τροχιές αυτές είναι οι διαδρομές που ακολουθήσανε τα αεροσκάφη κατά την πτήση τους χρησιμοποιώντας την μέθοδο. Στο επόμενο βήμα τρέχουμε τα αρχεία dataproc.m και makeresults.m τα οποίο μας δίνουν δεδομένα μετά από επεξεργασία των αποτελεσμάτων της εξομοίωσης όπως τον αριθμό των resolutions που έγιναν, δηλαδή τον αριθμό των αεροσκαφών που κάποιο αεροσκάφος συνάντησε, τον χρόνο που βρέθηκε το κάθε αεροσκάφος σε resolution, πότε ολοκληρώθηκε η πτήση του, τις ελάχιστες αποστάσεις που το συγκεκριμένο αεροσκάφος είχε και με ποια από αυτά, καθώς και διάφορα στοιχεία γενικά για την εξομοίωση.

2.5.2 Εξομοιώσεις

2.5.2.1Σενάριο των 1000 πρώτων αεροσκαφών

Εδώ θα κάνουμε μία εξομοίωση των 1000 πρώτων αεροσκαφών. Τα αεροσκάφη αυτά επιλέγονται από την αρχή και είναι μία εξομοίωση της πρώτης ημέρας. Είναι πολύ κοντά στα σημερινά δεδομένα. Τα πρώτα αεροσκάφη ξεκινάνε από τα μεσάνυχτα και τα τελευταία φτάνουν στην έξοδο από τον χώρο που ελέγχουμε περίπου μία ημέρα μετά. Σε αυτήν την εξομοίωση όπως και στις υπόλοιπες επιλέγεται τα αεροσκάφη να μπορούν να κινηθούν στις δύο διαστάσεις, παρόλο που ο κώδικας μπορεί να τρέξει και την τρίτη (κάθετες μανούβρες). Παρόλα αυτά επιλέξαμε να είναι στις δύο ,κάτι που είναι πιο δύσκολο αφού ο χώρος για μανούβρες των αεροσκαφών είναι πολύ μικρότερος, για να ελέγξουμε πως συμπεριφέρονται τα αεροσκάφη σε σενάρια που δεν επιτρέπεται να αλλάξουν ύψος(είτε λόγω έλλειψης ισχύος στους κινητήρες, καιρικών φαινομένων αλλά κυρίως για λόγους απόδοσης/κατανάλωσης καυσίμου οπότε και το αεροσκάφος αποφεύγει να αλλάξει το ύψος πτήσης του από το προκαθορισμένο). Αρχικά παίρνουμε τα παρακάτω διαγράμματα.





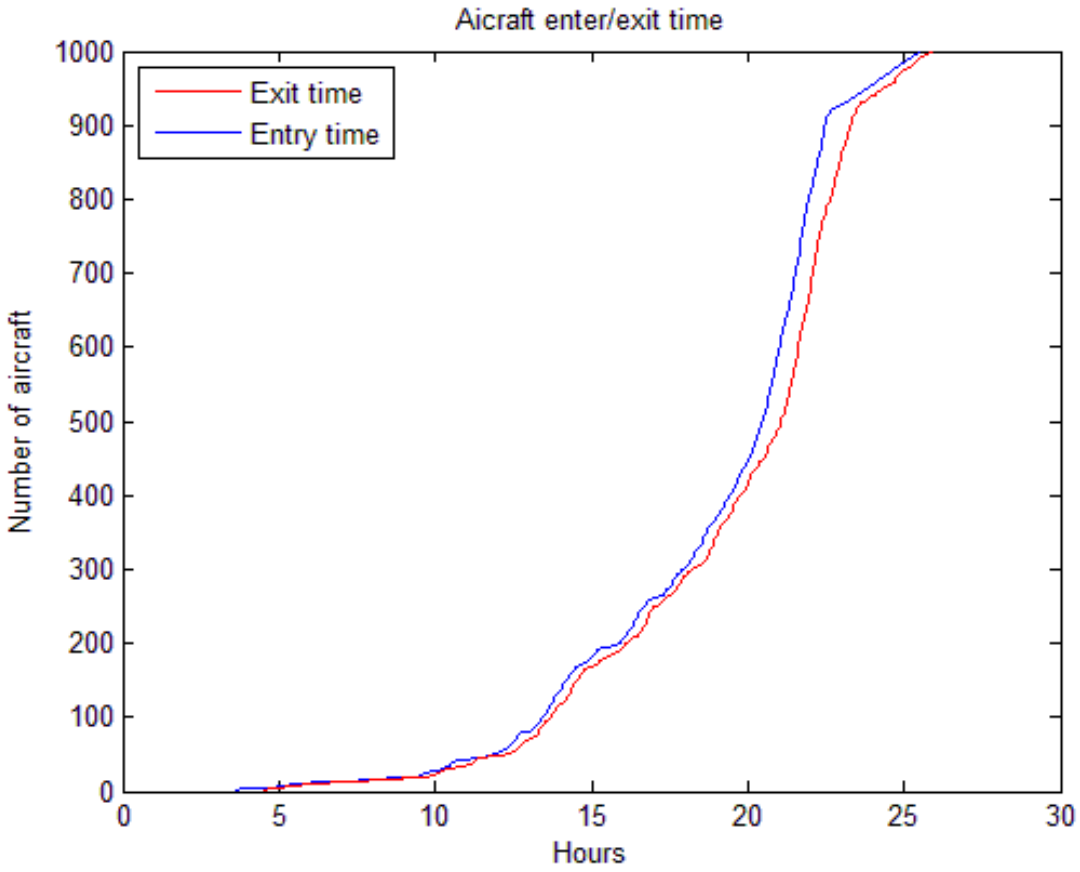
Εικόνες 2.5.2.1.1-2

Τα παραπάνω δύο διαγράμματα δίνουν τα εξής στοιχεία για την εξομοίωση. Το πρώτο δείχνει τον αριθμό των resolutions που συμβαίνουν κάθε χρονική στιγμή στο εναέριο χώρο μας στην περίοδο που έκαναν να πετάξουν όλα τα αεροσκάφη(που είναι περίπου 26 ώρες). Δηλαδή κάθε χρονική στιγμή από τα αεροσκάφη που βρίσκονταν σε πτήση αυτά που δεν ακολουθούσαν την ευθεία πορεία που ενώνει την θέση τους με τον στόχο αλλά προσπαθούσαν να αποφύγουν μία πιθανή σύγκρουση. Το δεύτερο από αυτά είναι ένα παρόμοιο διάγραμμα το οποίο μας δείχνει κάθε χρονική στιγμή πόσα αεροσκάφη βρίσκονταν σε πτήση μέσα στον εναέριο χώρο μας.

Στην αρχή των δύο διαγραμμάτων βλέπουμε ότι δεν υπάρχει δραστηριότητα αλλά αυτή αρχίζει περίπου 3-4 ώρες μετά. Αυτό συμβαίνει γιατί πολλά από τα αεροσκάφη εκκινούν από μέρη μακρινά από τον χώρο της Κεντρικής Ευρώπης και φτάνουν σε αυτή κάποια χρονική στιγμή μετά την απογείωσή τους. Στις εξομοιώσεις μετράμε τον χρόνο από την στιγμή που θα απογειωθεί το πρώτο αεροσκάφος αλλά λαμβάνουμε υπόψη την επίδρασή του την στιγμή που αυτό θα μπει στον εναέριο χώρο μας. (Οι χρόνοι αυτοί υπολογίστηκαν στην αρχή για όλο το δείγμα όπως αναφέραμε νωρίτερα). Παρατηρούμε ότι στην αρχή δεν υπάρχει μεγάλος “συνωστισμός” στον χώρο αλλά όσο περνάει η ώρα αυτός αυξάνει. Τα αεροσκάφη εκκινούν κάποια χρονική στιγμή και η επίδρασή τους στον χώρο φαίνεται λίγο αργότερα αφού δεν

είναι αναγκαστικό να ξεκινήσουν μέσα από τον χώρο αλλά από κάπου εκτός αυτού και να μπουν αργότερα. Παρατηρούμε ότι η κίνηση αρχίζει να αυξάνει γύρω στο μεσημέρι της επόμενης ημέρας και φτάνει στα μέγιστα το ίδιο απόγευμα προς βράδυ.

Παρακάτω δίνουμε ένα διάγραμμα που δείχνει σε κάθε χρονική στιγμή πόσοι έχουν απογειωθεί και πόσοι έχουν προσγειωθεί. (Να αναφέρουμε ότι τους όρους απογείωση και προσγείωση τους αναφέρουμε πιο γενικά εννοώντας τον χρόνο που το αεροσκάφος εισήλθε στον εναέριο χώρο και τον χρόνο που εξήλθε από αυτόν).

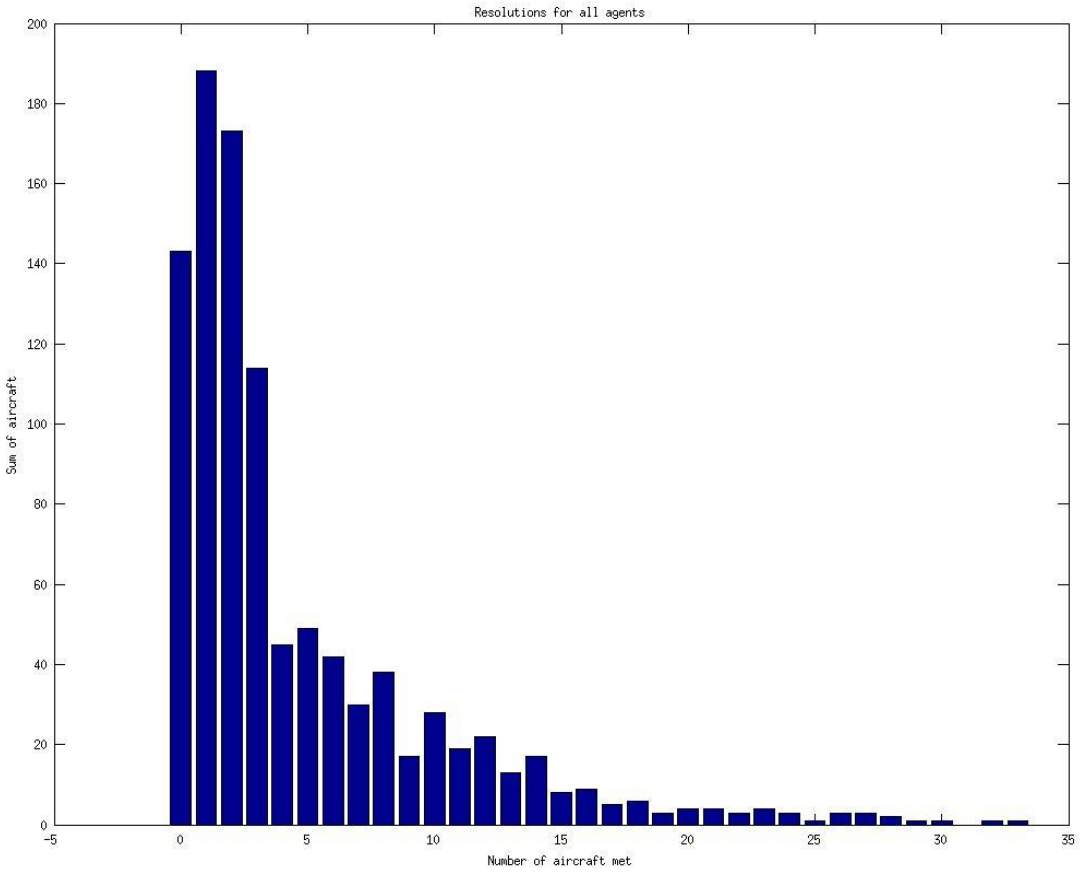


Εικόνα 2.5.2.1.3

Εδώ βλέπουμε την χρονική στιγμή απογείωσης και προσγείωσης των αεροσκαφών. Παρατηρούμε ότι η καμπύλη απογείωσης είναι παρόμοια με αυτήν της προσγείωσης με την καμπύλη προσγείωσης να έρχεται βεβαίως λίγο πιο αργά. Από την κλίση της καμπύλης απογείωσης μπορούμε να συμπεράνουμε ότι ο ρυθμός απογείωσης των αεροσκαφών αυξάνει κατά την διάρκεια της ημέρας συνεχώς και αρχίζει να μειώνεται μετά τις 21 περίπου ώρες δηλαδή γύρω στα μεσάνυχτα της επόμενης ημέρας. Επίσης παρατηρούμε ότι η κλίση της καμπύλης προσγείωσης είναι παρόμοια με αυτή της

καμπύλης απογείωσης δηλαδή ο ρυθμός απογείωσης των αεροσκαφών είναι παρόμοιος με τον ρυθμό προσγείωσης τους.

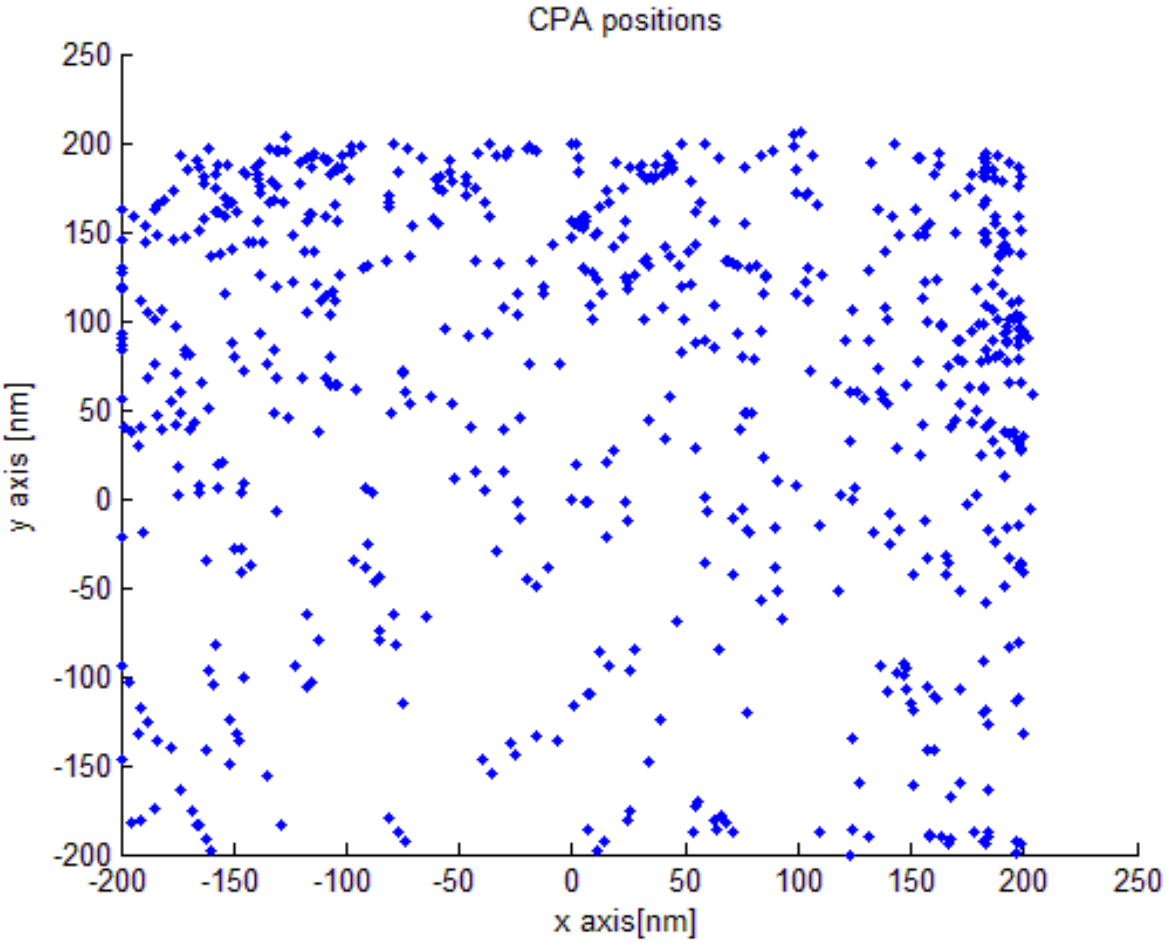
Τέλος δίνουμε το παρακάτω διάγραμμα



Εικόνα 2.5.2.1.4

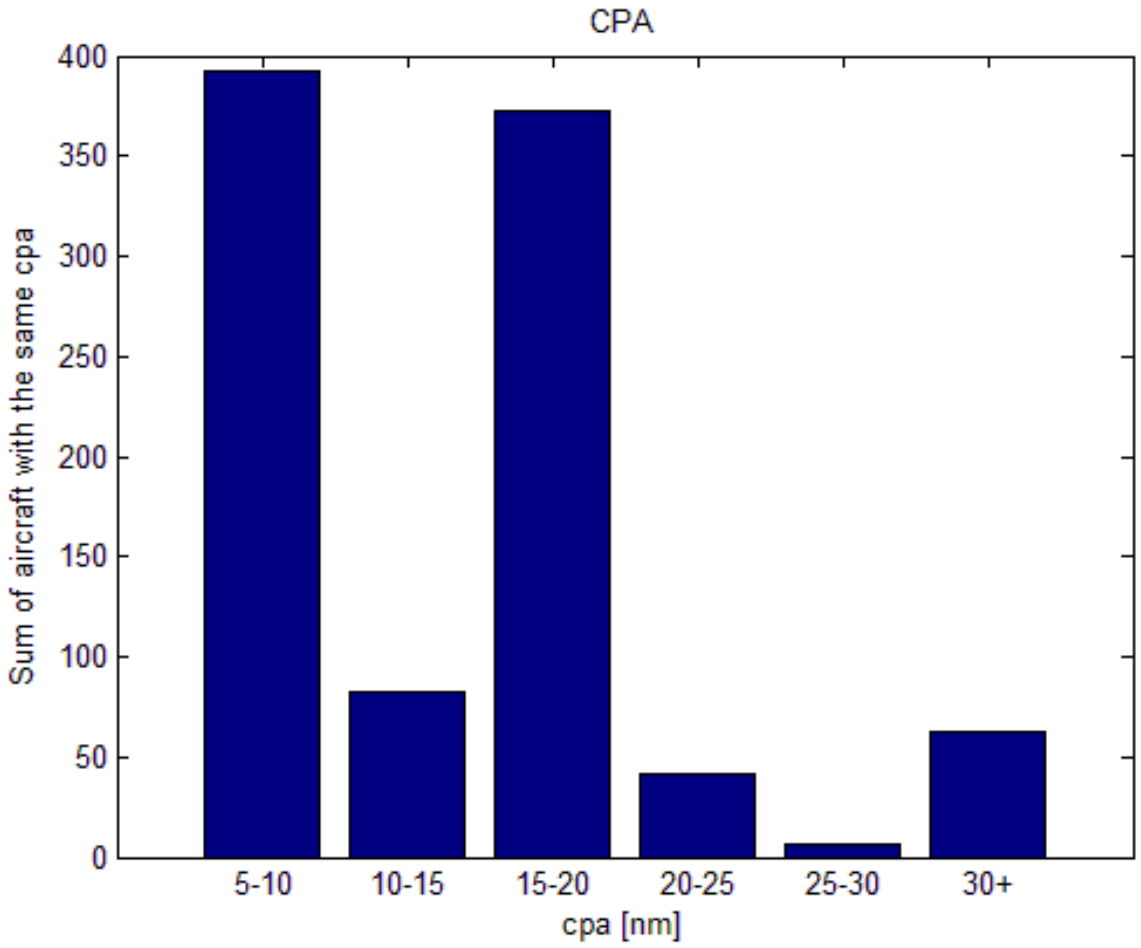
Εδώ ομαδοποιούμε τα αεροσκάφη σε σχέση με το πόσα αεροσκάφη συνάντησαν στον δρόμο τους οπότε και αναγκάστηκαν να αλλάξουν πορεία. Στον οριζόντιο άξονα έχουμε τον αριθμό τον αεροσκαφών που βρήκε ένα αεροσκάφος στον δρόμο του και στον κάθετο τον αριθμό των αεροσκαφών. Παρατηρούμε ότι τα περισσότερα αεροσκάφη συνάντησαν τουλάχιστον ένα ακόμα στον δρόμο τους και κάποια συνάντησαν περισσότερους από τριάντα.

Μπορούμε επίσης να δώσουμε διαγράμματα σχετικά με το cpa (closest point of approach) του κάθε αεροσκάφους δηλαδή πόσο ήταν η ελάχιστη απόσταση που βρέθηκε ένα αεροσκάφος σε σχέση με ένα άλλο, με ποιο αεροσκάφος και σε ποιο σημείο.



Εικόνα 2.5.2.1.5

Εδώ έχουμε ένα διάγραμμα που δείχνει όλα τα σημεία στα οποία το καθένα από τα 1000 αεροσκάφη βρέθηκαν σε ελάχιστη απόσταση με κάποιο άλλο αεροσκάφος σε όλη την διάρκεια της πτήσης τους. Παρατηρούμε ότι υπάρχουν αρκετά σημεία και στα σημεία εξόδου αφού πολλά αεροσκάφη βγαίνουν από το ίδιο η κοντινά σημεία αφού αρκετά πηγαίνουν προς τα ίδια αεροδρόμια.



Εικόνα 2.5.2.1.5

Εδώ έχουμε ένα διάγραμμα που ομαδοποιεί τα αεροσκάφη ανάλογα με το cpa. Ο άξονας των x μας δίνει ένα εύρος ελάχιστων αποστάσεων και ο άξονας τον y μας δίνει τον αριθμό τον αεροσκαφών που ανήκουν σε αυτή την κλάση.

Τέλος έχουμε και ένα αρχείο results.txt το οποίο μας δίνει αναλυτικά διάφορα στοιχεία για το κάθε αεροσκάφος. Παρακάτω παρουσιάζουμε τμήμα αυτού του αρχείου για την παραπάνω εξομοίωση.

SIMULATION RESULTS

Total number of agents: 1000

Agents Travelled(mean value): 1.154335 % more of their line route distance
Flight time: 1783143.000000 seconds or 495.317500 hours
Flight time(Mean Value): 1783.143000 seconds or 0.495318 hours
Total time in resolution(all agents): 1300341.000000 seconds or 361.205833 hours
Mean time in resolution: 1300.341000 seconds or 0.361206 hours
Mean value of resolutions for each agent: 4.652000
Average % time in resolution: 72.924101 %
Execution time: 5460.670337 seconds
Flight time/execution time: 326.542877

AGENT: 1

Start: 200.000000 , 60.473997 , 0.000000
Target: -200.000000 , 98.824198 , 0.000000
Total distance travelled: 399.898333 [nm]
Line distance: 399.898333 [nm]
Travelled: 0.000000 % more of their line route distance
Flight time: 3173.000000 seconds or 0.881389 hours
Maximum distance was 83.233333 [nm] with agent: 3
Minimum distance was 22.573889 [nm] with agent: 2
Total time in resolution: 0.000000 seconds or 0.000000 hours
Agent: 1 met 0 agent(s)

AGENT: 2

Start: 200.000000 , 60.473997 , 0.000000
Target: -200.000000 , 98.824198 , 0.000000
Total distance travelled: 399.898333 [nm]
Line distance: 399.898333 [nm]
Travelled: 0.000000 % more of their line route distance
Flight time: 3173.000000 seconds or 0.881389 hours
Maximum distance was 60.533333 [nm] with agent: 3
Minimum distance was 22.573889 [nm] with agent: 1
Total time in resolution: 2992.000000 seconds or 0.831111 hours
Agent: 2 met 1 agent(s)
Agent: 1 at 3.705278 hours for 0.831111 hours

AGENT: 3

Start: 200.000000 , 60.473997 , 0.000000
Target: -200.000000 , 98.824198 , 0.000000
Total distance travelled: 399.898333 [nm]
Line distance: 399.898333 [nm]
Travelled: 0.000000 % more of their line route distance
Flight time: 3173.000000 seconds or 0.881389 hours

```
Maximum distance was 83.233333 [nm] with agent: 1
Minimum distance was 60.407222 [nm] with agent: 2
Total time in resolution: 0.000000 seconds or 0.000000 hours
Agent: 3 met 0 agent(s)
```

AGENT: 4

```
Start: -153.488745 , 200.000000 , 0.000000
Target: -145.190946 , -100.759136 , 0.000000
Total distance travelled: 298.969444 [nm]
Line distance: 298.969444 [nm]
Travelled: 0.000000 % more of their line route distance
Flight time: 2505.000000 seconds or 0.695833 hours
Maximum distance was 324.195299 [nm] with agent: 6
Minimum distance was 12.366771 [nm] with agent: 7
Total time in resolution: 477.000000 seconds or 0.132500 hours
Agent: 4 met 2 agent(s)
Agent: 6 at 4.858333 hours for 0.013056 hours
Agent: 7 at 4.945278 hours for 0.045556 hours
```

AGENT: 5

```
Start: -153.488745 , 200.000000 , 0.000000
Target: -145.190946 , -100.759136 , 0.000000
Total distance travelled: 298.969444 [nm]
Line distance: 298.969444 [nm]
Travelled: 0.000000 % more of their line route distance
Flight time: 2505.000000 seconds or 0.695833 hours
Maximum distance was 340.078847 [nm] with agent: 6
Minimum distance was 12.808756 [nm] with agent: 7
Total time in resolution: 2371.000000 seconds or 0.658611 hours
Agent: 5 met 2 agent(s)
Agent: 4 at 4.895278 hours for 0.658611 hours
Agent: 7 at 5.496944 hours for 0.042500 hours
```

AGENT: 6

```
Start: -148.419666 , -200.000000 , 0.000000
Target: -171.614743 , 83.726964 , 0.000000
Total distance travelled: 282.741111 [nm]
Line distance: 282.741111 [nm]
Travelled: 0.000000 % more of their line route distance
Flight time: 2244.000000 seconds or 0.623333 hours
Maximum distance was 340.078847 [nm] with agent: 5
Minimum distance was 14.948624 [nm] with agent: 4
Total time in resolution: 13.000000 seconds or 0.003611 hours
Agent: 6 met 1 agent(s)
Agent: 4 at 5.034722 hours for 0.003611 hours
```

(…)

AGENT: 453

Start: -200.000000 , 92.576070 , 0.000000
Target: 200.000000 , -15.324077 , 0.000000
Total distance travelled: 412.383333 [nm]
Line distance: 412.383333 [nm]
Travelled: 0.000000 % more of their line route distance
Flight time: 3272.000000 seconds or 0.908889 hours
Maximum distance was 440.319533 [nm] with agent: 640
Minimum distance was 15.907323 [nm] with agent: 455
Total time in resolution: 0.000000 seconds or 0.000000 hours
Agent: 453 met 0 agent(s)

AGENT: 454

Start: 7.343776 , -109.691817 , 0.000000
Target: 200.000000 , -150.320245 , 0.000000
Total distance travelled: 204.987500 [nm]
Line distance: 195.151319 [nm]
Travelled: 5.040284 % more of their line route distance
Flight time: 1589.000000 seconds or 0.441389 hours
Maximum distance was 487.406033 [nm] with agent: 481
Minimum distance was 5.577820 [nm] with agent: 427
Total time in resolution: 448.000000 seconds or 0.124444 hours
Agent: 454 met 1 agent(s)
Agent: 427 at 20.053056 hours for 0.124444 hours

AGENT: 455

Start: -200.000000 , 85.822012 , 0.000000
Target: 200.000000 , -97.259758 , 0.000000
Total distance travelled: 437.983889 [nm]
Line distance: 437.983888 [nm]
Travelled: 0.000000 % more of their line route distance
Flight time: 3475.000000 seconds or 0.965278 hours
Maximum distance was 467.204767 [nm] with agent: 641
Minimum distance was 15.907323 [nm] with agent: 453
Total time in resolution: 273.000000 seconds or 0.075833 hours
Agent: 455 met 1 agent(s)
Agent: 453 at 20.035000 hours for 0.075833 hours

AGENT: 456

Start: 7.343776 , -109.691817 , 0.000000
Target: -105.404057 , -200.000000 , 0.000000
Total distance travelled: 142.463333 [nm]
Line distance: 142.460332 [nm]
Travelled: 0.002107 % more of their line route distance
Flight time: 1084.000000 seconds or 0.301111 hours
Maximum distance was 436.528464 [nm] with agent: 449
Minimum distance was 7.826490 [nm] with agent: 421
Total time in resolution: 839.000000 seconds or 0.233056 hours
Agent: 456 met 2 agent(s)

```
Agent: 421 at 20.140000 hours for 0.059444 hours
Agent: 452 at 20.087222 hours for 0.233056 hours

                              (…)

                          AGENT: 979

Start: -200.000000 , 184.395105 , 0.000000
Target: 200.000000 , -40.906137 , 0.000000
Total distance travelled: 460.936111 [nm]
Line distance: 457.877766 [nm]
Travelled: 0.667939 % more of their line route distance
Flight time: 3657.000000 seconds or 1.015833 hours
Maximum distance was 504.555555 [nm] with agent: 969
Minimum distance was 5.224641 [nm] with agent: 985
Total time in resolution: 3655.000000 seconds or 1.015278 hours
Agent: 979 met 10 agent(s)
Agent: 549 at 22.936389 hours for 0.084722 hours
Agent: 927 at 22.820278 hours for 0.048333 hours
Agent: 961 at 22.888333 hours for 0.031111 hours
Agent: 962 at 22.896944 hours for 0.041667 hours
Agent: 964 at 22.916111 hours for 0.039722 hours
Agent: 965 at 22.444444 hours for 0.511111 hours
Agent: 981 at 22.950278 hours for 0.024167 hours
Agent: 985 at 22.777778 hours for 0.681944 hours
Agent: 986 at 22.897500 hours for 0.562222 hours
Agent: 999 at 22.947222 hours for 0.008333 hours



                          AGENT: 980

Start: 30.744933 , -120.390519 , 0.000000
Target: -118.196982 , 193.083882 , 0.000000
Total distance travelled: 345.166111 [nm]
Line distance: 345.166050 [nm]
Travelled: 0.000018 % more of their line route distance
Flight time: 2739.000000 seconds or 0.760833 hours
Maximum distance was 446.003101 [nm] with agent: 823
Minimum distance was 6.166300 [nm] with agent: 955
Total time in resolution: 2737.000000 seconds or 0.760278 hours
Agent: 980 met 8 agent(s)
Agent: 909 at 22.733611 hours for 0.167778 hours
Agent: 927 at 22.905000 hours for 0.067222 hours
Agent: 955 at 23.175833 hours for 0.073056 hours
Agent: 961 at 22.984722 hours for 0.079444 hours
Agent: 967 at 22.488611 hours for 0.655833 hours
Agent: 972 at 22.488611 hours for 0.440278 hours
Agent: 973 at 22.488611 hours for 0.732500 hours
Agent: 983 at 22.596389 hours for 0.652500 hours



                          AGENT: 981

Start: 200.000000 , -39.091820 , 0.000000
Target: -200.000000 , 166.647536 , 0.000000
Total distance travelled: 459.629851 [nm]
Line distance: 447.846454 [nm]
```

```
Travelled: 2.631124 % more of their line route distance
Flight time: 3621.000000 seconds or 1.005833 hours
Maximum distance was 487.447832 [nm] with agent: 995
Minimum distance was 5.550510 [nm] with agent: 872
Total time in resolution: 3461.000000 seconds or 0.961389 hours
Agent: 981 met 20 agent(s)
Agent: 814 at 22.617222 hours for 0.033611 hours
Agent: 815 at 22.643056 hours for 0.046111 hours
Agent: 831 at 22.636389 hours for 0.005833 hours
Agent: 833 at 22.678889 hours for 0.073333 hours
Agent: 849 at 22.567222 hours for 0.067778 hours
Agent: 850 at 22.583889 hours for 0.069167 hours
Agent: 851 at 22.633611 hours for 0.041111 hours
Agent: 853 at 22.648333 hours for 0.036111 hours
Agent: 868 at 22.631667 hours for 0.063056 hours
Agent: 872 at 22.663889 hours for 0.061111 hours
Agent: 875 at 22.648889 hours for 0.064167 hours
Agent: 922 at 22.844444 hours for 0.033333 hours
Agent: 934 at 22.869444 hours for 0.029167 hours
Agent: 949 at 22.889444 hours for 0.027500 hours
Agent: 961 at 22.465556 hours for 0.255000 hours
Agent: 962 at 22.465556 hours for 0.896111 hours
Agent: 964 at 22.465556 hours for 0.950833 hours
Agent: 976 at 22.925833 hours for 0.047500 hours
Agent: 979 at 22.960000 hours for 0.014722 hours
Agent: 986 at 22.955000 hours for 0.039167 hours
```

AGENT: 982

```
Start: 159.925009 , -113.311295 , 0.000000
Target: 200.000000 , -135.761640 , 0.000000
Total distance travelled: 44.012778 [nm]
Line distance: 44.012778 [nm]
Travelled: 0.000000 % more of their line route distance
Flight time: 351.000000 seconds or 0.097500 hours
Maximum distance was 475.221364 [nm] with agent: 792
Minimum distance was 15.763889 [nm] with agent: 989
Total time in resolution: 193.000000 seconds or 0.053611 hours
Agent: 982 met 1 agent(s)
Agent: 801 at 22.625000 hours for 0.053611 hours
```

AGENT: 983

```
Start: 48.316773 , -107.012124 , 0.000000
Target: -146.991113 , 200.000000 , 0.000000
Total distance travelled: 379.463787 [nm]
Line distance: 362.783599 [nm]
Travelled: 4.597834 % more of their line route distance
Flight time: 3002.000000 seconds or 0.833889 hours
Maximum distance was 456.618390 [nm] with agent: 823
Minimum distance was 5.313742 [nm] with agent: 973
Total time in resolution: 2542.000000 seconds or 0.706111 hours
Agent: 983 met 8 agent(s)
Agent: 909 at 22.588611 hours for 0.225000 hours
Agent: 927 at 22.784444 hours for 0.103889 hours
Agent: 955 at 23.137778 hours for 0.035278 hours
```

Agent: 967 at 22.488889 hours for 0.556389 hours
Agent: 972 at 22.466944 hours for 0.367500 hours
Agent: 973 at 22.481111 hours for 0.640833 hours
Agent: 976 at 22.720833 hours for 0.064167 hours
Agent: 980 at 22.670000 hours for 0.480000 hours

AGENT: 984

Start: -200.000000 , -172.643662 , 0.000000
Target: 200.000000 , 150.827451 , 0.000000
Total distance travelled: 513.570000 [nm]
Line distance: 512.515691 [nm]
Travelled: 0.205713 % more of their line route distance
Flight time: 4030.000000 seconds or 1.119444 hours
Maximum distance was 532.384840 [nm] with agent: 951
Minimum distance was 6.099176 [nm] with agent: 573
Total time in resolution: 4008.000000 seconds or 1.113333 hours
Agent: 984 met 13 agent(s)
Agent: 549 at 23.053333 hours for 0.057500 hours
Agent: 573 at 23.412500 hours for 0.064167 hours
Agent: 576 at 23.580833 hours for 0.007222 hours
Agent: 952 at 22.750278 hours for 0.011944 hours
Agent: 969 at 22.945278 hours for 0.032500 hours
Agent: 971 at 22.784722 hours for 0.060556 hours
Agent: 975 at 22.474722 hours for 0.206389 hours
Agent: 976 at 23.039722 hours for 0.047500 hours
Agent: 978 at 22.680833 hours for 0.000278 hours
Agent: 979 at 23.083611 hours for 0.023611 hours
Agent: 985 at 23.098333 hours for 0.024444 hours
Agent: 986 at 23.087778 hours for 0.058333 hours
Agent: 995 at 22.906389 hours for 0.063333 hours

AGENT: 985

Start: -157.960358 , 200.000000 , 0.000000
Target: 200.000000 , -83.888146 , 0.000000
Total distance travelled: 484.535406 [nm]
Line distance: 456.198390 [nm]
Travelled: 6.211555 % more of their line route distance
Flight time: 3802.000000 seconds or 1.056111 hours
Maximum distance was 513.807091 [nm] with agent: 901
Minimum distance was 5.224641 [nm] with agent: 979
Total time in resolution: 3509.000000 seconds or 0.974722 hours
Agent: 985 met 10 agent(s)
Agent: 549 at 22.915833 hours for 0.082222 hours
Agent: 819 at 22.566667 hours for 0.018056 hours
Agent: 857 at 22.651667 hours for 0.013333 hours
Agent: 929 at 22.507222 hours for 0.051389 hours
Agent: 959 at 22.515278 hours for 0.209722 hours
Agent: 960 at 22.507222 hours for 0.071111 hours
Agent: 965 at 22.759444 hours for 0.196111 hours
Agent: 979 at 22.772778 hours for 0.687222 hours
Agent: 986 at 22.992500 hours for 0.477778 hours
Agent: 999 at 22.942500 hours for 0.013056 hours

Σε αυτό μπορούμε να βρούμε αναλυτικά δεδομένα σχετικά με την αρχή και προορισμό του κάθε αεροσκάφους, πόσο πέταξε παραπάνω από την ευθεία απόσταση μεταξύ αρχής και προορισμού, την ελάχιστη και μέγιστη απόσταση του και με ποιο αεροσκάφος, την συνολική ώρα που χρειάστηκε να λειτουργήσει ο αλγόριθμος για να αποφύγει κάποιο/α αεροσκάφος/η και τέλος ποια αεροσκάφη συνάντησε και για πόση ώρα.

Ένα πιο γενικό μέρος αυτού του αρχείο είναι η αρχή του η οποία μας δίνει διάφορες πληροφορίες για ολόκληρο το simulation. Εκεί μας δίνονται ο μέσος όρος και συνολικός αριθμός διαφόρων μεγεθών όπως:

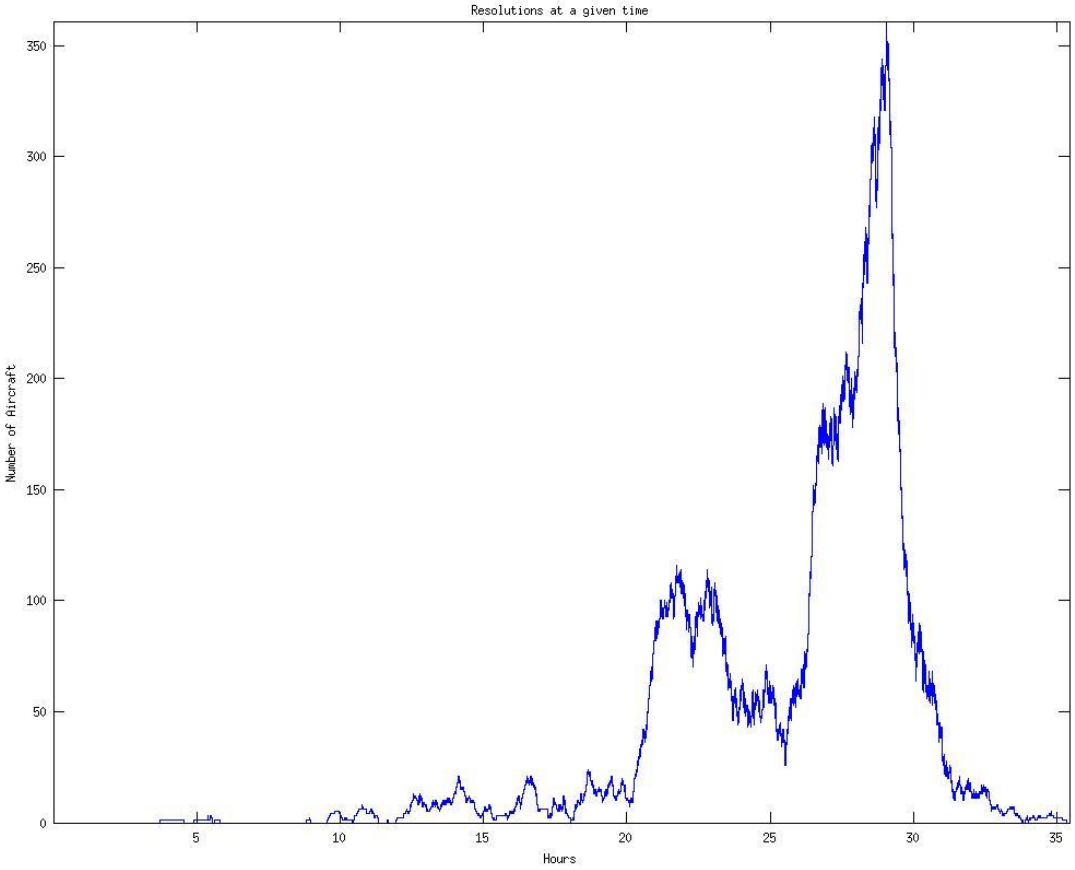
```
Agents Travelled(mean value): 1.154335 % more of their line route
distance
Flight time: 1783143.000000 seconds or 495.317500 hours
Flight time(Mean Value): 1783.143000 seconds or 0.495318 hours
Total time in resolution(all agents): 1300341.000000 seconds or
361.205833 hours
Mean time in resolution: 1300.341000 seconds or 0.361206 hours
Mean value of resolutions for each agent: 4.652000
Average % time in resolution: 72.924101 %
Execution time: 5460.670337 seconds
Flight time/execution time: 326.542877
```

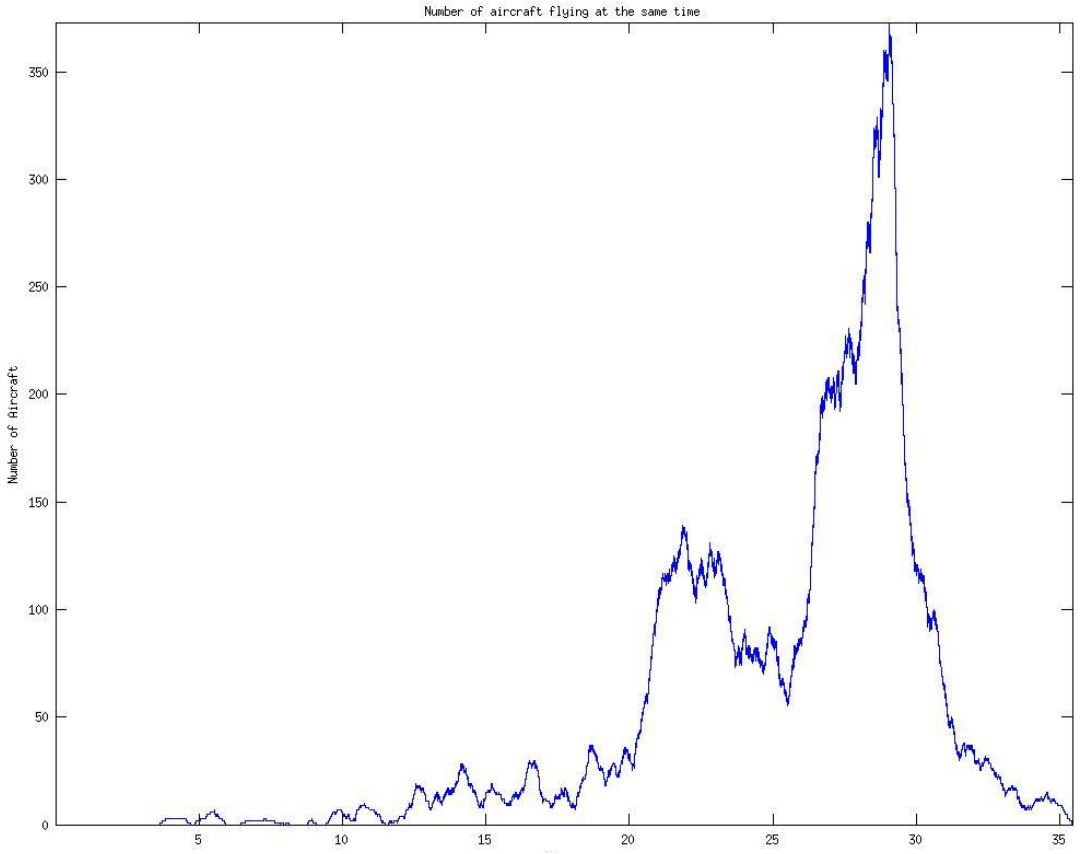
- Πόσο % πέταξε το κάθε αεροσκάφος περισσότερο από το αν ακολουθούσε ευθεία πορεία κατά μέσο όρο
- Οι συνολικές ώρες πτήσης και μία μέση τιμή αυτών
- Πόση ώρα τα αεροσκάφη αφιέρωσαν στην αποφυγή πιθανών συγκρούσεων και ο μέσος όρος αυτής
- Πόσα άλλα αεροσκάφη συνάντησε κατά μέσο όρο το κάθε αεροσκάφος
- Τον χρόνο εκτέλεσης του προγράμματος
- Ένα λόγο Χρόνου Πτήσης/Υπολογιστικού χρόνου

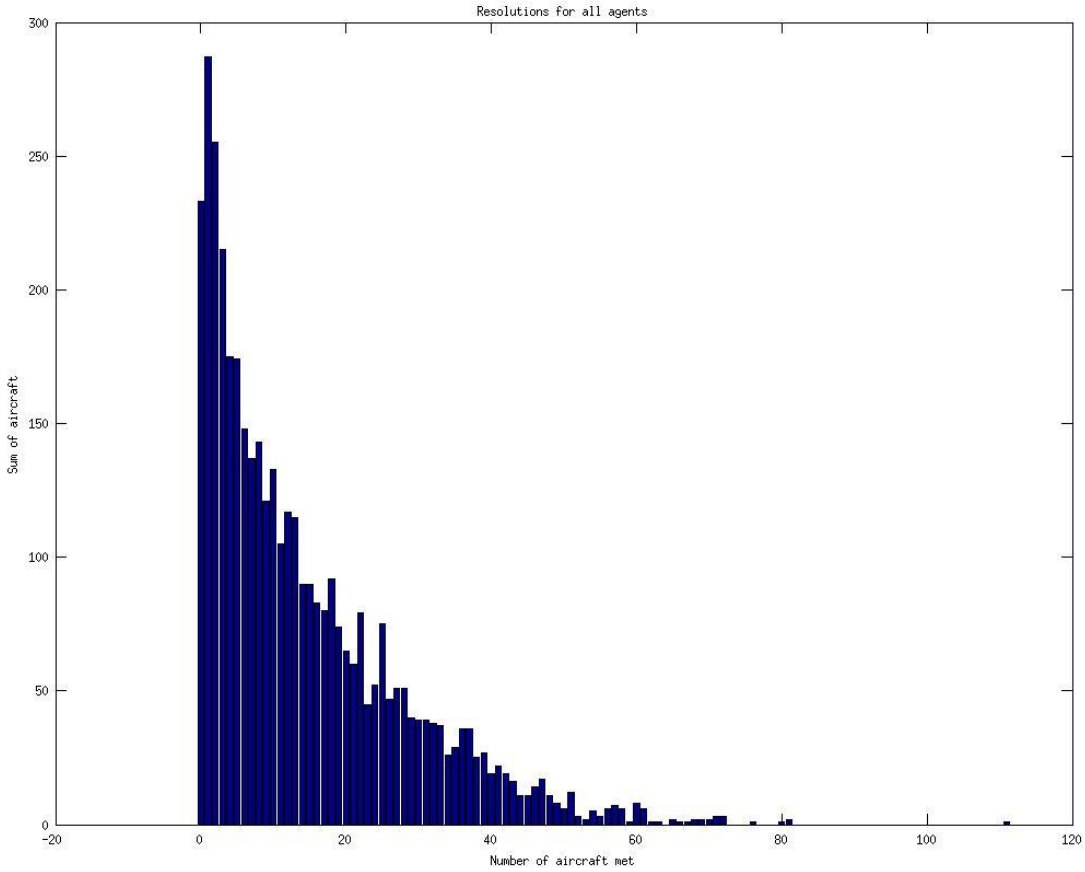
Το τελευταίο είναι ένα σημαντικό μέγεθος αφού μας δείχνει ότι η ώρα των πτήσεων είναι πολλή μεγαλύτερη από τον πραγματικό υπολογιστικό χρόνο της μεθόδου. Επίσης αυτό συμβαίνει με ένα σημερινό τυπικό υπολογιστή. Οπότε θεωρητικά θα μπορούσαμε να εφαρμόσουμε την συγκεκριμένη μέθοδο χρησιμοποιώντας έναν τυπικό υπολογιστή πάνω στο αεροσκάφος χωρίς να έχουμε ιδιαίτερες υπολογιστικές απαιτήσεις. Επίσης και το αρχικό μέγεθος είναι ένας σημαντικός δείκτης αφού μας δείχνει ένα μέσο όρο της απόκλισης από την πορεία του κάθε αεροσκάφους κάτι το οποίο έχει άμεση σχέση με κατανάλωση καυσίμου, άνεση των επιβατών κτλ.

2.5.2.2Σενάριο πρώτων 4000 αεροσκαφών

Σε αυτό το σενάριο τρέξαμε τα πρώτα 4000 αεροσκάφη. Αυτά μπαίνουν και στην δεύτερη ημέρα όπου η εναέρια κυκλοφορία είναι πολύ μεγαλύτερη. Παίρνουμε τα παρακάτω διαγράμματα.





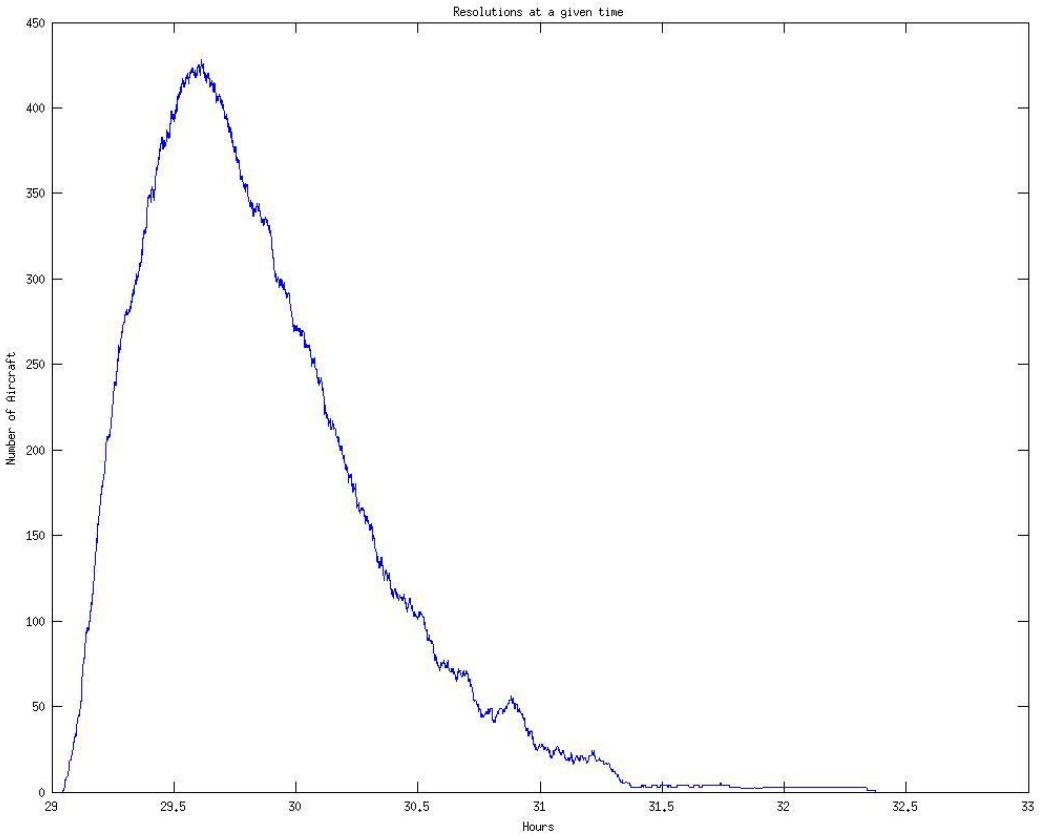


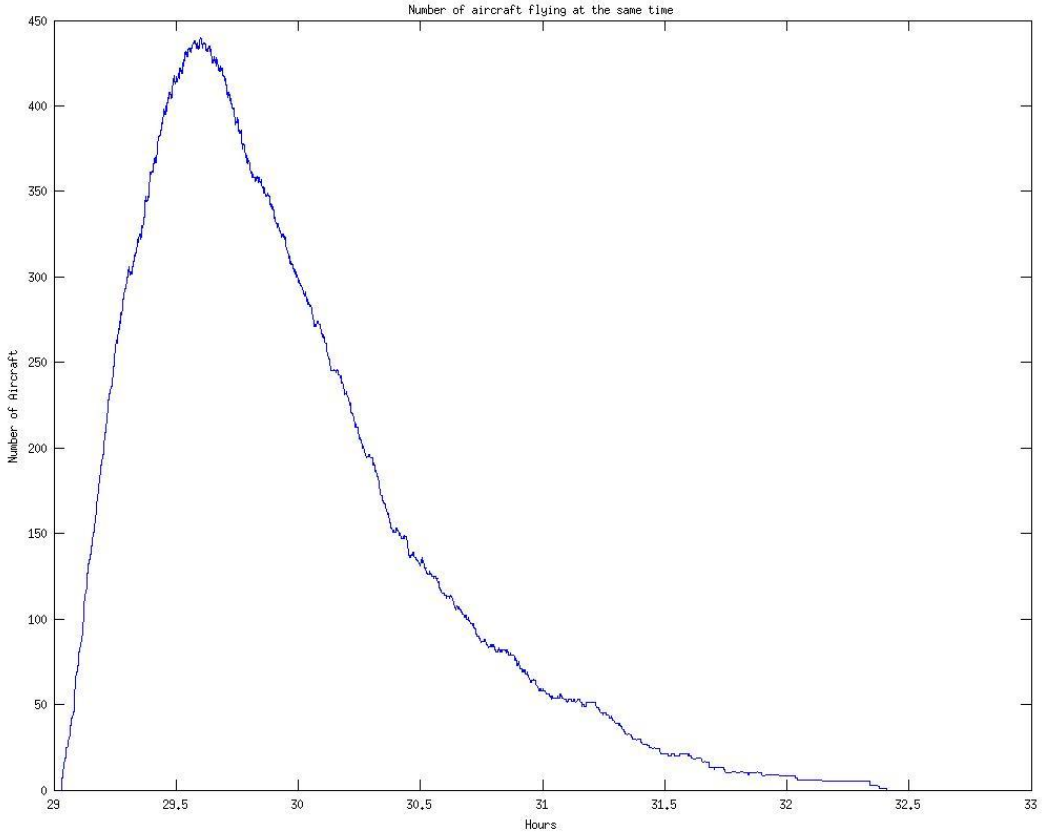
Εικόνες 2.5.2.2.1-3

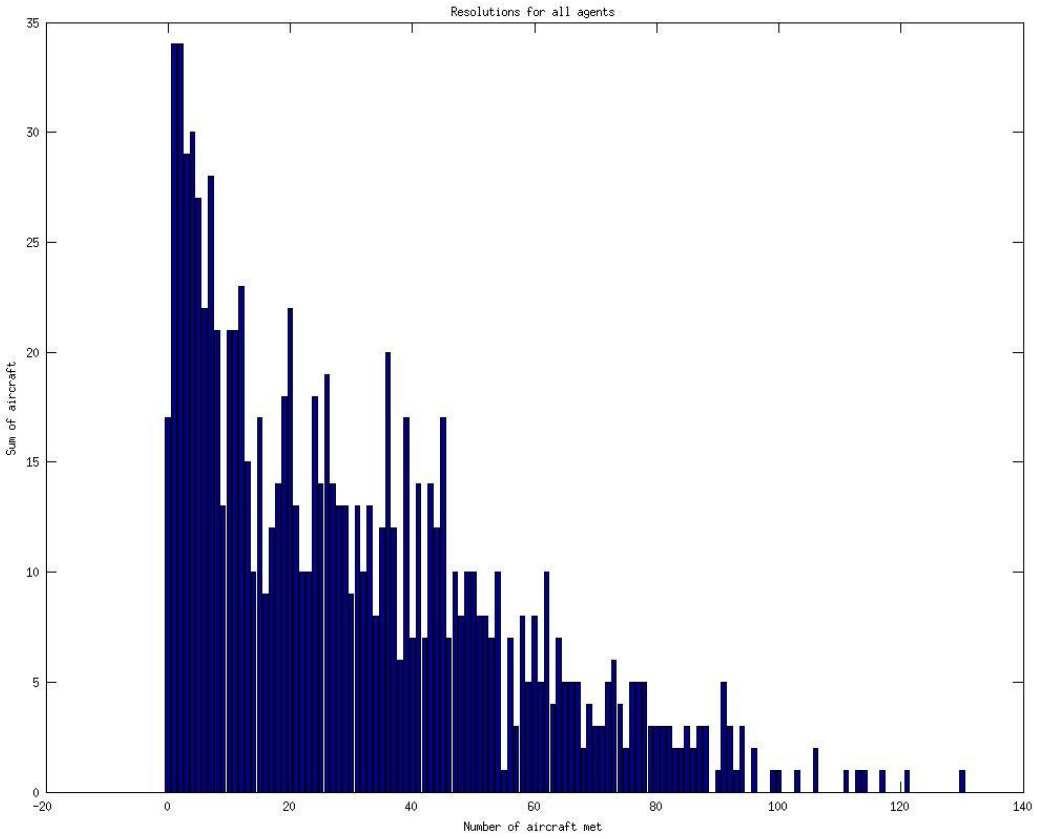
Όπως παρατηρούμε στην εικόνα 2.5.2.2.1 τα μέγιστα resolutions και αεροσκάφη που πετάνε κάθε χρονική στιγμή σε σχέση με την εξομοίωση των 1000 ξεπερνούν τα 350 σε σχέση με τις μέγιστες τιμές πριν που ήταν περίπου στο 140. Εδώ λόγω του μεγαλύτερου πλήθους των αεροσκαφών φτάνουμε μέχρι περίπου την ώρα 35. Παρατηρούμε ότι την δεύτερη ημέρα η κίνηση αυξάνει αρκετά. Περίπου από την ώρα 20 παρατηρούμε τις μέγιστες τιμές της πρώτης ημέρας οι οποίες είναι περίπου στις 140 όπως υπολογίσαμε και πριν. Εδώ ίσως είναι ελαφρώς αυξημένες αφού στο προηγούμενο παράδειγμα περίπου αυτήν την ώρα ο ρυθμός εισόδου των αεροσκαφών στον εναέριο χώρο έγινε μηδέν απότομα(φτάσαμε τους 1000) ενώ τους 4000 συνέχισε να υπάρχει είσοδος αεροσκαφών. Παρατηρούμε επίσης μία πτώση της κίνησης των αεροσκαφών περίπου γύρω στα μεσάνυχτα της πρώτης ημέρας που είναι λογική και πάλι σταδιακή αύξηση της από τα μεσάνυχτα και μετά. Σχετικά με το διάγραμμα 2.5.2.2.3 έχουμε παρόμοια αποτελέσματα με πριν, όμως αυξάνει ο αριθμός των αεροσκαφών της κάθε κλάσης και τώρα υπάρχουν αεροσκάφη που συναντάνε μέχρι και 80 άλλα αεροσκάφη στον δρόμο τους.

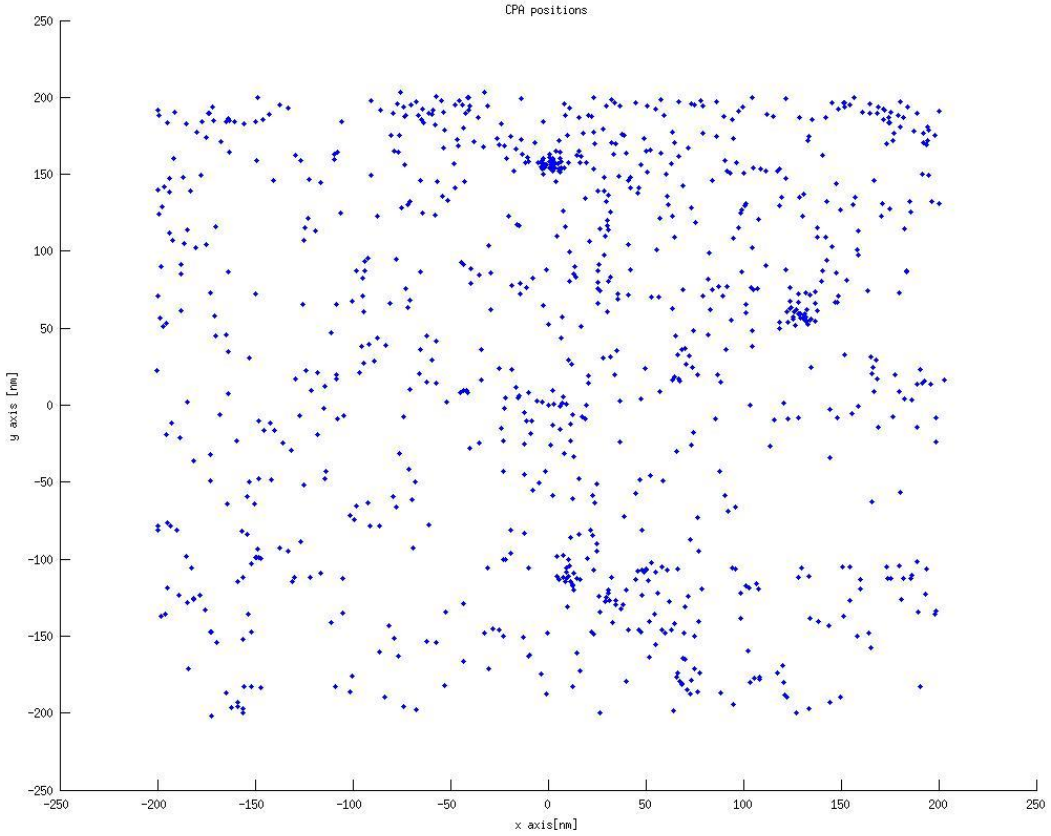
2.5.2.3Σενάριο 1000 αεροσκαφών σε υψηλή κίνηση(περίπου από την ώρα 29)

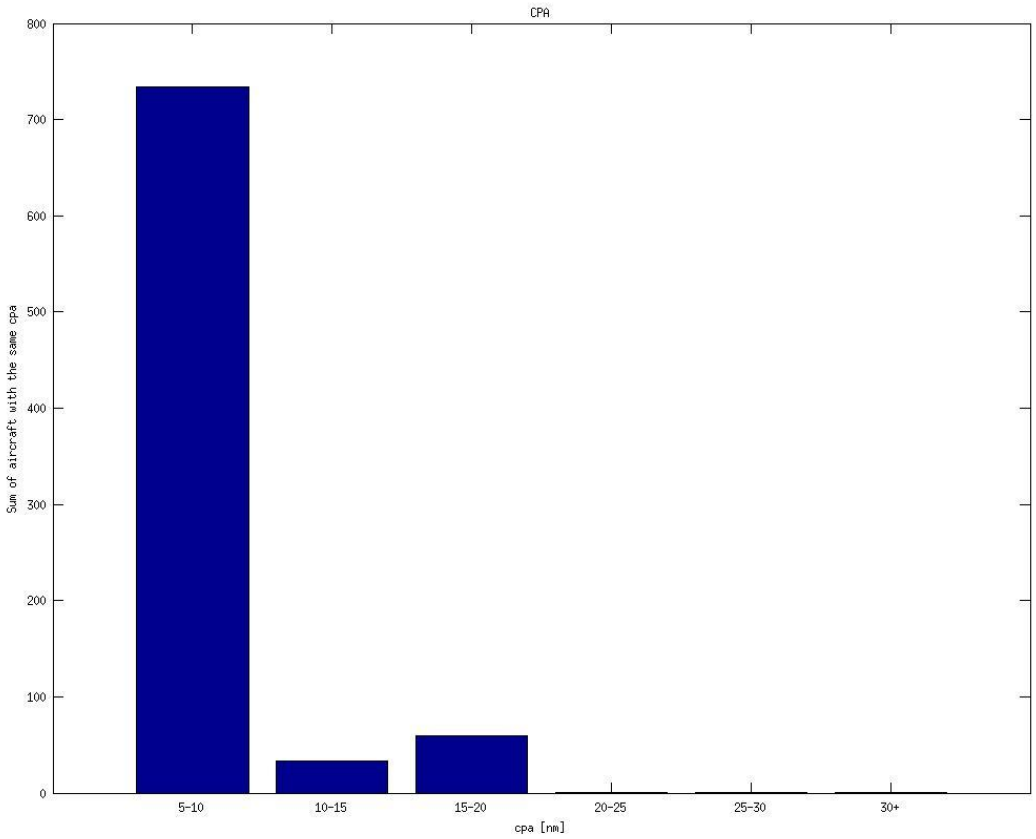
Εδώ δίνουμε ένα σενάριο σχετικά με το πώς συμπεριφέρεται η μέθοδος σε μία ώρα που η κίνηση αρχίζει να αυξάνεται πολύ σε μικρό χρονικό διάστημα το οποίο θα χρησιμεύσει για σύγκριση. Αυτή η εξομοίωση βρίσκεται περίπου στο τέλος των εξομοιώσεων που ακολουθούν και χωρίζουν σε διαφορετικά ύψη πτήσεις. Είναι ένα δείγμα του πως συμπεριφέρονται τα αεροσκάφη όταν πετάξουν πολλά μαζί στο ίδιο ύψος.


Resolutions at a given time
Number of Aircraft
Hours
0
50
100
150
200
250
300
350
400
450
29
29.5
30
30.5
31
31.5
32
32.5
33


Number of aircraft flying at the same time
Number of Aircraft
Hours
0
50
100
150
200
250
300
350
400
450
29
29.5
30
30.5
31
31.5
32
32.5
33


Resolutions for all agents
Sum of aircraft
Number of aircraft met
0
5
10
15
20
25
30
35
-20
0
20
40
60
80
100
120
140


CPA positions
y axis [nm]
x axis[nm]
250
200
150
100
50
0
-50
-100
-150
-200
-250



Εικόνες 2.5.2.3.1-5

Παρατηρούμε παραπάνω ότι τα αεροσκάφη που βρίσκονται εν πτήση την ίδια στιγμή και τα resolutions ξεπερνάνε τα 430 καθώς επίσης και ότι αυξάνεται ο μέσος αριθμός των αεροσκαφών που συναντάει το κάθε αεροσκάφος. Δίνουμε επίσης το διάγραμμα των κοντινότερων αποστάσεων που βρέθηκαν τα αεροσκάφη σε σχέση με κάποιο άλλο αεροσκάφος και τα σημεία που αυτό συνέβη. Εδώ πλέον τα περισσότερα βρίσκονται μεταξύ 5nm και 10nm λόγω του μεγάλου αριθμού των αεροσκαφών που πετάνε την ίδια στιγμή

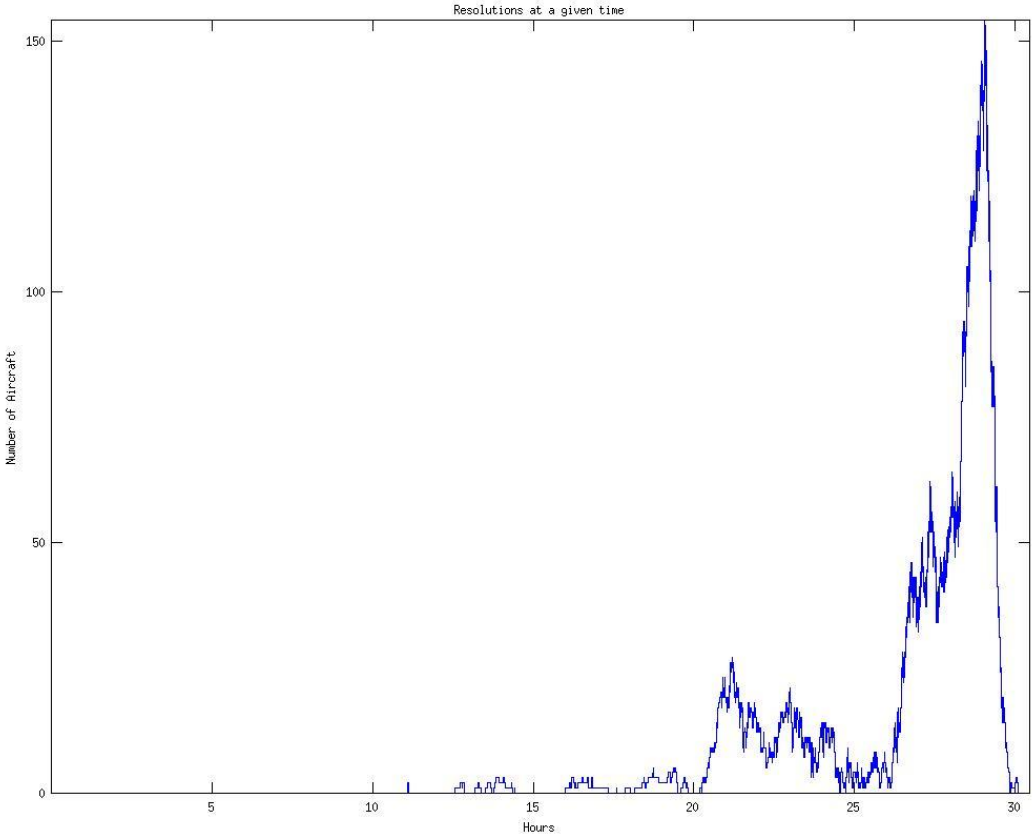
```
                    SIMULATION RESULTS

Total number of agents: 1000


Agents Travelled(mean value): 5.418844 % more of their line route
distance
Flight time: 1751668.000000 seconds or 486.574444 hours
Flight time(Mean Value): 1751.668000 seconds or 0.486574 hours
Total time in resolution(all agents): 1593592.000000 seconds or
442.664444 hours
Mean time in resolution: 1593.592000 seconds or 0.442664 hours
Mean value of resolutions for each agent: 30.543000
Average % time in resolution: 90.975687 %
Execution time: 21956.282228 seconds
Flight time/execution time: 79.779809
```

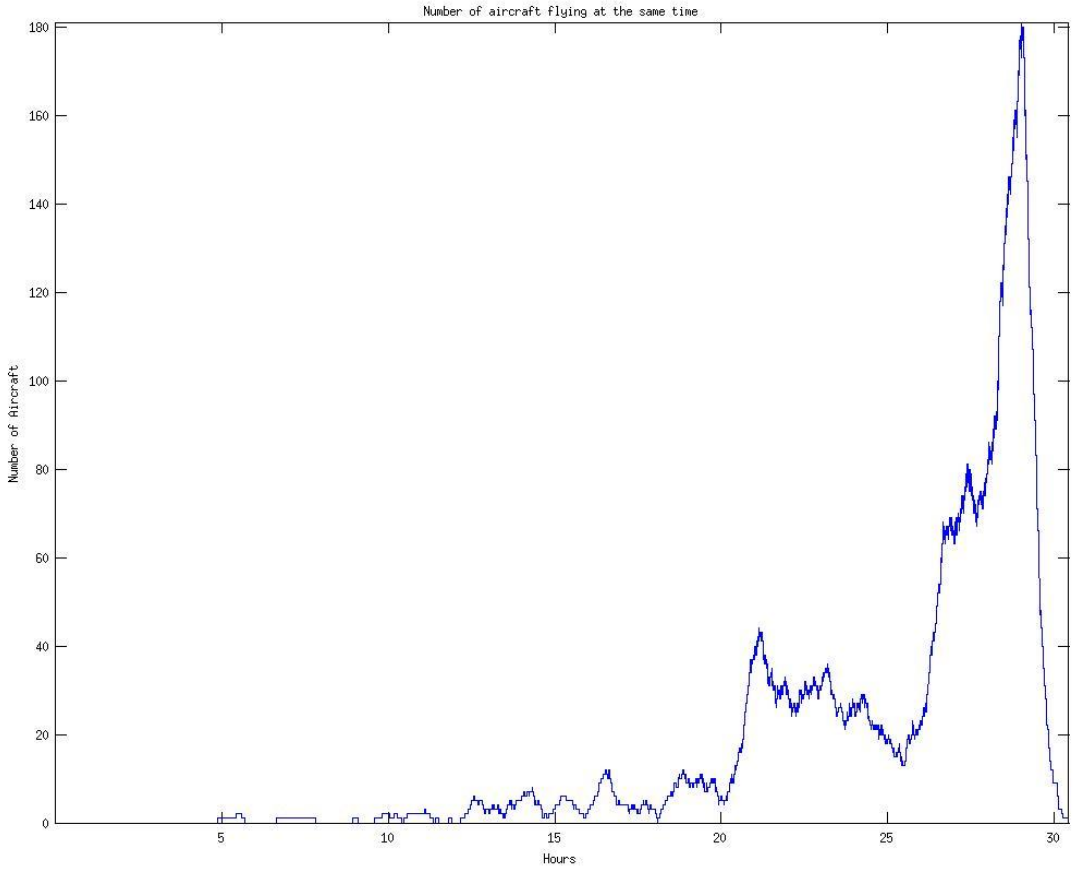
2.5.2.4 Σενάριο 4000 αεροσκαφών με επιλογή ανά 4

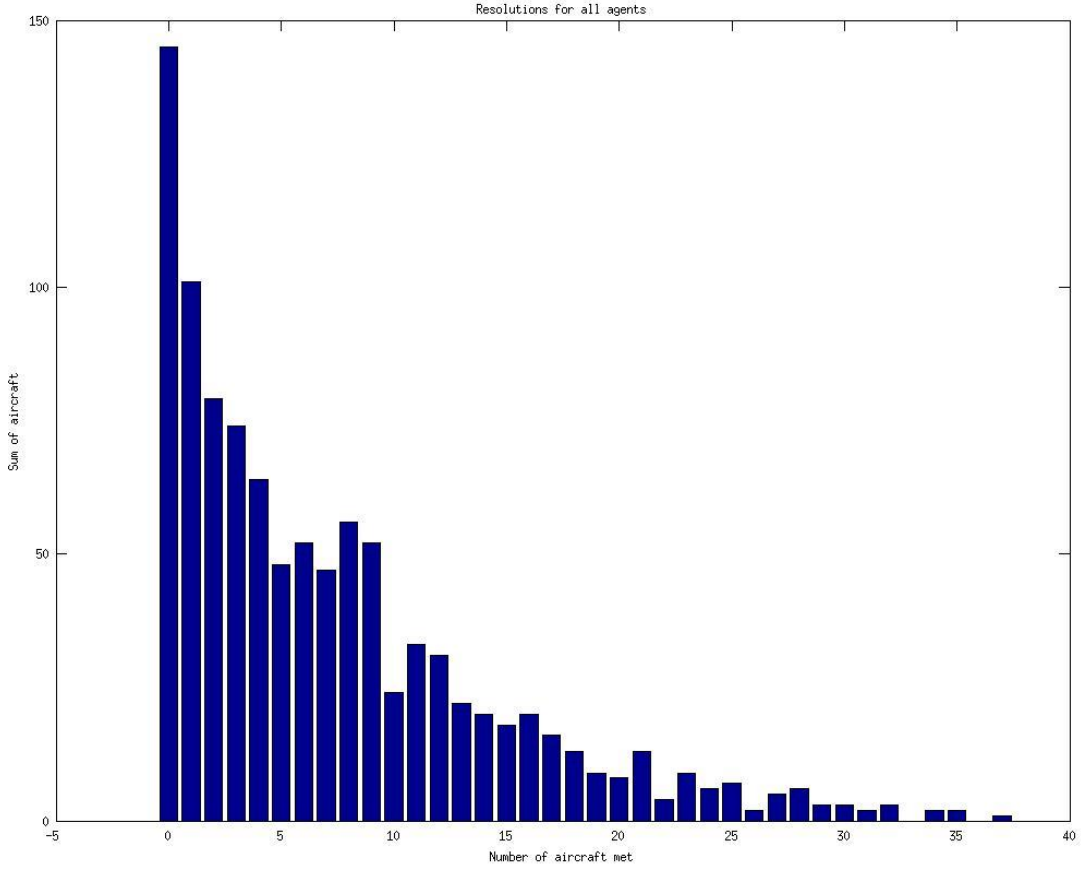
Στις τέσσερις εξομοιώσεις που ακολουθούν τρέξαμε το πρόγραμμα επιλέγοντας αεροσκάφη ανά 4 και ξεκινώντας από τα αεροσκάφη 1,2,3 και 4 αντίστοιχα. Έτσι πχ για την πρώτη εξομοίωση που θα παρουσιάσουμε τώρα τα αεροσκάφη που επιλέγονται είναι το 1,5,9,13,17 κτλ. Αυτό το κάνουμε για να δούμε τα αποτελέσματα που έχουμε αν χωρίζαμε τις συνολικές πτήσεις σε 4 διαφορετικές ομάδες καθαρά με βάση την ώρα απογείωσης τους. Έτσι γίνεται μία αποσυμφόρηση του εναέριου χώρου. Πρέπει να έχουμε υπόψη μας ότι εδώ πλέον τρέχουμε επιλέγουμε 1000 αεροσκάφη αλλά σε ένα συνολικό εύρος 4000 τα οποία χωρίζονται σε κάθε εξομοίωση. Θα δώσουμε παρακάτω όλα τα διαγράμματα και των τεσσάρων εξομοιώσεων και θα τα σχολιάσουμε στο τέλος.

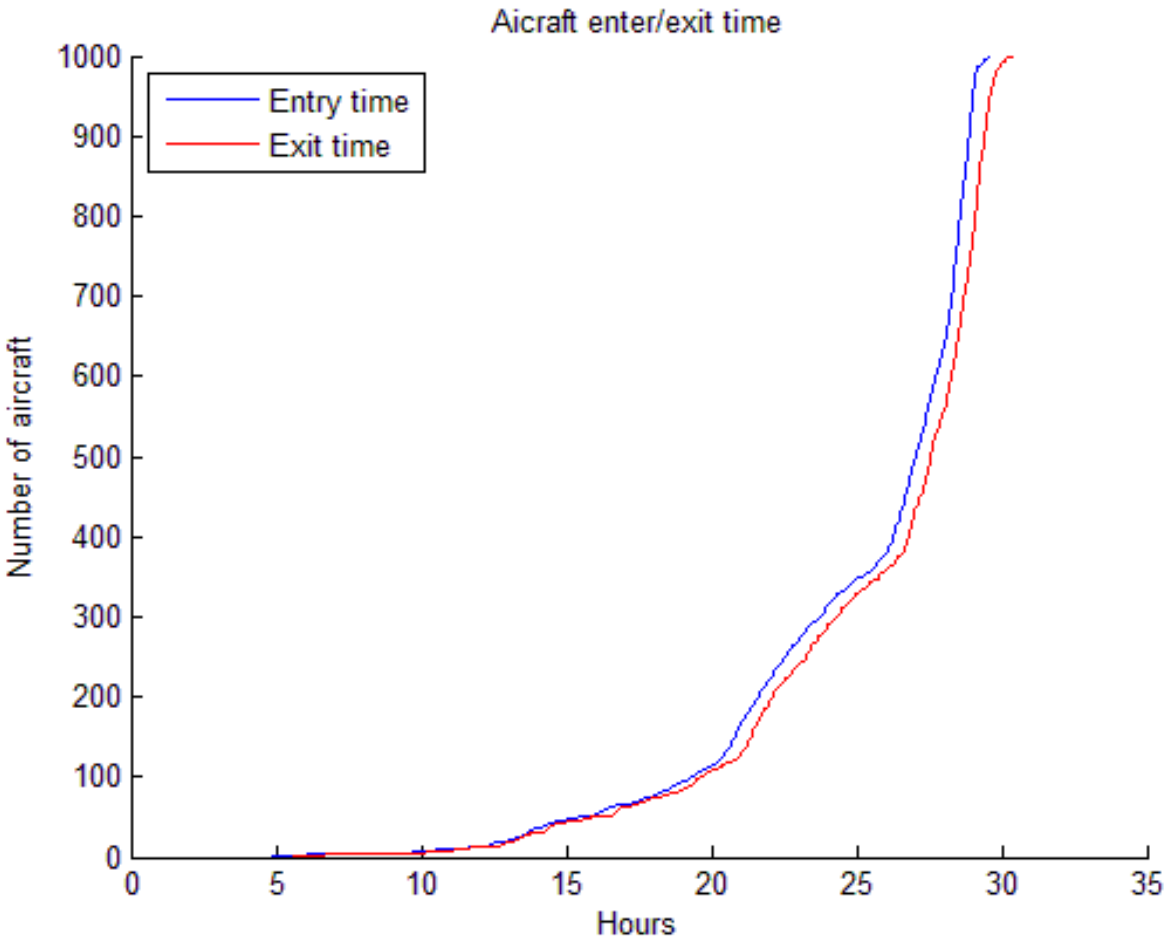
2.5.2.4.1Ξεκινώντας από το αεροσκάφος 1

Για αυτήν την περίπτωση παίρνουμε τα παρακάτω διαγράμματα




Number of aircraft flying at the same time
Number of Aircraft
Hours
0
20
40
60
80
100
120
140
160
180
5
10
15
20
25
30


Resolutions for all agents
Sum of aircraft
Number of aircraft met
0
50
100
150
-5
0
5
10
15
20
25
30
35
40

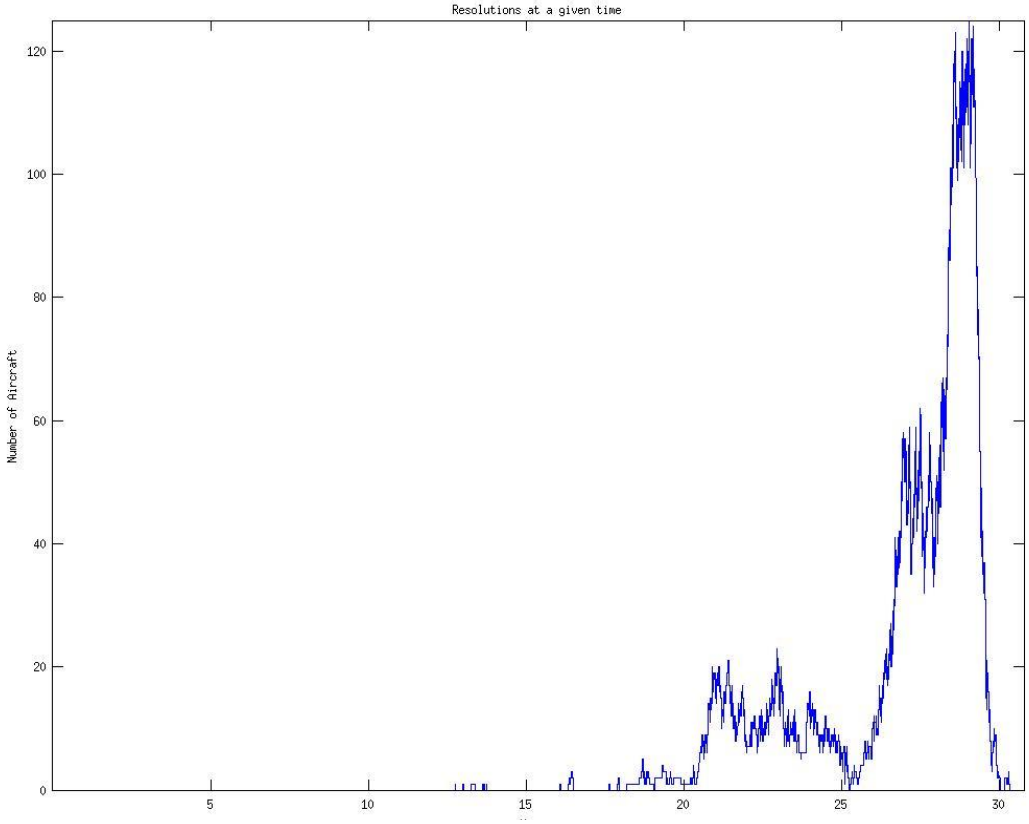


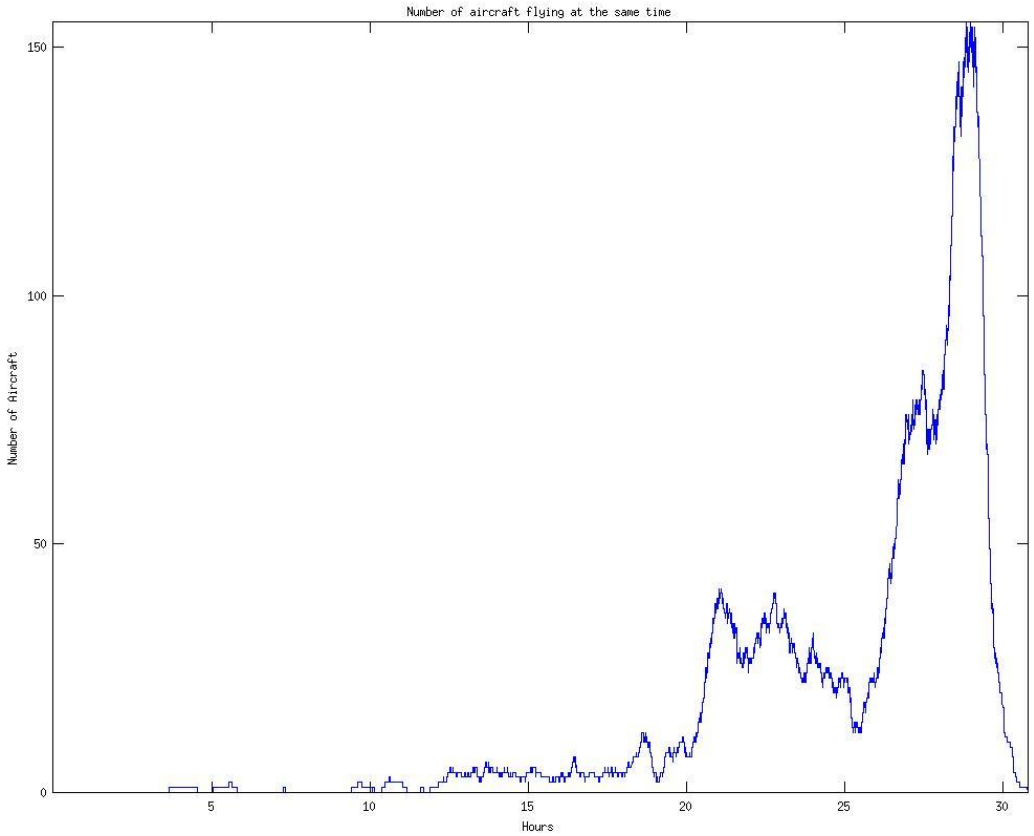
Εικόνες 2.5.2.4.1.1-4

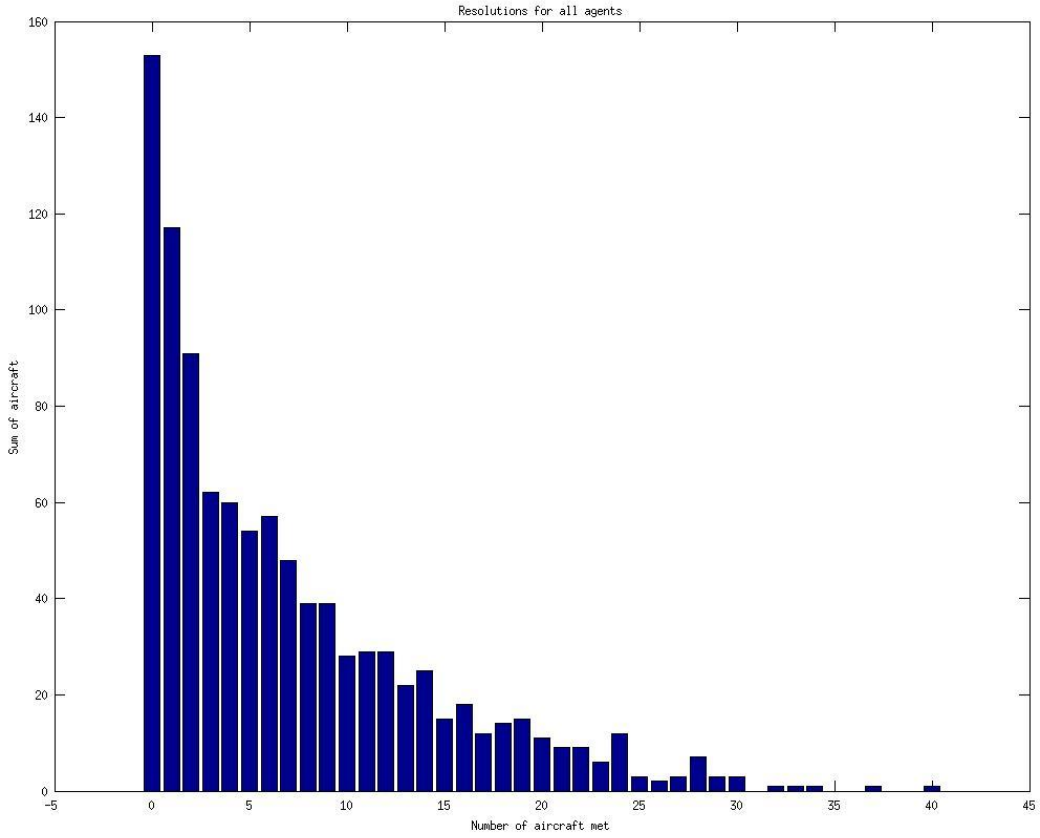
Και το makeresults.m δίνει

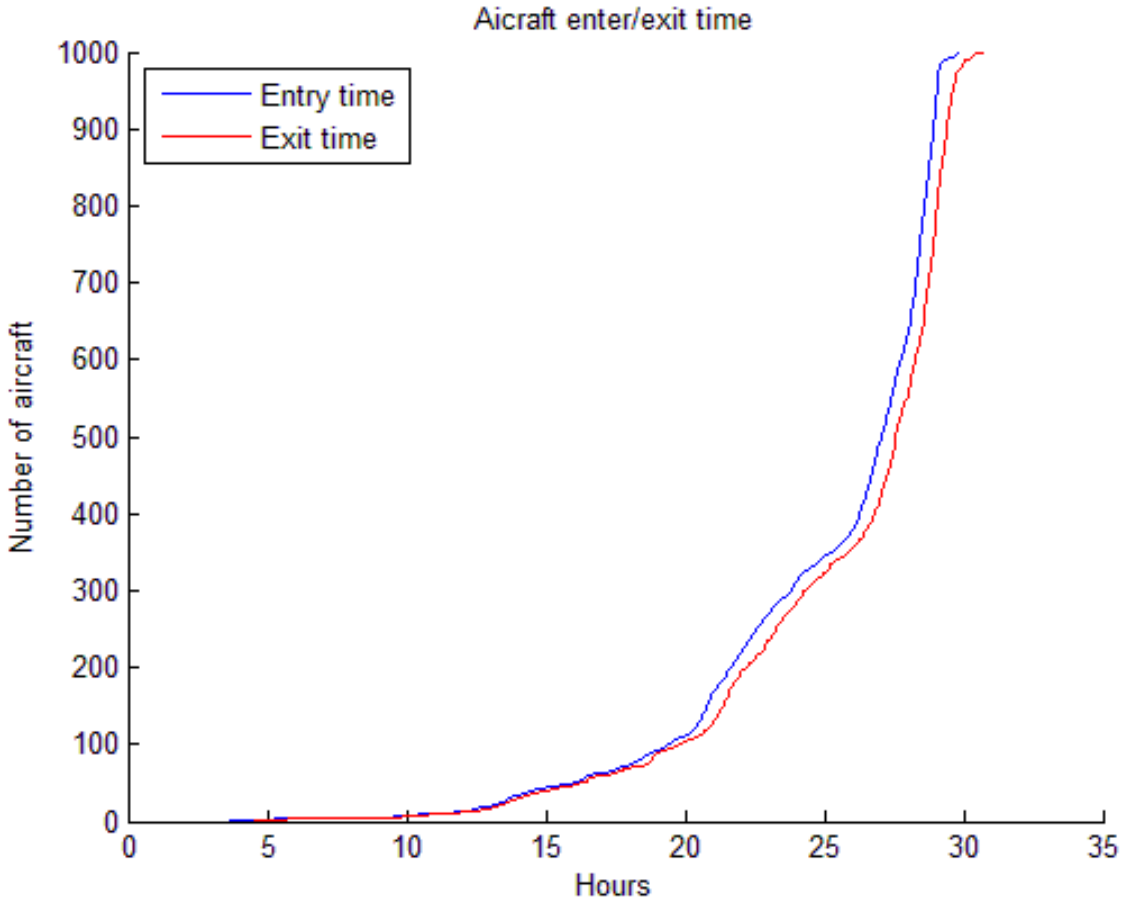
```
                    SIMULATION RESULTS

Total number of agents: 1000


Agents Travelled(mean value): 2.390727 % more of their line
route distance
Flight time: 1885084.000000 seconds or 523.634444 hours
Flight time(Mean Value): 1885.084000 seconds or 0.523634 hours
Total time in resolution(all agents): 1308367.000000 seconds
or 363.435278 hours
Mean time in resolution: 1308.367000 seconds or 0.363435 hours
Mean value of resolutions for each agent: 7.254000
Average % time in resolution: 69.406297 %
Execution time: 5886.301232 seconds
Flight time/execution time: 320.249326
```

2.5.2.4.2Ξεκινώντας από το αεροσκάφος 2









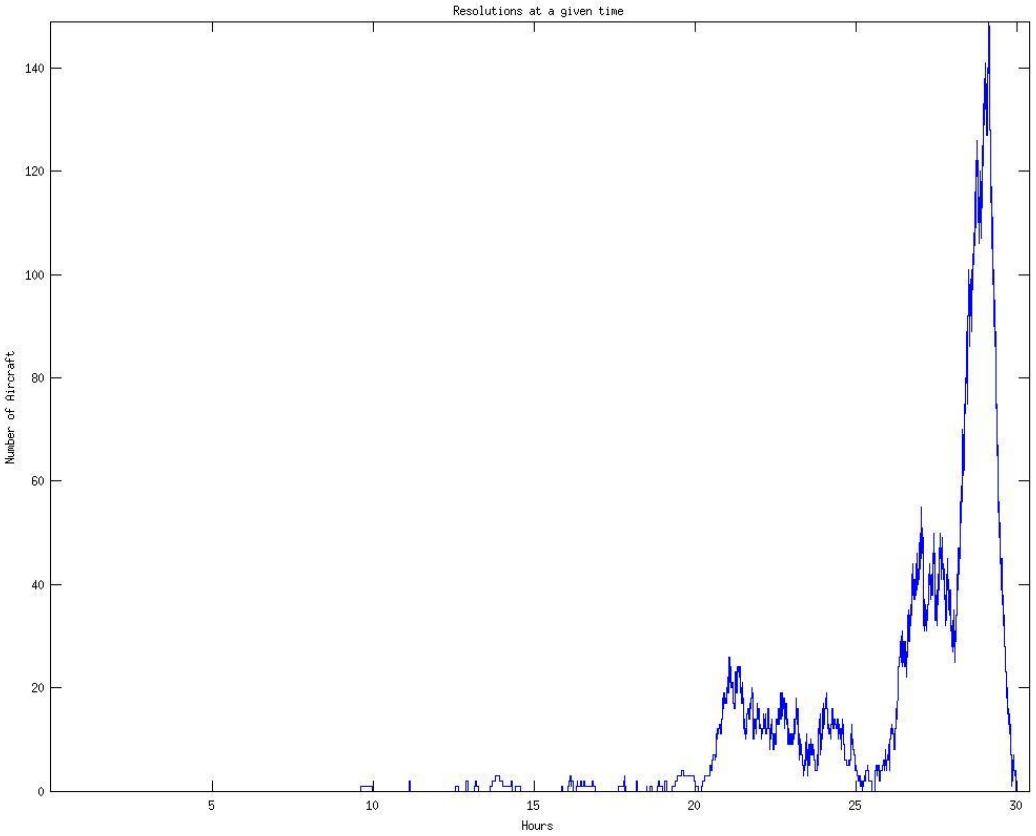
Εικόνες 2.5.2.4.2.1-4

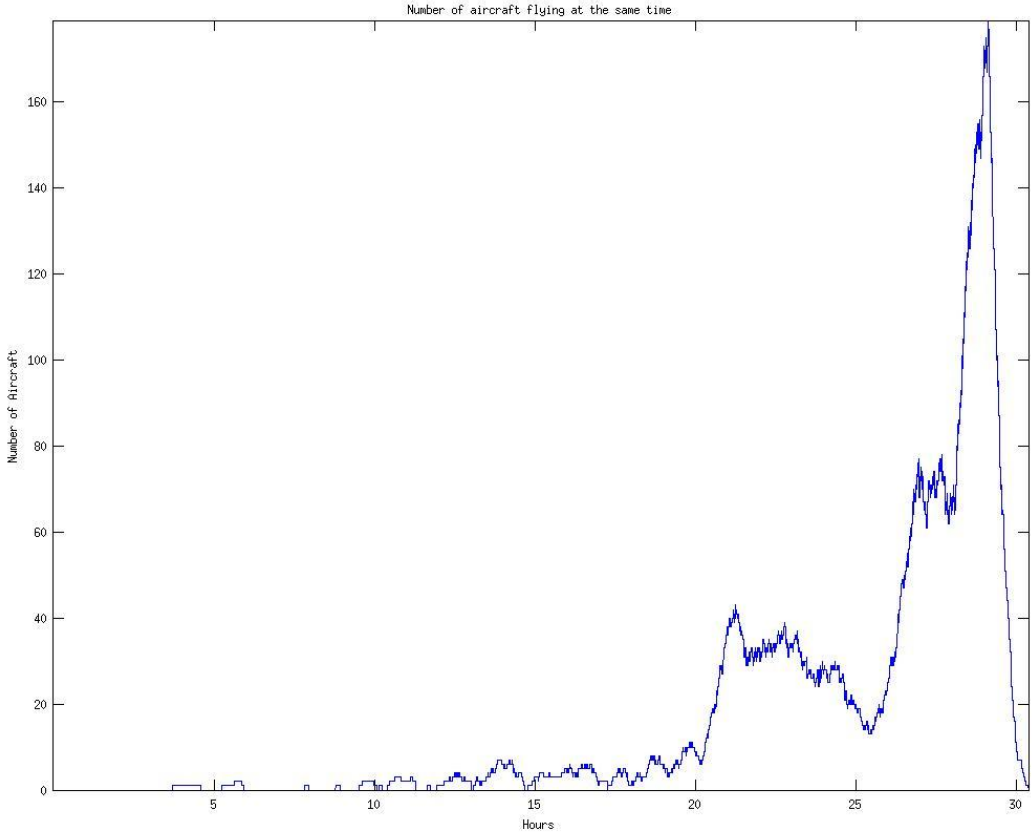
SIMULATION RESULTS

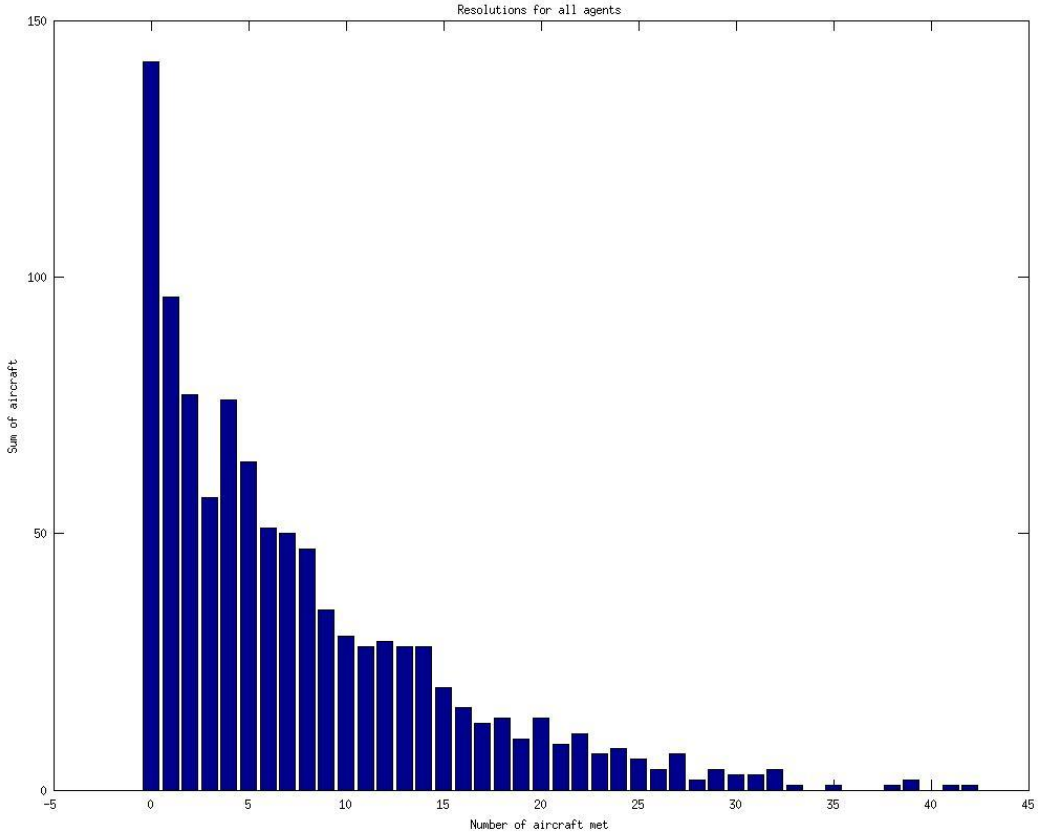
Total number of agents: 1000

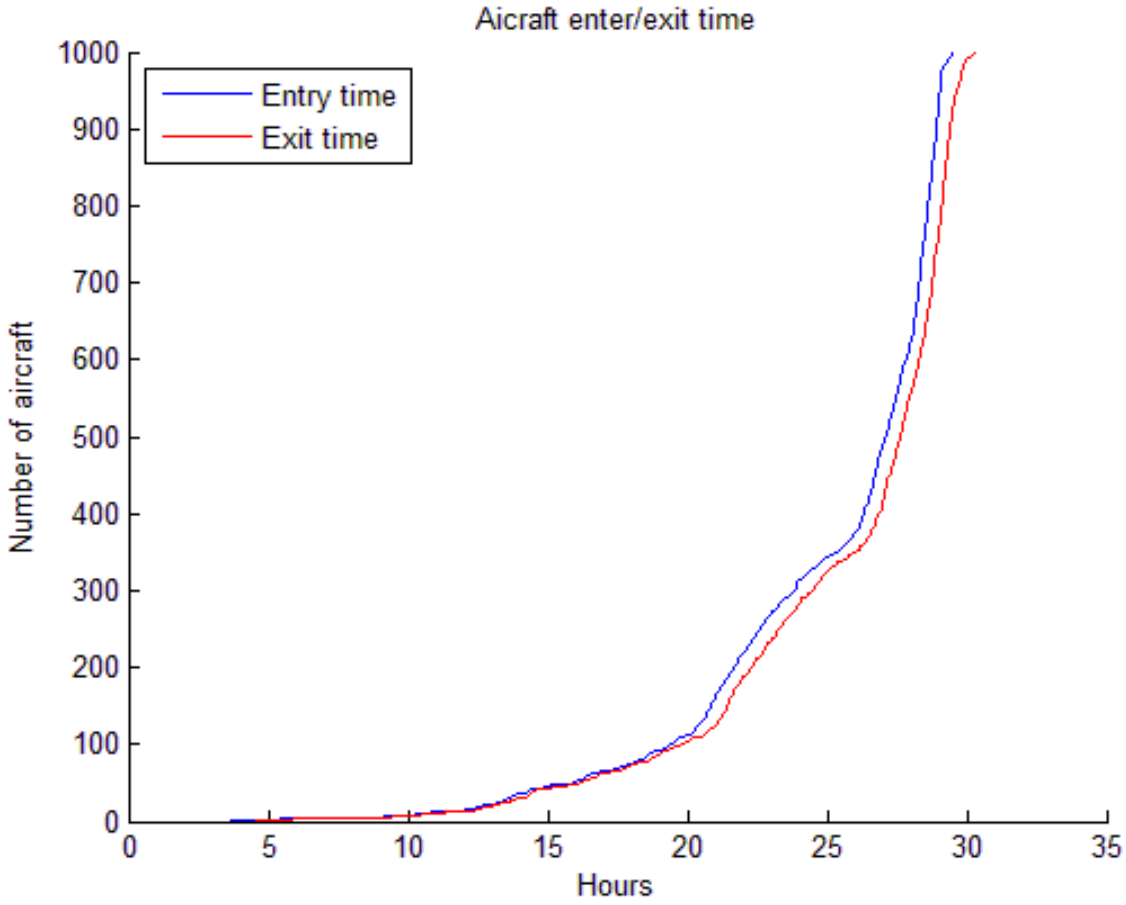
Agents Travelled(mean value): 2.377116 % more of their line route distance
Flight time: 1876940.000000 seconds or 521.372222 hours
Flight time(Mean Value): 1876.940000 seconds or 0.521372 hours
Total time in resolution(all agents): 1265490.000000 seconds or 351.525000 hours
Mean time in resolution: 1265.490000 seconds or 0.351525 hours
Mean value of resolutions for each agent: 6.963000
Average % time in resolution: 67.423040 %
Execution time: 5620.412478 seconds
Flight time/execution time: 333.950579

2.5.2.4.3Ξεκινώντας από το αεροσκάφος 3




Number of aircraft flying at the same time
Number of Aircraft
Hours
0
20
40
60
80
100
120
140
160
5
10
15
20
25
30


Resolutions for all agents
Sum of aircraft
Number of aircraft met
0
50
100
150
-5
0
5
10
15
20
25
30
35
40
45



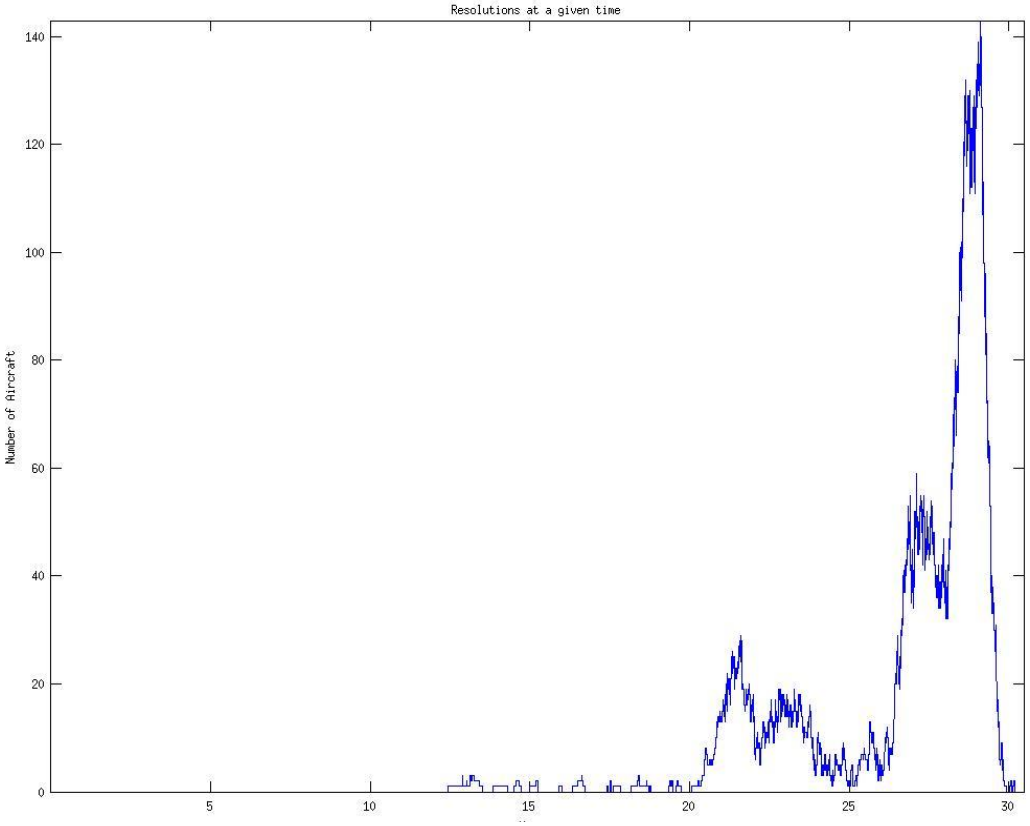
Εικόνες 2.5.2.4.3.1-4

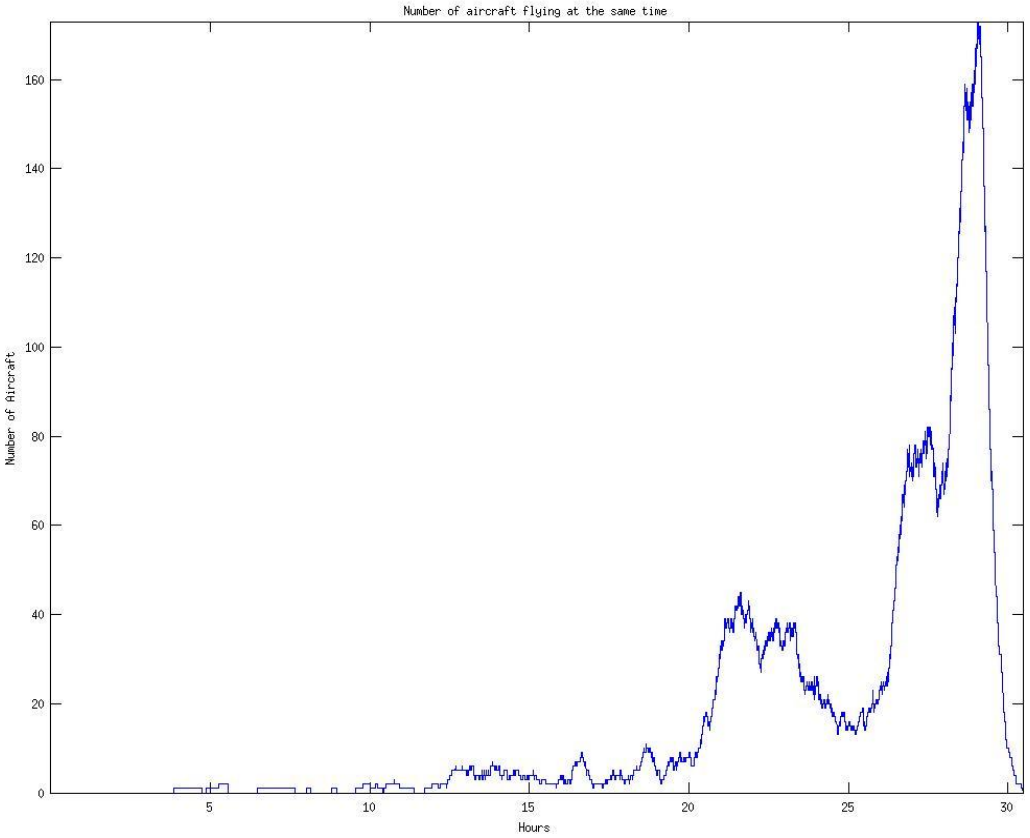
SIMULATION RESULTS

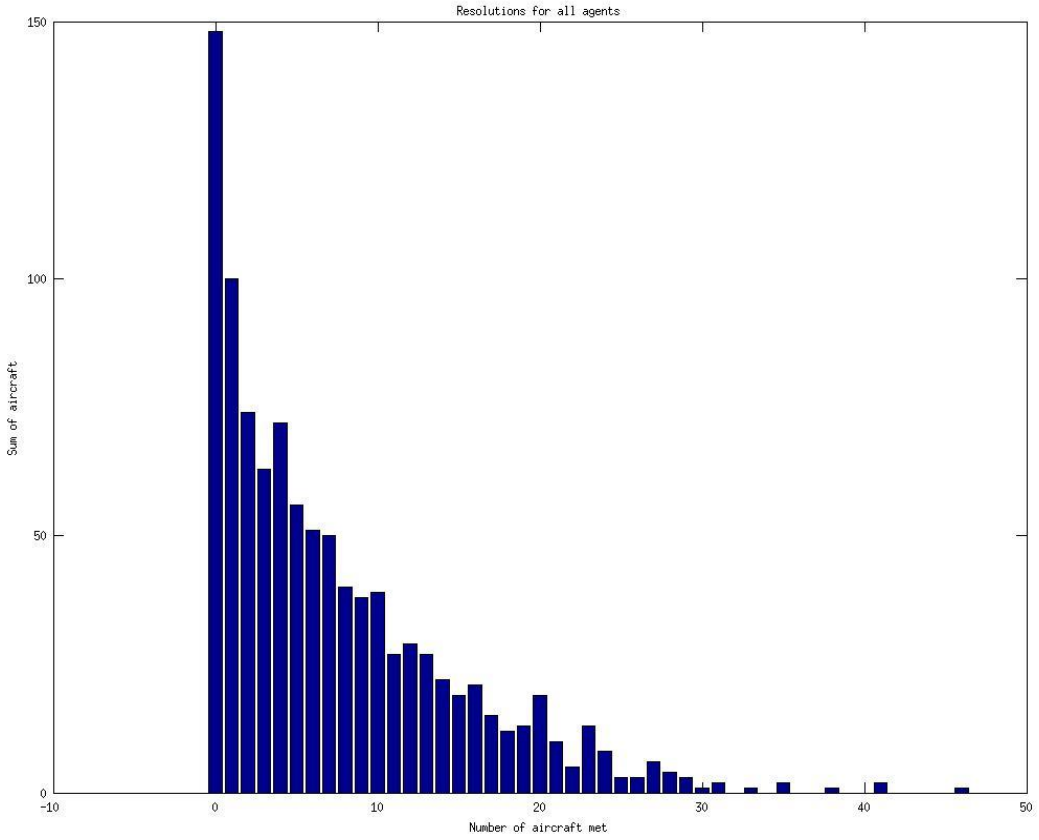
Total number of agents: 1000

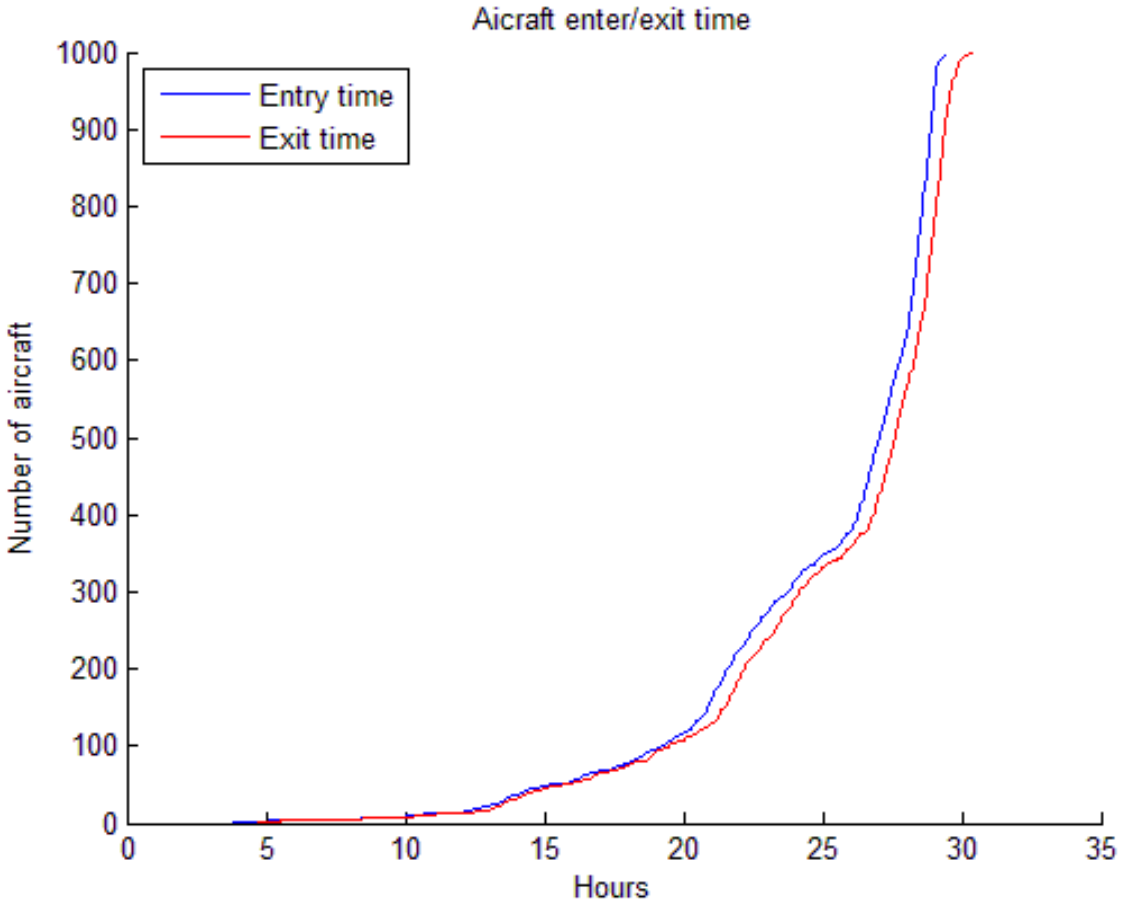
Agents Travelled(mean value): 2.444906 % more of their line route distance
Flight time: 1881920.000000 seconds or 522.755556 hours
Flight time(Mean Value): 1881.920000 seconds or 0.522756 hours
Total time in resolution(all agents): 1324389.000000 seconds or 367.885833 hours
Mean time in resolution: 1324.389000 seconds or 0.367886 hours
Mean value of resolutions for each agent: 7.580000
Average % time in resolution: 70.374352 %
Execution time: 5843.137189 seconds
Flight time/execution time: 322.073561

2.5.2.4.4Ξεκινώντας από το αεροσκάφος 4









Εικόνες 2.5.2.4.4.1-4

```
SIMULATION RESULTS

Total number of agents: 1000


Agents Travelled(mean value): 3.197039 % more of their line
route distance
Flight time: 1891159.000000 seconds or 525.321944 hours
Flight time(Mean Value): 1891.159000 seconds or 0.525322 hours
Total time in resolution(all agents): 1299418.000000 seconds
or 360.949444 hours
Mean time in resolution: 1299.418000 seconds or 0.360949 hours
Mean value of resolutions for each agent: 7.422000
Average % time in resolution: 68.710140 %
Execution time: 5985.912308 seconds
Flight time/execution time: 315.934966
```

2.5.2.4.5 Σχολιασμός

Στις τέσσερις παραπάνω περιπτώσεις παρατηρούμε ότι και τα resolutions και ο αριθμός των αεροσκαφών που πετάνε κάποια χρονική στιγμή δεν ξεπερνάνε του 140 με 160 ανάλογα με το διάγραμμα. Αυτό ουσιαστικά είναι ένα δείγμα 4000 αεροσκαφών τα οποία πετάνε σε 4 διαφορετικά ύψη πτήσης(flight levels) και έχουν χωριστεί ομοιόμορφα βάση μόνο της ώρας απογείωσης τους. Αν συγκρίνουμε τα διαγράμματα αυτά με τα διαγράμματα των 4000 αεροσκαφών που δόθηκαν στο ***2.5.2.2*** βλέπουμε μία μεγάλη μείωση (και αναμενόμενη βέβαια) όπου εκεί πέρα οι αντίστοιχες τιμές κυμαίνονταν κοντά στο 350. Στα διαγράμματα αυτών των simulation παρατηρούμε επίσης παρόμοια αύξηση και μείωση αντίστοιχα της κίνησης της πρώτης ημέρας και μετά τα μεσάνυχτα μία μεγάλη αύξηση της κίνησης. Σχετικά με τα διαγράμματα ***2.5.2.4.x.3*** δηλαδή τα διαγράμματα που ομαδοποιούμε τα αεροσκάφη ανάλογα με τους πόσους συνάντησαν στον δρόμο τους παρατηρούμε ότι και εδώ έχουμε παρόμοια φθίνουσα πορεία αλλά σε σχέση με το ***2.5.2.1.3*** , ***2.5.2.2.3*** και ***2.5.2.3.3*** των πρώτων 1000 ,4000 και 1000 σε υψηλή κίνηση αεροσκαφών που τα περισσότερα από τα αεροσκάφη συνάντησαν ένα ή δύο αεροσκάφη στον δρόμο τους πλέον τα αεροσκάφη που δεν συνάντησαν κανένα άλλο στο δρόμο τους είναι τα περισσότερα. Επίσης βλέπουμε ότι οι τιμές Agents Travelled(mean value) μειώθηκαν σε σχέση με το 2.5.2.3 από περίπου 5.4% σε 2.3% έως 3.2% κάτι που φανερώνει μείωση μέση απόκλιση των αεροσκαφών από τις πορείες τους και αυτό συνέβη σε ελαφρώς περισσότερες ώρες πτήσης. Συνοπτικά παρουσιάζουμε ότι

	1000a/c υψηλή κίνηση	4000a/c επιλεγμένα ανά 4
Agents Travelled more of their line route distance (mean value):	5.4 %	2.3 – 3.2%
Flight time(Mean Value):	0.48 hours	~0.52 hours
Mean time in resolution:	0.44 hours	~0.36 hours
Mean value of resolutions for each agent:	30.54	6.96-7.6
Average % time in resolution:	91 %	67.9 - 70%
Flight time/execution time:	79.9	~315-333

Έτσι λοιπόν παρόλο τις περισσότερες ώρες πτήσης κατά μέσο όρο (διαφορετικό δείγμα) έχουμε μία αισθητή μείωση της απόκλισης από την πορεία του κάθε αεροσκάφους, του μέσου αριθμού των αεροσκαφών που συνάντησε το κάθε αεροσκάφος στον δρόμο του και του μέσου χρόνου που χρειάστηκε κάθε αεροσκάφος να αποφύγει κάποιον άλλο αεροσκάφος. Επίσης ο λόγος Χρόνου Πτήσης/Χρόνου εκτέλεσης αυξάνεται αρκετά δηλαδή μειώνεται πολύ ο υπολογιστικός φόρτος.

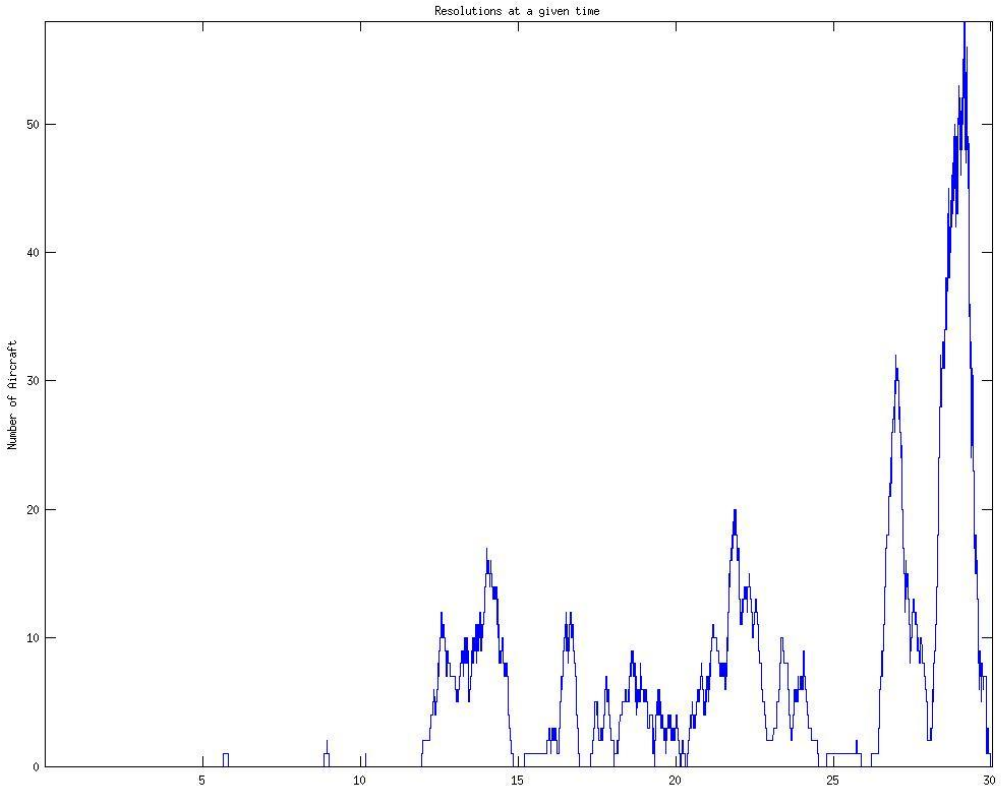
2.5.2.5 Σενάριο 4000 αεροσκαφών χωρισμένων ανά κατεύθυνση

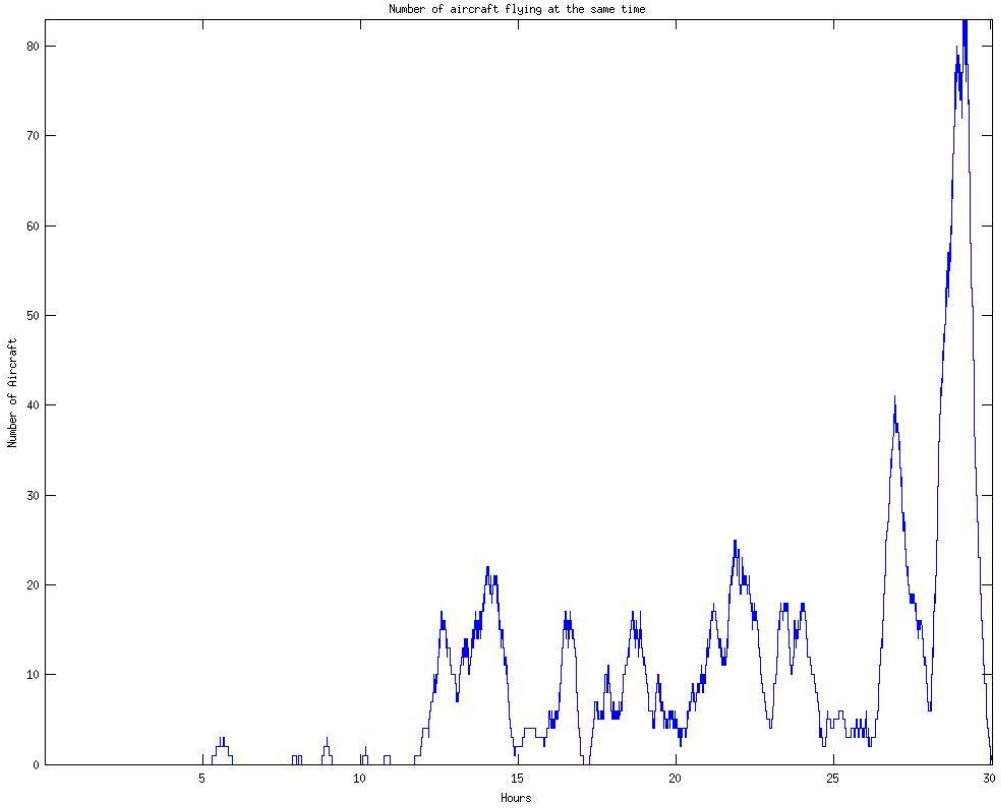
Σε αυτές τις εξομοιώσεις χωρίζουμε τα αεροσκάφη ανά κατεύθυνση και όχι με βάση τον χρόνο απογείωσης τους. Δηλαδή από τα 4000 πρώτα αεροσκάφη βρίσκουμε πόσα από αυτά πηγαίνουν προς την κάθε κατεύθυνση.

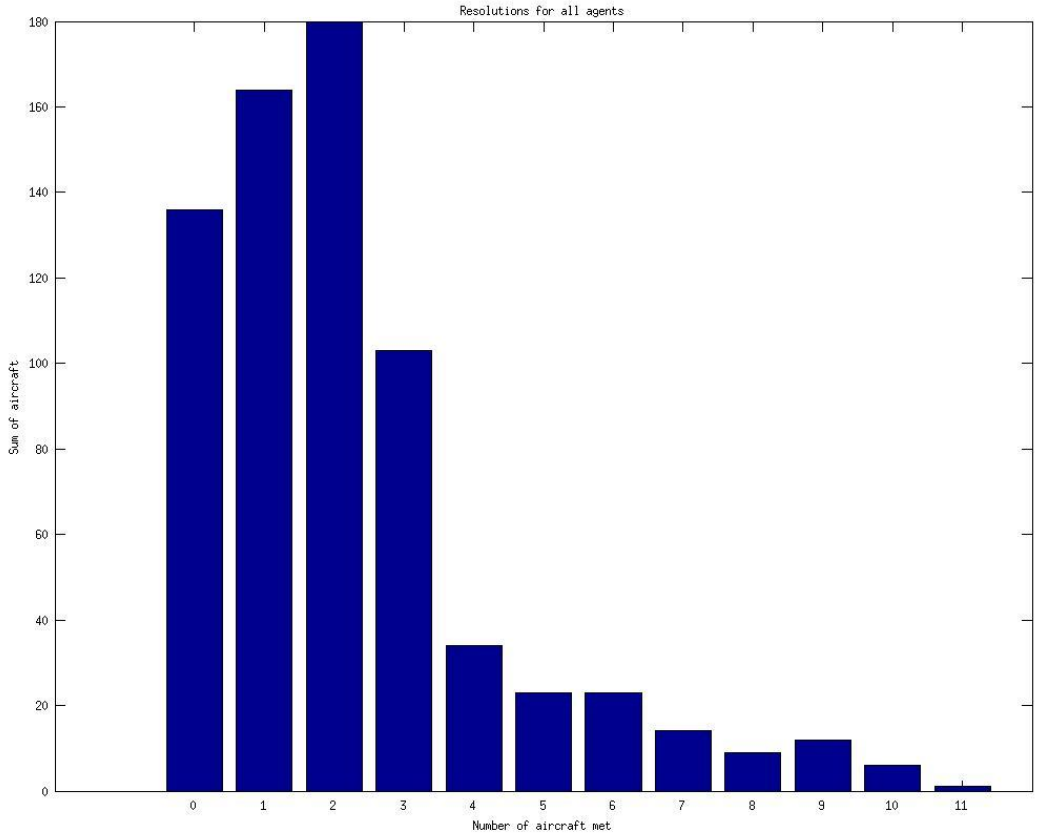
Βόρεια: 705/4000
Νότια: 1120/4000
Ανατολικά:789/4000
Δυτικά: 1386/4000

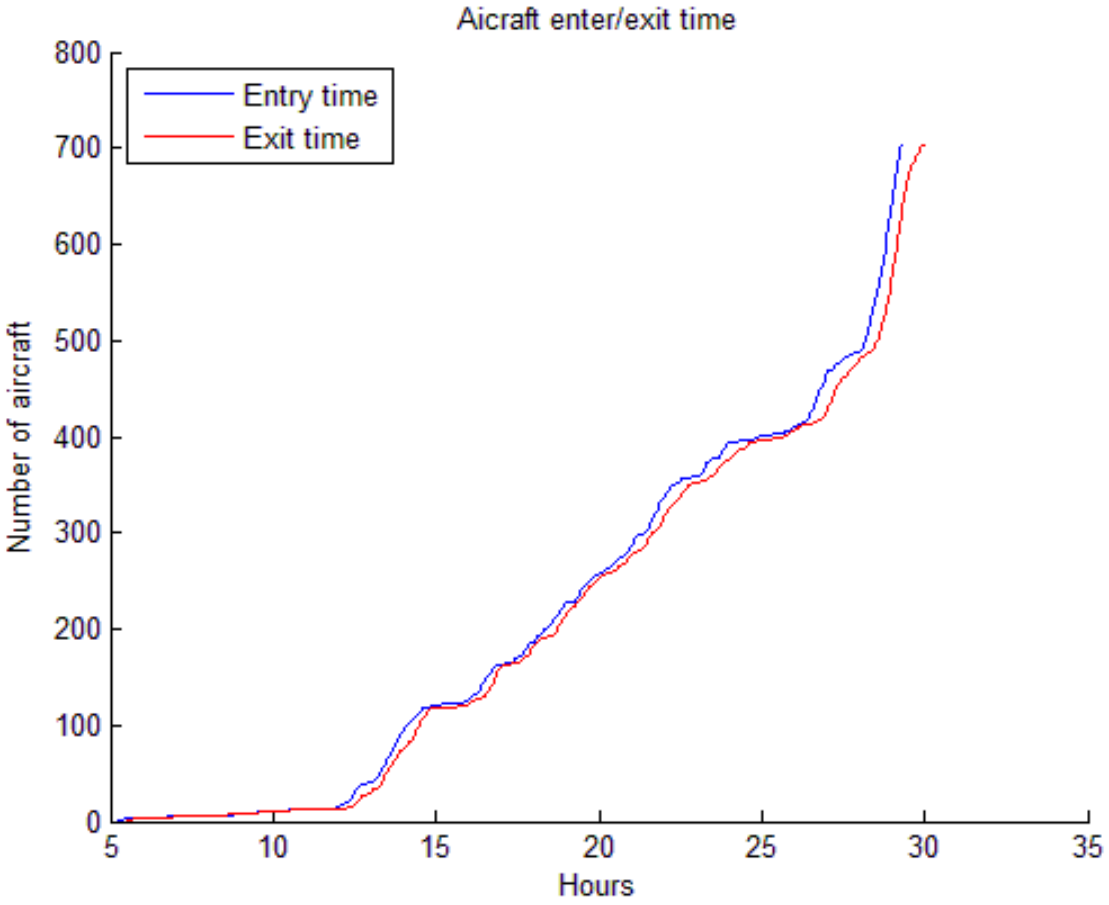
Δίνουμε τα παρακάτω αποτελέσματα τα οποία σχολιάζουμε στο τέλος.

2.5.2.5.1 Πτήσεις με βόρεια κατεύθυνση









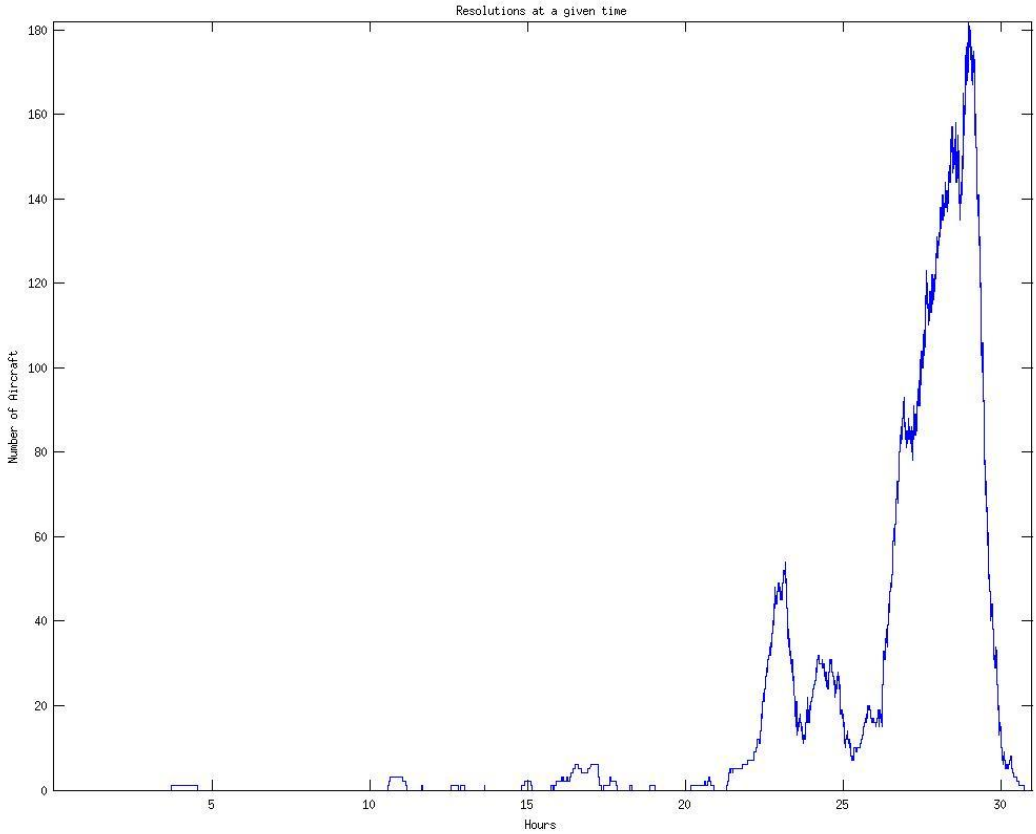
Εικόνες 2.5.2.5.1.1-4

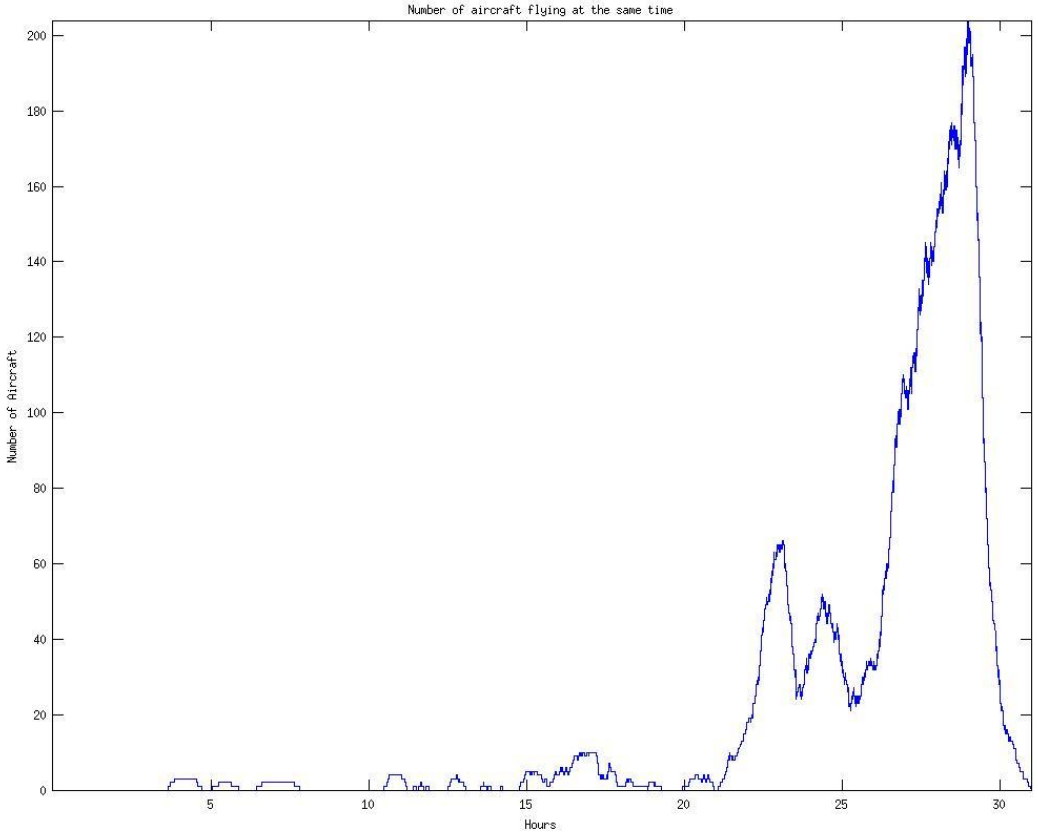
SIMULATION RESULTS

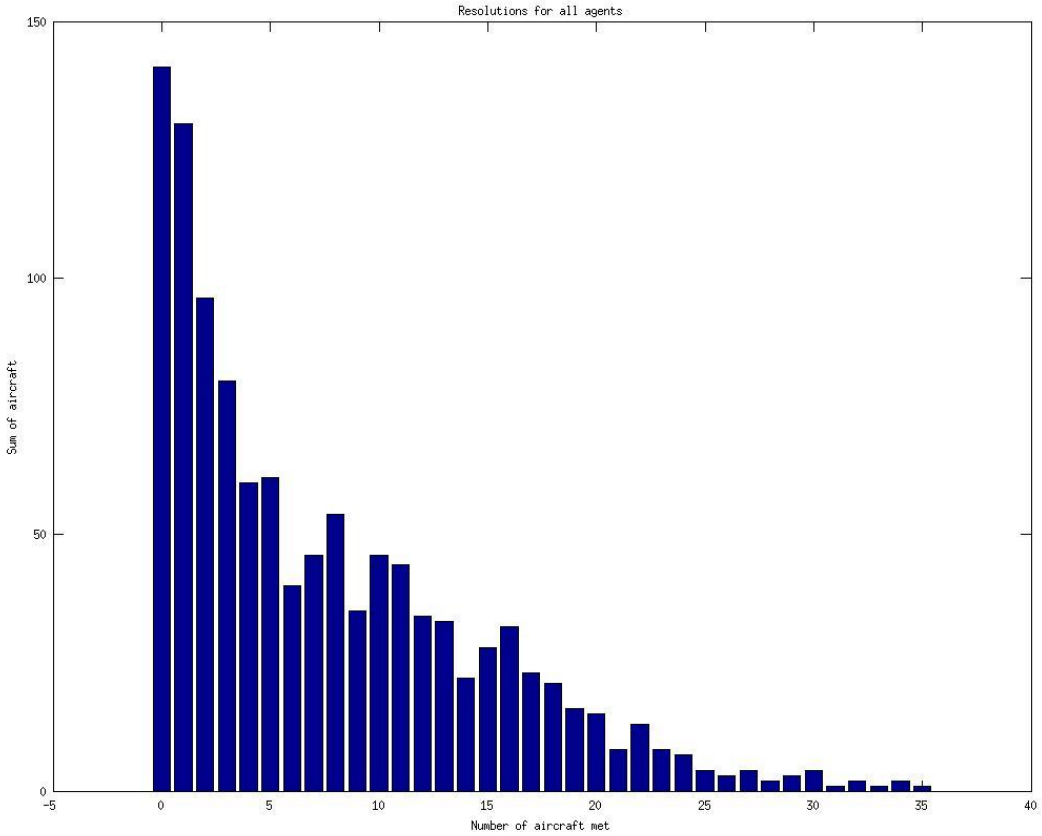
Total number of agents: 705

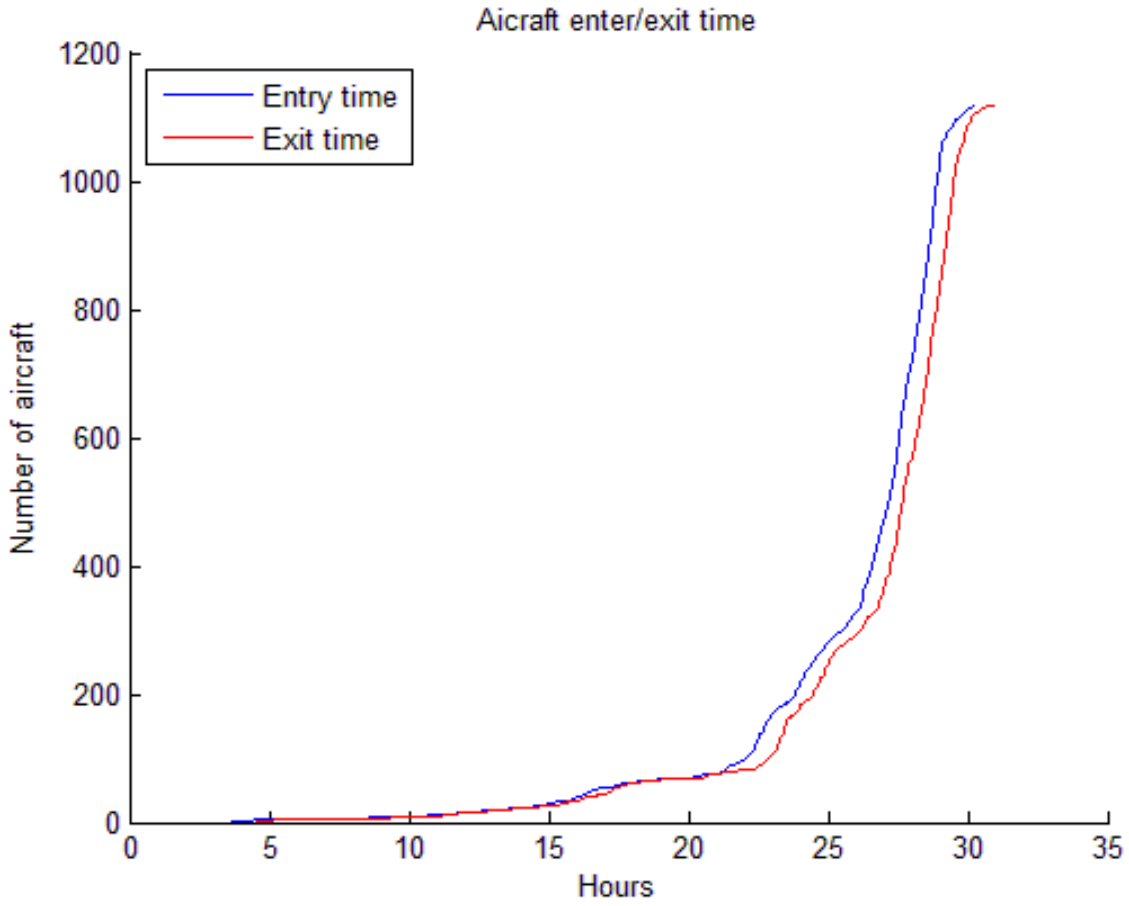
Agents Travelled(mean value): 0.392896 % more of their line route distance
Flight time: 930571.000000 seconds or 258.491944 hours
Flight time(Mean Value): 1319.958865 seconds or 0.366655 hours
Total time in resolution(all agents): 579163.000000 seconds or 160.878611 hours
Mean time in resolution: 821.507801 seconds or 0.228197 hours
Mean value of resolutions for each agent: 2.228369
Average % time in resolution: 62.237379 %
Execution time: 1360.956906 seconds
Flight time/execution time: 683.762282

2.5.2.5.2 Πτήσεις με νότια κατεύθυνση




Number of aircraft flying at the same time
Number of Aircraft
Hours
0
20
40
60
80
100
120
140
160
180
200
5
10
15
20
25
30

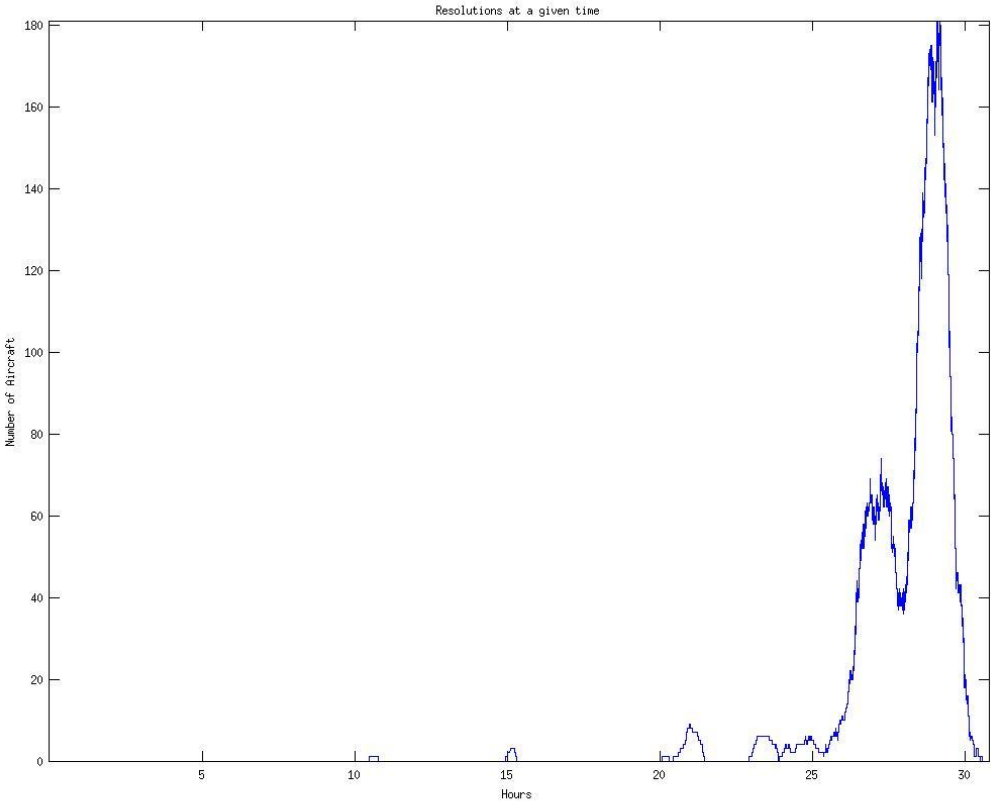

Resolutions for all agents
Sum of aircraft
Number of aircraft met
0
50
100
150
-5
0
5
10
15
20
25
30
35
40

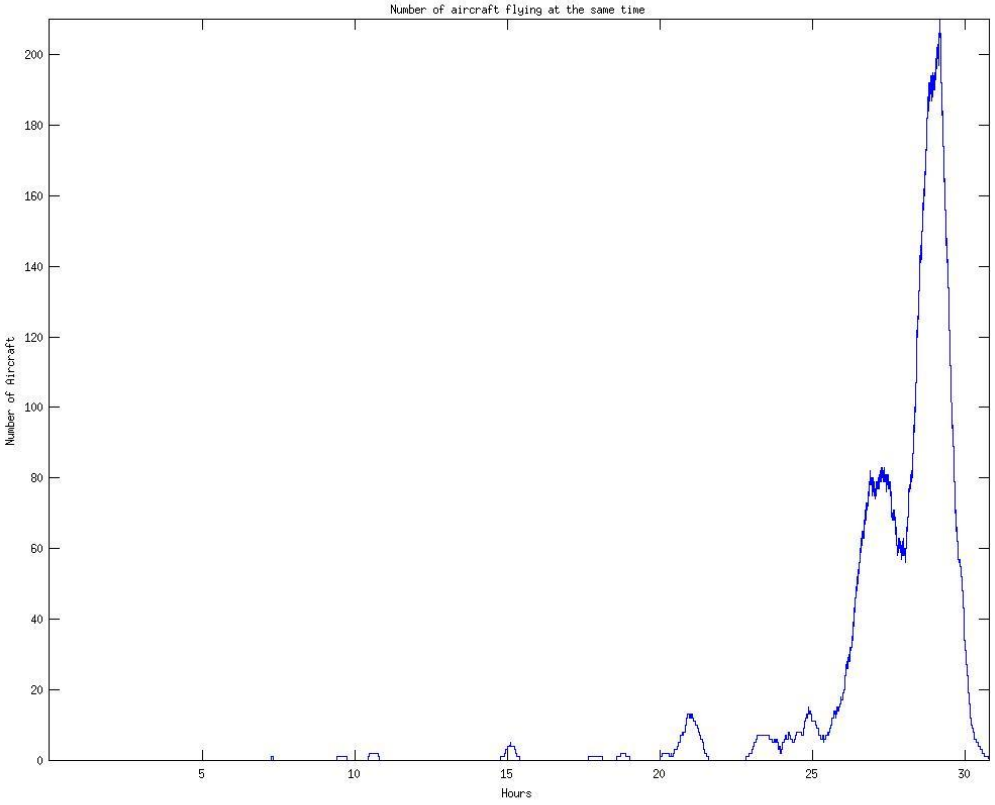


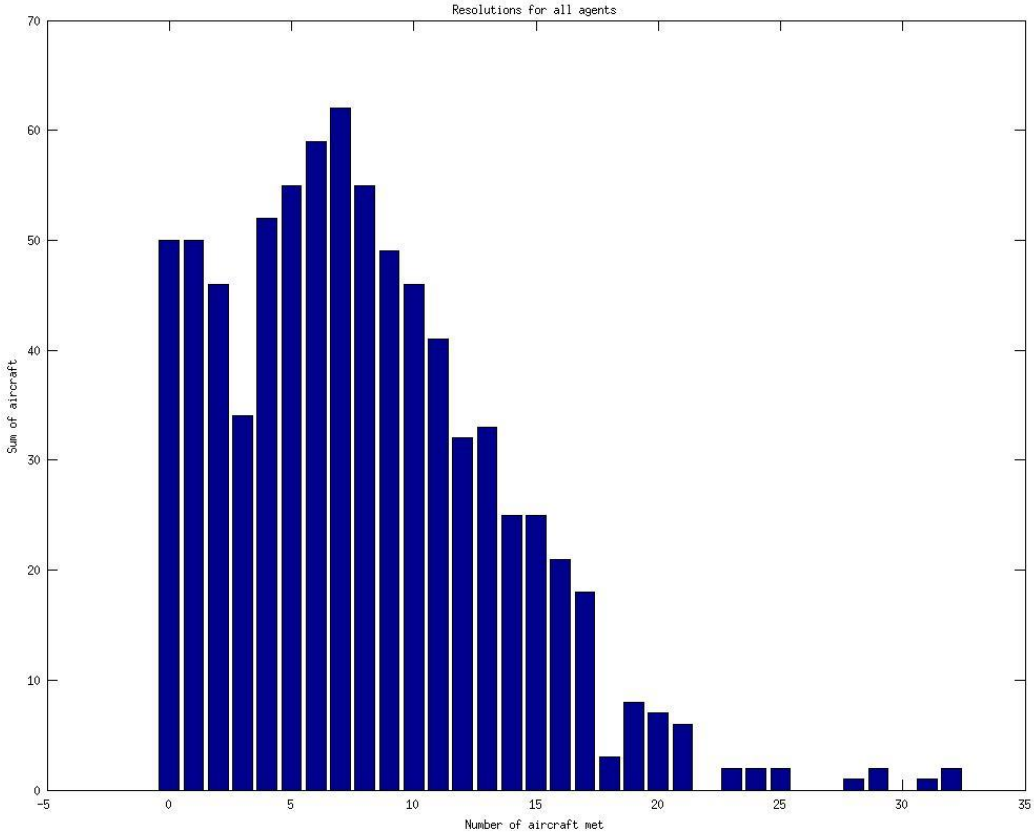
Εικόνες 2.5.2.5.2.1-4

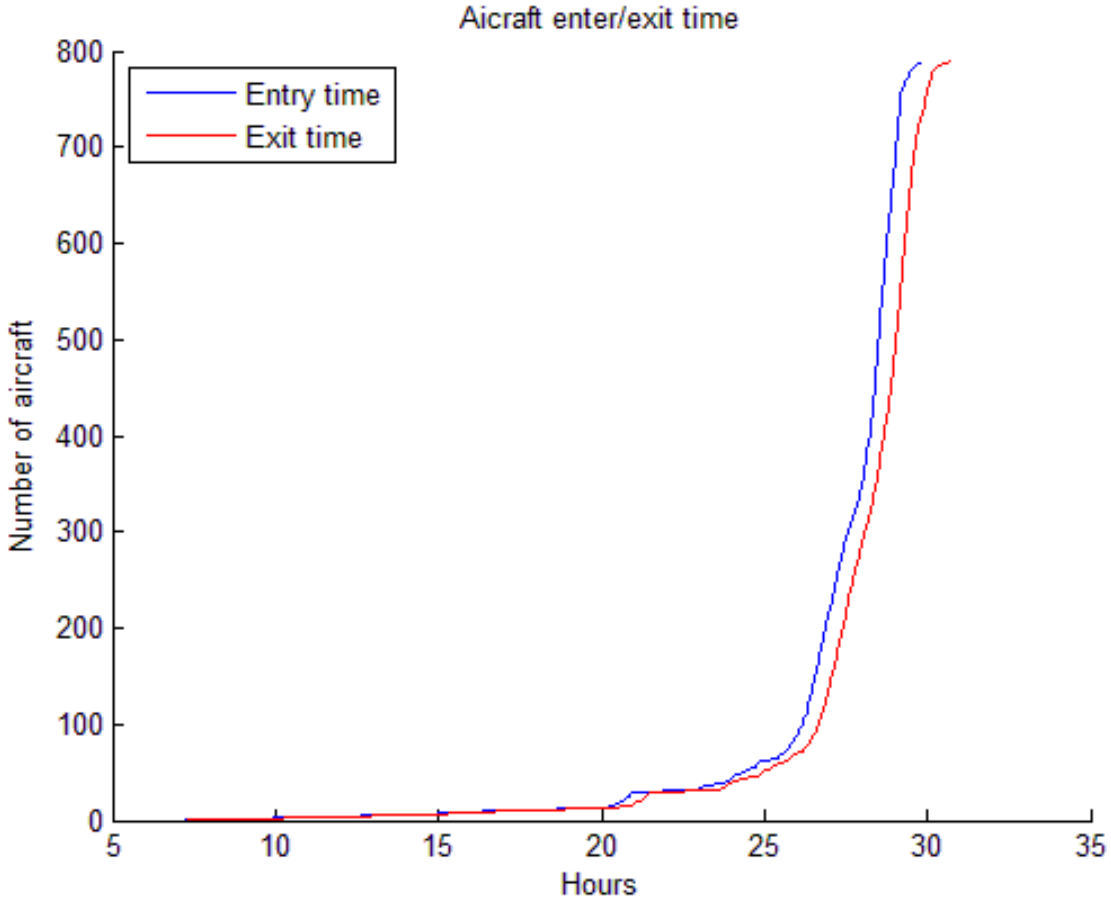
```
                    SIMULATION RESULTS

Total number of agents: 1120


Agents Travelled(mean value): 1.388187 % more of their line route
distance
Flight time: 2431036.000000 seconds or 675.287778 hours
Flight time(Mean Value): 2170.567857 seconds or 0.602936 hours
Total time in resolution(all agents): 1934696.000000 seconds or
537.415556 hours
Mean time in resolution: 1727.407143 seconds or 0.479835 hours
Mean value of resolutions for each agent: 7.496429
Average % time in resolution: 79.583190 %
Execution time: 10975.794446 seconds
Flight time/execution time: 221.490664
```

2.5.2.5.3 Πτήσεις με ανατολική κατεύθυνση









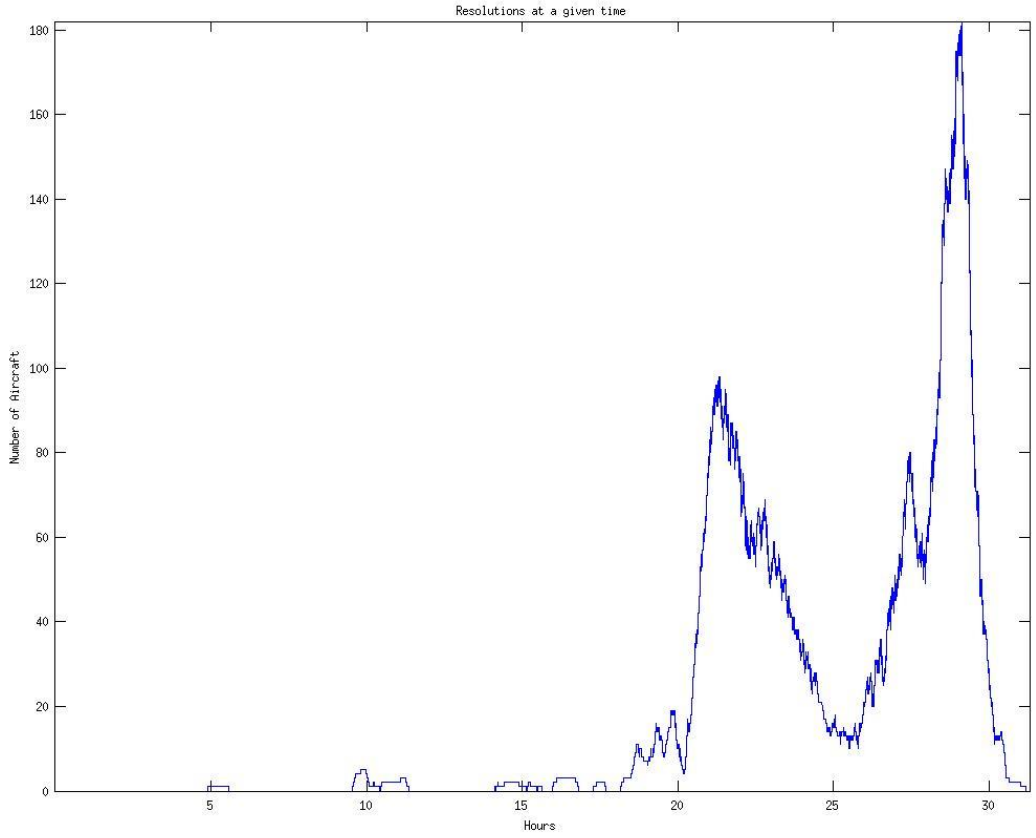
Εικόνες 2.5.2.5.3.1-4

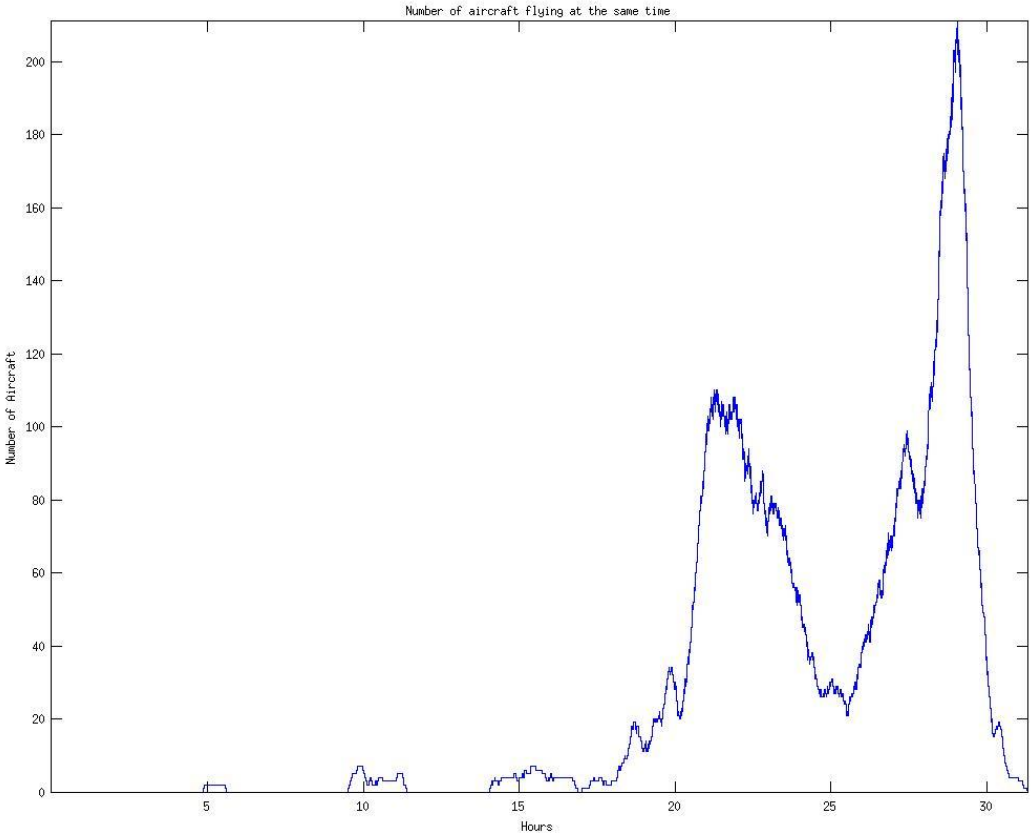
SIMULATION RESULTS

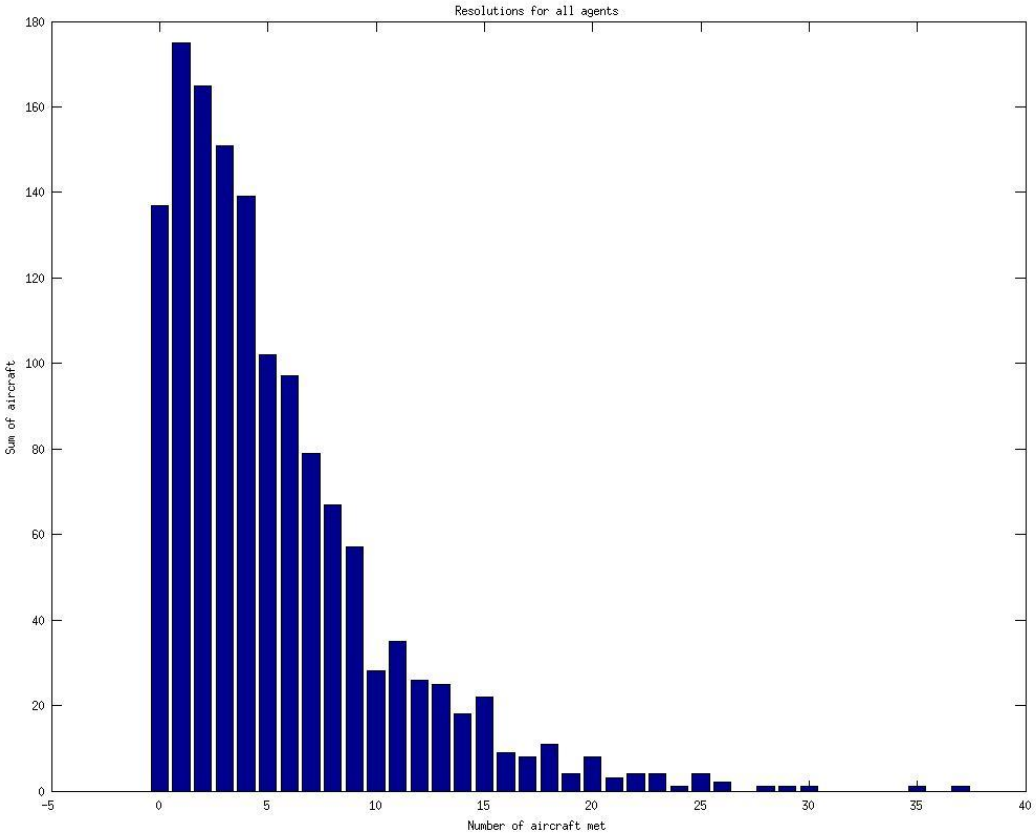
Total number of agents: 789

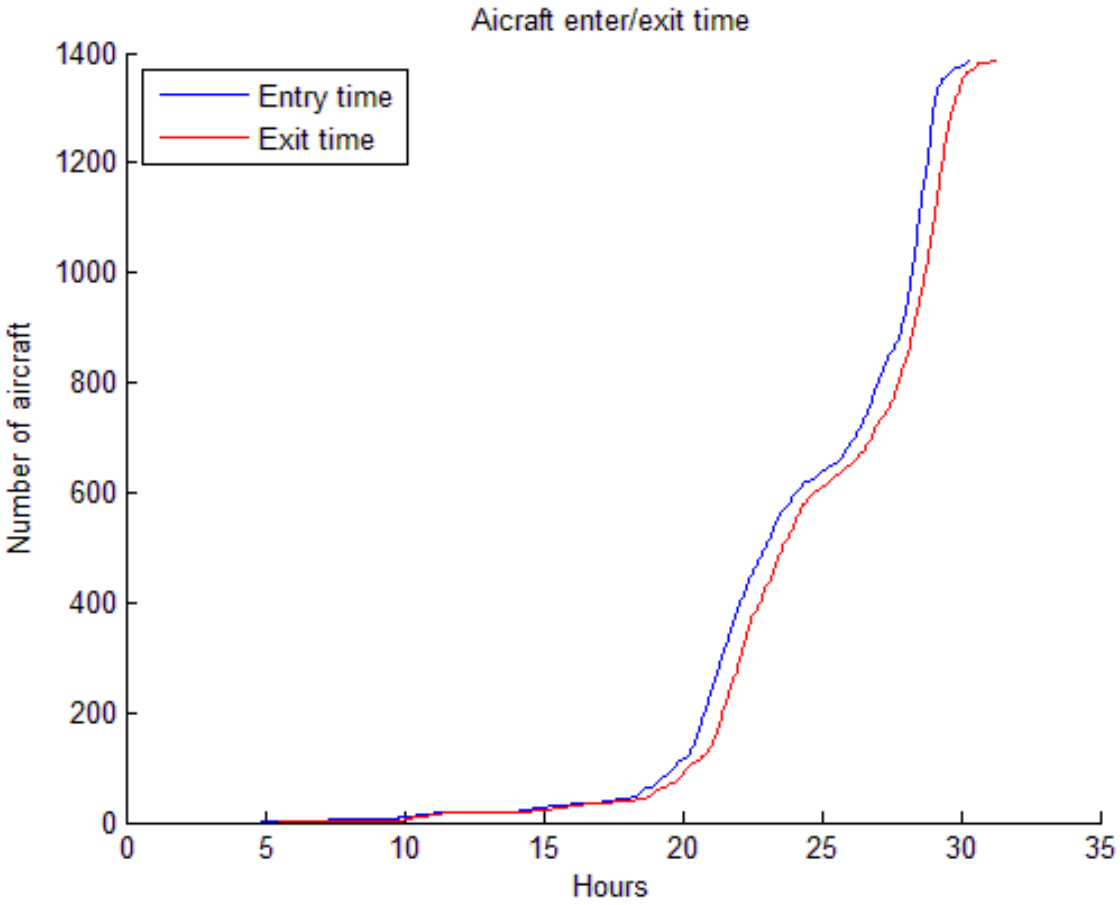
Agents Travelled(mean value): 2.559453 % more of their line route distance
Flight time: 1510453.000000 seconds or 419.570278 hours
Flight time(Mean Value): 1914.389100 seconds or 0.531775 hours
Total time in resolution(all agents): 1301960.000000 seconds or 361.655556 hours
Mean time in resolution: 1650.139417 seconds or 0.458372 hours
Mean value of resolutions for each agent: 7.964512
Average % time in resolution: 86.196658 %
Execution time: 6708.479083 seconds
Flight time/execution time: 225.155804

2.5.2.5.4 Πτήσεις με δυτική κατεύθυνση




Number of aircraft flying at the same time
Number of Aircraft
Hours
0
20
40
60
80
100
120
140
160
180
200
5
10
15
20
25
30


Resolutions for all agents
Sum of aircraft
Number of aircraft met
0
20
40
60
80
100
120
140
160
180
-5
0
5
10
15
20
25
30
35
40



Εικόνες 2.5.2.5.4.1-4

SIMULATION RESULTS

Total number of agents: 1386

Agents Travelled(mean value): 1.496161 % more of their line route distance
Flight time: 3025049.000000 seconds or 840.291389 hours
Flight time(Mean Value): 2182.575036 seconds or 0.606271 hours
Total time in resolution(all agents): 2393439.000000 seconds or 664.844167 hours
Mean time in resolution: 1726.867965 seconds or 0.479686 hours
Mean value of resolutions for each agent: 5.327561
Average % time in resolution: 79.120669 %
Execution time: 12320.168837 seconds
Flight time/execution time: 245.536327

2.5.2.5.5 Σχολιασμός

Οι παραπάνω πτήσεις πια χωρίστηκαν ανά κατεύθυνση. Στις πτήσεις με κατεύθυνση τον βορρά παρατηρούμε ότι δεν είναι τόσο εμφανές το μέγεθος της συνολικής κίνησης κατά τις διάφορες ώρες αφού τα αεροσκάφη είναι λίγα. Παρόλα αυτά παρατηρούμε ότι η κίνηση μετά την ώρα 24 αρχίζει και αυξάνει. Επίσης δεν είναι τόσο εμφανής η αυξομείωση της κίνησης πριν την ώρα 24 στις ανατολικές πτήσης. Στις άλλες δύο περιπτώσεις η αυξομείωση της κίνησης πριν την ώρα 24 είναι πιο εμφανής ιδιαίτερα με τις πτήσεις που πάνε προς τα δυτικά. Πάντως το συμπέρασμα που μπορούμε να βγάλουμε από τα παραπάνω είναι ότι στο δείγμα μας υπάρχει ένα μοτίβο στις πτήσεις και δεν είναι τελείως τυχαία κατανεμημένες με βάση την κατεύθυνση. Το πλεονέκτημα βέβαια αυτού του διαχωρισμού είναι ότι ελαχιστοποιούνται οι περιπτώσεις που τα αεροσκάφη πλησιάζουν από πλάγιες η μετωπικές κατευθύνσεις κάτι το οποίο έχει ως αποτέλεσμα να αναμένουμε μία μείωση στις αποκλίσεις που παρατηρούνται από την πορεία του καθενός. Όπως έχουμε τον παρακάτω πίνακα για να κάνουμε μία σύγκριση των αποτελεσμάτων.

	1000a/c υψηλή κίνηση	**4000a/c επιλεγμένα ανά 4**	**4000a/c επιλεγμένα ανά κατεύθυνση**
Agents Travelled more of their line route distance (mean value):	5.4 %	2.60%	1.46%
Flight time(Mean Value):	0.48 hours	0.52 hours	0.52 hours
Mean time in resolution:	0.44 hours	0.36 hours	0.41 hours
Mean value of resolutions for each agent:	30.54	7.3	5,75
Average % time in resolution:	91 %	68.97%	76.78%
Flight time/execution time:	79.9	~323	~343

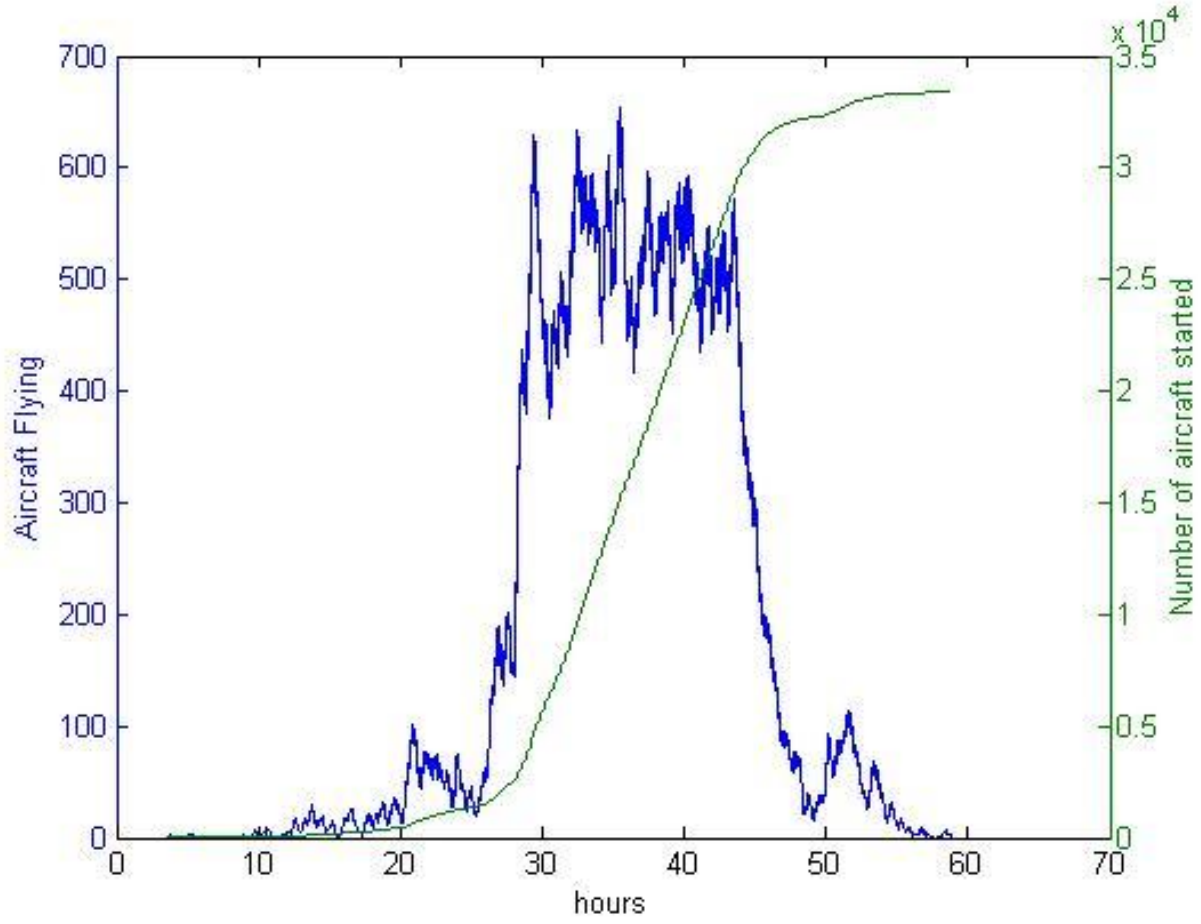
Από τα παραπάνω μπορούμε να συμπεράνουμε τα εξής. Επιλέγοντας τα αεροσκάφη να πετάξουν ανά κατεύθυνση παρατηρούμε ότι για τον ίδιο συνολικό αριθμό αεροσκαφών η απόκλιση από την ευθεία πορεία τους είναι η μικρότερη. Αυτό είναι αναμενόμενο γενικά αφού τα αεροσκάφη τείνουν να πετούν ανά σχηματισμούς και όχι σε τυχαίες κατευθύνσεις. Ο μέσος χρόνος πτήσης προκύπτει βέβαια ίδιος. Επίσης παρατηρούμε ότι τα αεροσκάφη βλέπουν κάποιο άλλο αεροσκάφος μέσα στο εύρος των αισθητήρων τους για περισσότερη ώρα από την περίπτωση που επιλέγουμε αεροσκάφη σε σχέση με την ώρα απογείωσης τους και περνάνε περισσότερο χρόνο προσπαθώντας να "αποφύγουν" κάποιο άλλο αεροσκάφος. Αυτό μπορεί να φαίνεται παράδοξο εκ πρώτης όψεως συγκρινόμενο με τα καλύτερα αποτελέσματα που έχουμε σε σχέση με τις άλλες περιπτώσεις. Ο λόγος που συμβαίνει είναι ο εξής. Τα αεροσκάφη μπαίνουν στον εναέριο χώρο ανά τακτά χρονικά διαστήματα . Πολλά αεροσκάφη στο δείγμα μας εκκινούν από αεροδρόμια προς παρόμοιες κατευθύνσεις. Αυτό συμβαίνει βέβαια γιατί το δείγμα είναι προσαρμοσμένο έτσι ώστε να περνάει

πάνω από την Ελβετία και γενικά την Κεντρική Ευρώπη. Έτσι αεροδρόμια που είναι πολύ δυτικά της Ευρώπης πχ. Και επειδή έχουν επιλεγεί πτήσεις που περνάνε από την Κεντρική Ευρώπη μόνο, οι πτήσεις που ξεκινάνε από αυτά θα έχουν μία κατεύθυνση προς τα ανατολικά. Στο δείγμα παρατηρούμε ότι υπάρχουν πτήσεις οι οποίες ξεκινάνε με μικρές χρονικές διαφορές από τα ίδια αεροδρόμια. Το ίδιο και για βορρά, νότο και ανατολή. Έτσι πολλά αεροσκάφη μπαίνουν στον χώρο σχετικά κοντινές χρονικές στιγμές. Επίσης και κάποια αεροσκάφη μπαίνουν σε κοντινές χρονικές στιγμές για λόγους τυχαίους. Αυτά τα αεροσκάφη λοιπόν που κινούνται προς παρόμοιες κατευθύνσεις και επειδή μπροστά βλέπουν περίπου 50nm αν υπάρχει κάποιο άλλο αεροσκάφος μπροστά τους το πιθανότερο είναι ότι θα είναι στο οπτικό τους πεδίο για το μεγαλύτερο μέρος της πτήσης τους κάτι το οποίο ο αλγόριθμος λαμβάνει σαν resolution γιατί βλέπουν κάποιο άλλο αεροσκάφος παρόλα αυτό δεν αποκλίνουν ιδιαίτερα από την πορεία τους. Αυτό σε σύγκριση με την περίπτωση της κίνησης των αεροσκαφών σε σχέση με τον χρόνο απογείωσης τους κάτι που κάνει ένα αεροσκάφος να ανήκει στην παραπάνω περίπτωση αλλά και να δει κάποια άλλα αεροσκάφη να έρχονται από αντίθετες πορείες κάτι που θα τα κάνει να αλλάξουν την πορεία τους δραστικά αλλά για μικρό χρονικό διάστημα. Αυτός είναι λοιπόν και ο λόγος που χωρίζοντας τα ανά κατεύθυνση έχουμε μικρότερες αποκλίσεις από την πορεία τους αλλά περισσότερο χρόνο σε resolution ενώ παράλληλα συναντάνε και λιγότερα άλλα αεροσκάφη στον δρόμο τους.

2.5.3 Εξομοιώσεις μεγαλύτερου βήματος

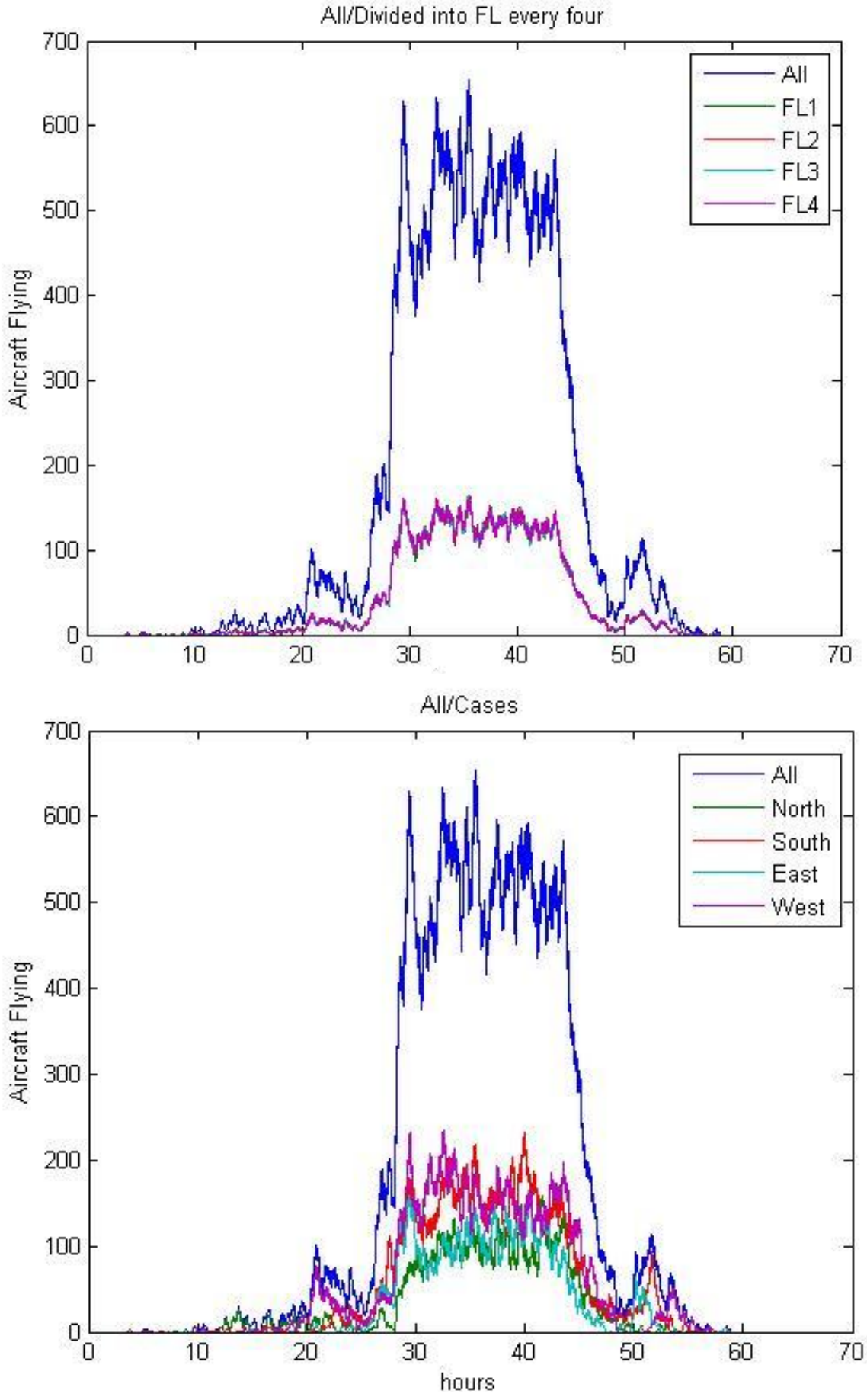
2.5.3.1 Εισαγωγή

Παραπάνω δώσαμε μία σειρά εξομοιώσεων η οποία μας έδωσε κάποια στατιστικά αποτελέσματα από τα οποία βγάλαμε κάποια συμπεράσματα σχετικά με τον διαχωρισμό των αεροσκαφών σε επίπεδα πτήσης έτσι ώστε να έχουμε όσο το δυνατόν μικρότερη απόκλιση από την ευθεία πορεία τους. Γενικά το δείγμα περιέχει περισσότερα από 33000 αεροσκάφη σε ένα χρονικό διάστημα δύο ημερών. Αυτό βάζει κάποια πρακτικά όρια cpu/RAM στον κώδικα καθώς και όρια στο χρονικό διάστημα που αυτός τρέχει. Οι πρώτες εξομοιώσεις έγιναν για ένα χρονικό βήμα ενός δευτερολέπτου. Αυτό είναι ένα καλό βήμα για να μπορέσουμε να βγάλουμε κάποια σχετικά ακριβή μεγέθη που αφορούν τις εξομοιώσεις. Οι πρώτες εξομοιώσεις έγιναν ουσιαστικά στον χρονικό ορίζοντα της πρώτης ημέρας και περιλαμβάνουν λίγο από την αρχή της δεύτερης όπου και η κίνηση αυξάνει εμφανώς στο τέλος των διαγραμμάτων. Παρακάτω δίνουμε μία σειρά εξομοιώσεων με βήμα δώδεκα δευτερολέπτων. Τα διαγράμματα που θα δώσουμε λοιπόν είναι προσεγγιστικά αλλά είναι αρκετά κοντά στα πραγματικά.



Εικόνες 2.5.3.1.1

Στο παραπάνω διάγραμμα δίνουμε μία καθαρή εκτίμηση του δείγματος με βάση μόνο την ώρα εισόδου και εξόδου του κάθε αεροσκάφους από τον εναέριο χώρο. Οι ώρες αυτές υπολογίζονται μόνο με βάση την ονομαστική ταχύτητα του αεροσκάφους και ότι ακολουθεί μόνο την ευθεία του πορεία. Έτσι παίρνουμε την παραπάνω εκτίμηση των πτήσεων που πετάνε την ίδια στιγμή τις δύο ημέρες. Βλέπουμε ότι υπάρχει μία δραστηριότητα σχετικά μικρή την πρώτη ημέρα η οποία αυξάνεται εμφανώς μετά την ώρα 24. Η μεγαλύτερη δραστηριότητα βρίσκεται από περίπου την ώρα 29 έως την ώρα 48. Με την πράσινη καμπύλη δίνουμε την τον αριθμό των αεροσκαφών που έχουν απογειωθεί μέχρι αυτή την στιγμή η οποία αυξάνει περίπου μέχρι τον αριθμό περίπου 33000. Η κλίση της καμπύλης είναι επίσης ένα μέτρο του ρυθμού απογείωσης των αεροσκαφών η οποία παρατηρούμε ότι αυξάνει μεταξύ τις ώρες υψηλής κίνησης. Ο μέγιστος αριθμός των αεροσκαφών εν πτήση ξεπερνάει κάποιες χρονικές περιόδους και τα 600.



Εικόνες 2.5.3.1.2-3

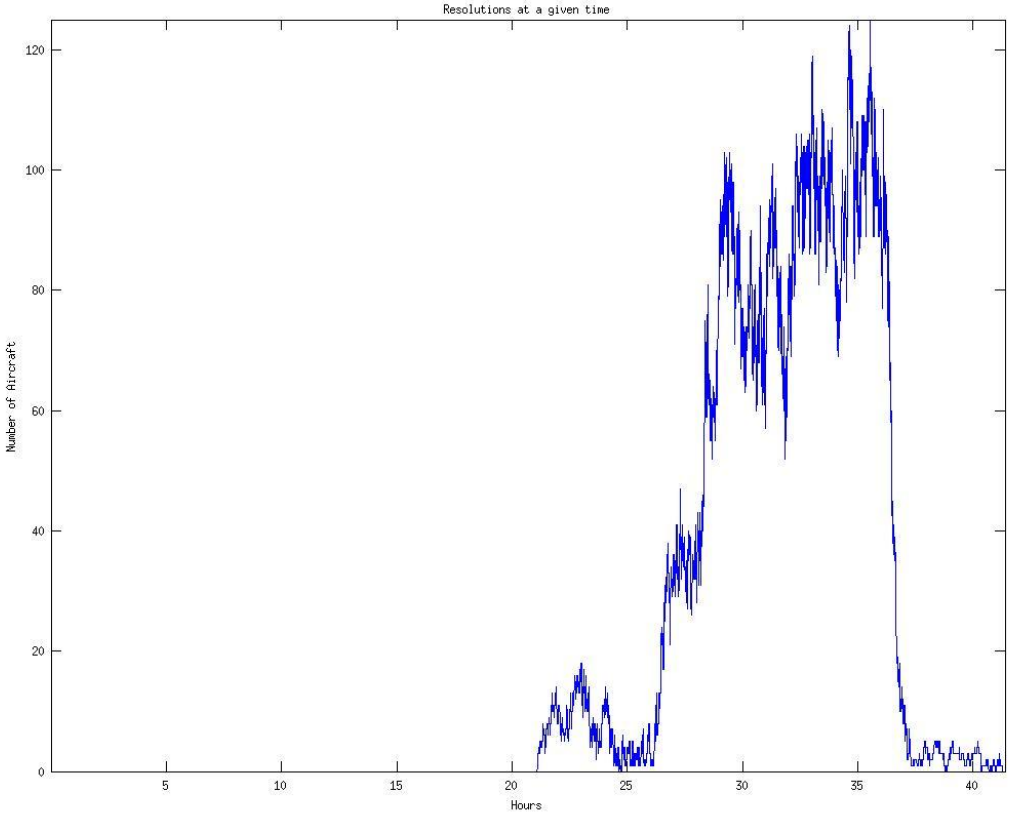
Στις παραπάνω δύο εικόνες παρουσιάζουμε το ίδιο διάγραμμα με την εικόνα 2.5.3.1.1 ενώ παρουσιάζουμε και τις επί μέρους εκτιμήσεις του αριθμού των αεροσκαφών που βρίσκονται εν πτήση την ίδια χρονική στιγμή χωρίζοντας τους ανάλογα με την ώρα απογείωσης τους και ανάλογα με την κατεύθυνσή τους. Παρατηρούμε ότι ανάλογα με την ώρα απογείωσης τους τα αεροσκάφη χωρίζονται πιο ομοιόμορφα με τις μέγιστες πτήσης να μην ξεπερνούν τις 200 για όλες τις περιπτώσεις. Αντίθετα στον διαχωρισμό τους με βάση την κατεύθυνση το δείγμα είναι περισσότερο ακανόνιστο και κάποιες χρονικές στιγμές ο αριθμός των αεροσκαφών ξεπερνάει τα 200. Ακολουθούν κάποιες εξομοιώσεις οι οποίες τρέχουν σε ένα διάστημα της περιόδου της υψηλής κίνησης. Ο σκοπός τους είναι να εξομοιώσουν τις περιπτώσεις οι οποίες προτείνονται από τα παραπάνω αποτελέσματα δηλαδή της περίπτωσης του διαχωρισμού των αεροσκαφών σε τέσσερα διαφορετικά επίπεδα πτήσης. Στα παρακάτω αποτελέσματα δεν θα δώσουμε την απόκλιση από την ευθεία πορεία. Αυτή δόθηκε στα στις παραπάνω εξομοιώσεις και είναι αυτή που θα χρησιμοποιηθεί χρησιμοποιήθηκε σαν μέτρο σύγκρισης μεταξύ των εξομοιώσεων. Ο λόγος που δεν θα δώσουμε αυτό το μέγεθος είναι ότι δεν θα είναι πλέον ιδιαίτερα ακριβές αφού ένα βήμα δώδεκα δευτερολέπτων είναι σχετικά μεγάλο για τέτοιου είδους εκτιμήσεις. Επίσης το μέγεθος Flight time/Execution Time δεν δίνεται αφού φυσικά για μεγαλύτερο βήμα αναμένεται αρκετά μεγαλύτερο. Τα δύο αυτά μεγέθη τα εκτιμάμε χρησιμοποιώντας τις παραπάνω εξομοιώσεις όπου και η ακρίβεια είναι καλύτερη για την εξαγωγή τους. Όσο αναφορά τα υπόλοιπα κρίνεται ότι μπορούν να χρησιμοποιηθούν ως ένας δείκτης αποτίμησης των δύο περιπτώσεων που ακολουθούν αφού είναι περισσότερο ανθεκτικά στην αλλαγή του βήματος.

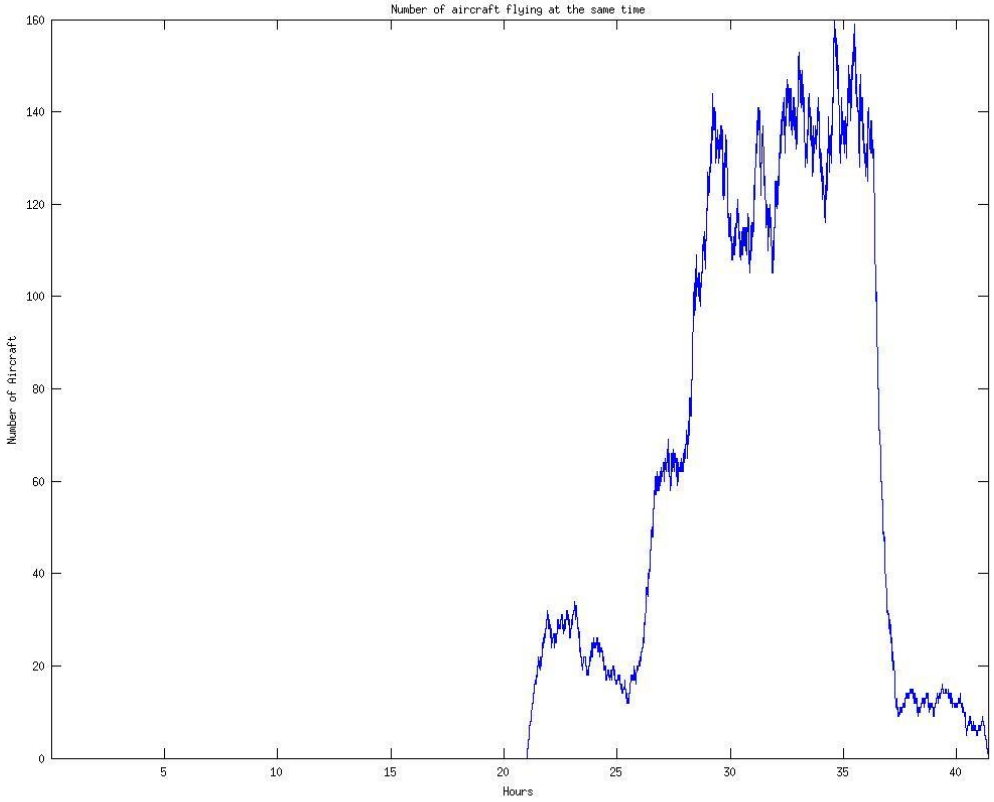
2.5.3.2 Σενάριο 16000 αεροσκαφών με αρχή το αεροσκάφος 701 με επιλογή ανά 4

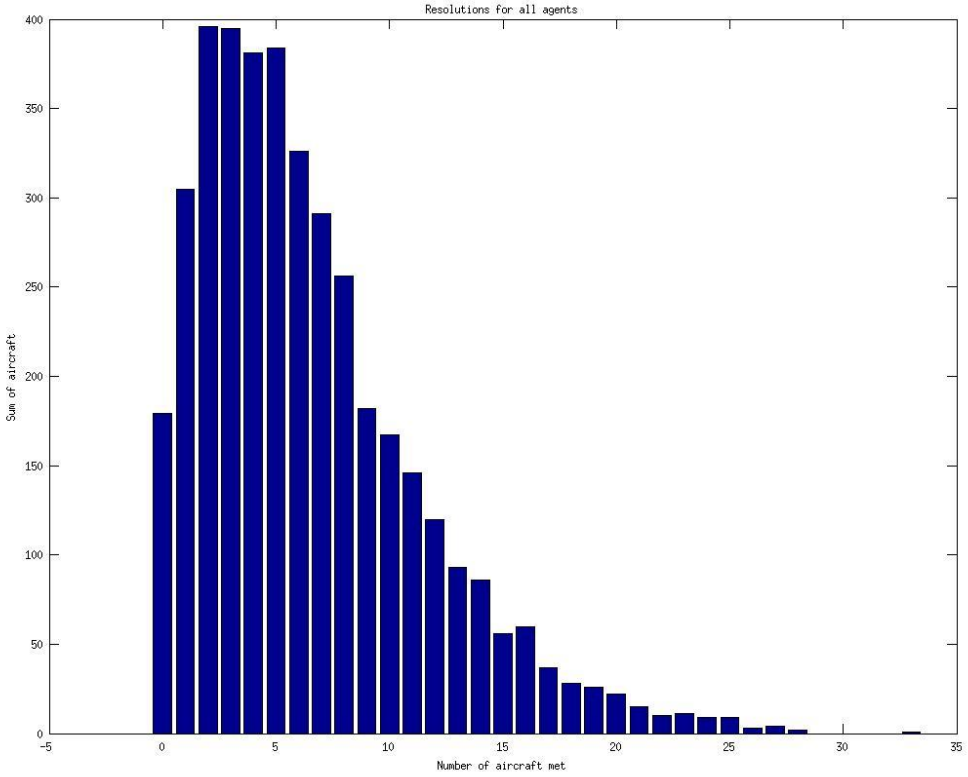
Σε αυτό το σενάριο επιλέγουμε σαν αρχή το αεροσκάφος 700 έτσι ώστε να ξεκινήσουμε από το τέλος της πρώτης ημέρας και να έχουμε μία εικόνα της κίνησης όταν εισέλθουμε στην δεύτερη ημέρα. Τα αεροσκάφη ανά περίπτωση είναι 4000 ανά περίπτωση οπότε και καλύπτουν ένα εύρος 16000 αεροσκαφών του δείγματος. Εδώ όπως και στην 2.5.2.4 χωρίζουμε τα αεροσκάφη ανάλογα με την ώρα απογείωσης τους ομοιόμορφα σε τέσσερα επίπεδα πτήσης.

2.5.3.2.1 Ξεκινώντας από το αεροσκάφος 701

Έχουμε τα παρακάτω διαγράμματα


Resolutions at a given time
Number of Aircraft
Hours
0
20
40
60
80
100
120
5
10
15
20
25
30
35
40


Number of aircraft flying at the same time
Number of Aircraft
Hours
0
20
40
60
80
100
120
140
160
5
10
15
20
25
30
35
40



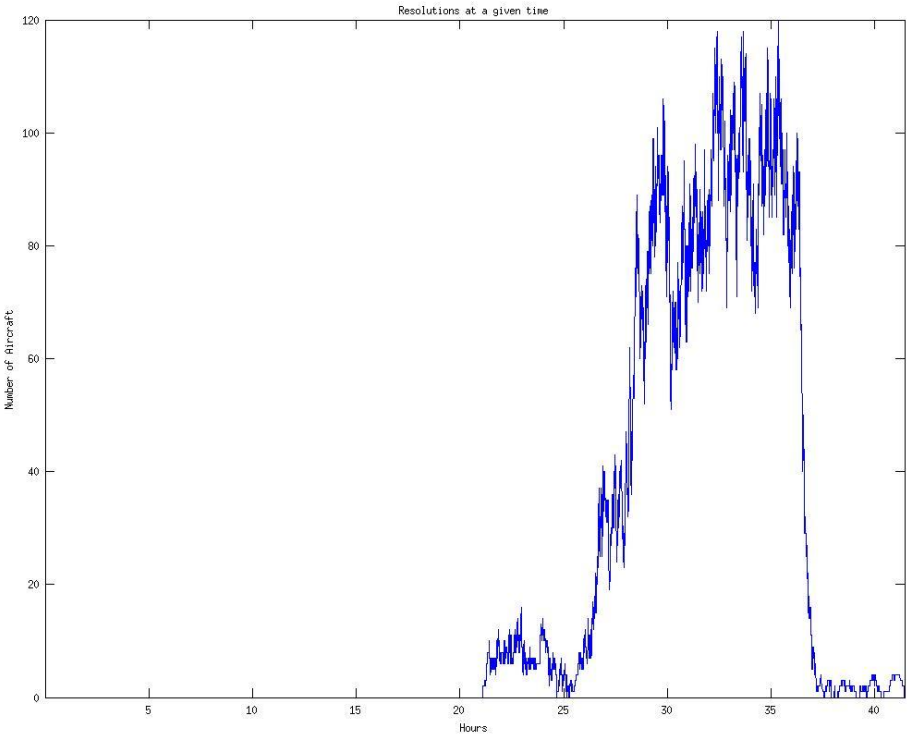
Εικόνες 2.5.3.2.1.1-3

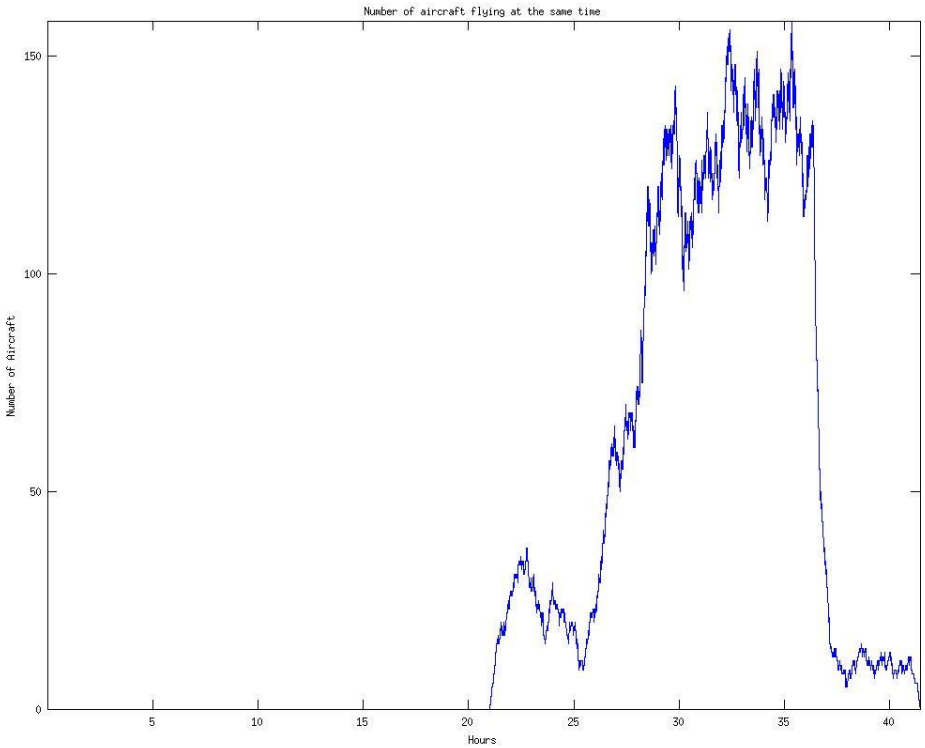
SIMULATION RESULTS

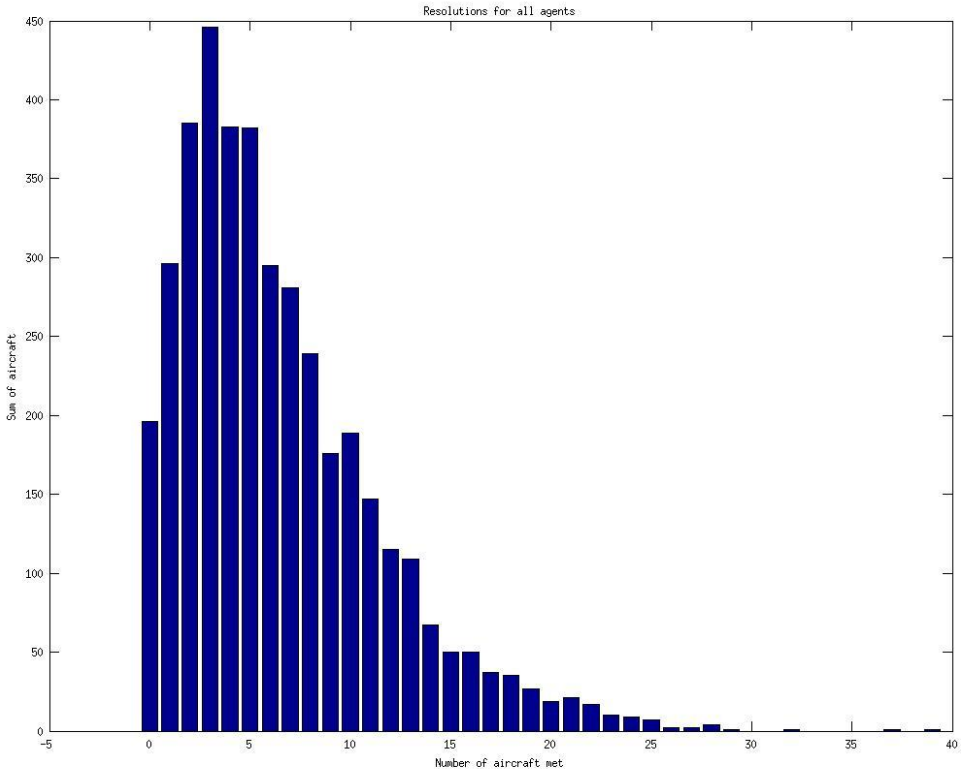
Total number of agents: 4000

Flight time: 4971876.000000 seconds or 1381.076667 hours
Flight time(Mean Value): 1242.969000 seconds or 0.345269 hours
Total time in resolution(all agents): 3951636.000000 seconds
or 1097.676667 hours
Mean time in resolution: 987.909000 seconds or 0.274419 hours
Mean value of resolutions for each agent: 6.476250
Average % time in resolution: 79.479778 %

2.5.3.2.2 Ξεκινώντας από το αεροσκάφος 702







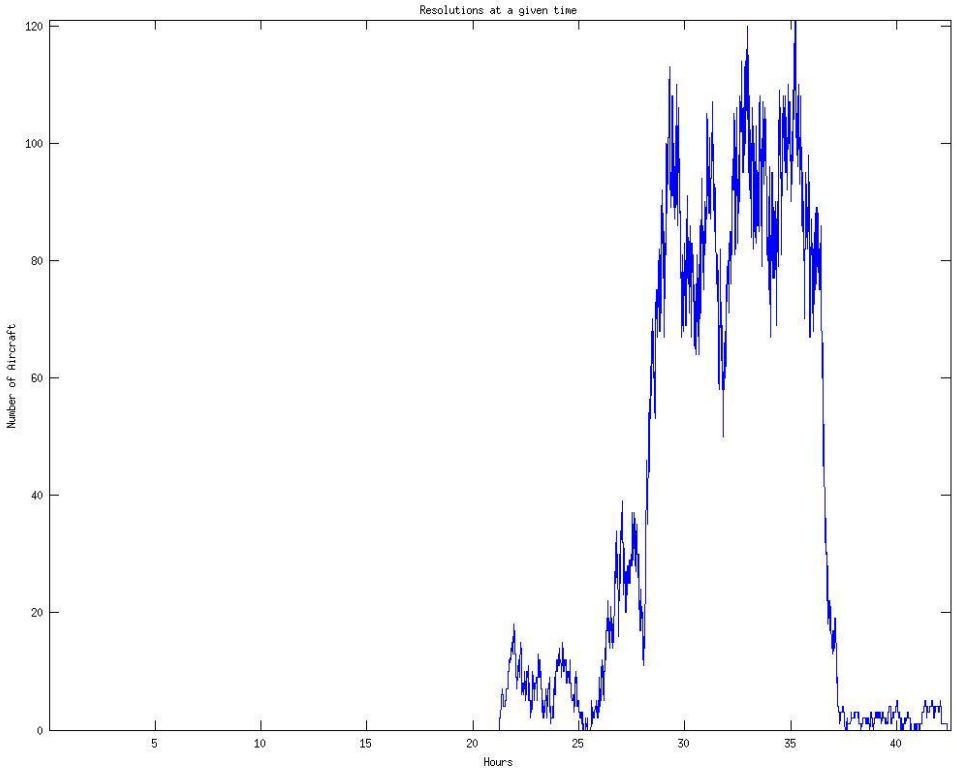
Εικόνες 2.5.3.2.2.1-3

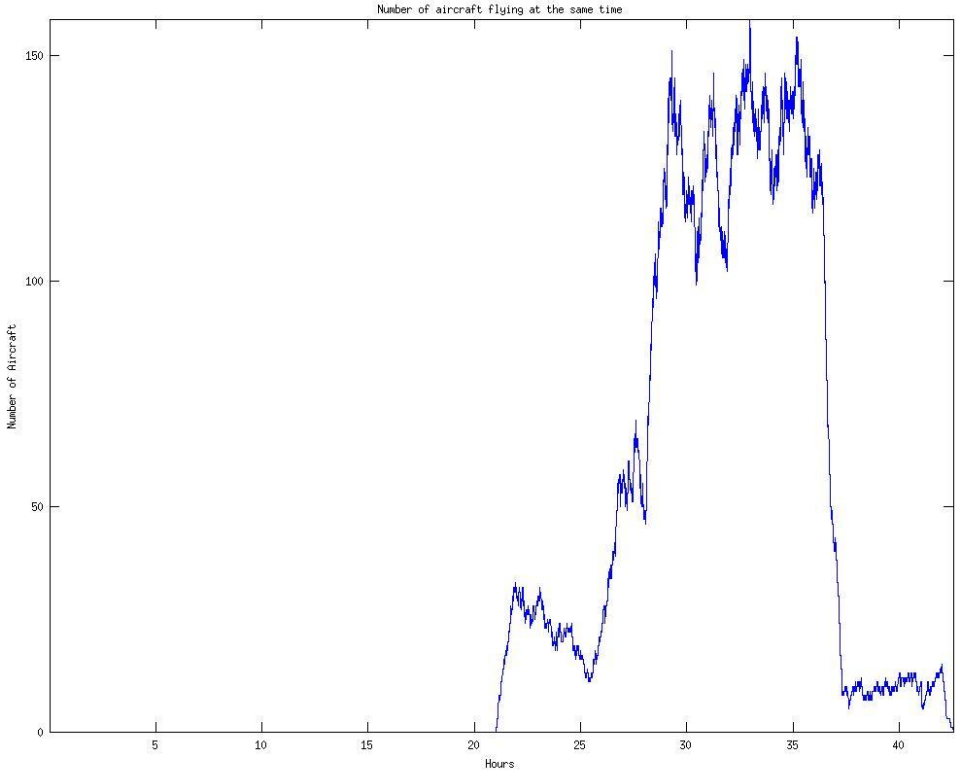
SIMULATION RESULTS

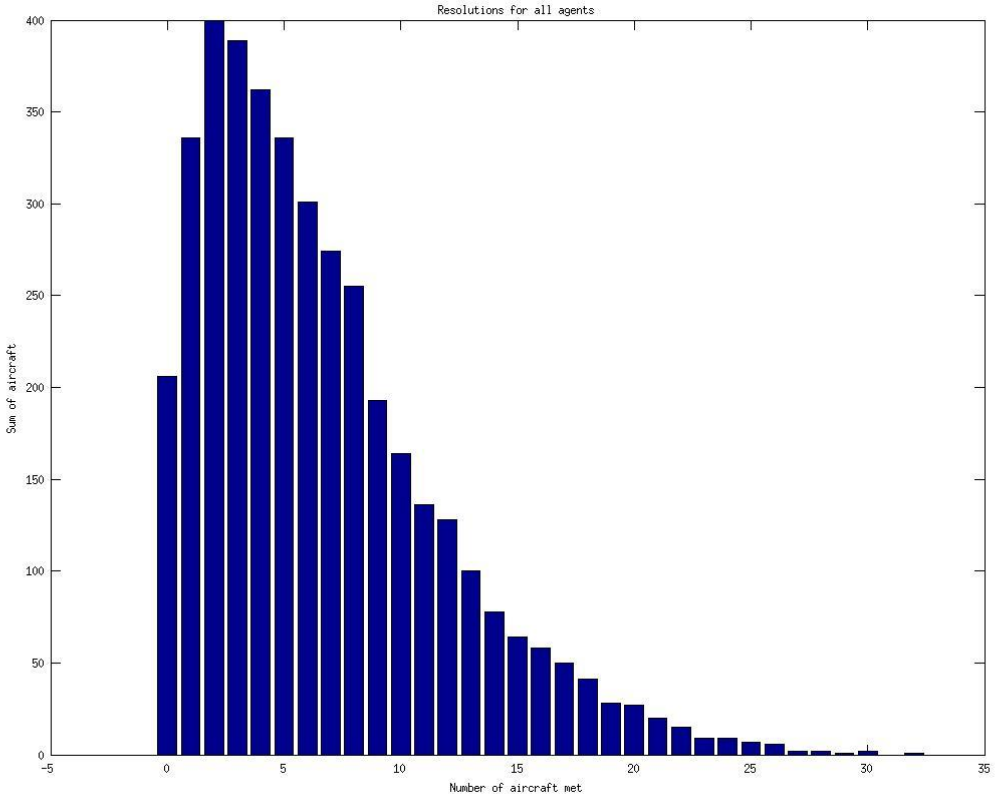
Total number of agents: 4000

Flight time: 4905192.000000 seconds or 1362.553333 hours
Flight time(Mean Value): 1226.298000 seconds or 0.340638 hours
Total time in resolution(all agents): 3880020.000000 seconds or 1077.783333 hours
Mean time in resolution: 970.005000 seconds or 0.269446 hours
Mean value of resolutions for each agent: 6.453500
Average % time in resolution: 79.100268 %

2.5.3.2.3 Ξεκινώντας από το αεροσκάφος 703







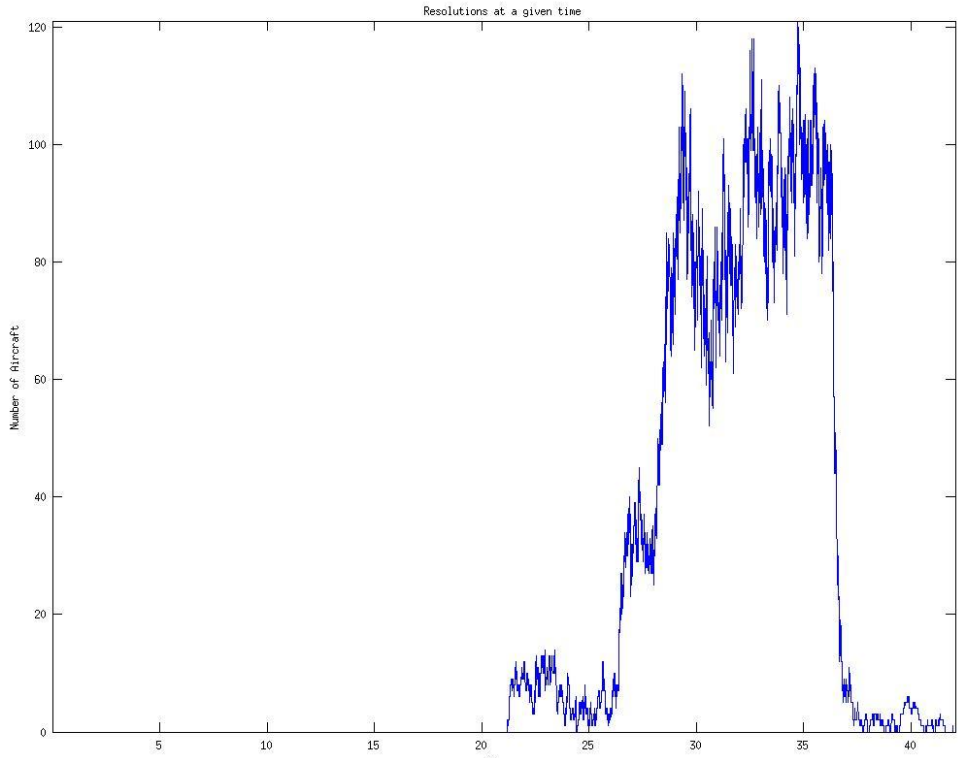
Εικόνες 2.5.3.2.3.1-3

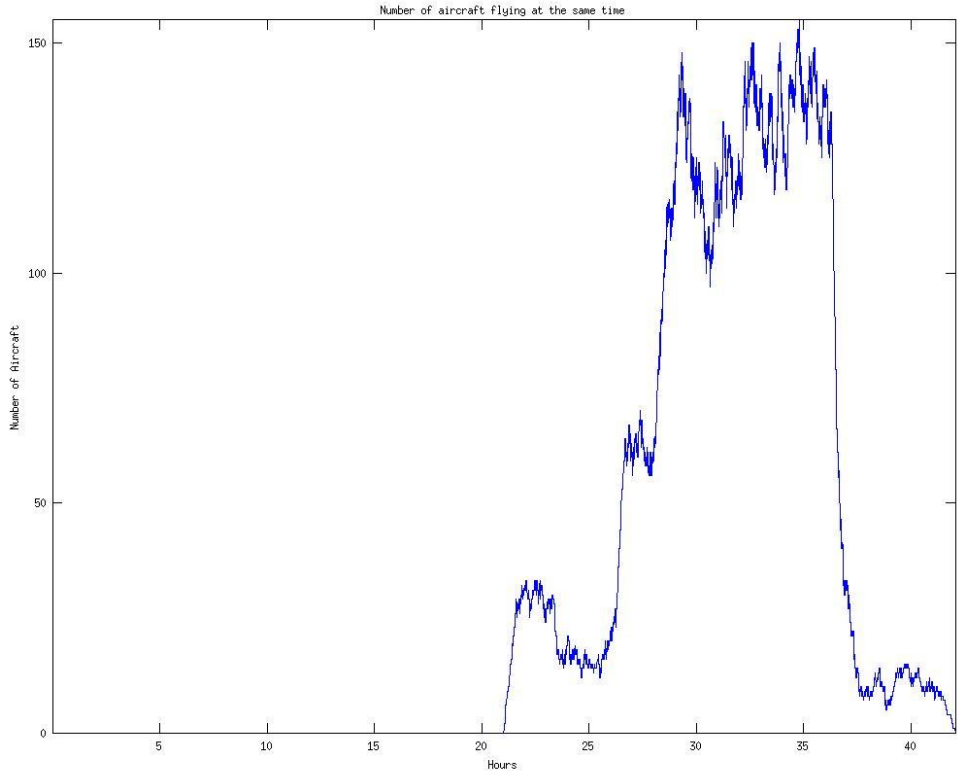
SIMULATION RESULTS

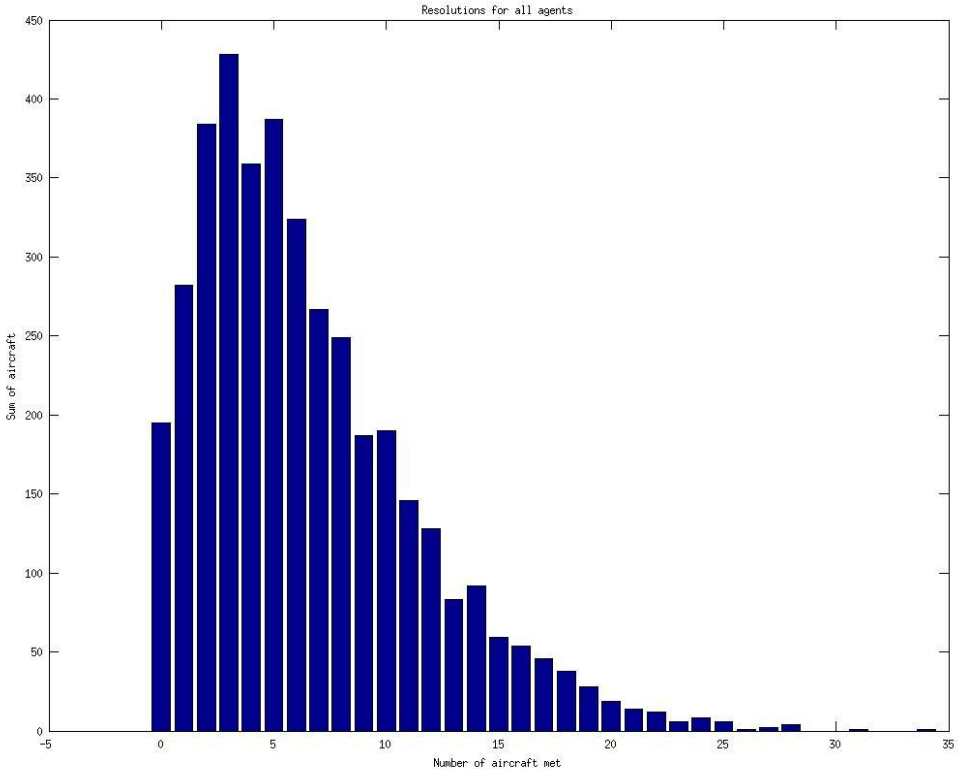
Total number of agents: 4000

Flight time: 4920312.000000 seconds or 1366.753333 hours
Flight time(Mean Value): 1230.078000 seconds or 0.341688 hours
Total time in resolution(all agents): 3889248.000000 seconds or 1080.346667 hours
Mean time in resolution: 972.312000 seconds or 0.270087 hours
Mean value of resolutions for each agent: 6.569750
Average % time in resolution: 79.044744 %

2.5.3.2.4 Ξεκινώντας από το αεροσκάφος 704







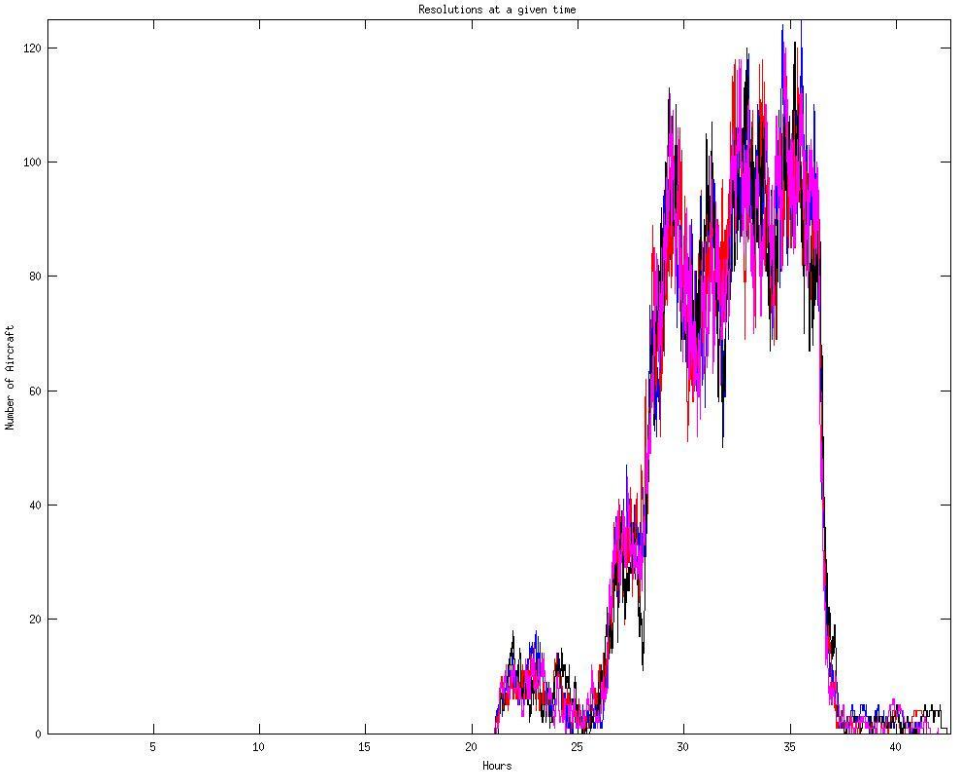
Εικόνες 2.5.3.2.4.1-3

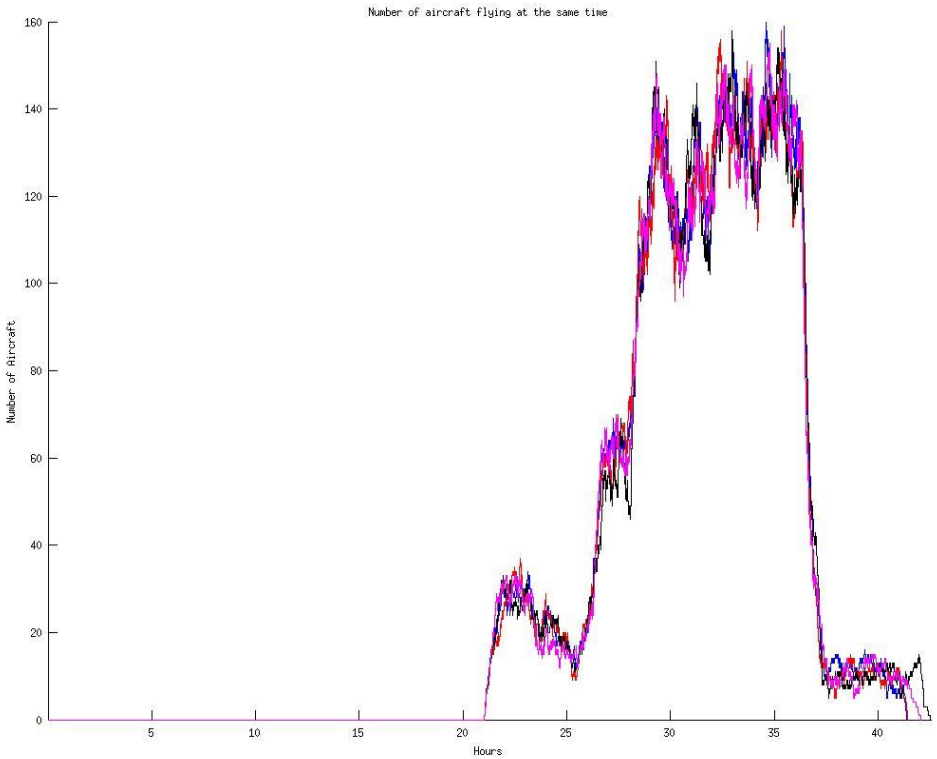
SIMULATION RESULTS

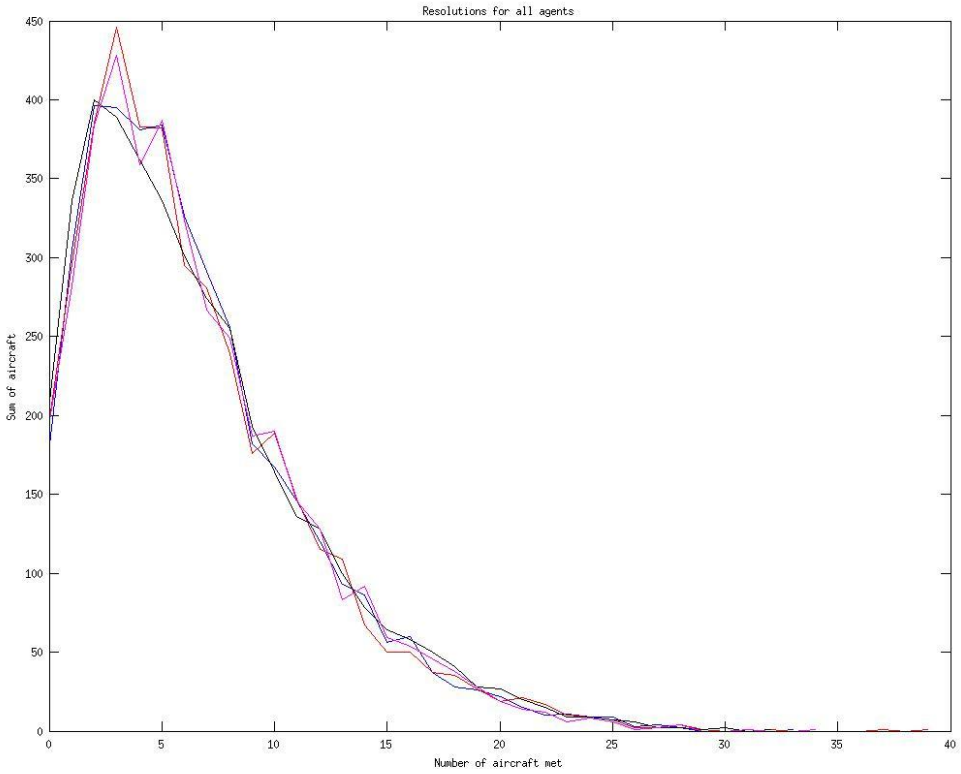
Total number of agents: 4000

Flight time: 4916604.000000 seconds or 1365.723333 hours
Flight time(Mean Value): 1229.151000 seconds or 0.341431 hours
Total time in resolution(all agents): 3883200.000000 seconds or 1078.666667 hours
Mean time in resolution: 970.800000 seconds or 0.269667 hours
Mean value of resolutions for each agent: 6.506000
Average % time in resolution: 78.981346 %

2.5.3.2.5 Συνολική παρουσίαση αποτελεσμάτων







Εικόνες 2.5.3.2.5.1-3

Παραπάνω παρουσιάζουμε τα αποτελέσματα συνοπτικά. Βλέπουμε ότι τα διαγράμματα δεν έχουν ιδιαίτερες διαφορές κάτι που ήταν αναμενόμενο με βάση των διαχωρισμό των αεροσκαφών.

	Start 701	**Start 702**	**Start 703**	**Start 704**	**Mean Value**
Mean time in resolution:	0.2744	0.2694	0.2700	0.2696	0.2709
Mean value of resolutions for each agent:	6.4762	6.4535	6.5697	6.5060	6.5014
Average % time in resolution:	79.4797 %	79.1002%	79.0447 %	78.9813 %	79.1515%

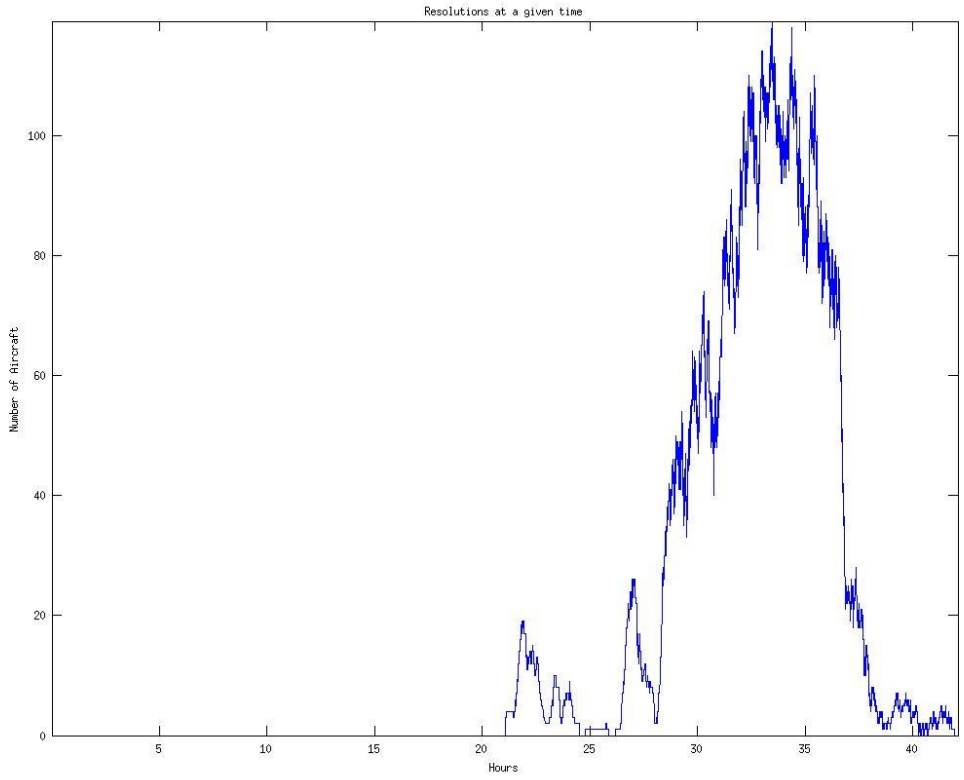
Στον παραπάνω πίνακα δίνουμε τα αποτελέσματα μαζί με την μέση τιμή τους. Όλα τα αποτελέσματα είναι σχετικά κοντά. Στις εξομοιώσεις βλέπουμε ότι υπάρχουν στιγμές κατά τις οποίες πετάνε περισσότερα από 150 αεροσκάφη την ίδια στιγμή. Παρακάτω δίνουμε και την επόμενη σειρά εξομοιώσεων και κατόπιν ακολουθεί σχολιασμός των αποτελεσμάτων.

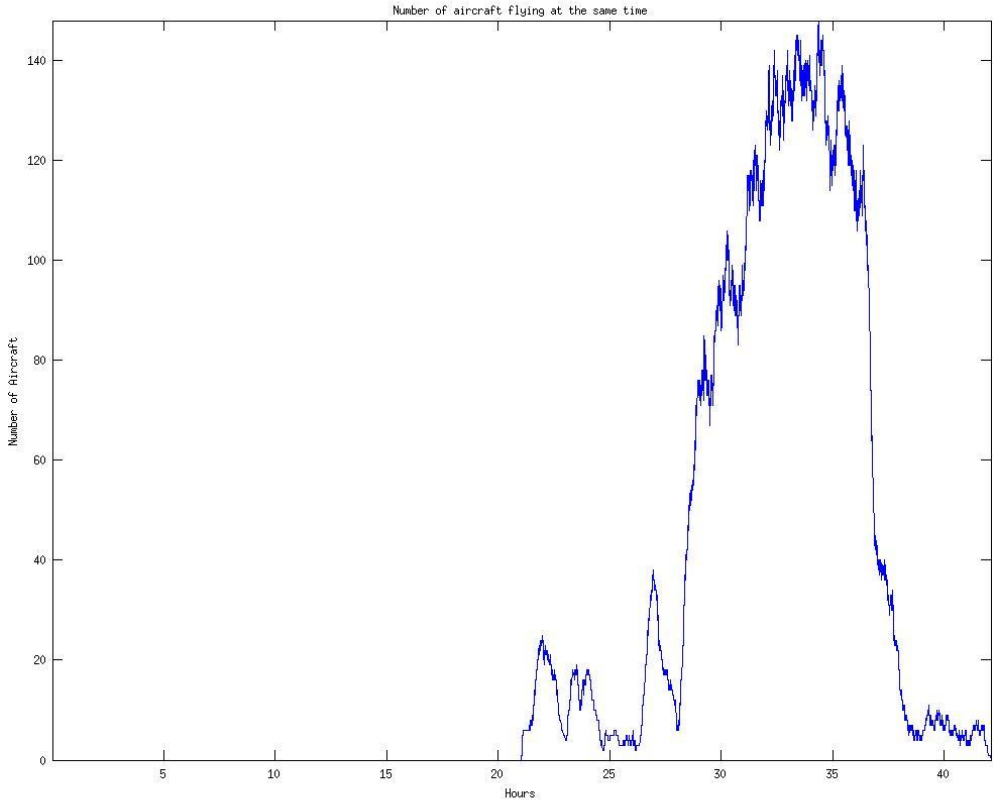
2.5.3.3 Σενάριο 16000 χιλιάδων αεροσκαφών χωρισμένων ανά κατεύθυνση

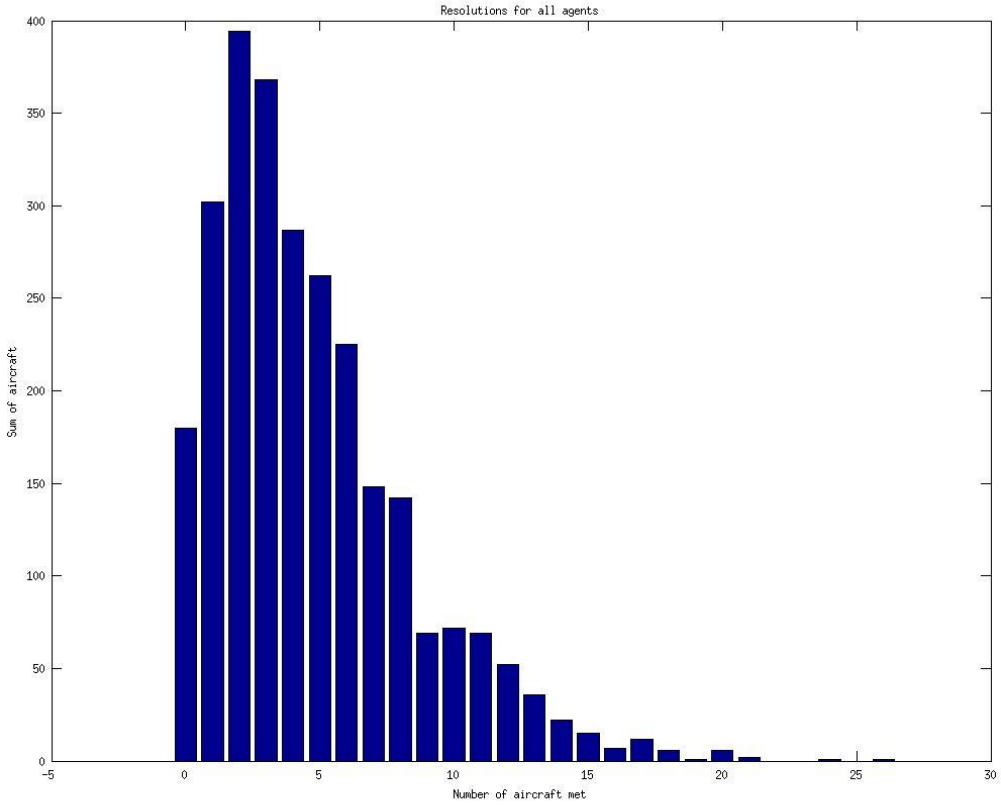
Σε αυτήν την σειρά εξομοιώσεων όπως και στην 2.5.2.5 τα αεροσκάφη χωρίζονται με βάση την κατεύθυνση τους. Από τα 16000 αεροσκάφη επιλέγονται αυτά που πετάνε στις αντίστοιχες κατευθύνσεις και γίνεται η εξομοίωση. Παρακάτω δίνουμε όλα τα αποτελέσματα και ακολουθεί σχολιασμός τους.

Βόρεια: 2679/16000
Νότια: 4857/16000
Ανατολικά: 3212/16000
Δυτικά: 5252/16000

2.5.3.3.1 Πτήσεις με βόρεια κατεύθυνση







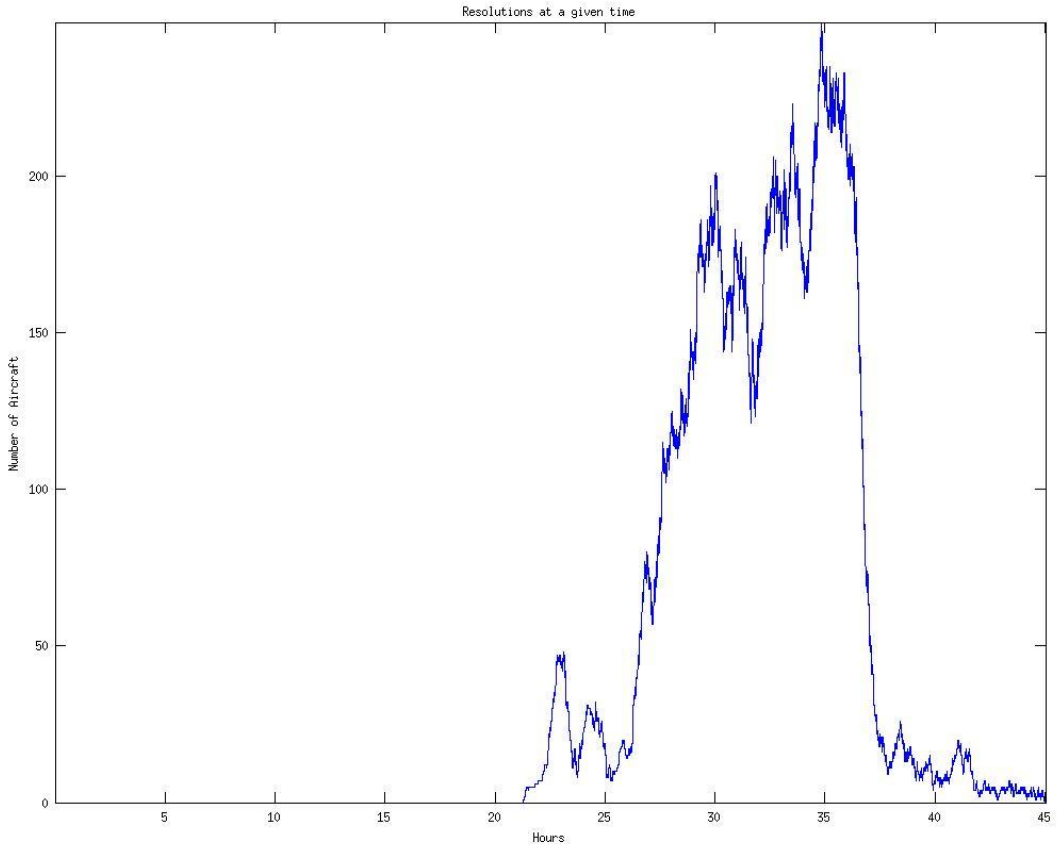
Εικόνες 2.5.3.3.1.1-3

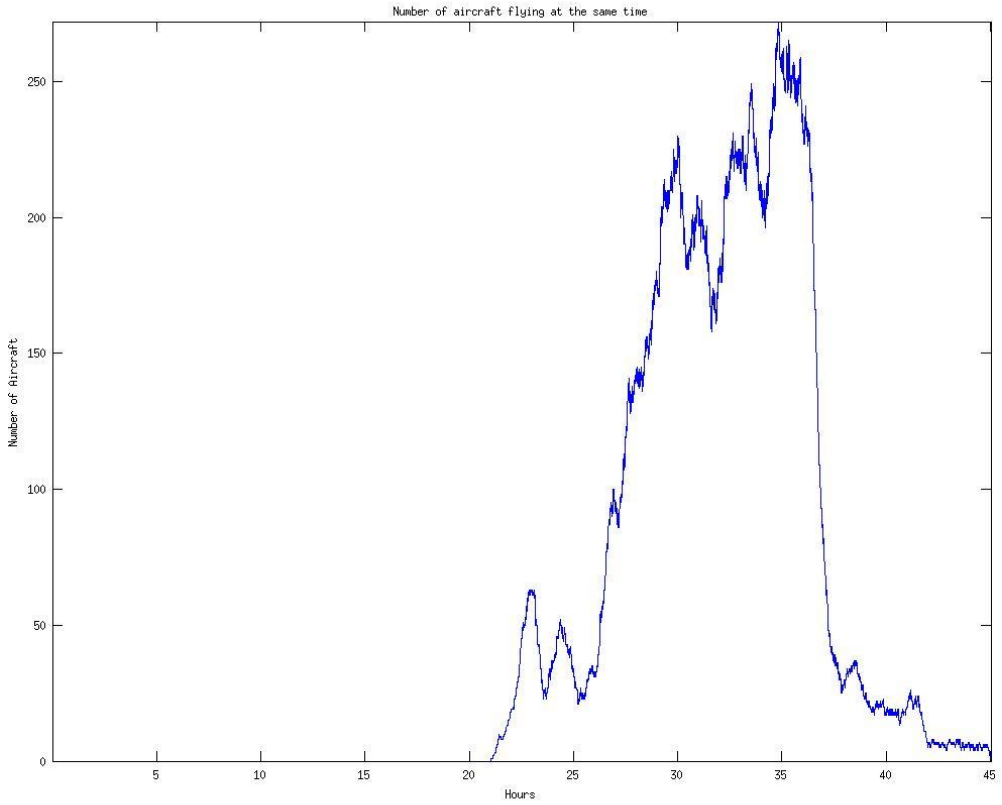
SIMULATION RESULTS

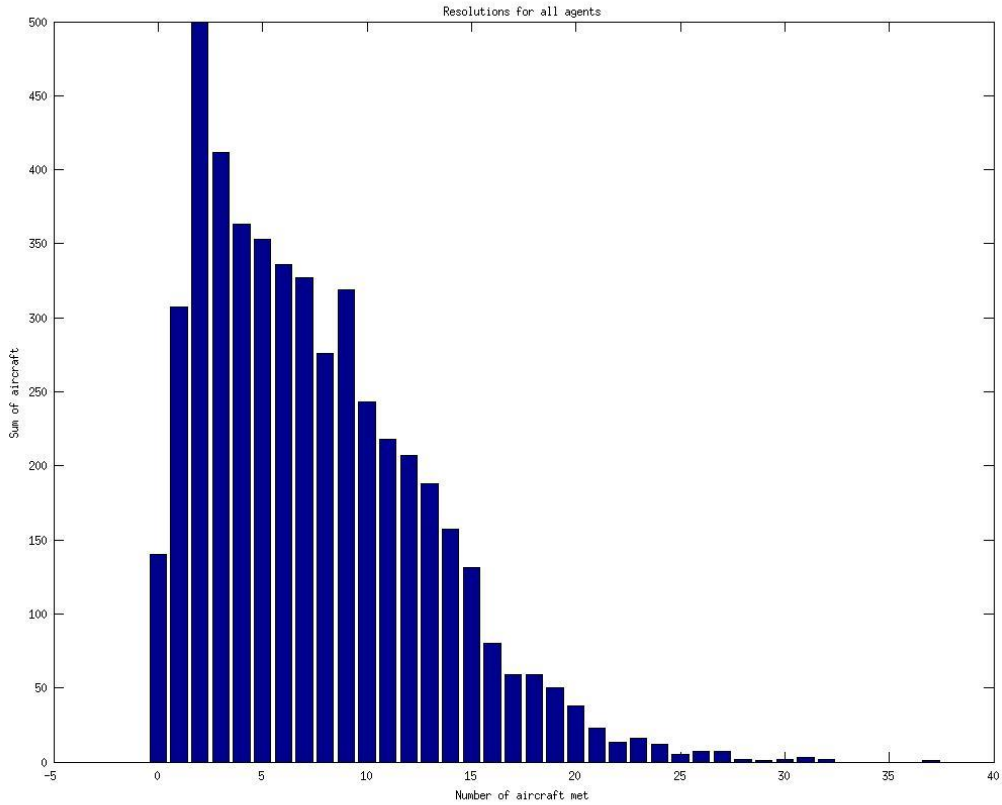
Total number of agents: 2679

Flight time: 3965748.000000 seconds or 1101.596667 hours
Flight time(Mean Value): 1480.309071 seconds or 0.411197 hours
Total time in resolution(all agents): 3123900.000000 seconds or 867.750000 hours
Mean time in resolution: 1166.069429 seconds or 0.323908 hours
Mean value of resolutions for each agent: 4.686077
Average % time in resolution: 78.772025 %

2.5.3.3.2 Πτήσεις με νότια κατεύθυνση







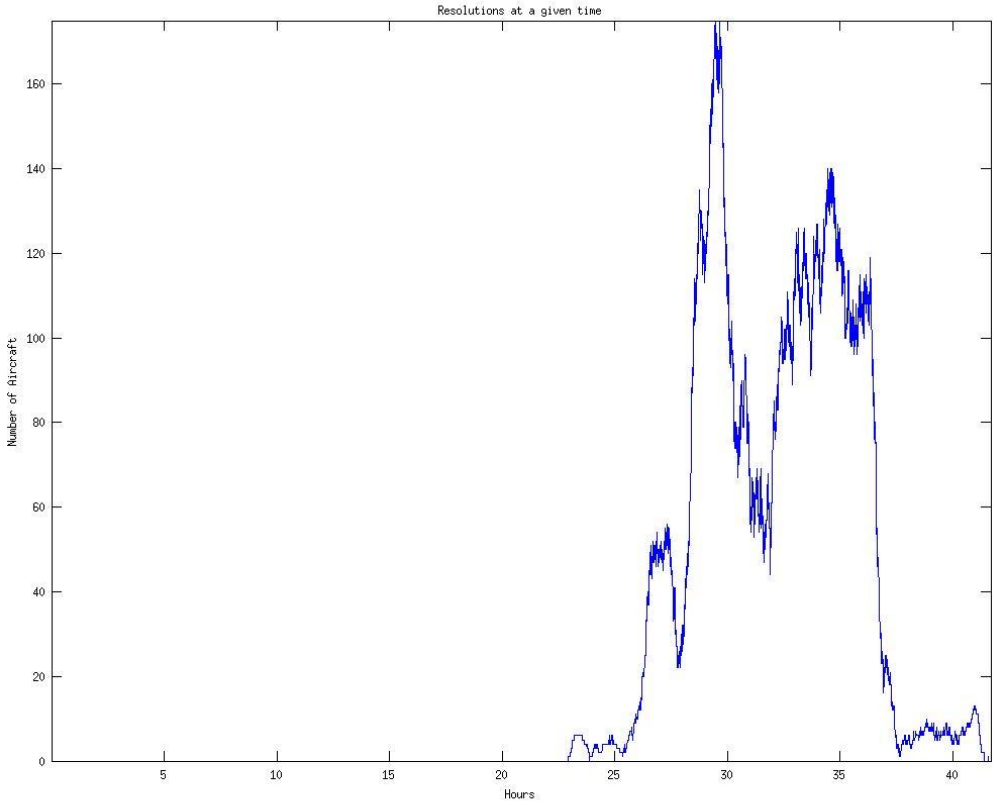
Εικόνες 2.5.3.3.2.1-3

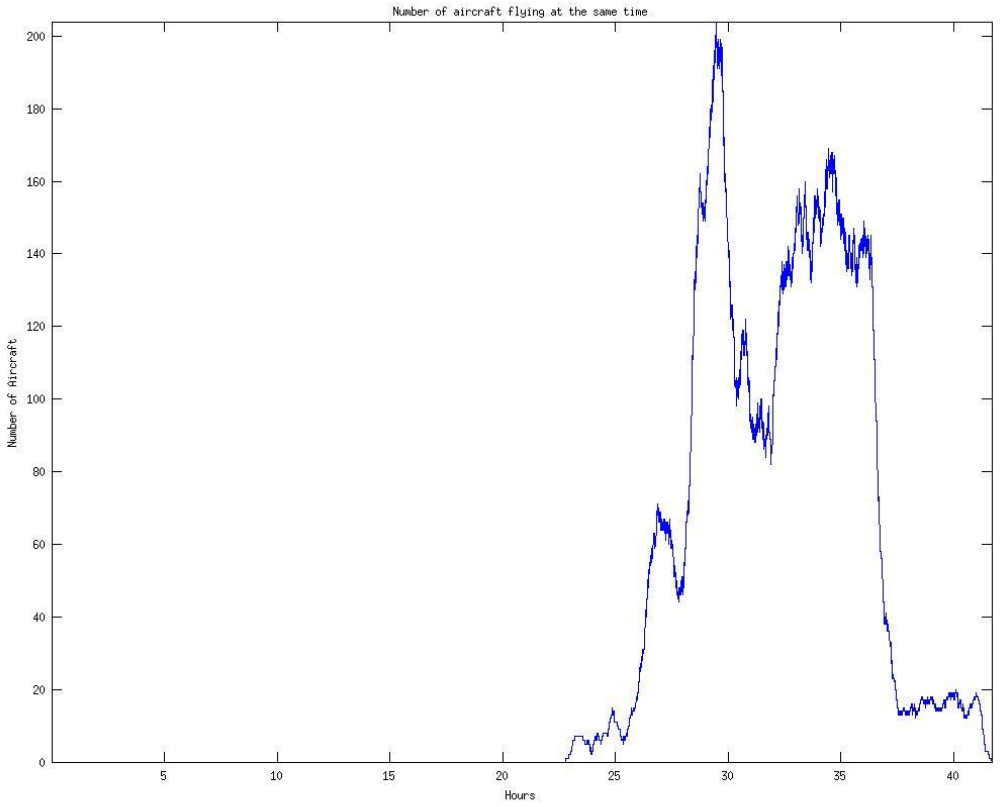
SIMULATION RESULTS

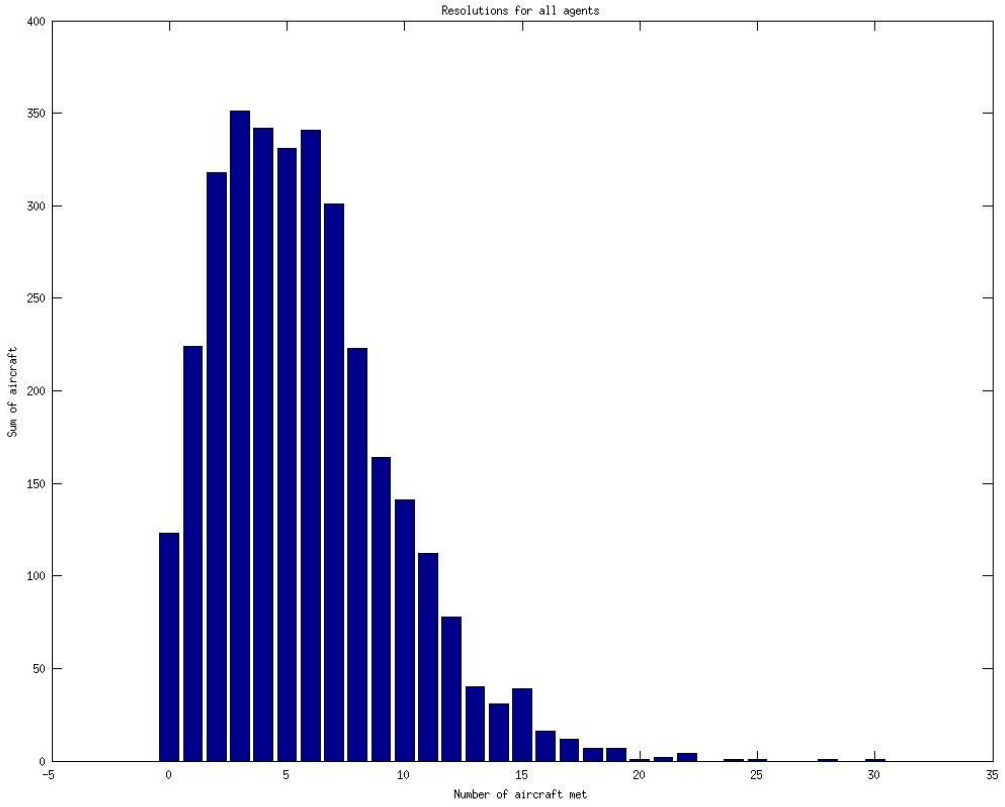
Total number of agents: 4857

Flight time: 8425956.000000 seconds or 2340.543333 hours
Flight time(Mean Value): 1734.806671 seconds or 0.481891 hours
Total time in resolution(all agents): 7322892.000000 seconds or 2034.136667 hours
Mean time in resolution: 1507.698579 seconds or 0.418805 hours
Mean value of resolutions for each agent: 7.489809
Average % time in resolution: 86.908738 %

2.5.3.3.3 Πτήσεις με ανατολική κατεύθυνση







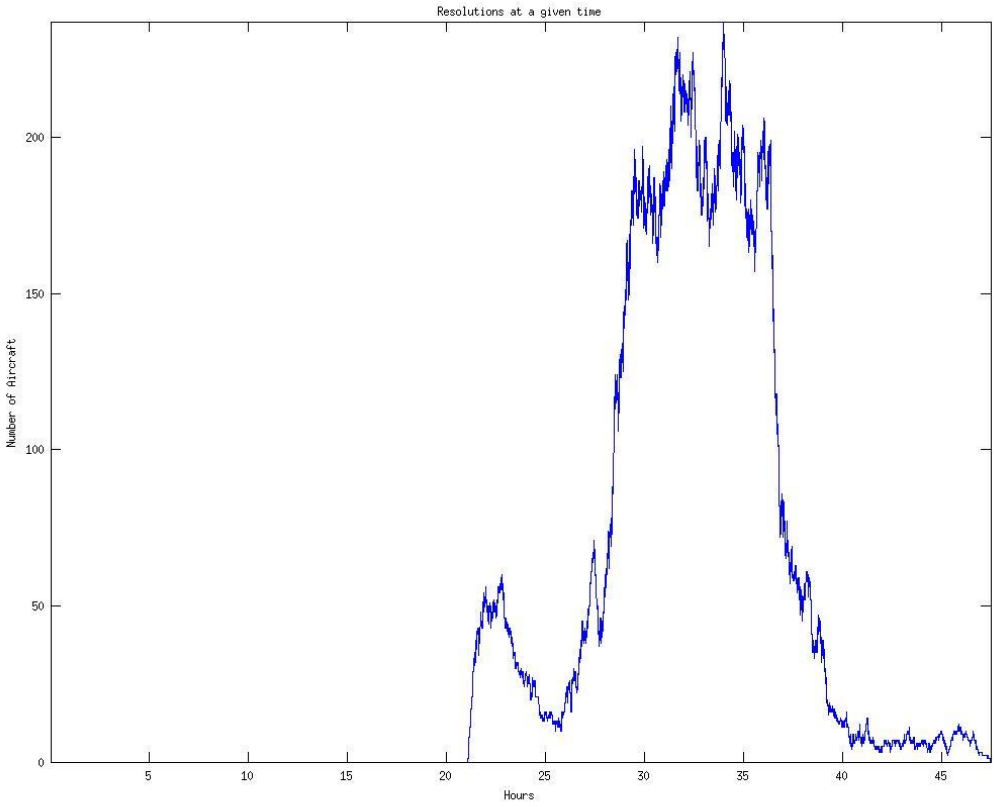
Εικόνες 2.5.3.3.3.1-3

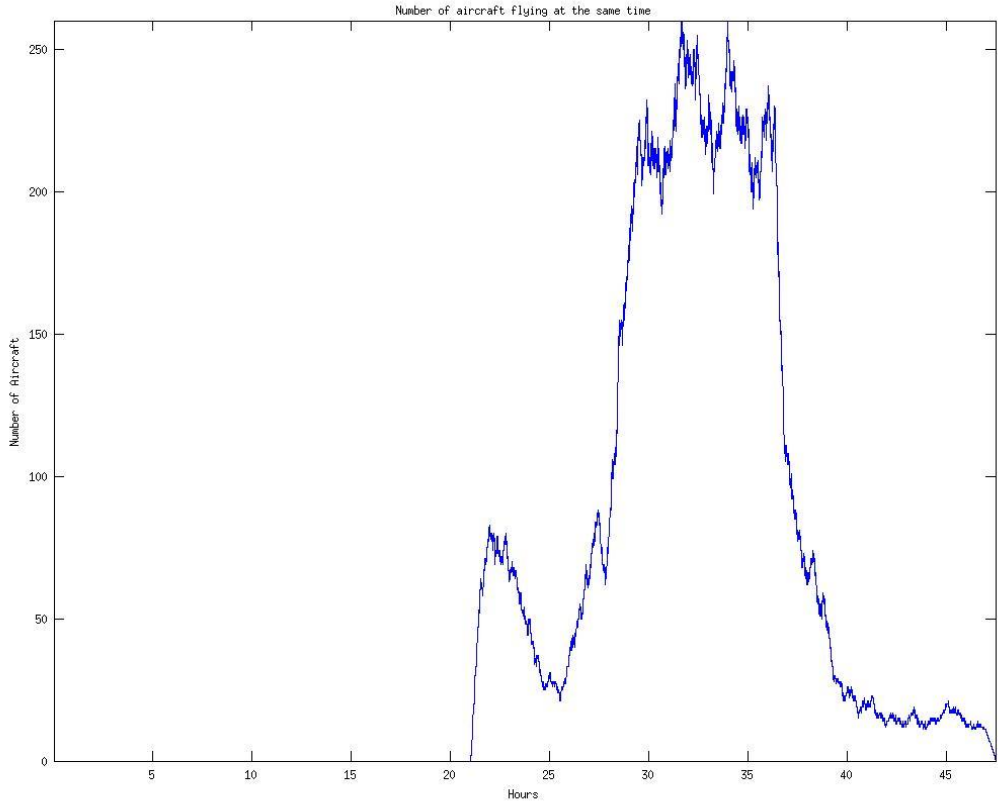
SIMULATION RESULTS

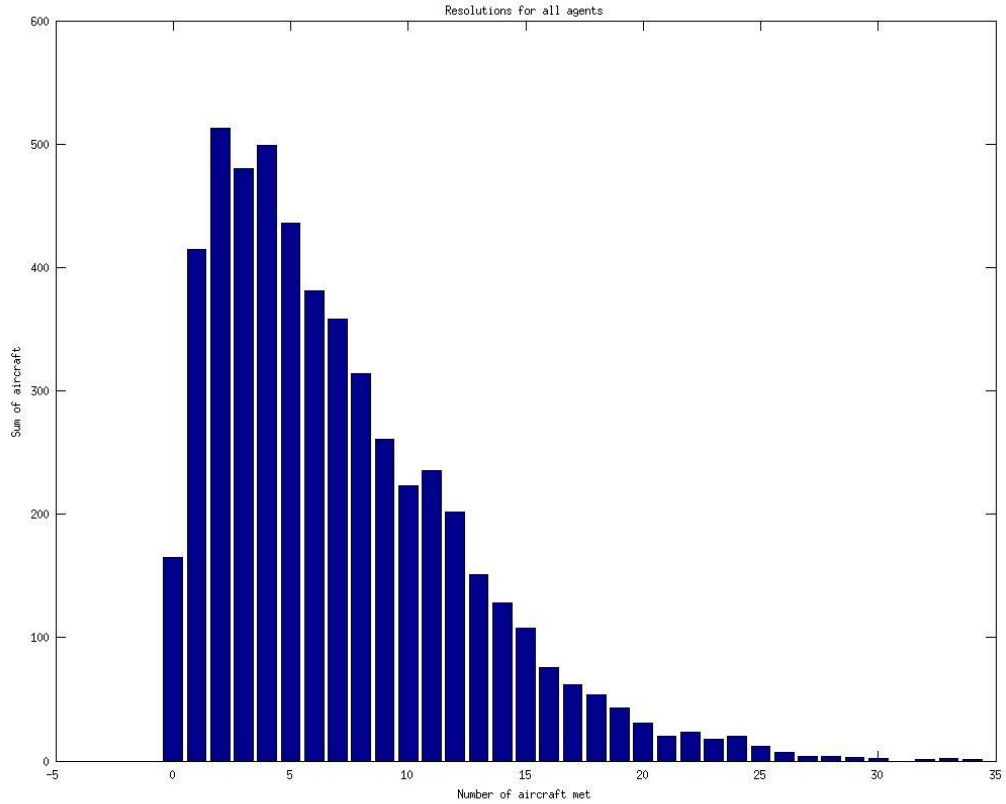
Total number of agents: 3212

Flight time: 4997484.000000 seconds or 1388.190000 hours
Flight time(Mean Value): 1555.879203 seconds or 0.432189 hours
Total time in resolution(all agents): 4166220.000000 seconds or 1157.283333 hours
Mean time in resolution: 1297.079701 seconds or 0.360300 hours
Mean value of resolutions for each agent: 5.741594
Average % time in resolution: 83.366350 %

2.5.3.3.3 Πτήσεις με δυτική κατεύθυνση







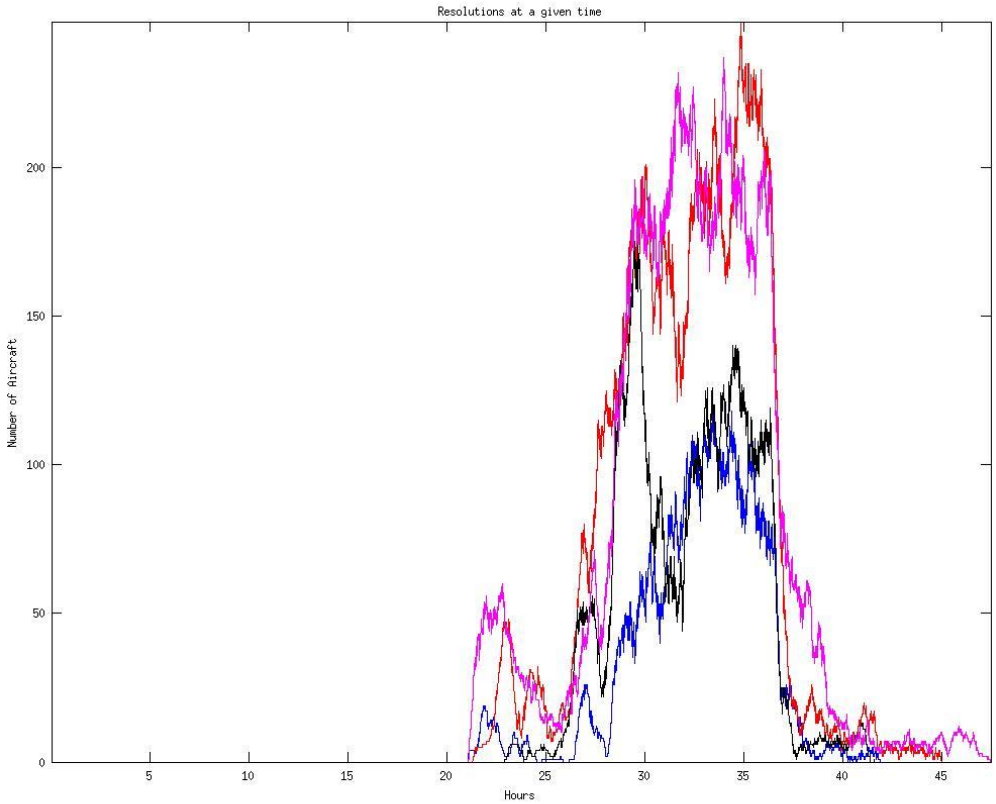
Εικόνες 2.5.3.3.4.1-3

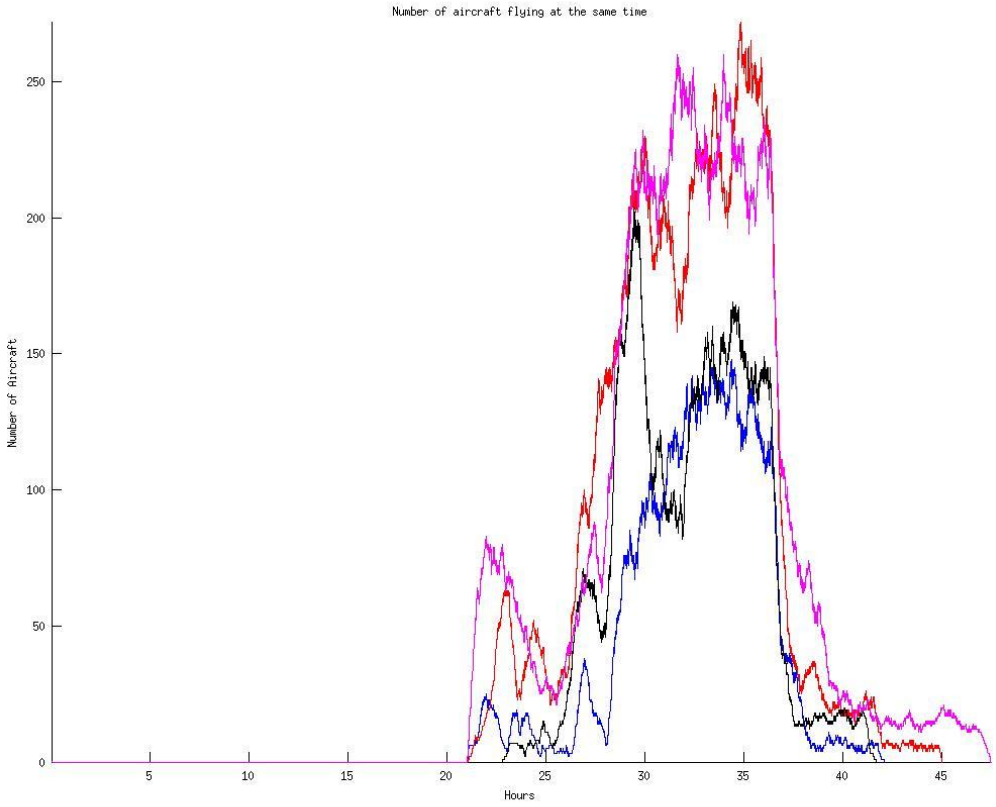
SIMULATION RESULTS

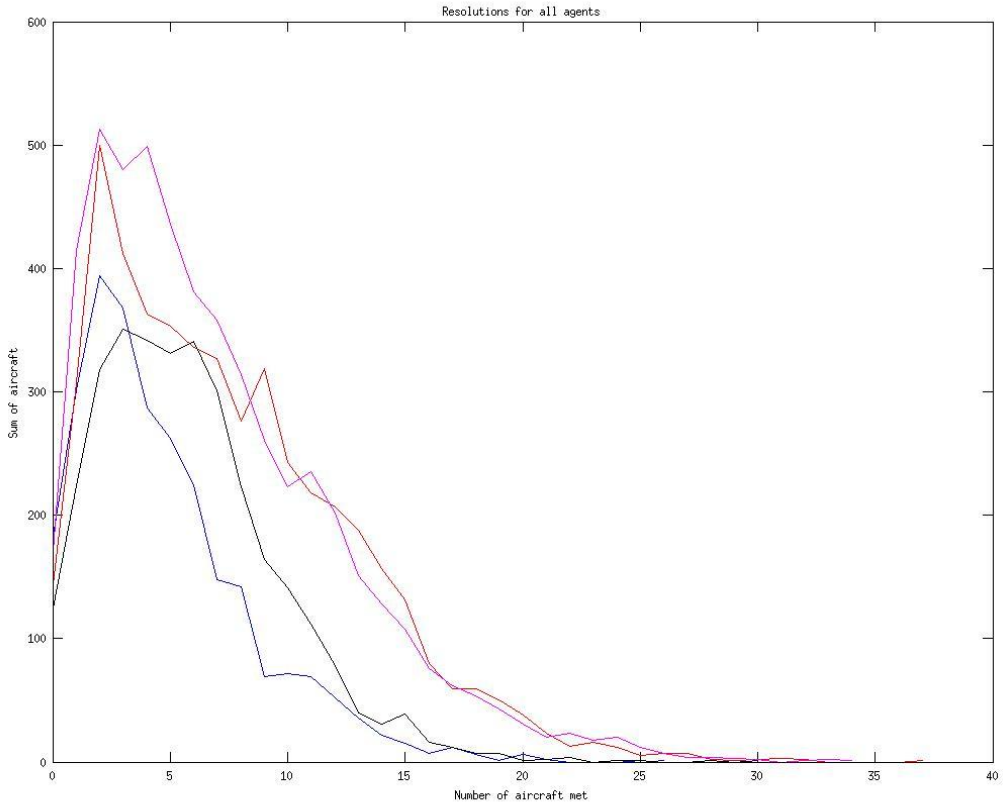
Total number of agents: 5252

Flight time: 9132636.000000 seconds or 2536.843333 hours
Flight time(Mean Value): 1738.887281 seconds or 0.483024 hours
Total time in resolution(all agents): 7755216.000000 seconds or 2154.226667 hours
Mean time in resolution: 1476.621478 seconds or 0.410173 hours
Mean value of resolutions for each agent: 7.007616
Average % time in resolution: 84.917608 %

2.5.3.3.5 Συνολική παρουσίαση αποτελεσμάτων







Εικόνες 2.5.3.3.5.1-3

	North	South	East	West	Mean Value
Number of aircraft	2679/16000	4857/16000	3212/16000	5252/16000	-
Mean time in resolution:	0.3239	0.4188	0.3603	0.4101	0.3782
Mean value of resolutions for each agent:	4.6860	7.4898	5.7415	7.0076	6.2312
Average % time in resolution:	78.772 %	86.908 %	83.366 %	84.917 %	83.491%

Σε αυτήν εδώ την σειρά εξομοιώσεων παρατηρούμε όπως και στις προηγούμενες ενότητες ότι έχουμε πιο ανομοιόμορφα διαγράμματα λόγω τις διαφορετικής κατανομής των αεροσκαφών. Στα σενάρια όπου τα αεροσκάφη πετάνε νότια και δυτικά όπου τα αεροσκάφη είναι και περισσότερα παρατηρούμε ότι τα αεροσκάφη που βρίσκονται στο αέρα κάθε στιγμή ξεπερνάνε τα 200 και σε ορισμένες χρονικές στιγμές τα 250. Επίσης ο μέσος χρόνος πτήσης για κάθε σενάριο κυμαίνεται περίπου στις 0.4 με 0.5 ώρες και ο μέσος αριθμός των αεροσκαφών που συναντάει κάθε αεροσκάφος είναι μεγαλύτερος στα σενάρια με την αυξημένη κίνηση(νότια και δυτική κατεύθυνση αεροσκαφών). Επίσης βλέπουμε ότι το ίδιο συμβαίνει με τον % χρόνο που τα αεροσκάφη βρίσκονται σε resolution όπου αυτός αυξάνεται για σενάρια με περισσότερο αυξημένη κίνηση με μεγαλύτερες τιμές αυτές των σεναρίων με νότια και δυτική κατεύθυνση.

2.5.3.4 Σχολιασμός των 2.5.3.2 και 2.5.3.3

Στις παραπάνω ενότητες δόθηκαν δύο εξομοιώσεις μεγαλύτερου χρονικού βήματος. Αυτές έγιναν για να πάρουμε μία εικόνα της περιόδου υψηλής κίνησης που αρχίζει μετά την ώρα 29 και τα σενάρια που ήδη έχουμε περιγράψει για διαχωρισμό των αεροσκαφών σε τέσσερα ύψη πτήσης. Τελικά τα αποτελέσματα έδωσαν για όλες τις εξομοιώσεις μία εικόνα της κίνησης από τα ώρες 29 έως περίπου 36 όπου ήδη έχει αρχίσει να συναντάται η υψηλή κίνηση της δεύτερης ημέρας. Στην εξομοίωση όπου διαχωρίζουμε τα αεροσκάφη με βάση το ύψος πτήσης τους βλέπουμε μία περισσότερο ομοιόμορφη κατανομή από άποψη αριθμού αεροσκαφών,(4000 στην κάθε εξομοίωση) και κατ επέκταση τον αριθμό των αεροσκαφών που βρίσκονται σε πτήση την κάθε χρονική στιγμή που και στα τέσσερα διαγράμματα ξεπερνάνε τους 150. Επίσης ο μέγιστος αριθμός resolutions που συμβαίνει έχει μέγιστες τιμές τα 120 resolutions. Αντίθετα στα διαγράμματα των εξομοιώσεων με βάση την κατεύθυνση των αεροσκαφών έχουμε μία πιο ανομοιόμορφη κατανομή. Εδώ ο μέγιστος αριθμός των αεροσκαφών που μπορεί να πετάνε την ίδια χρονική στιγμή είναι από 140 για την βόρεια και με λιγότερη κίνηση περίπτωση μέχρι και περισσότερο από 250 για τις περιπτώσεις με τα περισσότερα αεροσκάφη δηλαδή για τις περιπτώσεις που τα αεροσκάφη πετάνε νότια και δυτικά. Ο αριθμός των resolutions είναι από περίπου 140 για την βόρεια περίπτωση περίπου 240 για την νότια και δυτική περίπτωση.

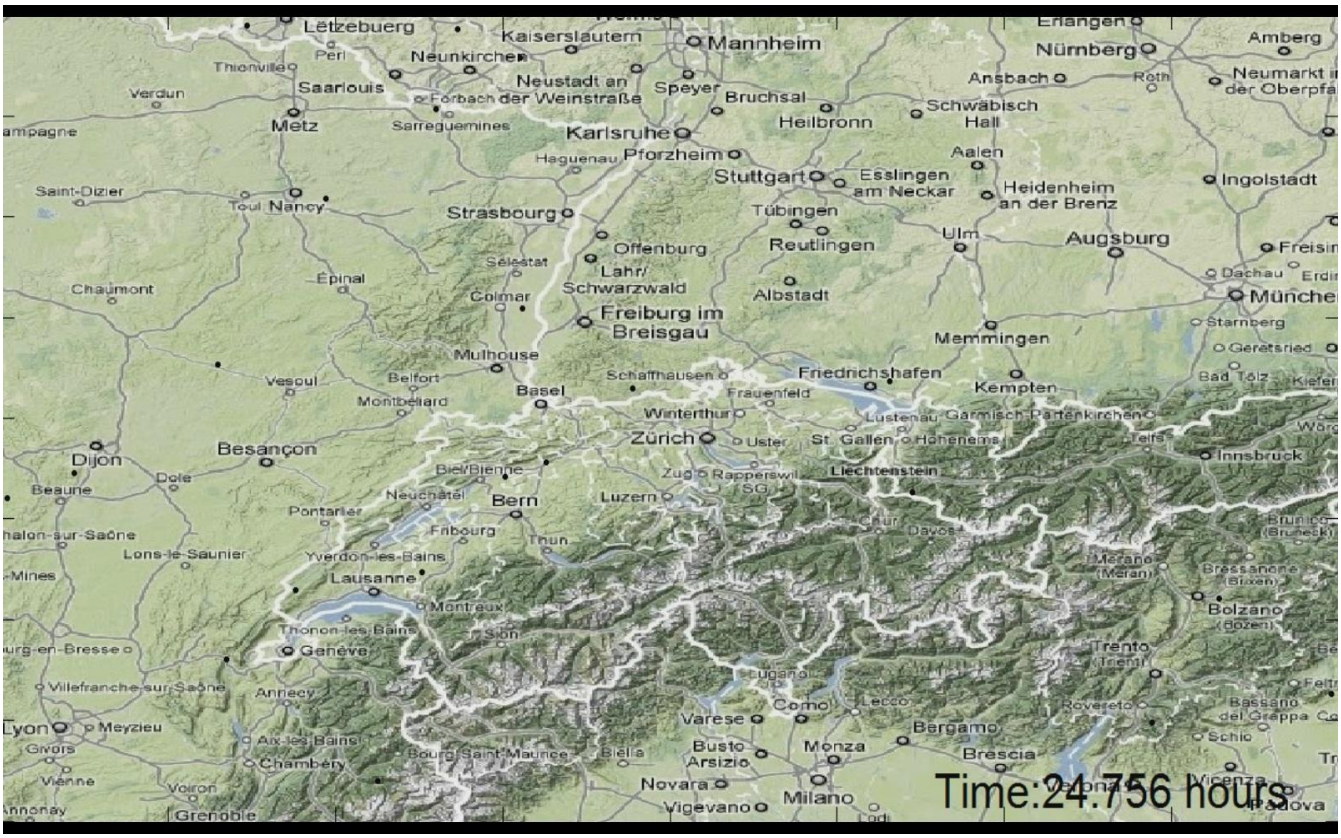
	Mean Value (a/c divided with respect to takeoff time)	**Mean Value (a/c divided with respect to direction)**
Number of aircraft	16000/16000	16000/16000
Mean time in resolution:	0.2709	0.3782
Mean value of resolutions for each agent:	6.5014	6.2312
Average % time in resolution:	79.1515%	83.491%

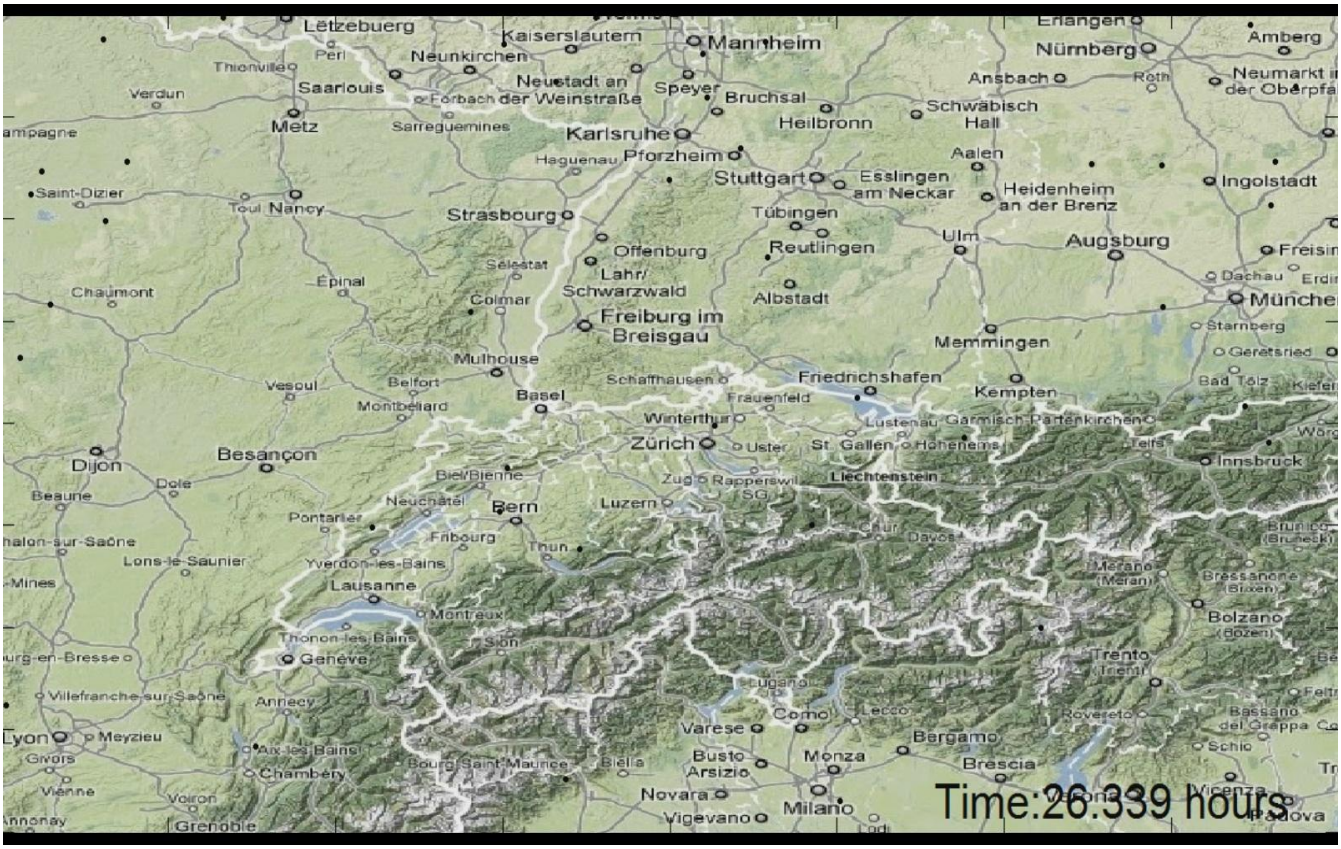
Στον παραπάνω πίνακα δίνουμε τους μέσους όρους διαφόρων μεγεθών προς σύγκριση. Όπως αναφέραμε και πριν λόγω του μεγαλύτερου βήματος της εξομοίωσης εδώ προτιμάμε να σχολιάσουμε τα αποτελέσματα συγκριτικά για να βγάλουμε κάποια συμπεράσματα και όχι τόσο τους αριθμούς για τους οποίους προτιμάμε τις εξομοιώσεις που δίνονται στην ενότητα 2.5.2 οι οποίες είναι και πιο ακριβείς. Σε αυτές τις εξομοιώσεις βλέπουμε ότι ο μέσος χρόνος πτήσης αυξάνεται ελαφρώς για το κάθε αεροσκάφος. Επίσης όπως και στα προηγούμενα αποτελέσματα τα αεροσκάφη που πετάνε με βάση την κατεύθυνση είναι περνάνε περισσότερο χρόνο σε resolutions από τα άλλα αλλά συναντάνε ελαφρώς λιγότερα αεροσκάφη στον δρόμο τους. Αυτό βρίσκεται σε συμφωνία με τα αποτελέσματα που δόθηκαν στην 2.5.2.5.5 και η εξήγηση είναι ίδια με αυτήν που δόθηκε εκεί.

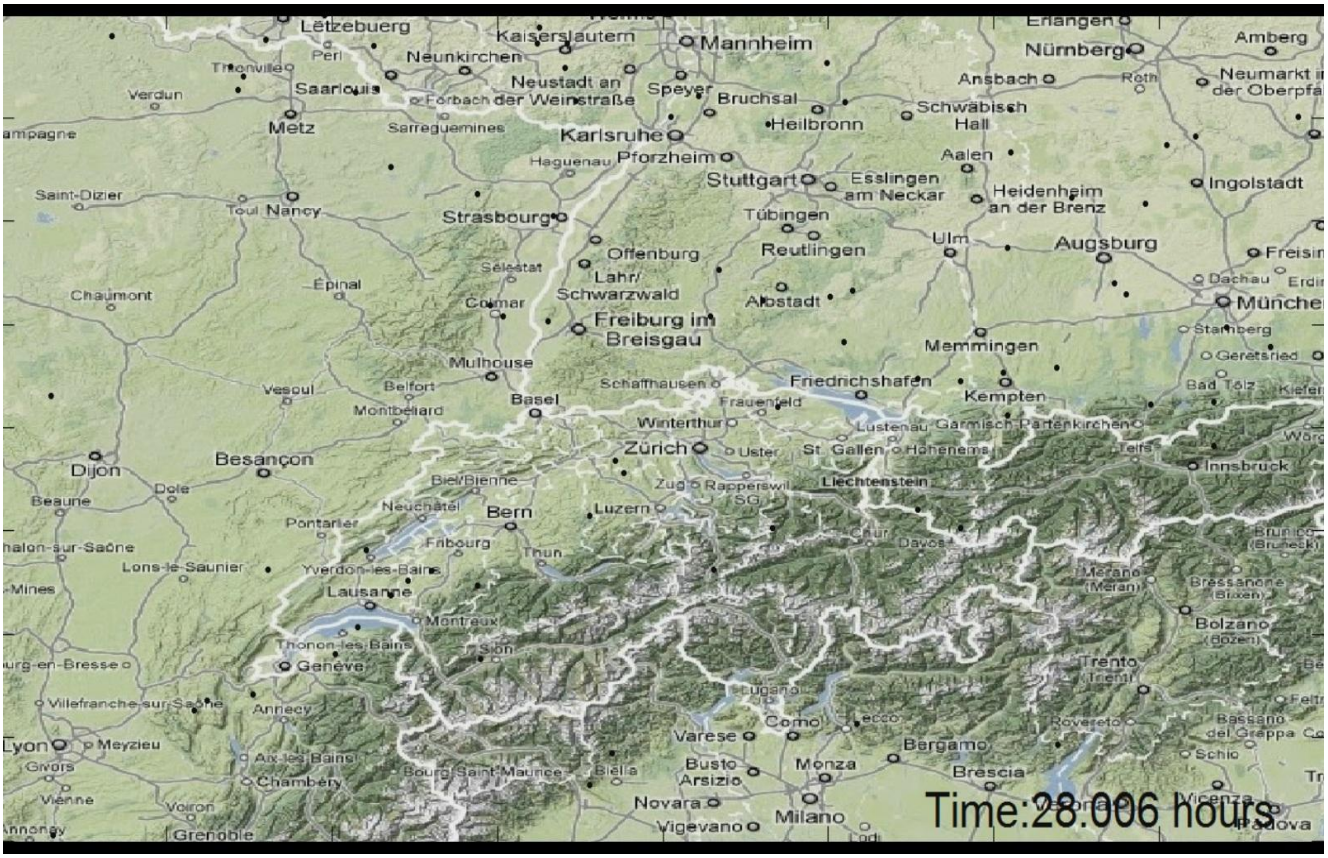
2.5.4 Συμπεράσματα

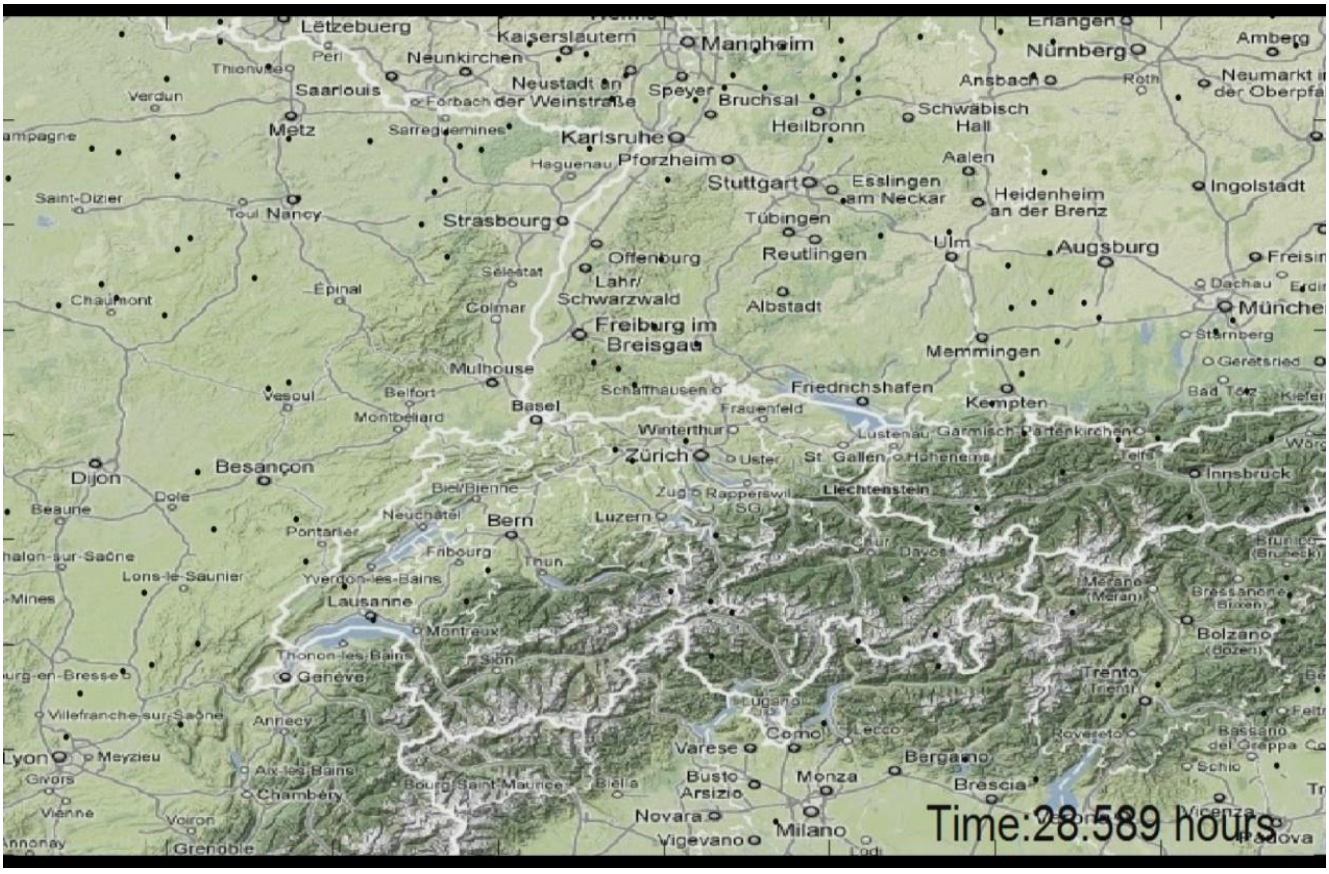
Ο βασικός σκοπός της εργασία αυτής ήταν η δημιουργία ενός κώδικα ο οποίος θα χρησιμοποιηθεί για μία σειρά εξομοιώσεων σε ένα δείγμα πτήσεων από την Κεντρική Ευρώπη. Η κατασκευή των συναρτήσεων πλοήγησης που χρησιμοποιήθηκε όπως και ο έλεγχος περιγράφεται στο Κεφάλαιο 1. Ο κώδικας αυτός δημιουργήθηκε έτσι ώστε εύκολα να μπορεί να αλλάξει η κατασκευή των συναρτήσεων πλοήγησης ή τα δεδομένα των πτήσεων που εισάγονται έτσι ώστε να μπορεί να χρησιμοποιηθεί και άλλες παρόμοιου τύπου εξομοιώσεις πολλών αεροσκαφών.
Οι εξομοιώσεις έδειξαν ότι ο κώδικας μπορεί να χειριστεί ένα μεγάλο αριθμό αεροσκαφών τρέχοντας με επιθυμητή ακρίβεια (όπως παρουσιάζουμε στην 2.5.2.2 έτρεξε 4000 αεροσκάφη με μικρό βήμα). Βασικοί περιορισμοί στον αριθμό των αεροσκαφών είναι τα πρακτικά όρια που θέτει η σημερινή επεξεργαστική ισχύς και RAM καθώς και το γεγονός ότι για πολύ μεγάλο αριθμό αεροσκαφών ο χρόνος που θα χρειαστεί για να ολοκληρωθεί μία σειρά εξομοιώσεων να είναι απαγορευτικός. Ένας βασικός λόγος είναι το μικρό βήμα που απαιτείται για να έχουμε μία αποδεκτή ακρίβεια με την μέθοδο των συναρτήσεων πλοήγησης. Πάντως έγιναν προσπάθειες βελτιστοποίησης του αλγόριθμο έτσι ώστε να μπορεί να χειρίζεται ένα μεγάλο αριθμό αεροσκαφών και δεδομένων όσο το δυνατό αποδοτικότερα.
Όσο αφορά τις εξομοιώσεις έγινε ένας μεγάλος αριθμός αυτών έτσι ώστε να μπορέσουμε να έχουμε μία εικόνα του κίνησης των αεροσκαφών. Ο επιμέρους σχολιασμός των αποτελεσμάτων γίνεται στις αντίστοιχες ενότητες. Όπως παρατηρούμε στις εξομοιώσεις της πρώτης ημέρας τα σενάρια για τα οποία υπάρχει μεγαλύτερη απόκλιση από την ευθεία πορεία τους είναι φυσικά τα σενάρια όπου αυτά τρέχουν μαζί χωρίς κάποιο διαχωρισμό. Εμείς προτείνουμε ότι θα ήταν πιο αποδοτικό να χωριστούν τα αεροσκάφη σε τέσσερα διαφορετικά ύψη πτήσης. Όπως θα περίμενε κανείς ο αποδοτικότερος τρόπος διαχωρισμού φάνηκε να είναι ο διαχωρισμός των αεροσκαφών κατά κατεύθυνση. Παρόλα αυτά αν χωρίσουμε τα αεροσκάφη με βάση την ώρα εκκίνησης έχουμε μία πιο ομοιόμορφη κατανομή της κίνησης στα τέσσερα επίπεδα πτήσης. Παρόλα αυτά κάποια τυχαία ίσως σενάρια τα οποία μπορεί να συμβούν κάποια μέρα ίσως περιλαμβάνουν υπερβολικό αριθμό αεροσκαφών σε κάποιο ύψος πτήσης αν κάνουμε διαχωρισμό με βάση την κατεύθυνση. Με τον διαχωρισμό με βάση την ώρα εκκίνησης διασφαλίζουμε ότι θα έχουμε γενικά πιο ομοιόμορφες κατανομές της κίνησης.
Τέλος τα διαγράμματα μεγαλύτερου βήματος δόθηκαν έτσι ώστε να φτιάξουμε μία προσεγγιστική εικόνα του χώρου την περίοδο υψηλής κίνησης. Και αυτά τα αποτελέσματα έδειξαν παρόμοια συμπεριφορά ότι δηλαδή τα αεροσκάφη με τον διαχωρισμό ανά κατεύθυνση συναντάνε λιγότερα άλλα αεροσκάφη(έστω και οριακά λιγότερα εδώ) ενώ περνάνε περισσότερο χρόνο μέσα σε resolution ακριβώς όπως οι προηγούμενες περιπτώσεις.

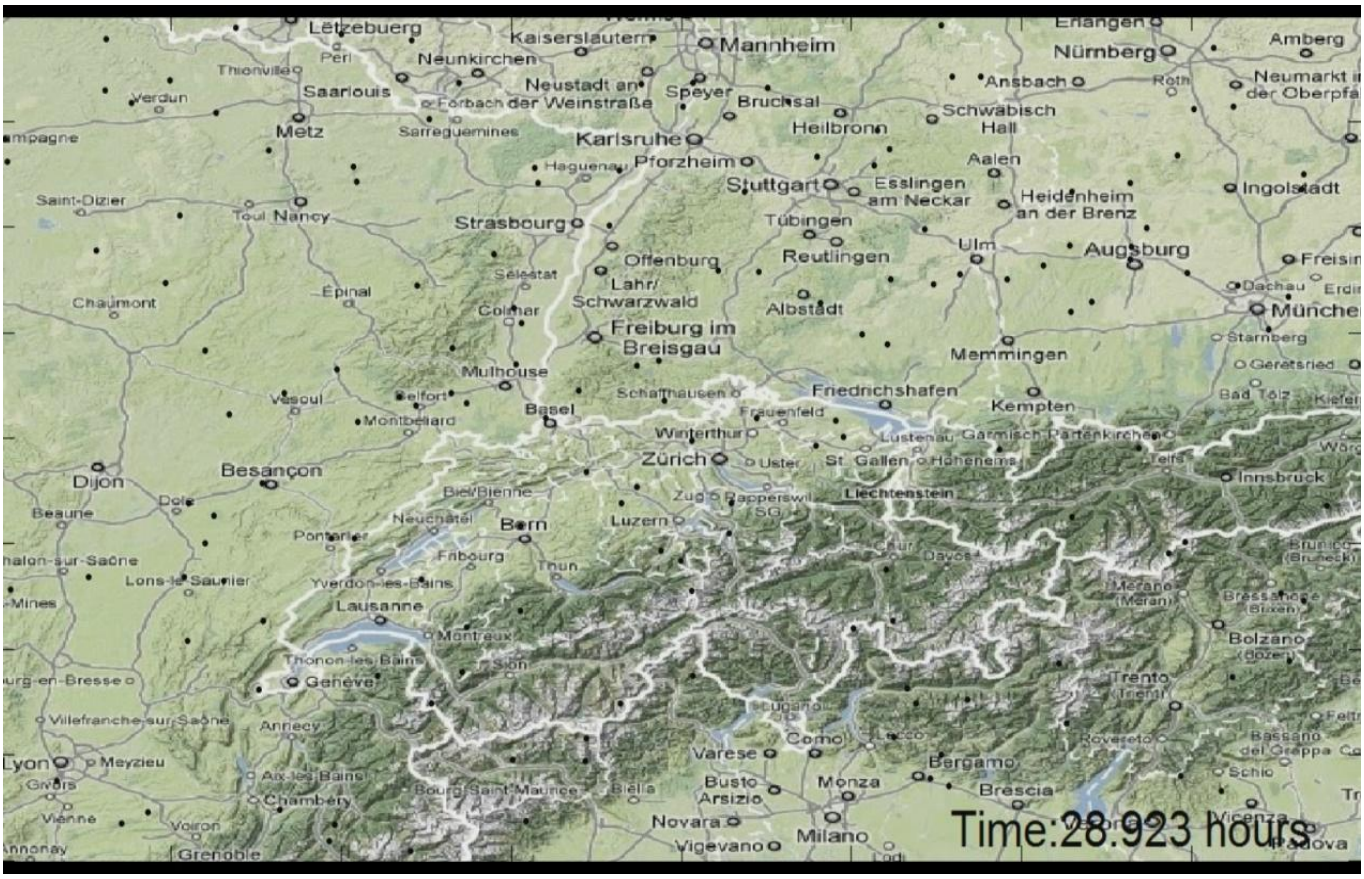
2.5.5Μερικές εικόνες από βίντεο

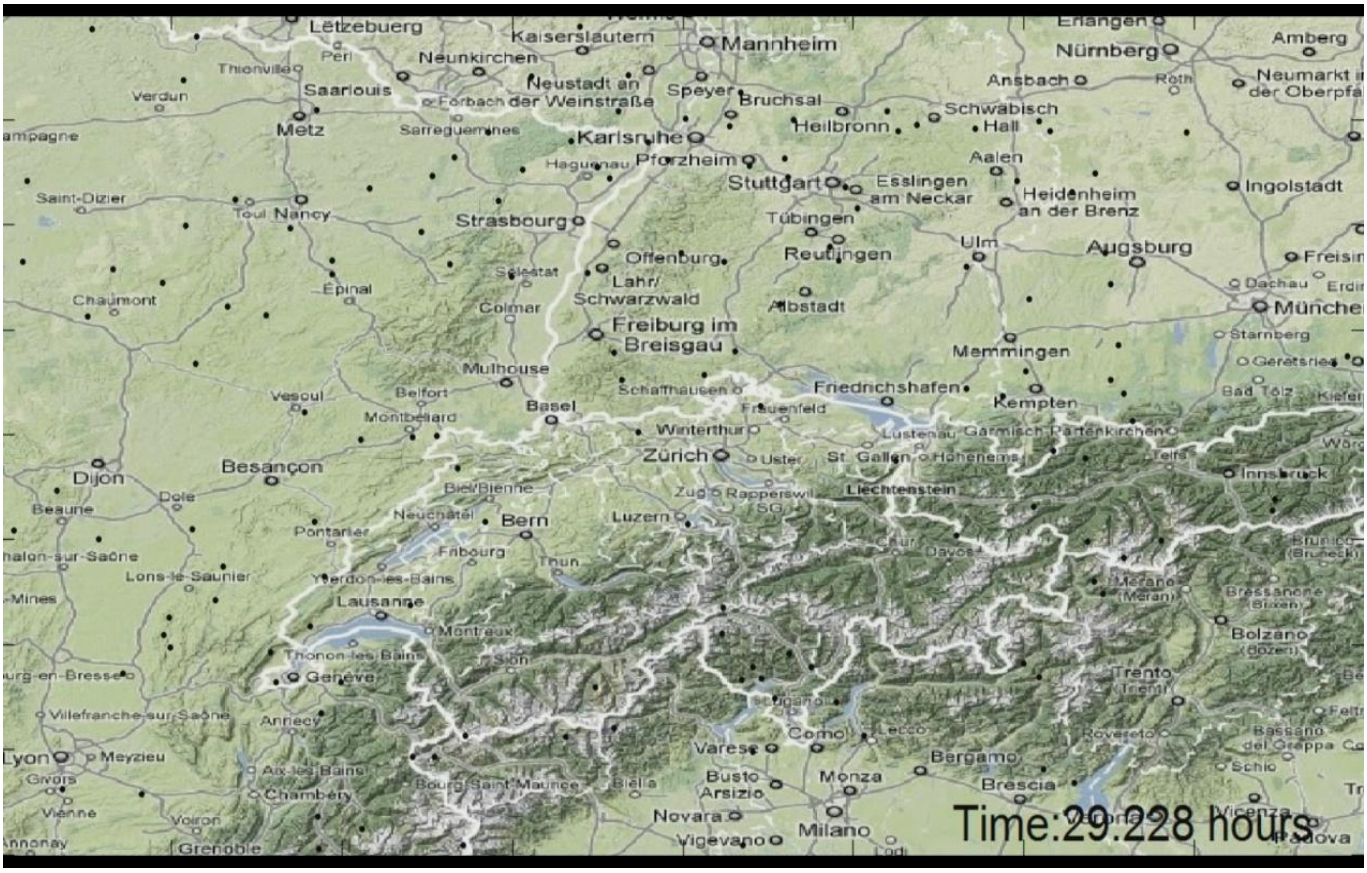





Time:28.006 hours


Time:28.589 hours


Time:28.923 hours


Time:29.228 hours

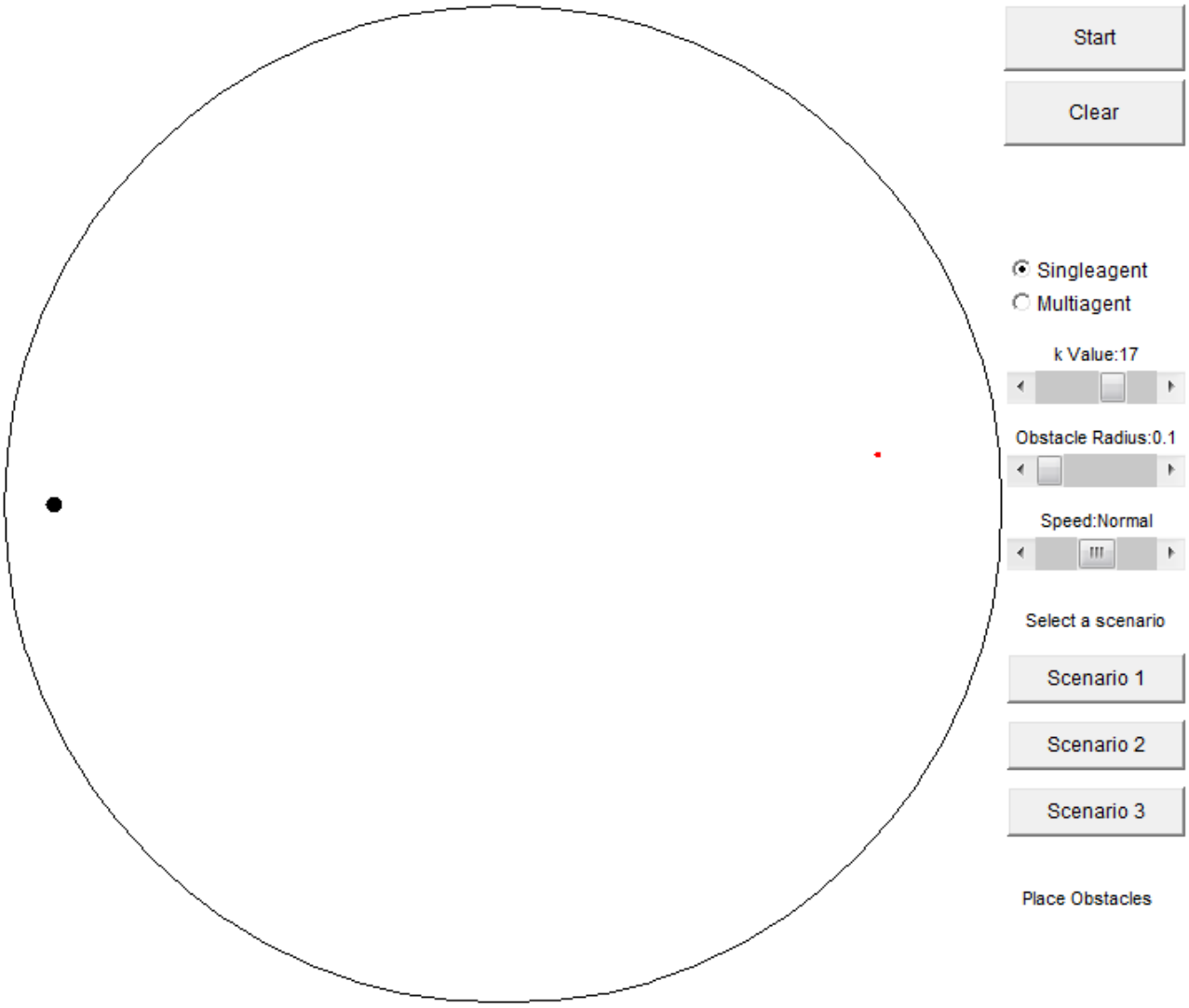
ΚΕΦΑΛΑΙΟ 3

1.Ανάπτυξη εφαρμογής σε Java για εξομοίωση δισδιάστατων σεναρίων των συναρτήσεων πλοήγησης

1.1Εισαγωγικά

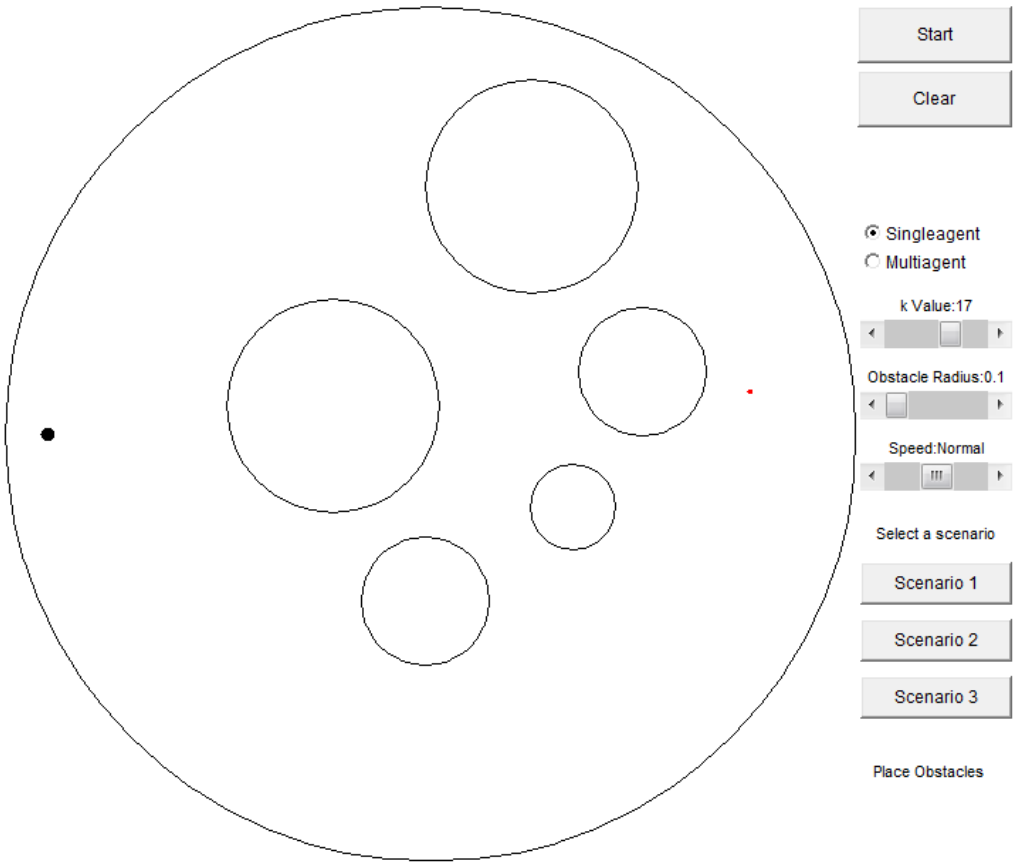
Παρακάτω παρουσιάζουμε μία εφαρμογή που αναπτύχθηκε σε Java και είναι ικανή να λύσει κάποια απλά σενάρια προγραμματισμού πορείας με την βοήθεια των συναρτήσεων πλοήγησης. Δημιουργήθηκε έτσι ώστε να υπάρχει ένα μέσο με την βοήθεια του οποίου κάποιος μπορεί να τρέξει κάποιες απλές περιπτώσεις single agent και multi agent σεναρίων ορίζοντας τα με τον τρόπο που αυτός επιθυμεί. Ο χώρος λειτουργίας επιλέγεται να είναι ένας σφαιρικός χώρος (sphere world-Κεφάλαιο 1) αφού οι συναρτήσεις πλοήγησης μπορούν να εξομοιώσουν προβλήματα σε οποιοδήποτε χώρο διφεομορφικό σε αυτόν. Σε αυτόν τον δισδιάστατο χώρο λειτουργίας ο χρήστης μπορεί να επιλέξει που θα τοποθετήσει μία σειρά εμποδίων μεταβλητής ακτίνας(μεταξύ κάποιον προεπιλεγμένων τυπικών τιμών) κλικάροντας στο σημείο που επιθυμεί, να αλλάξει την παράμετρο k (Κεφάλαιο 1) ή και την ταχύτητα της εξομοίωσης. Η εφαρμογή αυτή βλέπει το πρόβλημα πλοήγησης από μία πιο multimedia σκοπιά και ο βασικός της στόχος είναι να λειτουργήσει ως ένα μέσο με το οποίο κάποιος χρήστης ο οποίος δεν είναι εξοικειωμένος με τις συναρτήσεις πλοήγησης να δημιουργήσει μερικά σενάρια μόνος του και να τα δει να εκτελούνται. Πρέπει να σημειώσουμε (όπως αναφέραμε και παραπάνω) ότι ουσιαστικά ο κάθε κύκλος που τοποθετείται μπορεί να αντιπροσωπεύει πρακτικά οποιαδήποτε μορφή εμποδίου (αρκεί το εμπόδιο αυτό να είναι ένα συμπαγές σώμα χωρίς οπές ή οι οπές του να μοντελοποιηθούν σαν συμπαγή σημεία στο σώμα). Επίσης ο χώρος λειτουργίας μπορεί να θεωρηθεί ότι αντιπροσωπεύει οποιαδήποτε κλειστή καμπύλη που περιβάλει τα εμπόδια. Βεβαίως το πρόβλημα μπορεί να διαφοροποιηθεί ελάχιστα ανάλογα με την κάθε περίπτωση αλλά από την απόδειξη των συναρτήσεων πλοήγησης, όπως ήδη έχουμε αναφέρει, καθένα από τα παραπάνω άπειρα προβλήματα μπορεί να λυθεί μετασχηματίζοντας αυτό σε ένα σφαιρικό χώρο αρκεί να βρεθεί ένας διφεομορφισμός στον χώρο αυτό.(Βεβαίως αυτό το πρόβλημα μπορεί να αποδειχθεί ιδιαίτερα δύσκολο πολλές φορές ανάλογα και με την πολυπλοκότητα του χώρου και του αριθμού των διαστάσεων παρόλα αυτά η λύση θεωρητικά υπάρχει αλλά αυτό είναι ένα άλλου είδους πρόβλημα που δεν θα μας απασχολήσει εδώ). Ούτως ή άλλως πάντως η εφαρμογή αυτή έχει τον σκοπό να δώσει με ένα εποπτικό τρόπο την λύση κάποιων απλών δισδιάστατων προβλημάτων και όχι να χρησιμοποιηθεί για την λύση οποιουδήποτε προβλήματος.

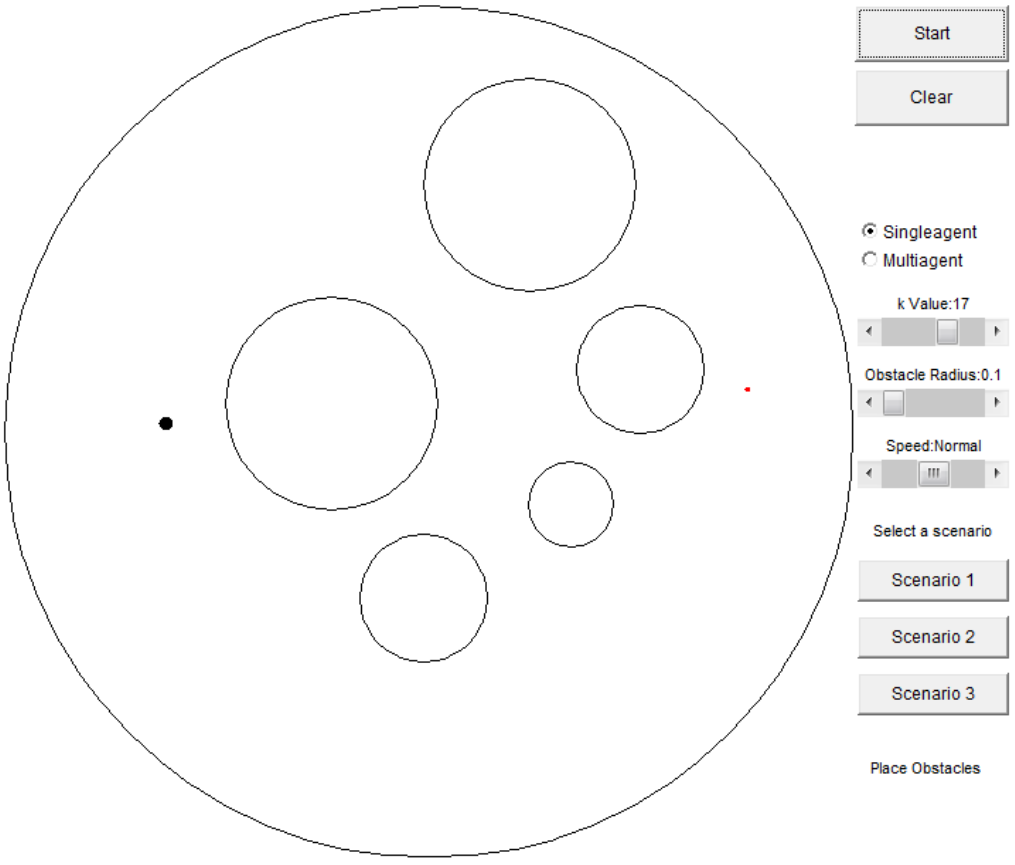
1.2Επιλογή Single-Agent

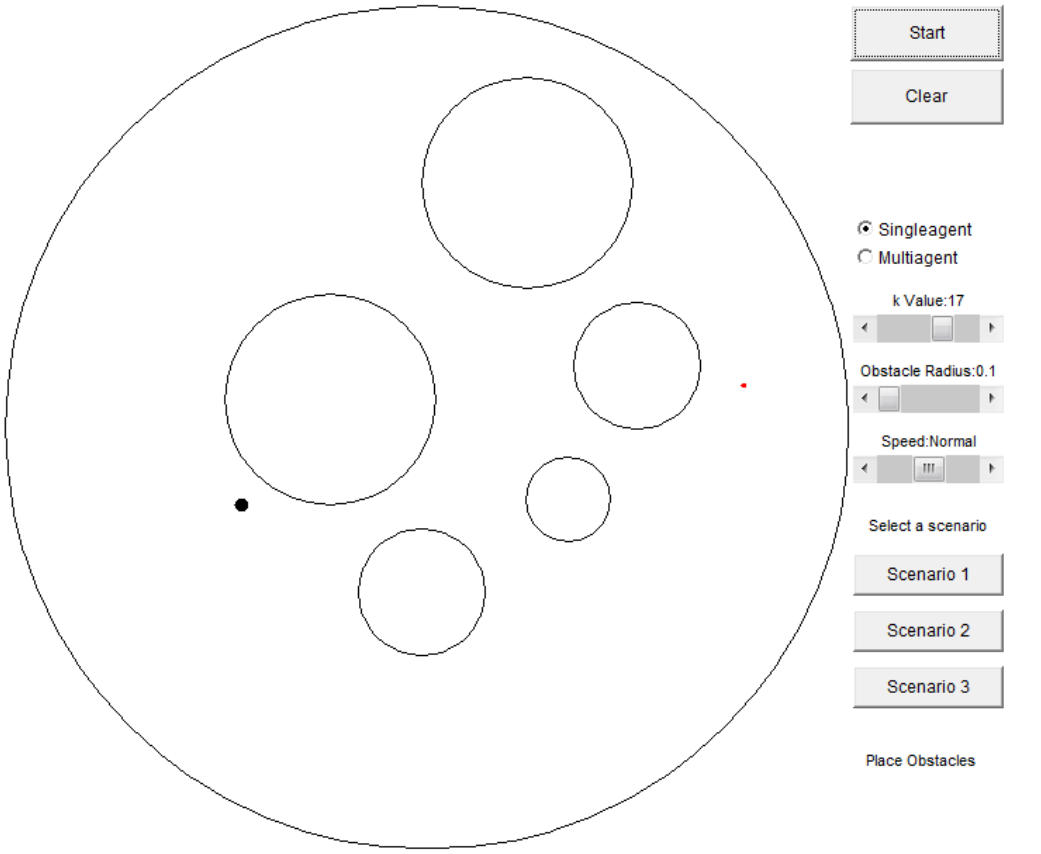


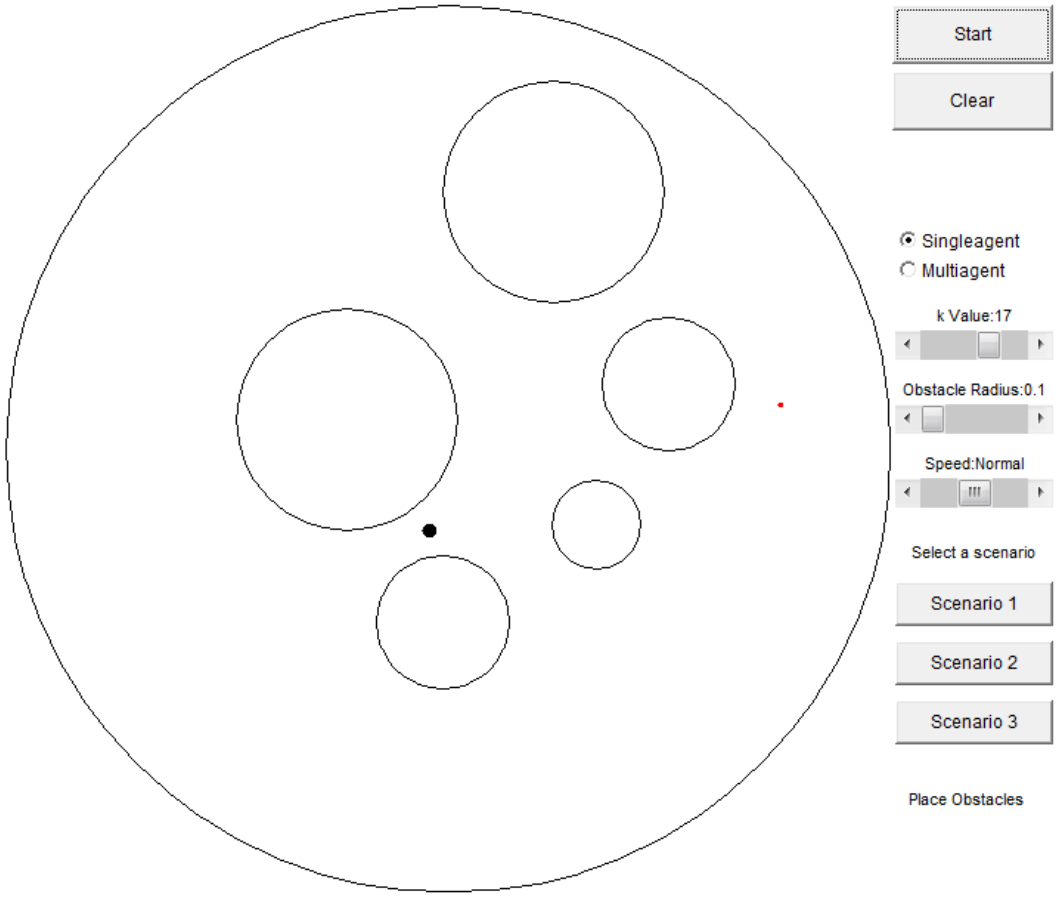
Σε αυτήν την επιλογή έχουμε το πρόβλημα του ενός μόνο agents. Έχουμε θέση τυχαία την αρχή εκεί που είναι ο μαύρος κύκλος και ο προορισμός είναι στο σημείο με την κόκκινη τελεία. Αυτό το σενάριο προσομοιώνει ουσιαστικά το πρόβλημα που έχουμε ένα ρομπότ να κινείται μέσα σε έναν χώρο με εμπόδια των οποίων η ακριβής θέση είναι απόλυτα γνωστή κάθε στιγμή. Εδώ μπορούμε να επιλέξουμε αυθαίρετα θέσεις εμποδίων σε οποιοδήποτε σημείο τους χώρου επιλέγοντας ένα μέγεθος ακτίνας από το αντίστοιχο scrollbar. Τα μεγέθη της ακτίνας είναι αδιαστατοποιημένα ως προς την ακτίνα του περιγράμματος του χώρου εργασίας(μεγάλος κύκλος). Επίσης υπάρχουν μερικά προεπιλεγμένα σενάρια που μπορούν να φορτωθούν άμεσα στο πρόγραμμα. Μπορούμε να ελέγξουμε και την ταχύτητα της εξομοίωσης όπως και την επιλογή της παραμέτρου k. Αφού τοποθετήσουμε τα εμπόδια μπορούμε να τρέξουμε την εξομοίωση. Τα εμπόδια δεν πρέπει να αλληλεπικαλύπτονται ούτε να "κόβουν" το περίγραμμα του χώρου εργασίας(από την βασική απόδειξη των συναρτήσεων πλοήγησης-Κεφάλαιο 1).

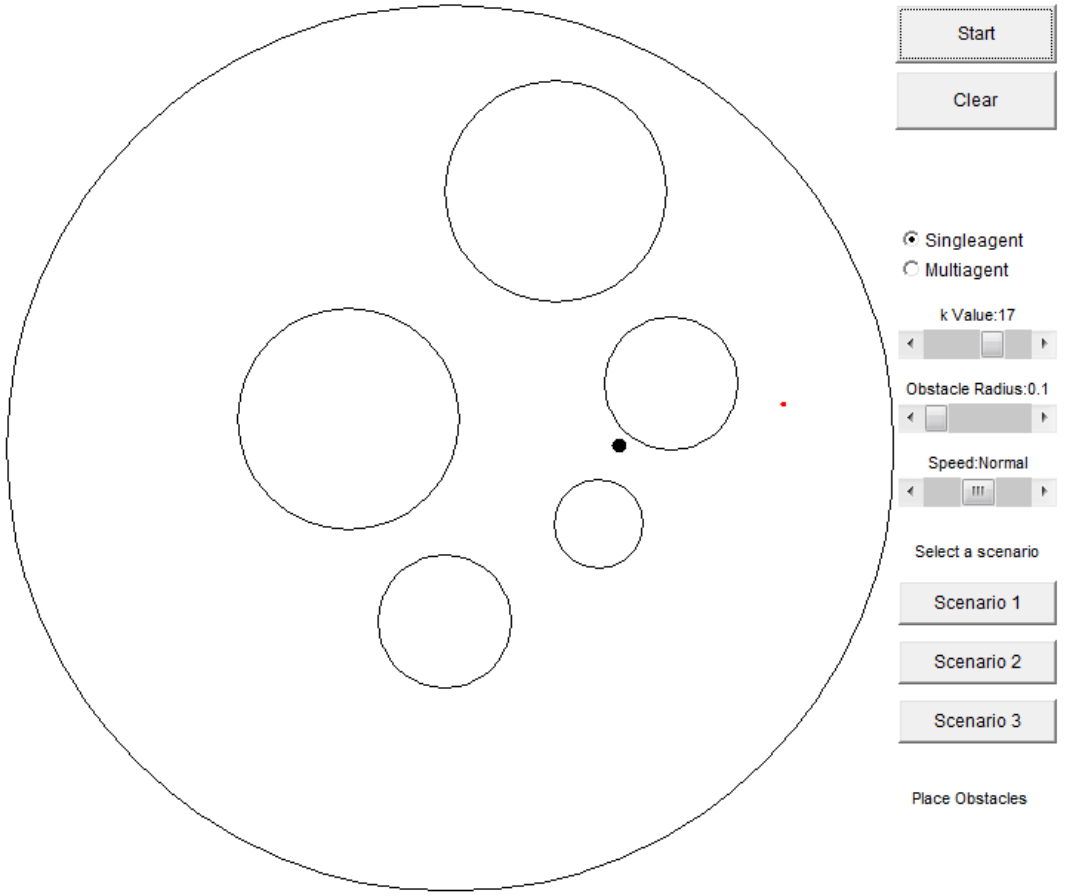
Σενάριο 1

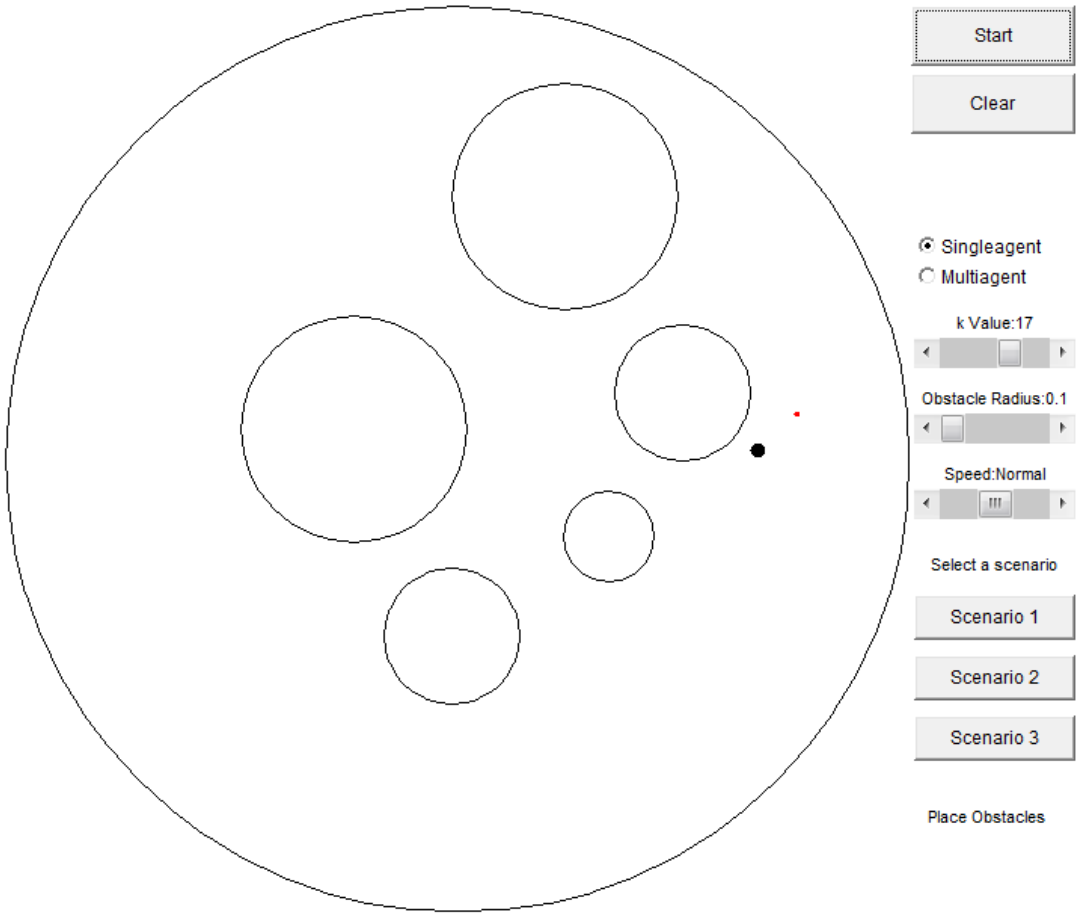

Start
Clear
Singleagent
Multiagent
k Value:17
Obstacle Radius:0.1
Speed:Normal
Select a scenario
Scenario 1
Scenario 2
Scenario 3
Place Obstacles


Start
Clear
Singleagent
Multiagent
k Value:17
Obstacle Radius:0.1
Speed:Normal
Select a scenario
Scenario 1
Scenario 2
Scenario 3
Place Obstacles

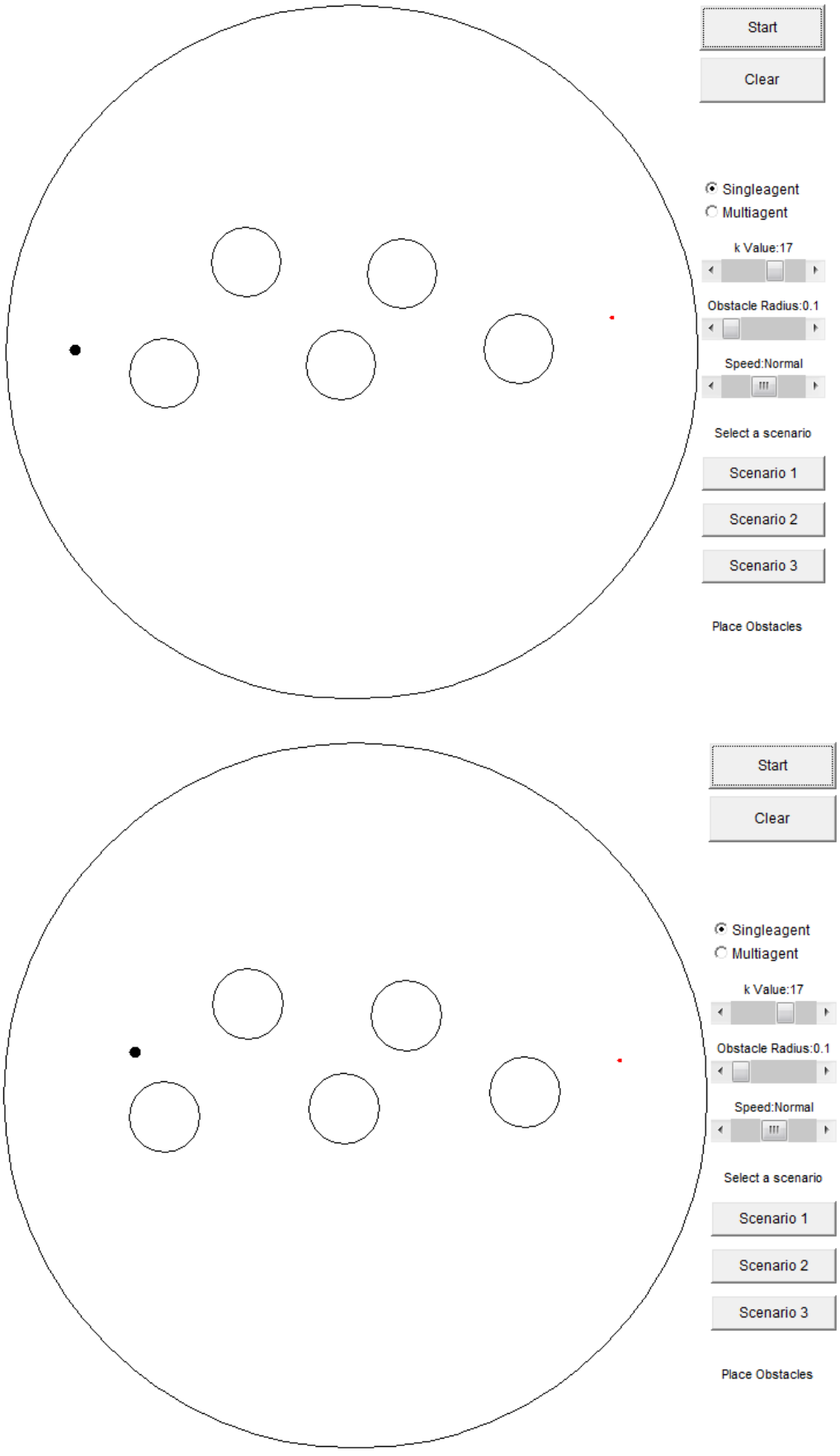

Start
Clear
Singleagent
Multiagent
k Value:17
Obstacle Radius:0.1
Speed:Normal
Select a scenario
Scenario 1
Scenario 2
Scenario 3
Place Obstacles

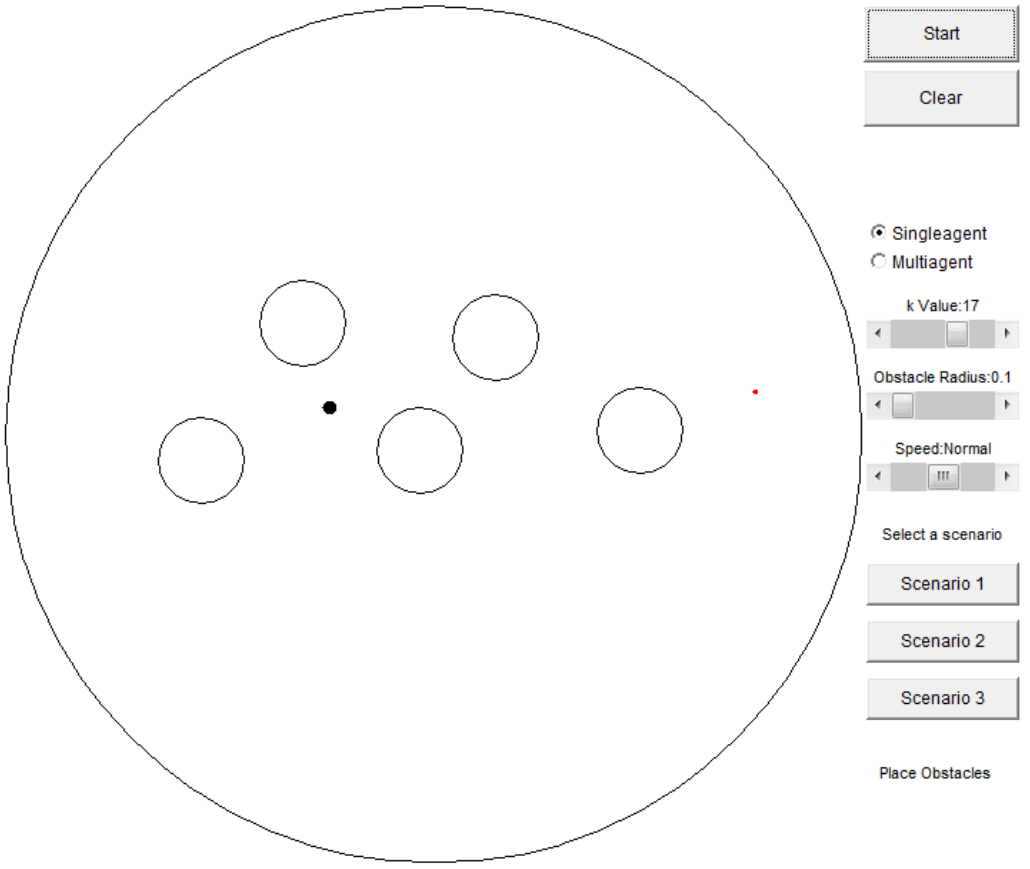

Start
Clear
Singleagent
Multiagent
k Value:17
Obstacle Radius:0.1
Speed:Normal
Select a scenario
Scenario 1
Scenario 2
Scenario 3
Place Obstacles

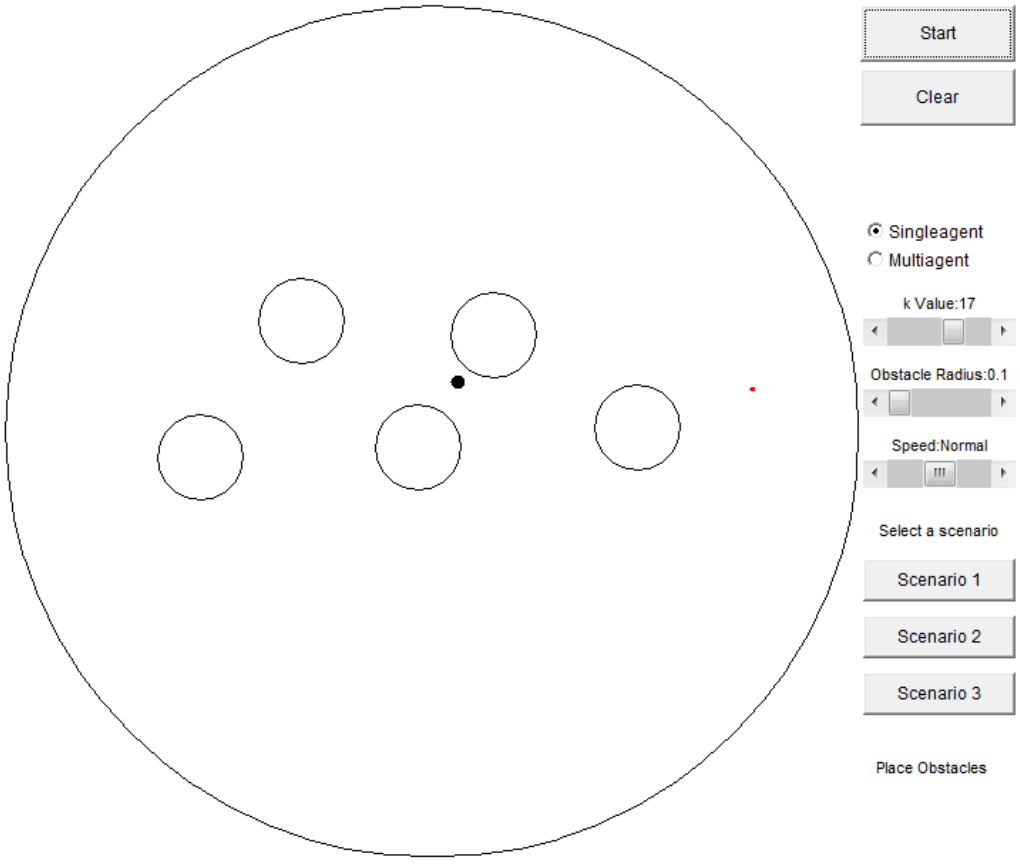

Start
Clear
Singleagent
Multiagent
k Value:17
Obstacle Radius:0.1
Speed:Normal
Select a scenario
Scenario 1
Scenario 2
Scenario 3
Place Obstacles

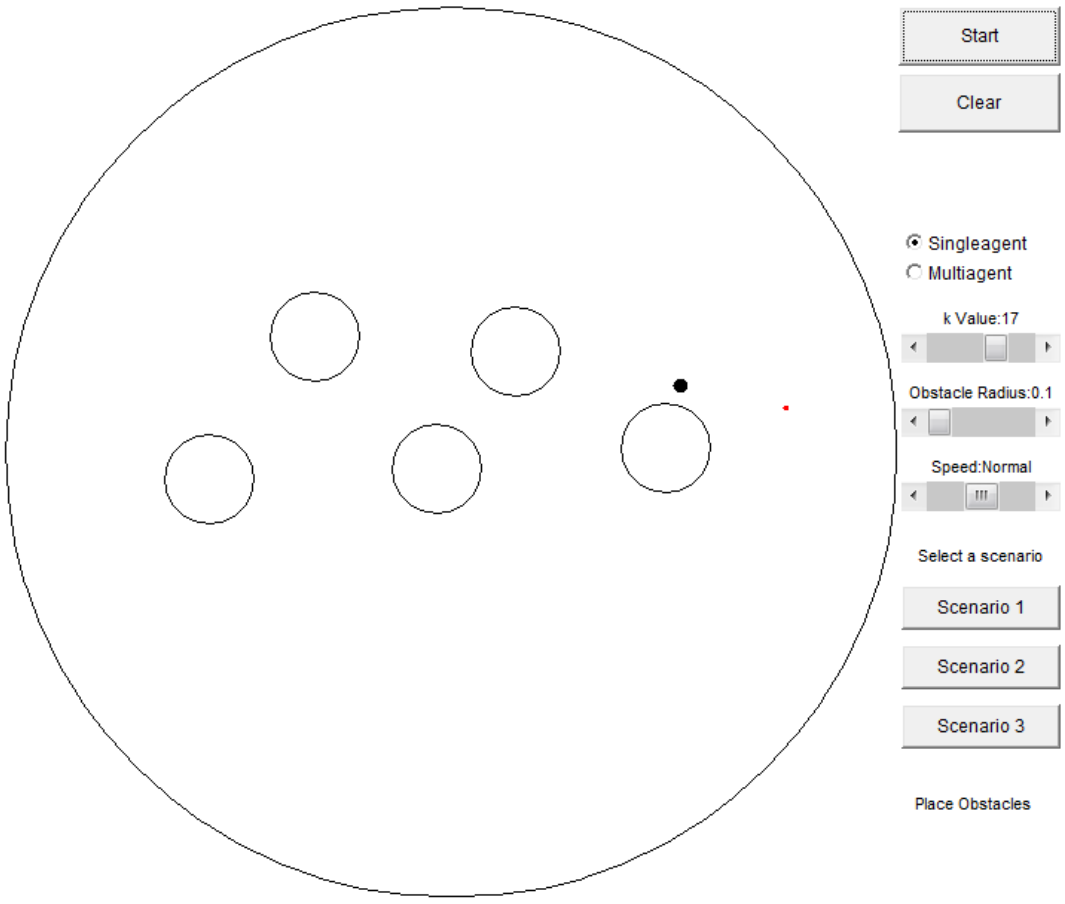

Start
Clear
Singleagent
Multiagent
k Value:17
Obstacle Radius:0.1
Speed:Normal
Select a scenario
Scenario 1
Scenario 2
Scenario 3
Place Obstacles

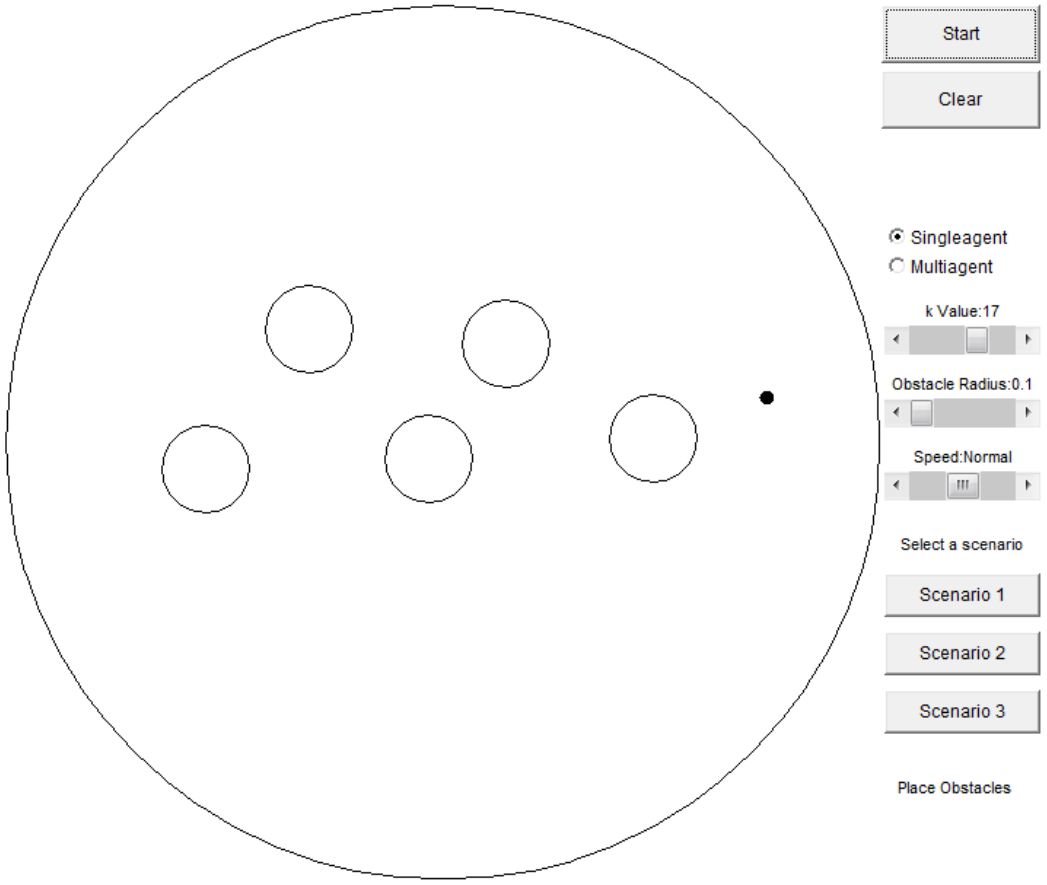
Σενάριο 2


Start
Clear
Singleagent
Multiagent
k Value:17
Obstacle Radius:0.1
Speed:Normal
Select a scenario
Scenario 1
Scenario 2
Scenario 3
Place Obstacles
Start
Clear
Singleagent
Multiagent
k Value:17
Obstacle Radius:0.1
Speed:Normal
Select a scenario
Scenario 1
Scenario 2
Scenario 3
Place Obstacles

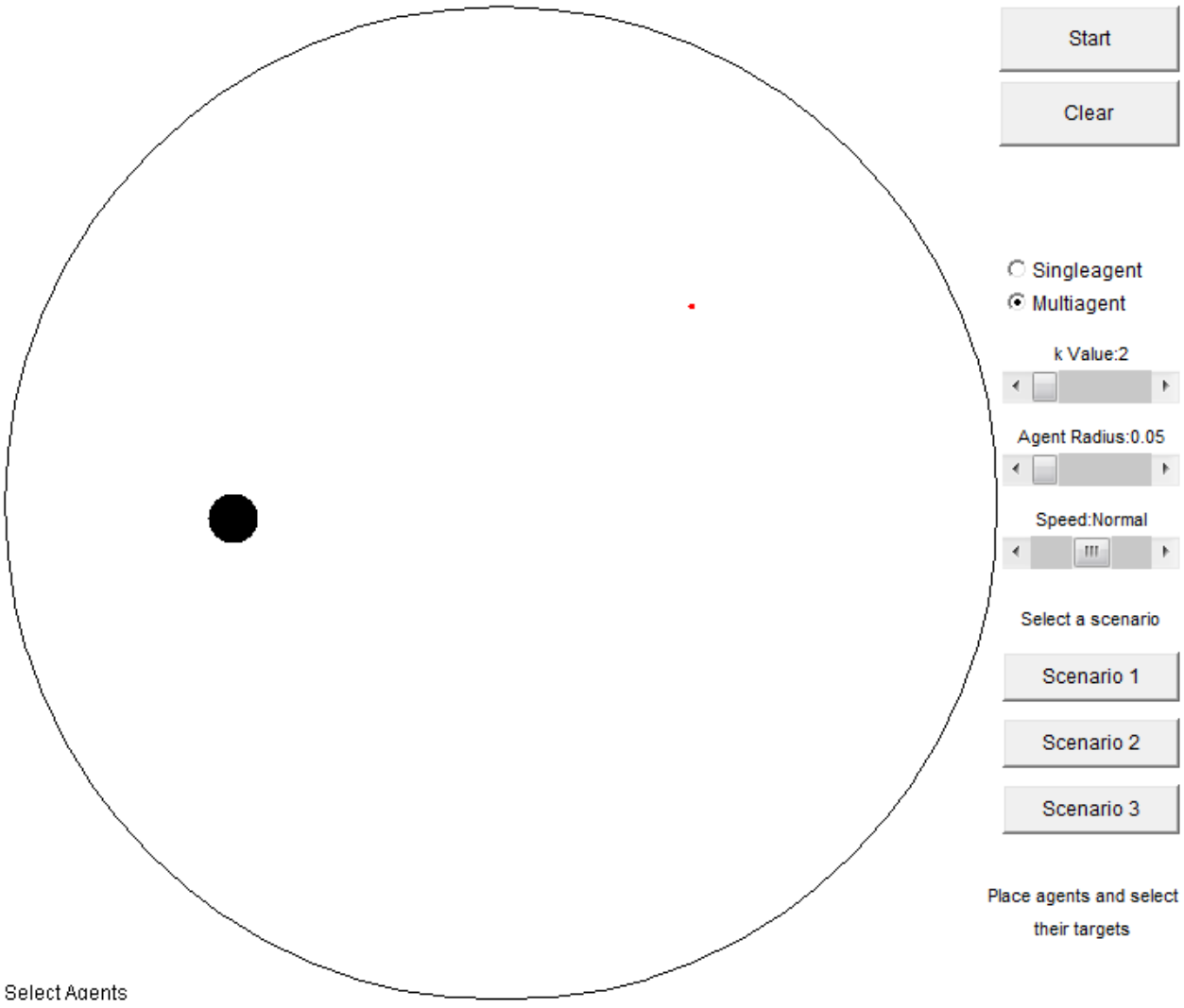

Start
Clear
Singleagent
Multiagent
k Value:17
Obstacle Radius:0.1
Speed:Normal
Select a scenario
Scenario 1
Scenario 2
Scenario 3
Place Obstacles


Start
Clear
Singleagent
Multiagent
k Value:17
Obstacle Radius:0.1
Speed:Normal
Select a scenario
Scenario 1
Scenario 2
Scenario 3
Place Obstacles


Start
Clear
Singleagent
Multiagent
k Value:17
Obstacle Radius:0.1
Speed:Normal
Select a scenario
Scenario 1
Scenario 2
Scenario 3
Place Obstacles

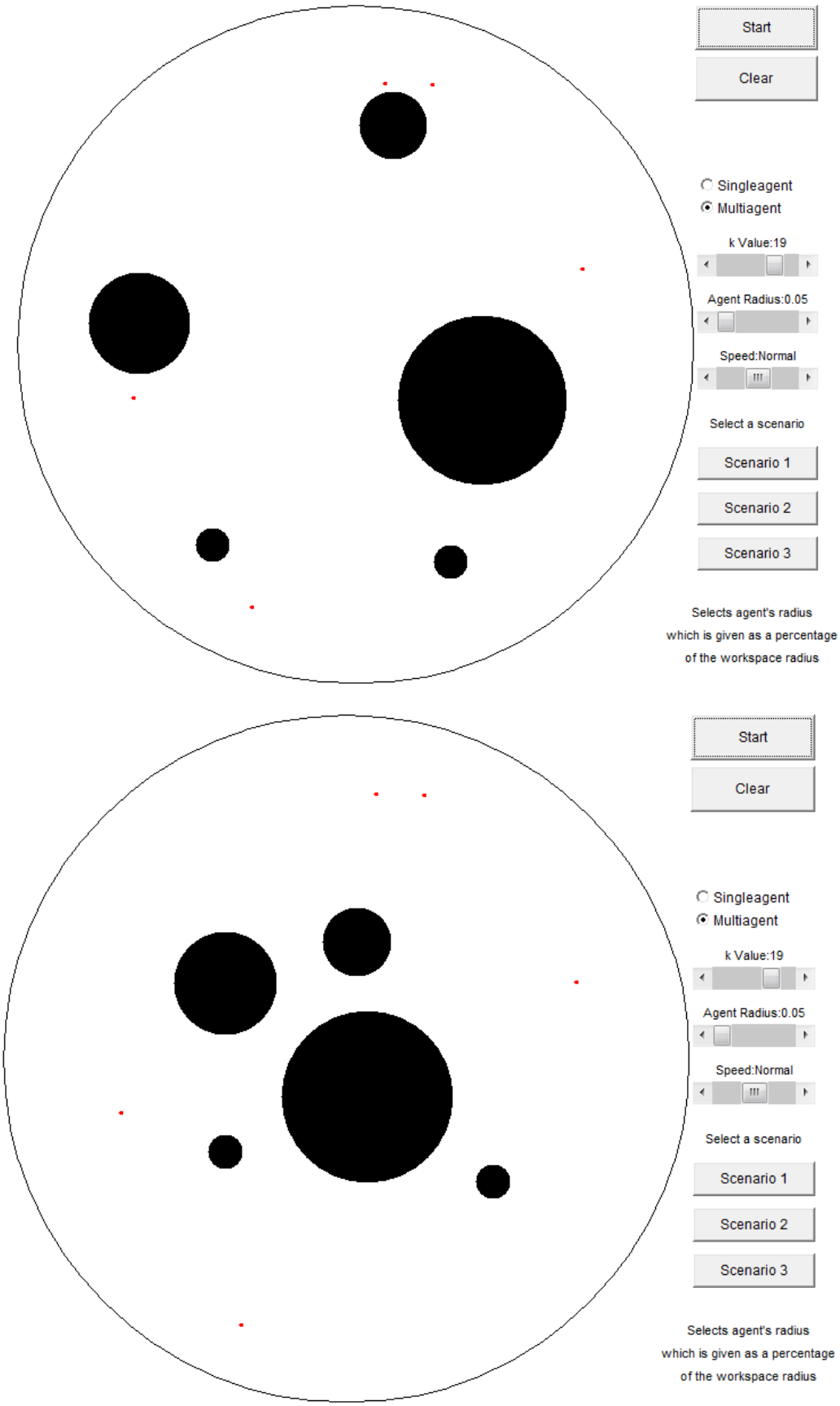

Start
Clear
Singleagent
Multiagent
k Value:17
Obstacle Radius:0.1
Speed:Normal
Select a scenario
Scenario 1
Scenario 2
Scenario 3
Place Obstacles

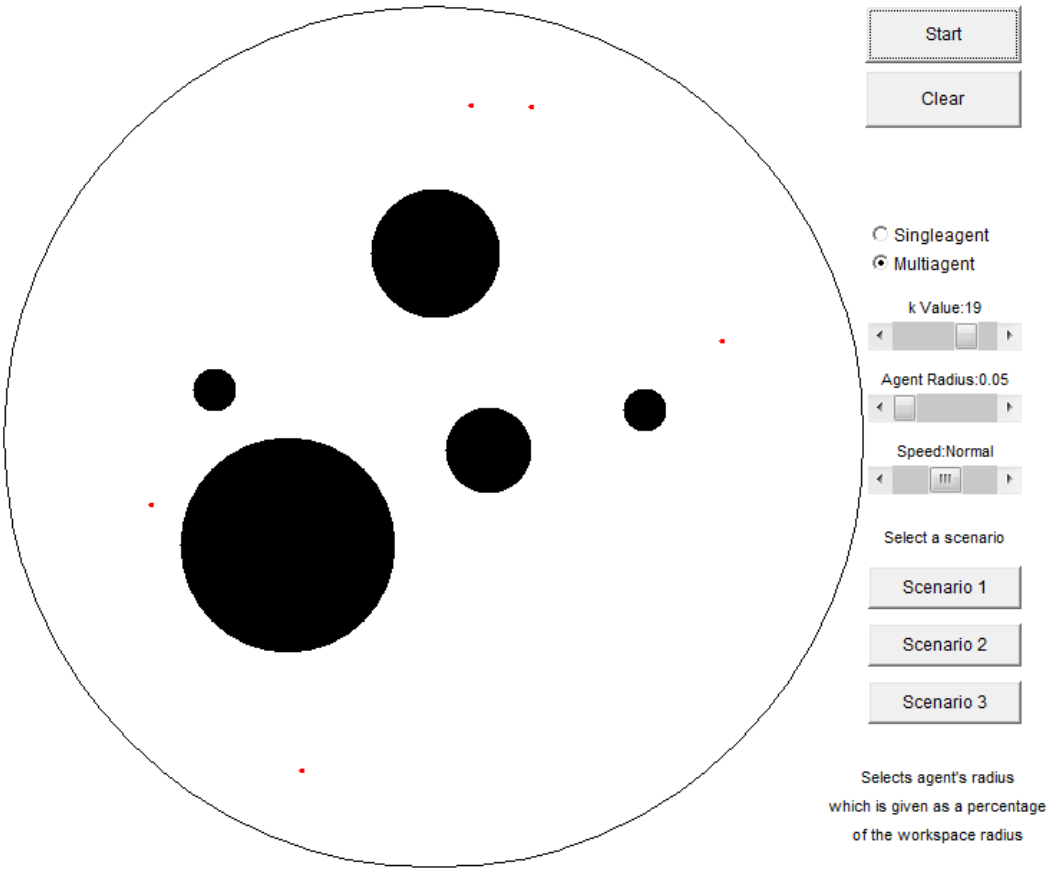
1.3Επιλογή Multi-Agent

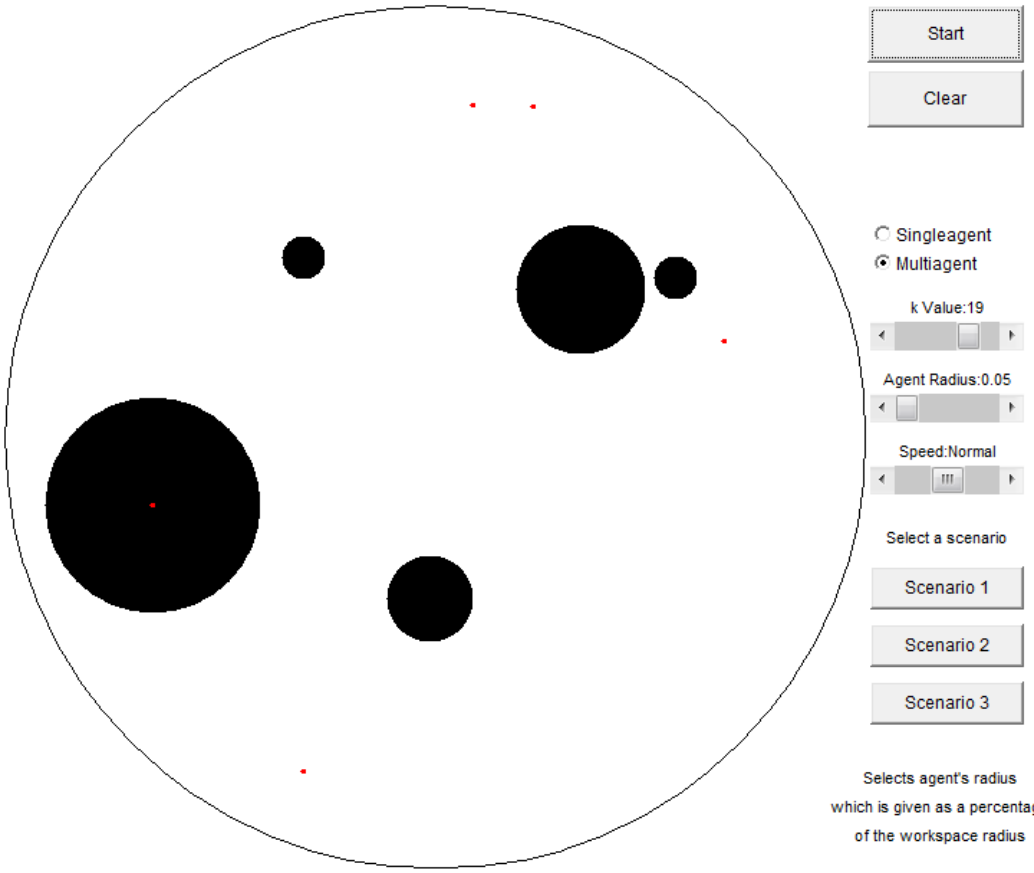


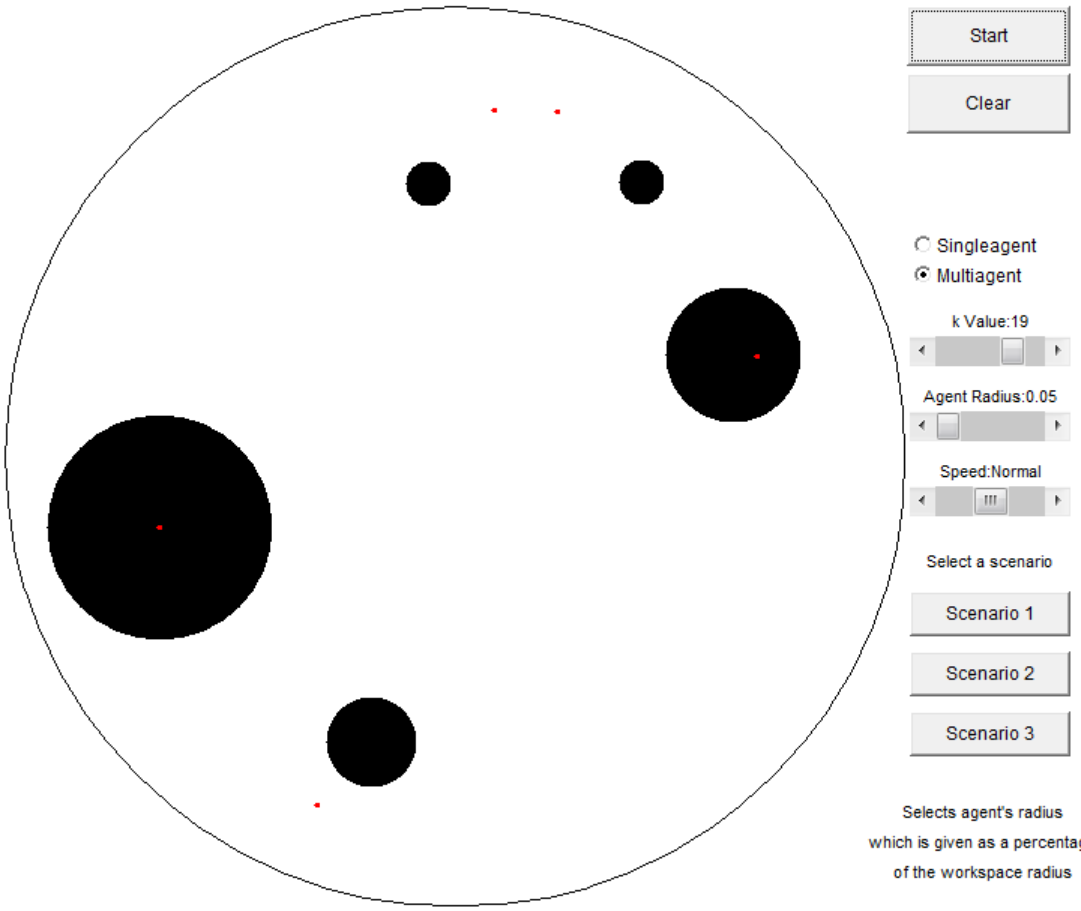
Εδώ πέρα έχουμε την επιλογή πολλών agent που κινούνται μαζί. Οι επιλογές είναι αντίστοιχες με πριν. Στην αρχή μπορούμε να επιλέξουμε με την σειρά την θέση του κάθε agent και μετά τον προορισμό του όπως φαίνεται στο σχήμα. Μπορούμε να επιλέξουμε διάφορες ακτίνες, τιμές k, καθώς και ταχύτητες εξομοίωσης. Εδώ επίσης πρέπει να προσέχουμε να μην αλληλεπικαλύπτονται οι agents καθώς και τα σημεία προορισμού να μην είναι αρκετά κοντά έτσι ώστε όταν φτάσουν οι agents στον στόχο τους να μην υπάρχει αρκετός χώρος. Παρακάτω δίνουμε δύο σενάρια όπως και πριν.

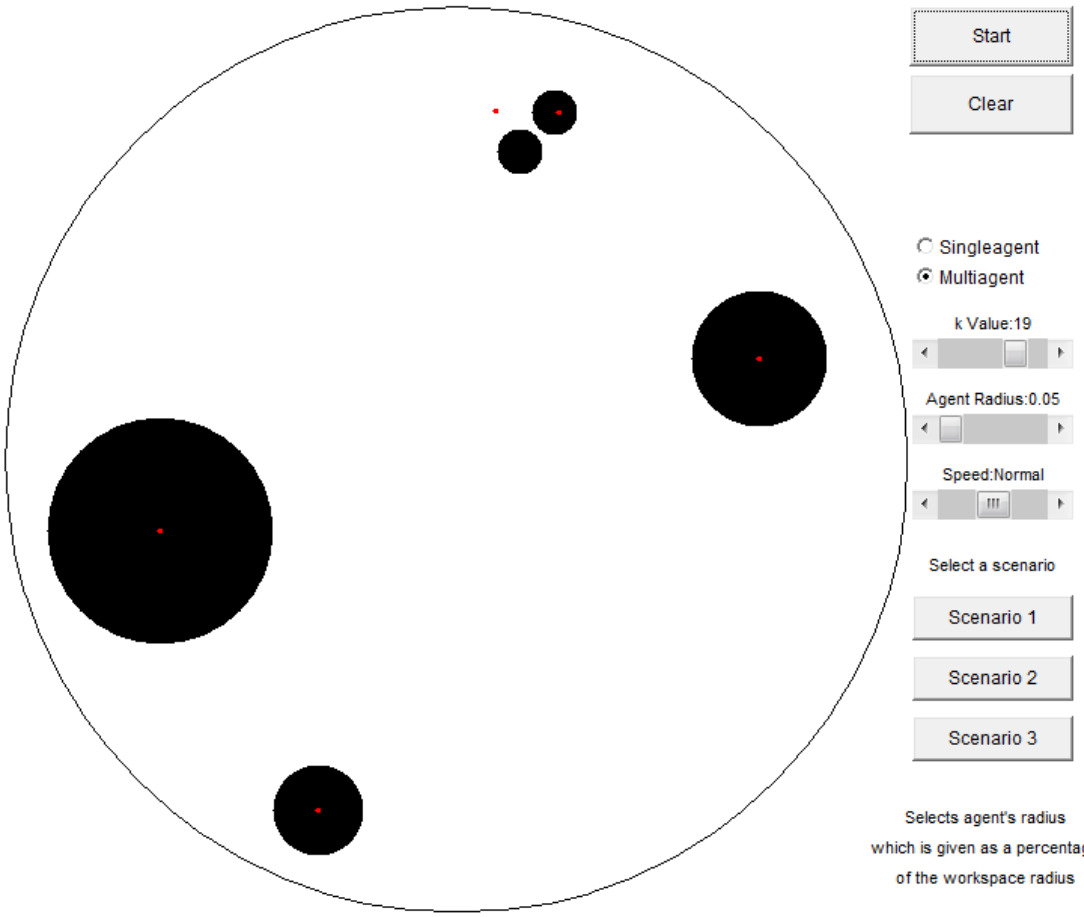
Σενάριο 1


Start
Clear
Singleagent
Multiagent
k Value:19
Agent Radius:0.05
Speed:Normal
Select a scenario
Scenario 1
Scenario 2
Scenario 3
Selects agent's radius
which is given as a percentage
of the workspace radius
Start
Clear
Singleagent
Multiagent
k Value:19
Agent Radius:0.05
Speed:Normal
Select a scenario
Scenario 1
Scenario 2
Scenario 3
Selects agent's radius
which is given as a percentage
of the workspace radius

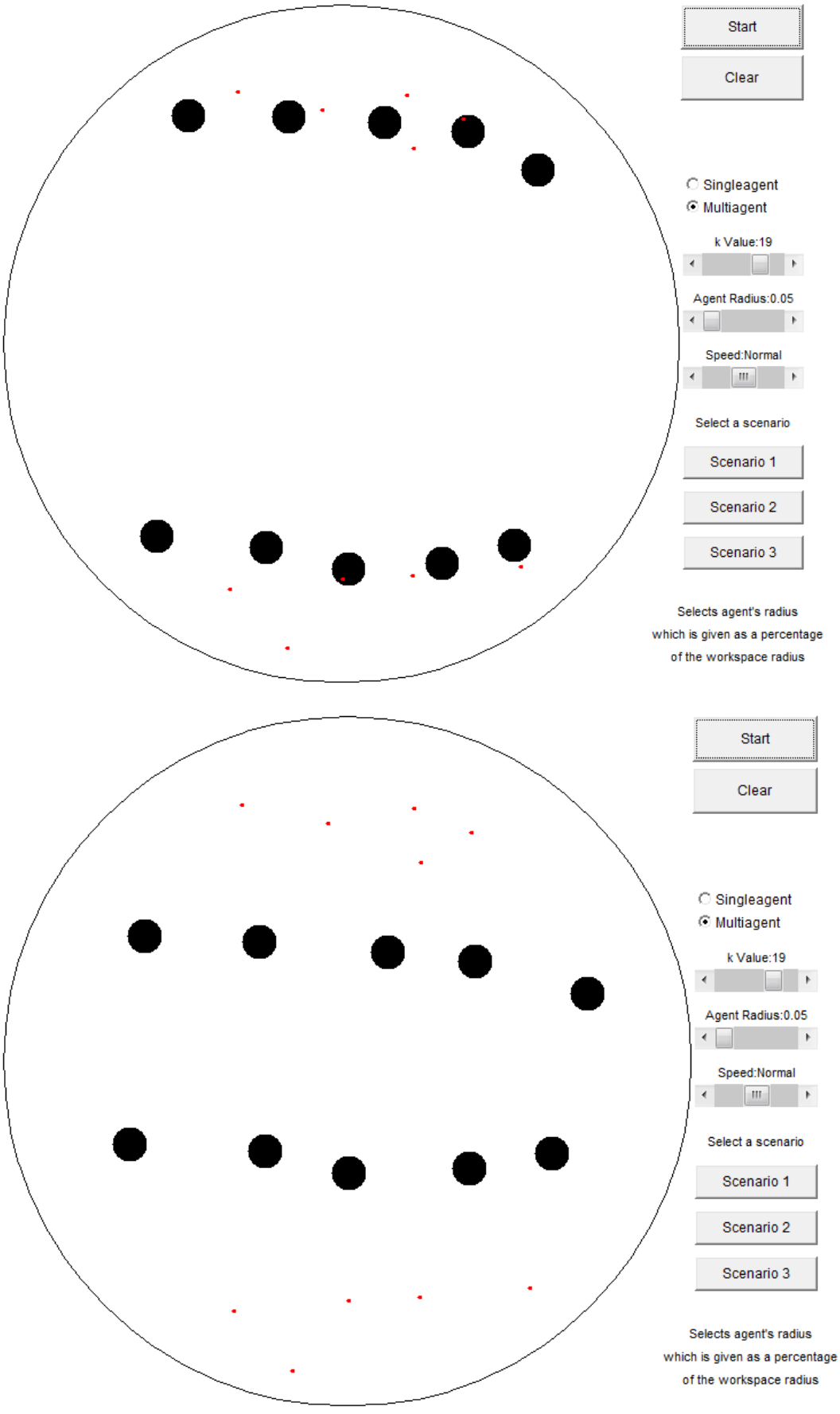

Start
Clear
Singleagent
Multiagent
k Value:19
Agent Radius:0.05
Speed:Normal
Select a scenario
Scenario 1
Scenario 2
Scenario 3
Selects agent's radius
which is given as a percentage
of the workspace radius

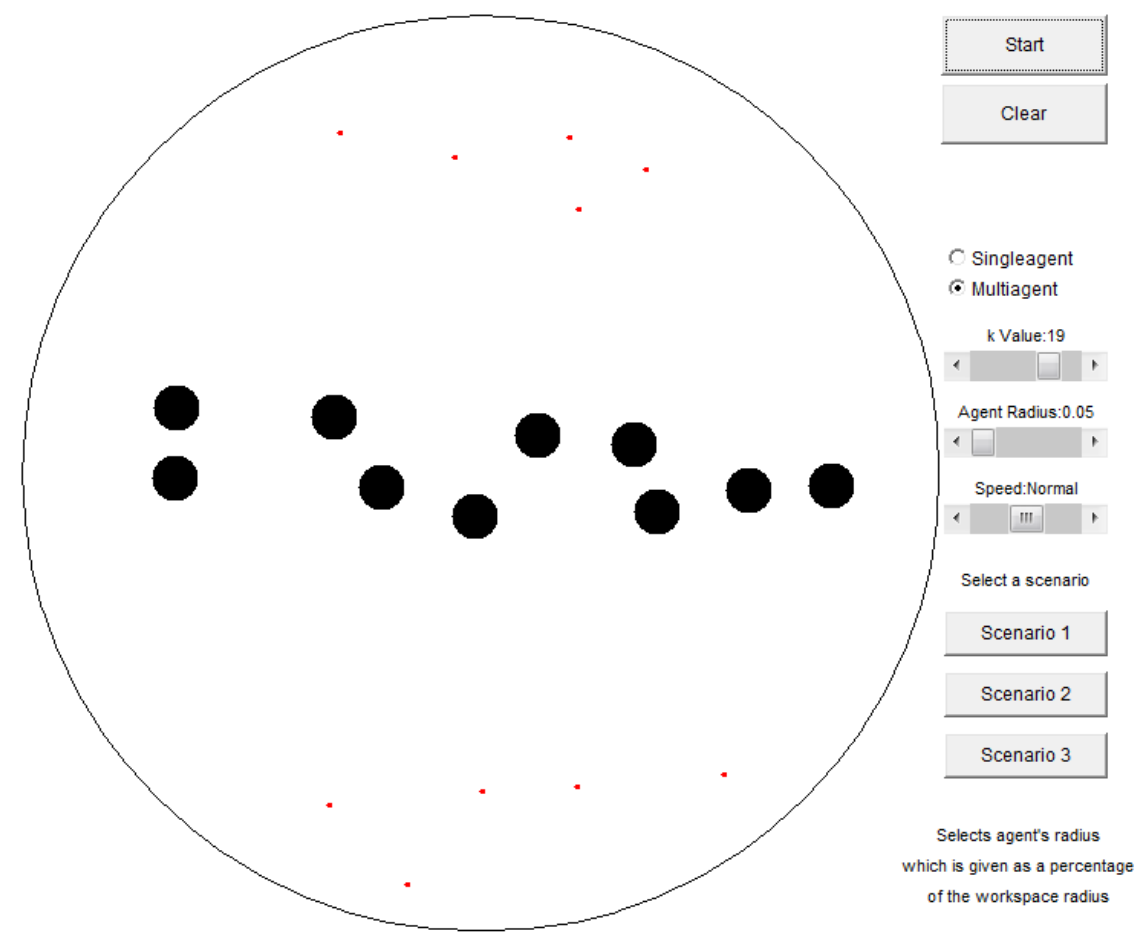

Start
Clear
Singleagent
Multiagent
k Value:19
Agent Radius:0.05
Speed:Normal
Select a scenario
Scenario 1
Scenario 2
Scenario 3
Selects agent's radius
which is given as a percentage
of the workspace radius

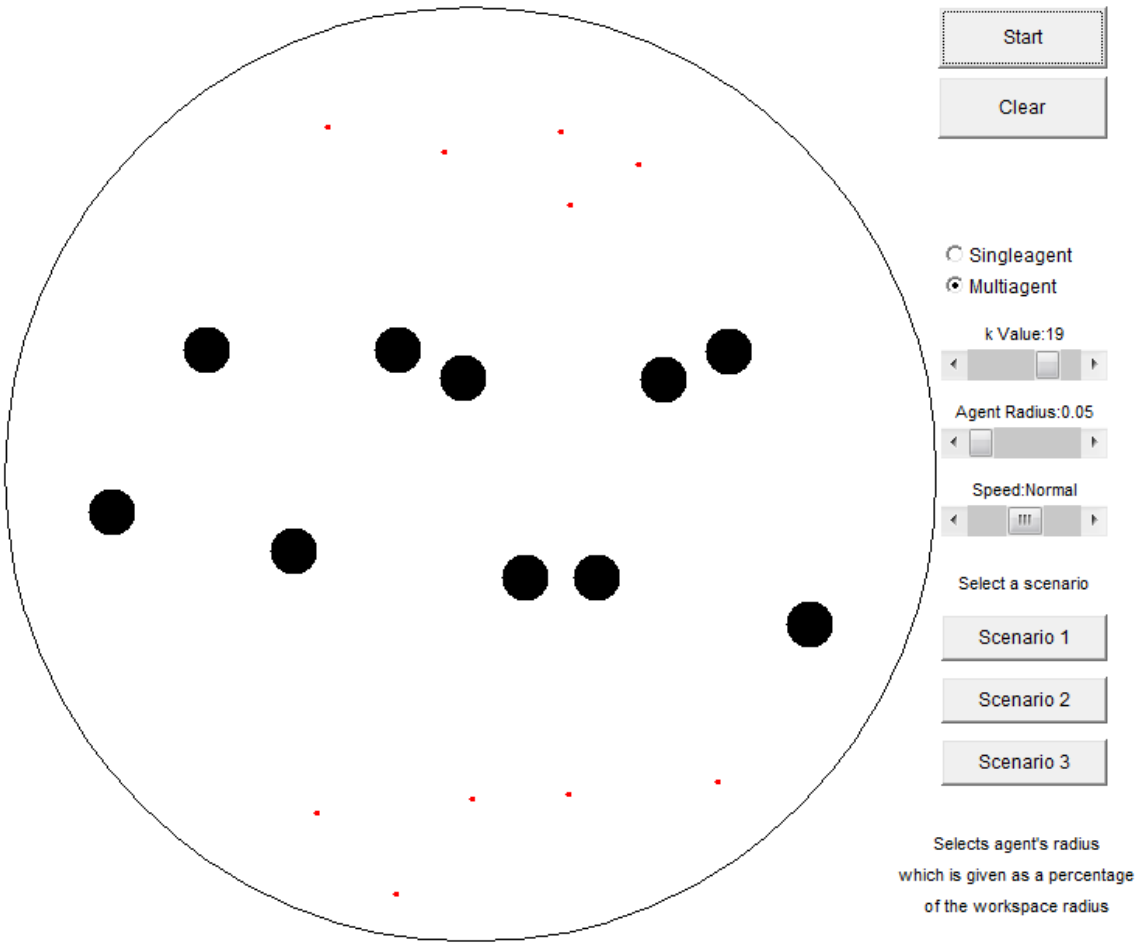

Start
Clear
Singleagent
Multiagent
k Value:19
Agent Radius:0.05
Speed:Normal
Select a scenario
Scenario 1
Scenario 2
Scenario 3
Selects agent's radius
which is given as a percentage
of the workspace radius

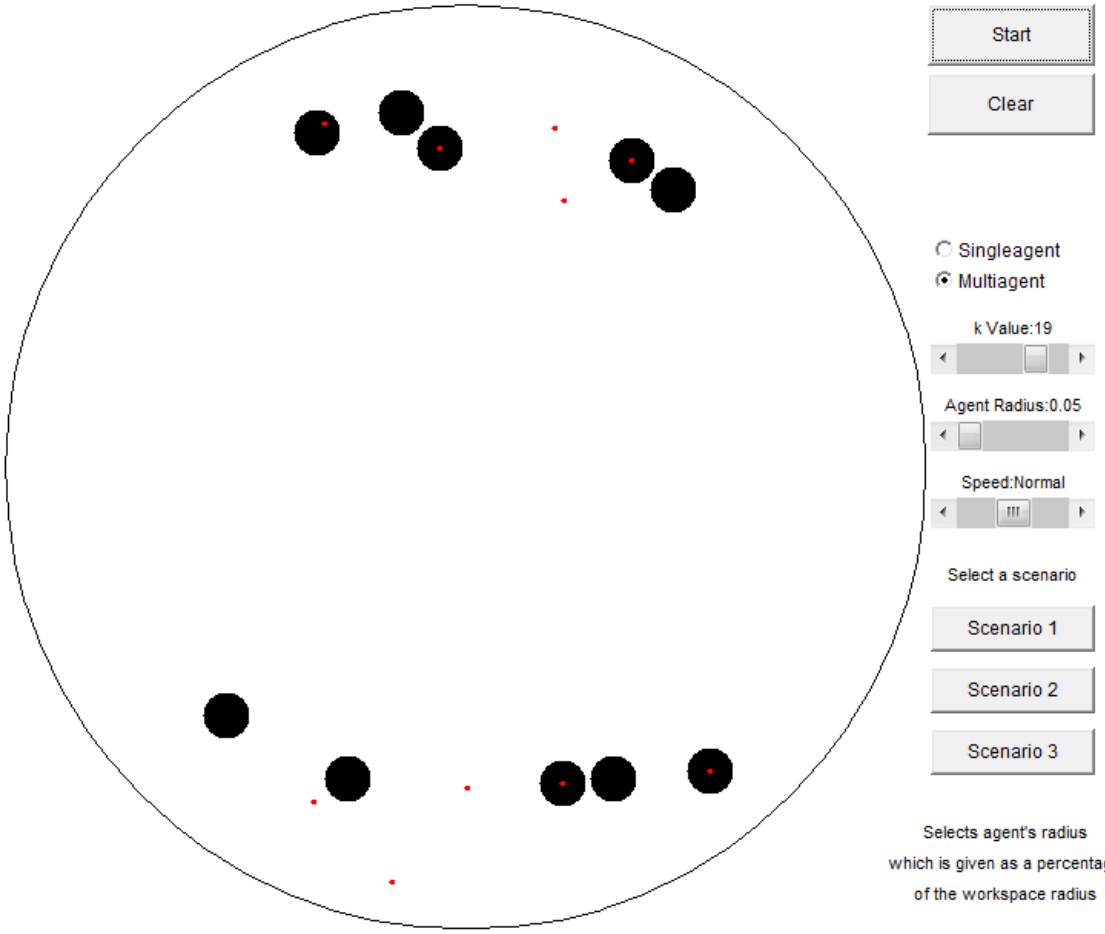

Start
Clear
Singleagent
Multiagent
k Value:19
Agent Radius:0.05
Speed:Normal
Select a scenario
Scenario 1
Scenario 2
Scenario 3
Selects agent's radius
which is given as a percentage
of the workspace radius

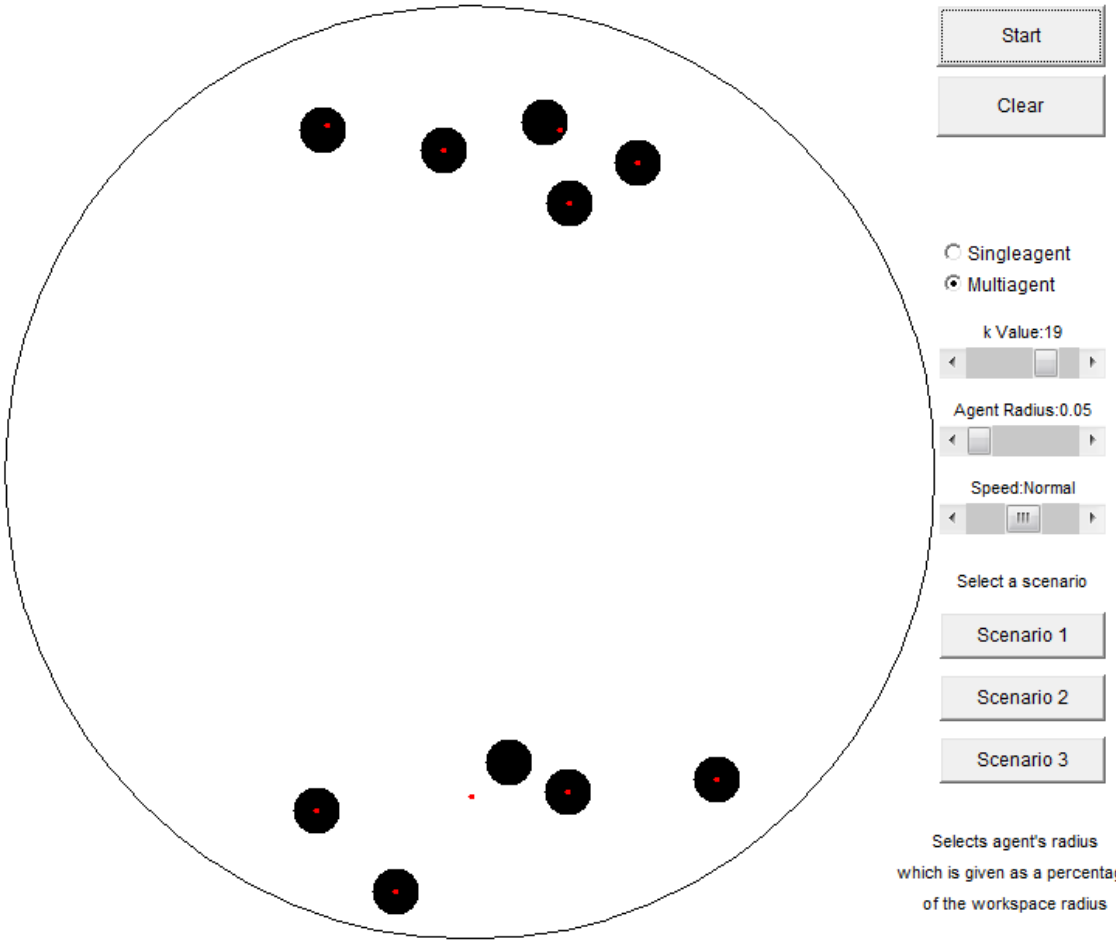
Σενάριο 2


Start
Clear
Singleagent
Multiagent
k Value:19
Agent Radius:0.05
Speed:Normal
Select a scenario
Scenario 1
Scenario 2
Scenario 3
Selects agent's radius
which is given as a percentage
of the workspace radius
Start
Clear
Singleagent
Multiagent
k Value:19
Agent Radius:0.05
Speed:Normal
Select a scenario
Scenario 1
Scenario 2
Scenario 3
Selects agent's radius
which is given as a percentage
of the workspace radius


Start
Clear
Singleagent
Multiagent
k Value:19
Agent Radius:0.05
Speed:Normal
Select a scenario
Scenario 1
Scenario 2
Scenario 3
Selects agent's radius
which is given as a percentage
of the workspace radius


Start
Clear
Singleagent
Multiagent
k Value:19
Agent Radius:0.05
Speed:Normal
Select a scenario
Scenario 1
Scenario 2
Scenario 3
Selects agent's radius
which is given as a percentage
of the workspace radius


Start
Clear
Singleagent
Multiagent
k Value:19
Agent Radius:0.05
Speed:Normal
Select a scenario
Scenario 1
Scenario 2
Scenario 3
Selects agent's radius
which is given as a percentage
of the workspace radius

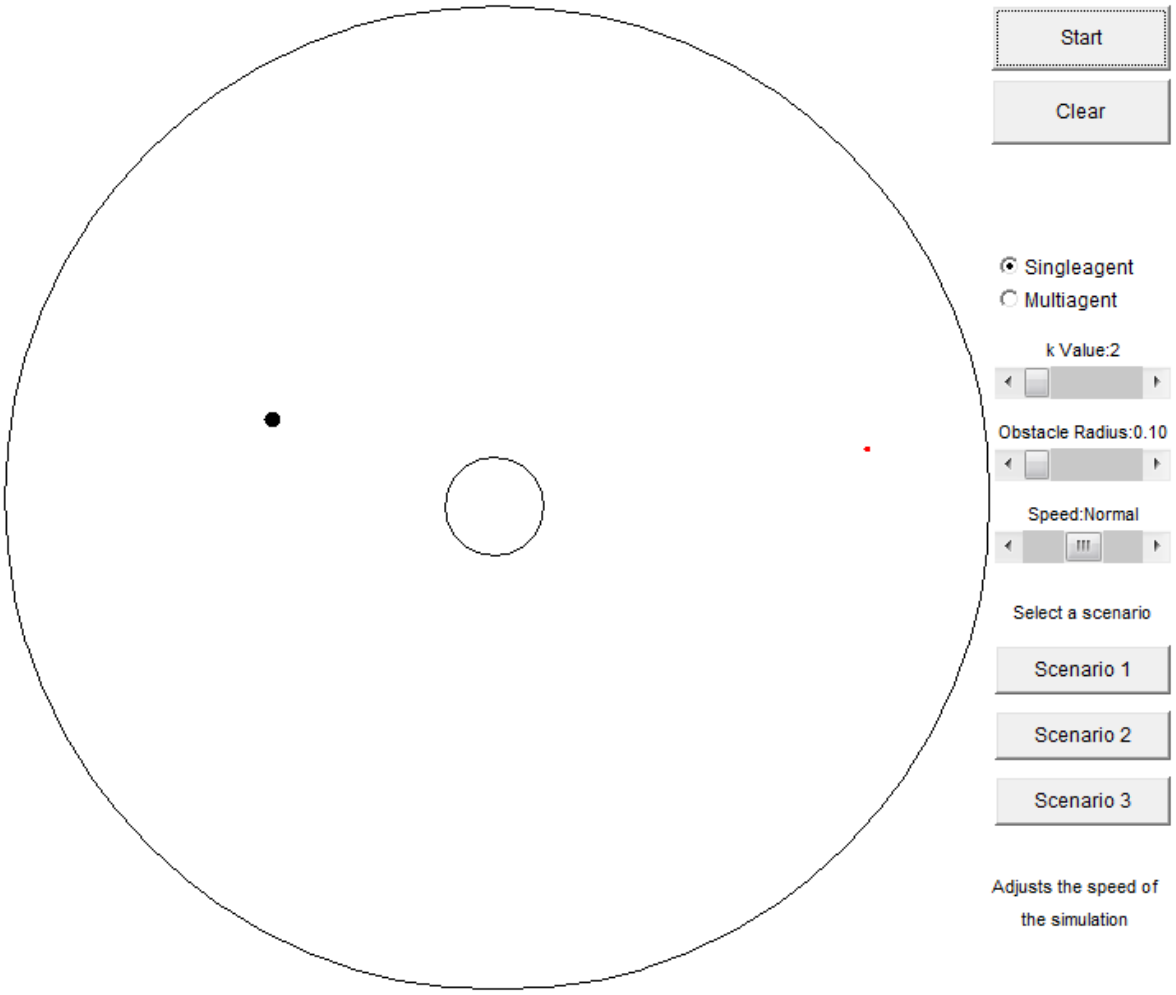

Start
Clear
Singleagent
Multiagent
k Value:19
Agent Radius:0.05
Speed:Normal
Select a scenario
Scenario 1
Scenario 2
Scenario 3
Selects agent's radius
which is given as a percentage
of the workspace radius

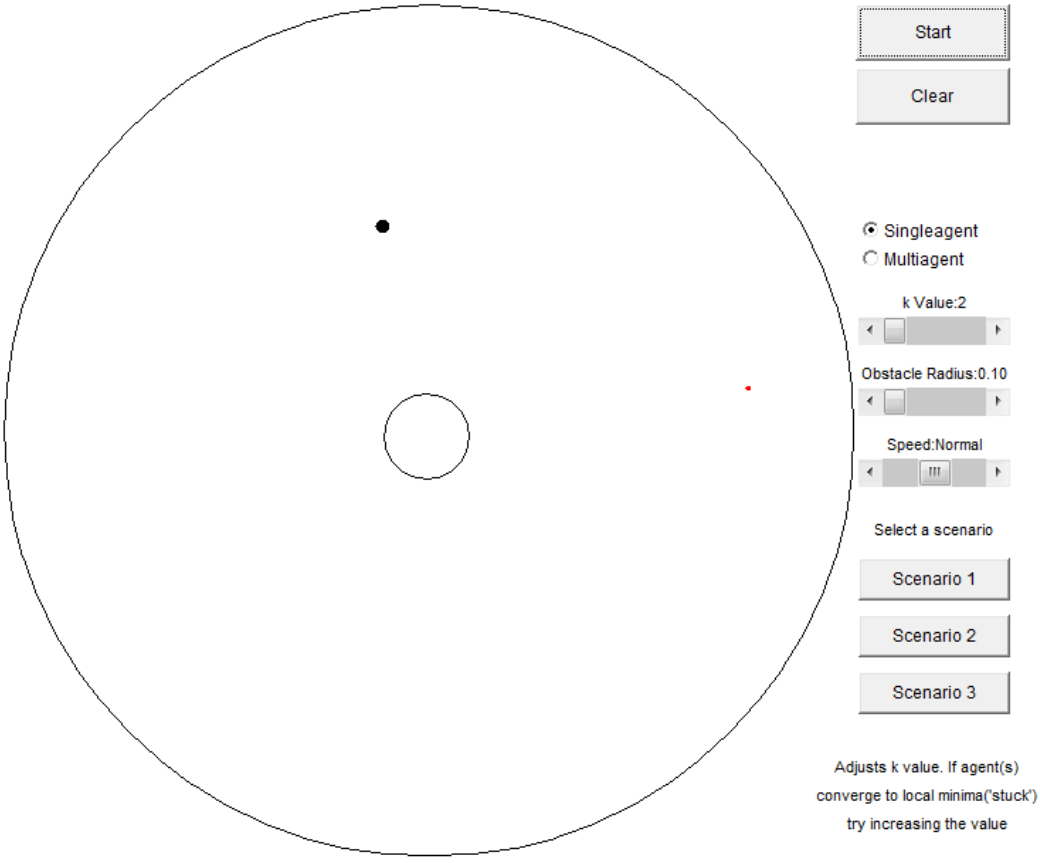
1.4 Συμπληρωματικά

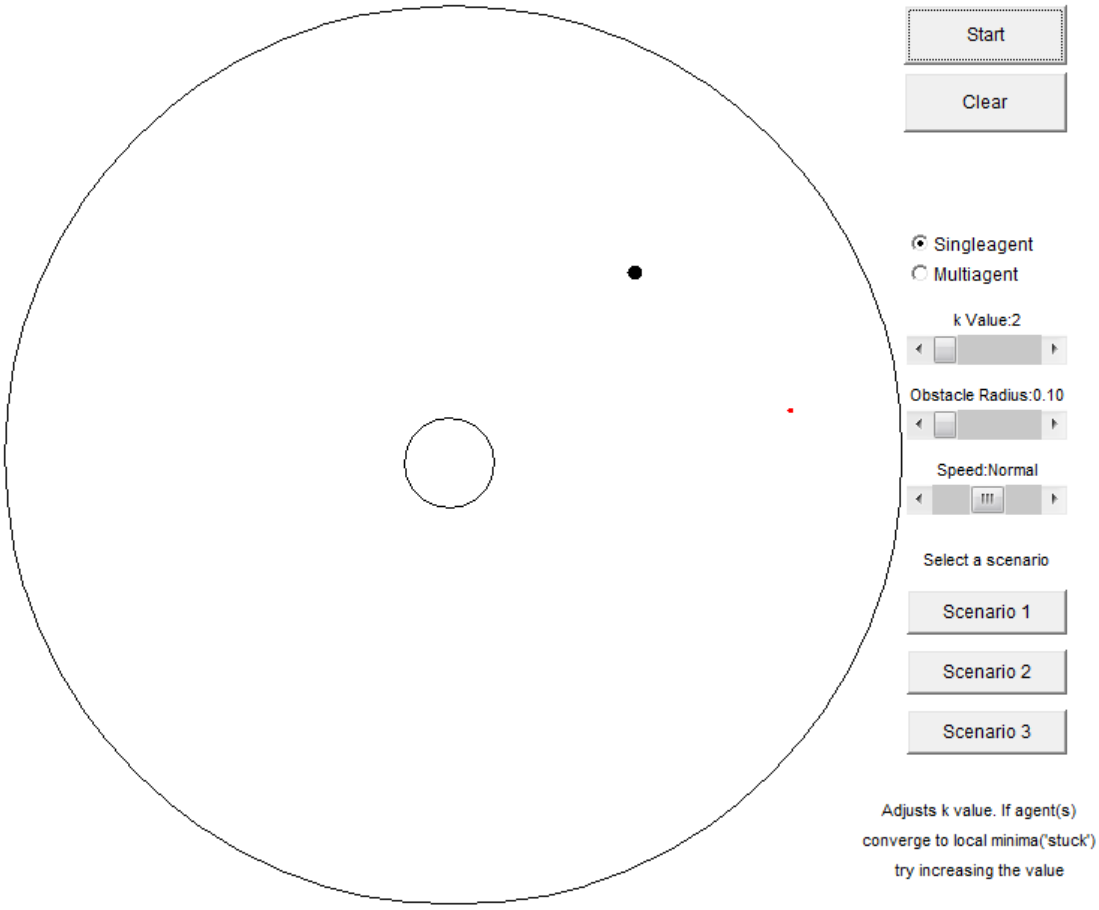
Εδώ είναι μία καλή ευκαιρία να δείξουμε δύο περιπτώσεις σχετικά με την επίδραση του k στις συναρτήσεις πλοήγησης. Αρχικά θα δείξουμε το πως επιδράει το k στην πορεία που ακολουθεί κάποιος agent.

1.4.1Επίδραση του k στην διαδρομή του agent

k=2

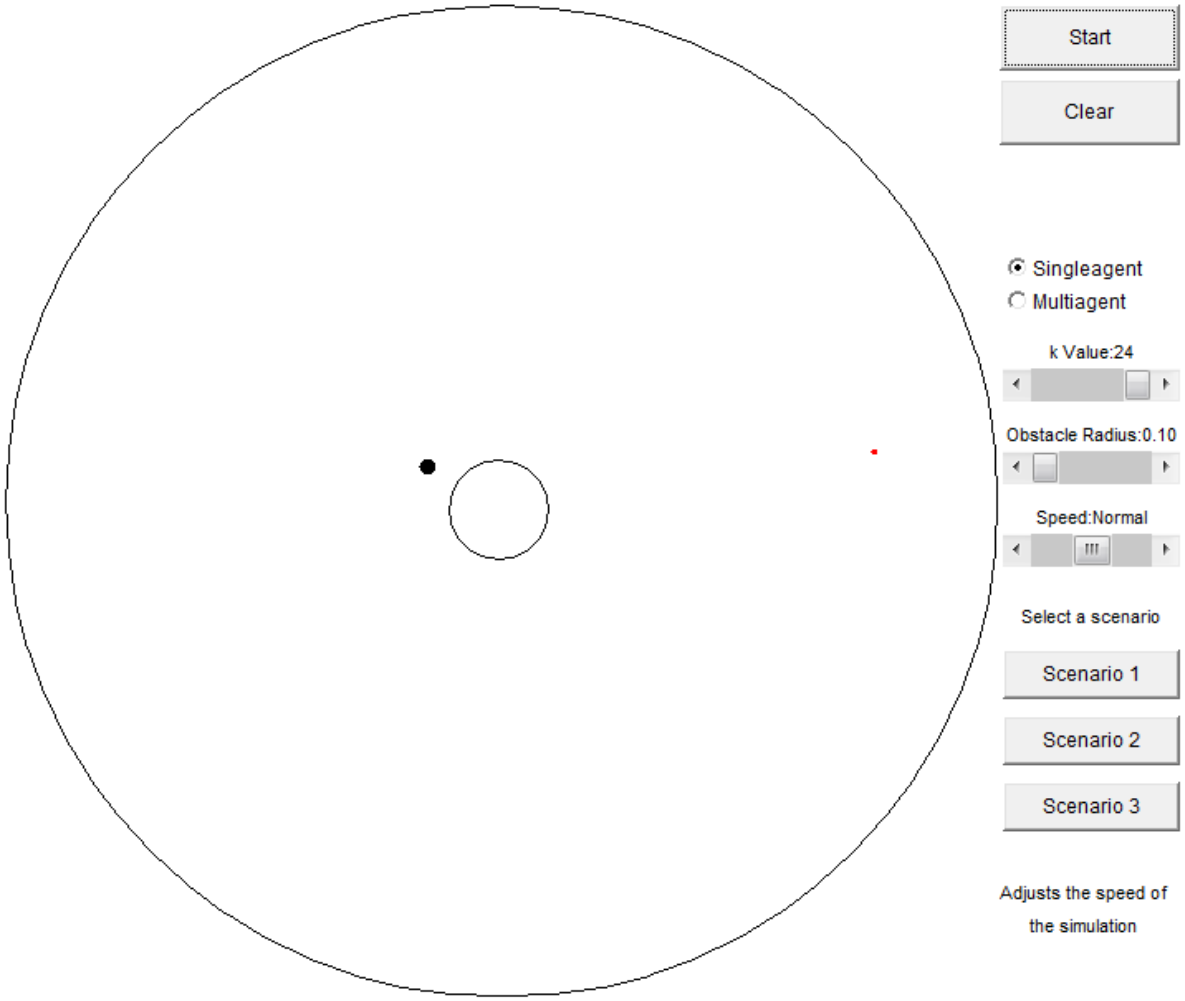


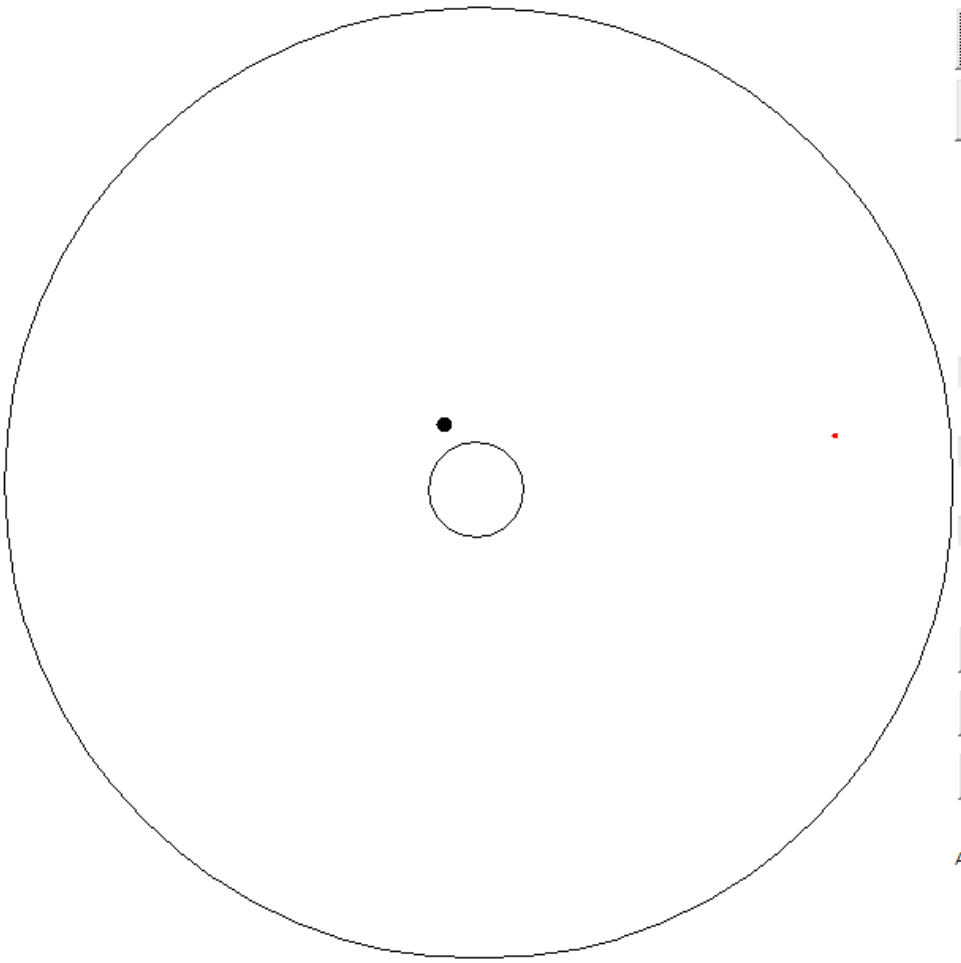

Start
Clear
Singleagent
Multiagent
k Value:2
Obstacle Radius:0.10
Speed:Normal
Select a scenario
Scenario 1
Scenario 2
Scenario 3
Adjusts k value. If agent(s)
converge to local minima('stuck')
try increasing the value

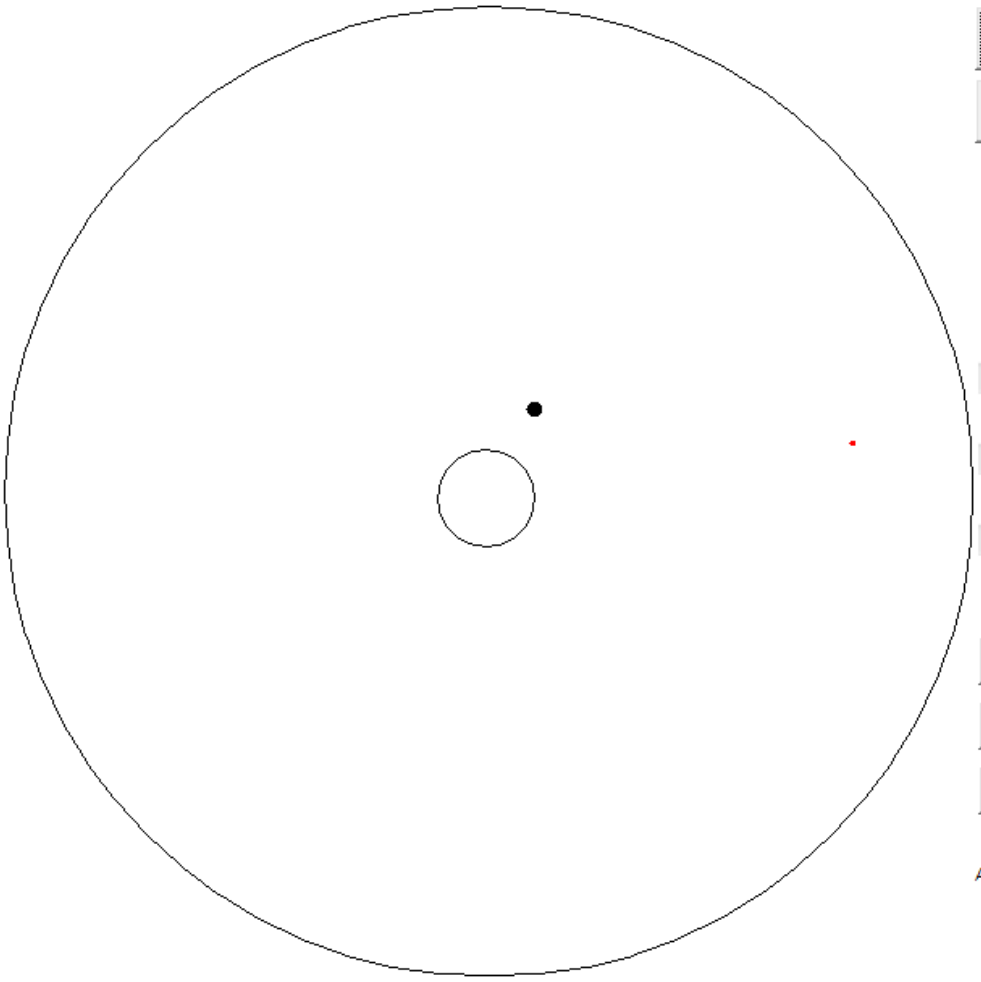

Start
Clear
Singleagent
Multiagent
k Value:2
Obstacle Radius:0.10
Speed:Normal
Select a scenario
Scenario 1
Scenario 2
Scenario 3
Adjusts k value. If agent(s)
converge to local minima('stuck')
try increasing the value

Εδώ έχουμε ένα απλό σενάριο ενός εμποδίου στον δρόμου του agent. Στην πρώτη εξομοίωση επιλέχτηκε k=2. Παρατηρούμε το πόσο μακριά από το εμπόδιο κινείται ο agent ή ισοδύναμα από πόσο μακριά αισθάνεται το δυναμικό πεδίο του εμποδίου.

k=24

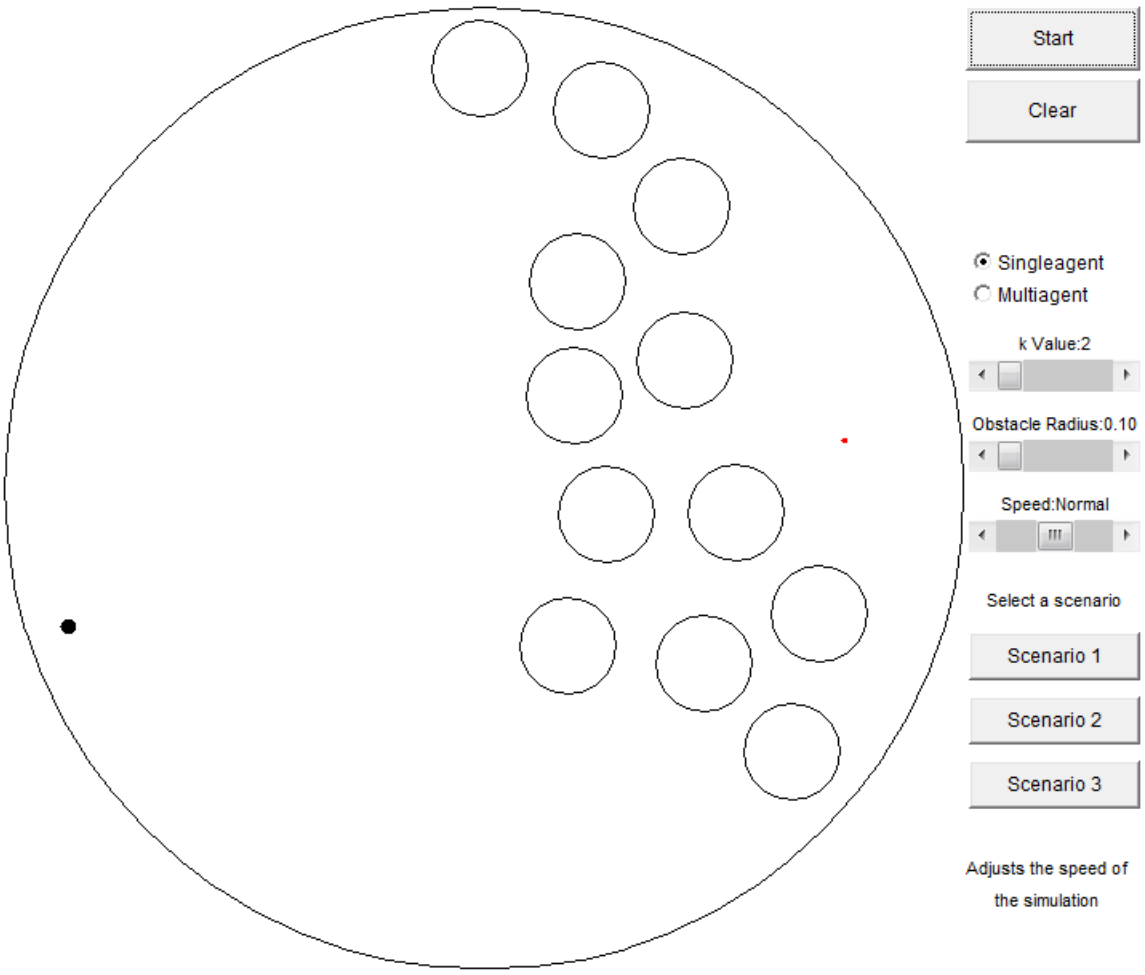



Start
Clear
Singleagent
Multiagent
k Value:24
Obstacle Radius:0.10
Speed:Normal
Select a scenario
Scenario 1
Scenario 2
Scenario 3
Adjusts the speed of the simulation

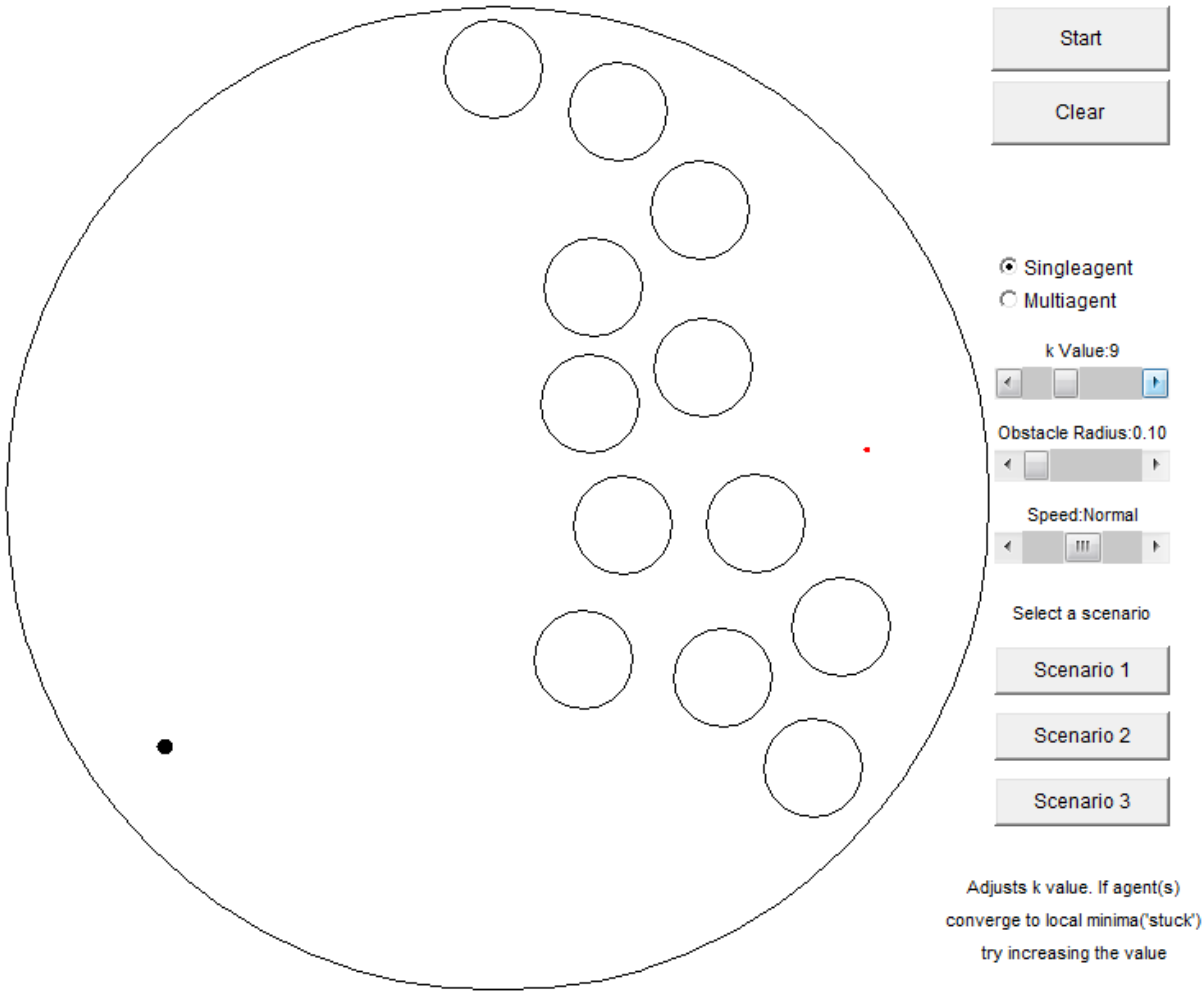

Start
Clear
Singleagent
Multiagent
k Value:24
Obstacle Radius:0.10
Speed:Normal
Select a scenario
Scenario 1
Scenario 2
Scenario 3
Adjusts the speed of the simulation

Εδώ τρέχουμε την ίδια εξομοίωση αλλά με k=24. Παρατηρούμε ότι ο agent πηγαίνει πολύ κοντά στο εμπόδια (δηλαδή η επίδραση του δυναμικού πεδίου του εμποδίου έχει "μαζέψει" αρκετά κοντά στο εμπόδιο). Εδώ είναι ένα καλό παράδειγμα της επίδρασης του k στο δυναμικό πεδίο. Ένα από τα αποτελέσματα που έχει το k στο δυναμικό πεδίο είναι να ενισχύει την ελκτική επίδραση του προορισμού και να μικραίνει την απωστική επίδραση των εμποδίων. Ρυθμίζοντας το k αντίστοιχα μπορούμε να κάνουμε τον κινούμενο agent να καταλαβαίνει την επίδραση κάποιου εμποδίου από μεγαλύτερες αποστάσεις. Βέβαια αυτή η δυνατότητα δίνεται αφού έχουμε εξασφαλίσει ότι το k είναι αρκετά μεγάλο έτσι ώστε να μην υπάρχουν τοπικά ελάχιστα στον χώρο λειτουργίας. Ένα παράδειγμα αυτής της περίπτωσης δίνεται παρακάτω.

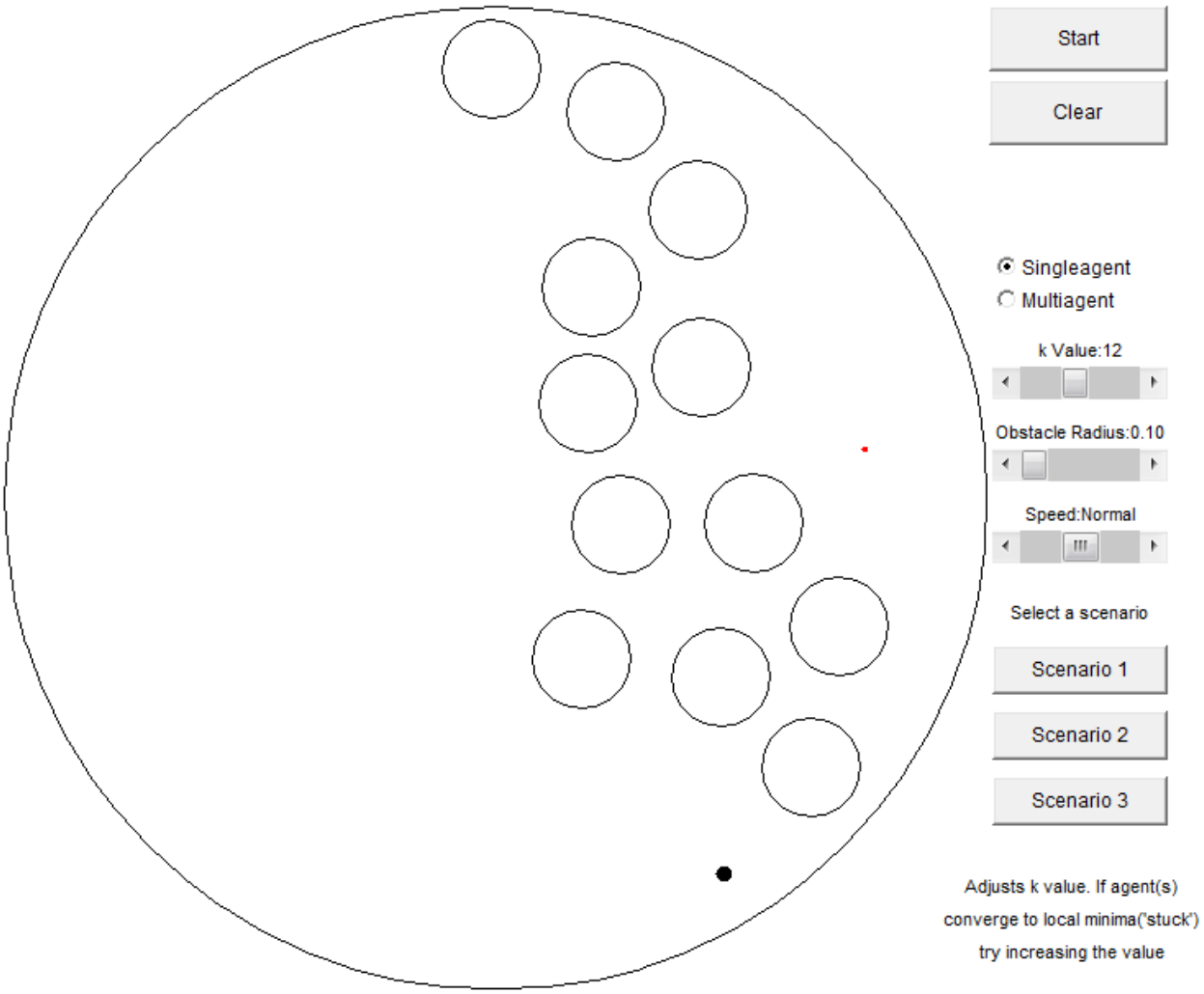
1..4.2Επίδραση του k στα τοπικά ελάχιστα



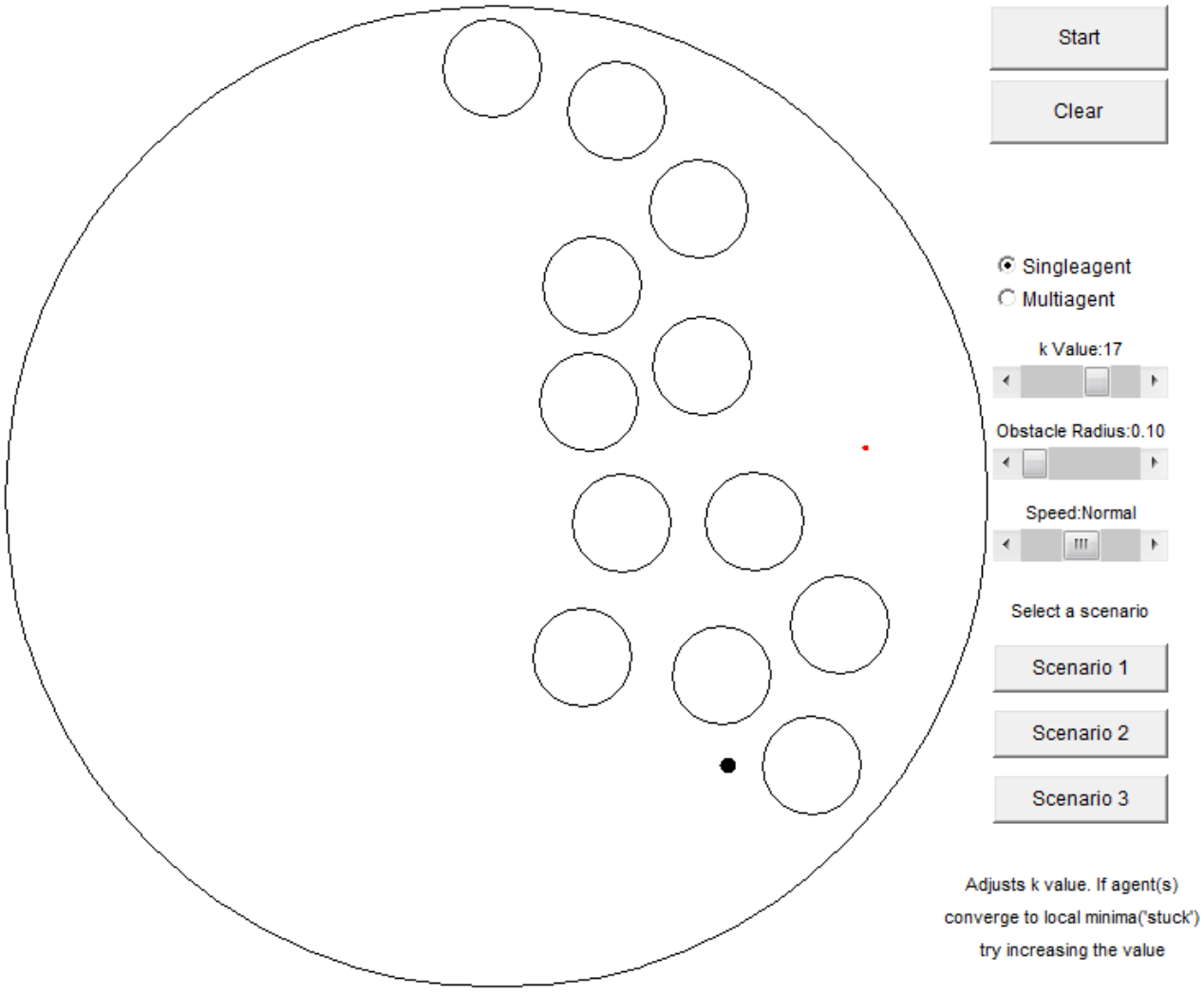
Παραπάνω δίνουμε μία περίπτωση με έναν μεγάλο αριθμό εμποδίων μεταξύ του agent και του προορισμού του. Ο μεγάλος αυτός αριθμός εμποδίων δημιουργεί ένα απωστικό πεδίο που πρακτικά εξουδετερώνει την ελκτική επίδραση του στόχου. Εδώ επίσης σημαντικό ρόλο παίζει και το γεγονός που δείξαμε στο προηγούμενο παράδειγμα ότι δηλαδή ο agent καταλαβαίνει το πεδίο των εμποδίων από μεγαλύτερη απόσταση. Η παραπάνω εικόνα δείχνει τον agent να έχει σταματήσει σε ένα συγκεκριμένο σημείο. Το k έχει επιλεγεί ίσο με δύο. Αυτό είναι μία τυπική περίπτωση όπου ο agent έχει πέσει σε ένα τοπικό ελάχιστο του πεδίου.



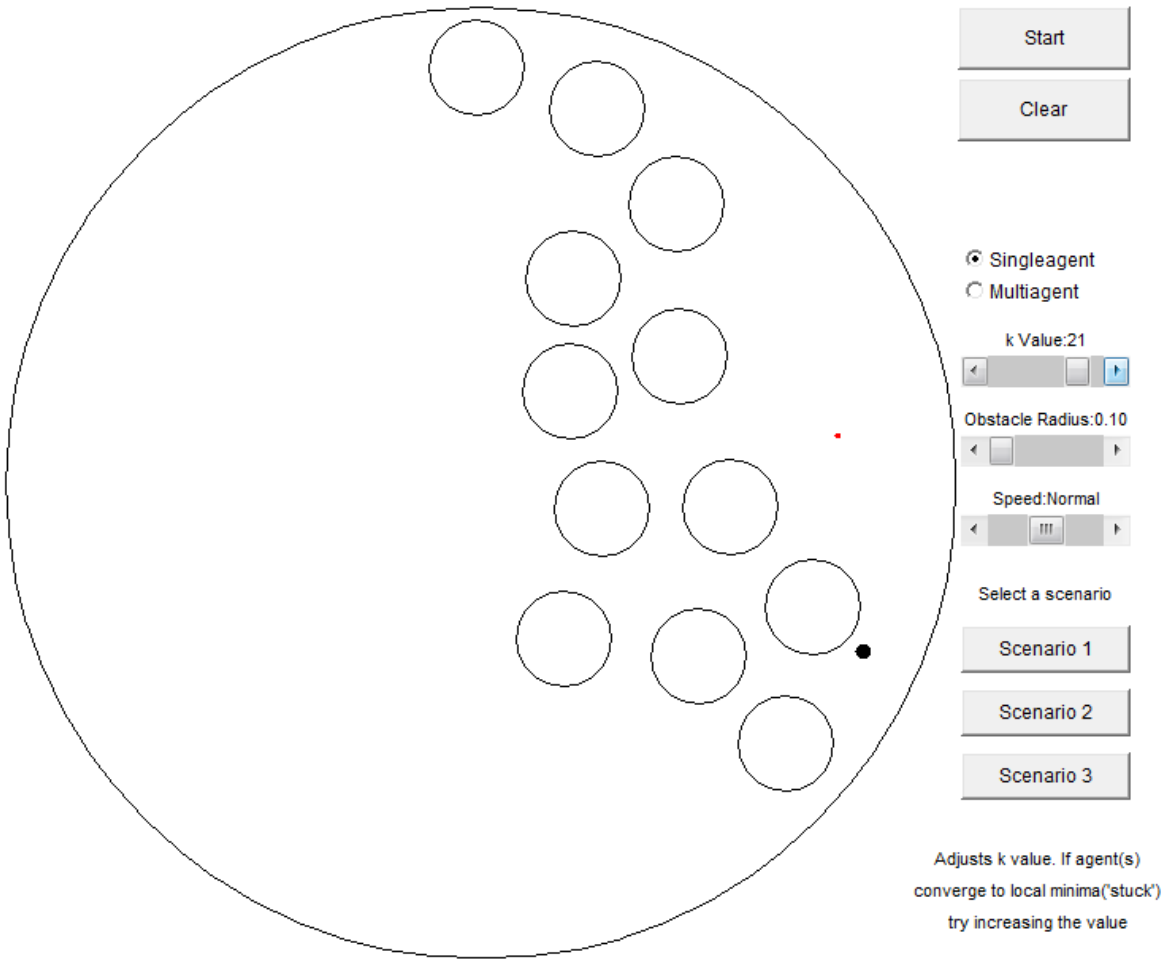
Αυξάνουμε το k σε εννέα. Παρατηρούμε μία μικρή μετατόπιση του agent ο οποίο σταματάει πάλι λίγο πιο μετά. Αυτό ουσιαστικά είναι μία μετατόπιση του τοπικού ελαχίστου από μία θέση σε μία άλλη.

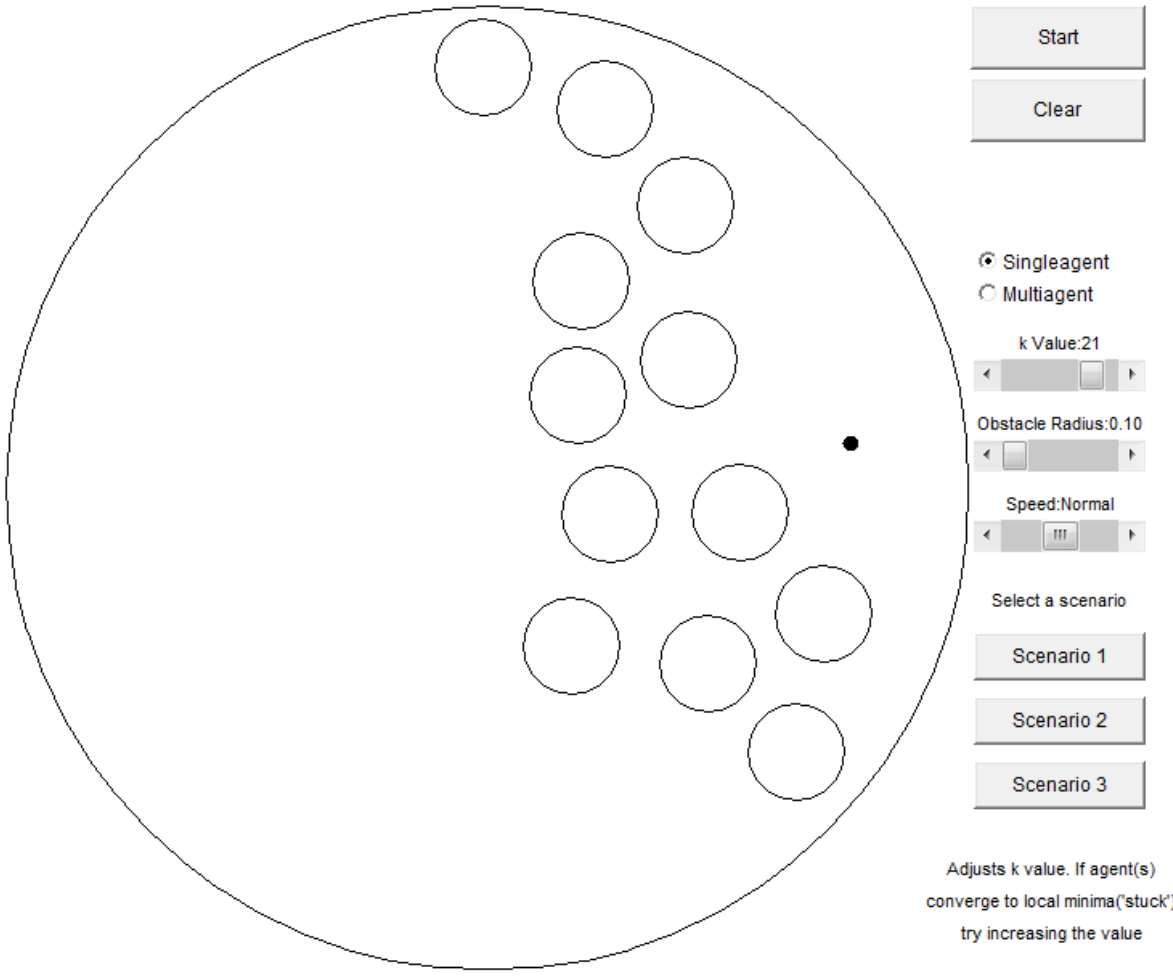


Το k αυξάνεται σε δώδεκα και το τοπικό ελάχιστο μετακινείται ακόμα περισσότερο. Εδώ παρατηρούμε αυτό ακριβώς που αναφέραμε και στο πρώτο κεφάλαιο. Η αύξηση του k τελικά αυξάνει την ελκτική δύναμη του προορισμού οπότε και το τοπικό ελάχιστο μετατοπίζεται προς μία θέση πιο κοντά στα εμπόδια. Μία διαφορετική διατύπωση αυτού είναι ότι με την αύξηση της παραμέτρου k τα τοπικά ελάχιστα του χώρου "μαζεύονται" πολύ κοντά στα εμπόδια



Αυτό είναι το τελευταίο τοπικό ελάχιστο που δείχνουμε. Παρατηρούμε ότι ο agent έχει φτάσει αρκετά κοντά στα εμπόδια και έχει σταματήσει εκεί δηλαδή ισοδύναμα το τοπικό ελάχιστο αυτό έχει φτάσει κοντά στα όρια των εμποδίων. Από την απόδειξη των συναρτήσεων πλοήγησης γνωρίζουμε ότι υπάρχει k τέτοιο ώστε να εξαλείφει τα τοπικά ελάχιστα του χώρου και το μόνο ελάχιστο να είναι αυτό του προορισμού. Τελικά για k ίσο με εικοσιένα τα ολικά ελάχιστα εξαφανίζονται και ο agent φτάνει στον προορισμό του όπως δείχνουν οι δύο τελευταίες εικόνες.


Start
Clear
Singleagent
Multiagent
k Value:21
Obstacle Radius:0.10
Speed:Normal
Select a scenario
Scenario 1
Scenario 2
Scenario 3
Adjusts k value. If agent(s)
converge to local minima('stuck')
try increasing the value


Start
Clear
Singleagent
Multiagent
k Value:21
Obstacle Radius:0.10
Speed:Normal
Select a scenario
Scenario 1
Scenario 2
Scenario 3
Adjusts k value. If agent(s)
converge to local minima('stuck')
try increasing the value

ΠΑΡΑΡΤΗΜΑ Α – ΚΩΔΙΚΑΣ ΕΞΟΜΟΙΩΣΕΩΝ ΣΕ Matlab

main_traffic.m

```
if (~exist('batch_mode'))
    clear all
    close all
    clc

    nagents=input('How many agents? ');
    case_range=input('Select aircraft from case (Enter for all): ');
    if isempty(case_range)
        case_range=1:4;
    end

    select_first=input('Start agent selection from (Enter for 1): ');
    if isempty(select_first)
        select_first=1;
    end


    select_every=input('Round Robin selection every (Enter for 1): ');
    if isempty(select_every)
        select_every=1;
    end
end

f2nm=@(x)x./6076.12;
nm2f=@(x)x.*6076.12;
km2nm=@(x)x./1.852;


init_NF_tr


[agent_startxy,agent_finishxy,theta_start,time_startall,unom] =
TrafficData(nagents,case_range,select_first,select_every);

agents(:).start = zeros(4,1);
agents(:).target = zeros(4,1);
agents(:).class = zeros(4,1);
% agents(:).start = zeros(nagents,1);
% agents(:).target = zeros(nagents,1);
% agents(:).class = zeros(nagents,1);
aaa = zeros(4,nagents);
bbb = zeros(4,nagents);



for i = 1:nagents
    aaa(:,i) = [agent_startxy(i,1) agent_startxy(i,2) 0 theta_start(i,1)]';
    bbb(:,i) = [agent_finishxy(i,1) agent_finishxy(i,2) 0 theta_start(i,1)]';
    agents(i).start=aaa(:,i);
    agents(i).target=bbb(:,i);
    agents(i).time = fix(time_startall(i,1)/8);
    agents(i).unom = unom(i,1);
    agents(i).class=1;
end



navigate_tr
```

TrafficData.m

```
function [agent_start,agent_finish,theta_out,time_start,unom] = TrafficData(
numberofagents,case_range,select_first,select_every )

if ~exist('case_range')
    case_range=1:4;
end
```

```
if ~exist('select_first')
    select_first=1;
end

if ~exist('select_every')
    select_every=1;
end


load flights_all_upd.mat

count = 1;
count2 = select_first;
while (count<numberofagents+1)

    if count == 171
        dsc =1;
    end

    if (flights.fliesfl330(1,flights.totimesorted(2,count2))
==1)&(sum(flights.case(flights.totimesorted(2,count2))==case_range))
        time_start(count,1) = flights.totimesorted(1,count2);
        agent_start(count,1) = flights.start(1,flights.totimesorted(2,count2));
        agent_start(count,2) = flights.start(2,flights.totimesorted(2,count2));
        agent_finish(count,1) = flights.target(1,flights.totimesorted(2,count2));
        agent_finish(count,2) = flights.target(2,flights.totimesorted(2,count2));
        unom(count,1) = flights.speednm(flights.totimesorted(2,count2),22);
        count = count+1;
        count2=count2+select_every;
    else
        count2=count2+select_every;
    end
end



    start_finish = agent_finish - agent_start;

    tangent_stfin = abs(start_finish(:,2))./abs(start_finish(:,1));
    theta_out = atan(tangent_stfin);

    for i = 1:numberofagents
        if (start_finish(i,1)>0)
            if (start_finish(i,2)>0)
                theta_out(i,1) = theta_out(i,1);
            elseif (start_finish(i,2)<0)
                theta_out(i,1) = 2*pi - theta_out(i,1);
            end
        elseif (start_finish(i,1)<0)
            if (start_finish(i,2)>0)
                theta_out(i,1) = pi - theta_out(i,1);
            elseif (start_finish(i,2)<0)
                theta_out(i,1) = pi + theta_out(i,1);
            end
        end

    end
```

updateflights.m

```
km2nm=@(x)x./1.852;

x_all = flights.X;
y_all = flights.Y;

x_all = x_all/1000;
y_all = y_all/1000;

x_all = km2nm(x_all);
y_all = km2nm(y_all);  %x,y in nm

areaxy = interest_area/1000;
```

```
areaxy = km2nm(areaxy);

a = (y_all(2,:)-y_all(1,:))./((x_all(2,:)-x_all(1,:)));
b = y_all(1,:) - a.*x_all(1,:);


for i = 1:length(x_all)

    check = [x_all(1,i) y_all(1,i)];
    if length(find(check<200 & check>-200))==2
        flights.startsin(1,i) = 1;
    else
        flights.startsin(1,i) = 0;
    end

    check = [x_all(2,i) y_all(2,i)];
    if length(find(check<200 & check>-200))==2
        flights.endsin(1,i) = 1;
    else
        flights.endsin(1,i) = 0;
    end

end
count=0;
for i = 1:length(x_all)
    if i == 270
        continue
    end
    if (flights.startsin(1,i)==1) && (flights.endsin(1,i)==1)
        flights.start(:,i) = [x_all(1,i) y_all(1,i) 0 0]';
        flights.target(:,i) = [x_all(2,i) y_all(2,i) 0 0]';


    elseif (flights.startsin(1,i)==1) && (flights.endsin(1,i)==0)
        flights.start(:,i) = [x_all(1,i) y_all(1,i) 0 0]';
        y12 = a(1,i)*[200 -200] + b(1,i);
        x12 = ([200 -200] - b(1,i))/a(1,i);
        xy = [y12 x12];
        result1 = find(xy<=200 & xy>=-200);
        xy = [200 -200 x12;y12 200 -200];
        xyborder = xy(:,result1);
        vec1 = xyborder(:,1)' - [x_all(1,i) y_all(1,i)];
        vec2 = xyborder(:,2)' - [x_all(1,i) y_all(1,i)];
        vec3 = [x_all(2,i) y_all(2,i)] - [x_all(1,i) y_all(1,i)];
        if (vec1*vec3' >0)
            flights.target(:,i) = [xyborder(:,1); 0 ;0]';
        elseif (vec2*vec3' >0)
            flights.target(:,i) = [xyborder(:,2); 0 ;0]';
        end


    elseif (flights.startsin(1,i)==0) && (flights.endsin(1,i)==1)
        flights.target(:,i) = [x_all(2,i) y_all(2,i) 0 0]';
        y12 = a(1,i)*[200 -200] + b(1,i);
        x12 = ([200 -200] - b(1,i))/a(1,i);
        xy = [y12 x12];
        result1 = find(xy<=200 & xy>=-200);
        xy = [200 -200 x12;y12 200 -200];
        xyborder = xy(:,result1);
        vec1 = xyborder(:,1)' - [x_all(2,i) y_all(2,i)];
        vec2 = xyborder(:,2)' - [x_all(2,i) y_all(2,i)];
        vec3 = [x_all(1,i) y_all(1,i)] - [x_all(2,i) y_all(2,i)];
        if (vec1*vec3' >0)
            flights.start(:,i) = [xyborder(:,1); 0 ;0]';
        elseif (vec2*vec3' >0)
            flights.start(:,i) = [xyborder(:,2); 0 ;0]';
        end



    elseif (flights.startsin(1,i)==0) && (flights.endsin(1,i)==0)
        y12 = a(1,i)*[200 -200] + b(1,i);
        x12 = ([200 -200] - b(1,i))/a(1,i);
        xy = [y12 x12];
        result1 = find(xy<=200 & xy>=-200);
        xy = [200 -200 x12;y12 200 -200];
```

```
        xyborder = xy(:,result1);
        distance1 = norm(xyborder(:,1) - [x_all(1,i);y_all(1,i)],2);
        distance2 = norm(xyborder(:,2) - [x_all(1,i);y_all(1,i)],2);
        if distance1<distance2
            flights.start(:,i) = [xyborder(:,1) ;0 ;0];
            flights.target(:,i) = [xyborder(:,2) ;0;0];
        else
            flights.start(:,i) = [xyborder(:,2);0 ;0];
            flights.target(:,i) = [xyborder(:,1);0 ;0];
        end
    end



end

flights.speednm = flights.speed*3.6/1.852;

for j = 1:flights.total_number
    if flights.speednm(j,22) == 0
        flights.fliesfl330(1,j) = 0;
    else
        flights.fliesfl330(1,j) = 1;
    end
end

for j = 1:length(flights.timestamp)
    if flights.startsin(j) == 0
        flights.distdiff(1,j) = norm(flights.start(1:2,j) - [x_all(1,j)
y_all(1,j)]',2);
        timediff = (flights.distdiff(1,j)/flights.speednm(j,22))*3600;
        flights.totime(1,j) = fix(60*flights.timestamp(1,j) + timediff);
    else
        flights.totime(1,j) = 60*flights.timestamp(1,j);
    end
end

flights.totime(2,:) = 1:flights.total_number;

flights.totimesorted = sortrows(flights.totime');
flights.totimesorted = flights.totimesorted';
```

init_NF_tr.m

```
dt=1/3600;
kf=15e2;
epsilon=1e-12;
ef=1e-20;

mind=2;
ro2=50*mind;
b=5*mind;
ro=15;

NF_params.r=[5 5 f2nm(1000)]';

NF_params.Rs=[50 15 f2nm(1800)]';


NF_params.Rw=[10^3 10^3 10^3]';
NF_params.Rsw=NF_params.Rw-NF_params.Rs([1,1,3]);
NF_params.rbound=max(NF_params.Rw);
NF_params.rbound2=NF_params.rbound^2;

NF_params.k=10;
NF_params.dn=1e-8;
NF_params.X=0.15;
NF_params.Y=5e-8;
NF_params.fipol=[2*NF_params.Y/NF_params.X^3 -3*NF_params.Y/NF_params.X^2 0
NF_params.Y];
aclimb=deg2rad(15);
adescent=deg2rad(-20);
```

```
aignore=deg2rad(0);
atrans=deg2rad(0.5);
if aignore+atrans>(min(aclimb,-adescent))
    disp('Bad selection of angles!')
    return
end
```

navigate_tr.m

```
sat=inline('max(0,min(1,x))');
tic
i=1;
fd=0;

maxsteps=4000;
for j=1:nagents


    agents(j).history=zeros(4,maxsteps);
    agents(j).convec=zeros(3,maxsteps);
    agents(j).pose=agents(j).start;
    pose(:,j) = agents(j).start;
    agents(j).history(:,1)=agents(j).start;
    agents(j).Jit=[cos(agents(j).target(4)) sin(agents(j).target(4))]';
    agents(j).test=norm(agents(j).target(1:3)-agents(j).start(1:3),2)>mind;
    agents(j).fdc=agents(j).start(4);
    agents(j).Vc=1;    agents(j).cthreats={};
    if agents(j).class==0
        agents(j).ispan=[];
    else
        agents(j).ispan=find([agents.class]<=agents(j).class);
        agents(j).ispan(find(agents(j).ispan==j))=[];
    end
end
NF_Rs = NF_params.Rs;
NF_r = NF_params.r;

test=0;
for j=1:nagents
    test=test+agents(j).test;
    timeall(1,j) = agents(j).time;%%%%%%%%%%%%%
    agentsfinished(1,j) = 0;
end


fd=zeros(3,1);
g=fd;
elapsedtime = 0;
nagentsinflight = find((timeall <= elapsedtime) & (agentsfinished == 0));
w=0;

while (test || ~isempty(find(agentsfinished==0)))


    ii=i+1;

    nagentsinflight_old = nagentsinflight;

    nagentsinflight = find((timeall <= elapsedtime) & (agentsfinished == 0));

    takeoffqueue = setxor(nagentsinflight_old,nagentsinflight);

        for new_agents_i = takeoffqueue
            collides = 0;
            for agents_old_i = nagentsinflight_old
                r_dis = ((agents(new_agents_i).start(1,1)-
agents(agents_old_i).pose(1,1))/NF_params.r(1,1))^2 ...
                    + ((agents(new_agents_i).start(2,1)-
agents(agents_old_i).pose(2,1))/NF_params.r(2,1))^2;

                if r_dis <=10
                    nagentsinflight(:,find(nagentsinflight==new_agents_i)) = [];
                    agents(new_agents_i).time = agents(new_agents_i).time + 3600*dt;
                    collides = 1;
```

```
                    break;
                end
            end
            if collides == 0
                nagentsinflight_old = [nagentsinflight_old new_agents_i];
            end
        end


    totalagentsinflight(1,elapsedtime+1) = length(nagentsinflight);

    if ~isempty(nagentsinflight)==1
        test=0;
    end


    for j=nagentsinflight
        distance=norm(agents(j).pose(1:3)-agents(j).target(1:3),2); %
        if distance>mind
            Vold=agents(j).Vc;
            Jii=[cos(agents(j).pose(4)); sin(agents(j).pose(4))];
            [dvdq, agents(j).Vc,agents(j).cthreats{i-
agents(j).time}]=dV_tr(agents(j).pose(1:3),agents(j).pose(4),agents(j).target(1:3),age
nts(j).Jit,j,NF_params,agents,Jii,nagentsinflight,pose,NF_Rs,NF_r);

            nq=(Jii'*dvdq(1:2));

            fd=atan2(-dvdq(2),-dvdq(1));
            ad=-atan2(dvdq(3),norm(dvdq(1:2),2));
            atilde=min(aclimb,max(adescent,ad));
            td=tan(atilde);
            if i==1
                fdot=0;
            else
                fdot=(fd-agents(j).fdc)/dt;
            end


            while abs(fdot)>pi
                fdot=fdot-sign(fdot)*2*pi;
            end

            agents(j).fdc=fd;

            fd=agents(j).pose(4)-fd;
            while abs(fd)>pi
                fd=fd-sign(fd)*2*pi;
            end

            fprod=fdot*fd;

            if fprod-ef>0
                g=0;
            elseif fprod>0
                g=(-kf*fd+fdot)*(1-fprod/ef);
            else
                g=-kf*fd+fdot;
            end


            if (i-1-agents(j).time)==0
                dfdt=(agents(j).Vc-Vold)/dt;
            else
                dfdt=(agents(j).Vc-Vold)/dt-[nq dvdq(3)]*[agents(j).convec(1:2,i-1-
agents(j).time)];
            end
                Fi=agents(j).unom;

            if  dfdt<Fi*(abs(nq)-epsilon)
                uic=-sign(nq)*Fi;
            else
                uic=-sign(nq)*(dfdt+Fi*epsilon)/(abs(nq));
                [j,dfdt];
            end
            if uic>500
                uic = 500;
            elseif uic <-500
```

```
                uic = -500;
            end

            sn=sat((distance-ro)/b);
            sf=sat((abs(uic)*td*dvdq(3)+uic*nq+dfdt+epsilon)/(abs(uic)*td*dvdq(3)));
            sa=sat((aignore-abs(ad))/atrans);
            w=(1-sn*sf*sa)*abs(uic)*td;

        else
            uic=0;
            g=0;
            w=0;
            agents(j).cthreats{i-agents(j).time}=[];
            agentsfinished(1,j) = 1;
            disp(['Agents Finished: ', num2str(length(nonzeros(agentsfinished)))]);
        end

        agents(j).convec(:,i-agents(j).time)=[8*uic; w; g];

        agents(j).history(:,ii-
agents(j).time)=[agents(j).pose(1:2)+[Jii*agents(j).convec(1,i-agents(j).time)]*dt;
agents(j).pose(3)+agents(j).convec(2,i-agents(j).time)*dt;
agents(j).pose(4)+agents(j).convec(3,i-agents(j).time)*dt];

        while abs(agents(j).history(4,ii-agents(j).time))>pi
            agents(j).history(4,ii-agents(j).time)=agents(j).history(4,ii-
agents(j).time)-sign(agents(j).history(4,ii-agents(j).time))*2*pi;
        end

        agents(j).pose=agents(j).history(:,ii-agents(j).time);
        pose(:,j)=agents(j).history(:,ii-agents(j).time);


test=test+(norm((agents(j).target(1:3)-agents(j).pose(1:3)),2)>mind);
    end


    i=i+1;
    elapsedtime = elapsedtime + 3600*dt;

end

elapsedtime = elapsedtime + 3600*dt;
totald=0;
totalf=0;
for j=1:nagents
    agents(j).cthreats(1,length(agents(j).cthreats)+1) =
agents(j).cthreats(1,length(agents(j).cthreats));
    agents(j).history = agents(j).history(:,1:length(agents(j).cthreats));
    agents(j).datahistory = agents(j).history;
    agents(j).convec = agents(j).convec(:,1:length(agents(j).cthreats));
    agents(j).history = [];
%%%%%%%%%%%%%%%%%%%
end


time=toc


for i = 1:nagents

    maxdistance(i,1) = norm(agents(i).target(1:3,1) -agents(i).start(1:3,1),2);

end

NF_params.maxdistance = max(maxdistance);

if ~exist('batch_mode')
    save data.mat
end
```

dV_tr.m

```
function [dV, V2c,sensing] =
dV_tr(n,f,target,Jit,iagent,NF_params,agents,Jii,nagentsinflight,pose,NF_Rs,NF_r)
Jrel=[Jii,flipdim(Jii,1).*[-1; 1]]';
Jrel=[Jrel,[0;0];0 0 1];

n2x=n+[1 0 0]'*NF_params.dn;
n2y=n+[0 1 0]'*NF_params.dn;
n2z=n+[0 0 1]'*NF_params.dn;
ispan=agents(iagent).ispan;

if (length(nagentsinflight) > 0)
    count = 0;
    ispan_new=[];
    for kk = nagentsinflight
        if find(ispan==kk)>0
            count = count + 1;
            ispan_new(1,count) = kk;
        end
    end
    ispan = ispan_new;
else
    ispan = [];
end

 [V2c,sensing]=nf_tr(n,f,target,Jit,ispan,NF_params,agents,Jrel,pose,NF_Rs,NF_r);
V2x=nf_tr(n2x,f,target,Jit,ispan,NF_params,agents,Jrel,pose,NF_Rs,NF_r);
V2y=nf_tr(n2y,f,target,Jit,ispan,NF_params,agents,Jrel,pose,NF_Rs,NF_r);
V2z=nf_tr(n2z,f,target,Jit,ispan,NF_params,agents,Jrel,pose,NF_Rs,NF_r);

dV=([V2x V2y V2z]'-V2c)/NF_params.dn;
return
```

nf_tr.m

```
function [ V2c, sensing, Gi ] =
nf_tr(n,f,target,Jit,ispan,NF_params,agents,Jrel,pose,NF_Rs,NF_r)


sensing=[];


qi = norm(n,2);


deviation=(n-target);
gd=norm(deviation,2)^2/(4*NF_params.rbound2);
cof=8;
w=1e-2;
wp=1e0;
l=1;
G=1;

for threat=ispan
    qij=pose(1:3,threat)-n;


        qij=Jrel*qij; %
        if qij(1)>0

            r=sum((qij./NF_Rs).^2)-1;
        else
            r=sum((qij./NF_Rs([2,2,3])).^2)-1;
        end



        if r<0
            b=sum((qij./NF_r).^2)-1;
            if b>0
                sensing=[sensing; threat];
                gi=((b)/(b-1*r));
                gi=gi^3-3*gi^2+3*gi;
                G=G*gi;
            else
```

```
                V2c=1;
                Gi=0;
                return
            end
    end
end


rw=sum((n./NF_params.Rsw).^2)-1;
if rw>0
    bw=sum((n./NF_params.Rw).^2)-1;
    if bw<0
        bound=bw/(bw-1*rw);
        bound=bound^3-3*bound^2+3*bound;
    else
        V2c=1;
        bound=0;
        return
    end
else
    bound=1;
end
if G<=NF_params.X
    fi=polyval(NF_params.fipol,G);

    gd=gd+fi;
end
Gi=G*bound;
k=NF_params.k;
V2c=gd/((gd^k+Gi)^(1/k));!

if (max(isnan(V2c)))
    disp('V2c is NaN')
    keyboard
end
```

dataproc.m

```
runcpa = 0;


linedistance = zeros(nagents,1);
distancetravelled = zeros(nagents,1);

for i = 1:nagents
    chksize = size(agents(i).datahistory);
    chksize = chksize(1,2);
    if chksize < 5
        agents(i) = agents(i-1);
    end
    datahistorylength(1,i) = length(agents(i).datahistory);
end


for i = 1:nagents

    for ii = 1:(datahistorylength(1,i)-1)

        dr = agents(i).datahistory(1:3,ii+1)-agents(i).datahistory(1:3,ii);
        d_distancetravelled = norm(dr,2);
        distancetravelled(i,1) = distancetravelled(i,1) + d_distancetravelled;

    end

    linedistance(i,1) = norm(agents(i).datahistory(1:3,datahistorylength(1,i)) -
agents(i).start(1:3,1),2);
    flighttime(i,1) = length(agents(i).datahistory);
end


distance_percentage = (distancetravelled - linedistance)./linedistance;

agents(1).whosees = [];
agents(1).whoseespos = [];
```

```
for i = 1:nagents
    clc
    disp(['Progress:',num2str(i*100/(2*nagents)),'%']);

    threatslength = cellfun('length',agents(i).cthreats);
    minpos = min(find(threatslength~=0))-1;
    unique_threats_it = [];
    if max(threatslength)~=0
        unique_threats = [];
        for jj = 1:max(threatslength)
            unique_threats_it = cell2mat(agents(i).cthreats(1,find(threatslength ==
jj)));
            if jj == 1
                unique_threats = union(unique_threats,unique(unique_threats_it));
            else
                unique_threats = union(unique_threats,unique(unique_threats_it)');
            end
        end
        for kk = unique_threats
            pos = find(cellfun(@(x) any(x==kk),agents(i).cthreats)==1);
            agents(i).whosees(pos-minpos,kk) = kk;
            agents(i).whoseespos(pos-minpos,kk) = pos;
        end
    end

    if ~isempty(agents(i).whosees)
        [row1 column1] = size(agents(i).whosees);
        for ii = column1:-1:1
            if  isempty(nonzeros(agents(i).whosees(:,ii)))
                agents(i).whosees(:,ii) = [];
                agents(i).whoseespos(:,ii) = [];
                continue;
            end
        end
    end
end

for i = 1:nagents
    agents(i).timeinresolution = length(agents(i).whosees);

aircraftinconf = zeros(1,elapsedtime);

for i = 1:nagents
    currentconf = ~cellfun('isempty',agents(i).cthreats);
    aircraftinconf = aircraftinconf + [zeros(1,agents(i).time) currentconf
zeros(1,elapsedtime-length(currentconf)-agents(i).time)];
end

figure(10)
plot((1:8:8*elapsedtime)/3600,aircraftinconf)
title('Resolutions at a given time')
xlabel('Hours')
ylabel('Number of Aircraft')

axis tight

figure(1)
totalagentsinflight(1,length(totalagentsinflight)+1)=0;
plot((1:8:8*elapsedtime)/3600,totalagentsinflight)
title('Number of aircraft flying at the same time')
xlabel('Hours')
ylabel('Number of Aircraft')
axis tight


figure(2)
maxagentsmet=0;
for i = 1:nagents
    if min(size(agents(i).whosees))>maxagentsmet;
        maxagentsmet = min(size(agents(i).whosees));
    end
end
agentsmetall = zeros(1,maxagentsmet+1);
for i = 1:nagents
    agentsmetall(1,min(size(agents(i).whosees))+1) =
agentsmetall(1,min(size(agents(i).whosees))+1)+1;
```

```
end
bar(0:(length(agentsmetall)-1),agentsmetall)
title('Resolutions for all agents')
xlabel('Number of aircraft met')
ylabel('Sum of aircraft')

for i = 1:nagents
    agents(i).timeended = agents(i).time + length(agents(i).datahistory);
end


if runcpa==1
for i = 1:nagents
    clc
    disp(['Progress:',num2str((i+nagents)*100/(2*nagents)),'%']);
    count1 = 0;
    for j = 1:nagents
        if i == j
            agents(i).cpaallpoints(:,i) = 0;
            continue;
        end
        if ((agents(i).timeended <= agents(j).time) || agents(j).timeended <=
agents(i).time)
            count1 = count1 + 1;
            agents(i).cpaallpoints(:,j) = 0;
            agents(j).cpaallpoints(:,i) = 0;
            agents(i).didntflewwith(1,count1) = j;
            continue;
        end

        if agents(i).time<=agents(j).time
            if agents(i).timeended <=agents(j).timeended
                dummat_x = [];
                dummat_y = [];
                k = 1:(agents(i).timeended - agents(j).time);
                dummat_x(1,:) = agents(i).datahistory(1,k+agents(j).time-
agents(i).time) - agents(j).datahistory(1,k);
                dummat_y(1,:) = agents(i).datahistory(2,k+agents(j).time-
agents(i).time) - agents(j).datahistory(2,k);
                agents(i).cpaallpoints(k,j) = ((dummat_x.^2 + dummat_y.^2).^(0.5))';
            elseif agents(i).timeended > agents(j).timeended
                dummat_x = [];
                dummat_y = [];
                k = 1:(agents(j).timeended - agents(j).time);
                dummat_x(1,:) = agents(i).datahistory(1,k+agents(j).time-
agents(i).time) - agents(j).datahistory(1,k);
                dummat_y(1,:) = agents(i).datahistory(2,k+agents(j).time-
agents(i).time) - agents(j).datahistory(2,k);
                agents(i).cpaallpoints(k,j) = ((dummat_x.^2 + dummat_y.^2).^(0.5))';
            end
        elseif agents(i).time>agents(j).time
            if agents(i).timeended <=agents(j).timeended
                dummat_x = [];
                dummat_y = [];
                k = 1:(agents(i).timeended - agents(i).time);
                dummat_x(1,:) = agents(j).datahistory(1,k+agents(i).time-
agents(j).time) - agents(i).datahistory(1,k);
                dummat_y(1,:) = agents(j).datahistory(2,k+agents(i).time-
agents(j).time) - agents(i).datahistory(2,k);
                agents(i).cpaallpoints(k,j) = ((dummat_x.^2 + dummat_y.^2).^(0.5))';
            elseif agents(i).timeended > agents(j).timeended
                dummat_x = [];
                dummat_y = [];
                k = 1:(agents(j).timeended - agents(i).time);
                dummat_x(1,:) = agents(j).datahistory(1,k+agents(i).time-
agents(j).time) - agents(i).datahistory(1,k);
                dummat_y(1,:) = agents(j).datahistory(2,k+agents(i).time-
agents(j).time) - agents(i).datahistory(2,k);
                agents(i).cpaallpoints(k,j) = ((dummat_x.^2 + dummat_y.^2).^(0.5))';
            end
        end

    end
    agents(i).flewwith = setxor(1:nagents,agents(i).didntflewwith);
    agents(i).flewwith = setxor(agents(i).flewwith,i);
```

```
    count22 = 0;
    if isempty(agents(i).flewwith) == 1
        agents(i).cpa(1,1) = -1;
        agents(i).fpa(1,1) = -1;
        agents(i).cpaallpoints = [];
        continue;
    end
    for jjj = agents(i).flewwith
        count22 = count22+1;
        [agents(i).cpa(1,count22) agents(i).cpapos(1,count22)] =
min(nonzeros(agents(i).cpaallpoints(:,jjj)));
        [agents(i).fpa(1,count22) agents(i).fpapos(1,count22)] =
max(nonzeros(agents(i).cpaallpoints(:,jjj)));
    end
    agents(i).cpaallpoints = [];

end

figure(3)
for i = 1:nagents
    if agents(i).cpa(1,1) == -1
        continue;
    end
    hold on
    [thispos1 thispos2] = find(agents(i).cpa==min(agents(i).cpa));

plot(agents(i).datahistory(1,agents(i).cpapos(1,thispos2)),agents(i).datahistory(2,age
nts(i).cpapos(1,thispos2)),'.')
end
title('CPA positions')
xlabel('x axis[nm]')
ylabel('y axis [nm]')
hold off

figure(4)
for i = 1:nagents
    allcpa(i,1) = min(agents(i).cpa);
end
cpadiag = zeros(1,6);
cpadiag(1,1) = max(size(find(allcpa>5&allcpa<10)));
cpadiag(1,2) = max(size(find(allcpa>10&allcpa<15)));
cpadiag(1,3) = max(size(find(allcpa>15&allcpa<20)));
cpadiag(1,4) = max(size(find(allcpa>20&allcpa<25)));
cpadiag(1,5) = max(size(find(allcpa>25&allcpa<30)));
cpadiag(1,6) = max(size(find(allcpa>30)));
bar(cpadiag)
title('CPA')
xlabel('cpa [nm]')
ylabel('Sum of aircraft with the same cpa')
Labels = {'5-10', '10-15', '15-20', '20-25','25-30','30+'};
set(gca, 'XTick', 1:6, 'XTickLabel', Labels);

end
```

makeresults.m

```
resultsfile = fopen('results.txt','w');
fprintf(resultsfile,'                          SIMULATION RESULTS \n\n');
fprintf(resultsfile,'Total number of agents: %d \n\n\n', nagents);

check12 = find(isnan(distance_percentage)==0);
newdis = distance_percentage(check12,1);
percentage = sum(newdis)/length(newdis);
fprintf(resultsfile,'Agents Travelled(mean value): %f %% more of their line route
distance\n', percentage*100);
fprintf(resultsfile,'Flight time: %f seconds or %f hours\n',
sum(flighttime),sum(flighttime/3600));
fprintf(resultsfile,'Flight time(Mean Value): %f seconds or %f hours\n',
sum(flighttime)/nagents,sum(flighttime/(3600*nagents)));
totaltinres = 0;
for jj = 1:nagents
    totaltinres = totaltinres + agents(jj).timeinresolution;
end
fprintf(resultsfile,'Total time in resolution(all agents): %f seconds or %f hours \n',
totaltinres, totaltinres/3600);
```

```
fprintf(resultsfile,'Mean time in resolution: %f seconds or %f hours \n',
totaltinres/nagents, totaltinres/(3600*nagents));
maxagentsmet=0;
for i = 1:nagents
    if min(size(agents(i).whosees))>maxagentsmet;
        maxagentsmet = min(size(agents(i).whosees));
    end
end
agentsmetall = zeros(1,maxagentsmet+1);
for i = 1:nagents
    agentsmetall(1,min(size(agents(i).whosees))+1) =
agentsmetall(1,min(size(agents(i).whosees))+1)+1;
end
meanres = ([0:(length(agentsmetall)-1)]*agentsmetall')/nagents;
fprintf(resultsfile,'Mean value of resolutions for each agent: %f\n', meanres);
fprintf(resultsfile,'Average %% time in resolution: %f %%\n',
100*totaltinres/sum(flighttime));
fprintf(resultsfile,'Execution time: %f seconds\n', time);
fprintf(resultsfile,'Flight time/execution time: %f\n\n\n', sum(flighttime)/time);

for i = 1:nagents
    fprintf(resultsfile,'                        AGENT: %d \n\n', i);
    fprintf(resultsfile,'Start: %f , %f , %f \n',
agents(i).start(1,1),agents(i).start(2,1),agents(i).start(3,1));
    fprintf(resultsfile,'Target: %f , %f , %f \n',
agents(i).target(1,1),agents(i).target(2,1),agents(i).target(3,1));
    fprintf(resultsfile,'Total distance travelled: %f [nm]\n',
distancetravelled(i,1));
    fprintf(resultsfile,'Line distance: %f [nm]\n', linedistance(i,1));

    if ((distancetravelled(i,1)-linedistance(i,1))<=0)
        fprintf(resultsfile,'Travelled: %f %% more of their line route distance\n',
0);
    else
        fprintf(resultsfile,'Travelled: %f %% more of their line route distance\n',
distance_percentage(i,1)*100);
    end

    fprintf(resultsfile,'Flight time: %f seconds or %f hours \n', flighttime(i,1),
flighttime(i,1)/3600);


    if agents(i).cpa == -1;
        fprintf(resultsfile,'Maximum distance was: No other agents where in
flight\n');
        fprintf(resultsfile,'Minimum distance was: No other agents where in
flight\n');
    else
        [maxagentdis maxagentpos] = max(agents(i).fpa);
        [minagentdis minagentpos] = min(agents(i).cpa);
        fprintf(resultsfile,'Maximum distance was %f [nm] with agent:
%d\n',maxagentdis,agents(i).flewwith(1,maxagentpos));
        fprintf(resultsfile,'Minimum distance was %f [nm] with agent:
%d\n',minagentdis,agents(i).flewwith(1,minagentpos));
    end

    fprintf(resultsfile,'Total time in resolution: %f seconds or %f hours \n',
agents(i).timeinresolution, agents(i).timeinresolution/3600);

    if ~isempty(agents(i).whosees)
        fprintf(resultsfile,'Agent: %d met %d agent(s)\n',
i,min(size(agents(i).whosees)));
        for j = 1:min(size(agents(i).whosees))
            fprintf(resultsfile,'Agent: %d at %f hours for %f hours\n',
max(agents(i).whosees(:,j)),(min(find(agents(i).whoseespos(:,j)>0))+agents(i).time)/36
00,length(nonzeros(agents(i).whosees(:,j)))/3600);
        end
    else
        fprintf(resultsfile,'Agent: %d met %d agent(s)\n', i,0);
    end

    fprintf(resultsfile,'\n\n\n');
end


fclose(resultsfile);
```

ΠΑΡΑΡΤΗΜΑ Β – ΚΩΔΙΚΑΣ εφαρμογής στην Java

```
package mainproject;

import java.awt.*;
import java.applet.*;
import java.awt.event.*;

public class Mainproject extends Applet implements Runnable, ActionListener,
MouseListener, ItemListener, AdjustmentListener
{
     Thread runner;
     int option = 1,speed_value=5;
     boolean running = true,obstacleselection = true,agentselection_multi =
true,agentsmouse=true;
     double k=17;
     //Single Agent Vars/////////////
     double n[] = new double[2];
     double start_single[] = new double[2];
     double target[] = new double[2];
     //Obstacles///////////////
     double obstacle_xpos[] = new double[1];
     double obstacle_ypos[] = new double[1];
     double obstacle_radius[] = new double[1];
     double getobsradius=0.1,getagentradius=0.05;
     //Multi Agent Vars/////////////
     double n_multi[] = new double[2];
     double target_multi[] = new double[2];
     double n_xpos[] = new double[1];
     double n_ypos[] = new double[1];
     double start_xpos_multi[] = new double[1];
     double start_ypos_multi[] = new double[1];
     double target_xpos[] = new double[1];
     double target_ypos[] = new double[1];
     double agent_radius[] = new double[1];
     ////////////////////////////////
     double mindistance = 0.01,convergence,maxconvergence=100;
     NFcalculation nfcalculation_o = new NFcalculation();
     //double buffering vars/////
     Graphics bufferGraphics;
     Image offscreen;
     //Gui components/////////////
     Button btn_start;
     Button btn_clear;
     Button btn_restart;
     CheckboxGroup optiongroup;
     Checkbox chk_Singleagent;
     Checkbox chk_Multiagent;
     Label lbl_k;
     Scrollbar sld_k;
     Label lbl_agentradius;
     Scrollbar sld_agentradius;
     Label lbl_obstacleradius;
     Scrollbar sld_obstacleradius;
     Label lbl_speed;
     Scrollbar sld_speed;
     Label lbl_scenario;
     Button btn_scenario1;
     Button btn_scenario2;
     Button btn_scenario3;
     Label lbl_info1;
     Label lbl_info2;
     Label lbl_info3;
     //Mouse Listener/////////////
     double mouse_xpos,mouse_ypos;



     public void init()
     {
         //Variables initialization
         n[0] = -0.9;
         n[1] = 0;
         target[0] = 0.75;
         target[1] = 0.1;
```

```
obstacle_radius[0] = 0.1;
agent_radius[0] = 0.05;
//////////////////////////
///double buffering////////
offscreen = createImage(600,600);
bufferGraphics = offscreen.getGraphics();
///Gui init/////////////////
setLayout(null);
btn_start = new Button("Start");
btn_clear = new Button("Clear");
optiongroup = new CheckboxGroup();
chk_Singleagent = new Checkbox("Singleagent", optiongroup,true);
chk_Multiagent = new Checkbox("Multiagent", optiongroup,false);
lbl_k = new Label("k Value:17",Label.CENTER);
sld_k = new Scrollbar(Scrollbar.HORIZONTAL, (int) k, 1, 2, 25);
lbl_agentradius = new Label("Agent Radius:0.05",Label.CENTER);
sld_agentradius = new Scrollbar(Scrollbar.HORIZONTAL, 0, 1, 0, 6);
lbl_obstacleradius = new Label("Obstacle Radius:0.1",Label.CENTER);
sld_obstacleradius = new Scrollbar(Scrollbar.HORIZONTAL, 0, 1, 0, 7);
lbl_speed = new Label("Speed:Normal",Label.CENTER);
sld_speed = new Scrollbar(Scrollbar.HORIZONTAL, 1, 1, 0, 3);
btn_restart = new Button("Restart");
lbl_scenario = new Label("Select a scenario",Label.CENTER);
btn_scenario1 = new Button("Scenario 1");
btn_scenario2 = new Button("Scenario 2");
btn_scenario3 = new Button("Scenario 3");
lbl_info1 = new Label("Place Obstacles",Label.CENTER);
lbl_info2 = new Label("",Label.CENTER);
lbl_info3 = new Label("",Label.CENTER);
//////////////////////////////////////////////////////////////
btn_start.setBounds(601, 0, 109, 40);
btn_clear.setBounds(601, 45, 109, 40);
btn_restart.setBounds(601, 90, 109, 40);
chk_Singleagent.setBounds(605, 150, 100, 20);
chk_Multiagent.setBounds(605, 170, 100, 20);
lbl_k.setBounds(603, 200, 107, 20);
sld_k.setBounds(603, 220, 107, 20);
lbl_agentradius.setBounds(603, 250, 107, 20);
sld_agentradius.setBounds(603, 270, 107, 20);
lbl_obstacleradius.setBounds(603, 250, 107, 20);
sld_obstacleradius.setBounds(603, 270, 107, 20);
lbl_speed.setBounds(603, 300, 107, 20);
sld_speed.setBounds(603, 320, 107, 20);
lbl_scenario.setBounds(603, 360, 107, 20);
btn_scenario1.setBounds(603, 390, 107, 30);
btn_scenario2.setBounds(603, 430, 107, 30);
btn_scenario3.setBounds(603, 470, 107, 30);
lbl_info1.setBounds(573, 530, 157, 15);
lbl_info2.setBounds(573, 550, 157, 15);
lbl_info3.setBounds(573, 570, 157, 15);
lbl_k.setFont(new Font("sansserif",Font.PLAIN,11));
lbl_agentradius.setFont(new Font("sansserif",Font.PLAIN,11));
lbl_obstacleradius.setFont(new Font("sansserif",Font.PLAIN,11));
lbl_speed.setFont(new Font("sansserif",Font.PLAIN,11));
lbl_scenario.setFont(new Font("sansserif",Font.PLAIN,11));
lbl_info1.setFont(new Font("sansserif",Font.PLAIN,11));
lbl_info2.setFont(new Font("sansserif",Font.PLAIN,11));
lbl_info3.setFont(new Font("sansserif",Font.PLAIN,11));
lbl_agentradius.setVisible(false);
sld_agentradius.setVisible(false);
btn_restart.setVisible(false);
//////////////////////////////////////////////////////////////
add(btn_start);
add(btn_clear);
add(chk_Singleagent);
add(chk_Multiagent);
add(sld_k);
add(lbl_k);
add(lbl_agentradius);
add(sld_agentradius);
add(lbl_obstacleradius);
add(sld_obstacleradius);
add(lbl_speed);
add(sld_speed);
add(btn_restart);
add(lbl_scenario);
add(btn_scenario1);
```

```
        add(btn_scenario2);
        add(btn_scenario3);
        add(lbl_info1);
        add(lbl_info2);
        add(lbl_info3);
        //////Action Listener///////
        btn_start.addActionListener(this);
        btn_clear.addActionListener(this);
        btn_restart.addActionListener(this);
        btn_scenario1.addActionListener(this);
        btn_scenario2.addActionListener(this);
        btn_scenario3.addActionListener(this);
        chk_Singleagent.addItemListener(this);
        chk_Multiagent.addItemListener(this);
        sld_k.addAdjustmentListener(this);
        sld_agentradius.addAdjustmentListener(this);
        sld_obstacleradius.addAdjustmentListener(this);
        sld_speed.addAdjustmentListener(this);
        ///////Mouse Listener///////
        addMouseListener(this);
    }

    public void start()
    {
        if (runner == null)
        {
            runner = new Thread(this);
            runner.start();
        }

    }

    public void stop()
    {
        if (runner != null)
        {
            runner = null;
        }

    }

    public void run()
    {

        while(true)//inf Loop
        {
            //Main Run Loop for Single and Multiagent Problem
             while(running)   //Inf while. Here we have new n calculation and then
paint
             {
                 if (option==1)
                 {
                     if (obstacleselection==true)
                     {
                         repaint();
                     }
                     else if (obstacleselection == false)
                     {
                         n = nfcalculation_o.newposition(n,
target,obstacle_xpos,obstacle_ypos,k,obstacle_radius,agent_radius);
                         convergence = Math.pow(n[0] - target[0], 2) + Math.pow(n[1]-
target[1], 2);
                         convergence = Math.pow(convergence, 0.5);
                         if (convergence<mindistance)
                         {
                             running = false;
                         }
                         repaint();
                         try {
                           Thread.sleep(speed_value);
                           }
                         catch (InterruptedException e) {
                           e.printStackTrace();
                           }
                     }
                 }
                 else if(option==2)
```

```
                {
                    if (agentselection_multi==true)
                    {
                        repaint();
                    }
                    else if (agentselection_multi == false)
                    {
                        maxconvergence = 0;
                        for(int i=0;i<n_xpos.length-1;i++)
                        {
                            n_multi[0] = n_xpos[i];
                            n_multi[1] = n_ypos[i];
                            target_multi[0] = target_xpos[i];
                            target_multi[1] = target_ypos[i];
                            convergence = Math.pow(n_multi[0] - target_xpos[i], 2) +
Math.pow(n_multi[1]-target_ypos[1], 2);
                            convergence = Math.pow(convergence, 0.5);
                            if (convergence<mindistance)
                            {
                                maxconvergence++;
                                if (maxconvergence==(n_xpos.length-1))
                                {
                                    running = false;
                                }
                                continue;
                            }
                         //   System.out.printf("%f\n", convergence);
                            n_multi = nfcalculation_o.newposition_multi(n_multi,
target_multi,n_xpos,n_ypos,i,k,agent_radius);
                            n_xpos[i] = n_multi[0];
                            n_ypos[i] = n_multi[1] ;

                        }
                        repaint();
                        try {
                          Thread.sleep(speed_value-1);
                          }
                        catch (InterruptedException e) {
                          e.printStackTrace();
                          }
                    }

                }
              }//end of running while
          }//end of inf while
     }

     public void paint(Graphics g)
     {
        //clears graphics object
         bufferGraphics.setColor(Color.white);
         bufferGraphics.clearRect(0,0,600,600);
         bufferGraphics.setColor(Color.black);
         bufferGraphics.setColor(Color.black );
         bufferGraphics.drawOval(0,0,600,600);
        //Main paint code for Single and Multiagent Problem
        if (option==1)
        {
            if (obstacleselection == true)
            {

                bufferGraphics.setColor(Color.RED);
                bufferGraphics.fillOval(nfcalculation_o.trans_x(target[0])-
2,nfcalculation_o.trans_y(target[1])-2,4,4);
                bufferGraphics.setColor(Color.black);
                bufferGraphics.fillOval(nfcalculation_o.trans_x(n[0])-
5,nfcalculation_o.trans_y(n[1])-5,10,10);
                for(int i=0;i<obstacle_xpos.length-1;i++)
                {
                    bufferGraphics.drawOval(nfcalculation_o.trans_x(obstacle_xpos[i])-
((int)(obstacle_radius[i]*300)),nfcalculation_o.trans_y(obstacle_ypos[i])-
((int)(obstacle_radius[i]*300)),((int)(obstacle_radius[i]*600)),((int)(obstacle_radius
[i]*600)));
                }
                g.drawImage(offscreen,0,0,this);
            }
            else if(obstacleselection == false)
```

```
            {
                for(int i=0;i<obstacle_xpos.length-1;i++)
                {
                    bufferGraphics.drawOval(nfcalculation_o.trans_x(obstacle_xpos[i])-
((int)(obstacle_radius[i]*300)),nfcalculation_o.trans_y(obstacle_ypos[i])-
((int)(obstacle_radius[i]*300)),((int)(obstacle_radius[i]*600)),((int)(obstacle_radius
[i]*600)));
                }
                bufferGraphics.setColor(Color.RED);
                bufferGraphics.fillOval(nfcalculation_o.trans_x(target[0])-
2,nfcalculation_o.trans_y(target[1])-2,4,4);
                bufferGraphics.setColor(Color.black);
                bufferGraphics.fillOval(nfcalculation_o.trans_x(n[0])-
5,nfcalculation_o.trans_y(n[1])-5,10,10);
                g.drawImage(offscreen,0,0,this);
            }
        }//end of option 1 if
        else if (option==2)
        {
            if (agentselection_multi==true)
            {
                bufferGraphics.drawString("Select Agents",0, 600);
                for(int i=0;i<n_xpos.length-1;i++)
                {
                    bufferGraphics.fillOval(nfcalculation_o.trans_x(n_xpos[i])-
((int)(agent_radius[i]*300)),nfcalculation_o.trans_y(n_ypos[i])-
((int)(agent_radius[i]*300)),((int)(agent_radius[i]*600)),((int)(agent_radius[i]*600))
);
                }
                for(int i=0;i<target_xpos.length-1;i++)
                {
                    bufferGraphics.setColor(Color.RED);
                    bufferGraphics.fillOval(nfcalculation_o.trans_x(target_xpos[i])-
2,nfcalculation_o.trans_y(target_ypos[i])-2,4,4);
                    bufferGraphics.setColor(Color.black);
                }
                g.drawImage(offscreen,0,0,this);
            }
            else if(agentselection_multi==false)
            {
                for(int i=0;i<n_xpos.length-1;i++)
                {
                    bufferGraphics.fillOval(nfcalculation_o.trans_x(n_xpos[i])-
((int)(agent_radius[i]*300)),nfcalculation_o.trans_y(n_ypos[i])-
((int)(agent_radius[i]*300)),((int)(agent_radius[i]*600)),((int)(agent_radius[i]*600))
);
                }
                for(int i=0;i<target_xpos.length-1;i++)
                {
                    bufferGraphics.setColor(Color.RED);
                    bufferGraphics.fillOval(nfcalculation_o.trans_x(target_xpos[i])-
2,nfcalculation_o.trans_y(target_ypos[i])-2,4,4);
                    bufferGraphics.setColor(Color.black);
                }
               // bufferGraphics.fillOval(nfcalculation_o.trans_x(n[0])-
5,nfcalculation_o.trans_y(n[1])-5,10,10);
                g.drawImage(offscreen,0,0,this);
            }
            ////////////////////////////////////
        //    System.out.printf("\n");
        //    for (int i=0;i<n_xpos.length-1;i++) System.out.printf("%f ",n_xpos[i]);
        //    System.out.printf("\n");
        //    for (int i=0;i<n_ypos.length-1;i++) System.out.printf("%f ",n_ypos[i]);
        //    System.out.printf("\n");
        //    for (int i=0;i<target_xpos.length-1;i++) System.out.printf("%f
",target_xpos[i]);
        //    System.out.printf("\n");
        //    for (int i=0;i<target_ypos.length-1;i++) System.out.printf("%f
",target_ypos[i]);
        //    System.out.printf("\n");
        //    for (int i=0;i<agent_radius.length-1;i++) System.out.printf("%f
",agent_radius[i]);
            ////////////////////////////////////
        }//end of option 2 if

    }
```

```
    public void update(Graphics g)
    {
         paint(g);
    }

//////////////////////MouseListener Methods//////////////////////////
    public void mouseClicked(MouseEvent me)
    {
        if (option==1)
        {
            mouse_xpos = nfcalculation_o.inv_trans_x(me.getX());
            mouse_ypos = nfcalculation_o.inv_trans_y(me.getY());
            obstacle_xpos[obstacle_xpos.length-1] = mouse_xpos;
            obstacle_ypos[obstacle_ypos.length-1] = mouse_ypos;
            obstacle_radius[obstacle_radius.length-1] = getobsradius;
            obstacle_xpos = nfcalculation_o.resize(obstacle_xpos);
            obstacle_ypos = nfcalculation_o.resize(obstacle_ypos);
            obstacle_radius = nfcalculation_o.resize(obstacle_radius);
            repaint();
        }
        else if (option==2)
        {
            if (agentsmouse == true)
            {
                mouse_xpos = nfcalculation_o.inv_trans_x(me.getX());
                mouse_ypos = nfcalculation_o.inv_trans_y(me.getY());
                n_xpos[n_xpos.length-1] = mouse_xpos;
                n_ypos[n_ypos.length-1] = mouse_ypos;
                start_xpos_multi[start_xpos_multi.length-1] = mouse_xpos;
                start_ypos_multi[start_ypos_multi.length-1] = mouse_ypos;
                agent_radius[agent_radius.length-1] = getagentradius;
                n_xpos = nfcalculation_o.resize(n_xpos);
                n_ypos = nfcalculation_o.resize(n_ypos);
                start_xpos_multi = nfcalculation_o.resize(start_xpos_multi);
                start_ypos_multi = nfcalculation_o.resize(start_ypos_multi);
                agent_radius = nfcalculation_o.resize(agent_radius);
            }
            else if (agentsmouse == false)
            {
                mouse_xpos = nfcalculation_o.inv_trans_x(me.getX());
                mouse_ypos = nfcalculation_o.inv_trans_y(me.getY());
                target_xpos[target_xpos.length-1] = mouse_xpos;
                target_ypos[target_ypos.length-1] = mouse_ypos;
                target_xpos = nfcalculation_o.resize(target_xpos);
                target_ypos = nfcalculation_o.resize(target_ypos);
            }
            repaint();
            agentsmouse = !agentsmouse;
        }
    }

    public void mousePressed(MouseEvent me)
    {

    }

    public void mouseReleased(MouseEvent me)
    {

    }

    public void mouseEntered(MouseEvent me)
    {

    }

    public void mouseExited(MouseEvent me)
    {

    }
///////////////Action Listener///////////////////////////////

    public void actionPerformed(ActionEvent evt)
    {
        if (evt.getSource() == btn_start)
        {
            if (option==1)
```

```
            {
                obstacleselection = false;
            }
            else if (option==2)
            {
                agentselection_multi=false;
            }
        }
        else if (evt.getSource() == btn_restart)
        {
            if (option==1)
            {

            }
            else if (option==2)
            {
                double value1;
                for (int i=0;n_xpos.length-1<i;i++)
                {
                    value1 = start_xpos_multi[i];
                    n_xpos[i] = value1;
                    value1 = start_ypos_multi[i];
                    n_ypos[i] = value1;
                }
                running = true;
            }
        }
        else if (evt.getSource() == btn_clear)
        {
            Clearall();
        }
        else if (evt.getSource() == btn_scenario1)
        {
            Clearall();
            if (option==1)
            {
                double obstacle_xpos_temp[] = new double[]{-0.543333,-0.306667,-
0.033333,0.143333,0.480000};
                double obstacle_ypos_temp[] = new double[]{-0.060000,0.260000,-
0.036667,0.226667,0.010000};
                for (int i=0;i<obstacle_xpos_temp.length;i++)
                {
                    obstacle_xpos[i] = obstacle_xpos_temp[i];
                    obstacle_ypos[i] = obstacle_ypos_temp[i];
                    obstacle_radius[i] = 0.1;
                    obstacle_xpos = nfcalculation_o.resize(obstacle_xpos);
                    obstacle_ypos = nfcalculation_o.resize(obstacle_ypos);
                    obstacle_radius = nfcalculation_o.resize(obstacle_radius);
                }
                k = 22;

            }
            else if (option == 2)
            {
                double n_xpos_temp[] = new double[]{-
0.383333,0.036667,0.413333,0.026667};
                double n_ypos_temp[] = new double[]{0.743333 ,0.770000, 0.680000 ,-
0.596667};
                double target_xpos_temp[] = new double[]{-0.266667 ,0.010000,
0.300000, 0.063333};
                double target_ypos_temp[] = new double[]{-0.780000, -0.773333 ,-
0.770000, 0.646667 };
                double agent_radius_temp[] = new double[]{0.100000 ,0.100000
,0.100000, 0.250000};
                for (int i=0;i<n_xpos_temp.length;i++)
                {
                    n_xpos[i] = n_xpos_temp[i];
                    n_ypos[i] = n_ypos_temp[i];
                    target_xpos[i] = target_xpos_temp[i];
                    target_ypos[i] = target_ypos_temp[i];
                    agent_radius[i] = agent_radius_temp[i];
                    n_xpos = nfcalculation_o.resize(n_xpos);
                    n_ypos = nfcalculation_o.resize(n_ypos);
                    target_xpos = nfcalculation_o.resize(target_xpos);
                    target_ypos = nfcalculation_o.resize(target_ypos);
                    agent_radius = nfcalculation_o.resize(agent_radius);
                }
```

```
                k=21;
            }
        }
        else if (evt.getSource() == btn_scenario2)
        {
            Clearall();
            if (option==1)
            {
                double obstacle_xpos_temp[] = new double[]{-
0.273333,0.093333,0.403333,0.343333,0.016667};
                double obstacle_ypos_temp[] = new
double[]{0.006667,0.543333,0.313333,-0.253333,-0.646667};
                double obstacle_radius_temp[] = new
double[]{0.400000,0.150000,0.150000,0.100000,0.200000};
                for (int i=0;i<obstacle_xpos_temp.length;i++)
                {
                    obstacle_xpos[i] = obstacle_xpos_temp[i];
                    obstacle_ypos[i] = obstacle_ypos_temp[i];
                    obstacle_radius[i] = obstacle_radius_temp[i];
                    obstacle_xpos = nfcalculation_o.resize(obstacle_xpos);
                    obstacle_ypos = nfcalculation_o.resize(obstacle_ypos);
                    obstacle_radius = nfcalculation_o.resize(obstacle_radius);
                }
                k=18;
            }
            else if (option == 2)
            {
                double n_xpos_temp[] = new double[]{-0.516667, -0.210000 ,0.033333
,0.280000 ,0.473333 ,-0.413333, -0.116667, 0.126667 ,0.383333 ,0.536667 };
                double n_ypos_temp[] = new double[]{-0.660000 ,-0.696667 ,-0.760000,
-0.743333 ,-0.686667 ,0.763333 ,0.760000 ,0.753333 ,0.726667 ,0.603333 };
                double target_xpos_temp[] = new double[]{-0.306667 ,0.193333 ,-
0.056667 ,0.360000 ,0.213333 ,-0.330000 ,-0.160000 ,0.210000 ,0.003333 ,0.530000 };
                double target_ypos_temp[] = new double[]{0.743333 ,0.733333 ,0.690000
,0.663333 ,0.576667 ,-0.723333 ,-0.896667 ,-0.683333 ,-0.693333 ,-0.656667 };
                double agent_radius_temp[] = new double[]{0.050000 ,0.050000,
0.050000 ,0.050000, 0.050000 ,0.050000 ,0.050000, 0.050000 ,0.050000 ,0.050000};
                for (int i=0;i<n_xpos_temp.length;i++)
                {
                    n_xpos[i] = n_xpos_temp[i];
                    n_ypos[i] = n_ypos_temp[i];
                    target_xpos[i] = target_xpos_temp[i];
                    target_ypos[i] = target_ypos_temp[i];
                    agent_radius[i] = agent_radius_temp[i];
                    n_xpos = nfcalculation_o.resize(n_xpos);
                    n_ypos = nfcalculation_o.resize(n_ypos);
                    target_xpos = nfcalculation_o.resize(target_xpos);
                    target_ypos = nfcalculation_o.resize(target_ypos);
                    agent_radius = nfcalculation_o.resize(agent_radius);
                }
                k=18;
            }
        }
        else if (evt.getSource() == btn_scenario3)
        {
            Clearall();
            if (option==1)
            {
                double obstacle_xpos_temp[] = new double[]{-0.230000,-
0.013333,0.333333,0.236667,0.496667 };
                double obstacle_ypos_temp[] = new double[]{0.066667,-0.390000,-
0.170000,0.580000,0.146667 };
                double obstacle_radius_temp[] = new
double[]{0.250000,0.150000,0.100000,0.250000,0.150000 };
                for (int i=0;i<obstacle_xpos_temp.length;i++)
                {
                    obstacle_xpos[i] = obstacle_xpos_temp[i];
                    obstacle_ypos[i] = obstacle_ypos_temp[i];
                    obstacle_radius[i] = obstacle_radius_temp[i];
                    obstacle_xpos = nfcalculation_o.resize(obstacle_xpos);
                    obstacle_ypos = nfcalculation_o.resize(obstacle_ypos);
                    obstacle_radius = nfcalculation_o.resize(obstacle_radius);
                }
                k = 19;
            }
            else if (option == 2)
            {
```

```
                double n_xpos_temp[] = new double[]{-0.463333 ,0.290000 ,0.476667,
0.140000,-0.743333};
                double n_ypos_temp[] = new double[]{-0.686667 ,-0.743333 ,-0.180000
,0.746667, 0.040000};
                double target_xpos_temp[] = new double[]{0.226667, 0.086667 ,-
0.656667 ,-0.306667, 0.670000 };
                double target_ypos_temp[] = new double[]{0.766667, 0.770000 ,-
0.156667 ,-0.773333, 0.223333};
                double agent_radius_temp[] = new double[]{0.050000, 0.050000,
0.250000, 0.100000 ,0.150000};
                for (int i=0;i<n_xpos_temp.length;i++)
                {
                    n_xpos[i] = n_xpos_temp[i];
                    n_ypos[i] = n_ypos_temp[i];
                    target_xpos[i] = target_xpos_temp[i];
                    target_ypos[i] = target_ypos_temp[i];
                    agent_radius[i] = agent_radius_temp[i];
                    n_xpos = nfcalculation_o.resize(n_xpos);
                    n_ypos = nfcalculation_o.resize(n_ypos);
                    target_xpos = nfcalculation_o.resize(target_xpos);
                    target_ypos = nfcalculation_o.resize(target_ypos);
                    agent_radius = nfcalculation_o.resize(agent_radius);
                }
                k=11;
            }
        }
    }

    public void itemStateChanged( ItemEvent event )
    {
        if (chk_Singleagent.getState())
        {
            option = 1;
            lbl_obstacleradius.setVisible(true);
            sld_obstacleradius.setVisible(true);
            lbl_agentradius.setVisible(false);
            sld_agentradius.setVisible(false);
            lbl_info1.setText("Place Obstacles");
            lbl_info2.setText("");
            lbl_info3.setText("");
            Clearall();
        }
        else if (chk_Multiagent.getState())
        {
            option = 2;
            lbl_obstacleradius.setVisible(false);
            sld_obstacleradius.setVisible(false);
            lbl_agentradius.setVisible(true);
            sld_agentradius.setVisible(true);
            lbl_info1.setText("Place agents and select");
            lbl_info2.setText("their targets");
            lbl_info3.setText("");
            Clearall();
        }
    }

    ////////Misc Methods/////
    public void Clearall()
    {
        if (option==1)
        {
            n[0] = -0.9;
            n[1] = 0;
            obstacle_xpos = nfcalculation_o.shorten(obstacle_xpos);
            obstacle_ypos = nfcalculation_o.shorten(obstacle_ypos);
            agent_radius = nfcalculation_o.shorten(agent_radius);
            obstacle_radius = nfcalculation_o.shorten(obstacle_radius);
            obstacleselection=true;
            running = true;
        }
        else if (option==2)
        {
            n_xpos = nfcalculation_o.shorten(n_xpos);
            n_ypos = nfcalculation_o.shorten(n_ypos);
            target_xpos = nfcalculation_o.shorten(target_xpos);
            target_ypos = nfcalculation_o.shorten(target_ypos);
            agent_radius = nfcalculation_o.shorten(agent_radius);
```

```
            agentsmouse = true;
            agentselection_multi=true;
            running = true;
         }
     }

    public void adjustmentValueChanged(AdjustmentEvent e) {
        int radvalue,obstvalue,kvalue;

        kvalue =  sld_k.getValue();
        radvalue = sld_agentradius.getValue();
        obstvalue = sld_obstacleradius.getValue();
        if ((sld_obstacleradius.getValueIsAdjusting()) || (getobsradius != (double)
0.1+(obstvalue*0.05)))
        {
            lbl_info1.setText("Selects obstacle's radius");
            lbl_info2.setText("which is given as a percentage");
            lbl_info3.setText("of the workspace radius");
        }
        else if ((sld_agentradius.getValueIsAdjusting()) || (getagentradius !=
(double) 0.05+(radvalue*0.05)))
        {
            lbl_info1.setText("Selects agent's radius");
            lbl_info2.setText("which is given as a percentage");
            lbl_info3.setText("of the workspace radius");
        }
        else if ((sld_k.getValueIsAdjusting()) || (k != (double) kvalue))
        {
            lbl_info1.setText("Adjusts k value. If agent(s)");
            lbl_info2.setText("converge to local minima('stuck')");
            lbl_info3.setText("try increasing the value");
        }
        else
        {
            lbl_info1.setText("Adjusts the speed of");
            lbl_info2.setText("the simulation");
            lbl_info3.setText("");
        }
        k = (double) kvalue;
        getagentradius = (double) 0.05+(radvalue*0.05);
        getobsradius = (double) 0.1+(obstvalue*0.05);
        lbl_k.setText("k Value:" + kvalue);
        if (radvalue ==0) lbl_agentradius.setText("Agent Radius:0.05" );
        else lbl_agentradius.setText("Agent Radius:0." + (5+radvalue*5));
        lbl_obstacleradius.setText("Obstacle Radius:0." + (10+obstvalue*5));
        if (sld_speed.getValue()==0)
        {
            lbl_speed.setText("Speed:Slow");
            speed_value = 10;
        }
        else if (sld_speed.getValue()==1)
        {
            lbl_speed.setText("Speed:Normal");
            speed_value = 5;
        }
        else if (sld_speed.getValue()==2)
        {
            lbl_speed.setText("Speed:Fast");
            speed_value = 3;
        }

    }

}//end of applet class


package mainproject;




public class NFcalculation

{
```

```
    //////////////////Main Navigation is Here//////////////////////////

    //////////////////Single Agent/////////////////////////////////////

    public double[] newposition(double n[],double target[],double
obstacle_xpos[],double obstacle_ypos[],double k,double obstacle_radius[],double
agent_radius[])

    {

        double newpos[] = new double[2];

        double n_in[] = new double[2];

        double dfidx,dfidy,fi,fi_new,dfnorm;

        double dn = 0.001;



        fi =
calculatenf(n,target,obstacle_xpos,obstacle_ypos,k,obstacle_radius,agent_radius);

        n_in = n;

        n_in[0] += dn;

        fi_new =
calculatenf(n_in,target,obstacle_xpos,obstacle_ypos,k,obstacle_radius,agent_radius);

        dfidx = (fi_new - fi)/dn;

        n_in[0] -= dn;

        n_in[1] += dn;

        fi_new =
calculatenf(n_in,target,obstacle_xpos,obstacle_ypos,k,obstacle_radius,agent_radius);

        dfidy = (fi_new - fi)/dn;

        n_in[1] -= dn;

        dfnorm = Math.pow(dfidx, 2)+Math.pow(dfidy, 2);

        dfnorm = Math.pow(dfnorm, 0.5);

        newpos[0] = n[0] - 0.002*dfidx/dfnorm;

        newpos[1] = n[1] - 0.002*dfidy/dfnorm;



        return newpos;

    }



    public double calculatenf(double n[],double target[],double obstacle_xpos[],double
obstacle_ypos[],double k,double obstacle_radius[],double agent_radius[])

    {

        double fi,gk,den;

        double b = 1;
```

```
        double onedivk = 1/k;


        gk = Math.pow(n[0] - target[0], 2) + Math.pow(n[1] - target[1], 2);

        for(int i=0;i<obstacle_xpos.length-1;i++)

        {

            b = b*(Math.pow(n[0] - obstacle_xpos[i], 2) + Math.pow(n[1] -
obstacle_ypos[i], 2) - Math.pow(obstacle_radius[i]+0.02,2));

        }

        b = b*(Math.pow(1-0.02, 2) -Math.pow(n[0], 2) - Math.pow(n[1], 2));

        den = Math.pow(gk, k) + b;

        den = Math.pow(den, onedivk);

        fi = gk/den;




        return fi;


    }

    //////////////Multi Agent////////////////////////////////////

    public double[] newposition_multi(double n[],double target[],double
n_xpos[],double n_ypos[],int agent_num,double k,double agent_radius[])

    {

        double newpos[] = new double[2];

        double n_in[] = new double[2];

        double dfidx,dfidy,fi,fi_new,dfnorm;

        double dn = 0.001;


        fi = calculatenf_multi(n,target,n_xpos,n_ypos,agent_num,k,agent_radius);

        n_in = n;

        n_in[0] += dn;

        fi_new =
calculatenf_multi(n_in,target,n_xpos,n_ypos,agent_num,k,agent_radius);

        dfidx = (fi_new - fi)/dn;

        n_in[0] -= dn;

        n_in[1] += dn;

        fi_new =
calculatenf_multi(n_in,target,n_xpos,n_ypos,agent_num,k,agent_radius);
```

```
        dfidy = (fi_new - fi)/dn;

        n_in[1] -= dn;

        dfnorm = Math.pow(dfidx, 2)+Math.pow(dfidy, 2);

        dfnorm = Math.pow(dfnorm, 0.5);

        newpos[0] = n[0] - 0.0010*dfidx/dfnorm;

        newpos[1] = n[1] - 0.0010*dfidy/dfnorm;


        return newpos;

    }


    public double calculatenf_multi(double n[],double target[],double n_xpos[],double
n_ypos[],int agent_num,double k,double agent_radius[])

    {

        double fi,gk,den;

        double b = 1;

        double onedivk = 1/k;


        gk = Math.pow(n[0] - target[0], 2) + Math.pow(n[1] - target[1], 2);

        for(int i=0;i<n_xpos.length-1;i++)

        {

            if (i==agent_num)

            {

                continue;

            }

            b = b*(Math.pow(n[0] - n_xpos[i], 2) + Math.pow(n[1] - n_ypos[i], 2) -
Math.pow(agent_radius[agent_num]+agent_radius[i], 2));

        }

        b = b*(Math.pow(1-agent_radius[agent_num], 2) -Math.pow(n[0], 2) -
Math.pow(n[1], 2));

        den = Math.pow(gk, k) + b;

        den = Math.pow(den, onedivk);

        fi = gk/den;




        return fi;
```

```
}

//////////////Misc///////////////

public int trans_x(double input)
{
    int coordinates;

    coordinates = (int) Math.rint(input*300+300);

    return coordinates;

}

public int trans_y(double input)
{
    int coordinates;

    coordinates = (int) Math.rint(-input*300+300);

    return coordinates;

}

public double inv_trans_x(double input)
{
    double coordinates;

    coordinates = input/300 - 1;

    return coordinates;

}
```

```
    public double inv_trans_y(double input)
    {
        double coordinates;

        coordinates = 1-input/300;

        return coordinates;

    }

    public double[] resize(double array[])
    {
        double newarray[] = new double[array.length+1];

        for(int i=0;i<array.length;i++)
        {
            newarray[i] = array[i];
        }

        return newarray;
    }

     public double[] shorten(double array[])
    {
        double newarray[] = new double[1];



        return newarray;
    }

}
```

ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] D.E.KODITSCHEK AND E.RIMON, “Robot navigation functions on manifolds with boundary. Advances in applied mathematics 11(4),412-442(1990)

[2]E.Rimon and D.E. Koditschek, “Exact robot navigation using artificial potential functions,” IEEE Transactions on Robotics and Automation, vol. 8, no. 5,pp.501-508,1992.

[3]D.V.Dimarogonas and K.J.Kyriakopoulos, “Decentralized navigation functions for multiple robotic agents with limited sensing capabilities,”,Journal of Intelligent and Robotic Systems, vol.48,no.3,pp.411-433,2007.

[4]Lionis,G.,Papageorgiou,X.,Kyriakopoulos,K.J.: Locally computable navigation functions for sphere worlds. Proceedings of the 2007 IEEE International Conference on Robotics and Automation pp. 1998-2003 (2007)

[5]G.P.Roussos and K.J.Kyriakopoulos “Towards constant velocity navigation and collision avoidance for autonomous nonholonomic aircraft-like vehicles,” Conference on Decision and Control, 2009

[6] D.E.KODITSCHEK AND E.RIMON, “Exact Robot Navigation Using Cost Functions: The Case of Distinct Spherical Boundaries in E^n,” Technical Report 8803, Center for Systems Science, Yale University, January 1988.

[7] S. SMALE, On gradient dynamical systems, Ann. Of Math. 74, No. 1 (1961),199-206.

[8] J.MILNOR, “Lectures on the h-Cobordism Theorem”, Princeton Univ.Press,Princeton,NJm1965.

[9] W.S.MASSEY, personal communications with the authors, Sept. 1986

[10] Roussos, G., Kyriakopoulos, K.J.:Decentralized & prioritized Navigation and Collision Avoidance for Multiple Mobile Robots

[11] Roussos, G., Kyriakopoulos, K.J.: Decentralized navigation and collision avoidance for aircraft in 3D space. 2010 American Control Conference, Baltimore, USA (2010)

[12] Dimarogonas, D.V., Kyriakopoulos, K.J.: Decentralized navigation functions for multiple robotic agents with limited sensing capabilities. Journal of Intelligent and Robotic Systems 48(3),411,433 (2007)

[13] G.P.Roussos and K.J.Kyriakopoulos,”Towards constant velocity navigation and collision avoidance for autonomous nonholonomic aircraft-like vehicles,” Conference on Decision and Control, 2009

[14]Steven M. LaValle ,PLANNING ALGORITHMS, Cambridge University Press 2006