

ΕΘΝΙΚΟ ΚΑΙ ΚΑΠΟΔΙΣΤΡΙΑΚΟ

ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΙΝΟΙΝΩΝΙΩΝ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

ΠΡΟΒΛΕΨΗ ΠΛΑΙΣΙΟΥ, ΕΦΑΡΜΟΓΗ ΣΤΟ ΧΩΡΙΚΟ ΠΛΑΙΣΙΟ

Παναγιώτης Μπουγιούρης

Α.Μ. 1016

Υπεύθυνος Καθηγητής: Ευστάθιος Χατζηευθυμιάδης

Επιβλέπων: Χρήστος Αναγνωστόπουλος

ΑΘΗΝΑ, ΝΟΕΜΒΡΙΟΣ 2004

ΠΕΡΙΕΧΟΜΕΝΑ

---

1. ΕΙΣΑΓΩΓΙΚΑ

1.1 ΠΡΟΛΟΓΟΣ

1.2 ΠΑΡΟΥΣΑ ΜΕΛΕΤΗ

1.3 ΠΡΟΕΚΤΑΣΕΙΣ

2. ΥΠΗΡΕΣΙΕΣ ΒΑΣΙΣΜΕΝΕΣ ΣΤΗΝ ΤΟΠΟΘΕΣΙΑ

2.1 ΠΕΡΙΓΡΑΦΗ

2.2 ΟΙ ΣΧΕΤΙΚΕΣ ΜΕ ΤΗ ΘΕΣΗ ΥΠΗΡΕΣΙΕΣ

2.3 ΚΑΤΗΓΟΡΙΕΣ ΣΧΕΤΙΚΩΝ ΜΕ ΤΗ ΘΕΣΗ ΥΠΗΡΕΣΙΩΝ

2.4 Ο ΠΡΟΣΔΙΟΡΙΣΜΟΣ ΤΗΣ ΘΕΣΗΣ

2.5 ΟΙ ΥΠΗΡΕΣΙΕΣ ΔΙΚΤΥΟΥ

2.6 ΥΠΗΡΕΣΙΟΣΤΡΕΦΗΣ ΑΡΧΙΤΕΚΤΟΝΙΚΗ

2.7 ΓΕΩΓΡΑΦΙΚΗ ΤΟΠΟΘΕΣΙΑ – ΧΩΡΙΚΟ ΠΛΑΙΣΙΟ - ΠΡΟΒΛΕΨΗ

2.8 ΠΡΟΒΛΕΨΗ ΣΕ ΥΒΡΙΔΙΚΑ ΚΑΙ AD HOC ΑΣΥΡΜΑΤΑ ΔΙΚΤΥΑ

3. ΚΙΝΗΤΙΚΟΤΗΤΑ - ΜΟΝΤΕΛΑ ΠΡΟΣΟΜΟΙΩΣΗΣ ΚΙΝΗΤΙΚΟΤΗΤΑΣ

3.1 ΕΙΣΑΓΩΓΗ

3.2 ΜΟΝΤΕΛΑ ΤΥΧΑΙΑΣ ΚΙΝΗΤΙΚΟΤΗΤΑΣ (MARKOV, BROWN, ΑΥΞΗΤΙΚΟ, SMOOTH, GAUSS-MARKOV, ΜΕΤΑΤΟΠΙΣΗΣ ΣΗΜΕΙΟΥ, ΔΙΑΝΥΣΜΑΤΙΚΟ, ΟΜΑΔΑΣ ΣΗΜΕΙΟΥ ΑΝΑΦΟΡΑΣ, BOIDS)

3.2 ΑΙΤΙΟΚΡΑΤΙΚΑ ΜΟΝΤΕΛΑ ΚΙΝΗΤΙΚΟΤΗΤΑΣ

3.3 ΥΒΡΙΔΙΚΑ ΜΟΝΤΕΛΑ ΚΙΝΗΤΙΚΟΤΗΤΑΣ

4. ΠΛΑΙΣΙΟ ΚΑΙ ΕΠΙΓΝΩΣΗ ΠΛΑΙΣΙΟΥ

4.1 ΕΙΣΑΓΩΓΗ

4.2 ΕΠΙΓΝΩΣΗ ΠΛΑΙΣΙΟΥ

4.3 ΣΤΗΝ ΕΦΑΡΜΟΓΗ

4.4 ΤΟ ΠΛΑΙΣΙΟ ΣΤΙΣ ΚΙΝΗΤΕΣ ΣΥΣΚΕΥΕΣ

5. ΠΡΟΒΛΕΨΗ ΠΛΑΙΣΙΟΥ

5.1 ΠΡΟΒΛΕΨΗ ΠΛΑΙΣΙΟΥ

5.2 ΠΡΟΒΛΕΨΗ ΜΕΣΩ ΚΑΤΑΣΤΑΣΕΩΝ

5.3 ΠΡΟΒΛΕΨΗ ΜΙΑΣ-ΚΑΤΑΣΤΑΣΗΣ

5.3 ΠΡΟΒΛΕΨΗ ΔΥΟ-ΚΑΤΑΣΤΑΣΕΩΝ

5.4 ΚΑΘΟΛΙΚΗ ΠΡΟΒΛΕΨΗ ΔΥΟ-ΕΠΙΠΕΔΩΝ

5.4 ΤΟΠΙΚΗ ΠΡΟΒΛΕΨΗ ΔΥΟ-ΕΠΙΠΕΔΩΝ

6. ΕΦΑΡΜΟΓΗ ΣΤΟ ΧΩΡΙΚΟ ΠΛΑΙΣΙΟ

6.1 ΕΙΣΑΓΩΓΗ

6.2 ΜΕΘΟΔΟΙ ΠΑΡΕΜΒΟΛΗΣ

6.3 ΠΑΡΕΜΒΟΛΗ NEWTON ΔΙΗΡΗΜΕΝΩΝ ΔΙΑΦΟΡΩΝ

6.4 ΠΑΡΕΜΒΟΛΗ LAGRANGE

6.5 ΚΥΒΙΚΕΣ SPLINES

6.6 ΧΩΡΙΚΟ ΠΛΑΙΣΙΟ ΤΗΣ ΠΡΟΣΟΜΟΙΩΣΗΣ

6.7 Η ΠΡΟΣΟΜΟΙΩΣΗ

6.8 ΣΕ ΒΑΘΟΣ ΧΡΟΝΟΥ

7. ΕΠΙΛΟΓΟΣ

ΒΙΒΛΙΟΓΡΑΦΙΑ

ΠΑΡΑΡΤΗΜΑΤΑ

Α. ΔΙΑΓΡΑΜΜΑΤΑ UML

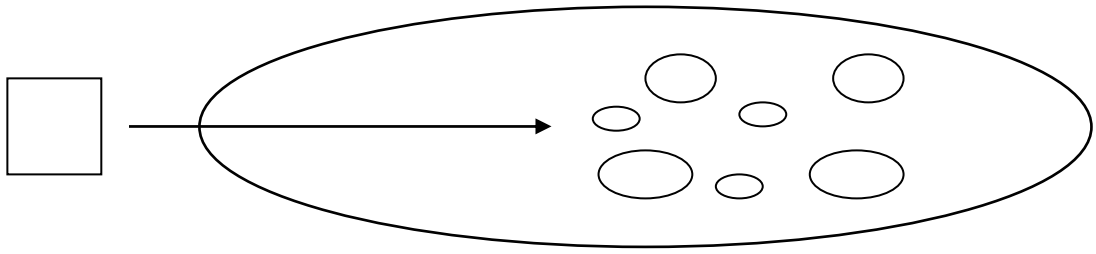
Β. ΠΗΓΑΙΟΣ ΚΩΔΙΚΑΣ

«Δεν ελπίζω τίποτα, δε φοβάμαι τίποτα, είμαι ελεύθερος»

Ν. Καζαντζάκης

ΕΝΟΤΗΤΑ 1η

«ΕΙΣΑΓΩΓΙΚΑ»



1.1 ΠΡΟΛΟΓΟΣ

Δε χρειάζεται να μακρηγορήσει κανείς ιδιαίτερα για να καταδείξει πόσο οι σύγχρονες τηλεπικοινωνίες έχουν επηρεάσει τη ζωή μας την τελευταία δεκαετία. Μάλιστα, πρόκειται για τεχνολογίες οι οποίες δεν έμειναν για πολύ καιρό στο στάδιο της έρευνας, είτε σαν ακαδημαϊκά προγράμματα είτε σαν στρατιωτικά σχέδια – από τα οποία πολλές ξεκίνησαν. Γρήγορα βρήκαν το δρόμο τους προς την αγορά μέσω των υπαρχόντων κρατικών αλλά και νέων τηλεπικοινωνιακών οργανισμών, για πολλούς από τους οποίους σήμερα γίνεται λόγος ως «κολοσσοί» της παγκόσμιας οικονομίας.

Τελικός παραλήπτης αυτών των νέων εκπληκτικών δυνατοτήτων είναι ο άνθρωπος και θα ήταν αντικείμενο μιας άλλης μελέτης, περισσότερο κοινωνικοοικονομικής, το πώς αυτό έγινε εφικτό τελικά. Το μόνο βέβαιο είναι πως η εξέλιξη εμφανίζεται ραγδαία και η χρήση των προϊόντων των τηλεπικοινωνιών συνδέεται άρρηκτα με τη σύγχρονη ζωή.

Μια πολύ πρόχειρη ματιά βρίσκει τον άνθρωπο των ανεπτυγμένων χωρών στην αλλαγή της χιλιετίας να χρησιμοποιεί μια πλειάδα από αυτές: κινητά τηλέφωνα (2ης και σιγά σιγά 3ης γενιάς) με όλο και περισσότερες δυνατότητες, κινητό διαδίκτυο (internet), υπηρεσίες βασισμένες στην τοποθεσία του χρήστη (location based services) και άλλες πολλές. Υπάρχει, δηλαδή, η απαραίτητη υποδομή στις μέρες μας για την παροχή χρήσιμων και ποιοτικών υπηρεσιών στον άνθρωπο, τον οποίο οφείλουν να έχουν στο επίκεντρο όλοι όσοι καταπιάνονται με τα τηλεπικοινωνιακά ζητήματα, ακόμη και στο ερευνητικό επίπεδο.

1.2 ΠΑΡΟΥΣΑ ΜΕΛΕΤΗ

Στα πλαίσια των υπηρεσιών βασισμένων στη γεωγραφική τοποθεσία του χρήστη η σύγχρονη μελέτη βρίσκεται σε προχωρημένο στάδιο και ήδη υπόσχεται ευρείας εμπορικής εφαρμογής, χωρίς, βέβαια, να έχει εξαντληθεί ακόμη. Στην παρούσα εργασία θεωρήθηκε χρήσιμο να

ξεκινήσουμε κάνοντας μια αναφορά σε αυτές και τη θέση τους στη σύγχρονη επιστήμη και επιχειρηματικότητα, μιας και αποτελούν κατεξοχήν συστήματα όπου το «πλαίσιο» (context) μπορεί να παίξει σημαντικό ρόλο και κατ' επέκταση το ίδιο και η πρόβλεψή του. Άλλωστε το «χωρικό πλαίσιο» (spatial context) αποτελεί και βασικό πεδίο μελέτης και εφαρμογής της παρούσας εργασίας.

Στη συνέχεια μελετούμε την έννοια της «επίγνωσης πλαισίου» (context awareness), ενός τεράστιου νέου κεφαλαίου που έχει ανοιχτεί στον τομέα της ανάλυσης και μοντελοποίησης συστημάτων που εξαρτώνται από πολλές παραμέτρους – με σημαντική εφαρμογή στις κινητές υπηρεσίες -, τη θέση του στη σύγχρονη έρευνα και τη χρησιμότητά του. Σκοπός μας είναι να προεκτείνουμε την έρευνα σε δυνατότητες πρόβλεψης του «πλαισίου» (context prediction).

Για λόγους πληρότητας της μελέτης μας παρεμβάλλουμε αναφορές σε υπάρχοντα μοντέλα πρόβλεψης (prediction models), καθώς και σε μοντέλα προσομοίωσης της κινητικότητας χρηστών (mobility), έτσι ώστε να μπορούμε να έχουμε μια πληρέστερη εικόνα και να εξάγουμε εμπεριστατωμένα συμπεράσματα.

Σε αυτή ακριβώς την κατεύθυνση βρίσκεται το δεύτερο – και κυριότερο - τμήμα της παρούσας εργασίας. Προσεγγίζουμε το «πλαίσιο» (context) στη χωρική (spatial) του έκφανση και προσπαθούμε να μελετήσουμε τρόπους με τους οποίους να επιδέχεται πρόβλεψη. Ο λόγος για αυτή την επιλογή έγκειται στο ότι το «χωρικό πλαίσιο» (spatial context) είναι από τα κατεξοχήν πεδία όπου η ανάγκη της πρόβλεψης είναι εξαιρετικά σημαντική αλλά και προσφέρεται ιδιαιτέρως για τη μελέτη της «πρόβλεψης του πλαισίου». Κύριος χώρος στον οποίο εφαρμόζεται η μελέτη μας είναι το περιβάλλον της γης, όπως αυτό μοντελοποιείται από το σύστημα GPS (Global Positioning System) αλλά όχι μόνο. Πηγή του χώρου μπορεί να είναι οποιοδήποτε σύστημα που παράγει χωρικές

συντεταγμένες με αποδεκτό τρόπο, όπως για παράδειγμα ο City Simulator της IBM.

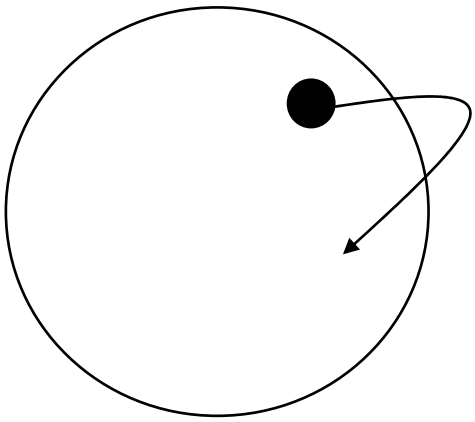
Μελετούμε τη συμπεριφορά μαθηματικών μεθόδων παρεμβολής (interpolation) στην πρόβλεψη χωρικών συντεταγμένων για να εκτιμήσουμε κατά πόσο ενδείκνυνται να χρησιμοποιηθούν στην πρόβλεψη της ίδιας της κινητικότητας, αρχικά, κι έπειτα στην πρόβλεψη του «πλαισίου». Οι μέθοδοι αυτές θα αναλυθούν και θα εφαρμοστούν σε ένα σύστημα προσομοίωσης με τεχνολογία Java 2 ώστε να εξαχθούν τα απαραίτητα συμπεράσματα.

1.3 ΠΡΟΕΚΤΑΣΕΙΣ

Σαν κατακλείδα της παρούσας εργασίας γίνεται μια σύντομη αναφορά στις προεκτάσεις που μπορεί να έχει μια τέτοια μελέτη, πώς μπορεί να μελετηθεί περαιτέρω και πώς θα ήταν δυνατόν να βελτιωθεί και να καταστεί εφαρμόσιμη, με γνώμονα πάντοτε τις αναγκαιότητες του πεδίου της «επίγνωσης πλαισίου».

ΕΝΟΤΗΤΑ 2η

«ΥΠΗΡΕΣΙΕΣ ΒΑΣΙΣΜΕΝΕΣ ΣΤΗΝ ΤΟΠΟΘΕΣΙΑ»



2.1 ΠΕΡΙΓΡΑΦΗ

Οι νέες τεχνολογίες των υπηρεσιών δικτύου αλλάζουν ολοένα και περισσότερο όχι μόνο τον τρόπο με τον οποίο σχεδιάζεται πλέον το λογισμικό αλλά κυρίως τον τρόπο με τον οποίο αυτό θα χρησιμοποιείται. Μια «υπηρεσία βασισμένη στην γεωγραφική τοποθεσία» - του χρήστη – (location based service) μπορεί να περιγραφεί σα μια εφαρμογή που αντιδρά σύμφωνα με ένα γεωγραφικό ερέθισμα. Το ερέθισμα αυτό μπορεί να είναι η είσοδος του ονόματος κάποιας τοποθεσίας (πόλη, ταχυδρομικός κώδικας, οδός κλπ) σε μια φόρμα (ιστοσελίδας, κινητού κ.α.), η τοποθεσία ενός χρήστη κινητής συσκευής, η ακριβής θέση ενός αυτοκινήτου την ώρα της οδήγησης.

«Χρησιμοποιώντας τη γνώση της τοποθεσίας κάποιου ανθρώπου ή/και αντικειμένου αποσκοπούν στο να παράσχουν χρήσιμη πληροφορία.»

Οι «υπηρεσίες βασισμένες στη γεωγραφική τοποθεσία» εκμεταλλεύονται την πληροφορία γύρω από τη θέση του χρήστη, μέσω ειδικά διαμορφωμένων συσκευών, με σκοπό να του παράσχουν δυνατότητες άμεσα σχετιζόμενες με αυτή (τη θέση του). Ταυτόχρονα, η σύγκλιση των τεχνολογιών του Διαδικτύου, των ασύρματων και κινητών συσκευών σε συνδυασμό με τις όλο και πιο αξιόπιστες μεθόδους προσδιορισμού της τοποθεσίας, υπόσχεται μια νέα γενιά υπηρεσιών, που θα απαρτίζουν ένα κατανεμημένο περιβάλλον, στο οποίο η γεωγραφική θέση των χρηστών θα χρησιμοποιείται με σκοπό την υπολογιστική βελτιστοποίηση και την παροχή αξιόπιστων και ποιοτικών υπηρεσιών.

Η υπηρεσιοστρεφής αρχιτεκτονική (service-oriented architecture) έρχεται να φέρει αλλαγές ανάλογες με αυτές της αντικειμενοστρεφούς αρχιτεκτονικής, που αφορούν όχι τόσο στην τεχνική ανάπτυξης του λογισμικού όσο στη γενικότερη αντίληψη που θα το διέπει. Με κεντρική έννοια ένα «παντού και πάντα (ubiquitous) δικτυωμένο υπολογιστικό σύστημα», δημιουργείται μια υπηρεσία για να καλύψει μια απόλυτα

συγκεκριμένη ανάγκη, η οποία - η υπηρεσία – αν και δε θα είναι διαθέσιμη τοπικά στο χρήστη, θα μπορεί ο ίδιος ανά πάσα στιγμή να επωφεληθεί από αυτήν μέσω κάποιου δικτύου (intranet, internet).

Συνδυάζοντας τώρα τη νέα αυτή τεχνολογία με τη δυναμική των σχετικών με τη θέση υπηρεσιών (location-based services), αποκτούμε μια πολύ χρήσιμη για τον κινητό χρήστη και προσοδοφόρα για τον παροχέα υπηρεσία. Ο τελικός χρήστης θα έχει τη δυνατότητα να επιλέξει μέσα από μια ποικιλία παρεχόμενων υπηρεσιών και να απολαύσει έτσι ολοκληρωμένες σχετικές με τη θέση υπηρεσίες δικτύου σε ένα πλήρως ανταγωνιστικό περιβάλλον. Για τους παροχείς υπηρεσιών το μοντέλο αυτό ενισχύει τις επιχειρηματικές συνεργασίες και διευρύνει τον κύκλο εργασιών τους. Οι περιορισμοί των κινητών συσκευών ως προς την υπολογιστική τους δεινότητα απαιτούν τη παρεμβολή ενός αντιπροσώπου, ο οποίος θα αναλάβει το ρόλο του αιτούντα (requestor) στην κλήση μιας υπηρεσίας, μέσα στα πλαίσια της υπηρεσιοστρεφούς αρχιτεκτονικής.

2.2 ΟΙ ΣΧΕΤΙΚΕΣ ΜΕ ΤΗ ΘΕΣΗ ΥΠΗΡΕΣΙΕΣ

Οι σχετικές με τη θέση υπηρεσίες είναι υπηρεσίες σχετικά νέες για τα δίκτυα κινητών τηλεπικοινωνιών. Παρότι η ανάπτυξή και η εφαρμογή τους έχει μελετηθεί σε μεγάλο βαθμό, δεν έχουμε δει ακόμα πολλές να λειτουργούν στην πράξη. Παρότι οι προδιαγραφές και για το υλικό αλλά και το λογισμικό που θα υποστηρίξουν αυτές τις υπηρεσίες είναι πλέον σαφείς, οι εταιρείες κινητής τηλεφωνίας παρουσιάζονται διστακτικές να τις εισάγουν στην καθημερινότητα, αναλογιζόμενες κυρίως την ανταποδοτικότητα μιας τέτοιας επένδυσης, κατά πόσο, δηλαδή, οι υπηρεσίες αυτές θα βρουν θετικό το συνδρομητικό κοινό, ώστε να τις αξιοποιήσει και να αποφέρει τελικά κέρδη στους τηλεπικοινωνιακούς παροχείς. Με σκοπό ακριβώς αυτόν, την ανάλυση, δηλαδή, της αγοράς για τις σχετικές με τη θέση υπηρεσίες, έχουν χρηματοδοτηθεί στο εξωτερικό έρευνες, οι οποίες αποδεικνύουν τη δυναμική των σχετικών με

τη θέση υπηρεσιών στο νέο τηλεπικοινωνιακό τοπίο, το οποίο χαρακτηρίζεται από έντονα σημάδια ωρίμανσης.

Σύμφωνα, λοιπόν, με αυτές τις έρευνες, οι σχετικές με τη θέση υπηρεσίες λαμβάνουν μια εξέχουσα θέση στις υπηρεσίες προστιθέμενης αξίας (value-added services) που παρέχει μια εταιρεία κινητής τηλεφωνίας στους συνδρομητές της. Ως τέτοια, όμως, είναι αναμενόμενο ότι δεν απευθύνεται στο σύνολο του συνδρομητικού κοινού αλλά σε μια μερίδα αυτού, που είναι περισσότερο εξοικειωμένη με τη χρήση του κινητών συσκευών αλλά και πιο απαιτητική από αυτά που έχει τη δυνατότητα μια κινητή συσκευή να της παρέχει.

2.3 ΚΑΤΗΓΟΡΙΕΣ ΣΧΕΤΙΚΩΝ ΜΕ ΤΗ ΘΕΣΗ ΥΠΗΡΕΣΙΩΝ

Υπάρχει μεγάλη ποικιλία σχετικών με τη θέση υπηρεσιών, από τις οποίες μεγαλύτερο ενδιαφέρον από άποψης χρησιμότητας έχουν οι παρακάτω τέσσερις κατηγορίες:

- Υπηρεσίες πληροφόρησης: Βρίσκουν εφαρμογή σε σχέσεις «Επιχείρησης-προς-Πελάτη» (B2C), «Πελάτη-προς-Επιχείρηση» (C2B) αλλά και «Καταναλωτή-προς-Καταναλωτή» (C2C). Με τις υπηρεσίες αυτές γνωστοποιείται η θέση της πλησιέστερης στο συνδρομητή υπηρεσίας (εστιατόριο, πρατήριο βενζίνης, φαρμακείο κτλ.), όπως επίσης νέα για την κίνηση στους δρόμους, βοήθεια για την καθοδήγησή του σε μια άγνωστή πόλη κοκ.
- Υπηρεσίες έκτακτης ανάγκης: Αφορούν τη γνωστοποίηση της θέσης κάποιου συνδρομητή που βρίσκεται σε κίνδυνο προς τις υπηρεσίες διάσωσης (Πυροσβεστική, Ασθενοφόρο κ.ά.) και δεν κομίζουν κέρδη στις υπηρεσίες κινητής τηλεφωνίας.
- Υπηρεσίες παρακολούθησης στόλου: Αναφέρονται περισσότερο σε σχέσεις «Επιχείρησης-προς-Επιχείρηση» (B2B) και χρησιμοποιούνται για τον εντοπισμό της θέσης εξωτερικών πόρων με σκοπό τη βελτιστοποίηση της χρήσης τους και την εγγύηση της ασφάλειάς τους.

- Υπηρεσίες για τις εταιρείες κινητής τηλεφωνίας: Η πληροφορία για τη θέση των συνδρομητών μπορεί να χρησιμοποιηθεί και εσωτερικά σε μια εταιρεία κινητής τηλεφωνίας με σκοπό τον καλύτερο σχεδιασμό του δικτύου, τη βελτιστοποίηση της χρήσης των ραδιοσυχνοτήτων και της κατανομής των καναλιών.

2.4 Ο ΠΡΟΣΔΙΟΡΙΣΜΟΣ ΤΗΣ ΘΕΣΗΣ

Βασικό δομικό στοιχείο για την ανάπτυξη και προσφορά των σχετικών με τη θέση υπηρεσιών είναι η τεχνολογία που χρησιμοποιείται για τον προσδιορισμό της θέσης του κινητού χρήστη. Η πληροφορία αυτή θα συσχετιστεί με την αντίστοιχη για τις θέσεις των διαθέσιμων υπηρεσιών και με αυτόν τον τρόπο θα δοθεί, για παράδειγμα, η πληροφόρηση για το κοντινότερο φαρμακείο ή ο τοπικός οδικός χάρτης.

Είναι τρεις οι βασικές μέθοδοι που χρησιμοποιούνται για τον προσδιορισμό της θέσης ενός κινητού σε ένα GSM δίκτυο. Με βάση τη γεωγραφική ακρίβεια που προσφέρουν και ξεκινώντας από αυτήν με τη μικρότερη ακρίβεια έχουν ως εξής:

- Cell-ID (Αναγνωστικού Κυψέλης): Ο προσδιορισμό της θέσης γίνεται σε σχέση με τα αναγνωριστικά των κυψελών του δικτύου, που είναι διαθέσιμα χωρίς επιπλέον εξοπλισμό. Η ακρίβεια της μεθόδου εξαρτάται από την πυκνότητα του κυψελωτού δικτύου και είναι ίση με το μέγεθος που μπορεί να έχει μια κυψέλη, δηλαδή από 100μ. έως και 35 χλμ. Μια βελτίωση της μεθόδου αυτής, που συνδυάζει το αναγνωριστικό της κυψέλης με δύο ραδιομετρήσεις του δικτύου και ονομάζεται Cell ID++, παρέχει ακρίβεια 100-200 μέτρα.
- Observed Time Difference (OTD – Σημειούμενη Χρονική Διαφορά): Είναι μια οικογένεια μεθόδων, από τις οποίες στα GSM δίκτυα χρησιμοποιείται η E-OTD (Enhanced OTD) και η οποία παρέχει ακρίβεια της τάξεως των 60-200 μέτρων ανάλογα με το αν ο συνδρομητής βρίσκεται σε αγροτική ή αστική περιοχή.

- Assisted GPS (A-GPS – Υποβοηθούμενου GPS): Χρησιμοποιεί την ήδη υπάρχουσα υποδομή των δορυφόρων του συστήματος GPS, αλλά απαιτεί και την εγκατάσταση επιπλέον εξοπλισμού στο δίκτυο, καθώς επίσης και την αλλαγή των συσκευών κινητών τηλεφώνων. Προσφέρει συνολικά την καλύτερη ακρίβεια, καθώς η θέση του κινητού προσδιορίζεται γεωγραφικά με περιθώριο λάθους 10-20μ.

2.5 ΟΙ ΥΠΗΡΕΣΙΕΣ ΔΙΚΤΥΟΥ

Οι υπηρεσίες δικτύου (web services) έρχονται να καλύψουν μια δεδομένη ανέκαθεν ανάγκη των επιχειρήσεων: την ευρύτερη δυνατή συνεργασία και ολοκλήρωση των επιμέρους συστημάτων λογισμικού τους. Η εισαγωγή των υπηρεσιών δικτύου και της υπηρεσιοστρεφούς αρχιτεκτονικής υπόσχεται να καλύψει αυτήν ακριβώς την απαίτηση και να προχωρήσει και παραπέρα προσφέροντας γενικότερα ένα πιο ευέλικτο και κατανεμημένο υπολογιστικό μοντέλο.

Η υπηρεσία δικτύου είναι μια συνιστώσα λογισμικού ανεξάρτητη από πλατφόρμα και υλοποίηση που μπορεί να:

- Περιγραφεί (described) με τη χρήση μιας γλώσσας περιγραφής υπηρεσιών (π.χ. WSDL).
- Δημοσιευτεί (published) σε μητρώο υπηρεσιών.
- Ανακαλυφθεί (discovered) μέσω τυποποιημένων μηχανισμών (είτε κατά την ώρα της εκτέλεσης είτε κατά την ώρα της ανάπτυξης).
- Κληθεί (invoked) μέσω μιας δηλωμένης Προγραμματιστικής διεπαφής Εφαρμογής – Π.Ε (Application Programming Interface – API), συνήθως πάνω από δίκτυο.
- Συντεθεί (composed) με άλλες υπηρεσίες.

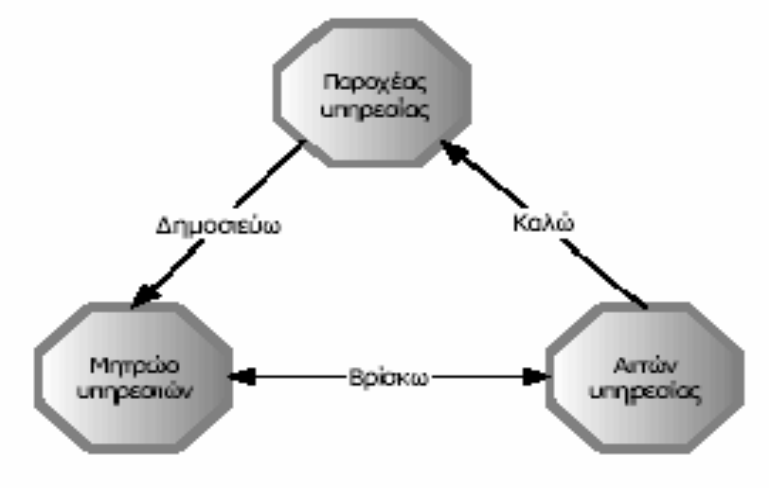
Ένα σημαντικό σημείο είναι ότι οι υπηρεσίες δικτύου δεν είναι απαραίτητο να υπάρχουν στο Παγκόσμιο Πλέγμα Πληροφοριών (ΠΠΠ).

Μια υπηρεσία δικτύου μπορεί να βρίσκεται οπουδήποτε στο δίκτυο, στο διαδίκτυο ή σε κάποιο εσωτερικό δίκτυο επιχείρησης.

Από τεχνική σκοπιά η υπηρεσία δικτύου δεν είναι τίποτα περισσότερο από μια συλλογή από μία ή περισσότερες σχετιζόμενες λειτουργίες που είναι προσβάσιμες πάνω από ένα δίκτυο και περιγράφονται από την περιγραφή υπηρεσίας. Μέχρι αυτό το επίπεδο η έννοια των υπηρεσιών δικτύου δεν είναι καινούρια.. Με τις υπηρεσίες δικτύου η βιομηχανία πληροφορικής προσπαθεί να αντιμετωπίσει την πρωταρχική πρόκληση της κατανεμημένης επεξεργασίας που παραμένει εδώ και δεκαετίες – τον εντοπισμό και την πρόσβαση σε απομακρυσμένα συστήματα. Η μεγάλη διαφορά είναι ότι τώρα η βιομηχανία προσεγγίζει το πρόβλημα χρησιμοποιώντας «ανοιχτή» τεχνολογία (XML και πρωτόκολλα Διαδικτύου) και «ανοιχτά» πρότυπα που διευθύνονται από ευρείες συνεργασίες όπως η World Wide Web Consortium (W3C), η οποία κατευθύνει την εξέλιξη των προδιαγραφών για το SOAP και το WSDL.

2.6 ΥΠΗΡΕΣΙΟΣΤΡΕΦΗΣ ΑΡΧΙΤΕΚΤΟΝΙΚΗ

Κάθε φορά που εφαρμόζονταν οι τεχνολογίες των υπηρεσιών δικτύου σε ένα πρόβλημα ολοκλήρωσης εφαρμογών ανέκυπτε ένα μοτίβο, το οποίο στη συνέχεια ονομάστηκε Υπηρεσιοστρεφής Αρχιτεκτονική *(YA) (service-oriented architecture – SOA).* Η ΥΑ είναι μια απλή αρχή, η οποία εφαρμόζεται σε μια μεγάλη ποικιλία καταστάσεων που κάνουν χρήση υπηρεσιών δικτύου. Το σχήμα που ακολουθεί (Σχήμα 2.1) απεικονίζει τους βασικούς ρόλους και λειτουργίες σε μια ΥΑ.



Σχήμα 2.1

Οποιαδήποτε υπηρεσιοστρεφής αρχιτεκτονική περιέχει τρεις ρόλους: τον αιτούντα την υπηρεσία (service requestor), τον παροχέα της υπηρεσίας (service provider) και το μητρώο των υπηρεσιών (service registry). Ο παροχέας της υπηρεσίας είναι υπεύθυνος για τη δημιουργία της περιγραφής της, δημοσιεύοντας την σε ένα ή περισσότερα μητρώα υπηρεσιών και αποδεχόμενός μηνύματα για την κλήση της υπηρεσίας δικτύου από έναν ή και περισσότερους αιτούντες την υπηρεσία. Ο αιτών την υπηρεσία είναι υπεύθυνος για την εύρεση μιας περιγραφής υπηρεσίας που μπορεί να έχει δημοσιευθεί σε ένα ή περισσότερα μητρώα και είναι επίσης υπεύθυνος για την κλήση και συνδυασμό, αν χρειάζεται, υπηρεσιών δικτύου που διατίθενται από παροχείς υπηρεσιών. Το μητρώο υπηρεσιών είναι υπεύθυνο για τη δημοσιοποίηση των περιγραφών των υπηρεσιών δικτύου που δημοσιεύονται προς αυτό από τους παροχείς αλλά και για τη διευκόλυνση των αιτούντων να ψάξουν τη συλλογή με τις περιγραφές υπηρεσιών που περιέχονται σ' αυτό. Μια ΥΑ περιλαμβάνει επίσης και τρεις λειτουργίες: δημοσιεύω (publish), βρίσκω (find) και καλώ (bind).

2.7 ΓΕΩΓΡΑΦΙΚΗ ΤΟΠΟΘΕΣΙΑ – ΧΩΡΙΚΟ ΠΛΑΙΣΙΟ – ΠΡΟΒΛΕΨΗ

Ας εστιάσουμε, στη συνέχεια, στην «τοποθεσία» ως μια εξέχουσα συνιστώσα του πλαισίου (context) των περισσότερων συστημάτων. Σε ένα

περιβάλλον κινητής φύσεως – π.χ. κινητή τηλεφωνία – η θέση ενός χρήστη προσδιορίζεται από δύο παραμέτρους: την ίδια την θέση (στο σύστημα στο οποίο αυτή μετράται) και τη χρονική στιγμή στην οποία συμβαίνει. Μείζονος σημασίας στο πεδίο της επίγνωσης πλαισίου δεν είναι μόνο η θέση καθεαυτή αλλά και η δυνατότητα πρόβλεψής της. Η ικανότητα, δηλαδή, του συστήματος να εκτιμά δυναμικά και αξιόπιστα τις μελλοντικές θέσεις ενός χρήστη, βασιζόμενο στην υπάρχουσα πληροφορία γύρω από το χώρο στον οποίο αυτός βρίσκεται (τρέχουσα θέση, προηγούμενες θέσεις κλπ).

Δύο είναι τα βασικά στοιχεία που θα σταθούν κύριοι αρωγοί στην προσπάθεια για την πρόβλεψη της θέσης:

- Η πραγματοποίηση της κινητικότητας (mobility): ο προσδιορισμός της τροχιάς ενός κινητού χρήστη.
- Ο προσδιορισμός της θέσης: η εκτίμηση/υπολογισμός μιας τοποθεσίας πάνω σε αυτή την τροχιά ή η χρονική στιγμή στην οποία ο χρήστης θα φτάσει σε μια συγκεκριμένη τοποθεσία.

Μέχρι σήμερα έχουν προταθεί διαφορετικές στρατηγικές πρόβλεψης, κάθε μια με διαφορετική στόχευση, που μπορούν να διαχωριστούν σε δύο κατηγορίες: προσέγγιση με βάση την κυψέλη (cell-based) και με βάση τον χάρτη (map-based).

Στην προσέγγιση με βάση την κυψέλη ο γεωγραφικός χώρος τμηματοποιείται σε κανονικά εξάγωνα – τις κυψέλες – οι οποίες, καθορίζονται από την αρχιτεκτονική του κυψελωτού δικτύου ή/και από τις ανάγκες της εκάστοτε εφαρμογής. Η κινητικότητα προσδιορίζεται από σα μια σειρά από κυψέλες που αποτελούν την τροχιά του χρήστη. Η συγκεκριμένη προσέγγιση έχει παίξει σημαντικό ρόλο στα σύγχρονα δίκτυα κινητής τηλεφωνίας στην προσπάθεια να αυξηθεί η αποδοτικότητά τους. Δύο από τα σημαντικότερα μειονεκτήματά της είναι τα εξής: Καταρχάς, δεν έχει την ικανότητα να εντοπίσει με μεγάλη ακρίβεια τη θέση ενός χρήστη μιας και η ακτίνα μιας κυψέλης εκτείνεται

από 150 έως και 30.000 μέτρα. Κι ακόμη, δε μπορεί να μοντελοποιήσει την τροχιά ενός χρήστη με ακρίβεια, μιας και το μόνο που μπορεί να υπολογίσει είναι η πιθανότητα αλλαγής κυψέλης του χρήστη, βασισμένη στην πλευρά της κυψέλης (κανονικό εξάγωνο) από την οποία εξέρχεται αυτός.

Αντίθετα, στην προσέγγιση με βάση το χάρτη η στρατηγική στοχεύει στον προσδιορισμό της θέσης του χρήστη σε επίπεδο δρόμου του οδικού δικτύου, χρησιμοποιώντας συστήματα όπως το GPS. Η αποδοχή που κάνει η μέθοδος έγκειται στο ότι οι διαδρομές των χρηστών πραγματοποιούνται κατά κύριο λόγο με οχήματα κι επομένως πάνω σε κάποιο υπάρχοντα δρόμο. Σημαντικό στοιχείο στην εφαρμογή της είναι η ύπαρξη μεγάλης ποσότητας οργανωμένης πληροφορίας που αφορά τα οδικά δίκτυα και είναι διαθέσιμη ανά πάσα στιγμή για να υποβοηθήσει την πρόβλεψη της κινητικότητας και των βασιζόμενων σε αυτή υπηρεσιών.

Στη δεύτερη προσέγγιση, επίσης, αν ένας χρήστης δηλώσει κάποιο επιθυμητό προορισμό, τότε ακόμη και ένας συμβατικός αλγόριθμος δρομολόγησης θα μπορούσε να χρησιμοποιηθεί για να προβλέψει μια τροχιά και να εκτιμήσει το χρόνο που θα χρειαστεί (ο χρήστης) για να φτάσει εκεί. Ο χρήστης έχει, βέβαια, πολλές επιλογές όσον αφορά την τροχιά που θα ακολουθήσει σε ένα συνηθισμένο οδικό δίκτυο για να φτάσει σε ένα δεδομένο προορισμό. Οι σχετικές μελέτες, όμως, αποδεικνύουν ότι οι χρήστες τις περισσότερες φορές παρουσιάζουν ένα σημαντικό βαθμό τακτικότητας στις κινήσεις τους. Μάλιστα, σε ένα σουηδικό συνέδριο έρευνας για τις μεταφορές παρουσιάστηκαν τα αποτελέσματα μιας μελέτης όπου οι χρήστες φέρονται να ακολουθούν τις ίδιες τροχιές για το ίδιο ζευγάρι εκκίνησης-προορισμού κατά 70%.

2.8 ΠΡΟΒΛΕΨΗ ΣΕ ΥΒΡΙΔΙΚΑ ΚΑΙ AD HOC ΑΣΥΡΜΑΤΑ ΔΙΚΤΥΑ

Η επόμενη μεγάλη γενιά στον τομέα των δικτύων θα εφαρμόζει υβριδικές αρχιτεκτονικές που συνδυάζουν ιδέες από κυψελωτά αλλά και ad hoc δίκτυα. Το όραμα της παροχής υπηρεσιών στους κινητούς χρήστες σε πραγματικό χρόνο απαιτεί να προσεγγίσουμε τη διαχείριση της κινητικότητας (mobility) με τρόπο διορατικό και προδραστικό (proactive). Αν οι κινήσεις ενός χρήστη μπορούσαν να προβλεφθούν με σχετική ακρίβεια σε ένα υβριδικό δικτυακό περιβάλλον τότε η διαχείριση των πόρων του συστήματος και απόκριση αυτού στα αιτήματα του χρήστη θα ήταν ιδανική.

Ενώ τα δίκτυα 3G είναι βασισμένα πρωτίστως στην ιδέα της τμηματοποίησης μέσω κυψελών τα δίκτυα 4G θα είναι κατ' ουσία υβριδικά, κάνοντας χρήση των κυψελών (ή σταθμών βάσης) αλλά και εκμεταλλευόμενα τις αρχές που διέπουν τα ad hoc δίκτυα (hotspots, wireless LANs etc). Ένα ad hoc δίκτυο είναι ένα αυτο-ρυθμιζόμενο και αυτο-συντηρούμενο δίκτυο απαρτιζόμενο από ισότιμους κόμβους (κινητούς δρομολογητές). Για να εκπληρωθεί το όραμα της επόμενης γενιάς δικτύων υπάρχει η ανάγκη για μια μεθοδολογία διαχείρισης συνεχούς ροής πληροφορίας παρόλη την, εκ των πραγμάτων, συχνή αλλαγή στην τοπολογία και έλλειψη κανονικότητας στην κίνηση των χρηστών. Για παράδειγμα, συνεχής ροή χρειάζεται για τη διαμεταγωγή πακέτων με αυστηρά χρονικά κριτήρια και γι' αυτό οι διαδικασίες αλληλοεύρεσης κόμβων και συντονισμού τους πρέπει να διαθέτει ένα βαθμό διορατικότητας (proactiveness). Αν οι τοπολογικές μεταβολές μπορούσαν να προβλέπονται με σχετική ακρίβεια, τότε και η ανακατασκευή των δρομολογήσεων ή η εξεύρεση κόμβων θα μπορούσε με τη σειρά της να προηγείται των τοπολογικών μεταβολών.

Η ανάγκη για ένα σχήμα που να παρέχει δυνατότητες πρόβλεψης στα υπάρχοντα 2G και 3G δίκτυα περνάει και μέσα από το πρίσμα της μελλοντική μετάβασης στις τεχνολογίες 4G, η οποία θα πρέπει να γίνει όσο το δυνατόν πιο ομαλά. Και μάλιστα θα πρέπει να είναι αποδοτικό τόσο σε δίκτυα με σταθερή υποδομή (π.χ. κυψελωτό δίκτυο κινητής

τηλεφωνίας) όσο και σε περιβάλλοντα χωρίς ξεκάθαρη υποδομή/τοπολογία (ad hoc δίκτυα). Το σχήμα αυτό θα ήταν επιθυμητό, ακόμη, να είναι ευαίσθητο στην τυχαιότητα που χαρακτηρίζει την κίνηση των χρηστών και να παράσχει διαφάνεια ανάμεσα σε ετερογενή συστήματα ανεξαρτήτου τεχνολογίας. Για να είναι βιώσιμο θα πρέπει να προσφέρει υψηλό βαθμό ακρίβειας στις προβλέψεις επιφέροντας, ταυτόχρονα, την ελάχιστη δυνατή υπολογιστική επιβάρυνση.

Τα περισσότερα συστήματα πρόβλεψης που έχουν προταθεί και βασίζονται σε κυψέλες κάνουν χρήση μιας βάσης γνώσης που καταγράφει τις προηγούμενες κινήσεις των χρηστών. Οι αλγόριθμοι που χρησιμοποιούν βασίζονται στην αναδρομική πιθανότητα του να πραγματοποιήσει ο χρήστης μια συγκεκριμένη κίνηση, σε συνδυασμό με παράγοντες όπως η κατεύθυνση κίνησης και η ταχύτητα. Όταν οι κινήσεις των χρηστών παρουσιάζουν κάποια κανονικότητα τα συστήματα αυτά επιτυγχάνουν πρόβλεψη με καλή ακρίβεια. Ακόμη και με μικρές, όμως παρεκκλίσεις, οι παραδοσιακοί αλγόριθμοι βασισμένοι σε κριτήρια θέσης, κατεύθυνσης, ταχύτητας, χρόνου, κανόνων Bayes αποτυγχάνουν.

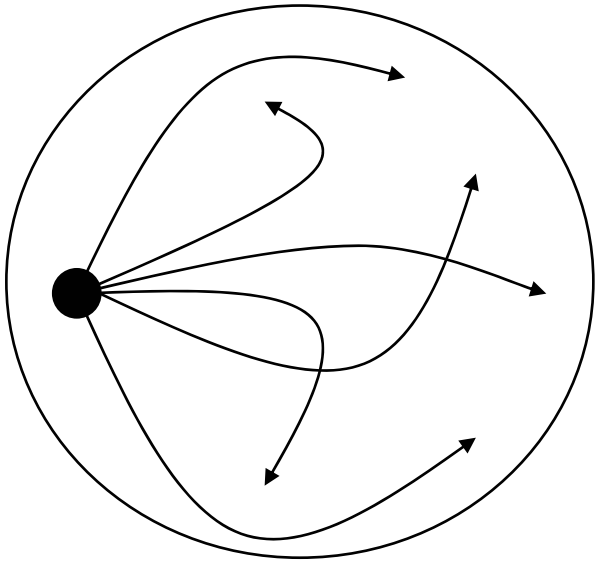
Αυτή τους η αποτυχία αποδίδεται στην έλλειψη προσαρμοστικότητας στις αλλαγές της συμπεριφοράς του χρήστη. Στη σύγχρονη βιβλιογραφία συναντά κανείς αλγόριθμους που επιχειρούν να εκμεταλλευτούν την κανονικότητα στην ανθρώπινη συμπεριφορά – αναζητώντας περιοδικές ή/και επαναλαμβανόμενες δραστηριότητες – και άλλους που χρησιμοποιούν το ιστορικό κινήσεων ενός χρήστη για να κατασκευάσουν μια πιθανή εικόνα των μελλοντικών κινήσεων και με βάση αυτή να διαθέσουν τους αναγκαίους πόρους κατά μήκος της προβλεπόμενης διαδρομής.

Τα ad hoc δίκτυα βρίσκουν σήμερα εφαρμογή κυρίως σε στρατιωτικά περιβάλλοντα ή σε επείγουσες επιχειρήσεις αναζήτησης και διάσωσης. Η ουσιαστική πρόβλεψη των μελλοντικών κινήσεων βασισμένη στο ιστορικό των κινήσεων δεν είναι, όμως, ιδιαίτερα εφαρμόσιμη ως τώρα εξαιτίας της

δυναμικής ανακατανομής της τοπολογίας και των ετερογενών αναγκών που προτάσσουν οι διάφορες περιπτώσεις.

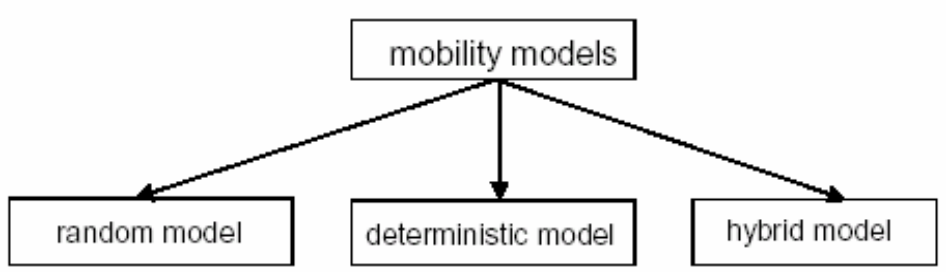
ΕΝΟΤΗΤΑ 3η

«ΚΙΝΗΤΙΚΟΤΗΤΑ - ΜΟΝΤΕΛΑ ΠΡΟΣΟΜΟΙΩΣΗΣ ΚΙΝΗΤΙΚΟΤΗΤΑΣ»



3.1 ΕΙΣΑΓΩΓΗ

Η προσομοίωση μοντέλων δικτύων κινητών τερματικών θεωρείται ιδιαίτερα κρίσιμη στη μελέτη της κινητικότητας και ανάγεται στην προσομοίωση της κινητικής ικανότητας του χρήστη. Στο σχήμα 3.1 κατηγοριοποιούνται οι προσεγγίσεις κινητικότητας (mobility) χρηστών σε δίκτυα.



Σχήμα 3.1

Τα Μοντέλα Τυχαίας Κινητικότητας προσομοιώνουν τυχαίες κινήσεις σε καθορισμένη περιοχή και η υλοποίησή τους είναι εύκολη. Οι παράμετροι της κίνησης αλλάζουν κατά την προσομοίωση παράγοντας έτσι τυχαίες κατευθύνσεις. Οι περισσότερες προσομοιώσεις δικτύων κινητών τερματικών ακολουθούν αυτό το μοντέλο το οποίο είναι κατανοητό επειδή οι αναλυτές εστιάζουν εκτενέστερα στην ανάπτυξη νέων επικοινωνιακών πρωτοκόλλων και όχι στην κατασκευή μοντέλων κίνησης. Τα μοντέλα αυτού του τύπου χρησιμοποιούνται για στρατιωτικές εφαρμογές ή για εφαρμογές έκτακτης ανάγκης. Αλλά η τυχαία κίνηση δεν αντιστοιχίζεται με την πραγματική κίνηση, επειδή για παράδειγμα υπάρχουν και φυσικοί περιορισμοί στην πραγματική κίνηση (π.χ. Βουνά, Τοίχοι κλπ).

Σε αντίθεση με τα Μοντέλα Τυχαίας Κινητικότητας, υπάρχουν τα Αιτιοκρατικά Μοντέλα Κίνησης που αναπαριστούν την πραγματική πορεία κίνησης χρηστών σε μια περιοχή προσομοίωσης. Τέτοια μοντέλα μπορούν κάλλιστα να αναπαραστήσουν στην πραγματική κίνηση αλλά

περιορίζονται σε προκαθορισμένη περιοχή. Η απόκτηση τέτοιων σημείων κίνησης αποτελεί υπερβολική δουλειά από την θέση του προσομοιωτή. Πρέπει να προετοιμαστούν ερωτηματολόγια θέσης, χιλιάδες αισθητήρες αναφοράς θέσης, μηχανές ιχνηλάτησης και άλλα όργανα που θα προωθούν την πληροφορία θέσης της περιοχής προσομοίωσης στον προσομοιωτή.

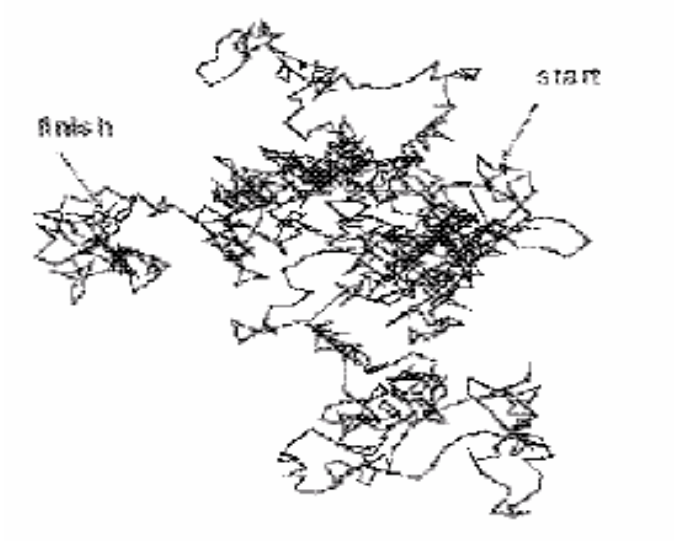
Οι χρήστες όμως δεν κινούνται τελικά πάντα τυχαία αλλά με έναν βαθμό γενικότητας της κίνησής τους. Έτσι πρέπει να βρεθεί μια χρυσή τομή μεταξύ των μη πραγματικών, μη προκαθορισμένων, τυχαίων κινήσεων με προκαθορισμένα και πραγματικά στίγματα κίνησης.

Τα Υβριδικά Μοντέλα Κίνησης προσθέτουν επιπλέον πληροφορίες τυχαιότητας κίνησης καθώς και πληροφορίες περιορισμού της, με σκοπό να αντιστοιχεί σε πραγματικές κινήσεις με κάποια αποδεκτή πιθανότητα. Τέτοια μοντέλα κυμαίνονται μεταξύ της απλότητας της τυχαιότητας την κίνησης και της αυξημένης πολυπλοκότητας της αιτιοκρατικής κίνησης.

3.2 ΜΟΝΤΕΛΑ ΤΥΧΑΙΑΣ ΚΙΝΗΤΙΚΟΤΗΤΑΣ

Οι πρωταρχικές ενέργειες προσομοίωσης σε δίκτυα κινητών τερματικών χρησιμοποιούν σενάρια που βασίζονται στον σχεδιασμό των κυψελών με την χρήση των σταθμών βάσεων (κεντρικοποιημένη δομή). Στόχος του προσομοιωτή είναι η δημιουργία μιας λίστας κυψελών από τις οποίες διέρχεται ο κινητός χρήστης. Στο μονοδιάστατο μοντέλο κίνησης η κάθε κυψέλη έχει δύο μόνο γειτονικές κυψέλες με δύο μόνο περιορισμούς κίνησης ενώ στο δυσδιάστατο μοντέλο μπορεί να υπάρξει ένα πλήθος γειτονικών κυψελών, κυρίως από 6 και πάνω, που προσομοιώνεται μια περιοχή κάλυψης όπως μια πόλη στην οποία οι χρήστες κινούνται από γειτονική κυψέλη προς γειτονική κυψέλη έχοντας πιθανότητες ύπαρξης σε μία από τις 6 κυψέλες. Αυτό το είδος πορείας των χρηστών καλείται Πορεία Κίνησης Τυχαίου Μοντέλου.

Το μοντέλο ορίζεται ως εξής: Σε κάθε διακριτή χρονική στιγμή t ο χρήστης κινείται σε μία από τις γειτονικές κυψέλες με πιθανότητα p ή μένει στην τρέχουσα κυψέλη με πιθανότητα 1-p ανεξάρτητη από την προηγούμενη κίνησή του. Εάν ο χρήστης αποφασίσει να κινηθεί σε άλλη κυψέλη τότε μια από τις γειτονικές υποψήφιες κυψέλες επιλέγεται με ίση πιθανότητα. Επισημαίνεται ότι οι περισσότεροι χρήστες σε τέτοιου είδους δίκτυα είναι πεζοί. Η κίνησή τους λαμβάνει χώρα μέσα σε μια βασική γεωγραφική έκταση με συχνές ενέργειες τύπου «κινούμαι-σταματώ» και συχνές αλλαγές κατευθύνσεων. Οι αλυσίδες Markov και οι διεργασίες Markov μπορούν να χρησιμοποιηθούν για τον μαθηματικό φορμαλισμό του μοντέλου.



Σχήμα 3.2

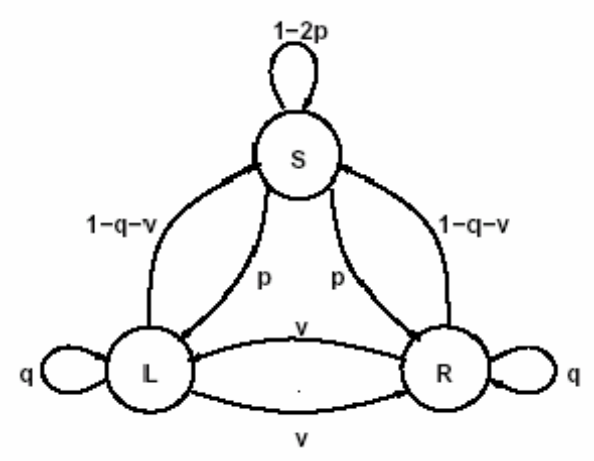
Η διεργασία Markov είναι μια στοχαστική διεργασία που η μελλοντική της συμπεριφορά προκαθορίζεται μόνο από την τρέχουσα κατάσταση. Αυτή η ιδιότητα επιτρέπει την διεργασία να μοντελοποιεί την Τυχαία Πορεία Κίνησης της οποίας η συμπεριφορά αποφασίζεται μόνο από την τρέχουσα κατάσταση. Εάν οι καταστάσεις της διεργασίας Markov λαμβάνουν μόνον διακριτές τιμές, η οποία καλείται αλυσίδα Markov, τότε μπορεί να περιγραφεί μια μετάβαση με της χρήση μια μήτρας πιθανοτήτων P. Ένα στοιχείο της μήτρας P[i,j] περιγράφει την πιθανότητα μετάβασης από την i κατάσταση στην j.

$$P = \begin{bmatrix} p_{00} & p_{01} & p_{02} & \cdots \\ p_{10} & p_{11} & p_{12} & \cdots \\ p_{20} & p_{21} & p_{22} & \cdots \\ \vdots & \vdots & \vdots & \ddots \end{bmatrix}$$

Σχήμα 3.3

ΜΟΝΤΕΛΟ ΚΙΝΗΣΗΣ MARKOV

Πολλοί αναλυτές χρησιμοποιούν το Μοντέλο Κίνησης Markov το οποίο αποτελεί την μονοδιάστατη έκδοση της Πορείας Τυχαίας Κίνησης. Η έκδοση αυτή περιγράφεται στον 3Χ3 πίνακα του σχήματος 3.4.



Σχήμα 3.4

Σύμφωνα με το μοντέλο ένας χρήστης μπορεί να βρίσκεται σε μία από τις τρεις καταστάσεις:

1. Κατάσταση L – κατάσταση αριστερής κίνησης
2. Κατάσταση R – κατάσταση δεξιάς κίνησης
3. Κατάσταση S – μόνιμη κατάσταση

Το μειονέκτημα του μοντέλου αυτού παρουσιάζεται στην έλλειψη της πραγματικής κίνησης επειδή δεν ορίζει συντεταγμένες σημείων στόχων αφήνοντας μια χαοτική και άσκοπη κίνηση να λαμβάνει χώρα.

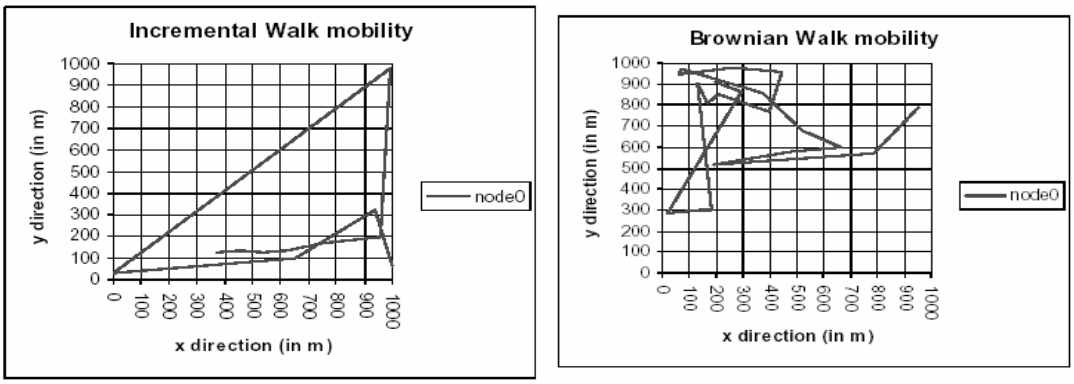
ΜΟΝΤΕΛΟ ΚΙΝΗΣΗΣ BROWN

Το μοντέλο αυτό υποστηρίζει την κίνηση σε πολικές συντεταγμένες. Ένα κινητό στοιχείο μπορεί να κινηθεί ελεύθερα μέσα στα όρια της περιοχής προσομοίωσης και η κίνησή του καθορίζεται από το διάνυσμα: V_d= (V,Θ) όπου: V = μέτρο ταχύτητας, Θ = κατεύθυνση ταχύτητας.

Η ταχύτητα και η κατεύθυνση καθορίζεται τυχαία από τα σύνολα [V_{dmin},V_{dmax}] και [Θ_{min},Θ_{max}] αντίστοιχα. Οι παράμετροι προσομοίωσης αλλάζουν αφού ο κινητός χρήστης διανύσει μια απόσταση d, η οποία αποτελεί τυχαία μεταβλητή από την εκθετική κατανομή με κανονικοποιημένη μέση τιμή, εστιάζοντας σε μια πιο ρεαλιστική κίνηση αφού το κινούμενο αντικείμενο δεν αλλάζει συχνά τις παραμέτρους κίνησης αλλά οι τιμές τους διαρκούν για κάποιο χρονικό διάστημα.

ΑΥΞΗΤΙΚΟ ΜΟΝΤΕΛΟ ΚΙΝΗΣΗΣ

Το αυξητικό μοντέλο κίνησης βασίζεται στην κίνηση Brown, αλλά οι παράμετροι αλλάζουν αυξητικά κάθε Δt δευτερόλεπτα παράγοντας έτσι μια πιο ομαλή κίνηση.



Σχήμα 3.5, 3.6

ΜΟΝΤΕΛΟ ΚΙΝΗΣΗΣ SMOOTH

Το μοντέλο αυτό προσθέτει στην κίνηση Brown συσχέτιση των προηγούμενων τιμών κίνησης τόσο στην τιμή της ταχύτητας όσο και στην

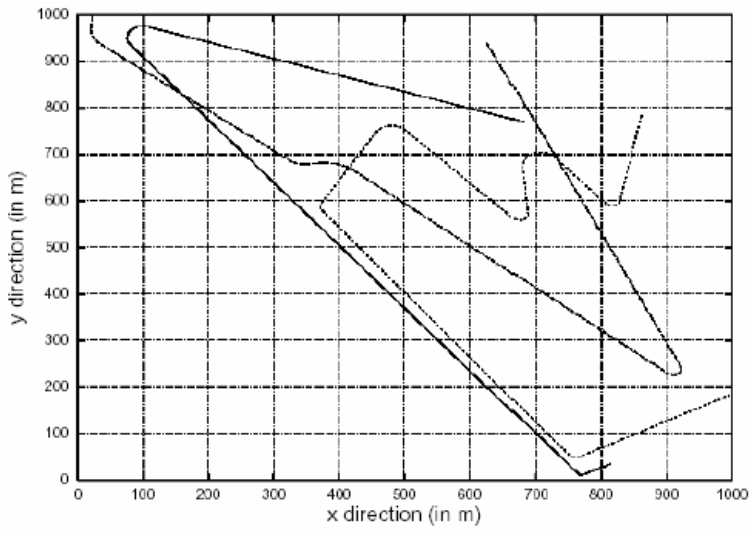
τιμή της κατεύθυνσης. Μια μεταβολή της κατεύθυνσης δεν αντιστοιχεί σε άμεση αλλαγή των τιμών των παραμέτρων απλά ενημερώνεται μια μέση τιμή λειαίνοντας έτσι τις τιμές του ρυθμού αλλαγής. Επίσης η ταχύτητα ανακτά τιμές από ένα καλά ορισμένο διάστημα τιμών που προσομοιώνει περισσότερο τις πραγματικές κινήσεις. Για παράδειγμα μια τυπική ταχύτητα μέσα στην πόλη για ένα κινούμενο τερματικό είναι από 50-60 χλμ/ώρα και για έξω από την πόλη 100-120 χλμ/ώρα. Ο χρηστής κινείται με μια σταθερή ταχύτητα μέχρι ένα γεγονός αλλαγής της τιμής της εμφανιστεί. Η νέα τιμή της ταχύτητας επιλέγονται από το σχήμα 3.7

$$V^* = \begin{cases} V_{pref1} & p(V_{pref1}) \\ V_{pref2} & p(V_{pref2}) \\ \dots & \dots \\ V_{prefn} & p(V_{prefn}) \\ 0 < V < V_{\max} & 1 - p(V_{pref}) \end{cases}, \qquad V_{pref} = \sum_{i=1}^{n} p(V_{prefn}) < 1$$

Σχήμα 3.7

Η ταχύτητα αλλάζει αυξητικά με επιτάχυνση που η τιμή της κατανέμεται κανονικά. Το χρονικό διάστημα μεταξύ δύο τιμών ταχύτητας επιλέγεται σύμφωνα με την εκθετική κατανομή. Ο αλγόριθμος του μοντέλου παράγει μια πιο ρεαλιστική συμπεριφορά της κίνησης του χρήστη σε σχέση με την κίνηση Brown.

Ο χρόνος της αλλαγής της κατεύθυνσης είναι ανεξάρτητος από τον χρόνο αλλαγής της τιμής της ταχύτητας και καθορίζεται από δική του εκθετική κατανομή. Η νέα κατεύθυνση κατανέμεται μέσα στα όρια $0<\varphi<2\pi$. Η κατεύθυνση αλλάζει αυξητικά σε καθορισμένα χρονικά βήματα. Η συνολικός χρόνος αλλαγής κατεύθυνσης ή «χρόνος καμπύλης» αποκτάται από μια κανονική κατανομή κάνοντας πιο ομαλή την κατεύθυνση της κίνησης όπως στο σχήμα 3.8.



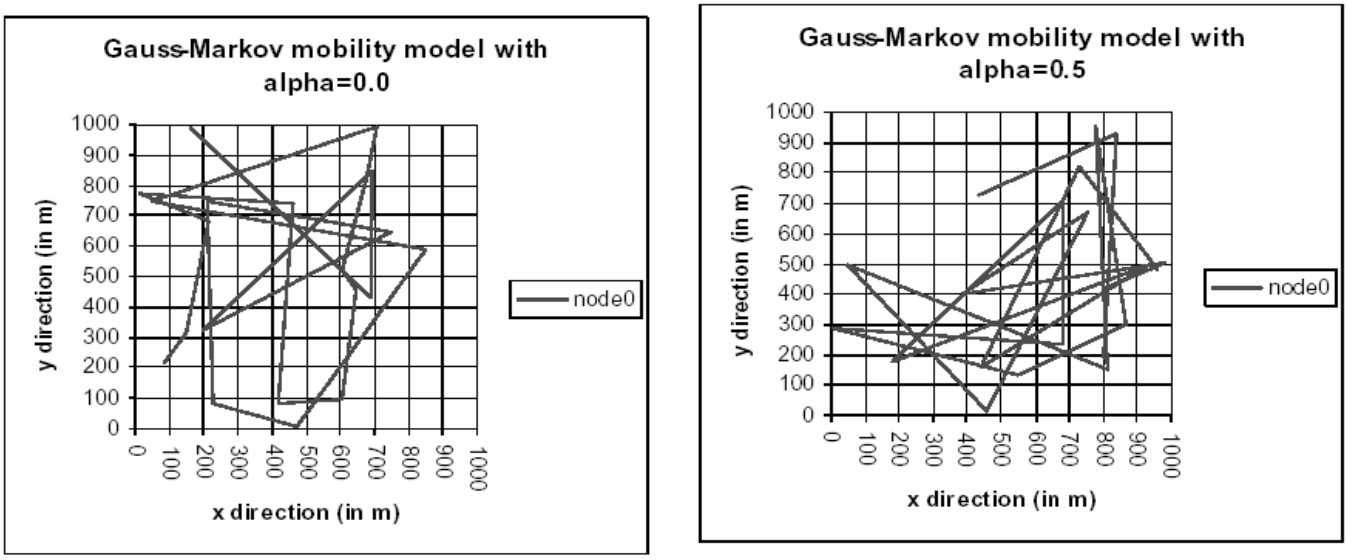
Σχήμα 3.8

ΜΟΝΤΕΛΟ ΚΙΝΗΣΗΣ GAUSS-MARKOV

Το μοντέλο αυτό χρησιμοποιεί και υλοποιεί τις ικανότητες των τυχαίων και των αιτιοκρατικών μοντέλων. Χαρακτηρίζεται από μια μεταβλητή που μπορεί να μεταπίπτει μεταξύ πλήρους τυχαιότητας και πλήρους καθοριστικότητας της κίνησης.

Το μοντέλο εκφράζεται με την σχέση: $V_n = \alpha V_{n-1} + (1-\alpha)\mu + \text{sqrt}(1-\alpha^2)\, x_{n-1}$, όπου: α = παράμετρος μετάπτωσης, V_n = κατεύθυνση κινητού αντικειμένου, x-ανεξάρτητη, μη συσχετιζόμενη κατανομή Gaussian με μέση μηδενική τιμή και προκαθορισμένη απόκλιση σ, μ-ασύμπτωτη μέση τιμή

Για α=0 η εξίσωση περιγράφει την τυχαία κίνηση Brown, για τιμή α=1 περιγράφει την αιτιοκρατική κίνηση, για $\alpha \in (0,1)$ αποτελεί μια υβριδική κίνηση, όπως στο σχήμα 3.9.



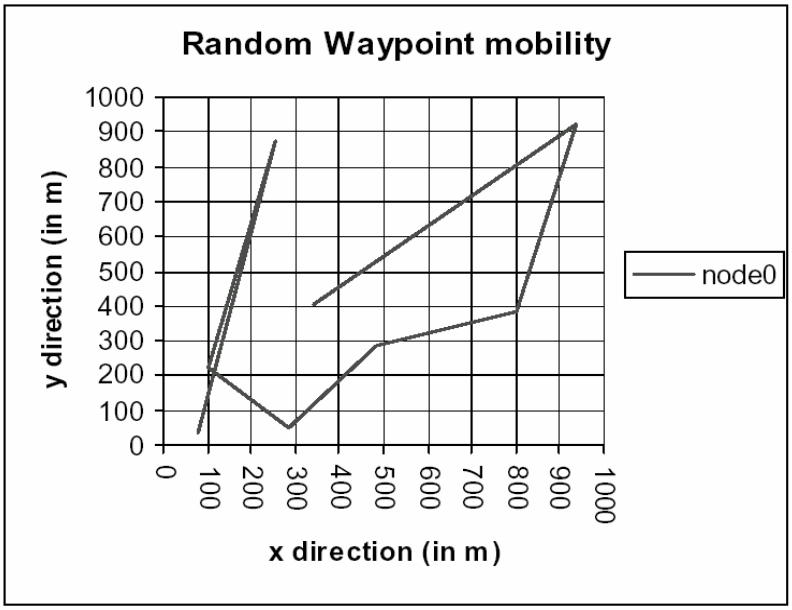
Σχήμα 3.9

Τα μειονεκτήματα του μοντέλου Brown και των μοντέλων που το επεκτείνουν δεν θεωρούν την αίσθηση της κίνησης. Τα αντικείμενα τροποποιούν τις κινητικές παραμέτρους αλλά ένα τελικό προβλεπόμενο σύνολο σημείων κίνησης δεν υπάρχει.

ΜΟΝΤΕΛΟ ΚΙΝΗΣΗΣ ΤΥΧΑΙΑΣ ΜΕΤΑΤΟΠΙΣΗΣ ΣΗΜΕΙΟΥ

Το σημείο προορισμού και η ταχύτητα της κίνησης είναι τα χαρακτηριστικά αυτού του μοντέλου. Το μοντέλο αυτό χρησιμοποιείται για προσομοιώσεις δικτύων κινητών τερματικών και υλοποιούνται από τους προσομοιωτές MANET. Το μοντέλο ορίζεται ως εξής:

Στη αρχή όλοι οι κόμβοι τοποθετούνται σε αρχικές θέσεις, και επιλέγουν το σημείο προορισμού τους μέσα από την περιοχή προσομοίωσης καθώς και επιλέγουν την ταχύτητα κίνησης. Η ταχύτητα παραμένει σταθερή κατά την διάρκεια της κίνησης. Αφού φτάσει στον προορισμό ο κόμβος στέκεται εκεί για συγκεκριμένο χρονικό διάστημα και έπειτα η κίνησή του συνεχίζεται με τον ίδιο τρόπο. Τα παραδείγματα των στιγμάτων θέσεων και ταχύτητας απεικονίζονται στο σχήμα 3.10.



Σχήμα 3.10

Σε μερικά μοντέλα προσομοίωσης η κίνηση ενός χρήστη ορίζει γενικά την κίνηση μιας ομάδας χρηστών. Αυτό αποτελεί βασικό χαρακτηριστικό στην προσομοίωση κινητών τερματικών. Για παράδειγμα η κίνηση ενός ξεναγού σε ένα μουσείο καθορίζει και την κίνηση μιας ομάδας επισκεπτών.

ΔΙΑΝΥΣΜΑΤΙΚΟ ΜΟΝΤΕΛΟ ΚΙΝΗΤΙΚΟΤΗΤΑΣ

Πολλοί ερευνητές χρησιμοποιούν το διανυσματικό μοντέλο κινητικότητας, το οποίο διαθέτει μια δυνατότητα επέκτασης της έκδοσης του μοντέλου BROWN. Το διάνυσμα κίνησης M εκφράζεται ως το διανυσματικό άθροισμα του βασικού διανύσματος κίνησης B και του διανύσματος απόκλισης V: M = B + αV, όπου α είναι ένας παράγοντας επιτάχυνσης. Από κατάλληλες ρυθμίσεις του παράγοντα επιτάχυνσης και την επιλογή των διανυσμάτων βάσεων και απόκλισης, διάφορα μοντέλα μπορούν να περιγραφούν:

1. Μοντέλο βαρύτητας - οι κόμβοι προσπαθούν να είναι πιο στενοί σε ένα ορισμένο σημείο (παραδείγματος χάριν, πιο κοντά στον σταθμό βάσης προκειμένου να αποκτηθεί ένα καλύτερο ράδιο

σήμα). Σε αυτή την περίπτωση το διάνυσμα βάσης είναι κατευθυνόμενο σε αυτό το σημείο.

2. Εξαρτώμενο μοντέλο θέσης - αντιπροσωπεύει ένα συλλογικό πρότυπο κινητικότητας στην περιοχή (για το παράδειγμα, για το δρόμο). Στην περίπτωση αυτή το διάνυσμα βάσης καθορίζεται από την περιοχή
3. Απευθυνόμενος στο μοντέλο - ο κόμβος κινείται σε έναν ορισμένο στόχο και κατά την διάρκεια προσέγγισής του η ταχύτητα μειώνεται στο μηδέν. Στη περίπτωση αυτή το διάνυσμα βάσης είναι το σημείο προορισμού.
4. Ομαδοποίηση του μοντέλου των κινήσεων - η κίνηση μιας ομάδας κόμβων. Στην περίπτωση αυτή το διάνυσμα βάσης είναι η μετακίνηση της ομάδας.

Το διάνυσμα απόκλισης σε αυτά τα μοντέλα αντιπροσωπεύει τη μεμονωμένη συμπεριφορά μετακίνησης του κόμβου (απόκλιση από το διάνυσμα βάσης).

ΜΟΝΤΕΛΟ ΚΙΝΗΤΙΚΟΤΗΤΑΣ ΟΜΑΔΑΣ ΣΗΜΕΙΟΥ ΑΝΑΦΟΡΑΣ

Ένα πολύ παρόμοιο μοντέλο που χρησιμοποιείται είναι το Μοντέλο κινητικότητας ομάδας. Κάθε ομάδα έχει ένα συνδεμένο λογικό σημείο αναφοράς και ένα διάνυσμα κίνησης V. Το διάνυσμα μετακίνησης ενός κόμβου υπολογίζεται ως το ποσό του διανύσματος της ομάδας V και μιας τυχαίας τιμής διανύσματος RM.

Το διάνυσμα RM έχει μήκος, το οποίο διανέμεται εξίσου μέσα σε μια ορισμένη ακτίνα από το κέντρο της ομάδας, και κατεύθυνση, η οποία διανέμεται ομοιόμορφα μέσα στο [0,2π]. Το διάνυσμα, το οποίο η ομάδα θα ακολουθήσει, καθορίζεται ως ένα σύνολο σημείων ελέγχου. Κάθε φορά που το κέντρο της ομάδας φθάνει σε ένα σημείο ελέγχου, το νέο διάνυσμα V και το επόμενο σημείο ελέγχου επιλέγονται και η μετακίνηση συνεχίζεται.

ΜΟΝΤΕΛΟ BOIDS

Διάφοροι συντάκτες εφάρμοσαν το μοντέλο Boids στην προσομοίωση της κίνησης ομάδας. Το Boids ήταν αρχικά δημιουργημένο για να διαμορφώσει τη σύνθετη κίνηση της δυναμικής ομάδας στη ζωολογία (κοπάδια πουλιών, ιχθυοτροφία, κοπάδια ζώων). Οι βιολόγοι έχουν βρει ότι η ομαδοποίηση της συμπεριφοράς των ζώων δημιουργείται από μια έλξη, η οποία διαμορφώνει την επιθυμία κάθε μέλους να προσχωρήσει στην ομάδα και ταυτοχρόνως να διατηρηθεί μια ικανοποιητική απόσταση από τα κοντινά πλάσματα. Υποθέτοντας ότι η μετακίνηση της ομάδας είναι ένα αποτέλεσμα της αλληλεπίδρασης μεταξύ των ενιαίων κόμβων, η μετακίνηση προσομοιώνεται όπως η κίνηση των μεμονωμένων κόμβων ξεχωριστά. Τρία υπό-πρότυπα καθορίζουν το σύστημα:

1. μοντέλο αντίληψης (το πεδίο θέας που έχει ένας κόμβος),
2. μοντέλο συμπεριφοράς (τι ένας κόμβος αποφασίζει στην τρέχουσα κατάστασή του)
3. μοντέλο μετακίνησης (πως ένας κόμβος κινείται πραγματικά, το οποίο είναι βασισμένο στο μοντέλο συμπεριφοράς).

Οι επόμενες συμπεριφορές οδηγούν στην ομαδοποίηση:

1. Ενοποίηση Ομάδας: ο κόμβος προσπαθεί να μείνει κοντά στους κοντινούς συντρόφους – κόμβους
2. Ταίριασμα ταχύτητας: ο κόμβος προσπαθεί να ταιριάξει με την ταχύτητα των πιο κοντινών του συντρόφων – κόμβων.
3. Αποφυγή σύγκρουσης: ο κόμβος προσπαθεί να αποφύγει τις συγκρούσεις με τους κοντινούς του συντρόφους – κόμβους.

Ο συνδυασμός αυτών των συμπεριφορών παράγει την κίνηση. Αυτό το μοντέλο κινητικότητας χρησιμοποιήθηκε στην προσομοίωση σεναρίων σε αίθουσες Τέχνης, σε αίθουσες Γεγονότων - Συναυλιών και σε πεδία Μάχης, στα οποία η διαφορά της ταχύτητας της κίνησης ενός κόμβου εξαρτάται από την συμπεριφορά της κίνησης των διπλανών του κόμβων.

3.2 ΑΙΤΙΟΚΡΑΤΙΚΑ ΜΟΝΤΕΛΑ ΚΙΝΗΤΙΚΟΤΗΤΑΣ

Μια άλλη κλάση πρότυπων μοντέλων είναι η κλάση των αιτιοκρατικών μοντέλων κινητικότητας, η οποία αντιπροσωπεύεται από τα ίχνη της κίνησης, που λήφθηκαν από την πραγματική ζωή χρησιμοποιώντας τις διαφορετικές κοινωνιολογικές μεθόδους, όπως προσωπικά ερωτηματολόγια, ταξίδια κλπ.

Παραδείγματος χάριν, ο δυναμικός αλγόριθμος διαχείρισης θέσης προσομοιώθηκε με πρότυπα από μια βάση δεδομένων, η οποία περιείχε τα αποτελέσματα της έρευνας ενός ταξιδιού, που διευθύνθηκε σε έναν περιφερειακό δήμο του Βατερλό. Χρησιμοποιώντας αυτό το στοιχείο, ο συντάκτης του μοντέλου υποστήριξε ότι τα αποτελέσματα τις εργασίας του απεικόνιζαν επαρκώς την πραγματικότητα.

Επάνω στην έρευνα, οι πληροφορίες για τα ταξίδια ανά ημέρα ελήφθησαν από οικογένειες με μέλη άνω των 5 ετών. Τα στοιχεία ταξιδιού περιείχαν πληροφορίες για το χρόνο της έναρξης και τερματισμού, την προέλευση, τις θέσεις προορισμού και τον σκοπό του ταξιδιού. Οι σκοποί του ταξιδιού ήταν ταξινομημένοι σε: εργασίας, σχετικός με την εργασία, σχολείο, αγορές, διακοπές, προσωπική επιχείρηση, επιστροφής, άλλος.

Οι πληροφορίες καταγράφηκαν στις μήτρες «διάρκειας δραστηριότητας» και «μετάβασης δραστηριότητας». Η μήτρα «διάρκειας δραστηριότητας» χρησιμοποιείται για να καταχωρήσει τις πληροφορίες για ένα ιδιαίτερο ταξίδι, ενώ η μήτρα «μετάβασης δραστηριότητας» εξυπηρετεί για τη μετατροπή μεταξύ των διαφορετικών δραστηριοτήτων κατά τη διάρκεια της διαδικασίας προσομοίωσης.

Το πλεονέκτημα του μοντέλου είναι η κοντινή του συσχέτιση με την πραγματικότητα, αλλά αφετέρου, απαιτεί υψηλή προετοιμασία και

δαπάνες για να διεξαχθεί μια έρευνα προκειμένου να ληφθούν τα στοιχεία πηγής.

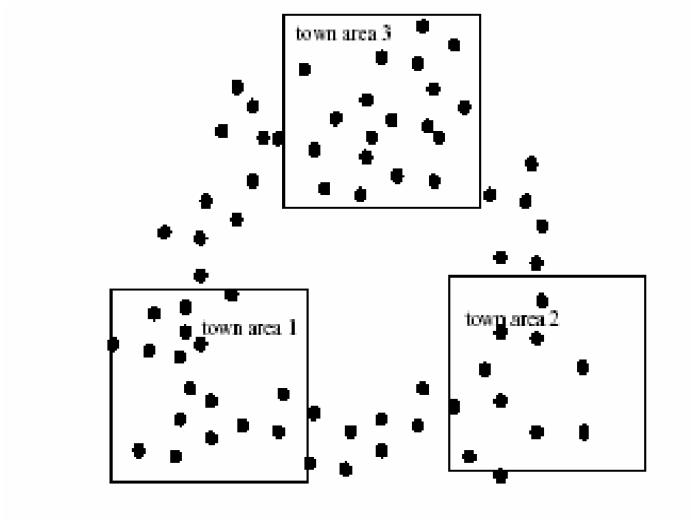
3.3 ΥΒΡΙΔΙΚΑ ΜΟΝΤΕΛΑ ΚΙΝΗΤΙΚΟΤΗΤΑΣ

Τα υβριδικά μοντέλα κινητικότητας συνδυάζουν την τυχαία μετακίνηση με την τακτικότητα της αιτιοκρατικής μετακίνησης.
Ένα μοντέλο χτίζεται μεταξύ δύο επιπέδων:

1. το καθολικό μοντέλο κινητικότητας (GMM)
2. το τοπικό μοντέλο κινητικότητας (LMM)

Το GMM χρησιμοποιείται για να δημιουργήσει intercell (μέσα στην κυψέλη) μετακινήσεις, ενώ LMM χρησιμοποιείται για να διαμορφώσει intracell (μεταξύ των κυψελών) μετακινήσεις. Η παρουσία του καθολικού μοντέλου παρακινείται από το γεγονός ότι οι περισσότεροι από τους κινητούς χρήστες εκθέτουν μερική τακτικότητα στην καθημερινή τους μετακίνηση. Αυτή η τακτικότητα μπορεί να χαρακτηριστεί καλύτερα από διάφορα πρότυπα κινητικότητας χρηστών (UMP), που καταγράφεται σε ένα σχεδιάγραμμα για κάθε χρήστη και που συντάσσεται μέχρι την ώρα του περιστατικού. Η ακρίβεια του UMP προτύπου μειώνεται με μικρές αποκλίσεις με την παραγωγή του πραγματικού μονοπατιού του χρήστη (UAP). Η LMM αντιπροσωπεύει τη μετακίνηση μέσα σε ένα στοιχείο και μπορεί να είναι περιγραμμένο από τρεις μεταβλητές: θέση, ταχύτητα, και κατεύθυνση.

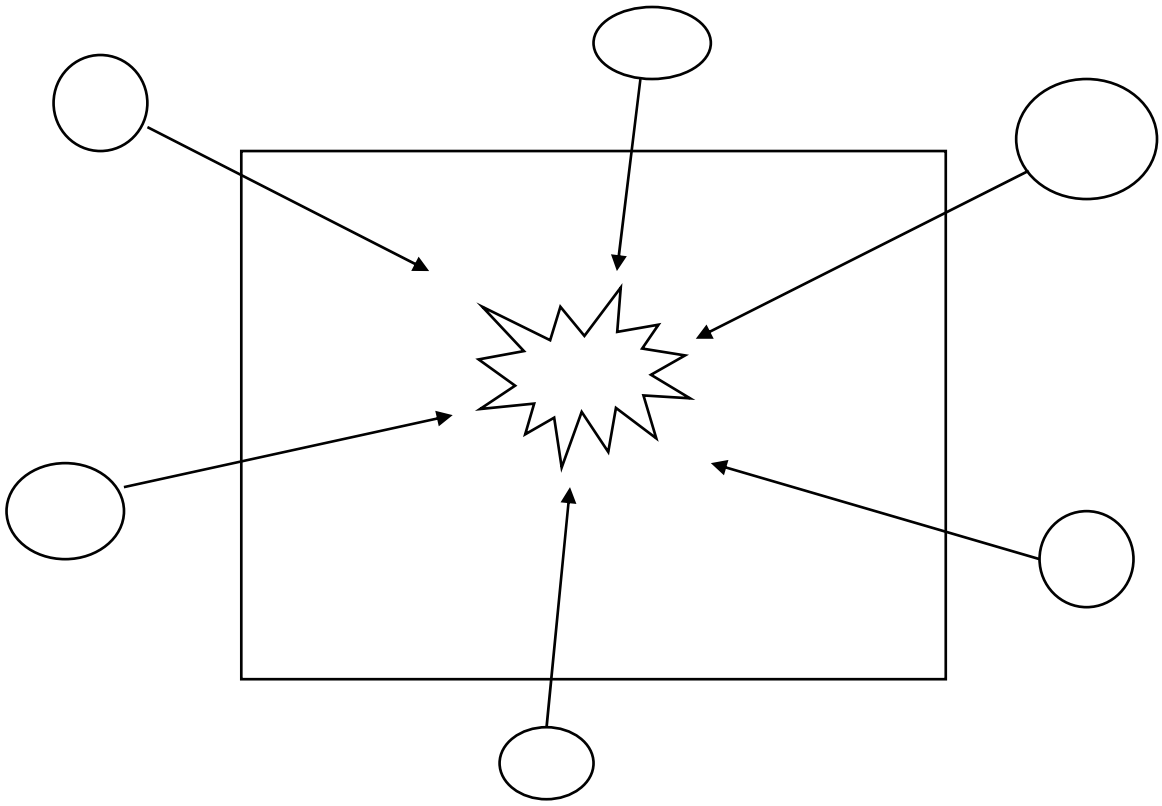
Χαρακτηριστικό μοντέλο τέτοιου τύπου είναι το Τυχαίο Μοντέλο Waypoint Με Εμπόδια. Η περιοχή προσομοίωσης αποτελείται από τρεις μεγάλες περιοχές μετακίνησης (πόλεις) και τρεις εθνικές οδούς, που συνδέουν τις πόλεις. Μέσα στην πόλη ο κόμβος χρησιμοποιεί το τυχαίο Waypoint μοντέλο κινητικότητας, αλλά με ορισμένη πιθανότητα μπορεί η κίνησή του να κατευθυνθεί σε μια άλλη πόλη από μια "εθνική οδό" όπως στο σχήμα 3.11.



Σχήμα 3.11

ΕΝΟΤΗΤΑ 4η

«ΠΛΑΙΣΙΟ ΚΑΙ ΕΠΙΓΝΩΣΗ ΠΛΑΙΣΙΟΥ»



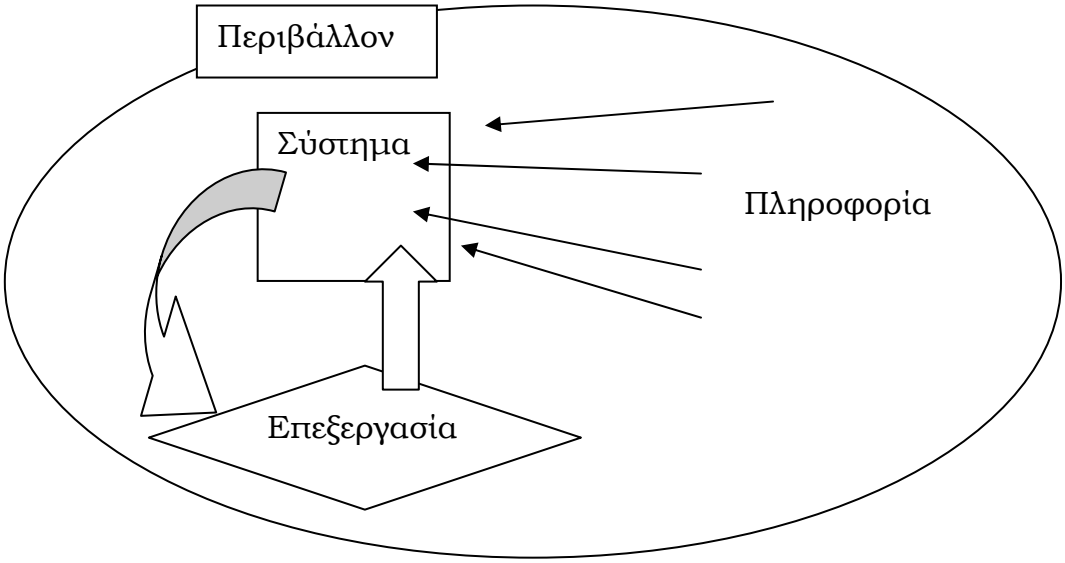
4.1 ΕΙΣΑΓΩΓΗ

Πολλή συζήτηση και επιστημονική έρευνα γίνεται στις μέρες μας γύρω από την έννοια του «πλαισίου» (context) και του πώς θα μπορούσε να αξιοποιηθεί στα δίκτυα του άμεσου μέλλοντος. Παράλληλα με το πλαίσιο - ή «περιρρέουσα κατάσταση» για πολλούς - εμφανίζεται η ανάγκη για τον κατάλληλο προσδιορισμό του σε συσχέτιση με τις ανάγκες κάθε εφαρμογής. Η έννοια είναι εύκολο να παρεξηγηθεί μιας και ως πλαίσιο θα μπορούσε να θεωρηθεί είναι οτιδήποτε βρίσκεται γύρω από μια οντότητα και αλληλεπιδρά μαζί της.

Η ευκολίες που μπορούν να παρασχεθούν σε ένα σύστημα όταν αυτό έχει πλήρη επίγνωση του τι συμβαίνει γύρω του είναι τεράστιες. Η πληροφορία που το επηρεάζει είναι αχανής και εκτείνεται από χωρικές και χρονικές παραμέτρους ως και παράγοντες που δεν είναι πάντα τόσο προφανείς αλλά μπορεί να αποδειχθούν κρίσιμοι. Αν ένα σύστημα μπορέσει να φτάσει στο σημείο να προσαρμόζεται κάθε φορά στο περιβάλλον του κι ακόμη στους ίδιους του τους χρήστες με αξιοπιστία και προνοητικότητα, τότε θα μπορούμε να μιλάμε για ένα πραγματικά εύχρηστο, χρήσιμο, κατά το δυνατόν διαθέσιμο και ανθρωποκεντρικό προϊόν τεχνολογίας.

Μέγιστη σημασία, λοιπόν, δίνεται στην έννοια της «επίγνωσης του πλαισίου» (context awareness). Η επίγνωση πλαισίου θα μπορούσε να οριστεί ως: «το μοντέλο εκείνο στο οποίο μια εφαρμογή θα μπορεί να ανακαλύπτει και να εκμεταλλεύεται την περιρρέουσα πληροφορία (όπως για παράδειγμα την τοποθεσία του χρήστη, την ώρα της ημέρας, τις τριγύρω συσκευές, τη δραστηριότητα του χρήστη κ.α.)». Ένας τέτοιος ορισμός στέκεται πολύ απλά σε δύο κομβικά σημεία. Την ανακάλυψη της πληροφορίας και την κατάλληλη εκμετάλλευσή της προς όφελος του χρήστη αλλά και του ίδιου του συστήματος. Στα πλαίσια της «ανακάλυψης» της πληροφορίας εντάσσεται η σωστή εξαγωγή της από το

περιβάλλον και η όσο το δυνατόν πιο έξυπνη επεξεργασία της για την παραγωγή συμπερασμάτων για το μέλλον (βλ. Σχήμα 4.1).



Σχήμα 4.1

4.2 ΕΠΙΓΝΩΣΗ ΠΛΑΙΣΙΟΥ

Για τον περισσότερο κόσμο η εμπειρία του με την δυνατότητα υπολογισμών (computation) έχει λάβει χώρα κυρίως μέσω των προσωπικών ηλεκτρονικών υπολογιστών (Η/Υ) και, τελευταία, μέσω συσκευών, όπως, για παράδειγμα, τα κινητά τηλέφωνα. Η υπολογιστική, όμως, ικανότητα ενσωματώνεται γοργά σε μια μεγάλη ποικιλία συσκευών και μηχανημάτων. Ειδικοί υπολογιστές χειρός, κινητά τηλέφωνα 3ης γενιάς, αυτοκίνητα με ηλεκτρονικούς εγκεφάλους, συσκευές εντοπισμού τοποθεσίας είναι μόνο μερικά από τα παραδείγματα που μπορεί να αναφέρει κανείς. Όλα αυτά δε θα μπορούσαν να γίνουν εφικτά χωρίς τη χρήση της ασύρματης τεχνολογίας που διασυνδέει τις φυσικές συσκευές με τον ηλεκτρονικό κόσμο. Η διάκριση ανάμεσα στην υπολογιστική και την επικοινωνιακή ικανότητα καθίσταται δυσδιάκριτη και διαπερνά σιγά σιγά κάθε δραστηριότητά μας, προσωπική, επαγγελματική και κοινωνική.

Με μια προσεκτική και δομημένη εισαγωγή της «επίγνωσης πλαισίου» στην υπολογιστική ικανότητα που διαπνέει την καθημερινότητά μας, πολλές δραστηριότητες μπορούν να γίνουν πολύ πιο απλές και αποδοτικές με ελάχιστο κόπο από πλευράς χρήστη. Η δυνατότητα για επικοινωνία μπορεί όχι μόνο να διευκολυνθεί αλλά και να ενσωματωθεί με φυσικό τρόπο στη ζωή μας. Αυτά τα δύο δένονται με συνεχή και διάφανο τρόπο έχοντας ως συνδετικό κρίκο το πλαίσιο, την επίγνωσή του και την έγκαιρη πρόβλεψή του.

Η έννοια του πλαισίου εκτιμάται όλο και πιο πολύ στις μέρες μας. Ο όρος «υπολογιστική ικανότητα με επίγνωση του πλαισίου» (context aware computing) είναι περισσότερο οικείος σε αυτούς που ασχολούνται με «πανταχού παρόντα» και «διεισδυτικά/προνοητικά» (pervasive) δίκτυα και συστήματα, όπου το πλαίσιο θεωρείται κρίσιμο στην προσπάθεια για παροχή αξιόπιστων υπηρεσιών που θα είναι συνυφασμένες με την ανθρώπινη δραστηριότητα.

Ένας πρωταρχικός στόχος είναι η πρόσληψη και εκμετάλλευση της πληροφορίας που σχετίζεται με το πλαίσιο μιας συσκευής έτσι ώστε να παρέχονται υπηρεσίες κατάλληλες σε συγκεκριμένους ανθρώπους, καταστάσεις, χώρο, χρόνο, γεγονότα κλπ. Η πληροφορία αυτή, από την άλλη μεριά είναι αξιοποιήσιμη μόνο όταν μπορεί να ερμηνευθεί με χρήσιμο τρόπο. Επομένως, η πληθώρα πληροφορίας δεν είναι εκ των προτέρων επιθυμητή - το αντίθετο.

Η επίγνωση πλαισίου ακούγεται άψογη στη θεωρία. Και είναι αντικείμενο επιστημονικής έρευνας το πώς θα μπορέσει να μπει πραγματικά σε εφαρμογή. Σημαντικά είναι, για παράδειγμα, τα προβλήματα που σχετίζονται με την αλληλεπίδραση ανθρώπου και υπολογιστή. Μέσω του πλαισίου σχέση αυτή ορίζεται σχεδόν εκ νέου. Πώς θα μπορούσε ο χρήστης να αισθάνεται ασφαλής απέναντι στο τρέχον σύστημά του ενώ ταυτόχρονα το τελευταίο να παίρνει όσο το δυνατόν περισσότερες και όσο

το δυνατόν πιο πρώιμες αποφάσεις; Πώς θα μπορούσε, δηλαδή, ο χρήστης να νιώσει και ωφελημένος αλλά και ασφαλής!

Ας δούμε ένα απλό παράδειγμα όπου μια ποικιλία τεχνολογιών αναζήτησης και επεξεργασίας πληροφορίας θα μπορούσαν να αξιοποιηθούν για να υποβοηθήσουν τη δραστηριότητα χρηστών που έχουν ανάγκη από υπηρεσίες βασισμένες στην τοποθεσία και την κίνησή τους. Ας φανταστούμε μία κινητή συσκευή που επιτρέπει σε αυτόν που τη χρησιμοποιεί να κινείται σε ένα συνεδριακό χώρο από δωμάτιο σε δωμάτιο να έχει πρόσβαση στις πληροφορίες που σχετίζονται άμεσα με το σημείο στο οποίο βρίσκεται (θέμα συζήτησης, ροή παρουσιάσεων κλπ), να κρατά ιστορικό της διαδρομής και των σημείων όπου ο χρήστης έδειξε το μεγαλύτερο ενδιαφέρον (επειδή ίσως στάθηκε εκεί για περισσότερο χρόνο), να τον ενημερώνει για τη δραστηριότητα των συναδέλφων του (τοποθεσία τους, τι θεώρησαν εκείνοι ενδιαφέρον και στάθηκαν να παρατηρήσουν) κ.α.

Γίνεται εύκολα αντιληπτό, λοιπόν, ότι η επίγνωση πλαισίου είναι ένα ιδιαίτερα ευρύ θέμα και γι' αυτό έχει και ευρύ φάσμα θεωρήσεων. Οι περισσότεροι επικεντρώνουν το ενδιαφέρον τους στην αρχιτεκτονική των συστημάτων που θα βασίζονται σε αυτήν και μιλούν για την απαραίτητη υποδομή σε υλικό και τις αντίστοιχες ενδιάμεσες (middleware) υπηρεσίες που θα είναι υπεύθυνες για τη συλλογή, επεξεργασία και προώθηση της πληροφορίας, σε συνδυασμό όχι μόνο με την αλληλεπίδραση με το χρήστη αλλά και με το υπόλοιπο κατανεμημένο σύστημα.

Αντικείμενο έρευνας για πολλούς, ακόμη, είναι το ζήτημα της διερμήνευσης της πληροφορίας πλαισίου. Ένα βασικό θέμα είναι στη συλλογή της πληροφορίας είναι το γεγονός ότι αυτή θα αποδομηθεί στη συνέχεια. Η πληροφορία που θα γίνει τελικά άμεσα σε ένα σύστημα είναι αναγκαστικά πολύ λιγότερο πλούσια σε περιεχόμενο από την πηγή από την οποία προήλθε και καλείται να περιγράψει. Επομένως, πρέπει να

αναπτυχθούν και μηχανισμοί μέσω των οποίων η πληροφορία να ελέγχεται και να διερμηνεύεται σωστά στο χώρο τον οποίο είναι κρίσιμη και με τη μέγιστη δυνατή αποτελεσματικότητα.

4.3 ΣΤΗΝ ΕΦΑΡΜΟΓΗ

Οι σημερινές συσκευές που έχουν ικανότητα επεξεργασίας γίνονται ολοένα και πιο δυνατές τεχνολογικά, διαθέτοντας ενσωματωμένη δυνατότητα αποθήκευσης, αυξανόμενη υπολογιστική ισχύ, δυνατότητα τηλεπικοινωνίας και υποστήριξη πολλών και διαφορετικών εφαρμογών. Η επίγνωση πλαισίου για τέτοιου είδους συσκευές και για τα συστήματα στα οποία θα περικλείονται κρίνεται σημείο κλειδί για τη μελλοντική τους εξέλιξη. Μεγάλη σημασία, επίσης, δίνεται όχι μόνο στην ικανότητα πρόσληψης της πληροφορίας που μπορεί να εξορυχτεί από το πλαίσιο αλλά και στη δυνατότητα πρόβλεψης του ίδιου του πλαισίου σε μελλοντικά χρονικά στάδια, καθιστώντας, έτσι, ικανό το σύστημα να δρα προνοητικά και να προλαμβάνει τις ανάγκες.

Τα συστήματα που θα μπορούσαν να εκμεταλλευτούν τέτοιου είδους πληροφορία είναι είτε καθεαυτό κινητά, όπως φορητοί υπολογιστές (laptops), υπολογιστές χειρός (palmtops), κινητά τηλέφωνα κ.α. είτε σταθερά, όπως καλωδιακές τηλεοράσεις, συστήματα οικιακής ψυχαγωγίας, ακόμη και ολόκληρα δωμάτια ή κτήρια εξοπλισμένα με διαδραστικές συσκευές και ειδικούς ανιχνευτές. Η επίγνωση πλαισίου επιτρέπει σε αυτά να προσαρμόζονται στο περιβάλλον στο οποίο βρίσκονται κάτι που δίνει χώρο για να οραματιστούμε μια πληθώρα από πλεονεκτήματα και δυνατότητες για νέες εφαρμογές. Μερικά από τα πλεονεκτήματα αυτά:

- Ευκολία στη χρήση. Ένα προσαρμοζόμενο σύστημα θα μπορεί να συνεργάζεται πολύ πιο αποτελεσματικά με τους χρήστες του.
- Εξοικονόμηση πόρων. Για παράδειγμα μια έξυπνη κινητή συσκευή θα μπορεί να μειώσει την κατανάλωσή της ανάλογα με το περιβάλλον στο οποίο βρίσκεται.

- Προδραστικότητα (proactiveness). Το σύστημα θα μπορεί να προλάβει τις ανάγκες που θα παρουσιαστούν στο κοντινό μέλλον και κατ' επέκταση να προσαρμοστεί κατάλληλα για να τις αντιμετωπίσει.
- Υπηρεσίες. Ένα σύστημα που μπορεί να βγάζει χρήσιμα συμπεράσματα από το πλαίσιό του μπορεί και να παρέχει πολύ πιο εξειδικευμένες υπηρεσίες στους χρήστες του.

Από την άλλη πλευρά διάφορα ζητήματα αναδύονται όταν η θεωρία οδεύει προς το να γίνει πράξη. Ενδεικτικά:

- Οι συσκευές/συστήματα θα πρέπει να διαθέτουν κατάλληλους ανιχνευτές για την πρόσληψη της αναγκαίας πληροφορίας και από το περιβάλλον αλλά και από τους χρήστες με έξυπνο και διακριτικό τρόπο.
- Η αναγνώριση και πρόβλεψη του πλαισίου είναι κυρίως αναγκαία σε συστήματα με χαμηλές υπολογιστικές ικανότητες, πράγμα το οποίο περιορίζει σημαντικά το σύνολο των αλγορίθμων που μπορούν να χρησιμοποιηθούν.
- Η εκμάθηση και η προσαρμογή από την πλευρά του συστήματος θα έπρεπε να γίνεται σε πραγματικό χρόνο και όχι να προηγείται της κατάστασης την όποία καλείται να διευκολύνει.
- Η αλληλεπίδραση με το χρήστη θα έπρεπε να μείνει σε χαμηλό επίπεδο και οπωσδήποτε να διεκπεραιώνεται όσο το δυνατόν πιο διακριτικά.
- Η πληροφορία που συλλέγεται από το σύστημα και τους ανιχνευτές του είναι σε μεγάλο βαθμό ετερογενής και έτσι πολλές μέθοδοι και αλγόριθμοι που βασίζονται στη στατιστική ανάλυση δε μπορούν να εφαρμοστούν.
- Η πρόβλεψη θα πρέπει να διατηρεί μια καλή ισορροπία ανάμεσα στην πρότερη γνώση που διαθέτει όσον αφορά κάποια μέτρηση (patterns, trends) και στην πιθανότητα αυθαίρετης εμφάνισης νέων στοιχείων, ώστε να μην συμβιβάζεται όσον αφορά την ακρίβειά της.

Έτσι, η επίγνωση πλαισίου είναι πλέον ένα πολύ αναπτυσσόμενο ερευνητικό πεδίο. Πολλές προτάσεις γίνονται καθημερινά σχετικά με την ανάκτηση της πληροφορίας από το περιβάλλον μέσω ανιχνευτών - περίπλοκων ή μη - και κάποια πειραματικά προϊόντα βρίσκονται στην ανάπτυξη. Ταυτόχρονα, μελετώνται τρόποι με τους οποίους η πληροφορία του πλαισίου θα μπορέσει να καταστεί χρήσιμη ή, διαφορετικά, πώς θα μπορούσε να εξαχθεί χρήσιμη πληροφορία από αυτή, τομέας ο οποίος έχει να αντιμετωπίσει και να συγκεράσει τη μεγάλη ποικιλία και ετερογένεια που την διέπει. Τέλος, ένα μεγάλο ζητούμενο είναι η εξισορρόπηση ανάμεσα στην παρεμβολή από πλευράς χρήστη και την προνοητικότητα που οφείλει να δείχνει το σύστημα. Καινοτόμα συστήματα που βασίζονται στο πλαίσιο δε μπορούν να κατασκευαστούν αν δεν εισάγουν στην αρχιτεκτονική τους μηχανισμούς πρόβλεψης του πλαισίου και κατάλληλης εκμετάλλευσης της παραγόμενης πληροφορίας.

4.4 ΤΟ ΠΛΑΙΣΙΟ ΣΤΙΣ ΚΙΝΗΤΕΣ ΣΥΣΚΕΥΕΣ

Η τεχνολογία των κινητών συσκευών με υπολογιστική ικανότητα έχει δώσει ιδιαίτερη ώθηση στη χρήση των υπολογιστών σε ποικίλα και συνεχώς μεταβαλλόμενα περιβάλλοντα. Οι κινητές συσκευές μπορούν να αυξήσουν τις ικανότητές τους αν τους δοθεί η δυνατότητα να λαμβάνουν υπόψη τους το πλαίσιο στο οποίο βρίσκονται και αλληλεπιδρούν. Για να μπορέσουν να προσλάβουν πληροφορία από το πλαίσιο στο οποίο βρίσκονται (περιβάλλον, περιρρέουσα κατάσταση) έχουν την ανάγκη να εξοπλιστούν με ειδικούς αισθητήρες που θα τους επιτρέψουν να έχουν πρόσβαση στην πληροφορία αυτή. Πρόκειται για εξειδικευμένα εξαρτήματα, τα οποία δεν έχουν σαν πρώτιστο στόχο την υπολογιστική ικανότητα αλλά την πρόσληψη των παραμέτρων του περιβάλλοντος στο μοντέλο του υπολογισμού.

Το «πλαίσιο» είναι ουσιαστικά ό,τι μας περιβάλλει και ο όρος αυτός χρησιμοποιείται κυρίως σε σχέση με το φυσικό κόσμο που περιβάλλει μια κινητή συσκευή. Στο τελευταίο γίνεται αναφορά ειδικά σαν «φυσικό πλαίσιο» για να διαχωρίζεται από άλλα είδη πλαισίου όπως οι συνθήκες, τα άλλα συστήματα με τα οποία αλληλεπιδρά, το τηλεπικοινωνιακό δίκτυο κλπ. Ένα ποσοστό ασάφειας υπεισέρχεται όταν ο όρος (πλαίσιο) χρησιμοποιείται σε διαφορετικά επίπεδα αφαίρεσης. Συγκεκριμένα, μπορεί να αναφέρεται:

- Στην πραγματική κατάσταση του κόσμου γύρω που περιβάλλει μια συσκευή/σύστημα.
- Σε μια οπτική γωνία από την οποία αντιλαμβανόμαστε - ή μετράμε - μια κατάσταση.
- Σε ένα συγκεκριμένο στιγμιότυπο κάποιας παραμέτρου, όπως μια τοποθεσία.

Η έννοια της «κατάστασης» για τον μελετητή του πλαισίου είναι αυτό που του παρέχεται από τον φυσικό κόσμο, ενώ «πλαίσιο» είναι αυτό που μπορούμε να εξάγουμε με τους κατάλληλους αισθητήρες και την απαραίτητη επεξεργασία από την «κατάσταση».

Η ανάγκη για εκμετάλλευση της πληροφορίας πλαισίου στις κινητές συσκευές αποκτά ολοένα και αυξανόμενο ενδιαφέρον από διάφορα ερευνητικά πεδία, όπως κινητής επεξεργασία (mobile computing), φορητή υπολογιστική ικανότητα (wearable computing), επαυξημένη πραγματικότητα (augmented reality), πανταχού-παρόντα συστήματα (ubiquitous computing) και επικοινωνία ανθρώπου μηχανής. Μια πρόκληση για κινητοποίηση γύρω από αυτό τον τομέα είναι ότι η επίγνωση πλαισίου μπορεί να εξυπηρετήσει ώστε να γίνει πιο ομαλή η αφαίρεση που απαιτείται έτσι κι αλλιώς για να έχει ένα σύστημα πρόσβαση σε ετερογενή περιβάλλοντα και συνθήκες.

Ακόμη, έχει γίνει κατανοητό ότι το πλαίσιο είναι χρήσιμο σε διαφορετικά επίπεδα μέσα σε μια κινητή συσκευή. Σε επίπεδο μεγάλων συστημάτων

μπορεί να αξιοποιηθεί στη διαχείριση ενέργειας και πόρων. Σε επίπεδο εφαρμογής εξυπηρετεί και τις εφαρμογές με ανάγκη για προσαρμοστικότητα αλλά και τις ίδιες τις εφαρμογές που βασίζονται άμεσα στο πλαίσιο. Στο επίπεδο δε της αλληλεπίδρασης με το χρήστη μπορεί να διευκολύνει στην εξέλιξη προς περισσότερο διαφανείς και όσο το δυνατόν φυσικές διαδικασίες.



Σχήμα 4.2

Το πλαίσιο όπως το αντιλαμβανόμαστε ενστικτωδώς πηγάζει από το φυσικό περιβάλλον μιας συσκευής, ανιχνεύεται από αισθητήρες και καταλήγει σαν προϊόν επεξεργασίας των προσλαμβανομένων δεδομένων. Οι συσκευές μπορούν να έχουν είτε άμεση είτε έμμεση επίγνωση του

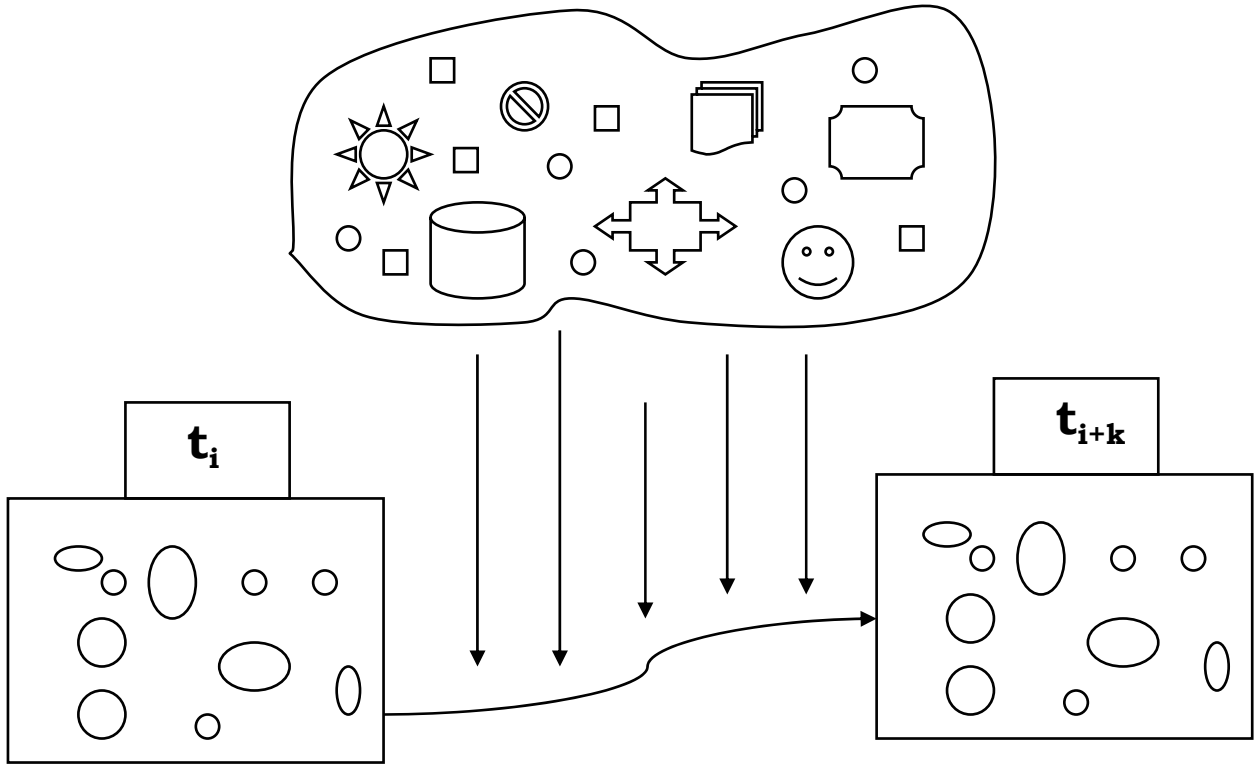
πλαισίου τους. Στη δεύτερη περίπτωση η όλη διαδικασία ανίχνευσης και επεξεργασίας βρίσκεται εγκυψελωμένη στην υποδομή του συστήματος και μια συσκευή έχει πρόσβαση στην πληροφορία πλαισίου μέσω επικοινωνίας με το σύστημα. Αντίθετα, στην πρώτη περίπτωση μια συσκευή μπορεί να έχει άμεση επίγνωση αν έχει την ικανότητα η ίδια να αποκομίζει το πλαίσιο αυτόνομα - ανεξάρτητα από την υποδομή του συστήματος. Για παράδειγμα αν μια διεργασία εξαρτάται από τη θερμοκρασία του περιβάλλοντος στο οποίο λαμβάνει χώρα τότε είναι πολύ πιο αξιόπιστο ή ίδια να μπορεί να την υπολογίζει αυτόνομα από το να την «μαθαίνει» μέσα από τους μηχανισμούς του όλου συστήματος.

Οι κινητές συσκευές που έχουν ενσωματωμένες δυνατότητες άμεσης επίγνωσης πλαισίου διαθέτουν πολλαπλούς αισθητήρες και αλγόριθμους για την επεξεργασία των δεδομένων που λαμβάνουν από το περιβάλλον σε δεδομένα χρήσιμα (αξιοποιήσιμα) για τις λειτουργίες τους. Μεγάλο κομμάτι της έρευνας έχει ασχοληθεί με τη χρήση μεμονωμένων αλλά ισχυρότατων αισθητήρων, όπως ανιχνευτές θέσης και κάμερες. Οι αισθητήρες θέσης δίνουν πρόσβαση σε πληροφορία που σχετίζεται με την τοποθεσία σαν ανεξάρτητο αλλά και ιδιαίτερα χρήσιμο δεδομένο. Συχνά, βέβαια, δεν είναι η τοποθεσία που συνιστά μόνη της άμεσα αξιοποιήσιμο δεδομένο αλλά σε συνδυασμό με επιπρόσθετη πληροφορία που μπορεί να παραχθεί (π.χ. οι πόροι που είναι διαθέσιμοι σε κάποιο σημείο). Οι κάμερες, αντίστοιχα, παρέχουν δυνητικά πολύ πλούσιο υλικό, από το οποίο είναι δυνατόν να εξορυχτεί η απαραίτητη πληροφορία.

Παρόλο που το χωρικό και το οπτικό πλαίσιο μπορούν να αποδειχθούν πολύ ισχυρά στη διευκόλυνση των συσκευών, έχουν κάποια ξεκάθαρα μειονεκτήματα. Η τοποθεσία δεν είναι παρά μια στατική περιγραφή ενός χώρου που δεν αντανακλά τη δυναμική σκοπιά του. Η όραση, από την άλλη, μπορεί να αξιοποιηθεί για την αναγνώριση της κίνηση και άλλων δυναμικών στοιχείων, όμως η εξαγωγή πολύ συγκεκριμένης πληροφορίας είναι υπολογιστικά γιγάντια για τις ικανότητες των κινητών συσκευών.

ΕΝΟΤΗΤΑ 5η

«ΠΡΟΒΛΕΨΗ ΠΛΑΙΣΙΟΥ»



5.1 ΠΡΟΒΛΕΨΗ ΠΛΑΙΣΙΟΥ

Η σύγχρονη βιβλιογραφία μιλά για τα «πανταχού παρόντα» (ubiquitous) συστήματα και την ανάγκη τους να χρησιμοποιήσουν την πληροφορία που μπορούν να εξάγουν από το πλαίσιο. Τα συστήματα αυτά γίνονται ακόμη πιο αποτελεσματικά αν μπορούν να προβλέπουν τις ανάγκες του χρήστη και να δρουν εκ των προτέρων όπου το κρίνουν χρήσιμο.

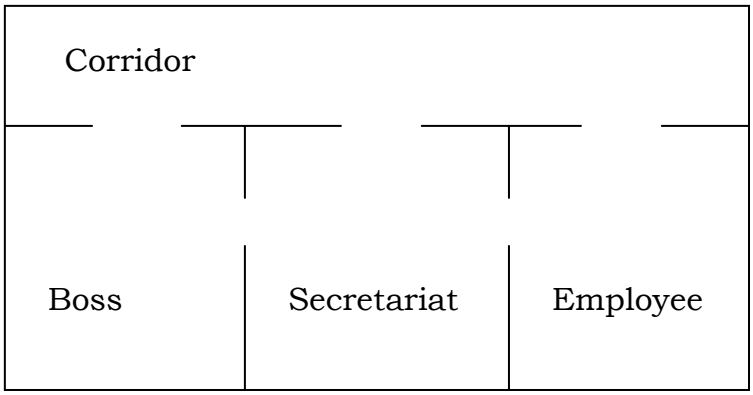
Μια σύντομη περιγραφή ενός «πανταχού παρόντος συστήματος» από την WikipediA είναι η εξής: «Τα συστήματα που δεν θεωρούν τις υπολογιστικές συσκευές σαν ξεχωριστές οντότητες αλλά στοχεύουν στο να ενοποιήσουν την υπολογιστική ικανότητα με το περιβάλλον. Η ενσωμάτωση αυτή - της υπολογιστικής ικανότητας στο περιβάλλον - πιστεύεται ότι θα ανάγει την αλληλεπίδραση του ανθρώπου με τις υπολογιστικές συσκευές σε μια φυσική διαδικασία.». Το πιο κλασσικό παράδειγμα που αναφέρεται είναι εκείνο των συσκευών που υποστηρίζουν το GPS σύστημα και δίνουν κατευθύνσεις στους χρήστες τους (για το GPS θα γίνει λόγος συγκεκριμένα σε επόμενη ενότητα).

5.2 ΠΡΟΒΛΕΨΗ ΜΕΣΩ ΚΑΤΑΣΤΑΣΕΩΝ

Ο άνθρωπος είναι ον που βασίζεται στη συνήθεια. Δρα τις περισσότερες φορές με συγκεκριμένα πρότυπα (patterns) συμπεριφοράς, σπάζοντας τη συνήθειά του μόνο ορισμένες φορές. Υπάρχουν μέθοδοι πρόβλεψης καταστάσεων που βασίζονται σε «πρότυπες δραστηριότητες» και μπορούν να διευκολύνουν τις δραστηριότητες που συμβαίνουν επαναλαμβανόμενα και με όμοιο τρόπο ή ροή πράξεων. Ένα σύστημα που εκμεταλλεύεται τέτοια πρότυπα θα πρέπει να μαθαίνει αυτόματα τις συνήθειες των χρηστών του και να προσαρμόζεται δυναμικά στις αλλαγές τους. Θα πρέπει, επομένως, να στηρίζεται στη γνώση της πρότυπης συμπεριφοράς του παρελθόντος για να προβλέψει συμπεριφορές του μέλλοντος. Αν πάλι η προσπάθεια πρόβλεψης αποτυγχάνει, η αποτυχία

αυτή θα πρέπει να αναγνωρίζεται και να λαμβάνεται υπόψη για τις επόμενες απόπειρες πρόγνωσης.

Μερικές από τις πιο προφανείς τεχνικές στην εκμάθηση προτύπων από τα συστήματα είναι οι αλυσίδες Markov, τα δίκτυα Bayes και τα νευρωνικά δίκτυα. Ωστόσο, η πρόκληση έγκειται στο πώς θα μπορέσουν να μεταφερθούν οι τεχνικές αυτές στην προκείμενη αναγκαιότητα της πληροφορίας πλαισίου. Ένα τέτοιο παράδειγμα είναι το «Adaptive House» του πανεπιστημίου του Colorado, ένα έξυπνο κτήριο που παρατηρεί τον τρόπο ζωής και τη συμπεριφορά των ενοίκων του προσπαθώντας να προσαρμοστεί στις ανάγκες τους. Οι ένοικοι παρακολουθούνται από ειδικούς ανιχνευτές και μια προσέγγιση νευρωνικού δικτύου εφαρμόζεται για να προβλέψει σε ποιο δωμάτιο θα βρεθεί σε επόμενη φάση ο χρήστης και σε τι πράξεις θα προχωρήσει.



Σχήμα 5.1

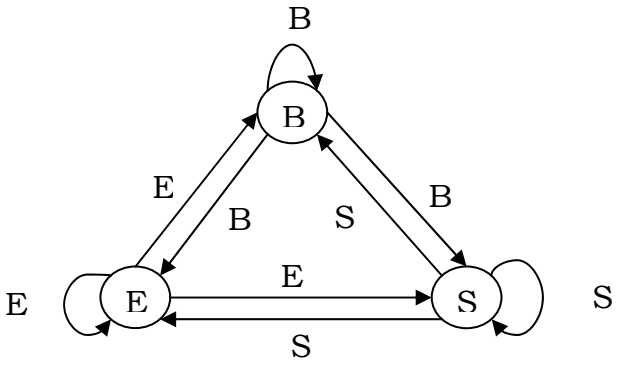
Στη συνέχεια θα παρουσιαστούν αλγόριθμοι/τεχνικές διακλάδωσης που χρησιμοποιούνται σε μικροεπεξεργαστές υψηλών αποδόσεων και μπορούν να μετατραπούν κατάλληλα για τη διαχείριση της επίγνωσης πλαισίου. Συγκεκριμένα επικεντρώνονται σε πρόγνωση καταστάσεων (state prediction) δύο-επιπέδων και δύο-καταστάσεων με τοπικό και καθολικό πρώτου-επιπέδου ιστορικό. Η πρόβλεψη για την οποία γίνεται απόπειρα αφορά στην επόμενη θέση ενός χρήστη - και όχι γενικά στην πρόβλεψη πλαισίου. Σαν παράδειγμα χρησιμοποιείται ένας όροφος κτηρίου όπως φαίνεται στο Σχήμα 5.1, όπου διακρίνονται ο διάδρομος,

το γραφείο του αφεντικού (Boss), της γραμματείας (Secretariat) και του εργαζομένου (Employee).

5.3 ΠΡΟΒΛΕΨΗ ΜΙΑΣ-ΚΑΤΑΣΤΑΣΗΣ

Το σύστημα αποθηκεύει μόνο μία προβλεπόμενη κατάσταση για κάθε γειτονικό δωμάτιο και γι' αυτό ο τίτλος «πρόβλεψη μίας-κατάστασης». Φεύγοντας ο χρήστης από ένα δωμάτιο R, το δωμάτιο προορισμού T αποθηκεύεται και όταν ξαναμπεί στο R τότε το T θα θεωρηθεί ως το επόμενο δωμάτιο στο οποίο προβλέπεται να μεταβεί. Μια τέτοια τεχνική λειτουργεί ανάλογα με τον διακλαδωτή του ενός bit στην περίπτωση όπου υπάρχουν μόνο δύο γειτονικά δωμάτια αλλά όταν αυξηθούν σε τρία χρειάζεται να επεκταθεί.

Στο παράδειγμα του Σχήματος 5.1 ο διάδρομος έχει τρία γειτονικά δωμάτια: του αφεντικού (B), της γραμματείας (S) και του υπαλλήλου (E). Ο γράφος πρόβλεψης για το διάδρομο φαίνεται στο Σχήμα 5.2. Οι κόμβοι δείχνουν πιο δωμάτιο θα προβλεφθεί κάθε φορά. Για παράδειγμα αν κάποιος μετακινηθεί από το διάδρομο στο δωμάτιο του αφεντικού τότε η κατάσταση θα τεθεί σε B. Και όταν ο ίδιος ξαναβρεθεί στο διάδρομο τότε η πρόβλεψη του συστήματος για το επόμενο δωμάτιο θα είναι πάλι το B. Αν η πρόβλεψη είναι επιτυχής τότε η κατάσταση θα παραμείνει B, διαφορετικά θα αλλάξει ανάλογα με το δωμάτιο στο οποίο έγινε η μετάβαση στην πραγματικότητα - π.χ. S, για το δωμάτιο της γραμματείας.



Σχήμα 5.2

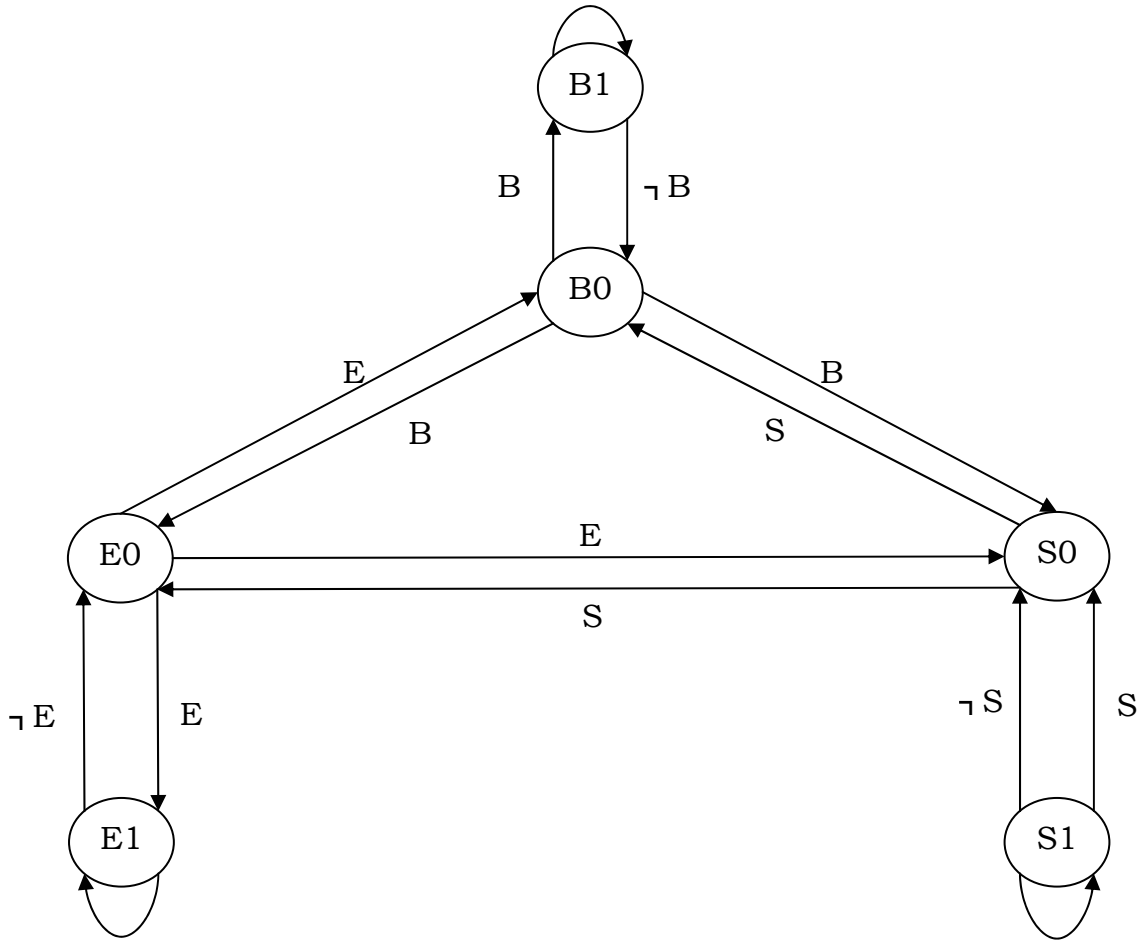
Η αρχή της κατασκευής του γράφου μπορεί να εφαρμοστεί διαδοχικά για αυθαίρετο πλήθος γειτονικών δωματίων όπου κάθε δωμάτιο αναπαρίσταται με ένα κόμβο - στο γράφο - και όλα τα γειτονικά δωμάτια είναι πλήρως αλληλοσυνδεδεμένα. Το αποθηκευτικό κόστος για μια-κατάσταση είναι ιδιαίτερα χαμηλό, μιας και μονάχα μια τιμή χρειάζεται να αποθηκεύεται κάθε φορά (αυτή του επόμενου προβλεπόμενου δωματίου). Το άμεσο πλεονέκτημα που παρέχει είναι η γρήγορη εκμάθηση. Αρκεί ένας χρήστης να μπει μία φορά σε ένα δωμάτια και την επόμενη φορά το σύστημα θα είναι έτοιμο να προβλέψει. Ένα μειονέκτημα είναι το ότι το σύστημα μάλλον μαθαίνει υπερβολικά γρήγορα. Για παράδειγμα αν κάποιος μπαίνει συνήθως από το διάδρομο στη γραμματεία το σύστημα το μαθαίνει αυτό και πράττει ανάλογα. Αν, όμως ο ίδιος σπάσει αυτή του τη συνήθεια έστω και για μια μόνο φορά τότε το σύστημα θα θεωρήσει τη νέα μετάβαση ως ικανή για τις επόμενες προβλέψεις. Η πρόβλεψη μιας-κατάστασης κατάστασης είναι, λοιπόν, πολύ ευαίσθητη στις σπάνιες εναλλαγές της συμπεριφοράς.

5.3 ΠΡΟΒΛΕΨΗ ΔΥΟ-ΚΑΤΑΣΤΑΣΕΩΝ

Η πρόβλεψη δύο-καταστάσεων προέρχεται από την τεχνική πρόβλεψης με διακλάδωση δύο bits με μετρητή κορεσμού. Η πρώτη εγγραφή δηλώνει το επόμενο δωμάτιο, όπως και στην πρόβλεψη μιας-κατάστασης, ενώ η δεύτερη χρησιμοποιείται για τις εναλλαγές ανάμεσα στις ισχυρές και τις ασθενείς καταστάσεις. Με αυτό τον τρόπο το δωμάτιο που καταχωρείται στην πρώτη εγγραφή προβλέπεται πάντα ανεξάρτητα από τη δεύτερη εγγραφή, κάτι που επηρεάζει την ταχύτητα εκμάθησης και ανανέωσης της γνώσης. Ο όρος «πρόβλεψη δύο καταστάσεων» προέρχεται από την ύπαρξη δύο καταστάσεων για κάθε δωμάτιο για το οποίο γίνεται πρόβλεψη.

Στο Σχήμα 5.3 φαίνεται ο γράφος πρόβλεψης ενός δωματίου (βλ. Σχήμα 5.1) με τρία άλλα γειτονικά δωμάτια. Η ονομασία των κόμβων είναι συνδυασμός του ονόματος του δωματίου μαζί με ένα μετρητή. Για

παράδειγμα αν ένας χρήστης μπει για πρώτη φορά στο γραφείο του αφεντικού από το διάδρομο τότε βρίσκεται στην κατάσταση B0 (ασθενής κατάσταση). Αν ο ίδιος χρήστης ξαναμπεί στο διάδρομο τότε το δωμάτιο του αφεντικού προβλέπεται σαν επόμενη κίνηση. Αν η πρόβλεψη αποδειχτεί επιτυχημένη θα βρεθεί σε κατάσταση B1(ισχυρή κατάσταση). Έτσι την επόμενη φορά θα προβλεφθεί πάλι το γραφείο του αφεντικού. Αν, όμως, ο χρήστης διακόψει τη συνήθειά του αυτή και μπει από το διάδρομο στη γραμματεία ή το γραφείο του υπαλλήλου η κατάσταση επαναφέρεται πίσω σε B0 και το γραφείο του αφεντικού θα είναι για ακόμη μία φορά η πρόβλεψη για την επόμενη κίνηση του χρήστη από το διάδρομο. Αν, ακόμη, μεταφερθεί ξανά από το διάδρομο σε κάποιο άλλο δωμάτιο εκτός του αφεντικού (π.χ. γραμματεία) τότε η κατάσταση θα γίνει ασθενής για το νέο δωμάτιο (π.χ. S0, για τη γραμματεία) και αυτό θα θεωρεί σαν προβλεπόμενο στο εξής με την ίδια διαδικασία. Ο γράφος πρόβλεψης των δύο-καταστάσεων φαίνεται στο Σχήμα 5.3



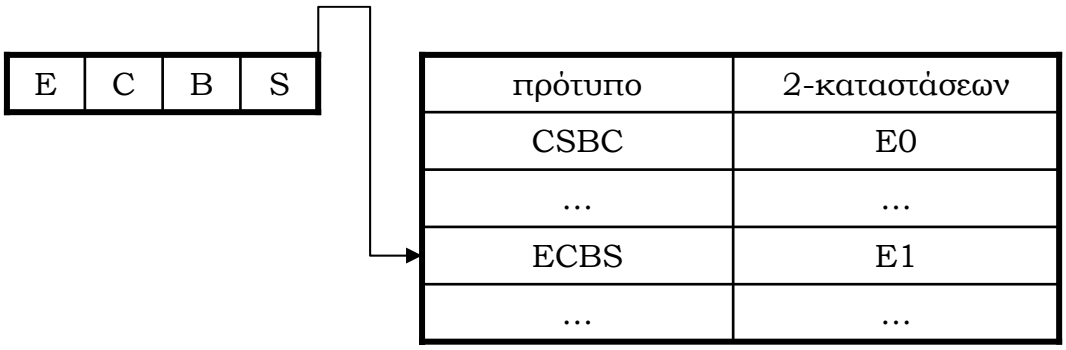
Σχήμα 5.3

Αν ένα δωμάτιο έχει περισσότερα από τρία γειτονικά δωμάτια τότε η ίδια αρχή μπορεί να εφαρμοστεί με παρόμοιο τρόπο. Αξίζει να σημειωθεί ότι μονάχα οι κόμβοι με 0 στη δεύτερη εγγραφή - οι ασθενείς καταστάσεις - σχηματίζουν έναν πλήρως διασυνδεδεμένο γράφο. Το κόστος αποθήκευσης για την πρόβλεψη δύο καταστάσεων είναι επίσης χαμηλό. Πέρα από το διακριτικό που ξεχωρίζει το κάθε δωμάτιο από τα υπόλοιπα χρειάζεται και ένας μετρητής ενός bit. Το υπολογιστικό κόστος σε σύγκριση με την πρόβλεψη μιας-κατάστασης είναι ελάχιστα υψηλότερο και το σύστημα εκπαιδεύεται ταχύτατα. Η ανανέωση της γνώσης καθυστερεί λίγο παραπάνω αλλά μια μικρή αλλαγή στις συνήθειες του χρήστη δεν προκαλεί επιπτώσεις. Στην περίπτωση όπου υπάρχει ανάγκη για περισσότερες των δύο διαδοχικές αποκλίσεις από την εγνωσμένη συμπεριφορά πριν το σύστημα επανέλθει σε ασθενή κατάσταση μπορεί κανείς απλά να αυξήσει τον αριθμό των καταστάσεων - όχι μόνο ασθενής και ισχυρή, δηλαδή - δημιουργώντας σύστημα πρόβλεψης k-καταστάσεων.

5.4 ΚΑΘΟΛΙΚΗ ΠΡΟΒΛΕΨΗ ΔΥΟ-ΕΠΙΠΕΔΩΝ

Η καθολική πρόβλεψη δύο-επιπέδων θεωρεί μια ακολουθία που αποτελείται από την καταγραφή τω τελευταίων δωματίων που διέσχισε ο χρήστης ώστε, βασισμένη σε αυτή, να προβλέψει την επόμενη μετάβαση. Τα δωμάτια της ακολουθίας αποθηκεύονται σε ένα είδος καταχωρητή μετατόπισης ο οποίος συνιστά το πρώτο από τα δύο επίπεδα του μηχανισμού. Αν ο χρήστης μπει σε ένα νέο δωμάτιο τότε όλες οι εγγραφές του καταχωρητή ολισθαίνουν μια θέση προς τα αριστερά και το νέο δωμάτιο προστίθεται στην κενή θέση που δημιουργήθηκε δεξιά. Το μέγεθος του καταχωρητή καλείται «τάξη» (order) και είναι ενδεικτικό του μεγέθους του ιστορικού της κίνησης - του χρήστη - που έχει επιρροή πάνω στις μελλοντικές προβλέψεις.

Το δεύτερο επίπεδο αποτελείται από ένα πίνακα ιστορικού προτύπων που διατηρεί όλα τα πιθανά πρότυπα διαδοχικών μεταβάσεων για τα δωμάτια. Κάθε εγγραφή του πίνακα κρατά επιπλέον μια υποεγγραφή πρόβλεψης δύο-καταστάσεων, η οποία θα μπορούσε να αντικατασταθεί με k-καταστάσεων ή μια άλλη στατιστική παράμετρο. Το πρότυπο στον καταχωρητή μετατόπισης χρησιμεύει στην επιλογή της κατάλληλης εγγραφής από τον πίνακα του ιστορικού των προτύπων. Ένα τμήμα ενός τέτοιου πίνακας φαίνεται στο Σχήμα 5.4.



Σχήμα 5.4

Ας θεωρήσουμε το ίδιο παράδειγμα σαν υπόθεση εργασίας (C=διάδρομος, B=αφεντικό, S=γραμματεία, E=υπάλληλος). Ταυτόχρονα ας επιλέξουμε τάξη 4. Έτσι θα υπάρχουν $4x3^3=108$ πρότυπα και ισάριθμες εγγραφές στον πίνακα. Μια πιθανή ακολουθία που θα μπορούσε να οδηγήσει στο στιγμιότυπο του Σχήματος 5.4 είναι η C S B C E C B S E S E C B S E C B S. Στην προκειμένη περίπτωση η πρόβλεψη είναι ότι το δωμάτιο E θα ακολουθήσει.

Το μέγεθος του k στη δεύτερη στήλη του πίνακα μπορεί να τροποποιηθεί ανάλογα με τις ανάγκες για να προσαρμόσει διαφορετικές ταχύτητες ανανέωσης της εκμάθησης του συστήματος ή να αντικατασταθεί με τις συχνότητες προσπέλασης όλων των γειτονικών δωματίων δεδομένου του τρέχοντος.

Ένα πλεονέκτημα αυτής της τεχνικής είναι ότι πιο περίπλοκα πρότυπα (patterns) κίνησης μπορούν να υποστηριχθούν. Επίσης, μιας και κάθε

πρότυπο είναι διαχειρίσιμο ανεξάρτητα από τα υπόλοιπα, οι παρεμβολές ανάμεσα σε δύο πρότυπα δεν πρόκειται να εμφανιστούν, όπως στις τεχνικές διακλάδωσης.

Ένα μειονέκτημα, από την άλλη μεριά, είναι ότι μετά από ένα μεγάλο αριθμό εμφανίσεων ενός προτύπου η διαδικασία ανανέωσης της γνώσης απαιτεί μεγάλο χρόνο. Για παράδειγμα, αν το δωμάτιο R προσπελαύνεται 1000 φορές ακολουθώντας το ίδιο πρότυπο μεταβάσεων τότε χρειάζονται άλλες 1000 φορές προσπέλασης ενός άλλου δωματίου που να ακολουθεί το ίδιο πρότυπο ώστε η προβλεπόμενη μετάβαση να αλλάξει.

5.4 ΤΟΠΙΚΗ ΠΡΟΒΛΕΨΗ ΔΥΟ-ΕΠΙΠΕΔΩΝ

Η τοπική πρόβλεψη δύο επιπέδων χρησιμοποιεί ιστορικό μόνο γειτονικών δωματίων και αδιαφορεί για τις αλληλουχίες μετακινήσεων στα υπόλοιπα δωμάτια. Εδώ ο καταχωρητής μετατόπισης δεν περιέχει όλο το ιστορικό (καθολικό)αλλά μόνο το τοπικό - δηλαδή, τα δωμάτια που ένας χρήστης έχει επισκεφτεί αμέσως μετά από το τρέχον. Για κάθε δωμάτιο δημιουργείται μία πρόβλεψη δύο-επιπέδων κι επομένως τα δωμάτια που διαδέχονται το ένα αμέσως μετά το άλλο σχηματίζουν το πρότυπο. Η τοπική δύο-επιπέδων πρόβλεψη λειτουργεί παρόμοια με την αντίστοιχη καθολική. Κι εδώ διαφορετικές μέθοδοι πρόβλεψης μπορούν να αξιοποιηθούν στο δεύτερο-επίπεδο (π.χ. δύο-καταστάσεων, k-καταστάσεων, στατιστική).

Για άλλη μια φορά κοιτώντας στο Σχήμα 5.1 και το παράδειγμα του διαδρόμου με τα τρία δωμάτια υποθέτουμε ότι ένας χρήστης τα διασχίζει με την ακόλουθη σειρά: C B C S B C B E C S C. Ο χρήστης αυτός βρίσκεται αυτή τη στιγμή στο διάδρομο και ο καταχωρητής μετατόπισης που χρησιμοποιείται για την επιλογή του προτύπου από τον πίνακα με τα ιστορικά περιέχει τα: B S B S.

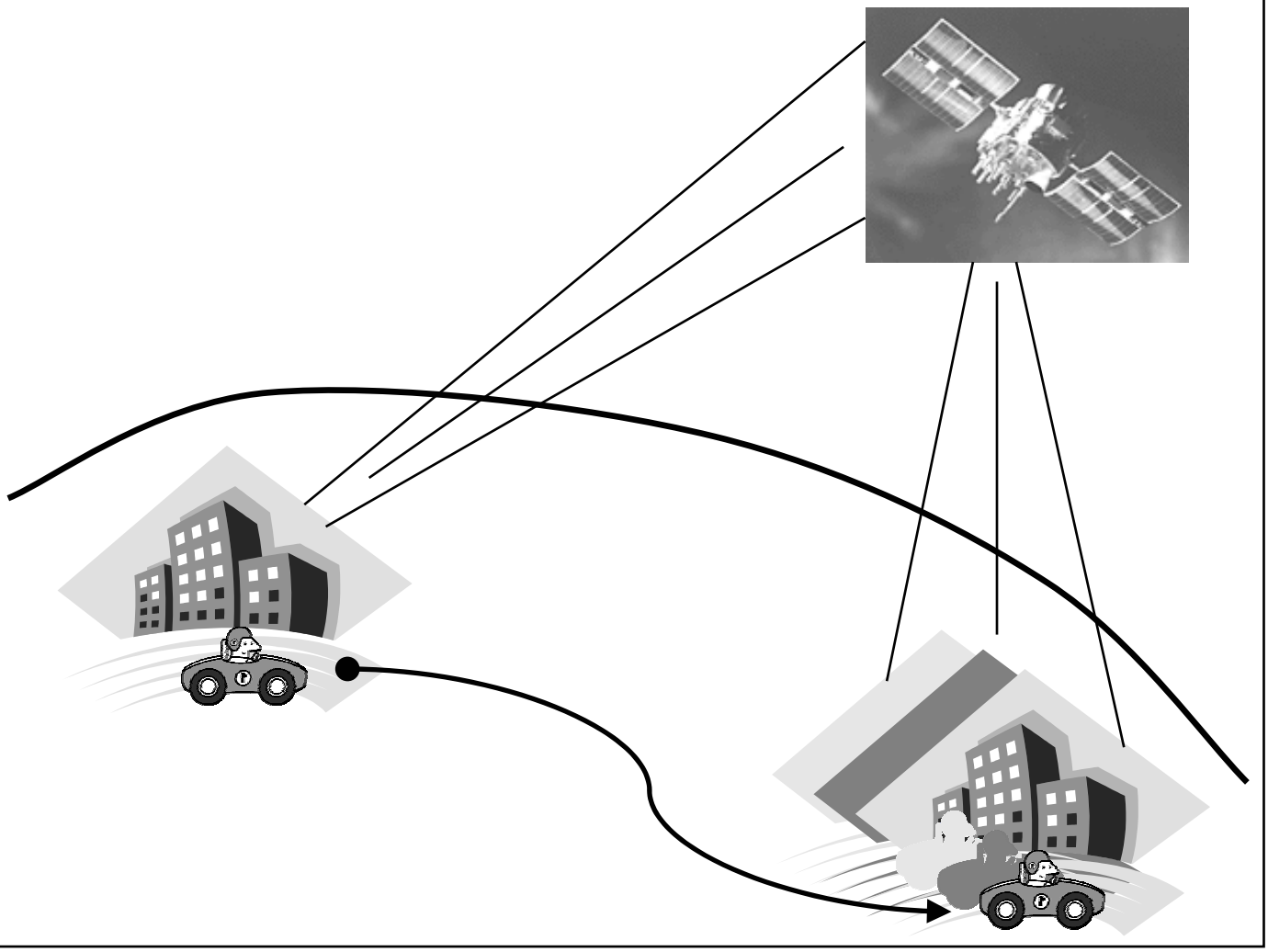
Το αποθηκευτικό κόστος έχει ως ακολούθως: Αν r είναι η τάξη και n_s ο αριθμός των γειτονικών δωματίων ενός δωματίου s, τότε ένας πίνακας με $(n_s)^r$ πρότυπα χρειάζεται να αποθηκευτεί για κάθε δωμάτιο s. Το ίδιο δωμάτιο υπάρχει η πιθανότητα να εμφανιστεί πολλαπλά σε ένα πρότυπο σε αντίθεση με την καθολική πρόβλεψη δύο-επιπέδων που περιγράφηκε παραπάνω. Ένα πλεονέκτημα απέναντί της είναι με σχετικά μικρή τάξη το σύστημα μπορεί να δικαιολογήσει μεγαλύτερες καθολικές αλληλουχίες μεταβάσεων.

Παρόλαυτα, η φάση της εκμάθησης είναι ιδιαίτερα μεγάλη, ενώ πιο γενικές ακολουθίες κινήσεων δεν είναι δυνατό να καταστούν πάντα διακριτές από το σύστημα. Για παράδειγμα η ακολουθία C B C S B C B E C S C και η C B C S E B E C B E C S C παράγουν το ίδιο πρότυπο - το B S B S - τάξης 4, μιας και μόνο τα δωμάτια που προσπελαύνονται μετά το διάδρομο C συνυπολογίζονται για να σχηματίσουν το τοπικό πρότυπο. Έτσι, από καθολική σκοπιά, παρεμβολές ανάμεσα στις καταστάσεις μπορούν να εμφανιστούν κατά την κατασκευή των προτύπων.

Τέλος, μπορούμε να αναφέρουμε πως τα πλεονεκτήματα πολλών συστημάτων πρόβλεψης θα μπορούσαν να συνδυαστούν προς την κατεύθυνση της δημιουργίας ενός υβριδικού συστήματος. Για παράδειγμα, η πρόβλεψη δύο-καταστάσεων μπορεί να χρησιμοποιηθεί στη διαδικασία εκμάθησης μιας πρόβλεψης δύο-επιπέδων. Ένας τέτοιος συνδυασμός θα μπορούσε να είναι άμεσα χρήσιμος μιας και παρέχει άμεση προσαρμοστικότητα.

ΕΝΟΤΗΤΑ 6η

«ΕΦΑΡΜΟΓΗ ΣΤΟ ΧΩΡΙΚΟ ΠΛΑΙΣΙΟ»



6.1 ΕΙΣΑΓΩΓΗ

Στην παρούσα εργασία θα προσπαθήσουμε να εφαρμόσουμε μερικές νέες τακτικές πρόβλεψης πληροφορίας πλαισίου και να μελετήσουμε τη συμπεριφορά τους επικεντρωμένοι στο χώρο (χωρικό πλαίσιο - spatial context). Αντίθετα με πολλές μεθόδους που έχουν προταθεί ως σήμερα και χρησιμοποιούν στατιστική ή/και πιθανοθεωρητική γνώση ώστε να επιτύχουν πρόβλεψη, εφαρμόζουμε γνωστές μαθηματικές μεθόδους.

Η κεντρική φιλοσοφία πίσω από μια τέτοια προσέγγιση είναι το γεγονός ότι για πολλά συστήματα είναι πολύ πιο χρήσιμο να γνωρίζουν την πορεία ενός μεγέθους - παραμέτρου του πλαισίου τους - με το σφάλμα πάντοτε που υπεισέρχεται στην πρόβλεψη που κάνουν, παρά να γνωρίζουν την πιθανή ή/και στατιστική πορεία του. Ίσως, δηλαδή, να είναι πολύ πιο χρήσιμο να έχει ένα σύστημα τη γνώση ότι το μέγεθος Α έχει τιμή $A_i \pm e\%$, όπου e=μέσο ποσοστό απόκλισης, από το να γνωρίζουν ότι το μέγεθος Α θα ακολουθήσει την πορεία A_i με πιθανότητα p_i ή βάσει του προτύπου του ιστορικού του που ταιριάζει στην τρέχουσα πραγματικότητά του.

Για το σκοπό αυτό επιλέξαμε να μελετήσουμε τις εξής μαθηματικές μεθόδους παρεμβολής: Newton Διηρημένων Διαφορών, Lagrange και Cubic Splines, για τις οποίες θα γίνει ιδιαίτερος λόγος παρακάτω. Στην πραγματικότητα η έννοια της παρεμβολής (interpolation) χρειάστηκε να επεκταθεί για τις συγκεκριμένες ανάγκες, ώστε να έχει τη δυνατότητα να υπολογίζει προσεγγίσεις μεγεθών έξω από τα στενά όρια των μετρήσεων που δέχεται σαν είσοδο (extrapolation). Η τακτική που επιλέξαμε για τη δοκιμή των μεθόδων παρεμβολής στην προσπάθεια πρόβλεψης πληροφορίας πλαισίου αναπτύχθηκε με όσο το δυνατόν πιο γενικό τρόπο έτσι ώστε να αποτελεί μια βάση για προβλέψεις και να επιδέχεται επεκτάσεις και προσθήκες όσον αφορά τις ίδιες τις μεθόδους πρόβλεψης χρησιμοποιεί.

Στη συνέχεια του κεφαλαίου αυτού θα παρουσιαστεί αναλυτικά η διαδικασία που ακολουθήθηκε, από τη μελέτη των μεθόδων παρεμβολής και το είδος της εισόδου που δέχεται στο χωρικό πλαίσιο ένα σύστημα έως και την ανάλυση και υλοποίηση της προσομοίωσης που χρειάστηκε να γίνει ώστε η μελέτη να είναι πλήρης και να παράγει αξιόλογα συμπεράσματα.

6.2 ΜΕΘΟΔΟΙ ΠΑΡΕΜΒΟΛΗΣ

Πολύ συχνά τα δεδομένα που παίρνουμε από κάποια μέτρηση έχουν διακριτές τιμές παρόλο που αναφέρονται σε συνεχή, κατά τα άλλα, μεγέθη. Είναι πολλές φορές αναγκαίο να υπολογίζουμε εκτιμήσεις για τις τιμές των μεγεθών που μας ενδιαφέρουν ανάμεσα στις διακριτές εκείνες τιμές τους που γνωρίζουμε. Αν, δηλαδή, γνωρίζουμε τις τιμές του μεγέθους Α, με

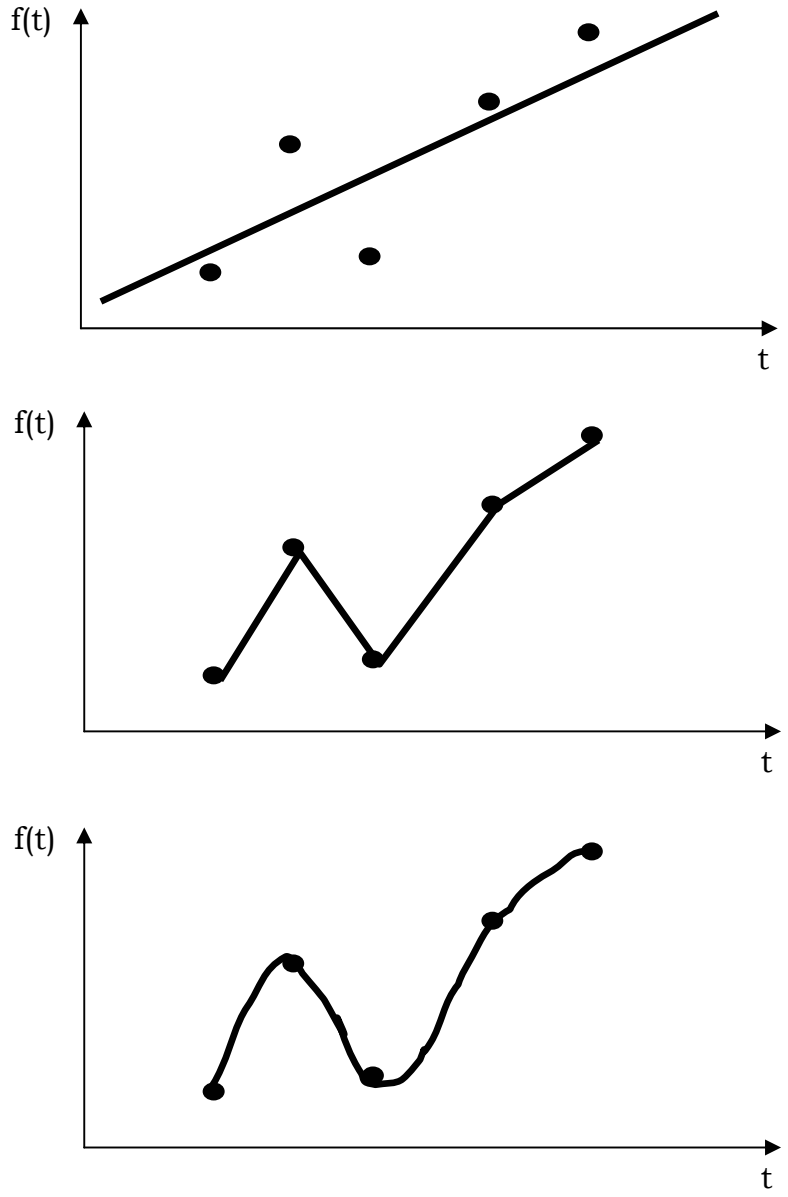
- $A=f(t)$, για $t=t_0,t_1,\ldots,t_n$,
- και $n \in N$

θέλουμε να έχουμε μια όσο το δυνατόν πιο ακριβή εκτίμηση για τις τιμές του Α για $t \neq t_i$, με $i=0,1,\ldots n$. Γνωρίζοντας, λοιπόν, τα $A(t_i)$ να μπορούμε να υπολογίσουμε όλα τα υπόλοιπα $A(t_j)$ στο εύρος τιμών από t_0 έως t_n. Αυτό μεταφράζεται αλγεβρικά στην προσπάθεια υπολογισμού μιας συνάρτησης $A'=f(t)$ η οποία για $t \in (t_0,t_n)$ να προσεγγίζει όσο το δυνατόν καλύτερα την $A=f(t)$.

Τα σημεία $t_i=t_0,t_1,\ldots,t_n$ καλούνται σημεία παρεμβολής και η Α' καλείται συνάρτηση παρεμβολής που αντιστοιχεί σε μια καμπύλη παρεμβολής.

Υπάρχουν δύο γενικές προσεγγίσεις σε αυτή την προσπάθεια που διαχωρίζονται με βάση το ποσοστό του σφάλματος που σχετίζεται με τα δεδομένα. Η πρώτη, όπου τα δεδομένα παρουσιάζουν σημαντικό βαθμό απόκλισης ή «θορύβου», βασίζεται στην παραγωγή μιας καμπύλης που να προσπαθεί να αποδώσει τη γενική τάση του μεγέθους που μελετάται. Η καμπύλη αυτή εκ φύσεως τείνει να ακολουθήσει το «πρότυπο» των

σημείων που μας είναι γνωστά και να τα θεωρήσει σα μια ομάδα. Η δεύτερη, όπου τα δεδομένα μας είναι γνωστά με ακρίβεια, προσπαθεί να παράγει μια καμπύλη που να διατρέχει όλα τα σημεία παρεμβολής ένα προς ένα. Στο Σχήμα 6.1 φαίνονται τρία παραδείγματα παρεμβολής. Το πρώτο ανήκει στην πρώτη κατηγορία μεθόδων παρεμβολής και τα άλλα δύο στη δεύτερη.



Σχήμα 6.1

Η επιλογή της κατηγορίας και της συγκεκριμένης μεθόδου που θα εφαρμοστεί κάθε φορά εξαρτάται ιδιαίτερα από τη φύση των δεδομένων εισόδου - σημεία παρεμβολής - και πρέπει να εξετάζεται κατά

περίπτωση. Στην παρούσα εργασία θα αποπειραθούμε να εφαρμόσουμε μεθόδους που εμπίπτουν και στις δύο κατηγορίες αλλά για να εκτιμήσουμε προσεγγίσεις μεγεθών έξω από τα όρια του πεδίου (t_0, t_n) και μάλιστα για $t=t_{n+k}$, $k>0$. Συγκεκριμένα, θα εφαρμόσουμε τις εξής μεθόδους της δεύτερης κατηγορίας:

- Newton Διηρημένων Διαφορών (Divided Differences)
- Παρεμβολή Lagrange (Lagrange Interpolation)

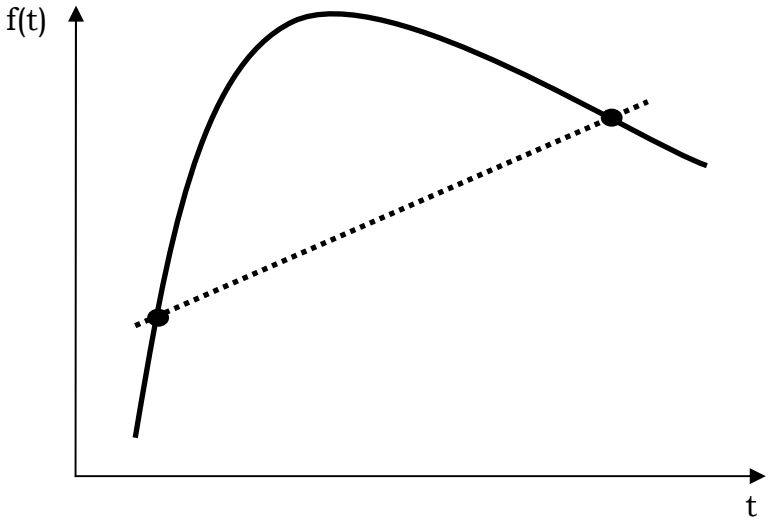
Και από την πρώτη κατηγορία:

- Κυβικές (cubic) Splines

6.3 ΠΑΡΕΜΒΟΛΗ NEWTON ΔΙΗΡΗΜΕΝΩΝ ΔΙΑΦΟΡΩΝ

Η παρεμβολή Newton διηρημένων διαφορών αποτελεί μια από τις πιο δημοφιλείς και χρήσιμες επιλογές. Ας την δούμε σταδιακά:

- **Γραμμική παρεμβολή:** ο πιο απλός τρόπος είναι να συνδέσει κανείς δύο σημεία με μία ευθεία γραμμή και για περισσότερα σημεία να το επαναλάβει αυτό για όλα τα σημεία ανά δύο. Στο Σχήμα 6.2 μας είναι γνωστά 2 σημεία της καμπύλης (με μαύρες κουκίδες) και την προσεγγίζουμε ενώνοντάς τα με ευθείες γραμμές (γραμμική παρεμβολή).

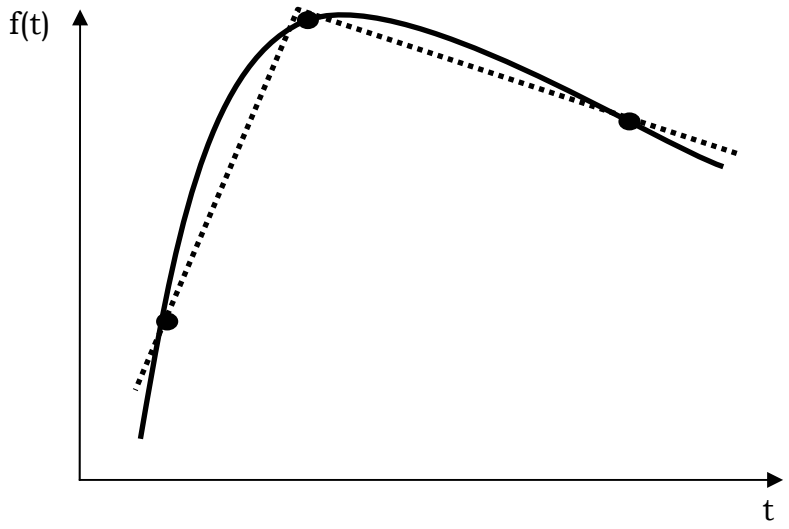


Σχήμα 6.2

Η συνάρτηση που δίνει την γραμμική παρεμβολή για την f είναι:

$$f_1(x) = f(x_0) + \frac{f(x_1) - f(x_0)}{x_1 - x_0} + (x - x_0)$$

Στο Σχήμα 6.3 φαίνεται η ίδια διαδικασία για τρία σημεία.



Σχήμα 6.3

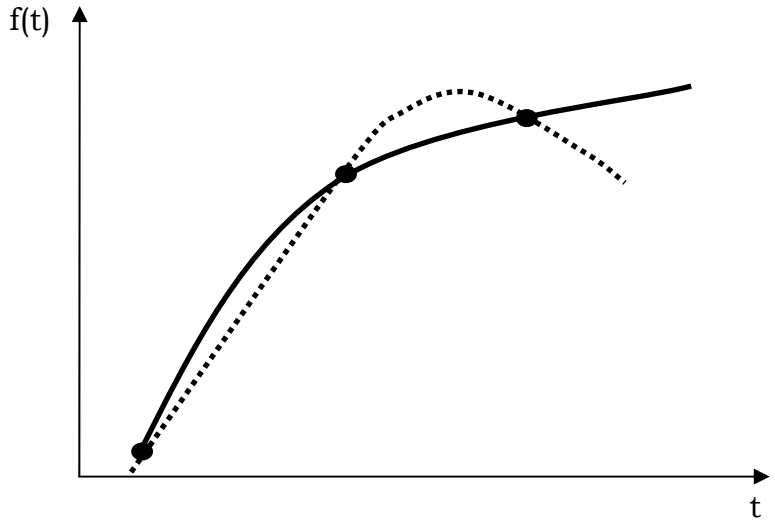
- **Τετραγωνική παρεμβολή:** χρειάζεται τουλάχιστον τρία σημεία για να επιτευχθεί και εξομαλύνει την γραμμική παρεμβολή ώστε να προσαρμόζεται καλύτερα στην πραγματική καμπύλη, όπως φαίνεται στο Σχήμα 6.4. Η συνάρτηση που δίνει την τετραγωνική παρεμβολή για την f είναι:

$$f_2(x) = b_0 + b_1(x - x_0) + b_2(x - x_0)(x - x_1)\text{, όπου}$$

$$b_0 = f(x_0)$$

$$b_1 = \frac{f(x_1) - f(x_0)}{x_1 - x_0}$$

$$b_2 = \frac{\dfrac{f(x_2) - f(x_1)}{x_2 - x_1} - \dfrac{f(x_1) - f(x_0)}{x_1 - x_0}}{x_2 - x_0}$$



Σχήμα 6.4

- **Γενική μορφή:** η γενική μορφή της παρεμβολής Newton με διηρημένες διαφορές είναι η εξής:

$f_n(x) = b_0 + b_1(x - x_0) + \ldots + b_n(x - x_0)(x - x_1)\ldots(x - x_{n-1})$, όπου

$$b_0 = f(x_0)$$

$$b_1 = f[x_1, x_0],$$

$$b_2 = f[x_2, x_1, x_0]$$

.

.

.

$$b_n = f[x_n, x_{n-1}, \ldots, x_1, x_0]$$

και ακόμη:

$$f[x_i, x_j] = \frac{f(x_i) - f(x_j)}{x_i - x_j}$$

$$f[x_i, x_j, x_k] = \frac{f[x_i, x_j] - f[x_j, x_k]}{x_i - x_k}$$

.

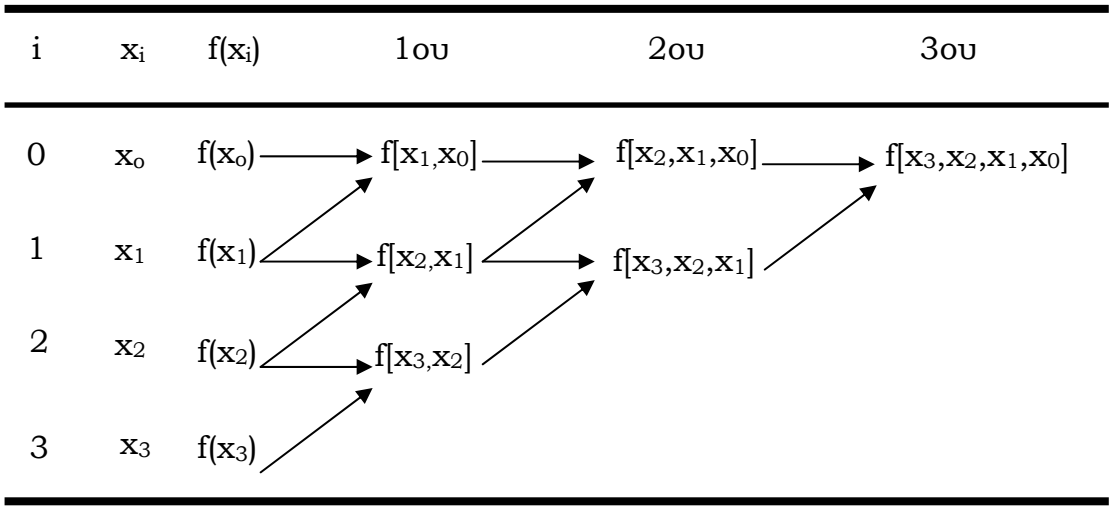
.

.

$$f[x_n, x_{n-1},..., x_1,.x_0] = \frac{f[x_n, x_{n-1},..., x_1] - f[x_{n-`}, x_{n-2},..., x_0]}{x_n - x_0},$$

οι οποίες είναι οι διηρημένες διαφορές 1ου, 2ου,...,n-οστού βαθμού αντίστοιχα, όπου n=το πλήθος των σημείων της παρεμβολής. Ο υπολογισμός τους φαίνεται και στον πίνακα του Σχήματος 6.5 όπου μπορεί κανείς να διακρίνει την αναδρομική τους φύση.

Είναι φανερό πως η μέθοδος αυτή υπολογίζει πολυωνυμικές συναρτήσεις και μάλιστα βαθμού τόσου όσα είναι τα σημεία της παρεμβολής.



Σχήμα 6.5

6.4 ΠΑΡΕΜΒΟΛΗ LAGRANGE

Η παρεμβολή Lagrange επαναδιατυπώνει το πολυώνυμο παρεμβολής Newton με διηρημένες διαφορές ώστε να αποφεύγονται οι υπολογισμοί των διηρημένων διαφορών και υπολογίζεται ως:

$$f_n(x) = \sum_{i=0}^{n} L_i(x) f(x_i)\text{ , όπου}$$

$$L_i(x) = \prod_{j=0,\, j\neq i}^{n} \frac{x - x_j}{x_i - x_j}$$

Τα $L_i(x)$ καλούνται συντελεστές Lagrange και το n πάλι δείχνει το πλήθος των σημείων παρεμβολής αλλά και το βαθμό του πολυωνύμου της παρεμβολής που υπολογίζεται.

Μπορούμε να παρατηρήσουμε πως στην παρεμβολή Lagrange η προσθήκη επιπλέον σημείων παρεμβολής σε μια προσπάθεια να αυξηθεί η ακρίβεια των αποτελεσμάτων μας οδηγεί σε πλήρη υπολογισμό όλων των συντελεστών από την αρχή. Κάτι τέτοιο είναι δυνατόν να αποφευχθεί με τη μέθοδο του Newton, η οποία μπορεί να εκμεταλλεύεται τους προηγούμενους υπολογισμούς των διηρημένων διαφορών.

6.5 ΚΥΒΙΚΕΣ SPLINES

Πολλές εφαρμογές απαιτούν ένα βαθμό πρακτικότητας που οι πολυωνυμικές μέθοδοι παρεμβολής, όπως αυτές που προαναφέρθηκαν, δεν αποδεικνύονται κατάλληλες. Ο λόγος είναι ότι τα πολυώνυμα μεγάλου βαθμού συνήθως παρουσιάζουν και μια μεγάλη ταλαντωτική συμπεριφορά, η οποία δεν είναι επιθυμητή για την προσέγγιση μεγεθών/συναρτήσεων που έχουν σχετικά ομαλή συμπεριφορά. Επιπρόσθετα, τα πολυώνυμα μεγάλου βαθμού είναι επιφέρουν και αυξημένη υπολογιστική πολυπλοκότητα όταν ο αριθμός των σημείων παρεμβολής αυξάνεται σημαντικά.

Για παράδειγμα θα ήταν ιδιαίτερα βαρύ να υπολογίσει κανείς το πολυώνυμο βαθμού 99 για ένα δοσμένο πλήθος 100 σημείων παρεμβολής. Η λύση σε αυτό το πρόβλημα είναι να βρούμε ένα πολυώνυμο μικρότερου βαθμού που να ταιριάζει όσο το δυνατόν καλύτερα στα δοσμένα σημεία. Ένα τέτοιο πολυώνυμο δε θα διασχίζει αναγκαστικά τα σημεία αυτά αλλά θα τα προσεγγίζει όσο το δυνατόν καλύτερα. Αυτό μπορεί να επιτευχθεί με την κατά τμήματα (piece-wise) πολυωνυμική παρεμβολή.

Στην κατά τμήματα πολυωνυμική παρεμβολή διάφορα πολυώνυμα μικρού βαθμού - και άρα υπολογίσιμα από πλευράς πολυπλοκότητας - ενώνονται μαζί με συνεχή τρόπο έτσι ώστε το τελικό πολυώνυμο να αποτελεί την προσέγγιση του μεγέθους που θέλουμε να παρεμβάλλουμε. Η απλούστερη περίπτωση θα ήταν να ενώσουμε απλά τα σημεία της παρεμβολής μεταξύ τους, κάτι που φαίνεται στη μεσαία γραφική παράσταση του Σχήματος 6.1, δημιουργώντας μια τεθλασμένη γραμμή.

Η απλή αυτή προσέγγιση - της τεθλασμένης γραμμής - δεν έχει ομαλότητα και ακόμη παρουσιάζει γωνιακά σημεία, στα οποία η πρώτη παράγωγος δεν είναι συνεχής. Γι' αυτό και δεν προσφέρεται παρά για να παράσχει μια «ιδέα» για την αυξομείωση του μεγέθους που μελετούμε. Για να ξεπεραστεί και αυτό το πρόβλημα χρησιμοποιείται η τεχνική των Splines.

Ας θεωρήσουμε ένα σύνολο σημείων $X_n=\{x_j\}$, $j=0,1,\ldots,n$ και $a=x_0<x_1<\ldots<x_n=b$ και το σύνολο τιμών $\{f(x_j)\}$, $j=0,1,\ldots,n$. Η ζητούμενη καμπύλη είναι εκείνη που περνά από τα σημεία $P_j=(x_j,f(x_j))$, $j=0,1,\ldots,n$.

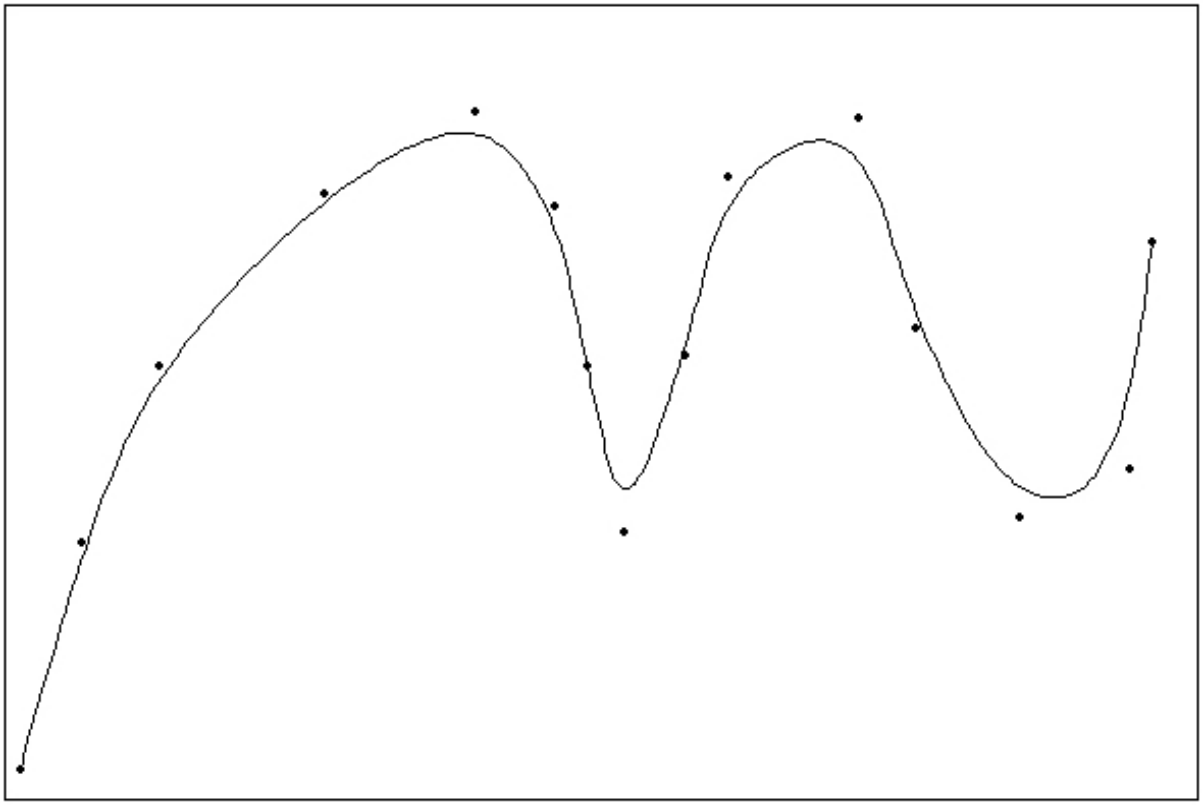
Ας θεωρήσουμε, ακόμη, το σύνολο των συναρτήσεων $Sp(X_n)$ τέτοιες ώστε αν $S(x)\in Sp(X_n)$ τότε η $S(x)$ να ικανοποιεί τις εξής τρεις ιδιότητες:

- $S(x)\in C^2[a,b]$, δηλαδή οι $S'(x)$ και $S''(x)$ να είναι συνεχείς συναρτήσεις στο διάστημα $[a,b]$.
- $S(x_j)=f(x_j)\equiv f_j$, $0\le j\le n$, δηλαδή η $S(x)$ να παρεμβάλλει την $f(x)$ στο διάστημα $[a,b]$.
- $S(x)$ να είναι πολυώνυμο $3^{ου}$ βαθμού σε κάθε υποδιάστημα $[x_j,x_{j+1}]$, $0\le j\le n-1$.

Η συνάρτηση $S(x)$ καλείται μια κυβική Spline και με όμοιο τρόπο μπορούν να οριστούν και Splines μεγαλύτερου βαθμού. Ακόμη, η συνάρτηση $S(x)$ μπορεί να είναι και διαφορετική σε κάθε υποδιάστημα $[x_j,x_{j+1}]$, $0\le j\le n-1$.

Για παράδειγμα ας πάρουμε την περίπτωση όπου γνωρίζουμε τέσσερα σημεία παρεμβολής και άρα n=3. Η S(x) θα αποτελείται από τα κυβικά πολυώνυμα $S_0(x)$ στο $[x_0,x_1]$, $S_1(x)$ στο $[x_1,x_2]$ και $S_2(x)$ στο $[x_2,x_3]$. Κάθε ένα από τα $S_j(x)$ είναι ένα πολυώνυμο τρίτου βαθμού και έχει τέσσερις συντελεστές που πρέπει να υπολογιστούν έτσι ώστε να ικανοποιούνται οι τρεις περιορισμοί που προαναφέρθηκαν για την S(x). Προχωρώντας με αυτές τις σκέψεις θα καταλήξουμε σε μια οικογένεια συναρτήσεων (σύστημα) με δύο παραμέτρους, όπου έχουμε τη δυνατότητα να επιλέξουμε αυθαίρετα είτε τις τιμές των πρώτων παραγώγων στα ακραία σημεία του συνόλου των σημείων παρεμβολής ($S_0'(x_0)$, $S_2'(x3)$) είτε τις τιμές των δεύτερων παραγώγων στα σημεία αυτά ($S_0''(x_0)$, $S_2''(x3)$). Αν οι πρώτες επιλεγούν ως μηδενικές, κάτι που αποτελεί συνήθη πρακτική τότε μιλούμε για φυσικές (natural) Splines.

Στο Σχήμα 6.6 φαίνεται η γενική μορφή μιας Spline.



Σχήμα 6.6

6.6 ΧΩΡΙΚΟ ΠΛΑΙΣΙΟ ΤΗΣ ΠΡΟΣΟΜΟΙΩΣΗΣ

Για να εφαρμόσουμε, λοιπόν, τις μεθόδους που αναφέρθηκαν παραπάνω χρειάστηκε να επιλέξουμε και κάποιο χώρο στον οποίο η μελέτη μας να προσομοιωθεί. Για το σκοπό αυτό προτιμήθηκε η εφαρμογή να γίνει όχι μόνο σε ένα χώρο αλλά σε τρεις διαφορετικούς, έτσι ώστε τα αποτελέσματα να είναι όσο το δυνατόν πιο γενικευμένα.

Το «σενάριο» της προσομοίωσης έχει ως εξής: Θεωρούμε ένα χρήστη που μετακινείται και διαθέτει μια συσκευή, η οποία έχει τη δυνατότητα να ενημερώνει σε τακτά χρονικά διαστήματα κάποιο σύστημα για την τοποθεσία του χρήστη αυτού. Το σύστημα λαμβάνει τις διαδοχικές τοποθεσίες του και προσπαθεί ανά πάσα στιγμή να δώσει εκτιμήσεις για την/τις μελλοντική/ές τοποθεσία/ες του.

Για να καταστεί δυνατή, επομένως, μια τέτοια προσομοίωση υπάρχει η ανάγκη για μία «πηγή» διαδοχικών τοποθεσιών για το φανταστικό μας χρήστη. Η προσομοίωση, όπως προαναφέρθηκε, επιλέχθηκε να εφαρμοστεί για λόγους γενικότητας σε τρεις διαφορετικές πηγές τοποθεσιών. Η «τοποθεσία» του χρήστης από εδώ και στο εξής θα καλείται «στίγμα» του χρήστη. Τα στίγματα καταχωρούνται σε αρχείο από όπου και διαβάζονται από την προσομοίωση όπως θα γινόταν σε πραγματικό χρόνο. Οι πηγές θα είναι οι εξής:

- **Αυθαίρετα (ή απλά στίγματα):** Τα στίγματα αυτού του τύπου είναι μόνο καταχρηστικά αποδεκτά μιας και δεν ακολουθούν κάποιο κανόνα ως προς την παραγωγή τους. Ο εκτελεστής της προσομοίωσης έχει τη δυνατότητα να θέσει ως είσοδο οποιαδήποτε διαδρομή θεωρεί χρήσιμη να έχει ακολουθήσει ο χρήστης. Στην περίπτωση αυτών των στιγμάτων επαφίεται σε αυτόν που δίνει την είσοδο στην προσομοίωση να επιλέξει στίγματα που να έχουν νόημα για τα συμπεράσματα που θέλει να εξάγει. Τα απλά στίγματα έχουν την ακόλουθη μορφή:

ID χρήστη	Χρόνος (t)	Θέση X - άξονα	Θέση Y - άξονα	Θέση Z - άξονα

Για παράδειγμα μερικές από τις γραμμές ενός τέτοιου αρχείου:
264215769 , 0.326378 , -73.9721 , 40.7855,16
476130054 , 0.634028 , -73.9735 , 40.7929,0
696020740 , 0.986450 , -73.9704 , 40.7914,34
969010951 , 1.376213 , -73.971 , 40.794,11
918008951 , 1.767245 , -73.9708 , 40.7943,0
965733157 , 2.103650 , -73.9683 , 40.7877,30 κ.ο.κ.

- **Στίγματα City Simulator:** Ο City Simulator είναι ένα τρισδιάστατο μοντέλο πόλης που επιτρέπει τη δημιουργία δυναμικών χωρικών δεδομένων προσομοιώνοντας την κίνηση πολλαπλών χρηστών. Είναι γραμμένος σε Java και σχεδιασμένος ώστε να παράγει ρεαλιστικά δεδομένα για τις ανάγκες άλλων εφαρμογών που τα χρειάζονται για να πραγματοποιήσουν τις δικές τους προσομοιώσεις. Είναι μια ιδιαίτερα παραμετροποιήσιμη εφαρμογή που μπορεί να προσομοιώσει την κίνηση πολλών χρηστών ανάμεσα σε δρόμους και κτήρια μιας πόλης αλλά και ανάμεσα στους ορόφους κτηρίων. Το αρχείο που παράγεται από τον City Simulator έχει εγγραφές με την παρακάτω μορφή:

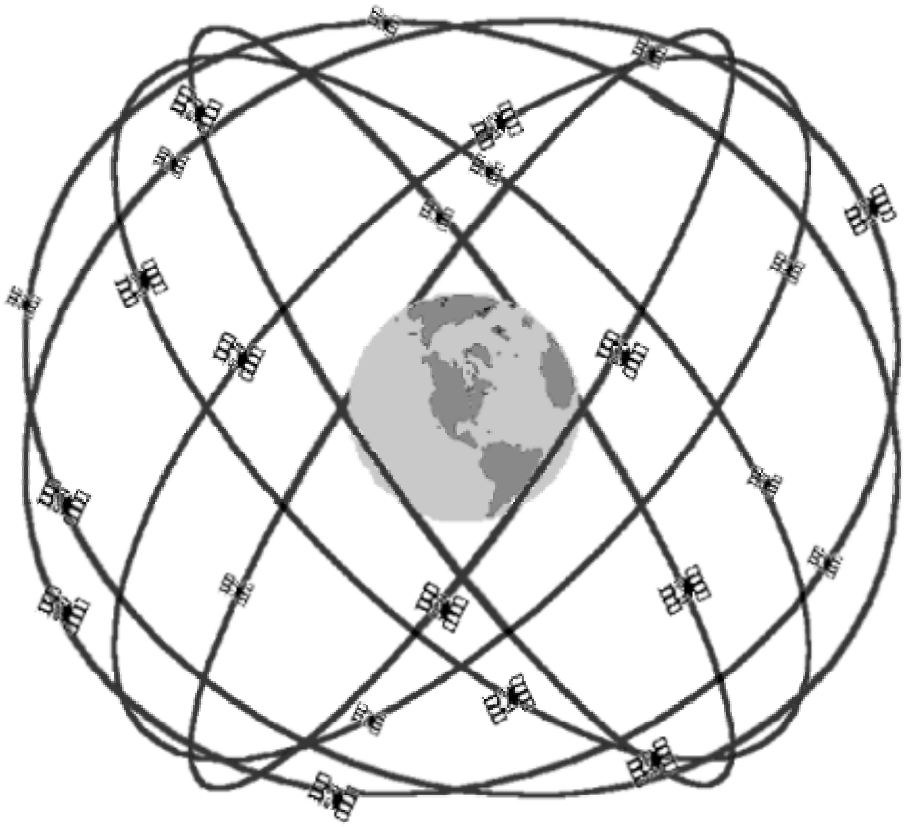
ID χρήστη	Χρόνος (t)	Θέση X - άξονα	Θέση Y - άξονα	Θέση Z - άξονα	Αριθμός κύκλου προσομοίωσης

Για παράδειγμα μερικές από τις γραμμές ενός τέτοιου αρχείου:
1 , 13.223364 , 802.8,261.6 , 4.6 # cycle = 1
1 , 31.193111 , 802.0,263.2 , 4.6 # cycle = 2
1 , 44.826985 , 803.8,262.0 , 4.6 # cycle = 3
1 , 62.873528 , 803.0,261.8 , 4.6 # cycle = 4

1 , 80.589481 , 802.4,262.4 , 4.6 # cycle = 5

1 , 97.932973 , 801.4,264.0 , 4.6 # cycle = 6

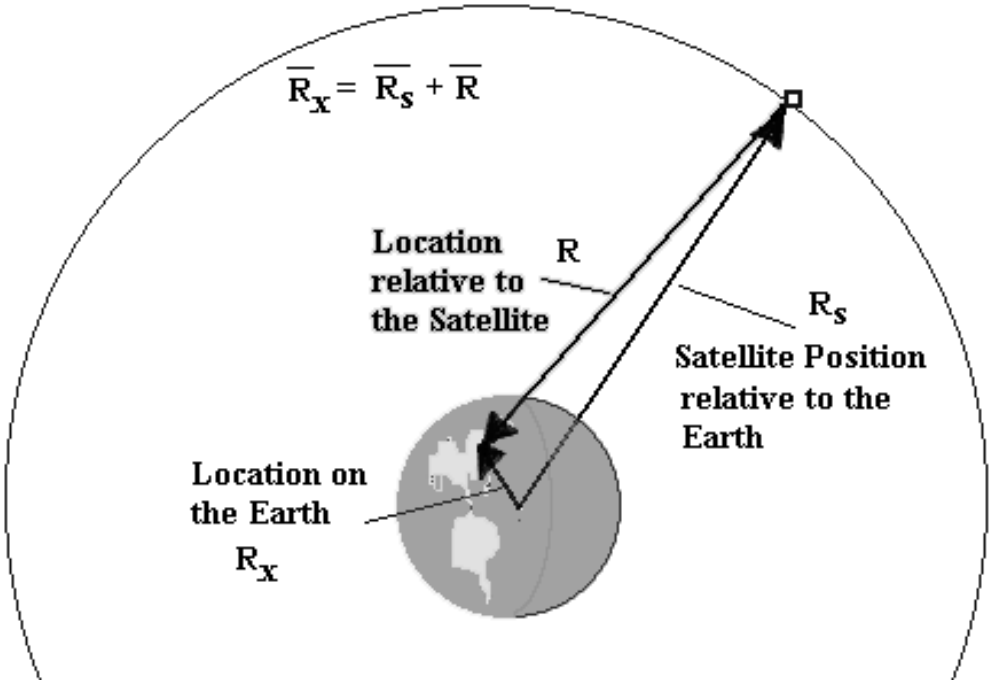
- **Στίγματα GPS:** Τα αρχικά GPS σημαίνουν Global Positioning System. Το GPS είναι ένα δίκτυο δορυφόρων που περιστρέφονται γύρω από τη γη σε συνδυασμό με συγκεκριμένους επίγειους σταθμούς ελέγχου, του οποίου η λειτουργία είναι ευθύνη της κυβέρνησης των Ηνωμένων Πολιτειών της Αμερικής. Το δίκτυο αυτό παράσχει πληροφορία χρονική και χωρική στους δέκτες του σε όλη την υφήλιο. Υπάρχουν 24 τέτοιοι ενεργοί δορυφόροι για οποιοδήποτε σημείο και για κάθε στιγμή. Κάθε δορυφόρος εκπέμπει συνεχόμενα πληροφορία για τη θέση του και για να πάρει ένας δέκτης ακριβή μέτρηση της τοποθεσίας του πάνω στη γη χρειάζεται να έχει επαφή με τέσσερις από αυτούς. Στο σχήμα 6.7 φαίνεται η διάταξη των δορυφόρων γύρω από τη γη.



Σχήμα 6.7

Υπάρχουν δύο δυνατότητες λήψης πληροφορίας από τους δορυφόρους: ακριβής (Precise Positioning Service - PPS) και standard (Standard Positioning Service - SPS). Η δεύτερη είναι διαθέσιμη στο κοινό και με αυτή θα ασχοληθούμε, ενώ η πρώτη εξυπηρετεί στρατιωτικές ανάγκες και γίνεται προσιτή έπειτα από ειδική άδεια.

Ο τρόπος με τον οποίο ένας δέκτης μπορεί να υπολογίσει την τοποθεσία στην οποία βρίσκεται πάνω στη γη παίρνοντας πληροφορία από κάθε έναν από τους τέσσερις δορυφόρους φαίνεται σχηματικά στο Σχήμα 6.8.



Σχήμα 6.8

Η πιο κοινή ανάγκη που εξυπηρετεί το GPS είναι η πλοήγηση. Δέκτες GPS υπάρχουν σε πλοία, αυτοκίνητα, αεροπλάνα κλπ. Συχνά οι εφαρμογές που το χρησιμοποιούν το συνδυάζουν με κάποια υπηρεσία χαρτογράφησης ώστε να παράσχουν καθοδήγηση στους χρήστες. Στο Σχήμα 6.9 φαίνονται δύο συσκευές δέκτες GPS.



Σχήμα 6.9

Για τις ανάγκες της προσομοίωσης δημιουργήθηκαν αρχεία με στίγματα GPS που έχουν παρθεί από την κίνηση χρηστών μέσα σε αστικό περιβάλλον σε πραγματικές συνθήκες. Ένα τέτοιο αρχείο περιέχει διάφορες μορφές GPS στιγμάτων από τις οποίες η πιο κοινή είναι η NMEA-0183 (National Marine Electronic Association). Η μορφή αυτών των στιγμάτων είναι η εξής: $GPGGA, hhmmss.ss, ddmm.mmmm, n, dddmm.mmmm, e, q, ss, y.y, a.a, z, g.g, z, t.t, iiii*CC

Οι έννοιες που περικλείονται στο στίγμα αυτού του τύπου επεξηγούνται αναλυτικά στον πίνακα που ακολουθεί.

Πεδίο	Τιμή	Έννοια
1	**hhmmss.ss** (π.χ. 180432.00)	UTC (Coordinated Universal Time) - ώρα Greenwich (π.χ. 18 ώρες και 04 λεπτά και 32.00 δευτερόλεπτα)
2	**ddmm.mmmm** (π.χ. 4027.027912)	Γεωγραφικό πλάτος (π.χ. 40 μοίρες και 27.027912 δεύτερα)
3	**n** (π.χ. N)	Κατεύθυνση γεωγραφικού πλάτους (N=Βόρεια, S=Νότια)
4	**dddmm.mmmm** (π.χ. 08704.857070)	Γεωγραφικό μήκος (π.χ. 87 μοίρες και 4.85707 δεύτερα)
5	**e** (π.χ. W)	Κατεύθυνση γεωγραφικού μήκους (E=Ανατολή, W=Δύση)
6	**q** (π.χ. 2)	Δείκτης ποιότητας GPS (0 - fix not valid, 1 - GPS fix, 2 - DGPS fix)
7	**ss** (π.χ. 07)	Αριθμός ενεργών δορυφόρων (00-24)
8	**y.y** (π.χ. 1.0)	Οριζόντια διάχυση ακριβείας (DOP - Dilution Of Precision)
9	**a.a** (π.χ 212.15)	Ύψος πάνω από το μέσο θαλάσσιο επίπεδο (π.χ. 212.15 m)
10	**z** (π.χ. M)	Μονάδα μέτρησης ύψους (M=μέτρα)
11	**g.g** (π.χ. -33.81)	Γεωειδής διαχωρισμός (π.χ.-33.81 m)
12	**z** (π.χ. M)	Μονάδα του γεωειδούς διαχωρισμού (M=μέτρα)
13	**t.t** (π.χ. 4.2)	Ηλικία της εγγραφής του διαφορικού GPS
14	**iiii** (π.χ. 0555)	ID Σταθμού βάσης (τιμές 0000-1023)
15	***CC**	Άθροισμα ελέγχου (checksum)

Για παράδειγμα μερικές από τις γραμμές ενός τέτοιου αρχείου, από τις οποίες μας ενδιαφέρουν μόνο εκείνες που ξεκινούν με $GPGGA και αναφέρονται στο φορμαλισμό NMEA-0183:

$GPGGA,105609.401,3758.0323,N,02346.5039,E,1,04,13.8,246.8,M,,,,0000*30

$GPGSA,A,3,19,22,31,15,,,,,,,,,18.1,13.8,11.7*09

$GPRMC,105609.401,A,3758.0323,N,02346.5039,E,18.45,87.76,140404,,*09

$GPGGA,105610.401,3758.0311,N,02346.5104,E,1,05,9.6,247.6,M,,,,0000*0D

$GPGSA,A,3,19,22,31,15,18,,,,,,,,11.9,9.6,7.0*04

$GPGSV,3,1,09,03,75,199,,19,61,307,32,22,55,046,36,31,52,304,33*7D
$GPGSV,3,2,09,15,46,064,35,14,40,131,33,18,19,046,32,11,14,287,23*7A
$GPGSV,3,3,09,16,11,198,*47
$GPRMC,105610.401,A,3758.0311,N,02346.5104,E,18.63,98.33,140404,,*04
$GPGGA,105611.401,3758.0294,N,02346.5168,E,1,05,9.6,248.4,M,,,,0000*07

6.7 Η ΠΡΟΣΟΜΟΙΩΣΗ

Η προσομοίωση υλοποιήθηκε σε Java 1.4 και ο πηγαίος κώδικας βρίσκεται στο παράρτημα της παρούσας εργασίας. Τρέχοντας κανείς την προσομοίωση χωρίς παραμέτρους μπορεί να δει ό,τι χρειάζεται να δώσει στην εφαρμογή σαν είσοδο για να πάρει αποτελέσματα:

```
% java simulation
 > Usage:
 > | param1 = DATA_MODE {SIMPLE, CITY_SIMULATOR, GPS}
 > | param2 = INPUT_FILE {path absolute/relative}
 > | param3 = QUEUE_LENGTH {integer}
 > | param4 = TIME_STEP {double}
 > | param5 = GRANULARITY - as a radius {double}
 > | param6 = OUTPUT_FILE {path absolute/relative}
 > | (use no parameters for HELP)
 > Simulation exiting...
```

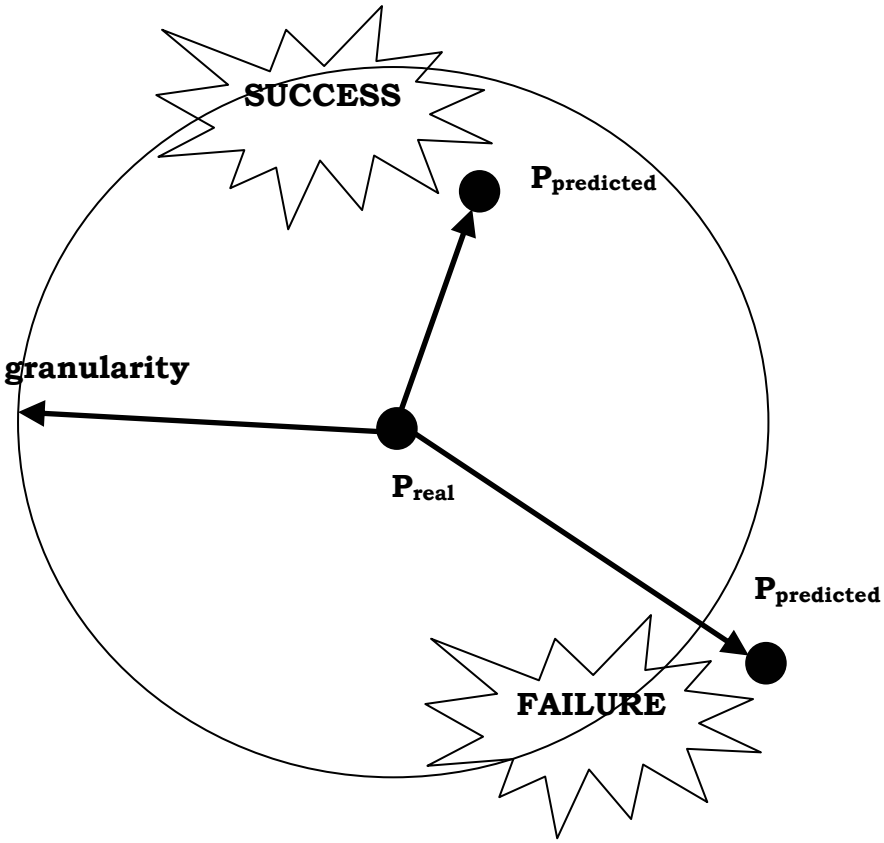
Εξήγηση των παραμέτρων:

- **DATA_MODE:** Το είδος της εισόδου που επιλέγει ο χρήστης. SIMPLE για απλά/αυθαίρετα στίγματα, CITY_SIMULATOR για στίγματα που προέρχονται από τον City Simulator και GPS για στίγματα από το σύστημα GPS.
- **INPUT_FILE:** Το όνομα του αρχείου που περιέχει τα στίγματα εισόδου - είτε απλό όνομα είτε διαδρομή (path) προς το αρχείο. Αν το αρχείο δεν αντανακλά το DATA_MODE που επιλέχθηκε τότε η εφαρμογή θα τερματίσει με αντίστοιχο μήνυμα λάθους.
- **QUEUE_LENGTH:** Το βάθος γνώσης που θα πρέπει να λαμβάνει υπόψη η προσομοίωση για να κάνει πρόβλεψη για τις μελλοντικές

τοποθεσίες του χρήστη. Καθώς η εφαρμογή τρέχει οι τοποθεσίες από τις οποίες περνά ο φανταστικός χρήστης καταγράφονται και κάθε φορά που έρχεται μία νέα τοποθεσία γίνεται και μία πρόβλεψη για το μέλλον. Αυτή η παράμετρος καθορίζει το πόσες περασμένες τοποθεσίες του χρήστη θέλει αυτός που εκτελεί την προσομοίωση να λαμβάνονται υπόψη στην προσπάθεια για προβλέψεις. Αποτελεί μία από τις πιο κρίσιμες παραμέτρους της προσομοίωσης που θα καθορίσει σε μεγάλο βαθμό τα συμπεράσματα που θα εξαχθούν στο τέλος.

- **TIME_STEP:** Το βήμα του χρόνου μέσα στο οποίο έρχεται ένα νέο στίγμα. Για παράδειγμα στο GPS νέα στίγματα έρχονται περίπου ανά ένα δευτερόλεπτο.
- **GRANULARITY:** Παράμετρος που καθορίζει το μέγιστο όριο λάθους που ανέχεται η προσομοίωση για να θεωρήσει μια πρόβλεψη ως επιτυχημένη ή αποτυχημένη. Η παράμετρος αυτή δίνεται σε μέτρα (m) και στο τέλος της προσομοίωσης εκτυπώνονται το πλήθος επιτυχημένων και αποτυχημένων προβλέψεων. Επιτυχημένη θα θεωρηθεί μια πρόβλεψη όταν η τοποθεσία που προέβλεψε η προσομοίωση βρίσκεται μέσα στον κύκλο που ορίζεται από την πραγματική τοποθεσία στην οποία βρέθηκε ο χρήστης (κέντρο του κύκλου) και με ακτίνα το granularity, όπως φαίνεται στο Σχήμα 6.10.
- **OUTPUT_FILE:** Το αρχείο στο οποίο η προσομοίωση θα καταγράψει αναλυτικά όλα τα βήματα που έκανε και τα αποτελέσματα που παράγει - και ανά βήμα αλλά και συνολικά κατά την έξοδό της.

Καθώς η προσομοίωση τρέχει νέα στίγματα έρχονται σε τακτά χρονικά διαστήματα (TIME_STEP) που μας πληροφορούν για τις τρέχουσες τοποθεσίες του χρήστη. Η εφαρμογή τις δέχεται, τις επεξεργάζεται και με τις μεθόδους παρεμβολής που συζητήσαμε κάνει κάθε φορά μια πρόβλεψη για την επόμενη τοποθεσία στην οποία πιστεύει πως θα βρεθεί ο χρήστης.



Σχήμα 6.10

Παραδείγματα εκτέλεσης:

```
% java simulation SIMPLE data/p1.txt 6 5.0 15 CitySim.txt
 > Starting mobility simulation...
 > DATA_MODE=SIMPLE
 > INPUT_FILE=D:\Java\mobility\data\p1.txt
 > QUEUE_LENGTH=6
 > TIME_STEP=5.0
 > GRANULARITY=15.0
 > OUTPUT_FILE=D:\Java\mobility\CitySim.txt
 > SIMPLE start...
 > Exporting results to D:\Java\mobility\CitySim.txt
 > Mean error percentages:
 >    N.D.D: 25.58407788736459%
 >    CbSpl: 2.9929517370348466%
 >    Lgrng: 25.58407788736458%
 > Successes (out of 45 tries):
```

```
 >    N.D.D: 0
 >    CbSpl: 26
 >    Lgrng: 0
 > SIMPLE end...
 > Ending mobility simulation...
```

Στο παραπάνω παράδειγμα η προσομοίωση έτρεξε για απλά στίγματα και στις τελευταίες γραμμές εκτύπωσε τα ποσοστά επιτυχίας της κάθε μιας από τις μεθόδους παρεμβολής που χρησιμοποιήθηκαν. Για την ακρίβεια εκτυπώνει το μέσο ποσοστιαίο λάθος της κάθε μιας.

Στο επόμενο παράδειγμα η προσομοίωση τρέχει για GPS στίγματα:

```
% java simulation GPS data/gps.txt 6 1.0 15 gps_res.txt
 > Starting mobility simulation...
 > DATA_MODE=GPS
 > INPUT_FILE=D:\Java\mobility\data\gps.txt
 > QUEUE_LENGTH=6
 > TIME_STEP=1.0
 > GRANULARITY=15.0
 > OUTPUT_FILE=D:\Java\mobility\gps_res.txt
 > GPS start...
 > Reading GPS signals as NMEA-0183 sentences (GPGGA)...
 > Exporting results to D:\Java\mobility\gps_res.txt
 > Mean error percentages:
 >    N.D.D: 0.1790736150031899%
 >    CbSpl: 0.015091436795098494%
 >    Lgrng: 0.17907361500322527%
 > Successes (out of 304 tries):
 >    N.D.D: 284
 >    CbSpl: 298
 >    Lgrng: 284
 > GPS end...
 > Ending mobility simulation...
```

Το αρχείο εξόδου έχει την παρακάτω μορφή:

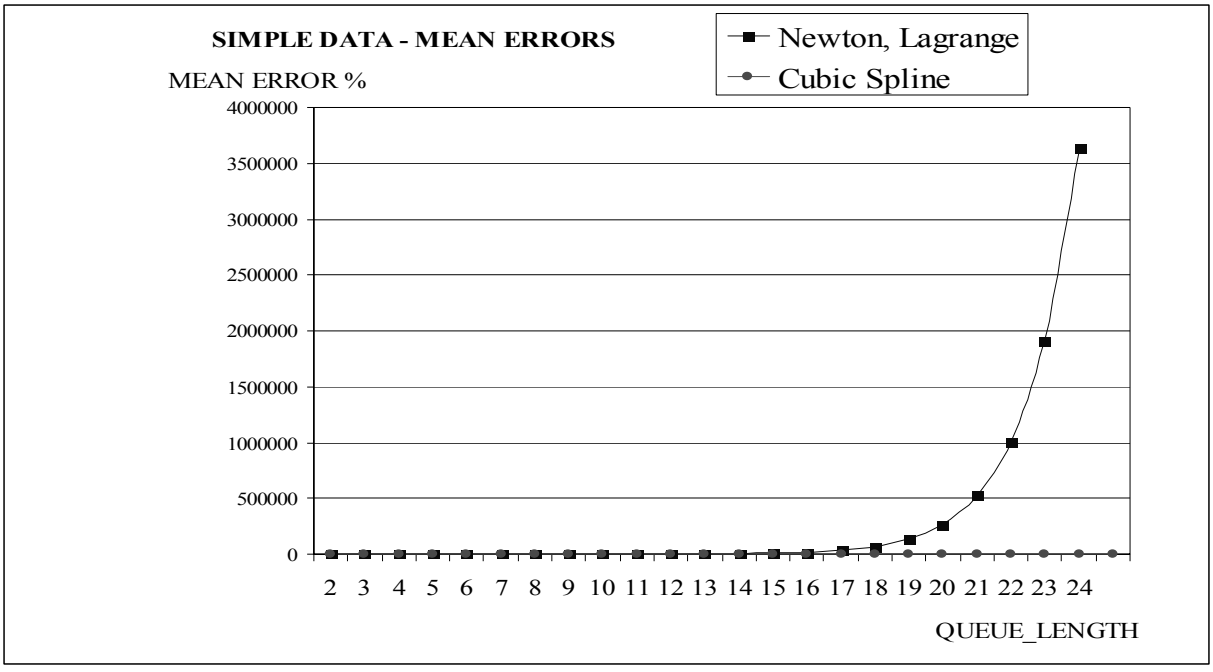
```
     MOBILITY SIMULATION RESULTS (Sun Oct 31 16:50:00 EET
2004)
DATA_MODE=GPS (NMEA 0183 format - GPGGA)
INPUT_FILE=D:\Java\mobility\data\gps.txt
QUEUE_SIZE=6
TIME_STEP=1.0
GRANULARITY=15.0
N.D.D=NEWTON  DIVIDED  DIFFERENCES,  CbSpl=CUBIC  SPLINES,
Lgrng=LAGRANGE
stigma = UTC - Latitude, Longitude, Altitude

stigma: 39233.409 - 136680.1247,85560.047,224.8
    Waiting for queue to fill with data...
stigma: 39234.409 - 136680.1247,85560.047,224.8
    Waiting for queue to fill with data...
stigma: 39235.408 - 136680.1247,85560.047,224.8
    Waiting for queue to fill with data...
stigma: 39236.408 - 136680.1247,85560.047,224.8
    Waiting for queue to fill with data...
stigma: 39237.408 - 136680.1247,85560.047,224.8
    Waiting for queue to fill with data...
stigma: 39238.408 - 136680.1247,85560.047,224.8
Next(?) N.D.D: 136680.1247,85560.047,224.8
        CbSpl: 136680.1247,85560.047,224.8
        Lgrng:
136680.12470000004,85560.0470000002,224.79999999999973
stigma: 39239.408 - 136680.1247,85560.047,224.8
Error N.D.D 0.0%,0.0%,0.0%    SUCCESS
      CbSpl 0.0%,0.0%,0.0%    SUCCESS
      Lgrng  -4.258677773467631E-14%,-2.2110190983037803E-
13%,1.2643109177225982E-13%    SUCCESS
Next(?) N.D.D: 136680.1247,85560.047,224.8
        CbSpl: 136680.1247,85560.047,224.8
```

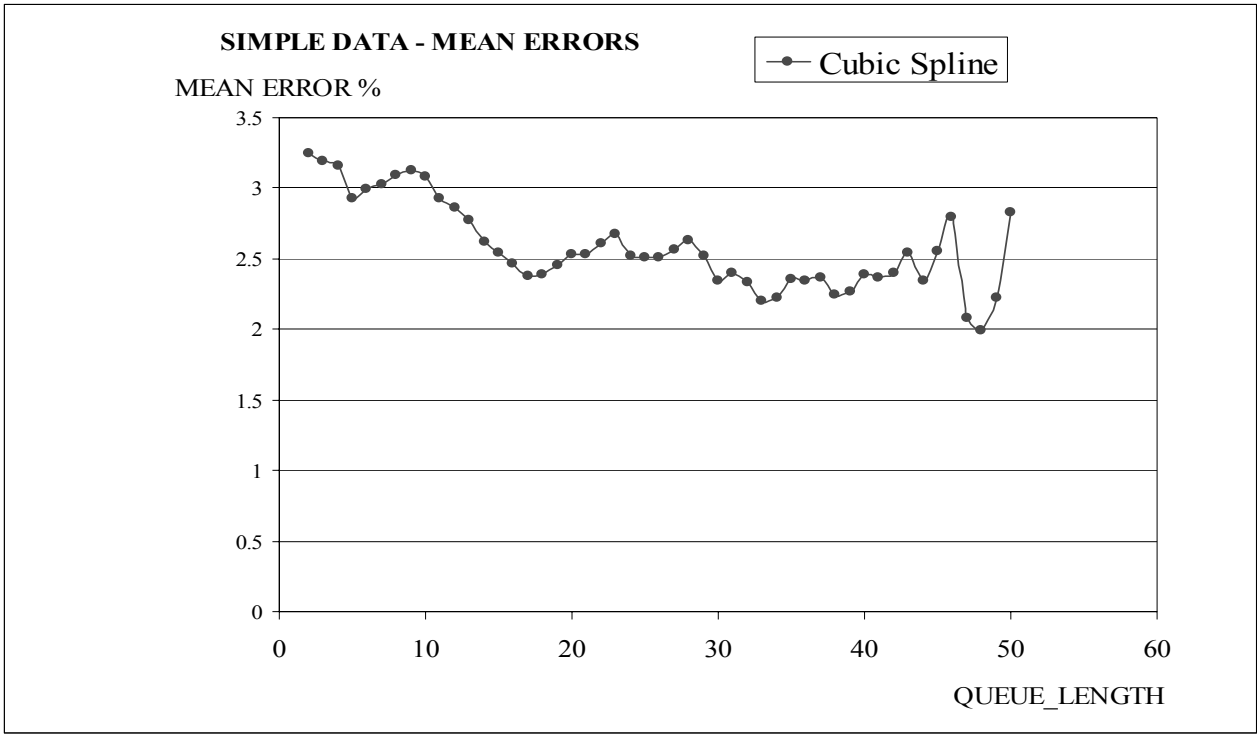
```
        Lgrng:
136680.1246999997,85560.04699999973,224.79999999999973
```

Μέχρι να περάσει αρκετός χρόνος ωσότου η εφαρμογή να μπορεί να παράσχει προβλέψεις δεν γίνεται τίποτα, απλώς διαβάζονται τα στίγματα του χρήστη. Από εκεί και πέρα η εφαρμογή κάνει συνεχώς - με την άφιξη κάθε νέου στίγματος προβλέψεις για το επόμενο και ταυτόχρονα ελέγχει πόσο καλά τα πήγε στην αμέσως προηγούμενη πρόβλεψη εκτυπώνοντας αναλυτικά για κάθε μέθοδο παρεμβολής.

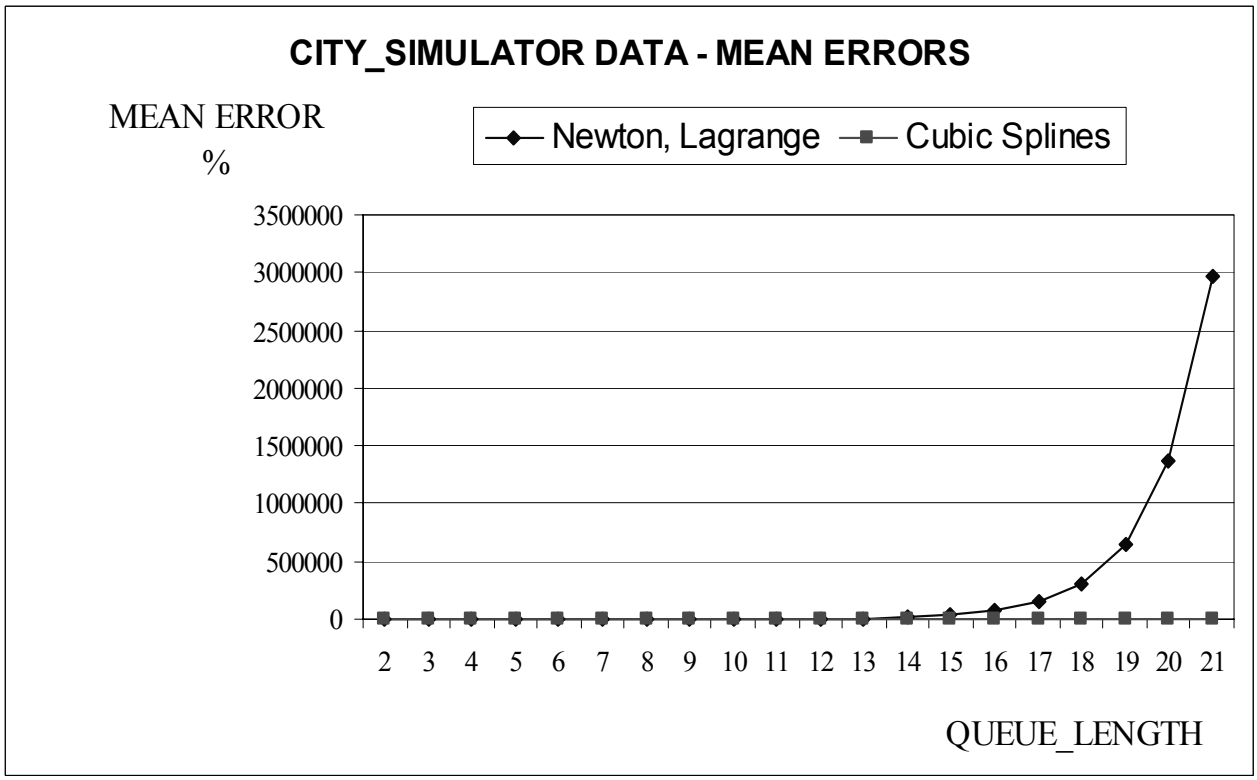
Ακολουθούν πίνακες με αποτελέσματα σχετικά με το πώς συμπεριφέρθηκε η κάθε μέθοδος τελικά όσον αφορά την ικανότητά της να προβλέπει σε συνάρτηση με το QUEUE_LENGTH.



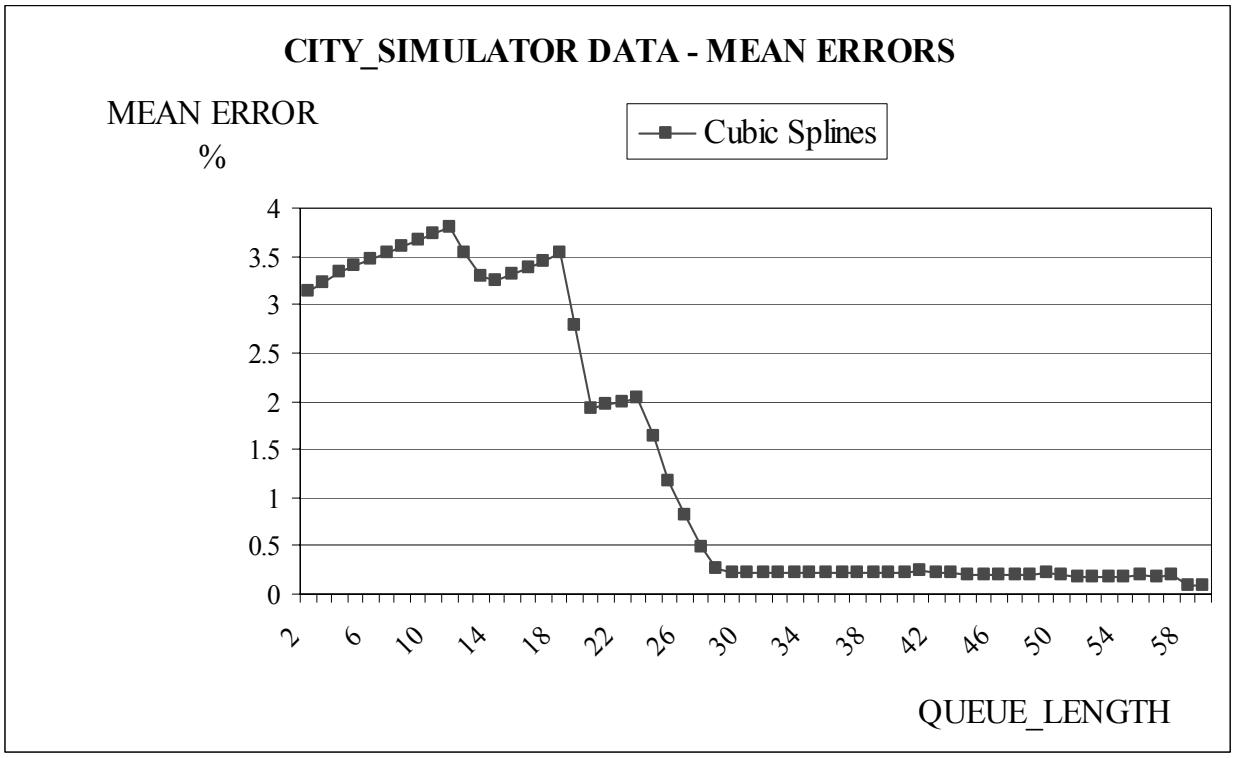
Μέσο λάθος των μεθόδων για απλά στίγματα



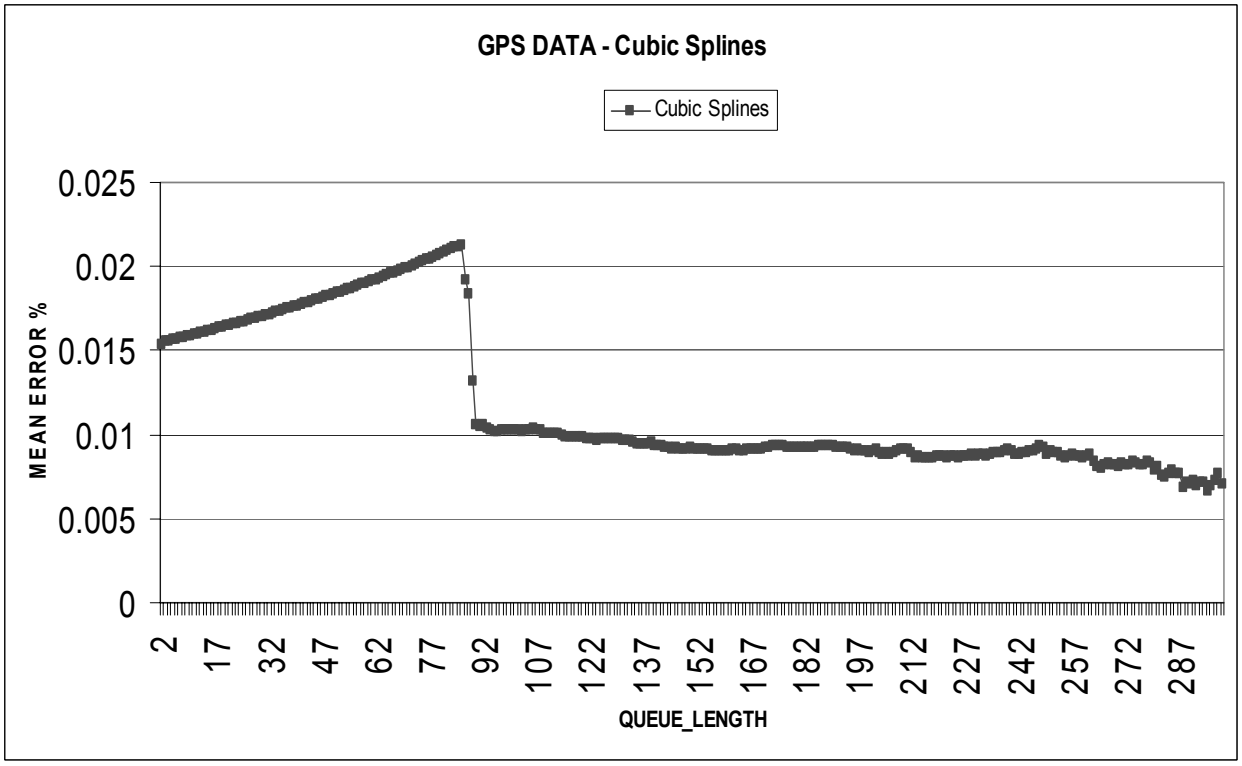
Μεγέθυνση του προηγούμενου διαγράμματος μόνο για τις Cubic Splines



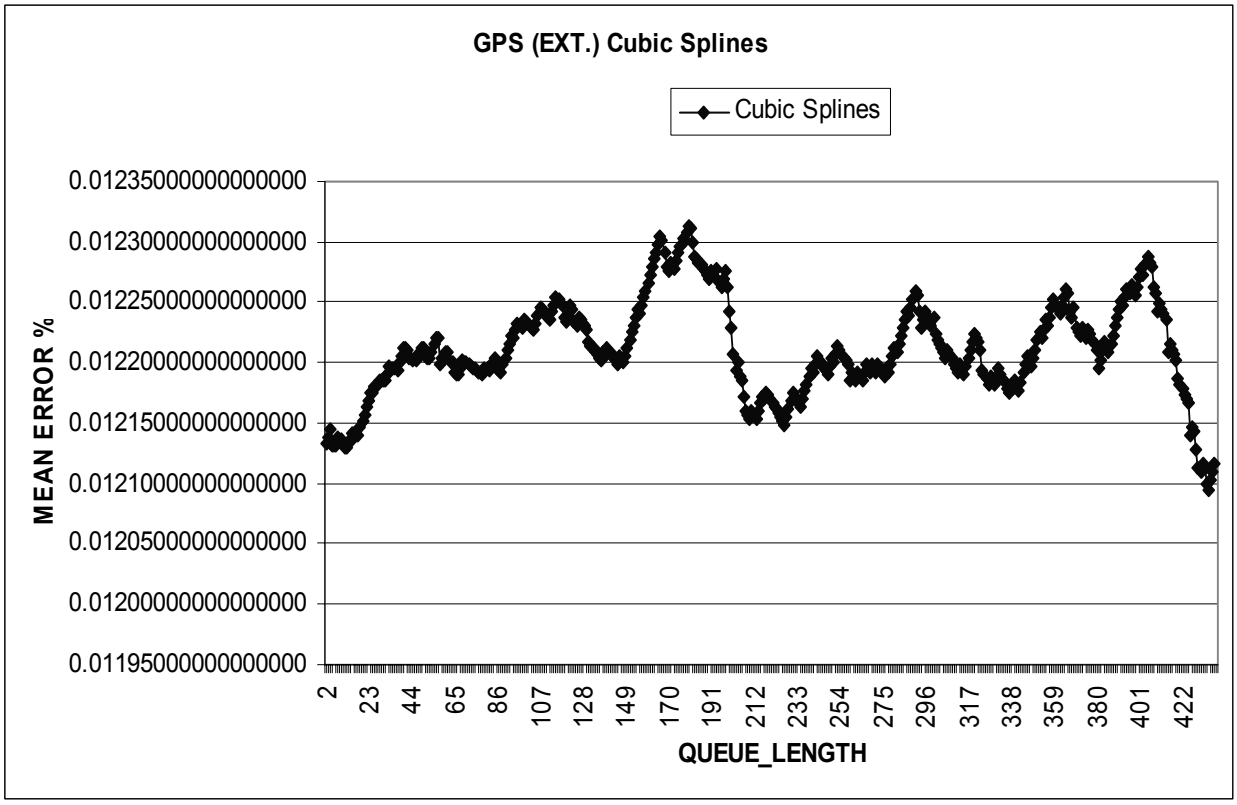
Μέσο λάθος των μεθόδων για στίγματα από τον City Simulator



Μεγέθυνση του προηγούμενου διαγράμματος μόνο για τις Cubic Splines



Μέσο λάθος μόνο για τις Cubic Splines για GPS στίγματα.



Μέσο λάθος μόνο για τις Cubic Splines για άλλη ομάδα GPS στιγμάτων.

Γίνεται εύκολα αντιληπτό από τα παραπάνω ότι οι κυβικές Splines τα κατάφεραν πολύ καλύτερα στην προσπάθεια να προβλέψουν την επόμενη θέση του χρήστη κι επομένως το χωρικό του πλαίσιο. Μάλιστα, μπορεί να παρατηρήσει κανείς πως ουσιαστικά όσο μεγάλο QUEUE_LENGTH και αν επιλέξει - το ιστορικό της γνώσης, δηλαδή - δεν κερδίζει κατά μέσο όρο ιδιαίτερη ακρίβεια στην πρόβλεψη που κάνει το σύστημα.

6.8 ΣΕ ΒΑΘΟΣ ΧΡΟΝΟΥ

Έτσι επιλέγουμε να επεκτείνουμε τη μελέτη μας στη δυνατότητα πρόβλεψης χρησιμοποιώντας πια μόνο τις Splines. Το επόμενο βήμα είναι να δούμε πόσο βαθιά μέσα στο χρόνο θα μπορούσαμε να προβλέψουμε και με τι ακρίβεια. Δηλαδή, η προσομοίωσή μας μπορεί

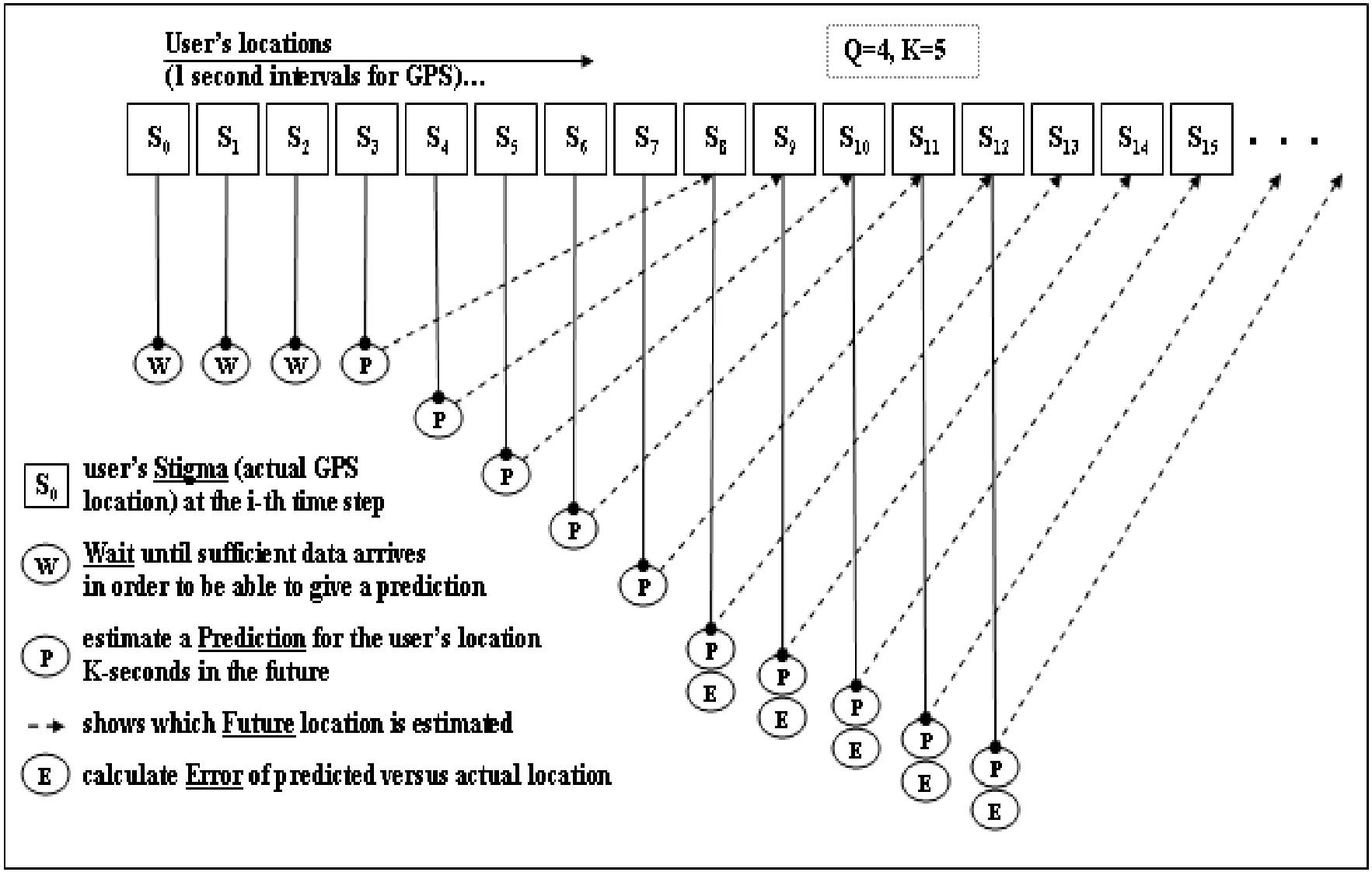
να προβλέπει μόνο την επόμενη τοποθεσία ενός κινητού χρήστη ή και τη μεθεπόμενη ή ακόμη και την επόμενη από αυτήν;

Τώρα, λοιπόν, κάθε φορά που έρχεται στην προσομοίωσή μας ένα νέο στίγμα - τοποθεσία - του χρήστη, η προσομοίωση προσπαθεί να προβλέψει τη θέση του k χρονικά βήματα στο μέλλον. Το πώς το κάνει αυτό φαίνεται στο Σχήμα 6.11.

```
% java simulationGpsNK
 > Usage:
 > | param1 = INPUT_FILE {path absolute/relative}
 > | param2 = QUEUE_LENGTH {integer}
 > | param3 = TIME_STEP {double}
 > | param4 = FUTURE_STEP {integer}
 > | param5 = OUTPUT_FILE {path absolute/relative}
 > | (use no parameters for HELP)
 > Simulation exiting...
```

Η μόνη διαφορά είναι ότι τώρα η εφαρμογή τρέχει ρητά για GPS στίγματα και υπάρχει και μια επιπλέον παράμετρος, η FUTURE_STEP, η οποία καθορίζει ακριβώς σε πόσο βάθος χρόνου θέλουμε να γίνονται οι προβλέψεις. Ας δούμε ένα παράδειγμα:

```
% java simulationGpsNK data\gps.txt 5 1.0 3 fin_gps.txt
 > Starting mobility simulation for GPS N+K future
predictions...
 > DATA_MODE=
 > INPUT_FILE=D:\Java\mobility\data\gps.txt
 > QUEUE_LENGTH=5
 > TIME_STEP=1.0
 > FUTURE_STEP=3
 > OUTPUT_FILE=D:\Java\mobility\fin_gps.txt
 > GPS-NK start (only with Cubic Splines)...
```


User's locations
(1 second intervals for GPS)...
Q=4, K=5
S0 S1 S2 S3 S4 S5 S6 S7 S8 S9 S10 S11 S12 S13 S14 S15 . . .
W W W P
P
P
P
P
P E
P E
P E
P E
P E
S0 user's Stigma (actual GPS location) at the i-th time step
W Wait until sufficient data arrives in order to be able to give a prediction
P estimate a Prediction for the user's location K-seconds in the future
shows which Future location is estimated
E calculate Error of predicted versus actual location

Σχήμα 6.11

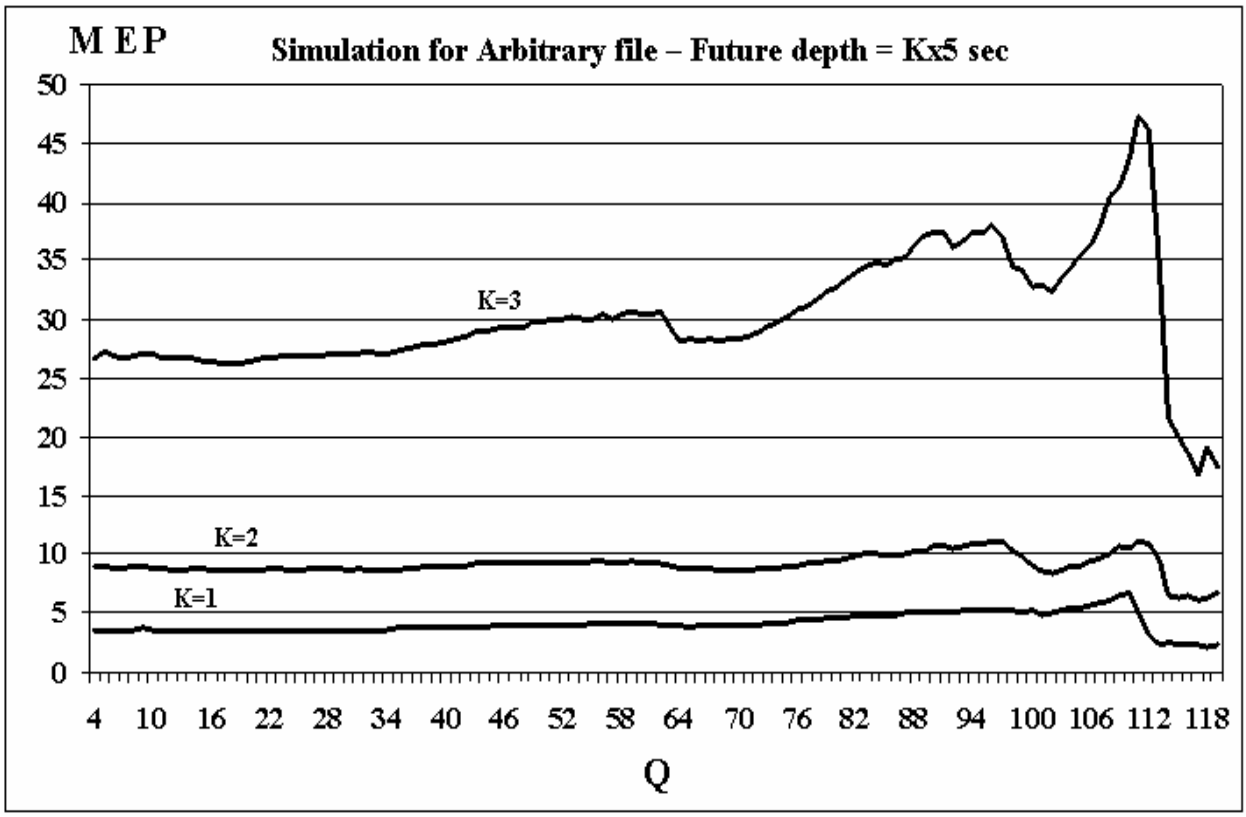
```
> Reading GPS signals as NMEA-0183 sentences (GPGGA)...
> Exporting results to D:\Java\mobility\fin_gps.txt

> Mean error percentages:
>    CbSpl: 0.12078052399025571%
> GPS-NK end...
> Ending mobility simulation...
```

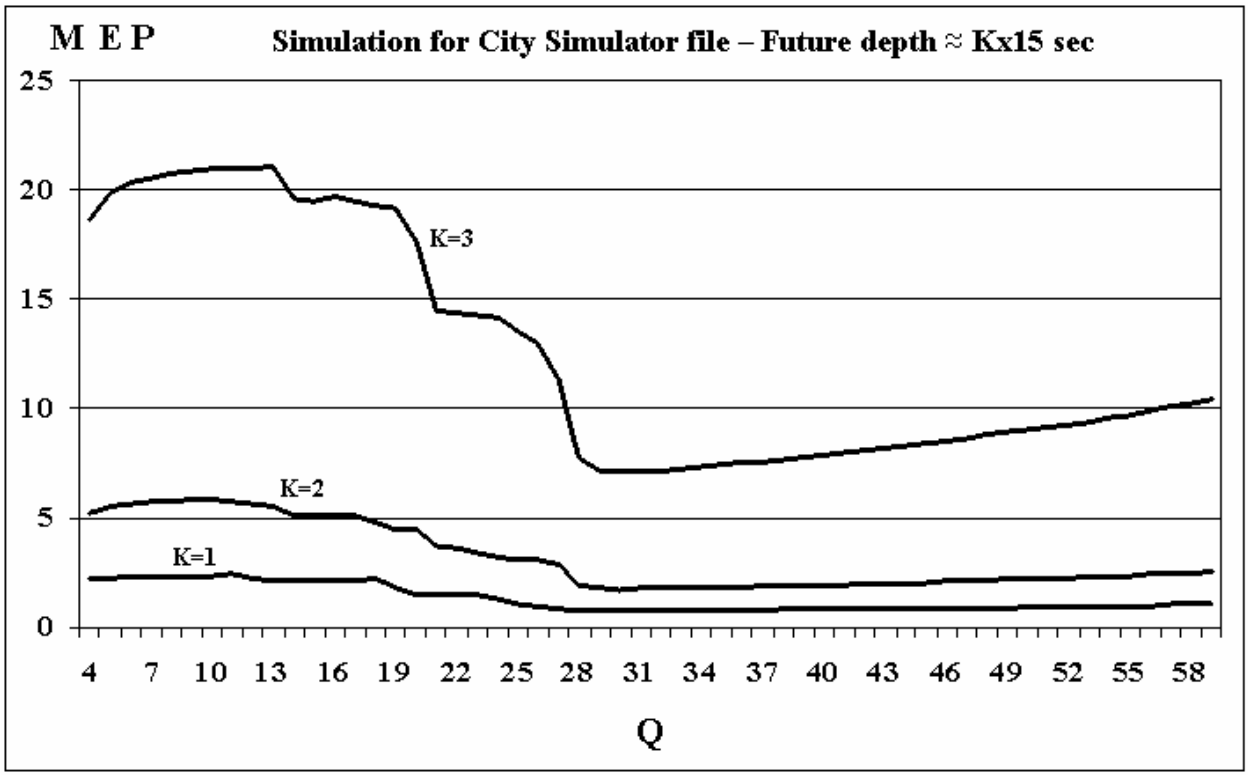
Αντίστοιχα τρέχει η εφαρμογή και για τις άλλες δύο μορφές στιγμάτων θέτοντας αντί για simulationGpsNK

- simulationCitySimNK για GPS στίγματα και
- simulationSimpleNK για απλά στίγματα.

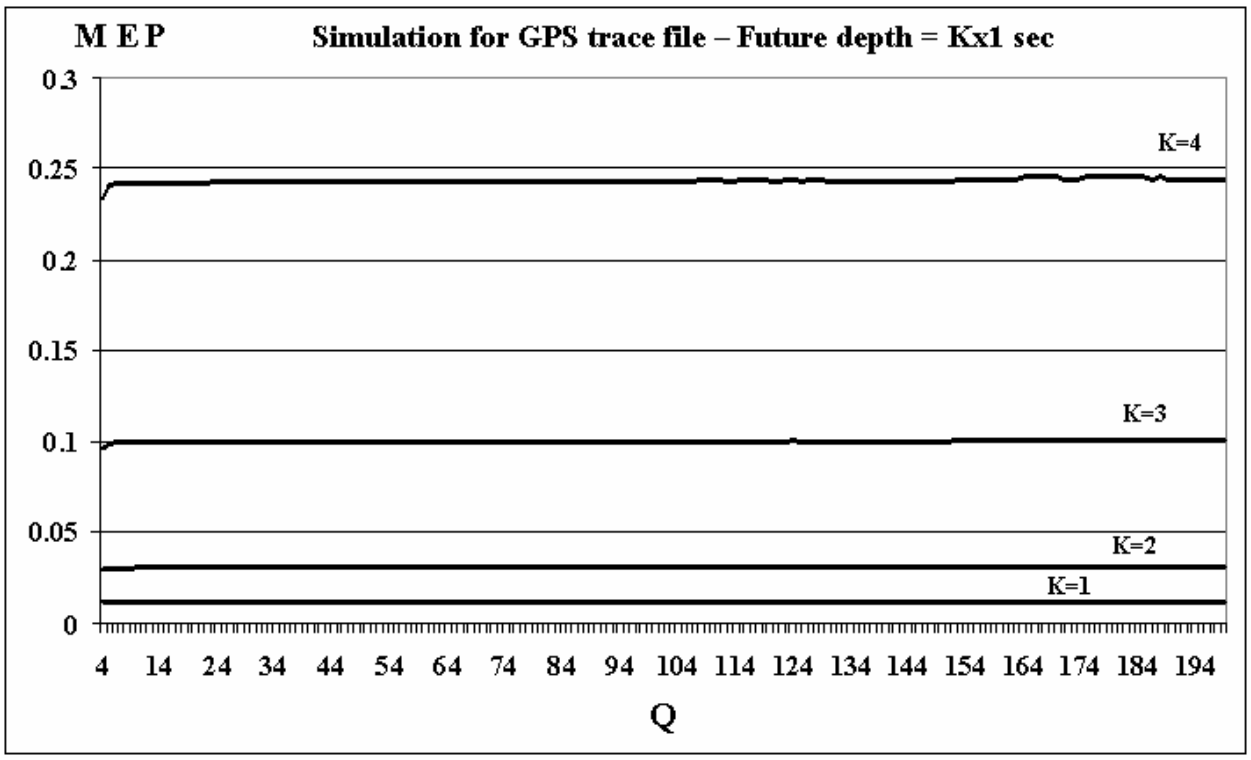
Τα αποτελέσματα της προσομοίωσης φαίνονται στα διαγράμματα που ακολουθούν:



Μέσο λάθος για την πρόβλεψη με Cubic Splines σε βάθος Κ απλών στιγμάτων



Μέσο λάθος για την πρόβλεψη με Cubic Splines σε βάθος Κ στιγμάτων City Simulator



Μέσο λάθος για την πρόβλεψη με Cubic Splines σε βάθος Κ GPS στιγμάτων

Παρατηρήσεις:

- Οι κυβικές Splines επιτυγχάνουν ιδιαίτερα χαμηλά ποσοστά αστοχίας και στα τρία είδη στιγμάτων που χρησιμοποιήθηκαν.
- Όσο και αν μεγαλώνει το QUEUE_LENGTH η ακρίβεια των προβλέψεών μας δεν γίνεται καλύτερη. Άρα και με σχετικά μικρό ιστορικό γνώσης μπορεί κανείς να προβλέπει την τοποθεσία ενός κινητού χρήστη με σχετικά καλή ακρίβεια.
- Από τα διαγράμματα συμπεραίνουμε πως για το GPS, για παράδειγμα, η προσομοίωσή μας πρόβλεψε τις θέσεις του χρήστη για τρία και πλέον δευτερόλεπτα στο μέλλον με σφάλμα κάτω του 0.5%.
- Ακόμη για τα στίγματα από τον City Simulator η προσομοίωση πρόβλεψε τις τοποθεσίες του χρήστη για 15 με 20 δευτερόλεπτα στο μέλλον με σφάλμα κάτω του 5%.
- Γνωρίζοντας ένα σύστημα με καλή ακρίβεια την τιμή ενός μεγέθους/παραμέτρου (π.χ. θέση) είναι ικανό να εκτιμήσει και τις τιμές άλλων μεγεθών τα οποία εξαρτώνται από το πρώτο (συμπερασμός). Για παράδειγμα ένας χρήστης κινητής συσκευής θα μπορούσε να ειδοποιείται πως στην κατεύθυνση στην οποία κινείται το εύρος ζώνης είναι χαμηλό - έστω BW=f(θέση) - και, επομένως, πρέπει να πράξει ανάλογα, ενώ η ίδια του η συσκευή πιθανόν να μειώσει τις δευτερεύουσες λειτουργίες της ώστε να εξοικονομήσει ενέργεια για την ώρα που το εύρος ζώνης θα πέφτει.

7. ΕΠΙΛΟΓΟΣ

Στην παρούσα εργασία μελετήσαμε την κομβική για τις σύγχρονες τεχνολογίες δικτύων και τηλεπικοινωνιών έννοια του πλαισίου (context). Το «πλαίσιο» περιγράφηκε, αναλύθηκε τόσο όσο αφορά το πόσο σημαντικό ρόλο καλείται να παίξει σε αυτό που σήμερα ονομάζουμε pervasive και ubiquitous computing όσο το τι απόπειρες έχουν γίνει ως τώρα στην κατεύθυνση αυτή.

Πρόκειται για ένα μεγάλο κεφάλαιο στη σύγχρονη έρευνα που φιλοδοξεί να φέρει μια νέα επανάσταση στην ίδια την αντίληψη του ανθρώπου απέναντι στον υπολογιστή και τις συσκευές που διαθέτουν υπολογιστικές ικανότητες. Αυτή η υπολογιστική ικανότητα διαπερνά την καθημερινότητά μας όλο και πιο πολύ και γι' αυτό η αλληλεπίδραση ανθρώπου μηχανής τείνει να εξελιχθεί σε μια όσο το δυνατόν πιο διαφανή και φυσική διαδικασία.

Στην πράξη είδαμε τρόπους πρόβλεψης πληροφορίας πλαισίου και εφαρμόσαμε δικές μας μεθόδους, δανεισμένες από τα μαθηματικά (παρεμβολή) για να χτίσουμε μια προσομοίωση που ερευνά μια πτυχή του χωρικού πλαισίου. Είδαμε πως με τη χρήση κυβικών Splines η πρόβλεψη του χωρικού πλαισίου είναι δυνατή. Μπορεί να βελτιωθεί σημαντικά, ακόμη, αν οι μαθηματικές μέθοδοι που χρησιμοποιήσαμε μπορέσουν να αποκτήσουν μεγαλύτερη προσαρμοστικότητα στις κινήσεις των χρηστών.

ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] “Numerical Methods for Engineers”, Steven C. Chapra, Raymond P. Canale, McGraw-Hill International Editions

[2] «Εισαγωγή στην Αριθμητική Ανάλυση», Νικόλαος Μισυρλής, Πανεπιστήμιο Αθηνών, Τμήμα Πληροφορικής

[3] «Γραμμική Άλγεβρα και Αναλυτική Γεωμετρία», Θανάσης Χρυσάκης, επίκουρος καθηγητής Πανεπιστημίου Αθηνών, Αθήνα 1992

[4] «Μαθήματα Γενικών Μαθηματικών», Λεωνίδας Τσίτσας, Εκδόσεις Συμμετρία, Αθήνα 1993

[5] «Απειροστικός Λογισμός», Σ. Νεγρεπόντης, Σ. Γιωτόπουλος, Ε. Γιαννακούλιας, Εκδόσεις Αίθρα, Αθήνα 1995

[6] «Πανεπιστημιακή Φυσική», Hugh D. Young, Εκδόσεις Παπαζήση

[7] “Mutli-Sensor Context Awareness in Mobile devices and smart Artifacts”, Hans Gellersen, Albrecht Schmidt (Lancaster University, Department of Computing), Michael Beigl (University of Karlsruhe)

[8] “Global and Local State Prediction”, Jan Petzold, Faruk Bagci, Wolfgang Trumbler, Theo Ungerer, University of Augsburg, Institute of Computer Science

[9] “An Architecture for Context Prediction”, Rene Mayerhofer, Institut fuer Pervasive Computing, Johannes Kepler Universitaet Linz

[10] “Introduction on Contex-Aware Computing”, Thomas P. Moran (IBM Almaden Ressearch Center), Paul Dourish (University of California, Irvine)

[11] “A Framework for Resource Discovery in Pervasive Computing for Mobile Aware Task Execution”, K Kalapriya, S.K. Nandy, Deepti Srinivasan, R. Uma Maheshwari, Satish. V (Indian Institute of Science),

[12] “Prediction Intelligence in Context Aware Applications”, Anagnostopoulos, Mpougiouris, Hadjieythymiades, University of Athens, Department of Informatics

[13] «Η χρήση των υπηρεσιών δικτύου για την παροχή υπηρεσιών που εξαρτώνται από τη γεωγραφική θέση», Κώστας Πολυχρονόπουλος, Πανεπιστήμιο Αθηνών, Τμήμα Πληροφορικής

[14] «Ανάπτυξη OSA/Parlay συμβατού προσομοιωτή κίνησης τερματικών σταθμών», Χρήστος Αναγνωστόπουλος, Πανεπιστήμιο Αθηνών, Τμήμα Πληροφορικής

[15] “Context-Awareness: some Compelling Applocations”, Peter Brown (University of Exeter), Winslow Burleson (MIT Media Lab), Mik Lamming, Dave Snowdon (Xerox Research Center Europe), Odd-Wiking Rahlff (SINTEF), Guy Romano (Motorola labs Ilinois), Jean Scholtz (DARPA)

[16] “Context-Aware Computing Applications”, Bill Schilit, Norman Adams, Roy Want - IEEE workshop on mobile computing systems and applications Dec 9,9 1994

[17] “Towards Context-Aware Adaptable Web Services”, Markus Keidl, Alfons Kemper, (Universitaet Passau, Germany)

[18] “Precision Agriculture - Untangling the GPS data string”, Viacheslav Adamchuk, Biological Systems Engineer, University of Nebraska, Lincoln

[19] www.ibiblio.org - The public's library and digital archive. - An Interactive Introduction to Splines

[20] www.citeseer.com - Scientific Literature Digital Library

[21] en.wikipedia.org/wiki/Main_Page - WikipediA, The Free Encyclopedia

[22] www.euclideanspace.com - 3D World Simulation - Martin John Baker

[23] www.gpsworld.com - The GPS magazine

[24] www.allgps.com - GPS Portal: Global Positioning System introduction and services

[25] igscb.jpl.nasa.gov - International GPS Service

[26] openmap.bbn.com - OpenMap 4.6 - Open Systems Mapping Technology - BBN Technologies

[27] www.chaeron.com - Enterprise System Solutions - Chaeron GPS Java Library 1.02)

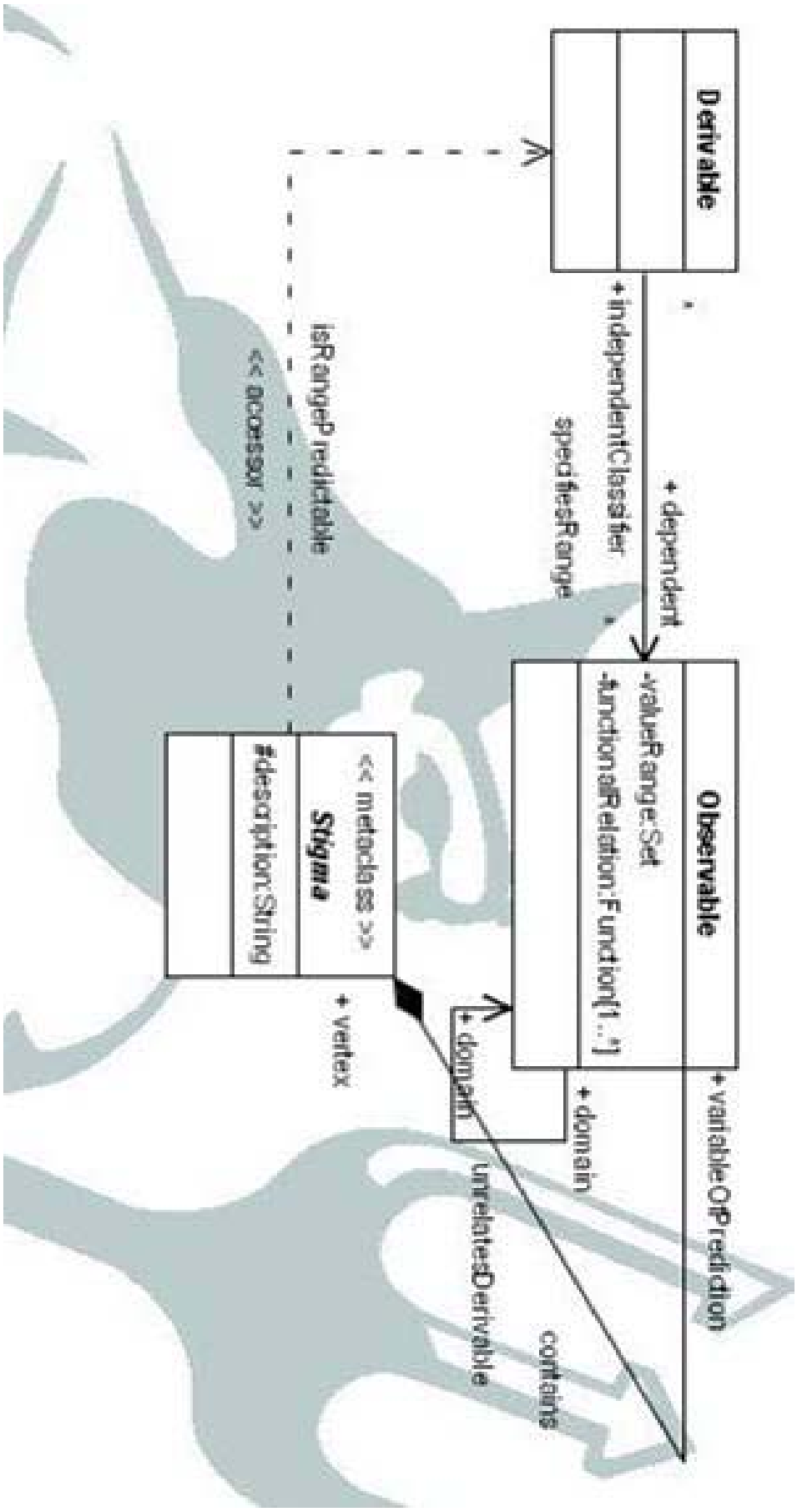
[28] www.whereonearth.com - L-Sphere: Location Based Services Technology Platform

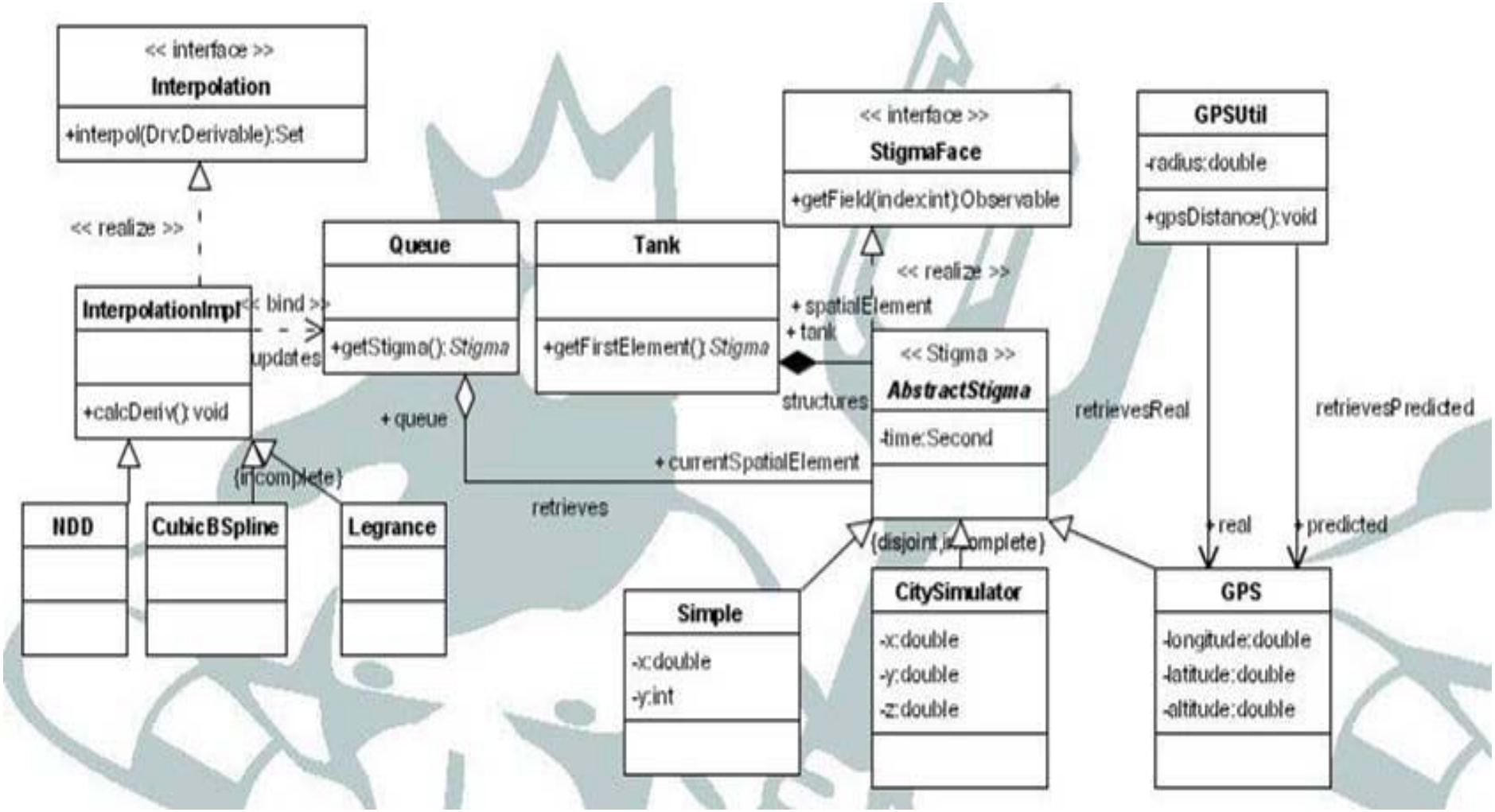
[29] UML Toolkit - Hans-Erik Eriksson, Magnus Penker - Wiley Computer Publishing

[30] JavaTM 2 Platform,Standard Edition, v 1.3 API Specification, Java Technology - java.sun.com - The source for developers (Sun's developer network site)

[31] Michael Thomas Flanagan's JAVA library on-line web page at www.ee.ucl.ac.uk/~mflanaga

ΠΑΡΑΡΤΗΜΑ Α: ΔΙΑΓΡΑΜΜΑΤΑ ΥΛΟΠΟΙΗΣΗΣ ΣΕ UML




<< interface >>
Interpolation
+interpol(Drv:Derivable):Set
<< realize >>
InterpolationImpl
+calcDeriv():void
<< bind >>
updates
Queue
+getStigma():Stigma
Tank
+getFirstElement():Stigma
+ queue
retrieves
{incomplete}
NDD
CubicBSpline
Legrance
<< interface >>
StigmaFace
+getField(index:int):Observable
<< realize >>
+ spatialElement
+ tank
structures
<< Stigma >>
AbstractStigma
-time:Second
+currentSpatialElement
{disjoint,incomplete}
Simple
-x:double
-y:int
CitySimulator
-x:double
-y:double
-z:double
GPSUtil
-radius:double
+gpsDistance():void
retrievesReal
retrievesPredicted
+real
+predicted
GPS
-longitude:double
-latitude:double
-altitude:double

ΠΑΡΑΡΤΗΜΑ Β: ΠΗΓΑΙΟΣ ΚΩΔΙΚΑΣ

Queue.java

```
package lbs.util;

import java.util.*;

/* Boithitikh klash pou ylopoiei mia oura proteraiothtas me stathero megisto megethos
*/
public class Queue {
	private Vector data;
	private int capacity;

	public Queue(int capacity, int dataIncrement) {
		this.capacity = capacity;
		this.data = new Vector(this.capacity, dataIncrement);
	}

	public Queue() {
		this.capacity = 4;
		this.data = new Vector(this.capacity, 1);
	}

	public synchronized int getCapacity() {
		return this.capacity;
	}

	public synchronized int getElementCount() {
		return this.data.size();
	}

	public synchronized boolean isFull() {
		return (this.data.size() == this.capacity);
	}

	public synchronized boolean isEmpty() {
		return (this.data.size() == 0);
	}

	public synchronized boolean add(Object newElement) {
		if (this.isFull()) {
			int i;
			for (i=0; i<this.capacity-1; i++) {
				this.data.set(i, this.data.elementAt(i+1));
			}
			this.data.set(i, newElement);
			return false;
		} else {
			this.data.add(newElement);
			return true;
		}
	}

	public synchronized Object elementAt(int index) {
		if (this.isEmpty()) {
			return null;
		}
		try {
			Object answer = data.elementAt(index);
			return answer;
		} catch (NoSuchElementException nsee) {
			System.out.println("Queue: getNextElement() -> " +
nsee.getMessage());
			nsee.printStackTrace();
			return null;
		} catch (ArrayIndexOutOfBoundsException aioob) {
			System.out.println("Queue: getNextElement() -> " +
aioob.getMessage());
			aioob.printStackTrace();
			return null;
		}
	}

	public synchronized Object [] getAllElements() {
```

```
                Object [] objects = new Object[this.data.size()];
                for (int i=0; i<this.data.size(); i++) {
                        objects[i] = this.data.elementAt(i);
                }
                return objects;
        }
}
```

Tank.java

```
package lbs.util;

import java.util.*;

/* Boithtikh klash pou ylopoiei mia stoiba dedomenwn */
public class Tank {
        private Vector data;

        public Tank() {
                data = new Vector();
        }

        public synchronized int getElementCount() {
                return data.size(); }

        public synchronized boolean isEmpty() {
                return (data.size() == 0);
        }

        public synchronized void add(Object newElement) {
                data.add(newElement);
        }

        public synchronized Object getFirstElement() {
                if (this.isEmpty()) {
                        return null;
                }
                try {
                        Object answer = data.elementAt(0);
                        data.remove(0);
                        return answer;
                } catch (NoSuchElementException nsee) {
                        System.out.println("Tank: getNextElement() -> " +
nsee.getMessage());
                        nsee.printStackTrace();
                        return null;
                } catch (ArrayIndexOutOfBoundsException aioob) {
                        System.out.println("Tank: getNextElement() -> " +
aioob.getMessage());
                        aioob.printStackTrace();
                        return null;
                }
        }

        public synchronized Object [] getAllElements() {
                Object [] objects = new Object[this.getElementCount()];
                for (int i=0; i<this.getElementCount(); i++) {
                        objects[i] = data.elementAt(i);
                }
                return objects;
        }
}
```

Stigma.java

```
package lbs.util;

/* Kathorizei thn ennoia tou stigmatos san ontothta pou diathetei "pedia" */
public interface Stigma {
        public Object getField(int position);
}
```

AbstractStigma.java

```
package lbs.util;
```

```
/* Abstract klash goneas twn stigmatwn - Kathorizei tis koines idiothtes olwn twn
eidwn twn stigmatwn */
public abstract class AbstractStigma implements Stigma {
	protected Object [] fields;

	public AbstractStigma(int fieldCount) {
		fields = new Object[fieldCount];
	}

	public AbstractStigma(Object [] fields) {
		this.fields = new Object[fields.length];
		for (int i=0; i<fields.length; i++) {
			this.fields[i] = fields[i];
		}
	}

	public Object getField(int position) {
		if (position>this.fields.length) {
			return null;
		} else {
			return this.fields[position];
		}
	}

	public String toString() {
		String answer = new String();
		for (int i=0; i<this.fields.length; i++) {
			answer += this.fields[i] + " ";
		}
		return answer;
	}
}
```

SimpleStigma.java

```
package lbs.util;

/* H aplousterh morfh stigmatos me ta parakatw 4 pedia */
public class SimpleStigma extends AbstractStigma {
	public static final int ID = 0;
	public static final int TIME = 1;
	public static final int X = 2;
	public static final int Y = 3;

    public SimpleStigma(int ID, float time, float x, float y) {
		super(4);
		this.fields[this.ID] = new Integer(ID);
		this.fields[this.TIME] = new Double(time);
		this.fields[this.X] = new Double(x);
		this.fields[this.Y] = new Double(y);
	}

	public SimpleStigma(String fileLine) {
		super(4);
		int to, from;
		from = 0;
		to = fileLine.indexOf(",", from);
		this.fields[this.ID] = new Integer(fileLine.substring(from, to));
		from = to+1;
		to = fileLine.indexOf(",", from);
		this.fields[this.TIME] =  new Double(fileLine.substring(from, to));
		from = to+1;
		to = fileLine.indexOf(",", from);
		this.fields[this.X] = new Double(fileLine.substring(from, to));
		from = to+1;
		to = fileLine.indexOf(",", from);
		this.fields[this.Y] = new Double(fileLine.substring(from, to));
	}
}
```

CitySimStigma.java

```
package lbs.util;

/* Stigma montelopoihmeno opws auto dinetai apo ton City Simulator */
public class CitySimStigma extends AbstractStigma {
	public static final int ID = 0;
```

```
        public static final int TIME = 1;
        public static final int X = 2;
        public static final int Y = 3;
        public static final int Z = 4;

        public CitySimStigma(int ID, float time, float x, float y, float z) {
                super(5);
                this.fields[ID] = new Integer(ID);
                this.fields[TIME] = new Double(time);
                this.fields[X] = new Double(x);
                this.fields[Y] = new Double(y);
                this.fields[Z] = new Double(z);
        }

        public CitySimStigma(String fileLine) {
                super(5);
                int to, from;
                from = 0;
                to = fileLine.indexOf(",", from);
                this.fields[ID] = new Integer(fileLine.substring(from, to));
                from = to+1;
                to = fileLine.indexOf(",", from);
                this.fields[TIME] =  new Double(fileLine.substring(from, to));
                from = to+1;
                to = fileLine.indexOf(",", from);
                this.fields[X] = new Double(fileLine.substring(from, to));
                from = to+1;
                to = fileLine.indexOf(",", from);
                this.fields[Y] = new Double(fileLine.substring(from, to));
                from = to+1;
                to = fileLine.indexOf(" ", from);
                this.fields[Z] = new Double(fileLine.substring(from, to));
        }
}
```

GpsStigma.java

```
package lbs.util;

/* Stigma GPS (GPGGA) me ta kyriotera pedia pou einai anagaia */
public class GpsStigma extends AbstractStigma {
        public static final int TYPE = 0;
        public static final int TIME = 1;
        public static final int LATITUDE = 2;
        public static final int LONGTITUDE = 3;
        public static final int ALTITUDE = 4;

        public GpsStigma(String type, float time, long latitude, long longtitude, long
altitude) {
                super(5);
                this.fields[TYPE] = new String(type);
                this.fields[TIME] = new Double(time);
                this.fields[LATITUDE] = new Double(latitude);
                this.fields[LONGTITUDE] = new Double(longtitude);
                this.fields[ALTITUDE] = new Double(altitude);
        }

        public GpsStigma(String fileLine) {
                super(5);
                int to, from;
                from = 0;
                to = fileLine.indexOf(",", from);
                this.fields[TYPE] = new String(fileLine.substring(from, to));
                from = to+1;
                to = fileLine.indexOf(",", from);
         String timeStr = fileLine.substring(from, to);
         int hours = new Integer(""+timeStr.charAt(0)+timeStr.charAt(1)).intValue();
         int minutes = new Integer(""+timeStr.charAt(2)+timeStr.charAt(3)).intValue();
         double seconds = new Double(timeStr.substring(4)).doubleValue();
                this.fields[TIME] =  new Double(hours*3600.0+minutes*60.0+seconds);
                from = to+1;
                to = fileLine.indexOf(",", from);
         String latitudeStr = fileLine.substring(from, to);
         int degrees = new
Integer(""+latitudeStr.charAt(0)+latitudeStr.charAt(1)).intValue();
         minutes = new
Integer(""+latitudeStr.charAt(2)+latitudeStr.charAt(3)).intValue();
```

```
        seconds = new Double("0"+latitudeStr.substring(4)).doubleValue();
                this.fields[LATITUDE] = new
Double(degrees*3600.0+minutes*60.0+seconds);
                from = to+1;
                to = fileLine.indexOf(",", from);
                from = to+1;
                to = fileLine.indexOf(",", from);
        String longtitudeStr = fileLine.substring(from, to);
        degrees = new
Integer(""+longtitudeStr.charAt(0)+longtitudeStr.charAt(1)+longtitudeStr.charAt(2)).in
tValue();
        minutes = new
Integer(""+longtitudeStr.charAt(3)+longtitudeStr.charAt(4)).intValue();
        seconds = new Double("0"+longtitudeStr.substring(5)).doubleValue();
                this.fields[LONGTITUDE] = new
Double(degrees*3600.0+minutes*60.0+seconds);
                from = to+1;
                to = fileLine.indexOf(",", from);
                from = to+1;
                to = fileLine.indexOf(",", from);
                from = to+1;
                to = fileLine.indexOf(",", from);
                from = to+1;
                to = fileLine.indexOf(",", from);
                from = to+1;
                to = fileLine.indexOf(",", from);
                this.fields[ALTITUDE] = new Double(fileLine.substring(from, to));
        }
}
```

GpsUtil.java

```
package lbs.util;

/* Klash boithitikh gia ton ypologismo tou granularity sta predictions twn GPS
stigmatwn */
public class GpsUtil {
    private static final double a = 6378137.0;
    private static final double b = 6356752.3;

    public static double degreesToRadians(double degrees) {
        double radians = 0.0;
        radians = (Math.PI*degrees)/180.0;
        return radians;
    }

    public static double secondsToDegrees(double seconds) {
        double degrees = 0.0;
        degrees = seconds/3600.0;
        return degrees;
    }

    public static double gpsDistance(GpsStigma stigma, double predLat, double
predLong) {
        double lat1 =
degreesToRadians(secondsToDegrees(((Double)stigma.getField(GpsStigma.LATITUDE)).double
Value()));
        double long1 =
degreesToRadians(secondsToDegrees(((Double)stigma.getField(GpsStigma.LONGTITUDE)).doub
leValue()));
        double lat2 = degreesToRadians(secondsToDegrees(predLat));
        double long2 = degreesToRadians(secondsToDegrees(predLong));
        double t1 = Math.sin(lat1)*Math.sin(lat2);
        double t2 = Math.cos(lat1)*Math.cos(lat2);
        double t3 = Math.cos(long1-long2);
        double t4 = t2*t3;
        double t5 = t1+t4;
        double rad_dist = Math.atan(-t5/Math.sqrt(-t5*t5+1))+2*Math.atan(1);
        //System.out.println(rad_dist*1000*1.6093470878864446*3437.74677*1.1508);
        return rad_dist*1000*1.6093470878864446*3437.74677*1.1508;
    }
}
```

Interpolation.java

```
package lbs.interpol;
```

```
/* Interface pou kathorizei tis methodous paremvolhs */
public interface Interpolation {
	public double interpolate(double xValue);
}
```

NewtonDividedDifferences.java

```
package lbs.interpol;

/* Parembolh Nweton me Dihrhmenes Diafores */
public class NewtonDividedDifferences implements Interpolation {
	private double [] x;
	private double [] y;
	private double [][] NDD;
	private int classSize;

	public NewtonDividedDifferences(double [] x, double [] y) {
		int n = x.length;
		if (n!=y.length) {
			throw new IllegalArgumentException("Arrays x and y are of
different length");
		}

		this.classSize = n;
		this.x = new double[n];
		this.y = new double[n];
		this.NDD = new double[n][];
		for (int i=0; i<n; i++) {
			this.x[i] = x[i];
			this.y[i] = y[i];
			this.NDD[i] = new double[n-i];
		}

		for (int i=0; i<n; i++) {
			NDD[i][0] = this.y[i];
		}
		for (int i=1; i<n; i++) {
			for (int k=0; k<n-i; k++) {
				NDD[k][i] = (NDD[k+1][i-1]-NDD[k][i-1]) / (x[k+i]-x[k]);
			}
		}
	}

	public double interpolate(double xValue) {
		double answer = 0.0;
		double factor = 1.0;
		for (int i=0; i<classSize; i++) {
			factor = 1.0;
			for (int k=0; k<i; k++) {
				factor *= (xValue - x[k]);
			}
			answer += factor * NDD[0][i];
		}
		return answer;
	}
}
```

Lagrange.java

```
package lbs.interpol;

/* H parembolh Lagrange */
public class Lagrange implements Interpolation {
	private double [] x;
	private double [] y;
	private int n;

	public Lagrange(double x[], double y[]) {
		this.n = x.length;
		if (this.n!=y.length) {
			throw new IllegalArgumentException("Arrays x and y are of
different length");
		}

		this.x = new double[this.n];
		this.y = new double[this.n];
		for (int i=0; i<this.n; i++) {
```

```
                        this.x[i] = x[i];
                        this.y[i] = y[i];
                }
        }

        public double interpolate(double x) {
                double lambda;
                double y = 0.0;
                int i,j;
                for( i=1; i <= this.n; i++ ) {
                  lambda = 1.0;
                  for ( j=1; j<=this.n; j++ )
                        if (i != j)
                          {lambda *= ((x - this.x[j-1])/(this.x[i-1] - this.x[j-1]));}
                   y += (this.y[i-1] * lambda);
                }
                return y;
        }
}
```

CubicSpline.java

```
package lbs.interpol;

/* Cubic BSpline interpolation - afou arxikoipoiithei thelei na klithei kai h
calcDeriv() */
public class CubicSpline implements Interpolation {

        private int n = 0;
        private double[] y = null;
        private double[] x = null;
        private double[] y2 = null;  // returned second derivatives of y
        private double yp1 = 0.0D;  // first derivative at point one, default value =
zero (natural spline)
        private double ypn = 0.0D;  // first derivative at point n, default value =
zero (natural spline)

        public CubicSpline(double[] x, double[] y) {
                this.n=x.length;
                if(this.n!=y.length) {
                        throw new IllegalArgumentException("Arrays x and y are of
different length");
                }
                this.x = new double[this.n];
                this.y = new double[this.n];
                this.y2 = new double[this.n];
                for(int i=0; i<this.n; i++){
                                this.x[i]=x[i];
                                this.y[i]=y[i];
                }
                this.yp1=1e40;
                this.ypn=1e40;
        }

        public void calcDeriv() {
                double  p=0.0D,qn=0.0D,sig=0.0D,un=0.0D;
                double[] u = new double[this.n];

                if (yp1 > 0.99e30) {
                        y2[0]=u[0]=0.0;
                } else {
                this.y2[0] = -0.5;
                        u[0]=(3.0/(this.x[1]-this.x[0]))*((this.y[1]-
this.y[0])/(this.x[1]-this.x[0])-this.yp1);
                }

                for(int i=1;i<=this.n-2;i++) {
                        sig=(this.x[i]-this.x[i-1])/(this.x[i+1]-this.x[i-1]);
                        p=sig*this.y2[i-1]+2.0;
                        this.y2[i]=(sig-1.0)/p;
                        u[i]=(this.y[i+1]-this.y[i])/(this.x[i+1]-this.x[i]) -
(this.y[i]-this.y[i-1])/(this.x[i]-this.x[i-1]);
                        u[i]=(6.0*u[i]/(this.x[i+1]-this.x[i-1])-sig*u[i-1])/p;
                }

                if (this.ypn > 0.99e30) {
                        qn=un=0.0;
```

```
                } else {
                        qn=0.5;
                        un=(3.0/(this.x[this.n-1]-this.x[this.n-2]))*(this.ypn-
(this.y[this.n-1]-this.y[this.n-2])/(this.x[this.n-1]-x[this.n-2]));
                }

                this.y2[this.n-1]=(un-qn*u[this.n-2])/(qn*this.y2[this.n-2]+1.0);
                for(int k=this.n-2;k>=0;k--) {
                        this.y2[k]=this.y2[k]*this.y2[k+1]+u[k];
                }
        }

        public double interpolate(double xx) {
                double h=0.0D,b=0.0D,a=0.0D, yy=0.0D;
                /*if (xx<this.x[0] || xx>this.x[this.npoints-1]){
             throw new IllegalArgumentException("x is outside the range of data
points");
                }*/
                int k=0;
                int klo=0;
                int khi=this.n-1;
                while (khi-klo > 1) {
                        k=(khi+klo) >> 1;
                        if(this.x[k] > xx){
                                khi=k;
                        }
                        else{
                                klo=k;
                        }
                }
                h=this.x[khi]-this.x[klo];

                if (h == 0.0) {
                        throw new IllegalArgumentException("Two identical values of x:
"+klo+" ("+this.x[klo]+") and "+khi+"("+this.x[khi]+")" );
                } else {
                        a=(this.x[khi]-xx)/h;
                        b=(xx-this.x[klo])/h;
                        yy=a*this.y[klo]+b*this.y[khi]+((a*a*a-a)*this.y2[klo]+(b*b*b-
b)*this.y2[khi])*(h*h)/6.0;
                }
                return yy;
        }
}
```

ReadSignals.java

```
package lbs.calculus;

/* Kathorizei th morfh twn klasewn pou diabazoun stigmata gia mobility prediction */
public interface ReadSignals {
    public void readSignals();
}
```

ReadSignalsSimple.java

```
package lbs.calculus;

import java.io.*;
import java.util.*;

import lbs.util.Tank;
import lbs.util.Queue;
import lbs.util.SimpleStigma;

import lbs.interpol.NewtonDividedDifferences;
import lbs.interpol.CubicSpline;
import lbs.interpol.Lagrange;

/* Diavazei stigmata morfhs Simple apo arxeio kai prosomoiwnei diadikasia mobility
prediction bgazontas ektenh apotelesmata se arxeio */
public class ReadSignalsSimple implements ReadSignals {
    private int queueSize;
    private double timeStep;
    private double granularity;
    private File fileIn;
    private File fileOut;
```

```
    private long time;
    private boolean exhaustive;

    public ReadSignalsSimple(File fileIn, File fileOut, int queueSize, double
timeStep, double granularity, boolean exhaustive) {
        this.queueSize = queueSize;
        this.timeStep = timeStep;
        this.granularity = granularity;
        this.fileIn = fileIn;
        this.fileOut = fileOut;
        this.exhaustive = exhaustive;
    }

    public void readSignals() {
        int userID = 0;
        Tank tank = new lbs.util.Tank();
        SimpleStigma stigma = null;
        time = new Date().getTime();
        try {
            if (!exhaustive) {
                System.out.println(" > SIMPLE start...");
            }
            BufferedReader in = new BufferedReader(new InputStreamReader(new
FileInputStream(fileIn)));
            in.readLine();
            in.readLine();
            String read;
            boolean firsttime = true;
            userID = 0;
            while ((read=in.readLine()) != null) {
                stigma = new SimpleStigma(read);
                if (firsttime) {
                    tank.add(stigma);
                    userID = ((Integer)stigma.getField(SimpleStigma.ID)).intValue();
                    firsttime = false;
                } else {
                    if (((Integer)stigma.getField(SimpleStigma.ID)).intValue() ==
userID) {
                        tank.add(stigma);
                    }
                }
            }
            in.close();
        } catch (Exception e) {
            System.out.println(" > COULD NOT COMPLETE SIMULATION!");
            System.out.println(" > Possible reason: Input file invalid format... Stack
Trace follows:");
            e.printStackTrace();
            return;
        }
        try {
            PrintWriter pw = null;
            if (!exhaustive) {
                System.out.println(" > Exporting results to
"+fileOut.getAbsolutePath());
                pw = new PrintWriter(new FileOutputStream(fileOut));
            }
            Queue tQ, xQ, yQ;
            tQ = new Queue(queueSize, 1);
            xQ = new Queue(queueSize, 1);
            yQ = new Queue(queueSize, 1);
            double nddXPrediction = 0.0;
            double nddYPrediction = 0.0;
            double csXPrediction = 0.0;
            double csYPrediction = 0.0;
            double lgXPrediction = 0.0;
            double lgYPrediction = 0.0;
            int counter = 0;
            double granularityProximity = 0.0;
            String success = "SUCCESS";
            String failure="FAILURE";
            double nddError = 0.0;
            double csError = 0.0;
            double lgError = 0.0;
            int nddSuccess = 0;
            int csSuccess = 0;
            int lgSuccess = 0;
```

```
            int totalPredictions = 0;

            if (!exhaustive) {
                pw.println("        MOBILITY SIMULATION RESULTS ("+new Date()+")");
                pw.println("DATA_MODE=SIMPLE");
                pw.println("INPUT_FILE="+fileIn.getAbsolutePath());
                pw.println("QUEUE_SIZE="+queueSize);
                pw.println("TIME_STEP="+timeStep);
                pw.println("GRANULARITY="+granularity);
                pw.println("N.D.D=NEWTON DIVIDED DIFFERENCES, CbSpl=CUBIC SPLINES,
Lgrng=LAGRANGE");
                pw.println("stigma = t - x(t), y(t)");
                pw.println("Chose user: " + userID + " for the simulation...");
                pw.println();
            }
            while (!tank.isEmpty()) {
                counter++;
                stigma = (SimpleStigma)tank.getFirstElement();
                if (!exhaustive) {
                    pw.println("stigma: "+stigma.getField(SimpleStigma.TIME)+" -
"+stigma.getField(SimpleStigma.X)+","+stigma.getField(SimpleStigma.Y));
                }
                tQ.add(stigma.getField(SimpleStigma.TIME));
                xQ.add(stigma.getField(SimpleStigma.X));
                yQ.add(stigma.getField(SimpleStigma.Y));
                if (counter<queueSize) {
                    if (!exhaustive) {
                        pw.println("    Waiting for queue to fill with data...");
                    }
                    continue;
                }
                double [] t = new double[queueSize];
                double [] x = new double[queueSize];
                double [] y = new double[queueSize];
                for (int i=0; i<queueSize; i++) {
                    t[i] = ((Double)tQ.elementAt(i)).doubleValue();
                    x[i] = ((Double)xQ.elementAt(i)).doubleValue();
                    y[i] = ((Double)yQ.elementAt(i)).doubleValue();
                }
                if (counter>queueSize) {
                    totalPredictions++;
                    nddError +=
Math.abs(100*(((Double)stigma.getField(SimpleStigma.X)).doubleValue()-
nddXPrediction)/((Double)stigma.getField(SimpleStigma.X)).doubleValue());
                    nddError +=
Math.abs(100*(((Double)stigma.getField(SimpleStigma.Y)).doubleValue()-
nddYPrediction)/((Double)stigma.getField(SimpleStigma.Y)).doubleValue());
                    granularityProximity = calculateGranularity(stigma,
nddXPrediction, nddYPrediction);
                    if (granularityProximity>=0.0) {
                      nddSuccess++;
                    }
                    if (!exhaustive) {
                        pw.println("Error N.D.D
"+100*(((Double)stigma.getField(SimpleStigma.X)).doubleValue()-
nddXPrediction)/((Double)stigma.getField(SimpleStigma.X)).doubleValue()+"%"+

","+100*(((Double)stigma.getField(SimpleStigma.Y)).doubleValue()-
nddYPrediction)/((Double)stigma.getField(SimpleStigma.Y)).doubleValue()+"%"+
                                                    "
"+(granularityProximity>=0.0?success:failure));
                    }
                    csError +=
Math.abs(100*(((Double)stigma.getField(SimpleStigma.X)).doubleValue()-
csXPrediction)/((Double)stigma.getField(SimpleStigma.X)).doubleValue());
                    csError +=
Math.abs(100*(((Double)stigma.getField(SimpleStigma.Y)).doubleValue()-
csYPrediction)/((Double)stigma.getField(SimpleStigma.Y)).doubleValue());
                    granularityProximity = calculateGranularity(stigma, csXPrediction,
csYPrediction);
                    if (granularityProximity>=0.0) {
                      csSuccess++;
                    }
                    if (!exhaustive) {
                        pw.println("      CbSpl
"+100*(((Double)stigma.getField(SimpleStigma.X)).doubleValue()-
csXPrediction)/((Double)stigma.getField(SimpleStigma.X)).doubleValue()+"%"+
```

```
","+100*(((Double)stigma.getField(SimpleStigma.Y)).doubleValue()-
csYPrediction)/((Double)stigma.getField(SimpleStigma.Y)).doubleValue()+"%"+
                                                "
"+(granularityProximity>=0.0?success:failure));
                    }
                    lgError +=
Math.abs(100*(((Double)stigma.getField(SimpleStigma.X)).doubleValue()-
lgXPrediction)/((Double)stigma.getField(SimpleStigma.X)).doubleValue());
                    lgError +=
Math.abs(100*(((Double)stigma.getField(SimpleStigma.Y)).doubleValue()-
lgYPrediction)/((Double)stigma.getField(SimpleStigma.Y)).doubleValue());
                    granularityProximity = calculateGranularity(stigma, lgXPrediction,
lgYPrediction);
                    if (granularityProximity>=0.0) {
                      lgSuccess++;
                    }
                    if (!exhaustive) {
                        pw.println("      Lgrng
"+100*(((Double)stigma.getField(SimpleStigma.X)).doubleValue()-
lgXPrediction)/((Double)stigma.getField(SimpleStigma.X)).doubleValue()+"%"+

","+100*(((Double)stigma.getField(SimpleStigma.Y)).doubleValue()-
lgYPrediction)/((Double)stigma.getField(SimpleStigma.Y)).doubleValue()+"%"+
                                                "
"+(granularityProximity>=0.0?success:failure));
                    }
                }
                NewtonDividedDifferences nddX = new NewtonDividedDifferences(t, x);
                NewtonDividedDifferences nddY = new NewtonDividedDifferences(t, y);
                CubicSpline csX = new CubicSpline(t, x);
                csX.calcDeriv();
                CubicSpline csY = new CubicSpline(t, y);
                csY.calcDeriv();
                Lagrange lgX = new Lagrange(t, x);
                Lagrange lgY = new Lagrange(t, y);
                nddXPrediction =
nddX.interpolate(((Double)stigma.getField(SimpleStigma.TIME)).doubleValue()+timeStep);
                nddYPrediction =
nddY.interpolate(((Double)stigma.getField(SimpleStigma.TIME)).doubleValue()+timeStep);
                csXPrediction =
csX.interpolate(((Double)stigma.getField(SimpleStigma.TIME)).doubleValue()+timeStep);
                csYPrediction =
csY.interpolate(((Double)stigma.getField(SimpleStigma.TIME)).doubleValue()+timeStep);
                lgXPrediction =
lgX.interpolate(((Double)stigma.getField(SimpleStigma.TIME)).doubleValue()+timeStep);
                lgYPrediction =
lgY.interpolate(((Double)stigma.getField(SimpleStigma.TIME)).doubleValue()+timeStep);
                if (!exhaustive) {
                    pw.println("Next(?) N.D.D: "+nddXPrediction+","+nddYPrediction);
                    pw.println("        CbSpl: "+csXPrediction+","+csYPrediction);
                    pw.println("        Lgrng: "+lgXPrediction+","+lgYPrediction);
                }
            }
            if (!exhaustive) {
                time = new Date().getTime() - time;
                pw.println("#");
                pw.println();
                pw.println("Mean error percentages");
                pw.println();
                pw.println("    N.D.D: "+nddError/((counter-queueSize)*2.0)+"%");
                pw.println();
                pw.println("    CbSpl: "+csError/((counter-queueSize)*2.0)+"%");
                pw.println();
                pw.println("    Lgrng: "+lgError/((counter-queueSize)*2.0)+"%");
                pw.println();
                pw.println("Successful predictions (out of "+totalPredictions+" tries)
for granularity="+granularity);
                pw.println();
                pw.println("    N.D.D: "+nddSuccess);
                pw.println();
                pw.println("    CbSpl: "+csSuccess);
                pw.println();
                pw.println("    Lgrng: "+lgSuccess);
                pw.println();
                pw.println("Running time: "+time+"ms");
                pw.close();
```

```
                System.out.println(" > Mean error percentages:");
                System.out.println(" >    N.D.D: "+nddError/((counter-
queueSize)*2.0)+"%");
                System.out.println(" >    CbSpl: "+csError/((counter-
queueSize)*2.0)+"%");
                System.out.println(" >    Lgrng: "+lgError/((counter-
queueSize)*2.0)+"%");
                System.out.println(" > Successes (out of "+totalPredictions+"
tries):");
                System.out.println(" >    N.D.D: "+nddSuccess);
                System.out.println(" >    CbSpl: "+csSuccess);
                System.out.println(" >    Lgrng: "+lgSuccess);
                System.out.println(" > SIMPLE end...");
            } else {
                System.out.println(nddSuccess+"\t"+csSuccess+"\t"+lgSuccess+"\t"+
                    nddError/((counter-queueSize)*2.0)+"\t\t"+csError/((counter-
queueSize)*2.0)+"\t\t"+lgError/((counter-queueSize)*2.0)+"\t\t");
            }
        } catch (Exception e) {
            System.out.println(" > COULD NOT COMPLETE SIMULATION!");
            System.out.println(" > Reason unpredicted... Stack trace follows:");
            e.printStackTrace();
            return;
        }
    }

    private double calculateGranularity(SimpleStigma stigma, double predX, double
predY) {
        double res = 0.0;
        double stigmaX = ((Double)stigma.getField(SimpleStigma.X)).doubleValue();
        double stigmaY = ((Double)stigma.getField(SimpleStigma.Y)).doubleValue();
        double predVector = Math.sqrt(Math.pow(stigmaX-predX, 2.0)+Math.pow(stigmaY-
predY, 2.0));
        res = granularity-predVector;
        return res;
    }
}
```

ReadSignalsCitySim.java

```
package lbs.calculus;

import java.io.*;
import java.util.*;

import lbs.util.Tank;
import lbs.util.Queue;
import lbs.util.Stigma;
import lbs.util.CitySimStigma;

import lbs.interpol.NewtonDividedDifferences;
import lbs.interpol.CubicSpline;
import lbs.interpol.Lagrange;

/* Diavazei stigmata morfhs City Simulator apo arxeio kai prosomoiwnei diadikasia
mobility prediction bgazontas ektenh apotelesmata se arxeio */
public class ReadSignalsCitySim implements ReadSignals {
    private int queueSize;
    private double timeStep;
    private double granularity;
    private File fileIn;
    private File fileOut;
    private long time;
    private boolean exhaustive;

    public ReadSignalsCitySim(File fileIn, File fileOut, int queueSize, double
timeStep, double granularity, boolean exhaustive) {
        this.queueSize = queueSize;
        this.timeStep = timeStep;
        this.granularity = granularity;
        this.fileIn = fileIn;
        this.fileOut = fileOut;
        this.exhaustive = exhaustive;
    }

    public void readSignals() {
        int userID = 0;
```

```
        Tank tank = new lbs.util.Tank();
        CitySimStigma stigma = null;
        time = new Date().getTime();
        try {
            if (!exhaustive) {
                System.out.println(" > CITY_SIMULATOR start...");
            }
            BufferedReader in = new BufferedReader(new InputStreamReader(new
FileInputStream(fileIn)));
            in.readLine();
            in.readLine();
            in.readLine();
            in.readLine();
            in.readLine();
            String read;
            boolean firsttime = true;
            userID = 0;
            while ((read=in.readLine()) != null) {
                stigma = new CitySimStigma(read);
                if (firsttime) {
                    tank.add(stigma);
                    userID = ((Integer)stigma.getField(CitySimStigma.ID)).intValue();
                    firsttime = false;
                } else {
                    if (((Integer)stigma.getField(CitySimStigma.ID)).intValue() ==
userID) {
                        tank.add(stigma);
                    }
                }
            }
            in.close();
        } catch (Exception e) {
            System.out.println(" > COULD NOT COMPLETE SIMULATION!");
            System.out.println(" > Possible reason: Input file invalid format... Stack
Trace follows:");
            e.printStackTrace();
            return;
        }
        try {
            PrintWriter pw = null;
            if (!exhaustive) {
                System.out.println(" > Exporting results to
"+fileOut.getAbsolutePath());
                pw = new PrintWriter(new FileOutputStream(fileOut));
            }
            Queue tQ, xQ, yQ, zQ;
            tQ = new Queue(queueSize, 1);
            xQ = new Queue(queueSize, 1);
            yQ = new Queue(queueSize, 1);
            zQ = new Queue(queueSize, 1);
            double nddXPrediction = 0.0;
            double nddYPrediction = 0.0;
            double nddZPrediction = 0.0;
            double csXPrediction = 0.0;
            double csYPrediction = 0.0;
            double csZPrediction = 0.0;
            double lgXPrediction = 0.0;
            double lgYPrediction = 0.0;
            double lgZPrediction = 0.0;
            int counter = 0;
            double granularityProximity = 0.0;
            String success = "SUCCESS";
            String failure="FAILURE";
            double nddError = 0.0;
            double csError = 0.0;
            double lgError = 0.0;
            int nddSuccess = 0;
            int csSuccess = 0;
            int lgSuccess = 0;
            int totalPredictions = 0;

            if (!exhaustive) {
                pw.println("        MOBILITY SIMULATION RESULTS ("+new Date()+")");
                pw.println("DATA_MODE=CITY_SIMULATOR");
                pw.println("INPUT_FILE="+fileIn.getAbsolutePath());
                pw.println("QUEUE_SIZE="+queueSize);
                pw.println("TIME_STEP="+timeStep);
```

```
                pw.println("GRANULARITY="+granularity);
                pw.println("N.D.D=NEWTON DIVIDED DIFFERENCES, CbSpl=CUBIC SPLINES,
Lgrng=LAGRANGE");
                pw.println("stigma = t - x(t), y(t), z(t)");
                pw.println("Chose user: " + userID + " for the simulation...");
                pw.println();
            }
            while (!tank.isEmpty()) {
                counter++;
                stigma = (CitySimStigma)tank.getFirstElement();
                if (!exhaustive) {
                    pw.println("stigma: "+stigma.getField(CitySimStigma.TIME)+" -
"+stigma.getField(CitySimStigma.X)+","+stigma.getField(CitySimStigma.Y)+","+stigma.get
Field(CitySimStigma.Z));
                }
                tQ.add(stigma.getField(CitySimStigma.TIME));
                xQ.add(stigma.getField(CitySimStigma.X));
                yQ.add(stigma.getField(CitySimStigma.Y));
                zQ.add(stigma.getField(CitySimStigma.Z));
                if (counter<queueSize) {
                    if (!exhaustive) {
                        pw.println("    Waiting for queue to fill with data...");
                    }
                    continue;
                }
                double [] t = new double[queueSize];
                double [] x = new double[queueSize];
                double [] y = new double[queueSize];
                double [] z = new double[queueSize];
                for (int i=0; i<queueSize; i++) {
                    t[i] = ((Double)tQ.elementAt(i)).doubleValue();
                    x[i] = ((Double)xQ.elementAt(i)).doubleValue();
                    y[i] = ((Double)yQ.elementAt(i)).doubleValue();
                    z[i] = ((Double)zQ.elementAt(i)).doubleValue();
                }
                if (counter>queueSize) {
                    totalPredictions++;
                    nddError +=
Math.abs(100*(((Double)stigma.getField(CitySimStigma.X)).doubleValue()-
nddXPrediction)/((Double)stigma.getField(CitySimStigma.X)).doubleValue());
                    nddError +=
Math.abs(100*(((Double)stigma.getField(CitySimStigma.Y)).doubleValue()-
nddYPrediction)/((Double)stigma.getField(CitySimStigma.Y)).doubleValue());
                    nddError +=
Math.abs(100*(((Double)stigma.getField(CitySimStigma.Z)).doubleValue()-
nddZPrediction)/((Double)stigma.getField(CitySimStigma.Z)).doubleValue());
                    granularityProximity = calculateGranularity(stigma,
nddXPrediction, nddYPrediction);
                    if (granularityProximity>=0.0) {
                      nddSuccess++;
                    }
                    if (!exhaustive) {
                        pw.println("Error N.D.D
"+100*(((Double)stigma.getField(CitySimStigma.X)).doubleValue()-
nddXPrediction)/((Double)stigma.getField(CitySimStigma.X)).doubleValue()+"%"+

","+100*(((Double)stigma.getField(CitySimStigma.Y)).doubleValue()-
nddYPrediction)/((Double)stigma.getField(CitySimStigma.Y)).doubleValue()+"%"+

","+100*(((Double)stigma.getField(CitySimStigma.Z)).doubleValue()-
nddZPrediction)/((Double)stigma.getField(CitySimStigma.Z)).doubleValue()+"%"+
                                                    "
"+(granularityProximity>=0.0?success:failure));
                    }
                    csError +=
Math.abs(100*(((Double)stigma.getField(CitySimStigma.X)).doubleValue()-
csXPrediction)/((Double)stigma.getField(CitySimStigma.X)).doubleValue());
                    csError +=
Math.abs(100*(((Double)stigma.getField(CitySimStigma.Y)).doubleValue()-
csYPrediction)/((Double)stigma.getField(CitySimStigma.Y)).doubleValue());
                    csError +=
Math.abs(100*(((Double)stigma.getField(CitySimStigma.Z)).doubleValue()-
csZPrediction)/((Double)stigma.getField(CitySimStigma.Z)).doubleValue());
                    granularityProximity = calculateGranularity(stigma, csXPrediction,
csYPrediction);
                    if (granularityProximity>=0.0) {
                      csSuccess++;
```

```
                    }
                    if (!exhaustive) {
                        pw.println("      CbSpl
"+100*(((Double)stigma.getField(CitySimStigma.X)).doubleValue()-
csXPrediction)/((Double)stigma.getField(CitySimStigma.X)).doubleValue()+"%"+

","+100*(((Double)stigma.getField(CitySimStigma.Y)).doubleValue()-
csYPrediction)/((Double)stigma.getField(CitySimStigma.Y)).doubleValue()+"%"+

","+100*(((Double)stigma.getField(CitySimStigma.Z)).doubleValue()-
csZPrediction)/((Double)stigma.getField(CitySimStigma.Z)).doubleValue()+"%"+
                                                "
"+(granularityProximity>=0.0?success:failure));
                    }
                    lgError +=
Math.abs(100*(((Double)stigma.getField(CitySimStigma.X)).doubleValue()-
lgXPrediction)/((Double)stigma.getField(CitySimStigma.X)).doubleValue());
                    lgError +=
Math.abs(100*(((Double)stigma.getField(CitySimStigma.Y)).doubleValue()-
lgYPrediction)/((Double)stigma.getField(CitySimStigma.Y)).doubleValue());
                    lgError +=
Math.abs(100*(((Double)stigma.getField(CitySimStigma.Z)).doubleValue()-
lgZPrediction)/((Double)stigma.getField(CitySimStigma.Z)).doubleValue());
                    granularityProximity = calculateGranularity(stigma, lgXPrediction,
lgYPrediction);
                    if (granularityProximity>=0.0) {
                       lgSuccess++;
                    }
                    if (!exhaustive) {
                        pw.println("      Lgrng
"+100*(((Double)stigma.getField(CitySimStigma.X)).doubleValue()-
lgXPrediction)/((Double)stigma.getField(CitySimStigma.X)).doubleValue()+"%"+

","+100*(((Double)stigma.getField(CitySimStigma.Y)).doubleValue()-
lgYPrediction)/((Double)stigma.getField(CitySimStigma.Y)).doubleValue()+"%"+

","+100*(((Double)stigma.getField(CitySimStigma.Z)).doubleValue()-
lgZPrediction)/((Double)stigma.getField(CitySimStigma.Z)).doubleValue()+"%"+
                                                "
"+(granularityProximity>=0.0?success:failure));
                    }
                }
                NewtonDividedDifferences nddX = new NewtonDividedDifferences(t, x);
                NewtonDividedDifferences nddY = new NewtonDividedDifferences(t, y);
                NewtonDividedDifferences nddZ = new NewtonDividedDifferences(t, z);
                CubicSpline csX = new CubicSpline(t, x);
                csX.calcDeriv();
                CubicSpline csY = new CubicSpline(t, y);
                csY.calcDeriv();
                CubicSpline csZ = new CubicSpline(t, z);
                csZ.calcDeriv();
                Lagrange lgX = new Lagrange(t, x);
                Lagrange lgY = new Lagrange(t, y);
                Lagrange lgZ = new Lagrange(t, z);
                nddXPrediction =
nddX.interpolate(((Double)stigma.getField(CitySimStigma.TIME)).doubleValue()+timeStep)
;
                nddYPrediction =
nddY.interpolate(((Double)stigma.getField(CitySimStigma.TIME)).doubleValue()+timeStep)
;
                nddZPrediction =
nddZ.interpolate(((Double)stigma.getField(CitySimStigma.TIME)).doubleValue()+timeStep)
;
                csXPrediction =
csX.interpolate(((Double)stigma.getField(CitySimStigma.TIME)).doubleValue()+timeStep);
                csYPrediction =
csY.interpolate(((Double)stigma.getField(CitySimStigma.TIME)).doubleValue()+timeStep);
                csZPrediction =
csZ.interpolate(((Double)stigma.getField(CitySimStigma.TIME)).doubleValue()+timeStep);
                lgXPrediction =
lgX.interpolate(((Double)stigma.getField(CitySimStigma.TIME)).doubleValue()+timeStep);
                lgYPrediction =
lgY.interpolate(((Double)stigma.getField(CitySimStigma.TIME)).doubleValue()+timeStep);
                lgZPrediction =
lgZ.interpolate(((Double)stigma.getField(CitySimStigma.TIME)).doubleValue()+timeStep);
                if (!exhaustive) {
```

```
                    pw.println("Next(?) N.D.D:
"+nddXPrediction+","+nddYPrediction+","+nddZPrediction);
                    pw.println("        CbSpl:
"+csXPrediction+","+csYPrediction+","+csZPrediction);
                    pw.println("        Lgrng:
"+lgXPrediction+","+lgYPrediction+","+lgZPrediction);
                }
            }
            if (!exhaustive) {
                time = new Date().getTime() - time;
                pw.println("#");
                pw.println();
                pw.println("Mean error percentages");
                pw.println();
                pw.println("    N.D.D: "+nddError/((counter-queueSize)*3.0)+"%");
                pw.println();
                pw.println("    CbSpl: "+csError/((counter-queueSize)*3.0)+"%");
                pw.println();
                pw.println("    Lgrng: "+lgError/((counter-queueSize)*3.0)+"%");
                pw.println("Successful predictions (out of "+totalPredictions+" tries)
for granularity="+granularity);
                pw.println();
                pw.println("    N.D.D: "+nddSuccess);
                pw.println();
                pw.println("    CbSpl: "+csSuccess);
                pw.println();
                pw.println("    Lgrng: "+lgSuccess);
                pw.println();
                pw.println("Running time: "+time+"ms");
                pw.close();
                System.out.println(" > Mean error percentages:");
                System.out.println(" >    N.D.D: "+nddError/((counter-
queueSize)*3.0)+"%");
                System.out.println(" >    CbSpl: "+csError/((counter-
queueSize)*3.0)+"%");
                System.out.println(" >    Lgrng: "+lgError/((counter-
queueSize)*3.0)+"%");
                System.out.println(" > Successes (out of "+totalPredictions+"
tries):");
                System.out.println(" >    N.D.D: "+nddSuccess);
                System.out.println(" >    CbSpl: "+csSuccess);
                System.out.println(" >    Lgrng: "+lgSuccess);
                System.out.println(" > CITY_SIMULATOR end...");
            } else {
                System.out.println(nddSuccess+"\t"+csSuccess+"\t"+lgSuccess+"\t"+
                    nddError/((counter-queueSize)*3.0)+"\t\t"+csError/((counter-
queueSize)*3.0)+"\t\t"+lgError/((counter-queueSize)*3.0)+"\t\t");
            }
        } catch (Exception e) {
            System.out.println(" > COULD NOT COMPLETE SIMULATION!");
            System.out.println(" > Reason unpredicted... Stack trace follows:");
            e.printStackTrace();
            return;
        }
    }

    private double calculateGranularity(CitySimStigma stigma, double predX, double
predY) {
        double res = 0.0;
        double stigmaX = ((Double)stigma.getField(CitySimStigma.X)).doubleValue();
        double stigmaY = ((Double)stigma.getField(CitySimStigma.Y)).doubleValue();
        double predVector = Math.sqrt(Math.pow(stigmaX-predX, 2.0)+Math.pow(stigmaY-
predY, 2.0));
        res = granularity-predVector;
        return res;
    }
}
```

ReadSignalsGps.java

```
package lbs.calculus;

import java.io.*;
import java.util.*;

import lbs.util.Tank;
import lbs.util.Queue;
```

```
import lbs.util.GpsStigma;
import lbs.util.GpsUtil;

import lbs.interpol.NewtonDividedDifferences;
import lbs.interpol.CubicSpline;
import lbs.interpol.Lagrange;

/* Diavazei stigmata morfhs Gps apo arxeio kai prosomoiwnei diadikasia mobility
prediction bgazontas ektenh apotelesmata se arxeio */
public class ReadSignalsGps implements ReadSignals {
    private int queueSize;
    private double timeStep;
    private double granularity;
    private File fileIn;
    private File fileOut;
    private long time;
    private boolean exhaustive;

    public ReadSignalsGps(File fileIn, File fileOut, int queueSize, double timeStep,
double granularity, boolean exhaustive) {
        this.queueSize = queueSize;
        this.timeStep = timeStep;
        this.granularity = granularity;
        this.fileIn = fileIn;
        this.fileOut = fileOut;
        this.exhaustive = exhaustive;
    }

    public void readSignals() {
        Tank tank = new lbs.util.Tank();
        GpsStigma stigma = null;
        time = new Date().getTime();
        String read = "";
        try {
            if (!exhaustive) {
                System.out.println(" > GPS start...");
                System.out.println(" > Reading GPS signals as NMEA-0183 sentences
(GPGGA)...");
            }
            BufferedReader in = new BufferedReader(new InputStreamReader(new
FileInputStream(fileIn)));
            in.readLine();
            in.readLine();
            boolean firsttime = true;
            while ((read=in.readLine()) != null) {
                if (!read.startsWith("$GPGGA")) {
                      continue;
                }
                stigma = new GpsStigma(read);
                tank.add(stigma);
            }
            in.close();
        } catch (Exception e) {
            System.out.println(" > COULD NOT COMPLETE SIMULATION!");
            System.out.println(" > Possible reason: Input file invalid format... Stack
Trace follows:");
            e.printStackTrace();
            return;
        }
        try {
            PrintWriter pw = null;
            if (!exhaustive) {
                System.out.println(" > Exporting results to
"+fileOut.getAbsolutePath());
                pw = new PrintWriter(new FileOutputStream(fileOut));
            }
            Queue tQ, latQ, longQ, altQ;
            tQ = new Queue(queueSize, 1);
            latQ = new Queue(queueSize, 1);
            longQ = new Queue(queueSize, 1);
            altQ = new Queue(queueSize, 1);
            double nddLatPrediction = 0.0;
            double nddLongPrediction = 0.0;
            double nddAltPrediction = 0.0;
            double csLatPrediction = 0.0;
            double csLongPrediction = 0.0;
            double csAltPrediction = 0.0;
```

```
            double lgLatPrediction = 0.0;
            double lgLongPrediction = 0.0;
            double lgAltPrediction = 0.0;
            int counter = 0;
            double granularityProximity = 0.0;
            String success = "SUCCESS";
            String failure="FAILURE";
            double nddError = 0.0;
            double csError = 0.0;
            double lgError = 0.0;
            int nddSuccess = 0;
            int csSuccess = 0;
            int lgSuccess = 0;
            int totalPredictions = 0;

            if (!exhaustive) {
                pw.println("        MOBILITY SIMULATION RESULTS ("+new Date()+")");
                pw.println("DATA_MODE=GPS (NMEA 0183 format - GPGGA)");
                pw.println("INPUT_FILE="+fileIn.getAbsolutePath());
                pw.println("QUEUE_SIZE="+queueSize);
                pw.println("TIME_STEP="+timeStep);
                pw.println("GRANULARITY="+granularity);
                pw.println("N.D.D=NEWTON DIVIDED DIFFERENCES, CbSpl=CUBIC SPLINES,
Lgrng=LAGRANGE");
                pw.println("stigma = UTC - Latitude, Longtitude, Altitude");
                pw.println();
            }
            while (!tank.isEmpty()) {
                counter++;
                stigma = (GpsStigma)tank.getFirstElement();
                if (!exhaustive) {
                    pw.println("stigma: "+stigma.getField(GpsStigma.TIME)+" -
"+stigma.getField(GpsStigma.LATITUDE)+","+stigma.getField(GpsStigma.LONGTITUDE)+","+st
igma.getField(GpsStigma.ALTITUDE));
                }
                tQ.add(stigma.getField(GpsStigma.TIME));
                latQ.add(stigma.getField(GpsStigma.LATITUDE));
                longQ.add(stigma.getField(GpsStigma.LONGTITUDE));
                altQ.add(stigma.getField(GpsStigma.ALTITUDE));
                if (counter<queueSize) {
                    if (!exhaustive) {
                        pw.println("    Waiting for queue to fill with data...");
                    }
                    continue;
                }
                double [] t = new double[queueSize];
                double [] lat = new double[queueSize];
                double [] lon = new double[queueSize];
                double [] alt = new double[queueSize];
                for (int i=0; i<queueSize; i++) {
                    t[i] = ((Double)tQ.elementAt(i)).doubleValue();
                    lat[i] = ((Double)latQ.elementAt(i)).doubleValue();
                    lon[i] = ((Double)longQ.elementAt(i)).doubleValue();
                    alt[i] = ((Double)altQ.elementAt(i)).doubleValue();
                }
                if (counter>queueSize) {
                    totalPredictions++;
                    nddError +=
Math.abs(100*(((Double)stigma.getField(GpsStigma.LATITUDE)).doubleValue()-
nddLatPrediction)/((Double)stigma.getField(GpsStigma.LATITUDE)).doubleValue());
                    nddError +=
Math.abs(100*(((Double)stigma.getField(GpsStigma.LONGTITUDE)).doubleValue()-
nddLongPrediction)/((Double)stigma.getField(GpsStigma.LONGTITUDE)).doubleValue());
                    nddError +=
Math.abs(100*(((Double)stigma.getField(GpsStigma.ALTITUDE)).doubleValue()-
nddAltPrediction)/((Double)stigma.getField(GpsStigma.ALTITUDE)).doubleValue());
                    granularityProximity = calculateGranularity(stigma,
nddLatPrediction, nddLongPrediction);
                    if (granularityProximity>=0.0) {
                      nddSuccess++;
                    }
                    if (!exhaustive) {
                        pw.println("Error N.D.D
"+100*(((Double)stigma.getField(GpsStigma.LATITUDE)).doubleValue()-
nddLatPrediction)/((Double)stigma.getField(GpsStigma.LATITUDE)).doubleValue()+"%"+
```

```
","+100*(((Double)stigma.getField(GpsStigma.LONGTITUDE)).doubleValue()-
nddLongPrediction)/((Double)stigma.getField(GpsStigma.LONGTITUDE)).doubleValue()+"%"+

","+100*(((Double)stigma.getField(GpsStigma.ALTITUDE)).doubleValue()-
nddAltPrediction)/((Double)stigma.getField(GpsStigma.ALTITUDE)).doubleValue()+"%"+
                                                    "
"+(granularityProximity>=0.0?success:failure));
                    }
                    csError +=
Math.abs(100*(((Double)stigma.getField(GpsStigma.LATITUDE)).doubleValue()-
csLatPrediction)/((Double)stigma.getField(GpsStigma.LATITUDE)).doubleValue());
                    csError +=
Math.abs(100*(((Double)stigma.getField(GpsStigma.LONGTITUDE)).doubleValue()-
csLongPrediction)/((Double)stigma.getField(GpsStigma.LONGTITUDE)).doubleValue());
                    csError +=
Math.abs(100*(((Double)stigma.getField(GpsStigma.ALTITUDE)).doubleValue()-
csAltPrediction)/((Double)stigma.getField(GpsStigma.ALTITUDE)).doubleValue());
                    granularityProximity = calculateGranularity(stigma,
csLatPrediction, csLongPrediction);
                    if (granularityProximity>=0.0) {
                      csSuccess++;
                    }
                    if (!exhaustive) {
                        pw.println("      CbSpl
"+100*(((Double)stigma.getField(GpsStigma.LATITUDE)).doubleValue()-
csLatPrediction)/((Double)stigma.getField(GpsStigma.LATITUDE)).doubleValue()+"%"+

","+100*(((Double)stigma.getField(GpsStigma.LONGTITUDE)).doubleValue()-
csLongPrediction)/((Double)stigma.getField(GpsStigma.LONGTITUDE)).doubleValue()+"%"+

","+100*(((Double)stigma.getField(GpsStigma.ALTITUDE)).doubleValue()-
csAltPrediction)/((Double)stigma.getField(GpsStigma.ALTITUDE)).doubleValue()+"%"+
                                                    "
"+(granularityProximity>=0.0?success:failure));
                    }
                    lgError +=
Math.abs(100*(((Double)stigma.getField(GpsStigma.LATITUDE)).doubleValue()-
lgLatPrediction)/((Double)stigma.getField(GpsStigma.LATITUDE)).doubleValue());
                    lgError +=
Math.abs(100*(((Double)stigma.getField(GpsStigma.LONGTITUDE)).doubleValue()-
lgLongPrediction)/((Double)stigma.getField(GpsStigma.LONGTITUDE)).doubleValue());
                    lgError +=
Math.abs(100*(((Double)stigma.getField(GpsStigma.ALTITUDE)).doubleValue()-
lgAltPrediction)/((Double)stigma.getField(GpsStigma.ALTITUDE)).doubleValue());
                    granularityProximity = calculateGranularity(stigma,
lgLatPrediction, lgLongPrediction);
                    if (granularityProximity>=0.0) {
                      lgSuccess++;
                    }
                    if (!exhaustive) {
                        pw.println("      Lgrng
"+100*(((Double)stigma.getField(GpsStigma.LATITUDE)).doubleValue()-
lgLatPrediction)/((Double)stigma.getField(GpsStigma.LATITUDE)).doubleValue()+"%"+

","+100*(((Double)stigma.getField(GpsStigma.LONGTITUDE)).doubleValue()-
lgLongPrediction)/((Double)stigma.getField(GpsStigma.LONGTITUDE)).doubleValue()+"%"+

","+100*(((Double)stigma.getField(GpsStigma.ALTITUDE)).doubleValue()-
lgAltPrediction)/((Double)stigma.getField(GpsStigma.ALTITUDE)).doubleValue()+"%"+
                                                    "
"+(granularityProximity>=0.0?success:failure));
                    }
                }
                NewtonDividedDifferences nddLat = new NewtonDividedDifferences(t,
lat);
                NewtonDividedDifferences nddLong = new NewtonDividedDifferences(t,
lon);
                NewtonDividedDifferences nddAlt = new NewtonDividedDifferences(t,
alt);
                CubicSpline csLat = new CubicSpline(t, lat);
                csLat.calcDeriv();
                CubicSpline csLong = new CubicSpline(t, lon);
                csLong.calcDeriv();
                CubicSpline csAlt = new CubicSpline(t, alt);
                csAlt.calcDeriv();
                Lagrange lgLat = new Lagrange(t, lat);
```

```
                Lagrange lgLong = new Lagrange(t, lon);
                Lagrange lgAlt = new Lagrange(t, alt);
                nddLatPrediction =
nddLat.interpolate(((Double)stigma.getField(GpsStigma.TIME)).doubleValue()+timeStep);
                nddLongPrediction =
nddLong.interpolate(((Double)stigma.getField(GpsStigma.TIME)).doubleValue()+timeStep);
                nddAltPrediction =
nddAlt.interpolate(((Double)stigma.getField(GpsStigma.TIME)).doubleValue()+timeStep);
                csLatPrediction =
csLat.interpolate(((Double)stigma.getField(GpsStigma.TIME)).doubleValue()+timeStep);
                csLongPrediction =
csLong.interpolate(((Double)stigma.getField(GpsStigma.TIME)).doubleValue()+timeStep);
                csAltPrediction =
csAlt.interpolate(((Double)stigma.getField(GpsStigma.TIME)).doubleValue()+timeStep);
                lgLatPrediction =
lgLat.interpolate(((Double)stigma.getField(GpsStigma.TIME)).doubleValue()+timeStep);
                lgLongPrediction =
lgLong.interpolate(((Double)stigma.getField(GpsStigma.TIME)).doubleValue()+timeStep);
                lgAltPrediction =
lgAlt.interpolate(((Double)stigma.getField(GpsStigma.TIME)).doubleValue()+timeStep);
                if (!exhaustive) {
                    pw.println("Next(?) N.D.D:
"+nddLatPrediction+","+nddLongPrediction+","+nddAltPrediction);
                    pw.println("        CbSpl:
"+csLatPrediction+","+csLongPrediction+","+csAltPrediction);
                    pw.println("        Lgrng:
"+lgLatPrediction+","+lgLongPrediction+","+lgAltPrediction);
                }
            }
            if (!exhaustive) {
                time = new Date().getTime() - time;
                pw.println("#");
                pw.println();
                pw.println("Mean error percentages");
                pw.println();
                pw.println("    N.D.D: "+nddError/((counter-queueSize)*3.0)+"%");
                pw.println();
                pw.println("    CbSpl: "+csError/((counter-queueSize)*3.0)+"%");
                pw.println();
                pw.println("    Lgrng: "+lgError/((counter-queueSize)*3.0)+"%");
                pw.println();
                pw.println("Successful predictions (out of "+totalPredictions+" tries)
for granularity="+granularity);
                pw.println();
                pw.println("    N.D.D: "+nddSuccess);
                pw.println();
                pw.println("    CbSpl: "+csSuccess);
                pw.println();
                pw.println("    Lgrng: "+lgSuccess);
                pw.println();
                pw.println("Running time: "+time+"ms");
                pw.close();
                System.out.println(" > Mean error percentages:");
                System.out.println(" >    N.D.D: "+nddError/((counter-
queueSize)*3.0)+"%");
                System.out.println(" >    CbSpl: "+csError/((counter-
queueSize)*3.0)+"%");
                System.out.println(" >    Lgrng: "+lgError/((counter-
queueSize)*3.0)+"%");
                System.out.println(" > Successes (out of "+totalPredictions+"
tries):");
                System.out.println(" >    N.D.D: "+nddSuccess);
                System.out.println(" >    CbSpl: "+csSuccess);
                System.out.println(" >    Lgrng: "+lgSuccess);
                System.out.println(" > GPS end...");
            } else {
                System.out.println(nddSuccess+"\t"+csSuccess+"\t"+lgSuccess+"\t"+
                    nddError/((counter-queueSize)*3.0)+"\t\t"+csError/((counter-
queueSize)*3.0)+"\t\t"+lgError/((counter-queueSize)*3.0)+"\t\t");
            }
        } catch (Exception e) {
            System.out.println(" > COULD NOT COMPLETE SIMULATION!");
            System.out.println(" > Reason unpredicted... Stack trace follows:");
            e.printStackTrace();
            return;
        }
    }
```

```
    private double calculateGranularity(GpsStigma stigma, double predLat, double
predLong) {
        double res = 0.0;
        res = granularity - GpsUtil.gpsDistance(stigma, predLat, predLong);
        return res;
    }
}
```

ReadSignalsSimpleNK.java

```
package lbs.calculus;

import java.io.*;
import java.util.*;

import lbs.util.Tank;
import lbs.util.Queue;
import lbs.util.SimpleStigma;

import lbs.interpol.CubicSpline;

/* Diavazei stigmata morfhs Simple apo arxeio kai prosomoiwnei diadikasia mobility
prediction bgazontas ektenh apotelesmata se arxeio
   H sygekrimenh diaferei apo thn aplh ReadSignalsSimple sto oti kanei problepsh gia k
bhmata mprosta sto xwro (oxi gia to epomeno)
   kai malista xrhsimopoiwntas mono Cubic Splines gia na kanei thn problepsh (oi
polywnymikes exoun krithei ligotero katallhles)*/
public class ReadSignalsSimpleNK implements ReadSignals {
    private int queueSize;
    private double timeStep;
    private double granularity;
    private File fileIn;
    private File fileOut;
    private long time;
    private boolean exhaustive;
    private int steps;

    public ReadSignalsSimpleNK(File fileIn, File fileOut, int queueSize, double
timeStep, int steps, boolean exhaustive) {
        this.queueSize = queueSize;
        this.timeStep = timeStep;
        this.fileIn = fileIn;
        this.fileOut = fileOut;
        this.exhaustive = exhaustive;
        this.steps = steps;
    }

    public void readSignals() {
        Tank tank = new lbs.util.Tank();
        SimpleStigma stigma = null;
        time = new Date().getTime();
        String read = "";
        try {
            if (!exhaustive) {
                System.out.println(" > SIMPLE-NK start (only with Cubic Splines)...");
            }
            BufferedReader in = new BufferedReader(new InputStreamReader(new
FileInputStream(fileIn)));
            in.readLine();
            in.readLine();
            boolean firsttime = true;
            while ((read=in.readLine()) != null) {
                stigma = new SimpleStigma(read);
                tank.add(stigma);
            }
            in.close();
        } catch (Exception e) {
            System.out.println(" > COULD NOT COMPLETE SIMULATION!");
            System.out.println(" > Possible reason: Input file invalid format... Stack
Trace follows:");
            e.printStackTrace();
            return;
        }
        try {
            PrintWriter pw = null;
            if (!exhaustive) {
```

```
                System.out.println(" > Exporting results to
"+fileOut.getAbsolutePath());
                pw = new PrintWriter(new FileOutputStream(fileOut));
            }
            Queue tQ, xQ, yQ, altQ;
            tQ = new Queue(queueSize, 1);
            xQ = new Queue(queueSize, 1);
            yQ = new Queue(queueSize, 1);
            //altQ = new Queue(queueSize, 1);
            double [] csXPred = new double[steps];
            double [] csYPred = new double[steps];
            //double [] csAltPred = new double[steps];
            int predCounter = 0;
            int counter = 0;
            String success = "SUCCESS";
            String failure="FAILURE";
            double csError = 0.0;

            if (!exhaustive) {
                pw.println("        MOBILITY SIMULATION RESULTS ("+new Date()+")");
                pw.println("DATA_MODE=SIMPLE");
                pw.println("INPUT_FILE="+fileIn.getAbsolutePath());
                pw.println("QUEUE_SIZE="+queueSize);
                pw.println("TIME_STEP="+timeStep);
                pw.println("CbSpl=CUBIC SPLINES");
                pw.println("stigma = t - x - y");
                pw.println();
            }
            while (!tank.isEmpty()) {
                counter++;
                stigma = (SimpleStigma)tank.getFirstElement();
                if (!exhaustive) {
                    pw.println("stigma: "+stigma.getField(SimpleStigma.TIME)+" -
"+stigma.getField(SimpleStigma.X)+","+stigma.getField(SimpleStigma.Y));
                }
                tQ.add(stigma.getField(SimpleStigma.TIME));
                xQ.add(stigma.getField(SimpleStigma.X));
                yQ.add(stigma.getField(SimpleStigma.Y));
                //altQ.add(stigma.getField(SimpleStigma.ALTITUDE));
                if (counter<queueSize) {
                    if (!exhaustive) {
                        pw.println("    Waiting for queue to fill with data...");
                    }
                    continue;
                }
                double [] t = new double[queueSize];
                double [] x = new double[queueSize];
                double [] y = new double[queueSize];
                //double [] alt = new double[queueSize];
                for (int i=0; i<queueSize; i++) {
                    t[i] = ((Double)tQ.elementAt(i)).doubleValue();
                    x[i] = ((Double)xQ.elementAt(i)).doubleValue();
                    y[i] = ((Double)yQ.elementAt(i)).doubleValue();
                    //alt[i] = ((Double)altQ.elementAt(i)).doubleValue();
                }
                if (counter>=queueSize+steps) {
                    csError +=
Math.abs(100*(((Double)stigma.getField(SimpleStigma.X)).doubleValue()-
csXPred[predCounter])/((Double)stigma.getField(SimpleStigma.X)).doubleValue());
                    csError +=
Math.abs(100*(((Double)stigma.getField(SimpleStigma.Y)).doubleValue()-
csYPred[predCounter])/((Double)stigma.getField(SimpleStigma.Y)).doubleValue());
                    //csError +=
Math.abs(100*(((Double)stigma.getField(GpsStigma.ALTITUDE)).doubleValue()-
csAltPred[predCounter])/((Double)stigma.getField(GpsStigma.ALTITUDE)).doubleValue());
                    if (!exhaustive) {
                        pw.println("Error CbSpl
"+100*(((Double)stigma.getField(SimpleStigma.X)).doubleValue()-
csXPred[predCounter])/((Double)stigma.getField(SimpleStigma.X)).doubleValue()+"%"+

","+100*(((Double)stigma.getField(SimpleStigma.Y)).doubleValue()-
csYPred[predCounter])/((Double)stigma.getField(SimpleStigma.Y)).doubleValue()+"%"+"");

//","+100*(((Double)stigma.getField(GpsStigma.ALTITUDE)).doubleValue()-
csAltPred[predCounter])/((Double)stigma.getField(GpsStigma.ALTITUDE)).doubleValue()+"%
");
                    }
```

```
                }
                CubicSpline csX = new CubicSpline(t, x);
                csX.calcDeriv();
                CubicSpline csY = new CubicSpline(t, y);
                csY.calcDeriv();
                //CubicSpline csAlt = new CubicSpline(t, alt);
                //csAlt.calcDeriv();
                csXPred[predCounter] =
csX.interpolate(((Double)stigma.getField(SimpleStigma.TIME)).doubleValue()+timeStep*st
eps);
                csYPred[predCounter] =
csY.interpolate(((Double)stigma.getField(SimpleStigma.TIME)).doubleValue()+timeStep*st
eps);
                //csAltPred[predCounter] =
csAlt.interpolate(((Double)stigma.getField(GpsStigma.TIME)).doubleValue()+timeStep*ste
ps);
                if (!exhaustive) {
                    pw.println("Next("+steps+") CbSpl:
"+csXPred[predCounter]+","+csYPred[predCounter]);//+","+csAltPred[predCounter]);
                }
                predCounter++;
                if (predCounter==steps) {
                      predCounter = 0;
                }
            }
            if (!exhaustive) {
                time = new Date().getTime() - time;
                pw.println("#");
                pw.println();
                pw.println("Mean error percentages");
                pw.println();
                pw.println("    CbSpl: "+csError/((counter-queueSize-steps)*2.0)+"%");
                pw.println();
                pw.println("Running time: "+time+"ms");
                pw.close();
                System.out.println(" > Mean error percentages:");
                System.out.println(" >    CbSpl: "+csError/((counter-queueSize-
steps)*2.0)+"%");
                System.out.println(" > SIMPLE-NK end...");
            } else {
                System.out.println(csError/((counter-queueSize-steps)*2.0));
            }
        } catch (Exception e) {
            System.out.println(" > COULD NOT COMPLETE SIMULATION!");
            System.out.println(" > Reason unpredicted... Stack trace follows:");
            e.printStackTrace();
            return;
        }
    }
}
```

ReadSignalsCitySimNK.java

```
package lbs.calculus;

import java.io.*;
import java.util.*;

import lbs.util.Tank;
import lbs.util.Queue;
import lbs.util.CitySimStigma;

import lbs.interpol.NewtonDividedDifferences;
import lbs.interpol.CubicSpline;
import lbs.interpol.Lagrange;

/* Diavazei stigmata morfhs City Simulator apo arxeio kai prosomoiwnei diadikasia
mobility prediction bgazontas ektenh apotelesmata se arxeio
   H sygekrimenh diaferei apo thn aplh ReadSignalsGps sto oti kanei problepsh gia k
bhmata mprosta sto xwro (oxi gia to epomeno)
   kai malista xrhsimopoiwntas mono Cubic Splines gia na kanei thn problepsh (oi
polywnymikes exoun krithei ligotero katallhles)*/
public class ReadSignalsCitySimNK implements ReadSignals {
    private int queueSize;
    private double timeStep;
    private double granularity;
    private File fileIn;
```

```
    private File fileOut;
    private long time;
    private boolean exhaustive;
    private int steps;

    public ReadSignalsCitySimNK(File fileIn, File fileOut, int queueSize, double
timeStep, int steps, boolean exhaustive) {
        this.queueSize = queueSize;
        this.timeStep = timeStep;
        this.fileIn = fileIn;
        this.fileOut = fileOut;
        this.exhaustive = exhaustive;
        this.steps = steps;
    }

    public void readSignals() {
        Tank tank = new lbs.util.Tank();
        CitySimStigma stigma = null;
        time = new Date().getTime();
        String read = "";
        try {
            if (!exhaustive) {
                System.out.println(" > CITY_SIMULATOR-NK start (only with Cubic
Splines)...");
            }
            BufferedReader in = new BufferedReader(new InputStreamReader(new
FileInputStream(fileIn)));
            in.readLine();
            in.readLine();
            in.readLine();
            in.readLine();
            in.readLine();
            boolean firsttime = true;
            while ((read=in.readLine()) != null) {
                stigma = new CitySimStigma(read);
                tank.add(stigma);
            }
            in.close();
        } catch (Exception e) {
            System.out.println(" > COULD NOT COMPLETE SIMULATION!");
            System.out.println(" > Possible reason: Input file invalid format... Stack
Trace follows:");
            e.printStackTrace();
            return;
        }
        try {
            PrintWriter pw = null;
            if (!exhaustive) {
                System.out.println(" > Exporting results to
"+fileOut.getAbsolutePath());
                pw = new PrintWriter(new FileOutputStream(fileOut));
            }
            Queue tQ, xQ, yQ, zQ;
            tQ = new Queue(queueSize, 1);
            xQ = new Queue(queueSize, 1);
            yQ = new Queue(queueSize, 1);
            zQ = new Queue(queueSize, 1);
            double [] csXPred = new double[steps];
            double [] csYPred = new double[steps];
            double [] csZPred = new double[steps];
            int predCounter = 0;
            int counter = 0;
            String success = "SUCCESS";
            String failure="FAILURE";
            double csError = 0.0;

            if (!exhaustive) {
                pw.println("        MOBILITY SIMULATION RESULTS ("+new Date()+")");
                pw.println("DATA_MODE=CITY_SIMULATOR");
                pw.println("INPUT_FILE="+fileIn.getAbsolutePath());
                pw.println("QUEUE_SIZE="+queueSize);
                pw.println("TIME_STEP="+timeStep);
                pw.println("CbSpl=CUBIC SPLINES");
                pw.println("stigma = UTC - Latitude, Longtitude, Altitude");
                pw.println();
            }
            while (!tank.isEmpty()) {
```

```
                counter++;
                stigma = (CitySimStigma)tank.getFirstElement();
                if (!exhaustive) {
                    pw.println("stigma: "+stigma.getField(CitySimStigma.TIME)+" -
"+stigma.getField(CitySimStigma.X)+","+stigma.getField(CitySimStigma.Y)+","+stigma.get
Field(CitySimStigma.Z));
                }
                tQ.add(stigma.getField(CitySimStigma.TIME));
                xQ.add(stigma.getField(CitySimStigma.X));
                yQ.add(stigma.getField(CitySimStigma.Y));
                zQ.add(stigma.getField(CitySimStigma.Z));
                if (counter<queueSize) {
                    if (!exhaustive) {
                        pw.println("    Waiting for queue to fill with data...");
                    }
                    continue;
                }
                double [] t = new double[queueSize];
                double [] x = new double[queueSize];
                double [] y = new double[queueSize];
                double [] z = new double[queueSize];
                for (int i=0; i<queueSize; i++) {
                    t[i] = ((Double)tQ.elementAt(i)).doubleValue();
                    x[i] = ((Double)xQ.elementAt(i)).doubleValue();
                    y[i] = ((Double)yQ.elementAt(i)).doubleValue();
                    z[i] = ((Double)zQ.elementAt(i)).doubleValue();
                }
                if (counter>=queueSize+steps) {
                    csError +=
Math.abs(100*(((Double)stigma.getField(CitySimStigma.X)).doubleValue()-
csXPred[predCounter])/((Double)stigma.getField(CitySimStigma.X)).doubleValue());
                    csError +=
Math.abs(100*(((Double)stigma.getField(CitySimStigma.Y)).doubleValue()-
csYPred[predCounter])/((Double)stigma.getField(CitySimStigma.Y)).doubleValue());
                    csError +=
Math.abs(100*(((Double)stigma.getField(CitySimStigma.Z)).doubleValue()-
csZPred[predCounter])/((Double)stigma.getField(CitySimStigma.Z)).doubleValue());
                    if (!exhaustive) {
                        pw.println("Error CbSpl
"+100*(((Double)stigma.getField(CitySimStigma.X)).doubleValue()-
csXPred[predCounter])/((Double)stigma.getField(CitySimStigma.X)).doubleValue()+"%"+

","+100*(((Double)stigma.getField(CitySimStigma.Y)).doubleValue()-
csYPred[predCounter])/((Double)stigma.getField(CitySimStigma.Y)).doubleValue()+"%"+

","+100*(((Double)stigma.getField(CitySimStigma.Z)).doubleValue()-
csZPred[predCounter])/((Double)stigma.getField(CitySimStigma.Z)).doubleValue()+"%");
                    }
                }
                CubicSpline csX = new CubicSpline(t, x);
                csX.calcDeriv();
                CubicSpline csY = new CubicSpline(t, y);
                csY.calcDeriv();
                CubicSpline csZ = new CubicSpline(t, z);
                csZ.calcDeriv();
                csXPred[predCounter] =
csX.interpolate(((Double)stigma.getField(CitySimStigma.TIME)).doubleValue()+timeStep*s
teps);
                csYPred[predCounter] =
csY.interpolate(((Double)stigma.getField(CitySimStigma.TIME)).doubleValue()+timeStep*s
teps);
                csZPred[predCounter] =
csZ.interpolate(((Double)stigma.getField(CitySimStigma.TIME)).doubleValue()+timeStep*s
teps);
                if (!exhaustive) {
                    pw.println("Next("+steps+") CbSpl:
"+csXPred[predCounter]+","+csYPred[predCounter]+","+csZPred[predCounter]);
                }
                predCounter++;
                if (predCounter==steps) {
                       predCounter = 0;
                }
            }
            if (!exhaustive) {
                time = new Date().getTime() - time;
                pw.println("#");
                pw.println();
```

```
                pw.println("Mean error percentages");
                pw.println();
                pw.println("    CbSpl: "+csError/((counter-queueSize-steps)*3.0)+"%");
                pw.println();
                pw.println("Running time: "+time+"ms");
                pw.close();
                System.out.println(" > Mean error percentages:");
                System.out.println(" >    CbSpl: "+csError/((counter-queueSize-
steps)*3.0)+"%");
                System.out.println(" > CITY_SIMULATOR-NK end...");
            } else {
                System.out.println(csError/((counter-queueSize-steps)*3.0));
            }
        } catch (Exception e) {
            System.out.println(" > COULD NOT COMPLETE SIMULATION!");
            System.out.println(" > Reason unpredicted... Stack trace follows:");
            e.printStackTrace();
            return;
        }
    }
}
```

ReadSignalsGpsNK.java

```
package lbs.calculus;

import java.io.*;
import java.util.*;

import lbs.util.Tank;
import lbs.util.Queue;
import lbs.util.GpsStigma;
import lbs.util.GpsUtil;

import lbs.interpol.NewtonDividedDifferences;
import lbs.interpol.CubicSpline;
import lbs.interpol.Lagrange;

/* Diavazei stigmata morfhs Gps apo arxeio kai prosomoiwnei diadikasia mobility
prediction bgazontas ektenh apotelesmata se arxeio
   H sygekrimenh diaferei apo thn aplh ReadSignalsGps sto oti kanei problepsh gia k
bhmata mprosta sto xwro (oxi gia to epomeno)
   kai malista xrhsimopoiwntas mono Cubic Splines gia na kanei thn problepsh (oi
polywnymikes exoun krithei ligotero katallhles)*/
public class ReadSignalsGpsNK implements ReadSignals {
    private int queueSize;
    private double timeStep;
    private double granularity;
    private File fileIn;
    private File fileOut;
    private long time;
    private boolean exhaustive;
    private int steps;

    public ReadSignalsGpsNK(File fileIn, File fileOut, int queueSize, double timeStep,
int steps, boolean exhaustive) {
        this.queueSize = queueSize;
        this.timeStep = timeStep;
        this.fileIn = fileIn;
        this.fileOut = fileOut;
        this.exhaustive = exhaustive;
        this.steps = steps;
    }

    public void readSignals() {
        Tank tank = new lbs.util.Tank();
        GpsStigma stigma = null;
        time = new Date().getTime();
        String read = "";
        try {
            if (!exhaustive) {
                System.out.println(" > GPS-NK start (only with Cubic Splines)...");
                System.out.println(" > Reading GPS signals as NMEA-0183 sentences
(GPGGA)...");
            }
            BufferedReader in = new BufferedReader(new InputStreamReader(new
FileInputStream(fileIn)));
```

```
            in.readLine();
            in.readLine();
            boolean firsttime = true;
            while ((read=in.readLine()) != null) {
                if (!read.startsWith("$GPGGA")) {
                      continue;
                }
                stigma = new GpsStigma(read);
                tank.add(stigma);
            }
            in.close();
        } catch (Exception e) {
            System.out.println(" > COULD NOT COMPLETE SIMULATION!");
            System.out.println(" > Possible reason: Input file invalid format... Stack
Trace follows:");
            e.printStackTrace();
            return;
        }
        try {
            PrintWriter pw = null;
            if (!exhaustive) {
                System.out.println(" > Exporting results to
"+fileOut.getAbsolutePath());
                pw = new PrintWriter(new FileOutputStream(fileOut));
            }
            Queue tQ, latQ, longQ, altQ;
            tQ = new Queue(queueSize, 1);
            latQ = new Queue(queueSize, 1);
            longQ = new Queue(queueSize, 1);
            altQ = new Queue(queueSize, 1);
            double [] csLatPred = new double[steps];
            double [] csLongPred = new double[steps];
            double [] csAltPred = new double[steps];
            int predCounter = 0;
            int counter = 0;
            String success = "SUCCESS";
            String failure="FAILURE";
            double csError = 0.0;

            if (!exhaustive) {
                pw.println("        MOBILITY SIMULATION RESULTS ("+new Date()+")");
                pw.println("DATA_MODE=GPS (NMEA 0183 format - GPGGA)");
                pw.println("INPUT_FILE="+fileIn.getAbsolutePath());
                pw.println("QUEUE_SIZE="+queueSize);
                pw.println("TIME_STEP="+timeStep);
                pw.println("CbSpl=CUBIC SPLINES");
                pw.println("stigma = UTC - Latitude, Longtitude, Altitude");
                pw.println();
            }
            while (!tank.isEmpty()) {
                counter++;
                stigma = (GpsStigma)tank.getFirstElement();
                if (!exhaustive) {
                    pw.println("stigma: "+stigma.getField(GpsStigma.TIME)+" -
"+stigma.getField(GpsStigma.LATITUDE)+","+stigma.getField(GpsStigma.LONGTITUDE)+","+st
igma.getField(GpsStigma.ALTITUDE));
                }
                tQ.add(stigma.getField(GpsStigma.TIME));
                latQ.add(stigma.getField(GpsStigma.LATITUDE));
                longQ.add(stigma.getField(GpsStigma.LONGTITUDE));
                altQ.add(stigma.getField(GpsStigma.ALTITUDE));
                if (counter<queueSize) {
                    if (!exhaustive) {
                        pw.println("    Waiting for queue to fill with data...");
                    }
                    continue;
                }
                double [] t = new double[queueSize];
                double [] lat = new double[queueSize];
                double [] lon = new double[queueSize];
                double [] alt = new double[queueSize];
                for (int i=0; i<queueSize; i++) {
                    t[i] = ((Double)tQ.elementAt(i)).doubleValue();
                    lat[i] = ((Double)latQ.elementAt(i)).doubleValue();
                    lon[i] = ((Double)longQ.elementAt(i)).doubleValue();
                    alt[i] = ((Double)altQ.elementAt(i)).doubleValue();
                }
```

```
                if (counter>=queueSize+steps) {
                    csError +=
Math.abs(100*(((Double)stigma.getField(GpsStigma.LATITUDE)).doubleValue()-
csLatPred[predCounter])/((Double)stigma.getField(GpsStigma.LATITUDE)).doubleValue());
                    csError +=
Math.abs(100*(((Double)stigma.getField(GpsStigma.LONGTITUDE)).doubleValue()-
csLongPred[predCounter])/((Double)stigma.getField(GpsStigma.LONGTITUDE)).doubleValue()
);
                    csError +=
Math.abs(100*(((Double)stigma.getField(GpsStigma.ALTITUDE)).doubleValue()-
csAltPred[predCounter])/((Double)stigma.getField(GpsStigma.ALTITUDE)).doubleValue());
                    if (!exhaustive) {
                        pw.println("Error CbSpl
"+100*(((Double)stigma.getField(GpsStigma.LATITUDE)).doubleValue()-
csLatPred[predCounter])/((Double)stigma.getField(GpsStigma.LATITUDE)).doubleValue()+"%
"+

","+100*(((Double)stigma.getField(GpsStigma.LONGTITUDE)).doubleValue()-
csLongPred[predCounter])/((Double)stigma.getField(GpsStigma.LONGTITUDE)).doubleValue()
+"%"+

","+100*(((Double)stigma.getField(GpsStigma.ALTITUDE)).doubleValue()-
csAltPred[predCounter])/((Double)stigma.getField(GpsStigma.ALTITUDE)).doubleValue()+"%
");
                    }
                }
                CubicSpline csLat = new CubicSpline(t, lat);
                csLat.calcDeriv();
                CubicSpline csLong = new CubicSpline(t, lon);
                csLong.calcDeriv();
                CubicSpline csAlt = new CubicSpline(t, alt);
                csAlt.calcDeriv();
                csLatPred[predCounter] =
csLat.interpolate(((Double)stigma.getField(GpsStigma.TIME)).doubleValue()+timeStep*ste
ps);
                csLongPred[predCounter] =
csLong.interpolate(((Double)stigma.getField(GpsStigma.TIME)).doubleValue()+timeStep*st
eps);
                csAltPred[predCounter] =
csAlt.interpolate(((Double)stigma.getField(GpsStigma.TIME)).doubleValue()+timeStep*ste
ps);
                if (!exhaustive) {
                    pw.println("Next("+steps+") CbSpl:
"+csLatPred[predCounter]+","+csLongPred[predCounter]+","+csAltPred[predCounter]);
                }
                predCounter++;
                if (predCounter==steps) {
                      predCounter = 0;
                }
            }
            if (!exhaustive) {
                time = new Date().getTime() - time;
                pw.println("#");
                pw.println();
                pw.println("Mean error percentages");
                pw.println();
                pw.println("    CbSpl: "+csError/((counter-queueSize-steps)*3.0)+"%");
                pw.println();
                pw.println("Running time: "+time+"ms");
                pw.close();
                System.out.println(" > Mean error percentages:");
                System.out.println(" >    CbSpl: "+csError/((counter-queueSize-
steps)*3.0)+"%");
                System.out.println(" > GPS-NK end...");
            } else {
                System.out.println(csError/((counter-queueSize-steps)*3.0));
            }
        } catch (Exception e) {
            System.out.println(" > COULD NOT COMPLETE SIMULATION!");
            System.out.println(" > Reason unpredicted... Stack trace follows:");
            e.printStackTrace();
            return;
        }
    }
}
```

simulation.java

```
import java.io.*;
import java.util.*;

import lbs.util.Tank;
import lbs.util.Queue;
import lbs.util.Stigma;
import lbs.util.AbstractStigma;
import lbs.util.SimpleStigma;
import lbs.util.CitySimStigma;
import lbs.util.GpsStigma;

import lbs.interpol.NewtonDividedDifferences;
import lbs.interpol.CubicSpline;
import lbs.interpol.Lagrange;

import lbs.calculus.ReadSignalsSimple;
import lbs.calculus.ReadSignalsCitySim;
import lbs.calculus.ReadSignalsGps;

/* H klash me th main pou diabazei tis parametrous apo command line kai ektelei thn
prosomoiwsh */
public class simulation {
    private static final String FS = System.getProperty("file.separator");
    private static Properties props;
    private static int queueSize;
    private static double timeStep;
    private static double granularity;
    private static File fileIn;
    private static File fileOut;

    public static void main(String args[]) {
        String message = "";
        String dataInput = "";
        String availableInputs = "";
        boolean validInput = false;
        if (args.length<1) {
            System.out.println(" > Usage:");
            System.out.println(" > | param1 = DATA_MODE {SIMPLE, CITY_SIMULATOR,
GPS}");
            System.out.println(" > | param2 = INPUT_FILE {path absolute/relative}");
            System.out.println(" > | param3 = QUEUE_LENGTH {integer}");
            System.out.println(" > | param4 = TIME_STEP {double}");
            System.out.println(" > | param5 = GRANULARITY - as a radius {double}");
            System.out.println(" > | param6 = OUTPUT_FILE {path absolute/relative}");
            validInput = false;
            message = "|";
        } else {
            dataInput = args[0];
            availableInputs = "SIMPLE;CITY_SIMULATOR;GPS;";
            StringTokenizer tokens = new StringTokenizer(availableInputs, ";");
            while (tokens.hasMoreTokens()) {
                if (dataInput.equals(tokens.nextToken())) {
                    validInput = true;
                    break;
                }
            }
            if (!validInput) {
               message = "Unknown DATA_MODE";
            }
            if (validInput) {
                if (args.length<2) {
                    validInput = false;
                    message = "INPUT_FILE not specified";
                } else {
                    fileIn = new File(args[1]);
                    if (!fileIn.exists()) {
                        validInput = false;
                        message = "INPUT_FILE does no exist";
                    }
                }
            }
            if (validInput) {
                if (args.length>2) {
                    try {
                        Integer Q = new Integer(args[2]);
```

```
                queueSize = Q.intValue();
            } catch (NumberFormatException nfe) {
                validInput = false;
                message = "Invalid QUEUE_LENGTH";
            }
        } else {
               validInput = false;
            message = "QUEUE_LENGTH not specified";
        }
    }
    if (validInput) {
        if (args.length>3) {
            try {
                Double Q = new Double(args[3]);
                timeStep = Q.doubleValue();
            } catch (NumberFormatException nfe) {
                validInput = false;
                message = "Invalid TIME_STEP";
            }
        } else {
               validInput = false;
            message = "TIME_STEP not specified";
        }
    }
    if (validInput) {
        if (args.length<5) {
               validInput = false;
            message = "GRANULARITY not specified";
        } else {
               try {
                      Double B = new Double(args[4]);
                granularity = B.doubleValue();
               }
               catch (NumberFormatException nfe) {
                validInput = false;
                message = "Invalid GRANULARITY";
               }
        }
    }
    if (validInput) {
        if (args.length<6) {
            validInput = false;
            message = "OUTPUT_FILE not specified";
        } else {
            fileOut = new File(args[5]);
            if (!fileOut.exists()) {
                try {
                    fileOut.createNewFile();
                }
                catch (IOException ioe) {
                    validInput = false;
                    message = "Could not create OUPUT_FILE";
                }
            }
        }
    }
}
boolean exhaustive = false;
if (validInput) {
       if (args.length>6) {
              if (args[6].equals("1")) {
            exhaustive = true;
              }
       }
}
if (validInput) {
    if (!exhaustive) {
        System.out.println(" > Starting mobility simulation...");
        System.out.println(" > DATA_MODE="+dataInput);
        System.out.println(" > INPUT_FILE="+fileIn.getAbsolutePath());
        System.out.println(" > QUEUE_LENGTH="+queueSize);
        System.out.println(" > TIME_STEP="+timeStep);
        System.out.println(" > GRANULARITY="+granularity);
        System.out.println(" > OUTPUT_FILE="+fileOut.getAbsolutePath());
    }
    if (dataInput.equals("SIMPLE")) {
```

```
                ReadSignalsSimple simple = new ReadSignalsSimple(fileIn, fileOut,
queueSize, timeStep, granularity, exhaustive);
                simple.readSignals();
            } else if (dataInput.equals("CITY_SIMULATOR")) {
                ReadSignalsCitySim citysim = new ReadSignalsCitySim(fileIn, fileOut,
queueSize, timeStep, granularity, exhaustive);
                citysim.readSignals();
            } else if (dataInput.equals("GPS")) {
                ReadSignalsGps gps = new ReadSignalsGps(fileIn, fileOut, queueSize,
timeStep, granularity, exhaustive);
                gps.readSignals();
            }
        } else {
            System.out.println(" > "+message+" (use no parameters for HELP)");
            System.out.println(" > Simulation exiting...");
            System.exit(1);
        }
        if (!exhaustive) {
            System.out.println(" > Ending mobility simulation...");
        }
    }
}
```

simulationSimpleNK.java

```
import java.io.*;
import java.util.*;

import lbs.util.Tank;
import lbs.util.Queue;
import lbs.util.Stigma;
import lbs.util.AbstractStigma;
import lbs.util.SimpleStigma;
import lbs.util.CitySimStigma;
import lbs.util.GpsStigma;

import lbs.interpol.NewtonDividedDifferences;
import lbs.interpol.CubicSpline;
import lbs.interpol.Lagrange;

import lbs.calculus.ReadSignalsSimpleNK;

/* H klash me th main pou diabazei tis parametrous apo command line kai ektelei thn
prosomoiwsh
   gia ola ta queue_length kanontas problepsh gia to k-osto mellontiko stigma*/
public class simulationSimpleNK {
    private static final String FS = System.getProperty("file.separator");
    private static int queueSize;
    private static double timeStep;
    private static int future_step;
    private static File fileIn;
    private static File fileOut;

    public static void main(String args[]) {
        String message = "";
        String dataInput = "";
        String availableInputs = "";
        boolean validInput = true;
        if (args.length<1) {
            System.out.println(" > Usage:");
            System.out.println(" > | param1 = INPUT_FILE {path absolute/relative}");
            System.out.println(" > | param2 = QUEUE_LENGTH {integer}");
            System.out.println(" > | param3 = TIME_STEP {double}");
            System.out.println(" > | param4 = FUTURE_STEP {integer}");
            System.out.println(" > | param5 = OUTPUT_FILE {path absolute/relative}");
            validInput = false;
            message = "|";
        } else {
            if (args.length>0) {
                fileIn = new File(args[0]);
                if (!fileIn.exists()) {
                    validInput = false;
                    message = "INPUT_FILE does no exist";
                }
            } else {
                validInput = false;
                message = "INPUT_FILE not specified";
```

```
            }
            if (validInput) {
                if (args.length>1) {
                    try {
                        Integer Q = new Integer(args[1]);
                        queueSize = Q.intValue();
                    } catch (NumberFormatException nfe) {
                        validInput = false;
                        message = "Invalid QUEUE_LENGTH";
                    }
                } else {
                       validInput = false;
                    message = "QUEUE_LENGTH not specified";
                }
            }
            if (validInput) {
                if (args.length>2) {
                    try {
                        Double Q = new Double(args[2]);
                        timeStep = Q.doubleValue();
                    } catch (NumberFormatException nfe) {
                        validInput = false;
                        message = "Invalid TIME_STEP";
                    }
                } else {
                       validInput = false;
                    message = "TIME_STEP not specified";
                }
            }
            if (validInput) {
                if (args.length>3) {
                       try {
                              Integer B = new Integer(args[3]);
                        future_step = B.intValue();
                       }
                       catch (NumberFormatException nfe) {
                        validInput = false;
                        message = "Invalid FUTURE_STEP";
                       }
                } else {
                       validInput = false;
                    message = "FUTURE_STEP not specified";
                }
            }
            if (validInput) {
                if (args.length>4) {
                    fileOut = new File(args[4]);
                    if (!fileOut.exists()) {
                        try {
                            fileOut.createNewFile();
                        }
                        catch (IOException ioe) {
                            validInput = false;
                            message = "Could not create OUPUT_FILE";
                        }
                    }
                } else {
                    validInput = false;
                    message = "OUTPUT_FILE not specified";
                }
            }
        }
        boolean exhaustive = false;
        if (validInput) {
               if (args.length>5) {
                       if (args[5].equals("1")) {
                    exhaustive = true;
                       }
               }
        }
        if (validInput) {
            if (!exhaustive) {
                System.out.println(" > Starting mobility simulation for SIMPLE N+K
future predictions...");
                System.out.println(" > DATA_MODE="+dataInput);
                System.out.println(" > INPUT_FILE="+fileIn.getAbsolutePath());
                System.out.println(" > QUEUE_LENGTH="+queueSize);
```

```
                System.out.println(" > TIME_STEP="+timeStep);
                System.out.println(" > FUTURE_STEP="+future_step);
                System.out.println(" > OUTPUT_FILE="+fileOut.getAbsolutePath());
            }
            ReadSignalsSimpleNK simpleNK = new ReadSignalsSimpleNK(fileIn, fileOut,
queueSize, timeStep, future_step, exhaustive);
            simpleNK.readSignals();
        } else {
            System.out.println(" > "+message+" (use no parameters for HELP)");
            System.out.println(" > Simulation exiting...");
            System.exit(1);
        }
        if (!exhaustive) {
            System.out.println(" > Ending mobility simulation...");
        }
    }
}
```

simulationCitySimNK.java

```
import java.io.*;
import java.util.*;

import lbs.util.Tank;
import lbs.util.Queue;
import lbs.util.Stigma;
import lbs.util.AbstractStigma;
import lbs.util.SimpleStigma;
import lbs.util.CitySimStigma;
import lbs.util.GpsStigma;

import lbs.interpol.NewtonDividedDifferences;
import lbs.interpol.CubicSpline;
import lbs.interpol.Lagrange;

import lbs.calculus.ReadSignalsCitySimNK;

/* H klash me th main pou diabazei tis parametrous apo command line kai ektelei thn
prosomoiwsh
   gia ola ta queue_length kanontas problepsh gia to k-osto mellontiko stigma*/
public class simulationCitySimNK {
    private static final String FS = System.getProperty("file.separator");
    private static int queueSize;
    private static double timeStep;
    private static int future_step;
    private static File fileIn;
    private static File fileOut;

    public static void main(String args[]) {
        String message = "";
        String dataInput = "";
        String availableInputs = "";
        boolean validInput = true;
        if (args.length<1) {
            System.out.println(" > Usage:");
            System.out.println(" > | param1 = INPUT_FILE {path absolute/relative}");
            System.out.println(" > | param2 = QUEUE_LENGTH {integer}");
            System.out.println(" > | param3 = TIME_STEP {double}");
            System.out.println(" > | param4 = FUTURE_STEP {integer}");
            System.out.println(" > | param5 = OUTPUT_FILE {path absolute/relative}");
            validInput = false;
            message = "|";
        } else {
            if (args.length>0) {
                fileIn = new File(args[0]);
                if (!fileIn.exists()) {
                    validInput = false;
                    message = "INPUT_FILE does no exist";
                }
            } else {
                validInput = false;
                message = "INPUT_FILE not specified";
            }
            if (validInput) {
                if (args.length>1) {
                    try {
                        Integer Q = new Integer(args[1]);
```

```
                    queueSize = Q.intValue();
                } catch (NumberFormatException nfe) {
                    validInput = false;
                    message = "Invalid QUEUE_LENGTH";
                }
            } else {
                   validInput = false;
                message = "QUEUE_LENGTH not specified";
            }
        }
        if (validInput) {
            if (args.length>2) {
                try {
                    Double Q = new Double(args[2]);
                    timeStep = Q.doubleValue();
                } catch (NumberFormatException nfe) {
                    validInput = false;
                    message = "Invalid TIME_STEP";
                }
            } else {
                   validInput = false;
                message = "TIME_STEP not specified";
            }
        }
        if (validInput) {
            if (args.length>3) {
                   try {
                          Integer B = new Integer(args[3]);
                    future_step = B.intValue();
                   }
                   catch (NumberFormatException nfe) {
                    validInput = false;
                    message = "Invalid FUTURE_STEP";
                   }
            } else {
                   validInput = false;
                message = "FUTURE_STEP not specified";
            }
        }
        if (validInput) {
            if (args.length>4) {
                fileOut = new File(args[4]);
                if (!fileOut.exists()) {
                    try {
                        fileOut.createNewFile();
                    }
                    catch (IOException ioe) {
                        validInput = false;
                        message = "Could not create OUPUT_FILE";
                    }
                }
            } else {
                validInput = false;
                message = "OUTPUT_FILE not specified";
            }
        }
    }
    boolean exhaustive = false;
    if (validInput) {
           if (args.length>5) {
                  if (args[5].equals("1")) {
                exhaustive = true;
                  }
           }
    }
    if (validInput) {
        if (!exhaustive) {
            System.out.println(" > Starting mobility simulation for CITY_SIMULATOR
N+K future predictions...");
            System.out.println(" > DATA_MODE="+dataInput);
            System.out.println(" > INPUT_FILE="+fileIn.getAbsolutePath());
            System.out.println(" > QUEUE_LENGTH="+queueSize);
            System.out.println(" > TIME_STEP="+timeStep);
            System.out.println(" > FUTURE_STEP="+future_step);
            System.out.println(" > OUTPUT_FILE="+fileOut.getAbsolutePath());
        }
```

```
            ReadSignalsCitySimNK citysim = new ReadSignalsCitySimNK(fileIn, fileOut,
queueSize, timeStep, future_step, exhaustive);
            citysim.readSignals();
        } else {
            System.out.println(" > "+message+" (use no parameters for HELP)");
            System.out.println(" > Simulation exiting...");
            System.exit(1);
        }
        if (!exhaustive) {
            System.out.println(" > Ending mobility simulation...");
        }
    }
}
```

simulationGpsNK.java

```
import java.io.*;
import java.util.*;

import lbs.util.Tank;
import lbs.util.Queue;
import lbs.util.Stigma;
import lbs.util.AbstractStigma;
import lbs.util.SimpleStigma;
import lbs.util.CitySimStigma;
import lbs.util.GpsStigma;

import lbs.interpol.NewtonDividedDifferences;
import lbs.interpol.CubicSpline;
import lbs.interpol.Lagrange;

import lbs.calculus.ReadSignalsGpsNK;

/* H klash me th main pou diabazei tis parametrous apo command line kai ektelei thn
prosomoiwsh
   gia ola ta queue_length kanontas problepsh gia to k-osto mellontiko stigma*/
public class simulationGpsNK {
    private static final String FS = System.getProperty("file.separator");
    private static int queueSize;
    private static double timeStep;
    private static int future_step;
    private static File fileIn;
    private static File fileOut;

    public static void main(String args[]) {
        String message = "";
        String dataInput = "";
        String availableInputs = "";
        boolean validInput = true;
        if (args.length<1) {
            System.out.println(" > Usage:");
            System.out.println(" > | param1 = INPUT_FILE {path absolute/relative}");
            System.out.println(" > | param2 = QUEUE_LENGTH {integer}");
            System.out.println(" > | param3 = TIME_STEP {double}");
            System.out.println(" > | param4 = FUTURE_STEP {integer}");
            System.out.println(" > | param5 = OUTPUT_FILE {path absolute/relative}");
            validInput = false;
            message = "|";
        } else {
            if (args.length>0) {
                fileIn = new File(args[0]);
                if (!fileIn.exists()) {
                    validInput = false;
                    message = "INPUT_FILE does no exist";
                }
            } else {
                validInput = false;
                message = "INPUT_FILE not specified";
            }
            if (validInput) {
                if (args.length>1) {
                    try {
                        Integer Q = new Integer(args[1]);
                        queueSize = Q.intValue();
                    } catch (NumberFormatException nfe) {
                        validInput = false;
                        message = "Invalid QUEUE_LENGTH";
```

```
                }
            } else {
                   validInput = false;
                message = "QUEUE_LENGTH not specified";
            }
        }
        if (validInput) {
            if (args.length>2) {
                try {
                    Double Q = new Double(args[2]);
                    timeStep = Q.doubleValue();
                } catch (NumberFormatException nfe) {
                    validInput = false;
                    message = "Invalid TIME_STEP";
                }
            } else {
                   validInput = false;
                message = "TIME_STEP not specified";
            }
        }
        if (validInput) {
            if (args.length>3) {
                   try {
                          Integer B = new Integer(args[3]);
                    future_step = B.intValue();
                   }
                   catch (NumberFormatException nfe) {
                    validInput = false;
                    message = "Invalid FUTURE_STEP";
                   }
            } else {
                   validInput = false;
                message = "FUTURE_STEP not specified";
            }
        }
        if (validInput) {
            if (args.length>4) {
                fileOut = new File(args[4]);
                if (!fileOut.exists()) {
                    try {
                        fileOut.createNewFile();
                    }
                    catch (IOException ioe) {
                        validInput = false;
                        message = "Could not create OUPUT_FILE";
                    }
                }
            } else {
                validInput = false;
                message = "OUTPUT_FILE not specified";
            }
        }
    }
    boolean exhaustive = false;
    if (validInput) {
           if (args.length>5) {
                   if (args[5].equals("1")) {
                exhaustive = true;
                   }
           }
    }
    if (validInput) {
        if (!exhaustive) {
            System.out.println(" > Starting mobility simulation for GPS N+K future
predictions...");
            System.out.println(" > DATA_MODE="+dataInput);
            System.out.println(" > INPUT_FILE="+fileIn.getAbsolutePath());
            System.out.println(" > QUEUE_LENGTH="+queueSize);
            System.out.println(" > TIME_STEP="+timeStep);
            System.out.println(" > FUTURE_STEP="+future_step);
            System.out.println(" > OUTPUT_FILE="+fileOut.getAbsolutePath());
        }
        ReadSignalsGpsNK gpsNK = new ReadSignalsGpsNK(fileIn, fileOut, queueSize,
timeStep, future_step, exhaustive);
        gpsNK.readSignals();
    } else {
        System.out.println(" > "+message+" (use no parameters for HELP)");
```

```
            System.out.println(" > Simulation exiting...");
            System.exit(1);
        }
        if (!exhaustive) {
            System.out.println(" > Ending mobility simulation...");
        }
    }
}
```