Ανοικτό Πανεπιστήμιο Κύπρου

Σχολή Θετικών και Εφαρμοσμένων Επιστημών

Μεταπτυχιακή Διατριβή στα Πληροφοριακά Συστήματα

Μετάπτωση Εφαρμογής Ταξινόμησης σε Δικτυακό Περιβάλλον για Βέλτιστη Διαχείριση Πόρων

Ανδρέας-Χαράλαμπος Παπανδρεάδης

Επιβλέπων Καθηγητής
ΔΗΜΗΤΡΙΟΣ ΚΑΛΛΕΣ

Μάιος 2012

Ανοικτό Πανεπιστήμιο Κύπρου

Σχολή Θετικών και Εφαρμοσμένων Επιστημών

Μετάπτωση Εφαρμογής Ταξινόμησης σε Δικτυακό Περιβάλλον για Βέλτιστη Διαχείριση Πόρων

Ανδρέας-Χαράλαμπος Παπανδρεάδης

Επιβλέπων Καθηγητής
Δημήτριος Καλλές

Η παρούσα μεταπτυχιακή διατριβή υποβλήθηκε
προς μερική εκπλήρωση των απαιτήσεων για απόκτηση

μεταπτυχιακού τίτλου σπουδών
στα Πληροφοριακά Συστήματα

από τη Σχολή Θετικών και Εφαρμοσμένων Επιστημών
του Ανοικτού Πανεπιστημίου Κύπρου

Μάιος 2012

Περίληψη

Στα πλαίσια της παρούσας μεταπτυχιακής διατριβής αναπτύχθηκε εφαρμογή Ιστού, η οποία υλοποιεί παραλλαγές του αλγορίθμου αυτόνομης ταξινόμησης Emerge-Sort, ενός αλγορίθμου που συνίσταται στην εφαρμογή απλών τοπικών τελεστών σε αριθμητικές ακολουθίες, ώστε να επιτευχθεί ταξινόμηση. Εκτός από την υλοποίηση παραλλαγών του αλγορίθμου, η εφαρμογή περιλαμβάνει έναν γενετικό αλγόριθμο αξιολόγησής τους, ενώ είναι βασισμένη σε αυτόνομη εφαρμογή που αναπτύχθηκε στα πλαίσια παλαιότερης έρευνας. Χρησιμοποιήθηκαν δύο μοντέλα εκτέλεσης, ένα για εκτέλεση υπολογισμών στην πλευρά του εξυπηρετητή Ιστού, με χρήση της τεχνολογίας των Java Servlets, και ένα για εκτέλεση υπολογισμών στο περιβάλλον της εφαρμογής περιήγησης Ιστού του χρήστη, με χρήση της τεχνολογίας των Java Applets. Μετά το πέρας της υλοποίησης της εφαρμογής εκτελέστηκαν πειράματα για τον έλεγχο ορθότητάς της και την αξιολόγηση των διαφόρων παραλλαγών του αλγορίθμου Emerge-Sort.

Λέξεις Κλειδιά

Νοημοσύνη Σμήνους, Αυτόνομη Ταξινόμηση, Τεχνολογίες Ιστού, Java Servlets, Java Applets, Emerge-Sort

Summary

In this thesis, the development of a Web application, which implements different variations of Emerge-Sort sorting algorithm, is presented. Emerge-Sort is an algorithm which applies local operators in parts of a given number sequence, in order for sorting to emerge. Apart from the implementation of the different variations of Emerge-Sort, a genetic algorithm is used in order to evaluate these variations. The development is based on a standalone Windows application which was developed in previous research project. Two models were implemented; one with server side computing, utilizing Java Servlets technology, and another with client-side computing, utilizing Java Applets technology. Implementation of the Web Application followed rules and techniques of proper software development. After application implementation, experiments regarding execution time and functionality control were conducted.

Keywords

Swarm Intelligence, Sorting, Web Technologies, Java Servlets, Java Applets, Emerge-Sort

Ευχαριστίες

Θα ήθελα να ευχαριστήσω θερμά για το χρόνο που "έκλεψα" από τα αγαπημένα μου πρόσωπα, Μαρία, Κυριάκο - Γεώργιο, Ευφροσύνη και τους αγαπημένους γονείς και αδέλφια κατά την διάρκεια των μεταπτυχιακών σπουδών μου αλλά κυρίως κατά την διάρκεια της εκπόνησης αυτής της διπλωματικής εργασίας.

Θα ήθελα να ευχαριστήσω θερμά τον επιβλέποντα καθηγητή κ. Δημήτριο Καλλέ γιατί μου έδωσε την ευκαιρία να ασχοληθώ με ένα ενδιαφέρον θέμα και αφετέρου για την καθοδήγηση και την ενθάρρυνσή του κατά την διάρκεια εκπόνησης αυτής της διπλωματικής εργασίας.

Περιεχόμενα

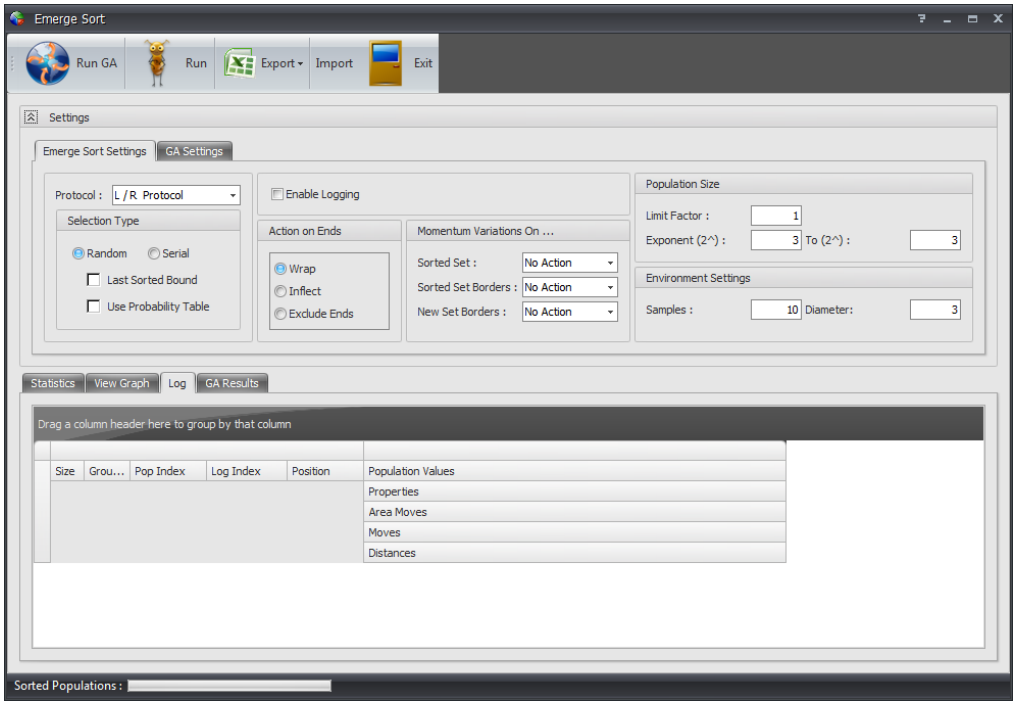
Λίστα Εικόνων

Κεφάλαιο 1
Εισαγωγή

Στα πλαίσια της παρούσας μεταπτυχιακής διατριβής μία αυτόνομη (standalone) εφαρμογή αξιολόγησης του αλγορίθμου αυτόνομης ταξινόμησης Emerge-Sort μετατράπηκε σε εφαρμογή Ιστού (Web Application), με σκοπό να επιτευχθεί καλύτερη διαχείριση των διαθέσιμων υπολογιστικών πόρων.

Ο αλγόριθμος Emerge-Sort επιλύει το πρόβλημα της ταξινόμησης αριθμητικών ακολουθιών με χρήση τεχνικών Νοημοσύνης Σμήνους ακολουθώντας δύο βασικές

προϋποθέσεις: την ισχυρή αυτό-οργάνωση και την έλλειψη πρόθεσης κατεύθυνσης. Ο αλγόριθμος Emerge-Sort, δηλαδή, λειτουργεί χωρίς κεντρικό έλεγχο και δεν γνωρίζει από την αρχή προς ποια κατεύθυνση θα ταξινομήσει. Η έλλειψη κεντρικού ελέγχου διέπει τόσο τη δομή του αλγορίθμου όσο και την εμβέλεια των αλληλεπιδράσεων των μελών της, προς ταξινόμηση, ακολουθίας. Τα μέλη της ακολουθίας δρουν ανεξάρτητα, έχοντας περιορισμένη εμβέλεια αλληλεπίδρασης, και χρησιμοποιούν ένα σύνολο απλών τοπικών τελεστών σύγκρισης και αντιμετάθεσης, ώστε να αλλάξουν τη θέση τους μέσα στην ακολουθία. Η ταξινόμηση της ακολουθίας δημιουργείται αυτόνομα, προκύπτει (emerges) δηλαδή από τις κινήσεις των μελών της ακολουθίας. Το γεγονός ότι δεν ορίζεται από την αρχή η κατεύθυνση ταξινόμησης οδηγεί σε αβεβαιότητα ως προς τη τελική σύγκλιση των ακολουθιών σε ταξινομημένη κατάσταση. Ο αλγόριθμος Emerge-Sort φαίνεται να επιτυγχάνει μία επίδοση της τάξης του $O(n^2)$, έχοντας ένα επιπλέον κόστος σε σχέση με βέλτιστους αλγορίθμους ταξινόμησης, της τάξης του $n / \log n$.



Εικόνα 1.1: Η αυτόνομη εφαρμογή Emerge-Sort UI.

Στα πλαίσια προηγούμενης εργασίας αναπτύχθηκε η αυτόνομη εφαρμογή EmergeSortUI που αξιολογεί παραλλαγές του αλγορίθμου Emerge-Sort και μπορεί να χρησιμοποιηθεί σε συστήματα με λειτουργικό σύστημα Windows. Παρά την επίτευξη του στόχου της αξιολόγησης παραλλαγών του αλγορίθμου Emerge-Sort, η αυτόνομη εφαρμογή EmergeSortUI είχε αρκετούς περιορισμούς όσον αφορά την εκτέλεσή της. Κύριος στόχος της διατριβής ήταν η παροχή της ίδιας λειτουργικότητας με την αυτόνομη εφαρμογή, μέσω μίας νέας εφαρμογής Ιστού με στόχο να αρθούν οι περισσότεροι από τους περιορισμούς εκτέλεσης. Η εφαρμογή Ιστού δεν απαιτεί

εγκατάσταση στο σύστημα κάθε χρήστη, μειώνοντας το κόστος χρήσης της, ενώ σε περίπτωση επέκτασης της λειτουργικότητας χρειάζεται να αναβαθμιστεί μόνο κεντρικά ώστε να γίνει διαθέσιμη σε όλους τους χρήστες. Χρησιμοποιώντας τεχνολογίες Ιστού και τη γλώσσα προγραμματισμού Java, η εφαρμογή Ιστού μπορεί να εκτελεστεί στα περισσότερα μοντέρνα λειτουργικά συστήματα, αρκεί να υπάρχει στο σύστημα του χρήστη εγκατεστημένη μία εφαρμογή περιήγησης Ιστού (Web Browser). Η εφαρμογή Ιστού δρα ως νέα διεπαφή που χρησιμοποιεί τις βιβλιοθήκες που έχουν ήδη αναπτυχθεί και απλά αντικαθιστά το γραφικό περιβάλλον της αυτόνομης εφαρμογής. Με αυτό τον τρόπο επετεύχθη επαναχρησιμοποίηση του κώδικα και το κόστος υλοποίησης, σε ανθρωποώρες, παρέμεινε σχετικά χαμηλό.

Σε πρώτο στάδιο η εφαρμογή Ιστού αναπτύχθηκε χρησιμοποιώντας την τεχνολογία των Java Servlets, ώστε η επιθυμητή λειτουργικότητα να γίνει διαθέσιμη στους χρήστες με το μοντέλο πελάτη-εξυπηρετητή, και το βάρος των υπολογισμών να ανήκει στον εξυπηρετητή που φιλοξενεί τα σχετικά Java Servlets. Με αυτό τον τρόπο ακόμα και χρήστες με συστήματα χωρίς ικανούς υπολογιστικούς πόρους μπορούν να αναθέσουν στον εξυπηρετητή την εκτέλεση της εφαρμογής και να λάβουν εκ των υστέρων τα παραγόμενα αποτελέσματα. Καθώς ο εξυπηρετητής διαθέτει περισσότερους πόρους από έναν συνηθισμένο ηλεκτρονικό υπολογιστή, οι υπολογισμοί εκτελούνται ταχύτερα. Οι «επιλογές» των χρηστών, που μέχρι πρότινος εισάγονταν μέσω του γραφικού περιβάλλοντος της αυτόνομης εφαρμογής, τώρα πλέον εισάγονται μέσω μίας HTML φόρμας. Τα δεδομένα της φόρμας μεταφέρονται σε ένα Java Servlet, το οποίο αναλαμβάνει να εκτελέσει τους απαιτούμενους υπολογισμούς, χρησιμοποιώντας την προγραμματιστική λογική και τις βιβλιοθήκες που έχουν ήδη υλοποιηθεί για την αυτόνομη εφαρμογή. Αφού εκτελεστούν οι απαραίτητοι υπολογισμοί, τα παραγόμενα αποτελέσματα επιστρέφονται στο χρήστη με τη μορφή HTML ιστοσελίδας. Η εφαρμογή Ιστού γίνεται διαθέσιμη μέσω ενός εξυπηρετητή Apache Tomcat (ο οποίος θα είναι ο servlet container της προτεινόμενης αρχιτεκτονικής). Για την εφαρμογή υλοποιήθηκαν Servlet κλάσεις συμβατές με το Java Servlet API που υλοποιεί η Java EE.

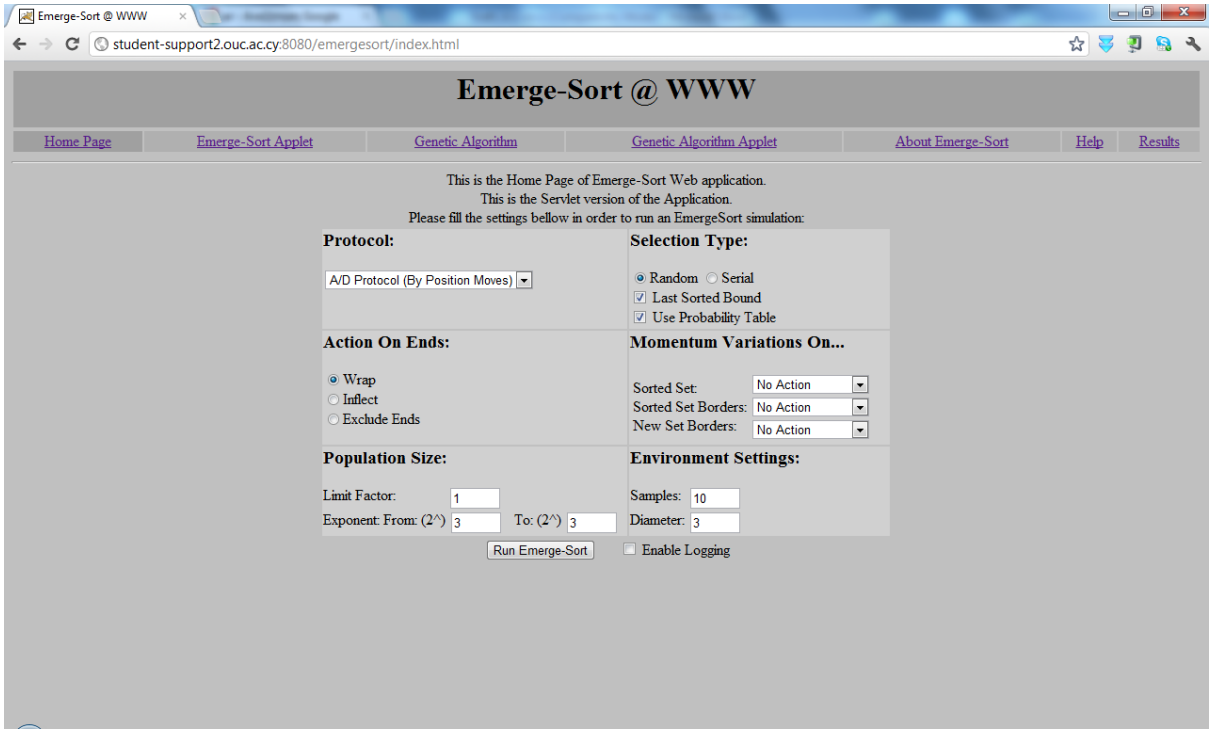
Σε δεύτερο στάδιο χρησιμοποιήθηκε η τεχνολογία των Java Applets (μικροεφαρμογές Java), ώστε οι υπολογισμοί να είναι δυνατό να εκτελούνται και στο περιβάλλον του χρήστη (client-side). Αυτό κρίθηκε απαραίτητο καθώς αν αυξηθεί ο αριθμός χρηστών που χρησιμοποιούν την εγκατεστημένη στον εξυπηρετητή εφαρμογή Ιστού του 1ου σταδίου, υπάρχει περίπτωση να παρατηρηθεί καθυστέρηση στην εκτέλεση των υπολογισμών. Με τη χρήση των Java Applets παρέχεται η δυνατότητα τοπικής εκτέλεσης της εφαρμογής χωρίς επιβάρυνση του εξυπηρετητή διατηρώντας πλεονεκτήματα όπως η μη απαίτηση εγκατάστασης και

αναβάθμισης και η δια-λειτουργικότητα σε πολλαπλές διαφορετικές πλατφόρμες. Για την εκτέλεση των Java Applets, απαιτείται ο κάθε χρήστης να έχει εγκατεστημένη στο σύστημα που χρησιμοποιεί μία τελευταία έκδοση της εικονικής μηχανής της Java (JVM – Java Virtual Machine). Επιπλέον, θα πρέπει το σύστημα να διαθέτει υλικό ικανό να εκτελέσει τους ζητούμενους υπολογισμούς. Τα Java Applets ενσωματώνονται σε αντίστοιχες HTML σελίδες που προστέθηκαν στην αρχική εφαρμογή Ιστού, ώστε να εκτελούνται σε κάποια εφαρμογή περιήγησης Ιστού (Web Browser) και περιλαμβάνουν τόσο την προγραμματιστική λογική όσο και ένα γραφικό περιβάλλον (GUI) στο οποίο εισάγονται οι επιθυμητές επιλογές. Για την υλοποίηση των Java Applets χρησιμοποιήθηκε το πακέτο java.applet.

Η αρχική, αυτόνομη, εφαρμογή (EmergeSortUI) υλοποιεί παραλλαγές του αλγορίθμου Emerge-Sort και, επιπλέον, χρησιμοποιεί έναν γενετικό αλγόριθμο ως μέσο αξιολόγησής τους. Η λειτουργικότητα της εκτέλεσης των παραλλαγών του αλγορίθμου Emerge-Sort αναπτύχθηκε με τη γλώσσα προγραμματισμού Java και μετατράπηκε σε βιβλιοθήκη τύπου jar (Java Archive). Για τη λειτουργικότητα της εκτέλεσης του γενετικού αλγορίθμου χρησιμοποιήθηκε μία έτοιμη βιβλιοθήκη εκτέλεσης γενετικών αλγορίθμων, η οποία επεκτάθηκε και εξήχθη και αυτή σε βιβλιοθήκη τύπου jar. Το γραφικό περιβάλλον της αυτόνομης εφαρμογής αναπτύχθηκε εντελώς χωριστά από την παρεχόμενη λειτουργικότητα με χρήση της γλώσσας προγραμματισμού C# και του περιβάλλοντος ανάπτυξης Microsoft Visual Studio. Το γραφικό περιβάλλον χρησιμοποιεί τις προαναφερθείσες βιβλιοθήκες για να εκτελέσει τη λειτουργικότητα που επιθυμεί ο χρήστης. Προϋπόθεση για την εγκατάσταση της εφαρμογής είναι η εγκατάσταση της βιβλιοθήκης Microsoft .Net Framework Version 2.0 [26]. Η αυτόνομη εφαρμογή εκτελεί υπολογισμούς μεγάλης διάρκειας και απαιτεί ένα περιβάλλον με υλικό (hardware) ικανό να εκτελέσει τους ζητούμενους υπολογισμούς, γεγονός που μπορεί να μην επιτρέψει την αποδοτική εκτέλεσή της σε παλαιότερα συστήματα και να οδηγήσει σε δέσμευση πολλών πόρων του συστήματος κατά την εκτέλεσή της. Τέλος, σημαντικός περιορισμός θεωρείται το γεγονός πως αν η λειτουργικότητα της εφαρμογής επεκταθεί ή διορθωθεί, θα πρέπει όλοι οι υπάρχοντες χρήστες της να εγκαταστήσουν τη νέα έκδοσή της, έχοντας ένα σχετικό κόστος σε ανθρωποώρες.

Η ανάπτυξη της Εφαρμογής Ιστού έγινε τμηματικά ώστε να εξασφαλιστεί η ορθή λειτουργικότητά της. Αφού ορίστηκαν οι προδιαγραφές, επιλέχθηκε το περιβάλλον ανάπτυξης, σχεδιάστηκε η εφαρμογή και ξεκίνησε η υλοποίησή της. Κάποια προβλήματα σχετικά με τη χρήση βιβλιοθηκών επιλύθηκαν με αναζήτηση λύσεων στο διαδίκτυο. Μετά το πέρας της υλοποίησης αλλά και στα ενδιάμεσα στάδια η εφαρμογή Ιστού αξιολογήθηκε ως προς τη

λειτουργικότητά της. Η τρέχουσα μορφή της βρίσκεται στη διεύθυνση http://student-support2.ouc.ac.cy:8080/emergesort/.



Εικόνα 1.2: Η εφαρμογή Ιστού Emerge-Sort @ WWW.

Στο επόμενο κεφάλαιο γίνεται μία σύντομη βιβλιογραφική επισκόπηση και παρουσιάζονται οι κυριότερες τεχνολογίες και τα σημαντικότερα εργαλεία που χρησιμοποιήθηκαν κατά την ανάπτυξη της εφαρμογής. Στο 3ο και στο 4ο κεφάλαιο παρουσιάζεται η λειτουργικότητα και η μέθοδοι υλοποίησης της εφαρμογής Ιστού για εκτέλεση στο περιβάλλον του εξυπηρετητή και στο περιβάλλον του χρήστη αντίστοιχα. Στο 5ο κεφάλαιο παρατίθενται τα αποτελέσματα ορισμένων πειραμάτων που εκτελέστηκαν με τη χρήση της εφαρμογής και, τέλος, στο 6ο κεφάλαιο γίνεται μία σύνοψη των μεθόδων που ακολουθήθηκαν, των αποτελεσμάτων που προέκυψαν και πιθανών μελλοντικών επεκτάσεων.

Κεφάλαιο 2
Επιστημονικό και Τεχνικό Υπόβαθρο

Σε αυτό το κεφάλαιο γίνεται μία σύντομη βιβλιογραφική επισκόπηση. Αρχικά αναφέρονται κάποιες παλαιότερες εργασίες σχετικά με τη Νοημοσύνη Σμήνους και ειδικότερα το πρόβλημα της Αυτόνομης Ταξινόμησης. Στη συνέχεια αναλύεται συνοπτικά ο αλγόριθμος Emerge-Sort και η αυτόνομη εφαρμογή αξιολόγησης EmergeSortUI που αναπτύχθηκε στα πλαίσια προηγούμενης εργασίας. Στο δεύτερο μέρος του κεφαλαίου περιγράφονται οι τεχνολογίες και τα εργαλεία που χρησιμοποιήθηκαν στα πλαίσια της παρούσας μεταπτυχιακής διατριβής.

2.1 Βιβλιογραφική Έρευνα

Η εφαρμογή που αναπτύχθηκε στα πλαίσια της παρούσας διατριβής επιλύει το πρόβλημα της ταξινόμησης αριθμητικών ακολουθιών με χρήση παραλλαγών του αλγορίθμου Emerge-Sort, ενός αλγορίθμου που χρησιμοποιεί τεχνικές Νοημοσύνης Σμήνους ώστε να επιτευχθεί η επιθυμητή ταξινόμηση και να προκύψει με αυτόνομο τρόπο (Αυτόνομη

Ταξινόμηση). Στις επόμενες παραγράφους αναλύονται οι σχετικές έννοιες και παρουσιάζονται παλαιότερες προσπάθειες στον εν λόγω τομέα έρευνας.

2.1.1 Νοημοσύνη Σμήνους

Ως Νοημοσύνη Σμήνους (Swarm Intelligence) ορίζεται η συλλογική συμπεριφορά αποκεντρωμένων συστημάτων, ικανών να επιδείξουν αυτό-οργάνωση, υψηλή προσαρμοστικότητα, ευελιξία και ευρωστία. Η Νοημοσύνη Σμήνους είναι όρος της Τεχνητής Νοημοσύνης και είναι εμπνευσμένη από φυσικά αποκεντρωμένα συστήματα, όπως οι αποικίες εντόμων, τα οποία χωρίς κεντρική διαχείριση, τείνουν να συγκλίνουν σε αναδυόμενες (emerging), επιθυμητές συμπεριφορές. Παρά το γεγονός ότι κάθε μία από τις συμμετέχουσες οντότητες έχει τοπική υπόσταση και εμβέλεια, χρησιμοποιώντας τοπικά ερεθίσματα και αντιδρώντας σε αυτά, είναι δυνατό το σύστημα να οδηγηθεί σε συλλογική απόφαση χωρίς κανέναν κεντρικό έλεγχο.

Το 1997 δόθηκε από τον Bonabeau ένας περιγραφικός και ταυτόχρονα συνοπτικός ορισμός για την αυτό-οργάνωση αποκεντρωμένων συστημάτων: «Η αυτό-οργάνωση αποτελείται από ένα σύνολο δυναμικών μηχανισμών, όπου η δομή εμφανίζεται στο συλλογικό επίπεδο ως αποτέλεσμα αλληλεπιδράσεων συστατικών χαμηλότερου επιπέδου. Οι κανόνες που καθορίζουν τις αλληλεπιδράσεις ανάμεσα στις συνιστώσες μονάδες του συστήματος εκτελούνται βάσει αποκλειστικά τοπικής πληροφορίας χωρίς δυνατότητα αναφοράς σε συλλογικό σχεδιασμό, ο οποίος αποτελεί περισσότερο μία αναδυόμενη ιδιότητα του συστήματος παρά μία ιδιότητα που του έχει επιβληθεί από εξωτερική επιρροή.» [01]

Σημαντικό χαρακτηριστικό συστημάτων με δυνατότητα αυτόνομης οργάνωσης είναι ο τρόπος με τον οποίο οδηγούνται σε μία κοινή, φαινόμενη (emerging) συμπεριφορά. Όπως αναφέρεται σε σχετική εργασία των Wolf και Holvoet [07], «Ένα σύστημα επιδεικνύει φαινόμενη (προκύπτουσα) συμπεριφορά όταν σε μακροσκοπικό επίπεδο παρατηρούνται φαινόμενα συμπεριφορών, δομών, ιδιοτήτων ή προτύπων, τα οποία απορρέουν δυναμικά από αλληλεπιδράσεις των τμημάτων σε μικροσκοπικό επίπεδο. Αυτά τα φαινόμενα είναι καινοφανή όσον αφορά τα μέρη του συστήματος». Η προκύπτουσα κοινή συμπεριφορά ενός συστήματος καταδεικνύει ότι το σύστημα αυτό είναι κάτι περισσότερο από το άθροισμα των οντοτήτων που το αποτελούν. Η συνολική λειτουργικότητα ενός αυτό-οργανούμενου συστήματος δεν προκύπτει άμεσα από την επιμέρους λειτουργικότητα των οντοτήτων που το αποτελούν, αλλά

έμμεσα και με αμφίδρομο τρόπο, οδηγεί σε ένα ευφυές αποτέλεσμα, που καθοδηγεί τη συμπεριφορά των οντοτήτων και ανατροφοδοτείται από τις μεταξύ τους αλληλεπιδράσεις.

Κύρια πηγή έμπνευσης των θεωριών σχετικά με την αυτό-οργάνωση και την φαινόμενη συμπεριφορά συστημάτων, αποτέλεσαν φυσικά αυτό-οργανούμενα σύνολα όπως οι αποικίες εντόμων ή τα σμήνη πτηνών. Σε αυτά τα σύνολα διακρίνουμε κατά κανόνα τα παρακάτω χαρακτηριστικά:

- **θετική ανάδραση (positive feedback):** Το χαρακτηριστικό αυτό εξασφαλίζει την ενίσχυση των θετικών αποτελεσμάτων και συχνά εφαρμόζεται ως συντελεστής βάρους ανάλογος με την ποιότητα των αποτελεσμάτων της επιμέρους δράσης μίας οντότητας. Ένα παράδειγμα από φυσικό σύστημα είναι η προσέλκυση εντόμων από άλλα έντομα (συνήθως με κινήσεις με συγκεκριμένο πρότυπο) προς περιοχές με καλύτερη ποιότητα τροφής.

- **αρνητική ανάδραση (negative feedback):** Το χαρακτηριστικό αυτό είναι η αντίθετη έκφανση του προηγούμενου και εξασφαλίζει την υποβάθμιση των δράσεων που προκαλούν αρνητικά αποτελέσματα. Ένα παράδειγμα από τη φύση είναι η συνεννόηση σμηνών πτηνών ή εντόμων για απομάκρυνση από περιοχές όπου δεν υπάρχει πλέον τροφή με τη χρήση ειδικών σχηματισμών.

- **ενίσχυση των διακυμάνσεων (amplification of fluctuations):** Το χαρακτηριστικό αυτό εισάγει έναν παράγοντα τύχης δίνοντας τη δυνατότητα σε ένα σύστημα να οδηγηθεί σε νέα αποτελέσματα και να αποκτήσει ευελιξία, ώστε να ανταποκρίνεται αποδοτικότερα σε αλλαγές του περιβάλλοντος που θα μπορούσαν να διαταράξουν τη συμπεριφορά του. Ένα παράδειγμα από τις αποικίες εντόμων είναι περιπτώσεις όπου τυχαίες αλλαγές κατεύθυνσης μπορεί να οδηγήσουν σε νέες πηγές τροφής ή σε καλύτερες εν γένει συνθήκες που δε θα ήταν δυνατό να βρεθούν σε ένα πιο αυστηρό μοντέλο.

- **πολλαπλά είδη αλληλεπιδράσεων (multiple interactions):** Το χαρακτηριστικό αυτό αφορά τις αλληλεπιδράσεις των οντοτήτων του συστήματος με το περιβάλλον και άλλες οντότητες. Στη βιβλιογραφία συναντάται ο όρος stigmergy, που αποτελείται από τις ελληνικές λέξεις στίγμα και έργο. Χαρακτηριστικό παράδειγμα του φυσικού περιβάλλοντος αποτελούν τα έντομα που επιτυγχάνουν κοινούς στόχους, όπως η εύρεση

συντομότερων μονοπατιών, μέσω ουσιών (φερομόνες) που απελευθερώνουν κατά την κίνησή τους. Τα έντομα ανταποκρίνονται σε αυτές τις ουσίες έτσι τα καλύτερα μονοπάτια προσελκύουν όλο και περισσότερα άτομα.

Οι ιδιότητες συστημάτων όπως αυτά που προαναφέρθηκαν αποκτούν ιδιαίτερη σημασία στο ετερογενές περιβάλλον του Διαδικτύου, όπως αυτό έχει διαμορφωθεί. Στο περιβάλλον αυτό που αλλάζει διαρκώς και ο όγκος των συναλλαγών διαρκώς μεταβάλλεται, τα εμπλεκόμενα συστήματα και οι επιθυμητές εφαρμογές πρέπει να αναπτύσσονται με αποκεντρωμένα πρότυπα ώστε ανταποκρίνονται σε διακυμάνσεις του περιβάλλοντος αλλά και των τμημάτων που τις αποτελούν. Τα παραδοσιακά πρωτόκολλα με λήψη αποφάσεων σε κεντρικό επίπεδο, που βασίζονται στην ορθή λειτουργία μιας ισχυρής οντότητας ελέγχου, μπορεί να μοιάζουν ιδανικά σε μικρά και στατικά σενάρια, αλλά σε περίπλοκα και δυναμικά συστήματα αποδεικνύονται ανεπαρκή ή μη αποδοτικά και σταδιακά αντικαθίστανται.

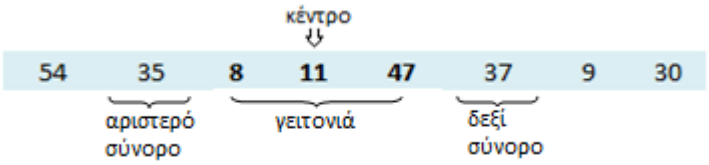
2.1.2 Η Αυτόνομη Ταξινόμηση και ο Αλγόριθμος Emerge-Sort

Το πρόβλημα της ταξινόμησης ακολουθιών είναι ένας από τους πλέον εξαντλητικά μελετημένους τομείς της επιστήμης των υπολογιστών. Έχουν αναπτυχθεί στο πέρασμα των χρόνων πολλοί αλγόριθμοι με υψηλή αποδοτικότητα, οι οποίοι μάλιστα συνέβαλαν και βελτίωσαν αλγορίθμους για πολλά άλλα προβλήματα. Αλγόριθμοι όπως ο merge-sort και ο quick-sort [04] με πολυπλοκότητες κοντά στο O(nlogn) αποδείχθηκαν πρακτικά εξαιρετικοί και η χρήση τους καθιερώθηκε. Το ερευνητικό ενδιαφέρον δεν έπαψε αλλά στράφηκε στην ανάπτυξη μεθόδων για καλύτερο χειρισμό ειδικών περιπτώσεων, όπως ακολουθίες που είναι εν μέρει ταξινομημένες ή που προέρχονται από κάποια ειδική κατανομή, όπως στην περίπτωση των αλγορίθμων spread-sort[05] και bucket-sort[06]. Σε αυτό το πλαίσιο, παλαιότεροι, μη βέλτιστοι αλγόριθμοι ταξινόμησης, όπως ο Insertion-Sort με πολυπλοκότητα $O(n^2)$, παραμένουν αντικείμενο μελέτης με στόχο τη δημιουργία παραλλαγών όπως ο Library-Sort που δημοσιεύτηκε το 2006 [08]. Με την ανάπτυξη των τεχνολογιών δικτύωσης, την ύπαρξη πολλαπλών επεξεργαστών στο ίδιο σύστημα αλλά και τη δημιουργία Υπολογιστικών Πλεγμάτων (Computational GRIDs), το επιστημονικό ενδιαφέρον στράφηκε σε αλγορίθμους κατανεμημένης ταξινόμησης, οι οποίοι περιλαμβάνοντας τεχνικές παράλληλης εκτέλεσης επιτυγχάνουν γραμμικές O(n) ή και καλύτερες πολυπλοκότητες[09]. Τα τελευταία χρόνια αρκετές ερευνητικές ομάδες ανέπτυξαν μεθόδους ταξινόμησης οι οποίες κάνουν χρήση τεχνικών

Νοημοσύνης Σμήνους. Η κοινή συμπεριφορά αποικιών που ακολουθούν, μετά από αυτό-οργάνωση, συγκεκριμένη κατεύθυνση μπορεί να αντιστοιχηθεί με ένα πρόβλημα ταξινόμησης.

Οι τεχνικές της Νοημοσύνης Σμήνους, εκτός από την ταξινόμηση ακολουθιών μπορούν να εφαρμοσθούν σε αρκετά προβλήματα βελτιστοποίησης. Στο βιβλίο Swarm Intelligence [01] των Bonabeau και Dorigo περιγράφονται διάφορες εφαρμογές τεχνικών Νοημοσύνης Σμήνους όπως η δρομολόγηση σε δίκτυα επικοινωνιών, η ανάθεση διεργασιών και πόρων, η τμηματοποίηση γράφων και ρομποτικές εφαρμογές μαζί με τα φυσικά μοντέλα που αντιστοιχούν σε αυτά όπως η αυτό-οργάνωση αποικιών εντόμων. Το ερευνητικό πεδίο της Νοημοσύνης Σμήνους είναι εξαιρετικά ενεργό. Αλγόριθμοι Νοημοσύνης Σμήνους όπως ο Cuckoo Search [10] και ο αλγόριθμος Firefly[11] έχουν προταθεί για προβλήματα βελτιστοποίησης. Αλγόριθμοι όπως ο "Inteligent Water Drops"[12] και ο "River Formation Dynamics"[13] που βασίζονται στη συμπεριφορά ρευστών χρησιμοποιούνται για προβλήματα συντομότερου μονοπατιού σε γράφους. Η Νοημοσύνη Σμήνους μπορεί να βρει εφαρμογή και σε κλασσικά προβλήματα της επιστήμης υπολογιστών όπως η ανάσυρση δεδομένων (Data Mining) [14].

Στην κατηγορία αλγορίθμων Νοημοσύνης Σμήνους ανήκει ο αλγόριθμος Αυτόνομης Ταξινόμησης EmergeSort. Ο αλγόριθμος Emerge-Sort αντιμετωπίζει το πρόβλημα ταξινόμησης, εφαρμόζοντας σε δύο βασικές προϋποθέσεις: την ισχυρή αυτό-οργάνωση και την έλλειψη πρόθεσης κατεύθυνσης. Πρόκειται για έναν αλγόριθμο ταξινόμησης χωρίς κεντρική διαχείριση, ο οποίος δεν γνωρίζει από την αρχή προς ποια κατεύθυνση θα ταξινομήσει. Η έλλειψη κεντρικού ελέγχου εφαρμόζεται τόσο σε δομικό επίπεδο όσο και σε επίπεδο εμβέλειας των αλληλεπιδράσεων των εμπλεκόμενων μελών της δοθείσης ακολουθίας. Οι αριθμητικές ακολουθίες θεωρούνται «συστήματα αυτόνομων οντοτήτων» των οποίων τα μέλη έχουν περιορισμένη εμβέλεια επικοινωνίας και δρουν ανεξάρτητα, βάσει ενός πρωτοκόλλου – ενός συνόλου απλών τοπικών τελεστών σύγκρισης και αντιμετάθεσης. Η έλλειψη προϋπόθεσης κατεύθυνσης οδηγεί σε αβεβαιότητα ως προς τη τελική σύγκλιση των ακολουθιών σε ταξινόμηση. Προφανώς, αν τα μέλη της ακολουθίας είχαν δεδηλωμένη προτίμηση ως προς την κατεύθυνση ταξινόμησης, τότε η σύγκλιση της ακολουθίας σε ταξινόμηση θα ήταν δεδομένη και ο αλγόριθμος τετριμμένος. Εκτός από την επίλυση του προβλήματος της ταξινόμησης, ο Emerge-Sort επιδιώκει να ανακαλύψει το μικρότερο σύνολο τοπικών τελεστών, ων οποίων μπορεί να οδηγήσει σε λύση με ένα λογικά αποδεκτό κόστος. Όπως αναφέρεται και στη σχετική εργασία[02], ο Emerge-Sort φαίνεται να επιτυγχάνει μία επίδοση της τάξης του $O(n^2)$. Το τίμημα των περιορισμών του Emerge-Sort σε σχέση με βέλτιστους αλγορίθμους ταξινόμησης φαίνεται να είναι της τάξης του $n / \log n$.



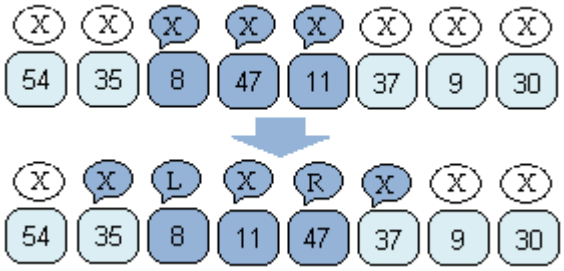
Εικόνα 2.1: Η γειτονιά όπως ορίζεται στον αλγόριθμο Emerge-Sort (Πηγή [03]).

Με τον αλγόριθμο Emerge-Sort η ταξινόμηση εκτελείται σε γύρους (rounds). Σε κάθε γύρο εξετάζεται ένα μόνο υποσύνολο της ακολουθίας, μία γειτονιά, που στην αρχική έκδοση του αλγορίθμου έχει 3 αριθμούς (μέλη). Το κέντρο της γειτονιάς είναι η θέση που επιλέγεται προς επεξεργασία σε κάθε γύρο. Τα άκρα της γειτονιάς αποτελούν τα όριά της. Μέλη της ίδιας γειτονιάς μπορούν να επικοινωνούν μεταξύ τους με αμελητέο κόστος. Μέλη που βρίσκονται στα όρια της γειτονιάς μπορούν να επικοινωνήσουν με μέλη εκτός των ορίων της και ανήκουν σε διπλανές γειτονιές. Παρά τη δυνατότητα επικοινωνίας των μελών μίας γειτονιάς και τη γνώση τους ως προς την κατάσταση ταξινόμησης της γειτονιάς τους, δεν υπάρχει γνώση της κατάστασης των υπόλοιπων γειτονιών της ακολουθίας. Επομένως, οι αποφάσεις κίνησης προς μία κατεύθυνση βασίζονται σε τοπικά κριτήρια (την τοπική τους εμβέλεια) και ένα παράγοντα ορμής (momentum) που χαρακτηρίζει τα μέλη. Ο παράγοντας ορμής είναι ένας τριαδικός τελεστής με τιμές, R = 'Right' L='Left' και X = 'Don't care', που χαρακτηρίζει κάθε μέλος και αποτελεί για αυτό ένα είδος στοιχειώδους σύντομης μνήμης. Κάθε μέλος «θυμάται» υπό συνθήκες την κατεύθυνση προς την οποία είχε κινηθεί ή σκόπευε να κινηθεί την τελευταία φορά.

Σημαντικό ρόλο διαδραματίζει το πρωτόκολλο που ακολουθείται, το οποίο δεν είναι παρά ένα σύνολο από κανόνες που αξιολογούν τη θέση των ατόμων μέσα στη γειτονιά σε σχέση με τον παράγοντα ορμής που φέρουν. Το αρχικό πρωτόκολλο του Emerge-Sort προβλέπει ότι στην αρχή όλα τα άτομα έχουν αδιάφορο παράγοντα ορμής – X. Η επιλογή του κέντρου της κάθε γειτονιάς γίνεται σειριακά. Με αυτό τον τρόπο ο αλγόριθμος ξεκινά από το 1ο μέλος αριθμό) της ακολουθίας και προχωρά εξετάζοντας σε κάθε γύρο μία θέση δεξιότερα, μέχρι το τελευταίο μέλος και πάλι από την αρχή, έως ότου επέλθει ταξινόμηση είτε σε αύξουσα είτε σε φθίνουσα διάταξη. Κάθε φορά που επιλέγεται μία γειτονιά, εξετάζεται πρώτα αν είναι ήδη ταξινομημένη. Επειδή δεν επιβάλλεται κατεύθυνση, αν η γειτονιά είναι ταξινομημένη, ο αλγόριθμος προχωρά στον επόμενο γύρο. Αν όμως η εν λόγω γειτονιά δεν είναι ταξινομημένη, τότε εφαρμόζονται μία σειρά από κριτήρια που βασίζονται στους παράγοντες ορμής των μελών της. Αν όλα τα μέλη φέρουν αδιάφορο παράγοντα ορμής τότε επιλέγεται η κατεύθυνση με το μικρότερο κόστος. Τα μέλη της γειτονιάς αλλάζοντας θέσεις μεταξύ τους προς την επιλεγμένη κατεύθυνση. Σε αυτή τη φάση αθροίζονται οι κινήσεις του κάθε μέλους και ανατίθενται παράγοντες ορμής δίνοντας στο

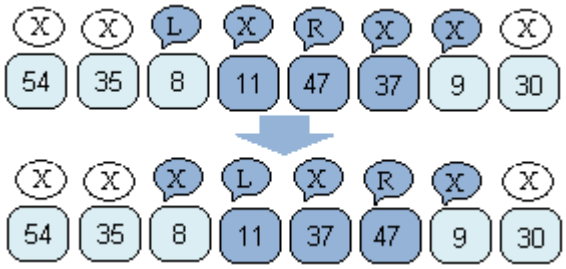
μέλος που βρίσκεται αριστερά του κέντρου τον παράγοντα L, στο μέλος δεξιά του κέντρου παράγοντα R, ενώ το κεντρικό μέλος λαμβάνει παράγοντα X. Επιπλέον, μέλη που συνορεύουν με την εν λόγω γειτονιά προσλαμβάνουν αδιάφορο παράγοντα. Ειδική μέριμνα λαμβάνεται για την αντιμετώπιση συνθηκών εγκλωβισμού μελών σε άκρα της ακολουθίας.

Τα επόμενα δύο σχήματα προέρχονται από παλαιότερη σχετική εργασία [03] και περιγράφουν ένα παράδειγμα ανάθεσης παραγόντων ορμής. Στην Εικόνα 2.2 παρουσιάζεται η κατάσταση ενός μέρους μιας ακολουθίας όπου όλα τα άτομα έχουν αδιάφορο παράγοντα ορμής. Στον πρώτο γύρο επιλέγεται η γειτονιά με κέντρο τον αριθμό 47. Επειδή η γειτονιά αυτή δεν είναι ταξινομημένη και οι παράγοντες όλων των μελών της είναι X (αδιάφοροι), η γειτονιά ταξινομείται προς την κατεύθυνση που απαιτεί τις λιγότερες εναλλαγές (swaps), δηλαδή σε αύξουσα διάταξη. Επιπλέον, τα όρια της γειτονιάς, δηλαδή οι αριθμοί 8 και 11 παίρνουν παράγοντες ορμής L και R αντίστοιχα, ενώ το νέο κέντρο της γειτονιάς παραμένει με παράγοντα X. Τέλος, επειδή οι συνοριακοί αριθμοί της γειτονιάς, το 35 και το 37, έχουν ήδη αδιάφορο παράγοντα, δεν επηρεάζονται σε αυτόν τον γύρο.



Εικόνα 2.2: Ανάθεση παραγόντων ορμής L, R, X μετά την ταξινόμηση γειτονιάς (1/2) (Πηγή [03]).

Στην εικόνα 2.3 παρουσιάζεται η εκτέλεση του επόμενου γύρου, όπου η τρέχουσα γειτονιά αποτελείται από τα μέλη 11, 47, 37. Σε αυτό το γύρο το μέλος 47 «θυμάται» ότι θα ήθελε να κινηθεί προς τα δεξιά (αφού έχει παράγοντα ορμής R από τον προηγούμενο γύρο). Επειδή στη γειτονιά δεν υπάρχουν άτομα με αντίθετο παράγοντα ορμής (L) η κατεύθυνση που ορίζει το 47 επικρατεί και η γειτονιά ταξινομείται και πάλι σε αύξουσα διάταξη. Στη συνέχεια αποδίδονται οι παράγοντες στα όρια της γειτονιάς, ενώ τα συνοριακά της άτομα, το 8 και το 9 προσλαμβάνουν αδιάφορο παράγοντα..

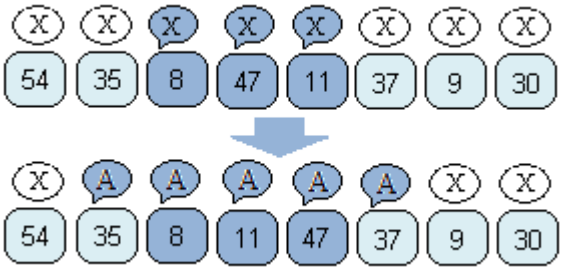


Εικόνα 2.3: Ανάθεση παραγόντων ορμής L, R, X μετά την ταξινόμηση γειτονιάς (2/2) (Πηγή [03]).

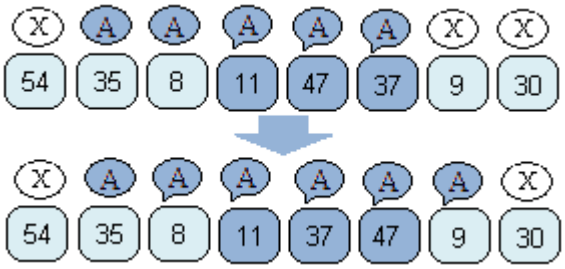
2.1.3 Η Αυτόνομη Εφαρμογή EmergeSortUI

Στα πλαίσια πρόσφατης εργασίας [03], ο αλγόριθμος Emerge-Sort αναλύθηκε πιο διεξοδικά και αναπτύχθηκαν παραλλαγές του. Οι παραλλαγές προέκυψαν από διαφορές στον τρόπο διάσχισης της ακολουθίας, στην επιλογή του κέντρου της γειτονιάς, στο μέγεθος της γειτονιάς, και στις ενέργειες στα άκρα της ακολουθίας. Δημιουργήθηκαν επίσης πρωτόκολλα με διαφορετικούς παράγοντες ορμής ενώ διαφοροποιήθηκε επίσης και ο τρόπος ή ο τόπος εφαρμογής τους.

Επεκτείνοντας το αρχικό πρωτόκολλο του Emerge-Sort, με σκοπό να ενισχυθεί η πληροφορία των παραγόντων ορμής, δημιουργήθηκε ένα νέο πρωτόκολλο βασιζόμενο σε παράγοντες ορμής με διαφορετική σημασία. Ο τελεστής παραμένει τριαδικός αλλά οι νέες πιθανές τιμές είναι A (Ascending, - Αύξων), D (Descending – Φθίνων) και (X - Αδιάφορος). Οι νέοι παράγοντες ορμής δεν περιγράφουν αν κάποιο μέλος κινήθηκε προς το ένα ή το άλλο άκρο της ακολουθίας, αλλά αν στην προηγούμενη γειτονιά που βρισκόταν είχε ταξινομηθεί με αύξουσα ή φθίνουσα διάταξη. Επιπλέον, αν επιλεγεί, η εφαρμογή του παράγοντα ορμής γίνεται σε όλα τα μέλη της γειτονιάς και όχι μόνο στα όριά της. Διαφορετικές παραλλαγές του αλγορίθμου Emerge-Sort μπορούν να προκύψουν αν η ανάθεση παραγόντων ορμής επιλεγεί να γίνεται ή όχι στις ταξινομήμένες ακολουθίες, στις αταξινόμητες και στα συνοριακά τους μέλη. Αν μία γειτονιά δεν είναι ταξινομημένη και όλα τα μέλη της έχουν αδιάφορο παράγοντα, επιλέγεται ο συντομότερος δρόμος, η κατεύθυνση δηλαδή που θα απαιτήσει τις λιγότερες κινήσεις. Αν τα άτομα φέρουν παράγοντες ορμής που δε συμφωνούν μεταξύ τους, επικρατεί η κατεύθυνση που δηλώνεται από τους περισσότερους. Στην περίπτωση που οι ανταγωνιστικοί παράγοντες ισοβαθμήσουν επικρατεί η κατεύθυνση των μελών που έχουν κάνει μεγαλύτερη διαδρομή μέσα στην ακολουθία. Τέλος, αν και με αυτό τον τρόπο προκύψει ισοβαθμία, τότε δεν επιλέγεται καμία κατεύθυνση και ο αλγόριθμος προχωρά σε επόμενη γειτονιά. Στις επόμενες δύο εικόνες παρουσιάζεται ένα παράδειγμα εφαρμογής των νέων παραγόντων ορμής σε δύο συνεχόμενους γύρους εκτέλεσης του αλγορίθμου.



Εικόνα 2.4: Ανάθεση παραγόντων ορμής A/D/X μετά την ταξινόμηση γειτονιάς (1/2) (Πηγή [03]).



Εικόνα 2.5: Ανάθεση παραγόντων ορμής A/D/X μετά την ταξινόμηση γειτονιάς (2/2) (Πηγή [03]).

Όσον αφορά τις ενέργειες στα άκρα της ακολουθίας σε περιπτώσεις εντοπισμού εγκλωβισμών αναπτύχθηκαν δύο νέες παραλλαγές, πέραν της αρχικής μεθόδου που ονομάζεται Wrapping και προϋποθέτει τη μετακίνηση όλων των ατόμων της γειτονιάς κατά μία θέση αριστερά (ή αντίστοιχα δεξιά) ώστε να απελευθερωθούν. Η πρώτη εναλλακτική μέθοδος (Inflect) περιλαμβάνει την ανάθεση συγκεκριμένων παραγόντων ορμής στο τελευταίο και προτελευταίο μέλος της ακολουθίας. Η δεύτερη εναλλακτική μέθοδος (Exclude Ends) δεν επιτρέπει στην πρώτη και τελευταία θέση μιας ακολουθίας να επιλεγούν ως «κέντρο» οπότε τα ακραία μέλη κινούνται μόνο στα πλαίσια της γειτονιάς τους και επικοινωνούν μόνο με το μέλος στα αριστερά ή δεξιά τους αντίστοιχα.

Πρόσθετες παραλλαγές δημιουργήθηκαν αλλάζοντας τον τρόπο με τον οποίο επιλέγεται το κέντρο της επόμενης γειτονιάς που θα μελετηθεί από τον αλγόριθμο. Η πρώτη διαθέσιμη επιλογή ορίζει αν το κέντρο της γειτονιάς θα επιλεγεί με σειριακό ή τυχαίο τρόπο. Η επιλογή κέντρου μπορεί να ενισχυθεί με δύο επιπλέον μεθόδους. Η πρώτη ορίζει ότι στο τέλος κάθε γύρου τα μέλη που βρίσκονταν στο κέντρο της γειτονιάς που εξετάστηκε εισέρχονται σε κατάσταση αδράνειας και σταματούν να ανταποκρίνονται σε μηνύματα. Για τη διατήρηση αυτής της πληροφορίας χρησιμοποιείται ένας πίνακας ο οποίος περιλαμβάνει τις υποψήφιες προς επιλογή θέσεις της ακολουθίας. Η μέθοδος αυτή ονομάζεται “Use Probability Table”. Η δεύτερη μέθοδος έχει ως στόχο να δώσει προτεραιότητα σε γειτονιές που θεωρητικά έχουν μεγαλύτερη πιθανότητα να συμβάλουν στην εξέλιξη της ταξινόμησης, επιλέγοντας άμεσα ως κέντρο της

επόμενης γειτονιάς ένα από τα δύο άκρα της γειτονιάς που ταξινομήθηκε την τελευταία φορά. Αν προκύψει κίνηση με αύξουσα ταξινόμηση, ως επόμενο κέντρο επιλέγεται το δεξί όριο της προηγούμενης γειτονιάς ενώ αν προκύψει κίνηση με φθίνουσα ταξινόμηση επιλέγεται το αριστερό όριο της προηγούμενης γειτονιάς. Η μέθοδος αυτή ονομάζεται "Last Sorted Bound". Οι 2 επιπλέον μέθοδοι μπορούν να συνδυαστούν μεταξύ τους και είτε με σειριακή ή τυχαία επιλογή κέντρου.

Τέλος, όσον αφορά τον τρόπο και τον τόπο εφαρμογής των παραγόντων ορμής μπορούν να χρησιμοποιηθούν αρκετές εκδοχές. Ο τρόπος εφαρμογής μπορεί να είναι ένας από τους παρακάτω:

- **Καμία Ενέργεια (No Action)** όπου δεν αλλάζουν οι παράγοντες ορμής των μελών της ακολουθίας.
- **Εφαρμογή (Set Properties)** όπου τα μέλη προσλαμβάνουν νέους παράγοντες ορμής ανεξάρτητα από τους προηγούμενους παράγοντες ορμής τους.
- **Απάλειψη (Reset Properties)** όπου στα μέλη ανατίθενται αδιάφορος παράγοντας ορμής.
- **Κλιμακωτή Εφαρμογή (Scale Properties)** όπου, αν ένα μέλος έχει αδιάφορο παράγοντα ορμής τότε προσλαμβάνει νέο, αλλιώς, αν ο παράγοντας ορμής που έχει είναι αντίθετος του νέου, τότε ο παλαιότερος απαλείφεται και ανατίθεται ο αδιάφορος παράγοντας.

Κάθε ένας από τους παραπάνω τρόπους μπορεί να εφαρμοστεί με διαφορετικό τρόπο στις παρακάτω ομάδες συνόλων:

- **Ταξινομημένη γειτονιά (Sorted Set)** που αφορά τις γειτονιές που τη στιγμή που τις επισκέπτεται ο αλγόριθμος είναι ήδη ταξινομημένες.
- **Σύνορα ταξινομημένης γειτονιάς (Sorted Set Borders)** που αφορά τα άτομα που συνορεύουν με την προς τρέχουσα γειτονιά.
- **Σύνορα γειτονιάς που μόλις ταξινομήθηκε (New Set Borders)** που αφορά τα συνοριακά άτομα γειτονιάς η οποία στην έναρξη του τρέχοντος γύρου δεν ήταν ταξινομημένη, αλλά ταξινομήθηκε κατά τη διάρκειά του.

Στα πλαίσια της προαναφερθείσας εργασίας[03], για την αξιολόγηση των παραλλαγών του αλγορίθμου Emerge-Sort, αναπτύχθηκε η εφαρμογή EmergeSortUI (EmergeSort User Interface – Διεπαφή Χρήστη αλγορίθμου Emerge-Sort). Υλοποιήθηκαν, με τη γλώσσα προγραμματισμού Java, μία σειρά από κλάσεις με στόχο να διευκολύνουν την εφαρμογή των

διάφορων παραλλαγών αλλά και των διαφορετικών πρωτοκόλλων. Οι κλάσεις ομαδοποιήθηκαν και σχημάτισαν βιβλιοθήκη τύπου jar (Java Archive). Το γραφικό περιβάλλον της εφαρμογής σχεδιάστηκε στο περιβάλλον Visual Studio .NET 2008 με χρήση της γλώσσας C#. Η παρεχόμενη λειτουργικότητα από τη διεπαφή του χρήστη παρέμειναν πλήρως διαχωρισμένες. Το γραφικό περιβάλλον δημιουργεί αντικείμενα των κλάσεων της βιβλιοθήκης και καλεί μεθόδους τους. Η εφαρμογή παρέχει την ευκαιρία στους χρήστες να διεξάγουν πειράματα και να κατανοήσουν τις παραλλαγές του αλγορίθμου και να εξοικειωθούν με τον τρόπο λειτουργίας του.

Όσον αφορά τον τρόπο χρήσης της εφαρμογής, δόθηκε μεγάλο βάρος στην ελευθερία επιλογών. Από το κεντρικό παράθυρο της εφαρμογής ο χρήστης μπορεί να ορίσει τις παραμέτρους προσομοίωσης (το πρωτόκολλο που θα χρησιμοποιηθεί, τον τρόπο επιλογής της γειτονιάς και του κέντρου της, τις ενέργειες στα άκρα της ακολουθίας και τις παραλλαγές στην εφαρμογή των παραγόντων ορμής) καθώς επίσης το μέγεθος της δοκιμαστικής ακολουθίας και του άνω ορίου εκτέλεσης. Η εφαρμογή μπορεί να καταγράψει λεπτομερώς τα βήματα εκτέλεσης και την κατάσταση των ακολουθιών και των συντελεστών των ατόμων σε κάθε βήμα αν ο χρήστης ενεργοποιήσει τη σχετική επιλογή (Enable Logging). Μόλις η ταξινόμηση τελειώσει, η εφαρμογή παρουσιάζει τα στατιστικά αποτελέσματα σε πίνακα, ώστε να είναι δυνατό να χρησιμοποιηθούν για σύγκριση. Επιπλέον παρουσιάζονται οι ακολουθίες που παράχθηκαν και αν έχει ενεργοποιηθεί η επιλογή Enable Logging, παρουσιάζονται αναλυτικά και τα βήματα του αλγορίθμου καθώς και η τελική κατάσταση των μελών.

Εκτός από την εξοικείωση με τις παραλλαγές του αλγορίθμου Emerge-Sort, η εφαρμογή παρέχει έναν γενετικό αλγόριθμο αξιολόγησης των διαφόρων παραλλαγών, για την αναπαράσταση των οποίων χρησιμοποιήθηκαν άτομα τα οποία αποτελούνται από έξι γονίδια, ένα για το βασικό πρωτόκολλο και πέντε για τις βασικές παραμέτρους του κάθε πρωτοκόλλου. Ο γενετικός αλγόριθμος εκτελεί όλους τους δυνατούς συνδυασμούς παραμέτρων των διαθέσιμων παραλλαγών και θεωρήθηκε αναμενόμενο ότι συνδυασμοί που δεν οδηγούν σε ταξινόμηση θα αντιπροσωπεύονται από άτομα που πολύ γρήγορα θα εξαλειφθούν από τους πληθυσμούς του γενετικού αλγορίθμου ενώ αντίθετα θα αναδειχθούν οι πιο αποδοτικοί από αυτούς με αντικειμενικό τρόπο. Για την εκτέλεση του γενετικού αλγόριθμου χρησιμοποιήθηκε μία βιβλιοθήκη που αναπτύχθηκε στο Evolutionary Computation Laboratory του George Mason University με την ονομασία ECJ[16].

Η βιβλιοθήκη ECJ είναι στην ουσία μία πλατφόρμα εφαρμογής γενετικών αλγορίθμων με πολλαπλές δυνατότητες (π.χ. ταυτόχρονη εκτέλεση διαφορετικών σεναρίων), πολύ καλή

απόδοση και υψηλή προσαρμοστικότητα. Η βιβλιοθήκη είναι διαθέσιμη σε μορφή jar αρχείου και ομάδες Java κλάσεων που υλοποιούν συγκεκριμένες λειτουργίες όπως διάφορες τεχνικές δημιουργίας, επιλογής και εξέλιξης πληθυσμών. Στα πλαίσια της ανάπτυξης της εφαρμογής EmergeSortUI, η βιβλιοθήκη ECJ επεκτάθηκε ώστε να μπορεί να κληθεί ως αυτόνομη διεργασία μέσα από το περιβάλλον της εφαρμογής. Για το σκοπό αυτό, αναπτύχθηκαν αρχεία παραμέτρων ώστε να μοντελοποιηθούν οι παραλλαγές του αλγορίθμου Emerge-Sort και αναπτύχθηκαν ορισμένες επιπλέον Java κλάσεις για την ορθή έναρξη της διεργασίας και εισόδου των παραμέτρων. Προέκυψε με αυτό τον τρόπο η επεκταμένη βιβλιοθήκη με το όνομα MyECJ.

2.2 Τεχνολογίες και Εργαλεία

Ο κυριότερος στόχος της παρούσας μεταπτυχιακής διατριβής ήταν η μετατροπή της λειτουργικότητας που αναπτύχθηκε για την αυτόνομη εφαρμογή EmergeSortUI, ώστε να γίνει διαθέσιμη μέσα από μία εφαρμογή Ιστού. Για τη μετατροπή αυτή χρησιμοποιήθηκαν αρκετά εργαλεία και τεχνολογίες, τα σημαντικότερα από τα οποία παρουσιάζονται στις επόμενες παραγράφους.

2.2.1 Η Γλώσσα HTML

Η HTML (Hyper-Text Markup Language) είναι μία περιγραφική γλώσσα (markup language), δηλαδή ένας ειδικός τρόπος γραφής κειμένου που περιλαμβάνει πληροφορίες μορφοποίησης με τη μορφή ετικετών (tags). Οι εφαρμογές περιήγησης Ιστού (Web Browsers) αναγνωρίζουν αυτό τον τρόπο γραφής και μορφοποιούν το σχετικό κείμενο ανάλογα με τις ενσωματωμένες πληροφορίες μορφοποίησης. Οι ετικέτες που χρησιμοποιούνται στην HTML ώστε να δώσουν τις απαραίτητες οδηγίες στον web browser συνήθως ορίζουν την αρχή ή το τέλος μιας λειτουργίας και βρίσκονται πάντα μεταξύ των συμβόλων < και >, (π.χ. <BODY>).

Έχουν θεσπιστεί προδιαγραφές ώστε οι ιστοσελίδες να προβάλλονται με τον ίδιο τρόπο σε όλες τις εφαρμογές περιήγησης Ιστού. Καθώς όμως οι εφαρμογές αυτές έγιναν πιο περίπλοκες ενσωματώνοντας αρκετές επιπλέον λειτουργίες, εμφανίζονται περιπτώσεις ασυμβατότητας στην εμφάνιση ιστοσελίδων. Για την παρούσα διατριβή επιλέχθηκε απλή σχεδίαση των HTML σελίδων της Εφαρμογής Ιστού και των παραγόμενων αποτελεσμάτων. Χρησιμοποιήθηκαν, εκτός από τις βασικές ετικέτες μορφοποίησης της σελίδας, ετικέτες για τη δημιουργία φορμών, την αποστολή δεδομένων στον εξυπηρετητή και τη δημιουργία πινάκων.

Πληροφορίες σχετικά με τη σύνταξη και τις επιλογές των διαθέσιμων ετικετών της HTML βρέθηκαν στην ιστοσελίδα W3Schools [19].

2.2.2 Η Τεχνολογία των Java Servlets

Τα Java Servlets [20] είναι μία τεχνολογία βασισμένη στη γλώσσα προγραμματισμού Java, με την οποία είναι δυνατό να παραχθεί δυναμικό διαδικτυακό περιεχόμενο. Ένα Java Servlet είναι στην ουσία ένα αντικείμενο μίας κλάσης της Java η οποία επεκτείνει την κλάση HttpServlet. Για να εκτελεστεί ο κώδικας της κλάσης πρέπει να περιληφθεί σε έναν ειδικό εξυπηρετητή που μπορεί να φιλοξενήσει Servlets, έναν Servlet Container. Στα πλαίσια αυτής της εργασίας αναπτύχθηκαν Java Servlets, τα οποία παίρνουν τις ρυθμίσεις εκτέλεσης μίας προσομοίωσης του αλγορίθμου Emerge-Sort και καλούν κατάλληλα τις βιβλιοθήκες (jar files), οι οποίες ενσωματώνουν την επιθυμητή λειτουργικότητα. Αφού η προσομοίωση ολοκληρωθεί, τα Servlets επιστρέφουν ένα HTML αρχείο με τα αποτελέσματα.

Η τεχνολογία των Java Servlets είναι στην ουσία εξέλιξη της τεχνολογίας CGI (Common Gateway Interface "Κοινό Περιβάλλον Διεπαφής") η οποία ξεκίνησε ως μέθοδος επέκτασης της λειτουργικότητας των εξυπηρετητών Ιστού (Web Servers) με την δυνατότητα εκτέλεσης εξειδικευμένων προγραμμάτων. Το μοντέλο αυτό προέβλεπε τη μεταφορά των δεδομένων από το χρήστη συνήθως μέσω μίας φόρμας HTML, την εκτέλεση ενός προγράμματος στο server και την επιστροφή των αποτελεσμάτων στο χρήστη μέσω HTML. Το μοντέλο CGI, αν και ευρέως διαδεδομένο, δημιουργούσε σε κάποιες περιπτώσεις προβλήματα ασφάλειας ή διαχείρισης πόρων. Η τεχνολογία των Java Servlets επέκτεινε αυτό το μοντέλο παρέχοντας μηχανισμούς ευκολότερου χειρισμού των δεδομένων εισόδου και εξόδου, βελτιώνοντας τη διαχείριση πόρων και συμβάλλοντας σε πιο αυστηρές προϋποθέσεις εκτέλεσης για βελτίωση της ασφάλειας.

Για την εργασία χρησιμοποιήθηκαν Java Servlets συμβατά με το νεότερο API, το Java Servlet 3.0. Περισσότερες πληροφορίες για τη χρήση των Java Servlets μπορούν να αναζητηθούν σε σχετική ισοσελίδα της εταιρείας Sun [15].

2.2.3 Ο Εξυπηρετητής Tomcat

Όπως αναφέρθηκε, για να εκτελεστεί ο κώδικας ενός Java Servlet πρέπει να εγκατασταθεί σε έναν Servlet Container, έναν εξυπηρετητή ο οποίος αναλαμβάνει να διαχειριστεί

τις αιτήσεις που φτάνουν σε αυτό ή κάποιο άλλο Servlet. Για την εργασία αυτή χρησιμοποιήθηκε ο Apache Tomcat[21] έκδοσης 7. Ο λόγος που επιλέχθηκε είναι γιατί είναι ευρέως διαδεδομένος και υπάρχει διαθεσιμότητα στο κέντρο δικτύων του πανεπιστημίου. Εξαιτίας του γεγονότος ότι είναι πολύ διαδεδομένος, υπάρχει αρκετή πληροφορία σχετικά με τη ρύθμισή του αλλά και πολλά εργαλεία τα οποία αυτοματοποιούν μεγάλο μέρος της.

2.2.4 Το Περιβάλλον Ανάπτυξης Eclipse

Το eclipse[22] είναι ένα περιβάλλον ανάπτυξης λογισμικού (Integrated Development Environment - IDE), το οποίο χρησιμοποιείται για την ανάπτυξη εφαρμογών στη γλώσσα προγραμματισμού Java (και άλλες γλώσσες). Έχουν αναπτυχθεί αρκετές επεκτάσεις (plug-ins) για το eclipse οι οποίες παρέχουν επιπλέον δυνατότητες. Μία από αυτές παρέχει «συνεργασία» του eclipse με τον εξυπηρετητή tomcat ώστε να δημιουργηθεί ένα ενοποιημένο περιβάλλον στο οποίο γράφεται ο κώδικας, μεταφράζεται, εγκαθίσταται (γίνεται deploy) στον servlet container και δοκιμάζεται. Με τη χρήση του eclipse αρκετές διαδικασίες γίνονται αυτόματα με αποτέλεσμα να περιορίζονται τα λάθη και να μειώνεται ο χρόνος των δοκιμών. Επιπλέον, αν ο κώδικας γίνει περίπλοκος, το περιβάλλον μπορεί να παράσχει δυνατότητες αποσφαλμάτωσης (debugging) ώστε να βρεθούν ευκολότερα τα λάθη. Αξίζει να τονιστεί ότι η έκδοση του eclipse που χρησιμοποιείται (eclipse for JEE Developers) περιέχει όλες τις απαραίτητες επεκτάσεις (plug-ins) αλλά και τις βιβλιοθήκες (Software Development Kit) που χρειάζονται για την ανάπτυξη Java Servlets και την εγκατάσταση στον Tomcat (Tomcat deployment).

2.2.5 Η Τεχνολογία των Java Applets

Τα Java Applets [17] είναι μικρο-εφαρμογές γραμμένες συνήθως στη γλώσσα προγραμματισμού Java οι οποίες είναι σχεδιασμένες για να εκτελούνται από μία εφαρμογή περιήγησης Ιστού (Web Browser). Συνήθως ενσωματώνονται σε μία HTML σελίδα (όπως ενσωματώνεται παραδείγματος χάριν μία εικόνα) και μεταφορτώνονται αυτόματα στην εφαρμογή περιήγησης Ιστού που ζήτησε τη σελίδα παρέχοντας τη λειτουργικότητα που υλοποιούν. Εμφανίστηκαν με την 1η έκδοση της Java το 1995 και γενικά επιτρέπεται να είναι γραμμένα σε οποιαδήποτε γλώσσα μπορεί να μεταφραστεί σε κώδικα μηχανής Java (Java Byte-code) αλλά συνήθως είναι γραμμένα σε Java.

Λόγω της φορητότητας του κώδικα μηχανής Java, τα Java Applets μπορούν να εκτελεστούν σε περιηγητές που εκτελούνται σε οποιαδήποτε πλατφόρμα αρκεί σε αυτή να έχει εγκατασταθεί μία συμβατή έκδοση της εικονικής μηχανής Java (JVM- Java Virtual Machine). Επίσης είναι πολύ εύκολο ένα Java Applet να μετατραπεί σε αυτόνομη εφαρμογή Java.

Τα Java Applets, αν και θεωρούνται αρκετά ώριμη τεχνολογία, χρησιμοποιούνται ακόμα και σήμερα για εφαρμογές που απαιτούν πολλούς υπολογισμούς ή περίπλοκα γραφικά, καθώς οι εφαρμογές περιήγησης Ιστού μόλις πρόσφατα ενσωμάτωσαν τέτοιες δυνατότητες. Τα Java Applets κατά κανόνα περιλαμβάνουν γραφικό περιβάλλον και αντιδρούν στις επιλογές του χρήστη με αλλαγές στο γραφικό περιβάλλον ή εκκίνηση υπολογιστικών διαδικασιών. Συχνά λοιπόν χρησιμοποιούνται για εφαρμογές επίδειξης ή για εκπαιδευτικές εφαρμογές. Για λόγους ασφάλειας η εκτέλεση των Java Applets περιορίζεται σε ένα κλειστό περιβάλλον εκτέλεσης (sandbox) από τις εφαρμογές περιήγησης Ιστού. Με αυτό τον τρόπο έχουν περιορισμένη πρόσβαση σε πόρους του συστήματος στο οποίο εκτελούνται και μόνο μετά από σχετική έγκριση του χρήστη μπορούν να λάβουν επιπλέον δικαιώματα εκτέλεσης.

2.2.6 Το WindowBuilder Plug-in για το Eclipse

Για την ευκολότερη σχεδίαση του γραφικού περιβάλλοντος των Applets επιλέχθηκε να χρησιμοποιηθεί το eclipse plug-in WindowBuilder [23]. Το plug-in αυτό επιτρέπει τη σχεδίαση ενός γραφικού περιβάλλοντος (GUI – Graphical User Interface) για εφαρμογές Java ή Java Applets και την αυτόματη παραγωγή κώδικα Java. Στα πλαίσια της εργασίας, τα περισσότερα μέρη του γραφικού περιβάλλοντος των Applets δημιουργήθηκαν και τοποθετήθηκαν με τη χρήση της προβολής σχεδίασης (Design View) του plug-in. Στη συνέχεια επιλέχθηκε η προβολή κώδικα (Source View) του plug-in και ο κώδικας που παρήγαγε το plug-in τροποποιήθηκε κατάλληλα για τις ανάγκες της εφαρμογής.

2.2.7 Εγκατάσταση και Παραμετροποίηση του Περιβάλλοντος Ανάπτυξης

Για την ευκολότερη ανάπτυξη και τον πιο αποδοτικό έλεγχο της εφαρμογής Ιστού που αναπτύχθηκε οι προαναφερθείσες τεχνολογίες συνδυάστηκαν ώστε να δημιουργήσουν ένα ενοποιημένο περιβάλλον ανάπτυξης και δοκιμών. Χρησιμοποιήθηκε ο Apache Tomcat έκδοσης 7 και το eclipse για ανάπτυξη JEE (Java Enterprise Edition) εφαρμογών. Ρυθμίστηκαν κατάλληλα ώστε τα Java Servlets που αναπτύσσονται στο eclipse να μπορούν να δοκιμαστούν στον τοπικό

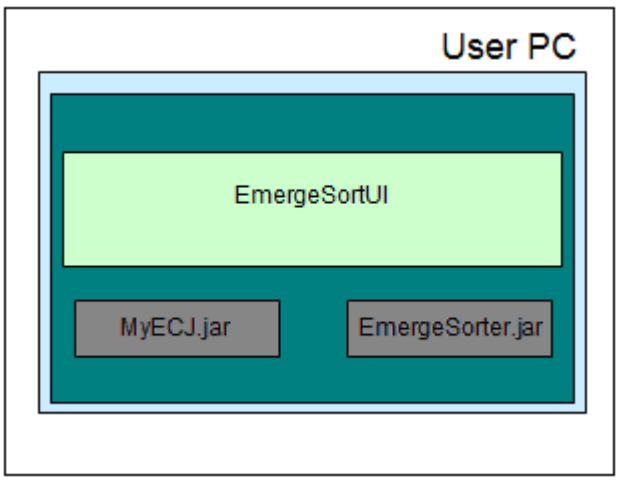
Apache Tomcat. Επιπλέον, για την ευκολότερη ανάπτυξη του γραφικού περιβάλλοντος των Java Applets, το plug-in WindowBuilder ενσωματώθηκε στο eclipse. Η διαδικασία παραμετροποίησης του περιβάλλοντος βασίστηκε σε έναν διαδικτυακό οδηγό [24] και περιγράφεται αναλυτικά στο παράρτημα Α.

Κεφάλαιο 3
Ανάπτυξη Εφαρμογής Ιστού με Εκτέλεση Υπολογισμών στον Εξυπηρετητή

Όπως αναφέρθηκε και στην εισαγωγή, το 1ο στάδιο της παρούσας μεταπτυχιακής διατριβής περιελάμβανε την υλοποίηση εφαρμογής Ιστού η οποία μπορεί να δεχτεί παραμέτρους προσομοίωσης παραλλαγών του αλγορίθμου Emerge-Sort και να εκτελέσει τους απαραίτητους υπολογισμούς μέσω ενός Java Servlet που βρίσκεται σε εξυπηρετητή Ιστού. Ο κυριότερος λόγος ανάπτυξης της συγκεκριμένης εφαρμογής Ιστού είναι το γεγονός ότι μπορεί να εκτελεστεί χωρίς να απαιτεί εγκατάσταση στο σύστημα κάθε χρήστη. Επιπλέον αν η λειτουργικότητά της επεκταθεί, απαιτείται να αναβαθμιστεί μόνο κεντρικά ώστε να γίνει διαθέσιμη με τη νέα της μορφή σε όλους τους χρήστες. Χρησιμοποιώντας ώριμες τεχνολογίες Ιστού όπως η HTML και τη γλώσσα προγραμματισμού Java, η εφαρμογή Ιστού μπορεί να εκτελεστεί στα περισσότερα συστήματα αρκεί να διαθέτουν κάποια εφαρμογή περιήγησης Ιστού (Web Browser), αίροντας περιορισμούς εκτέλεσης λόγω πόρων ή είδους πλατφόρμας. Η εφαρμογή Ιστού δρα ουσιαστικά ως νέα διεπαφή που χρησιμοποιεί βιβλιοθήκες που έχουν ήδη αναπτυχθεί και απλά αντικαθιστά το γραφικό περιβάλλον της αυτόνομης εφαρμογής EmergeSortUI. Έτσι επετεύχθη

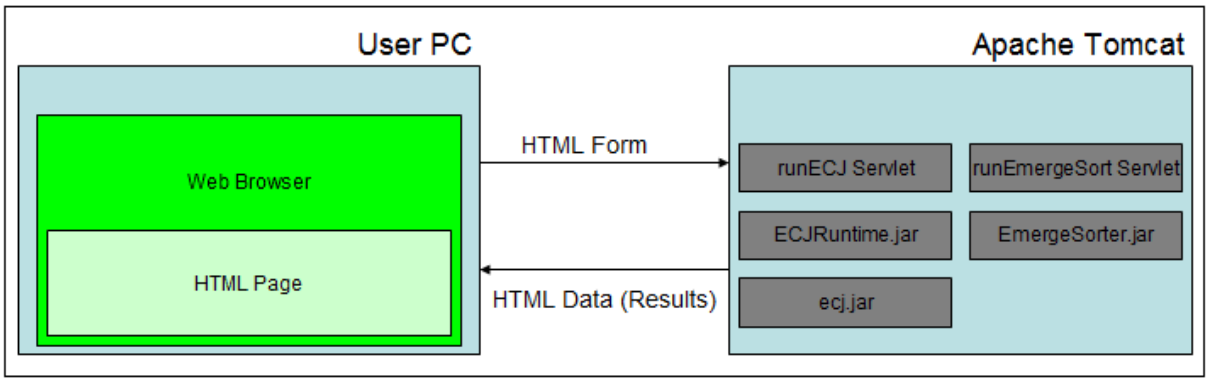
επαναχρησιμοποίηση του κώδικα διατηρώντας χαμηλά το κόστος υλοποίησης σε ανθρωποώρες.

Στην αρχική σελίδα της εφαρμογής ο χρήστης μπορεί να συμπληρώσει τις παραμέτρους της προσομοίωσης του Emerge-Sort και όταν επιλέξει «Run Emerge Sort» εκτελείται ένα Java Servlet, το οποίο διαβάζει τις παραμέτρους που εισήγαγε ο χρήστης στη φόρμα της HTML σελίδας, εκτυπώνει τις τιμές των μεταβλητών, και στη συνέχεια, αφού εκτελέσει την προσομοίωση τυπώνει τα Statistics και το Log της προσομοίωσης. Επιπλέον, από διαφορετική σελίδα της εφαρμογής έχει τη δυνατότητα να μεταφορτώσει ένα αρχείο παραμέτρων γενετικού αλγορίθμου και να εκτελέσει τον γενετικό αλγόριθμο αξιολόγησης. Σε αυτή την περίπτωση εκτελείται ένα άλλο Java Servlet, το οποίο διαβάζει τις παραμέτρους του αρχείου και ξεκινά την εκτέλεση του γενετικού αλγορίθμου. Τα αποτελέσματα αποθηκεύονται σε αρχείο το όνομα του οποίου έχει ορίσει ο χρήστης και μπορεί να μεταμορφωθεί μετά από έναν εύλογο χρόνο εκτέλεσης. Εκτός από τις προαναφερθείσες σελίδες υπάρχει μία σελίδα η οποία παρέχει γενικές πληροφορίες για τον αλγόριθμο καθώς επίσης και μία σελίδα που λειτουργεί ως βοήθεια για τη συμπλήρωση των παραμέτρων. Υπάρχει ακόμα μία σελίδα από την οποία μπορούν να ανασυρθούν παλαιότερα αποτελέσματα εκτέλεσης του γενετικού αλγορίθμου. Η λειτουργικότητα που παρέχει η εφαρμογή Ιστού έχει «κληρονομηθεί» από την εφαρμογή EmergeSortUI. Πιο συγκεκριμένα, η λειτουργικότητα του Emerge-Sort και της εκτέλεσης γενετικών αλγορίθμων ήταν ήδη διαθέσιμη με τη μορφή αρχείων jar (Java ARchive). Η λειτουργικότητα του γενετικού αλγορίθμου περιλαμβάνεται στο jar MyECJ.jar ενώ η λειτουργικότητα του αλγορίθμου EmergeSort στο jar EmergeSorter.jar. Στην εφαρμογή EmergeSortUI τα jar αρχεία χρησιμοποιούνται από το γραφικό περιβάλλον της εφαρμογής ώστε κατάλληλα αντικείμενα να δημιουργηθούν και να εκτελεστούν οι μέθοδοί τους.



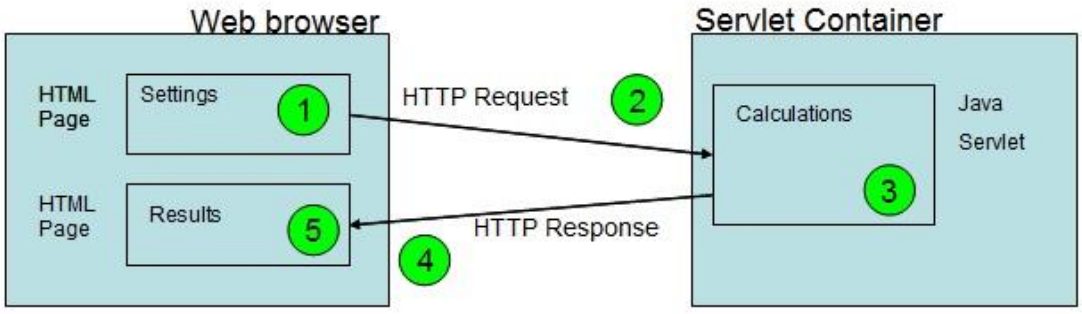
Εικόνα 3.1: Το μοντέλο εκτέλεσης της αυτόνομης εφαρμογής EmergeSortUI.

Στην εφαρμογή Ιστού που αναπτύχθηκε στα πλαίσια της παρούσας διατριβής, τα Java Servlets που δημιουργήθηκαν ενσωματώνουν αυτά τα jar αρχεία ώστε, αφού δεχτούν από το χρήστη τις σχετικές εντολές, δημιουργούν αντικείμενα αυτών των βιβλιοθηκών και καλούν μεθόδους τους. Τα jars «ζουν» και εκτελούνται στο περιβάλλον του εξυπηρετητή. Στο περιβάλλον του χρήστη μεταφέρεται απλά μία HTML σελίδα με τα αποτελέσματα. Η λειτουργικότητα του γενετικού αλγορίθμου περιλαμβάνεται πλέον στα jar ECJRuntime.jar και ecj.jar ενώ η λειτουργικότητα του αλγορίθμου EmergeSort στο jar EmergeSorter.jar.



Εικόνα 3.2: Το μοντέλο εκτέλεσης των Java Servlet.

Στο μοντέλο εκτέλεσης των Java Servlets, η ροή των δεδομένων ακολουθεί την πορεία που φαίνεται στην επόμενη εικόνα: Οι παράμετροι εισάγονται από το χρήστη μέσω μίας HTML φόρμας (1). Τα δεδομένα της φόρμας μεταφέρονται (2) μέσω ενός HTTP request στο Java Servlet το οποίο αναλαμβάνει να εκτελέσει (3) τους απαιτούμενους υπολογισμούς χρησιμοποιώντας την προγραμματιστική λογική και τις βιβλιοθήκες που έχουν ήδη υλοποιηθεί για την αυτόνομη εφαρμογή. Αφού οι υπολογισμοί εκτελεστούν το Servlet στέλνει ως απάντηση (4) ένα νέο HTML αρχείο το οποίο περιλαμβάνει τα αποτελέσματα. Τέλος η HTML σελίδα με τα αποτελέσματα απεικονίζεται από τον περιηγητή Ιστού του χρήστη (5).

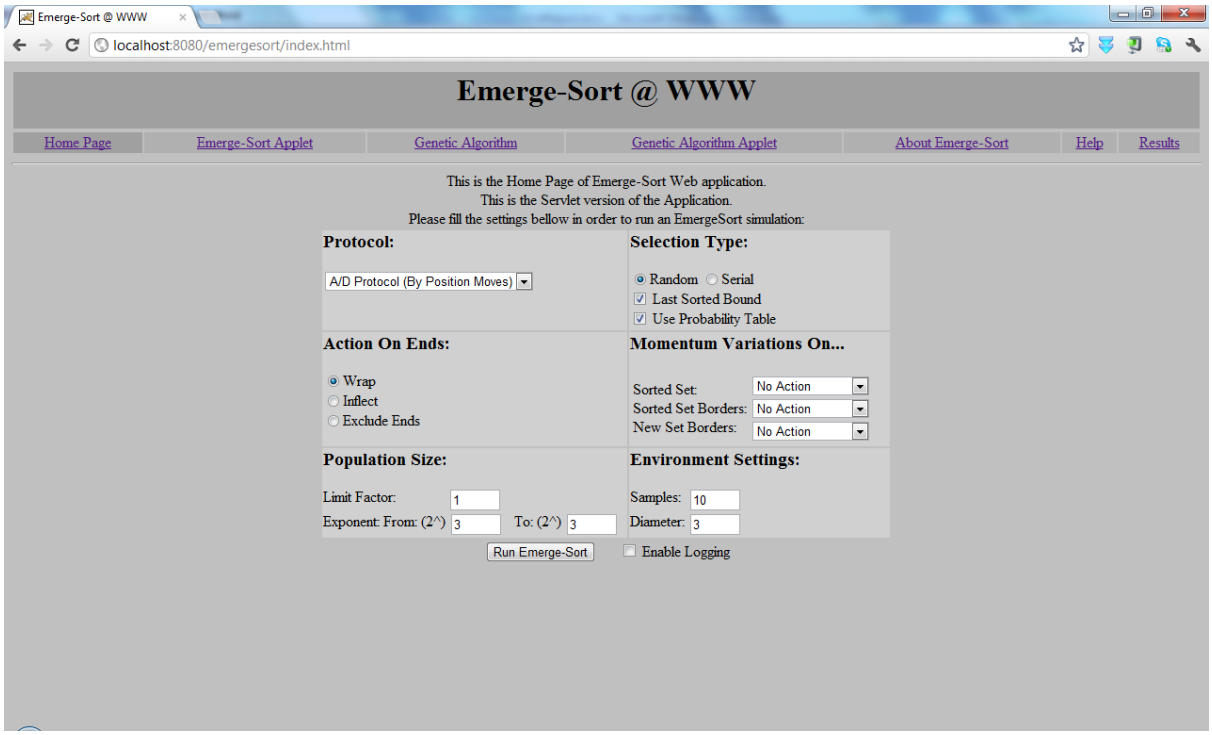


Εικόνα 3.3: Η ροή δεδομένων στην Εφαρμογή Ιστού με εκτέλεση υπολογισμών στον εξυπηρετητή.

Στο κεφάλαιο αυτό περιγράφεται η λειτουργικότητα που προσφέρει η Εφαρμογή Ιστού μέσω Java Servlets και η μέθοδος υλοποίησής της.

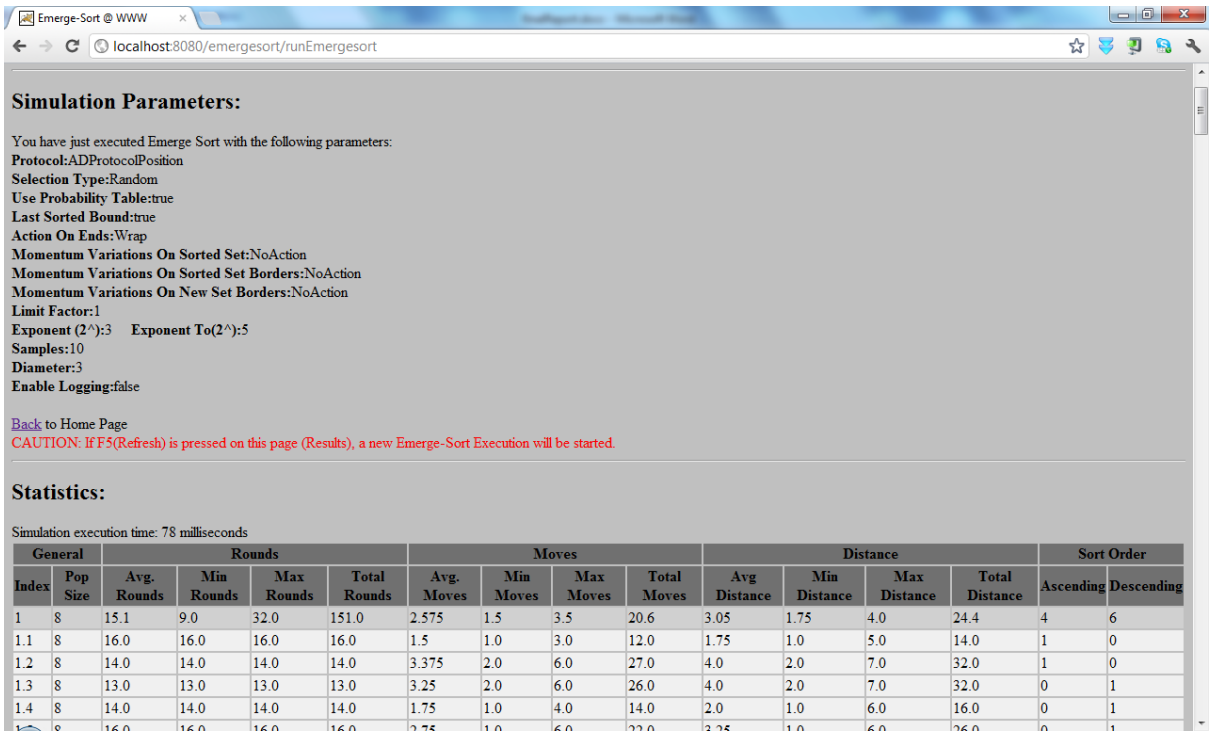
3.1 Λειτουργικότητα

Στην αρχική οθόνη της εφαρμογής ο χρήστης μπορεί να συμπληρώσει τις παραμέτρους της προσομοίωσης διαφόρων παραλλαγών του Emerge-Sort.



Εικόνα 3.4: Η αρχική σελίδα της Εφαρμογής Ιστού.

Αν ο χρήστης συμπληρώσει τις παραμέτρους του Emerge-Sort και επιλέξει Run Emerge-Sort, εκτελείται το Servlet runEmergesort το οποίο διαβάζει τις παραμέτρους που εισήγαγε ο χρήστης στη φόρμα της HTML σελίδας, εκτυπώνει τις τιμές των μεταβλητών, και στη συνέχεια, αφού εκτελέσει την προσομοίωση τυπώνει τα Statistics και το Log της προσομοίωσης.



Emerge-Sort @ WWW

localhost:8080/emergesort/runEmergesort

Simulation Parameters:

You have just executed Emerge Sort with the following parameters:
Protocol:ADProtocolPosition
Selection Type: Random
Use Probability Table:true
Last Sorted Bound:true
Action On Ends: Wrap
Momentum Variations On Sorted Set: NoAction
Momentum Variations On Sorted Set Borders: NoAction
Momentum Variations On New Set Borders: NoAction
Limit Factor:1
Exponent (2^):3 **Exponent To(2^):**5
Samples:10
Diameter:3
Enable Logging:false

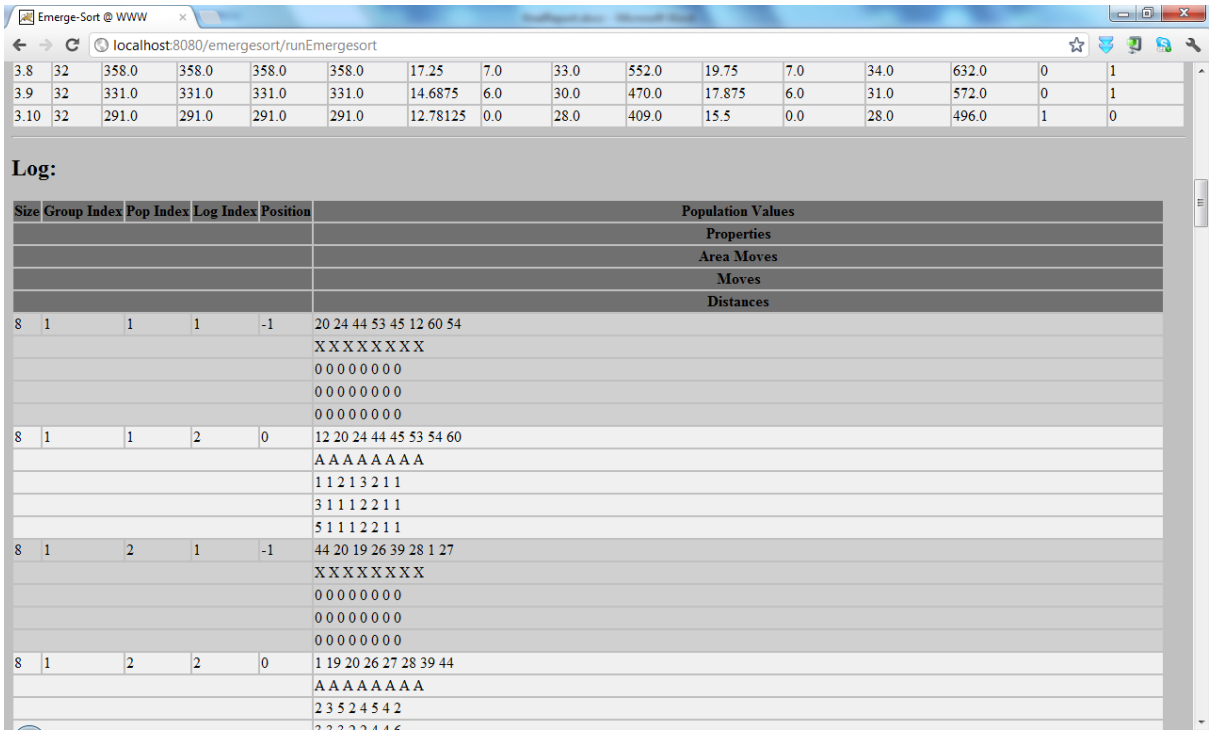
Back to Home Page
CAUTION: If F5(Refresh) is pressed on this page (Results), a new Emerge-Sort Execution will be started.

Statistics:

Simulation execution time: 78 milliseconds

General		Rounds				Moves				Distance				Sort Order	
Index	Pop Size	Avg. Rounds	Min Rounds	Max Rounds	Total Rounds	Avg. Moves	Min Moves	Max Moves	Total Moves	Avg Distance	Min Distance	Max Distance	Total Distance	Ascending	Descending
1	8	15.1	9.0	32.0	151.0	2.575	1.5	3.5	20.6	3.05	1.75	4.0	24.4	4	6
1.1	8	16.0	16.0	16.0	16.0	1.5	1.0	3.0	12.0	1.75	1.0	5.0	14.0	1	0
1.2	8	14.0	14.0	14.0	14.0	3.375	2.0	6.0	27.0	4.0	2.0	7.0	32.0	1	0
1.3	8	13.0	13.0	13.0	13.0	3.25	2.0	6.0	26.0	4.0	2.0	7.0	32.0	0	1
1.4	8	14.0	14.0	14.0	14.0	1.75	1.0	4.0	14.0	2.0	1.0	6.0	16.0	0	1

Εικόνα 3.5: Τα αποτελέσματα της προσομοίωσης (1/2).



Emerge-Sort @ WWW

localhost:8080/emergesort/runEmergesort

3.8	32	358.0	358.0	358.0	358.0	17.25	7.0	33.0	552.0	19.75	7.0	34.0	632.0	0	1
3.9	32	331.0	331.0	331.0	331.0	14.6875	6.0	30.0	470.0	17.875	6.0	31.0	572.0	0	1
3.10	32	291.0	291.0	291.0	291.0	12.78125	0.0	28.0	409.0	15.5	0.0	28.0	496.0	1	0

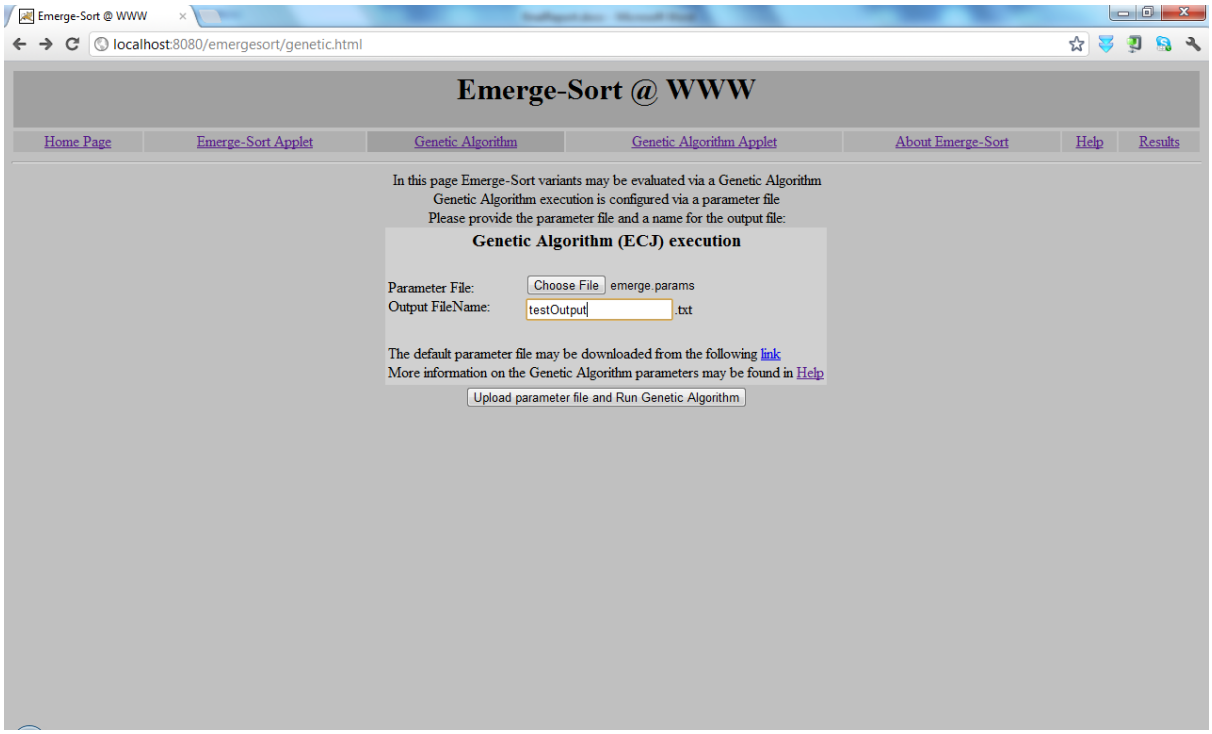
Log:

Size	Group Index	Pop Index	Log Index	Position	Population Values
					Properties
					Area Moves
					Moves
					Distances
8	1	1	1	-1	20 24 44 53 45 12 60 54
					X X X X X X X X
					0 0 0 0 0 0 0 0
					0 0 0 0 0 0 0 0
					0 0 0 0 0 0 0 0
8	1	1	2	0	12 20 24 44 45 53 54 60
					A A A A A A A A
					1 1 2 1 3 2 1 1
					3 1 1 1 2 2 1 1
					5 1 1 1 2 2 1 1
8	1	2	1	-1	44 20 19 26 39 28 1 27
					X X X X X X X X
					0 0 0 0 0 0 0 0
					0 0 0 0 0 0 0 0
					0 0 0 0 0 0 0 0
8	1	2	2	0	1 19 20 26 27 28 39 44
					A A A A A A A A
					2 3 5 2 4 5 4 2

Εικόνα 3.6: Τα αποτελέσματα της προσομοίωσης (2/2).

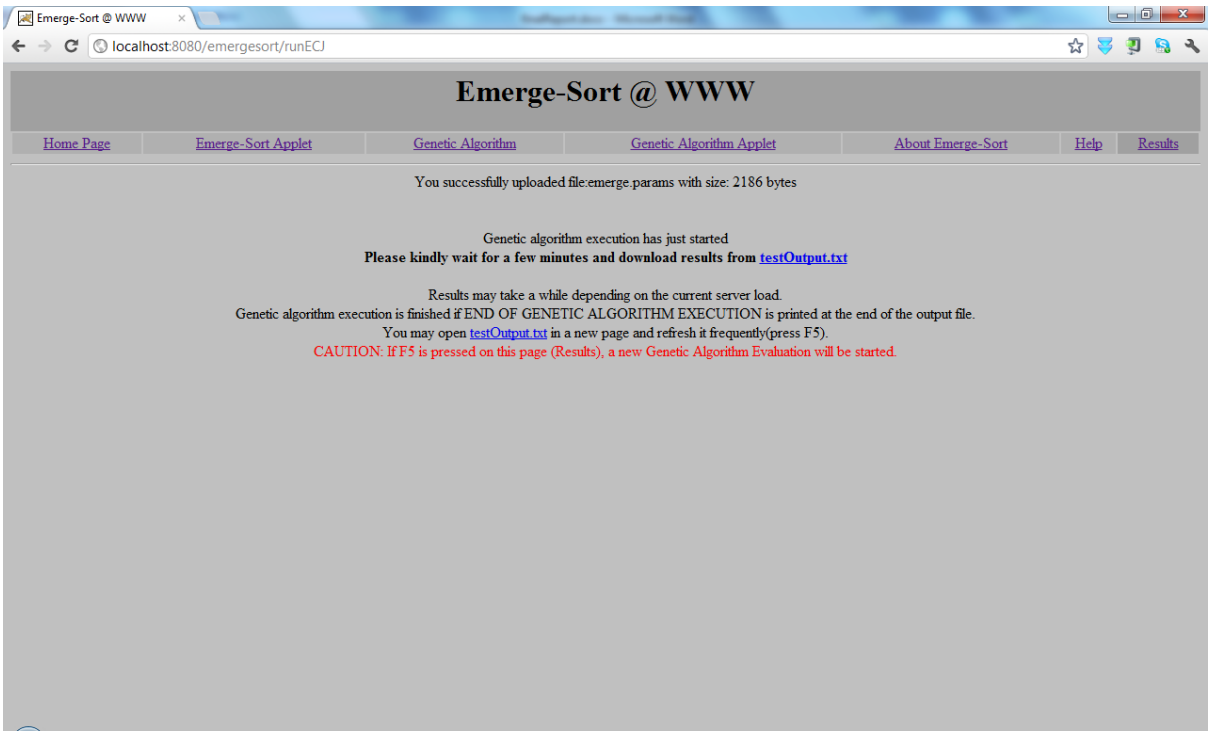
Αν ο χρήστης επιλέξει τη σελίδα Genetic Algorithm μπορεί να εκτελέσει το γενετικό αλγόριθμο. Για να εκτελεστεί ο γενετικός αλγόριθμος απαιτείται ένα αρχείο παραμέτρων (με την κατάληξη .params). Ο χρήστης μπορεί να μεταφορτώσει το σχετικό αρχείο από σύνδεσμο που παρέχεται στη σελίδα και να το τροποποιήσει ανάλογα με το πείραμα γενετικού αλγορίθμου που

επιθυμεί να εκτελέσει. Στη συνέχεια, μέσω της HTML φόρμας επιλέγει τον κατάλογο όπου έχει αποθηκεύσει το αρχείο παραμέτρων. Ακολούθως επιλέγει το όνομα του αρχείου των αποτελεσμάτων που θα δημιουργήσει το Servlet. Επιλέγοντας "Upload Parameter File and Run Genetic Algorithm" εκτελείται το Servlet runECJ το οποίο εκκινεί το γενετικό αλγόριθμο με είσοδο το αρχείο που μεταφόρτωσε ο χρήστης. Το servlet επιστρέφει το σύνδεσμο από όπου ο χρήστης μπορεί να μεταφορτώσει το αρχείο με τα αποτελέσματα (αρχείο εξόδου).

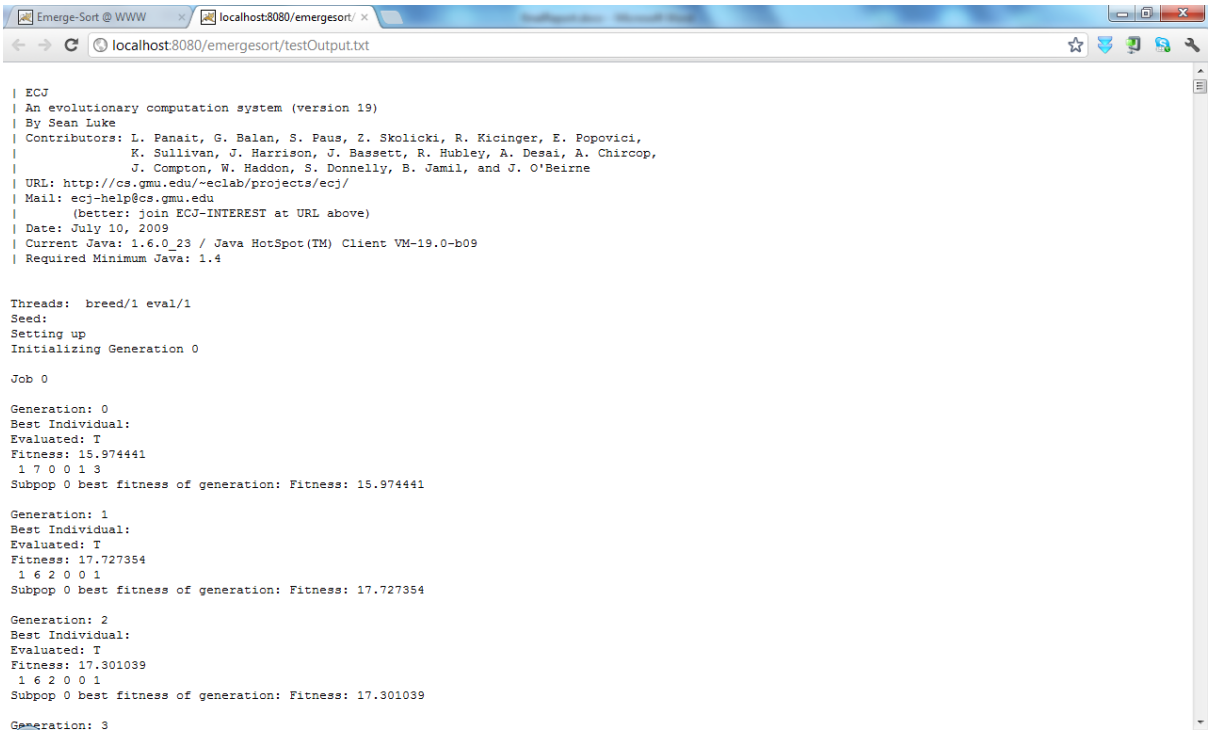


Εικόνα 3.7: Η σελίδα εκτέλεσης γενετικού αλγορίθμου (Genetic Algorithm).

Αν ο χρήστης δεν εισάγει αρχείο παραμέτρων ή όνομα αρχείου εξόδου χρησιμοποιούνται προκαθορισμένες (default) τιμές. Ειδική μέριμνα έχει ληφθεί αν ο χρήστης εισάγει όνομα αρχείου εξόδου που ήδη υπάρχει και για τη διαγραφή παλαιότερων αρχείων εξόδου για αποφυγή υπερφόρτωσης του αποθηκευτικού χώρου το εξυπηρετητή.

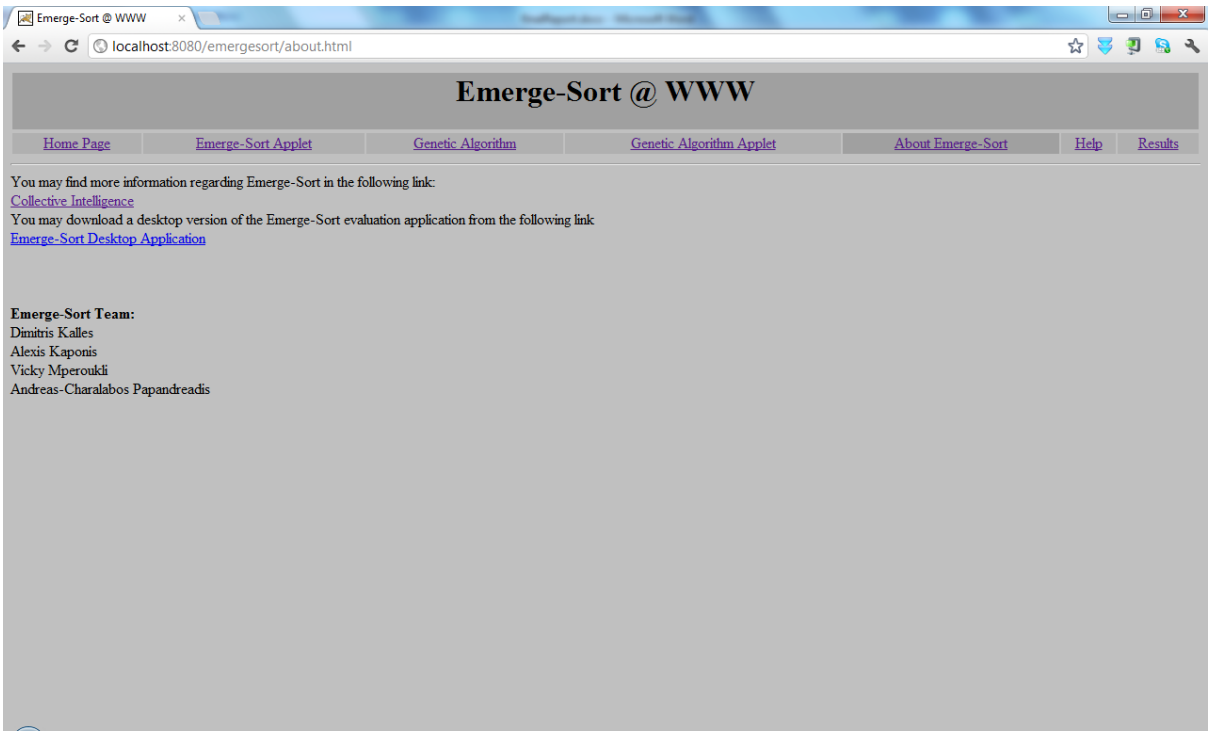


Εικόνα 3.8: Η σελίδα αποτελεσμάτων γενετικού αλγορίθμου (Results).

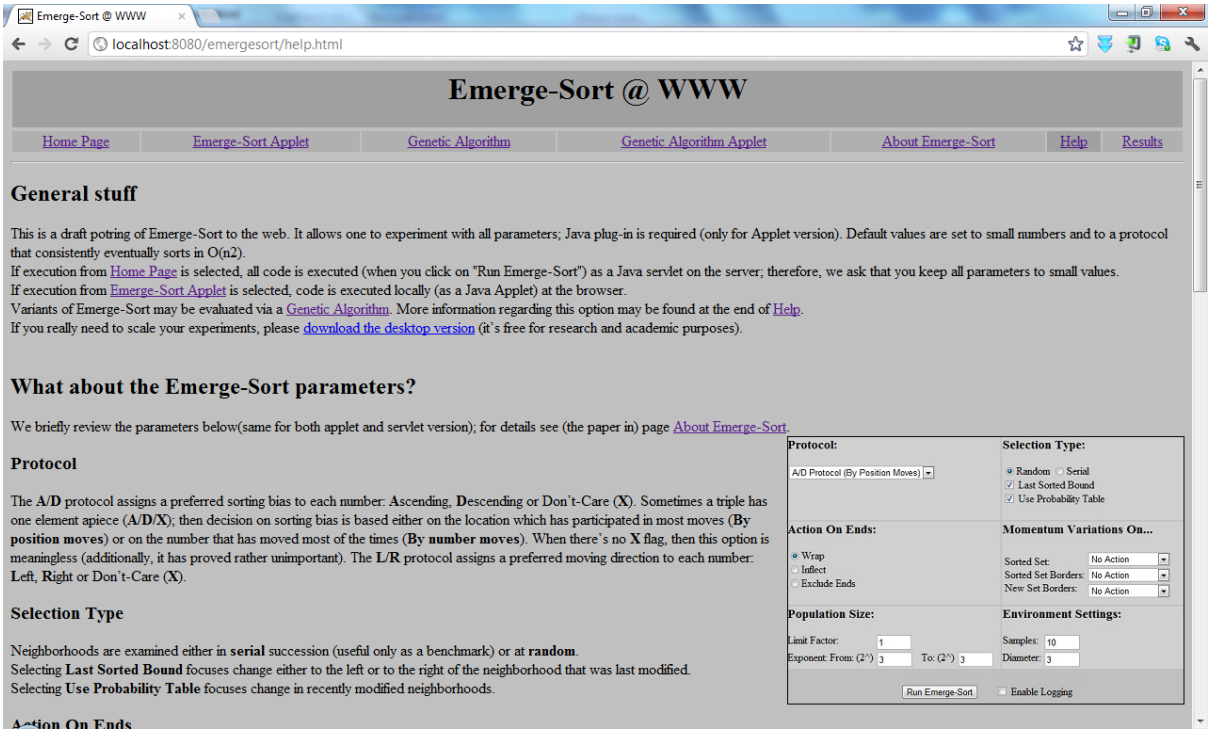


Εικόνα 3.9: Ενδεικτικό αρχείο εξόδου γενετικού αλγορίθμου.

Εκτός από τις σελίδες υποβολής παραμέτρων προσομοίωσης και τη σελίδα των αποτελεσμάτων, υπάρχει μία σελίδα η οποία παρέχει γενικές πληροφορίες για τον αλγόριθμο και τη σχετική έρευνα (About Emerge-Sort) καθώς επίσης και μία σελίδα που λειτουργεί ως βοήθεια για τη συμπλήρωση των παραμέτρων (Help).



Εικόνα 3.10: Η σελίδα που παρέχει γενικές πληροφορίες για τον αλγόριθμο.



Εικόνα 3.11: Η σελίδα που παρέχει βοήθεια για τη συμπλήρωση των παραμέτρων.

Ο χρήστης μπορεί να πλοηγηθεί μεταξύ των σελίδων της εφαρμογής μέσω ενός απλού οριζόντιου μενού (Οι σελίδες σχετικές με Applets παρουσιάζονται στο επόμενο κεφάλαιο).

Εικόνα 3.12: Το μενού της εφαρμογής.

3.2 Υλοποίηση

Τα αρχεία που περικλείουν το μεγαλύτερο μέρος της υλοποίησης του 1ου παραδοτέου είναι η αρχική σελίδα (**WebContent/index.html**), η σελίδα με τις πληροφορίες για τον αλγόριθμο (**WebContent/about.html**), η σελίδα βοήθειας (**WebContent/help.html**) και τα servlet που χειρίζεται την εκτέλεση της προσομοίωσης του Emerge-Sort (**src\emergesort\runEmergesort.java**) και την εκτέλεση του γενετικού αλγορίθμου (**src\emergesort\runECJ.java**). Τα υπόλοιπα αρχεία της εφαρμογής και η δομή των φακέλων δημιουργήθηκε αυτόματα από το eclipse.

3.2.1 Η Αρχική Σελίδα (index.html)

Η αρχική σελίδα της εφαρμογής έχει στην ουσία αντικαταστήσει το GUI της αυτόνομης εφαρμογής. Έχουν οριστεί 2 φόρμες (tag form) οι οποίες επιτρέπουν την εισαγωγή των ρυθμίσεων της εφαρμογής με μορφή παραμέτρων. Κάθε παράμετρος (tag input ή select) έχει ένα όνομα και πιθανές τιμές. Π.χ. η παράμετρος Protocol έχει όνομα Protocol και πιθανές τιμές LRProtocol, ADProtocolPosition, ADProtocolNumber και ορίζεται ως εξής:

```
<select name="Protocol">
<option value="LRProtocol" >L/R Protocol</option>
<option value="ADProtocolPosition" selected="selected">A/D Protocol (By
Position Moves)</option>
<option value="ADProtocolNumber">A/D Protocol (By Number Moves)</option>
</select> .
```

Κάποιες παράμετροι δέχονται απευθείας τιμές, όπως η Limit Factor η οποία έχει όνομα LimitFactor, default τιμή 1 και ορίζεται ως εξής:

```
Limit Factor:
<input type="text" name="LimitFactor" value="1"><BR>
```

Τέλος κάποιες παράμετροι είναι checkboxes, όπως η Last Sorted Bound η οποία ορίζεται ως εξής:

```
<input type="checkbox" name="SelectionTypeExtra" value="LastSortedBound" />
Last Sorted Bound<br/>
```

Στη φόρμα ορίζεται το servlet το οποίο θα αναλάβει να χειριστεί το παραγόμενο request και ο τύπος του request (GET ή POST).

```
<form action="/emergesort/runEmergesort"
      method="POST">
```

Επίσης ορίζεται το Submit Button, το κουμπί που ξεκινά την αποστολή των δεδομένων στον server.

```
    <input type="submit" value="Run Emerge-Sort">
```

Στην αρχική σελίδα, αλλά και στις υπόλοιπες σελίδες στην αρχή του σώματος (body) ορίζεται ένας πίνακας ο οποίος ορίζει ένα απλό οριζόντιο μενού πλοήγησης. Σε κάθε σελίδα αλλάζει το κελί του πίνακα που είναι highlighted.

```
<table align="center" bgcolor="#C0C0C0">
<tr >
<td width="20%" bgcolor="#A0A0A0">
<center>
<a href="./index.html">Home Page</a>
</center>
</td>
<td width="25%" bgcolor="#B0B0B0">
<center>
<a href="./about.html">About Emerge-Sort</a>
</center>
</td>
<td width="20%" bgcolor="#B0B0B0">
<center>
<a href="./help.html">Help</a>
</center>
</td>
</tr>
</table>
```

3.2.2 Η Σελίδα Εκτέλεσης του Γενετικού Αλγορίθμου (genetic.html)

Η σελίδα αυτή είναι πιο απλή από την προηγούμενη καθώς περιλαμβάνει μία απλούστερη φόρμα με ένα πεδίο τύπου file για το αρχείο παραμέτρων που επιθυμεί να ανεβάσει ο χρήστης και ένα πεδίο τύπου text για το όνομα του αρχείου εξόδου. Όταν επιλεγεί το κουμπί της φόρμας εκτελείται το servlet runECJ.

3.2.3 Η Σελίδα με Πληροφορίες για τον Emerge-Sort (about.html)

Η σελίδα αυτή είναι ακόμα πιο απλή από την προηγούμενη καθώς περιλαμβάνει μόνο το μενού πλοήγησης, 2 συνδέσμους σχετικά με τον Emerge-Sort και τη σύσταση της ομάδας.

3.2.4 Η Σελίδα Βοήθειας για τη Συμπλήρωση των Παραμέτρων (help.html)

Κρίθηκε σκόπιμο να προστεθεί αυτή η σελίδα που περιλαμβάνει στην ουσία έναν οδηγό χρήσης της εφαρμογής περιγράφοντας τη σημασία των παραμέτρων που χρησιμοποιούνται και των τιμών που μπορούν να δεχτούν.

3.2.5 Το Servlet για την Εκτέλεση του Emerge-Sort (runEmergesort.java)

Το servlet αυτό αναλαμβάνει να επεξεργαστεί το POST request που προκαλεί η χρήση της πρώτης φόρμας (με κουμπί Run Emerge Sort). Για την εκτέλεση της ταξινόμησης χρησιμοποιείται το jar EmergeSort.jar. Για να συμπεριληφθεί στο Project εισήχθη ως βιβλιοθήκη από το menu Project->Properties->Java Build Path->Add JARs του eclipse.

Η μέθοδος doGet, όπως και ο constructor του servlet, δε χρησιμοποιείται στην ουσία. Η doGet καλεί την doPost μόνο για λόγους debugging.

Η μέθοδος doPost αναλαμβάνει να εκτελέσει τον αλγόριθμο Emerge-Sort και να επιστρέψει τα αποτελέσματα.

Καλεί τις κατάλληλες μεθόδους ώστε:

- να προετοιμάσει την επιστρεφόμενη HTML Σελίδα.
- να συγκεντρώσει τις παραμέτρους του request και να τις αποθηκεύσει σε ξεχωριστές μεταβλητές.
- να μετατρέψει τις μεταβλητές αυτές σε κατάλληλους τύπους και να περάσει τις παραμέτρους σε ένα αντικείμενο τύπου Sorter (από τη βιβλιοθήκη Emergesorter.jar).
- να εκκινήσει την ταξινόμηση.
- να επιστρέψει στην HTML σελίδα τα αποτελέσματα της ταξινόμησης.
- να «κλείσει» την HTML σελίδα.

Ακολουθούν αναλυτικά οι μέθοδοι που χρησιμοποιούνται για τις παραπάνω διαδικασίες (και βοηθητικές μέθοδοι):

printPageStart: Χρησιμοποιείται για να εκκινήσει την HTML σελίδα. Σε αυτή τη συνάρτηση παράγεται HTML κώδικας για την υλοποίηση του μενού πλοήγησης.

printPageEnd: Χρησιμοποιείται για να κλείσει την HTML σελίδα.

getSimulationParameters: Χρησιμοποιείται για να πάρει τις παραμέτρους από το request (με τη χρήση της request.getParameter) και να τις αποθηκεύσει σε ξεχωριστές μεταβλητές τύπου string.

printSimulationParameters: Χρησιμοποιείται για να τυπώσει στην HTML σελίδα τις τιμές των παραμέτρων.

runSimulation: Χρησιμοποιείται για να αναθέσει τις παραμέτρους σε ένα αντικείμενο της κλάσης sorter και να εκτελέσει την προσομοίωση (sorter.Sort();). Περιλαμβάνει λειτουργικότητα αντίστοιχη της μεθόδου Run() από τη standalone εφαρμογή.

printSimulationStatistics: Χρησιμοποιείται για να τυπώσει τα simulation Stats. Περιλαμβάνει λειτουργικότητα αντίστοιχη της μεθόδου FillStatsGrid (Sorter sorter) από τη standalone εφαρμογή.

createStatisticsTable: Χρησιμοποιείται από την προηγούμενη συνάρτηση για να αρχικοποιήσει τον σχετικό πίνακα και να εισάγει επικεφαλίδες. Περιλαμβάνει λειτουργικότητα αντίστοιχη της μεθόδου CreateTable από τη standalone εφαρμογή.

printSimulationLog: Χρησιμοποιείται για να τυπώσει το simulation Log. Περιλαμβάνει λειτουργικότητα αντίστοιχη της μεθόδου FillLogGrid (Sorter sorter) από τη standalone εφαρμογή.

createLogTable: Χρησιμοποιείται από την προηγούμενη συνάρτηση για να αρχικοποιήσει τον σχετικό πίνακα και να εισάγει επικεφαλίδες. Περιλαμβάνει λειτουργικότητα αντίστοιχη της μεθόδου createLogTable από τη standalone εφαρμογή.

Οι συναρτήσεις **getEmergeTypeFromString, getAtomSelectionTypeFromString, getWrapTypeFromString, getSortedSetActionFromString, getSortedSetBorders-ActionFromString, getUnSortedSetBordersActionFromString** μετατρέπουν τις αντίστοιχες String ή Boolean μεταβλητές σε τύπους της κλάσης Enm που μπορεί να χειριστεί ο Sorter.

Περιλαμβάνουν λειτουργικότητα αντίστοιχη των μεθόδων get_____Type από τη standalone εφαρμογή.

3.2.6 Το Servlet για την Εκτέλεση του Γενετικού Αλγορίθμου (runECJ.java)

Το servlet αυτό αναλαμβάνει να επεξεργαστεί το POST request που προκαλεί η χρήση της φόρμας εκτέλεσης του γενετικού αλγορίθμου. Για την εκτέλεση του γενετικού αλγορίθμου χρησιμοποιούνται τα jar αρχεία ECJRuntime.jar, ecj.jar, EmergeSorter.jar. Για να συμπεριληφθούν στο Project εισήχθησαν ως βιβλιοθήκες από το menu Project->Properties->Java Build Path->Add JARs του eclipse. Το ecj.jar είναι μία βιβλιοθήκη για την εκτέλεση σεναρίων γενετικών αλγορίθμων, το ECJRuntime.jar είναι ένα project που αναπτύχθηκε στα πλαίσια προηγούμενης εργασίας και καλεί τη βιβλιοθήκη ecj.jar ενώ το EmergeSorter.jar είναι η βιβλιοθήκη που αναπτύχθηκε στα πλαίσια προηγούμενης εργασίας και περιλαμβάνει τη λειτουργικότητα των παραλλαγών του Emerge-Sort. Για τις ανάγκες της παρούσας εργασίας χρειάστηκε να γίνουν κάποιες αλλαγές στον κώδικα του ECJRuntime ώστε η έξοδος του γενετικού αλγορίθμου να περάσει από τη standard έξοδο σε ένα σαφώς ορισμένο αρχείο εξόδου.

Η μέθοδος doGet, όπως και ο constructor του servlet, δε χρησιμοποιείται στην ουσία. Η doGet καλεί την doPost μόνο για λόγους debugging.

Η μέθοδος doPost αναλαμβάνει να εκκινήσει την εκτέλεση του γενετικού αλγορίθμου και να επιστρέψει τα αποτελέσματα.

Καλεί τις κατάλληλες μεθόδους ώστε:

- να προετοιμάσει την επιστρεφόμενη HTML σελίδα.
- να διαβάσει και να αποθηκεύσει το αρχείο που «ανέβασε» ο χρήστης.
- να εκτελέσει τις κατάλληλες ενέργειες αν ο χρήστης δεν όρισε αρχείο παραμέτρων ή αρχείο εξόδου.
- να εκκινήσει την εκτέλεση του γενετικού αλγορίθμου με αρχείο εισόδου το αρχείο που «ανέβασε» ο χρήστης.
- να επιστρέψει στην HTML σελίδα το σύνδεσμο από όπου μπορεί να ληφθεί το αρχείο εξόδου.
- να «κλείσει» την HTML σελίδα.

Ακολουθούν αναλυτικά οι μέθοδοι που χρησιμοποιούνται για τις παραπάνω διαδικασίες (και βοηθητικές μέθοδοι):

printPageStart: Χρησιμοποιείται για να εκκινήσει την HTML σελίδα. Σε αυτή τη συνάρτηση παράγεται HTML κώδικας για την υλοποίηση του μενού πλοήγησης.

printPageEnd: Χρησιμοποιείται για να κλείσει την HTML σελίδα.

runGeneticAlgorithm: Χρησιμοποιείται για να διαβάσει και να αποθηκεύσει το αρχείο που «ανέβασε» ο χρήστης, να εκτελέσει τις κατάλληλες ενέργειες αν ο χρήστης δεν όρισε αρχείο παραμέτρων ή αρχείο εξόδου και να χειριστεί περιπτώσεις λάθους, να εκκινήσει την εκτέλεση του γενετικού αλγορίθμου με αρχείο εισόδου το αρχείο που «ανέβασε» ο χρήστης και αν όλα πανε καλά να επιστρέψει στην HTML σελίδα το σύνδεσμο από όπου μπορεί να ληφθεί το αρχείο εξόδου.

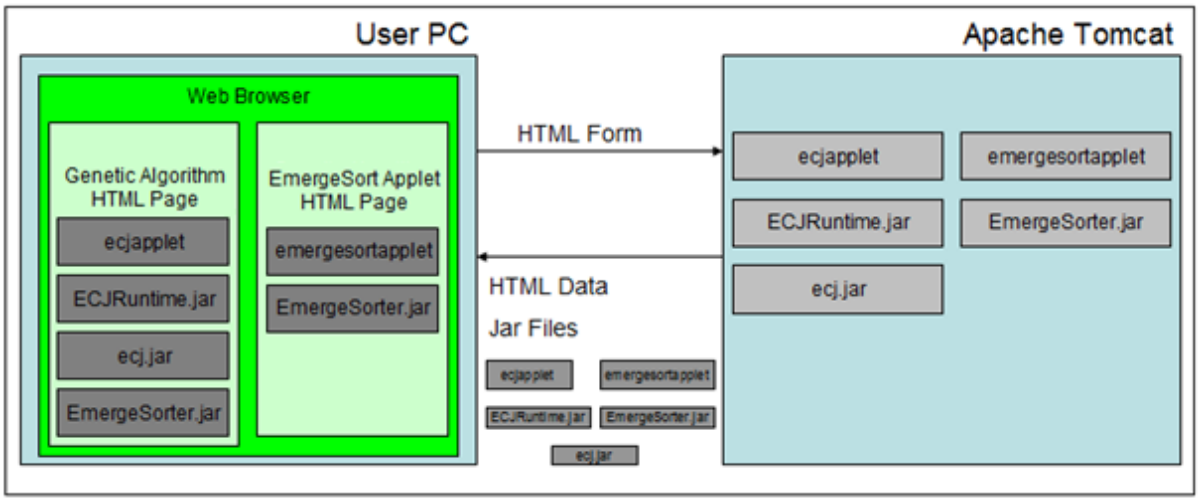
Κεφάλαιο 4
Επέκταση Εφαρμογής Ιστού για Εκτέλεση Υπολογισμών στο Περιβάλλον Χρήστη

Αφού αναπτύχθηκε το κομμάτι της Εφαρμογής Ιστού για εκτέλεση των υπολογισμών στο περιβάλλον του εξυπηρετητή, χρησιμοποιήθηκε η τεχνολογία των Java Applets ώστε η υπολογιστική λογική του αλγορίθμου Emerge-Sort και του γενετικού αλγορίθμου αξιολόγησης να είναι διαθέσιμη στους χρήστες μέσω του Ιστού με το βάρος, όμως, των υπολογισμών να μπορεί να μετακυλιστεί στον υπολογιστή του εκάστοτε χρήστη. Με τη χρήση των Java Applets παρέχεται η δυνατότητα τοπικής εκτέλεσης της εφαρμογής χωρίς επιβάρυνση του εξυπηρετητή διατηρώντας πλεονεκτήματα όπως η μη απαίτηση εγκατάστασης και αναβάθμισης και η δια-λειτουργικότητα σε πολλαπλές διαφορετικές πλατφόρμες.

Η Εφαρμογή Ιστού επεκτάθηκε ώστε να περιλαμβάνει νέες HTML σελίδες οι οποίες ενσωματώνουν Java Applets (η μία για εκτέλεση παραλλαγών του Emerge-Sort και η άλλη για εκτέλεση του γενετικού αλγορίθμου) που αναλαμβάνουν τόσο την εκτέλεση των υπολογισμών της προσομοίωσης όσο και την εμφάνιση των αποτελεσμάτων στην εφαρμογή περιήγησης Ιστού του χρήστη. Σε αυτή την περίπτωση ο κώδικας των Java Applets (και των απαραίτητων

βιβλιοθηκών) μεταφορτώνεται στην εφαρμογή περιήγησης Ιστού που χρησιμοποιεί ο χρήστης και εκτελείται εκεί και όχι στον εξυπηρετητή Ιστού. Οι παράμετροι που εισάγονταν μέσω της HTML φόρμας της Εφαρμογής Ιστού μπορούν πλέον να εισάγονται και στο γραφικό περιβάλλον (GUI) του Applet και τα αποτελέσματα θα εμφανίζονται σε πίνακες του γραφικού περιβάλλοντος. Από τη στιγμή που ο χρήστης επιλέξει τη σχετική HTML σελίδα και ο κώδικας του Applet μεταφορτωθεί δεν απαιτείται καμία επικοινωνία με τον server για την εκτέλεση των υπολογισμών.

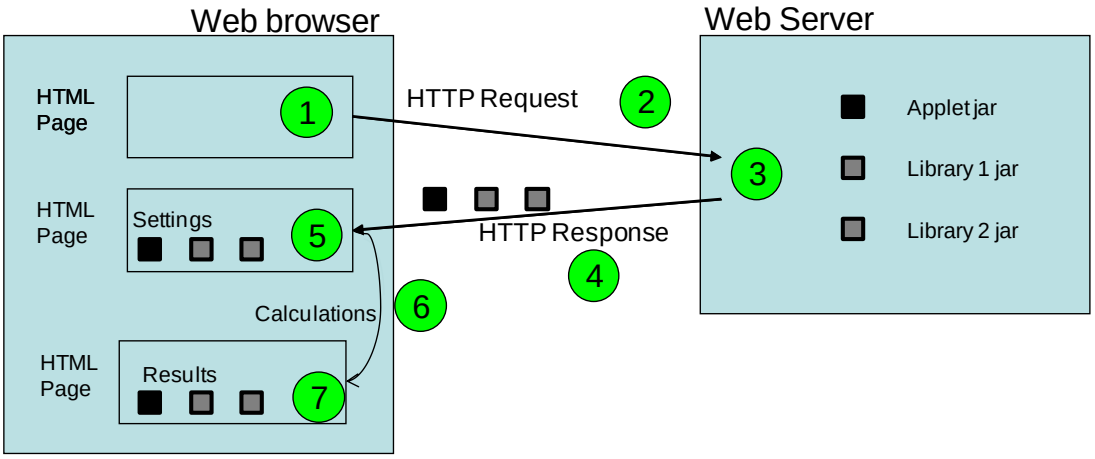
Σε αυτό το στάδιο τα jar αρχεία (βιβλιοθήκες) που ήταν διαθέσιμα από προηγούμενη εργασία, ενσωματώθηκαν και σε Java Applets. Το γραφικό περιβάλλον των Java Applets αφού δεχτεί από το χρήστη τις σχετικές εντολές, δημιουργεί αντικείμενα αυτών των βιβλιοθηκών και καλεί μεθόδους τους. Τα jars των Applets με το γραφικό περιβάλλον εκτέλεσης του γενετικού αλγορίθμου (ecjapplet.jar) και του αλγορίθμου EmergeSort (emergesortapplet.jar) μεταφορτώνονται και εκτελούνται στο περιβάλλον του χρήστη όντας ενσωματωμένα σε αντίστοιχες HTML σελίδες. Τα jars ECJRuntime.jar, ecj.jar και EmergeSorter.jar μεταφορτώνονται επίσης στο περιβάλλον του χρήστη για να παράσχουν την απαιτούμενη λειτουργικότητα.



Εικόνα 4.1: Το μοντέλο εκτέλεσης των Java Applets.

Στο μοντέλο εκτέλεσης των Java Applets η ροή των δεδομένων ακολουθεί την πορεία που φαίνεται στην επόμενη εικόνα : Ο χρήστης επιλέγει να επισκεφτεί την HTML σελίδα που ενσωματώνει το Java Applet (1). Μεταφέρεται στον εξυπηρετητή Ιστού το αίτημα (HTTP request) για τη σελίδα (2). Ο εξυπηρετητής Ιστού «αντιλαμβάνεται» ότι η ζητούμενη σελίδα περιέχει ενσωμάτωση Java Applet οπότε βρίσκει το Java Applet και τις απαραίτητες βιβλιοθήκες

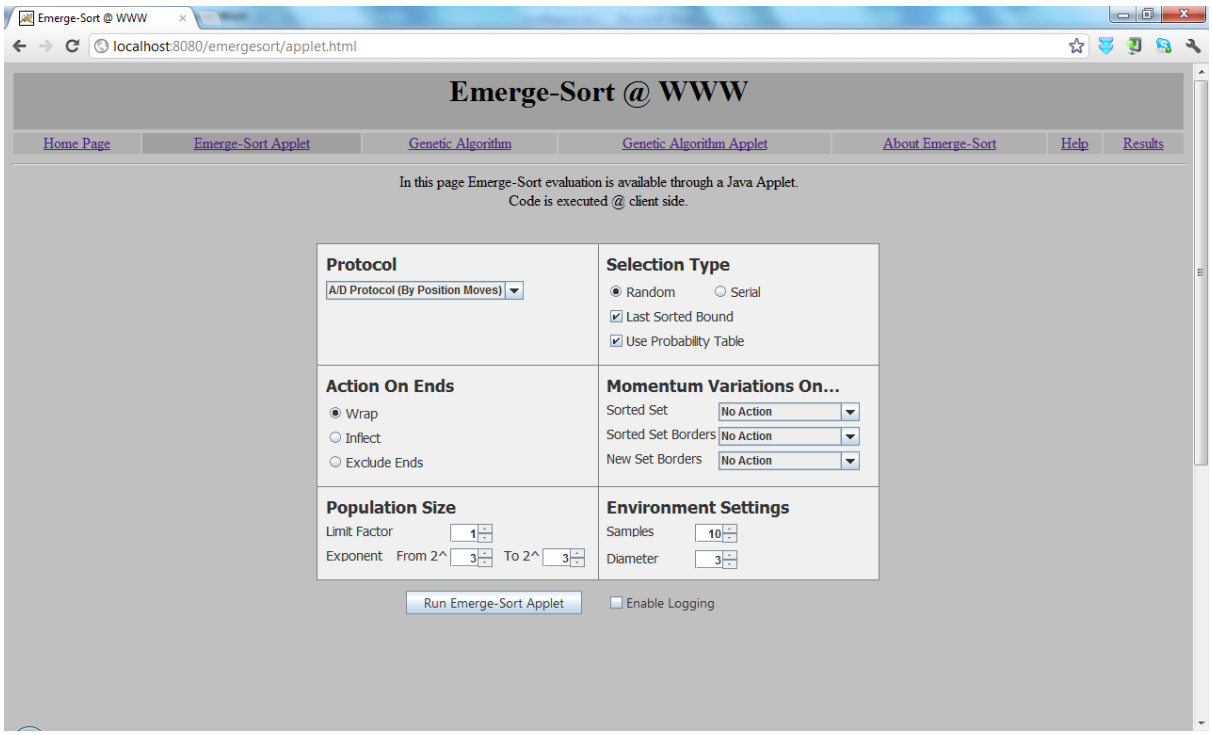
σε μορφή jar (3) και τα στέλνει πίσω στην εφαρμογή περιήγησης Ιστού του χρήστη (4). Από τη στιγμή αυτή ο χρήστης μπορεί να εισάγει (5) τις παραμέτρους προσομοίωσης στο γραφικό περιβάλλον του Java Applet. Οι υπολογισμοί εκτελούνται από την εφαρμογή περιήγησης Ιστού (6) και τα αποτελέσματα εμφανίζονται (7) στο περιβάλλον εκτέλεσης του Applet.



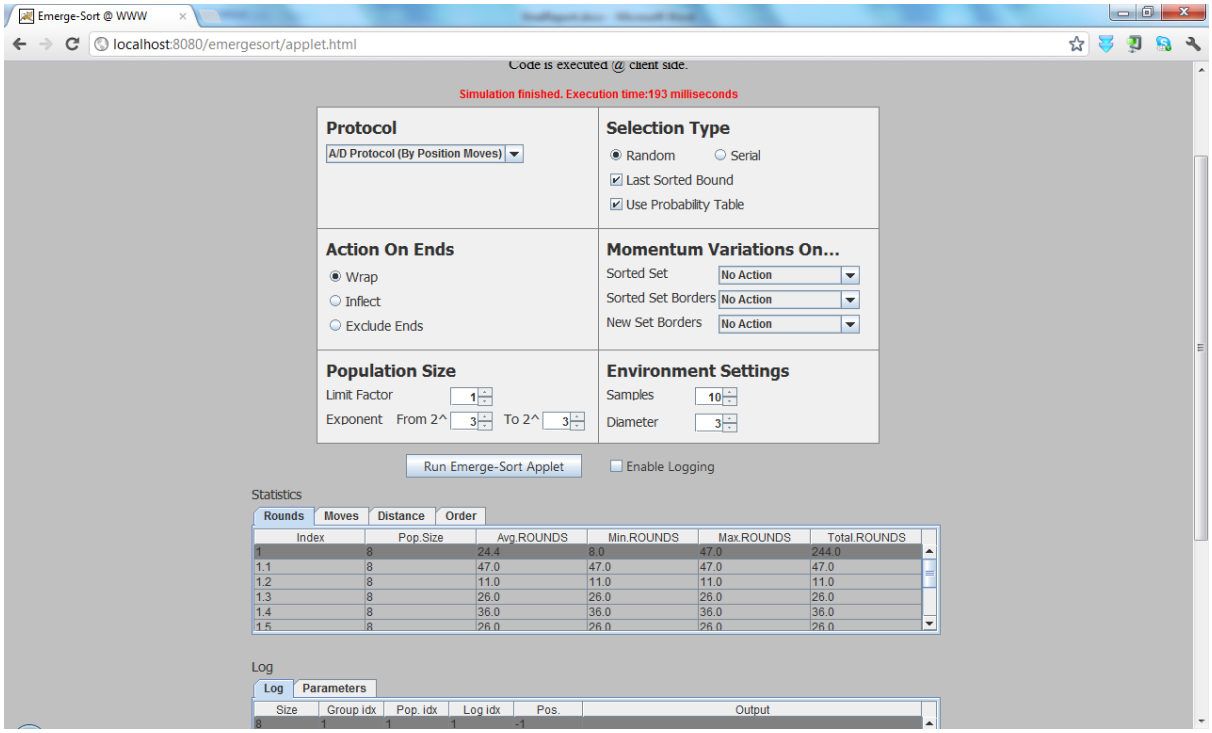
Εικόνα 4.2: Η ροή δεδομένων στην Εφαρμογή Ιστού με εκτέλεση υπολογισμών στον χρήστη.

4.1 Λειτουργικότητα

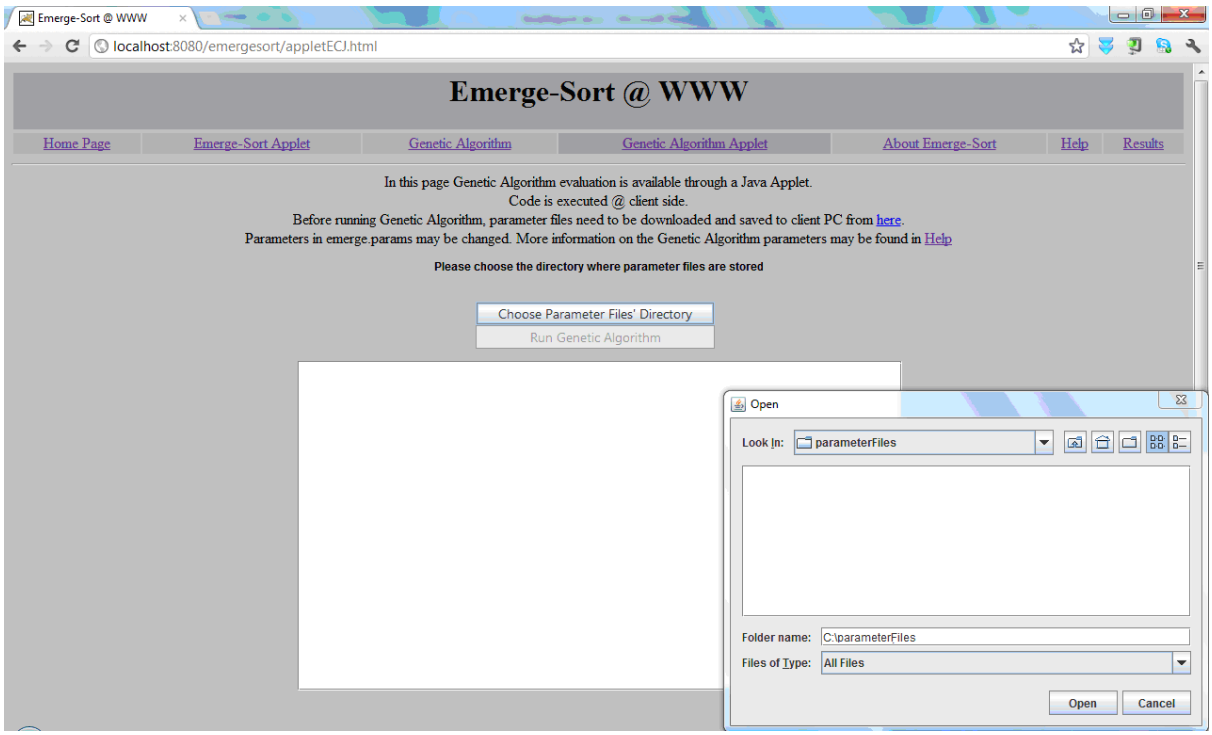
Επιλέχθηκε τα δύο νέα Java Applets για την εκτέλεση των παραλλαγών του Emerge-Sort και του γενετικού αλγορίθμου να ενσωματωθούν σε αντίστοιχες HTML σελίδες, μέρος της διαδικτυακής εφαρμογής ώστε ο χρήστης να έχει τη δυνατότητα να επιλέξει μεταξύ της εκτέλεσης σε Servlet ή σε Applet. Οι νέες σελίδες ονομάστηκαν Emerge-Sort Applet (για την ενσωμάτωση του Applet εκτέλεσης παραλλαγών του Emerge-Sort) και Genetic Algorithm Applet (για την ενσωμάτωση του Applet εκτέλεσης γενετικού αλγορίθμου) και ενσωματώθηκαν στο μενού της διαδικτυακής εφαρμογής.



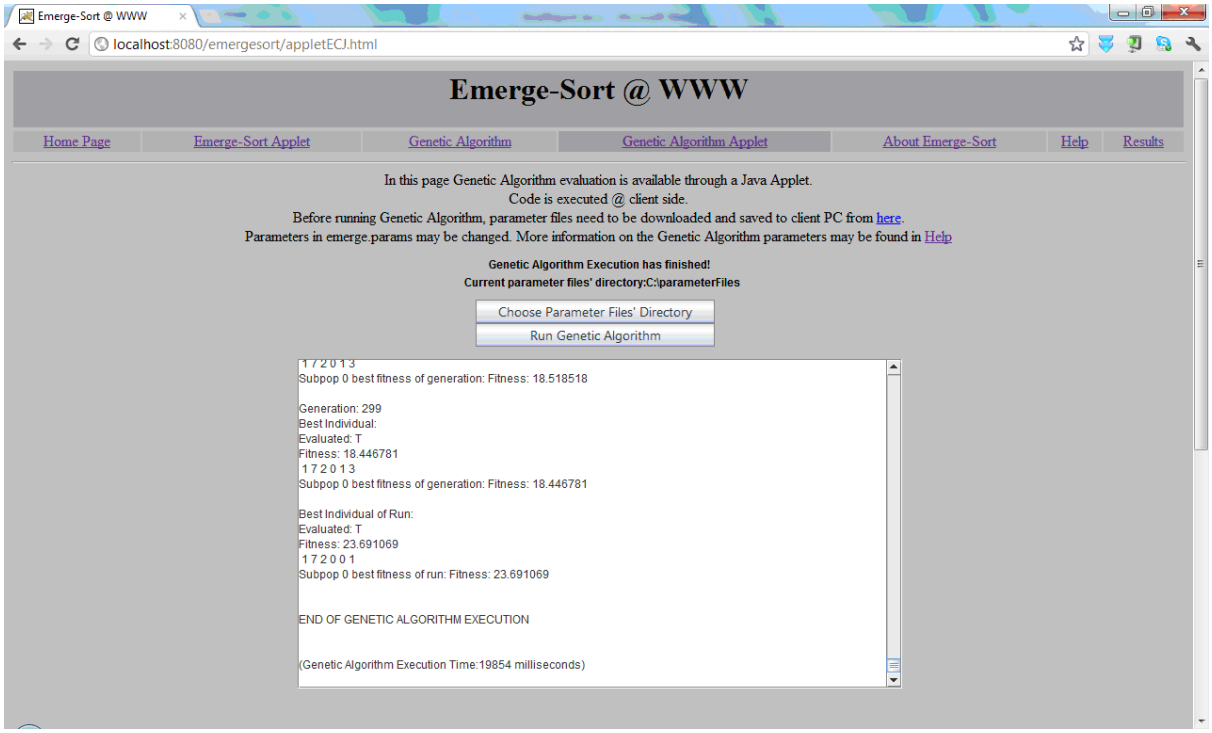
Εικόνα 4.3: Η σελίδα που ενσωματώνει το Applet εκτέλεσης παραλλαγών του Emerge-Sort.



Εικόνα 4.4: Τα αποτελέσματα εκτέλεσης παραλλαγών του Emerge-Sort στο Applet.



Εικόνα 4.5: Η σελίδα που ενσωματώνει το Applet εκτέλεσης του γενετικού αλγορίθμου.



Εικόνα 4.6: Τα αποτελέσματα εκτέλεσης του γενετικού αλγορίθμου στο αντίστοιχο Applet.

Και για τα δύο applets επιλέχθηκε παρόμοια σχεδίαση με τις αντίστοιχες HTML φόρμες της αρχικής σελίδας, χρησιμοποιώντας βέβαια γραφικά στοιχεία της Java. Η έξοδος και των δύο

applets γίνεται στα πλαίσια του γραφικού περιβάλλοντός και δε δημιουργείται κάποιο αρχείο εξόδου. Το Applet του γενετικού αλγορίθμου διαφέρει λίγο από το αντίστοιχο Servlet καθώς δεν ορίζεται το όνομα του αρχείου παραμέτρων αλλά ο κατάλογος που βρίσκεται. Πρέπει επίσης να σημειωθεί πως για την ορθή εκτέλεση των Java Applets χρειάστηκε να υπογραφούν ψηφιακά με μη έμπιστο πιστοποιητικό. Σε κάθε μήνυμα προειδοποίησης ασφάλειας για τα συγκεκριμένα Applets ο χρήστης πρέπει να επιτρέψει την εκτέλεσή τους.

4.2 Υλοποίηση

Για την ανάπτυξη των Java Applets επιλέχθηκε να δημιουργηθούν 2 νέα eclipse projects, ένα για το applet εκτέλεσης παραλλαγών του emergesort (emergesortapplet) και ένα για το applet εκτέλεσης του γενετικού αλγορίθμου (ecjapplet). Τα αρχεία που αποτελούν την υλοποίηση αυτού του κομματιού της εφαρμογής είναι τα source αρχεία του project emergesortapplet (**emergeSortApplet.java, MyLogTableCellRenderer.java, MyStatisticsTableCellRenderer.java**), τα source αρχεία του project ecjapplet (**ECJApplet.java**) και οι σελίδες (**WebContent/applet.html και WebContent/appletECJ**) που ενσωματώνουν τα applet στο emergesort project.

4.2.1 Το Applet για την Εκτέλεση του Emerge-Sort (emergeSortApplet.java)

Το Java Applet αντικαθιστά την HTML φόρμα. Για την εκτέλεση της ταξινόμησης χρησιμοποιείται το jar EmergeSort.jar. Για να συμπεριληφθεί στο Project εισήχθη ως βιβλιοθήκη από το menu Project->Properties->Java Build Path->Add JARs.

Η μέθοδος **emergeSortApplet** είναι ο constructor της κλάσης και εκεί γίνεται η αρχικοποίηση των gui components του Applet.

Το πάτημα του κουμπιού εκτέλεσης ενεργοποιεί τη συνάρτηση **mouseReleased** η οποία παίρνει τις παραμέτρους προσομοίωσης από τα GUI components και καλεί τη συνάρτηση runEmergeSortApplet.

Η **runEmergeSortApplet** αναλαμβάνει:

- να μετατρέψει τις μεταβλητές αυτές σε κατάλληλους τύπους και να περάσει τις παραμέτρους σε ένα αντικείμενο τύπου Sorter (από τη βιβλιοθήκη Emergesorter.jar).

- να εκκινήσει την ταξινόμηση.

- να εμφανίσει στο GUI τα αποτελέσματα της ταξινόμησης

Αναλυτικά οι μέθοδοι που χρησιμοποιούνται για τις παραπάνω διαδικασίες (και βοηθητικές μέθοδοι):

runSimulation: Χρησιμοποιείται για να αναθέσει τις παραμέτρους σε ένα αντικείμενο της κλάσης sorter και να εκτελέσει την προσομοίωση (sorter.Sort();). Περιλαμβάνει λειτουργικότητα αντίστοιχη της μεθόδου Run() από τη standalone εφαρμογή.

printSimulationStatistics: Χρησιμοποιείται για να τυπώσει τα simulation Stats. Περιλαμβάνει λειτουργικότητα αντίστοιχη της μεθόδου FillStatsGrid(Sorter sorter) από τη standalone εφαρμογή.

printSimulationLog: Χρησιμοποιείται για να τυπώσει το simulation Log. Περιλαμβάνει λειτουργικότητα αντίστοιχη της μεθόδου FillLogGrid (Sorter sorter) από τη standalone εφαρμογή.

Οι συναρτήσεις: **getEmergeTypeFromString, getAtomSelectionTypeFromString, getWrapTypeFromString, getSortedSetActionFromString, getSortedSetBorders ActionFromString, getUnSortedSetBordersActionFromString** μετατρέπουν τις αντίστοιχες String ή Boolean μεταβλητές σε τύπους της κλάσης Enm που μπορεί να χειριστεί ο Sorter. Περιλαμβάνουν λειτουργικότητα αντίστοιχη των μεθόδων get_____Type από τη standalone εφαρμογή.

Περισσότερες λεπτομέρειες για την υλοποίηση μπορούν να βρεθούν στα σχόλια του κώδικα.

4.2.2 Η κλάση MyLogTableCellRenderer (MyLogTableCellRenderer.java)

Η κλάση αυτή χρησιμοποιείται για τη μορφοποίηση του Log Table .

4.2.3 Η κλάση MyStatisticsTableCellRenderer (MyStatisticsTableCellRenderer.java)

Η κλάση αυτή χρησιμοποιείται για τη μορφοποίηση του Statistics Table .

4.2.4 Το Applet για την Εκτέλεση του Γενετικού Αλγορίθμου (emergeSortApplet.java)

Το Java Applet αντικαθιστά την αντίστοιχη HTML φόρμα. Για την εκτέλεση του γενετικού αλγορίθμου χρησιμοποιούνται τα jar αρχεία ECJRuntime.jar, ecj.jar, EmergeSorter.jar. Για να συμπεριληφθούν στο Project εισήχθησαν ως βιβλιοθήκες από το menu Project->Properties->Java Build Path->Add JARs του eclipse. Το ecj.jar είναι μία βιβλιοθήκη για την εκτέλεση σεναρίων γενετικών αλγορίθμων, το ECJRuntime.jar είναι ένα project που αναπτύχθηκε στα πλαίσια προηγούμενης εργασίας και καλεί τη βιβλιοθήκη ecj.jar ενώ το EmergeSorter.jar είναι η βιβλιοθήκη που αναπτύχθηκε στα πλαίσια προηγούμενης εργασίας και περιλαμβάνει τη λειτουργικότητα των παραλλαγών του Emerge-Sort. Για τις ανάγκες της παρούσας εργασίας χρειάστηκε να γίνουν κάποιες αλλαγές στον κώδικα του ECJRuntime ώστε η έξοδος του γενετικού αλγορίθμου να περάσει από τη standard έξοδο σε ένα αντικείμενο τύπου JTextArea. Επιπλέον, για να καταστεί δυνατό το ECJApplet να εκτελέσει τον ECJ χρειάστηκε να υπογραφεί ψηφιακά με την ακόλουθη διαδικασία[25]:

- εκτελέστηκε η εντολή **keytool -genkey -alias key** για να δημιουργηθεί ένα κλειδί με το ψευδόνυμο key.
- εκτελέστηκε η εντολή **keytool -selfcert -alias key** ώστε να δημιουργηθεί ένα πιστοποιητικό για αυτό το κλειδί.
- τέλος, για όλα τα jar files που χρησιμοποιούνται από το applet εκτελέστηκε η εντολή **jarsigner <JAR_FILE> key**

Τα νέα, υπογεγραμμένα jar αρχεία αντιγράφηκαν στο eclipse project της συνολικής εφαρμογής (emergesort). Οι παραπάνω εντολές είναι μέρος του Java Development Kit και είναι διαθέσιμες από το command promt των Windows αν έχουν τεθεί σωστά οι μεταβλητές περιβάλλοντος PATH και JAVA_HOME.

Η μέθοδος **ECJApplet** είναι ο constructor της κλάσης και εκεί γίνεται η αρχικοποίηση των gui components του Applet.

Το πάτημα του κουμπιού “Choose Parameter Files” ενεργοποιεί τη συνάρτηση **actionPerformed** η οποία ελέγχει αν ο κατάλογος που επιλέχθηκε περιέχει αρχεία παραμέτρων και τοτε ενεργοποιεί το κουμπί Run Genetic το οποίο αν πατηθεί καλεί τη συνάρτηση runGenetic.

Η **runGenetic** αναλαμβάνει να ανακατευθύνει τη standard έξοδο σε ένα αντικείμενο τύπου JTextArea, να εκκινήσει τον γενετικό αλγόριθμο και να εμφανίσει στο GUI τα αποτελέσματά του.

Κεφάλαιο 5
Πειραματική Ανάλυση και Συμπεράσματα

Η εφαρμογή Ιστού EmegreSort @ WWW που αναπτύχθηκε στα πλαίσια της παρούσας διατριβής, ελέχγθηκε σε όλα τα στάδια υλοποίησής της ώστε να εξασφαλιστεί η παροχή ισοδύναμης λειτουργικότητας με την αυτόνομη εφαρμογή EmergeSortUI. Μετά το πέρας της υλοποίησης η εφαρμογή χρησιμοποιήθηκε για την εκτέλεση δοκιμών σχετικά με την απόδοσή της σε περιβάλλον εκτέλεσης υπολογισμών τόσο στον εξυπηρετητή όσο και στο χρήστη. Επιπλέον, η εφαρμογή χρησιμοποιήθηκε για την αξιολόγηση της επίδρασης στα αποτελέσματα του αλγορίθμου EmergeSort ορισμένων από τις διαθέσιμες παραλλαγές του.

Όπως αναφέρθηκε σε προηγούμενο κεφάλαιο, ο κύριος λόγος μετατροπής της αυτόνομης εφαρμογής EmergeSortUI σε εφαρμογή Ιστού ήταν η αποδοτικότερη εκτέλεση των απαιτούμενων υπολογισμών. Μεταφέροντας την εκτέλεση από έναν προσωπικό υπολογιστή σε έναν εξυπηρετητή Ιστού με περισσότερους υπολογιστικούς πόρους, είναι αναμενόμενο να παρατηρηθούν συντομότεροι χρόνοι εκτέλεσης, γεγονός που επιβεβαιώνεται από τα πειράματα που παρουσιάζονται. Η εναλλακτική μέθοδος εκτέλεσης στο περιβάλλον του χρήστη μέσω Java

Applets δεν είναι τόσο αποδοτική όσο η εκτέλεση στον εξυπηρετητή, αλλά παρ'όλα αυτά πετυχαίνει οριακά καλύτερους χρόνους σε σχέση με την αυτόνομη εφαρμογή και μπορεί να χρησιμοποιηθεί για να αποφευχθεί υπερφόρτωση του εξυπηρετητή.

5.1 Πειράματα Αξιολόγησης της Απόδοσης

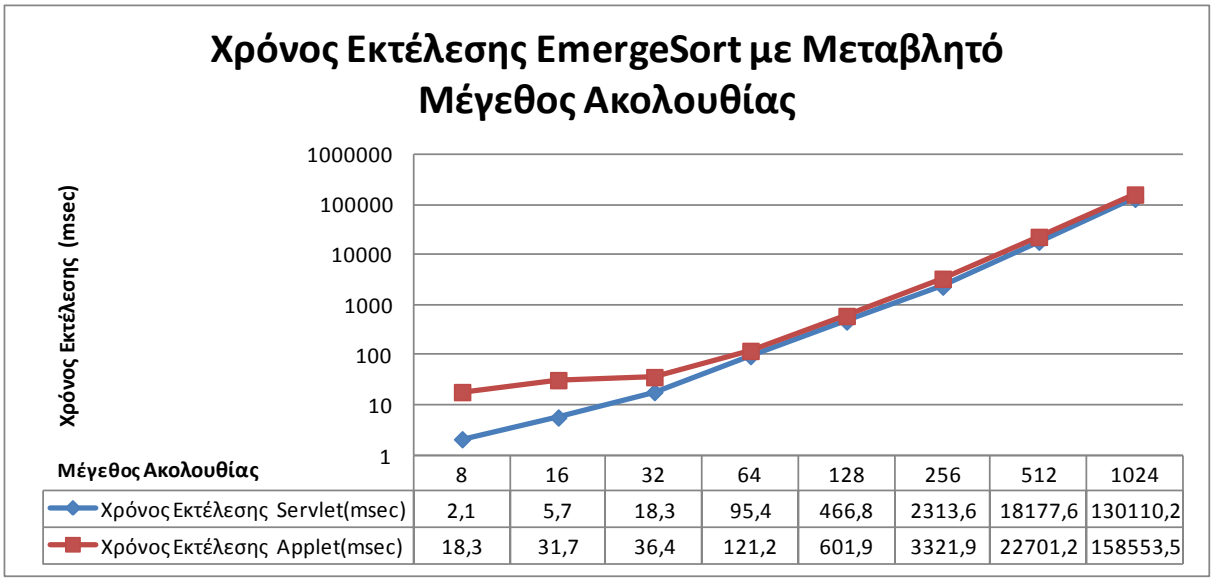
Σε πρώτο στάδιο εκτελέστηκαν ορισμένες δοκιμές, ώστε να αξιολογηθεί η απόδοση της εφαρμογής Ιστού όσον αφορά το χρόνο εκτέλεσης. Ίδιες παράμετροι εκτέλεσης, τόσο για τον αλγόριθμο EmergeSort όσο και για τον γενετικό αλγόριθμο, χρησιμοποιήθηκαν στα αντίστοιχα Servlet και Applet. Καταγράφηκαν οι μέσοι όροι των χρόνων εκτέλεσης και επιχειρήθηκε μία σύγκριση της απόδοσης εκτέλεσης σε περιβάλλον εξυπηρετητή έναντι της απόδοσης εκτέλεσης στο περιβάλλον χρήστη. Στις επόμενες παραγράφους περιγράφονται οι ρυθμίσεις και τα αποτελέσματα των προαναφερθέντων πειραμάτων καθώς επίσης και τα συμπεράσματα που προκύπτουν. Τα πρώτα τέσσερα πειράματα αφορούν τον αλγόριθμο EmergeSort ενώ το τελευταίο το γενετικό αλγόριθμο. Αξίζει να σημειωθεί ότι, όπου γίνεται αναφορά σε χρόνους εκτέλεσης, εννοούνται χρόνοι εκτέλεσης υπολογισμών από την έναρξη μέχρι τη λήξη τους. Δεν περιλαμβάνονται σε αυτούς χρόνοι μεταφοράς από το δίκτυο ή χρόνοι επεξεργασίας των δεδομένων εισόδου. Κάθε πείραμα διεξήχθη δέκα (10) φορές και ελήφθησαν οι μέσοι όροι των χρόνων εκτέλεσης.

Σε αυτό το σημείο πρέπει να σημειωθεί πως οι αντίστοιχες επιδόσεις εκτέλεσης στην αυτόνομη εφαρμογή EmergeSortUI ήταν υποδεέστερες από τις αντίστοιχες επιδόσεις με το Java Applet μοντέλο. Παρ'ολο που η αυτόνομη εφαρμογή δεν περιλαμβάνει μέτρηση χρόνου εκτέλεσης, οι δοκιμές απέδειξαν ότι η χρήση του EmergeSortUI απαιτεί επιπλέον χρόνο σε σχέση με την εκτέλεση των αντίστοιχων Applet. Η εκτέλεση των αντίστοιχων Servlet είναι, όπως περιγράφεται στη συνέχεια, ακόμη πιο γρήγορη καθώς γίνεται στον εξυπηρετητή ο οποίος διαθέτει περισσότερους υπολογιστικούς πόρους.

5.1.1 EmergeSort: Μεταβλητά Μεγέθη Ακολουθίας

Σε αυτό το πείραμα αναλύεται η διαφορά στο χρόνο εκτέλεσης του αλγορίθμου Emerge-Sort μεταξύ Servlet και Applet περιβάλλοντος. Μεταβλητή του πειράματος ήταν το μέγεθος της ακολουθίας (δυνάμεις του 2 από το 3 έως το 10). Οι υπόλοιπες παράμετροι διατηρήθηκαν σταθερές και είναι οι εξής:

- **Protocol** : A/D with Position Moves
- **Selection Type**:Random
- **Use Probability Table**: Ενεργό
- **Last Sorted Bound**: Ενεργό
- **Action On Ends**: Wrap
- **Momentum Variations On Sorted Set**: NoAction
- **Momentum Variations On Sorted Set Borders**: NoAction
- **Momentum Variations On New Set Borders**: NoAction
- **Limit Factor**: 1
- **Samples**:20
- **Diameter**:3
- **Enable Logging**: Μη ενεργό



Εικόνα 5.1: Χρόνοι εκτέλεσης του αλγορίθμου Emerge-Sort με μεταβλητό μέγεθος ακολουθίας.

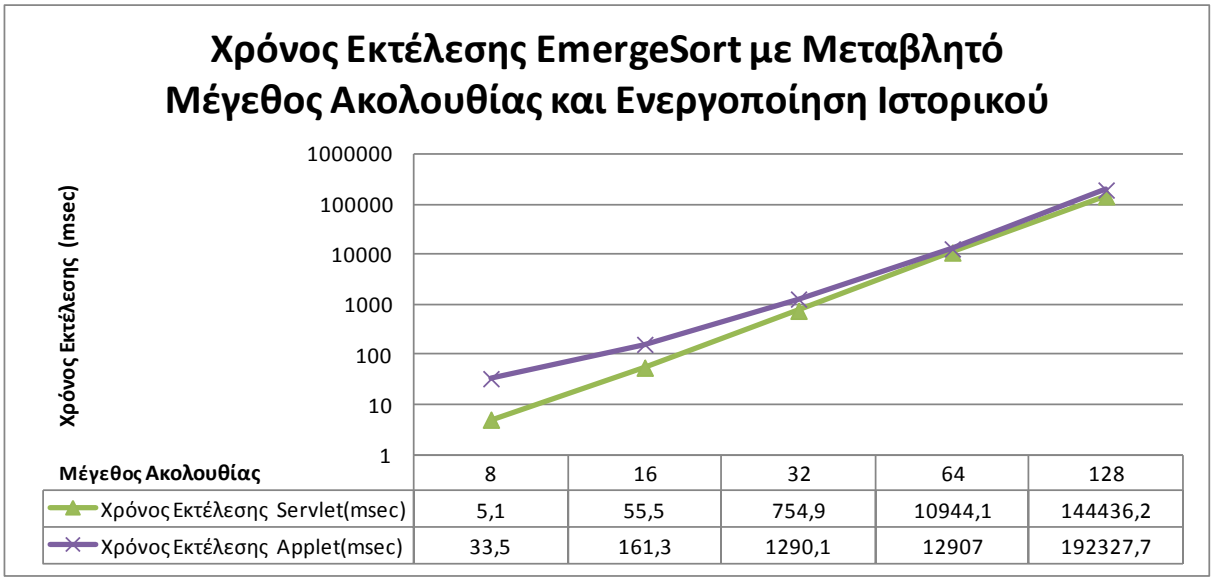
Από το πείραμα αυτό έγινε φανερό ότι ο χρόνος εκτέλεσης αυξάνεται μη γραμμικά σε σχέση με τις δυνάμεις του 2 όσο αυτές αυξάνονται. Ο χρόνος εκτέλεσης ήταν μεγαλύτερος όταν χρησιμοποιήθηκε το Applet μοντέλο εκτέλεσης για κάθε μέγεθος ακολουθίας.

Πρέπει να τονισθεί σε αυτό το σημείο ότι οι διαφορές στο χρόνο εκτέλεσης μεταξύ εκτέλεσης στον εξυπηρετητή (χρήση Servlet) ή στο περιβάλλον του χρήστη (χρήση Applet) εξαρτώνται από την υπολογιστική ισχύ και τον φόρτο των συστημάτων που χρησιμοποιήθηκαν ως εξυπηρετητής και ως σύστημα χρήστη. Αν επέλθει περισσότερος φόρτος στον εξυπηρετητή (από περισσότερες προσπελάσεις της εφαρμογής) ή χρησιμοποιηθεί σύστημα χρήστη με περισσότερη υπολογιστική ισχύ τα αποτελέσματα μπορεί να αντιστραφούν. Στο τρέχον περιβάλλον εκτέλεσης η εκτέλεση στον εξυπηρετητή παρουσίασε καλύτερα αποτελέσματα.

5.1.2 EmergeSort: Μεταβλητά Μεγέθη Ακολουθίας και Καταγραφή Ιστορικού

Σε αυτό το πείραμα αναλύεται πάλι η διαφορά στο χρόνο εκτέλεσης του αλγορίθμου Emerge-Sort μεταξύ Servlet και Applet περιβάλλοντος. Μεταβλητή του πειράματος και αυτή τη φορά ήταν το μέγεθος της ακολουθίας (δυνάμεις του 2 από το 3 έως το 10). Η διαφορά σε σχέση με το προηγούμενο πείραμα είναι η ενεργοποίηση της καταγραφής ιστορικού (Enable Logging), ώστε να καταγράφονται όλες οι κινήσεις της εκάστοτε εκτέλεσης του Emerge-Sort. Οι υπόλοιπες παράμετροι διατηρήθηκαν σταθερές και είναι οι εξής:

- **Protocol** : A/D with Position Moves
- **Selection Type**:Random
- **Use Probability Table**: Ενεργό
- **Last Sorted Bound**: Ενεργό
- **Action On Ends**: Wrap
- **Momentum Variations On Sorted Set**: NoAction
- **Momentum Variations On Sorted Set Borders**: NoAction
- **Momentum Variations On New Set Borders**: NoAction
- **Limit Factor**: 1
- **Samples**:20
- **Diameter**:3
- **Enable Logging**: Ενεργό



Εικόνα 5.2: Χρόνοι **εκτέλεσης του αλγορίθμου Emerge-Sort με μεταβλητό μέγεθος ακολουθίας και ενεργοποιημένη την καταγραφή ιστορικού.**

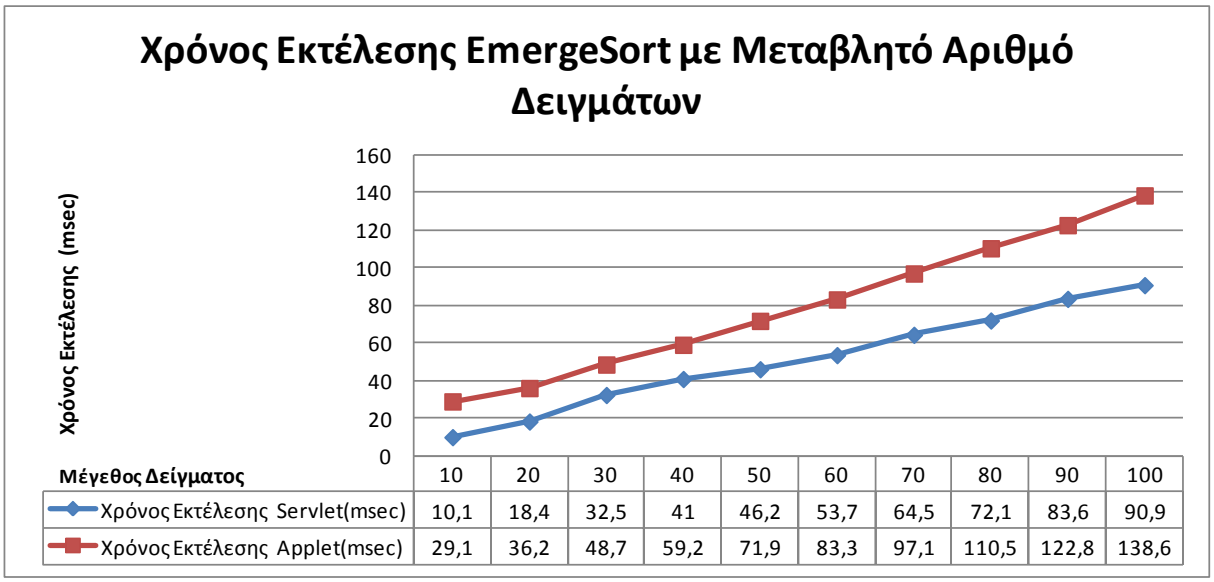
Από το πείραμα αυτό έγινε φανερό ότι ο χρόνος εκτέλεσης αυξάνεται μη γραμμικά σε σχέση με τις δυνάμεις του 2 όσο αυτές αυξάνονται. Ο χρόνος εκτέλεσης ήταν μεγαλύτερος όταν χρησιμοποιήθηκε το Applet μοντέλο εκτέλεσης για κάθε μέγεθος ακολουθίας. Επιπλέον οι αντίστοιχοι χρόνοι εκτέλεσης ήταν σημαντικά μεγαλύτεροι σε σχέση με το αμέσως προηγούμενο

πείραμα. Η ενεργοποίηση καταγραφής ιστορικού αναμενόμενα απαιτεί μεγαλύτερο χρόνο εκτέλεσης για την καταγραφή αποθήκευση όλων των ενδιάμεσων αποτελεσμάτων.

5.1.3 EmergeSort: Μεταβλητά Δείγματα ανά Ακολουθία

Σε αυτό το πείραμα αναλύεται η διαφορά στο χρόνο εκτέλεσης του αλγορίθμου Emerge-Sort μεταξύ Servlet και Applet περιβάλλοντος. Μεταβλητή του πειράματος ήταν το πλήθος των δειγμάτων (από 10 έως 100 με βήμα 10). Οι υπόλοιπες παράμετροι διατηρήθηκαν σταθερές και είναι οι εξής:

- **Protocol** : A/D with Position Moves
- **Selection Type**:Random
- **Use Probability Table**: Ενεργό
- **Last Sorted Bound**: Ενεργό
- **Action On Ends**: Wrap
- **Momentum Variations On Sorted Set**: NoAction
- **Momentum Variations On Sorted Set Borders**: NoAction
- **Momentum Variations On New Set Borders**: NoAction
- **Limit Factor**: 1
- **Exponent (2^)**:5 **Exponent To(2^)**:5
- **Diameter**:3
- **Enable Logging**: Μη ενεργό



Μέγεθος Δείγματος	10	20	30	40	50	60	70	80	90	100
Χρόνος Εκτέλεσης Servlet(msec)	10,1	18,4	32,5	41	46,2	53,7	64,5	72,1	83,6	90,9
Χρόνος Εκτέλεσης Applet(msec)	29,1	36,2	48,7	59,2	71,9	83,3	97,1	110,5	122,8	138,6

Εικόνα 5.3: Χρόνοι **εκτέλεσης του αλγορίθμου Emerge-Sort με μεταβλητό πλήθος δειγμάτων.**

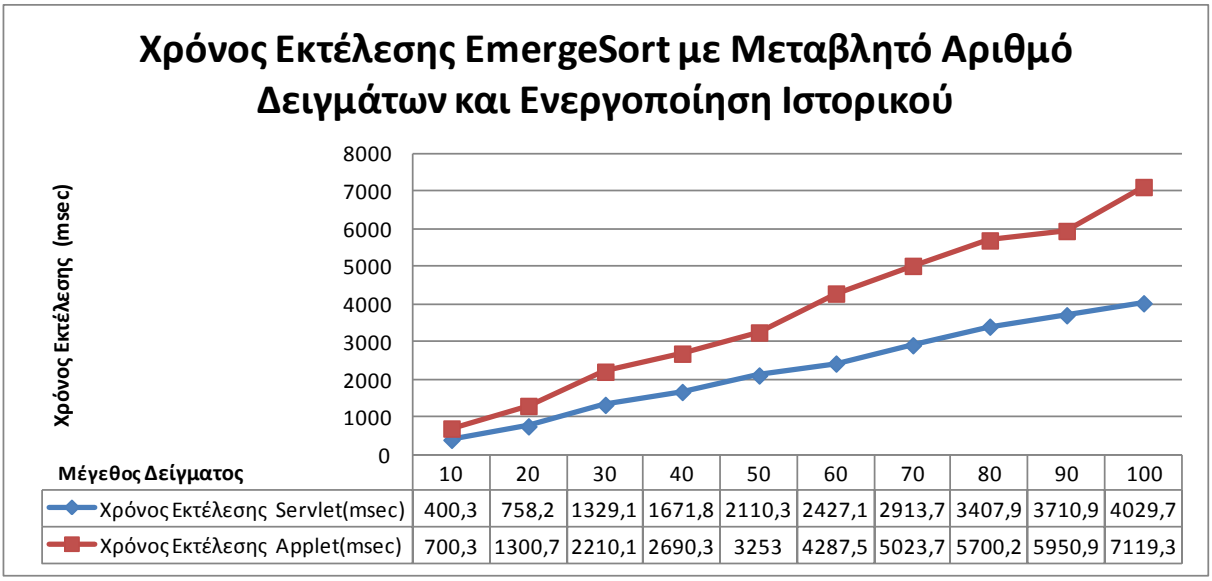
Από το πείραμα αυτό έγινε φανερό ότι ο χρόνος εκτέλεσης αυξάνεται σχεδόν γραμμικά σε σχέση με το πλήθος των δειγμάτων ανά ακολουθία. Στο τρέχον περιβάλλον εκτέλεσης η

εκτέλεση στον εξυπηρετητή παρουσίασε καλύτερα αποτελέσματα καθώς ο χρόνος εκτέλεσης ήταν μεγαλύτερος όταν χρησιμοποιήθηκε το Applet μοντέλο εκτέλεσης για κάθε μέγεθος ακολουθίας.

5.1.4 EmergeSort: Μεταβλητά Δείγματα ανά Ακολουθία και Καταγραφή Ιστορικού

Σε αυτό το πείραμα αναλύεται η διαφορά στο χρόνο εκτέλεσης του αλγορίθμου Emerge-Sort μεταξύ Servlet και Applet περιβάλλοντος. Μεταβλητή του πειράματος ήταν το μέγεθος της ακολουθίας (δυνάμεις του 2 από το 3 έως το 10). Η διαφορά σε σχέση με το αμέσως προηγούμενο πείραμα είναι η ενεργοποίηση της καταγραφής ιστορικού (Enable Logging) ώστε να καταγράφονται όλες οι κινήσεις της εκάστοτε εκτέλεσης του Emerge-Sort. Οι υπόλοιπες παράμετροι διατηρήθηκαν σταθερές και είναι οι εξής:

- **Protocol** : A/D with Position Moves
- **Selection Type**:Random
- **Use Probability Table**: Ενεργό
- **Last Sorted Bound**: Ενεργό
- **Action On Ends**: Wrap
- **Momentum Variations On Sorted Set**: NoAction
- **Momentum Variations On Sorted Set Borders**: NoAction
- **Momentum Variations On New Set Borders**: NoAction
- **Limit Factor**: 1
- **Exponent (2^)**:5 **Exponent To(2^)**:5
- **Diameter**:3
- **Enable Logging**: Ενεργό



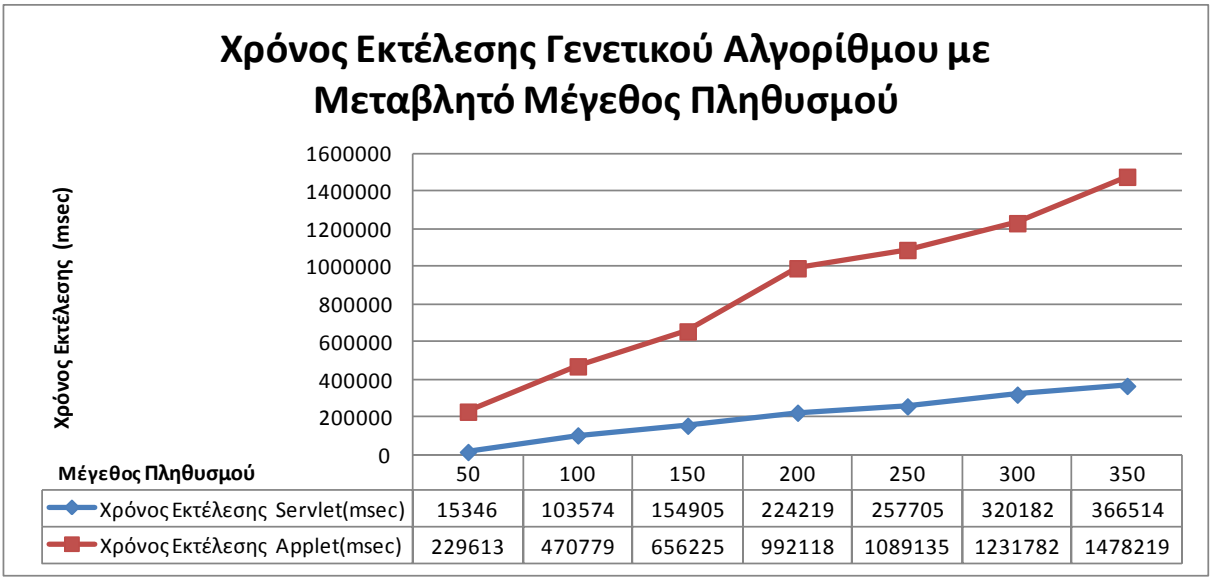
Εικόνα 5.4: Χρόνοι **εκτέλεσης του αλγορίθμου Emerge-Sort με μεταβλητό πλήθος δειγμάτων και ενεργοποιημένη την καταγραφή ιστορικού.**

Από το πείραμα αυτό έγινε φανερό ότι ο χρόνος εκτέλεσης αυξάνεται σχεδόν γραμμικά σε σχέση με το πλήθος των δειγμάτων ανά ακολουθία. Ο χρόνος εκτέλεσης ήταν μεγαλύτερος όταν χρησιμοποιήθηκε το Applet μοντέλο εκτέλεσης για κάθε μέγεθος ακολουθίας. Επιπλέον οι αντίστοιχοι χρόνοι εκτέλεσης ήταν σημαντικά μεγαλύτεροι σε σχέση με το αμέσως προηγούμενο πείραμα. Η ενεργοποίηση καταγραφής ιστορικού αναμενόμενα απαιτεί μεγαλύτερο χρόνο εκτέλεσης για την καταγραφή αποθήκευση όλων των ενδιάμεσων αποτελεσμάτων.

5.1.5 Γενετικός Αλγόριθμος: Μεταβλητό Μέγεθος Πλυθυσμού

Σε αυτό το πείραμα αξιολογήθηκε η διαφορά στην απόδοση του γενετικού αλγορίθμου μεταξύ Servlet και Applet περιβάλλοντος. Ως μεταβλητή του πειράματος χρησιμοποιήθηκε το μέγεθος του πληθυσμού της προς ταξινόμηση ακολουθίας. Από τις υπόλοιπες παραμέτρους του γενετικού αλγορίθμου οι σημαντικότερες είχαν τις εξής τιμές:

- **generations** : 300
- **emerge.LimitFactor**:1
- **emerge.ExponentSize**:4
- **emerge.Diameter**:3
- **emerge.Samples**:5



Εικόνα 5.5: Χρόνοι **εκτέλεσης του γενετικού αλγορίθμου με μεταβλητό μέγεθος ακολουθίας.**

Από το πείραμα αυτό έγινε φανερό ότι και ο χρόνος εκτέλεσης του γενετικού αλγορίθμου αυξάνεται σχεδόν γραμμικά σε σχέση με το μέγεθος της, προς ταξινόμηση, ακολουθίας. Στο τρέχον περιβάλλον εκτέλεσης και με μικρό φόρτο χρήσης του εξυπηρετητή, ο χρόνος εκτέλεσης ήταν μεγαλύτερος όταν χρησιμοποιήθηκε το Applet μοντέλο εκτέλεσης για κάθε μέγεθος ακολουθίας.

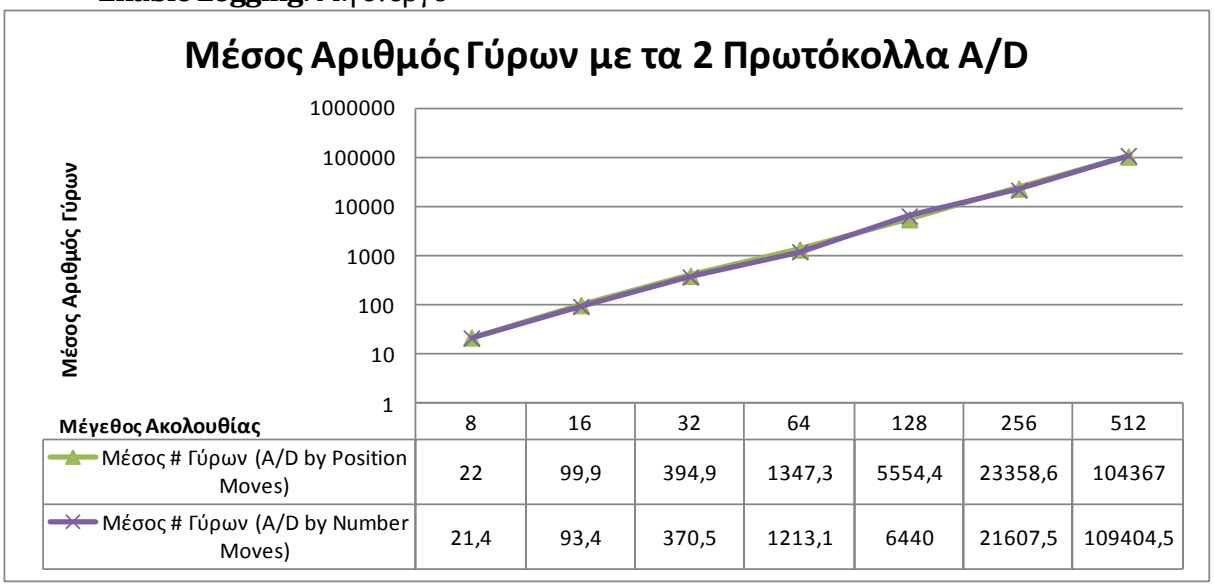
5.2 Πειράματα Αξιολόγησης της Επίδρασης Διαφορετικών Παραμέτρων

Σε δεύτερο στάδιο εκτελέστηκαν πειράματα ώστε να μελετηθεί η επίδραση στην επιλογή διαφορετικών παραμέτρων στην απόδοση του αλγορίθμου Emerge-Sort και του γενετικού αλγορίθμου. Σε αυτό το στάδιο δεν ενδιέφερε ο χρόνος εκτέλεσης αλλά κυρίως ο μέσος αριθμός κινήσεων και γύρων των εκτελέσεων του Emerge-Sort. Για την εκτέλεση των πειραμάτων χρησιμοποιήθηκαν τα αντίστοιχα Applet. Τα Java Servlet χρησιμοποιήθηκαν μόνο για διασταύρωση των αποτελεσμάτων ώστε να επιβεβαιωθεί ότι η λειτουργικότητα που παρέχεται είναι ίδια για το Servlet και το Applet μοντέλο. Στις επόμενες παραγράφους περιγράφονται οι ρυθμίσεις και τα αποτελέσματα των προαναφερθέντων πειραμάτων καθώς επίσης και τα συμπεράσματα που προκύπτουν. Τα πρώτα τέσσερα πειράματα αφορούν τον αλγόριθμο EmergeSort ενώ το τελευταίο το γενετικό αλγόριθμο.

5.2.1 EmergeSort: Διαφορετικά Πρωτόκολλα A/D

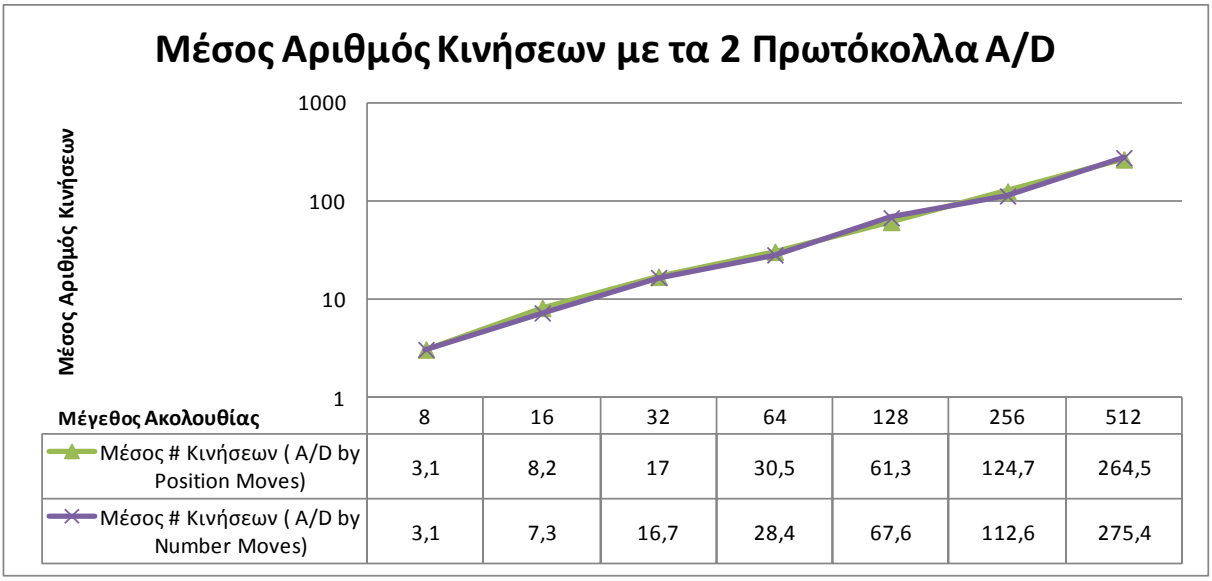
Σε αυτό το πείραμα αναλύεται η διαφορά στο μέσο αριθμό γύρων και τον μέσο αριθμό κινήσεων του αλγορίθμου Emerge-Sort για τα 2 διαθέσιμα A/D πρωτόκολλα (A/D By Position Moves και A/D by Number Moves). Μεταβλητή του πειράματος ήταν το μέγεθος της ακολουθίας (δυνάμεις του 2 από το 3 έως το 9) και ο τύπος του πρωτοκόλλου. Οι υπόλοιπες παράμετροι διατηρήθηκαν σταθερές και είναι οι εξής:

- **Selection Type**:Random
- **Use Probability Table**: Ενεργό
- **Last Sorted Bound**: Ενεργό
- **Action On Ends**: Wrap
- **Momentum Variations On Sorted Set**: NoAction
- **Momentum Variations On Sorted Set Borders**: NoAction
- **Momentum Variations On New Set Borders**: NoAction
- **Limit Factor**: 1
- **Exponent (2^)**:3 **Exponent To(2^)**:9
- **Diameter**:3
- **Enable Logging**: Μη ενεργό



Μέγεθος Ακολουθίας	8	16	32	64	128	256	512
Μέσος # Γύρων (A/D by Position Moves)	22	99,9	394,9	1347,3	5554,4	23358,6	104367
Μέσος # Γύρων (A/D by Number Moves)	21,4	93,4	370,5	1213,1	6440	21607,5	109404,5

Εικόνα 5.6: Μέσος αριθμός γύρων για τα διαφορετικά A/D πρωτόκολλα και μεταβλητό μέγεθος ακολουθίας.



Μέγεθος Ακολουθίας	8	16	32	64	128	256	512
Μέσος # Κινήσεων (A/D by Position Moves)	3,1	8,2	17	30,5	61,3	124,7	264,5
Μέσος # Κινήσεων (A/D by Number Moves)	3,1	7,3	16,7	28,4	67,6	112,6	275,4

Εικόνα 5.7: Μέσος αριθμός κινήσεων για τα διαφορετικά A/D πρωτόκολλα και μεταβλητό μέγεθος ακολουθίας

Όπως φαίνεται στις δύο προηγούμενες εικόνες, το πρωτόκολλο A/D by Number Moves εμφανίζεται οριακά καλύτερο για ακολουθίες μεγέθους 8, 16, 32, 256 αριθμών. Οι διαφορές είναι σχετικά μικρές ενώ για μεγέθη ακολουθίας 128 και 512 αντιστρέφονται οπότε πιθανώς να βρίσκονται στο όριο του στατιστικού σφάλματος.

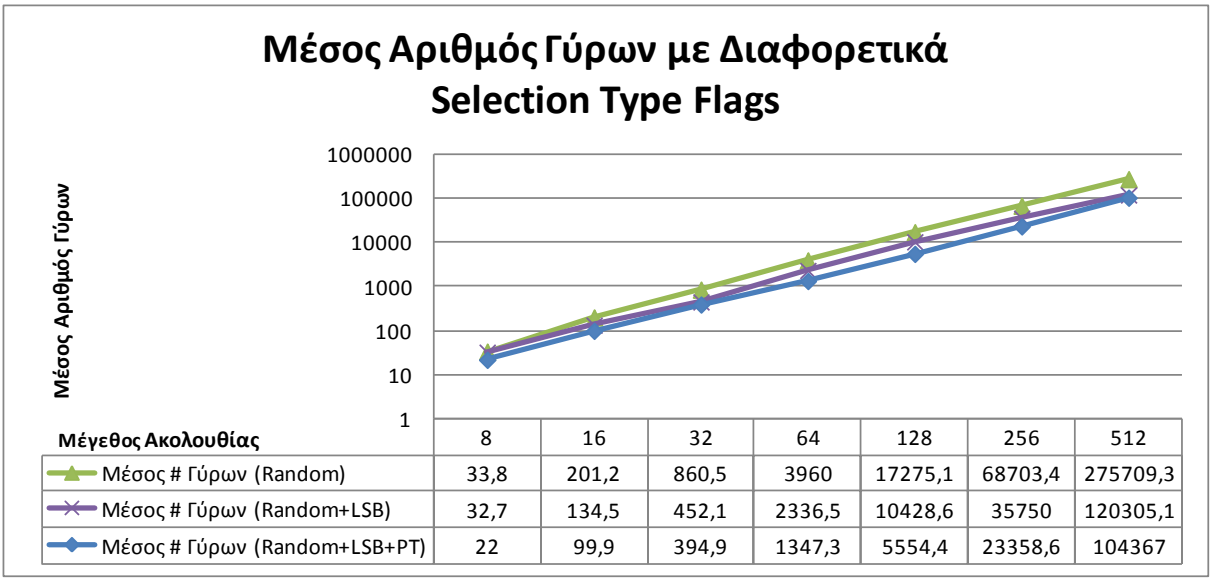
5.2.2 EmergeSort: Διαφορετικός Τρόπος Επιλογής Κέντρου Γειτονιάς

Σε αυτό το πείραμα αναλύεται η διαφορά στο μέσο αριθμό γύρων και τον μέσο αριθμό κινήσεων του αλγορίθμου Emerge-Sort για διαφορετικούς τρόπους επιλογής του κέντρου της γειτονιάς. Μελετόνται οι περιπτώσεις της τυχαίας επιλογής (Random) με ενεργοποιημένες (ή όχι) τις επιλογές Last Sorted Bound και Use Probability Table. Στα πλαίσια παλαιότερης εργασίας [03] δείχτηκε ότι η ενεργοποίηση των επιλογών Last Sorted Bound και Use Probability Table λειτουργεί θετικά οπότε ελέγχθηκαν μόνο οι εξής συνδυασμοί:

- **Selection Type**:Random, **Use Probability Table**: Μη Ενεργό, **Last Sorted Bound**: Μη Ενεργό
- **Selection Type**:Random **Use Probability Table**: Ενεργό **Last Sorted Bound**: Μη Ενεργό
- **Selection Type**:Random **Use Probability Table**: Ενεργό **Last Sorted Bound**: Ενεργό

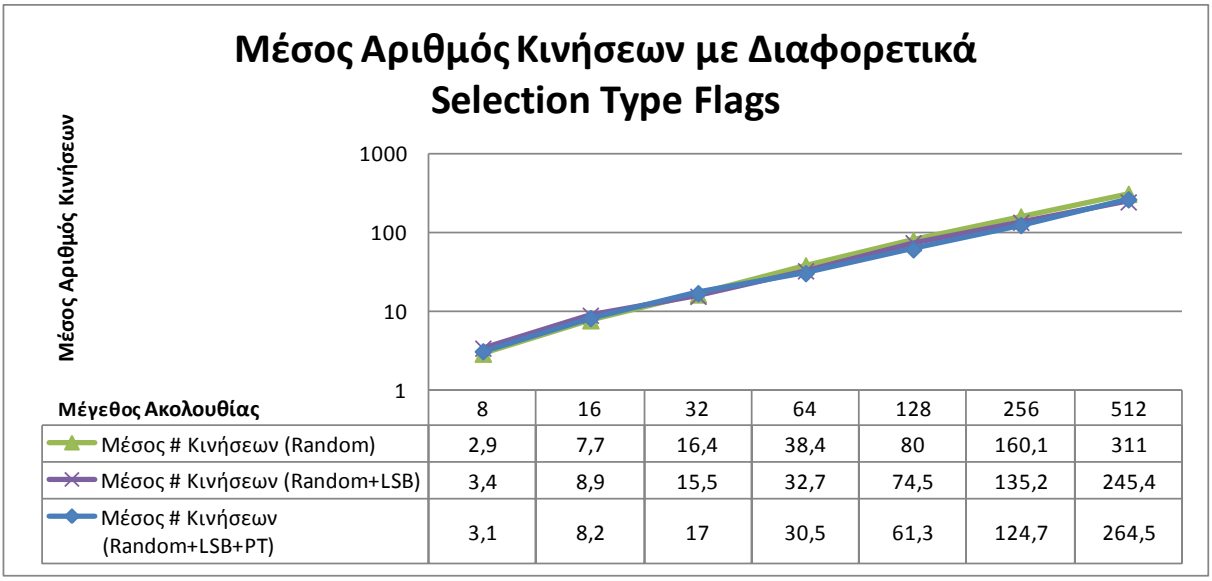
Μεταβλητή του πειράματος ήταν το μέγεθος της ακολουθίας (δυνάμεις του 2 από το 3 έως το 9) και ο τύπος του πρωτοκόλλου. Οι υπόλοιπες παράμετροι διατηρήθηκαν σταθερές και είναι οι εξής:

- **Protocol** : A/D by Position Moves
- **Action On Ends**: Wrap
- **Momentum Variations On Sorted Set**: NoAction
- **Momentum Variations On Sorted Set Borders**: NoAction
- **Momentum Variations On New Set Borders**: NoAction
- **Limit Factor**: 1
- **Exponent (2^)**:3 **Exponent To(2^)**:9
- **Diameter**:3
- **Enable Logging**: Μη ενεργό



Μέγεθος Ακολουθίας	8	16	32	64	128	256	512
Μέσος # Γύρων (Random)	33,8	201,2	860,5	3960	17275,1	68703,4	275709,3
Μέσος # Γύρων (Random+LSB)	32,7	134,5	452,1	2336,5	10428,6	35750	120305,1
Μέσος # Γύρων (Random+LSB+PT)	22	99,9	394,9	1347,3	5554,4	23358,6	104367

Εικόνα 5.8: Μέσος αριθμός γύρων για διαφορετικούς τρόπους επιλογής κέντρου γειτονιάς.



Μέγεθος Ακολουθίας	8	16	32	64	128	256	512
Μέσος # Κινήσεων (Random)	2,9	7,7	16,4	38,4	80	160,1	311
Μέσος # Κινήσεων (Random+LSB)	3,4	8,9	15,5	32,7	74,5	135,2	245,4
Μέσος # Κινήσεων (Random+LSB+PT)	3,1	8,2	17	30,5	61,3	124,7	264,5

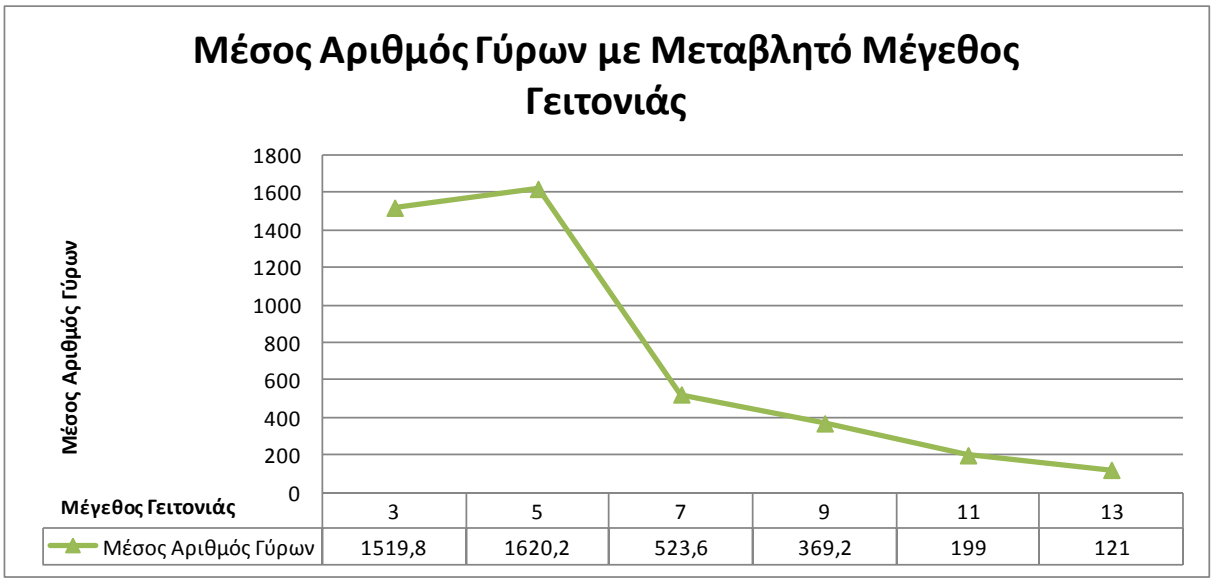
Εικόνα 5.9: Μέσος αριθμός κινήσεων για διαφορετικούς τρόπους επιλογής κέντρου γειτονιάς.

Όπως φαίνεται στις δύο προηγούμενες εικόνες, η ενεργοποίηση των επιλογών Last Sorted Bound και Use Probability Table (ειδικά της 2ης) λειτουργεί θετικά και οδηγεί σε μείωση τόσο του μέσου αριθμού κινήσεων όσο και του μέσου αριθμού γύρων εκτέλεσης. Η μείωση αυτή τείνει να αυξάνεται όσο αυξάνει το μέγεθος της ακολουθίας.

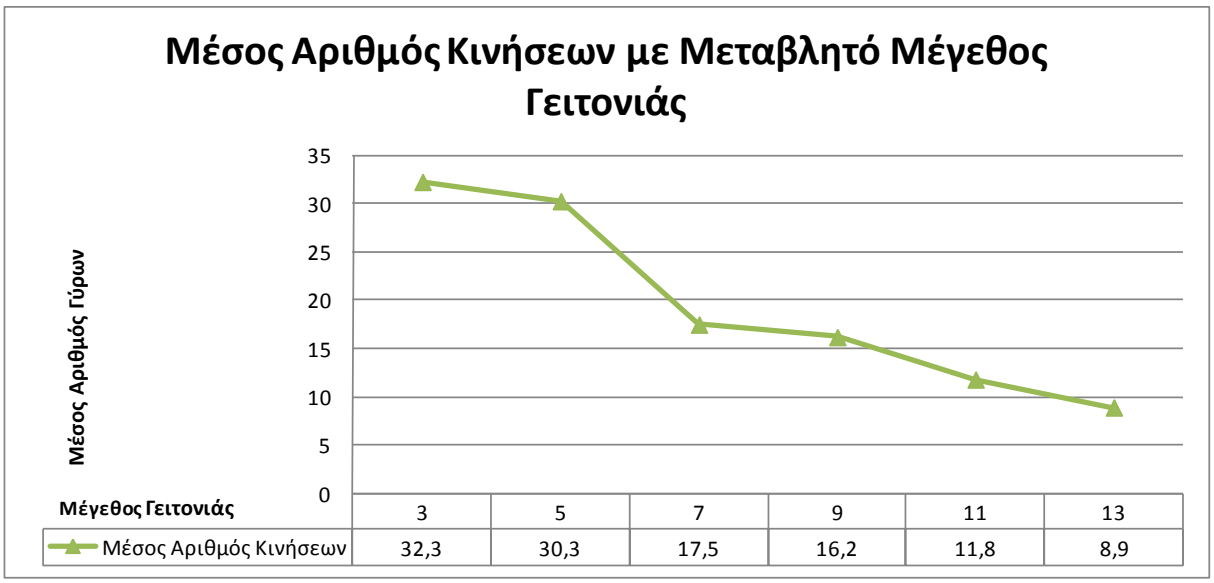
5.2.3 EmergeSort: Μεταβλητά Μεγέθη Γειτονιάς

Σε αυτό το πείραμα αναλύεται η επίδραση που έχει στην απόδοση του αλγορίθμου Emerge-Sort το μέγεθος της γειτονιάς. Ως μεταβλητή του πειράματος χρησιμοποιήθηκε το μέγεθος της γειτονιάς παίρνοντας περιττές τιμές από 3 έως 13. Πρέπει να σημειωθεί ότι οι άρτιες τιμές οδηγούσαν σε μη ταξινομημένες ακολουθίες. Οι υπόλοιπες παράμετροι διατηρήθηκαν σταθερές και είναι οι εξής:

- **Protocol** : A/D by Position Moves
- **Action On Ends**: Wrap
- **Selection Type**:Random
- **Use Probability Table**: Ενεργό
- **Last Sorted Bound**: Ενεργό
- **Momentum Variations On Sorted Set**: NoAction
- **Momentum Variations On Sorted Set Borders**: NoAction
- **Momentum Variations On New Set Borders**: NoAction
- **Limit Factor**: 1
- **Exponent (2^)**:6 **Exponent To(2^)**:6
- **Diameter**:3
- **Enable Logging**: Μη ενεργό
- **Samples**: 50



Εικόνα 5.10: Μέσος αριθμός κινήσεων για διαφορετικούς τρόπους επιλογής κέντρου γειτονιάς.

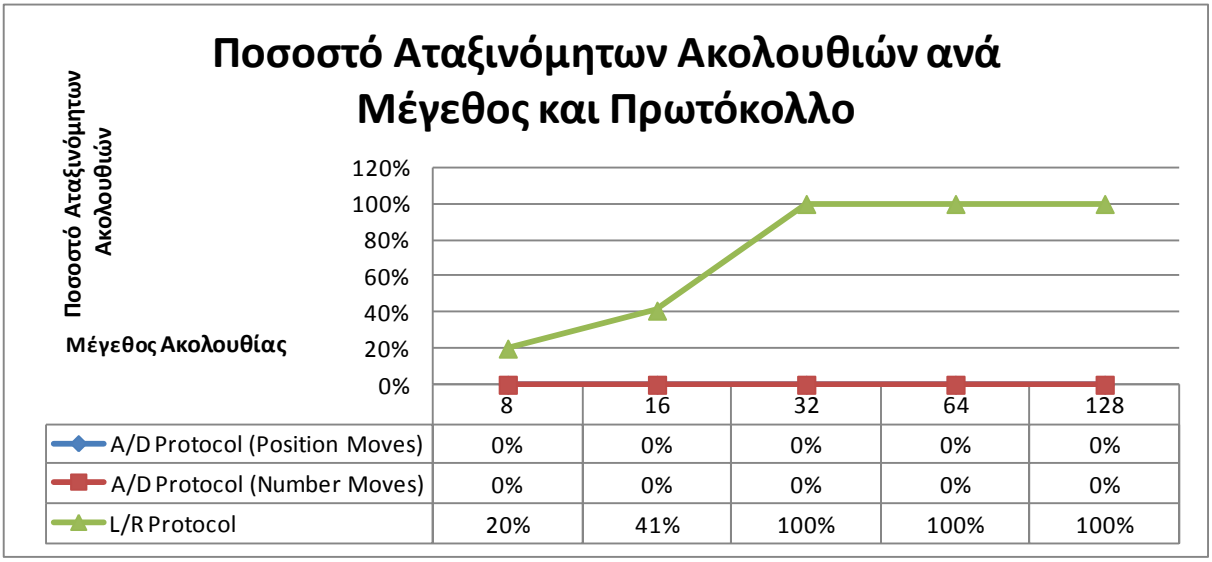


Εικόνα 5.11: Μέσος αριθμός κινήσεων για διαφορετικούς τρόπους επιλογής κέντρου γειτονιάς.

Όπως φαίνεται στις δύο προηγούμενες εικόνες, η αύξηση του μεγέθους της γειτονιάς οδηγεί σε μικρότερο αριθμό γύρων εκτέλεσης και κινήσεων επομένως φαίνεται να λειτουργεί θετικά για το πρωτόκολλο A/D by Position Moves που επιλέχθηκε.

5.2.4 EmergeSort: Ποσοστά Αταξινόμητων Ακολουθιών ανά Πρωτόκολλο

Από τις πρώτες δοκιμές της εφαρμογής φάνηκε ότι το πρωτόκολλο L/R οδηγεί συχνά σε μη ταξινομημένες ακολουθίες. Το γεγονός αυτό αποτυπώνεται στην παρακάτω εικόνα όπου παρατίθενται τα ποσοστά μη ταξινομημένων ακολουθιών (για κάθε μέγεθος) σε ένα πλήθος 10 δειγμάτων κάθε φορά. Τα ποσοστά αποτυχίας του L/R πρωτοκόλλου είναι μεγάλα για ακολουθίες μεγέθους μεγαλύτερου του 16.



Εικόνα 5.12: Ποσοστό αταξινόμητων ακολουθιών ανά μέγεθος και πρωτόκολλο.

5.2.5 Γενετικός Αλγόριθμος: Γενικά Σχόλια

Όπως και στην παλαιότερη εργασία [03] όπου αξιολογήθηκαν οι διαφορετικές παραλλαγές του Emerge-Sort, για την αναπαράσταση των διαφορετικών παραμέτρων χρησιμοποιήθηκαν άτομα τα οποία αποτελούνται από έξι γονίδια: ένα για το βασικό πρωτόκολλο και πέντε για τις βασικές παραμέτρους του κάθε πρωτοκόλλου. Ο γενετικός αλγόριθμος θα εκτελεί τυφλά όλους τους δυνατούς συνδυασμούς παραμέτρων και είναι αναμενόμενο ότι συνδυασμοί που δεν οδηγούν σε ταξινόμηση θα αντιπροσωπεύονται από άτομα που θα εξαλειφθούν από τους πληθυσμούς του γενετικού αλγορίθμου, ενώ η αυτοματοποιημένη εκτέλεση αναμένεται να αναδείξει τους πιο αποδοτικούς από αυτούς με αντικειμενικό τρόπο. Στους παρακάτω πίνακες περιγράφονται οι δυνατές τιμές για κάθε γονίδιο και η σημασία τους.

Τιμές	Περιγραφή
0	Πρωτόκολλο L/R

1	Πρωτόκολλο A/D

Πίνακας 5.1: 1ο γονίδιο – Επιλογή πρωτοκόλλου (Πηγή [03])

Τιμές	Περιγραφή
0	*Serial* - ο αλγόριθμος επισκέπτεται τις γειτονιές διαδοχικά, από αριστερά προς τα δεξιά
1	*Serial + LastSortedBound* - διαδοχικά εκτός αν προηγηθεί ταξινόμηση οπότε εξετάζεται ένα από τα δύο όρια της γειτονιάς που ταξινομήθηκε
2	*Serial + Use Probability Table* – σειριακή επίσκεψη αλλά αποκλείονται οι θέσεις που δεν έχουν πιθανότητα να βοηθήσουν στην εξέλιξη, χρήση βοηθητικού πίνακα
3	*Serial + LastSortedBound + Use Probability Table* – Συνδυασμός των τριών παραπάνω μεθόδων.
4	*Random* – ο αλγόριθμος επισκέπτεται τις γειτονιές με τυχαίο τρόπο, όλοι οι αριθμοί συμμετέχουν με ίση πιθανότητα
5	*Random+ LastSortedBound* – με τυχαίο τρόπο εκτός αν προηγηθεί ταξινόμηση οπότε εξετάζεται ένα από τα δύο όρια της γειτονιάς που ταξινομήθηκε
6	*Random + Use Probability Table* – με τυχαίο τρόπο αλλά αποκλείονται οι θέσεις που δεν έχουν πιθανότητα να βοηθήσουν στην εξέλιξη, χρήση βοηθητικού πίνακα
7	*Random + LastSortedBound + Use Probability Table* – Συνδυασμός των τριών προηγούμενων μεθόδων.

Πίνακας 5.2: 2ο γονίδιο – Επιλογή επόμενης γειτονιάς (Πηγή [03])

Τιμές	Περιγραφή
0	*Wrap* – αν ανιχνευθεί εγκλωβισμένος αριθμός στο αριστερό ή δεξί άκρο της ακολουθίας όλοι οι αριθμοί μετακινούνται μία θέση αριστερά ή δεξιά ανάλογα.
1	*Inflect* – σε αντίστοιχη περίπτωση δίνονται κατάλληλοι παράγοντες ορμής ώστε να δοθεί η δυνατότητα στον εγκλωβισμένο αριθμό να μετακινηθεί προς τη μέση της ακολουθίας
2	*No Action* – δεν εξετάζονται τα άκρα της ακολουθίας. Ο πρώτος και ο τελευταίος αριθμός μετακινούνται μόνο στα πλαίσια της ταξινόμησης της γειτονιάς τους.

Πίνακας 5.3: 3ο γονίδιο – Επιλογή ενέργειας στα άκρα της ακολουθίας (Πηγή [03])

Τιμές	Περιγραφή
0	*No Action* – Καμία ενέργεια
1	*Set Momentum* – αποδίδονται παράγοντες ορμής ανεξάρτητα με τους παράγοντες που είχαν ήδη οι αριθμοί
2	*Reset Momentum* – όλοι οι αριθμοί παίρνουν αδιάφορο παράγοντα ορμής

Πίνακας 5.4: 4ο γονίδιο – Επιλογή ενέργειας όταν η γειτονιά είναι ήδη ταξινομημένη (Πηγή [03])

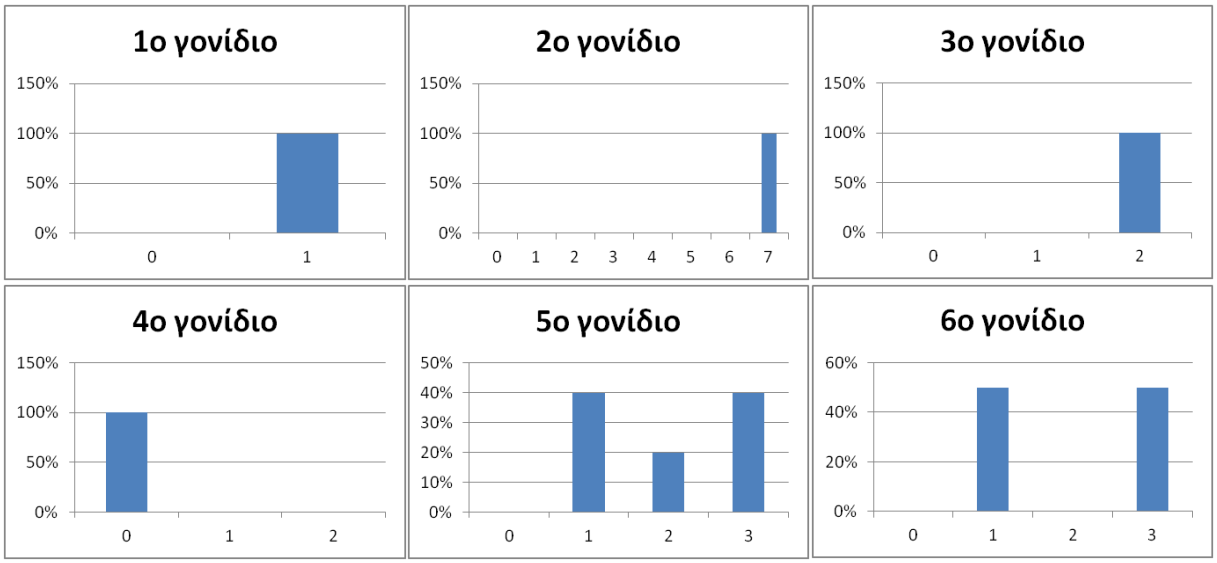
Τιμές	Περιγραφή
0	*No Action* – Καμία ενέργεια
1	*Reset Momentum* – οι δύο συνοριακοί αριθμοί της γειτονιάς παίρνουν αδιάφορο παράγοντα ορμής
2	*Set Momentum* – οι δύο συνοριακοί αριθμοί της γειτονιάς παίρνουν νέο παράγοντα ορμής ανάλογα με το πρωτόκολλο
3	*Scale Momentum* – Κλιμακωτή ανάθεση παραγόντων ορμής

Πίνακας 5.5: 5ο γονίδιο – Επιλογή ενέργειας στα σύνορα ταξινομημένης γειτονιάς (Πηγή [03])

Τιμές	Περιγραφή
0	*No Action* – Καμία ενέργεια
1	*Reset Momentum* – οι δύο συνοριακοί αριθμοί της γειτονιάς παίρνουν αδιάφορο παράγοντα ορμής
2	*Set Momentum* – οι δύο συνοριακοί αριθμοί της γειτονιάς παίρνουν νέο παράγοντα ορμής ανάλογα με το πρωτόκολλο
3	*Scale Momentum* – Κλιμακωτή ανάθεση παραγόντων ορμής

Πίνακας 5.6: 6ο γονίδιο – Επιλογή ενέργειας στα σύνορα μη ταξινομημένης γειτονιάς (Πηγή [03])

Κατά την εκτέλεση του γενετικού αλγορίθμου έγινε φανερό ότι στα 4 πρώτα γονίδια επικράτησε ο συνδυασμός {1,7,2,0}. Στο παρακάτω σχήμα φαίνονται οι συχνότητες εμφάνισης των επιμέρους τιμών κάθε γονιδίου σε ένα σύνολο από περίπου 20 εκτελέσεις του γενετικού αλγορίθμου στο αντίστοιχο Java Applet της εφαρμογής Ιστού Emerge-Sort @ WWW.



Εικόνα 5.13: Συχνότητες τιμών για κάθε επιμέρους γονίδιο.

Οι επικρατούσες τιμές των πρώτων 4 γονιδίων συμβολίζουν τον εξής συνδυασμό παραμέτρων:

- Πρωτόκολλο: A/D
- Επιλογή Γειτονιάς: Random + LastSortedBound + Use Probability Table
- Ενέργειες στα άκρα της ακολουθίας: No Action – δεν εξετάζονται τα άκρα της ακολουθίας
- Ενέργειες όταν η γειτονιά είναι ήδη ταξινομημένη: No Action – Καμία ενέργεια

Για τις υπόλοιπες θέσεις δεν καταγράφηκαν επικρατούντα γονίδια με βάση τις συγκεκριμένες ρυθμίσεις. Για τις 2 τελευταίες επιλογές παρατηρείται μία ελαφρά επικράτηση των επιλογών 1 και 3.

	Gene...	Individual	Fitness	Individual Description
	288	1 7 2 0 3 3	3,691399	ADProtocolP SemiRandomProb NoEnds NoAction ScaleFlags ScaleFlags
	289	1 7 2 0 3 1	3,8954463	ADProtocolP SemiRandomProb NoEnds NoAction ScaleFlags ResetFlags
	290	1 7 2 0 3 1	3,594407	ADProtocolP SemiRandomProb NoEnds NoAction ScaleFlags ResetFlags
	291	1 7 2 0 3 3	3,889386	ADProtocolP SemiRandomProb NoEnds NoAction ScaleFlags ScaleFlags
	292	1 7 2 0 3 1	3,6008787	ADProtocolP SemiRandomProb NoEnds NoAction ScaleFlags ResetFlags
	293	1 7 0 0 3 3	4,0516996	ADProtocolP SemiRandomProb Wrap NoAction ScaleFlags ScaleFlags
	294	1 7 0 0 3 1	3,629764	ADProtocolP SemiRandomProb Wrap NoAction ScaleFlags ResetFlags
	295	1 7 0 0 3 3	3,6036036	ADProtocolP SemiRandomProb Wrap NoAction ScaleFlags ScaleFlags
	296	1 7 0 0 3 3	3,7064493	ADProtocolP SemiRandomProb Wrap NoAction ScaleFlags ScaleFlags
	297	1 7 1 0 3 3	3,7791467	ADProtocolP SemiRandomProb Inflect NoAction ScaleFlags ScaleFlags
	298	1 7 2 0 3 1	3,6402023	ADProtocolP SemiRandomProb NoEnds NoAction ScaleFlags ResetFlags
	299	1 7 0 0 3 1	3,7863011	ADProtocolP SemiRandomProb Wrap NoAction ScaleFlags ResetFlags
		1 7 2 0 0 3	4,636069	ADProtocolP SemiRandomProb NoEnds NoAction NoAction ScaleFlags

Εικόνα 5.14: Αποτελέσματα εκτέλεσης γενετικού αλγορίθμου στην αυτόνομη εφαρμογή.

Τα παραπάνω αποτελέσματα είναι συμβατά με τα αποτελέσματα εκτέλεσης του γενετικού αλγορίθμου στην αυτόνομη εφαρμογή χρησιμοποιόντας ίδιες παραμέτρους, όπως φαίνεται και στο παραπάνω σχήμα, γεγονός που αποδεικνύει την ορθή μετάπτωση της λειτουργικότητας του γενετικού αλγορίθμου στην εφαρμογή Ιστού. Τα αποτελέσματα εκτέλεσης του γενετικού αλγορίθμου γενικά συμφωνούν και με τα πειραματικά δεδομένα εκτέλεσης του Emerge-Sort.

Κεφάλαιο 6
Συμπεράσματα

Στόχος της παρούσας μεταπτυχιακής διατριβής ήταν η ανάπτυξη της εφαρμογής Ιστού EmergeSort @ WWW ώστε η λειτουργικότητα της αυτόνομης εφαρμογής EmergeSortUI να γίνει διαθέσιμη με πιο ευέλικτο και αποδοτικό τρόπο. Η λειτουργικότητα της αυτόνομης εφαρμογής επαναχρησιμοποιήθηκε καθώς ήταν διαθέσιμη μέσω βιβλιοθηκών (jar – Java ARchive) οδηγώντας σε εξοικονόμηση στον χρόνο ανάπτυξης της εφαρμογής Ιστού.

Το κυριότερο πλεονέκτημα της εφαρμογής Ιστού έναντι της αυτόνομης εφαρμογής είναι η δια-λειτουργικότητά της καθώς κάθε χρήστης μπορεί να έχει πρόσβαση στην εφαρμογή ανεξάρτητα από το λειτουργικό σύστημα που χρησιμοποιεί ή τους πόρους του συστήματός του. Εξίσου σημαντικό είναι το γεγονός ότι δε χρειάζεται να εγκατασταθεί κάποιο κομμάτι της εφαρμογής στο σύστημα του χρήστη, εξοικονομώντας αρκετές ανθρωποώρες που θα απαιτούνταν για την εγκατάσταση της εφαρμογής και την αναβάθμισή της σε περίπτωση αλλαγών. Για να αποφευχθεί υπερφόρτωση του εξυπηρετητή που φιλοξενεί την εφαρμογή Ιστού, παρέχεται ταυτόχρονα η δυνατότητα εκτέλεσης των υπολογισμών στο περιβάλλον του χρήστη με την τεχνολογία των Java Applets. Η εφαρμογή βρίσκεται αποθηκευμένη στον εξυπηρετητή αλλά όποτε κάποιος χρήστης το επιθυμήσει, την εκτελεί τοπικά στο σύστημά του μέσω της εφαρμογής περιήγησης Ιστού που διαθέτει. Η ανάπτυξη της εφαρμογής Ιστού EmergeSort @ WWW ολοκληρώθηκε με επιτυχία, καθώς επετεύχθη η μετατροπή της λειτουργικότητας της αυτόνομης εφαρμογής EmergeSortUI. Αφού αναπτύχθηκε το κομμάτι της Εφαρμογής Ιστού για εκτέλεση των υπολογισμών στο περιβάλλον του εξυπηρετητή, χρησιμοποιήθηκε η τεχνολογία των Java Applets ώστε η υπολογιστική λογική του αλγορίθμου να είναι διαθέσιμη στους χρήστες μέσω του Ιστού αλλά με το βάρος των υπολογισμών να πέφτει στον υπολογιστή του εκάστοτε χρήστη.

Εκτός από τα παραπάνω πολύ σημαντικά πλεονεκτήματα, προέκυψαν αρκετά που δεν υπήρχαν στην αρχική στοχοθεσία της παρούσας διατριβής. Ένα από αυτά είναι ο κεντρικός έλεγχος της εφαρμογής. Αν ο διαχειριστής της για κάποιο λόγο αποφασίσει ότι προσωρινά πρέπει να μην είναι διαθέσιμη, μπορεί να το κάνει από τον εξυπηρετητή Ιστού για όλους τους

χρήστες. Επιπλέον υπάρχει δυνατότητα ανάλυσης των προσβάσεων των χρηστών ώστε ο διαχειριστής της εφαρμογής να γνωρίζει πόσοι και με ποιο τρόπο χρησιμοποιούν την εφαρμογή. Μάλιστα, τα αποτελέσματα από τις εκτελέσεις γενετικού αλγορίθμου όλων των χρηστών αποθηκεύονται κεντρικά δίνοντας έτσι τη δυνατότητα συνολικής μελέτης τους.

Όσον αφορά τα πειράματα που εκτελέστηκαν χρησιμοποιώντας την εφαρμογή EmergeSort @ WWW, αυτά κατέδειξαν κυρίως:

- Το γεγονός ότι η λειτουργικότητα του αλγορίθμου Emerge-Sort και του γενετικού αλγορίθμου παρέχεται με ισοδύναμο, ως προς την αυτόνομη εφαρμογή EmergeSortUI, τρόπο, τόσο στο μοντέλο εκτέλεσης υπολογισμών στον εξυπηρετητή όσο και στο μοντέλο εκτέλεσης υπολογισμών στο περιβάλλον του χρήστη.

- Ότι το τρέχον περιβάλλον εκτέλεσης με περιορισμένο φόρτο χρήσης λειτουργεί αποδοτικότερα για εκτέλεση υπολογισμών στον εξυπηρετητή.

- Ότι ο μέσος αριθμός κινήσεων και γύρων εκτέλεσης του αλγορίθμου Emerge-Sort επηρεάζεται θετικά από τη χρήση των επιλογών Last Sorted Bound και Use Probability Table.

- Ότι ο μέσος αριθμός κινήσεων και γύρων εκτέλεσης του αλγορίθμου Emerge-Sort επηρεάζεται θετικά από αύξηση του μεγέθους της γειτονιάς

- Ότι το L/R πρωτόκολλο έχει μικρά ποσοστά επιτυχούς ταξινόμησης.

Παρά την επιτυχία ως προς τον στόχο της παρούσας διατριβής υπάρχουν αρκετές βελτιώσεις που θα μπορούσαν να γίνουν στο μέλλον χρησιμοποιώντας ως βάση την υπάρχουσα εφαρμογή. Στη συνέχεια παρατίθενται ορισμένες προτάσεις για μελλοντική έρευνα:

- Έλεγχος πρόσβασης: Μέχρι στιγμής δεν υπάρχει κανένας απολύτως έλεγχος πρόσβασης. Κάθε χρήστης που γνωρίζει το url της εφαρμογής, μπορεί να τη χρησιμοποιήσει. Θα μπορούσε να χρησιμοποιηθεί μία βάση δεδομένων, η οποία θα επέτρεπε τη διαχείριση χρηστών πιθανώς και με διαφορετικά προνόμια (π.χ. κάποιοι χρήστες θα μπορούσαν να χρησιμοποιούν την εκτέλεση Java Servlets και κάποιοι μόνο των Java Applets για να

αποφευχθεί συνωστισμός). Με αυτό τον τρόπο θα μπορούσαν να καταγράφονται και στατιστικά χρήσης της εφαρμογής.

- Εξαγωγή αποτελεσμάτων: Ο τρόπος που εξάγονται τα αποτελέσματα θα μπορούσε να βελτιωθεί, είτε δίνοντας τη δυνατότητα εξαγωγής τους σε άλλες μορφές αρχείων (.xls, .xml κ.α.) είτε παρέχοντας και γραφήματα μέσω σχετικών βιβλιοθηκών.

- Εκτέλεση επιπλέον πειραμάτων: Οι συνδυασμοί διαφορετικών παραμέτρων του αλγορίθμου Emerge-Sort είναι πολυάριθμοι. Στα πλαίσια της παρούσας διατριβής αναλύθηκε η επίδραση μόνο ορισμένων από αυτούς. Σε επόμενο στάδιο μπορούν να εκτελεστούν λεπτομερέστερες αναλύσεις για περισσότερους συνδυασμούς μεταβλητών με χρήση της υπάρχουσας υποδομής.

- Παράμετροι στο γενετικό αλγόριθμο: Στην τρέχουσα έκδοση της εφαρμογής, η εισαγωγή των παραμέτρων του γενετικού αλγορίθμου γίνεται μέσω ενός αρχείου παραμέτρων. Είναι δυνατό να αναπτυχθεί μία φόρμα (και η επέκταση του σχετικού ενός Applet GUI) που θα επιτρέπει την ευκολότερη εισαγωγή διαφορετικών παραμέτρων από το χρήστη.

- Εκτέλεση σε Υπολογιστικό Πλέγμα (GRID): Για την πιο αποδοτική εκτέλεση του αλγορίθμου Emerge-Sort και του γενετικού αλγορίθμου σε μεγαλύτερα μεγέθη ακολουθιών και γενεών αντίστοιχα θα μπορούσε να χρησιμοποιηθεί η εφαρμογή κάποιο υπολογιστικό πλέγμα [18]. Με τις κατάλληλες μετατροπές, οι βιβλιοθήκες (jar αρχεία) που παρέχουν την απαραίτητη λειτουργικότητα, μπορούν να εκτελεστούν σε Υπολογιστικό Πλέγμα πετυχαίνοντας καλύτερα αποτελέσματα.

Βιβλιογραφία

[01] E. Bonabeau, M. Dorigo and G. Theraulaz. '*Swarm Intelligence: 'From Natural to Artificial Systems'*, Santa Fe Institute Studies on the Sciences of Complexity. Oxford University Press, 1999, σελ. 7, 22-25.

[02] Kalles D., Kaporis, Alexis. *'Emerge-Sort: Converging to Ordered Sequences by Simple Local Operators'* (2008), http://arxiv.org/abs/0812.1126, σελ. 1

[03] Βασιλική Μπερουκλή. Νοημοσύνη Σμήνους και Γενετικός Προγραμματισμός. ΕΛΛΗΝΙΚΟ ΑΝΟΙΧΤΟ ΠΑΝΕΠΙΣΤΗΜΙΟ, Πτυχιακή Εργασία HOU-CS-UGP-2010-11

[04] D.E. Knuth: *The Art of Computer Programming, Vol. 3 - Sorting and Searching*. Addison-Wesley (1973)

[05] Steven J. Ross. '*The Spreadsort High-performance General-case Sorting Algorithm*'. Parallel and Distributed Processing Techniques and Applications, Volume 3, pp.1100–1106. Las Vegas Nevada. 2002

[06] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *'Introduction to Algorithms, Second Edition'*. MIT Press and McGraw-Hill, 2001

[07] T. de Wolf, T. Holvoet, '*Emergence versus self-organization: different concepts but promising when combined'*, vol. 3464 of Lecture Notes in Computer Science, Springer, http://www.springerlink.com/content/p7k83kc3fkj8e42n/fulltext.pdf, 2005, σελ. 7

[08] Bender, M. A., Farach-Colton, M., and Mosteiro M. (2006). "*Insertion Sort is O(n log n)*", Theory of Computing Systems 39 (3): 391

[09] Roger Wattenhofer, Fabian Kuhn, '*Principles of Distributed Computing*', http://www.dcg.ethz.ch/lectures/fs08/distcomp/lecture/chapter9.pdf , 2008

[10] X.-S. Yang; S. Deb. "*Cuckoo search via Lévy flights*". World Congress on Nature & Biologically Inspired Computing (NaBIC 2009).

[11] Yang X. S.: '*Nature-Inspired Metaheuristic Algorithms*'. Luniver Press, 2008

[12] Shah-Hosseini, Hamed. "The intelligent water drops algorithm: a nature-inspired swarm-based optimization algorithm". International Journal of Bio-Inspired Computation, 2009

[13] Pablo Rabanal, Ismael Rodríguez and Fernando Rubio :Finding Minimum Spanning/Distances Trees by Using River Formation Dynamics, Springer, 2008.

[14] Martens, D.; Baesens, B.; Fawcett, T. (2011). "Editorial Survey: Swarm Intelligence for Data Mining". Machine Learning 82 (1): 1–42, 2011.

[15] Ιστοσελίδα εκμάθησης της τεχνολογίας των Java Servlets: http://java.sun.com/j2ee/tutorial/1_3-fcs/doc/Servlets.html

[16] Η Ιστοσελίδα του Project ecj: http://cs.gmu.edu/~eclab/projects/ecj/

[17] Ιστοσελίδα σχετική με τα Java Applets : http://java.sun.com/applets/

[18] I. Foster, C. Kesselman. Computational Grids

[19] Η Ιστοσελίδα W3Schools: http://www.w3schools.com/html/

[20] Ιστοσελίδα σχετική με την τεχνολογία των Java Servlets: http://www.oracle.com/technetwork/java/javaee/servlet/index.html

[21] Η Ιστοσελίδα του εξυπηρετητή Apache Tomcat : http://tomcat.apache.org/

[22] Η Ιστοσελίδα του περιβάλλοντος ανάπτυξης eclipse: http://www.eclipse.org/

[23] Η Ιστοσελίδα του eclipse plugin Window Builder: http://code.google.com/javadevtools/wbpro/

[24] Ιστοσελίδα με οδηγίες παραμετροποίησης του περιβάλλοντος ανάπτυξης: http://www.coreservlets.com/Apache-Tomcat-Tutorial/tomcat-7-with-eclipse.html

[25] Ιστοσελίδα για τη Διαδικασία Ψηφιακής Υπογραφής εκτελέσιμου αρχείου jar : http://java.sun.com/developer/onlineTraining/Programming/JDCBook/signed.html

[26] Η βιβλιοθήκη Microsoft .Net Framework Version 2.0: http://www.microsoft.com/downloads/details.aspx?familyid=0856eacb-4362-4b0d-8edd- aab15c5e04f5&displaylang=en

Παράρτημα Α
Εγκατάσταση και Παραμετροποίηση του Περιβάλλοντος Ανάπτυξης

Για την ανάπτυξη της εφαρμογής Ιστού επιλέχθηκε να χρησιμοποιηθεί το περιβάλλον ανάπτυξης του eclipse σε συνδυασμό με τον apache tomcat για την ευκολότερη ανάπτυξη, εγκατάσταση και δοκιμή των Java Servlets και Java Applets της διαδικτυακής εφαρμογής που αναπτύχθηκε. Η χρήση της συνδυασμένης αυτής λύσης βοήθησε στην ταχύτερη αντιμετώπιση σφαλμάτων και μείωσε σημαντικά το χρόνο ανάπτυξης. Επιπλέον, για την ταχύτερη υλοποίηση του γραφικού περιβάλλοντος των Java Applets χρησιμοποιήθηκε το WindowBuilder plug-in για το eclipse. Στο παράρτημα αυτό περιγράφονται τα βήματα που απαιτήθηκαν για την ορθή εγκατάσταση και παραμετροποίηση του συνολικού περιβάλλοντος ανάπτυξης. Ως βάση χρησιμοποιήθηκε ένας διαδικτυακός οδηγός (tutorial) εγκατάστασης και παραμετροποίησης του Tomcat έκδοσης 7 για χρήση με το eclipse (http://www.coreservlets.com/Apache-Tomcat-Tutorial/tomcat-7-with-eclipse.html).

A.1 Εγκατάσταση της Java

Από το σχετικό σύνδεσμο (http://www.oracle.com/technetwork/java/javase/downloads/jdk-6u29-download-513648.html) μεταφορτώθηκε και εγκαταστάθηκε το Java Development Kit έκδοσης 1.6.0_29. Οι Servlet 3.0 containers (όπως ο tomcat έκδοσης 7 που χρησιμοποιήθηκε) απαιτούν Java τουλάχιστον έκδοσης 6 και δε λειτουργούν με Java 5.

A.2 Εγκατάσταση του Tomcat

Από το σχετικό σύνδεσμο (http://www.coreservlets.com/Apache-Tomcat-Tutorial/tomcat7-files/tomcat-7.0.8-preconfigured.zip) μεταφορτώθηκε ο Tomcat. Αποσυμπιέστηκε στο δίσκο C δημιουργώντας το φάκελο C:\apache-tomcat-7.0.8. Το συμπιεσμένο αρχείο που μεταφορτώθηκε είναι μία «προ-ρυθμισμένη» έκδοση του Tomcat με αλλαγές στην παραμετροποίηση (configuration) ώστε να χρησιμοποιηθεί ευκολότερα. Θα μπορούσε να χρησιμοποιηθεί και κάποια από τις επίσημες εκδόσεις από τον ιστοχώρο της Apache (http://tomcat.apache.org/download-70.cgi). Οι ρυθμίσεις του Tomcat επέβαλαν να ξεκινάει στην πόρτα 80. Επειδή διαπιστώθηκε ότι σε αρκετά συστήματα η πόρτα αυτή είναι δεσμευμένη από το λειτουργικό (πχ Windows 7) ή κάποια διεργασία (για παράδειγμα το πρόγραμμα Skype ή ο SQL Server) η πόρτα έπρεπε να αλλάξει σε 8080. Για το σκοπό αυτό άλλαξε η σχετική γραμμή στο αρχείο C:\apache-tomcat-7.0.8\conf\server.xml από:

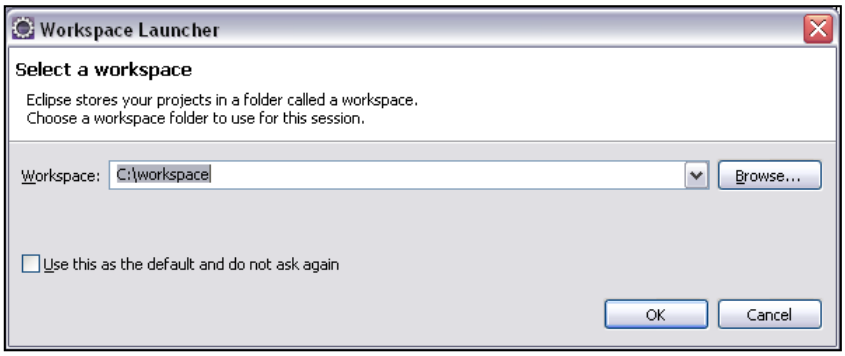
```
<Connector port="80" protocol="HTTP/1.1"
```

σε

```
<Connector port="8080" protocol="HTTP/1.1".
```

A.3 Εγκατάσταση του Eclipse

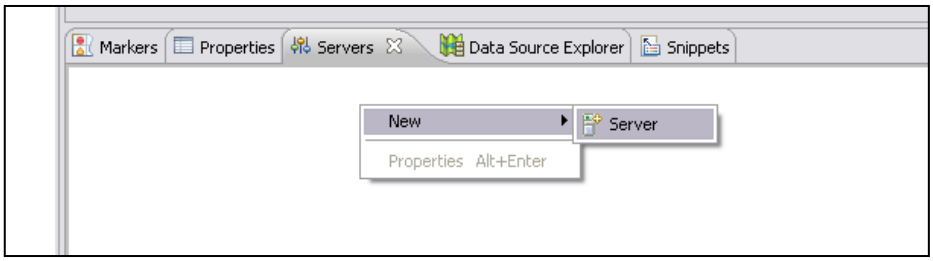
Από το σχετικό σύνδεσμο (http://www.eclipse.org/downloads/packages/eclipse-ide-java-ee-developers/heliossr2) μεταφορτώθηκε και εγκαταστάθηκε το eclipse jee και αποσυμπιέστηκε στο φάκελο C:\Program Files\eclipse\. Το eclipse εκτελείται για πρώτη φορά κάνοντας διπλό κλικ στο eclipse.exe και επιλέγοντας workbench. Κάθε φορά που εκτελείται το eclipse ζητείται να οριστεί ο χώρος εργασίας (workspace), ο χώρος δηλαδή που θα αποθηκευτούν τα projects. Προτείνεται να χρησιμοποιηθεί ο φάκελος C:\workspace για να αποφευχθούν ασυμβατότητες.



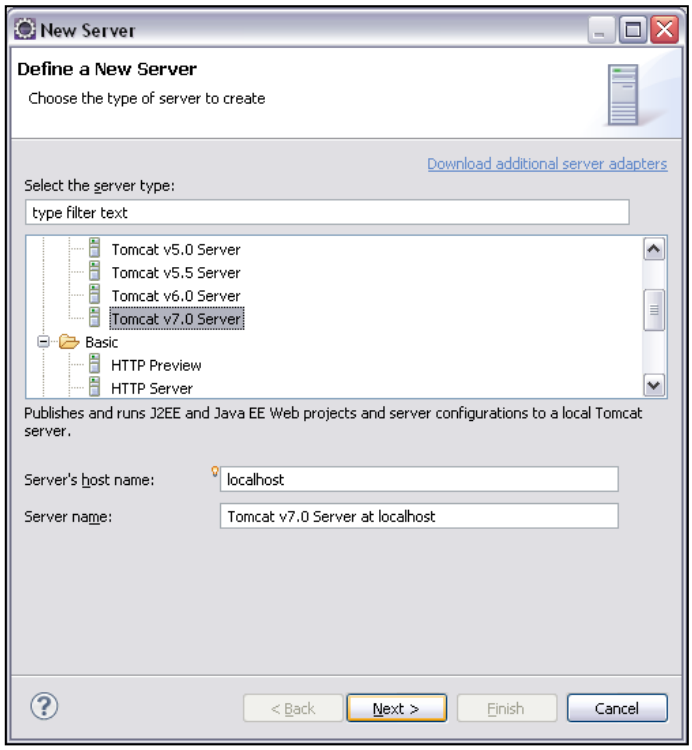
Εικόνα Α.1: Επιλογή Workspace στο eclipse.

Α.4 Συνεργασία eclipse-Tomcat

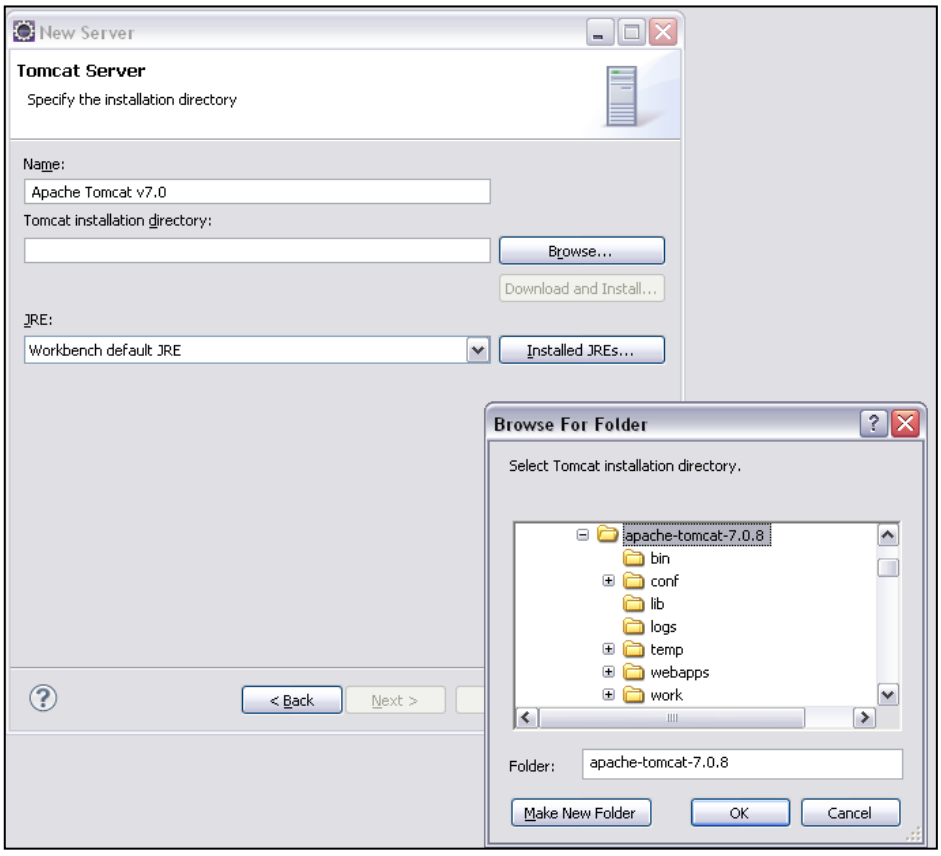
Για να εκκινήσει ο tomcat μέσα από το eclipse πρέπει την πρώτη φορά να προστεθεί στους servers που διαχειρίζεται το eclipse. Στο tab Servers στο κέντρο και κάτω επιλέγεται : R-click, New, Server, Apache, Tomcat v7.0, επιλογή του καταλόγου εγκατάστασης του Tomcat 7 (e.g., C:\apache-tomcat-7.0.8), OK. Τα βήματα φαίνονται και στις παρακάτω εικόνες.



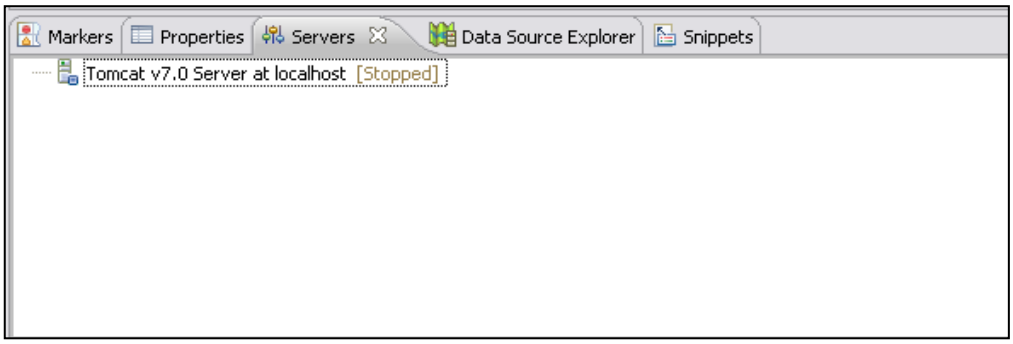
Εικόνα Α.2: Ρύθμιση νέου Server στο eclipse (1/4).



Εικόνα Α.3: Ρύθμιση νέου Server στο eclipse (2/4).



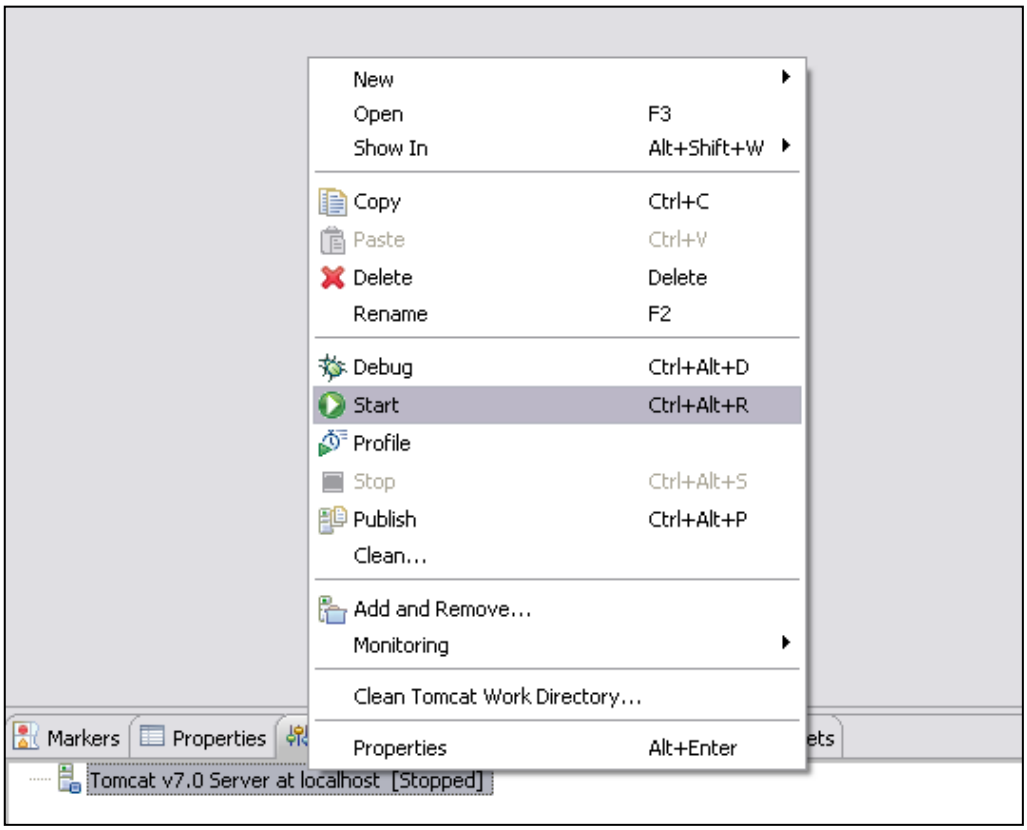
Εικόνα Α.4: Ρύθμιση νέου Server στο eclipse (3/4).



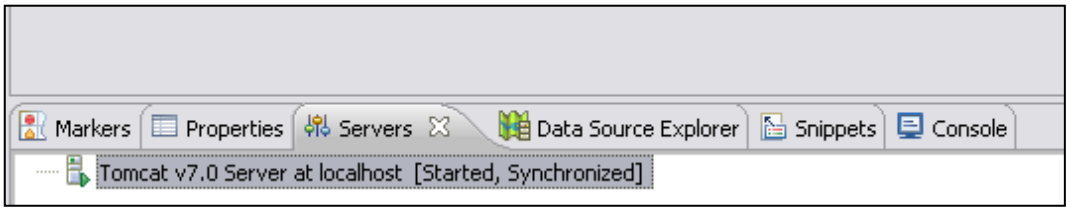
Εικόνα Α.5: Ρύθμιση νέου Server στο eclipse (4/4).

Α.5 Εκκίνηση του Tomcat

Στο Servers tab εκτελείται δεξί κλικ στον Tomcat v7.0 και επιλέγεται "Start".



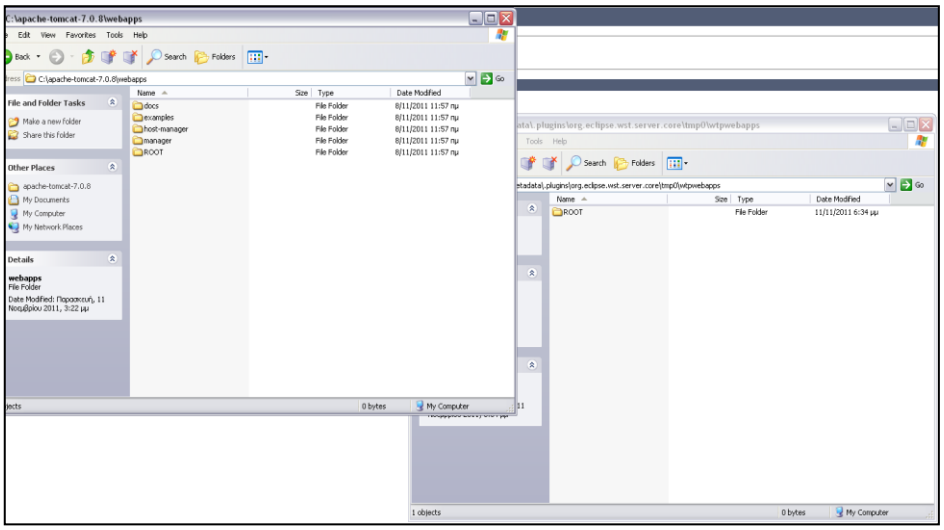
Εικόνα Α.6: Εκκίνηση του tomcat μέσα από το eclipse.



Εικόνα Α.7: Η κατάσταση (status) του tomcat μετά την εκκίνηση.

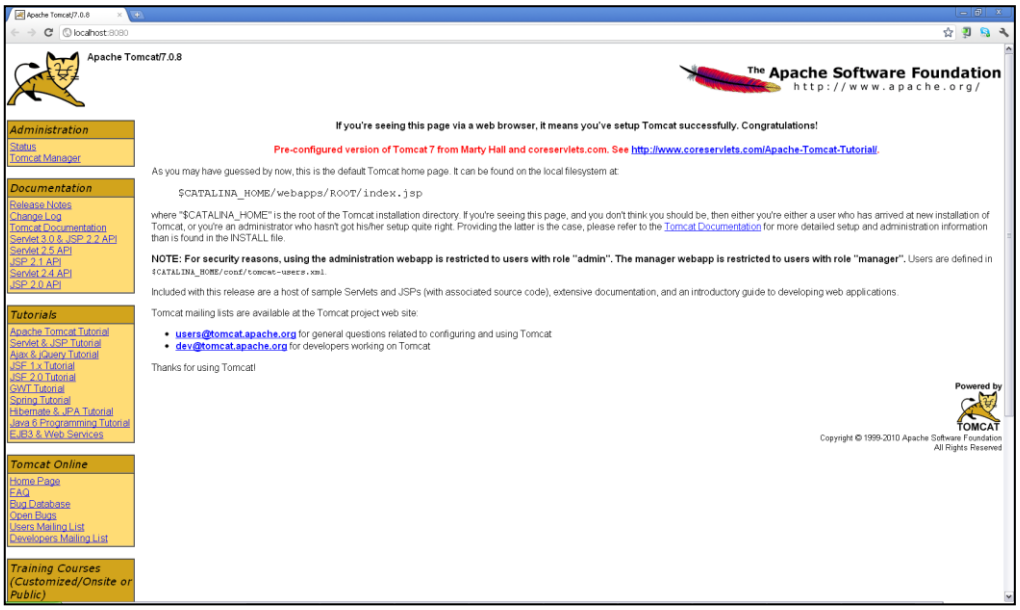
Στη συνέχεια πρέπει να γίνει copy από το φάκελο C:\apache-tomcat-7.0.8\webapps, ο φάκελος ROOT και να γίνει paste στο φάκελο

C:\workspace\.metadata\.plugins\org.eclipse.wst.server.core\tmp0\wtpwebapps.



Εικόνα Α.8: Μετακίνηση των αρχείων της κεντρικής σελίδας του Tomcat.

Στη συνέχεια εκτελείται δεξί κλικ->Restart στον Server. Για να ελεγχθεί αν ο Server έχει εκκινήσει κανονικά ο χρήστης μπορεί να επισκεφτεί στη διεύθυνση http://localhost:8080/ (8080 γιατί άλλαξε η πόρτα στο server configuration) με οποιονδήποτε browser. Θα πρέπει να εμφανιστεί η παρακάτω οθόνη.



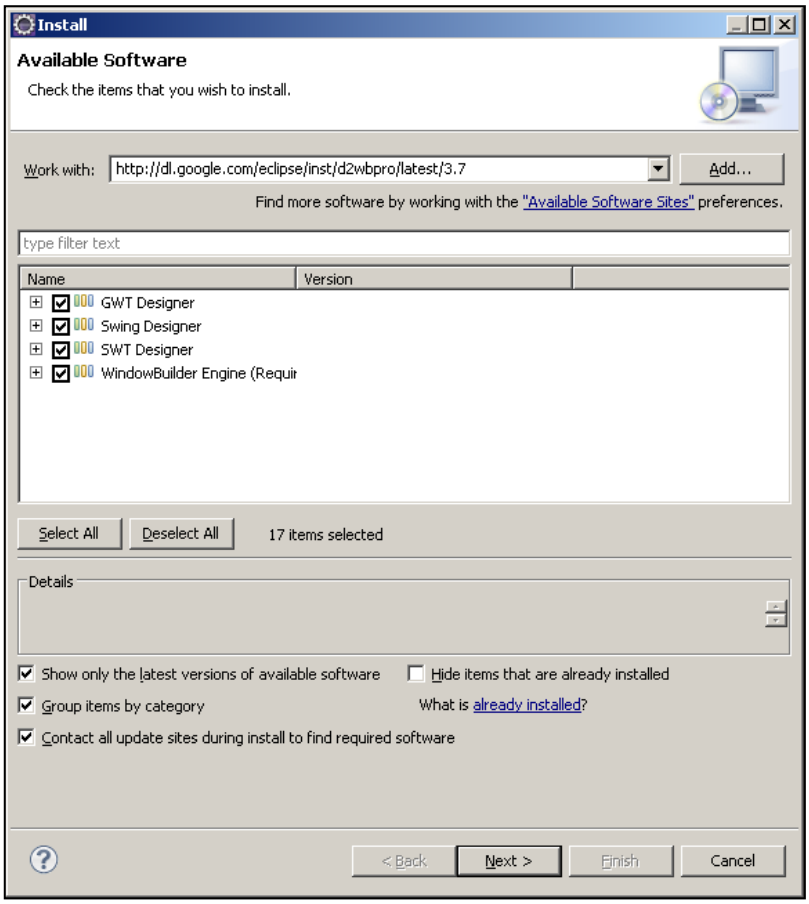
Εικόνα Α.9: Ρύθμιση νέου Server στο eclipse (4/4).

Α.6 Εγκατάσταση του WindowBuilder plug-in

Για την ευκολότερη υλοποίηση του GUI του Applet εγκαταστάθηκε στο eclipse το Window Builder plugin. Για την εγκατάσταση ακολουθήθηκαν οι οδηγίες από το σύνδεσμο http://code.google.com/javadevtools/wbpro/installation/updatesite_3.7.html επιλέγοντας ως σύνδεσμο του νέου software το: http://dl.google.com/eclipse/inst/d2wbpro/latest/3.7.

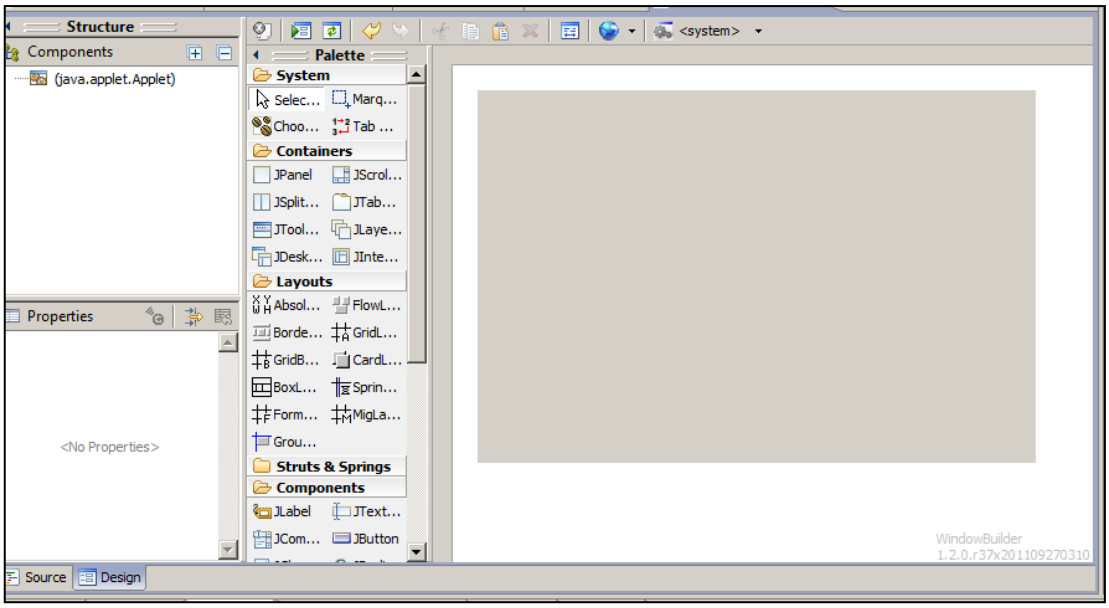
Πιο συγκεκριμένα:

- επιλέχθηκε στο eclipse Help->Install New Software
- στο παράθυρο που εμφανίστηκε, στο πεδίο Work With έγινε paste ο σύνδεσμος http://dl.google.com/eclipse/inst/d2wbpro/latest/3.7
- στο ακριβώς κάτω παράθυρο επιλέχθηκαν όλα τα Group που εμφανίστηκαν (Select All) (βλ. παρακάτω εικόνα) και στη συνέχεια Next.
- σε μηνυμα περί αποδοχής της άδειας επιλέχθηκε Accept και τελικά Finish.
- όταν ζητήθηκε, έγινε restart στο eclipse για να περιληφθούν τα νέα στοιχεία λογισμικού.



Εικόνα Α.10: Εγκατάσταση του Window Builder plugin.

Το plugin παρέχει αυτόματη παραγωγή κώδικα Java μετά από τη σχεδίαση του GUI με γραφικό τρόπο (Design View).



Εικόνα Α.11: Το Design View του WindowBuilder plug-in.

Παράρτημα Β

Περιγραφή των Τμημάτων Κώδικα της Εφαρμογής

Σε αυτό το παράρτημα περιγράφονται εν συντομία τα διάφορα κομμάτια του κώδικα της εφαρμογής. Παρουσιάζονται συνοπτικά τα eclipse projects που δημιουργήθηκαν για τους σκοπούς της εργασίας καθώς και οι έτοιμες βιβλιοθήκες που χρησιμοποιήθηκαν. Αναλύεται επίσης ο τρόπος με τον οποίο μπορεί να εισαχθούν αυτά τα project στο eclipse αν υπάρξει στο μέλλον ανάγκη επέκτασης του κώδικα. Τέλος περιγράφεται η διαδικασία εγκατάστασης της εφαρμογής Ιστού σε έναν εξυπηρετητή Apache Tomcat.

Β.1 Το Τελικό Παραδοτέο της Εφαρμογής Ιστού

Για την ανάπτυξη της εφαρμογής δημιουργήθηκαν ξεχωριστά eclipse projects για κάθε κομμάτι του κώδικα. Επιπλέον τα περισσότερα από αυτά μετατράπηκαν σε αρχεία jar (Java Archive) ώστε να χρησιμοποιηθούν ως βιβλιοθήκες όπου απαιτήθηκε. Στις επόμενες παραγράφους περιγράφονται τα περιεχόμενα κάθε φακέλου του τελικού παραδοτέου.

Β.1.1 Ο Φάκελος oldSources

Για την ανάπτυξη της εφαρμογής δημιουργήθηκαν ξεχωριστά eclipse projects για κάθε κομμάτι του κώδικα. Επιπλέον τα περισσότερα από αυτά μετατράπηκαν σε αρχεία jar (Java Archive) ώστε να χρησιμοποιηθούν ως βιβλιοθήκες όπου απαιτήθηκε. Στις επόμενες παραγράφους περιγράφονται τα περιεχόμενα κάθε φακέλου του τελικού παραδοτέου.

Β.1.1 Ο Φάκελος oldSources

Ο φάκελος αυτός περιλαμβάνει τα αρχεία πηγαίου κώδικα (source files) από προηγούμενη εργασία τα οποία χρησιμοποιούνται για την επίτευξη λειτουργικότητας του γενετικού αλγορίθμου. Κάποια από αυτά τα αρχεία είναι τα πηγαία αρχεία της βιβλιοθήκης ECJ ενώ κάποια άλλα χρησιμοποιήθηκαν για τη χρήση της βιβλιοθήκης.

B.1.2 Ο Φάκελος eclipseProjects

Ο φάκελος αυτός περιλαμβάνει τα ξεχωριστά projects του eclipse για κάθε κομμάτι της εφαρμογής. Αναλυτικά:

- project ecj: Το project αυτό περιλαμβάνει τα πηγαία αρχεία της βιβλιοθήκης ecj. Το project αυτό εξήχθη ως jar αρχείο (ecj.jar).

- project ECJRuntime: Το project αυτό περιλαμβάνει αρχεία από προηγούμενη υλοποίηση τα οποία «καλούν» τη βιβλιοθήκη ecj με τρόπο συμβατό με το πρόβλημα του emergesort. Ως dependencies απαιτεί το ecj.jar και το jar με τη λειτουργικότητα του EmergeSort (EmergeSorter.jar) που ήταν διαθέσιμο από προηγούμενη εργασία. Το project αυτό εξήχθη ως jar αρχείο (ECJRuntime.jar).

- project emergesortapplet: Το project αυτό περιλαμβάνει όλα τα αρχεία για το applet που εκτελεί τη λειτουργικότητα του Emerge-Sort. . Ως dependency απαιτεί το jar με τη λειτουργικότητα του EmergeSort (EmergeSorter.jar) που ήταν διαθέσιμο από προηγούμενη εργασία. Το project αυτό εξήχθη ως jar αρχείο (emergesortapplet.jar).

- project ecjapplet: Το project αυτό αυτό περιλαμβάνει όλα τα αρχεία για το applet που εκτελεί τη λειτουργικότητα του γενετικού αλγορίθμου. Ως dependency απαιτεί το jar με τη λειτουργικότητα του EmergeSort (EmergeSorter.jar) που ήταν διαθέσιμο από προηγούμενη εργασία και τα jars ecj.jar, ECJRuntime.jar, EmergeSorter.jar. Το project αυτό εξήχθη ως jar αρχείο (ecjapplet.jar).

- project emergesort: Το project αυτό είναι η συνολική εφαρμογή ιστού. Περιλαμβάνει τις HTML σελίδες της εφαρμογής, τα Servlet που εκτελούν το γενετικό αλγόριθμο και τον αλγόριθμο emergesort και τα jar αρχεία που μπορούν να μεταφορτωθούν από τους χρήστες της. Το project αυτό εξήχθη σε war αρχείο (emergesort.war) ώστε να γίνεται εύκολα deploy στον apache tomcat.

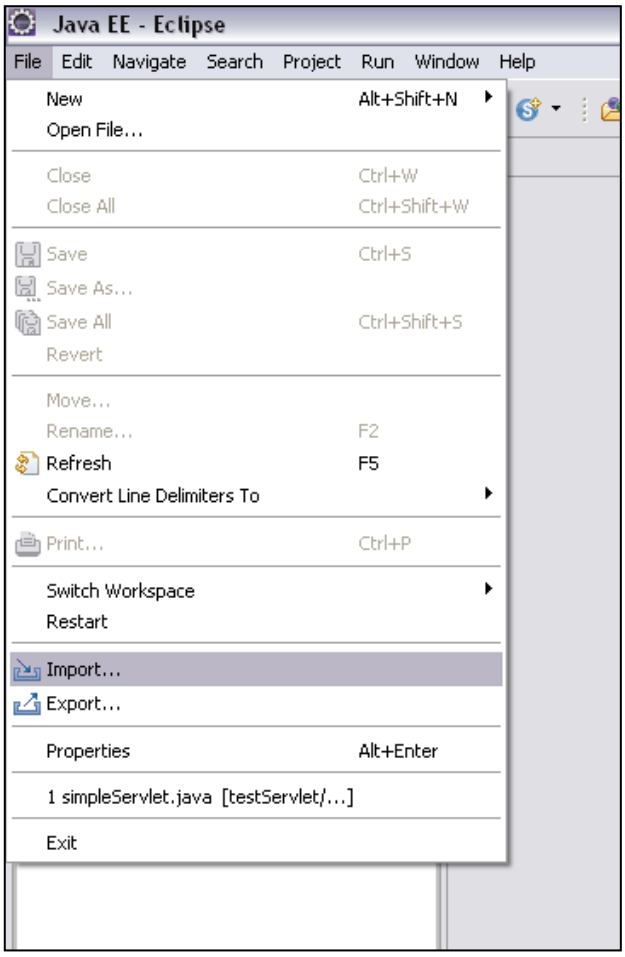
B.1.3 Ο Φάκελος finalJars

Ο φάκελος αυτός περιλαμβάνει τα jar files που χρησιμοποιούνται από την εφαρμογή Ιστού. Από αυτά τα **ecj.jar, ECJRuntime.jar, ecjapplet.jar, emergesortapplet.jar** δημιουργήθηκαν από τα αντίστοιχα eclipse projects. Το **EmergeSorter.jar** χρησιμοποιήθηκε αυτούσιο από προηγούμενη εργασία ενώ τα iText**-2.1.5.jar, jcommon-1.0.17.jar, jfreechart-1.0.14.jar** είναι απλά dependencies που απαιτούνται από τη βιβλιοθήκη ecj.

Β.2 Εγκατάσταση των Eclipse Projects στο Περιβάλλον Ανάπτυξης

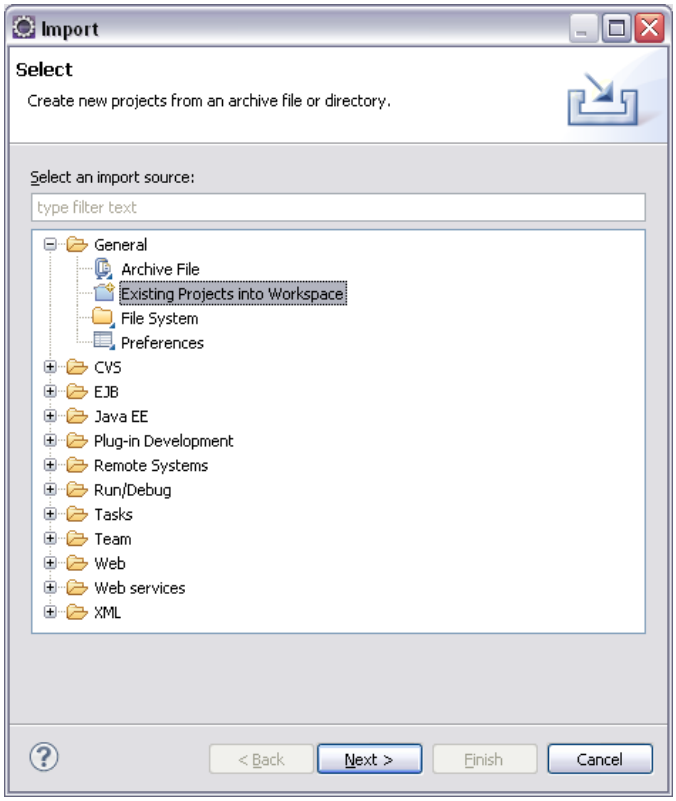
Για να εγκατασταθούν τα eclipse projects στο περιβάλλον ανάπτυξης που περιγράφηκε στο παράρτημα Α ακολουθούνται τα παρακάτω βήματα.

Στο eclipse επιλέγεται File->Import.



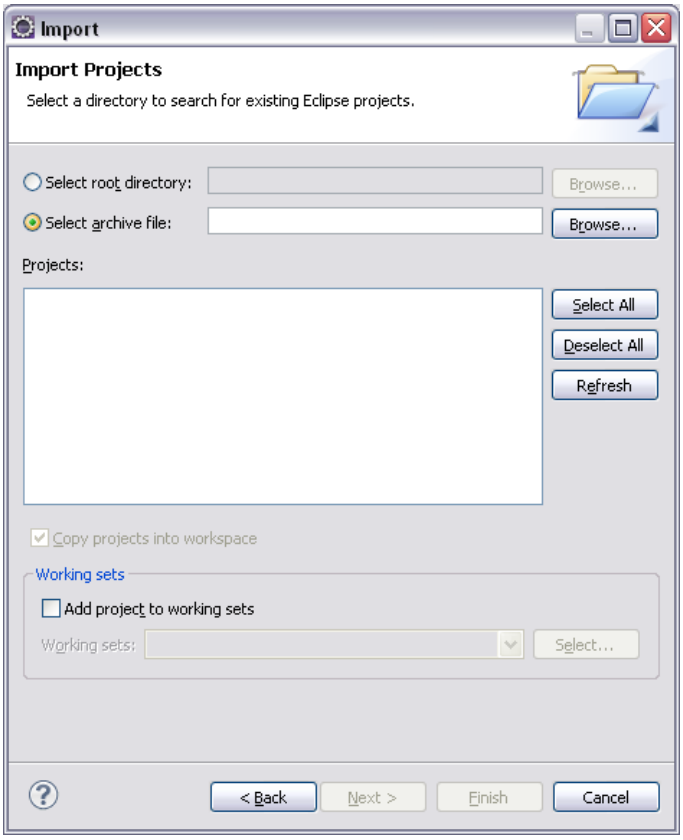
Εικόνα Β.1: Εγκατάσταση eclipse project (1/7).

Στη συνέχεια επιλέγεται “Existing Projects into Workspace” και Next.



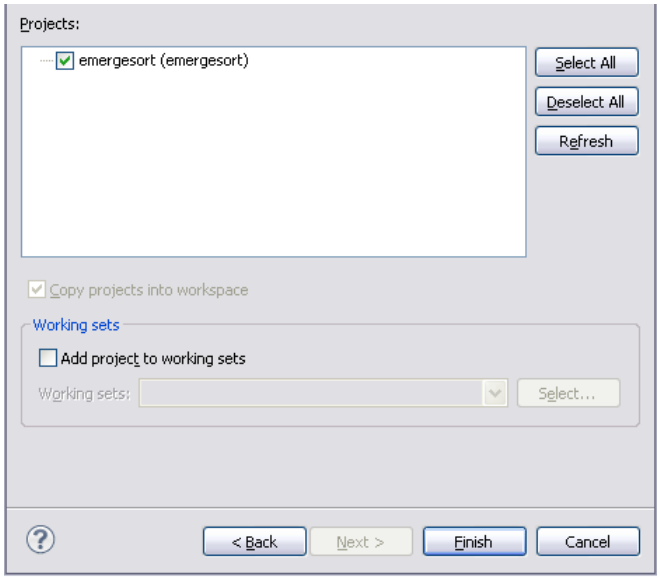
Εικόνα Β.2: Εγκατάσταση eclipse project (2/7).

Κατόπιν επιλέγεται Archive file και ανοίγει το project zip (π.χ. το emergesort.zip).



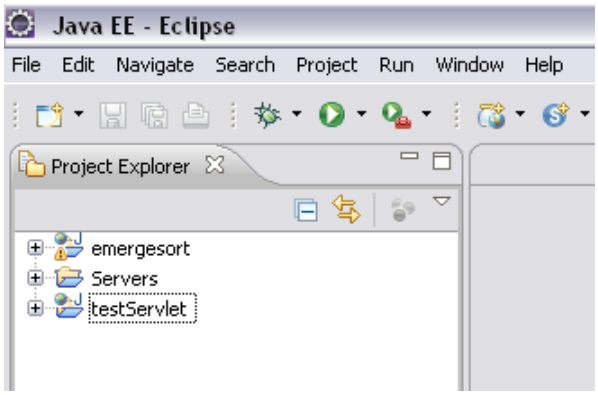
Εικόνα Β.3: Εγκατάσταση eclipse project (3/7).

Τέλος επιλέγεται Finish.



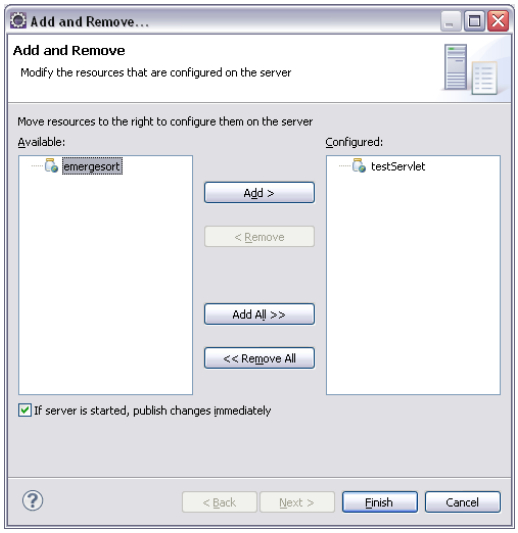
Εικόνα Β.4: Εγκατάσταση eclipse project (4/7).

Το Project είναι πλέον ορατό στο eclipse.



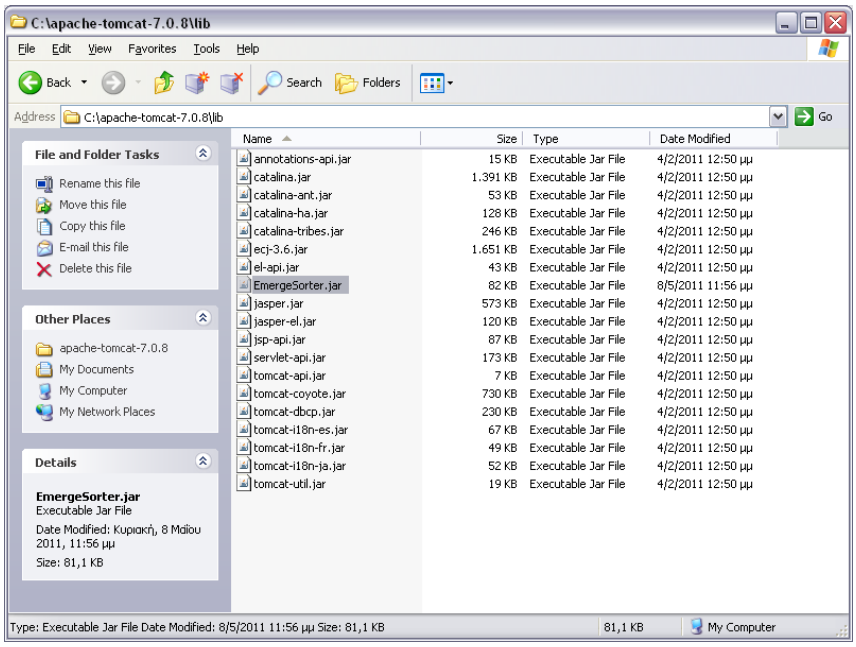
Εικόνα Β.5: Εγκατάσταση eclipse project (5/7).

Αν πρόκειται για το project archive της συνολικής εφαρμογής (emergesort.zip), το project γίνεται add στον server.



Εικόνα Β.6: Εγκατάσταση eclipse project (6/7).

Επιπλέον, τα jar που παρέχουν την απαιτούμενη λειτουργικότητα πρέπει να γίνουν copy στο φάκελο lib του Tomcat (C:\apache-tomcat-7.0.8\lib).



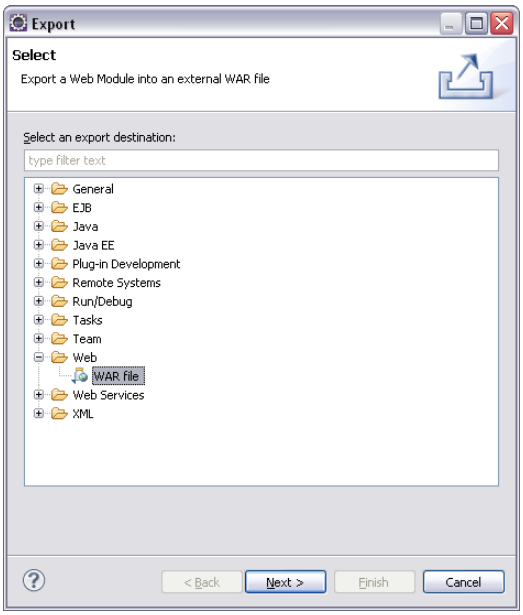
Εικόνα Β.7: Εγκατάσταση eclipse project (7/7).

Θα πρέπει μετά από τα παραπάνω βήματα η εφαρμογή να είναι διαθέσιμη από τη διεύθυνση http://localhost:8080/emergesort/.

Β.3 Εγκατάσταση της Εφαρμογής στον Apache Tomcat

Η εφαρμογή Ιστού μπορεί να εγκατασταθεί πολύ εύκολα σε οποιονδήποτε εξυπηρετητή Tomcat έκδοσης 7 με τη μορφή war file (emergesort.war). Το war file περιέχει όλα τα αρχεία που χρειάζονται για να εκτελεστεί σωστά η διαδικτυακή εφαρμογή.

Για να κατασκευαστεί το war file emergesort.war χρησιμοποιήθηκε το eclipse menu File->Export->Web->War File.



Εικόνα Β.8: Εξαγωγή war file.

Για να εγκατασταθεί το war file σε έναν εξυπηρετητή Tomcat:

- Γίνεται copy το emergesort.war στο φάκελο TOMCAT_DIR\webapps (όπου TOMCAT_DIR ο ριζικός φάκελος του Tomcat π.χ. C:\apache-tomcat-7.0.8).
- Γίνονται copy όλα τα απαιτούμενα jar στο φάκελο lib του Tomcat.
- Γίνεται start ο Tomcat TOMCAT_DIR\bin\startup.bat .

Μετά την εκκίνηση θα πρέπει να έχει δημιουργηθεί ένας φάκελος με το όνομα emergesort στον φάκελο webapps και η εφαρμογή να είναι διαθέσιμη.

Παράρτημα Γ
Ο κώδικας των Java Servlets και Java Applets

Σε αυτό το παράρτημα παρατίθεται ο κώδικας Java των Java Servlets και Java Applets που εκκινούν την εκτέλεση του αλγορίθμου Emerge-Sort ή του γενετικού αλγορίθμου.

Γ.1 Ο κώδικας του Java Servlet runEmergesort

```
package emergesort;

/*java and servlet imports*/
import java.io.IOException;
import java.io.PrintWriter;
```

```
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import emergesorter.EmergeOptions;
import emergesorter.Enm;
import emergesorter.Population;
import emergesorter.PopulationGroup;
import emergesorter.PopulationLog;
import emergesorter.SortCycleEvent;
import emergesorter.SortCycleListener;
import emergesorter.Sorter;
import emergesorter.Stats;

/**
 * Servlet implementation class runEmergesort
 * This class represents the Servlet that runs the emergesort simulation
 */
@WebServlet("/runEmergesort")
public class runEmergesort extends HttpServlet implements SortCycleListener {
	private static final long serialVersionUID = 1L; //class serialization id


	/*Server-wide variables*/
	private PrintWriter out; //the Servlet output stream, will be used for
writing the response

	private String ProtocolString;                              //String variable
that will keep the Protocol value
	private String SelectionTypeString;                         //String variable
that will keep the Selection value
	private String[] SelectionTypeExtraStringArray;      //String variable that
will keep the Selection type extra values
	private boolean LastSortedBoundBoolean;             //Boolean variable that
will keep the "Last Sorted Bound" value
	private boolean UseProbabilityTableBoolean;         //String variable that
will keep the "Use Probability Table" value
	private String ActionOnEndsString;                          //String variable
that will keep the "Action On Ends" value
	private String SortedSetString;                             //String variable
that will keep the "Sorted Set" value
	private String SortedSetBordersString;              //String variable that
will keep the "Sorted Set Borders" value
	private String NewSetBordersString;                         //String variable
that will keep the "New Set Borders" value
	private String LimitFactorString;                   //String variable that
will keep the "Limit Factor" value
	private String ExponentString;                              //String variable
that will keep the "Exponent" value
	private String ExponentToString;                    //String variable that
will keep the "Exponent To" value
	private String SamplesString;                               //String variable
that will keep the "Samples" value
	private String DiameterString;                              //String variable
that will keep the "Diameter" value
	private boolean EnableLoggingBoolean;               //Boolean variable that
will keep the "Enable Logging" value
	private long startTime=0,endTime=0;

	/*Server side variables used for running emergesort simulation*/
	Sorter sorter;                          //the Sorter object
    EmergeOptions options;       //the EmergeOptions object


    /**
     * @see HttpServlet#HttpServlet()
     * the servlet constructor...not overriden in this implementation
     */
    public runEmergesort() {
```

```
        super(); //HttpServlet "default" constructor functionality is used
    }

    /**
     * @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse
response)
     * this function is used for handling and responding to an HTTP GET request
     * in this implementation, a POST GET request is sent
     * we call the same functionality with doPost for debugging/test purposes
     */
    protected void doGet(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
            doPost(request, response);
    }

    /**
     * @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse
response)
     * this function is used for handling, parsing and responding to an HTTP POST
request to the servlet
     */
    protected void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {

            /*set the response's content type*/
            response.setContentType("text/html");

            /*assign to out variable, the response's output writer*/
            /*from now on, everything written on out, will be sent as response to
the calling browser*/
        out=response.getWriter();

        /*call the function that parses the request and initializes the
emergesort simulation parameters*/
        getSimulationParameters(request);

        /*call the function that sets up the HTML page that is sent as response*/
        printPageStart();

        /*call the function that "prints" the simulation parameters to response*/
        printSimulationParameters();

        /*call the function that runs the simulation*/
        runSimulation();

        /*call the function that "prints" the simulation Stats to response*/
        printSimulationStatistics();

        /*call the function that "prints" the simulation Log to response*/
        printSimulationLog();

        /*call the function that "closes" the HTML page that is sent as
response*/
        printPageEnd();

    }

    /**
     * This function prints to the response HTML tags that initialize the HTML
page
     * It also prints the page title and main header
     */
    private void printPageStart(){
        String title = "Emerge-Sort @ WWW";
        out.println("<!DOCTYPE HTML PUBLIC \"-//W3C//DTD HTML 4.0
Transitional//EN\">\n" +
                    "<HTML>\n" +
                    "<HEAD><TITLE>" + title + "</TITLE></HEAD>\n"+
                           "<body bgcolor=\"#C0C0C0\">\n" +
                           "<table  width=\"100%\"\n" +
                           "<tr bgcolor=\"#A0A0A0\">\n" +
```

```
                                "<td>\n" +
                                "<h1 ALIGN=\"CENTER\">Emerge-Sort @ WWW</h1>\n" +
                                "</td>\n" +
                                "</tr>\n" +
                                "</table>\n" +

                                "<table width=\"100%\" align=\"center\"
bgcolor=\"#C0C0C0\">\n" +
                                "<tr >\n" +

                                "<td bgcolor=\"#B0B0B0\">\n" +
                                "<center>\n" +
                                "<a href=\"./index.html\">Home Page</a>\n" +
                                "</center>\n" +
                                "</td>\n" +

                                "<td bgcolor=\"#B0B0B0\">\n" +
                                "<center>\n" +
                                "<a href=\"./applet.html\">Emerge-Sort Applet</a> \n" +
                                "</center>\n" +
                                "</td>\n" +

                                "<td bgcolor=\"#B0B0B0\">\n" +
                                "<center>\n" +
                                "<a href=\"./genetic.html\">Genetic Algorithm</a> \n" +
                                "</center>\n" +
                                "</td>\n" +

                                "<td bgcolor=\"#B0B0B0\">\n" +
                                "<center>\n" +
                                "<a href=\"./appletECJ.html\">Genetic Algorithm
Applet</a> \n" +
                                "</center>\n" +
                                "</td>\n" +

                                "<td bgcolor=\"#B0B0B0\">\n" +
                                "<center>\n" +
                                "<a href=\"./about.html\">About Emerge-Sort</a> \n" +
                                "</center>\n" +
                                "</td>\n" +

                                "<td bgcolor=\"#B0B0B0\">\n" +
                                "<center>\n" +
                                "<a href=\"./help.html\">Help</a>\n" +
                                "</center>\n" +
                                "</td>\n" +

                                "<td bgcolor=\"#A0A0A0\">\n" +
                                "<center>\n" +
                                "<a href=\"#\">Results</a>\n" +
                                "</center>\n" +
                                "</td>\n" +

                                "</tr>\n" +
                                "</table>\n" +
                                "<hr>\n");
        }

        /**
         * This function prints to the response HTML tags that end the HTML page
         */
        private void printPageEnd(){
            out.println("\n</body></html>");
        }

        /**
         * This function gets the simulation parameters from the request
         */
        private void getSimulationParameters(HttpServletRequest request){

              /*For string parameters, request.getParameter(paramName) is used*/
```

```
            /*paramName is the name of the parameter as it was in the HTML form in
index.html*/
            /*For boolean parameters special handling is used*/

          ProtocolString=request.getParameter("Protocol");
          SelectionTypeString=request.getParameter("SelectionType");

SelectionTypeExtraStringArray=request.getParameterValues("SelectionTypeExtra");
          LastSortedBoundBoolean=false;
          UseProbabilityTableBoolean=false;
          if(SelectionTypeExtraStringArray!=null){
             for(int i=0;i<SelectionTypeExtraStringArray.length;i++){
                    if(SelectionTypeExtraStringArray[i].equals("LastSortedBound")){
                           LastSortedBoundBoolean=true;
                    }

      if(SelectionTypeExtraStringArray[i].equals("UseProbabilityTable")){
                           UseProbabilityTableBoolean=true;
                    }
             }
          }
             ActionOnEndsString=request.getParameter("ActionOnEnds");
          SortedSetString=request.getParameter("SortedSet");
          SortedSetBordersString=request.getParameter("SortedSetBorders");
          NewSetBordersString=request.getParameter("NewSetBorders");
          LimitFactorString=request.getParameter("LimitFactor");
          ExponentString=request.getParameter("Exponent");
          ExponentToString=request.getParameter("ExponentTo");
          SamplesString=request.getParameter("Samples");
          DiameterString=request.getParameter("Diameter");
          EnableLoggingBoolean=false;
          if(request.getParameter("EnableLogging")!=null){
             EnableLoggingBoolean=true;

          }

      }


      /**
       * This function prints the simulation parameters that were previously parsed
by the request
       */
      private void printSimulationParameters(){

          out.println("<h2>Simulation Parameters:</h2>" );
          out.println("You have just executed Emerge Sort with the following
parameters:<br>");
          out.println("<b>Protocol:</b>"+ProtocolString+"<br>");
          out.println("<b>Selection Type:</b>"+SelectionTypeString+"<br>");
             out.println("<b>Use Probability
Table:</b>"+UseProbabilityTableBoolean+"<br>");
             out.println("<b>Last Sorted
Bound:</b>"+LastSortedBoundBoolean+"<br>");
          out.println("<b>Action On Ends:</b>"+ActionOnEndsString+"<br>");
          out.println("<b>Momentum Variations On Sorted
Set:</b>"+SortedSetString+"<br>");
          out.println("<b>Momentum Variations On Sorted Set
Borders:</b>"+SortedSetBordersString+"<br>");
          out.println("<b>Momentum Variations On New Set
Borders:</b>"+NewSetBordersString+"<br>");
          out.println("<b>Limit Factor:</b>"+LimitFactorString+"<br>");
          out.println("<b>Exponent
(2^):</b>"+ExponentString+"    ");
          out.println("<b>Exponent To(2^):</b>"+ExponentToString+"<br>");
          out.println("<b>Samples:</b>"+SamplesString+"<br>");
          out.println("<b>Diameter:</b>"+DiameterString+"<br>");
          out.println("<b>Enable Logging:</b>"+EnableLoggingBoolean);

          out.println("<br><br>");
          out.println("<a href=\".\\index.html\">Back</a> to Home Page<br>");
```

```
            out.println("<font color=\"red\">CAUTION: If F5(Refresh) is pressed on
this page (Results), a new Emerge-Sort Execution will be started.</font>");

            out.println("<hr/>");
        }
        /**
         * This function runs the emergesort simulation based on the parameters from
the request
         */
        private void runSimulation(){

              startTime=System.currentTimeMillis();
              /*initialize sorter object*/
              sorter = new Sorter();

              /*initialize options object*/
         options = new EmergeOptions();

         /*set emergesort limit factor(double)*/
         options.setLimitFactor((float)Double.parseDouble(LimitFactorString));

         /*set emergesort enable logging (true/false)*/
         options.setEnableLogging(EnableLoggingBoolean);

         /*set emergesort log rate (1-default)*/
         options.setLogRate(1);

         /*set emergesort Emerge Type*/
         options.setEmergeType(getEmergeTypeFromString());

         /*set emergesort Selection Type*/
         options.setSelAtomType(getAtomSelectionTypeFromString());

         /*set emergesort Wrap Type*/
         options.setWrapType(getWrapTypeFromString());

         /*set Actions on set when selected atoms are sorted*/
         options.setSortedSetAction(getSortedSetActionFromString());

         /*set Actions on set borders when selected atoms are sorted*/
         options.setSortedSetBorderAction(getSortedSetBordersActionFromString());

         /*set Actions on unsorted set borders when selected atoms are sorted*/
         options.setSetBorderAction(getUnSortedSetBordersActionFromString());

         /*set Sorter options*/
         sorter.setSampleSize(Integer.parseInt(SamplesString));           //Number
of Samples
         sorter.setDiameter(Integer.parseInt(DiameterString));            //Size of
neighbourhood
         sorter.setBase(2);
        //Base of the population size
         sorter.setPowerMin(Integer.parseInt(ExponentString)-2);    //Exponent of the
population size
         sorter.setPowerMax(Integer.parseInt(ExponentToString)-2);        //Exponent
of the population size

         sorter.setEmergeOptions(options);

         //debugging sorter.addSortCycleListener((SortCycleListener) this);
         /*Start sorting*/
         sorter.Sort();
         //debugging sorter.removeSortCycleListener((SortCycleListener) this);
         endTime=System.currentTimeMillis();

        }

        /**
         * This function "prints"-> fills the response HTML with the simulation stats
         *
         */
```

```
        private void printSimulationStatistics(){


                PopulationGroup popGroup;
                Stats st;
                /*Print statistics header*/
            out.println("<h2>Statistics:</h2>");
            out.println("Simulation execution time:  "+(endTime-startTime)+"
milliseconds");

            /*call this function in order to initialize Statistics table (start-end
tags and headers)*/
            createStatisticsTable();
            for (int i = 0; i < sorter.getPopulationGroups().size(); i++){

                  /*for each population group print the aggeregated statistics*/
                  popGroup = (PopulationGroup)sorter.getPopulationGroups().get(i);
                  st = (Stats)popGroup.getGroupStats();
                  out.println("<tr bgcolor=\"#D0D0D0\">");
                  out.println("<td>"+(i+1)+"</td>");
                  out.println("<td>"+popGroup.getPopSize()+"</td>");
                  out.println("<td>"+st.getAvgRounds()+"</td>");
                  out.println("<td>"+st.getMinRounds()+"</td>");
                  out.println("<td>"+st.getMaxRounds()+"</td>");
                  out.println("<td>"+st.getRounds()+"</td>");
                  out.println("<td>"+st.getAvgMoves()+"</td>");
                  out.println("<td>"+st.getMinMoves()+"</td>");
                  out.println("<td>"+st.getMaxMoves()+"</td>");
                  out.println("<td>"+st.getTotalMoves()+"</td>");
                  out.println("<td>"+st.getAvgDistance()+"</td>");
                  out.println("<td>"+st.getMinDistance()+"</td>");
                  out.println("<td>"+st.getMaxDistance()+"</td>");
                  out.println("<td>"+st.getTotalDistance()+"</td>");
                  out.println("<td>"+st.getSortedAscending()+"</td>");
                  out.println("<td>"+st.getSortedDescending()+"</td>");
                  out.println("</tr>");

                  for (int j = 0; j < popGroup.getPopulations().size(); j++){
                        /*for each population print its statistics*/
                        st =
(Stats)((Population)popGroup.getPopulations().get(j)).getStats();
                        out.println("<tr bgcolor=\"#F0F0F0\">");    //different
colour than the aggregated group results
                        out.println("<td>"+(i+1)+"."+(j+1)+"</td>");
                        out.println("<td>"+popGroup.getPopSize()+"</td>");
                        out.println("<td>"+st.getAvgRounds()+"</td>");
                        out.println("<td>"+st.getMinRounds()+"</td>");
                        out.println("<td>"+st.getMaxRounds()+"</td>");
                        out.println("<td>"+st.getRounds()+"</td>");
                        out.println("<td>"+st.getAvgMoves()+"</td>");
                        out.println("<td>"+st.getMinMoves()+"</td>");
                        out.println("<td>"+st.getMaxMoves()+"</td>");
                        out.println("<td>"+st.getTotalMoves()+"</td>");
                        out.println("<td>"+st.getAvgDistance()+"</td>");
                        out.println("<td>"+st.getMinDistance()+"</td>");
                        out.println("<td>"+st.getMaxDistance()+"</td>");
                        out.println("<td>"+st.getTotalDistance()+"</td>");
                        out.println("<td>"+st.getSortedAscending()+"</td>");
                        out.println("<td>"+st.getSortedDescending()+"</td>");
                        out.println("</tr>");
                  }
            }
            out.println("</table>");
            out.println("<hr/>");

        }

        /**
         * This function initializes Statistics table (start tags and headers)
         */
    private void createStatisticsTable()
```

```
    {

       out.println("<table>");
       out.println("<tr bgcolor=\"#707070\">"); //"darker" colour on the header
       out.println("<th colspan=2>General</th>");
       out.println("<th colspan=4>Rounds</th>");
       out.println("<th colspan=4>Moves</th>");
       out.println("<th colspan=4>Distance</th>");
       out.println("<th colspan=2>Sort Order</th>");
       out.println("</tr>");

       out.println("<tr bgcolor=\"#707070\">"); //"darker" colour on the header
       out.println("<th>Index</th>");
       out.println("<th>Pop Size</th>");
       out.println("<th>Avg. Rounds</th>");
       out.println("<th>Min Rounds</th>");
       out.println("<th>Max Rounds</th>");
       out.println("<th>Total Rounds</th>");
       out.println("<th>Avg. Moves</th>");
       out.println("<th>Min Moves</th>");
       out.println("<th>Max Moves</th>");
       out.println("<th>Total Moves</th>");
       out.println("<th>Avg Distance</th>");
       out.println("<th>Min Distance</th>");
       out.println("<th>Max Distance</th>");
       out.println("<th>Total Distance</th>");
       out.println("<th>Ascending</th>");
       out.println("<th>Descending</th>");
       out.println("</tr>");
    }


       /**
        * This function "prints"-> fills the response HTML with the simulation log
        *
        */
    private void printSimulationLog(){
             String colour="\"#D0D0D0\"";
             Population pop ;
             PopulationGroup popGroup;
             PopulationLog log;
             /*Print statistics header*/
             out.println("<h2>Log:</h2>");

             /*call this function in order to initialize Log table (start tags and
headers)*/
             createLogTable();

             for (int i = 0; i < sorter.getPopulationGroups().size(); i++){

                    /*for each population group find the populations */
                    popGroup = (PopulationGroup)sorter.getPopulationGroups().get(i);
                    for (int j = 0; j < popGroup.getPopulations().size(); j++){

                           /*for each population get the log */
                           pop = (Population)popGroup.getPopulations().get(j);
                           log = (PopulationLog)pop.getLog();
                           for (int k = 0; k < log.getFormattedValues().size(); k++)
               {
                                  /*for each log entry print the related values */

                                  /*general log line*/
                                  out.println("<tr bgcolor="+colour+">");
                                  out.println("<td>"+pop.getSize()+"</td>");
                                  out.println("<td>"+(i+1)+"</td>");
                                  out.println("<td>"+(j+1)+"</td>");
                                  out.println("<td>"+(k+1)+"</td>");

       out.println("<td>"+log.getFormattedHeaders().get(k)+"</td>");

       out.println("<td>"+log.getFormattedValues().get(k)+"</td>");
```

```
                                        out.println("</tr>");

                                        /*formated directions line*/
                                        out.println("<tr bgcolor="+colour+">");
                                        out.println("<td colspan=5></td>");
                                        out.println("<td
colspan=1>"+log.getFormattedDirections().get(k)+"</td>");
                                        out.println("</tr>");

                                        /*formated arrea moves line*/
                                        out.println("<tr bgcolor="+colour+">");
                                        out.println("<td colspan=5></td>");
                                        out.println("<td
colspan=1>"+log.getFormattedAreaMoves().get(k)+"</td>");
                                        out.println("</tr>");

                                        /*formated moves line*/
                                        out.println("<tr bgcolor="+colour+">");
                                        out.println("<td colspan=5></td>");
                                        out.println("<td
colspan=1>"+log.getFormattedMoves().get(k)+"</td>");
                                        out.println("</tr>");

                                        /*formated distances line*/
                                        out.println("<tr bgcolor="+colour+">");
                                        out.println("<td colspan=5></td>");
                                        out.println("<td
colspan=1>"+log.getFormattedDistances().get(k)+"</td>");
                                        out.println("</tr>");

                                        /*change colour between continuous log entries*/
                                        if(colour.equals("\"#D0D0D0\"")){
                                                colour="\"#F0F0F0\"";
                                        }
                                        else{
                                                colour="\"#D0D0D0\"";
                                        }
                                }
                                    }
                        }

                        /*"close" Log table*/
                        out.println("</table>");
                        out.println("<hr/>");

                }

                /**
                 * This function initializes Log table (start tags and headers)
                 */
                private void createLogTable()
                {
                        out.println("<table>");

                        out.println("<tr bgcolor=\"#707070\">");
                        out.println("<th>Size</th>");
                        out.println("<th>Group Index</th>");
                        out.println("<th>Pop Index</th>");
                        out.println("<th>Log Index</th>");
                        out.println("<th>Position</th>");
                        out.println("<th>Population Values</th>");
                        out.println("</tr>");

                        out.println("<tr bgcolor=\"#707070\">");
                        out.println("<th colspan=5> </th>");
                        out.println("<th colspan=1>Properties</th>");
                        out.println("</tr>");

                        out.println("<tr bgcolor=\"#707070\">");
                        out.println("<th colspan=5> </th>");
                        out.println("<th colspan=1>Area Moves</th>");
```

```
                out.println("</tr>");

                out.println("<tr bgcolor=\"#707070\">");
                out.println("<th colspan=5> </th>");
                out.println("<th colspan=1>Moves</th>");
                out.println("</tr>");

                out.println("<tr bgcolor=\"#707070\">");
                out.println("<th colspan=5> </th>");
                out.println("<th colspan=1>Distances</th>");
                out.println("</tr>");
    }

       /**
        * This function reads String variable ProtocolString and returns the
appropriate Enm.EmergeType
        */
       private Enm.EmergeType getEmergeTypeFromString()
       {
              if (ProtocolString.equals("LRProtocol")){
              return Enm.EmergeType.LRProtocol;
        }
        else if(ProtocolString.equals("ADProtocolPosition")){
              return Enm.EmergeType.ADProtocolP;
        }
        else if (ProtocolString.equals("ADProtocolNumber")){
              return Enm.EmergeType.ADProtocolM;
        }
        else{
            /*warning printed in server log if invalid parameter is passed*/
            /*default Enm type assigned*/
              getServletContext().log("Warning:Invalid EmergeType, set to default");
              return Enm.EmergeType.ADProtocolP;
        }
    }

       /**
        * This function reads String variable SelectionTypeString and returns the
appropriate Enm.SelAtomType
        */
       private Enm.SelAtomType getAtomSelectionTypeFromString()
    {
        if (SelectionTypeString.equals("Serial"))
        {
            if (!UseProbabilityTableBoolean && !LastSortedBoundBoolean)
                return Enm.SelAtomType.Serial;
            if (!UseProbabilityTableBoolean && LastSortedBoundBoolean)
                return Enm.SelAtomType.SemiSerial;
            if (UseProbabilityTableBoolean && LastSortedBoundBoolean)
                return Enm.SelAtomType.SemiSerialProb;
            if (UseProbabilityTableBoolean && !LastSortedBoundBoolean)
                return Enm.SelAtomType.SerialProb;
        }
        else
        {
            if (!UseProbabilityTableBoolean && !LastSortedBoundBoolean)
                return Enm.SelAtomType.RandomUni;

            if (!UseProbabilityTableBoolean && LastSortedBoundBoolean)
                return Enm.SelAtomType.SemiRandom;

            if (UseProbabilityTableBoolean && !LastSortedBoundBoolean)
                return Enm.SelAtomType.RandomProb;

            if (UseProbabilityTableBoolean && LastSortedBoundBoolean)
                return Enm.SelAtomType.SemiRandomProb;

        }
        /*warning printed in server log if invalid parameter is passed*/
        /*default Enm type assigned*/
```

```
        getServletContext().log("Warning:Invalid AtomSelectionType, set to
default(Random)");
        return Enm.SelAtomType.RandomUni;
    }

       /**
        * This function reads String variable ActionOnEndsString and returns the
appropriate Enm.WrapType
        */
       private Enm.WrapType getWrapTypeFromString()
    {
        if(ActionOnEndsString.equals("Wrap")){
             return Enm.WrapType.Wrap;
        }
        else if(ActionOnEndsString.equals("Inflect")){
             return Enm.WrapType.Inflect;
        }
        else if(ActionOnEndsString.equals("ExcludeEnds")){
             return Enm.WrapType.NoEnds;
        }
        else{
            /*warning printed in server log if invalid parameter is passed*/
            /*default Enm type assigned*/
             getServletContext().log("Warning:Invalid WrapType, set to
default(Wrap)");
             return Enm.WrapType.Wrap;
        }

    }

       /**
        * This function reads String variable SortedSetString and returns the
appropriate Enm.SortedSetAction
        */
    private Enm.SortedSetAction getSortedSetActionFromString()
    {
        if (SortedSetString.equals("NoAction")){
            return Enm.SortedSetAction.NoAction;
        }

        else if (SortedSetString.equals("SetProperties")){
            return Enm.SortedSetAction.SetFlags;
        }

        else if (SortedSetString.equals("ResetProperties")){
            return Enm.SortedSetAction.ResetFlags;
        }

        else if (SortedSetString.equals("ByNumber")){
            return Enm.SortedSetAction.ByNumber;
        }
        else{
            /*warning printed in server log if invalid parameter is passed*/
            /*default Enm type assigned*/
             getServletContext().log("Warning:Invalid SortedSetAction, set to
default(NoAction)");
             return Enm.SortedSetAction.NoAction;
        }


    }

       /**
        * This function reads String variable SortedSetBordersString and returns the
appropriate Enm.SortedSetBorderAction
        */
    private Enm.SortedSetBorderAction getSortedSetBordersActionFromString()
    {
        if (SortedSetBordersString.equals("NoAction")){
            return Enm.SortedSetBorderAction.NoAction;
        }
```

```
        else if (SortedSetBordersString.equals("SetProperties")){
            return Enm.SortedSetBorderAction.SetFlags;
        }

        else if (SortedSetBordersString.equals("ResetProperties")){
            return Enm.SortedSetBorderAction.ResetFlags;
        }

        else if (SortedSetBordersString.equals("ScaleProperties")){
            return Enm.SortedSetBorderAction.ScaleFlags;
        }
        else {
            /*warning printed in server log if invalid parameter is passed*/
            /*default Enm type assigned*/
             getServletContext().log("Warning:Invalid SortedSetBordersAction, set
to default(NoAction)");
             return Enm.SortedSetBorderAction.NoAction;
        }


    }

       /**
        * This function reads String variable NewSetBordersString and returns the
appropriate Enm.SetBorderAction
        */
    private Enm.SetBorderAction getUnSortedSetBordersActionFromString()
    {
        if (NewSetBordersString.equals("NoAction")){
            return Enm.SetBorderAction.NoAction;
        }
        else if (NewSetBordersString.equals("SetProperties")){
            return Enm.SetBorderAction.SetFlags;
        }

        else if (NewSetBordersString.equals("ResetProperties")){
            return Enm.SetBorderAction.ResetFlags;
        }

        else if (NewSetBordersString.equals("ScaleProperties")){
            return Enm.SetBorderAction.ScaleFlags;
        }
        else {
            /*warning printed in server log if invalid parameter is passed*/
            /*default Enm type assigned*/
             getServletContext().log("Warning:Invalid UnSortedSetBordersAction, set
to default(NoAction)");
             return Enm.SetBorderAction.NoAction;
        }

    }

       /**
        * This function is created for debugging purposes
        */
    public void SortCycleEnded(SortCycleEvent e)
    {
        //debugging getServletContext().log("hello");
    }

       }
```

Γ.2 Ο κώδικας του Java Applet emergeSortApplet

```
package emergesortapplet;

import java.awt.Color;

public class emergeSortApplet extends JApplet {
	/**
	 *
	 */
	private static final long serialVersionUID = 1L;
	private final ButtonGroup ActionOnEnds = new ButtonGroup();
	private final ButtonGroup SelectionType = new ButtonGroup();

	private String ProtocolString; // String variable that will keep the Protocol
value
	private String SelectionTypeString; // String variable that will keep the
Selection value
	private boolean LastSortedBoundBoolean; // Boolean variable that will keep
the "Last Sorted Bound" value
	private boolean UseProbabilityTableBoolean; // Boolean variable that will
keep the "Use Probability Table" value
	private String ActionOnEndsString; // String variable that will keep the
"Action On Ends" value
	private String SortedSetString; // String variable that will keep the "Sorted
Set" value
	private String SortedSetBordersString; // String variable that will keep the
"Sorted Set Borders" value
	private String NewSetBordersString; // String variable that will keep the
"New Set Borders" value
	private String LimitFactorString; // String variable that will keep the
"Limit Factor" value
	private String ExponentString; // String variable that will keep the
"Exponent" value
	private String ExponentToString; // String variable that will keep the
"Exponent To" value
	private String SamplesString; // String variable that will keep the "Samples"
value
	private String DiameterString; // String variable that will keep the
"Diameter" value
	private boolean EnableLoggingBoolean; // Boolean variable that will keep the
"Enable Logging" value

	/* Server side variables used for running emergesort simulation */
	Sorter sorter; // the Sorter object
	EmergeOptions options; // the EmergeOptions object

	/*global gui variables*/
	private JTable tableRounds;
	private JTable tableDistance;
	private JTable tableMoves;
	private JTable tableOrder;
	private JTable tableLog;
	private JTable tableParameters;
	JTabbedPane statisticsPane;
	JTabbedPane logPane;
	JLabel Messages;
	JLabel statisticsLabel;
	JLabel logLabel;
	PrintStream stdout;
	PrintStream stderr;
	private long startTime;
	private long endTime;

	/**
	 * Create the applet.
	 * This function actually prepares the applet GUI
```

```
        */
       public emergeSortApplet() {

              /*GUI setup*/

              /*Setup the applet window*/
              getContentPane().setBackground(Color.LIGHT_GRAY);
              setSize(new Dimension(800, 1200));
              getContentPane().setLayout(null);

              /*applet window is divided in JPanels that contain the various GUI
controls*/

              /*Setup the JPanel that contains the Protocol Parameters*/
              JPanel panel_1 = new JPanel();
              panel_1.setBackground(SystemColor.control);
              panel_1.setBorder(new LineBorder(Color.GRAY));
              panel_1.setBounds(100, 35, 300, 130);
              getContentPane().add(panel_1);
              panel_1.setLayout(null);

              JLabel lblNewLabel = new JLabel("Protocol");//Protocol Label
              lblNewLabel.setBounds(10, 11, 97, 22);
              panel_1.add(lblNewLabel);
              lblNewLabel.setFont(new Font("Tahoma", Font.BOLD, 18));

              final JComboBox Protocol = new JComboBox();//Protocol Combo Box
              Protocol.setModel(new DefaultComboBoxModel(new String[] {
                            "L/R Protocol", "A/D Protocol (By Position Moves)",
                            "A/D Protocol (By Number Moves)" }));
              Protocol.setSelectedIndex(1);
              Protocol.setBounds(10, 40, 210, 20);
              panel_1.add(Protocol);


              /*Setup the JPanel that contains the Selection Type Parameters*/
              JPanel panel = new JPanel();
              panel.setBorder(new LineBorder(Color.GRAY));
              panel.setLayout(null);
              panel.setBackground(SystemColor.control);
              panel.setBounds(398, 35, 300, 130);
              getContentPane().add(panel);

              JLabel lblSelectionType = new JLabel("Selection Type");//Selection
Type Label
              lblSelectionType.setBounds(10, 11, 139, 22);
              panel.add(lblSelectionType);
              lblSelectionType.setFont(new Font("Tahoma", Font.BOLD, 18));

              final JRadioButton Random = new JRadioButton("Random");//"Random"
Radio Button
              SelectionType.add(Random);
              Random.setSelected(true);
              Random.setBackground(SystemColor.control);
              Random.setFont(new Font("Tahoma", Font.PLAIN, 14));
              Random.setBounds(10, 40, 109, 23);
              panel.add(Random);

              final JRadioButton Serial = new JRadioButton("Serial");//"Serial"
Radio Button
              SelectionType.add(Serial);
              Serial.setBackground(SystemColor.control);
              Serial.setFont(new Font("Tahoma", Font.PLAIN, 14));
              Serial.setBounds(121, 40, 109, 23);
              panel.add(Serial);

              final JCheckBox LastSortedBound = new JCheckBox("Last Sorted
Bound");//"Last Sorted Bound" Checkbox
              LastSortedBound.setSelected(true);
              LastSortedBound.setFont(new Font("Tahoma", Font.PLAIN, 14));
              LastSortedBound.setBackground(SystemColor.control);
```

```
            LastSortedBound.setBounds(10, 66, 167, 23);
            panel.add(LastSortedBound);

            final JCheckBox UseProbabilityTable = new JCheckBox("Use Probability
Table");//"Use Probability Table" Checkbox
            UseProbabilityTable.setSelected(true);
            UseProbabilityTable.setFont(new Font("Tahoma", Font.PLAIN, 14));
            UseProbabilityTable.setBackground(SystemColor.control);
            UseProbabilityTable.setBounds(10, 92, 167, 23);
            panel.add(UseProbabilityTable);


            /*Setup the JPanel that contains the Action on Ends Parameters*/
            JPanel panel_2 = new JPanel();
            panel_2.setBorder(new LineBorder(Color.GRAY));
            panel_2.setLayout(null);
            panel_2.setBackground(SystemColor.control);
            panel_2.setBounds(100, 164, 300, 130);
            getContentPane().add(panel_2);

            JLabel lblActionOnEnds = new JLabel("Action On Ends");//"Action On
Ends" Label
            lblActionOnEnds.setBounds(10, 11, 179, 22);
            panel_2.add(lblActionOnEnds);
            lblActionOnEnds.setFont(new Font("Tahoma", Font.BOLD, 18));

            final JRadioButton Wrap = new JRadioButton("Wrap");//"Wrap" Radio
Button
            ActionOnEnds.add(Wrap);
            Wrap.setSelected(true);
            Wrap.setFont(new Font("Tahoma", Font.PLAIN, 14));
            Wrap.setBackground(SystemColor.control);
            Wrap.setBounds(10, 40, 109, 23);
            panel_2.add(Wrap);

            final JRadioButton Inflect = new JRadioButton("Inflect");//"Wrap"
Radio Button
            ActionOnEnds.add(Inflect);
            Inflect.setFont(new Font("Tahoma", Font.PLAIN, 14));
            Inflect.setBackground(SystemColor.control);
            Inflect.setBounds(10, 66, 109, 23);
            panel_2.add(Inflect);

            final JRadioButton ExcludeEnds = new JRadioButton("Exclude
Ends");//"Exclude Ends" Radio Button
            ActionOnEnds.add(ExcludeEnds);
            ExcludeEnds.setFont(new Font("Tahoma", Font.PLAIN, 14));
            ExcludeEnds.setBackground(SystemColor.control);
            ExcludeEnds.setBounds(10, 92, 109, 23);
            panel_2.add(ExcludeEnds);


            /*Setup the JPanel that contains the Action on Ends Parameters*/

            JPanel panel_3 = new JPanel();
            panel_3.setBorder(new LineBorder(Color.GRAY));
            panel_3.setLayout(null);
            panel_3.setBackground(SystemColor.control);
            panel_3.setBounds(398, 164, 300, 130);
            getContentPane().add(panel_3);

            JLabel lblMomentumVariationsOn = new JLabel("Momentum Variations
On...");//"Momentum Variations" Label
            lblMomentumVariationsOn.setBounds(10, 11, 255, 22);
            panel_3.add(lblMomentumVariationsOn);
            lblMomentumVariationsOn.setFont(new Font("Tahoma", Font.BOLD, 18));

            JLabel lblSortedSet = new JLabel("Sorted Set");//"Sorted Set" Label
            lblSortedSet.setFont(new Font("Tahoma", Font.PLAIN, 14));
            lblSortedSet.setBounds(10, 40, 84, 14);
            panel_3.add(lblSortedSet);
```

```
            JLabel lblSortedSetBorders = new JLabel("Sorted Set
Borders");//"Sorted Set Borders" Label
            lblSortedSetBorders.setFont(new Font("Tahoma", Font.PLAIN, 14));
            lblSortedSetBorders.setBounds(10, 66, 122, 14);
            panel_3.add(lblSortedSetBorders);

            JLabel lblNewSetBorders = new JLabel("New Set Borders");//"New Set
Borders" Label
            lblNewSetBorders.setFont(new Font("Tahoma", Font.PLAIN, 14));
            lblNewSetBorders.setBounds(10, 92, 109, 14);
            panel_3.add(lblNewSetBorders);

            final JComboBox SortedSet = new JComboBox();//"Sorted Set" options'
Combo Box
            SortedSet.setModel(new DefaultComboBoxModel(new String[] { "No
Action",
                        "Set Properties", "Reset Properties", "Scale
Properties" }));
            SortedSet.setBounds(129, 40, 150, 20);
            panel_3.add(SortedSet);

            final JComboBox SortedSetBorders = new JComboBox();//"Sorted Set
Borders" options' Combo Box
            SortedSetBorders.setModel(new DefaultComboBoxModel(new String[] {
                        "No Action", "Set Properties", "Reset Properties",
                        "Scale Properties" }));
            SortedSetBorders.setBounds(129, 66, 150, 20);
            panel_3.add(SortedSetBorders);

            final JComboBox NewSetBorders = new JComboBox();//"New Set Borders"
options' Combo Box
            NewSetBorders.setModel(new DefaultComboBoxModel(new String[] {
                        "No Action", "Set Properties", "Reset Properties",
                        "Scale Properties" }));
            NewSetBorders.setBounds(129, 92, 150, 20);
            panel_3.add(NewSetBorders);


            /*Setup the JPanel that contains the Population Size Parameters*/
            JPanel panel_4 = new JPanel();
            panel_4.setBorder(new LineBorder(Color.GRAY));
            panel_4.setLayout(null);
            panel_4.setBackground(SystemColor.control);
            panel_4.setBounds(100, 293, 300, 100);
            getContentPane().add(panel_4);

            JLabel lblPopulationSize = new JLabel("Population Size");//"Population
Size" Label
            lblPopulationSize.setBounds(10, 11, 179, 22);
            panel_4.add(lblPopulationSize);
            lblPopulationSize.setFont(new Font("Tahoma", Font.BOLD, 18));

            JLabel lblLimitFactor = new JLabel("Limit Factor");//"Limit Factior"
Label
            lblLimitFactor.setFont(new Font("Tahoma", Font.PLAIN, 14));
            lblLimitFactor.setBounds(10, 40, 84, 14);
            panel_4.add(lblLimitFactor);

            JLabel lblExponent = new JLabel("Exponent");//"Exponent" Label
            lblExponent.setFont(new Font("Tahoma", Font.PLAIN, 14));
            lblExponent.setBounds(10, 66, 84, 14);
            panel_4.add(lblExponent);

            final JSpinner LimitFactor = new JSpinner();//"Limit Factor" Spinner
            LimitFactor.setModel(new SpinnerNumberModel(new Float(1), null, null,
new Float(1)));
            LimitFactor.setBounds(142, 40, 45, 20);
            panel_4.add(LimitFactor);

            final JSpinner ExpFrom = new JSpinner();//"Exponent From" Spinner
```

```
            ExpFrom.setModel(new SpinnerNumberModel(new Integer(3), new
Integer(3),
                        null, new Integer(1)));
            ExpFrom.setBounds(142, 66, 45, 20);
            panel_4.add(ExpFrom);

            JLabel lblFrom = new JLabel("From 2^");//"Exponent" Label
            lblFrom.setFont(new Font("Tahoma", Font.PLAIN, 14));
            lblFrom.setBounds(85, 66, 65, 14);
            panel_4.add(lblFrom);

            JLabel lblTo = new JLabel("To 2^");//"Exponent To" Label
            lblTo.setFont(new Font("Tahoma", Font.PLAIN, 14));
            lblTo.setBounds(199, 66, 38, 14);
            panel_4.add(lblTo);

            final JSpinner ExpTo = new JSpinner();//"Exponent To" Spinner
            ExpTo.setModel(new SpinnerNumberModel(new Integer(3), new Integer(3),
                        null, new Integer(1)));
            ExpTo.setBounds(240, 66, 45, 20);
            panel_4.add(ExpTo);


            /*Setup the JPanel that contains the Environment Settings Parameters*/
            JPanel panel_5 = new JPanel();
            panel_5.setBorder(new LineBorder(Color.GRAY));
            panel_5.setLayout(null);
            panel_5.setBackground(SystemColor.control);
            panel_5.setBounds(398, 293, 300, 100);
            getContentPane().add(panel_5);

            JLabel lblEnvironmentSettings = new JLabel("Environment
Settings");//"Environment Settings" Label
            lblEnvironmentSettings.setBounds(10, 11, 260, 22);
            panel_5.add(lblEnvironmentSettings);
            lblEnvironmentSettings.setFont(new Font("Tahoma", Font.BOLD, 18));

            JLabel lblSamples = new JLabel("Samples");//"Samples" Label
            lblSamples.setFont(new Font("Tahoma", Font.PLAIN, 14));
            lblSamples.setBounds(10, 40, 84, 14);
            panel_5.add(lblSamples);

            JLabel lblDiameter = new JLabel("Diameter");//"Diameter" Label
            lblDiameter.setFont(new Font("Tahoma", Font.PLAIN, 14));
            lblDiameter.setBounds(10, 69, 84, 14);
            panel_5.add(lblDiameter);

            final JSpinner Samples = new JSpinner();//"Samples" Spinner
            Samples.setModel(new SpinnerNumberModel(new Integer(10), null, null,
                        new Integer(1)));
            Samples.setBounds(104, 40, 45, 20);
            panel_5.add(Samples);

            final JSpinner Diameter = new JSpinner();//"Diameter" Spinner
            Diameter.setModel(new SpinnerNumberModel(new Integer(3), null, null,
                        new Integer(1)));
            Diameter.setBounds(104, 68, 45, 20);
            panel_5.add(Diameter);

            final JCheckBox EnableLogging = new JCheckBox("Enable
Logging");//"Enable Logging" Checkbox
            EnableLogging.setBackground(Color.LIGHT_GRAY);
            EnableLogging.setFont(new Font("Segoe UI", Font.PLAIN, 14));
            EnableLogging.setBounds(408, 405, 130, 23);
            getContentPane().add(EnableLogging);



            Messages= new JLabel("");//Empty-at start- label used for execution
information messages
            Messages.setHorizontalAlignment(SwingConstants.CENTER);
```

```
		Messages.setBounds(100, 13, 598, 16);
		Messages.setForeground(Color.RED);
		getContentPane().add(Messages);

		statisticsLabel = new JLabel("");//The label used for Statistics
heading after algorithm execution
		statisticsLabel.setFont(new Font("Tahoma", Font.PLAIN, 14));
		statisticsLabel.setBounds(31, 435, 99, 23);
		getContentPane().add(statisticsLabel);

		logLabel = new JLabel("");//The label used for Log heading after
algorithm execution
		logLabel.setFont(new Font("Tahoma", Font.PLAIN, 14));
		logLabel.setBounds(31, 618, 46, 23);
		getContentPane().add(logLabel);

		/*Run button is created*/
		JButton btnRunEmergesortApplet = new JButton("Run Emerge-Sort
Applet");
		btnRunEmergesortApplet.addActionListener(new ActionListener() {
			public void actionPerformed(ActionEvent e) {
			}
		});
		btnRunEmergesortApplet.setFont(new Font("Segoe UI", Font.PLAIN, 14));
		btnRunEmergesortApplet.setBounds(195, 404, 187, 25);
		getContentPane().add(btnRunEmergesortApplet);

		/*Run button action listeners*/
		btnRunEmergesortApplet.addMouseListener(new MouseAdapter() {
			/*when button is pressed, simulation message is printed and
panes are cleared*/
			/*needed in order for the information message to be visible when
simulation is executed*/
			@Override
			public void mousePressed(MouseEvent e) {
				Messages.setText("Please Wait...Simulation is
Running...");
				if (statisticsPane != null) {
					//if simulation already run, delete Statistics
Pane
					getContentPane().remove(statisticsPane);
					statisticsPane = null;
					statisticsLabel.setText("");
					getContentPane().repaint();
				}
				if (logPane != null) {
					//if simulation already run, delete Log Pane
					getContentPane().remove(logPane);
					logPane = null;
					logLabel.setText("");
					getContentPane().repaint();
				}
			}
		}); /*end of mousePressed listener*/

		btnRunEmergesortApplet.addMouseListener(new MouseAdapter() {
			/*when button is released, simulation variables are filled and
simulation starts*/
			@Override
			public void mouseReleased(MouseEvent e) {

				/*fill global simulation parameters' variables from GUI
controls*/
				ProtocolString = Protocol.getSelectedItem().toString();
				if (Random.isSelected()) {
					SelectionTypeString = "Random";
				} else if (Serial.isSelected()) {
					SelectionTypeString = "Serial";
				}
				LastSortedBoundBoolean = LastSortedBound.isSelected();
```

```
                        UseProbabilityTableBoolean =
UseProbabilityTable.isSelected();
                        if (Wrap.isSelected()) {
                              ActionOnEndsString = "Wrap";
                        } else if (Inflect.isSelected()) {
                              ActionOnEndsString = "Inflect";
                        } else if (ExcludeEnds.isSelected()) {
                              ActionOnEndsString = "ExcludeEnds";
                        }
                        SortedSetString = SortedSet.getSelectedItem().toString();
                        SortedSetBordersString =
SortedSetBorders.getSelectedItem().toString();
                        NewSetBordersString =
NewSetBorders.getSelectedItem().toString();
                        LimitFactorString = LimitFactor.getValue().toString();

                        ExponentString = ExpFrom.getValue().toString();
                        ExponentToString = ExpTo.getValue().toString();
                        SamplesString = Samples.getValue().toString();
                        DiameterString = Diameter.getValue().toString();
                        EnableLoggingBoolean = EnableLogging.isSelected();

                        //main functionality of applet is started
                        runEmergeSortApplet();
                  }
            }); /*end of mousePressed listener*/
      }/*end of emergeSortApplet constructor*/

      /**
       *This function contains the main functionality of the applet,
       *ie running the simulation and printing the results
       */
      private void runEmergeSortApplet() {

            /* call the function that runs the simulation */
            runSimulation();

            /* call the function that "prints" the simulation Statistics  */
            printSimulationStatistics();

            /* call the function that "prints" the simulation Log  */
            printSimulationLog();


      }/*end of runEmergeSortApplet*/


      /**
       * This function initializes the Sorter object, initializes its options and
runs the simulation
       */
      private void runSimulation() {
            startTime=System.currentTimeMillis();
            /* initialize sorter object */
            sorter = new Sorter();

            /* initialize options object */
            options = new EmergeOptions();

            /* set emergesort limit factor(double) */
            options.setLimitFactor((float) Float.parseFloat(LimitFactorString));

            /* set emergesort enable logging (true/false) */
            options.setEnableLogging(EnableLoggingBoolean);

            /* set emergesort log rate (1-default) */
            options.setLogRate(1);

            /* set emergesort Emerge Type */
            options.setEmergeType(getEmergeTypeFromString());
```

```
                /* set emergesort Selection Type */
                options.setSelAtomType(getAtomSelectionTypeFromString());

                /* set emergesort Wrap Type */
                options.setWrapType(getWrapTypeFromString());

                /* set Actions on set when selected atoms are sorted */
                options.setSortedSetAction(getSortedSetActionFromString());

                /* set Actions on set borders when selected atoms are sorted */

        options.setSortedSetBorderAction(getSortedSetBordersActionFromString());

                /* set Actions on unsorted set borders when selected atoms are sorted
*/
                options.setSetBorderAction(getUnSortedSetBordersActionFromString());

                /* set Sorter options */
                sorter.setSampleSize(Integer.parseInt(SamplesString)); // Number of
Samples
                sorter.setDiameter(Integer.parseInt(DiameterString)); // Size of
neighbourhood
                sorter.setBase(2); // Base of the population size
                sorter.setPowerMin(Integer.parseInt(ExponentString) - 2); // Exponent
of the population size
                sorter.setPowerMax(Integer.parseInt(ExponentToString) - 2); //
Exponent of the population size
                sorter.setEmergeOptions(options);

                /* Start sorting */
                sorter.Sort();

                /*Set information message to empty when simulation is finished*/
                Messages.setText("");

        }/*end of runSimulation*/

        /**
         * This function fills the statistics pane with the simulation statisticss
         *
         */
        private void printSimulationStatistics(){


                PopulationGroup popGroup;
                Stats st;


           /*initialize Statistics pane */
                Object rows[][] = null;
                Object headersRounds[] = { "Index", "Pop.\nSize", "Avg.\nROUNDS",
                                "Min.\nROUNDS", "Max.\nROUNDS", "Total.\nROUNDS"};
                Object headersMoves[] = { "Index", "Pop.\nSize",
                                "Avg.\nMOVES", "Min.\nMOVES", "Max.\nMOVES",
"Total.\nMOVES"};
                Object headersDistance[] = { "Index", "Pop.\nSize",
                                "Avg.\nDISTANCE", "Min.\nDISTANCE",
"Max.\nDISTANCE","Total.\nDISTANCE"};
                Object headersOrder[] = { "Index", "Pop.\nSize","Ascending ORDER",
"Descending ORDER" };

                statisticsLabel.setText("Statistics");
                statisticsPane = new JTabbedPane(JTabbedPane.TOP);
                statisticsPane.setBounds(31, 458, 732, 138);
                getContentPane().add(statisticsPane);


                /*table models initialized, one for each one of the 4 tabs*/
                DefaultTableModel modelRounds = new DefaultTableModel(rows,
headersRounds) {
                        private static final long serialVersionUID = 1L;
```

```
                @Override
                public boolean isCellEditable(int row, int column) {
                        // all cells not editable
                        return false;
                }
        };

        DefaultTableModel modelMoves = new DefaultTableModel(rows,
headersMoves) {
                private static final long serialVersionUID = 1L;
                @Override
                public boolean isCellEditable(int row, int column) {
                        // all cells not editable
                        return false;
                }
        };

        DefaultTableModel modelDistance = new DefaultTableModel(rows,
headersDistance) {
                private static final long serialVersionUID = 1L;
                @Override
                public boolean isCellEditable(int row, int column) {
                        // all cells not editable
                        return false;
                }
        };

        DefaultTableModel modelOrder = new DefaultTableModel(rows,
headersOrder) {
                private static final long serialVersionUID = 1L;

                @Override
                public boolean isCellEditable(int row, int column) {
                        // all cells not editable
                        return false;
                }
        };

        /*tables are created, one for each one of the 4 tabs*/
        tableRounds = new JTable(modelRounds);
        tableMoves = new JTable(modelMoves);
        tableDistance = new JTable(modelDistance);
        tableOrder = new JTable(modelOrder);

        /*modified cell rendered selected in order to format row colour when
needed*/
        MyStatisticsTableCellRenderer renderer=new
MyStatisticsTableCellRenderer();

        /*pass the samples variable to the cell renderer*/
        renderer.samples=Integer.parseInt(SamplesString);

        /*set cell rendered for each table to MyStatisticsTableCellRenderer*/
        tableRounds.setDefaultRenderer(Object.class, renderer);
        tableMoves.setDefaultRenderer(Object.class, renderer);
        tableDistance.setDefaultRenderer(Object.class, renderer);
        tableOrder.setDefaultRenderer(Object.class, renderer);

        /*add a scrollpane to each table*/
        JScrollPane scrollPaneRounds = new JScrollPane(tableRounds);
        tableRounds.setFillsViewportHeight(true);
        JScrollPane scrollPaneMoves = new JScrollPane(tableMoves);
        tableMoves.setFillsViewportHeight(true);
        JScrollPane scrollPaneDistance = new JScrollPane(tableDistance);
        tableDistance.setFillsViewportHeight(true);
        JScrollPane scrollPaneOrder = new JScrollPane(tableOrder);
        tableOrder.setFillsViewportHeight(true);

        /*add each table to the respective tab*/
        statisticsPane.addTab("Rounds", null, scrollPaneRounds, null);
        statisticsPane.addTab("Moves", null, scrollPaneMoves, null);
```

```
            statisticsPane.addTab("Distance", null, scrollPaneDistance, null);
            statisticsPane.addTab("Order", null, scrollPaneOrder, null);


        for (int i = 0; i < sorter.getPopulationGroups().size(); i++){

            /*for each population group print the aggeregated statistics*/
            popGroup = (PopulationGroup)sorter.getPopulationGroups().get(i);
            st = (Stats)popGroup.getGroupStats();
            modelRounds.insertRow(modelRounds.getRowCount(), new Object[] {
                    (i+1),
                    popGroup.getPopSize(),
                    st.getAvgRounds(),
                    st.getMinRounds(),
                    st.getMaxRounds(),
                    st.getRounds()
            });
            modelMoves.insertRow(modelMoves.getRowCount(), new Object[] {
                    (i+1),
                    popGroup.getPopSize(),
                    st.getAvgMoves(),
                    st.getMinMoves(),
                    st.getMaxMoves(),
                    st.getTotalMoves()
            });
            modelDistance.insertRow(modelDistance.getRowCount(), new Object[] {
                    (i+1),
                    popGroup.getPopSize(),
                    st.getAvgDistance(),
                    st.getMinDistance(),
                    st.getMaxDistance(),
                    st.getTotalDistance()
            });
            modelOrder.insertRow(modelOrder.getRowCount(), new Object[] {
                    (i+1),
                    popGroup.getPopSize(),
                    st.getSortedAscending(),
                    st.getSortedDescending(),
            });

            for (int j = 0; j < popGroup.getPopulations().size(); j++){
                /*for each population print its statistics*/
                st =
(Stats)((Population)popGroup.getPopulations().get(j)).getStats();
                modelRounds.insertRow(modelRounds.getRowCount(), new Object[]
{
                        (i+1)+"."+(j+1),
                        popGroup.getPopSize(),
                        st.getAvgRounds(),
                        st.getMinRounds(),
                        st.getMaxRounds(),
                        st.getRounds()
                });
                modelMoves.insertRow(modelMoves.getRowCount(), new Object[] {
                        (i+1)+"."+(j+1),
                        popGroup.getPopSize(),
                        st.getAvgMoves(),
                        st.getMinMoves(),
                        st.getMaxMoves(),
                        st.getTotalMoves()
                });
                modelDistance.insertRow(modelDistance.getRowCount(), new
Object[] {
                        (i+1)+"."+(j+1),
                        popGroup.getPopSize(),
                        st.getAvgDistance(),
                        st.getMinDistance(),
                        st.getMaxDistance(),
                        st.getTotalDistance()
                });
                modelOrder.insertRow(modelOrder.getRowCount(), new Object[] {
```

```
                                (i+1)+"."+(j+1),
                                    popGroup.getPopSize(),
                                    st.getSortedAscending(),
                                    st.getSortedDescending(),
                                    st.getMaxRounds(),
                                    st.getRounds()
                            });

                    }
                }

            }/*end of printSimulationStatistics*/

            /**
             * This function "prints"-> fills the log pane with simulation log
             *
             */
        private void printSimulationLog(){

                    Population pop ;
                    PopulationGroup popGroup;
                    PopulationLog log;

                /*initialize Log pane */
                    Object rows[][] = null;
                    Object headersLog[] = { "Size", "Group idx", "Pop. idx",
                                "Log idx", "Pos.", "Output"};
                    Object headersParameters[] = { "Parameter", "Value"};

                    logLabel.setText("Log");
                    logPane = new JTabbedPane(JTabbedPane.TOP);
                    logPane.setBounds(31, 641, 732, 338);
                    getContentPane().add(logPane);

                    /*table models initialized, one for each one of the 4 tabs*/
                    DefaultTableModel modelLog = new DefaultTableModel(rows, headersLog) {
                           private static final long serialVersionUID = 1L;
                           @Override
                           public boolean isCellEditable(int row, int column) {
                                  // all cells not editable
                                  return false;
                           }
                    };

                    DefaultTableModel modelParameters = new DefaultTableModel(rows,
headersParameters) {
                           private static final long serialVersionUID = 1L;
                           @Override
                           public boolean isCellEditable(int row, int column) {
                                  // all cells not editable
                                  return false;
                           }
                    };

                    /*create tables from the models*/
                    tableLog = new JTable(modelLog);
                    tableParameters = new JTable(modelParameters);

                    /*set cell renderer for Log table to MyLogTableCellRenderer*/
                    tableLog.setDefaultRenderer(Object.class, new
MyLogTableCellRenderer());

                    tableLog.setFillsViewportHeight(true);
                    tableParameters.setFillsViewportHeight(true);
                    tableLog.getColumnModel().getColumn(4).setPreferredWidth(80);
                    tableLog.getColumnModel().getColumn(5).setPreferredWidth(400);

                    /*add a scrollpane to each table*/
                    JScrollPane scrollPaneLog = new JScrollPane(tableLog);
                    JScrollPane scrollPaneParameters = new JScrollPane(tableParameters);
```

```
                /*add each table to the respective tab*/
                logPane.addTab("Log", null, scrollPaneLog, null);
                logPane.addTab("Parameters", null, scrollPaneParameters, null);

                for (int i = 0; i < sorter.getPopulationGroups().size(); i++){

                        /*for each population group find the populations */
                        popGroup = (PopulationGroup)sorter.getPopulationGroups().get(i);
                        for (int j = 0; j < popGroup.getPopulations().size(); j++){

                                /*for each population get the log */
                                pop = (Population)popGroup.getPopulations().get(j);
                                log = (PopulationLog)pop.getLog();
                                for (int k = 0; k < log.getFormattedValues().size(); k++)
                    {
                                        /*for each log entry print the related values */
                                        modelLog.insertRow(modelLog.getRowCount(), new
Object[] {
                                                pop.getSize(),
                                                (i+1),
                                                (j+1),
                                                (k+1),
                                                log.getFormattedHeaders().get(k),
                                        });
                                        modelLog.insertRow(modelLog.getRowCount(), new
Object[] {
                                                "","","","","Pop.
Values:",log.getFormattedValues().get(k).replaceAll("\\s+", ",")});
                                        modelLog.insertRow(modelLog.getRowCount(), new
Object[] {

        "","","","","Properties:",log.getFormattedDirections().get(k).replaceAll("\\s
+", ",")});
                                        modelLog.insertRow(modelLog.getRowCount(), new
Object[] {
                                                "","","","","Area
Moves:",log.getFormattedAreaMoves().get(k).replaceAll("\\s+", ",")});
                                        modelLog.insertRow(modelLog.getRowCount(), new
Object[] {

        "","","","","Moves:",log.getFormattedMoves().get(k).replaceAll("\\s+",
",")});
                                        modelLog.insertRow(modelLog.getRowCount(), new
Object[] {

        "","","","","Distances:",log.getFormattedDistances().get(k).replaceAll("\\s+"
, ",")});


                    }
                        }
                }

                /*fill the Parameters table*/
                modelParameters.insertRow(modelParameters.getRowCount(), new Object[]
{
                        "Protocol",ProtocolString
                });
                modelParameters.insertRow(modelParameters.getRowCount(), new Object[]
{
                        "Selection Type",SelectionTypeString
                });
                modelParameters.insertRow(modelParameters.getRowCount(), new Object[]
{
                        "Last Sorted Bound",LastSortedBoundBoolean
                });
                modelParameters.insertRow(modelParameters.getRowCount(), new Object[]
{
                        "Use Probability Table",UseProbabilityTableBoolean
                });
```

```
                modelParameters.insertRow(modelParameters.getRowCount(), new Object[]
{
                        "Action On Ends:",ActionOnEndsString
                });
                modelParameters.insertRow(modelParameters.getRowCount(), new Object[]
{
                        "Sorted Set",SortedSetString
                });
                modelParameters.insertRow(modelParameters.getRowCount(), new Object[]
{
                        "Sorted Set Borders",SortedSetBordersString
                });
                modelParameters.insertRow(modelParameters.getRowCount(), new Object[]
{
                        "New Set Borders",NewSetBordersString
                });
                modelParameters.insertRow(modelParameters.getRowCount(), new Object[]
{
                        "Limit Factor",LimitFactorString
                });
                modelParameters.insertRow(modelParameters.getRowCount(), new Object[]
{
                        "Exponent From",ExponentString
                });
                modelParameters.insertRow(modelParameters.getRowCount(), new Object[]
{
                        "Exponent To",ExponentToString
                });
                modelParameters.insertRow(modelParameters.getRowCount(), new Object[]
{
                        "Samples",SamplesString
                });
                modelParameters.insertRow(modelParameters.getRowCount(), new Object[]
{
                        "Diameter",DiameterString
                });
                modelParameters.insertRow(modelParameters.getRowCount(), new Object[]
{
                        "Enable Logging",EnableLoggingBoolean
                });

                endTime=System.currentTimeMillis();
                Messages.setText("Simulation finished. Execution time:"+(endTime-
startTime)+" milliseconds");
                /*end of printSimulationLog*/

        }


        /**
         * This function reads String variable ProtocolString and returns the
         * appropriate Enm.EmergeType
         */
        private Enm.EmergeType getEmergeTypeFromString() {
                if (ProtocolString.equals("L/R Protocol")) {
                        return Enm.EmergeType.LRProtocol;
                } else if (ProtocolString.equals("A/D Protocol (By Position Moves)"))
{
                        return Enm.EmergeType.ADProtocolP;
                } else if (ProtocolString.equals("A/D Protocol (By Number Moves)")) {
                        return Enm.EmergeType.ADProtocolM;
                } else {
                        /* warning printed in server log if invalid parameter is passed
*/
                        /* default Enm type assigned */
                        System.out.println("Warning:Invalid EmergeType, set to
default");
                        return Enm.EmergeType.ADProtocolP;
                }
```

```
	}

	/**
	 * This function reads String variable SelectionTypeString and returns the
	 * appropriate Enm.SelAtomType
	 */
	private Enm.SelAtomType getAtomSelectionTypeFromString() {
		if (SelectionTypeString.equals("Serial")) {
			if (!UseProbabilityTableBoolean && !LastSortedBoundBoolean)
				return Enm.SelAtomType.Serial;
			if (!UseProbabilityTableBoolean && LastSortedBoundBoolean)
				return Enm.SelAtomType.SemiSerial;
			if (UseProbabilityTableBoolean && LastSortedBoundBoolean)
				return Enm.SelAtomType.SemiSerialProb;
			if (UseProbabilityTableBoolean && !LastSortedBoundBoolean)
				return Enm.SelAtomType.SerialProb;
		} else {
			if (!UseProbabilityTableBoolean && !LastSortedBoundBoolean)
				return Enm.SelAtomType.RandomUni;

			if (!UseProbabilityTableBoolean && LastSortedBoundBoolean)
				return Enm.SelAtomType.SemiRandom;

			if (UseProbabilityTableBoolean && !LastSortedBoundBoolean)
				return Enm.SelAtomType.RandomProb;

			if (UseProbabilityTableBoolean && LastSortedBoundBoolean)
				return Enm.SelAtomType.SemiRandomProb;

		}
		/* warning printed in server log if invalid parameter is passed */
		/* default Enm type assigned */
		System.out
				.println("Warning:Invalid AtomSelectionType, set to
default(Random)");
		return Enm.SelAtomType.RandomUni;
	}

	/**
	 * This function reads String variable ActionOnEndsString and returns the
	 * appropriate Enm.WrapType
	 */
	private Enm.WrapType getWrapTypeFromString() {
		if (ActionOnEndsString.equals("Wrap")) {
			return Enm.WrapType.Wrap;
		} else if (ActionOnEndsString.equals("Inflect")) {
			return Enm.WrapType.Inflect;
		} else if (ActionOnEndsString.equals("ExcludeEnds")) {
			return Enm.WrapType.NoEnds;
		} else {
			/* warning printed in server log if invalid parameter is passed
*/
			/* default Enm type assigned */
			System.out
					.println("Warning:Invalid WrapType, set to
default(Wrap)");
			return Enm.WrapType.Wrap;
		}

	}

	/**
	 * This function reads String variable SortedSetString and returns the
	 * appropriate Enm.SortedSetAction
	 */
	private Enm.SortedSetAction getSortedSetActionFromString() {
		if (SortedSetString.equals("No Action")) {
			return Enm.SortedSetAction.NoAction;
		}

		else if (SortedSetString.equals("Set Properties")) {
```

```
                return Enm.SortedSetAction.SetFlags;
            }

            else if (SortedSetString.equals("Reset Properties")) {
                return Enm.SortedSetAction.ResetFlags;
            }

            else if (SortedSetString.equals("Scale Properties")) {
                return Enm.SortedSetAction.ByNumber;
            } else {
                /* warning printed in server log if invalid parameter is passed
*/
                /* default Enm type assigned */
                System.out
                        .println("Warning:Invalid SortedSetAction, set to
default(NoAction)");
                return Enm.SortedSetAction.NoAction;
            }

        }

        /**
         * This function reads String variable SortedSetBordersString and returns
         * the appropriate Enm.SortedSetBorderAction
         */
        private Enm.SortedSetBorderAction getSortedSetBordersActionFromString() {
            if (SortedSetBordersString.equals("No Action")) {
                return Enm.SortedSetBorderAction.NoAction;
            } else if (SortedSetBordersString.equals("Set Properties")) {
                return Enm.SortedSetBorderAction.SetFlags;
            }

            else if (SortedSetBordersString.equals("Reset Properties")) {
                return Enm.SortedSetBorderAction.ResetFlags;
            }

            else if (SortedSetBordersString.equals("Scale Properties")) {
                return Enm.SortedSetBorderAction.ScaleFlags;
            } else {
                /* warning printed in server log if invalid parameter is passed
*/
                /* default Enm type assigned */
                System.out.println("Warning:Invalid SortedSetBordersAction, set
to default(NoAction)");
                return Enm.SortedSetBorderAction.NoAction;
            }

        }

        /**
         * This function reads String variable NewSetBordersString and returns the
         * appropriate Enm.SetBorderAction
         */
        private Enm.SetBorderAction getUnSortedSetBordersActionFromString() {
            if (NewSetBordersString.equals("No Action")) {
                return Enm.SetBorderAction.NoAction;
            } else if (NewSetBordersString.equals("Set Properties")) {
                return Enm.SetBorderAction.SetFlags;
            }

            else if (NewSetBordersString.equals("Reset Properties")) {
                return Enm.SetBorderAction.ResetFlags;
            }

            else if (NewSetBordersString.equals("Scale Properties")) {
                return Enm.SetBorderAction.ScaleFlags;
            } else {
                /* warning printed in server log if invalid parameter is passed
*/
                /* default Enm type assigned */
```

```
                    System.out.println("Warning:Invalid UnSortedSetBordersAction,
set to default(NoAction)");
                    return Enm.SetBorderAction.NoAction;
                }

        }

}
```

Γ.3 Ο κώδικας του Java Servlet runECJ

```
package emergesort;

import java.io.File;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.OutputStream;
import java.io.PrintStream;
import java.io.PrintWriter;
import java.util.Iterator;
import java.util.LinkedList;
import java.util.List;

import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import org.apache.tomcat.util.http.fileupload.FileItem;
import org.apache.tomcat.util.http.fileupload.FileUploadException;
import org.apache.tomcat.util.http.fileupload.disk.DiskFileItemFactory;
import org.apache.tomcat.util.http.fileupload.servlet.ServletFileUpload;

import ECJRuntime.TestRun;

/**
 * Servlet implementation class runECJ
 */
@WebServlet("/runECJ")
public class runECJ extends HttpServlet {
       private static final long serialVersionUID = 1L;
       static int i;
       static int MAXOUTPUTFILES=1000;
       static LinkedList<File> outputFiles;
       private PrintWriter out;
    /**
     * @see HttpServlet#HttpServlet()
     */
    public runECJ() {
        super();
        i=1;
        outputFiles=new LinkedList<File>();
    }

       /**
        * @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse
response)
        */
       protected void doGet(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
            doPost(request,response);
       }

       /**
        * @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse
response)
        */
       protected void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {


            /*set the response's content type*/
            response.setContentType("text/html");
```

```
            /*assign to out variable, the response's output writer*/
            /*from now on, everything written on out, will be sent as response to
the calling browser*/
            out=response.getWriter();

            /*call the function that sets up the HTML page that is sent as response*/
            printPageStart();

            /*call the function that parses the request and initializes the
emergesort simulation parameters*/
            runGeneticAlgorithm(request);

            printPageEnd();


        }

        /**
         * This function prints to the response HTML tags that initialize the HTML
page
         * It also prints the page title and main header
         */
        private void printPageStart(){
            String title = "Emerge-Sort @ WWW";
            out.println("<!DOCTYPE HTML PUBLIC \"-//W3C//DTD HTML 4.0
Transitional//EN\">\n" +
                        "<HTML>\n" +
                        "<HEAD><TITLE>" + title + "</TITLE></HEAD>\n"+
                                "<body bgcolor=\"#C0C0C0\">\n" +
                                "<table  width=\"100%\"\n" +
                                "<tr bgcolor=\"#A0A0A0\">\n" +
                                "<td>\n" +
                                "<h1 ALIGN=\"CENTER\">Emerge-Sort @ WWW</h1>\n" +
                                "</td>\n" +
                                "</tr>\n" +
                                "</table>\n" +

                                "<table width=\"100%\" align=\"center\"
bgcolor=\"#C0C0C0\">\n" +
                                "<tr >\n" +

                                "<td bgcolor=\"#B0B0B0\">\n" +
                                "<center>\n" +
                                "<a href=\"./index.html\">Home Page</a>\n" +
                                "</center>\n" +
                                "</td>\n" +

                                "<td bgcolor=\"#B0B0B0\">\n" +
                                "<center>\n" +
                                "<a href=\"./applet.html\">Emerge-Sort Applet</a> \n" +
                                "</center>\n" +
                                "</td>\n" +

                                "<td bgcolor=\"#B0B0B0\">\n" +
                                "<center>\n" +
                                "<a href=\"./genetic.html\">Genetic Algorithm</a> \n" +
                                "</center>\n" +
                                "</td>\n" +

                                "<td bgcolor=\"#B0B0B0\">\n" +
                                "<center>\n" +
                                "<a href=\"./appletECJ.html\">Genetic Algorithm
Applet</a> \n" +
                                "</center>\n" +
                                "</td>\n" +

                                "<td bgcolor=\"#B0B0B0\">\n" +
                                "<center>\n" +
                                "<a href=\"./about.html\">About Emerge-Sort</a> \n" +
                                "</center>\n" +
```

```
                        "</td>\n" +

                        "<td bgcolor=\"#B0B0B0\">\n" +
                        "<center>\n" +
                        "<a href=\"./help.html\">Help</a>\n" +
                        "</center>\n" +
                        "</td>\n" +

                        "<td bgcolor=\"#A0A0A0\">\n" +
                        "<center>\n" +
                        "<a href=\"#\">Results</a>\n" +
                        "</center>\n" +
                        "</td>\n" +

                        "</tr>\n" +
                        "</table>\n" +
                        "<hr>\n");
}

/**
 * This function prints to the response HTML tags that end the HTML page
 */
private void printPageEnd(){
    out.println("\n</body></html>");
}

/**
 * This function gets the simulation parameters from the request
 */
private void runGeneticAlgorithm(HttpServletRequest request){
      boolean fileFound=false;
    File tempDirectory=null;
       FileOutputStream fileOut=null;
       PrintStream p=null;
    List<FileItem> items=null;
    DiskFileItemFactory factory=null;
    ServletFileUpload upload=null;
    String paramFilename=null;
    String subFilename=null;
    File newParamFile=null;
    String outputFilename=null;
    String outputFilenameFull=null;
    String finalOutputFilename=null;
    boolean filenameChanged=false;

    tempDirectory=new File(getServletContext().getRealPath("/temp/"));

       // Create a factory for disk-based file items
    factory = new DiskFileItemFactory();

    // Set factory constraints
    factory.setSizeThreshold(100000);
    factory.setRepository(tempDirectory);

    // Create a new file upload handler
    upload= new ServletFileUpload(factory);

    // Set overall request size constraint
    upload.setSizeMax(100000);

    // Parse the request
    try {
              items= upload.parseRequest(request);
             // Process the uploaded items
             Iterator iter = items.iterator();
             while (iter.hasNext()) {
                    FileItem item = (FileItem) iter.next();
                if (!item.isFormField()) {
                    fileFound=true;
                    if(item.getSize()==0){
```

```
                                    out.println("<center>You did not upload a file or
file was empty!</center>");
                                    out.println("<center>Default parameter file is
going to be used for the execution</center>");

       paramFilename=getServletContext().getRealPath("/emerge.params");
                                }
                                else{
                                    out.println("<center>You successfully uploaded
file:"+item.getName()+" with size: "+item.getSize()+" bytes </center>");
                                    subFilename="/emerge"+i+".params";

       paramFilename=getServletContext().getRealPath(subFilename);
                                    newParamFile=new File(paramFilename);
                                    try {
                                            item.write(newParamFile);
                                        } catch (Exception e) {
                                            out.println("<center>Parameter file
could not be stored!!!</center>");
                                        }
                                }
                            }
                            else{
                                String name = item.getFieldName();
                                String value = item.getString();
                                if((name !=null) &&(name.compareTo("OutName")==0)){
                                    outputFilename=value;
                                }
                            }

                        }
                        if(!fileFound){
                            out.println("<center>You did not upload a
file!</center>");
                            out.println("<center>Default parameter file is going to
be used for the execution.</center>");
                        }
                        if((outputFilename==null)||outputFilename.isEmpty()){
                            outputFilename="ECJOutput";
                            out.println("<center>You did not define the name of the
output file</center>");
                            out.println("<center>Default output
filename(ECJOutputxxx.txt) is going to be used.</center>");
                        }
                        try {

       outputFilenameFull=getServletContext().getRealPath("/"+outputFilename+".txt")
;
                            File outputFile=new File(outputFilenameFull);
                            i=0;
                            finalOutputFilename=outputFilename+".txt";
                            while(outputFile.exists()){
                                filenameChanged=true;
                                outputFile=null;

       outputFilenameFull=getServletContext().getRealPath("/"+outputFilename+i+".txt
");
                                outputFile=new File(outputFilenameFull);
                                finalOutputFilename=outputFilename+i+".txt";
                                i++;
                            }

       if((filenameChanged)&&(!finalOutputFilename.contains("ECJOutput"))){
                                out.println("<center>Another output file with the
name that you defined already existed.</center>");
                                out.println("<center>Output filename was
changed</center>");
                            }
                            outputFile=new File(outputFilenameFull);
                            outputFiles.addLast(outputFile);
                            while(outputFiles.size()>MAXOUTPUTFILES){
```

```
                                    File fileToDelete=outputFiles.removeFirst();

                                    out.println("<center>Another (older) output file
("+fileToDelete.getName()+") was deleted.</center>");
                                    fileToDelete.delete();
                                }
                        } catch (Exception e1) {
                                out.println("<center>Error!!! could not create output
file!</center>");
                                out.println("<center>Please contact the
administrator</center>");
                        }
                    //p= new PrintStream( fileOut );
                        String args[]={paramFilename,outputFilenameFull};
                        TestRun tr = new TestRun(args);
                        //System.setOut(p);
                        //System.setErr(p);
                        out.println("<br><br><center>Genetic algorithm execution has
just started</center>");
                        out.println("<b><center>Please kindly wait for a few minutes and
download results from <a href=\"./"+finalOutputFilename+"\"
target=\"_blank\">"+finalOutputFilename+"</a></b></center>");
                        out.println("<br><center>Results may take a while depending on
the current server load.</center>");
                        out.println("<center>Genetic algorithm execution is finished if
END OF GENETIC ALGORITHM EXECUTION is printed at the end of the output
file.</center>");
                        out.println("<center>You may open <a
href=\"./"+finalOutputFilename+"\" target=\"_blank\">"+finalOutputFilename+"</a> in
a new page and refresh it frequently(press F5).</center>");
                        out.println("<center><font color=\"red\">CAUTION: If F5 is
pressed on this page (Results), a new Genetic Algorithm Evaluation will be
started.</font></center>");

                } catch (FileUploadException e) {
                        out.println("<center>An error occured during file
upload!!!</center>");
                }
            i++;
            if(i>=(MAXOUTPUTFILES+1)) i=1;
        }


}
```

Γ.4 Ο κώδικας του Java Applet ECJApplet

```
package ecjapplet;

import java.awt.Color;
import java.awt.Dimension;
import java.awt.Font;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.MouseAdapter;
import java.awt.event.MouseEvent;
import java.io.File;
import java.io.IOException;
import java.io.OutputStream;
import java.io.PrintStream;

import javax.swing.JApplet;
import javax.swing.JButton;
import javax.swing.JFileChooser;
import javax.swing.JLabel;
import javax.swing.JScrollPane;
import javax.swing.SwingConstants;
import javax.swing.SwingUtilities;

import ECJRuntime.TestRun;
import javax.swing.JTextArea;

public class ECJApplet extends JApplet {
	/**
	 *
	 */
	private static final long serialVersionUID = 1L;
	private JTextArea textArea;
	JButton btnRunGenetic;
	JButton btnChooseParameterFiles;
	JFileChooser fc;
	File directory;


	/*global gui variables*/
	JLabel Messages,Messages2;
	private PrintStream stdout;
	private PrintStream stderr;

	/**
	 * Create the applet.
	 * This function actually prepares the applet GUI
	 */
	public ECJApplet() {

		/*GUI setup*/

		/*Setup the applet window*/
		getContentPane().setBackground(Color.LIGHT_GRAY);
		setSize(new Dimension(800, 500));
		getContentPane().setLayout(null);

		/*Setup */
		Messages= new JLabel("Please choose the directory where parameter
files are stored");//Empty-at start- label used for execution information messages
		Messages.setHorizontalAlignment(SwingConstants.CENTER);
		Messages.setBounds(10, 11, 780, 16);
		Messages.setForeground(Color.BLACK);
		getContentPane().add(Messages);

		Messages2= new JLabel("");//Empty-at start- label used for execution
information messages
```

```
                Messages2.setHorizontalAlignment(SwingConstants.CENTER);
                Messages2.setBounds(10, 30, 780, 16);
                Messages2.setForeground(Color.BLACK);
                getContentPane().add(Messages2);

                fc = new JFileChooser();
                fc.setFileSelectionMode( JFileChooser.DIRECTORIES_ONLY);

                /*Run button is created*/
                btnRunGenetic = new JButton("Run Genetic Algorithm");
                btnRunGenetic.addActionListener(new ActionListener() {
                      public void actionPerformed(ActionEvent arg0) {
                            runGenetic();
                       }
                });
                btnRunGenetic.setFont(new Font("Segoe UI", Font.PLAIN, 14));
                btnRunGenetic.setBounds(269, 82, 254, 25);
                getContentPane().add(btnRunGenetic);
                btnRunGenetic.setEnabled(false);

                textArea = new JTextArea();
                textArea.setWrapStyleWord(true);
                JScrollPane jScrollPane1 = new JScrollPane(textArea);
                jScrollPane1.setBounds(80, 120, 642, 350);
                getContentPane().add(jScrollPane1);


                btnChooseParameterFiles = new JButton("Choose Parameter Files'
Directory");
                btnChooseParameterFiles.addActionListener(new ActionListener() {
                      public void actionPerformed(ActionEvent e) {
                            File testFile;
                            int error=0;
                            String failedFiles="";
                            int returnVal = fc.showOpenDialog(ECJApplet.this);

                     if (returnVal == JFileChooser.APPROVE_OPTION) {
                         directory = fc.getSelectedFile();
                            testFile=new File(directory,"emerge.params");
                            if(!testFile.exists()){
                             error++;
                             failedFiles+=" emerge.params,";
                            }
                            testFile=null;
                            testFile=new File(directory,"simple.params");
                            if(!testFile.exists()){
                             error++;
                             failedFiles+=" simple.params,";
                            }
                            testFile=null;
                            testFile=new File(directory,"ec.params");
                            if(!testFile.exists()){
                             error++;
                             failedFiles+=" ec.params,";
                            }
                            testFile=null;
                            if(!failedFiles.isEmpty()){
                             failedFiles=failedFiles.substring(0,
failedFiles.length()-1);
                            }

                            if(error>0){
                             Messages.setForeground(Color.RED);
                             Messages2.setForeground(Color.RED);
                             if(error==1){
                                    Messages.setText("ERROR:Please choose the correct
directory!File "+failedFiles+" doesn't exist in the directory:");
                              }
                             else{
                                    Messages.setText("ERROR:Please choose the correct
directory!Files "+failedFiles+" don't exist in the directory:");
```

```
                        }
                        Messages2.setText(directory.getPath());
                        btnRunGenetic.setEnabled(false);
                       }
                       else{
                        Messages.setForeground(Color.BLACK);
                        Messages2.setForeground(Color.BLACK);
                        Messages.setText("You may start the Genetic algorithm.
Parameter files' directory:");
                        Messages2.setText(directory.getPath());
                        btnRunGenetic.setEnabled(true);
                       }
                   }



                    }
            });
            btnChooseParameterFiles.setFont(new Font("Segoe UI", Font.PLAIN, 14));
            btnChooseParameterFiles.setBounds(269, 57, 254, 25);
            getContentPane().add(btnChooseParameterFiles);

      }/*end of emergeSortApplet constructor*/

      /**
       *This function contains the main functionality of the applet,
       *ie running the simulation and printing the results
       */
      private void runGenetic() {

       Messages.setForeground(Color.RED);
            Messages.setText("Please Wait...Genetic algorithm is Running...");
            Messages2.setText("");
            btnRunGenetic.setEnabled(false);
            btnChooseParameterFiles.setEnabled(false);
            String args[]={""+directory.getPath()+"\\emerge.params"};
            redirectSystemStreams();
            TestRun tr = new TestRun(args);


      }/*end of runGenetic*/

       private void updateTextArea(final String text) {
                 SwingUtilities.invokeLater(new Runnable() {


                    public void run() {
                      textArea.append(text);
                      if(text.contains("END OF")){
                            Messages.setForeground(Color.BLACK);
                            Messages.setText("Genetic Algorithm Execution has
finished!");
                            Messages2.setForeground(Color.BLACK);
                            Messages2.setText("Current parameter files'
directory:"+directory);
                            btnRunGenetic.setEnabled(true);
                            btnChooseParameterFiles.setEnabled(true);

                      }
                    }
                  });
                }

                private void redirectSystemStreams() {
                  OutputStream out = new OutputStream() {
                    @Override
                    public void write(int b) throws IOException {
                      updateTextArea(String.valueOf((char) b));
                    }

                    @Override
```

```
                public void write(byte[] b, int off, int len) throws IOException
{
                  updateTextArea(new String(b, off, len));
                }

                @Override
                public void write(byte[] b) throws IOException {
                  write(b, 0, b.length);
                }
              };

              stdout=System.out;
              stderr=System.err;
              System.setOut(new PrintStream(out, true));
              System.setErr(new PrintStream(out, true));
            }

            private void resetSystemStreams() {
                   System.setOut(stdout);
                   System.setErr(stderr);
            }
}
```