ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΥΠΡΟΥ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

ΣΧΕΔΙΟ ΠΑΡΟΥΣΙΑΣΗΣ

ΑΤΟΜΙΚΗΣ ΔΙΠΛΩΜΑΤΙΚΗΣ ΕΡΓΑΣΙΑΣ

15/5/2015

Μάιος 2015

Ατομική Διπλωματική Εργασία

NGS (NEXT-GENERATION SEQUENCING)

DATA ANALYSES PLATFORM

Χρίστα Φιλίππου

ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΥΠΡΟΥ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Μάιος 2015

ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΥΠΡΟΥ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Τίτλος Ατομικής Διπλωματικής Εργασίας

Χρίστα Φιλίππου

Επιβλέπων Καθηγητής

Δρ.Κωνσταντίνος Παττίχης

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

Μάιος 2015

ΕΥΧΑΡΙΣΤΙΕΣ

Αρχικά θα ήθελα να εκφράσω τις θερμές μου ευχαριστίες προς τον καθηγητή του Τμήματος Πληροφορικής του Πανεπιστημίου Κύπρου και επιβλέπων καθηγητή Δρ. Κωνσταντίνο Παττίχη για τη βοήθεια που μου παρείχε κατά τη διάρκεια εκπόνησης της διπλωματικής εργασίας. Η καθοδήγηση του και οι συμβουλές του έπαιξαν καθοριστικό ρόλο για την ολοκλήρωση της εργασίας αυτής.

Επίσης θα ήθελα να ευχαριστήσω εκ βάθους καρδίας τον Δρ. Άθω Αντωνιάδη για την πολύτιμη και συνεχή υποστήριξη του. Μου πρόσφερε γνώσεις γύρω από πολλά θέματα των κλάδων της Πληροφορικής και της Βιολογίας και μου έδωσε την ευκαιρία και το έναυσμα για να τα ερευνήσω και να τα κατανοήσω. Η αγαστή συνεργασία μας μου έδωσε τα εχέγγυα τόσο για την ολοκλήρωση της εργασίας αυτής, αλλά και της απόκτησης γνώσεων και εμπειριών σε πιο ευρύ πεδίο.

Επιπρόσθετα θα ήθελα να ευχαριστήσω το Cyprus Institute για την παροχή χώρου στο Cy-Tera server για την εκτέλεση των αλγορίθμων που εξέτασα στα πλαίσια της διατριβής, καθώς και τους υπεύθυνους που παρείχαν έγκαιρη βοήθεια σε προβλήματα που αντιμετώπιζα.

Τέλος, θα ήθελα να ευχαριστήσω, την οικογένεια μου και το φιλικό μου περιβάλλον για την ηθική συμπαράσταση που μου παρείχαν ώστε να δώσω το καλύτερο των δυνατοτήτων μου.

ΠΕΡΙΛΗΨΗ

Η ευθυγράμμιση γενετικών ακολουθιών, προκάλεσε το ενδιαφέρον για την μελέτη τους με στόχο την εύρεση πιθανών αιτιών, που προκαλούν διάφορες ασθένειες. Με τη δημιουργία μεθόδων ευθυγράμμισης γενετικών ακολουθιών, όπως Dot Matrix, BWA (Burrows-Wheeler Aligner), BFAST και Bowtie, επιδιώκεται η εύρεση καλύτερης αντιστοίχισης ζευγών (best-matching) ακολουθιών.

Οι μεθοδολογίες που υφίστανται παρουσιάζουν μερικά μειονεκτήματα, τα οποία αποτέλεσαν λόγο για τη πυροδότηση δημιουργίας μιας νέας πλατφόρμας. Το βασικότερο από τα μειονεκτήματα αυτών είναι η απώλεια ποιότητας/ το μεγάλο ποσοστό σφάλματος των ακολουθιών και η αδυναμία εφαρμογής τους χρησιμοποιώντας High Performance Computing (HPC) resources [45].

Στα πλαίσια της διπλωματικής αυτής, έγινε μια προσπάθεια δημιουργίας μίας καινοτόμας πλατφόρμας, αντί των αλγορίθμων, η οποία θα επιτρέπει τη δημιουργία εφαρμογών ανάλυσης γενετικών ακολουθιών που προέρχονται από αυτή τη νέα NGS (Next Generation Sequencing) data analyses πλατφόρμα, η οποία θα αναλυθεί στο κεφάλαιο 4 εκτενέστερα. Η πρωτοποριακή NGS πλατφόρμα, την οποία υλοποίησα, κάνει χρήση ενός νέου και εύκολα επεκτάσιμου aligner αλγορίθμου, ο οποίος βασίζεται στη μέθοδο Dot Matrix (θα επεξηγηθεί στο κεφάλαιο 3). Η ίδια αποτελείται από επιπρόσθετους αλγορίθμους, οι οποίοι ενισχύουν αυτόν τον aligner αλγόριθμο, όσον αφορά τη ανάλυση τεράστιου όγκου γενετικών δεδομένων και σύγκριση των αποτελεσμάτων αυτών, μετά την εφαρμογή του αλγορίθμου επεξεργασίας γενετικών ακολουθιών και συγκεκριμένα του αλγορίθμου ευθυγράμμισης.

Η ζητούμενη πλατφόρμα αντικαθιστά τα υφιστάμενα προγράμματα που κάνουν χρήση των ήδη υπάρχον αλγόριθμων ευθυγράμμισης (όπως ο BWA και parallel BWA), με βασικό στόχο την επίλυση νέων πολύπλοκων βιολογικών προβλημάτων και την υλοποίηση καλύτερων αλγορίθμων σε ένα περιβάλλον,

τέτοιο, που να επιτρέπει την εύκολη και γρήγορη εφαρμογή και σύγκριση αποτελεσμάτων.

Περιεχόμενα

Κεφάλαιο 1

Εισαγωγή

1.1 Μελέτη Θεωρητικού Υπόβαθρου

Σύμφωνα με έρευνα , η οποία έγινε από τον W. James Kent (Department of Biology and Center for Molecular Biology of RNA, University of California, Santa Cruz, Santa Cruz, California 95064, USA) [43] μελετείται ο αλγόριθμος ευθυγράμμισης γενετικών ακολουθιών BLAST και αναλύεται ο τρόπος υλοποίησης ενός νέου αλγορίθμου, ο οποίος ονομάστηκε «BLAT», ο οποίος στηρίζεται στο προηγούμενο, αλλά με κάποιες ουσιαστικές βελτιστοποιήσεις. Αυτός ο αλγόριθμος ιδανικά θα εξάλειφε μειονεκτήματα του BLAST αλγόριθμου σχετικά με το χρόνο εκτέλεσης και την απαιτούμενη μνήμη για το ταίριασμα ενός read πάνω σε ολόκληρο το γονιδίωμα.

Ο BLAST [20] κατά την ευθυγράμμιση ακολουθιών διατηρεί τη θέση του κάθε k-mer (k¼ 11 by default) της ακολουθίας που αναζητείται σε μία hash table δομή με key value το k-mer. Σαρώνει τις ακολουθίες που βρίσκονται στη βάση δεδομένων για εύρεση ακολουθιών με ακριβή επικάλυψη k -mer (k -mer matches ή seeds), αναζητώντας τον πίνακα κατακερματισμού. Ο BLAST [2], [3] εκτείνεται και ενώνει τις δύο k -mer επικαλυπτόμενες ακολουθίες και τις ραφινάρει χρησιμοποιώντας τον Smith–Waterman alignment [41], [17] .

Συγκεκριμένα, ο ήδη υπάρχον aligner αλγόριθμος BLAST απαιτεί αρκετά μεγάλο χρόνο εφαρμογής του για εύρεση των τμημάτων του DNA που ταιριάζουν με κάποιο συγκεκριμένο read, ενώ ο αλγόριθμος BLAT είναι μέχρι και 15 φορές γρηγορότερος, με βάσει τα αποτελέσματα της μελέτης. Αν βέβαια αναλογιστούμε, με βάσει τις πληροφορίες της πιο πάνω έρευνας, ότι για εφαρμογή του αλγόριθμου BLAT πάνω σε 7.51×10^9 βάσεις σε 1.33×10^7 reads χρειάζεται χρόνο 16,300 CPU hours, κατανοούμε ότι, αφενός είναι πρακτικά εφικτό να υλοποιηθεί αυτός ο αλγόριθμος στην NGS data analyses platform που ανέπτυξα, αφετέρου είναι αδύνατο να εφαρμοστεί σε High Performance Computing (HPC) machine όπως ο Cy-Tera server του Cyprus Institute που χρησιμοποιήθηκε για την εφαρμογή της στην NGS data analyses πλατφόρμας.

1.1.1 Cy-Tera server Cyprus Institute

Το Cy-Tera του Cyprus Institute χρησιμοποιήθηκε για την εφαρμογή των αλγορίθμων της στην NGS data analyses πλατφόρμας. Το Cy-Tera αποτελεί ένα project του Cyprus Institute με στόχο τη δημιουργία ενός μηχανισμού έρευνας, καθώς και υπολογιστών υψηλής επίδοσης (HPC), που υποστηρίζουν επιστημονικές εφαρμογές, με τη σχετική υποστήριξη των χρηστών και υπολογιστικών προγραμμάτων επιστημονικής έρευνας και εκπαίδευσης. Η εγκατάσταση Cy-Tera είναι η πρώτη εγκατάσταση HPC σε multi-TFLOPS στη Κύπρο, εξυπηρετώντας τις ανάγκες του Cyprus Institute και των εταίρων της για την έρευνα εφαρμογών σε πολλούς τομείς μεγάλου επιστημονικού ή/ και κοινωνικού ενδιαφέροντος. Παρέχει την αρχική χρηματοδότηση για την υποδομή του υπολογισμού με βάση το Κέντρο Επιστήμης και Τεχνολογίας Έρευνας (CaSToRC). Τα κύρια συστατικά του έργου Cy-Tera έχουν ως εξής: Μία Tier-2 εγκατάσταση HPC που βασίζεται σε πολλαπλή cores αρχιτεκτονική με επεκτάσιμη σχεδίαση με μέγιστη απόδοση, μεγαλύτερη από 20 TFLOPS και 100 TBYTES αποθήκευσης, για ανάπτυξη ερευνητικών δραστηριοτήτων στο Scientific Computing, δημιουργώντας έτσι την τοπική τεχνογνωσία σε νέες αρχιτεκτονικές, multithreading και άλλες εξειδικευμένες τεχνολογίες. Νέες αρχιτεκτονικές λύσεις

θα πρέπει να διερευνηθούν, συμπεριλαμβανομένων των GPUs και άλλα σχέδια, τα οποία μειώνουν τις απαιτήσεις ισχύος και ψύξης, ενώ προσφέρει πολύ υψηλότερες επιδόσεις στο κόστος μεγαλύτερης πολυπλοκότητας προγραμματισμού. Περιβάλλοντα παράλληλου προγραμματισμού και εργαλεία λογισμικού διευκολύνουν την ανάπτυξη της επεκτασιμότητας, δεδομένου ότι multi-core chips, με εκατοντάδες απλούς πυρήνες στο chip, χρησιμοποιούνται.

Συγκεκριμένα, ακολουθείται η εξής στρατηγική για την υλοποίηση της μεθόδου Near Perfect Matching:

Βάσει του τρόπου υλοποίησης του BLAT αλγόριθμου, παρατηρείται ότι επιδιώκεται το σχεδόν απόλυτο ταίριασμα μεταξύ ενός read με κάποια περιοχή ολόκληρου του γονιδιώματος («Near Perfect Matching» όπως ονομάστηκε από τους μελετητές).

Αν μία περιοχή ταιριάζει κατά 95% και πάνω με το συγκεκριμένο read, για το οποίο ψάχνουμε να βρούμε τη περιοχή του (ανθρώπινου) γονιδιώματος στην οποία ταιριάζει, τότε γίνεται αποδεκτό το ταίριασμα. Σε αντίθετη περίπτωση απορρίπτεται η συγκεκριμένη περιοχή, ως περιοχή ταιριάσματος, για αυτό το συγκεκριμένο read. Η πιο πάνω στρατηγική δίνει ένα ξεκάθαρο ταίριασμα με μέγιστο ανεκτό ποσοστό mismatching 5% (mismatching threshold).

Όμως, αφού βιολογικά γνωρίζουμε ότι σε ένα read είναι πολύ πιθανό να υπάρχει σημειακός πολυμορφισμός [30] σε επίπεδο ενός SNP (νουκλεοτιδίου) ή να έχουμε αρκετά SNPs που απουσιάζουν (missing data) [39], καταλήγουμε στο συμπέρασμα ότι με την εφαρμογή της πιο πάνω μεθοδολογίας χάνουμε πάρα πολλές πληροφορίες (βλέπε Εικόνα 2.6), αφού απορρίπτονται ταιριάσματα που θα έπρεπε να γίνουν αποδεκτά, για παράδειγμα σε περιπτώσει που υπάρχει πολυμορφισμός σε ποσοστό πάνω από το 5% των SNPs του συγκεκριμένου read (Ο ορισμός του πολυμορφισμού εξηγείται στο υποκεφάλαιο 2.5). Αναντίλεκτα, αυτό προκαλεί losing quality of sequence, πράγμα μη επιθυμητό για βιολογικά προβλήματα.

Μία άλλη παρατήρηση που έκανα μέσα από τη έρευνα της πτυχιακής μου είναι ότι, οι διάφοροι αλγόριθμοι που υπάρχουν, επιχειρούν να λύσουν **μεμονωμένα** κάποιο **συγκεκριμένο** πρόβλημα, με αποτέλεσμα τη μη επεκτασιμότητα και την ανέφικτη εφαρμογή τους σε άλλα παραπλήσια βιοιατρικά προβλήματα. Παραδείγματος χάριν, μελέτη που έγινε από τους Yi Wang, Henry C.M. Leung, S.M. Yiu και Francis Y.L. Chin το 2012 [47], ασχολείται με το γεγονός ότι οι **Next-generation sequencing (NGS)** τεχνολογίες που ασχολούνται με το προσδιορισμό αλληλουχίας του DNA χρησιμοποιούν σαν είσοδο δεδομένων ένα τεράστιο αριθμό reads από γονιδιώματα διαφόρων άγνωστων ειδών, καθιστώντας την ομαδοποίηση των reads ένα κρίσιμο βήμα και αδυνατώντας να τα διαχειριστούν. Συγκεκριμένα, αναπτύσσει ένα καινούριο εργαλείο, το «MetaCluster 4.0», το οποίο θα επιλύει προβλήματα που υπάρχουν στις μέχρι στιγμής ανεπτυγμένες μεθοδολογίες ομαδοποίησης και αποθήκευσης γωνιδιομάτων. Το MetaCluster 4.0 δύναται να διαιρεί τα reads εισόδου σε ομάδες, να εκτιμά τη κατανομή 4-μερές της κάθε ομάδας και να υπολογίζει τον αριθμό των ειδών των γονιδίων.

Καταλήγοντας, υπάρχουν αμέτρητες μεθοδολογίες που ασχολούνται με τέτοια θέματα, οι οποίες επιδιώκουν την μείωση μειονεκτημάτων των υφισταμένων, χωρίς ιδιαίτερη έμφαση στη ποιότητα αλληλουχίας που παράγεται και χωρίς να υπάρχει δυνατότητα τροποποίησής τους για βελτιστοποίηση.

1.2 Κίνητρο Διπλωματικής Εργασίας

Το μεγάλο ενδιαφέρον για την κατανόηση του τεράστιου όγκου πληροφοριών που προέρχονται από την μελέτη του γονιδιώματος των οργανισμών, οδήγησε στην ανάγκη για μελέτη των περιοχών του DNA για την εύρεση πιθανών αιτιών που προκαλούν διάφορες ασθένειες, όπως για παράδειγμα ο καρκίνος, ο οποίος μαστίζει στις μέρες μας.

Η ταχεία εξέλιξη της Next-generation sequencing (NGS) τεχνολογίας, που θέτει προκλήσεις στο τομέα της βιοπληροφορικής, αποτέλεσε βασικό κίνητρο της

διπλωματικής αυτής. Βάσει άρθρου που γράφτηκε και δημοσιεύτηκε από τους Jay Shendure και Hanlee Ji το 2008 [24], το θέμα που αφορά το quality of sequence, έχει γίνει μια περιοχή έντονου ενδιαφέροντος, δεδομένης της σχετικά χαμηλής ποιότητας των νέων πλατφόρμων επεξεργασίας αλληλουχιών. Συγκεκριμένα, οι περισσότερες μεθοδολογίες ευθυγράμμισης γονοτύπων έχουν καλύτερη απόδοση όσον αφορά τη ποιότητα παραγωγής ακολουθιών DNA σε μικρού μήκους reads, παρά σε μεγάλου μήκους, όπως παρατηρήθηκε. Σε περιπτώσεις μεγάλου μήκους reads παρουσιάζουν υψηλότερο ποσοστό σφάλματος. Τα πράγματα γίνονται ακόμη πιο δύσκολα όταν γίνεται χρήση σειριακών αλγορίθμων όπως ο BWA (Burrows Wheeler Aligner) [5], αντί παράλληλων όπως ο pBWA (parallel Burrows Wheeler Aligner) [8].

Τα μειονεκτήματα των ήδη υπάρχον πλατφόρμων και τα χαρακτηρίστηκα τους, τα οποία απείχαν αρκετά από τα επιθυμητά, όσον αφορά τα ποσοστό σφάλματος αποτελεσμάτων, το losing quality των ακολουθιών DNA που παράγουν, την επαναχρησιμοποίηση, αλλά και την επεκτασιμότητα τους για μελλοντικούς βιοιατρικούς σκοπούς καθόρισαν την επίτευξη της πτυχιακής αυτής.

1.3 Στόχος Διπλωματικής Εργασίας

Ο βασικός στόχος της διπλωματικής αυτής καθορίζεται από την υλοποίηση μίας νέας πρωτοποριακής NGS data analyses platform, η οποία θα αποτελείται από κάποιους αλγόριθμους διαχείρισης και ανάλυσης δεδομένων, η οποία να είναι εύκολα επαναχρησιμοποιήσιμη και επεκτάσιμη όσον αφορά μελλοντικές αλλαγές που θα βοηθήσουν σε διάφορα σοβαρά βιοϊατρικά και βιολογικά προβλήματα. Παράλληλα, η γρήγορη διαχείριση μεγάλου όγκου γενετικών δεδομένων (ακολουθιών) για την ανάλυση και επεξεργασία τους (scalability), μαζί με την αποφυγή του losing quality των γενετικών ακολουθιών, αποτελεί ένα άλλο σοβαρό σκοπό της ανάπτυξης της πλατφόρμας αυτής.

Σύμφωνα με την έρευνα που έκανα μέσω διαδικτύου, δεν υπάρχει, μέχρι σήμερα, κάποια πλατφόρμα, η οποία να επιτρέπει την επίλυση νέων, σύγχρονων και πολύπλοκων βιολογικών προβλημάτων, με χρήση καλύτερων ποιοτικά αλγόριθμων. Διάφοροι αλγόριθμοι που υπάρχουν, επιχειρούν να λύσουν μεμονωμένα κάποιο συγκεκριμένο βιολογικό πρόβλημα, χρησιμοποιώντας μεγάλο όγκο δεδομένων, με αποτέλεσμα τη μη επεκτασιμότητα, το μεγάλο απαιτούμενο χρόνο για την ανάλυσή τους και κατ' επέκταση την ανέφικτη εφαρμογή τους σε άλλα παραπλήσια βιοιατρικά προβλήματα με τη χρήση χρησιμοποιώντας High Performance Computing πόρων. Έτσι, αυτή η data analyses platform που ανέπτυξα έχει σαν μείζων στόχο:

✓ Να μπορεί να διαχειριστεί, γρήγορα, μεγάλα γενετικά δεδομένα εισόδου, για παράδειγμα διασπώντας τα σε μικρότερες ομάδες δεδομένων εισόδου.

✓ Να αποφεύγει το κίνδυνο για losing quality των ακολουθιών, που μπορεί να υπάρξει λόγω πολυμορφισμού, deletion, insertion [44] ή missing data [39] της DNA ακολουθίας.

✓ Να χειρίζεται, **το ίδιο αποδοτικά**, ακολουθίες (ανθρώπινου) γονιδιώματος άσχετα με το μέγεθος του μήκους τους.

✓ Να είναι γενικά μία εύρωστη, σε κώδικα, πλατφόρμα ώστε να επαναχρησιμοποιείται και να επεκτείνεται εύκολα και γρήγορα.

✓ Να υποστηρίζει την εύκολη και γρήγορη υλοποίηση νέων αλγορίθμων για προσαρμογή αυτής στην επίλυση άλλων ιατρικών προβλημάτων.

✓ Να υποστηρίζει τη σύγκριση αποτελεσμάτων που εξάγονται.

✓ Να εφαρμόζεται, χρησιμοποιώντας High Performance Computing (HPC) resources [45], για γρήγορη εφαρμογή της πλατφόρμας.

Κεφάλαιο 2

Βασικές Έννοιες Μοριακής Βιολογίας

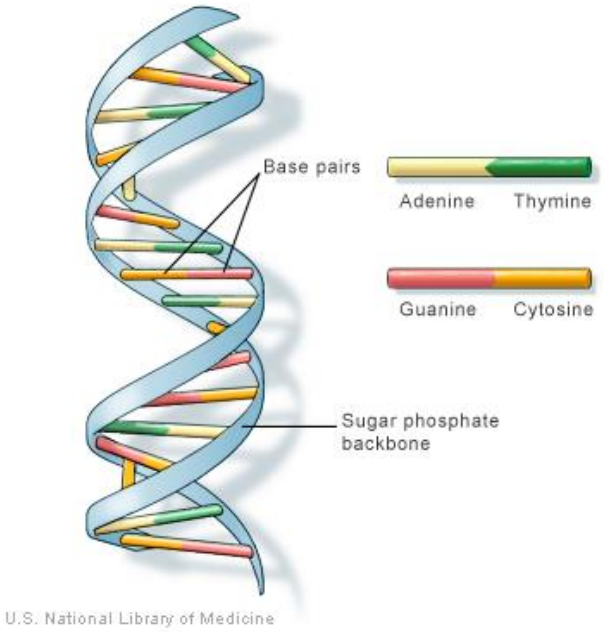
2.1 Δεσοξυριβοζονουκλεϊνικό οξύ

Το DNA (Δεσοξυριβοζονουκλεϊνικό οξύ) περιέχει όλες τις γενετικές πληροφορίες των περισσότερων οργανισμών και είναι διαφορετικό για κάθε άνθρωπο, αλλά κατά ένα ποσοστό του 98%-99% είναι κοινό. Βρίσκεται στον πυρήνα των κυττάρων και ο κύριος ρόλος του είναι η μακροχρόνια αποθήκευση πληροφοριών και οδηγιών για την παραγωγή άλλων συστατικών των κυττάρων, όπως είναι τα μόρια RNA και οι πρωτεΐνες. Η ανακάλυψη της δομής του έγινε το 1953 από τους Watson & Crick, οι οποίοι παρουσίασαν το μοντέλο της δομής του DNA, που ονομάσθηκε «μοντέλο της διπλής έλικας». Σύμφωνα με το μοντέλο

αυτό, το DNA αποτελείται από δύο πολυνουκλεοτιδικές παράλληλες αλυσίδες σε μορφή δύο κλώνων που σχηματίζουν δεξιόστροφη διπλή έλικα. Οι αλυσίδες αυτές συγκροτούνται από αζωτούχες βάσεις, φωσφορικές ρίζες και ένα σάκχαρο με πέντε άτομα άνθρακα (πεντόζη). Οι αζωτούχες βάσεις είναι η αδενίνη (A), η θυμίνη (T), η γουανίνη (G) και η κυτοσίνη (C). Οι έλικες του DNA είναι αντι-παράλληλες και συνδέονται πάντοτε σύμφωνα με τον κανόνα συμπληρωματικότητας των Watson & Crick:

- ✓ αδενίνη (A) - θυμίνη (Τα) (ενώνονται με δύο δεσμούς υδρογόνου).
- ✓ κυτοσίνη (C) - γουανίνη (G) (ενώνονται με τρεις δεσμούς υδρογόνου).

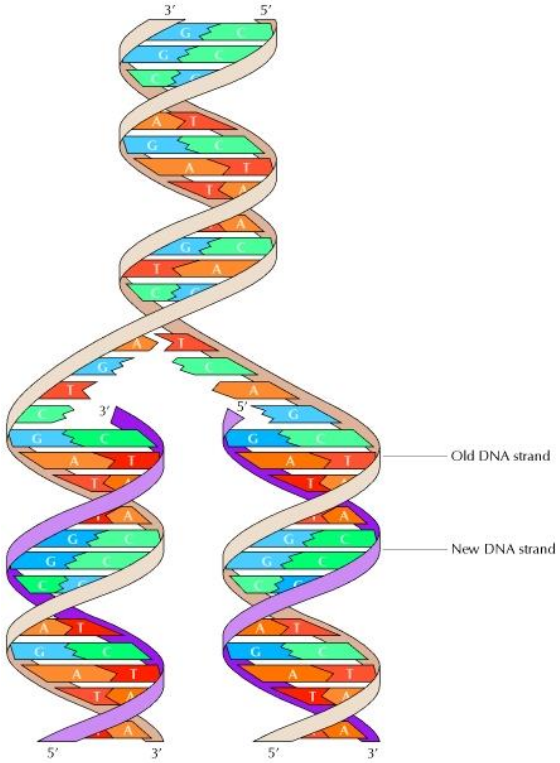
Η αλληλουχία των νουκλεοτιδίων είναι υπεύθυνη για την κωδικοποίηση και την αποθήκευση της πληροφορίας όπου κάθε μία από τις βάσεις μπορεί να θεωρηθεί ως ένα γράμμα από ένα αλφάβητο τεσσάρων γραμμάτων που χρησιμοποιείται για την αποθήκευση της χημικής πληροφορίας, για το πως εκφράζεται μία πρωτεΐνη. Η διαδικασία με την οποία αναπαράγεται το DNA ονομάζεται αντιγραφή [34].



Εικόνα 2.1 Η δομή του DNA. (Πάρθηκε από τη 'U.S. National Library of Medicine')
(http://ghr.nlm.nih.gov/handbook/illustrations/dnastructure.jpg)

2.2 Αντιγραφή

Κατά τη διαδικασία της αντιγραφής, οι δύο έλικες του DNA διαχωρίζονται, με αποτέλεσμα να μένουν ελεύθερες οι βάσεις των νουκλεοτιδίων και να χωρίζεται τα ζεύγη A-T και C-G. Τότε ένζυμα, φέρνουν απέναντι από κάθε βάση την συμπληρωματική της. Με αυτό τον τρόπο, κάθε αλυσίδα, λειτουργεί ως καλούπι για την δημιουργία της συμπληρωματικής της, και το DNA αυτό-διπλασιάζεται. Η διαδικασία αυτή οδηγεί στην δημιουργία δύο μορίων DNA τα οποία είναι όμοια μεταξύ τους, αλλά και με το αρχικό μόριο DNA [6]. Δηλαδή, κάθε βάση έχει την ιδιότητα να δημιουργεί τη συμπληρωματική της.



Εικόνα 2.2 Η αντιγραφή του DNA (Πάρθηκε από το 'NCBI')
(http://www.ncbi.nlm.nih.gov/bookshelf/br.fcgi?book=cooper&part=A417&rendertype=figure&id=A430)

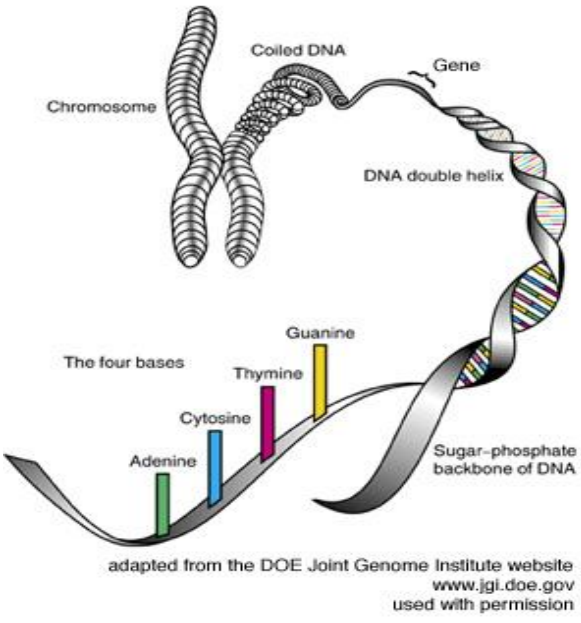
2.3 Χρωμοσώματα

Τα χρωμοσώματα αποτελούν οργανωμένες δομές DNA ή διαφορετικά, δομές που σχηματίζουν πακέτα DNA. Κάθε ένα από αυτά περιέχει ένα πάρα πολύ μακρύ

μόριο DNA, που είναι πακεταρισμένο με τέτοιο τρόπο ώστε να έχει 50000 φορές μικρότερο μήκος.

Στη διαδικασία δημιουργίας των χρωμοσωμάτων λαμβάνουν μέρος κάποιες ειδικές πρωτεΐνες που συνδέονται μαζί με το μακρομόριο του DNA σχηματίζοντας ένα σύμπλοκο που ονομάζεται χρωματίνη. Το βασικό στοιχείο από το οποίο αποτελείται η χρωματίνη είναι το νουκλεόσωμα, ένα πρωτεϊνικό οκταμερές που αποτελείται από τέσσερις διαφορετικούς τύπους πρωτεϊνών που ονομάζονται ιστόνες.

Στον άνθρωπο κάθε σωματικό κύτταρο περιέχει δύο αντίγραφα από κάθε χρωμόσωμα από τα οποία το ένα προέρχεται από τον πατέρα και το άλλο από τη μητέρα. Τα δύο αυτά αντίγραφα σχηματίζουν ζεύγη χρωμοσωμάτων που ονομάζονται ομόλογα λόγω της ίδιας δομής και οργάνωσης που παρουσιάζουν. Υπάρχουν 23 τέτοια ζεύγη ομόλογων χρωμοσωμάτων στον άνθρωπο, κάθε ένα από τα οποία είναι υπεύθυνο για την έκφραση ενός διαφορετικού χαρακτηριστικού του ανθρώπινου οργανισμού και ένα από αυτά είναι υπεύθυνο για το φύλο του ανθρώπου. Το 23° ζεύγος δεν αποτελεί ομόλογο ζεύγος χρωμοσωμάτων, γιατί στην περίπτωση άρρεν ατόμου, το ένα αποτελείται από το X και το άλλο από το Y. Σε περίπτωση που τα δύο αυτά χρωμοσώματα είναι ομόλογα, δηλαδή XX, το άτομο αυτό είναι θηλυκό.

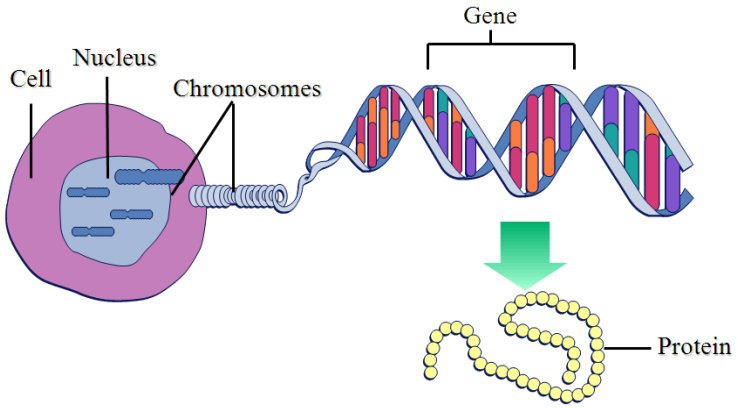


Εικόνα 2.3 Το χρωμόσωμα
(http://groups.nbp.northwestern.edu/science-

Εικόνα 2.4 Τα 23 ζεύγη ομόλογων χρωμοσωμάτων
(http://publications.nigms.nih.gov/thenewgenetics/chapter1.html)

2.4 Γονίδια

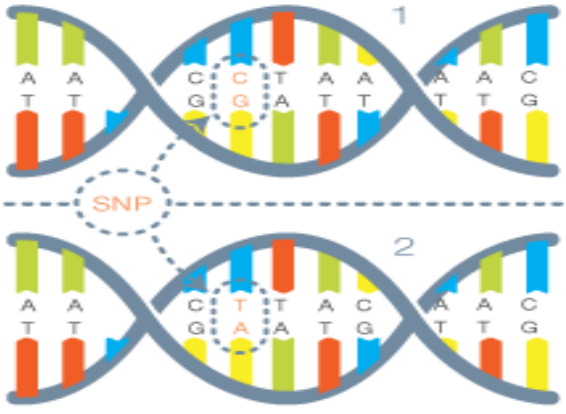
Τα γονίδια αποτελούν τμήματα DNA που μπορούν να μεταφραστούν σε ένα προϊόν πρωτεΐνης. Τα τμήματα αυτά του DNA που δεν μεταφράζονται δηλαδή που δεν παράγουν κάποιο προϊόν ονομάζονται ιντρόνια ενώ τα τμήματα που μεταφράζονται παράγοντας κάποιο προϊόν ονομάζονται εξώνια. Τα γονίδια αποτελούν εξώνια γιατί κάθε μετάφραση γονιδίου παράγει κάποιο προϊόν πρωτεΐνης [45].



Εικόνα 2.5 Το γονίδιο (Πάρθηκε από το 'National Fragile X Foundation')
(http://www.fragilex.org/fragile-x-associated-disorders/genetics-and-inheritance/fmr1-gene/)

2.5 Σημειακοί Νουκλεοτιδικοί Πολυμορφισμοί (SNP)

Σημειακοί νουκλεοτιδικοί πολυμορφισμοί (single nucleotide polymorphism SNP), είναι παραλλαγές που παρατηρούνται στις ακολουθίες του DNA και οι οποίες εμφανίζονται όταν ένα νουκλεοτίδιο (A,T,C ή G) στην ακολουθία του γονιδιώματος αλλαχθεί, για παράδειγμα λόγω μετάλλαξης. Για παράδειγμα ένα SNP θα μπορούσε να αλλάξει μια ακολουθία DNA από A**A**GGCTAA σε A**T**GGCTAA. Τα SNPs καλύπτουν 90% των γενετικών παραλλαγών που παρατηρούνται στον άνθρωπο. Ένας τέτοιος πολυμορφισμός παρατηρείται κάθε 100 μέχρι και 300 βάσεις κατά μήκος των 3 δισεκατομμυρίων βάσεων του ανθρώπινου γονιδιώματος. Στο παράδειγμα που προαναφέρθηκε, τα δύο διαφορετικά νουκλεοτίδια που παρουσιάζονται στην ίδια θέση στις δύο νουκλεοτιδικές ακολουθίες, θεωρούνται **αλληλόμορφα**. Τα περισσότερα κοινά SNPs αποτελούνται από 2 μόνο αλληλόμορφα. Οι τέσσερις διαφορετικές περιπτώσεις SNPs είναι AA, Aa, aa και η τέταρτη περίπτωση αυτή στην οποία παρατηρούνται missing data δηλαδή έλλειψη δεδομένων, διαφορετικά έλλειψη νουκλεοτιδίων σε κάποιες θέσεις των ακολουθιών DNA. Τα SNPs μπορούν να εμφανιστούν και στα εξώνια αλλά και στα ιντρόνια. Δηλαδή μπορούν να εμφανιστούν σε περιοχές του DNA που κωδικοποιούν κάποιο προϊόν αλλά και στις περιοχές που δεν κωδικοποιούν κάποια πρωτεΐνη ή κάποιο μόριο mRNA. Πολλά από τα SNPs δεν επηρεάζουν τη λειτουργία του κυττάρου, όμως πιστεύεται πως κάποια άλλα θα μπορούσαν να δημιουργήσουν την προδιάθεση στον άνθρωπο να ασθενήσει ή ακόμη να επηρεάσουν την αντίδραση του οργανισμού τους σε κάποιο φάρμακο [45].



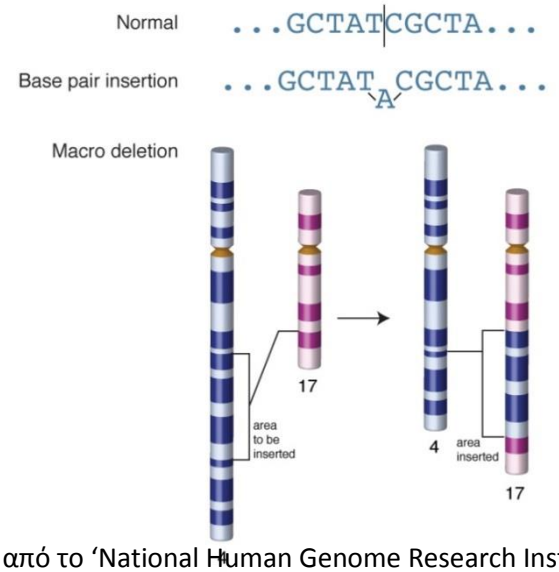
Εικόνα 2.6 Σημειακός Νουκλεοτιδικός Πολυμορφισμός –SNP (Πάρθηκε από το 'Sirius Genomics') http://www.siriusgenomics.com/images/snp_diagram.png)

2.6 Insertion/Deletion

Ο κάθε άνθρωπος μπορεί να έχει κάποιες (μετα)αλλαγές στο DNA του κατά τη διάρκεια της ζωής του. Αυτές οι αλλαγές συμβαίνουν σε μια σειρά από τρόπους. Μερικές φορές υπάρχουν απλά λάθη αντιγραφής που εισάγονται κατά την αναπαραγωγή του DNA. Άλλες αλλαγές μπορεί να είναι αποτέλεσμα κάποιας βλάβης του DNA λόγω περιβαλλοντικών παραγόντων, συμπεριλαμβανομένων του φωτός του ήλιου, του καπνού του τσιγάρου, και της ακτινοβολίας. Τα κύτταρά μας έχουν κατασκευαστεί με τέτοιους μηχανισμούς που να εντοπίζουν και να επιδιορθώνουν τις περισσότερα από τις αλλαγές που συμβαίνουν κατά τη διάρκεια της αντιγραφής του DNA ή από την περιβαλλοντική ζημιά . Καθώς γερνάμε, όμως, η επιδιόρθωση του DNA μας δεν λειτουργεί τόσο αποτελεσματικά κι έτσι συσσωρεύονται αλλαγές στο DNA μας [16].

2.6.1 Insertion

Η εισαγωγή (insertion) είναι ένας τύπος μετάλλαξης που αφορά την προσθήκη γενετικού υλικού. Μια μετάλλαξη εισαγωγής μπορεί να είναι μικρή και να περιλαμβάνει ένα επιπλέον ζεύγος βάσεων (base pair) του DNA, ή μεγάλη και να περιλαμβάνει ένα κομμάτι χρωμοσώματος [5]. Η εισαγωγή μπορεί να προκαλέσει μεταλλάξεις μετατόπισης πλαισίου (frame-shift mutation), δηλαδή να αλλάξει το νόημα του DNA και να καταλήξει σε μία συντομευμένη και μη λειτουργική πρωτεΐνη [8].

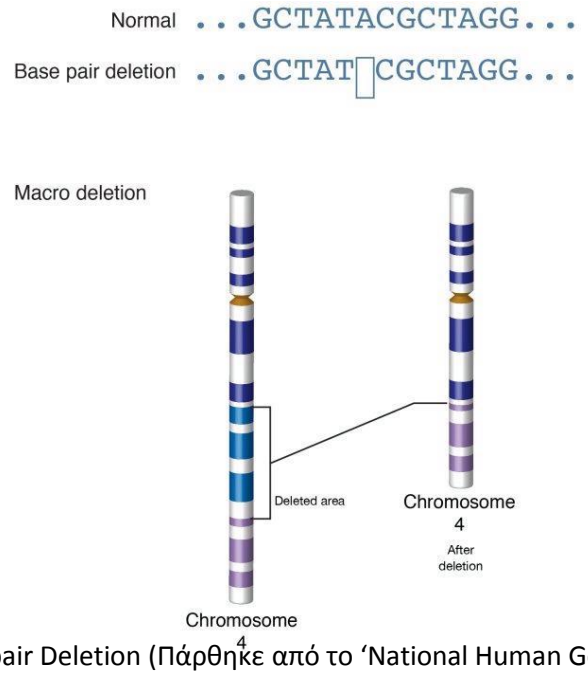


(Πάρθηκε από το 'National Human Genome Research Institute')

(http://www.genome.gov/glossary/resources/insertion_lg.jpg)

2.6.2 Deletion

Η διαγραφή (deletion) είναι ένας τύπος μετάλλαξης που αφορά την απώλεια γενετικού υλικού. Μια μετάλλαξη διαγραφής μπορεί να είναι μικρή και να περιλαμβάνει την απώλεια ενός ζεύγους βάσεων (base pair) του DNA, ενδιάμεση και να αφορά μερικά γονίδια ή μεγάλη και να περιλαμβάνει ένα κομμάτι χρωμοσώματος [25]. Οι μεγάλες διαγραφές μπορούν να ανιχνευθούν με μια εξέταση ρουτίνας των χρωμοσωμάτων, οι ενδιάμεσες διαγραφές με τη χρήση φθορίζοντος in situ υβριδισμού (FISH) και οι μικρές διαγραφές μόνο με την ανάλυση του DNA [33]. Όπως οι εισαγωγές, μπορούν και οι διαγραφές να προκαλέσουν μετατόπιση πλαισίου [8].



Εικόνα 2.7 Base pair Deletion (Πάρθηκε από το 'National Human Genome Research Institute') (http://www.genome.gov/dmd/previews/85289_large.jpg)

2.7 Sequencing

2.7.1 DNA sequencing

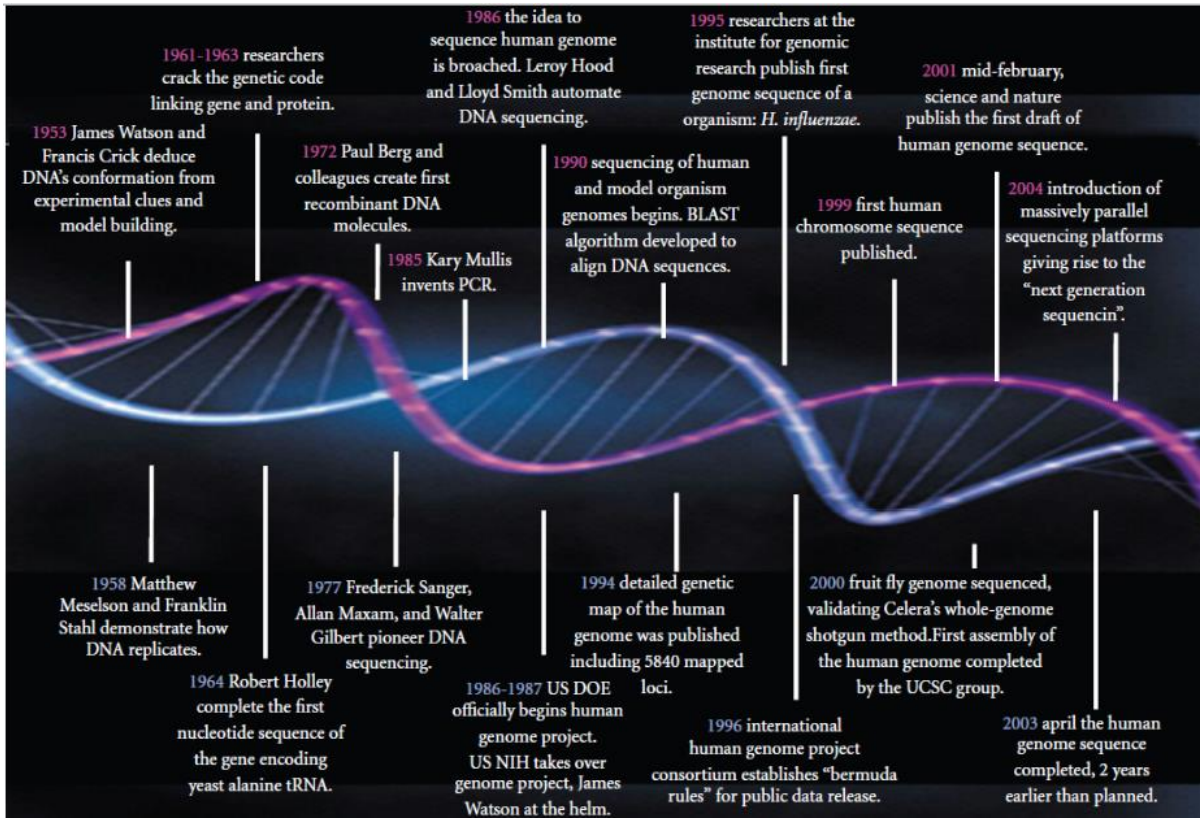
Το DNA Sequencing αποτελεί τη διαδικασία καθορισμού της ακριβής σειράς των νουκλεοτιδίων (οργανικές ενώσεις, σύνθετα οργανικά μόρια, που σχηματίζουν τη

βασική μονάδα των νουκλεϊκών οξέων (DNA και RNA)) μέσα σε ένα μόριο DNA.

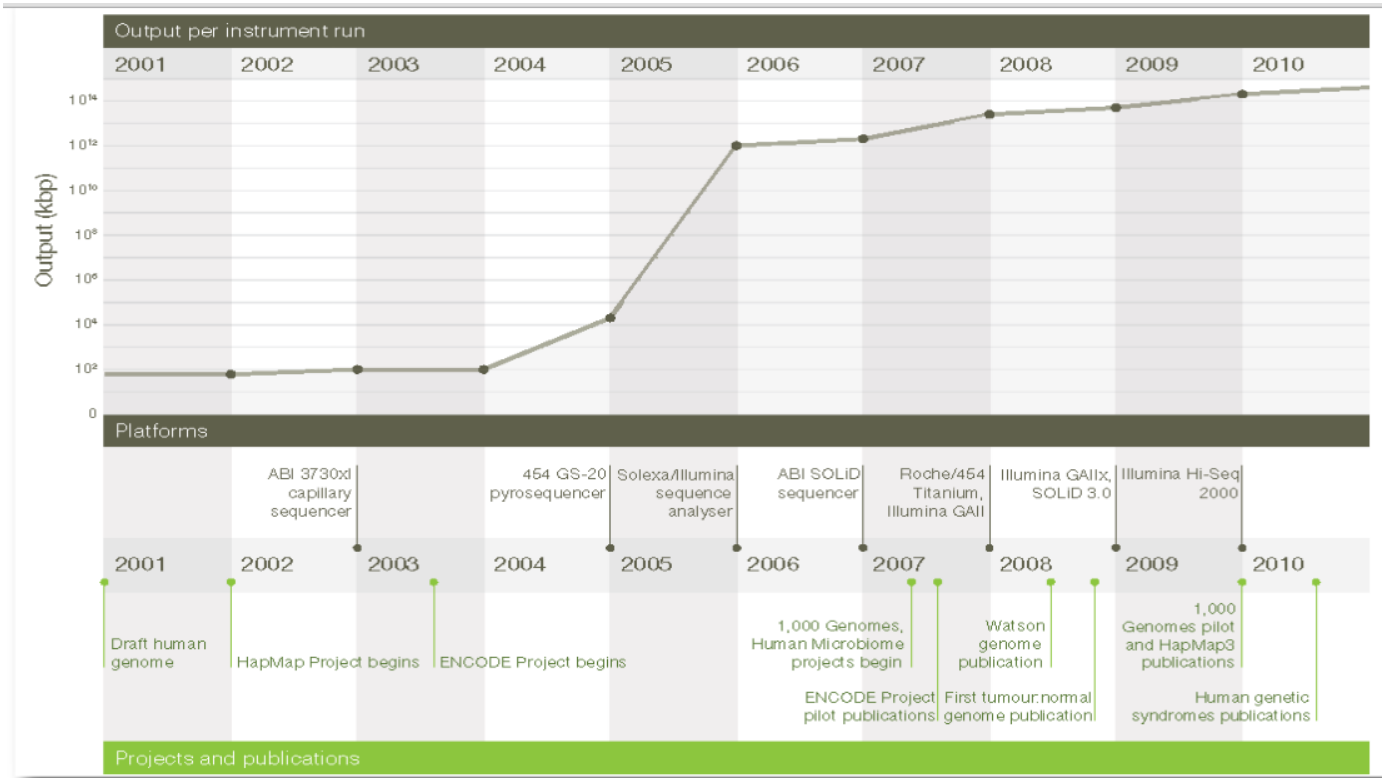
Περιλαμβάνει κάθε μέθοδο ή τεχνολογία που χρησιμοποιείται για τον καθορισμό της σειράς των τεσσάρων βάσεων (A, T C, G) [37].

2.7.1.1 Ιστορία του DNA sequencing

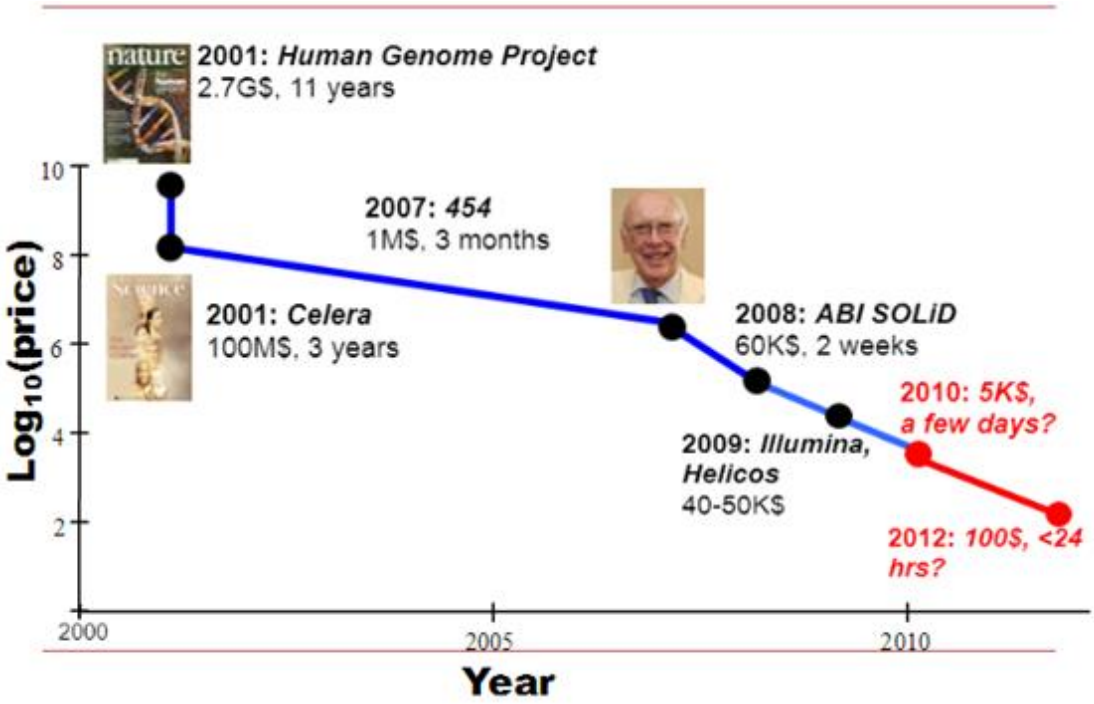
Οι πρώτες DNA ακολουθίες βρέθηκαν στις αρχές του 1970 (1975) από ακαδημαϊκούς ερευνητές που χρησιμοποίησαν εργαστηριακές μεθόδους βασισμένες στη δισδιάστατη χρωματογραφία. Στη συνέχεια ανακαλύφθηκαν μέθοδοι βασισμένες στο φθορισμό με αυτοματοποιημένη ανάλυση, που έκαναν το DNA sequencing πιο εύκολο και γρήγορο [37].



Εικόνα 2.8 Εξέλιξη της επανάστασης του DNA [37].



Εικόνα 2.9 Η επανάσταση στη sequencing τεχνολογία [37].



Εικόνα 2.10 Sequencing the Human Genome [1].

2.7.1.2 Χρήση του DNA sequencing

Το DNA sequencing βοήθησε [44] :

- Στον καθορισμό της σειράς των ανεξάρτητων γονιδίων.
- Στη χρήση μεγαλύτερων γενετικών περιοχών, πλήρων χρωμοσωμάτων ή όλου του γονιδιώματος.
- Στη χρήση των σειρών από ερευνητές της μοριακής βιολογίας ή της γενετικής για περαιτέρω επιστημονική πρόοδο ή χρήση τους από το ιατρικό προσωπικό για ιατρικές αποφάσεις ή βοήθεια για γενετικές συμβουλές.

2.7.2 Next Generation sequencing (NGS)

Οι τεχνολογίες NGS άρχισαν να χρησιμοποιούνται το 2005. Ουσιαστικά οι τεχνολογίες αυτές, παίρνουν ολόκληρο το γονιδίωμα, το σπάζουν σε εκατομμύρια πολύ μικρά κομμάτια και δημιουργούν ακολουθία για όλα. Χρησιμοποιούν μικροτεχνολοίες και νανοτεχνολογίες για τη μείωση του μεγέθους των δειγμάτων, του κόστους και την αύξηση των δειγμάτων που αναλύονται και για να επιτρέπει την εύρεση της σειράς παράλληλα.

Το NGS έχει ως βασικό πλεονέκτημα το διάβασμα χιλιάδων ή εκατομμυρίων δεδομένων σε κάθε δείγμα για να κατανοήσουν τις μολυσματικές εστίες των ασθενειών ενώ με το Sanger sequencing διαβάζεται μόνο ένα δεδομένο/δείγμα.

Τα **πλεονεκτήματα** των NGS τεχνολογιών είναι [42] :

- ✓ Μείωση του κόστους και αύξηση απόδοσης της γονιδιωματικής αλληλουχίας λόγω λιγότερου χρόνου, ανθρωπίνου δυναμικού και αντιδραστήρων.
- ✓ Μείωση του χρόνου λόγω του ότι η χημική αντίδραση και εύρεση του σήματος γίνονται σε ένα βήμα.
- ✓ Ακριβές και έμπιστο λόγω εκτέλεσης περισσότερων επαναλήψεων.
- ✓ Εντοπισμός αιτίας πολλών ασθενειών άγνωστης αιτιολογίας .

✓ Προβολή χιλιάδων θέσεων στο γονιδίωμα για εύρεση των παθογόνων μεταλλάξεων.

✓ Αλληλουχία βιολογικών δειγμάτων για τις γονιδιακές υπογραφές των νέων μολυσματικών παραγόντων.

✓ Αύξηση της γνώσης μας γύρω από τη φαρμακογενετική, τη γενετική του καρκίνου, την επιγενετική και των σύνθετων χαρακτηριστικών.

✓ Αντικατάσταση συστοιχιών και αλληλουχιών Sanger σε κλινικές εφαρμογές που ήδη χρησιμοποιούνται.

Αντίθετα, κάποια **μειονεκτήματα** των NGS τεχνολογιών που υπάρχουν είναι ότι [42]:

- Απαιτείται μεγάλη ακρίβεια και καλύτεροι τρόποι για επιλογή των υποσυνόλων των γονιδιωμάτων που τυχαίνουν ενδιαφέροντος
- Απαιτούνται βελτιώσεις στη λειτουργικότητα, την ταχύτητα, την ευκολία χρήσης του λογισμικού ανάλυσης δεδομένων στις υποδομές του εργαστηρίου Πληροφορικής, την αποθήκευση δεδομένων και την ικανότητα μεταφοράς δεδομένων, αφού θα παράγονται εξαιρετικά μεγάλα σύνολα δεδομένων
- Απαιτείται επίσης η κατάρτιση των ιατρών και του προσωπικού των εργαστηρίων

Επιπρόσθετα δημιουργούνται κάποια **ηθικά ζητήματα** όπως [42]:

- Η χρήση προσωπικών ιατρικών δεδομένων
- Το γεγονός ότι το μεγαλύτερο μέρος των ακολουθιών δεν έχει σχέση με το συγκεκριμένο κλινικό πρόβλημα, αλλά μπορεί να έχει σημασία για τον ασθενή με άλλου τρόπους ή μελλοντικά
- Η αδυναμία μας να ερμηνεύσουμε κάποια δεδομένα
- Η δυνατότητα να συνδέσουμε τις πληροφορίες με συγκεκριμένο άτομο παρά την παντελή απουσία οποιασδήποτε προσωπικής πληροφορίας αναγνώρισης του ατόμου

Στη συνέχεια, αναφέρουμε τους τρεις τύπους sequencing [37]:

- Με σύνθεση

- Με σύνδεση (ligation)
- Single-molecule

Στον πρώτο τρόπο (**σύνθεση**) χρησιμοποιείται ο πολυμερισμός. Στη διαδικασία του Sanger sequencing χρειάζονται διαφόρων μεγεθών template fragments, ενώ με η NGS σύνθεση βρίσκει το κατάλληλο μέγεθος, συνδέεται στις σειρές του adaptor και ενισχύει το φθορίζον ή χημικό σήμα. Στη συνέχεια τα templates μοιράζονται και ακινητοποιούνται για την προετοιμασία των flow-cell κύκλων.

Ο δεύτερος τρόπος (**σύνδεση**) χρησιμοποιεί την ευαισθησία στην ανομοιότητα του DNA για την εύρεση της σειράς των νουκλεοτιδίων. Αυτή η μέθοδος χρησιμοποιεί ανιχνευτές ολιγονουκλεοτιδίων διαφόρων μεγεθών τα οποία επισημαίνονται με ετικέτες φθορισμού ανάλογα με το νουκλεοτίδιο που αντιπροσωπεύουν. Τα μειονεκτήματα αυτού του τρόπου είναι ότι μπορεί να αντικατασταθούν λανθασμένες βάσεις γονιδιωμάτων κι επίσης έχει ελλιπή αντιπροσώπευση AT και CG περιοχών.

Ο τρίτος τρόπος (single-molecule) βρίσκει τη σειρά των DNA μορίων απευθείας χωρίς της χρήση ενίσχυσης. Έτσι έχει ως πλεονεκτήματα το ψηλό throughput και λιγότερα λάθη.

Method	Single-molecule real-time sequencing (Pacific Bio)	Ion semiconductor (Ion Torrent sequencing)	Pyrosequencing (454)	Sequencing by synthesis (Illumina)	Sequencing by ligation (SOLiD sequencing)	Chain termination (Sanger sequencing)
Read length	5,500 bp to 8,500 bp avg (10,000 bp N50); maximum read length >30,000 bases[42][43][44]	up to 400 bp	700 bp	50 to 300 bp	50+35 or 50+50 bp	400 to 900 bp
Accuracy	99.999% consensus accuracy; 87% single-read accuracy[45]	98%	99.9%	98%	99.9%	99.9%
Reads per run	50,000 per SMRT cell, or ~400 megabases[46][47]	up to 80 million	1 million	up to 3 billion	1.2 to 1.4 billion	N/A
Time per run	30 minutes to 2 hours [48]	2 hours	24 hours	1 to 10 days, depending upon sequencer and specified read length[49]	1 to 2 weeks	20 minutes to 3 hours
Cost per 1 million bases (in US$)	$0.33-$1.00	$1	$10	$0.05 to $0.15	$0.13	$2400
Advantages	Longest read length. Fast. Detects 4mC, 5mC, 6mA.[50]	Less expensive equipment. Fast.	Long read size. Fast.	Potential for high sequence yield, depending upon sequencer model and desired application.	Low cost per base.	Long individual reads. Useful for many applications.
Disadvantages	Moderate throughput. Equipment can be very expensive.	Homopolymer errors.	Runs are expensive. Homopolymer errors.	Equipment can be very expensive. Requires high concentrations of DNA.	Slower than other methods. Have issue sequencing palindromic sequence.[51]	More expensive and impractical for larger sequencing projects.

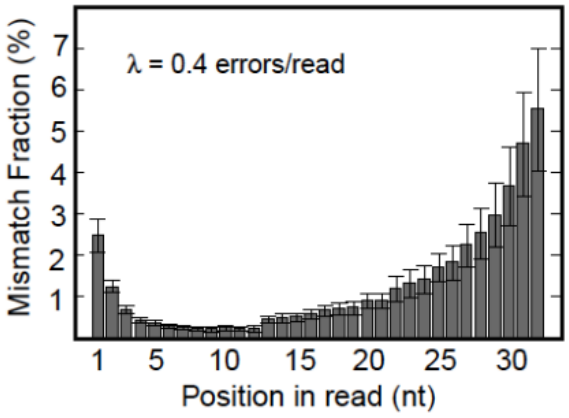
Πίνακας 2.1 Σύγκριση NGS μεθόδων/πλατφόρμων
(http://en.wikipedia.org/wiki/Next-generation_sequencing#Next-generation_methods)

	Throughput	Length	Quality	Costs
Sanger	6 Mb/day	800nt	10^{-4} - 10^{-5}	500$/Mb
454	750Mb/day	400nt	10^{-3} - 10^{-4}	~20$/Mb
Ion Torrent	1600Mb/day	200nt	10^{-2} - 10^{-3}	~10$/Mb
Illumina	100000Mb/day	125nt	10^{-2} - 10^{-3}	~0.40$/Mb
SOLiD 4	100000Mb/day	125nt	10^{-2} - 10^{-3}	~0.40$/Mb
Helicos	5000Mb/day	32nt	10^{-2}	~0.40$/Mb

Πίνακας 2.2 Σύγκριση NGS μεθόδων
(http://www.uni-leipzig.de/~strimmer/lab/courses/ss12/current-topics/slides/1-NGS.pdf)

2.7.5 Προβλήματα με το NGS

- Τα δεδομένα είναι μικρά κι έτσι είναι δύσκολο να μπουν στην ακολουθία οι επαναλαμβανόμενες περιοχές [12], [42] .
- Το ποσοστό λάθους αυξάνεται όσο αυξάνονται οι βάσεις της ακολουθίας [42] .
- Το alignment είναι αργό, για αυτό και χρειάζεται η παραλληλοποίηση.



Σχήμα 2.1 Ποσοστό λάθους/αριθμός βάσεων [42].

Κεφάλαιο 3

Σχεδιασμός και Υλοποίηση Πλατφόρμας

3.1 Αρχιτεκτονική Πλατφόρμας

Το γεγονός ότι η επεξεργασία, που θα κάνει η NGS data analyses πλατφόρμα, θα γίνεται σε μεγάλου όγκου δεδομένα, αναντίρρητα, μας οδηγεί στο συμπέρασμα ότι η παράλληλη εκτέλεση σε υψηλής απόδοσης υπολογιστή, θα αποτελεί μία

καλή λύση. Άρα μία τέτοια πλατφόρμα θα πρέπει να εφαρμόζεται σε High Performance Computing (HPC) μηχανή, γι' αυτό, χρησιμοποιήθηκε ο Cy-Tera server του Cyprus Institute.

Η γενικός διαχωρισμός αλγορίθμων/ μερών, από τους οποίους αποτελείται η NGS data analyses πλατφόρμα, **αφαιρετικά**, είναι:

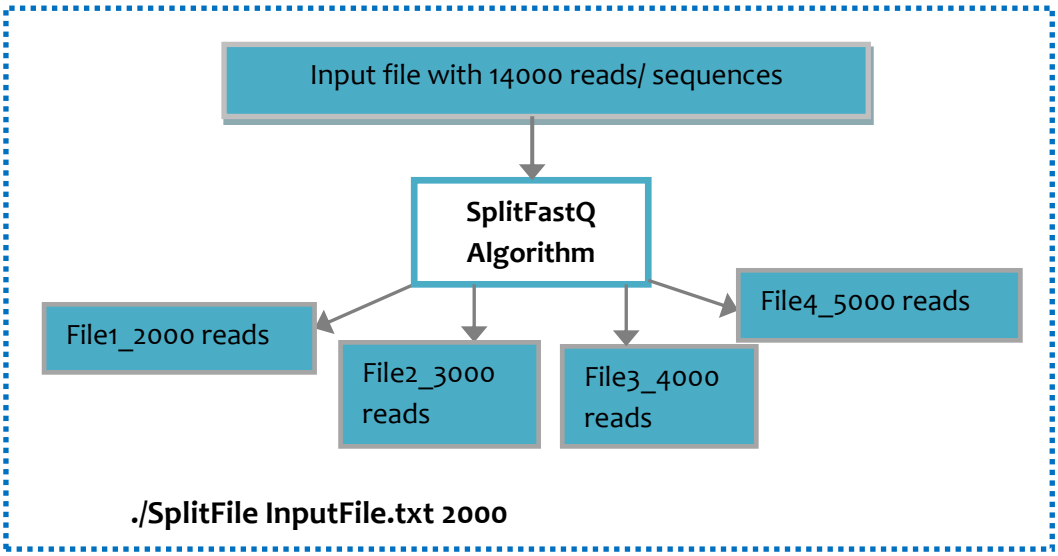
3.1.1 Αλγόριθμος Διάσπασης FASTQ Αρχείων:

Όνομα Αλγορίθμου: SplitFastQ

Είσοδος: Λαμβάνει σαν είσοδο ένα μεγάλο αρχείο μορφότυπου FASTQ, με reads ανθρώπινων γωνιδιωμάτων και ένα ακέραιο αριθμό Χ. Το συγκεκριμένο format του αρχείου (θα επεξηγηθεί πιο κάτω).

Έξοδος: Πολλά αρχεία τύπου FASTQ, με reads ανθρώπινων γωνιδιωμάτων, όπου το πρώτο αρχείο αποτελείται από Χ reads (τα Χ πρώτα reads του αρχείου εισόδου), το δεύτερο από X_1=X+(1000*1) reads (από το read X+1 μέχρι το read (X+1)+(1000) του αρχείου εισόδου), το τρίτο από X_2 = X+(1000*2) reads (από το read X_1+1 μέχρι το read (X_1+1)+X_2, του αρχείου εισόδου), κοκ. Δηλαδή, διασπούμε ένα μεγάλο αρχείο εισόδου σε κομμάτια/ μικρότερα αρχεία ώστε η επεξεργασία τους να γίνεται παράλληλα, πιο εύκολα και γρήγορα. Άρα, το πρώτο αρχείο θα περιλαμβάνει X_o=X reads, το δεύτερο Το X_1=X_o+1000, X_2=X_1+1000, . . . , το τελευταίο αρχείο (ας το υποθέσουμε ότι είναι το n-οστό) θα αποτελείται από X_n=X_{n-1}+1000 γονιδιώματα. Το '1000', δηλαδή το ποσό αύξησης πλήθους αλληλουχιών από αρχείο σε αρχείο, δύναται να τροποποιηθεί μέσα από το κώδικα όπως επιθυμεί ο χρήστης. Το αρχείο εισόδου θα διασπαστεί σε πολλαπλά αρχεία ίδιας μορφής, FASTQ, η οποία αναλύεται στο υποκεφάλαιο 3.2.1.1. Ο αλγόριθμος αυτός για να εκτελεστεί καλείται ως εξής:

./SplitFastQ <name of FASTQ file> <number of reads in first group/file of data>



Εικόνα 3.1 Γραφική αναπαράσταση λειτουργίας αλγορίθμου "SplitFile" με αρχείο εισόδου 14000 reads και ποσό αύξησης αλληλουχιών ανά αρχείο 1000 reads

3.1.2 Αλγόριθμος Επεξεργασίας Αλληλουχιών

Όνομα Αλγορίθμου: Αλγόριθμος Trimming

Είσοδος: Λαμβάνει σαν είσοδο ένα αρχείο τύπου FASTQ και ένα χαρακτήρα, έστω C, με ASCII code να ανήκει στο διάστημα [33, 126]. Αυτό το αρχείο FASTQ αποτελεί αρχείο εξόδου του Αλγόριθμου Διάσπασης Αρχείων "SplitFastQ", με reads ανθρώπινων γωνιδιωμάτων και των αντίστοιχων quality scores αυτών , με συγκεκριμένο format (θα επεξηγηθεί πιο κάτω – βλέπε υποκεφάλαιο 3.2.1.1). Ο χαρακτήρας εισόδου C αναπαριστά τον αντίστοιχο βαθμό ποιότητας , με βάσει τον οποίο θα γίνει η αφαίρεση νουκλεοτιδίων με ποιότητα μικρότερη ή ίση αυτής που αντιστοιχεί ο χαρακτήρας με βάσει το Phred quality score (η αντιστοιχία του χαρακτήρα με το επίπεδο ποιότητας νουκλεοτιδίων θα αναλυθεί στη συνέχεια- βλέπε Πίνακας 3.1) .

Έξοδος: Δύο αρχεία. Ένα αρχείο, με reads ανθρώπινων γωνιδιωμάτων, των οποίων τα αντίστοιχα quality scores έχουν κωδικοποιηθεί (η στρατηγική κωδικοποίησης αναλύεται στο υποκεφάλαιο 3.2) και ένα αρχείο με reads

ανθρώπινων γωνιδιωμάτων, μετά από συγκεκριμένη επεξεργασία (η επεξεργασία αυτή λέγεται 'κλαδεμα' ακολουθίας [38] – βλέπε υποκεφάλαιο 3.2) και τα αντίστοιχα κωδικοποιημένα quality scores.

Για την εκτέλεση του αλγορίθμου χρησιμοποιείται η εντολή:
./NGS_platform <name of input file> <character with ASCII code in range [33 , 126]>

3.1.3 Αλγόριθμος Παραλληλοποίησης:

Όνομα Αλγορίθμου: RunSubFiles

Είσοδος: Λαμβάνει σαν είσοδο μία ακολουθία ονομάτων των αρχείων τα οποία αποτελούν έξοδο του αλγορίθμου 2, «Αλγόριθμος Trimming».

Έξοδος: Πλήθος Script Files, όπου το κάθε ένα από αυτά καλεί τον επόμενο αλγόριθμο 4, «Aligner», 12 φορές με 12 διαφορετικά ζεύγη αρχείων κάθε φορά, καλύπτοντας κάθε δυνατό ζεύγος που δύναται να υπάρχει μεταξύ των αρχείων εισόδου. Αναλυτικότερα, αν για παράδειγμα ο χρήστης δώσει 10 αρχεία εισόδου, θα υπάρχουν$\binom{10}{2}$+ 10 δυνατά ζεύγη (κάθε αρχείο συνδυάζεται με όλα τα υπόλοιπα και τον εαυτό του).

Παράδειγμα κλήσης του αλγορίθμου:
./RunSubFiles file1.txt, file2.txt, file3.txt, file4.txt, . . . , fileN.txt

3.1.4 Αλγόριθμος Ευθυγράμμισης Αλληλουχιών

Όνομα Αλγορίθμου: Aligner.

Είσοδος: Λαμβάνει σαν είσοδο **δύο** αρχεία, με reads ανθρώπινων γωνιδιωμάτων, με συγκεκριμένο format (θα επεξηγηθεί πιο κάτω), και **ένα ακέραιο X**

αριθμό που υποδηλώνει τον ελάχιστο αριθμό επικάλυψης/ ταιριάσματος νουκλεοτιδίων (minimum overlap/ matching) μεταξύ δύο reads.

Έξοδος: Ένα αρχείο με τα reads που επικαλύπτονται μεταξύ τους τουλάχιστον κατά Χ νουκλεοτίδια και τον ακριβή αριθμό που πράγματι επικαλύπτονται/ ταιριάζουν (βλέπε υποκεφάλαιο 3.2.4).

Όνομα αρχείου εξόδου:

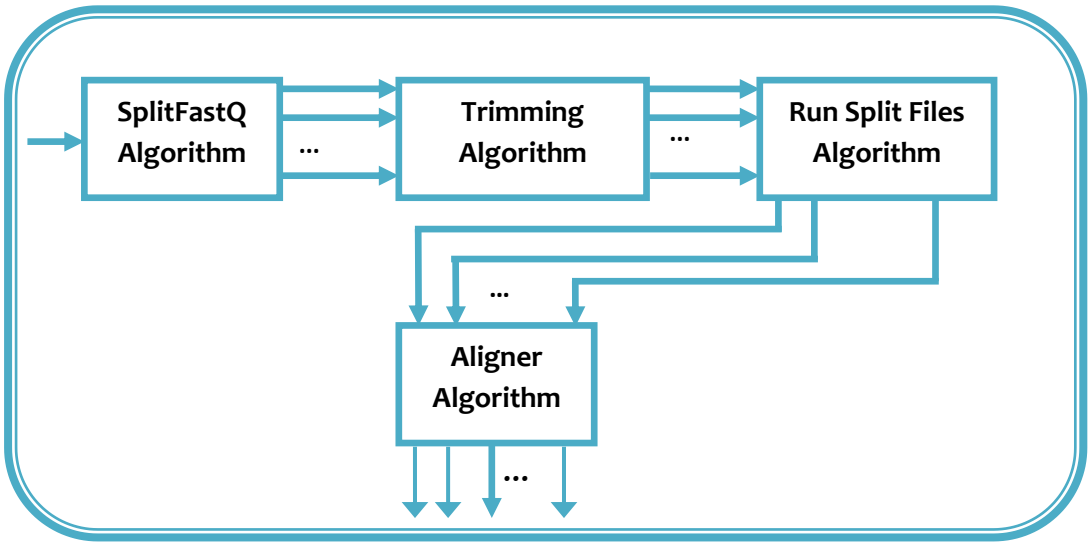
<file name1>_<file name2>. Overlap< minimum overlap >.Align.

Παράδειγμα τρόπου κλήσης του αλγορίθμου:

./aligner file_name1, file_name2 4

3.2 Δομή Πλατφόρμας

Αναπαράσταση Δομής NGS data analyses πλατφόρμας



Εικόνα 3.1 Γραφική αναπαράσταση της πλατφόρμας

Η νέα πλατφόρμα υλοποιήθηκε στη γλώσσα προγραμματισμού C++. Οι λειτουργίες της καινούριας NGS (Next Generation Sequence) data analyses platform διαχωρίστηκαν, έτσι ώστε κάθε τμήμα της (αλγόριθμος) να ικανοποιεί διαφορετικό σκέλος/ στόχο από τα υπόλοιπα. Για την ανάπτυξη αυτής,

δημιουργήθηκαν επίσης διάφοροι αλγόριθμοι, κλάσεις, βιβλιοθήκες και συναρτήσεις οι οποίες θα περιγραφούν εκτενώς πιο κάτω.

3.2.1 Αλγόριθμος Διάσπασης FASTQ Αρχείων

Αναντίλεκτα, για την επίλυση οποιουδήποτε βιολογικού ή/και βιοϊατρικού προβλήματος απαιτείται η γρήγορη διαχείριση μεγάλων δεδομένων εισόδου, για παράδειγμα διασπώντας τα σε μικρότερες ομάδες δεδομένων εισόδου, αφού ως επί το πλείστον τέτοιου είδους προβλήματα αναφέρονται σε τεράστιου όγκου γενετικά δεδομένα. Πολλοί υφιστάμενο αλγόριθμοι κατασκευάστηκαν χωρίς να δύναται να εφαρμοστούν σε μεγάλα δεδομένα, λόγο του μεγάλου χρόνου εκτέλεσης που θα προκύψει [43], όπως για παράδειγμα ο αλγόριθμος BLAT που περιγράφεται και μελετήθηκε από τους από τον W. James Kent (Department of Biology and Center for Molecular Biology of RNA, University of California, Santa Cruz, Santa Cruz, California 95064, USA) και εξηγήθηκε στο υποκεφάλαιο 1.1. Αποτελεί μία βελτιστοποιημένη παραλλαγή του αλγορίθμου ευθυγράμμισης BLAST, όμως με βάση τις μετρήσεις που έγιναν απαιτεί κατά μέσο όρο χρόνο 16,300 CPU hours για την ευθυγράμμιση 7.51×10^9 βάσεων σε 1.33×10^7 reads. Άρα για 1000 ακολουθίες απαιτεί 1,2256 CPU hours, χρόνος αρκετά μεγάλος για τόσες λίγες ακολουθίες DNA. Γεγονός που αναντίλεκτα δηλώνει την ανέφικτη εφαρμογή του σε High Performance Computing (HPC) machine, σε αντίθεση με τη πλατφόρμα που υλοποίησα.

Για το λόγο αυτό, ανέπτυξα ένα απλό, εύχρηστο και εύρωστο αλγόριθμο, ως προς το κώδικα, διάσπασης και ομαδοποίησης των γενετικών δεδομένων του αρχείου εισόδου σε μικρότερα. Ο αλγόριθμος αυτός ονομάστηκε «SplitFastQ Algorithm» .

Ο αλγόριθμος αυτός, διασπά τα δεδομένα των ακολουθιών και των πληροφοριών αυτών σε πολλά διαφορετικά αρχεία. Συγκεκριμένα, διαβάζει το περιεχόμενο του αρχείου εισόδου ανά 4 γραμμές και **αντιγράφει πιστά** τα δεδομένα, έτσι ώστε το πρώτο αρχείο εξόδου αποτελείται από X reads **(=number**

of reads in first group/file of data. X is given from user as input) (τα X πρώτα reads του αρχείου εισόδου), το δεύτερο από X_1=X+(1000*1) reads (από το read X+1 μέχρι το read (X+1)+ X_1, του αρχείου, το τρίτο από X_2 = X+(1000*2) reads (από το read X_1+1 μέχρι το read (X_1+1)+ X_2 του αρχείου), κοκ.

Τα αρχεία εξόδου έχουν την ακόλουθη μορφή ονομασίας:
<name of FASTQ file>Split<number of reads in output file>.F< serial number of file>

Η επιλογή για το ρυθμό αύξησης των ακολουθιών (+1000 reads σε κάθε επόμενο αρχείο εξόδου) ανά αρχείο δύναται εύκολα και γρήγορα να αλλάξει μέσα από το κώδικα, όπως έχει ήδη επισημανθεί.

3.2.1.1 Μορφή FASTQ αρχείου

Η μορφή ενός FASTQ αρχείου είναι μία μορφή που βασίζεται σε κείμενο αποθήκευσης τόσο των βιολογικών ακολουθιών (νουκλεοτιδική ακολουθία), όσο και των αντίστοιχων τιμών ποιότητας αυτών. Η ακολουθία, αλλά και οι τιμές ποιότητάς της αποτελούν μία σειρά από κωδικοποιημένα σύμβολα ASCII για λόγους συντομίας. Η μορφή αυτή αναπτύχθηκε αρχικά στο Wellcome Trust Sanger Institute για την συνένωση ακολουθιών FASTA και τη ποιότητα των στοιχείων της ακολουθίας.

Πιο κάτω καταγράφεται και εξηγείται η μορφή FASTQ [14] αρχείου εισόδου:
Για κάθε αλληλουχία χρησιμοποιούνται 4 γραμμές της μορφής:

@SEQ_ID <optional description>
<genome sequence>
+
< quality score of each SNP of sequence>

Γραμμή 1: Ξεκινά με το χαρακτήρα "@" και ακολουθείται από ένα μοναδικό αναγνωριστικό ακολουθίας και μια προαιρετική περιγραφή.

Γραμμή 2: Τα γράμματα (νουκλεοτίδια: A, T, C και G) της συγκεκριμένης ακολουθίας. Για απουσία ενός SNP (missing data) εισάγεται στο γράμμα **N** ως συμβολισμός.

Γραμμή 3: Ξεκινά με ένα «+» χαρακτήρα, προαιρετικά ακολουθούμενη από το ίδιο αναγνωριστικό αλληλουχίας (ή/και τη προαιρετική περιγραφή).

Γραμμή 4: Κωδικοποιεί τις τιμές ποιότητας (Sequence quality score) για την ακολουθία της γραμμής 2. Πρέπει να αποτελείται από ίδιο πλήθος συμβόλων με την αλληλουχία της γραμμής 2 (μία τιμή ποιότητας για κάθε νουκλεοτίδιο).

3.2.1.2 Sequence quality score

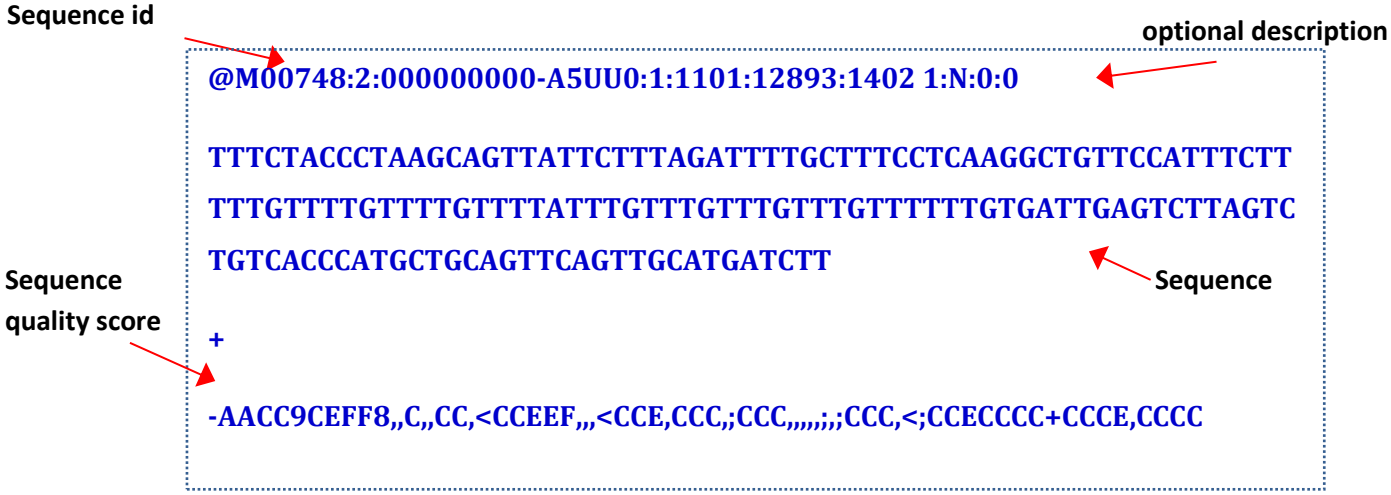
Για το συμβολισμό της ποιότητας του κάθε SNP μίας ακολουθίας **(Γραμμή 4)** , ξεχωριστά, χρησιμοποιούνται οι χαρακτήρες ASCII από το 33 μέχρι το 126. Αναφερόμενοι στη τιμή ποιότητας ενός νουκλεοτιδίου, εννοούμε κατά πόσο ένα νουκλεοτίδιο αντιγράφτηκε λάθος ή να προέκυψε λόγω πολυμορφισμού, insertion ή deletion (υποκεφάλαιο 2.6). Αυτή η κωδικοποίηση , εκφράζει τη πιθανότητα σφάλματος ενός SNP και είναι ευρέως αποδεκτή σαν μία μετρική αξιολόγησης της αποδοτικότητας μεταξύ διαφορετικών μεθόδων, αλλά και για χαρακτηρισμό της ποιότητας μίας ακολουθίας DNA.

Το «Phred quality scores» αναπτύχθηκε αρχικά για την αυτοματοποίηση της αλληλουχίας DNA στην έρευνα «Human Genome Project» [36]. Το Phred quality scores καθορίζει για ένα συγκεκριμένο νουκλεοτίδιο τη πιθανότητα για εσφαλμένο base. Ο χαρακτήρας ' ! ' αναπαριστά την ελάχιστη δυνατή ποιότητα ενός νουκλεοτιδίου, ενώ ο χαρακτήρας ' **~** ' τη μέγιστη,όπως φαίνεται στο Πίνακα 3.1.Η ακόλουθη σειρά συμβόλων παρουσιάζει σε αύξουσα ταξινόμηση τη σειρά συμβόλων ποιότητας, από τα αριστερά προς τα δεξιά(ASCII 33 έως 126, συμπεριλαμβανομένων):

!"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]
^_`abcdefghijklmnopqrstuvwxyz{|}~

Εικόνα 3.2 Quality value characters in left-to-right increasing order of quality (ASCII)
http://en.wikipedia.org/wiki/FASTQ_format#Format_converters

Ένα αρχείο FASTQ που περιέχει μια ενιαία ακολουθία μπορεί να μοιάζει κάπως έτσι:



Εικόνα 3.3 Παράδειγμα ενός read του αρχείου FASTQ

Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL (null)	32	20	040		Space	64	40	100	@	@	96	60	140	`	`
1	1	001	SOH (start of heading)	33	21	041	!	!	65	41	101	A	A	97	61	141	a	a
2	2	002	STX (start of text)	34	22	042	"	"	66	42	102	B	B	98	62	142	b	b
3	3	003	ETX (end of text)	35	23	043	#	#	67	43	103	C	C	99	63	143	c	c
4	4	004	EOT (end of transmission)	36	24	044	$	$	68	44	104	D	D	100	64	144	d	d
5	5	005	ENQ (enquiry)	37	25	045	%	%	69	45	105	E	E	101	65	145	e	e
6	6	006	ACK (acknowledge)	38	26	046	&	&	70	46	106	F	F	102	66	146	f	f
7	7	007	BEL (bell)	39	27	047	'	'	71	47	107	G	G	103	67	147	g	g
8	8	010	BS (backspace)	40	28	050	(	(	72	48	110	H	H	104	68	150	h	h
9	9	011	TAB (horizontal tab)	41	29	051	)	)	73	49	111	I	I	105	69	151	i	i
10	A	012	LF (NL line feed, new line)	42	2A	052	*	*	74	4A	112	J	J	106	6A	152	j	j
11	B	013	VT (vertical tab)	43	2B	053	+	+	75	4B	113	K	K	107	6B	153	k	k
12	C	014	FF (NP form feed, new page)	44	2C	054	,	,	76	4C	114	L	L	108	6C	154	l	l
13	D	015	CR (carriage return)	45	2D	055	-	-	77	4D	115	M	M	109	6D	155	m	m
14	E	016	SO (shift out)	46	2E	056	.	.	78	4E	116	N	N	110	6E	156	n	n
15	F	017	SI (shift in)	47	2F	057	/	/	79	4F	117	O	O	111	6F	157	o	o
16	10	020	DLE (data link escape)	48	30	060	0	0	80	50	120	P	P	112	70	160	p	p
17	11	021	DC1 (device control 1)	49	31	061	1	1	81	51	121	Q	Q	113	71	161	q	q
18	12	022	DC2 (device control 2)	50	32	062	2	2	82	52	122	R	R	114	72	162	r	r
19	13	023	DC3 (device control 3)	51	33	063	3	3	83	53	123	S	S	115	73	163	s	s
20	14	024	DC4 (device control 4)	52	34	064	4	4	84	54	124	T	T	116	74	164	t	t
21	15	025	NAK (negative acknowledge)	53	35	065	5	5	85	55	125	U	U	117	75	165	u	u
22	16	026	SYN (synchronous idle)	54	36	066	6	6	86	56	126	V	V	118	76	166	v	v
23	17	027	ETB (end of trans. block)	55	37	067	7	7	87	57	127	W	W	119	77	167	w	w
24	18	030	CAN (cancel)	56	38	070	8	8	88	58	130	X	X	120	78	170	x	x
25	19	031	EM (end of medium)	57	39	071	9	9	89	59	131	Y	Y	121	79	171	y	y
26	1A	032	SUB (substitute)	58	3A	072	:	:	90	5A	132	Z	Z	122	7A	172	z	z
27	1B	033	ESC (escape)	59	3B	073	;	;	91	5B	133	[	[	123	7B	173	{	{
28	1C	034	FS (file separator)	60	3C	074	<	<	92	5C	134	\	\	124	7C	174			|
29	1D	035	GS (group separator)	61	3D	075	=	=	93	5D	135	]	]	125	7D	175	}	}
30	1E	036	RS (record separator)	62	3E	076	>	>	94	5E	136	^	^	126	7E	176	~	~
31	1F	037	US (unit separator)	63	3F	077	?	?	95	5F	137	_	_	127	7F	177	&#127;	DEL

Source: www.LookupTables.com

Εικόνα 3.4 ASCII για αντιστοιχία αυτών με το επίπεδο ποιότητας ενός νουκλεοτίδιου

Phred quality scores: Ορίζεται ως η ιδιότητα έκφρασης της πιθανότητας λάθους ενός νουκλεοτιδίου ενός read (base-calling error), συναρτήση του λογαρίθμου. Τα Phred quality scores ορίζονται ως μια ιδιότητα, η οποία είναι λογαριθμικά συσχετιζόμενη με τη πιθανότητα σφάλματος ενός νουκλεοτιδίου, P, ως εξής:

$$Q = -10 \log_{10} P$$

or

$$P = 10^{\frac{-Q}{10}}$$

Phred quality scores are logarithmically linked to error probabilities

Phred Quality Score	Probability of incorrect base call	Base call accuracy
10	1 in 10	90%
20	1 in 100	99%
30	1 in 1000	99.9%
40	1 in 10,000	99.99%
50	1 in 100,000	99.999%
60	1 in 1,000,000	99.9999%

Πίνακας 3.1 Phred quality scores είναι συνδεδεμένες με το λογάριθμο της πιθανότητας λάθους
http://en.wikipedia.org/wiki/Phred_quality_score#Definition

Για παράδειγμα, εάν ένα quality score μίας βάσης είναι 30, η πιθανότητα αυτή η βάση να είναι λανθασμένη αποδίδεται στο 1/1000 ή αλλιώς στο 0.1%. Άρα η ακρίβεια της βάσης αυτής αντιστοιχείται στο 99.9%, όπως καταγράφεται στο πιο πάνω Πίνακα 3.1.

3.2.1.3 Quality Trimming/ Κλάδεμα

Η πιο κοινώς διαδεδομένη μέθοδος αποφυγής χαμηλών ποιοτικά βάσεων μίας (υπο)ακολουθίας DNA, είναι να λαμβάνονται υπόψη μόνο οι βάσεις με quality score μεγαλύτερο ή ίσο με 20 (ή με άλλο επίπεδο quality score το οποίο καθορίζει ο χρήστης-ερευνητής). Για παράδειγμα, υπάρχουν εργαλεία «αποκοπής» ακολουθίας βάσει της ποιότητας των αντίστοιχων στοιχείων αυτής, όπως το

εργαλείο ‘gatk ClipReads’ [36]. Το εργαλείο αυτό παρέχει τη δυνατότητα διαγραφής βάσεων του αριστερού άκρου της ακολουθίας, που έχουν χαμηλή ποιότητα. Υπάρχουν όμως και άλλες προσεγγίσεις [4] κατά τις οποίες γίνεται διαγραφή κάθε νουκλεοτιδίου από όλη την αλληλουχία, με quality score χαμηλότερο του επιθυμητού ή διαγραφή κάθε νουκλεοτιδίου από τα τελευταία Χ νουκλεοτίδια της αλληλουχίας με quality score χαμηλότερο του επιθυμητού (για Χ καθοριζόμενο από το χρήστη).

Στο δικό μας αλγόριθμο λαμβάνονται υπόψη SNPs με quality score μεγαλύτερο από Q , (για Q τόσο όσο επιθυμεί ο χρήστης), για λόγους αποφυγής νουκλεοτιδίων στα άκρα της ακολουθίας με ποιότητα χαμηλότερη ή ίση από Q και για περισσότερη ευελιξία. Συγκεκριμένα, δεν αγνοούνται/ διαγράφονται **όλα** τα νουκλεοτίδια, **από όλη** την ακολουθία, με ποιότητα μικρότερη η ίση του Q, αλλά αφαιρούνται νουκλεοτίδια από τα δύο άκρα της κάθε (υπο)αλληλουχίας ενόσο αυτά έχουν ποιότητα μικρότερη ι ίση από την επιθημητή. Δηλαδή, διασχίζεται η γενετική ακολουθία από τα αριστερά προς τα δεξιά (από την αρχή του read) και ενόσω υπάρχουν νουκλεοτίδια με ποιότητα μικρότερη ή ίση του Q το τρέχον νουκλεοτίδιο που εξετάζεται/διαβάζεται, αγνοείται και μόλις βρεθεί ένα νουκλεοτίδιο με quality score > Q ,τότε η διάσχιση για τη διαγραφή χαμηλών ποιοτικά SNPs τερματίζει. Άρα από το σημείο αυτό σταματά η μέθοδος διαγραφής των νουκλεοτιδίων με quality score <= Q.

Η ίδια μέθοδος εφαρμόζεται διασχίζοντας το γονιδίωμα και από δεξιά προς τα αριστερά (από το τέρμα της ακολουθίας προς την αρχή). Βλέπε Πίνακα 3.2 για την αντιστοιχία χαρακτήρων με το επίπεδο ποιότητας. Αυτό συμβαίνει αφού είναι βιολογικά αποδεδειγμένο ότι στα άκρα μίας ακολουθίας DNA βρίσκονται οι περισσότεροι πολυμορφισμοί, τους οποίος θέλαμε προφανώς να αποφύγουμε.

Εικόνα 3.5 Παράδειγμα διαδικασίας κλαδέματος από τη πλατφόρμα που αναπτύχθηκε (τα στοιχεία με γαλάζιο χρώμα είναι αυτά που κλαδεύονται αφού έχουν ποιότητα <30 βάσει του Πίνακα 3.2)

Τα Phred quality scores χρησιμοποιούνται για:

- Εκτίμηση της ποιότητας ακολουθίας.
- Αναγνώριση και η αφαίρεση της ακολουθίας χαμηλής ποιότητας (αποκοπή/ κλάδεμα από το άκρο της αρχής και του τέλους της ακολουθίας – «clipping»). **→Χρησιμοποιείται στον αλγόριθμο Επεξεργασίας Αλληλουχιών της NGS data analyses platform.**
- Προσδιορισμό της ακρίβειας/ορθότητας της αλληλουχίας.

3.2.1.4 Βηματική Περιγραφή Αλγορίθμου

Βήμα 1: Έλεγχος εισαγωγής ονόματος αρχείου εισόδου και θετικού ακεραίου, έστω file1 και X αντίστοιχα. Αν κάποιο από αυτά δεν έχει εισαχθεί ή δεν έχει εισαχθεί ορθά δώσε μήνυμα λάθους και τερμάτισε, διαφορετικά αρχικοποίησε *num_of_outputFile=1* και προχώρησε στο Βήμα 2.

Βήμα 2: Δημιούργησε ένα αρχείο εξόδου και αν υπάρχουν προς διάβασμα περισσότερες από Χ ακολουθίες γονιδιωμάτων, στο αρχείο εισόδου, αντέγραψε από το αρχείο file1 στο αρχείο εξόδου *num_of_outputFile,* Χ ακολουθίες, κλείσε το αρχείο που δημιούργησες και προχώρα στο Βήμα 3. Αν οι ακολουθίες που απέμειναν προς διάβασμα και αντιγραφή, στο αρχείο εισόδου file1 είναι λιγότερες από Χ, τότε αντίγραψε όσες απέμειναν στο τρέχον αρχείο εξόδου, *num_of_outputFile,* και τερμάτισε.

Βήμα 3: Αύξανε τη μεταβλητή Χ κατά 1000 και το *num_of_outputFile* κατά 1.

Βήμα 4: Αν υπάρχουν ακόμα ακολουθίες που δεν έχουν διαβαστεί και αντιγραφεί από το αρχείο εισόδου file1 επανέλαβε το Βήμα 2, αν όχι τερμάτισε.

3.2.2 Αλγόριθμος Κλαδέματος Γενετικών Ακολουθιών

Ο αλγόριθμος για την κωδικοποιημένη αποθήκευση και επεξεργασία των ακολουθιών DNA ονομάστηκε «Αλγόριθμος Trimming» και υλοποιήθηκε έτσι ώστε να διεκπεραιώνει το στόχο κατά τον οποίο επιθυμείται δέσμευση ελάχιστης δυνατής μνήμης για αποθήκευση των ακολουθιών εισόδου, μετά από τη κωδικοποίησή τους, και για επεξεργασία αυτών. Ταυτόχρονα επιδιώκεται η αποφυγή πιθανότητας για sequence losing quality, με τη στρατηγική που αναλύθηκε ήδη στο υποκεφάλαιο 3.2.1.1.

Πολλές από τους τεχνολογίες που αναπτύχθηκαν, όπως η «GenCompress» μέθοδος των Xin Chen, Ming Li, Bin Ma και John Tromp (Bioinformatics Laboratory, Computer Science Department, University of California, Santa Barbara) [46] και αυτή των Markus Hsi-Yang Fritz, Rasko Leinonen, Guy Cochrane και Ewan Birney (European Molecular Biology Laboratory's European Bioinformatics Institute (EMBL-EBI), Wellcome Trust Genome Campus, Hinxton, Cambridgeshire CB10 1SD, United Kingdom) [28] , αποτελούν μία λιγότερο δαπανηρή τεχνολογία προσδιορισμού, χαρτογράφησης και αποθήκευσης της αλληλουχίας, στην οποία

αντιμετωπίστηκαν προβλήματα που αφορούσαν τη ποιότητα ακολουθίας και τον μεγάλο όγκο μνήμης που απαιτεί η αποθήκευσή της. Οι πλύστες μέθοδοι ακλουθούν το μηχανισμό «κλαδέματος» (sequence trimming/clipping) βάσει του Phred Score [36] , όπως έπραξα και εγώ με τη βοήθεια των καθηγητών μου στον αλγόριθμο αυτό, για αντιμετώπιση των ποιοτικά κακών νουκλεοτιδίων. Το ίδιο υποστηρίζεται και κατά την έρευνα των Heng Li, Jue Ruan και Richard Durbin («Mapping short DNA sequencing reads and calling variants using mapping quality scores», The Wellcome Trust Sanger Institute, Hinxton CB10 1SA, United Kingdom;Beijing Genomics Institute, Chinese Academy of Science, Beijing 100029, China) [19]. Όσον αφορά την αποθήκευση της ακολουθίας χρησιμοποιήθηκε μία εύκολη, κατανοητή και αποδοτική διαδικασία που εξηγείται πιο κάτω στο υποκεφάλαιο 3.2.2.1.

Στη πλατφόρμα λοιπόν που υλοποίησα, πριν τον αλγόριθμο επεξεργασίας ακολουθιών, για την αποθήκευση τους, προηγήθηκε μία μεθοδολογία κωδικοποίησης των DNA ακολουθιών και του quality score του κάθε SNP της ακολουθίας, η οποία περιγράφεται στη συνέχεια σε αυτό το υποκεφάλαιο, χρησιμοποιώντας το γνωστό Phred Score [36], για χρήση μόνο των ποιοτικά καλών ακολουθιών/read και αφαίρεσης των ποιοτικά κακών (δηλαδή αυτών με χαμηλό επίπεδο ποιότητας).

Αρχικά, για την ανάπτυξη του αλγόριθμου υλοποιήθηκαν κάποιες βιβλιοθήκες/κλάσεις, που βοήθησαν στην επίτευξη του στόχου αυτού του αλγορίθμου, στην επαναχρησιμοποίηση και στην ορθή δόμηση. Αυτές αναλύονται πιο κάτω:

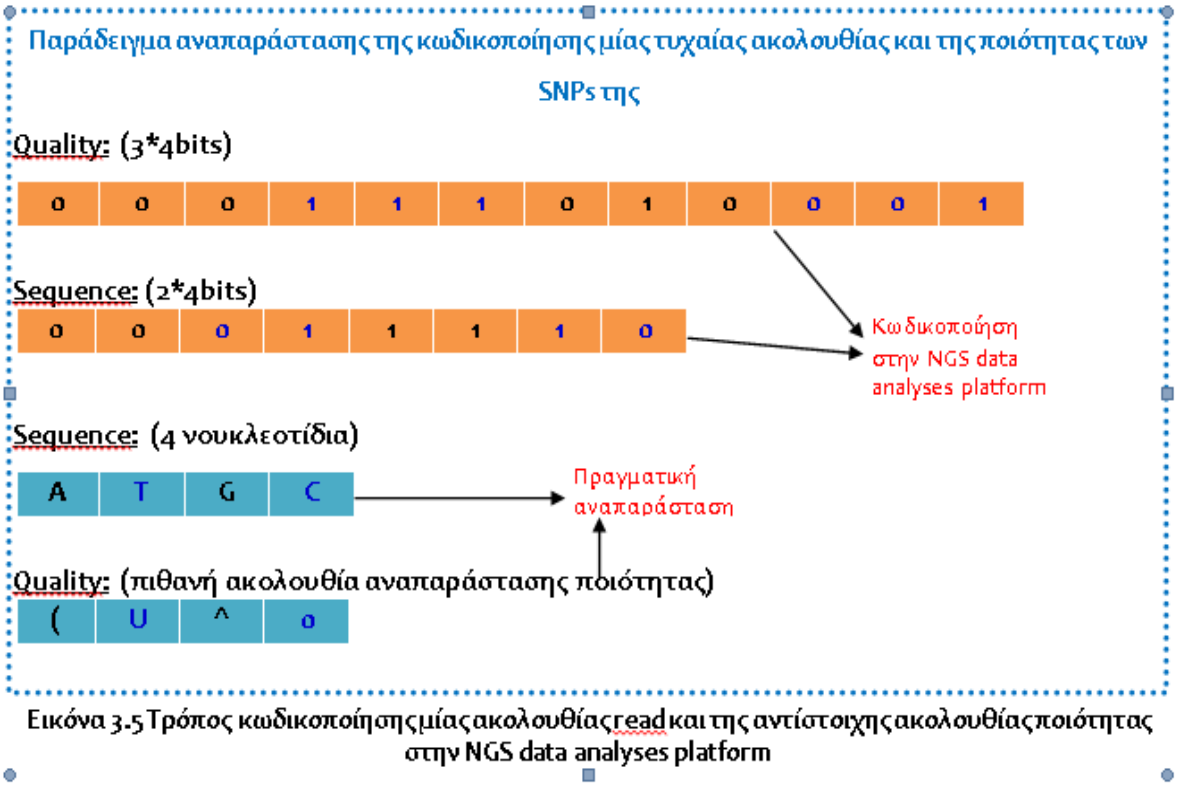
3.2.2.1 Βιβλιοθήκες Αλγόριθμου Επεξεργασίας Αλληλουχιών

Βιβλιοθήκη «read.h»:

Στη βιβλιοθήκη αυτή υλοποιήθηκε μία νέα κλάση «Reads» που αντιπροσωπεύει τη κωδικοποίηση και τη δομή αποθήκευσης του κάθε bp (base pair) [31] και του αντίστοιχου quality score του [14], για ένα read ενός DNA.
Συγκεκριμένα, η κλάση δομείται από 2 bitsets, ένα για αναπαράσταση των base pair μίας ακολουθίας/ενός read και ένα για αναπαράσταση των αντίστοιχων quality scores. Το πρώτο bitset περιέχει 300 * 2 bits για αποθήκευση των κωδικοποιημένων bps και το δεύτερο 300 * 3 bits για αποθήκευση των quality estimate των αντίστοιχων bps του DNA ενός ατόμου. Το μήκος των δομών bitsets (300), αλλάζει ανάλογα με τη τιμή που επιθυμεί ο χρήστης καθώς το μήκος αποτελεί καθολική μεταβλητή μέσα στον αλγόριθμο (read_length).

Αναλογιζόμενοι ότι τα νουκλεοτίδια ενός DNA είναι τέσσερα (η αδενίνη (A), η θυμίνη (T), η γουανίνη (G) και η κυτοσίνη (C) – βλέπε υποκεφάλαιο 2.1) για την κωδικοποίησή τους απαιτούνται 2 bits ($4=2^2$ διαφορετικά νουκλεοτίδια). Η συγκεκριμένη κωδικοποίηση επιλέχθηκε για λόγους γρήγορης αναζήτησης, ανάλυσης των bps ενός read και για δέσμευση ελάχιστης δυνατής μνήμης.

Κωδικοποίηση των SNP: A=00, T=01, C=10, G=01, (N=00 – for missing SNP).
Κωδικοποίηση των quality score του κάθε SNP: βλέπε Πίνακα 3.2.



Εικόνα 3.5 Τρόπος κωδικοποίησης μίας ακολουθίας read και της αντίστοιχης ακολουθίας ποιότητας στην NGS data analyses platform

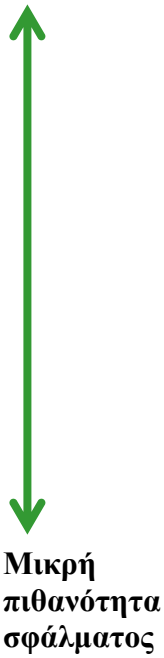
Επιπρόσθετα, στη δομή της κλάσης αυτής, υπάρχει και ένας μονοδιάστατος πίνακας από structs τύπου (end_id, end_bsPosition) που αντιστοιχούν στο μοναδικό αναγνωριστικό της υπο-ακολουθίας DNA με το οποίο το συγκεκριμένο read, έστω A, έχει κάποια επικάλυψη (το τέλος της ακολουθίας A με την αρχή του read με αναγνωριστικό= end_id). Δηλαδή ο πίνακας αυτός, αντιπροσωπεύει το πλήθος των reads με τα οποία το τρέχον read A έχει κάποιο ταίριασμα. Έτσι, στο πίνακα αυτόν αποθηκεύονται οι πληροφορίες που μας ενδιαφέρουν, αναφορικά με το matching/ overlapping των υπο-ακολουθιών, κατά τη διαδικασία ευθυγράμμισης.

Παραπέμποντας στο υποκεφάλαιο 3.2.1.1, προσθέτω ότι για την αποφυγή απώλειας της υψηλής ποιότητας της αλληλουχίας, που παράγεται τελικά από την NGS data analyses platform, εφαρμόζεται η μεθοδολογία «κλαδέματος» ή αλλιώς η «αποκοπής» [38], από τα άκρα της αλληλουχίας, χαμηλών ποιοτικά βάσεων (με ποιότητα μικρότερη ή ίση από αυτή που εισάγει ο χρήστης, με βάση το Phred Quality Score [36]) . Με αυτό τον τρόπο διασφαλίζουμε καλύτερα την μείωση

πιθανότητας εσφαλμένων βάσεων σε όποιο βαθμό επιθυμεί ο χρήστης (αφού αυτός αποφασίζει το βαθμό «κλαδέματος» [31] των ακολουθιών reads), παρά με τη στρατηγική «κλαδέματος» μόνο των βάσεων με ποιότητα μικρότερη ή ίση με 20 και έτσι έχουμε καλύτερα ποιοτικά ακολουθίες.

Για σκοπούς κωδικοποίησης του quality score των SNPs,κατηγοριοποιήσαμε το Phred Quality Score σε 7 ομάδες (βλέπε Πίνακα 3.2), ανάλογα με το χαρακτήρα σε ASCII. Για το λόγω αυτό χρειαστήκαμε 3 bit για τη κωδικοποίηση ενός SNP (αφού θέλουμε τιμές στο εύρος [0-7] για τις 7 κατηγορίες ποιότητας, τότε στο δυαδικό σύστημα θα χρησιμοποιήσουμε τις τιμές στο εύρος [000-111] αντίστοιχα).

Διάστημα χαρακτήρων κατηγορίας	**Phred Quality Score**	**Κωδικοποίηση**	**Διάστημα κατηγορίας σε ASCII code**	**Χαρακτήρας Εξόδου**
Missing	------	000	------	'o'
'!' – '*'	10	001	33-42	'!'
'+' – '4'	20	010	43-52	'+'
'5' – '>'	30	011	53-62	'5'
'?' – 'H'	40	100	63-72	'?'
'I' – 'R'	50	101	73-82	'I'
'S' – '\'	60	110	83-92	'S'
']' – '~'	70 και άνω	111	93 και άνω	'J'



Πίνακας 3.2 Relation of Phred Quality Score and ASCII characters based on FASTQ Protocol.

Έτσι επιτυγχάνουμε χαμηλότερο fail rate, αφού κατά τον αλγόριθμο 2 – «Αλγόριθμος Trimming», γίνεται διαγραφή των βάσεων των οποίων η ποιότητα – η πιθανότητα λάθους είναι μεγάλη (η πιθανότητα εγγύησης ορθότητας είναι μικρή). Το πρόβλημα αυτό αποτέλεσε μειονέκτημα σε πολλές μελέτες για αυτό και δεν είχαν τα επιθυμητά αποτελέσματα, όπως φαίνεται και από τη μελέτη των David Altshuler, Victor J. Pollara, Chris R. Cowles, William J. Van Etten, Jennifer Baldwin, Lauren Linton & Eric S. Lander (Whitehead Institute/MIT Center for Genome Research, Nine Cambridge Center, Cambridge, Massachusetts 02142, USA / Diabetes Unit, Department of Medicine, Massachusetts General Hospital, Harvard Medical School, Boston, Massachusetts 02114, USA / Department of Biology, Massachusetts Institute of Technology, Cambridge, Massachusetts 02139, USA) [9]. Στην έρευνα αυτή επισημαίνεται ότι για το ταίριασμα δύο υπο-ακολουθιών single nucleotide polymorphisms (SNPs) σημαντικό ρόλο έχει η ποιότητα αυτών, για το λόγο αυτό οι συγγραφείς της μελέτης αυτής, επιχείρησαν να αναπτύξουν ένα μηχανισμό «reduced representation shotgun (RRS) sequencing», το οποία θα αντιμετώπιζε τα διάφορα εμπόδια όπως η χαμηλή ποιότητα ακολουθιών και η ακρίβεια στο ταίριασμα δύο αλληλουχιών.

Βιβλιοθήκη «fastq.h»:

Η βιβλιοθήκη αυτή φορτώνει ένα αρχείο εισόδου Fastq File, το οποίο δίνει ο χρήστης και αποθηκεύει στη RAM όλα τα γενετικά δεδομένα που περιέχει το αρχείο εισόδου, μέσα στη κλάση που υλοποιεί η βιβλιοθήκη «fastq.h» (έτσι δημιουργείται ένα αντικείμενο fastq που περιέχει πολλά αντικείμενα της βιβλιοθήκης/ κλάσης read.h). Ακολούθως, για κάθε 4 γραμμές του αρχείου εισόδου, οι πληροφορίες που αφορούν ένα συγκεκριμένο read (ακολουθία και τιμές ποιότητας της ακολουθίας) αποθηκεύονται μέσα σε μία δομή, αφού αρχικά κωδικοποιηθούν με τη μέθοδο που αναλύθηκε πιο πάνω (βλέπε σε αυτό το υποκεφάλαιο 3.2.2.1 - βιβλιοθήκη «read.h»). Ακόμη, αυτή η βιβλιοθήκη χρησιμοποιείται για τη δημιουργία του αρχείου εξόδου.

3.2.2.2 Βηματική Περιγραφή Αλγορίθμου

Βήμα 1: Έλεγχος εισαγωγής ενός ονόματος αρχείου εισόδου και ενός χαρακτήρα που θα αντιπροσωπεύει μία ποιότητα, έστω input_file.fastq και X αντίστοιχα. Αν δεν έχει εισαχθεί ή δεν υπάρχει τέτοιο αρχείο δώσε μήνυμα λάθους και τερμάτισε, διαφορετικά προχώρησε στο Βήμα 2.

Βήμα 2: Δημιούργησε αντικείμενο της κλάσης fastq.h και αποθήκευσε κωδικοποιημένα, διαβάζοντας γραμμή προς γραμμή το αρχείο εισόδου, τις πληροφορίες για το κάθε τρέχον read που διαβάζεται (ακολουθία νουκλεοτιδίων και αντίστοιχη quality score sequence). Αναλυτικότερα, για κάθε SNP = A αποθήκευσε στο αντίστοιχο bitset «00», για κάθε SNP = T αποθήκευσε στο αντίστοιχο bitset «01», για κάθε SNP = C αποθήκευσε στο αντίστοιχο bitset «10» και για κάθε SNP = G αποθήκευσε στο αντίστοιχο bitset «11». **Για SNP =N (missing data) αποθήκευσε στο αντίστοιχο bitset «00».** Όταν όλη η κωδικοποίηση του αρχείου τερματίσει προχώρα στο Βήμα 3.

Βήμα 3: Για κάθε κωδικοποιημένη ακολουθία γίνεται η εφαρμογή του «κλαδέματος», όπως περιγράφτηκε πιο πάνω. Διασχίζεται το bitset του quality της ακολουθίας **από τη ουρά προς την αρχή και αντίστροφα**. Για κάθε SNP που δεν αγνοείται λόγω ψηλής ποιότητας αντιγράφεται μαζί με το αντίστοιχο χαρακτήρα ποιότητας σε ένα αρχείο εξόδου. Αν **το πρώτο** SNP που θα εξεταστεί, έχει ποιότητα με βάσει το Phred Quality Score, μικρότερη η ίση με X, τότε διαγράφεται αυτό και η αντίστοιχη ποιότητά του από το bitset και συνεχίζεται η διάσχιση, επαναλαμβάνοντας το Βήμα 3. Αν υπάρξει τρέχον εξεταζόμενο SNP, για το οποίο ισχύει ότι η ποιότητά του είναι μεγαλύτερη από X, το κλάδεμα τερματίζει και δεν αγνοούμε άλλα νουκλεοτίδια.

3.2.3 Αλγόριθμος Παραλληλοποίησης

Για την αποδοτική εφαρμογή του νέου aligner αλγόριθμου που αναλύεται στο υποκεφάλαιο 3.2.4, εκτελείται ο αλγόριθμος «runSubFiles», ο οποίος λαμβάνει σαν είσοδο ένα πλήθος αρχείων εισόδου (τα οποία αποτελούν έξοδο του αλγορίθμου2, «Αλγόριθμος Trimming»). Τα FASTQ αρχεία εισόδου (με τις «κλαδεμένες» ακολουθίες [36] και τις αντίστοιχες ποιότητες των νουκλεοτιδίων τους) περιέχουν γενετικά δεδομένα, μετά από επεξεργασία (κωδικοποίηση και «κλάδεμα» - βλέπε υποκεφάλαιο 3.2.2), συνδυάζονται για **να συγκριθεί κάθε γενετική ακολουθία κάθε αρχείου με τις ακολουθίες κάθε άλλου αρχείου εισόδου** μέσο του βελτιστοποιημένου αλγορίθμου ευθυγράμμισης [28] , που περιγράφεται στο επόμενο υποκεφάλαιο 3.2.4.

Η κύρια λοιπόν λειτουργία αυτού του αλγορίθμου, είναι η παραγωγή scripts files [13] ,τα οποία θα καλούν τον αλγόριθμο Ευθυγράμμισης Αλληλουχιών για σύγκριση του περιεχομένου όλων των αρχείων με κάθε αρχείο εισόδου. Άρα, ο «aligner algorithm» καλείται μία φορά για κάθε ζεύγος αρχείων εισόδου μέσω των απαιτούμενων (sub files in SLURM για το Cy-Tera server [27]), που θα δημιουργούνται και θα εκτελούνται αυτόματα, από τη πλατφόρμα, για aligning των ακολουθιών των δύο ζητούμενων αρχείων εισόδου. Αυτός ο αλγόριθμος θα ονομάζεται «runSubFiles algorithm». Για κάποιο αριθμό αρχείων εισόδου θα έχουμε το κάλεσμα του «aligner algorithm» για κάθε δυνατό ζεύγος αρχείων (κάθε αρχείο με κάθε άλλο και με τον εαυτό του). Αν για παράδειγμα τα αρχεία εισόδου είναι 10 θα υπάρχουν 55 ζεύγη αρχείων (($\binom{10}{2}$)+ 10 δυνατά ζεύγη - κάθε αρχείο συνδυάζεται με όλα τα υπόλοιπα και τον εαυτό του), για να συγκριθούν μεταξύ τους (για N αρχεία εισόδου έχουμε (($\binom{N}{2}$)+ N ζεύγη αρχείων σύγκρισης).

Στα scripts files μπορεί να καθοριστεί ο επιθυμητός αριθμός nodes και των διεργασιών ανά κόμβο που θα δεσμεύσουμε. Επιπρόσθετα, ο χρήστης δύναται να καθορίσει από το κώδικα τον ελάχιστο επιθυμητό αριθμό νουκλεοτιδίων επικάλυψης που θα δοθεί σαν είσοδο στις κλίσεις του αλγορίθμου

ευθυγράμμισης μέσα στα scripts files. Για κάθε 12-αδα ζευγών δημιούργησε ένα script files καλώντας τον αλγόριθμο «aligner» για κάθε ένα από αυτά.

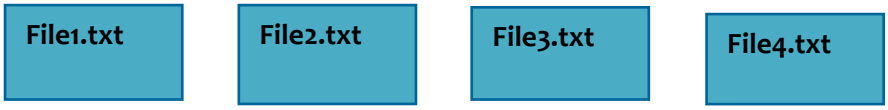
3.2.3.1 Βηματική Περιγραφή Αλγορίθμου

Βήμα 1: Έλεγχος εισαγωγής πλήθος ονομάτων αρχείων εισόδου, έστω file1.out, file2.out, …, fileN.out. Αν δεν έχει εισαχθεί τουλάχιστο ένα όνομα αρχείου, δώσε μήνυμα λάθους και τερμάτισε, διαφορετικά προχώρησε στο Βήμα 2.

Βήμα 2: Δημιούργησε κάθε δυνατό ζεύγος αρχείων (ζεύγος 1: file1.out, file1.out, ζεύγος 2: file1.out, file2.out, ζεύγος 3: file1.out, file3.out, . . ., ζεύγος N: fileN.out, fileN.out) και για κάθε 12-αδα ζευγών δημιούργησε ένα script files καλώντας τον αλγόριθμο «aligner» για κάθε ένα από αυτά. Αν απομείνει ομάδα ζευγών με πλήθος μικρότερο από 12 ζεύγη, δημιούργησε ένα script file καλώντας τον αλγόριθμο «aligner» για κάθε ζεύγος από την ομάδα αυτή. Για κάθε script file που δημιουργείται, τρέξε το αυτόματα.

Βήμα 4: Όταν δημιουργηθούν όλα τα δυνατά script files τερμάτισε.

Input files:



Script File:

… (άλλες αναγκαίες πληροφορίες του αρχείου) …
./aligner File1.txt,File1.txt, 7
./aligner File1.txt, File2.txt, 7
./aligner File1.txt, File3.txt, 7
./aligner File1.txt, File4.txt, 7
./aligner File2.txt, File2.txt, 7
./aligner File2.txt, File3.txt, 7
./aligner File2.txt, File4.txt, 7
./aligner File3.txt, File3.txt, 7
./aligner File3.txt,File4.txt, 7
./aligner File4.txt, File4.txt, 7

Για κλήση του αλγόριθμου RunSubFiles: .\RunSubFiles.out File1.txt, File2.txt, File3.txt,

Εικόνα 3.6 Παράδειγμα εξόδου του αλγορίθμου RunSubFiles της NGS data analyses platform με είσοδο 4 αρχεία

3.2.4 Αλγόριθμος Ευθυγράμμισης Αλληλουχιών

3.2.4.1 Θεωρητική Επισκόπηση Αλγορίθμων Ευθυγράμμισης

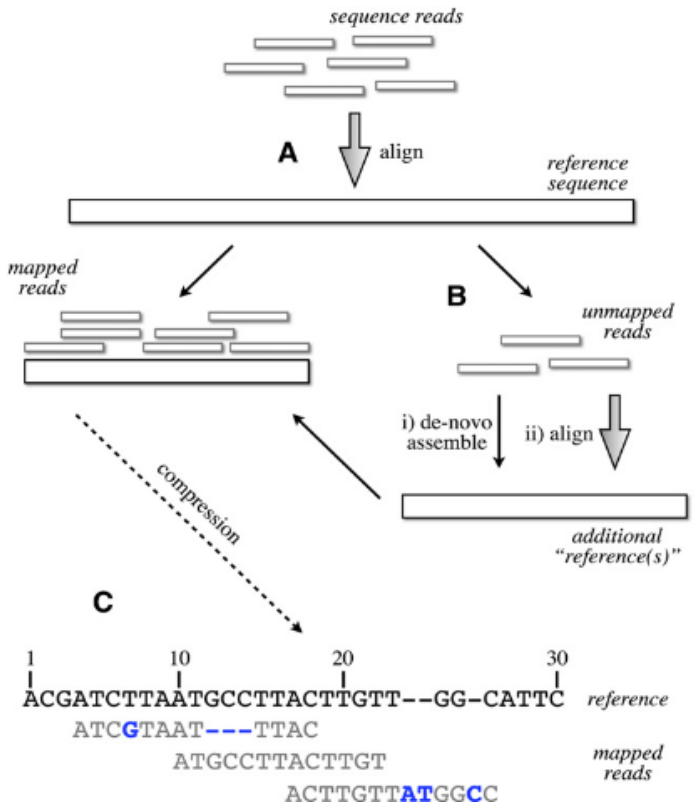
Στο συγκεκριμένο τμήμα της πλατφόρμας παρουσιάζεται ιδιαίτερα η ευελιξία στην επεκτασιμότητα και τη προσαρμογή της πλατφόρμας σε οποιοδήποτε βιολογικό πρόβλημα. Παραδείγματος χάριν, αν δε μας ενδιέφερε η διαδικασία ευθυγράμμισης γενετικών ακολουθιών [10] , αλλά κάποια άλλη επεξεργασία αυτών, θα μπορούσαμε να αντικαταστήσουμε το κομμάτι αυτό, δηλαδή το τμήμα κατά το οποίο εφαρμόζεται ο αλγόριθμος 4- «Aligner Algorithm», με τον αντίστοιχο επιθυμητό αλγόριθμο. Αυτό θα έχει σαν αποτέλεσμα τη δυνατότητα παράλληλης εκτέλεσης του επιθυμητού αλγορίθμου σε High Performance Computing (HPC) μηχανή, γρηγορότερα και με χαμηλότερο fail rateαφού κατά τον αλγόριθμο 2 – «Αλγόριθμος Trimming», γίνεται διαγραφή των βάσεων των οποίων η ποιότητα – η πιθανότητα λάθους είναι μεγάλη (η πιθανότητα εγγύησης είναι μικρή).

Αξιοσημείωτο είναι η το γεγονός που ωθεί και θα συνεχίσει να ωθεί εκατομμύρια ερευνητές της ιατρικής και της πληροφορικής να ασχολούνται με το θέμα της ευθυγράμμισης και της χαρτογράφησης γενετικών ακολουθιών. Το γεγονός αυτό είναι ότι οι περισσότερες γονιδιακές μεταβολές οφείλονται σε οφείλεται σε σημειακούς πολυμορφισμούς νουκλεοτιδίων (SNPs) , επομένως επιχειρείται η υψηλή γενετική ανάλυση με στόχο τον εντοπισμό των γονιδίων που προκαλούν διάφορες ασθένειες. Με βάσει προτάσεις που έγιναν [9], η γονιδιακή περιοχή στο διάστημα 5’ – 3’ νουκλεοτιδίων μπορεί να χρησιμοποιηθεί για ανίχνευση των ανθρώπινων γονιδίων που συνδέονται με κοινές ασθένειες, όπως για παράδειγμα ο καρκίνος, αλλά και την εξέταση πατρότητας.

3.2.4.2 Αλγόριθμος Ευθυγράμμισης της NGS data analyses platform

Για τη διπλωματική αυτή ανέπτυξα, με τη βοήθεια των καθηγητών μου, μία νέα διαλλακτική και καινοτόμα μεθοδολογία ευθυγράμμισης ανθρώπινων γενετικών ακολουθιών DNA, της οποίας η κύρια ιδέα στηρίζεται στον αλγόριθμο Dot Matrix Sequence Alignment [10], η οποία περιγράφεται πιο κάτω και χρησιμοποιείται στη χαρτογράφηση ακολουθιών DNA (mapping) , αλλά με κάποια ουσιαστικά optimizations για μείωση χρόνου εκτέλεσης, απαιτούμενης μνήμης και για ευκολία μελλοντικών αλλαγών.

Γενικότερα, η πιο βασική εργασία στην ανάλυση αλληλουχιών είναι η ευθυγράμμιση δύο ακολουθιών σε ζεύγη και η εύρεση τμημάτων των δύο αλληλουχιών που συσχετίζονται μεταξύ τους. Αυτό αποτελεί κάτι το θετικό όσον αφορά το computational sequence analysis, αφού το ποσοστό συσχέτισης μεταξύ δύο ακολουθιών (ή και πρωτεϊνών) αποτελεί κριτήριο αξιολόγησης της ομοιότητας (ή της ανεξαρτησίας) που υπάρχει μεταξύ τους και κατ' επέκταση, αν οι δύο ακολουθίες είναι ομόλογες. Αν είναι ομόλογες, τότε μπορούμε να καταλήξουμε σε συμπεράσματα που αφορούν τη δομή και τη λειτουργία τους. Με τους aligner algorithms μπορούμε να αναπαραστήσουμε το DNA στη μνήμη, ενώνοντας τα τμήμα τα του DNA που ήδη έχουμε αποθηκεύσει και να βρούμε αν δύο αλληλουχίες είναι κοινές, παρόμοιες, ομόλογες, ή έχουν κοινό πρόγονο και έτσι να τις χαρτογραφήσουμε πάνω σε όλη την ακολουθία γονότυπου.



Εικόνα 3.7 Γραφική αναπαράσταση χαρτογράφησης αλληλουχίας DNA μέσο της ευθυγράμμισης των γενετικών τμημάτων αυτού
http://genome.cshlp.org/content/21/5/734.full.pdf+html

3.2.4.3 Dot Matrix Sequence Alignment

Η ευθυγράμμιση αλληλουχίας είναι μία διαδικασία της σύγκρισης δύο ή περισσότερων αλληλουχιών (pair - wise alignment) για την αναζήτηση για μίας σειράς από επιμέρους χαρακτήρες [11] .Υπάρχουν δύο τύποι ευθυγράμμισης: αυτή σε τοπικό και αυτή σε παγκόσμιο επίπεδο.

• Σε παγκόσμιο επίπεδο, γίνεται μία προσπάθεια να ευθυγραμμιστεί **ολόκληρη** η αλληλουχία. Εάν δυο αλληλουχίες έχουν περίπου το ίδιο μήκος και είναι αρκετά παρόμοιες (κατά ένα ποσοστό το οποίο συνήθως καθορίζει ο χρήστης), είναι κατάλληλα για αυτό το τύπο ευθυγράμμισης.

• Σε τοπικό επίπεδο ευθυγράμμισης επικεντρωνόμαστε στην εξεύρεση **τμημάτων** των αλληλουχιών με υψηλό επίπεδο ταιριάσματος (high level of

matches). Δηλαδή, καθορίζεται κατά πόσον δύο τμήματα ακολουθιών ταιριάζουν/ επικαλύπτονται κατά ένα αποδεκτό ποσοστό.

Sequence Alignment:

Κατά τη διαδικασία ευθυγράμμισης πολλών γενετικών υπο-ακολουθιών με στόχο να βρεθεί η αρχική ολοκληρωμένη γενετική αλληλουχία (DNA sequence), υπάρχουν **δύο μέθοδοι επίτευξη**ς του στόχου.

1) **Align:**

   Η **πρώτη είναι έχοντας στη διάθεση ολόκληρο των γενετικό χάρτη (όλη την γενετική αλληλουχία),** την οποία διαθέτουμε στη βάση δεδομένων μας, σε κομματάκια (πολυμερίζουμε την αρχική αλληλουχία DNA), προσπαθούμε να τοποθετήσουμε στη σωστή θέση πάνω στο γενετικό χάρτη κάθε ένα κομμάτι της βάσης μας. Η διαδικασία αυτή ονομάζεται απλή ευθυγράμμιση αλληλουχιών σε ολόκληρο το ανθρώπινο γονιδίωμα, διαθέτοντας το γενετικό χάρτη.

2) **Align De novo Assemble:**

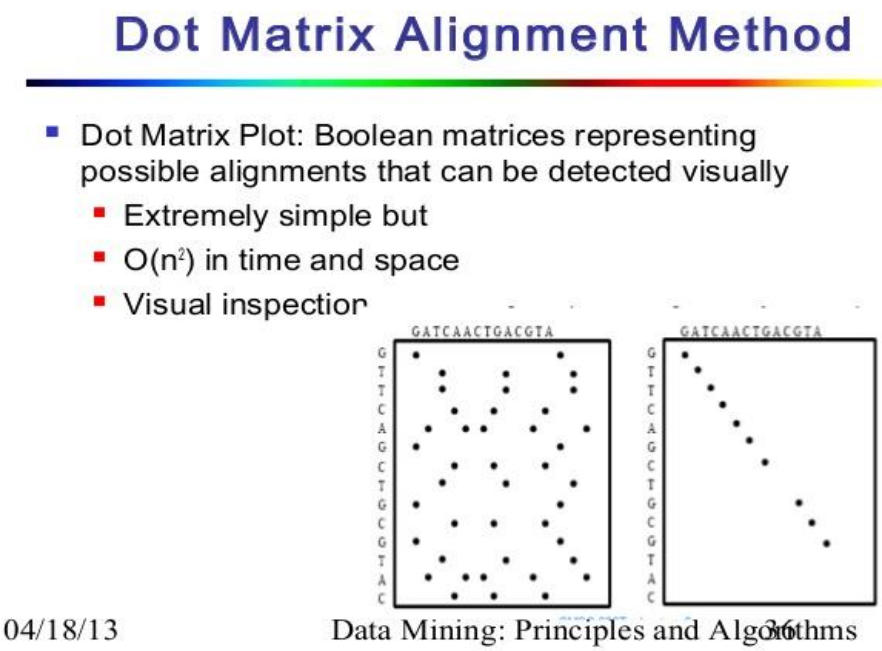
   Η δεύτερη μέθοδος, η οποία ακολουθείται από τη νέα πλατφόρμα NGS data analyses, είναι η ευθυγράμμιση των κομματιών της αρχικής αλληλουχίας DNA (εικόνα3.7), χωρίς όμως να έχουμε στη διάθεσή μας τον γενετικό χάρτη. Επομένως, ουσιαστικά **προσπαθούμε να «μαντέψουμε» την σωστή σειρά τοποθέτησης των γενετικών τμημάτων/ κομματιών, ώστε να μας δώσουν τον αρχικό γενετικό χάρτη.**

Dot Matrix Sequence Alignment :

Με βάσει τη μέθοδο Dot Matrix Sequence Alignment [39, 40] , λαμβάνοντας δύο ακολουθίες (πχ. reads) μήκους M και N, αντίστοιχα, τοποθετούνται με μία δομή πίνακα μεγέθους M*N, η μία κάθετα, πριν τη πρώτη στήλη, και η άλλη οριζόντια, πριν τη πρώτη γραμμή. Έτσι κάθε κελλί [α,β] αντιπροσωπεύει τη σύγκριση των

νουκλεοτιδίων α της κάθετης ακολουθίας και β της οριζόντιας. Ακολούθως, διασχίζουμε τη δομή κελί προς κελί συγκρίνοντας κάθε base/ γράμμα της μίας ακολουθίας με κάθε base/ γράμμα της άλλης. **Αν και μόνο αν** συμπίπτουν (είναι ίδια) το σημειώνουμε (πχ. βάζοντας τη τιμή 0 στα ανόμοια νουκλεοτίδια και 1 στα όμοια) στην αντίστοιχη συντεταγμένη ([γραμμή, στήλη]). Αυτό γίνεται μέχρις ότου να συγκριθεί κάθε στοιχείο της μίας ακολουθίας με κάθε άλλο της δεύτερης ακολουθίας.

Τέλος, η μέγιστη επικάλυψη/ ταίριασμα μεταξύ των δύο ακολουθιών θα είναι το μήκος της διαγωνίου με **συνεχόμενες** σημειωμένες συντεταγμένες με 1, δηλαδή ως κοινά/ίδια στοιχεία [48] . Η δυσκολία σε αυτό τον αλγόριθμο ευθυγράμμισης υφίσταται όταν οι ενιαίες διαγώνιοι που προκύπτουν είναι περισσότερες από μία και το μειονέκτημα του είναι η χρονική πολυπλοκότητά του και ο απαιτούμενος όγκος μνήμης, O(M*N) (ή για 2 ακολουθίες ίδιου μήκους N θα είναι $O(N^2)$).



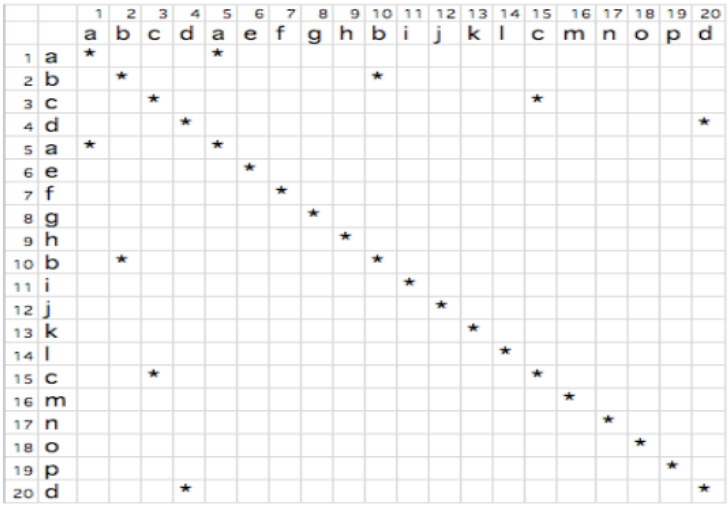
Εικόνα3.8 Παράδειγμα Dot Matrix Sequence Alignment

http://image.slidesharecdn.com/084-130418102030-phpapp02/95/chapter-84-data-mining-concepts-and-techniques-2nd-ed-slides-han-kamber-36-38.jpg?cb=1366298475

(Demonstration A6, Sequence 1 vs. 2)

abcdaefghbijklcmnopd

abcdaefghbijklcmnopd



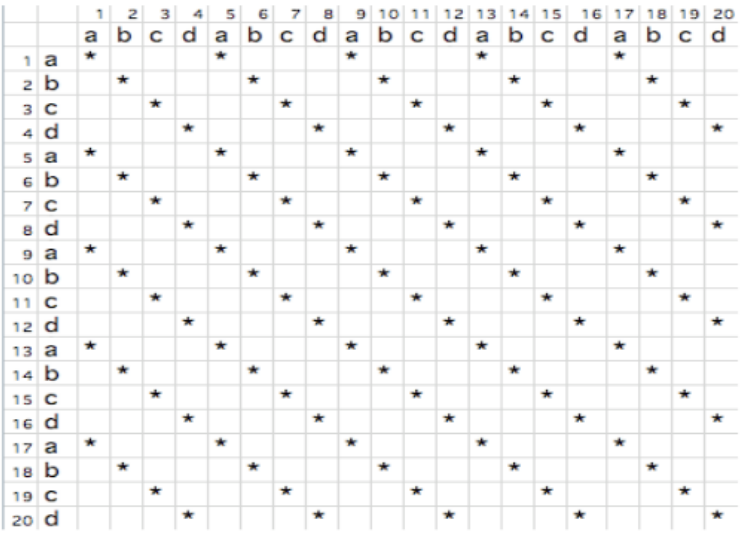
Εικόνα 3.9 Παράδειγμα αλγόριθμου Dot Matrix Sequence Alignment

http://www.srmuniv.ac.in/sites/default/files/files/5%286%29.pdf

(Demonstration A6, Sequence 5 vs. 5)

abcdabcdabcdabcdabcd

abcdabcdabcdabcdabcd

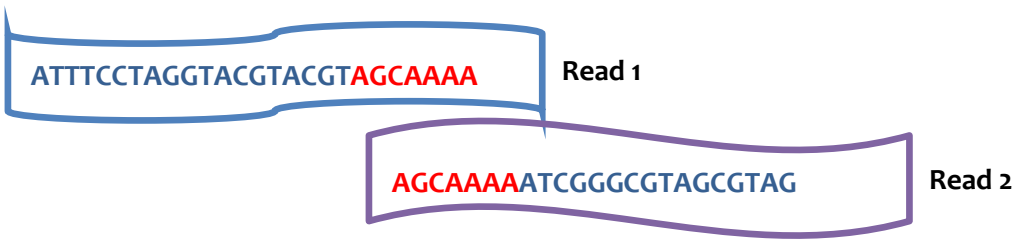


Εικόνα 3.10 Παράδειγμα αλγόριθμου Dot Matrix Sequence Alignment με περισσότερες από μία διαγώνιο ταιριάσματος/ επικάλυψης
http://www.srmuniv.ac.in/sites/default/files/files/5%286%29.pdf

3.2.4.4 NGS data analyses platform «aligner algorithm»

Κατά την εύρεση ενός βέλτιστου aligner αλγορίθμου που να στηρίζεται στη μέθοδο Dot Matrix Sequence Alignment [13] , μας ενδιέφερε να βρούμε αν ένα read (κομμάτι αλληλουχίας DNA) επικαλύπτει ένα άλλο έτσι ώστε ένα τμήμα της ουράς/του τέλους του πρώτου να αποτελεί τμήμα της αρχής του άλλου κατά X νουκλεοτίδια ή αντίστροφα. Το X είναι ο ελάχιστος αριθμός επικάλυψης νουκλεοτιδίων σε SNPs (minimum SNPs overlap/ matching) μεταξύ δύο reads και δίνεται από το χρήστη ως είσοδο στον αλγόριθμο (βλέπε υποκεφάλαιο 3.1). Αν, για παράδειγμα, ψάχνουμε τις ακολουθίες που επικαλύπτονται κατά 3 νουκλεοτίδια τουλάχιστον, τότε X=3, άρα κατά 6 bits, αφού με βάσει τη κωδικοποίηση που έγινε πριν την αποθήκευση των ακολουθιών και εξηγήθηκε στο υποκεφάλαιο 3.2.2 απαιτούνται 2 bits ανά νουκλεοτίδιο → 2*3=6).

Στον αλγόριθμο που υλοποίησα αντιμετωπίζονται, επίσης, περιπτώσεις κατά τις οποίες υπάρχουν πολλές πιθανές επικαλύψεις μεταξύ δύο συγκεκριμένων ακολουθιών. Αν η αρχή μίας ακολουθίας A επικαλύπτει την ουρά μίας άλλης B, για X ή περισσότερα SNPs και ταυτόχρονα η ουρά της A επικαλύπτει την αρχή της ακολουθίας B, τότε γίνονται αποδεκτές και οι δύο περιπτώσεις σαν αποδεκτές επικαλύψεις.



Εικόνα 3.11 Παράδειγμα επικάλυψης μεταξύ 2 ακολουθιών – για minimum nucleotides overlap = 7 (διαφορετικά 14 SNPs).

Έπειτα, αφού εφαρμοστεί η μέθοδος ευθυγράμμισης, παράγεται το αρχείο εξόδου με καταγεγραμμένα:

1. Τον αριθμό νουκλεοτιδίων επικάλυψης (πρέπει να είναι > ή = με X).
2. Το αναγνωριστικό της ακολουθίας, της οποίας η αρχή επικαλύπτεται.
3. Το αναγνωριστικό της ακολουθίας, της οποίας η ουρά επικαλύπτεται.

3.2.4.3 Αρχείου Εξόδου του aligner αλγορίθμου για DNA ακολουθίες

Μορφή αρχείου εξόδου:

overlap neucleotypes	<file name 1>	<file name 2>
<# of overlapping neucleotypes>	<read id of file 1>	<read id of file 2>
. . .	. . .	. . .

Πίνακας 3.3 Μορφή αρχείου εξόδου του Βελτιστοποιημένου «aligner algorithm» της NGS data analyses platform

Παράδειγμα αρχείου εξόδου (minimum overlap σε bits = 16 - (8 SNPs)) :

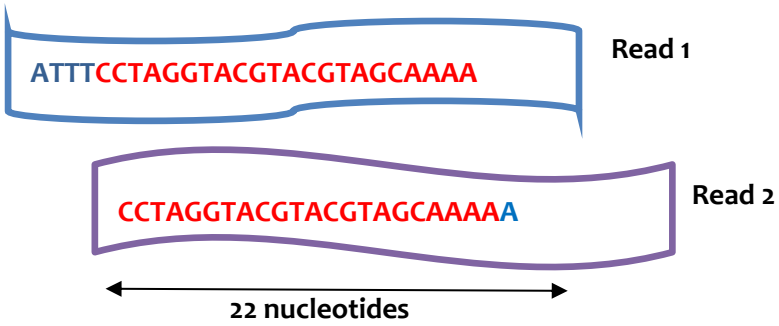
overlap neucleotypes	GeneFile1.txt	GeneFile2.txt
22	4	0
20	0	8
8	22	20

Πίνακας 3.4 Παράδειγμα αρχείου εξόδου του Βελτιστοποιημένου «aligner algorithm» της NGS data analyses platform

Βάσει του πίνακα 3.3, που φαίνεται πιο πάνω, λαμβάνουμε τη γνώση ότι τα 22 πρώτα νουκλεοτίδια του read με μοναδικό αναγνωριστικό/ id = 4, του αρχείου εισόδου με όνομα «GeneFile1.txt» (το $5^{ο}$ στη σειρά), συμπίπτουν/ είναι τα ίδια με τα τελευταία 22 νουκλεοτίδια του read με μοναδικό αναγνωριστικό/ id = 0 (το $1^{ο}$ στη σειρά), του αρχείου εισόδου με όνομα «GeneFile2.txt». Αντίστοιχα για τις υπόλοιπες γραμμές του αρχείου.

Γραφικά η πιο πάνω πληροφορία θα ήταν η ακόλουθη:

Για minimum overlap σε SNPs = 16 = (8 nucleotides):



Εικόνα 3.8 Παράδειγμα επικάλυψης μεταξύ 2 ακολουθιών – για minimum nucleotides overlap = 8 νουκλεοτίδια.

Αντίστοιχα για τις επόμενες γραμμές του αρχείου εξόδου. Το αρχείο εξόδου θα περιλαμβάνει μόνο ακολουθίες που επικαλύπτονται μεταξύ τους κατά 8 νουκλεοτίδια [21] (=16 SNPs – 1 νουκλεοτίδιο αποτελείται από 2 νουκλεοτίδιο) **τουλάχιστον.**

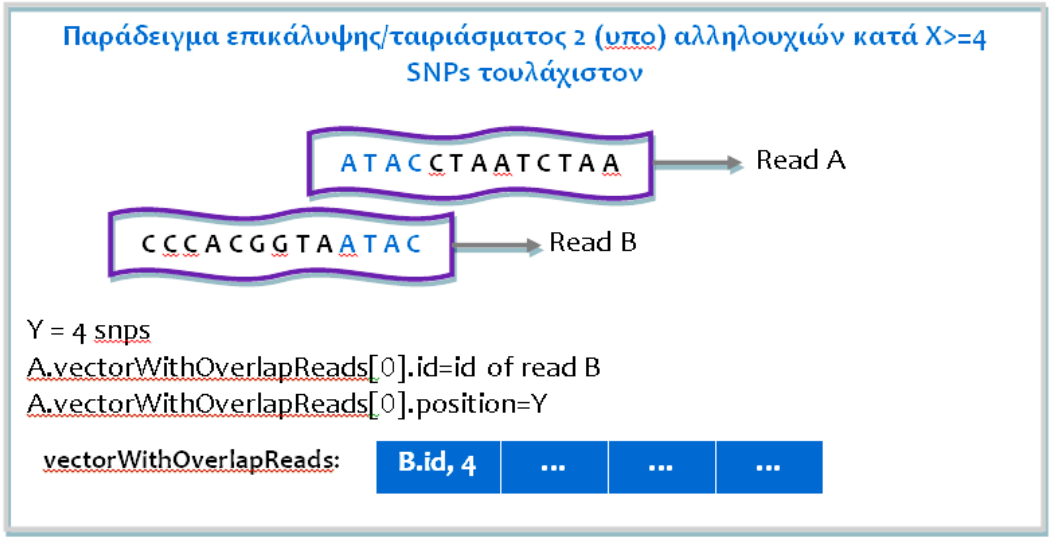
3.2.4.6 Αποθήκευση πληροφορίας χαρτογραφημένων αλληλουχιών

Στο υποκεφάλαιο 3.2.2.1 εξηγήθηκε αναλυτικά οι δομές που περιέχονται στη βιβλιοθήκη read.h. Μία από αυτές είναι ο μονοδιάστατος πίνακας (vector) που χρησιμοποιείται για την αποθήκευση πληροφοριών που αφορούν την επικάλυψη δύο υπο-αλληλουχιών, για τη χαρτογράφηση ολόκληρης της γονιδιακής ακολουθίας. Πολλές μελέτες επισημαίνουν, όπως ήδη έχει λεχθεί κατά τη διπλωματική αυτή, ότι μερικά από τα μείζων ζητήματα της ευθυγράμμισης κατά τη γονιδιακή χαρτογράφηση είναι ότι οι πλύστες μεθοδολογίες χαρτογράφησης δύναται να εφαρμόζονται μόνο σε μικρές γονιδιακές υπο-ακολουθίες σε μήκος και ότι απαιτούν μεγάλη χωρητικότητα μνήμης για την απαιτούμενη αναφορά από ένα τμήμα του DNA ένα άλλο (από μία ακολουθία σε μία άλλη με την οποία επικαλύπτεται). Μία από τις μελέτες που το επιβεβαιώνουν είναι αυτή των Heng Li, Jue Ruan and Richard Durbin (The Wellcome Trust Sanger Institute, Hinxton CB10 1SA, United Kingdom; Beijing Genomics Institute, Chinese Academy of Science, Beijing 100029, China) [19].

Η διαδικασία βελτιστοποίησης που ακολουθήθηκε στο τρόπο που αποθηκεύουμε αυτές τις πληροφορίες, στα πλαίσια της διπλωματικής μου, ήταν η εξής:

Για κάθε ξεχωριστή γενετική υπο-ακολουθία, έστω Α, που θα αποθηκευτεί στη μνήμη, δημιουργείται ένα αντικείμενο τύπου read (της κλάσης read.h), με ένα bitset που αναπαριστά τη κωδικοποιημένη σειρά νουκλεοτιδίων A,T,C,G σε δυαδική μορφή (όπως εξηγήθηκε στο υποκεφάλαιο 3.2.2.1), ένα bitset με κωδικοποιημένο το επίπεδο ποιότητας του κάθε αντίστοιχου SNP της ακολουθίας αυτής και ένα vector για την αποθήκευση των πληροφοριών των reads, με τα οποία το συγκεκριμένο read A επικαλύπτεται / επιτυγχάνει matching κατά Χ νουκλεοτίδια τουλάχιστον (για Χ ακέραιος καθοριζόμενος από το χρήστη). Κάθε φορά που εντοπίζεται μία επικάλυψη/ ταίριασμα με ένα άλλο read, έστω Β, όπου η αρχή του read A είναι ταιριάζει απόλυτα με το τέλος της υπο-ακολουθίας Α κατά Χ τουλάχιστον SNPs (έστω κατά Υ νουκλεοτίδια) και το read Β είναι το πρώτο με το οποίο εντοπίζουμε κάποιο ταίριασμα σε σχέση με το read Α, τότε στη πρώτη θέση του vector του αντικειμένου Α θα αποθηκευτεί το id του read Β και το Υ. Δηλαδή, στο vector του read Α αποθηκεύονται μόνο οι πληροφορίες για τα reads με τα οποία η αρχή της υπο-ακολουθίας Α (από τα αριστερά προς τα δεξιά) ταιριάζει απόλυτα με το τέλος άλλων reads, κατά τουλάχιστον Χ νουκλεοτίδια.

Το πιο πάνω παράδειγμα σκιαγραφείται πιο κάτω, για κατανόηση:



Εικόνα 3.9 Παράδειγμα τρόπου αποθήκευσης πληροφοριών σχετικά με το ταίριασμα 2 υπο (αλληλουχιών) στη μνήμη

Κεφάλαιο 4

Μετρήσεις και Αποτελέσματα

4.1 Σχέδιο Επικύρωσης NGS data analyses platform

Για την επικύρωση της NGS data analyses platform που αναπτύχθηκε στα πλαίσια της διπλωματικής αυτής, παρά το γεγονός ότι **ο στόχος της δεν ήταν η υλοποίηση των συγκεκριμένων αλγορίθμων που την απαρτίζουν καθ' εαυτό, αλλά η συνεργασία τους για την υλοποίησης μίας μη στατικής, καινοτόμας πλατφόρμας**

(δυναμική ως προς τις αλλαγές) που θα εκτελείται παράλληλα, αρχικά αξιολογήθηκε ο κάθε αλγόριθμος ξεχωριστά. Ακολούθως, με βάση τις παραστάσεις που ακολουθούν σχολιάστηκε κατά πόσον υπάρχει η ανάγκη για παραλληλοποίηση του κάθε αλγορίθμου.

4.1.1 Μετρικές Αποτελεσμάτων

- **Επεκτασιμότητα (Scalability) αλγορίθμου:** Είναι η δυνατότητα του λογισμικού (software) να διαχειριστεί μεγαλύτερα προβλήματα και άρα μεγαλύτερο όγκο δεδομένων με σχετικά μικρή αύξηση στο κόστος του αλγόριθμου. Όταν περισσότεροι επεξεργαστές μπορούν να χρησιμοποιηθούν χωρίς να επηρεάζεται αισθητά η αποδοτικότητα του αλγορίθμου, λέμε ότι ο αλγόριθμος είναι επεκτάσιμος.

- **Χρόνος Εκτέλεσης (Run Time)/ Χρονική Πολυπλοκότητα (Time Complexity):** Είναι ανάλογος του όγκου δεδομένων εισόδου ή αλλιώς του μεγέθους του προβλήματος που επιλύει. Στην ανάλυση μετρικών λαμβάνεται ο μέσος χρόνος εκτέλεσης πολλών στιγμιότυπων για αντικειμενικότερες μετρικές.

    - **Πολυπλοκότητα (Complexity):** Ο ρυθμός με τον οποίο μπορεί να αυξηθεί μία γραφική πολυπλοκότητας (πχ. χρονική) συναρτήσει του μεγέθους της εισόδου n, είναι:
        - Γραμμική Πολυπλοκότητα: $T(n) = c\ n$
        - Τετραγωνική Πολυπλοκότητα: $T(n) = c\ n^2$
        - Εκθετική Πολυπλοκότητα: $T(n) = c\ n^k$

- **Cost:** Το κόστος που χρειάστηκε για την εκτέλεση μιας συγκεκριμένης εκτέλεσης.
  Ο τύπος για υπολογισμό του Cost είναι: **Cost = wall time * #cpus**

Η πληροφορία για το walltime υπάρχει στο τέλος του report file που παράγεται μετά την εκτέλεση ενός script στο Cy-Tera server. Γενικά, το walltime πρέπει να μειώνεται όσο αυξάνονται οι επεξεργαστές, όπως παρατηρείται και σε κάποιες δοκιμές που έγιναν σε άλλους servers [8].

- **Speedup:** Μετρά το όφελος που υπάρχει χρησιμοποιώντας ένα παράλληλο αλγόριθμο για την εκτέλεση ενός προγράμματος σε σύγκριση με το καλύτερο σειριακό αλγόριθμο.
  Ο τύπος για υπολογισμό του Speedup είναι: **Speedup = Ts / Tp**

  Ts: ο χρόνος εκτέλεσης (walltime) που απαιτείται με χρήση ενός σειριακού αλγορίθμου.
  Tp: ο χρόνος εκτέλεσης (walltime) που απαιτείται με χρήση ενός παράλληλου αλγορίθμου.

  Γενικά, το speedup πρέπει να αυξάνεται όσο αυξάνονται οι επεξεργαστές, όπως παρατηρείται και σε κάποιες δοκιμές που έγιναν σε άλλους servers [8].

  Το speedup μπορεί να περιοριστεί λόγω:
  - του κόστους επικοινωνίας
  - της εξισορρόπησης φορτίου
  - του κόστους παραγωγής
  - του χρονοπρογραμματισμού των διεργασιών και των I/O λειτουργιών

- **Efficiency:** Η μετρική αυτή συνδέει το speedup με τον αριθμό των επεξεργαστών που χρησιμοποιούνται. Εκφράζει το ποσό αξιοποίησης των cpus που δεσμεύονται.

Ο τύπος για υπολογισμό του Efficiency είναι: **Efficiency = Speedup / #processors**

#processors = αριθμός των κόμβων * το πλήθος των cpus ανά κόμβο

Το efficiency πρέπει:

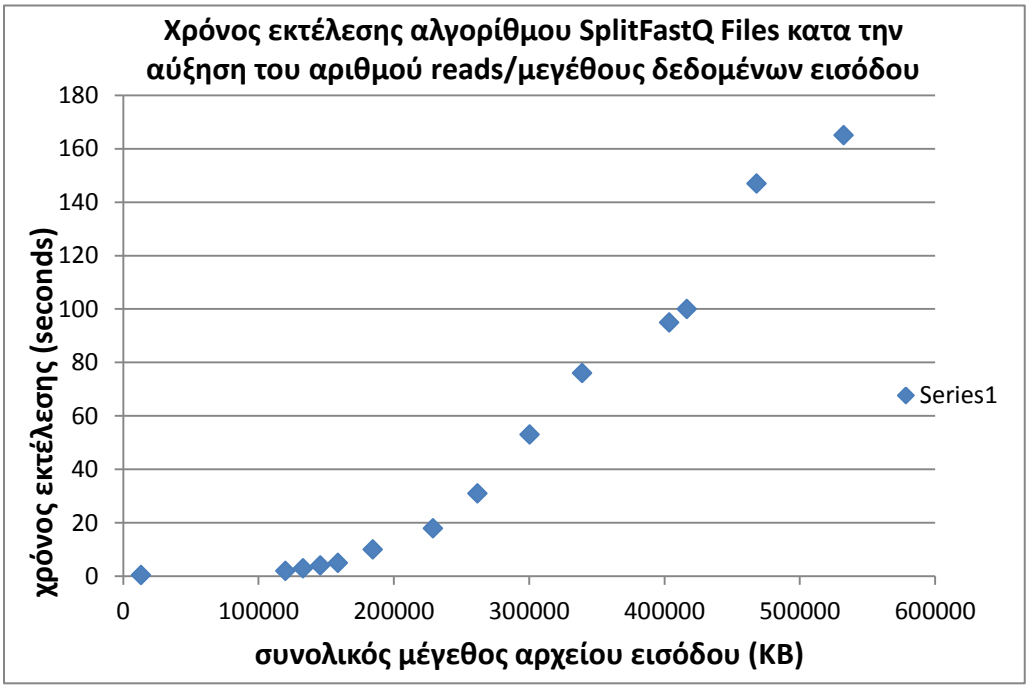
- Ιδανικά να έχει τιμή 1, αλλά πρακτικά να είναι μεταξύ των τιμών 0 και 1
- Να **μειώνεται *όσο αυξάνεται ο αριθμός των επεξεργαστών*.**

4.2 Επικύρωση Αλγορίθμου 1–Διάσπασης Αρχείων

Για την επικύρωση και την επαλήθευση του αλγορίθμου «SplitFastQ» δεν χρειάστηκαν περίπλοκες και χρονοβόρες μετρικές λόγο της απλότητας της λειτουργίας του. Όπως αναλύθηκε πιο πάνω ο αλγόριθμος απλά λαμβάνει ένα αρκετά μεγάλο αρχείο εισόδου σε πλήθος γενετικών (υπο)ακολουθιών/ reads και το διασπά σε πολλά μικρότερα αρχεία εξόδου, ώστε να επεξεργαστούν παράλληλα αν αυτό είναι αναγκαίο, αντιγράφοντας το αρχείο εισόδου πιστά, γραμμή προς γραμμή. Τα αρχεία εξόδου αποτελούνται από συγκεκριμένο πλήθος γραμμών του αρχείου εισόδου και κάθε κ-οστό αρχείο εξόδου αποτελείται από 1000 περισσότερες γενετικές ακολουθίες από ότι το κ-1 –οστό αρχείο εξόδου (η σταθερά 1000 μεταβάλλεται από το κώδικα όπως επιθυμεί ο χρήστης), ενώ το πρώτο αρχείο αποτελείται από X reads, όπου X ακέραια σταθερά που δίνεται από τον χρήστη σαν είσοδος στο πρόγραμμα.

Ο αλγόριθμος εφαρμόστηκε, αρχικά, σε πραγματικά δεδομένα ασθενών, μεγέθους 9.725 MB και $2.50875*10^{10}$ ακολουθιών, με χρόνο εκτέλεσης 8 λεπτά (8*60=480 sec). Το ποσό αλληλουχιών αύξησης ανά αρχείο, κατά την επαλήθευση, ήταν 1000 και το πλήθος αρχείων εξόδου ήταν 238, με κλήση του αλγορίθμου **«./SplitFastQ PLASMA_S1_L001_R2_001.fastq 1000».** Το πρώτο αρχείο εξόδου διέθετε **1000** reads, το $2^{ο}$ διέθετε 2000 reads, το $3^{ο}$ διέθετε 3000 reads, . . ., το $238^{ο}$ αρχείο διέθετε 239000 reads. Ακολούθως, εφαρμόστηκε πολλές φορές, σε συνολικά 1274000 ακολουθίες. Για τη δημιουργία της πιο κάτω γραφικής παράστασης εφαρμόστηκε ο αλγόριθμος «SplitFastQ Files>» σε αρχεία διαφόρων

μεγεθών, όπως παρουσιάζει η πιο κάτω γραφική παράσταση 4.1 και στο πίνακα 4.1.



Σχήμα 4.1 Απεικόνιση αποδοτικότητας αλγορίθμου Διάσπασης Αρχείων ανάλογα με το όγκο δεδομένων εισόδου σε σχέση με το χρόνο εκτέλεσης

Μέγεθος αρχείου εισόδου(KB)	Χρόνος εκτέλεσης (seconds)
531620	165
468084	145
416516	100
403624	95
339164	76
300488	53
261812	31
228942	18
184460	10
158676	5
145784	4
132892	3
120000	2

Πίνακας 4.1 Ακριβής μετρήσεις χρόνου εκτέλεσης αλγορίθμου «Διάσπασης Αρχείων FastQ» με την αύξηση του πλήθους των reads

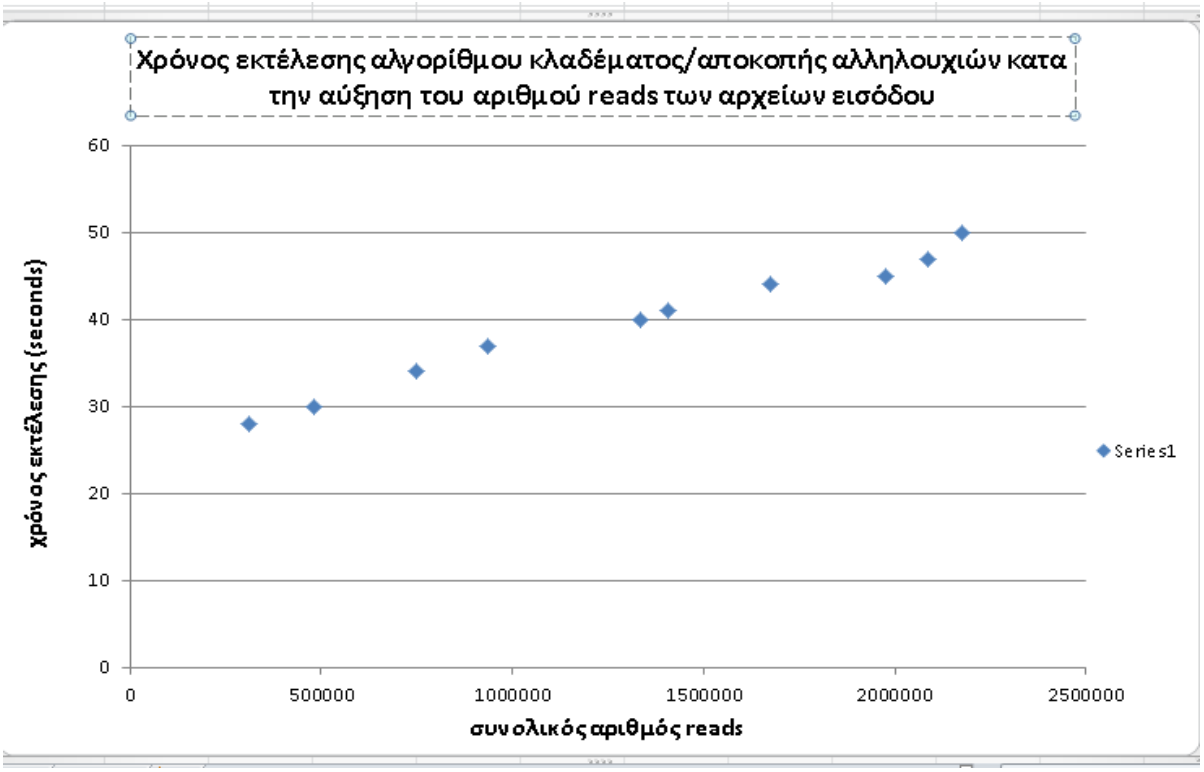
Η γραφική παράσταση του σχήματος 4.1 παρουσιάζει το χρόνο εκτέλεσης κατά την αύξηση του πλήθους των reads του αρχείου εισόδου στον αλγόριθμο «Split FatsQ File». Ο άξονας Χ δείχνει το πλήθος των reads του αρχείου που δόθηκε σαν είσοδο και ο άξονας ο Ψ τον απαιτούμενο χρόνο εκτέλεσης για κάθε συγκεκριμένη είσοδο.

```
@M00748:3:000000000-A5UP3:1:1101:9083:3511 1:N:0:1
GACTGTTTTCCTGGTCCTTCTTCTCTTCCTCCTTGCACCAGCATCACAGCAGAAGGCACAGGGGAGGAGAAGGGGCTGAGA
GAGAGCTCACCCCATCTCCCCACACTAATTCCATTCCAGGATCTGGGAGGAGGAGCCAGGAAGGGGAAGG
+
CCCCCGGGGGGGGGGGGGGGGGDGGGGGGGGGGGGGFFGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGG
EGGGGGGGGGGGGFFGGGGGGGGGGGGGGGGGGGGFGFGGGGFGGGGGGGGFGGGGEEFFGGGGDEFEGG+BFG
@M00748:3:000000000-A5UP3:1:1101:14405:3511 1:N:0:1
TGTGGGGTTTTCTTCACCGGCACAGGTGCAGACGCACTGTCAGCATGGAGACTGGATGCCAAAATGTCCAACTCTGTTCCA
CCATCCTCAATGTGGCCTTTCCACCTGAAGTCATTGGACCACTTTTCTTCTTTCCCCTCCTCTACATGAT
+
CCCCCGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGG
GGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGFGGGGGGGGGGGGGGGGGGGGCG
@M00748:3:000000000-A5UP3:1:1101:16307:3512 1:N:0:1
TCCTTTGATAGGTCTCAGTTCTGGTCCTGAAGGTGAATGAGGTGTCTTTTGGGATTGAGGTACGTGGTGAGGACCTGACTGT
GGCCCTGCAAGCAGAGGAACTGACCCTCCAGCAGCTGGGCACCGTGGGACCCTGGAGATCGGAAGAGCA
+
CCCC<EC<AEEFFC<EFFGFGFGGE@FFF,C<<E@F@EEEFF<FGFEFFGFD@EDCFGGGCFFGGEFGGDGEDDCEEGCCGFG
FGDCCFFGFGCFGGGGGGGCCFG<C6C,CFGGFGGG,,?FCDGEGG7==BCEFGFF?@@<BFGGG?=F
...
```

Εικόνα 4.1 Δείγμα του αρχείου εισόδου επικύρωσης του αλγορίθμου με reads μήκους 151 νουκλεοτιδίων

4.3 Επικύρωση Αλγορίθμου2– Κλάδεμα Γενετικών Ακολουθιών

Για την επικύρωση του αλγορίθμου 2, «Αλγόριθμος Trimming», κατά τον οποίο αποκόπτονται οι βάσεις με low quality score από τα δύο άκρα των αλληλουχιών, δημιουργήθηκαν script files για την εκτέλεση του με διαφορετικές εισόδους, σε μέγεθος/πλήθος από reads και περιεχόμενο. Τα script files εκτελέστηκαν για τη δημιουργία της γραφικής παράστασης που ακολουθεί στη συνέχεια, με παραμέτρους το χρόνο και το πλήθος αλληλουχιών εισόδου. Ο αλγόριθμος αξιολογήθηκε ως προς το χρόνο εκτέλεσής του σε σχέση με το μέγεθος αρχείου εισόδου και ως προς την ορθότητά του, δηλαδή κατά πόσον τα αποτελέσματα – περιεχόμενο αρχείων εξόδου, ήταν τα αναμενόμενα και πόσο χρόνο (run time) χρειάστηκε για την εξαγωγή των αποτελεσμάτων αυτών.



Σχήμα 4.2 Γραφική Παράσταση του χρόνου εκτέλεσης κατά την αύξηση του πλήθους των reads, για τον αλγόριθμο «trimming».

Πλήθος reads εισόδου	Χρόνος εκτέλεσης (seconds)
2177000	50
2086000	47
1977000	45
1677000	44
1407000	41
1337000	40
937000	37
750000	34
480000	30
310000	28

Πίνακας 4.2 Ακριβής μετρήσεις χρόνου εκτέλεσης αλγορίθμου «Κλαδέματος Γενετικών Ακολουθιών» με την αύξηση του πλήθους των reads

Η γραφική παράσταση του σχήματος 4.2 παρουσιάζει το χρόνο εκτέλεσης κατά την αύξηση του πλήθους των reads του αρχείου εισόδου στον αλγόριθμο «Trimming». Ο άξονας Χ δείχνει το πλήθος των reads του αρχείου που δόθηκε σαν είσοδο και ο άξονας ο Ψ τον απαιτούμενο χρόνο εκτέλεσης για κάθε συγκεκριμένη είσοδο.

```
AGAATTTACCCAGTCTGAACTGAAGAGA
II,IIIII?II?IIII5,,,,?!,?,,5

TTTTATTGTAAAATTATAAAATTATTAAT
IIII5III5??5,II5I5,5,?5,5,,,5

ATACATTTCAGCTTC
?III5II??55,?55
```

Εικόνα 4.2 Δείγμα αρχείου εξόδου του αλγορίθμου αποκοπής βάσεων από τα άκρα των γενετικών υπο-ακολουθιών, με low quality score (για 3 υπο-αλληλουχίες).

4.4 Επικύρωση Αλγορίθμου 3–Αλγόριθμος Παραλληλοποίησης

Για τη διαδικασία επικύρωσης του αλγορίθμου παραλληλοποίησης, κατά τον οποίο δημιουργούνται script αρχεία με 12 εντολές εκτέλεσης του αλγορίθμου 4 –

«Aligner Algorithm» (αφού σε ένα κόμβο δύναται να εκτελεστούν το πολλή 12 διεργασίες), εφαρμόστηκε ο αλγόριθμος σε αρχεία τα οποία αποτελούσαν την έξοδο του αλγορίθμου – «Διάσπασης FastQ Αρχείων». Δηλαδή ο αλγόριθμος παραλληλοποίησης εκτελέστηκε, για την επικύρωση αυτή, σε αρχεία με υπο-ακολουθίες DNA οι ο ποίες έχουν **ίδιο μήκος/πλήθος από SNPs**. Η είσοδος της εκτέλεσης αυτής ήταν αρχεία τα οποία προέκυψαν μετά την εφαρμογή του αλγορίθμου «Διάσπαση FASTQ αρχείων» με είσοδο το αρχείο **«PLASMA_S1_L001_R2_001.fastq»** και **«SRR003000_1.filt.fastq»**, μεγέθους 9.49673 GBytes και 0.75773 GBytes, αντίστοιχα και μήκος reads 36 και 151 βάσεις, αντίστοιχα. Όσον αφορά το τελευταίο έχει ληφθεί από το 1000 Genomes Project ανθρώπινες paired-end ακολουθίες από το φάκελο **NA19239** (http://ftp.1000genomes.ebi.ac.uk/vol1/ftp/data/NA19239/sequence_read/).
Δηλαδή, αφού τα αρχεία αυτά διασπάστηκαν μέσω του αλγορίθμου «Split FastQ files» σε πολλαπλά. Τα διαμελισμένα πλέον αρχεία (αυτά που προέκυψαν μετά την εκτέλεση του αλγορίθμου «Split FastQ files) εισάχθηκαν σαν είσοδο στον αλγόριθμο αυτό, της «Παραλληλοποίησης**».** Με αυτό τον τρόπο δημιουργήθηκαν script αρχεία. Εκτελούνται αυτόματα από τον αλγόριθμο παραλληλοποίησης, και καλούν τον επόμενο αλγόριθμο «Aligner Algorithm», για **κάθε δυνατό ζεύγος αρχείων εισόδου.** Δηλαδή κάθε αρχείο που εισάχθηκε σαν είσοδο στον «Αλγόριθμο Παραλληλοποίησης» έγινε alignment με κάθε άλλο αρχείο που εισάχθηκε, αλλά και με τον εαυτό του.

Άρα, ο αλγόριθμος RunSubFiles δημιούργησε πολλά script files καλώντας τον aligner algorithm για κάθε δυνατό ζεύγος αρχείων, ώστε να συγκριθούν οι ακολουθίες του κάθε αρχείου με κάθε άλλου αρχείου αλλά και με του ίδιου του αρχείου (του εαυτού του), όπως εξηγείται στο υποκεφάλαιο 3.2.3. Σύμφωνα με τη περιγραφή του υποκεφαλαίου 3.2.3, για είσοδο N αρχείων συνολικά στον αλγόριθμο «Parallelization», θα καλεστεί ο αλγόριθμος (μέσω των script files που θα δημιουργηθούν) $\binom{N}{2}$+N φορές, αφού τόσο είναι το πλήθος των ζευγών που υπάρχουν.

Σε κάθε script file καθορίστηκαν 12 κλήσεις του αλγόριθμου ευθυγράμμισης, αναλογιζόμενοι ότι κάθε επεξεργαστής αποτελείται από 12 κόμβους. Άρα για να είναι βέβαιο για τον αριθμό εκτελέσεων, στο σύνολο, χρησιμοποίησα με κάθε script file όλους τους δυνατούς κόμβους. Δηλαδή, ο μέγιστος δυνατός αριθμός διεργασιών ανά κόμβο είναι 12 και για αυτό αξιοποιήθηκε ο αριθμός αυτός.

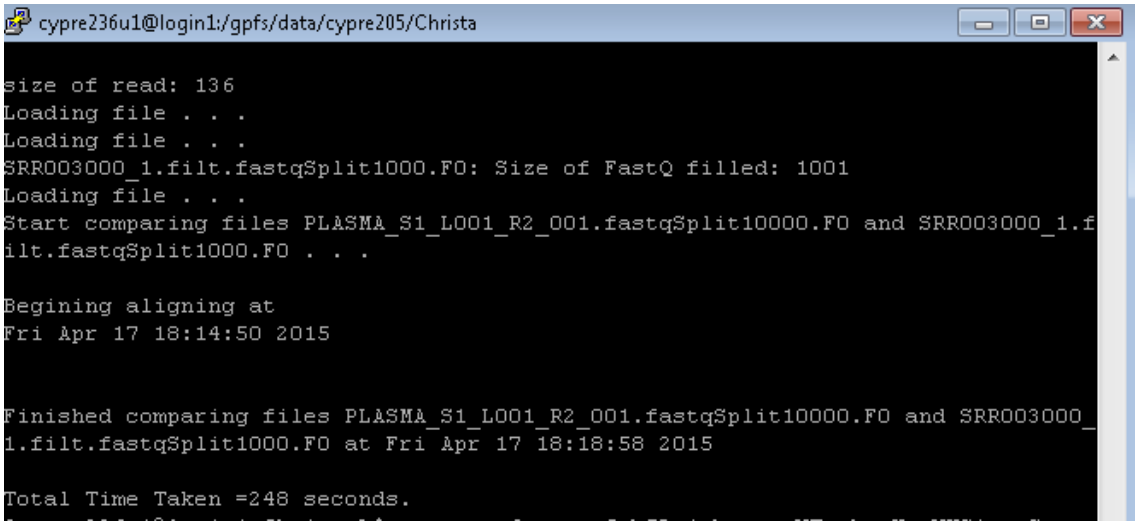
4.5 Επικύρωση Αλγορίθμου 4-Ευθηγράμμιση Αλληλουχιών

Για έλεγχο του αλγόριθμου ευθυγράμμισης γενετικών ακολουθιών, της πλατφόρμας, ως προς τη σχέση run time – number of input reads , εκτελέστηκε ο αλγόριθμος ευθυγράμμισης αρκετές φορές με διαφορετικές εισόδους κάθε φορά, όπως φαίνεται στο σχήμα και πίνακα 4.3. Συγκεκριμένα, μετά την εφαρμογή του αλγορίθμου διάσπασης αρχείων, «SplitFastQ», δημιουργήθηκαν αρχεία μεγέθους 1000, 2000, 3000, 4000, … , 22000 reads, τα οποία ακολούθως αποτέλεσαν είσοδο στον αλγόριθμο «RunSubFiles» (βλέπε υποκεφάλαιο 3.2.4), ώστε να ελεγχθεί κατά πόσον δουλεύει ορθά ο αλγόριθμος ευθυγράμμισης, ο οποίος καλείται αυτόματα από τον προηγούμενο, μέσω των script files που εξαγάγει σαν έξοδο.

Ο αλγόριθμος RunSubFiles δημιούργησε πολλά script files καλώντας τον aligner algorithm για κάθε δυνατό ζεύγος αρχείων, ώστε να συγκριθούν οι ακολουθίες του κάθε αρχείου με κάθε άλλου αρχείου αλλά και με του ίδιου του αρχείου (του εαυτού του), όπως εξηγείται στο υποκεφάλαιο 3.2.3. Σε κάθε script file καθορίστηκαν 12 κλήσεις του αλγόριθμου ευθυγράμμισης, όπως ήδη επισημάνθηκε στο προηγούμενο υποκεφάλαιο 4.3. Πιο κάτω φαίνεται η γραφική παράσταση, η οποία παρουσιάζει το Scalability/επεκτασιμότητα του αλγορίθμου αυτού, όπως αναλύθηκε στη παράγραφο αυτή. Αναλογιζόμενοι το τεράστιο πλήθος συγκρίσεων που μπορεί να χρειαστούν, για παράδειγμα σε μία ιατρική γενετική ανάλυση, ο αλγόριθμος αυτός **χρειάζεται παραλληλοποίηση, την οποία κάναμε**.

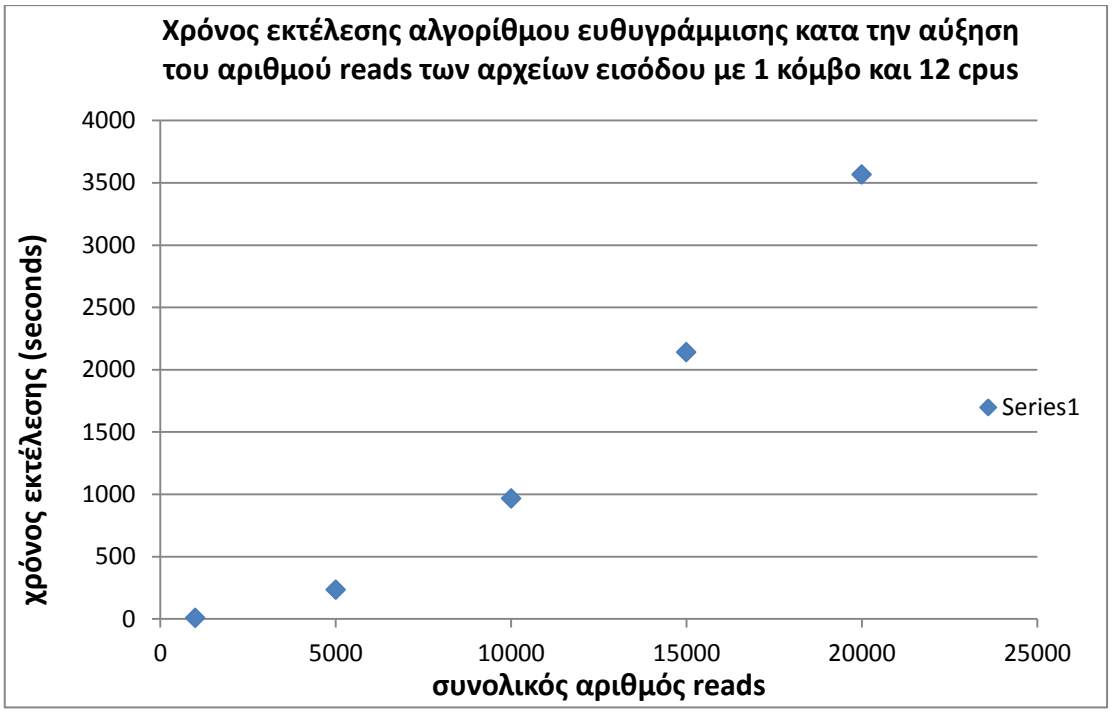
4.5.1Παραλληλοποίηση Αλγορίθμου ευθυγράμμισης

Για λήψη αντικειμενικών μετρήσεων ανεξάρτητων των πιο πάνω, ο αλγόριθμος ευθυγράμμισης εφαρμόστηκε σε δύο αρχεία, μεγέθους 10000 και 1000 reads, αντίστοιχα, με χρήση 1 μόνο επεξεργαστή και μίας μόνο διεργασίας. Άρα συνολικά εκτελέστηκαν 10000*1000=10000000 συγκρίσεις. Ο χρόνος εκτέλεσης που απαιτήθηκε μέχρι την ολοκλήρωση των συγκρίσεων ήταν 248 δευτερόλεπτα, όπως φαίνεται στη πιο κάτω εικόνα 4.3. Κάνοντας την αναλογία καταλήγουμε στη μετρική ότι σε 1 second γίνονται 40323 συγκρίσεις.



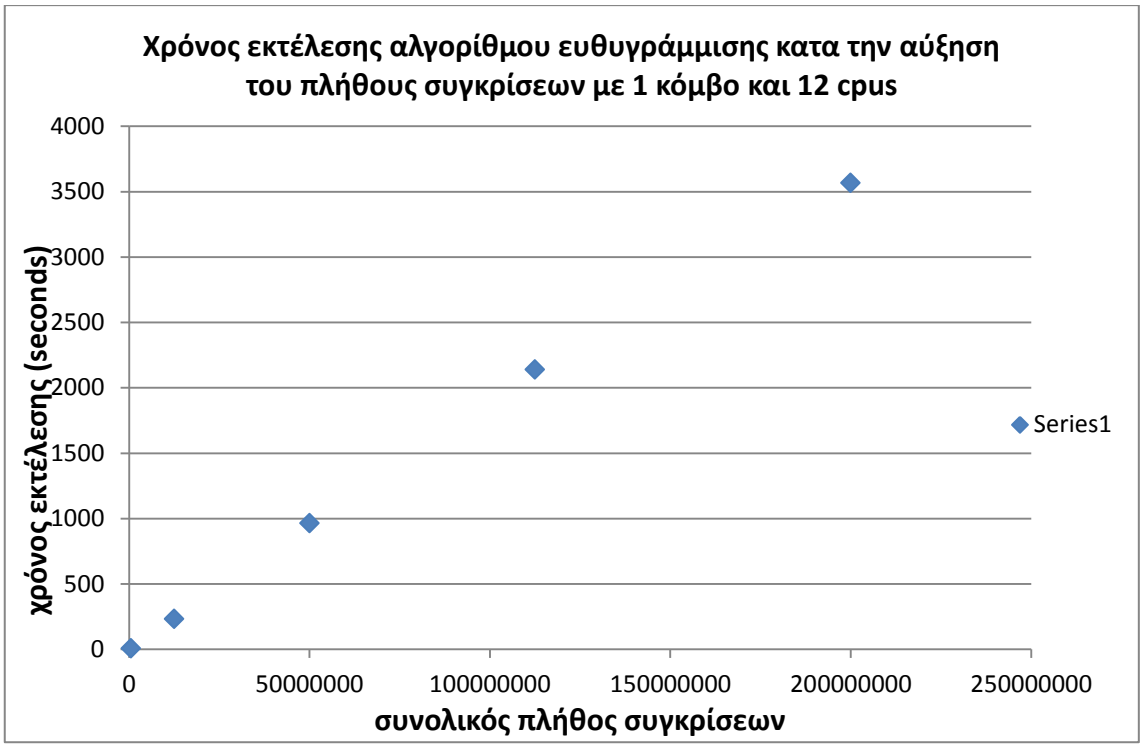
Εικόνα 4.3 Οθόνη αποτελέσματος αλγορίθμου «Aligner» για σύγκριση τμημάτων από τα αρχεία «PLASMA_S1_L001_R2_001.fastq» και SRR003000_1.filt», για συνολικά 10000000 συγκρίσεις.

Για τα διάφορα αρχεία του πίνακα 4.4, εκτελέστηκε ο αλγόριθμος ευθυγράμμισης με 1 κόμβο και 12 επεξεργαστές ανά κόμβο (12 διεργασίες/επεξεργαστές συνολικά) για τα αρχεία που προέκυψαν μετά τον αλγόριθμο διάσμασης αρχείων «Split FastQ Files» **,READS151_L001_R1_001.fastq** και **SRR003000_1.filt.fastq**. Ακολούθως εξάχθηκαν η πιο κάτω γραφικές παραστάσεις 4.4 και 4.5 και πίνακες 4.4 και 4.5, αντίστοιχα, οι οποίοι συσχετίζουν το χρόνο εκτέλεσης κατά την αύξηση του συνολικού πλήθους reads που δέχτηκε ο αλγόριθμος σαν είσοδο και κατ' επέκταση το πλήθος απαιτούμενων συγκρίσεων μεταξύ των reads κατά την εκτέλεση του αλγορίθμου.



Σχήμα 4.4 Γραφική Παράσταση του χρόνου εκτέλεσης κατά την αύξηση του πλήθους των reads στην είσοδο, για τον αλγόριθμο «Aligner», για reads μήκους 36 νουκλεοτιδίων.

Η γραφική παράσταση του σχήματος 4.4 παρουσιάζει το χρόνο εκτέλεσης κατά την αύξηση του πλήθους των reads του αρχείου εισόδου στον αλγόριθμο «Aligner», με χρήση 1 κόμβου και 12 επεξεργαστών ανά κόμβο. Ο άξονας Χ δείχνει το πλήθος των reads του αρχείου που δόθηκε σαν είσοδο και ο άξονας ο Ψ τον απαιτούμενο χρόνο εκτέλεσης για κάθε συγκεκριμένη είσοδο.



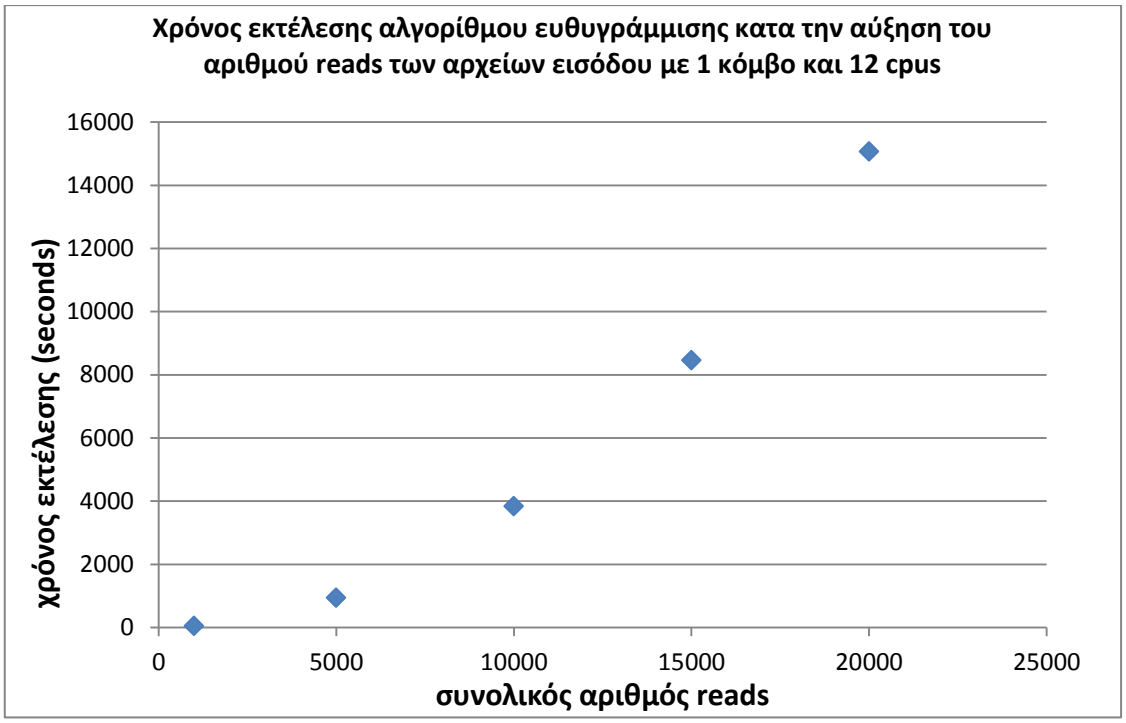
Σχήμα 4.5 Γραφική Παράσταση του χρόνου εκτέλεσης κατά την αύξηση του πλήθους συγκρίσεων, για τον αλγόριθμο «Aligner», με », για reads μήκους 36 νουκλεοτιδίων.

Η γραφική παράσταση του σχήματος 4.4 παρουσιάζει το χρόνο εκτέλεσης κατά την αύξηση του πλήθους των reads του αρχείου εισόδου στον αλγόριθμο «Aligner», με χρήση 1 κόμβου και 12 επεξεργαστών ανά κόμβο. Ο άξονας Χ δείχνει το πλήθος των reads του αρχείου που δόθηκε σαν είσοδο και ο άξονας ο Ψ τον απαιτούμενο χρόνο εκτέλεσης για κάθε συγκεκριμένη είσοδο.

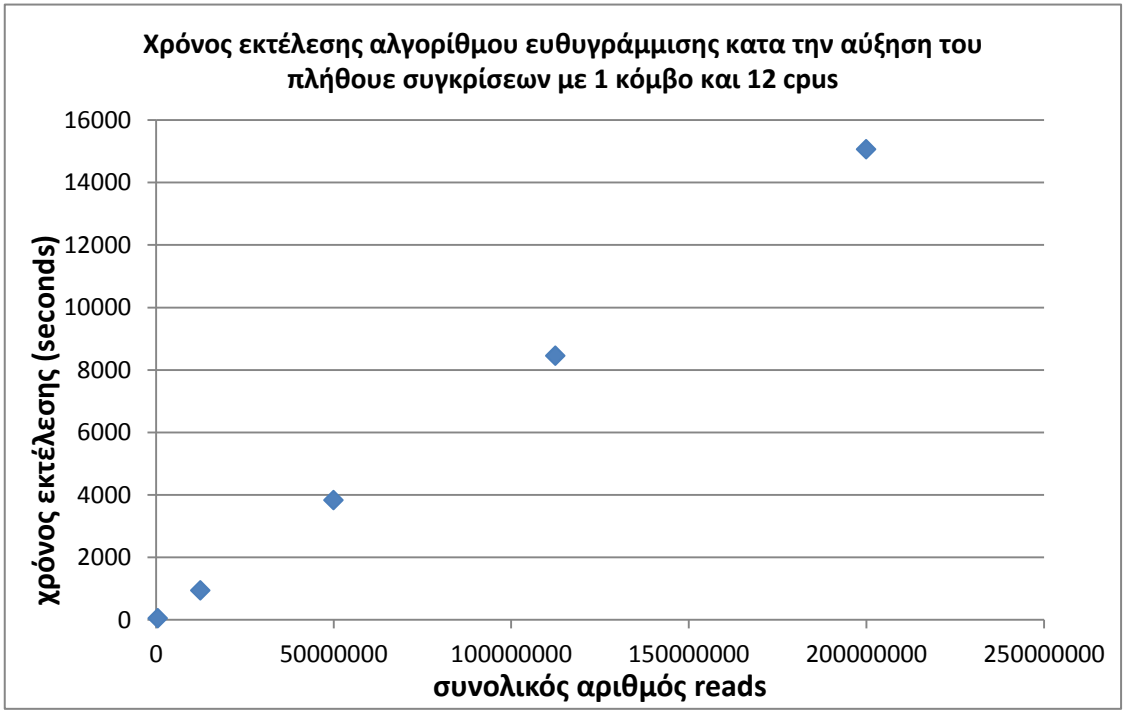
Με 12 διεργασίες συνολικά: ***(1 node , 12 tasks per node)***

Πλήθος reads 1ου αρχείου	Πλήθος reads 2ου αρχείου	Πλήθος συγκρίσεων μεταξύ των reads	Χρόνος εκτέλεσης (seconds)
1000	1000	(1000*(1000-1))/2 =499500	292
5000	5000	(5000*(5000-1))/2 =12497500	380
10000	10000	(10000*(10000-1))/2=49995000	602
15000	15000	(15000*(15000-1))/2 =1.12E+08	777
20000	20000	(20000*(20000-1))/2 =2E+08	880

Πίνακας 4.3 Ακριβής μετρήσεις χρόνου εκτέλεσης αλγορίθμου «Aligner» με την αύξηση του πλήθους των reads, για reads μήκους 36 νουκλεοτιδίων.



Σχήμα 4.6 Γραφική Παράσταση του χρόνου εκτέλεσης κατά την αύξηση του πλήθους συγκρίσεων, για τον αλγόριθμο «Aligner», για reads μήκους 151 νουκλεοτιδίων.

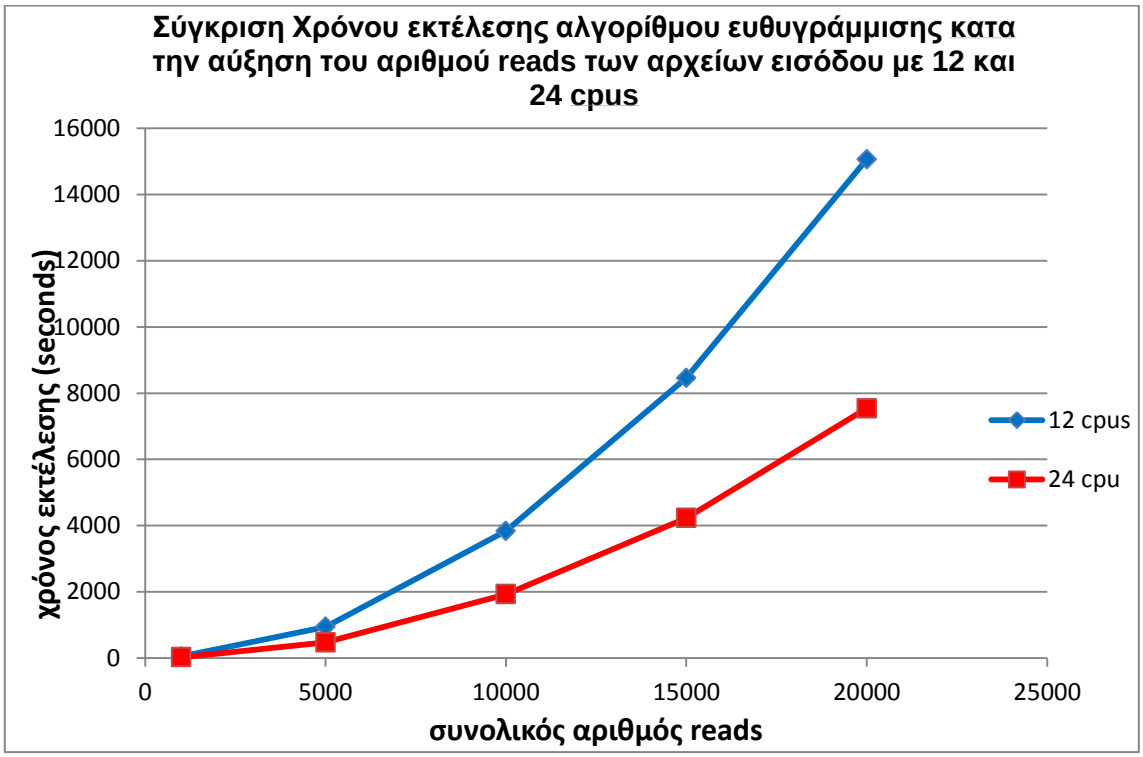


Σχήμα 4.7 Γραφική Παράσταση του χρόνου εκτέλεσης κατά την αύξηση του πλήθους συγκρίσεων, για τον αλγόριθμο «Aligner», με », για reads μήκους 151 νουκλεοτιδίων.

Με 12 διεργασίες συνολικά: ***(1 node , 12 tasks per node)***

Πλήθος reads 1ου αρχείου	Πλήθος reads 2ου αρχείου	Πλήθος συγκρίσεων μεταξύ των reads	Χρόνος εκτέλεσης (seconds)
1000	1000	(1000*(1000-1))/2 =499500	46
5000	5000	(5000*(5000-1))/2 =12497500	940
10000	10000	(10000*(10000-1))/2 =49995000	3835
15000	15000	(15000*(15000-1))/2 =1.12E+08	8455
20000	20000	(20000*(20000-1))/2 =2E+08	15060

Πίνακας 4.5 Ακριβής μετρήσεις χρόνου εκτέλεσης αλγορίθμου «Aligner» με την αύξηση του πλήθους των reads (και των συγκρίσεων), για reads μήκους 151 νουκλεοτιδίων.



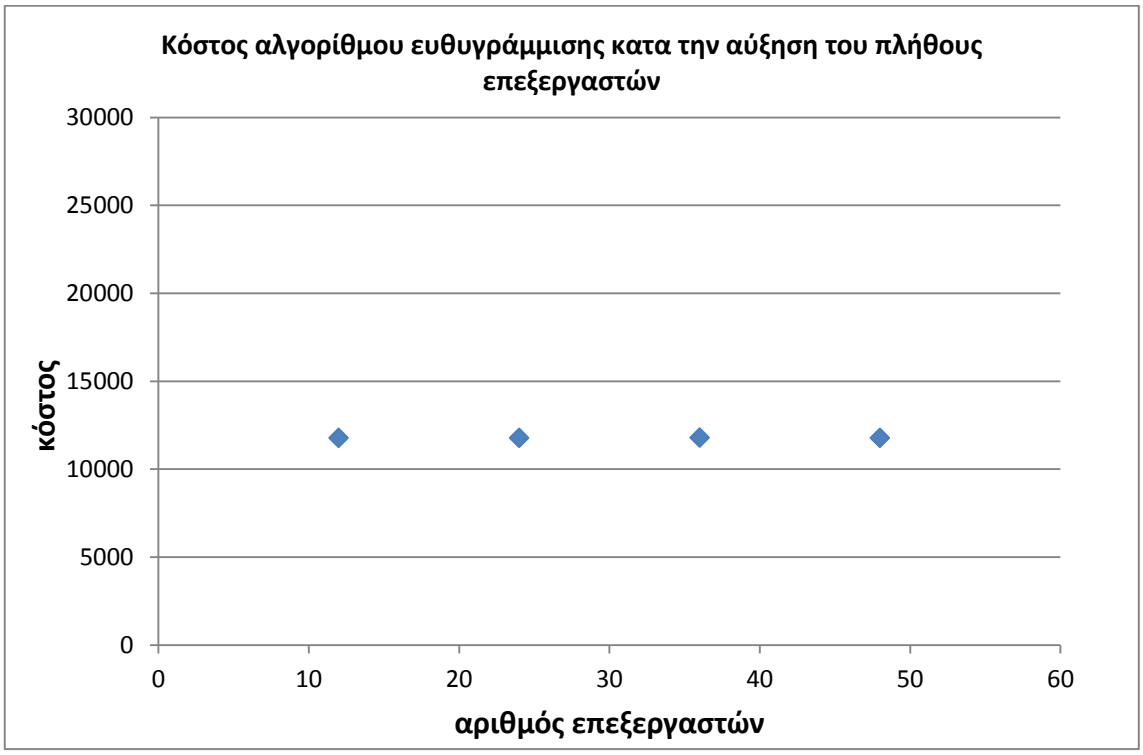
Σχήμα 4.8 Γραφική παράσταση σύγκρισης του χρόνου εκτέλεσης, του αλγορίθμου Ευθυγράμμισης, κατά την αύξηση του πλήθους των επεξεργαστών, με 12 επεξεργαστές και με 24 επεξεργαστες.

4.5.2 Επεκτασιμότητα

- Κόστος Αλγορίθμου:

Wall Time (seconds)	#cpus	Cost
980	12	11760
490	24	11760
327	36	11772
245	48	11760

Πίνακας 4.6 Μέτρηση του κόστους του αλγορίθμου «Aligner»



Σχήμα 4.9 Γραφική παράσταση κόστους κατά την αύξηση του πλήθους των επεξεργαστών, του αλγορίθμου «Aligner».

- Speedup Αλγορίθμου:

  Για τη **λήψη του χρόνου του καλύτερου σειριακού αλγόριθμου** έτρεξα τον αλγόριθμο με 12000 reads, μήκους 151 νουκλεοτιδίων το κάθε ένα, **με ένα κόμβο και 1 επεξεργαστή ανά κόμβο**. Ακολούθως, για τη **λήψη του παράλληλου χρόνου** έτρεξα τον αλγόριθμο με τα ίδια δεδομένα (ίδιο μέγεθος προβλήματος) αφού κομμάτιασα το αρχείο σε μικρότερα (με τν αλγόριθμο «Split FastQ File»), μεγέθους 1000 reads το κάθε ένα**, με 1 κόμβο και 12 επεξεργαστές ανά κόμβο**.

Αντίστοιχα, έπραξα την ίδια διαδικασία για δεδομένα εισόδου 60000 reads, 120000 reads, 180000 reads και 240000 reads.

Με βάσει τα αποτελέσματα, παρήγαγα τον ακόλουθο πίνακα 4.7.

Όγκος εισόδου (reads)	Σειριακός Χρόνος Εκτέλεσης(seconds)	Παράλληλος Χρόνος Εκτέλεσης(seconds) #cpus=12	Speedup
120000	550	46	11,96
60000	11280	940	12
120000	46015	3835	12
180000	101465	8455	11,99
240000	180700	15060	11,99
Average:			**~11,99**

Πίνακας 4.7 Speedup αλγορίθμου «Aligner» με 12 cpus, για διάφορους όγκους δεδομένων.

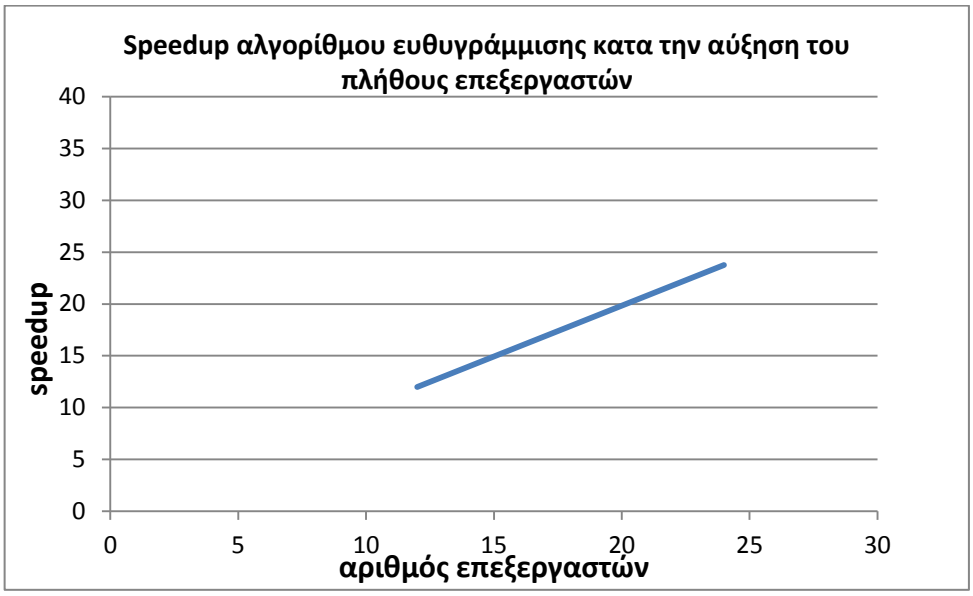
Την ίδια διαδικασία έπραξα κομματιάζοντας αυτή τη φορά το αρχείο εισόδου σε περισσότερα αρχεία, ώστε στη παράλληλη εκτέλεση να χρησιμοποιηθούν 24 επεξεργαστές συνολικά (2 κόμβοι με 12 επεξεργαστές ανά κόμβο).

Με βάσει τα αποτελέσματα, παρήγαγα τον ακόλουθο πίνακα 4.8, ο οποίος δείχνει το speedup για κάθε διαφορετική σε όγκο είσοδο.

Όγκος εισόδου (reads)	Σειριακός Χρόνος Εκτέλεσης(seconds)	Παράλληλος Χρόνος Εκτέλεσης(seconds) #cpus=24	Speedup
120000	550	24	22,92
60000	11280	470	24
120000	46015	1925	23,90
180000	101465	4230	23,99
240000	180700	7537	23,98
Average:			**~23,77**

Πίνακας 4.8 Speedup αλγορίθμου «Aligner» με 24 cpus, για διάφορους όγκους δεδομένων [35].

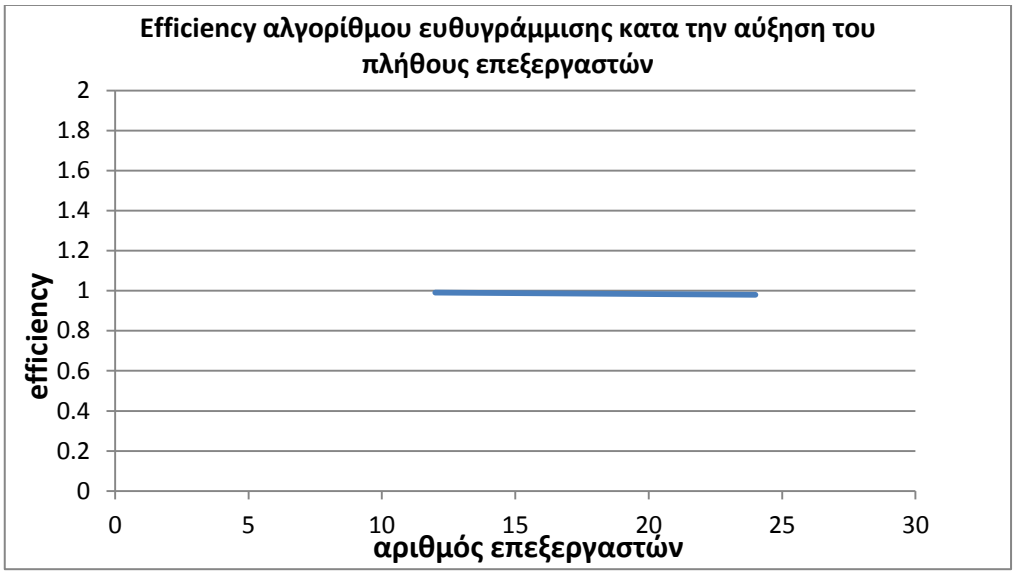
Στη συνέχεια εξήγαγα το πίνακα 4.7, όπου στον άξονα Χ παρουσιάζει το πλήθος τον cpus (δηλαδή το πλήθος των processors) και στον άξονα Ψ το αντίστοιχο speedup value.



Σχήμα 4.10 Γραφική Παράσταση speedup αλγορίθμου «Aligner», για διάφορα πλήθη cpus.

- Efficiency Αλγορίθμου:

  Για τη καταγραφή του Efficiency του speedup αλγορίθμου «Aligner» της NGS data analyses platform, με βάσει τους ορισμούς του υποκεφαλαίου 4.1.1, διαίρεσα το speedup με το αντίστοιχο πλήθος επεξεργαστών και ακολούθως παρήγαγα τον το σχήμα 4.8, όπου στον άξονα Χ παρουσιάζει το πλήθος τον cpus (δηλαδή το πλήθος των processors) και στον άξονα Ψ το αντίστοιχο Efficiency value.

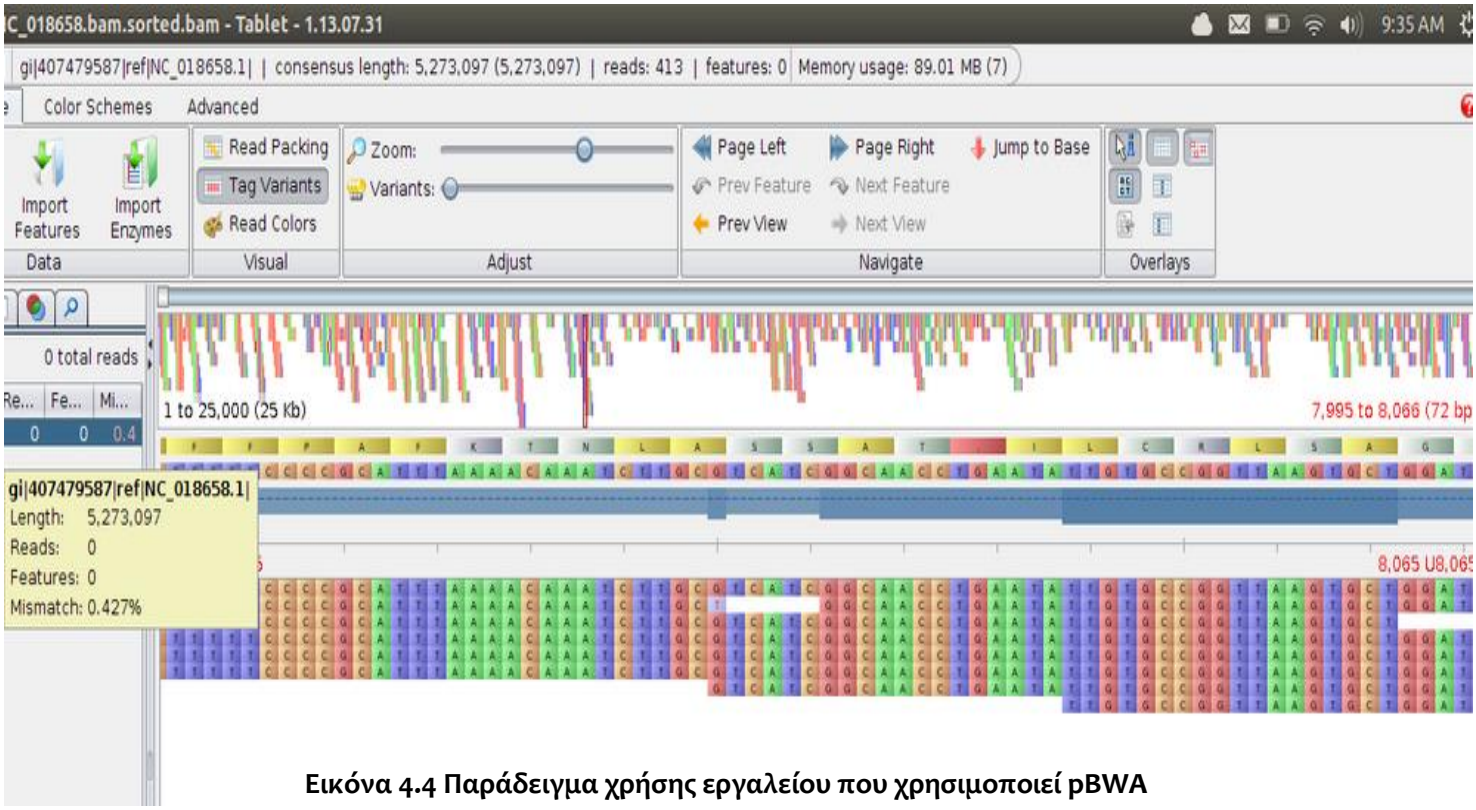


Σχήμα 4.11Γραφική παράσταση efficiency κατά την αύξηση του πλήθουε των cpus

4.5.3 Σύγκριση παραλληλοποίησης του Aligner αλγόριθμου της NGS data analyses platform με τον αλγόριθμο του pBWA

4.5.3.1 Αλγόριθμος BWA

Ο αλγόριθμος BWA (Burrows-Wheeler Alignment Tool) είναι ένα πακέτο λογισμικού για τη χαρτογράφηση (mapping) ακολουθιών χαμηλής απόκλισης προς ένα μεγάλο reference genome, όπως το ανθρώπινο γονιδίωμα [21]. Κατά το **paired-end mapping** [26], που υπάρχει σε κάποιους από τους αλγόριθμους, πρώτα βρίσκει τις θέσεις των καλών hits, τα ταξινομεί με βάση τις χρωμοσωμικές συντεταγμένες και στη συνέχεια κάνει μια γραμμική σάρωση διαμέσου όλων των πιθανών hits, για να συνδυάσει τα δυο άκρα [26].



Εικόνα 4.4 Παράδειγμα χρήσης εργαλείου που χρησιμοποιεί pBWA
Πηγή: http://2013-caltech-workshop.readthedocs.org/en/latest/bwa_mapping.html

Αποτελείται από κάποιους αλγορίθμους:

- BWA –backtrack
  - σχεδιάστηκε από τη Illumina [23]
  - έχει πάνω από 100 base pair (bp) reads
- BWA – SW
  - έχει πάνω από 70 bp reads έως και 1 Mbp
  - υποστηρίζει μεγάλη ανάγνωση και alignment διάσπασης
- BWA –MEM [5, 36]
  - είναι ο απόγονος του BWA-SW και ο πιο πρόσφατος αλγόριθμος που δημιουργήθηκε
  - έχει πάνω από 70 bp reads έως και 1 Mbp
  - υποστηρίζει μεγάλη ανάγνωση και alignment διάσπασης
  - δουλεύει με γονιδιώματα μεγέθους μεγαλύτερα των 4 GB
  - είναι πιο ανεκτικός στα μεγάλα κενά αλληλουχίας
  - έχει καλύτερη paired-end ανάγνωση, δηλαδή για κάθε DNA κομμάτι μπορεί να διαβάσει δεδομένα και από τα δύο άκρα. Οι ακολουθίες τότε αποθηκεύονται σε δύο ξεχωριστά αρχεία. Μπορεί να γίνει ακολουθία για το ένα άκρο και ακολούθως να γυρίσει από την άλλη και να γίνει ακολουθία για το άλλο άκρο.
  - έναντι του BWA- SW προτείνεται για queries υψηλής απόδοσης καθώς είναι πιο γρήγορος (έχει διπλάσια ταχύτητα από το BWA-SW για 100 bp reads) και πιο ακριβής (σε σύγκρισή με το BWA-SW)

4.5.3.2 Αποτελέσματα pBWA

Τα παρακάτω αποτελέσματα (πίνακας 4.7) σχετικά με τον αλγόριθμο pBWA έχουν ληφθεί από μελέτη της Παναγιώτας Μυτιληναίου, το 2014 (Πανεπιστήμιο Κύπρου) [35]. Για την εξαγωγή των αποτελεσμάτων χρησιμοποιήθηκαν αρχεία από το 1000 Genomes Project ανθρώπινες paired-end ακολουθίες από το φάκελο **NA19239** (http://ftp.1000genomes.ebi.ac.uk/vol1/ftp/data/NA19239/sequence_read/).

Τα fastq αρχεία ήταν τα «**SRR014819_1.filt.fastq**» και «**SRR014819_2.filt.fastq**» (paired-end) με μέγεθος ***1.84425* GB** το καθένα.

Αριθμός Κόμβων	Επεξεργαστές ανά κόμβο	Run Time (sec)	Επεκτασιμότητα
15	2	733	59%
15	4	455	52%
15	6	344	53%
15	12	235	55%

Πίνακας 4.7 Αποτελεσμάτων pBWA αλγορίθμου, ως προς το ποσοστό χρήσης του cpu κατά την αύξηση των επεξεργαστών/διεργασιών [35]

Πιο κάτω, στο πίνακα 4.8, φαίνονται οι αντίστοιχες μετρικές για τον αλγορίθμου «Aligner» της NGS data analyses platform, με είσοδο τα αρχεία **READS151_L001_R1_001.fastq**.

Αριθμός Κόμβων	Επεξεργαστές ανά κόμβο	Run Time (sec)	Επιθυμητό Run Time (sec)	Επεκτασιμότητα
1	12	940	940	98,2%
1	24	475	470	98,95%
2	24	240	235	97,92%

Πίνακας 4.8 Αποτελεσμάτων «Aligner» αλγορίθμου, ως προς το ως προς το ποσοστό χρήσης του cpu κατά την αύξηση των επεξεργαστών/διεργασιών

4.5.3.3 Σύγκριση αποτελεσμάτων

Στόχος της ακόλουθης σύγκρισης είναι να εξηγηθεί και να αποδειχθεί ότι με την παραλληλοποίηση του αλγορίθμου ευθυγράμμισης της NGS data analyses platform εκμεταλλεύεται (στο μέγιστο δυνατόν) σε μεγαλύτερο ποσοστό το κάθε επεξεργαστή που χρησιμοποιείται, σε σχέση με τον pBWA. Για να επιτευχθεί αυτό θα υπολογίσουμε το ποσοστό εκμετάλλευσης των cpus κατά τη παραλληλοποίηση του αλγορίθμου «Aligner» της NGS data analyses platform.

Βάσει το πίνακα 4.6, παρατηρούμε ότι ιδανικά η μείωση του παράλληλου RunTime, για τον αλγόριθμο pBWA [21], από 30 διεργασίες (πρώτη γραμμή του πίνακα) μέχρι τις 60 διεργασίες (δεύτερη γραμμή του πίνακα) που εκτελούνται παράλληλα, θα ήταν 2 φορές μικρότερο, δηλαδή ο χρόνος 733 seconds να μειωθεί στα 733/2=366,5. Αυτό όμως δεν ισχύει, αφού μειώνεται στα 450 seconds. Ο Parallel Burrows-Wheeler Aligner αλγόριθμος, μειώνει το RunTime/CPUtime κατά 733/450=1,629 φορές. Με βάσει τις μετρήσεις που έγιναν κατά τη μελέτη από την οποία λήφθηκαν τα δεδομένα το **ποσοστό επεκτασιμότητας είναι 55%.**

Αντίστοιχα, για τον αλγόριθμο ευθηγράμμισης της της NGS data analyses platform, με βάσει το πίνακα 4.7 εκμεταλλεβόμαστε σχεδόν στο έπακρο, κάθε διαθέσιμο επεξεργαστή που μας δίνεται μειώνοντας το run time που χρειάζεται για να ολοκληροθούν οι απαιτούμενες συγκρίσεις.Για να υπολογίσουμε την ακριβή αξιοποίηση κάθε διαθέσιμου επεξεργαστή κάνουμε τους εξής υπολογισμούς:

- Ιδανική μείσωση run time απο 12 cpus σε 24: 2 φορές.
- Πραγματική μείσωση run time απο 12 cpus σε 24: 940/475=1,9789
- Ποσοστό πραγματικής αξιοποίησης κάθε νήματος: 1,9789/2 = 0.9895 = **98,95% επεκτασιμότητας NGS data analyses πλατφόρμα.**

Αναλογιζόμενοι ότι το ακριβές ποσοστό αξιοποίησης κάθε διαθέσιμου νήματος που δίνεται για παράλληλη εκτέλεση από τον αλγόριθμο pBWA είναι 55% , ενώ από τον Aligner αλγόριθμο της NGS data analyses platform είναι: 98,95% , συμπεραίνουμε ότι ο Aligner αλγόριθμος της NGS data analyses πλατφόρμας που υλοποιήθηκε, εκμεταλλεύεται κατά 98,95-55=44,95% περισσότερο τη παραλληλοποίηση. Το σημείο αυτό αποτελεί ένα πλεονέκτημα της νέας πλατφόρμας, σε (βιο)ϊατρικά γενετικά θέματα όπου ο χρόνος για μία γενετική ανάλυση έχει καίριο ρόλο.

4.5.4 Σύγκριση κωδικοποίησης του Aligner αλγόριθμου της NGS data analyses platform με αυτή της εταιρίας 'Illumina'

4.5.4.1 Illumina

Η Illumina, Inc [23] είναι μια αμερικανική εταιρεία που έχει συσταθεί τον Απρίλιο του 1998 που αναπτύσσει, κατασκευάζει και εμπορεύεται ολοκληρωμένα συστήματα για την ανάλυση της γενετικής ποικιλομορφίας και βιολογικές λειτουργίες και αλγορίθμους. Χρησιμοποιώντας τις τεχνολογίες της, η εταιρεία παρέχει μια σειρά προϊόντων και υπηρεσιών που εξυπηρετούν αλληλουχίες, γονότυπους και την έκφραση του γονιδίου. Οι πελάτες περιλαμβάνουν γονιδιωματικά ερευνητικά κέντρα, φαρμακευτικές εταιρείες, ακαδημαϊκά ιδρύματα, ερευνητικούς οργανισμούς κλινικές και εταιρείες βιοτεχνολογίας. Παρέχουν εργαλεία στους ερευνητές, τα οποία δίνουν τη δυνατότητα να εκτελέσουν γενετικές εξετάσεις/ αναλύσεις, που απαιτούνται για την εξαγωγή ιατρικών πληροφοριών, εκ των προτέρων, στον τομέα της γονιδιωματικής και πρωτεομικής (genomics and proteomics). Η έδρα της βρίσκεται στο Σαν Ντιέγκο, στη Καλιφόρνια.

4.5.4.1.1 HiSeq 2500 System

Το HiSeq 2500 System sequencer [32] επεξεργάζεται αλληλουχίες υψηλής απόδοσης (high-throughput sequencing), με την ταχύτητα, απλότητα και προσιτότητα ενός desktop NGS (next-generation sequencing/ αλληλουχίας επόμενης γενιάς) συστήματος. Η γρήγορη, ολοκληρωμένη ροή εργασίας από το δείγμα μέχρι τα αποτελέσματα, επιτρέπει την εφαρμογή πολλών sequencing εφαρμογών, συμπεριλαμβανομένων ολόκληρων των γονιδιωμάτων, και των αντιγράφων (transcriptomes), σε μια ενιαία λειτουργία. Αυτό το σύστημα NGS ταιριάζει απόλυτα σε ερευνητικά εργαστήρια, χωρίς την ανάγκη για εξειδικευμένο εξοπλισμό. Τώρα, οι ερευνητές μπορεί να εκτελέσουν οποιοδήποτε

συνδυασμό high- and mid-throughput sequencing εφαρμογών, για να προωθήσουν τις μελέτες τους.

Το HiSeq 2500 System [15] είναι ευέλικτο και επεκτάσιμο NGS σύστημα για οποιαδήποτε εφαρμογή. Με το σύστημα HiSeq 2500, οι ερευνητές μπορούν να εκτελούν πολλαπλές εφαρμογές αλληλουχίας σε μια πλατφόρμα. Το HiSeq 2500 System είναι το μόνο desktop σύστημα NGS, που είναι ικανό να προσδιορίσει την αλληλουχία ενός ολόκληρου του ανθρώπινου γονιδιώματος, σε υψηλό ποσοστό κάλυψης, σε μία εκτέλεση.

Εικόνα 4.5 illumina HiSeq 2500 [22]

Πηγή:http://cdn.trendhunterstatic.com/thumbs/illumina-hiseq-2500.jpeg

4.5.4.2 Αποτελέσματα

	RAPID RUN MODE	HIGH OUTPUT MODE	
		TruSeq v3 Chemistry	HiSeq v4 Chemistry
ChIP-Seq Transcription Factor 15M Reads 1 × 36 bp	40 samples 7 hours	200 samples 2 days	260 samples 29 hrs
mRNA-Seq 40M Reads 2 × 50 bp	15 samples 16 hours	120 samples 5 days	100 samples 2.5 days
Nextera Rapid Capture Exome 37Mb 100x Coverage 2 × 100 bp	20 samples 27 hours	115 samples 10 days	150 samples 5 days
Human Whole Genome >30x coverage	1 sample 27 hours	6 samples 11 days	10 samples 6 days
De novo Sequencing 2.5 Gb Genome 100x Coverage 2 × 250 bp	1 sample 60 hours		

Πίνακας 4.9 Illumina Sequencing System Specifications

Πηγή: http://www.illumina.com/systems/hiseq_2500_1500/system.html

Όπως φαίνεται στο πίνακα 4.9 (μέσα στο κίτρινο περίγραμμα), για μία de novo ευθυγράμμιση (χωρίς τον αρχικό γενετικό χάρτη) , σε αλληλουχίες αρχείων FastQ συνολικού μεγέθους 2,5 Gbits, όπου η κάθε μία από αυτές αποτελείται από 250 bp (base pairs) → 250*2=500 νουκλεοτίδια χρειάστηκαν 60 ώρες συνολικά. Για κάθε ένα από τα νουκλεοτίδια ξοδεύουν 8 bits =1 byte για την αποθήκευσή του και άλλα 8 bits για το αντίστοιχο quality score [15]. Αν όμως, για κάθε νουκλεοτίδιο χρησιμοποιούταν μία άλλη κωδικοποίηση, η οποία θα σπαταλά λιγότερο από 16 bits/νουκλεοτίδιο, τότε θα μπορούσε να εκμεταλλευτεί η υπόλοιπη μνήμη για άλλες επιπλέον γενετικές αλληλουχίες.

Η NGS data analyses platform κωδικοποιεί τις βάσεις των γενετικών αλληλουχιών και των αντίστοιχων quality scores, με συνολικά 5bits/nucleotide, όπως εξηγήθηκε στο υποκεφάλαιο 3.2.2.1. Συγκεκριμένα, χρησιμοποιεί 2 bitsets, ένα για αναπαράσταση των base pair μίας ακολουθίας/ενός read και ένα για αναπαράσταση των αντίστοιχων quality scores. Το πρώτο bitset περιέχει 2 bits για αποθήκευση ενός κωδικοποιημένου νουκλεοτιδίου και το δεύτερο 3 bits για αποθήκευση του αντίστοιχου quality estimate. Αφού το πλήθος των διαφορετικών νουκλεοτιδίων ενός DNA είναι τέσσερα (η αδενίνη (A), η θυμίνη (T), η γουανίνη (G) και η κυτοσίνη (C) – βλέπε υποκεφάλαιο 2.1) για την κωδικοποίησή τους απαιτούνται 2 bits ($4=2^2$ διαφορετικά νουκλεοτίδια). Επίσης εφόσον κατηγοριοποιήσαμε τα quality scores σε 7 επίπεδα, χρειαστήκαμε 3 bits για τη κωδικοποιημένη bits του quality score για μία βάση μόνο. Η συγκεκριμένη κωδικοποίηση επιλέχθηκε για λόγους γρήγορης αναζήτησης, ανάλυσης των bps ενός read και για δέσμευση ελάχιστης δυνατής μνήμης.

4.5.4.3 Σύγκριση Αποτελεσμάτων

Από τα αποτελέσματα του υποκεφαλαίου 4.5.4.2, καταλήγουμε στο συμπέρασμα ότι για ένα read μήκους 250 bp (base pairs) → 250*2=500 νουκλεοτιδίων **η Illumina στο σύστημα HiSeq 2500 χρειάζεται 500*16 bits=8000 bits** (16 bits ανά νουκλεοτίδιο και το quality score της), **δηλαδή 8000/8=1000 bytes** για κάθε ακολουθία

μήκους 250 bp. Ενώ, **η NGS data analyses platform, χρειάζεται 500*5=2500 bits (5 bits ανά base), άρα κάνοντας τους υπολογισμούς σπαταλά 2500/8=312,5 bytes για μία ακολουθία 250 bp.**

Για κατανόηση της διαφοράς αυτής, ας υποθέσουμε ότι ένας ερευνητής επιθυμεί να κάνει alignment ένα αρχείο με 1000000 read μήκους 250 bp, άρα:

- Συνολική απαιτούμενη μνήμη από Illumina HiSeq 2500 σύστημα με FastQ:
  1000 bytes (για κάθε ακολουθία μήκους 250 bp)*1000000(πλήθος read)
  =1000000000 bytes

  953.67 Mbytes = 0.93 GBytes

- Συνολική απαιτούμενη μνήμη από NGS data analyses platform:
  312,5 bytes (για κάθε ακολουθία μήκους 250 bp)*1000000=312500000 bytes

  298.02 Mbytes = 0.29 GBytes

- **Ποσοστό/Λόγος διαφοράς χρήσης της μνήμης:**

  953.67/298.02 = 3,206

  Η NGS data analyses platform εξοικονομεί πάνω από 3 φορές περισσότερη μνήμη από ότι το HiSeq 2500 σύστημα με κωδικοποίηση FastQ.

Κεφάλαιο 5

Συζήτηση Αποτελεσμάτων

5.1 Σχολιασμός Αποτελεσμάτων

5.1.1 Αλγόριθμος Split FastQ File

Γενικά για μία εκτέλεση του αλγορίθμου «Split FastQ File», ο χρόνος είναι ανάλογος του πλήθους αλληλουχιών (ή των γραμμών) του αρχείου εισόδου. Αυτό ισχύει αφού ανεξάρτητα από το πλήθος αρχείων εξόδου, ο αλγόριθμος θα διαβάσει κάθε γραμμή του αρχείου (εισόδου) μία φορά ακριβώς, ώστε να τη μεταφέρει σε κάποιο αρχείο εξόδου. Αυτό επιβεβαιώνεται και από τη γραφική παράσταση 4.1, στην οποία είναι οφθαλμοφανές ότι ο χρόνος εκτέλεσης αυξάνεται γραμμικά κατά την αύξηση του όγκου δεδομένων του αρχείου εισόδου που δέχεται το πρόγραμμα.

Παρατηρώντας το γραφική παράσταση 4.1, για εισαγωγή αρχείου μεγέθους ~500000KB απαιτείται χρόνος εκτέλεσης ~160 seconds, άρα σε 1 second δύναται να διαβαστούν δεδομένα εισόδου (αλληλουχίες) μεγέθους 31250 KB συνολικά. Σε ένα τέτοιο χρόνο (1 second) ο αριθμός αλληλουχιών που διαβάζεται είναι αρκετά ικανοποιητικός, γεγονός που μας οδηγεί στο συμπέρασμα ότι δεν απαιτείται καμία παραλληλοποίηση του αλγορίθμου αυτού, σαν ανεξάρτητο τμήμα της

πλατφόρμας μας. Είναι αρκετά γρήγορο για αυτό το λόγο δεν απαιτεί παραλληλοποίηση.

5.1.2 Αλγόριθμος Trimming

Στη γραφική παράσταση 4.2, παρατηρείται **η γραμμική αύξηση του χρόνου εκτέλεσης του αλγορίθμου με την αύξηση του πλήθους reads** (υπο-ακολουθιών της αρχικής αλληλουχίας DNA. Αναλύοντας περισσότερο το πιο πάνω γράφημα, για την αποθήκευση της κωδικοποιημένης αλληλουχίας και του επιπέδου ποιότητας των βάσεων αυτής, αλλά και για το «κόψιμο» των βάσεων με low quality score από τα άκρα των γενετικών ακολουθιών (με τη μέθοδο που αναλύθηκε στο υποκεφάλαιο 3.2.1.3, εικόνα 3.5) σε 2086000 ακολουθίες χρειάστηκαν μόνο 47 seconds. Αναλογικά σκεπτόμενοι, σε 1 second μπορούν να κωδικοποιηθούν, να αποθηκευτούν και να αποκοπούν συνολικά ≈44383 reads,ένας σεβαστός αριθμός για ένα τέτοιο μικρό χρονικό διάστημα. Για το λόγο αυτό, **δε χρειάζεται καμία παραλληλοποίηση** με στόχο την επιτάχυνση της εκτέλεσής του.

5.1.3 Αλγόριθμος Aligner

Παρατηρώντας τη γραφική παράσταση του σχήματος 4.3, με τη χρήση 12 διεργασιών ανά κόμβο και 1 κόμβου, καθώς αυξάνεται το πλήθος των ακολουθιών που θα συγκριθούν, κατά την εφαρμογή του Aligner Algorithm**, ο χρόνος εκτέλεσης αυξάνεται τετραγωνικά**. Άρα κατά την αύξηση του αριθμού reads των δύο αρχείων εισόδου, που θα συγκριθούν μεταξύ τους για να βρεθούν τα επικαλυπτόμενα ζεύγη ακολουθιών (overlapping/matching reads), ο χρόνος για την ολοκλήρωση της εκτέλεσης της διαδικασίας αυτής (run time) αυξάνεται ανάλογα.

Για δύο παράλληλες εκτελέσεις του aligner αλγορίθμου του υποκεφαλαίου 4.5.1, παράχθηκαν τα σχήματα και οι πίνακες 4.4 και 4.5. Με βάσει αυτά, παρατηρούμε ότι **όσο το πλήθος συγκρίσεων που επιβάλλεται να πραγματοποιηθούν αυξάνεται τόσο περισσότερος είναι ο απαιτούμενος χρόνος ολοκλήρωσης του**

αλγορίθμου. Κατ' επέκταση, όσα περισσότερα τα reads εισόδου τόσο περισσότερος το run time. **Είναι δηλαδή, μεγέθη ανάλογα**. Στα Σχήματα 4.5 και 4.7, παρατηρούμε **τη γραμμική αύξηση του χρόνου εκτέλεσης καθώς το πλήθος των συγκρίσεων μεταξύ των reads αυξάνεται**. Αντίστοιχα, για τια σχήματα 4.4 και 4.6 παρατηρούμε την τετραγωνική πολυπλοκότητα χρόνου εκτέλεσης κατά την αύξηση του πλήθους των reads που δέχεται σαν είσοδο ο αλγόριθμος.

Επίσης στο Σχήμα4.8 παρατηρούμε ότι καθώς το πλήθος των επεξεραστών που δεσμεύτικαν για παράλληλη εκτ'ελεση του αλγορίθμου Ευθυγράμμισης ο χρόνος εκτέλεσης μοιράστικε (περίπου) στο μισό. Για παράδειγμα, με αρχεία εισόδου με πλήθος 15000 ακολουθίες το κάθε ένα ο χρόνος εκτέλεσης με 12 επεξεργαστές (1 κόμβο και 12 επεξεργαστές ανα κόμβο) ήταν 8000 δευτερόλεπτα, ενώ για τα ίδια αρχεία εισόδου, με 24 επεξεργαστές (2 κόμβους και 12 επεξεργαστές ανα κόμβο), ο χρόνος εκτέλεσης ήταν λίγο πιο πάνω από 4000 δευτερόλεπτα.

5.1.3.1 Επεκτασιμότητα Αλγόριθμου Aligner

Στα υποκεφάλαια 4.5.3.2 και 4.5.3.3, παρουσιάστηκε το ποσοστό επεκτασιμότητας του αλγορίθμου του parallel BWA βάσει έρευνας που έγινε στο Πανεπιστήμιο Κύπρου το 2014 [35] και του αλγόριθμου «Ευθυγράμμισης» της NGS data analyses platform. Τα ποσά αυτά είναι 55% και 98,95%, αντίστοιχα, βάσει των πινάκων 4.7 και 4.8. Για το λόγο αυτό, παρατηρούμε ότι η NGS data analyses platform είναι αρκετά επεκτάσιμη.

5.1.3.1.1 Κόστος

Παρατηρώντας το κόστος κατά την αύξηση του πλήθους των επεξεργαστών, στο σχήμα 4.9, συμπεραίνουμε ότι το κόστος παραμένει σταθερό κατά την αύξηση του πλήθους των επεξεργαστών. Αυτό ισχύει αφού όπως έχει ήδη παρατηρηθεί με την αύξηση του πλήθους των cpus ο χρόνος εκτέλεσης μειώνεται ανάλογα και έτσι ο πολλαπλασιασμός των δύο αυτών τιμών (δηλαδή το κόστος) παραμένει σταθερό.

5.1.3.1.12 Speedup

Παρατηρώντας το κόστος κατά την αύξηση του πλήθους των επεξεργαστών, στο σχήμα 4.10, συμπεραίνουμε ότι το speedup αυξάνεται όσο αυξάνονται οι επεξεργαστές. Δηλαδή το speedup value και το πλήθος των επεξεργαστών είναι μεγέθη ανάλογα. Η μορφή της παράστασης speedup - #cpus του σχήματος 4.10, έχει γραμμική αύξηση.

5.1.3.1.3 Efficiency

Το σχήμα 4.11 δείχνει την αποδοτικότητα (Efficiency) του αλγορίθμου «Ευθυγράμμισης» της NGS data analyses platform. Βλέποντας το σχήμα συμπεραίνουμε ότι η τιμή της αποδοτικότητας είναι (σχεδόν) πάντα σταθερή κοντά στη τιμή 1. Άρα η αποδοτικότητα είναι ψηλή και οι επεξεργαστές που δεσμεύονται αξιοποιούνται στο έπακρο.

5.1.3.1.4 Χωρική Πολυπλοκότητα

Μέγεθος Αρχείου (reads)	NGS data analyses platform	HiSeq 2500 της illumina με κωδικοποίηση FastQ
100	12500 bytes (100000 bits)	40000bytes (320000 bits)
1000	125000 bytes (1000000 bits)	400Kbytes (3200000 bits)
10 000	1250Kbytes (1250000 bytes)	4Mbytes (4000000 bytes)
100 000	12500Kbytes (12500000 bytes)	40Mbytes (40000000bytes)
1 000 000	125Mbytes (125000 Kbytes)	4000Mbytes (400000Kbits)
10 000 000	1250Mbytes (1250000Kbytes)	40Gbytes (40000M bits)
100 000 000	12500Mbytes (12500000Kbytes)	400Gbytes (400000Mbits)

Πίνακας 5.1 Μέγεθος μνήμης που απαιτεί το κάθε σύστημα για τη κωδικοποίησης διαφόρου όγκου δεδομένων-reads και των αντίστοιχων quality score.

****οι υπολογισμοί έγιναν νοουμένου ότι τα reads είναι μήκους 250bp=500 βάσεων.**

Βάσει των υπολογισμών του υποκεφαλαίου 4.5.4.3, για την απαιτούμενη μνήμη που χρειάζεται η NGS data analyses platform για τη κωδικοποίηση των reads και των αντίστοιχων quality scrores, παρατηρούμε ότι η μνήμη αυτή είναι περίπου 3 φορές λιγότερη από μία κωδικοποίηση ενός FastQ αρχείου που κωδικοποιείται από το software HiSeq 2500 της illumina. Αυτό φαίνεται και από το πιο κάτω πίνακα 5.1, ο οποίος συσχετίζει τη κωδικοποίηση της NGS data analyses platform και του software HiSeq 2500 της illumina με FastQ, για reads μήκους 100bps (=200 νουκλεοτιδίων).

Όπως φαίνεται από το πίνακα 5.1 η μνήμη που χρειάζεται η NGS data analyses platformείναι ικανοποιητική εφόσον εξοικονομεί πάνω από το 1/3 της RAM σε σχέση με αυτή του software HiSeq 2500 της illumina με FastQ . Αφενός, αυτό είναι θετικό, αφετέρου όμως, υπάρχει απώλεια ορισμένης πληροφορίας με την εξοικονόμηση αυτή.

Συγκεκριμένα η απώλεια αφορά το quality score. Αφού με βάσει το πίνακα 3.2 το quality score έχει κατηγοριοποιηθεί σε 7 επίπεδα με βάσει το Phred quality [36]. Όμως μία κωδικοποίηση χωρίς απώλειες θα συμπεριλάμβανε κάθε χαρακτήρα από τους 93 (από το ASCII 33 εώς το 126) έτσι θα απαιτούσε περισσότερα από 3bits για τη κωδικοποίηση ενός quality score που αφορά ένα νουκλεοτίδιο. Αυτό βέβαια ίσως μπορεί να αποφευχθεί σε μελλοντική εργασία.

Κεφάλαιο 6

Συμπεράσματα

6.1 Γενικά Συμπεράσματα

- **Αλγόριθμος Διάσπασης Αρχείων FastQ (Split FastQ File):**

Από το σχήμα 4.1 εξάγαμε το συμπέρασμα ότι ο χρόνος εκτέλεσης είναι ανάλογος του μεγέθους του αρχείου εισόδου και ότι λόγο της απλότητας της λειτουργίας του αλγορίθμου δεν απαιτείται κάποια παραλληλοποίηση στην εκτέλεση. Επίσης, με τη λειτουργία του αλγορίθμου αυτού κατορθώνουμε τη παράλληλη επεξεργασία των αρχείων που εξάγει.

- **Αλγόριθμος Κλαδέματος Γενετικής Ακολουθίας (Trimming):**

Από το σχήμα 4.2 εξάγαμε το συμπέρασμα ότι ο χρόνος εκτέλεσης είναι ανάλογος του πλήθους των υπο-ακολουθιών (reads) του αρχείου εισόδου και ότι λόγο της απλότητας της λειτουργίας του αλγορίθμου δεν απαιτείται κάποια παραλληλοποίηση στην εκτέλεση. Επίσης, με τη λειτουργία του αλγορίθμου αυτού κατορθώνουμε να έχουμε βάσεις/ νουκλεοτίδια με ψηλή τιμή ποιότητας (να έχουν χαμηλή πιθανότητα σφάλματος) στα δύο άκρα του κάθε read, εφόσον ξέρουμε ότι στο κέντρο των read δεν τίνουμε να έχουμε βάσεις με μεγάλη πιθανότητα σφάλματος.

- **Αλγόριθμος Παραλληλοποίησης**

Από τη λειτουργία του αλγορίθμου που ήδη αναλύθηκε στα υποκεφάλαια 3.1.3 και 3.2.3, καταλήξαμε στο συμπέρασμα ότι για πλήθος αρχείων στην είσοδο καλεί δημιουργεί αρχεία script με $\binom{N}{2}$+N κλήσεις συνολικά (μία για κάθε δυνατό

ζεύγος αρχείων και κάθε αρχείο με τον εαυτό του). Ο αλγόριθμος εκτελείται παράλληλα, εφόσον ο χρόνος αυξάνεται τετραγωνικά με την αύξηση του μεγέθους των αρχείων εισόδου και εφόσον γνωρίζουμε ότι μετά το τεμαχισμό μιας αλληλουχίας DNA μπορούν να προκύψει ένας τεράστιος αριθμός από reads, όπου το κάθε read μπορεί να υπάρχει πολλές φορές το ίδιο σαν αντίγραφο.

- **Αλγόριθμος Ευθυγράμμισης (Aligner):**

  Μετά από παρατήρηση των σχημάτων 4.4 μέχρι και 4.8 συμπεραίνουμε ότι ο χρόνος εκτέλεσης του αλγορίθμου είναι τετραγωνικά ανάλογος του μεγέθους του αρχείου εισόδου (βλέπε σχήμα 4.4) (πολύ μεγάλος για ανάλυση γενετικών ακολουθιών), για αυτό εκτελείται παράλληλα. Από το σχήμα 4.4 και 4.5, βλέπουμε ότι όσο πιο πολλά τα reads των δύο αρχείων εισόδου τόσο πιο πολύ αυξάνεται τετραγωνικά ο χρόνος. Αυτό είναι λογικό, εάν σκεφτούμε ότι όσο πιο πολλά τα reads των αρχείων εισόδου τόσο πιο πολύ αυξάνετε τετραγωνικά το πλήθος των συνολικ΄ν συγκρίσεων που θα πρέπει να κάνει ο αλγόριθμος μεταξύ των reads. Αρά κατ' επέκταση και ο χρόνος θα αυξάνεται τετραγωνικά. Μετά τη παράλληλη εκτέλεση του αλγορίθμου είχαμε τα εξής χαρακτηριστικά επεκτασιμότητας (βλέπε υποκεφάλαιο 4.5.1 και 5.1.3.1):

  - Κόστος: Παραμένει σταθερό κατά την αύξηση του πλήθους των επεξεργαστών, εφόσον από το σχήμα 4.8, κατανοούμε ότι με την αύξηση των δεσμευμένων επεξεργαστών ο χρόνος μειώνεται.
  - Speedup: Αυξάνεται όσο αυξάνονται οι επεξεργαστές. Δηλαδή το speedup value και το πλήθος των επεξεργαστών είναι μεγέθη ανάλογα, εφόσον με την αύξηση των δεσμευμένων επεξεργαστών ο χρόνος μειώνεται. Άρα ο παράλληλος χρόνος σε σχέση με το σειριακό (με 1 επεξεργαστή) θα είναι όλο και πιο μικρός.

- Αποδοτικότητα (Efficiency): Είναι ψηλή αφού, η τιμή της αποδοτικότητας είναι σταθερή, κοντά στη τιμή 1. Γνωρίζοντας ότι η τιμή της αποδοτικότητας κυμαίνεται στο διάστημα [0,1] και ότι όσο πιο κοντά στην τιμή 1 βρισκόμαστε τόσο πιο αποδοτικά χρησιμοποιούνται οι επεξεργαστές που δεσμεύτηκαν, τότε, οι επεξεργαστές που δεσμεύονται αξιοποιούνται στο έπακρο για αυτόν τον αλγόριθμο.
- Χωρική Πολυπλοκότητα (Μνήμη): Είναι αρκετά μικρή αφού δεσμεύουμε το 1/3 της απαιτούμενης μνήμης για μία κανονική κωδικοποίηση αρχείου FastQ, σε σχέση με τη μνήμη που απαιτεί την από την κωδικοποίηση της εταιρίας Illumina για το σύστημα αλληλούχισης HiSeq2500.

  Λόγο αυτής της μείωσης στην απαιτούμενη μνήμη για την αποθήκευση της κωδικοποιημένης αλληλουχίας και των αντίστοιχων quality score των νουκλεοτιδίων της, υπάρχει το **μειονέκτημα της απώλειας μερικής πληροφορίας σχετικά με τη ποιότητα των νουκλεοτιδίων.**

6.2 Μελλοντική Εργασία

Σε μελλοντικό στάδιο θα μπορούσαν να γίνουν βελτιστοποιήσεις και επεκτάσεις της πλατφόρμας που υλοποιήθηκε στα πλαίσια της διπλωματικής αυτής. Καθώς επισημάνθηκε ήδη ότι η αποδοτική χρήση τις μνήμης για αποθήκευση της κωδικοποιημένης γενετικής ακολουθίας και των ποιοτήτων των νουκλεοτιδίων της έχει το μειονέκτημα της **απώλειας πληροφοριών όσον αφορά τη ποιότητα των νουκλεοτιδίων**, θα μπορούσε να τροποποιηθεί η μέθοδος με την οποία κωδικοποιούνται οι ακολουθίες μειώνοντας αυτό το μειονέκτημα ή καλύτερα να διατηρείται στη μνήμη κάποια επιπλέων πληροφορία για την ακριβή ποιότητα κάθε βάσης της κάθε υπο-ακολουθίας.

Τέλος, μία άλλη επεκτασιμότητα θα ήταν η πρόσθεση ενός νέου πέμπτου αλγορίθμου στη πλατφόρμα που να αξιοποιεί τα δεδομένα εξόδου του αλγορίθμου Ευθυγράμμισης. Δηλαδή, ένας αλγόριθμος που να λαμβάνει τα δεδομένα που χαρτογραφούν το DNA και να τα χρησιμοποιεί για να κάνει κάποια άλλη γενετική ανάλυση, όπως για παράδειγμα να συγκρίνει τη χαρτογραφημένη αλληλουχία DNA με κάποια άλλη (η οποία θα τεμαχιστεί) για να υπολογίζει κατά πόσον η πρώτη αλληλουχία έχει κάποιο insertion (αυθαίρετη επιπρόσθετη βάση) ή deletion (αυθαίρετη διαγραφή κάποιας βάσης), όπου να προκαλεί κάποια σοβαρή ασθένεια στο άτομο με αυτό το DNA.

Βιβλιογραφία

[1] Adrian Heilbut. «next-gen sequencing bioinformatics primer», by bioinfo program retreat, may 22, 2010” .

[2] Altschul SF, Gish W, Miller W. "Basic local alignment search tool. J Mol Biol 1990; 215”.

[3] Altschul SF, Madden TL, Schaefer AA. "Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. Nucleic Acids Res 1997.”

[4] Anthony Bolger et al, Usadel Lab, Rheinsch - Westfalische Technische Hochschule Aachen “GenePattern”.

[5] Blaise Barney, Lawrence Livermore “Introduction to Parallel Computing.”

[6] Bruce Alberts, Alexander Johnson, Julian Lewis, Martin Raff, Keith Roberts, and Peter Walter. “Molecular Biology of the Cell - NCBI Bookshelf”.

[7] “Base pair - Wikipedia, the free encyclopedia”.

[8] Darren Peters, Xuemei Luo, Ke Qiu, Ping Liang. “Speeding Up Large-Scale Next Generation Sequencing Data Analysis with pBWA.”

[9] David Altshuler , Victor J. Pollara, Chris R. Cowles, William J. Van Etten, Jennifer Baldwin, Lauren Linton & Eric S. Lander. “An SNP map of the human genome generated by reduced representation shotgun sequencing : Article : Nature”.

[10] “Dot Matrix Sequence Alignment”.

[11] “dot matrix algorithms”.

[12] Eric T. Wang1,2,7, Rickard Sandberg1,3,7, Shujun Luo4, Irina Khrebtukova4, Lu Zhang4, Christine Mayr5, Stephen F. Kingsmore6, Gary P. Schroth4 & Christopher B. Burge. “Alternative isoform regulation in human tissue transcriptomes : Abstract : Nature”.

[13] “Examples, scripts (job submission files) | www.hpc2n.umu.se”.

[14] “FASTA format - Wikipedia, the free encyclopedia”.

[15] “FASTQ Format”.

[16] Geoffrey M Cooper. “The Cell - NCBI Bookshelf”.

[17] Gotoh O. "An improved algorithm for matching biologicalsequences.J Mol Biol1982."
[18] “GATK | Tool Documentation Index”.
[19] Heng Li, Jue Ruan, and Richard Durbin. “Mapping short DNA sequencing reads and calling variants using mapping quality scores”.
[20] Heng Li and Nils Homer. “A survey of sequence alignment algorithms for next-generation sequencing.”
[21] Heng Li, Richard Durbin. “Fast and accurate short read alignment with Burrows-Wheeler transform”.
[22] “HiSeq 2500 System | Rapid and high output run modes”.
[23] “Illumina (company) - Wikipedia, the free encyclopedia”.
[24] Jay Shendure & Hanlee Ji. “Next-generation DNA sequencing : Article : Nature Biotechnology.”
[25] J. W. Thomas, J. W. Touchman, R. W. Blakesley, G. G. Bouffard,S, M. Beckstrom-Sternberg, E. H. Margulies, M. Blanchette,A. C. Siepel, P. J. Thomas, J. C. McDowell.“Comparative analyses of multi-species sequences from targeted genomic regions : Article : Nature”.
[26] Korbel JO, Urban AE, Affourtit JP, Godwin B, Grubert F, Simons JF, Kim PM, Palejev D, Carriero NJ, Du L, Taillon BE, Chen Z, Tanzer A, Saunders AC, Chi J, Yang F, Carter NP, Hurles ME, Weissman SM, Harkins TT, Gerstein MB, Egholm M, Snyder M. “Paired-end mapping reveals extensive structural variation in the human genome. - PubMed - NCBI”.
[27] “LRZ: SLURM example job scripts”.
[28] Markus Hsi-Yang Fritz, Ras . “Efficient storage of high throughput DNA sequencing data using reference-based compression”.
[29] Mickael Goujon, Hamish McWilliam, Weizhong Li, Franck Valentin, Silvano Squizzato, Juri Paern and Rodrigo Lopez . “Bioinformatics Tools for Multiple Sequence Alignment < EMBL-EBI”.
[30] Νικόλαος Δάβανος. “πολυμορφισμός γονιδιωματος.”
[31] “Next Generation Sequencing (NGS)/Pre-processing - Wikibooks, open books for an open world”.

[32] “NextSeq 500 NGS System | Both high- and low-throughput desktop sequencing”.

[33] Pagon RA, Adam MP, Ardinger HH. “Illustrated Glossary - GeneReviews® - NCBI Bookshelf”.

[34] Pamela Gehron Robey. “JCI - Series Introduction: Stem cells near the century mark”.

[35] Παναγιώτα Μυτιληναίου. “ΕΦΑΡΜΟΓΗ ΕΥΘΥΓΡΑΜΜΙΣΗΣ ΓΕΝΕΤΙΚΩΝ ΑΚΟΛΟΥΘΙΩΝ ΜΕΣΩ ΤΩΝ ΝΕΑΣ ΓΕΝΙΑΣ ΜΕΘΟΔΩΝ ΠΡΟΣΔΙΟΡΙΣΜΟΥ ΓΟΝΟΤΥΠΟΥ ΣΕ ΥΨΗΛΗΣ ΑΠΟΔΟΣΗΣ ΥΠΟΛΟΓΙΣΤΗ ΜΕ ΧΡΗΣΗ ΤΟΥ BURROWS WHEELER ALIGNER ΑΛΓΟΡΙΘΜΟΥ”.

[36] “Phred quality score - Wikipedia, the free encyclopedia”.

[37] Ramasamy S Annadurai, Vasanthan Jayakumar, Raja C Mugasimangalam, Mohan AVSK Katta, Sanchita Anand, Sreeja Gopinathan, Santosh Prasad Sarma, Sunjay Jude Fernandes, Nandita Mullapudi, S Murugesan and Sudha Narayana Rao. “BMC Genomics | Full text | Next generation sequencing and de novo transcriptome analysis of Costus pictus D. Don, a non-model plant with potent anti-diabetic properties”.

[38] Ray R. Hashemi, Mahmood Bahar, Joshua H. Early, D. Curtis Jamison, and Alexander A. Tyler. “AN ASSOCIATION-BASED WEIGHTING APPROACH FOR PRUNING OF DNA-KEY SEQUENCES”.

[39] Sharon R. Browning. “Missing data imputation and haplotype phase inference for genome-wide association studies - Springer.”

[40] Steven M. Thompson. “sequence aligning dot matrices”.

[41] T.F. Smith, M.S. Waterman. “Identification of common molecular subsequences.”

[42] Tracy Tucker , Marco Marra, Jan M. Friedman. “Massively Parallel Sequencing: The Next Big Thing in Genetic Medicine”.

[43] W.James Kent. “BLAT—The BLAST-Like Alignment Tool.”

[44] W. James Kent, Robert Baertsch, Angie Hinrichs, Webb Miller‡, and David Haussler. “National Human Genome Research Institute. Insertion and Deletion.”

[45] Wei Huang,Jiuxing Liu, Bulent Abali, Dhabaleswar K. Panda. “High Performance Computing (HPC)”

[46] Xin Chen,Ming Li,Bin Ma and John Tromp. “dna sequence encoding cmpression”.

[47] Yi Wang, Henry C.M. Leung, S.M. Yiu, and Francis Y.L. Chin. “MetaCluster 4.0: A Novel Binning Algorithm for NGS Reads and Huge Number of Species | Abstract.”

[48] “23andMe - Genetics 101: What are SNPs?”.

Παράρτημα Α

Στο παράρτημα Α παρουσιάζεται ο κώδικας για τον πρώτο αλγόριθμο Διάσπασης αρχείων FastQ (Split FastQ file Algorithm).Οι βιβλιοθήκες 'string_functions.h' και 'functions_io.h' χρησιμοποιούνται σε όλους τους αλγορίθμους και για αυτό ο κώδικάς τους παρουσιάζεται στο τέλος.

```
/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                          Split Fastq File
 *
 *                        Libraries and Constants
 *
 *     This program, loads the Fastq File which user gives as input in command-
 *line and split it into many files, in which contains x reads in first output
 *file (x=positive intiger that user gives as input in command-line), x+a reads
 *in secnd output line, x+a+a in third, etc. The constant a is equal to 1000, but
 *user can change it as his like.
 *
 *_
 * author:Christa Philippou       email:christa.phil@hotmail.com
 * date: May 2014                 alternative email:christa.philippou@gmail.com
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

#include "string_functions.h"
#include "functions_io.h"
#define lines_per_read 4//standart for FastQ files
using namespace std;
std::string itoa(int value, unsigned int base);

int main(int argc, char *argv[])
{
    cout <<"Version 4May 2015\n\n\n";

       if(argc<2)
       {
       cerr<<"ERROR: must give input file fastq as argument . . . !"<<endl;
       cerr<<"(eg. my_fastq.fastq-don't care about extensions)"<<endl;
       return 1;
       }

       if(atoi(argv[2])<1)
       {
       cerr<<"ERROR: must give integer gratter than 0 as second
argument . . . !"<<endl;
       return 1;
       }
//======================== Load Fastq File ==================================//
       ifstream fileInput;
       //system("PAUSE");
       //check if file is open
       inputfileopen(fileInput,(string)argv[1]);
       int lines=atoi(argv[2]);
       int countf=0;//counter of output files.
       int countr=0;//counter of rows/reads of current output file.


       string str=(string)argv[1]+"Split"+itoa(lines,10)+".F"+itoa(countf,10);
```

```
        ofstream fileOutput;
        outputfileopen(fileOutput,str);
        string str_sequence;

        getline(fileInput, str_sequence);


//====================== Split Fastq File and Write The Outputs ================//
        while(!fileInput.eof())
        {
                for(int i=0; i<lines_per_read;++i)
                {
                        fileOutput << str_sequence<<endl;
                        getline(fileInput, str_sequence);
                }

                ++countr;

                if(countr==lines && !fileInput.eof())
                {
                        countr=0;
                        lines+=1000;//the current output file will contains 1000
                                    //reads more than the previous outut file.
                                    //you can change this constant as you like.
                        fileOutput.close();
                        countf++;

        str=(string)argv[1]+"Split"+itoa(lines,10)+".F"+itoa(countf,10);
                        outputfileopen(fileOutput,str);
                }
        }
        fileOutput.close();
        fileInput.close();
        return 0;
}
```

Παράρτημα Β

Στο παράρτημα Β παρουσιάζεται ο κώδικας για τον δεύτερο αλγόριθμο Κλαδέματος Γενετικών Ακολουθιών (Trimming Algorithm).Οι βιβλιοθήκες 'string_functions.h' και 'functions_io.h' χρησιμοποιούνται σε όλους τους αλγορίθμους και για αυτό ο κώδικάς τους παρουσιάζεται στο τέλος.

```
/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                                Trimming Of Reads
 *
 *                              Libraries and Constants
 *
 *     This program, loads the Fastq File which user gives as input in command-
 *line and save all data that contained in this file in Fastq class.
 *After, loads the Fastq File that user give as input in command-line and save in
 *ram all bps of each line contained in Fastq File. These encoded bps and their
 *qualities stores in a vector. Finally, it prunes the the two ends of each reads
 *according to the quality character that user gived as input. It writes two
 *outputs files:
 *
 *1)the first outpus contains the same content with fastQ file-input but only the
 *lines with reads and with quality scores.
 *2) the reads and quality scores after the trimming.
 *
 *_
 * author:Christa Philippou       email:christa.phil@hotmail.com
 * date: July 2014                alternative email:charista.philippou@gmail.com
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

#include "fastq.h"
#include "myfunctions_io.h"
#include <iostream>
#include <fstream>
#include <string>
#include <stdlib.h> //exit, EXIT_FAILURE

using namespace std;

int main(int argc, char *argv[])
{

	Reads r;
	cout<<sizeof(r);
	if(argc==2)
	{
	cerr<<"ERROR: must give input file fastq as first argument . . . !"<<endl;
	cerr<<"(eg. my_fastq.fastq-don't care about extensions)"<<endl<<endl;
	cerr<<"ERROR: must give character for quality that will remove from fastq file
as second argument . . . !"<<endl;
	return 1;
	}



//==================== Load Fastq File in RAM================================//

	string quality_romove=(string)argv[2];
	ifstream fileInput;
```

```
        ifstream fileInput2;
        ofstream fileOutput;
        ofstream fileProme;

        //check if file is open
        inputfileopen(fileInput,(string)argv[1]);
        Fastq fastq_file;
        cout<<"Size of FastQ empty: "<<fastq_file.size()<<endl;
        fastq_file.load_file(fileInput);
        cout << "Size of FastQ filled: "<<fastq_file.size()<<endl;

        //system("PAUSE");

        string output=(string)argv[1]+"Quality";
        outputfileopen(fileOutput,output);
        fastq_file.create_output(fileOutput);

#ifdef DEBUG
cout<<"avg= ";
cout<< fastq_file.average_quality();
cout<<"\n";
#endif

        inputfileopen(fileInput2,output);
        output=output+"Prune";
        outputfileopen(fileProme,output);
        fastq_file.prune(fileInput2,fileProme,quality_romove[0]);
    //file without prune is not necessary.
        return 0;
}
```

Οι βιβλιοθήκες 'fastq.h' και 'read.h' που χρησιμοποιούνται από τον αλγόριθμο αυτό , όπως και από τους επόμενους αλγορίθμους παρουσιάζονται πιο κάτω (για επεξήγηση της λειτουργίας τους βλέπε υποκεφάλαιο 3.2.2.1).

Βιβλιοθήκη 'fastq.h' :

```
/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                                Loading Fastq File
 *
 *                              Libraries and Constants
 *
 *     This program, loads the Fastq File which user gives as input in command-
 *line and save all data that contained in this file in Fastq class.
 *After, loads the Fastq File that user give as input in command-line and save in
 *ram all bps of each line contained in Fastq File. These encoded bps and thir
 *qualities stores in a vector.
 *
 *_
 * author:Christa Philippou        email:christa.phil@hotmail.com
 * date: Sept 2014                 alternative email:charista.philippou@gmail.com
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

#define read_length 151 //you can change it as you like according to the length
                        //of the reads in input file (=number of nucleotides).
#include "string_functions.h"
```

```
#include "my_functions_io.h"
#include "read.h"
#include <ctime>
#include <iterator>
#include <iostream>
#include <fstream>
#include <string>
#include <stdlib.h>
#include <vector>
#define format_err 9
#define empty_file 22
using namespace std;

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
*                               Class Fastq
*
* This class is used for representing the FastQ file that user gived as input.
*
*
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

class Fastq
{
          public:
              Fastq()
              {
              }
              unsigned long long int size()
              {
                     return my_fastq.size();
              }

              void load_file(ifstream & input, string fname)
              {
                     filename=fname;
                     cout<<"Loading file . . . "<<endl;
                     string str_sequence;
                     string str_quality;
                     string str;
                     short int index=0;
                     Reads tempRead;
                            while(!input.eof())
                                   {
                                   getline(input, str);
                                   getline(input, str_sequence);
                                   getline(input,str) ;
                                   getline(input, str_quality);


if(str_sequence.length()!=str_quality.length())
                                          {
cerr <<"\nError, Nucleotide seq has length "<<str_sequence.length()<< "while quality
has length "<<str_quality.length()<<endl;

exit(format_err);
}
else{
       if(str_sequence.size()==0)
       {
       cerr<<"There is an empty input file . . . !"<<endl;
       exit(empty_file);
       }
```

```
        unsigned short int i=0;
        string::iterator itr_q=str_quality.begin() ;
        string::iterator itr_s=str_sequence.begin();
        tempRead.clear();
               for (; itr_s!=str_sequence.end() || itr_q!=str_quality.end();)
               {
               char neuc=*itr_s;
               char qual=*itr_q;

               tempRead.set_read(neuc,qual,i);

               ++i;
               ++itr_s;
               ++itr_q;
               }
        //cout<<tempRead.get_length()<<endl;
        tempRead.find_end_pos();
        my_fastq.push_back(tempRead);

        }
        }
        input.close();

        }

        void create_output(ofstream&  output)
        {
         cout<<"Creating output . . . "<<endl;
         unsigned long long  int i=0;
         unsigned long long int j=0;
         for(i=0; i<my_fastq.size(); i++)
         {

               for(j=0; j<read_length; j++){

                      if((my_fastq[i].get_qual_c(j))!='0')
                      {
                      output.put(my_fastq[i].get_neuc_c(j));
                      }
                      }

                      output.put('\n');

                      for(j=0; j<read_length; j++){

                             if((my_fastq[i].get_qual_c(j))!='0')
                             {
                             output.put(my_fastq[i].get_qual_c(j));
                              }
                      }

                      output.put('\n');

#ifdef DEBUG
cout<<"avg= ";
cout<<my_fastq[i].average_quality();
cout<<"\n";
#endif
                      }

#ifdef DEBUG
```

```
cout<<"reads= ";
cout<<my_fastq.size();
cout<<"\n";
#endif
                        output.close();
                }

                float average_quality(){
                                int i=0;
                                float sum=0;
                                for(i=0;i<my_fastq.size();++i)
                                {
                                        sum=sum+my_fastq[i].average_quality();
                                }

#ifdef DEBUG
cout<<"sum= ";
cout<<sum;
cout<<"\n";

cout<<"my_fastq.size();= ";
cout<<my_fastq.size();
cout<<"\n";

cout<<"fastq avg= ";
cout<<sum/my_fastq.size();
cout<<"\n";
#endif

                                return sum/my_fastq.size();
                                        }

//create output file with all overlap information of all reads of two input files.

        void createAlignOutput(ofstream &fout)
        {
                for(unsigned long long int i=0; i<my_fastq.size();++i)
                {
                        fout <<my_fastq[i].get_endOverlaps(i);
                }
        }

//compare each read from file1 with each read from file2 and write in a third //file

//"file1_file2.aling" all read's id that overlap.
void compare_fastq(Fastq fq1,Fastq fq2, short int overlap,ofstream &foutAllign)
{
        cout<<"Start comparing files "<<fq1.filename<<" and "<<fq2.filename<<" . . .
"<<endl;

                time_t seconds_begin; time_t seconds_end; struct tm * timeinfo;

                time(&seconds_begin);
                timeinfo=localtime(&seconds_begin);

                cout <<"\nBegining aligning at "<<endl;
                cout <<asctime(timeinfo)<<endl;
//The following 4 lines of code would crash if you had input files of different
//sizes!.
                for(int i=0; i<fq1.size();++i){
                        for(int j=0; j<fq2.size();++j){
```

```
                fq2.my_fastq[j].overlapEndversion11(fq1.my_fastq[i],overlap,i);

                fq1.my_fastq[i].overlapEndversion11(fq2.my_fastq[j],overlap,j);
                        }

                }
                time(&seconds_end);
                timeinfo = localtime ( &seconds_end );
                cout <<"\nFinished comparing files "<<fq1.filename<<" and
"<<fq2.filename<<" at "<<asctime(timeinfo)<<endl;
                cout <<"Total Time Taken ="<<difftime(seconds_end,seconds_begin)<<"
seconds.\n";


                fq1.createAlignOutput(foutAllign);

        }

//for each read removes all suffix and prefix bps that their quality is lower //than
or equal to <qual>.
                void prune(ifstream & input,ofstream&  output,char qual_prome)
                {
                        cout<<"Starting prune of file . . . "<<endl;
                        string str_sequence;
                        string str_quality;
                        bool flag=false;

                        unsigned short int i=0;

                        while(!input.eof())
                        {i=0;
                        getline(input, str_sequence);
                        getline(input, str_quality);

                        string::iterator itr_q=str_quality.begin() ;
                        string::iterator itr_s=str_sequence.begin();
                        Reads *read=new Reads();
                        vector <char> quality;

                        for (; itr_s!=str_sequence.end() || itr_q!=str_quality.end();)
                {
                        char neuc=*itr_s;
                        char qual=*itr_q;



                        if(qual<=qual_prome && flag==false)
                        {
#ifdef DEBUG
cout<<"neuc= ";
cout<<neuc;
cout<<"qual= ";
cout<<","<<qual;
cout<<"\n";
cout<<"qual_prome= ";
cout<<(qual<=qual_prome);
cout<<"\n";
#endif
                        }
                        else{

                        if(!input.eof()){
```

```
                        read->set_read(neuc,qual,i);
                        quality.push_back(qual);
                        flag=true;
                        ++i;
                        }
                        else
                                break;
#ifdef DEBUG
cout<<"neuc= ";
cout<<neuc;
cout<<"\n";
cout<<"qual= ";
cout<<qual;
cout<<"\n";
#endif

                        }
                        ++itr_s; ++itr_q;
                        }//prome the prefix
                        flag=false;
                        itr_q=str_quality.end() ;
                        short int a=i;
                        short int b=0;

                        for (; itr_q!=str_quality.begin();)
                        {
                        char qual=*itr_q;

                        if(qual<=qual_prome && flag==false)
                        b++;
                        else
                        {

                        flag=true;
                        }

                        --itr_q;
                        }//prome the sufix

                        a=a-b;
                        i=0;

                        for (; i<=a;)
                        {
                        output.put(read->get_neuc_c(i));
                        ++i;
                        }//write sequence after prome.

                i=0;
                if(!input.eof())
                {
                        output.put('\n');
                }
                for (; i<=a;)
                {
                        output.put(quality[i]);
                        ++i;
                }//write quality after prome.

                if(!input.eof())
                {
                output.put('\n');
```

```
            }

            free(read);
            //output.put('\n');
            flag=false;
            }

            output.close();
            input.close();

        }
        void print_sizeOfread(){
                cerr<<"\nsize of read: "<<sizeof(my_fastq[1])<<endl;
        }
        private:
            vector <Reads> my_fastq;
            string filename;


};
```

Βιβλιοθήκη 'read.h' :

```
/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
*                              Class Reads
*
* This class is used for representing the coding for each bp
* It contains a bitset of 300*2 bits to save the coding bp and
* 300*3 bits to save bp's qualities.
*_
* author:Christa Philippou               email:christa.phil@hotmail.com
* date: Sept 2014
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

#ifndef READ_____H
#define READ_____H

#include <limits>
#include <vector>
#include <bitset>
#include <iostream>
#include <string>
#include <iterator>
#include <stdlib.h>
#include "string_functions.h"
//#define DEBUG
#define read_length 151
#define out_of_bounds 11
#define if_err 12
using namespace std;

struct overlap_info{
            unsigned short int end_bsPosition;
            int end_id;
            //integer that show in a specific position of specific read that
            //the beggining of this read overlap with.
            //specific read's id that overlap with.
            };
```

```
/*
00-A,  01-T,  10-C,  11-G
*/

class Reads{
        public:
              Reads()
              {
              this->end_pos=read_length;
              }

              Reads(const Reads &in)
              {
              *this=in;

              }
              void clear()
              {
                     sequence.reset();
                     quality.reset();
              }

                //set the 1st bit in our bitset
                void set_read(char neucleotide, char qual, short index)
                {
                     if(index>=read_length || index<0)
                     {
                            cerr<<"out of bounds . . . !\n";

                     exit(out_of_bounds);
                     }

                     if (neucleotide=='A' || neucleotide=='N')
                     {

                     sequence.set(2*index,0);

                     sequence.set(2*index+1,0);
                                                              }
                     else
                     if (neucleotide=='T'){

                     sequence.set(2*index,0);

                     sequence.set(2*index+1,1);
                     }
                     else
                     if (neucleotide=='C'){

                     sequence.set(2*index,1);

                     sequence.set(2*index+1,0);
                     }
                     else
                     if (neucleotide=='G'){

                     sequence.set(2*index,1);

                     sequence.set(2*index+1,1);
                     }
                     else
```

```
                    {
                    cerr<<"There is a neucleotype that isn't
'A','T','C','G' . . . !("<<neucleotide<<")'\n";

               exit(out_of_bounds);
                    }

       if(((int)qual)>=33 && (qual)<43)
       {
       this->quality.set(3*index,0);
                                                          quality.set(3*index+1,0);
                                                          quality.set(3*index+2,1);

       }

       else
       if(((int)qual)>=43 && ((int)qual)<53){
                                                           quality.set(3*index,0);
                                                          quality.set(3*index+1,1);
                                                          quality.set(3*index+2,0);

       }

       else
       if(((int)qual)>=53 && ((int)qual<63))
       {
                                                           quality.set(3*index,0);
                                                          quality.set(3*index+1,1);
                                                          quality.set(3*index+2,1);

       }

       else
if(((int)qual)>=63 && ((int)qual)<73)
{
quality.set(3*index,1);
quality.set(3*index+1,0);
quality.set(3*index+2,0);
}
else
if(((int)qual)>=73 && ((int)qual<83))
{
quality.set(3*index,1);
quality.set(3*index+1,0);
quality.set(3*index+2,1);
}
else
if(((int)qual)>=83 && ((int)qual)<93)
{
quality.set(3*index,1);
quality.set(3*index+1,1);
quality.set(3*index+2,0);
}
//((((int)qual)>=93)
else
{
quality.set(3*index,1);
quality.set(3*index+1,1);
quality.set(3*index+2,1);
}
}

////return the specific read from bit set
char get_neuc_c(short int index)
{
```

```
        if(index>read_length || index<0)
        {
        cout<<"out of bounds . . . ! (in function get_neuc_c)\n";
        exit(out_of_bounds);
        }

#ifdef DEBUG
cerr <<endl<<"bit1="<<sequence[2*index]<<" bit2="<<sequence[2*index+1]<<endl;
#endif

        if(sequence[2*index]==false)
                if(sequence[2*index+1]==false)
                        return 'A';
                 else//if sequence[2*index+1]==true
                        return 'T';
                  else//if sequence[2*index]==true
                        if(sequence[2*index+1]==false)
                                return 'C';
                         else//if sequence[2*index+1]==true
                                return 'G';
   }

   ////return the quality of specific read from bit set
   char get_qual_c(short int index)
   {
      if(index>read_length || index<0)
        {
        cout<<"out of bounds . . . !  (in function get_qual_c)\n";
        exit(out_of_bounds);
        }

        if(quality[3*index]==false)
                if(quality[3*index+1]==false)
                        if(quality[3*index+2]==false)
                                return '0';

                else//quality[3*index]==false && quality[3*index+1]==false

                if(quality[3*index+2]==true)
                                return '!';
                        else//quality[3*index]==false && quality[3*index+1]==true

                        if(quality[3*index+2]==false)
                                return ',';
                         else//quality[3*index]==false && quality[3*index+1]==true

                if(quality[3*index+2]==true)
                                return '5';
                        else//quality[3*index]==true

        if(quality[3*index+1]==false)
        if(quality[3*index+2]==false)
                                return '?';
                        else//quality[3*index]==true && quality[3*index+1]==false &&
                            //quality[3*index+2]==true
                                return 'I';

                else//quality[3*index]==true

        if(quality[3*index+1]==true)
        if(quality[3*index+2]==false)
                                return 'S';
```

```
                    else//quality[3*index]==true && quality[3*index+1]==true &&
                         //quality[3*index+2]==true)
                         return ']';
                    else
                        return if_err;
}


  int get_qual_d(short int index)
  {
       if(index>read_length || index<0)
       {
              cerr<<"Out of bounds with index "<<index<<" . . . ! (in function
get_qual_d)"<<endl;
              exit(out_of_bounds);
       }

       if(quality[3*index]==false)
              if(quality[3*index+1]==false)
              if(quality[3*index+2]==false)
                                  return 0;

              else//quality[3*index]==false && quality[3*index+1]==false
                         //&& quality[3*index+2]==true
                                return 1;
                   else//quality[3*index]==false
                                                        if(quality[3*index+1]==true)
                                                        if(quality[3*index+2]==false)
                       return 2;

       else//quality[3*index]==false && quality[3*index+1]==true &&
           //quality[3*index+2]==true
           return 3;
                                                        else//quality[3*index]==true &&
quality[3*index+1]==false

       if(quality[3*index+2]==false)
              return 4;

       else//quality[3*index]==true && quality[3*index+1]==false &&
           //quality[3*index+2]==true
              return 5;
                                                        else//quality[3*index]==true

       if(quality[3*index+1]==true)
       if(quality[3*index+2]==false)
                    return 6;
              else//quality[3*index]==true && quality[3*index+1]==true &&
                  //quality[3*index+2]==true
                 return 7;
            else
                    return if_err;
     }


       is_missing(short int index){
              return (quality[3*index]==quality[3*index+1]==quality[3*index+2] &&
quality[3*index]==0);
       }


Reads& operator=(const Reads &in)
```

```
{
	this->end_pos=in.end_pos;
	this->sequence=in.sequence;
	this->quality=in.quality;
	this->overlap_with=in.overlap_with;
	return *this;
}


float average_quality()
{
	int i=0;
	float sum=0;
	int count=0;

	for(i=0;i<read_length;++i)
	{
		if(this->get_qual_c(i)!='0')
		{
			sum+=get_qual_d(i);
			count++;
		}
	}

#ifdef DEBUG
cout<<"sum= ";
cout<<sum;
cout<<"\n";

cout<<"count= ";
cout<<count;
cout<<"\n";
#endif

	return sum/count;
}


void find_end_pos()
{

#ifdef DEBUG
cout<<"length: "<<read_length<<endl;
#endif

	for(unsigned int pos=read_length; pos>=0; --pos)
	{
		if(this->get_qual_d(pos)!=0)
		{
			this->end_pos=pos+1;
#ifdef DEBUG
cout<<"pos: "<<this->end_pos<<endl;
#endif

			break;
		}
	}

#ifdef DEBUG
for (unsigned int x=read_length; x>=0;--x)
	cout<<"position: "<<x<<" has neuc: "<<this->sequence[2*x]<<this-
>sequence[2*x+1]<<endl;
```

```
#endif

	return;
}

//*****************************************************************************
//	It calculate if tow specific reads have overlap (if the begining of
//this read is a part of ending of r2 read).If exist overlap between them
//save all information about overlap (this.id and the position that overlapping
// starts) and returns 0 otherwise returns -1.
//*****************************************************************************

short int overlapEndversion11(Reads &r2, unsigned short int minOverlap, int id)
{
	overlap_info oi;
	int r2i;

	for (int thisI=0; thisI+minOverlap<this->end_pos*2;)
	{

		for(r2i=0;
			this->sequence[thisI+r2i]==r2.sequence[r2i] &&
			this->sequence[thisI+r2i+1]==r2.sequence[r2i+1]
			;)
		{
			if(r2i+2==(r2.end_pos*2))
			{
				oi.end_id=id;
				oi.end_bsPosition=0;
				overlap_with.push_back(oi);
				//set the new overlap association between two reads

				return 0;//this->end_pos;
			}
			else if (thisI+r2i+2>=(this->end_pos*2))
			{
				oi.end_id=id;
				oi.end_bsPosition=thisI/2;
				overlap_with.push_back(oi);
				//set the new overlap association between two reads

				return 0;
			}
			r2i=r2i+2;
		}
		thisI=thisI+2;
		}
		return -1;
		}

//version10
short int overlapEndversion10(Reads r2, short int overlap)
{
	vector<unsigned short int> startOverlap;
	vector<unsigned short int> endOverlap;
	vector<unsigned short int> deletePositions;
	int maxOvelap=numeric_limits<int>::max();

	for(int i=0; i<r2.end_pos; ++i)
	{
		if(r2.get_neuc_c(i)==this->get_neuc_c(0))
		{
```

```
                        startOverlap.push_back(i);
                        endOverlap.push_back(i);
                }

        }//find all positions in r2 object-read that is equal to the first
         // neuc in the second read and save the positions in vectors.

        bool flag=true;
        for(int i=1; (endOverlap.empty()==false) &&
                        (flag==true) && (i<this->end_pos); ++i)
        {
                for(int x=0; x<endOverlap.size(); ++x)
                {
                        vector<unsigned short int>::iterator y;

                        if(endOverlap.at(x)<r2.end_pos-1)
                        {

                        if(r2.get_neuc_c(endOverlap.at(x)+1)==this->get_neuc_c(i))
                        {
                                ++endOverlap.at(x);
                        }
                        else
                        {
                                deletePositions.push_back(x);
                        }
                }

        }

for(int z=deletePositions.size()-1; z>=0; --z)
{
        endOverlap.erase(endOverlap.begin()+deletePositions.at(z));
        startOverlap.erase(startOverlap.begin()+deletePositions.at(z));
}

deletePositions.clear();

int countOverlaps=0;

for(int k=0; k<endOverlap.size(); ++k)
        if(endOverlap.at(k)==r2.end_pos-1)
                countOverlaps++;

if(countOverlaps==endOverlap.size())
        flag=false;

}

vector<unsigned short int>::iterator y1;
vector<unsigned short int>::iterator y2;
y2=startOverlap.begin();

for(y1=endOverlap.begin(); (y2!=startOverlap.end() && y1!=endOverlap.end());
        ++y1)
{
        if(maxOvelap>*y2)//max ovelap start from minimum position
                maxOvelap=*y2;
        ++y2;
}
if(maxOvelap==numeric_limits<int>::max())
        return -1;
```

```
else
	return maxOvelap;
}

//It prints in binary athe sequence and the quality of a specific read.
ostream& print_binary(ostream& strm)
{
	strm<<"sequence: ";

	for (unsigned int x=0; x<read_length;++x)
		strm<<sequence[2*x]<<sequence[2*x+1];

		strm<<endl<<endl<<"quality: ";

	for (unsigned int x=0; x<read_length;++x)
		strm<<quality[3*x]<<quality[3*x+1]<<quality[3*x+2];

	strm<<endl;
}

//It prints the sequence and the quality of a specific read as characters.
ostream& print_char(ostream& strm)
{
	strm<<"sequence: ";

	for (unsigned int x=0; x<read_length;++x)
	strm<<this->get_neuc_c(x);

	strm<<endl<<endl<<"quality: ";

	for (unsigned int x=0; x<read_length;++x)
	strm<<this->get_qual_d(x);

	strm<<endl;
}

//return a string with all read's id and lenth of overlap that overlaps with read with
// id that's equal to argument that be given
string get_endOverlaps(int id)
{
	string str;

	for(int i=0;i<this->overlap_with.size(); ++i)
	{
		str+=itoa(lengthOfOverlap(this-
>overlap_with.at(i).end_bsPosition),10)+"\t"+itoa(id,10)+"\t"+itoa(this-
>overlap_with.at(i).end_id,10)+"\n";
	}

	return str;
}

short int lengthOfOverlap(short int start_pos_overlap)
{
	return this->end_pos-start_pos_overlap;
}

//It prints the position that starts the overlap between a specific read and all other
reads.
void print_overlap()
{
	for(short int i=0; i<overlap_with.size(); ++i)
```

```
	{
		cout<<overlap_with.at(i).end_id<<" overlap "<<this-
>overlap_with.at(i).end_bsPosition<<endl<<endl;

	}
}

protected:
		vector<overlap_info> overlap_with; //array of a struct that //contains
		the read's id that overlap with this specific read and
		//the position that overlap starts, in each cell.

		unsigned short int end_pos;//the position that this specific read
					//ends.

	       bitset<2*read_length> sequence;//bitset that will contain 2*300 bps
					//of a subject's dna

		bitset<3*read_length> quality;//bitset that will contain 3*300 bits
					//for quality estimates about 300 bps
					//of a subject's dna
};

#endif
```

Παράρτημα Γ

Στο παράρτημα Γ παρουσιάζεται ο κώδικας για τον τρίτο αλγόριθμο Παραλληλοποίησης (Parallelization Algorithm).Οι βιβλιοθήκες 'string_functions.h' και 'functions_io.h' χρησιμοποιούνται σε όλους τους αλγορίθμους και για αυτό ο κώδικάς τους παρουσιάζεται στο τέλος.

```
/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* *
 *                                 Run Sub Files
 *
 *                          Libraries and Constants
 *     It creates script files in SLURM with 12 maximun runs in each, with as
 *nodes and tasks
 * per node as users wants. As input user must give as file as he wants for
 *example:
 *
 *file1, file2, file3. This program will creates script files with runs of
 *aligner algorithm,
 *that align the reads of two input files. These inout files of alogner algorithm,
 *could be any pair of possible pairs of initial input files that are given in
 *parallelization algorithm.
 *_
 * author:Christa Philippou              email:christa.phil@hotmail.com
 * date: November 2014
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

#include <sys/types.h>
#include <unistd.h>
#include <cstdlib>
#include <iostream>
#include <string.h>
#include "functions_io.h"
#include "string_functions.h"
#define arg_err 12
#define executableFile "aligner.out"//you can change the name of executable, that
                                  //will be called in script files, as you like.
#define threshold 6//you can change the value of threshold as you like (eg. if
                    //threshold=x that's mean aligner algorithm will
                    //run and will try to find reads that have at least x
                    //neucleotypes as overlap.
#define specification_subFile "#!/bin/bash\n#SBATCH --job-
name=geniaXsingleRun9\n#SBATCH --nodes=2\n# 2 nodes\n#SBATCH --ntasks-per-node=12\n#
Number of tasks to be invoked on each node\n#SBATCH --mem-per-cpu=1024       # Minimum
memory required per CPU (in megabytes)\n#SBATCH --time=23:00:00          # Run time in
hh:mm:ss\n#SBATCH --error=job.%J.out\n#SBATCH --output=job.%J.out\necho \"Starting at
`date`\" \necho \"Running on hosts: $SLURM_NODELIST\" \necho \"Running on
$SLURM_NNODES nodes.\necho \"Running on $SLURM_NPROCS processors.\necho \"Job id is
$SLURM_JOBID\" \n\n";
//in string that contain all spesification of sub file in Slurm you can change
//the number of nodes and of tasks per node as you like.
//by default nodes=1 and tasks-per-node=12.
using namespace std;

int main(int argc, char *argv[])
{
	pid_t pid;

	//in case that user doesn't gave any filename
```

```
    if (argc<2)
    {
     cerr<<"You have to give one or more filenames (eg. filename1 filename2
filename3) . . . !\n";
        exit(arg_err);
    }

    int count=0;
    int file=0;
    string filename="";
    string str=specification_subFile;
    ofstream fileOutput;

    for(unsigned int file_no1=1; file_no1<argc; file_no1++)
    {
      for(unsigned int file_no2=file_no1; file_no2<argc; file_no2++)
      {
       if(count==12)
       {
        file++;

filename=(string)executableFile+"_thresh"+itoa(threshold,10)+"_F"+itoa(file,10)+"_1nod
e_12task.sub";
        outputfileopen(fileOutput,filename);
        fileOutput << str<<"\nwait\necho \"Program finished with exit code $? at:
`date`\"";
        fileOutput.close();
       //execute current subfile

       pid = fork();

       if (pid ==-1)
        perror("fork error");
       else if (pid == 0)
          { //Child
        execlp("qsub", "qsub", filename.c_str(), NULL);
        cout<<"Return not expected. Must be an execve error.n";
       }//InParent
        cout<<"\nParent: Process to submit file "<<filename<<" was born .\n";

        //system("PAUSE");
        //write and close previus subfile
        str=specification_subFile;
        //start a new subfile
        count=0;
        }
       str+="./"+(string)executableFile+" "+(string)argv[file_no1]+"
"+(string)argv[file_no2]+" "+itoa(threshold,10)+" &\n";
       count++;
       }
    }

//write and close previus subfile
    file++;

outputfileopen(fileOutput,(string)executableFile+"_thresh"+itoa(threshold,10)+"_F"+ito
a(file,10)+"_1node_12task.sub");
    fileOutput << str<<"\nwait\necho \"Program finished with exit code $? at:
`date`\"";
    fileOutput.close();
       pid = fork();
```

```
    if (pid ==-1)
      perror("fork error");
    else if (pid == 0)
       { //Child
     execlp("qsub", "qsub", filename.c_str(), NULL);
     cout<<"Return not expected. Must be an execve error.n";
    }//InParent
    cout<<"\nParent: Process to submit file "<<filename<<" was born .\nFinish
creatting sub files\n";

    //system("PAUSE");
    return 0;
}
```

Παράρτημα Δ

Στο παράρτημα Γ παρουσιάζεται ο κώδικας για τον τέταρτο αλγόριθμο Ευθυγράμμισης (Aligner Algorithm).Οι βιβλιοθήκες 'string_functions.h' και 'functions_io.h' χρησιμοποιούνται σε όλους τους αλγορίθμους και για αυτό ο κώδικάς τους παρουσιάζονται στο τέλος.

```
/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                        Aligner Algorithm
 *
 *                        Libraries and Constants
 *
 *     This program, loads the Fastq File which user gives as input in command-
 *line and save all data that contained in this file in Fastq class.
 *After, loads the Fastq File that user give as input in command-line and save in
 *ram all bps of each line contained in Fastq File. These encoded bps and thier
 *qualities stores in a vector. After that, it align each read of the first input
 *file with each read of the second input file and creates one output file, which
 *contains the read's id that overlap at least in X neucleotides (X is given as
 *input parameter from the user)and the number of neucleotides that overlap.
 *
 *_
 * author:Christa Philippou       email:christa.phil@hotmail.com
 * date: Sept 2014               alternative email:christa.philippou@gmail.com
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

#include "my_functions_io.h"
#include "fastq.h"
#include <string>
#include <iostream>
#define ErrorArguments 52

using namespace std;

/* run this program using the console pauser or add your own getch, system("pause") or
input loop */

int main(int argc, char* argv[]) {
	if(argc!=4)
	{
	cerr<<"ERROR: must give two input files fastq as argument . . . !"<<endl;
	cerr<<"ERROR: must give a char-threshold as argument . . . !"<<endl;
	cerr<<"(eg. my_fastq1.fastq my_fastq2 7.fastq-don't care about
extensions)"<<endl<<endl;
	system("PAUSE");
	return ErrorArguments;
	}
//==================== Load Fastq File in RAM==============================//

	unsigned short int threshold=atoi(argv[3]);
	ifstream fileInput1;
	ifstream fileInput2;
	ofstream foutAllign;
	ofstream fileProme;

	//check if file is open

	inputfileopen(fileInput1,(string)argv[1]);
	inputfileopen(fileInput2,(string)argv[2]);
```

```
        Fastq fastq_file1;
        fastq_file1.load_file(fileInput1,(string)argv[1]);
        cout<<(string)argv[1]<<": Size of FastQ filled: "<<fastq_file1.size()<<endl;
        //load file1
        fastq_file1.print_sizeOfread();
        cerr<<"Loading file . . . "<<endl;
        Fastq fastq_file2;
        fastq_file2.load_file(fileInput2,(string)argv[2]);
        cout<<(string)argv[2]<<": Size of FastQ filled: "<<fastq_file2.size()<<endl;
        //load file2
        cerr<<"Loading file . . . "<<endl;
        string files_name=(string)argv[1]+"\t"+(string)argv[2];
        string
str=(string)argv[1]+"_"+(string)argv[2]+".Overlap"+(string)argv[3]+".Align";
                system("PAUSE");
        outputfileopen(foutAllign,str);
        foutAllign <<"overlap neucleotypes\t"<<argv[1]<<"\t"<<argv[2]<<endl;

        fastq_file1.compare_fastq(fastq_file1,fastq_file2,threshold,foutAllign);
        //align two input files.
        return 0;
}
```

Παράρτημα Ε

Ο κώδικας των βασικών βιβλιοθηκών 'string_functions.h' και 'functions_io.h' παρουσιάζεται πιο κάτω.

Βιβλιοθήκη string_functions.h:

```
#ifndef FileSTR____h
#define FileSTR____h
#include <cstdlib>
#include <iostream>
#include <string>
#include <stdlib.h>

using namespace std;

std::string itoa(int value, unsigned int base)
{
	const char digitMap[] = "0123456789abcdef";
	std::string buf;
	if (base == 0 || base > 16)
	{
		cout <<"Wrong base";
		return buf;
	}
	// negative int:
	std::string sign;
	int _value = value;
	// Check for case when input is zero:
	if (_value == 0)
		return "0";
	if (value < 0) {
		_value = -value;
		sign = "-";
	}
	// Translating number to string with base:
	for (int i = 30; _value && i ; --i)
	{
		buf = digitMap[ _value % base ] + buf;
		_value /= base;
	}
	return sign.append(buf);
}
#endif
```

Βιβλιοθήκη functions_io.h:

```
#ifndef athosFileIO_h
#define athosFileIO_h
#include <cstdlib>
#include <iostream>
#include <fstream>

using namespace std;

bool inputfileopen(ifstream &fin, string filename)
{
```

```
        fin.open(filename.c_str());
        if (fin.fail())
        {
           cout <<"\n Error opening file "<<filename.c_str()<<" , perhaps it does
      not exist? \n";
           exit(-1);
           return false;
        }
       return true;
}

bool outputfileopen(ofstream &fout, string filename,bool append=false)
{
        if (append)
           fout.open(filename.c_str(),fstream::app);
        else
           fout.open(filename.c_str());

        if (fout.fail())
        {
           cout <<"\n Error opening file "<<filename<<" .\n";
           exit(-1);
           return false;
        }
        return true;
}

#endif
```