ΔΗΜΟΚΡΙΤΕΙΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΡΑΚΗΣ

**ΤΜΗΜΑ ΜΟΡΙΑΚΗΣ ΒΙΟΛΟΓΙΑΣ ΚΑΙ ΓΕΝΕΤΙΚΗΣ**

# ΥΠΟΛΟΓΙΣΤΙΚΟ ΠΡΟΓΡΑΜΜΑ ΔΙΑΔΙΚΤΥΟΥ ΓΙΑ ΟΜΟΛΟΓΗ ΜΟΝΤΕΛΟΠΟΙΗΣΗ ΚΑΙ DOCKING

---

Διπλωματική Εργασία για Μερική Εκπλήρωση
των Απαιτήσεων για το Πτυχίο του
Μοριακού Βιολόγου και Γενετιστή

---

του Δημοκρίτειου Πανεπιστημίου Θράκης
τμήμα Μοριακής Βιολογίας και Γενετικής

---

του
Κουτσόβουλου Γεώργιου
ΕΑΡΙΝΟ ΕΞΑΜΗΝΟ 2008

ΕΠΙΒΛΕΠΩΝ ΚΑΘΗΓΗΤΗΣ ΓΙΑ ΤΗΝ ΕΚΠΟΝΗΣΗ ΤΗΣ ΠΑΡΟΥΣΑΣ ΔΙΠΛΩΜΑΤΙΚΗΣ ΕΡΓΑΣΙΑΣ

---

Dr Νικόλαος Μ. Γλυκός
Λέκτορας Υπολογιστικής και Δομικής Βιολογίας
Τμήμα Μοριακής Βιολογίας και Γενετικής
Δημοκρίτειο Πανεπιστήμιο Θράκης

# ΠΕΡΙΛΗΨΗ

Η παρούσα διπλωματική εργασία έχει ως σκοπό τη δημιουργία ενός προγράμματος διαδικτύου, με την ονομασία HMDock, σε γλώσσα προγραμματισμού Perl, ικανού για την επιτέλεση δύο διεργασιών. Η πρώτη διεργασία αφορά τη δημιουργία της τρισδιάστατης δομής μιας πρωτεΐνης από την αλληλουχία αμινοξέων που έχει εισαγάγει ο χρήστης, με βάση τις ήδη υπάρχουσες δομές της PDB βάσης δεδομένων (homology modeling). Η δεύτερη διεργασία πραγματεύεται τον υποθετικό προκαθορισμό της ύπαρξης ή μη σύνδεσης μεταξύ της πρωτεΐνης που έχει δημιουργηθεί από την προηγούμενη διαδικασία με καθορισμένα μόρια-προσδέτες (docking).

Για τη δημιουργία του προγράμματος HMDock καθώς και για την ταχύτερη και αποτελεσματικότερη λειτουργία του, χρησιμοποιήθηκαν ορισμένα ήδη υπάρχοντα προγράμματα που διατίθενται ελεύθερα στο διαδίκτυο. Σημείο αφετηρίας για τη λειτουργία του προγράμματος αποτελεί η αμινοξική αλληλουχία εισαγωγής από τον χρήστη. Στη συνέχεια αναζητούνται όλες οι πρωτεϊνικές ακολουθίες της PDB βάσης δεδομένων για την εύρεση αυτών που να έχουν τη μεγαλύτερη ομοιότητα με την αμινοξική αλληλουχία που έχει εισαγάγει ο χρήστης. Εφόσον το ποσοστό ομοιότητας είναι υψηλό, προσδιορίζεται η τρισδιάστατη δομή της υπό μελέτη πρωτεΐνης μέσω της ομόλογης μοντελοποίησης. Τέλος, ο χρήστης μπορεί να προσδιορίσει τις αλληλεπιδράσεις της πρωτεΐνης με ορισμένα μόρια με τη μέθοδο του docking.

Όσον αφορά τις γλώσσες προγραμματισμού που χρησιμοποιήθηκαν, εκτός από τη Perl, με την οποία είναι γραμμένο το κυριότερο μέρος του προγράμματος, κρίθηκε αναγκαίο να χρησιμοποιηθούν και άλλες γλώσσες προγραμματισμού για να καλύψουν τις ανάγκες του προγράμματος όπως η JavaScript, η Python και η Java, ενώ για τη δημιουργία της ιστοσελίδας χρησιμοποιήθηκαν tags της HTML. Μέρος των αποτελεσμάτων από τις παραπάνω διεργασίες παρατίθενται στην ιστοσελίδα, ενώ στο σύνολό τους είναι διαθέσιμα προς "κατέβασμα" μέσω FTP.

Παρουσιάζει ενδιαφέρον το γεγονός ότι παρά την αυξημένη χρήση των διεργασιών αυτών από όλο και περισσότερους ανθρώπους, τα προγράμματα στην πλειονότητα τους είναι δύσχρηστα και ιδιαίτερα πολύπλοκα ως προς τη λειτουργία. Συνεπώς, σκοπός της διπλωματικής εργασίας, εκτός από την επίτευξη της σωστής λειτουργίας του προγράμματος ήταν και η απλοποίηση των μεθόδων, χωρίς βέβαια να μειώνεται η αποτελεσματικότητα τους, αλλά και η δημιουργία ενός περιβάλλοντος που να είναι φιλικό και ευκόλως κατανοητό από τους χρήστες.

Στο παρόν κείμενο γίνεται μια πρώτη εισαγωγή στις επιστήμες της Βιοπληροφορικής και της Υπολογιστικής Βιολογίας, παρουσιάζονται τα προγράμματα που ενσωματώθηκαν και συνδυάστηκαν στο HMDock, αναλύεται λεπτομερώς η λειτουργία του προγράμματος, αναφέρεται βήμα προς βήμα ο τρόπος χρησιμοποίησης του, και τέλος παρουσιάζονται ορισμένες εφαρμογές του.


ΚΟΥΤΣΟΒΟΥΛΟΥ ΓΕΩΡΓΙΟΥ
Τμήμα Μοριακής Βιολογίας και Γενετικής
ΔΗΜΟΚΡΙΤΕΙΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΡΑΚΗΣ

# ABSTRACT

This work includes the creation of a web oriented program, named HMDock, in Perl programming language, capable of conducting two processes. In the first process the user provides a protein sequence to be modelled with known related structures from PDB database (homology modeling). In the second process the newly created three dimensional protein structure is tested against specific ligand molecules to define whether the two molecules bind (docking).

For the creation of the HMDock program, some programs, which are freely distributed in the internet, were used to increase its speed and effectiveness. The user enters an amino acid sequence that works as a starting point for the beginning of the application. Then, the sequence entered by the user is tested against the protein sequences of PDB database to find the homologous sequences. If suitable homologous sequences are found, the prediction of the 3D structure of the protein through homology modeling occurs. The last step is to try to find putative interactions between the newly created protein and some ligands by performing the operation of docking.

The programming languages that were used, apart from Perl in which the main program is written, are JavaScript, Python, and Java. For the creation of the webpage tags of HTML were used. Some results of the operations performed by HMDock can be viewed in the webpage, and all results are available for download by FTP.

We also tried to simplify the methods, without losing their effectiveness, and to create a user-friendly and easy to understand web page enviroment, achieving the reduction of the complexity and difficulty of the program usage.

In the next pages we make an introduction to Bioinformatics and to Computational Biology, we present the programs used, we analyze the functionality of the program, we show step by step its usage, and we present some examples.



KOUTSOVOULOS GEORGIOS
Department of Molecular Biology and Genetics
DEMOCRITUS UNIVERCITY OF THRACE

# ΕΥΧΑΡΙΣΤΙΕΣ

Κατ' αρχάς θα ήθελα να ευχαριστήσω τον επιβλέποντα καθηγητή μου, κύριο Νικόλαο Μ. Γλυκό για το χρόνο που μου διέθεσε, την εμπιστοσύνη που μου έδειξε, καθώς και την καθοδήγησή που μου παρείχε καθ' όλη την προσπάθεια της εκπόνησης της παρούσης διπλωματικής εργασίας

Επίσης ευχαριστώ την οικογένεια μου για την υποστήριξή της κατά τη συγγραφή του παρόντος κειμένου, και ιδιαίτερα την αδερφή μου Κατερίνα Κουτσοβούλου για την πολύτιμη βοήθεια της κατά τη μορφοποίηση του κειμένου.

# ΠΙΝΑΚΑΣ ΠΕΡΙΕΧΟΜΕΝΩΝ

# ΚΕΦΑΛΑΙΟ 1 – ΕΙΣΑΓΩΓΗ

Τα τελευταία χρόνια παρατηρείται μια ραγδαία εξέλιξη των επιστημών της Βιοπληροφορικής και της Υπολογιστικής Βιολογίας, η οποία οφείλεται κατά ένα μεγάλο ποσοστό στις συγκεκριμένες προϋποθέσεις που δημιουργήθηκαν αλλά και στη βαθύτερη κατανόηση των κανόνων που διέπουν την εξέλιξη και την λειτουργία των πρωτεϊνών [1].

Υπάρχει ήδη συσσωρευμένος ένας μεγάλος αριθμός δεδομένων αμινοξικών αλληλουχιών από μια μεγάλη ποικιλία γονιδίων και διαφορετικών οργανισμών. Τα δεδομένα αυτά αντιπροσωπεύουν την βασική γνώση που βρίσκεται στη διάθεσή μας για τα βιολογικά συστήματα. Εντούτοις, η δυνατότητα εκμετάλλευσης αυτών των δεδομένων είναι ακόμα αρκετά περιορισμένη, λόγω της ανικανότητας ακριβής πρόβλεψης της λειτουργίας ή/και της δομής μια πρωτεΐνης εξ' ολοκλήρου από την αμινοξική αλληλουχία της και των περιορισμών που υπάρχουν για την πειραματική εξακρίβωση της λειτουργίας ή/και της δομής της. Είναι αξιοσημείωτο το γεγονός ότι είναι ευκολότερη η πρόβλεψη της λειτουργίας παρά της δομής, ακόμα και αν μέσω της δομής μπορεί να γίνει κατανοητή η βιοχημεία που καθορίζει τη λειτουργία μιας πρωτεΐνης. Ο προσδιορισμός της λειτουργίας ενός μικρού μόνο ποσοστού γονιδίων έχει επιτευχθεί μέσω πειραμάτων, τα οποία αποτέλεσαν τη βάση για την εύρεση της λειτουργίας και χιλιάδων άλλων γονιδίων μέσω της αναγνώρισης ομόλογων αλληλουχιών ή παρόμοιων δομικών μοντέλων. Γίνεται έτσι αρχικά μια υπόθεση για τη λειτουργία της πρωτεΐνης από την αμινοξική αλληλουχία της, και ως εκ τούτου χρειάζεται να διεξαγχθούν λιγότερα πειράματα για τον επακριβή προσδιορισμό της λειτουργίας της [1] [6] [7].

Η χρήση κοινών δομικών μοντέλων προκύπτει από την εξελικτική ιστορία των οργανισμών, καθιστώντας έτσι προφανή την καιροσκοπική και συντηρητική διαδικασία πίσω από την εξελικτική πορεία. Ένας οργανισμός χρησιμοποιεί μόνο τα στοιχεία που έχει στη διάθεσή του για τη λύση ενός συγκεκριμένου "προβλήματος", ανεξάρτητα από την πιθανότητα ύπαρξης καλύτερης λύσης. Εφόσον λυθεί το πρόβλημα, ο οργανισμός θα διατηρήσει με συντηρητικό τρόπο τα στοιχεία που είναι σημαντικά για την λύση, ενώ παράλληλα θα δοκιμάζει διάφορες τροποποιήσεις. Παρατηρείται λοιπόν κάτι προφανές, ότι δηλαδή όσο μεγαλύτερη συγγένεια εμφανίζουν δύο αλληλουχίες, τόσο αυξάνεται η πιθανότητα να έχουν πανομοιότυπη δομή και να επιτελούν παρόμοια λειτουργία. Ένα από τα πολλά παραδείγματα που επαληθεύει τους παραπάνω ισχυρισμούς είναι αυτό του κυτοχρώματος c των θηλαστικών, που με βάση την ποσοστιαία ομοιότητα της αλληλουχίας γίνεται ο σχεδιασμός του φυλογενετικού δέντρου με μεγάλη ακρίβεια. Ένα ακόμα παράδειγμα αποτελεί η ύπαρξη κοινής δομής με συγκεκριμένα αμινοξικά πρότυπα σε κάθε πρωτεΐνη που προσδένει ATP [7].

Εντούτοις, η ουσία των πραγμάτων είναι πιο πολύπλοκη και όχι απόλυτα κατανοητή. Γνωρίζουμε ότι μερικά μόνο αμινοξέα έχουν πανομοιότυπες βιοχημικές ιδιότητες, και ότι πρωτεΐνες με ποσοστό ομοιότητας αλληλουχιών μικρότερο από 20% είναι σχετικά απίθανο να σχετίζονται και άρα θα έχουν διαφορετικές λειτουργίες. Ωστόσο σχεδόν παρόμοιες πρωτεϊνικές δομές μπορεί να προέρχονται από τελείως διαφορετικές αμινοξικές αλληλουχίες, οδηγώντας στο συμπέρασμα ενός πεπερασμένου αριθμού πρωτεϊνικού "πακεταρίσματος" που απαντάται στη φύση (περίπου εκατό) [7]. Η τρισδιάστατη δομή μιας πρωτεΐνης είναι πολύ δεκτική στις περισσότερες αλλαγές αμινοξέων, ακόμα και στη περιοχή λειτουργίας της εκτός από κάποια περιορισμένα τμήματα αμινοξέων που πρέπει να παραμείνουν σταθερά για να προσδίδουν στην πρωτεΐνη τη συγκεκριμένη χημική της ιδιότητα. Παρατηρείται επίσης ότι σε πρωτεΐνες με μήκος εκατό ή παραπάνω αμινοξέων, υπάρχουν συγκεκριμένες τοπολογίες αναδίπλωσης ώστε να ικανοποιούν τις δύο βασικές αρχές των δομικών περιορισμών, την ελαχιστοποίηση του αριθμού των εκτεθειμένων υδροφοβικών πλευρικών αλυσιδών και συγχρόνως την μεγιστοποίηση του αριθμού των δεσμών υδρογόνου στο εσωτερικό της πρωτεΐνης [7].

Είναι λογικό να υποθέσουμε ότι ένας από τους λόγους που οι αλληλουχίες έχουν κοινά χαρακτηριστικά είναι ότι προέρχονται από έναν κοινό κυτταρικό πρόγονο. Αυτό επιβεβαιώνεται από το γεγονός ότι τα πρότυπα αλληλουχίας που είναι σημαντικά για την τριτοταγή αναδίπλωση της πρωτεΐνης έχουν συντηρηθεί, αλλά δεν είναι τελείως συνδεδεμένα με την εκάστοτε λειτουργία της. Για παράδειγμα, κάθε πρωτεΐνη παρόμοιου μεγέθους, με δομή α-έλικα – β-πτυχωτή επιφάνεια θα έχει τα ίδια υδροφοβικά πρότυπα που θα της επιτρέπουν να διατηρήσει αυτήν τη στερεοδιαμόρφωση. Το σύνολο αυτών των προτύπων, ανεξάρτητα από τη λειτουργία της πρωτεΐνης, βοήθησαν στη δημιουργία στατιστικών προσεγγίσεων και στην πρόβλεψη της τριτοταγούς δομής μέσω της ομόλογης μοντελοποίησης [1] [6] [7].

Ένα άλλο πεδίο εφαρμογής των επιστημών της Βιοπληροφορικής και της Υπολογιστικής Βιολογίας, είναι η κατασκευή νέων φαρμάκων. Η δημιουργία ενός ligand-based φαρμάκου (μικρό μόριο που προσδένεται και αλληλεπιδρά με πρωτεΐνες) εξολοκλήρου μέσω πειραματικών μεθόδων είναι μια ακριβή και χρονοβόρα διαδικασία. Η χρήση υπολογιστικών μεθόδων για την εύρεση ή την κατασκευή νέων φαρμάκων έχει συμβάλλει στη μείωση των εξόδων της διαδικασίας [7].

Πρέπει να σημειωθεί ότι οι βιολογικές διεργασίες λαμβάνουν χώρα σε μοριακό επίπεδο και βασίζονται στην αλληλεπίδραση πολύπλοκων βιολογικών μορίων. Συνεπώς η επενέργεια των φαρμάκων ορίζεται από τον τρόπο με τον οποίο αυτά επηρεάζουν τις φυσικές βιολογικές διεργασίες. Έχοντας ως στόχο την πλήρη γνώση των κανόνων που διέπουν την χειραγώγηση των βιολογικών διεργασιών, οι πρωτεΐνες και η αλληλεπίδρασή τους με το φάρμακο πρέπει να παρατηρηθούν με τη βοήθεια υπολογιστικών προγραμμάτων σε μοριακό επίπεδο, ειδάλλως ο καθορισμός των φαρμάκων προσδιορίζεται πειραματικά. Η ιδέα του σχεδιασμού φαρμάκων που βασίζεται στη λεπτομερή γνώση της λειτουργίας και της αλληλεπίδρασής τους με τα βιολογικά συστήματα, με τον όρο ορθολογική σχεδίαση φαρμάκων, έχει αρχίσει να βρίσκει πολλούς υποστηρικτές [7].

Συνεπώς η ορθολογική σχεδίαση είναι μια εναλλακτική μέθοδος για την εύρεση νέων φαρμάκων, καθώς επιστημονικά δεδομένα από πολλούς τομείς, όπως η Μοριακή Βιολογία και η Γενωμική, έχουν αρχίσει να γίνονται διαθέσιμα σε μεγάλη κλίμακα. Αρχικά, μεμονωμένα αποτελέσματα όπως δομές πρωτεϊνών υψηλού ενδιαφέροντος βοήθησαν στο να ξεκινήσει η πρόβλεψη πιθανών ligand-based φαρμάκων με την ανάλυση της αλληλεπίδρασης τους μέσω υπολογιστικών μεθόδων. Με την αύξηση των διαθέσιμων δεδομένων και της γνώσης καθίστατο αναγκαίο ένα σύστημα αυτοματοποίησης για την ορθολογική σχεδίαση φαρμάκων. Σήμερα υπάρχει μια μεγάλη βάση δεδομένων δομής πρωτεϊνών, με κυρίαρχη την PDB βάση δεδομένων και βάσεις ολιγοπεπτιδιών, οι οποίες αυξάνουν συνεχώς. Κρίνεται λοιπόν επιτακτική η χρήση υπολογιστικών εργαλείων για την διαχείριση αυτών των δεδομένων [3] [6] [7].

Φαίνεται καθαρά ότι η σωστή αξιοποίηση της τεχνολογίας που είναι ευρέως διαθέσιμη, αλλά και των γνώσεων που έχουν συσσωρευτεί, προβλέπεται ότι θα οδηγήσει σε σαφώς καλύτερες υπολογιστικές μεθόδους, είτε με βελτίωση των ήδη υπαρχόντων, είτε με τη δημιουργία νέων, με σκοπό την καλύτερη κατανόηση των βιολογικών συστημάτων.

## 1.1 Γενωμική

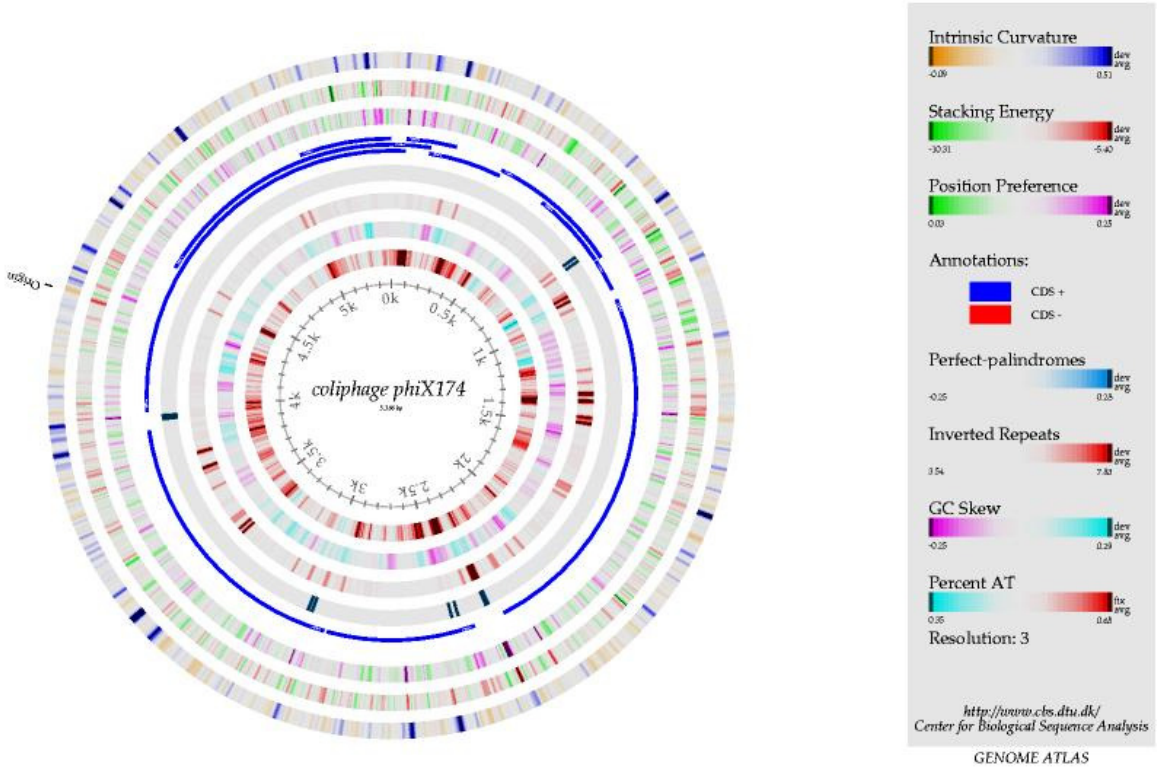
Γενωμική είναι η μελέτη του γονιδιώματος ενός ολόκληρου οργανισμού, και περιλαμβάνει τον καθορισμό ολόκληρης της DNA αλληλουχίας των οργανισμών και τη δημιουργία φυσικής κλίμακας γενετικού χάρτη. Η επιστήμη της Γενωμικής είναι στενά συνδεδεμένη με την επιστήμη της Πληροφορικής, διότι χωρίς την ύπαρξη μεγάλης υπολογιστικής ισχύος υπολογιστών, αλγορίθμων και λογισμικού, οι DNA αλληλουχίες θα ήταν

μικρής πρακτικής σημασίας. Ένα ακόμα προφανές παράδειγμα της σύγκλισης αυτών των επιστημών είναι ο προσδιορισμός πρωτεϊνικών δομών, αφού οι τεχνικές της κρυσταλλογραφίας και του NMR για τον καθορισμό των δομών υπόκεινται στο νόμο του Moore ο οποίος διατυπώθηκε το 1965 και αφορά την εκθετική αύξηση του αριθμού των τρανζίστορ (διπλασιάζεται περίπου κάθε δύο χρόνια) που μπορούν να τοποθετηθούν σε ένα ολοκληρωμένο κύκλωμα. Ο ίδιος νόμος διέπει και άλλες εφαρμογές, όπως τον αριθμό των πρωτεϊνικών δομών στη PDB βάση δεδομένων κ.ά. Προβλέπεται έτσι ότι στο μέλλον θα παράγονται περισσότερα δεδομένα από ότι θα μπορούν να χρησιμοποιηθούν και να αφομοιωθούν [2] [3] [4] [5].

Από την πρώτη αλληλούχιση γονιδιώματος το 1977 του βακτηριοφάγου Φ-X174 (5,386 bp), την ολοκλήρωση του προγράμματος του ανθρώπινου γονιδιώματος (3,2 Gbp) το 2001 και τη μετέπειτα αλληλούχιση διάφορων άλλων γονιδιωμάτων, έχουν αλληλουχηθεί μέχρι σήμερα συνολικά 854 γονιδιώματα διαφόρων οργανισμών (αριθμός που συνεχώς αυξάνεται). Από αυτά 728 είναι γονιδιώματα ιών, 112 γονιδιώματα προκαρυωτικών οργανισμών, 6 γονιδιώματα μονοκύτταρων ευκαρυωτικών οργανισμών, και 8 γονιδιώματα πολυκύτταρων ευκαρυωτικών οργανισμών. Αξίζει να σημειωθεί ότι σε διαφορετικά στελέχη του ίδιου είδους παρατηρούνται μερικές διαφορές, όπως για παράδειγμα το γονιδίωμα της E.coli ποικίλει από 4,6 έως 5,5 Mbp, καθώς και ο αριθμός των γονιδίων της από 4,085 έως 5,361. Οι λόγοι αυτής της ποικιλομορφίας είναι οι διπλασιασμοί γονιδιώματος, οι ενθέσεις γενετικού υλικού, οι διαγραφές βάσεων DNA, οι αναδιατάξεις γενετικών τόπων, και οι σημειακές μεταλλάξεις [8].

Ένας τρόπος παρουσίασης της πληθώρας των πληροφοριών γύρω από το γονιδίωμα ενός οργανισμού είναι με την κατασκευή DNA ατλάντων. Όλο το βακτηριακό χρωμόσωμα δίδεται ως κυκλικό, με διαφορετικά χρώματα να αντιστοιχούν σε διαφορετικές παραμέτρους που σχετίζονται με ορισμένα δομικά στιχεία του DNA και θα μπορούσαν να συμμετέχουν στην οργάνωση της χρωματίνης. Για τον ίδιο οργανισμό κατασκευάζονται άτλαντες ανάλογα με τις πληροφορίες που θέλουμε να απεικονίζουν (άτλας ιδιοσύστασης βάσεων, γονιδιακός άτλας, άτλας έκφρασης πρωτεϊνών κ.ά). Για τους ευκαρυωτικούς οργανισμούς, ο τρόπος απεικόνισης των πληροφοριών είναι η παρουσίαση καθενός χρωμοσώματος ξεχωριστά και η επέκτασή κάθε τμήματός του που χρωματίζεται ανάλογα με τις διαφορετικές παραμέτρους που θέλουμε να δείξουμε.

Η εικόνα 1-1 παριστά όλο το γονιδίωμα του βακτηριοφάγου Φ-X174, η παρουσίαση του οποίου γίνεται με τη μορφή γονιδιακού άτλαντος. Ο άτλας αποτελείται από τέσσερις εσωτερικούς και τρεις εξωτερικούς κύκλους, που χωρίζονται από τον κύκλο γονιδίων με τις μπλε μπάντες να αντιπροσωπεύουν τα γονίδια. Κάθε κύκλος του άτλαντος αντιστοιχεί σε μια παράμετρο που προσδιορίζεται στον διπλανό πίνακα (διαβάζοντας τον πίνακα από πάνω έως κάτω η παράμετρος αντιστοιχεί από τον εξωτερικό έως τον εσωτερικό κύκλο).




**Εικόνα 1-1**. Γονιδιακός άτλας του βακτηριοφάγου Φ-Χ174.

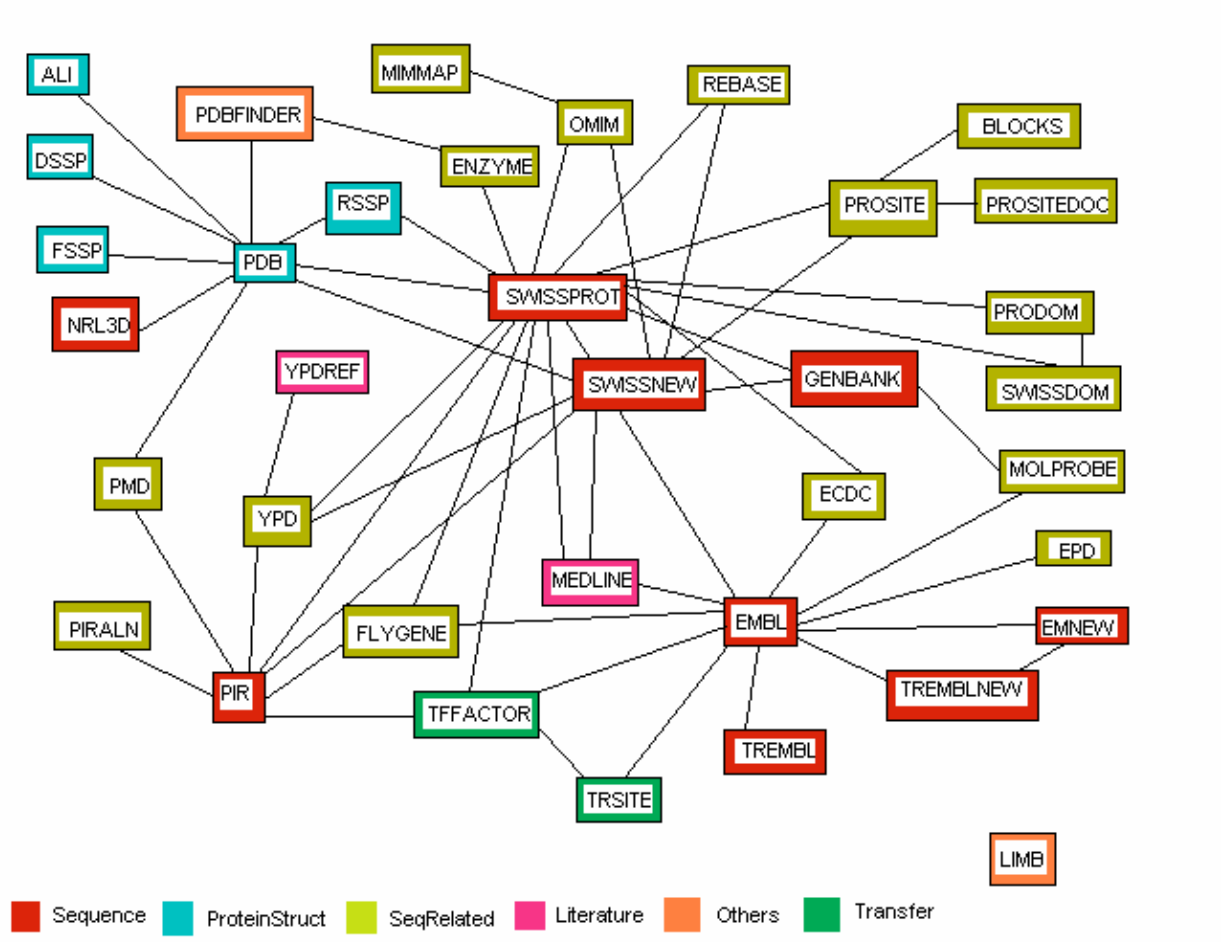
Σε αντίθεση με τις αρχικές προσεγγίσεις γονιδίων, οι DNA αλληλουχίες αποκομίζονται (μαζί με την υποτιθέμενη πρωτεϊνική αλληλουχία που καθορίζουν) χωρίς την προϋπάρχουσα γνώση για την λειτουργία τους. Κατ' ουσίαν τα γονίδια αλληλουχίζονται και ανακαλύπτονται ταυτόχρονα. Όλη αυτή η εξέλιξη δημιούργησε την ανάγκη ανάπτυξης ενός νέου κλάδου στη Βιοπληροφορική ο οποίος θα είναι υπεύθυνος για την ανάλυση μεγάλων βάσεων δεδομένων DNA αλληλουχιών σε επιμέρους στοιχεία (γονίδια, μεταγραφικές μονάδες, περιοχές που κωδικοποιούν πρωτεΐνες, ρυθμιστικά στοιχεία κ.τ.λ) που στη συνέχεια θα ακολουθείται από μια μεγαλύτερη φάση πρόβλεψης της βιολογικής λειτουργίας των γονιδίων.

Σε παραλληλία με την Κλασσική Γενωμική, ένας κλάδος της, η Δομική Γενωμική, έχει δεσμευτεί με την αποσαφήνιση ολόκληρου του ρεπερτορίου των πρωτεϊνικών δομών. Οι τεχνικές κρυσταλλογραφίας ακτίνων-Χ και φασματοσκοπίας NMR παράγουν την τρισδιάστατη δομή από ένα συγχρονισμένο πληθυσμό μορίων -συγχρονισμένων στο χώρο ως ένα διατεταγμένο κρυσταλλικό πλέγμα ή στη συμπεριφορά ως κατάσταση της ιδιοστροφορμής (spin) των πυρήνων, που οργανώνονται από ένα εξωτερικό μαγνητικό πεδίο. Έτσι με την επιλογή πρωτεϊνικών δομών, οι οποίες θα χρησιμοποιηθούν ως αντιπροσωπευτικά μοντέλα για κάθε πιθανή τρισδιάστατη διαμόρφωση, θα γίνει εφικτή η εφαρμογή της τεχνικής της ομόλογης μοντελοποίησης για κάθε πρωτεΐνη. Η Δομική Γενωμική συνδυάζει, περισσότερο από τη Κλασσική Γενωμική, αποτελέσματα από διαφορετικούς οργανισμούς. Οι στόχοι της Δομικής Γενωμικής έγιναν περισσότερο εφικτοί μερικώς από τις πειραματικές τεχνικές που καθιστούν δυνατόν τον προσδιορισμό των πρωτεϊνικών δομών, και μερικώς από την προσπάθεια κατανόησης των νόμων που διέπουν τις πρωτεϊνικές δομές[2] [3] [4] [5].

## 1.2 Βάσεις Δεδομένων

Το αντικείμενο των βάσεων δεδομένων ακολουθιών έχει αντιμετωπιστεί από μεγάλο αριθμό δημοσιεύσεων που δίνουν έμφαση στο αυξημένο ρυθμό ανάπτυξης των βάσεων και προτείνουν διάφορους τρόπους εκμετάλλευσης του μεγάλου μεγέθους των βιολογικών δεδομένων. Από την πρακτική επιστημονική σκοπιά, όπως και από την ιστορική προοπτική, τα δεδομένα των ακολουθιών έχουν διαχωριστεί σε βάσεις δεδομένων για πρωτεϊνικές ακολουθίες και σε βάσεις δεδομένων για νουκλεοτιδικές ακολουθίες. Τα νουκλεοτίδια αποτελούν τα πρωταρχικά σημεία εισόδου στις βάσεις δεδομένων, τόσο για τις πρωτεϊνικές, όσο και για τις νουκλεοτιδικές ακολουθίες και διαφαίνεται μια τάση μετατόπισης προς τη διαχείρηση των συνόλων δεδομένων των ακολουθιών των πρωτεϊνών από τις νουκλεοτιδικές βάσεις δεδομένων. Δεν πρόκειται για αναπάντεχη εξέλιξη, αφού οι καταθέτες ενθαρρύνονται να παράσχουν σχολιασμό για το χαρακτηρισμό της κωδικής αλληλουχίας που προσδιορίζει τον τρόπο παραγωγής ενός προϊόντος μετάφρασης. Αυτή η τάση προς συνδιαχείριση των πρωτεϊνικών και νουκλεοτιδικών ακολουθιών γίνεται φανερή από τις νουκλεοτιδικές ακολουθίες που είναι διαθέσιμες μέσω του Entrez, όπως και από την GenBank και τη μορφοποίηση των εγγράφων στη μορφή GenPept [1].

Ιστορικά, οι πρωτεϊνικές βάσεις δεδομένων αλληλουχιών προηγήθηκαν των νουκλεοτιδικών. Στις αρχές του 1960, ο Dayhoff και οι συνάδελφοι του συνέλλεξαν όλες τις πρωτεϊνικές ακολουθίες που ήταν γνωστές εκείνη την εποχή και τις καταχώρησαν στον Άτλα των Πρωτεϊνικών Ακολουθιών και Δομών (Dayhoff et al., 1965). Η εμφάνιση στο προσκήνιο των βάσεων δεδομένων ακολουθιών DNA το 1982, μετά από πρωτοβουλία του EMBL, οδήγησε στην επόμενη φάση, αυτή της πληροφοριακής έκρηξης στο πεδίο των βάσεων δεδομένων ακολουθιών, και οδηγηθήκαμε στην γένεση μιας πληθώρας βάσεων δεδομένων. Το 1988 συμφωνήθηκε να χρησιμοποιείται μια κοινή μορφή για τα δεδομένα των στοιχείων κάθε μοναδιαίας εγγραφής και, επίσης, η κάθε βάση να ενημερώνει μόνο τις εγγραφές αυτές που κατατέθηκαν άμεσα στην ίδια. Σήμερα οι άμεσες καταθέσεις διανέμονται έτσι ώστε η κάθε μια να έχει αντίγραφα όλων των ακολουθιών, ωστόσο κάθε εγγραφή ανήκει στη βάση δεδομένων που τη δημιούργησε και είναι δυνατό να αναθεωρηθεί μόνο από αυτή (Εικ. 1-2). Πέρα όμως από τις πρωτογενείς αυτές βάσεις δεδομένων (με σημαντικότερες τις EMBL, GenBank και DDBJ), υπάρχουν και οι δευτερογενείς βάσεις δεδομένων όπως η SWISS-PROT και η PIR που ρόλος τους είναι να προσδίδουν προστιθέμενη αξία σε ό,τι ήδη υπάρχει στις πρωτεύουσες βάσεις δεδομένων. Τόσο η SWISS-PROT όσο και η PIR αντλούν την πλειονότητα των πρωτεϊνικών τους ακολουθιών από τις νουκλεοτιδικές βάσεις δεδομένων. Ένα μικρό μέρος των δεδομένων των ακολουθιών της SWISS-PROT κατατίθεται άμεσα ή εισάγεται μετά από αναζήτηση στη βιβλιογραφία, οπότε η ακολουθία αποκτιέται κυριολεκτικά απευθείας από τις δημοσιεύσεις. Παρουσιάζει ενδιαφέρον το γεγονός ότι υπάρχει μία μόνο πρωτογενής βάση δομών επιπέδου ατομικών μοντέλων, η PDB βάση δεδομένων. Ωστόσο κάτι τέτοιο δε προκαλεί έκπληξη, γιατί ήταν η πρώτη βάση δεδομένων "Βιοπληροφορικής" που χτίστηκε για την αποθήκευση πολύπλοκων τρισδιάστατων δεδομένων. Η Protein Data Bank διευθύνεται και υποστηρίζεται τώρα από το Research Collaboratory for Structural Bioinformatics (RCSB), και η συλλογή της περιλαμβάνει όλες τις διαθέσιμες τρισδιάστατες δομές πρωτεϊνών, νουκλεϊκών οξέων, υδατανθράκων και ποικίλων άλλων συμπλόκων οι οποίες έχουν προσδιοριστεί πειραματικά από κρυσταλλογράφους ακτίνων-Χ και φασματοσκοπιστές NMR [1].




**Εικόνα 1-2.** Μερικές βάσεις δεδομένων που είναι διαθέσιμες στο διαδίκτυο.

Από τα προαναφερθέντα, καθίσταται εμφανές ότι υπάρχει μια σημαντική διάκριση μεταξύ πρωτογενών και δευτερογενών βάσεων δεδομένων. Η πιο σημαντική συνεισφορά των βάσεων δεδομένων ακολουθιών στην επιστημονική κοινότητα είναι το ότι καθιστούν τις ίδιες τις ακολουθίες προσβάσιμες. Οι πρωτογενείς βάσεις δεδομένων αναπαριστούν πειραματικά αποτελέσματα, συνοδευόμενα από κάποια ερμηνεία, αλλά δεν αποτελούν φροντισμένη ανασκόπηση (η οποία απαντάται στις δευτερογενείς βάσεις δεδομένων). Υπάρχουν λοιπόν συνέπειες στην ερμηνεία της ανάλυσης των ακολουθιών. Στις περισσότερες περιπτώσεις, όλη η πληροφορία που χρειάζεται ο ερευνητής είναι η ακολουθία. Καθεμιά ακολουθία DNA και RNA θα σχολιαστεί, ώστε να περιγραφεί η ανάλυση των πειραματικών αποτελεσμάτων, που καταδεικνύει για ποιο λόγο έγινε αρχικά ο προσδιορισμός της. Η μεγάλη πλειονότητα των πρωτεϊνικών ακολουθιών δεν έχουν προσδιοριστεί πειραματικά, γεγονός το οποίο μπορεί να έχει επιπτώσεις όταν αυτές αναλυθούν. Για παράδειγμα, η ανάθεση κάποιου ονόματος πρωτεϊνικού προϊόντος ή κάποιου προσδιοριστή λειτουργίας με βάση την υποκειμενική ερμηνεία μιας ανάλυσης ομοιοτήτων μπορεί να είναι πολύ χρήσιμη, αλλά μπορεί να γίνει και παραπλανητική [1] [3].

Είναι απαραίτητο να τονιστεί ότι η ποιότητα της πληροφορίας που είναι αποθηκευμένη στο διαδίκτυο πρέπει να υποβάλλεται σε προσωπική κρίση. Ακόμα και αν τα δεδομένα και οι πληροφορίες προέρχονται από μεγάλες διεθνείς ιστοσελίδες αυτό δε σημαίνει ότι είναι εξ' ολοκλήρου σωστά. Αν και αυτά τα λάθη δε καθιστούν απαραίτητα τα δεδομένα άχρηστα, είναι καλύτερα να το έχουμε υπόψη μας όταν χρησιμοποιούμε αυτές τις πηγές.

## 1.3 Υπολογιστικά Εργαλεία Βιολογίας

Αυτή η ενότητα ασχολείται με δύο σημαντικά εργαλεία της Υπολογιστικής Βιολογίας, την ομόλογη μοντελοποίηση και το docking. Η τεχνική της ομόλογης μοντελοποίησης βρίσκει εφαρμογή στην πρόβλεψη τριτοταγούς δομής πρωτεϊνών και η μέθοδος του docking βρίσκει εφαρμογή στην διαλεύκανση υποθέσεων σύζευξης δύο πεπτιδίων.

### 1.3.1 Homology Modeling

Η πρόγνωση τριτοταγούς δομής που βασίζεται σε δεδομένα πρωτεϊνικών ακολουθιών είναι μια πολύπλοκη και τεχνικά απαιτητική μέθοδος, η οποία βασίζεται σε πιθανότητες πρόσδεσης χωρίς να είναι απόλυτα ακριβής ακόμα. Η σημασία της επαρκούς και ακριβούς πρόγνωσης της δομής με βάση την ακολουθία, έγκειται στο γεγονός ότι γνωρίζουμε ότι, ενώ η ακολουθία μπορεί να προσδιορίσει τη δομή, η ίδια στερεοδιάταξη μπορεί να προκύψει από πολλές διαφορετικές ακολουθίες. Συνεπώς παρατηρείται ότι η δομή συντηρείται σε πολύ μεγαλύτερο βαθμό από ό,τι η ακολουθία και ότι υπάρχει ένας περιορισμένος αριθμός μοτίβων του σκελετού των α-ανθράκων, υποδεικνύοντας έτσι ότι οι ομοιότητες μεταξύ των πρωτεϊνών μπορεί να μην ανευρεθούν απαραιτήτως μόνο μέσω των παραδοσιακών μεθόδων που βασίζονται στην ακολουθία. Συμπεραίνοντας ότι η σχέση μεταξύ ακολουθίας και δομής αποτελεί τη ρίζα του προβλήματος του πρωτεϊνικού διπλώματος, η τωρινή έρευνα γύρω από το πρόβλημα αποτελεί σημείο εστίασης διαφόρων ανασκοπήσεων [1].

Μία από τις ισχυρές τεχνικές πρόγνωσης τριτοταγούς δομής είναι η ομόλογη μοντελοποίηση (homology modeling). Αυτή η τεχνική στηρίζεται στην συσχέτιση μιας πρωτεϊνικής αλληλουχίας-στόχου σε μια πρωτεϊνική αλληλουχία με γνωστή δομή. Αν η αλληλουχία-στόχος έχει μεγάλη ομοιότητα με την αλληλουχία γνωστής δομής, τότε μπορεί να χρησιμοποιηθεί η γνωστή δομή ως οδηγός για την κατασκευή της δομής της πρωτεΐνης στόχου με εντυπωσιακό αποτέλεσμα επιπέδου απόδοσης σε σχέση με την πραγματική δομή [2] [22].

Υπάρχει μια συγκεκριμένη διαδικασία η οποία χρησιμοποιείται στην τεχνική της ομόλογης μοντελοποίησης. Αρχικά χρησιμοποιείται η αλληλουχία της πρωτεΐνης άγνωστης δομής για να βρεθούν όσο το δυνατόν περισσότερο δομές ομόλογων πρωτεϊνών. Στη συνέχεια δημιουργείται μια συσχέτιση μεταξύ της αλληλουχίας-στόχου και των αλληλουχιών-οδηγών και κατασκευάζεται ένα μοντέλο-σκελετός (χωρίς τις πλευρικές ομάδες των αμινοξέων) που θα λειτουργήσει ως οδηγός με βάση τις δομές των πρωτεϊνών οδηγών. Στις περιοχές που δεν μπόρεσε να επιτευχθεί ακριβής μοντελοποίηση (με αποτέλεσμα τη εμφάνιση διακοπτόμενων τμημάτων), είτε στην πρωτεΐνη στόχο είτε στο μοντέλο-οδηγό, χρησιμοποιείται μια διαδικασία μοντελοποίησης (loop modeling) για την αντικατάσταση των τμημάτων στο σωστό μήκος. Τέλος, προστίθενται οι πλευρικές ομάδες των αμινοξέων στο μοντέλο-σκελετό και γίνεται βελτιστοποίηση του μοντέλου με βάση την ενέργεια και άλλες γνώσεις βελτιστοποίησης [2] [3] [22].

Από την παρατήρηση της παραπάνω διαδικασίας προκύπτει ότι το σημείο-κλειδί για την επιτυχία της ομόλογης μοντελοποίησης είναι ο σωστός σχεδιασμός του προγράμματος συσχέτισης των αλληλουχιών και η δημιουργία του κατάλληλου οδηγού δομής. Εάν σχεδιαστεί σωστά το πρόγραμμα τότε αυξάνεται σημαντικά το επίπεδο απόδοσης της υποθετικής δομής. Επίσης, λογικά συμπεραίνεται ότι η ποιότητα του ομόλογου μοντέλου εξαρτάται από το βαθμό ομοιότητας της αλληλουχίας-στόχου με την αλληλουχία-οδηγό. Η προσέγγιση έτσι περιπλέκεται από τα κενά στη δημιουργία του μοντέλου-σκελετού τα οποία καλύπτονται από τη διαδικασία loop modeling, και είναι λιγότερο επακριβή από ότι το υπόλοιπο μοντέλο, ειδικά εάν τα κενά είναι μεγάλα σε μήκος. Τα λάθη στην προσθήκη και την

σωστή τοποθέτηση των πλευρικών ομάδων των αμινοξέων είναι χαρακτηριστικά της ομόλογης μοντελοποίησης σε περιπτώσεις χαμηλού βαθμού ομοιότητας μεταξύ των αλληλουχιών [2] [3].

Πέραν τούτων, η τεχνική της ομόλογης μοντελοποίησης είναι ιδιαίτερα χρήσιμη στην επίτευξη ποιοτικών συμπερασμάτων για τις βιοχημικές ιδιότητες που κρύβονται σε μια πρωτοταγή δομή πρωτεϊνικής αλληλουχίας και ειδικότερα στην επίλυση υποθέσεων που αφορούν ορισμένα συντηρημένα αμινοξέα που βρίσκονται σε συγκεκριμένες θέσεις κατά την τριτοταγή δομή των πρωτεϊνών. Επίσης η τεχνική της ομόλογης μοντελοποίησης μπορεί να παράγει υψηλής ακρίβειας μοντέλα όταν ο βαθμός ομολογίας μεταξύ των αλληλουχιών είναι υψηλός, γεγονός που πυροδότησε την αρχή δημιουργίας αντιπροσωπευτικών δομών για κάθε είδους πρωτεϊνικό "πακετάρισμα". Ακόμα και τα χαμηλής ακρίβειας πρωτεϊνικά μοντέλα μπορούν να χρησιμοποιηθούν με αρκετά καλά αποτελέσματα γιατί τα λάθη δομής τείνουν να παρατηρούνται σε θέσεις εκτός των λειτουργικών περιοχών των πρωτεϊνών, που μπορεί να διαφέρουν αρκετά ακόμα και σε στενά συγγενείς πρωτεΐνες. Οι λειτουργικές περιοχές, και ειδικότερα το ενεργό κέντρο μιας πρωτεΐνης τείνουν να είναι πιο συντηρημένες και έτσι να μοντελοποιούνται με μεγαλύτερη ακρίβεια [2] [3] [22].

### 1.3.2 Docking

Docking είναι η μέθοδος μέσα από την οποία γίνεται πρόβλεψη σύζευξης μεταξύ δύο μορίων. Παράλληλα με την ύπαρξη πρόσδεσης γίνεται και προσδιορισμός του προσανατολισμού του ενός μορίου σε σχέση με το άλλο όταν δημιουργείται ένα σταθερό σύμπλεγμα. Η γνώση της πληροφορίας προσανατολισμού συμβάλλει στην πρόβλεψη της ισχύος πρόσδεσης με τη χρήση διάφορων συναρτήσεων βαθμολογίας. Η μέθοδος του docking χρησιμοποιείται συχνά για την πρόβλεψη σύζευξης μεταξύ μιας πρωτεΐνης και ενός υποψήφιου φαρμάκου για να διαπιστωθεί η περιοχή πρόσδεσης και η ενεργότητα στη θέση αυτή του μικρού μορίου, και συνεπώς, παίζει σημαντικό ρόλο στην ορθολογική σχεδίαση φαρμάκων. Φαίνεται έτσι ότι η αξία της μεθόδου στην φαρμακευτική καθώς και τις βιολογικές διεργασίες είναι σημαντική, για αυτό και γίνονται προσπάθειες βελτιστοποίησης της μεθόδου docking. Docking γίνεται μεταξύ ενός υποδοχέα/μόριο-δέκτης (receptor) πάνω στο οποίο προσδένεται ένα μόριο-προσδέτης (ligand) που μπορεί να είναι ένα μικρό μόριο, μια πρωτεΐνη ή νουκλεϊκό οξύ.

Η ουσία πίσω από τη μέθοδο docking είναι πολύ απλή στη κατανόηση της, η υλοποίηση όμως της μεθόδου προϋποθέτει ο χρήστης να γνωρίζει τις βασικές αρχές της βιολογίας, της χημείας, της φυσικής και των μαθηματικών. Η σκέψη πίσω από τη μέθοδο συνοψίζεται στη λύση ενός προβλήματος κλειδιού-κλειδαριάς (όπως π.χ στα ένζυμα) κατά το οποίο μας ενδιαφέρει η γνώση του σωστού προσανατολισμού του κλειδιού ώστε να συνδεθεί με την κλειδαριά. Η όλη μέθοδος διακατέχεται από μια συνεχή βελτιστοποίηση της λύσης, έτσι ώστε να προβλεφθεί ο καλύτερος προσανατολισμός του προσδέτη σε σχέση με την πρωτεΐνη με την οποία προσδένεται. Πρέπει να ικανοποιούνται λοιπόν δύο κριτήρια, στο σημείο πρόσδεσης τα δύο μόρια να έχουν συμπληρωματικές επιφάνειες και ο προσανατολισμός τους να είναι τέτοιος ώστε η ελεύθερη ενέργεια του συστήματος να ελαχιστοποιείται [24].

Για τις πρωτεΐνες που έχουν αναλυθεί σχεδόν πλήρως, δηλαδή έχουν χαρακτηριστεί οι περισσότερες αλληλεπιδράσεις τους με άλλα μόρια, ο βιολογικός τους ρόλος δεν έχει προσδιοριστεί επακριβώς. Ακόμα και πρωτεΐνες που συμμετέχουν σε απόλυτα κατανοητές βιολογικές διεργασίες (όπως πχ Γλυκόλυση) μπορεί να έχουν λειτουργίες ή βιολογικούς εταίρους εκτός της διεργασίας για την οποία ανακαλύφθηκαν αρχικά. Επιπλέον με την απότομη εξέλιξη της επιστημης της Γενωμικής ένας μεγάλος αριθμός "υποθετικών" πρωτεϊνών ανακαλύφθηκε χωρίς προηγούμενη γνώση επί της λειτουργίας τους, πέραν της αμινοξικής αλληλουχίας. Τέλος, σε καλά μελετημένες αλληλεπιδράσης πρωτεϊνών προκύπτουν νέα

ερωτήματα. Οι γενετικές ασθένειες είναι γνωστό πως προκαλούνται από αλλαγές στη στερεοδιαμόρφωση πρωτεϊνών ή στην αλλαγή της αμινοξικής αλληλουχίας μέσω μετάλλαξης, οπότε υπάρχει ανάγκη να κατανοήσουμε πόσες και ποιες διαφορετικές αλληλεπιδράσεις μπορεί μια μεταλλαγμένη πρωτεΐνη να προκαλέσει [2] [3] [4] [5].

Η μέθοδος docking για αλληλεπιδράσεις μεταξύ πρωτεϊνών προσπαθεί να απαντήσει σε ορισμένα ερωτήματα. Αναλύοντας την υπόθεση ύπαρξης ή μη σύζευξης μεταξύ των πρωτεϊνών ανιχνεύεται και ο προσανατολισμός των πρωτεϊνών όταν είναι συνδεδεμένες προβλέποντας παράλληλα και την ισχύ της πρόσδεσης. Εάν δε διαπιστωθεί ύπαρξη σύζευξης εξετάζεται το ενδεχόμενο πιθανής αλληλεπίδρασης μέσω μεταλλάξεων. Επιπλέον επειδή η μέθοδος του docking στηρίζεται καθαρά σε φυσικές παραμέτρους και κανόνες, ακόμα και πρωτεΐνες άγνωστης λειτουργίας μπορούν να χρησιμοποιηθούν για την εξακρίβωση υποθέσεων πιθανής λειτουργίας. Μοναδική προϋπόθεση είναι να έχει προσδιοριστεί η τριτοταγής δομή των πρωτεϊνών είτε μέσω πειραμάτων (κρυσταλλογραφία, NMR) είτε με προγράμματα πρόβλεψης δομής (πχ τεχνική της ομόλογης μοντελοποίησης) [2] [3] [4] [5].

Η επιτυχία μιας μεθόδου docking εξαρτάται από δυο παραμέτρους, τον αλγόριθμο αναζήτησης και τη συνάρτηση βαθμολογίας. Ο αλγόριθμος αναζήτησης προσδιορίζει το πεδίο αναζήτησης που αποτελείται από όλους τους πιθανούς προσανατολισμούς και τις στεροδιαμορφώσεις της πρωτεΐνης με προσδεδεμένο το μόριο-προσδέτης. Με την παρούσα υπολογιστή ισχύ είναι αδύνατο να εξερευνήσουμε εξονυχιστικά ολόκληρο το πεδίο αναζήτησης – θα έπρεπε να περιλάβουμε όλες τις πιθανές στερεοδιαμορφώσεις καθενός μορίου (ας μην ξεχνάμε ότι τα μόρια δεν είναι στατικά αλλά παρουσιάζονται από ένα σύνολο διαφορετικών καταστάσεων στερεοδιαμόρφωσης). Υπάρχουν πολλές τεχνικές που προσδιορίζουν το πεδίο αναζήτησης ώστε να αυξάνεται η απόδοση των αποτελεσμάτων με την ελάχιστη δυνατή υπολογιστική ισχύ. Η συνάρτηση βαθμολογίας είναι υπεύθυνη για τον υπολογισμό της πιθανότητας ώστε κάθε δυνατή πρόσδεση, που έχει προηγουμένως προσδιοριστεί από τον αλγόριθμο αναζήτησης, να εξεταστεί ως πρός την πιθανότητα ύπαρξης αλληλεπίδρασης. Οι περισσότερες συναρτήσεις βαθμολογίας χρησιμοποιούν φυσικές παραμέτρους με κύρια παράμετρο την ενέργεια κάθε δυνατής πρόσδεσης, δηλαδή χαμηλή ενέργεια υποδηλώνει ένα σταθερό σύστημα και άρα πιθανή υπαρκτή θέση πρόσδεσης [2] [3] [4] [5].

Πρέπει να τονιστεί ότι διαφορετικά προβλήματα docking χρειάζονται διαφορετικές διαδικασίες για να επιλυθούν. Τα σενάρια για docking συνήθως διαχωρίζονται σε μία από τις εξής δύο κατηγορίες, blind docking όταν το ενεργό κέντρο είναι άγνωστο και direct docking όταν το ενεργό κέντρο είναι γνωστό.

## 1.4 Στόχος Εργασίας

Από τα προαναφερθέντα φαίνεται καθαρά το παράδοξο που έχει δημιουργηθεί. Η εύρεση νέων βιολογικών δεδομένων αυξάνεται με ραγδαίους βαθμούς, ενώ η χρησιμοποίηση και κατανόηση αυτών προχωράει με σταθερά βήματα. Δημιουργήθηκε έτσι η κατάσταση των μεγάλων βάσεων δεδομένων που περιέχουν πληροφορίες οι οποίες δεν μπορούν να αξιοποιηθούν πλήρως από τη βιολογική κοινότητα.

Από τις πρωτεϊνικές αλληλουχίες που υπάρχουν διαθέσιμες στο διαδίκτυο υπολογίζεται ότι 20%-30% αυτών έχουν προσδιοριστεί για τη λειτουργία τους (μπορεί όμως να είναι μερική), ενώ για ένα ακόμα μικρότερο ποσοστό αυτών έχει βρεθεί η τριτοταγής δομή τους. Τα ποσοστά αυτά όλο και θα μικραίνουν γιατί η εύρεση των πρωτεϊνικών αλληλουχιών γίνεται ταχύτερα από την αποσαφήνιση της λειτουργίας και της δομής τους.

Στόχος λοιπόν της εργασίας είναι να προσπαθήσει να αντιμετωπίσει αυτή την κατάσταση, βοηθώντας και στην πρόβλεψη των δομών των πρωτεϊνών μέσω της ομόλογης

μοντελοποίησης αλλά και της πιθανής λειτουργίας αυτών μέσω της μεθόδου docking.

## 1.5 Προγραμματισμός

Για την δημιουργία του προγράμματος χρησιμοποιήθηκαν οι γλώσσες προγραμματισμού Perl, Python, Java, και JavaScript, ενώ για το σχεδιασμό της ιστοσελίδας χρησιμοποιήθηκαν tags της HTML. Επιπλέον έγινε χρήση των προγραμμάτων Blast, Modeller, Jmol, και Autodock, τα οποία διατίθενται ελεύθερα στο διαδίκτυο για την επιτέλεση ορισμένων διεργασιών.

### 1.5.1 Γλώσσες Προγραμματισμού

Στο διαδίκτυο υπάρχει ένας μεγάλος αριθμός προγραμμάτων για εφαρμογές Βιοπληροφορικής. Όμως για την καλύτερη κατανόηση του τρόπου με τον οποίο λειτουργούν καθώς και για τη δυνατότητα δημιουργίας ενός προγράμματος που να προσαρμόζεται στις εκάστοτε ανάγκες του κάθε προβλήματος είναι απαραίτητες οι γνώσεις προγραμματισμού.

Το μεγαλύτερο μέρος του προγράμματος δημιουργήθηκε σε γλώσσα προγραμματισμού Perl (Practical Extraction and Reporting Language). Η επιλογή της Perl δεν έγινε τυχαία, αλλά οφείλεται στη δυνατότητα που προσφέρει για γρήγορη και εύκολη διαχείρηση μεγάλων βάσεων δεδομένων, για ενσωμάτωση και άλλων γλώσσεων προγραμματισμού, και για την δημιουργία ιστοσελίδας μέσα στις προγραμματιστικές εντολές της. Είναι γνωστό ότι τα βιολογικά δεδομένα είναι αποθηκευμένα σε πελώριες βάσεις δεδομένων και αρχεία κειμένου. Η Perl με την μεγάλη ικανότητα της να ξεχωρίζει μοτίβα μέσα από δεδομένα και ιδιαίτερα ακολουθίες κείμενου, ήταν η πιο λογική επιλογή. Η Perl είναι σχετικά εύκολη στην εκμάθηση και αφού περιλαμβάνει τη λειτουργική μοναδα CGI.pm καθιστά εύκολο τον σχεδιασμό και τον προγραμματισμό μιας φόρμας καταχωρήσεων. Η όλη αλληλεπίδραση της φόρμας με την εφαρμογή CGI μπορεί να συνοψιστεί σε 6 βήματα [10].

1. Ο χρήστης ζητά μια φόρμα.
2. Ο διακομιστής στέλνει τη φόρμα στον χρήστη.
3. Ο χρήστης συμπληρώνει και υποβάλλει την φόρμα.
4. Ο διακομιστής προωθεί τις καταχωρίσεις στην εφαρμογή CGI
5. Η εφαρμογή CGI επεξεργάζεται τα δεδομένα και στέλνει τα αποτελέσματα στον διακομιστή
6. Ο διακομιστής στέλνει τα αποτελέσματα στον χρήστη

Όλη αυτή η ευκολία της Perl φάνηκε ιδιαίτερα χρήσιμη στο σχεδιασμό του προγράμματος.

Πρέπει να τονιστεί ότι για τον συνδυασμό των προγραμμάτων έπρεπε να γραφτούν κάποια scripts σε γλώσσες προγραμματισμού Python (Modeller, Autodock) και Java (Jmol). Επιπλέον, για τον σχεδιασμό της ιστοσελίδας χρησιμοποιήθηκε η HTML, tags της οποίας απλοποιήθηκαν μέσω συναρτήσεων που διαθέτει η λειτουργική μονάδα CGI.pm. Τέλος χρησιμοποιήθηκε και η JavaScript η οποία πρόσθεσε νέα δυναμικά στοιχεία στην κατασκευή της ιστοσελίδας.

### 1.5.2 Προγράμματα που χρησιμοποιήθηκαν

Τα προγράμματα που χρησιμοποιήθηκαν καθώς και οι εκδόσεις αυτών παρουσιάζονται στον παρακάτω πίνακα.

| **Πρόγραμμα** | **Έκδοση** | **Αναφορά** |
|---|---|---|
| BLAST | 2.6.11.10 | [21] |
| Modeller | 9.1 | [22] |
| Jmol | 10.2.0 | [23] |
| Autodock<br>ADT (Autodock Tools) | 4.0.1<br>1.4.6 | [24] |

#### 1.5.2.1 BLAST

Το πρόγραμμα BLAST (Basic Local Alignment Search Tool) αποτελεί μια μέθοδο εύρεσης ομοιοτήτων μεταξύ ακολουθιών. Το BLAST δέχεται μια ακολουθία επερώτησης από το χρήστη και την αναζητά έναντι ολόκληρης καθορισμένης βάσης δεδομένων. Η έξοδος του προγράμματος για κάθε ακολουθία της βάσης δεδομένων που βρέθηκε όμοια αποτελεί ένα Seq-align (αντιστοίχιση αλληλουχιών) και όλες μαζί οι αντιστοιχίσεις συνδυάζονται σε ένα Seq-annot (παρουσίαση όλων των αντιστοιχίσεων) [21].

Το πρόγραμμα BLAST χρησιμοποιεί ως είσοδο αρχεία τα οποία έχουν τη μορφή FASTA και τα αρχεία εξόδου είναι της ίδιας μορφής.

#### 1.5.2.2 Modeller

Το πρόγραμμα Modeller είναι ικανό για την επιτέλεση διεργασιών ομόλογης μοντελοποίησης. Το Modeller δεν έχει γραφικό περιβάλλον αλλά λειτουργεί μέσω εντολών από τη γραμμή-εντολών. Ως είσοδο, το πρόγραμμα χρειάζεται δύο αρχεία, ένα αρχείο που να περιέχει τις στοιχίσεις αλληλουχιών και ένα script που να θέτει τις παραμέτρους. Η έξοδος του προγράμματος είναι ένα αρχείο pdb της προβλεπόμενης δομής καθώς και αρχεία κειμένου που παρουσιάζουν τα δεδομένα από τη διεργασία της ομόλογης μοντελοποίησης [22].

Το HMDock προϋποθέτει το αρχείο που περιέχει τις αλληλουχίες να έχει τη μορφή PIR για να λειτουργήσει το πρόγραμμα Modeller.

#### 1.5.2.3 Jmol

Το Jmol είναι είναι ένα πρόγραμμα απεικόνισης μορίων. Είναι γραμμένο σε γλώσσα προγραμματισμού Java και μπορεί να ενσωματωθεί σε ιστοσελίδες, μέσω της εφαρμογής applet που διαθέτει, για την απεικόνιση μορίων με διάφορους τρόπους [23].

Στο HMDock ως είσοδος για το πρόγραμμα Jmol χρησιμοποιούνται μόνο αρχεία pdb.

#### 1.5.2.4 Autodock

Το πρόγραμμα Autodock είναι υπεύθυνο για την επιτέλεση της διεργασίας του docking. Ως είσοδος χρησιμοποιούνται περαιτέρω επεξεργασμένα αρχεία pdb του μορίου-δέκτη και του

μορίου-προσδέτη καθώς και ένας χάρτης πλέγματος και σαν έξοδος παράγεται ένα αρχείο που πιθανολογεί την ύπαρξη ή μη σύζευξης μεταξύ των δύο μορίων [24].

Λεπτομερέστερη ανάλυση της λειτουργίας των προγραμμάτων BLAST, Modeller, και Autodock ακολουθεί στο ΠΑΡΑΡΤΗΜΑ – Α.

# ΚΕΦΑΛΑΙΟ 2 – ΠΑΡΟΥΣΙΑΣΗ ΠΡΟΓΡΑΜΜΑΤΟΣ

Στη δημιουργία ενός υπολογιστικού προγράμματος πρέπει να συνυπολογιστούν διάφοροι παράμετροι ώστε να λειτουργεί αποτελεσματικά το πρόγραμμα αλλά και να είναι φιλικό προς τον χρήστη. Ιδιαίτερα στην δημιουργία ενός διαδικτυακού προγράμματος πρέπει να γίνει μια προσεκτική σχεδίαση των ιστοσελίδων καθώς και των λειτουργιών που αυτές θα επιτελούν, ώστε ο χρήστης να μπορεί εύκολα, απλά και γρήγορα να τις χρησιμοποιεί για να επιτύχει το σκόπο του.

Σε περίπτωση που ο χρήστης είναι γνώστης της προγραμματιστικής γλώσσας που χρησιμοποιήθηκε θα πρέπει να έχει τη δυνατότητα να μπορεί να καταλάβει εύκολα τον κώδικα του προγράμματος ώστε να μπορεί, αν χρειαστεί, να τον τροποποιήσει για να καλύψει τις δικές του ανάγκες. Έτσι θα πρέπει και η συγγραφή του κώδικα να γίνει με συγκεκριμένη δομή ώστε να μην είναι χαοτική και δύσκολη η κατανόηση αυτού.

Επομένως σε αυτό το τμήμα, δίνεται έμφαση στην γενική δομή του προγράμματος, γίνεται επεξήγηση του τρόπου λειτουργίας του μέσω του περιβάλλοντος χρήσης, καθώς και του κώδικα αυτού. Τέλος, γίνεται και μια αναφορά στην υπολογιστική ισχύ που χρειάζεται το HMDock κατά την επιτέλεση των λειτουργιών του.

## 2.1 Γενική δομή

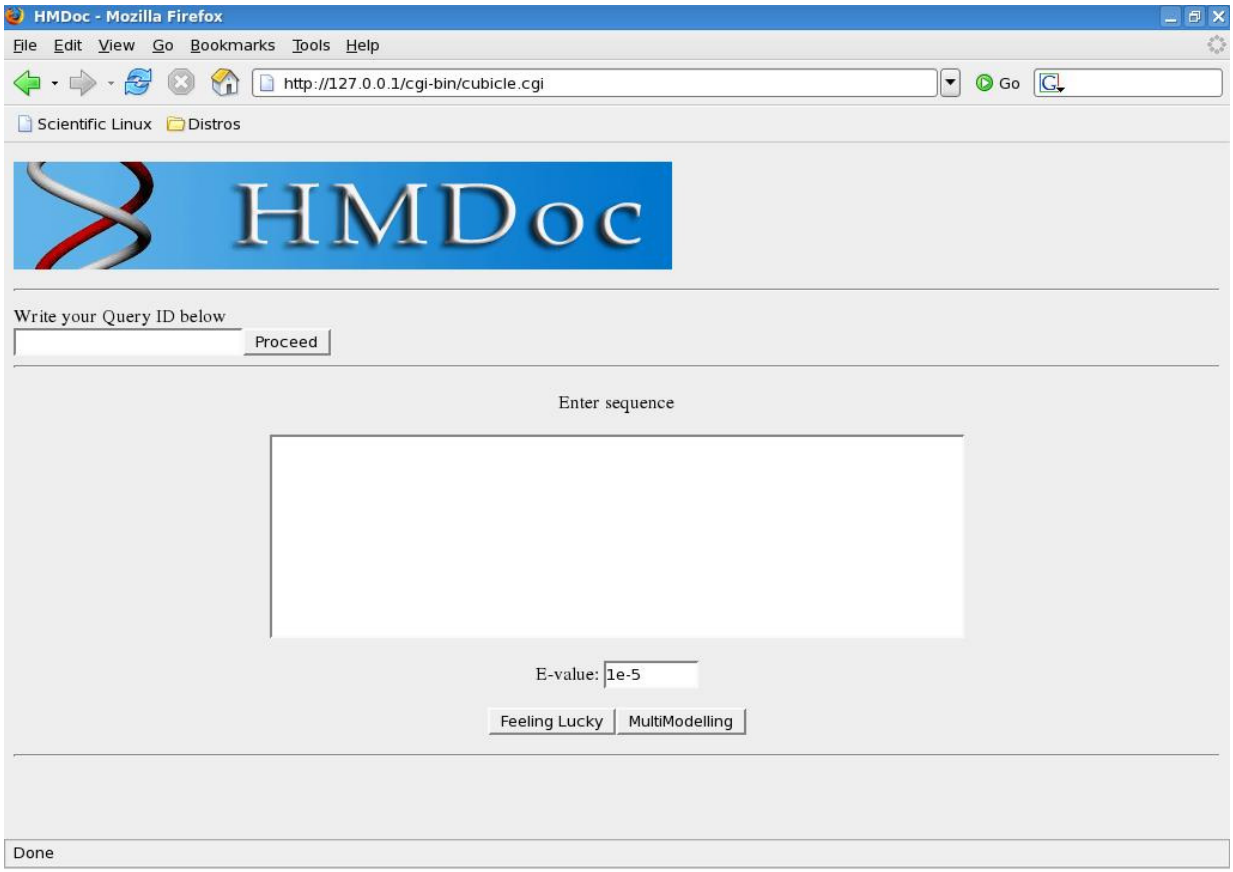
Συνολικά η λειτουργία του προγράμματος μπορεί να συνοψιστεί σε 7 βήματα.

1. Ο χρήστης πληκτρολογεί στον φυλλομετρητή του την διεύθυνση του προγράμματος και μεταφέρεται στην αρχική ιστοσελίδα.
2. Συμπληρώνει τη φόρμα επερώτησης για την ομόλογη μοντελοποίηση πληκτρολογώντας την αμινοξική αλληλουχία και αν το επιθυμεί την τιμή E-value, και την υποβάλλει.
3. Γίνεται η διεργασία της ομόλογης μοντελοποίησης και παράγεται ένας κωδικός που τον πληκτρολογεί ο χρήστης στο κατάλληλο “widget” πεδίου κειμένου ώστε να μεταβεί στην ιστοσελίδα των αποτελεσμάτων της ομόλογης μοντελοποίησης και της φόρμας επερώτησης για το docking.
4. Στην τρέχουσα ιστοσελίδα μπορεί να “κατεβάσει” όλα τα αποτελέσματα της ομόλογης μοντελοποίησης, καθώς και να επεξεργαστεί ορισμένα από αυτά.
5. Αν το επιθυμεί μπορεί να “ανεβάσει” ορισμένα μόρια-προσδέτες ώστε να τα “τρέξει” ενάντια στην τρισδιάστατη δομή που δημιουργήθηκε από την ομόλογη μοντελοποίηση με τη διαδικασία του docking.
6. Μόλις τελειώσει η διαδικασία του docking ο χρήστης έχει τη δυνατότητα να “κατεβάσει” τα αποτελέσματα της διαδικασίας.
7. Μελλοντικά μπορεί να επιστρέψει στην τρέχουσα ιστοσελίδα εάν θυμάται τον κωδικό συμπληρώνοντάς τον στην αρχική ιστοσελίδα.

Στη συνέχεια παρουσιάζεται και επεξηγείται το περιβάλλον χρήσης του προγράμματος καθώς και το διάγραμμα ροής αυτού.

### 2.1.1 Περιβάλλον χρήσης

Η εκκίνηση της λειτουργίας του προγράμματος γίνεται όταν ο χρήστης πληκτρολογήσει τη διεύθυνση του προγράμματος στο φυλλομετρητή του. Τότε μεταφέρεται στην ιστοσελίδα που διαθέτει τη φόρμα επερώτησης για την ομόλογη μοντελοποίηση (Εικ. 2-1).

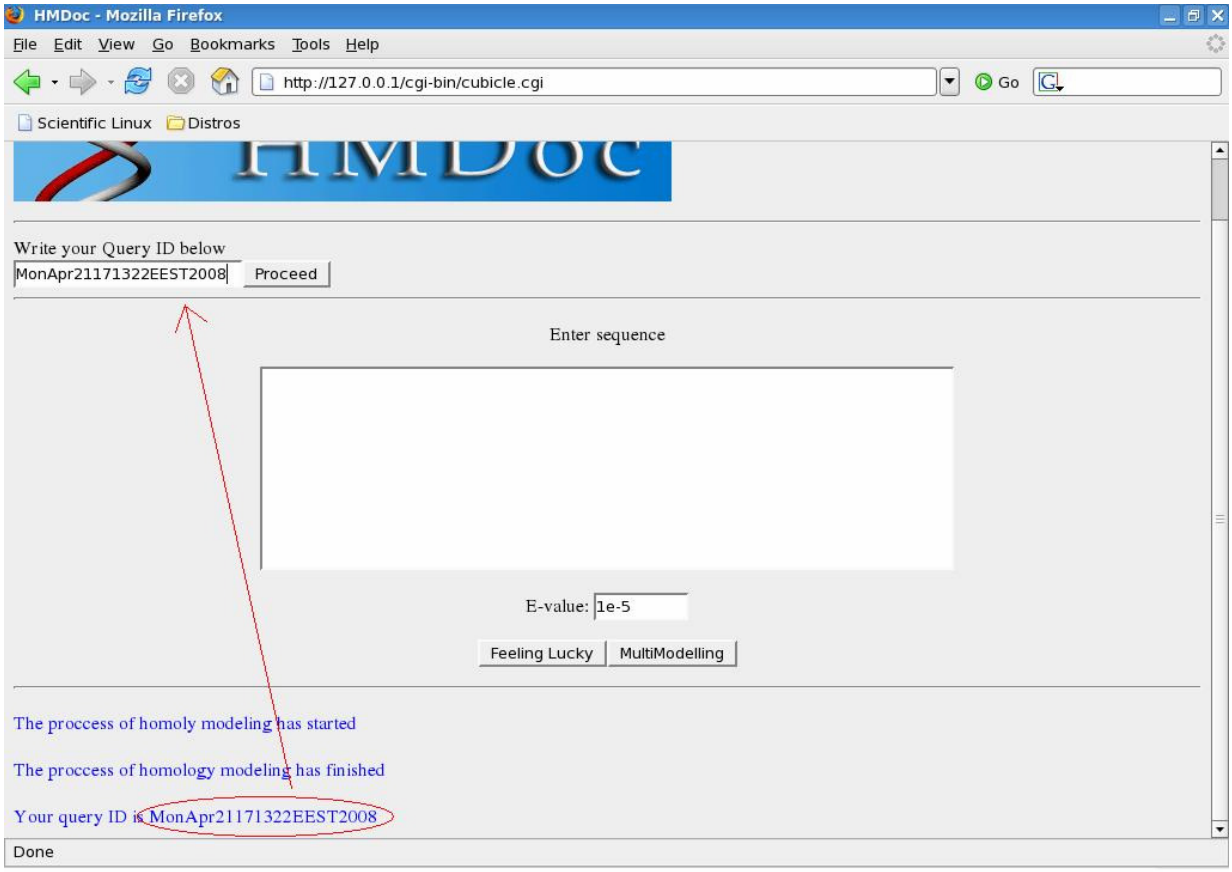



**Εικόνα 2-1.** Αρχική ιστοσελίδα.

Κάτω από το banner βρίσκεται το “widget” πεδίου κειμένου στο οποίο ο χρήστης πληκτρολογεί τον κωδικό που θα εμφανιστεί στην ιστοσελίδα μετά το πέρας της ομόλογης μοντελοποίησης. Χωρισμένη με μια γραμμή βρίσκεται η φόρμα επερώτησης για την ομόλογη μοντελοποίηση που περιλαμβάνει το “widget” πεδίου κειμένου στο οποίο ο χρήστης πληκτρολογεί την αμινοξική αλληλουχία σε συμβολισμό ενός κεφαλαίου γράμματος (εάν η αμινοξική αλληλουχία αποτελείται από δύο πολυπεπτιδικές αλυσίδες τότε ο χρήστης πρέπει να τις διαχωρίσει με το σύμβολο / ), το “widget” πεδίου αριθμού στο οποίο ο χρήστης θέτει την τιμή E-value που είναι το όριο πάνω από το οποίο θα προτιμούνται οι ομόλογες αλληλουχίες, και δύο κουμπιά υποβολής, το Feeling Lucky και το MultiModelling. Εάν ο χρήστης επιλέξει το κουμπί υποβολής Feeling Lucky τότε η ομόλογη μοντελοποίηση επιτυγχάνεται με την πρωτεΐνη με τη μεγαλύτερη ομοιότητα ως υπόστρωμα για το μοντέλο της πρωτεΐνης που έχει εισαχθεί από τον χρήστη, ενώ αν ο χρήστης επιλέξει το MultiModelling τότε το υπόστρωμα για το μοντέλο δημιουργείται από πέντε πρωτεΐνες με τη μεγαλύτερη ομοιότητα. Πρέπει να τονιστεί ότι αν η αμινοξική αλληλουχία του χρήστη αποτελείται από δύο πολυπεπτιδικές αλυσίδες πρέπει να επιλέγεται το κουμπί υποβολής Feeling Lucky διότι πρέπει να χρησιμοποιηθεί ως υπόστρωμα μία πρωτεΐνη που να έχει σε μεγάλο ποσοστό ομοιότητας και τις δύο πολυπεπτιδικές αλυσίδες.

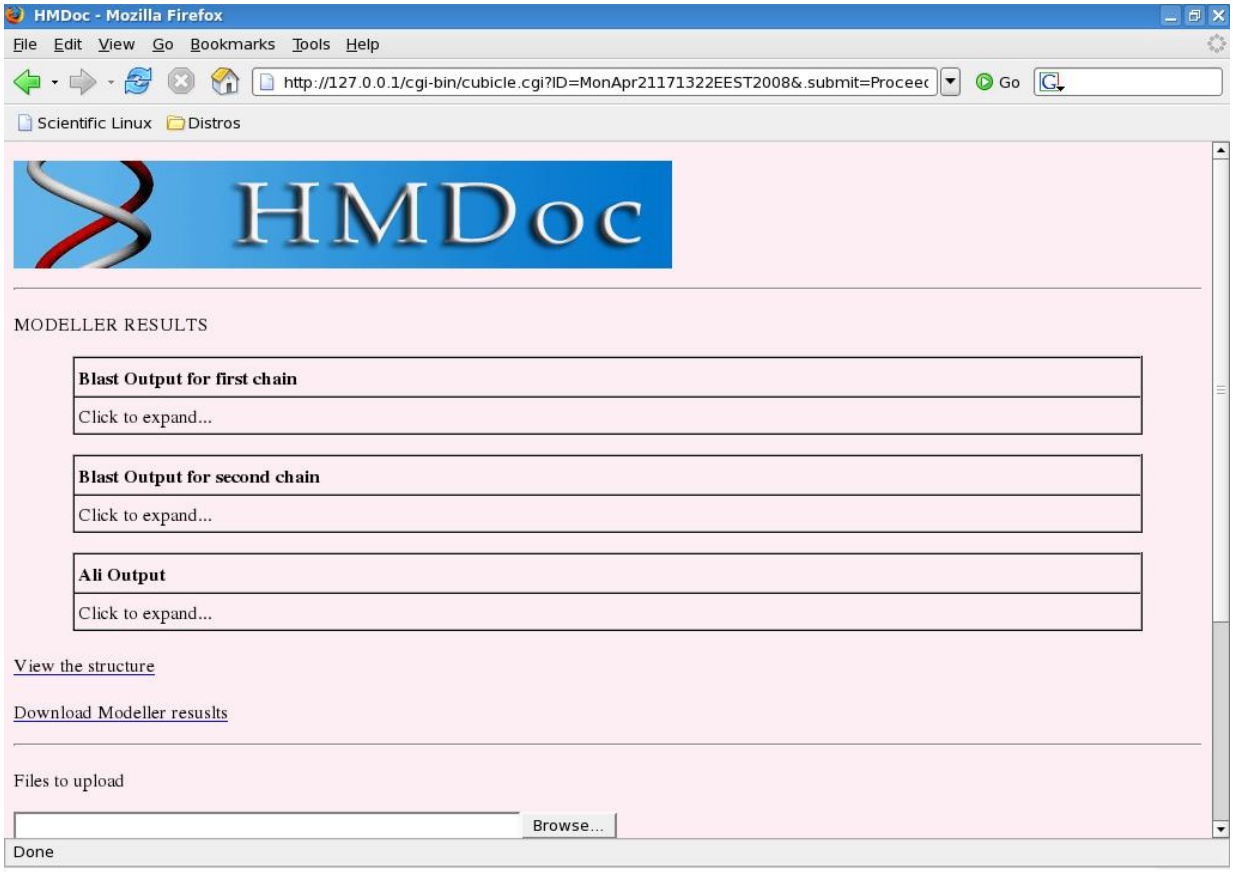
Μόλις ο χρήστης συμπληρώσει και υποβάλλει τη φόρμα επερώτησης τότε γίνονται όλες οι διαδικασίες της ομόλογης μοντελοποίησης και στο τέλος παράγεται ένας κωδικός που είναι διαφορετικός για κάθε φόρμα επερώτησης που επεξεργάζεται από το πρόγραμμα.

**Εικόνα 2-2.** Η ιστοσελίδα μετά την αποπεράτωση της ομόλογης μοντελοποίησης.



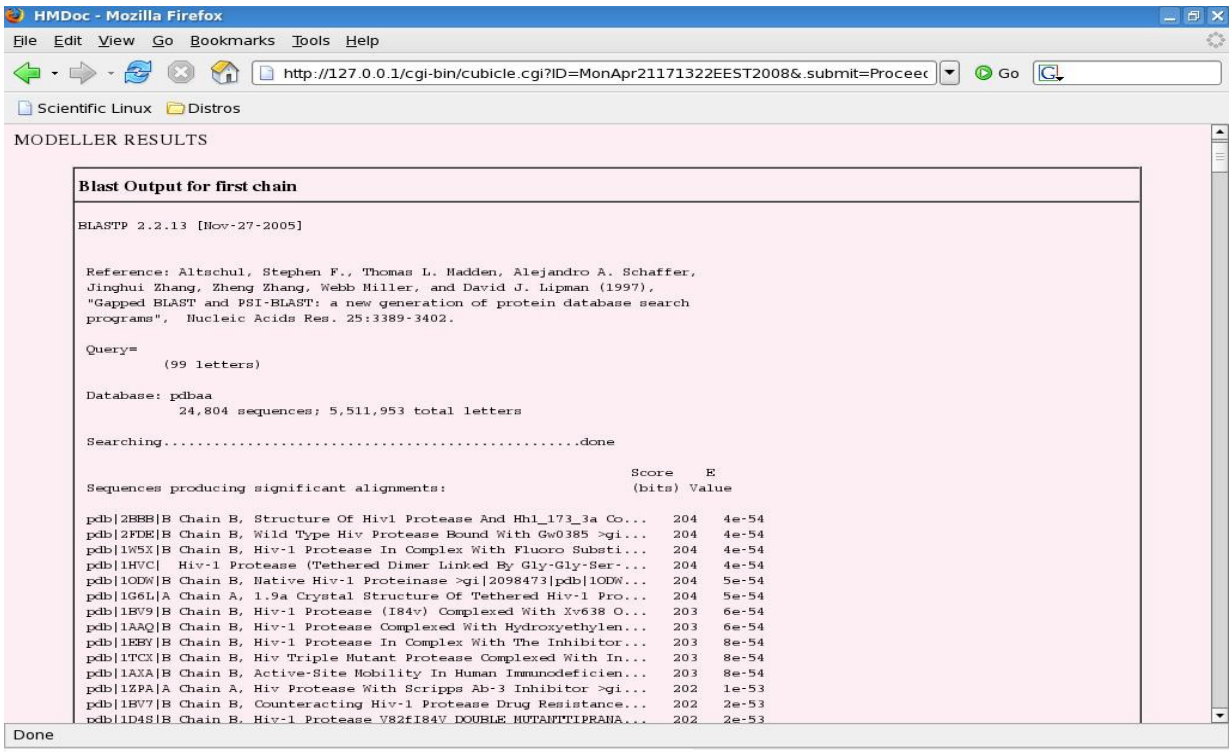

Πληκτρολογώντας αυτόν τον κωδικό στο "widget" πεδίου κειμένου που βρίσκεται κάτω από το banner και επιλέγοντας το κουμπί υποβολής Proceed (Εικ. 2-2), ο φυλλομετρητής παρουσιάζει στον χρήστη την ιστοσελίδα όπου βρίσκονται τα αποτελέσματα της ομόλογης μοντελοποίησης (Εικ. 2-3) καθώς και τη φόρμα επερώτησης για τη διαδικασία του docking.

Αν ο χρήστης κάνει click με το ποντίκι του στο κατάλληλο κουτί μπορούν να προβληθούν στην ιστοσελίδα ορισμένα αποτελέσματα της ομόλογης μοντελοποίησης όπως τα αποτελέσματα του προγράμματος BLAST καθώς και το υπόστρωμα που χρησιμοποιήθηκε από το Modeller (Εικ. 2-4). Ακόμα παρέχεται η δυνατότητα προβολής και περαιτέρω μελέτης του μοντέλου που δημιουργήθηκε μέσω του Jmol, αν ο χρήστης κάνει click στη φράση "View the structure". Τέλος κάνοντας click στη φράση "Download Modeller results" ο χρήστης μπορεί να "κατεβάσει" μέσω ftp όλα τα αποτελέσματα της ομόλογης μοντελοποίησης καθώς και τα αρχεία τα οποία χρησιμοποιήθηκαν για να επιτελεστεί η διαδικασία (Εικ. 2-5).

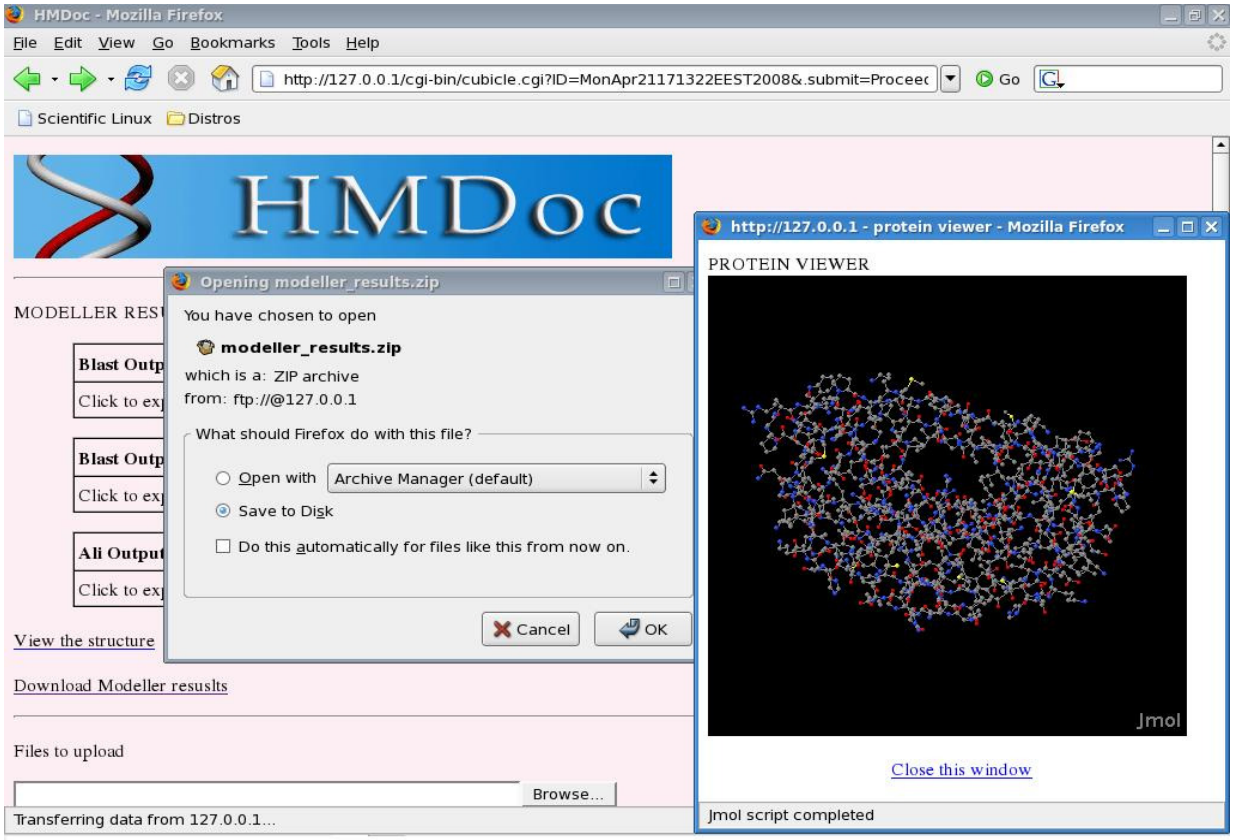



**Εικόνα 2-3.** Η ιστοσελίδα προβολής των αποτελεσμάτων της ομόλογης μοντελοποίησης.

Αφού ο χρήστης επεξεργαστεί τα δεδομένα της ομόλογης μοντελοποίησης, μπορεί να "ανεβάσει" αρχεία pdb που να περιέχουν τα μόρια-προσδέτες από το τμήμα της ιστοσελίδας που βρίσκεται κάτω από τη φράση "Files to upload" για να αρχίσει τη διαδικασία του docking. Αφού "ανεβάσει" όσα αρχεία θέλει, ο χρήστης έχει δύο επιλογές. Στη φόρμα επερώτησης για τη διεργασία του docking, είτε διαλέγει ένα μόριο-προσδέτη και κάνει click στο κουμπί υποβολής "Start Docking" για να επιτελεστεί η διεργασία μεταξύ της πρωτεΐνης που μόλις μοντελοποιήθηκε και του μορίου-προσδέτη που έχει επιλέξει, είτε κάνει click στο κουμπί υποβολής "MultiDock" ώστε να επιτελεστεί η διαδικασία με όλα τα μόρια-προσδέτες που έχει "ανεβάσει", ένα τη φορά. Αφού τελειώσει η διαδικασία του docking δίνεται η δυνατότητα ο χρήστης να κατεβάσει τα αρχεία που δημιουργήθηκαν κατά τη διεργασία μέσω ftp.



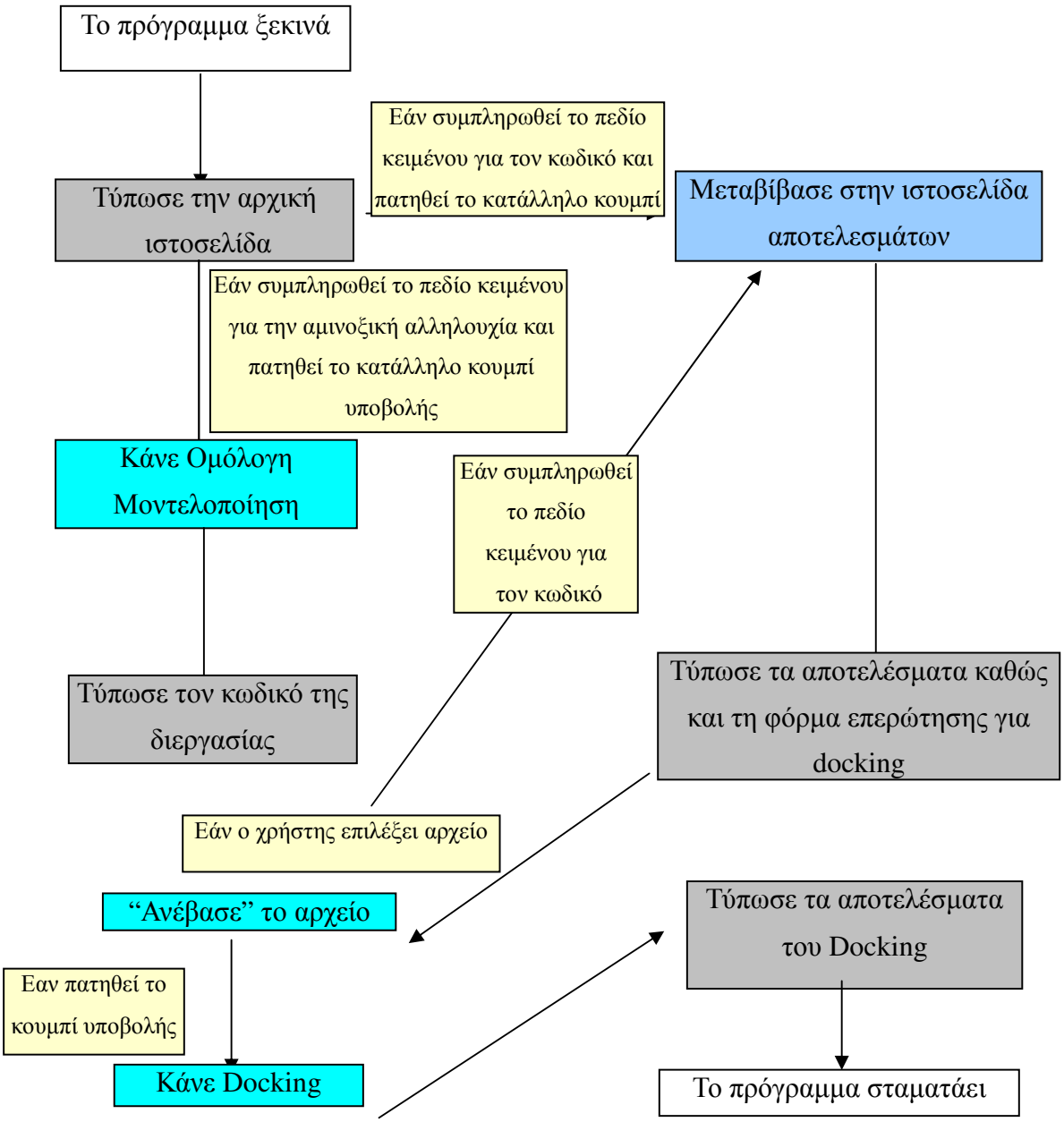

**Εικόνα 2-4.** Σε ανοιγμένη μορφή το κουτί που περιέχει τα αποτελέσματα του BLAST.




**Εικόνα 2-5.** Προβολή του μοντέλου της πρωτεΐνης και δυνατότητα "κατεβάσματος" των αρχείων.

### 2.1.2 Διάγραμμα Ροής

Στη συνέχεια παρουσιάζεται το διάγραμμα ροής του προγράμματος που φαίνεται να έχει μια σχετικά γραμμική ροή.



## 2.2 Επεξήγηση του Κώδικα

Σε αυτήν την ενότητα γίνεται διεξοδική επεξήγηση του κώδικα του προγράμματος βήμα προς βήμα. Γίνεται ανάλυση της χρήσης κάθε υπορουτίνας καθώς και αναφορά σε συγκεκριμένες σημαντικές μεταβλητές. Στο ΠΑΡΑΡΤΗΜΑ – Β παρατίθεται ολόκληρος ο κώδικας του προγράμματος.

### 2.2.1 Κύριο Τμήμα

Ο ρόλος του κώδικα του κυρίως τμήματος είναι να προσθέσει τις λειτουργικές μονάδες, να δηλώσει τις μεταβλητές, και να καθορίζει την ιστοσελίδα που θα δειχθεί στον φυλλομετρητή.

```
#!/usr/bin/perl -w
use strict;
use CGI qw(:standard);
use Net::FTP;
use File::Basename;

my(
   $query_string,
   $trash,
.................
   @count1,
   @jmolscript,
   );

$query_string = $ENV{'QUERY_STRING'};
($trash, $QID, $trash2) = split (/=/, $query_string);
if (defined($QID)) {
  ($QIDtrue, $trash) = split (/&/, $QID);
  }
unless (defined($QIDtrue)) {
  html1();
  }
else {
  if (-e "/tmp/$QIDtrue/") {
  html2();
  }
  else {
  bail2("Wrong ID entered");
  }
}
```

Επεξήγηση του κώδικα του κυρίως τμήματος

Η πρόταση `use` είναι υπεύθυνη για την προσθήκη των λειτουργικών μονάδων ώστε να επιτραπεί η πρόσβαση ορισμάτων και συναρτήσεων από αυτές. Ο τελεστής `my` χρησιμοποιείται για τη δήλωση των μεταβλητών που χρησιμοποιούνται από το πρόγραμμα. Ύστερα γίνεται διάκριση των ιστοσελίδων από το url που έχει πληκτρολογήσει ο χρήστης στον

φυλλομετρητή, καλώντας την υπορουτίνα `html1` για την πρώτη ιστοσελίδα ή την υπορουτίνα `html2` για την δεύτερη ιστοσελίδα. Η πρώτη ιστοσελίδα είναι υπεύθυνη για την υποβολή επερωτήματος για ομόλογη μοντελοποίηση, ενώ η δεύτερη δείχνει τα αποτελέσματα της ομόλογης μοντελοποίησης και αν το επιθυμεί ο χρήστης συμπληρώνεται και η φόρμα για να επιτελεστεί η διαδικασία του docking. Η υπορουτίνα `bail2` τυπώνει ένα σημείωμα λάθους σε περίπτωση που ο χρήστης έχει εισαγάγει λάθος ID.

### 2.2.2 Τμήμα Ομόλογης Μοντελοποίησης

Ο ρόλος του κώδικα της ομόλογης μοντελοποίησης είναι να επεξεργαστεί τα δεδομένα που έχει παραθέσει ο χρήστης μέσω της φόρμας της πρώτης ιστοσελίδας αφού υποβληθεί επερώτηση και να κάνει όλες τις προεργασίες ώστε να επιτελεστεί η διαδικασία της ομόλογης μοντελοποίησης.

```
sub html1 {
  print header, start_html("HMDoc"); print p("<IMG src='../banner.gif'>");
  print hr();
  ID_form();
  sequence_form();
  if (defined($Seq)) {
    if ($Seq ne '') {
      if (param) {
        if ($Begin eq 'Feeling Lucky') {
          $Seq =~ s/ //g;
          if ($Seq =~ /\//) {
            ID();
            Seq_writer("1");
            Seq_checker();
            blast_program("1");
            parse_blast("1");
            get_ftp("1");
            modeller_program_feeling_lucky("1");
            java_files();
            }
          else {
            ID();
            Seq_writer("2");
            Seq_checker();
            blast_program("2");
            parse_blast("2");
            get_ftp("1");
            modeller_program_feeling_lucky("2");
            java_files();
            }
          }
        else {
          ID();
          Seq_writer();
          Seq_checker();
          blast_program();
          parse_blast("3");
          get_ftp("2");
```

```
          modeller_program_multimodelling();
          java_files();
          }
        }
      }
    else {
      bail2 ("No sequence entered");
      }
    print end_html;
    }
  }
```

Επεξήγηση υπορουτίνας `html1`

Οι πρώτες τέσσερις γραμμές τις υπορουτίνας `html1` που αφορούν την μορφή της ιστοσελίδας, με τις υπορουτίνες `ID_form` και `sequence_form` να αφορούν τις φόρμες επερώτησης για την μετάβαση στην ιστοσελίδα που αφορά το docking και τα αποτελέσματα της ομόλογης μοντελοποίησης και για την διαδικασία τις ομόλογης μοντελοποίησης αντίστοιχα, ενώ στις υπόλοιπες γραμμές καλούνται οι υπορουτίνες για την προετοιμασία και επιτέλεση της ομόλογης μοντελοποίησης, εφόσον υποβληθεί η φόρμα επερώτησης. Η τέταρτη πρόταση `if` διακρίνει τις διαδικασίες που θα ακολουθηθούν εάν ο χρήστης υποβάλει την επερώτηση για μοντελοποίηση από μια ομόλογη δομή (`Feeling Lucky`) ή από πολλές ομόλογες δομές (`MultiModeling`). Ενώ η πέμπτη πρόταση `if` διακρίνει τις διαδικασίες που θα ακολουθηθούν εάν η αλληλουχία που εισήχθη από τον χρήστη περιλαμβάνει μία ή δυο πολυπεπτιδικές αλυσίδες. Η υπορουτίνα `bail2` τυπώνει ένα μύνημα λάθους σε περίπτωση που ο χρήστης δεν έχει εισαγάγει αλληλουχία.

### 2.2.2.1 Υπορουτίνες για τις φόρμες επερώτησης της `html1`

2.2.2.1.1 Υπορουτίνα `ID_form`

```
sub ID_form {
   $cur2 = new CGI;
   print "Write your Query ID below";
   print $cur2->start_form(-method=>'GET',
                     -action=>$action,
                     -enctype=>$encoding);
   print $cur2->textfield(-name=>'ID',
                    -size=>21,);
   print $cur2->submit(-value=>'Proceed');
   print $cur2->endform;
   print hr();
  }
```

Επεξήγηση υπορουτίνας `ID_form`

Όταν ολοκληρωθεί η ομόλογη μοντελοποίηση παράγεται ένας αριθμός που είναι διαφορετικός για κάθε φόρμα επερώτησης που έχει επεξεργαστεί. Πληκτρολογώντας τον αριθμό αυτό στην φόρμα επερώτησης που παράγει η υπορουτίνα `ID_form`, ο χρήστης μεταφέρεται στην ιστοσελίδα που αφορά το docking και τα αποτελέσματα της ομόλογης μοντελοποίησης.

2.2.2.1.2 Υπορουτίνα `sequence_form`

```
sub sequence_form {
  $cur1 = new CGI;
  print $cur1->start_form(-method=>'POST',
                          -action=>$action,
                          -enctype=>$encoding);
  print p("<BODY bgcolor=#EEEEEE>");
  print p("<center>Enter sequence");
  print p();
  print  $cur1->textarea(-name=>'Seq',
                         -default=>'',
                         -rows=>10,
                         -columns=>80,
                         -override=>1,);
  print p();
  print (" E-value: ");
  print $cur1->textfield(-name=>'Evalue',
                         -default=>'1e-5',
                         -size=>7,);
  print p();
  print $cur1->submit(-name=>'begin',
                      -value=>'Feeling Lucky');
  print $cur1->submit(-name=>'begin',
                      -value=>'MultiModelling');
  print p("</center>");
  print p("<font color=blue>");
  print $cur1->endform;
  print hr();
  $Seq=(param('Seq'));
  if (defined($Seq)) {
    $Seq =~ s/ //g;
    }
  $Begin=(param('begin'));
  $Evalue=(param('Evalue'));
  }
```

Επεξήγηση υπορουτίνας `sequence_form`

Στην φόρμα επερώτησης της υπορουτίνας `sequence_form` ο χρήστης συμπληρώνει το κεντρικό “widget” πεδίου κειμένου με την αμινοξική αλληλουχία προς μοντελοποίηση. Αν ο χρήστης επιλέξει το κουμπί υποβολής `MultiModeling` μπορεί να θέσει αν επιθυμεί και την τιμή E-value που προσδιορίζει το κατώτερο όριο που θα γίνονται δεκτές οι ομόλογες αλληλουχίες που θα χρησιμοποιηθούν ως υπόστρωμα για την ομόλογη μοντελοποίηση. Η αμινοξική αλληλουχία ανατίθεται στη μεταβλητή `$Seq` και η τιμή της E-value ανατίθεται στη μεταβλητή `$Evalue`.

### 2.2.2.2 Υπορουτίνες Ομόλογης Μοντελοποίησης

Λόγω της ύπαρξης δύο διεργασιών για την επιτέλεση της ομόλογης μοντελοποίησης, μέσω μιας ομόλογης δομής ή μέσω πολλών ομόλογων δομών, ορισμένες υπορουτίνες

διαχωρίζονται ανάλογα με το κουμπί υποβολής που χρησιμοποίησε ο χρήστης. Η διαχώριση αυτή επιτυγχάνεται με το κάλεσμα μιας υπορουτίνας μαζί με τον αριθμό 1 για ομόλογη μοντελοποίηση μέσω μιας δομής, ή με τον αριθμό 2 για ομόλογη μοντελοποίηση μέσω πολλών δομών.

#### 2.2.2.2.1 Υπορουτίνα `ID`

```
sub ID {
  system ("date > /tmp/DATE.txt");
  open DATE , "/tmp/DATE.txt" || bail("$!");
  @date = <DATE>;
  close DATE;
  for ( $i=0; $i <@date; $i++) {
    $datenew .= $date[$i];
    }
  $datenew =~ s/ //g;
  $datenew =~ s/://g;
  chop ($datenew);
  mkdir("/var/ftp/pub/$datenew/",0777);
  mkdir("/var/ftp/pub/$datenew/ligands/",0777);
  mkdir("/tmp/$datenew/",0777);
  }
```

Επεξήγηση υπορουτίνας `ID`

Μέσω της υπορουτίνας `ID` παράγεται ο αριθμός που είναι διαφορετικός για κάθε φόρμα επερώτησης και ανατίθεται στη μεταβλητή `$datenew`. Η υπορουτίνα `bail` ανακοινώνει στον χρήστη αν υπάρχουν λειτουργικά προβλήματα. Επίσης δημιουργούνται τρεις κατάλογοι με την εντολή `mkdir` που είναι διαφορετικοί για κάθε φόρμα επερώτησης που επεξεργάζεται, με τον πρώτο κατάλογο να μπορεί να περιέχει τα αρχεία τα οποία θα μπορεί να κατεβάζει ο χρήστης, με τον δεύτερο κατάλογο να μπορεί να περιέχει τα ligands που θα ανεβάζει ο χρήστης, και με τον τρίτο κατάλογο μέσα στον οποίο θα δημιουργούνται τα αρχεία υπεύθυνα για την ομόλογη μοντελοποίηση και το docking καθώς και τα αποτελέσματα αυτών των διεργασιών.

#### 2.2.2.2.2 Υπορουτίνα `Seq_writer`

```
sub Seq_writer {
  open PIRPROTEIN, "> /tmp/$datenew/UnkP.ali" || bail("$!");
  print PIRPROTEIN "\>P1;UnkP\n";
  print PIRPROTEIN "sequence:UnkP:::::::0.00: 0.00\n";
  print PIRPROTEIN "$Seq";
  print PIRPROTEIN "\*";
  close PIRPROTEIN;
  if ($_[0]=='1') {
    ($Seq1, $Seq2) = split (/\//, $Seq);
    open PROTEINONE, "> /tmp/$datenew/protein1.txt" || bail("$!");
    print PROTEINONE "$Seq1";
    close PROTEINONE;
    open PROTEINTWO, "> /tmp/$datenew/protein2.txt" || bail("$!");
    print PROTEINTWO "$Seq2";
    close PROTEINTWO;
```

```
    $Seq =~ s/\///g;
    open PROTEIN, "> /tmp/$datenew/protein.txt" || bail("$!");
    print PROTEIN "$Seq";
    close PROTEIN;
    }
  else {
    open PROTEIN, "> /tmp/$datenew/protein.txt" || bail("$!");
    print PROTEIN "$Seq";
    close PROTEIN;
    }
  }
```

Επεξήγηση υπορουτίνας `Seq_writer`

Μέσω της υπορουτίνας `Seq_writer` δημιουργούνται δύο αρχεία εάν ο χρήστης έχει υποβάλει μια αμινοξική αλληλουχία που κωδικοποιεί μια πεπτιδική αλυσίδα, το αρχείο `protein.txt` το οποίο περιέχει την αμινοξική αλληλουχία σε μορφή FASTA (για να χρησιμοποιηθεί από το BLAST), και το αρχείο `UnkP.ali` το οποίο την περιέχει σε μορφή PIR (για να χρησιμοποιηθεί από το Modeller). Εάν ο χρήστης έχει υποβάλει μια αμινοξική αλληλουχία που περιέχει δύο πολυπεπτιδικές αλυσίδες τότε δημιουργούνται τέσσερα αρχεία, τα αρχεία `protein1.txt` και `protein2.txt` που περιέχουν ξεχωριστά τις αλληλουχίες που κωδικοποιούν κάθε αλυσίδα, το `protein.txt` που περιέχει ολόκληρη την αμινοξική αλληλουχία, αρχεία σε μορφή FASTA, και το αρχείο `UnkP.ali` που περιέχει ολόκληρη την αμινοξική αλληλουχία σε μορφή PIR.

2.2.2.2.3 Υπορουτίνα `Seq_checker`

```
sub Seq_checker {
  @true = qw(A C D E F G H I K L M N P Q R S T V W Y /);
  open PROTEIN, "/tmp/protein.txt"
  or die ("Filehandle problems");
  @check = <PROTEIN>;
  close PROTEIN;
  $check = join( '', @check);
  $check =~ s/\s//g;
  $pcheck = $check;
  @check = split( '', $check);
  @pcheck = split( '', $pcheck);
  $hmm='';
  for ( $dt=0; $dt<@check; $dt+=5) {
    $hmm = "$hmm" . " ";
    for ( $t=$dt; $t < $dt+5  && $t < @check; $t++) {
      $hmm= "$hmm" . "$pcheck[$t]";
      }
    }
    for ( $d=0; $d<@check; $d++ ) {
    if ( $check[$d] ne $true[0]  && $check[$d] ne $true[1]  && $check[$d] ne
$true[2]  && $check[$d] ne $true[3] && $check[$d] ne $true[4]
      && $check[$d] ne $true[5]  && $check[$d] ne $true[6]  && $check[$d] ne
$true[7] && $check[$d] ne $true[8]  && $check[$d] ne $true[9]
      && $check[$d] ne $true[10] && $check[$d] ne $true[11] && $check[$d] ne
$true[12] && $check[$d] ne $true[13] && $check[$d] ne $true[14]
```

```
      && $check[$d] ne $true[15] && $check[$d] ne $true[16] && $check[$d] ne
$true[17] && $check[$d] ne $true[18] && $check[$d] ne $true[19]) {
      $dt=$d+1;
        print "Sorry the character $check[$d] at place number $dt of the
query you have entered, shown below, is invalid\n";
        print p();
        print "$hmm";
        die ;
      }
    }
  }
```

Επεξήγηση υπορουτίνας `Seq_checker`

Επειδή η αλληλουχία του χρήστη πρέπει να περιέχει μόνο γράμματα που να αντιστοιχούν σε αμινοξέα, ελέγχουμε με την υπορουτίνα `Seq_checker` για τυχόν στοιχεία της αλληλουχίας που να είναι λανθασμένα. Αν διαπιστωθεί κάτι τέτοιο το πρόγραμμα σταματάει και υποδεικνύει στον χρήστη που έχει γίνει το λάθος.

2.2.2.2.4 Υπορουτίνα `blast_program`

```
sub blast_program {
  if ($_[0]=='1') {
    system("/usr/local/blast/bin/blastall -p blastp   -d
/usr/local/blast/data/pdbaa -i /tmp/$datenew/protein1.txt -o
/tmp/$datenew/blast1.out");
    system("/usr/local/blast/bin/blastall -p blastp -d
/usr/local/blast/data/pdbaa -i /tmp/$datenew/protein2.txt -o
/tmp/$datenew/blast2.out");
  }
  else {
    system("/usr/local/blast/bin/blastall -p blastp -d
/usr/local/blast/data/pdbaa -i /tmp/$datenew/protein.txt -o
/tmp/$datenew/blast.out");
    }
  }
```

Επεξήγηση υπορουτίνας `blast_program`

Με την υπορουτίνα `blast_program` καλούμε το πρόγραμμα BLAST να βρει συγγενείς αλληλουχίες της πρωτεΐνης που βρίσκεται στο `protein.txt`, από τη βάση δεδομένων `pdbaa`, και να καταγράψει τα αποτελέσματα στο αρχείο `blast.out`. Εάν ο χρήστης έχει υποβάλει αλληλουχία με δύο πολυπεπτιδικές αλυσίδες, τότε κάθε αμινοξική ακολουθία που κωδικοποιεί μια αλυσίδα εισάγεται στο πρόγραμμα BLAST ώστε να βρεθούν ομόλογες αλληλουχίες και τα αποτελέσματα καταγράφονται στα αρχεία `blast1.out` και `blast2.out`.

2.2.2.2.5 Υπορουτίνα `parse_blast`

```
sub parse_blast {
  if ($_[0]=='1') {
    open BLASTFILE, "/tmp/$datenew/blast1.out" || bail("$!");
```

```
    while (<BLASTFILE>) {
      chomp;
      if (/>pdb/) {
        @line=split(/ /,$_);
        $queryline=$line[0];
      ( $trash, $nquery, $segment) = split (/\|/, $queryline);
        last;
        }
      }
    close BLASTFILE;
    $testb=0;
    open BLASTFILETWO, "/tmp/$datenew/blast2.out" || bail("$!");
    while (<BLASTFILETWO>) {
      if (/$nquery/) {
        $testb=1;
      }
    }
    unless ($testb == '1') {
      bail2("Homology modelling cant be completed, not enough homologous
proteins");
      }
    close BLASTFILETWO;
    $nquery="\L$nquery";
    $query=$nquery.$segment;
    }
  elsif ($_[0]=='2') {
    open BLASTFILE, "/tmp/$datenew/blast.out" || bail("$!");
    while (<BLASTFILE>) {
      chomp;
      if (/>pdb/) {
        @line=split(/ /,$_);
        $queryline=$line[0];
      ( $trash, $nquery, $segment) = split (/\|/, $queryline);
        last;
        }
      }
    $nquery="\L$nquery";
    $query=$nquery.$segment;
    close BLASTFILE;
    }
  elsif ($_[0]=='3') {
    $k = 0;
    open BLASTFILE, "/tmp/$datenew/blast.out" || bail("$!");
    while (<BLASTFILE>) {
      chomp;
      if (/pdb\|/) {
        @line=split(/ /,$_);
        $line_number = @line;
        $line_number = $line_number - 1;
        $multiqueryline[$k]=$line[0];
        if ( $line[0] =~ /^>pdb/) {
          last;
          }
        else {
```

```
          ($trash, $nmultiquery[$k], $multisegment[$k]) = split (/\|/,
$multiqueryline[$k]);
           $multievalue[$k] = $line[$line_number];
           $k++;
           }
         }
       }
     for ( $k =0; $k < @nmultiquery; $k++) {
       $nmultiquery[$k] = "\L$nmultiquery[$k]";
       $multiquery[$k] = $nmultiquery[$k].$multisegment[$k];
       }
     close BLASTFILE;
     }
  }
```

Επεξήγηση υπορουτίνας `parse_blast`

Μέσω της υπορουτίνας `parse_blast` αναζητούνται από το αρχείο `blast.out` η αλληλουχία με το μεγαλύτερο ποσοστό ομοιότητας εάν ο χρήστης έχει επιλέξει το κουμπί υποβολής `FeelingLucky` ή όλες οι αλληλουχίες που βρέθηκαν να έχουν ένα default ποσοστό ομοιότητας εάν έχει επιλέξει το κουμπί υποβολής `MultiModeling`. Αν έχει επιλεχθεί το κουμπί `FeelingLucky` διακρίνονται δύο περιπτώσεις ανάλογα με το αν η αλληλουχία περιέχει δύο ή μία πολυπεπτιδικές αλυσίδες. Στην πρώτη περίπτωση, ελέγχονται τα αρχεία `blast1.out` και `blast2.out` για την εύρεση της πρωτεΐνης που περιέχει σε μεγαλύτερη ομοιότητα και τις δύο αλυσίδες και το όνομα αυτής ανατίθεται στην μεταβλητή `$nquery`, ενώ στη δεύτερη περίπτωση ελέγχεται το αρχείο `blast.out` για την εύρεση της πρωτεΐνης με τη μεγαλύτερη ομοιότητα και το όνομα αυτής ανατίθεται στη μεταβλητή `$nquery`, ο χαρακτηρισμός της πολυπεπτιδικής αλυσίδας ανατίθεται στη μεταβλητή `$segment`, ενώ η μεταβλητή `$query` περιέχει τις δύο παραπάνω μεταβλητές ενωμένες. Αν έχει επιλεγεί το κουμπί `MultiModeling` ελέγχεται το αρχείο `blast.out` για την εύρεση όλων των πρωτεϊνών που έχουν ποσοστό ομοιότητα. Το όνομα κάθε πρωτεΐνης αποτελεί ένα στοιχείο του πίνακα `@nmultiquery`, ο χαρακτηρισμός της πολυπεπτιδικής αλυσίδας καθεμιάς πρωτεΐνης αποτελεί ένα στοιχείο του πίνακα `@multisegment`, ενώ καθένα από τα παραπάνω στοιχεία ενωμένα αποτελούν ένα στοιχείο του πίνακα `@multiquery`. Η τιμή E-value κάθε πρωτεΐνης αποτελεί ένα στοιχείο του πίνακα `@multievalue`.

#### 2.2.2.2.6 Υπορουτίνα `get_ftp`

```
sub get_ftp {
  if ($_[0]=='1') {
    unless (-e "/tmp/pdb$nquery\.ent") {
      $ftp = Net::FTP->new("ftp.rcsb.org", Debug => 0)
        || bail("Cannot connect to ftp.rcsb.org: $@");
      $ftp->login("anonymous",'-anonymous@')
        || bail("Cannot login ", $ftp->message);
      $ftp->cwd("/pub/pdb/data/structures/all/pdb/")
        || bail("Cannot change working directory ", $ftp->message);
      $ftp->get("pdb$nquery\.ent.Z","/tmp/pdb$nquery\.ent.Z")
        || bail("get failed ", $ftp->message);
      $ftp->quit;
      system("gunzip /tmp/pdb$nquery\.ent.Z");
```

```
      }
    system ("cp /tmp/pdb$nquery\.ent /tmp/$datenew/pdb$nquery\.ent");
    print p("The proccess of homology modeling has started");
    }
  elsif ($_[0]=='2') {
    $ftp_number = 0;
    for ($i=0; $i<@multievalue; $i++) {
      if ( $multievalue[$i] < $Evalue && $multievalue[$i] ne "") {
        $ftp_number++;
        }
      }
    if ( $ftp_number > 5 ) {
      $ftp_number = 5;
      }
    for ($i=0; $i<$ftp_number; $i++) {
      unless (-e "/tmp/pdb$nmultiquery[$i]\.ent") {
        $ftp = Net::FTP->new("ftp.rcsb.org", Debug => 0)
          || bail("Cannot connect to ftp.rcsb.org: $@");
        $ftp->login("anonymous",'-anonymous@')
          || bail("Cannot login ", $ftp->message);
        $ftp->cwd("/pub/pdb/data/structures/all/pdb/")
          || bail("Cannot change working directory ", $ftp->message);
        $ftp-
>get("pdb$nmultiquery[$i]\.ent.Z","/tmp/pdb$nmultiquery[$i]\.ent.Z")
          || bail("get failed ", $ftp->message);
        $ftp->quit;
        }
      system("gunzip /tmp/pdb$nmultiquery[$i]\.ent.Z");
      system ("cp /tmp/pdb$nmultiquery[$i]\.ent
/tmp/$datenew/pdb$nmultiquery[$i]\.ent");
      }
    print p("The proccess of homology modeling has started");
    }
  }
```

Επεξήγηση υπορουτίνας `get_ftp`

Με αυτή την υπορουτίνα το πρόγραμμα αναζητά τα δεδομένα στην PDB βάση δεδομένων για τις πρωτεΐνες που βρέθηκαν να έχουν ομολογία στην υπορουτίνα `parse_blast` και κατεβάζει τα αντίστοιχα αρχεία στον φάκελο `/tmp/`. Στην περίπτωση που αναζητείται μια ομόλογη πρωτεΐνη κατεβαίνει το αρχείο της πρωτεΐνης που βρίσκεται στη μεταβλητή `$nquery`, ενώ στην περίπτωση που γίνεται multiple modeling κάθε πρωτεΐνη που αποτελεί στοιχείο του πίνακα `@nmultiquery` ελέγχεται αν τηρεί το κριτήρια του E-value που έχει θέσει ο χρήστης κατά τη συμπλήρωση της φόρμας επερώτησης. Αν διαπιστωθεί ότι πάνω από πέντε πρωτεΐνες τηρούν το κριτήριο του E-value, τότε το πρόγραμμα κατεβάζει τα αρχεία των πέντε πρωτεϊνών με τη μεγαλύτερη ομολογία. Αυτό συμβαίνει διότι διαπιστώθηκε ότι παραπάνω από πέντε ομόλογες πρωτεΐνες αυξάνουν κατά πολύ την υπολογιστική ισχύ που χρειάζεται δυσανάλογα με την καλυτέρευση της απόδοσης του ομόλογου μοντέλου ( Η διαπίστωση αυτή προήλθε από την παρατήρηση του χρόνου που χρειάζεται το πρόγραμμα με την οπτική απόδοση του μοντέλου ). Τέλος κάθε αρχείο πρωτεΐνης που κατέβηκε μεταφέρεται στον κατάλογο που έχει οριστεί από την υπορουτίνα `ID` που είναι διαφορετικός για κάθε φόρμα επερώτησης που θα επεξεργαστεί.

2.2.2.2.7 Υπορουτίνα `modeller_program_feeling_lucky`

```
sub modeller_program_feeling_lucky {
  $sequence='UnkP';
  open  BUILD_PROFILE , ">/tmp/$datenew/build_profile\.py" || bail("$!");
  print BUILD_PROFILE "from modeller import *\n";
  print BUILD_PROFILE "log.verbose()\n";
  print BUILD_PROFILE "env = environ()\n";
  print BUILD_PROFILE "sdb = sequence_db(env)\n";
  print BUILD_PROFILE
"sdb.read(seq_database_file='/usr/local/blast/data/pdbaa.bin',
seq_database_format='BINARY',\n";
  print BUILD_PROFILE "chains_list='ALL')\n";
  print BUILD_PROFILE "aln = alignment(env)\n";
  print BUILD_PROFILE "aln.append(file='/tmp/$datenew/$sequence\.ali',
alignment_format='PIR', align_codes='ALL')\n";
  print BUILD_PROFILE "prf = aln.to_profile()\n";
  print BUILD_PROFILE "prf.build(sdb, matrix_offset=-450,
rr_file='\${LIB}/blosum62.sim.mat',\n";
  print BUILD_PROFILE "gap_penalties_1d=(-500, -50),
n_prof_iterations=1,\n";
  print BUILD_PROFILE "check_profile=False, max_aln_evalue=0.01)\n";
  print BUILD_PROFILE "prf.write(file='/tmp/$datenew/build_profile.prf')\n";
  print BUILD_PROFILE "aln = prf.to_alignment()\n";
  print BUILD_PROFILE "aln.write(file='/tmp/$datenew/build_profile.ali',
alignment_format='PIR')\n";
  print BUILD_PROFILE "\n";
  close BUILD_PROFILE;
  system("/usr/bin/mod9v1 /tmp/$datenew/build_profile.py");

  open  ALIGNTWOD , ">/tmp/$datenew/align2d\.py" || bail("$!");
  print ALIGNTWOD "from modeller import *\n";
  print ALIGNTWOD "env = environ()\n";
  print ALIGNTWOD "aln = alignment(env)\n";
  if ($_[0]=='1') {
    print ALIGNTWOD "mdl = model(env, file='/tmp/$datenew/pdb$nquery\.ent',
model_segment=('FIRST:A','LAST:B'))\n";
    print ALIGNTWOD "aln.append_model(mdl, align_codes='$nquery',
atom_files='/tmp/$datenew/pdb$nquery\.ent')\n";
    print ALIGNTWOD "aln.append(file='/tmp/$datenew/$sequence\.ali',
align_codes='$sequence')\n";
    print ALIGNTWOD "aln.align2d()\n";
    print ALIGNTWOD "aln.write(file='/tmp/$datenew/$sequence\-$nquery\.ali',
alignment_format='PIR')\n";
    print ALIGNTWOD "aln.write(file='/tmp/$datenew/$sequence\-$nquery\.pap',
alignment_format='PAP')\n";
    }
  elsif ($_[0]=='2') {
    print ALIGNTWOD "mdl = model(env, file='/tmp/$datenew/pdb$nquery\.ent',
model_segment=('FIRST:$segment','LAST:$segment'))\n";
    print ALIGNTWOD "aln.append_model(mdl, align_codes='$query',
atom_files='/tmp/$datenew/pdb$nquery\.ent')\n";
    print ALIGNTWOD "aln.append(file='/tmp/$datenew/$sequence\.ali',
align_codes='$sequence')\n";
    print ALIGNTWOD "aln.align2d()\n";
```

```
    print ALIGNTWOD "aln.write(file='/tmp/$datenew/$sequence\-$query\.ali',
alignment_format='PIR')\n";
    print ALIGNTWOD "aln.write(file='/tmp/$datenew/$sequence\-$query\.pap',
alignment_format='PAP')\n";
    }
  print ALIGNTWOD "\n";
  close ALIGNTWOD;
  system("/usr/bin/mod9v1 /tmp/$datenew/align2d.py");

  open  MODELSINGLE , ">/tmp/$datenew/model-single\.py" || bail("$!");
  print MODELSINGLE "from modeller import *\n";
  print MODELSINGLE "from modeller.automodel import *\n";
  print MODELSINGLE "env = environ()\n";
  if ($_[0]=='1') {
    print MODELSINGLE "a = automodel(env, alnfile='/tmp/$datenew/$sequence\-
$nquery\.ali',\n";
    print MODELSINGLE "knowns='$nquery\', sequence='$sequence')\n";
    }
  elsif ($_[0]=='2') {
    print MODELSINGLE "a = automodel(env, alnfile='/tmp/$datenew/$sequence\-
$query\.ali',\n";
    print MODELSINGLE "knowns='$query\', sequence='$sequence')\n";
    }
  print MODELSINGLE "a.starting_model = 1\n";
  print MODELSINGLE "a.ending_model = 1\n";
  print MODELSINGLE "a.make()\n";
  print MODELSINGLE "\n";
  close MODELSINGLE;
  system("(cd /tmp/$datenew/ ; /usr/bin/mod9v1 /tmp/$datenew/model-
single.py)");

  print p("The proccess of homology modeling has finished");
  system ("mv /tmp/$datenew/UnkP.B99990001.pdb /tmp/$datenew/UnkP.pdb");
  print " Your query ID is $datenew";
  }
```

Επεξήγηση υπορουτίνας `modeller_program_feeling_lucky`

Με την υπορουτίνα `modeller_program_feeling_lucky` δημιουργούνται και τρέχουν τα python scripts που χρειάζεται το Modeller για να επιτελεστεί η ομόλογη μοντελοποίηση μέσω της πρωτεΐνης με το μεγαλύτερο ποσοστό ομοιότητας. Το script `build_profile.py` είναι υπεύθυνο για την δημιουργία του περιβάλλοντος που θα επιτελεσθεί η ομόλογη μοντελοποίηση, το script `align2d.py` είναι υπεύθυνο για την στοίχιση της αμινοξικής ακολουθίας του χρήστη με την ακολουθία που έχει τη μεγαλύτερη ομοιότητα ώστε να χρησιμοποιηθεί ως υπόστρωμα για το script `model-single.py` που κατασκευάζει το τελικό ομόλογο μοντέλο. Γίνεται μια διαφοροποίηση της σύνθεσης των python scripts ανάλογα με το αν γίνεται ομόλογη μοντελοποίηση μίας ή δύο πολυπεπτιδικών αλυσίδων. Μόλις ολοκληρωθεί η ομόλογη μοντελοποίηση τυπώνεται στον φυλλομετρητή το ID.

2.2.2.2.8 Υπορουτίνα `modeller_program_multimodelling`

```
sub modeller_program_multimodelling {
```

```
  $sequence='UnkP';
  open  SALIGN , ">/tmp/$datenew/salign.py" || bail("$!");
  print SALIGN "from modeller import *\n";
  print SALIGN "log.verbose()\n";
  print SALIGN "env = environ()\n";
  print SALIGN "env.io.atom_files_directory = './:../atom_files/'\n";
  print SALIGN "aln = alignment(env)\n";
  print SALIGN "for (code, chain) in (";
  for ( $i=0 ; $i<$ftp_number ; $i++) {
    if ( $i == $ftp_number-1) {
      print SALIGN "('$nmultiquery[$i]','$multisegment[$i]')";
      }
    else {
      print SALIGN "('$nmultiquery[$i]', '$multisegment[$i]'),";
      }
    }
  print SALIGN "):\n";
  print SALIGN "  mdl = model(env, file=code, model_segment=('FIRST:'+chain,
'LAST:'+chain))\n";
  print SALIGN "  aln.append_model(mdl, atom_files=code,
align_codes=code+chain)\n";
  print SALIGN "for (weights, write_fit, whole) in (((1., 0., 0., 0., 1.,
0.), False, True),\n";
  print SALIGN "                                    ((1., 0.5, 1., 1., 1.,
0.), False, True),\n";
  print SALIGN "                                    ((1., 1., 1., 1., 1., 0.),
True, False)):\n";
  print SALIGN "  aln.salign(rms_cutoffs=(3.5, 6., 60, 60, 15, 60, 60, 60,
60, 60, 60),\n";
  print SALIGN "             normalize_pp_scores=False,\n";
  print SALIGN "             rr_file='\$(LIB)/as1.sim.mat', overhang=30,\n";
  print SALIGN "             gap_penalties_1d=(-450, -50),\n";
  print SALIGN "             gap_penalties_3d=(0, 3), gap_gap_score=0,
gap_residue_score=0,\n";
  print SALIGN "
dendrogram_file='/tmp/$datenew/fm00495.tree',\n";
  print SALIGN "             alignment_type='tree',
  print SALIGN "             feature_weights=weights,
  print SALIGN "             improve_alignment=True, fit=True,
write_fit=write_fit,\n";
  print SALIGN "             write_whole_pdb=whole, output='ALIGNMENT
QUALITY')\n";
  print SALIGN "aln.write(file='/tmp/$datenew/fm00495.pap',
alignment_format='PAP')\n";
  print SALIGN "aln.write(file='/tmp/$datenew/fm00495.ali',
alignment_format='PIR')\n";
  print SALIGN "aln.salign(rms_cutoffs=(1.0, 6., 60, 60, 15, 60, 60, 60, 60,
60, 60),\n";
  print SALIGN "         normalize_pp_scores=False,
rr_file='\$(LIB)/as1.sim.mat', overhang=30,\n";
  print SALIGN "         gap_penalties_1d=(-450, -50), gap_penalties_3d=(0,
3),\n";
  print SALIGN "         gap_gap_score=0, gap_residue_score=0,
dendrogram_file='1is3A.tree',\n";
  print SALIGN "         alignment_type='progressive',
feature_weights=[0]*6,\n";
```

```
  print SALIGN "         improve_alignment=False, fit=False,
write_fit=True,\n";
  print SALIGN "         write_whole_pdb=False, output='QUALITY')\n";
  close SALIGN;
  system("(cd /tmp/$datenew/ ; /usr/bin/mod9v1 /tmp/$datenew/salign.py)");

  open  ALIGNTWODMULT , ">/tmp/$datenew/align2dmult.py" || bail("$!");
  print ALIGNTWODMULT "from modeller import *\n";
  print ALIGNTWODMULT "log.verbose()\n";
  print ALIGNTWODMULT "env = environ()\n";
  print ALIGNTWODMULT
"env.libs.topology.read(file='\$(LIB)/top_heav.lib')\n";
  print ALIGNTWODMULT "aln = alignment(env)\n";
  print ALIGNTWODMULT "aln.append(file='/tmp/$datenew/fm00495.ali',
align_codes='all')\n";
  print ALIGNTWODMULT "aln_block = len(aln)\n";
  print ALIGNTWODMULT "aln.append(file='/tmp/$datenew/$sequence\.ali',
align_codes='$sequence')\n";
  print ALIGNTWODMULT "aln.salign(output='', max_gap_length=20,\n";
  print ALIGNTWODMULT "           gap_function=True,\n";
  print ALIGNTWODMULT "           alignment_type='PAIRWISE',
align_block=aln_block,\n";
  print ALIGNTWODMULT "           feature_weights=(1., 0., 0., 0., 0., 0.),
overhang=0,\n";
  print ALIGNTWODMULT "           gap_penalties_1d=(-450, 0),\n";
  print ALIGNTWODMULT "           gap_penalties_2d=(0.35, 1.2, 0.9, 1.2,
0.6, 8.6, 1.2, 0., 0.),\n";
  print ALIGNTWODMULT "           similarity_flag=True)\n";
  print ALIGNTWODMULT "aln.write(file='/tmp/$datenew/$sequence\-mult.ali',
alignment_format='PIR')\n";
  print ALIGNTWODMULT "aln.write(file='/tmp/$datenew/$sequence\-mult.pap',
alignment_format='PAP')\n";
  close ALIGNTWODMULT;
  system ("(cd /tmp/$datenew/ ; /usr/bin/mod9v1
/tmp/$datenew/align2dmult.py)");

  open  MODELMULT , ">/tmp/$datenew/modelmult.py" || bail("$!");
  print MODELMULT "from modeller import *\n";
  print MODELMULT "from modeller.automodel import *\n";
  print MODELMULT "env = environ()\n";
  print MODELMULT "a = automodel(env, alnfile='$sequence\-mult.ali',\n";
  print MODELMULT "              knowns=(";
  for ( $i=0 ; $i<$ftp_number ; $i++) {
    if ( $i == $ftp_number-1) {
      print MODELMULT "'$multiquery[$i]'";
      }
    else {
      print MODELMULT "'$multiquery[$i]',";
      }
    }
  print MODELMULT "), sequence='$sequence')\n";
  print MODELMULT "a.starting_model = 1\n";
  print MODELMULT "a.ending_model = 1\n";
  print MODELMULT "a.make()\n";
  close MODELMULT;
```

```
  system ("(cd /tmp/$datenew/ ; /usr/bin/mod9v1
/tmp/$datenew/modelmult.py)");

  print p("The proccess of homology modeling has finished");
  system ("mv /tmp/$datenew/UnkP.B99990001.pdb /tmp/$datenew/UnkP.pdb");
  print " Your query ID is $datenew";
  }
```

Επεξήγηση υπορουτίνας `modeller_program_multimodelling`

Με την υπορουτίνα `modeller_program_multimodelling` δημιουργούνται και τρέχουν τα python scripts που χρειάζεται το Modeller ώστε να επιτευχθεί η ομόλογη μοντελοποίηση μέσω πολλών ομόλογων πρωτεϊνών. Το script `salign.py` είναι υπεύθυνο για την δημιουργία του περιβάλλοντος που θα επιτελεσθεί η ομόλογη μοντελοποίηση, το script `align2dmult.py` είναι υπεύθυνο για την στοίχιση της αμινοξικής ακολουθίας του χρήστη με τις μέχρι πέντε ακολουθίες που έχουν τη μεγαλύτερη ομοιότητα ώστε να χρησιμοποιηθεί ως υπόστρωμα για το script `modelmult.py` που κατασκευάζει το τελικό ομόλογο μοντέλο. Μόλις ολοκληρωθεί η ομόλογη μοντελοποίηση τυπώνεται στον φυλλομετρητή το ID.
Η επεξήγηση των τιμών που έχουν οι παράμετροι που χρησιμοποιούνται από τα python scripts μπορεί να βρεθεί στην διεύθυνση http://saliab.org/modeller/8v1/manual/node1.htm.

2.2.2.2.9 Υπορουτίνα `java_files`

```
sub java_files {
  open HTML, ">/tmp/$datenew/start.html" || bail("$!");
  print HTML "PROTEIN VIEWER
  <html>
   <head>
    <title>protein viewer</title>
    <script src=\"./jmol/Jmol.js\" type=\"text/javascript\"></script>
    </head>
   <body>
   <form>
      <script type=\"text/javascript\">
        jmolInitialize(\"./jmol\");
        jmolApplet(400, \"load ./jmol/structures/$datenew.pdb\");
    </script>
    </form>
    <CENTER>
    <A HREF=\"javascript:window.close()\">Close this window</A>
    </CENTER>
   </body>
  </html>";
  close HTML;
  open JMOLSCRIPT, ">/tmp/$datenew/jmolscript.txt" || bail("$!");
  print JMOLSCRIPT "<A HREF=\"javascript:void(0)\"

onclick=\"window.open('../$datenew.html','welcome','width=440,height=490')\"
>
    <font color=black>View the structure</font></A>";
  close JMOLSCRIPT;
```

```
  system ("mv /tmp/$datenew/start.html /var/www/html/$datenew.html");
  system ("cp /tmp/$datenew/UnkP.pdb
/var/www/html/jmol/structures/$datenew.pdb");
  system ("(cd /tmp/$datenew ; /usr/bin/zip -q -r modeller_results . -i
UnkP.* *.log blast*.out *.ali *.pap *.prf protein*.txt)");
  chmod (0666,"/tmp/$datenew/modeller_results.zip");
  system ("mv /tmp/$datenew/modeller_results.zip
/var/ftp/pub/$datenew/modeller_results.zip");
  }
```

Επεξήγηση υπορουτίνας `java_files`

Με την υπορουτίνα `java_files` προετοιμάζεται το αρχείο που θα χρησιμοποιηθεί από το Jmol ώστε να δείξει την τρισδιάστατη μορφή της πρωτεΐνης, καθώς και το zip αρχείο που περιέχει τα αποτελέσματα της ομόλογης μοντελοποίησης και θα είναι προσβάσιμο προς κατέβασμα από τον χρήστη.

### 2.2.3 Τμήμα παρουσίασης Αποτελεσμάτων και Docking

Ο ρόλος του κώδικα της παρουσίασης αποτελεσμάτων και docking είναι να εμφανίσει στον φυλλομετρητή τα αποτελέσματα της ομόλογης μοντελοποίησης που διεξήχθη καθώς και την παρουσίαση των φορμών επερώτησης ώστε αν το επιθυμεί ο χρήστης να επιτελεσθεί και η διαδικασία του docking και η εμφάνιση των αποτελεσμάτων αυτής μόλις τελειώσει.

```
sub html2 {
  print header, start_html("HMDoc"); print p("<IMG src='../banner.gif'>");
  print hr();
  print p("<BODY bgcolor=#FDEEF4>");
  print p("MODELLER RESULTS");
  print p();
  if (-e "/tmp/$QIDtrue/blast.out") {
    java_script("1");
    }
  else {
    java_script("2");
    print p();
    java_script("3");
    }
  print p();
  java_script("4");
  print p();
  java();
  print p();
  ftp();
  print hr();
  upload_form();
  print hr();
  dock_results();
  print end_html;
  }
```

Επεξήγηση υπορουτίνας `html2`

Οι υπορουτίνες `java_script` και `java` είναι υπεύθυνες για την εμφάνιση των αποτελεσμάτων της ομόλογης μοντελοποίησης, η υπορουτίνα `ftp` για το κατέβασμα αρχείων που είναι τα αποτελέσματα της ομόλογης μοντελοποίησης, η υπορουτίνα `upload_form` για την φόρμα επερώτησης για τη διαδικασία του docking και η υπορουτίνα `dock_results` για το κατέβασμα αρχείων που είναι τα αποτελέσματα του docking.

### 2.2.3.1 Υπορουτίνες προβολής αποτελεσμάτων

#### 2.2.3.1.1 Υπορουτίνα `java_script`

```
sub java_script {
  if (-e "/tmp/$QIDtrue/blast.out") {
  open BLASTFILE, "/tmp/$QIDtrue/blast.out" || bail("$!");
  @count = <BLASTFILE>;
  close BLASTFILE;
  }
  else {
  open BLASTFILEONE, "/tmp/$QIDtrue/blast1.out" || bail("$!");
  @countone= <BLASTFILEONE>;
  close BLASTFILE;
  open BLASTFILETWO, "/tmp/$QIDtrue/blast2.out" || bail("$!");
  @counttwo= <BLASTFILETWO>;
  close BLASTFILETWO;
  }
  $leest = </tmp/$QIDtrue/*-*.pap>;
  if (defined($leest)) {
    open ALI, "$leest" || bail("$!");
    @count1 = <ALI>;
    close ALI;
    }
  print q(<script type="text/javascript">);
  print q(<!--
        function advance(obj) {
          var child;
          child = obj.firstChild;
          sibling = child.nextSibling;
          if(child.style.display != 'none') {
            child.style.display = 'none';
            sibling.style.display = 'inline';
            }
          else {
          child.style.display = 'inline';
          sibling.style.display = 'none';
          }
        }
        // -->);
  print q(</script>);
  if ($_[0]=='1') {
    print q(<table align="center" border="1" width="90%" cellpadding="3"
cellspacing="0">
```

```
            <tr><td ><b>Blast Output</b></td></tr><tr><td
onclick="advance(this)" class="quote"
            style="cursor: pointer;"><span style="display: none">);
            print q(<pre>);
            print ("@count");
            print q(</pre>);
    print q(</span><span>Click to expand...</span></td></tr></table>);
    }
.................................
  elsif ($_[0]=='4') {
    print q(<table align="center" border="1" width="90%" cellpadding="3"
cellspacing="0">
            <tr><td ><b>Ali Output</b></td></tr><tr><td
onclick="advance(this)" class="quote"
            style="cursor: pointer;"><span style="display: none">);
            print q(<pre>);
            print ("@count1");
            print q(</pre>);
    print q(</span><span>Click to expand...</span></td></tr></table>);
    }
  }
```

Επεξήγηση υπορουτίνας `java_script`

Με την υπορουτίνα `java_script` προβάλλονται στον φυλλομετρητή μέσω της γλώσσας προγραμματισμού JavaScript τα αποτελέσματα του BLAST καθώς και η αλληλούχιση που χρησιμοποίησε το Modeller ώστε να ολοκληρωθεί η ομόλογη μοντελοποίηση.

2.2.3.1.2 Υπορουτίνα `java`

```
sub java {
  open JMOLSCRIPT, "/tmp/$QIDtrue/jmolscript.txt" || bail("$!");
  @jmolscript=<JMOLSCRIPT>;
  close JMOLSCRIPT;
  print "@jmolscript";
  }
```

Επεξήγηση υπορουτίνας `java`

Χρησιμοποιούνται τα αρχεία από την υπορουτίνα `java_files` ώστε να προβληθεί μέσω του Jmol η πρωτεΐνη που μοντελοποιήθηκε.

### 2.2.3.2 Υπορουτίνες για τη διαδικασία του Docking

2.2.3.2.1 Υπορουτίνα `upload_form`

```
sub upload_form {
  $k=0;
  system ("(cd /var/ftp/pub/$QIDtrue/ligands ; /bin/ls >
list_of_files.txt)");
  open LIST, "/var/ftp/pub/$QIDtrue/ligands/list_of_files.txt";
```

```
while (<LIST>) {
  if (/pdb/) {
    @line1=$_;
    $line_number = @line1;
    $line_number = $line_number - 1;
    $dock_choices[$k]=$line1[0];
    $k++;
    }
  }
  close LIST;
my $upload_dir = "/var/ftp/pub/$QIDtrue/ligands";
$cur4 = new CGI;
print $cur4->startform(-method=>'POST',
                       -action=>$action,
                       -enctype=>'multipart/form-data');
print p("Files to upload");
print $cur4->filefield(-name=>'uplfile',
                       -default=>'starting value',
                       -size=>50,
                       -maxlength=>80);
print p();
print $cur4->submit(-name=>'start',
                    -value=>'Upload Files');
print hr();
print p ("DOCK AREA");
print $cur4->start_form(-method=>'POST',
                        -action=>$action,
                        -enctype=>$encoding);
print p("Perform docking with molecule");
print $cur4->popup_menu('choice',\@dock_choices);
print p();
print $cur4->submit(-name=>'start',
                    -value=>'Start Docking');
print $cur4->submit(-name=>'start',
                    -value=>'MultiDock');
print $cur4->endform;
$d_choice=(param('choice'));
$Start=(param('start'));
my $filename = (param('uplfile'));
if (defined($Start)) {
  if (param) {
    if ($Start eq 'Upload Files' && $filename ne '') {
      my ( $name, $path, $extension ) = fileparse ( $filename, '\..*' );
      $filename = $name . $extension;
      $filename =~ tr/ /_/;
      my $upload_filehandle = (upload("uplfile"));
      open ( UPLOADFILE, ">$upload_dir/$filename", ) || bail("$!");
      chmod(0666,"/$upload_dir/$filename");
      binmode UPLOADFILE;
      while ( <$upload_filehandle> ) {
        print UPLOADFILE;
        }
      close UPLOADFILE;
      }
```

```
      elsif ($Start eq 'Start Docking') {
        dock("1");
        }
      elsif ($Start eq 'MultiDock') {
        dock("2");
        }
      }
    }
  }
```

Επεξήγηση υπορουτίνας `upload_form`

Η υπορουτίνα `upload_form` περιέχει δύο λειτουργίες. Η πρώτη είναι να δίνει τη δυνατότητα στο χρήστη να ανεβάσει τα αρχεία που περιέχουν τα μόρια που θα λειτουργήσουν ως ligands στη διαδικασία του Docking, και τα τοποθετεί σε ένα κατάλογο. Η δεύτερη λειτουργία είναι να μπορεί να διαλέξει ο χρήστης ένα ligand για να ξεκινήσει η διαδικασία πατώντας το κουμπί υποβολής `Start Docking` ή πατώντας το κουμπί υποβολής `MultiDock` όλα τα ligands που έχει ανεβάσει ο χρήστης θα ξεκινήσουν ένα-ένα τη διαδικασία..

2.2.3.2.2 Υπορουτίνα `dock`

```
sub dock {
  $k=0;
  system ("(cd /var/ftp/pub/$QIDtrue/ligands ; /bin/ls >
list_of_files.txt)");
  open LIST, "/var/ftp/pub/$QIDtrue/ligands/list_of_files.txt";
  while (<LIST>) {
    if (/pdb/) {
      @line1=$_;
      $line_number = @line1;
      $line_number = $line_number - 1;
      $dock_choices[$k]=$line1[0];
      $k++;
      }
    }
    close LIST;
  if ($_[0]=='1') {
    if ($d_choice ne '') {
    $lig = $d_choice;
    $lig =~ s/.pdb//;
    unless (-e "/tmp/$QIDtrue/UnkP_$lig/") {
      mkdir ("/tmp/$QIDtrue/UnkP_$lig/",0777);
      system ("cp /var/ftp/pub/$QIDtrue/ligands/$d_choice
/tmp/$QIDtrue/UnkP_$lig/$d_choice");
      system ("cp /tmp/$QIDtrue/UnkP.pdb /tmp/$QIDtrue/UnkP_$lig/UnkP.pdb");
      print hr();
      print p();
      print ("The autodock proccess with $lig ligand has started....");
      }
    unless (-e "/tmp/$QIDtrue/UnkP_$lig/$lig\.pdbqt") {
      system ("( cd /tmp/$QIDtrue/UnkP_$lig/ ; /usr/local/MGLTools-
1.4.6/share/bin/pythonsh /usr/local/MGLTools-
```

```
1.4.6/MGLToolsPckgs/AutoDockTools/Utilities24/prepare_ligand4.py -l
/tmp/$QIDtrue/UnkP_$lig/$d_choice)");
      }
    unless (-e "/tmp/$QIDtrue/UnkP_$lig/UnkP.pdbqt") {
      system ("( cd /tmp/$QIDtrue/UnkP_$lig/ ; /usr/local/MGLTools-
1.4.6/share/bin/pythonsh /usr/local/MGLTools-
1.4.6/MGLToolsPckgs/AutoDockTools/Utilities24/prepare_receptor4.py -r
/tmp/$QIDtrue/UnkP_$lig/UnkP.pdb)");
      }
    unless (-e "/tmp/$QIDtrue/UnkP_$lig/UnkP.gpf") {
      system ("( cd /tmp/$QIDtrue/UnkP_$lig/ ; /usr/local/MGLTools-
1.4.6/share/bin/pythonsh /usr/local/MGLTools-
1.4.6/MGLToolsPckgs/AutoDockTools/Utilities24/prepare_gpf4.py -l
/tmp/$QIDtrue/UnkP_$lig/$lig\.pdbqt -r
/tmp/$QIDtrue/UnkP_$lig/UnkP.pdbqt)");
      }
    unless (-e "/tmp/$QIDtrue/UnkP_$lig/$lig\_UnkP.dpf") {
      system ("( cd /tmp/$QIDtrue/UnkP_$lig/ ; /usr/local/MGLTools-
1.4.6/share/bin/pythonsh /usr/local/MGLTools-
1.4.6/MGLToolsPckgs/AutoDockTools/Utilities24/prepare_dpf4.py -l
/tmp/$QIDtrue/UnkP_$lig/$lig\.pdbqt -r
/tmp/$QIDtrue/UnkP_$lig/UnkP.pdbqt)");
      }
    unless (-e "/tmp/$QIDtrue/UnkP_$lig/UnkP.glg") {
      system ("( cd /tmp/$QIDtrue/UnkP_$lig/ ; /usr/local/autodock/autogrid4
-p /tmp/$QIDtrue/UnkP_$lig/UnkP.gpf -l /tmp/$QIDtrue/UnkP_$lig/UnkP.glg)");
      }
    unless (-e "/tmp/$QIDtrue/UnkP_$lig/$lig\_UnkP.dlg") {
      system ("( cd /tmp/$QIDtrue/UnkP_$lig/ ; /usr/local/autodock/autodock4
-p /tmp/$QIDtrue/UnkP_$lig/$lig\_UnkP.dpf -l
/tmp/$QIDtrue/UnkP_$lig/$lig\_UnkP.dlg)");
      system ("(cd /tmp/$QIDtrue ; /usr/bin/zip -q -r dock_$lig\_results . -
i UnkP*.map* *.glg *.gpf *.dpf *.dlg)");
      chmod (0666,"/tmp/$QIDtrue/dock_$lig\_results.zip");
      system ("mv /tmp/$QIDtrue/dock_$lig\_results.zip
/var/ftp/pub/$QIDtrue/dock_$lig\_results.zip");
      my $woot1='<A
HREF='.'"ftp://anonymous:@127.0.0.1/pub/'.$QIDtrue.'/dock_'.$lig.'_results.z
ip"<font color=black>Download dock resuslts</font></A>';
      print ("and finished, $woot1");
      }
    }
    else {
    bail2("No ligand chosen");
    }
  }
  elsif ($_[0]=='2') {
    if ($d_choice ne '') {
      print hr();
      for ( $i=0 ; $i < @dock_choices ; $i++) {
        $lig = $dock_choices[$i];
        $lig =~ s/.pdb//;
        chomp ($lig);
        unless (-e "/tmp/$QIDtrue/UnkP_$lig/") {
          mkdir ("/tmp/$QIDtrue/UnkP_$lig/",0777);
          system ("cp /var/ftp/pub/$QIDtrue/ligands/$lig\.pdb
```

```
/tmp/$QIDtrue/UnkP_$lig/$lig\.pdb");
          system ("cp /tmp/$QIDtrue/UnkP.pdb
/tmp/$QIDtrue/UnkP_$lig/UnkP.pdb");
          print p();
          print ("The autodock proccess with $lig ligand has started....");
          }
        unless (-e "/tmp/$QIDtrue/UnkP_$lig/$lig\.pdbqt") {
          system ("( cd /tmp/$QIDtrue/UnkP_$lig/ ; /usr/local/MGLTools-
1.4.6/share/bin/pythonsh /usr/local/MGLTools-
1.4.6/MGLToolsPckgs/AutoDockTools/Utilities24/prepare_ligand4.py -l
/tmp/$QIDtrue/UnkP_$lig/$dock_choices[$i])");
          }
        unless (-e "/tmp/$QIDtrue/UnkP_$lig/UnkP.pdbqt") {
          system ("( cd /tmp/$QIDtrue/UnkP_$lig/ ; /usr/local/MGLTools-
1.4.6/share/bin/pythonsh /usr/local/MGLTools-
1.4.6/MGLToolsPckgs/AutoDockTools/Utilities24/prepare_receptor4.py -r
/tmp/$QIDtrue/UnkP_$lig/UnkP.pdb)");
          }
        unless (-e "/tmp/$QIDtrue/UnkP_$lig/UnkP.gpf") {
          system ("( cd /tmp/$QIDtrue/UnkP_$lig/ ; /usr/local/MGLTools-
1.4.6/share/bin/pythonsh /usr/local/MGLTools-
1.4.6/MGLToolsPckgs/AutoDockTools/Utilities24/prepare_gpf4.py -l
/tmp/$QIDtrue/UnkP_$lig/$lig\.pdbqt -r
/tmp/$QIDtrue/UnkP_$lig/UnkP.pdbqt)");
          }
        unless (-e "/tmp/$QIDtrue/UnkP_$lig/$lig\_UnkP.dpf") {
          system ("( cd /tmp/$QIDtrue/UnkP_$lig/ ; /usr/local/MGLTools-
1.4.6/share/bin/pythonsh /usr/local/MGLTools-
1.4.6/MGLToolsPckgs/AutoDockTools/Utilities24/prepare_dpf4.py -l
/tmp/$QIDtrue/UnkP_$lig/$lig\.pdbqt -r
/tmp/$QIDtrue/UnkP_$lig/UnkP.pdbqt)");
          }
        unless (-e "/tmp/$QIDtrue/UnkP_$lig/UnkP.glg") {
          system ("( cd /tmp/$QIDtrue/UnkP_$lig/ ;
/usr/local/autodock/autogrid4 -p /tmp/$QIDtrue/UnkP_$lig/UnkP.gpf -l
/tmp/$QIDtrue/UnkP_$lig/UnkP.glg)");
          }
        unless (-e "/tmp/$QIDtrue/UnkP_$lig/$lig\_UnkP.dlg") {
          system ("( cd /tmp/$QIDtrue/UnkP_$lig/ ;
/usr/local/autodock/autodock4 -p /tmp/$QIDtrue/UnkP_$lig/$lig\_UnkP.dpf -l
/tmp/$QIDtrue/UnkP_$lig/$lig\_UnkP.dlg)");
          system ("(cd /tmp/$QIDtrue ; /usr/bin/zip -q -r dock_$lig\_results
. -i UnkP*.map* *.glg *.gpf *.dpf *.dlg)");
          chmod (0666,"/tmp/$QIDtrue/dock_$lig\_results.zip");
          system ("mv /tmp/$QIDtrue/dock_$lig\_results.zip
/var/ftp/pub/$QIDtrue/dock_$lig\_results.zip");
          $wooty[$i]='<A
HREF='.'"ftp://anonymous:@127.0.0.1/pub/'.$QIDtrue.'/dock_'.$lig.'_results.z
ip"<font color=black>Download dock resuslts</font></A>';
          print ("and finished, ");
          print ($wooty[$i]);
          }
        }
      }
    else {
      bail2("No ligands chosen");
```

```
      }
    }
  }
```

Επεξήγηση υπορουτίνας `dock`

Η υπορουτίνα `dock` ξεκινά τις κατάλληλες διεργασίες για την προετοιμασία του μορίου-στόχου και του/των ligand/s για να τα δεχθεί ως input το Autodock που θα κάνει τη διεργασία του Docking.

### 2.2.3.3 Υπορουτίνες για κατέβασμα αρχείων

#### 2.2.3.3.1 Υπορουτίνα `ftp`

```
sub ftp {
  $woot='<A HREF='.'"ftp://anonymous:@127.0.0.1/pub/'.
$QIDtrue.'/modeller_results.zip"<font color=black>Download Modeller
resuslts</font></A>';
  print p($woot);
  }
```

#### 2.2.3.3.2 Υπορουτίνα `dock_results`

```
sub dock_results {
  $k=0;
  system ("(cd /var/ftp/pub/$QIDtrue/ ; /bin/ls >
list_of_zipped_files.txt)");
  open LISTZIP, "/var/ftp/pub/$QIDtrue/list_of_zipped_files.txt";
  while (<LISTZIP>) {
    if (/dock/) {
      @line2=$_;
      $line_number = @line2;
      $line_number = $line_number - 1;
      $dock_zip[$k]=$line2[0];
      $k++;
      }
    }
  close LISTZIP;
  for ($i=0; $i < @dock_zip ; $i++) {
    chomp ($dock_zip[$i]);
    ($trash,$ligand,$trash2) = split (/_/, $dock_zip[$i]);
    $wootty[$i]='<A
HREF='.'"ftp://anonymous:@127.0.0.1/pub/'.$QIDtrue.'/'.$dock_zip[$i].'"<font
color=black>Download dock resuslts with '.$ligand.' ligand</font></A>';
    print ($wootty[$i]);
    print p();
    }
  }
```

Επεξήγηση υπορουτινών `ftp` και `dock_results`

Αυτές οι υπορουτίνες δίνουν την δυνατότητα στο χρήστη να κατεβάσει τα αποτελέσματα της ομόλογης μοντελοποίησης και του Docking αντίστοιχα σε zip αρχεία.

## 2.2.4 Υπορουτίνες προβολής Λαθών

2.2.4.1 Υπορουτίνα `bail`

```
sub bail {
  my $error = "@_";
  print h1 ("Error"), p($error), end_html;
  die $error;
  }
```

2.2.4.2 Υπορουτίνα `bail2`

```
sub bail2 {
  if ($_[0] eq 'Wrong ID entered') {
    print header, start_html("Error");
    }
  my $error2 = "@_";
  print h1 ("Notification"), p($error2), end_html;
  die $error2;
  }
```

Επεξήγηση υπορουτινών `bail` και `bail2`

Αν συμβεί κάποιο λάθος κατά τη λειτουργία του προγράμματος οι υπορουτίνες αυτές αναλαμβάνουν να ενημερώσουν τον χρήστη για τα λάθη του συστήματος ή του χρήστη αντίστοιχα.

## 2.3 Υπολογιστικά κόστη

Όλα τα πειράματα έγιναν σε υπολογιστή με τα εξής χαρακτηριστικά:
Επεξεργαστής: Intel Pentium 4 CPU 2.00 GHz
Μνήμη: 1,00 GB RAM
Λειτουργικό σύστημα: Scientific Linux 4

Για την ολοκλήρωση της διαδικασίας της ομόλογης μοντελοποίησης παρατηρήθηκε ότι χρειάζεται από 5 έως 10 λεπτά για μοντελοποίηση μέσω μίας ομόλογης δομής, και από 9 έως 15 λεπτά για μοντελοποίηση μέσω 2 μέχρι 5 ομόλογων δομών. Η διαφορά στο χρόνο που χρειάζεται οφείλεται σε παράγοντες όπως ο αριθμός και το είδος των αμινοξέων της πρωτεΐνης προς μοντελοποίηση, το ποσοστό ομοιότητας αυτής με τις πρωτεΐνες που θα χρησιμοποιηθούν ως υπόστρωμα, το πλήθος των δευτεροταγών δομών που περιέχονται στην αμινοξική ακολουθία καθώς και την ύπαρξη ή μη άλλων διαδικασιών που να εκτελούνται παράλληλα στον υπολογιστή. Για τον έλεγχο αυτών των ισχυρισμών έγιναν περίπου 100 δοκιμές διαφορετικών πρωτεϊνών στο υπολογιστικό σύστημα που προαναφέρθηκε.

Για την ολοκλήρωση της διαδικασίας του docking παρατηρήθηκε ότι χρειάζεται από

1,5 έως 3 ώρες. Η διαφορά στο χρόνο έγκειται σε παράγοντες όπως το μέγεθος της πρωτεΐνης, το μέγεθος και το είδος των στοιχείων που απαρτίζουν το μόριο-προσδέτη, το πλήθος των πιθανών θέσεων σύζευξης μεταξύ πρωτεΐνης-δέκτη και μορίου-προσδέτη, το πλήθος των δευτεροταγών δομών που περιέχονται στην πρωτεΐνη, το πλήθος των δεσμών που μπορούν να δημιουργηθούν καθώς και την ύπαρξη ή μη άλλων διαδικασιών που να εκτελούνται παράλληλα στον υπολογιστή. Για τον έλεγχο αυτών των ισχυρισμών έγιναν περίπου 20 δοκιμές μεταξύ διαφορετικών πρωτεϊνών και μορίων που παίζουν το ρόλο προσδέτη στο υπολογιστικό σύστημα που προαναφέρθηκε.

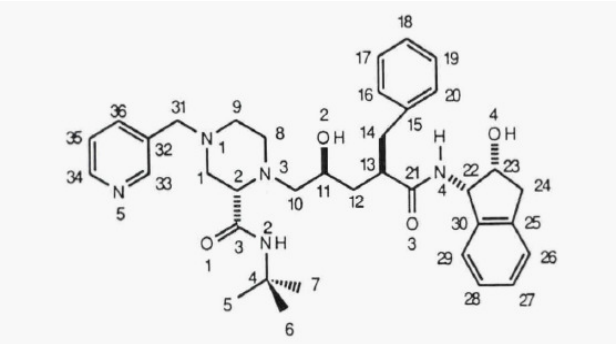
Γενικά, για τη λειτουργία των παραπάνω διαδικασιών παρατηρήθηκε ακόμα ότι χρησιμοποιούσαν το 60%-70% της υπολογιστικής ισχύος (το σύστημα κρατούσε ελεύθερο ένα ποσοστό της ισχύος για τυχόν μελλοντική χρήση) καθώς και το 80% της ελεύθερης μνήμης RAM. Από δοκιμές σε άλλα υπολογιστικά συστήματα προέκυψε το συμπέρασμα ότι η αύξηση της ταχύτητας του επεξεργαστή επιταχύνει σημαντικά όλες τις διαδικασίες (σε επεξεργαστή Intel Pentium 4 CPU 2.6 GHz έγινε ταχύτερα η ομόλογη μοντελοποίηση κατά 1 λεπτό και η διαδικασία του docking κατά 20 λεπτά), ενώ η αύξηση της μνήμης RAM πέραν του 1,00 GB παρουσιάζει αμελητέα επιτάχυνση των διαδικασιών.

# ΚΕΦΑΛΑΙΟ 3 – ΠΑΡΑΔΕΙΓΜΑ ΕΦΑΡΜΟΓΗΣ

Στο παρόν κεφάλαιο παρουσιάζεται ένα παράδειγμα από την υποθετική ανακάλυψη μιας πρωτεΐνης μέχρι την περαιτέρω μελέτη της, δηλαδή την ανάλυση των αποτελεσμάτων από τις διαδικασίες της ομόλογης μοντελοποίησης και του docking. Κρίθηκε αναγκαίο να χρησιμοποιηθεί ένα σύμπλεγμα γνωστής στερεοδιάταξης ώστε να γίνει και επαλήθευση της σωστής λειτουργίας του προγράμματος.

## 3.1 Εισαγωγή

Η πρωτεΐνη με την κωδική ονομασία 1hsg (της PDB βάσης δεδομένων) είναι μια HIV-2 πρωτεάση του ιού της ανθρώπινης ανοσοανεπάρκειας (HIV) – ο ιός HIV-1 είναι υπεύθυνος για την αρρώστια του AIDS στην Ευρώπη και την Βόρεια Αμερική, ενώ ο ιός HIV-2 στην Δυτική Αφρική. Πρέπει να τονιστεί ότι δομικά οι πρωτεάσες των δύο αυτών ιών είναι πανομοιότυπες. Στην προσπάθεια να βρεθούν ικανοί αναστολείς της HIV πρωτεάσης χρησιμοποιήθηκε ο αναστολέας MK1 (ή αλλιώς L-735,524) με ικανοποιητικά αποτελέσματα. Ο αναστολέας MK1 ανήκει στην υδροξυαμινοπεντανική αμιδική κλάση των μιμητών μικροπεπτιδίου (Εικ. 3-1) και είναι βιοδιαθέσιμος από τρία είδη εργαστηριακών ζώων [9].



**Εικόνα 3-1**. Δομή αναστολέα MK1.

In vitro ο αναστολέας MK1 δρα ανασταλτικά στις πέψεις πεπτιδίων που καταλύονται από την HIV πρωτεάση. Συγκεκριμένα, οι αλληλεπιδράσεις μεταξύ του ενζύμου και του αναστολέα περιλαμβάνουν τα υδροξικά άκρα του MK1 να προσδένονται στα καρβοξικά άκρα των αμινοξέων ASP-25 και ASP-25', καθώς και τα αμιδικά οξυγόνα του αναστολέα να αλληλεπιδρούν με τα αμινοξέα του σκελετού του ενζύμου Ile-50 και Ile-50' μέσω ενός μορίου νερού. Αν και άλλοι δεσμοί υδρογόνου συνεισφέρουν στην σταθερότερη πρόσδεση, οι δύο παραπάνω προσδέσεις θεωρούνται οι πιο σημαντικές [9].

Έχοντας υπόψη τα προαναφερθέντα, κλινικές έρευνες μπορούν να διεξαχθούν ώστε να επαληθευθούν οι πιθανές θεραπευτικές χρήσεις του αναστολέα MK1 στην καταπολέμηση της αρρώστιας του AIDS.

## 3.2 Εκτέλεση προγράμματος

Για την εκτέλεση του προγράμματος επιλέγχθηκε η πρωτεάση του ιού HIV-2 και η εύρεση των πιθανών αλληλεπιδράσεών της (που θα μπορούσε να έχει ως στόχο την εύρεση θεραπείας για την αρρώστια του AIDS). Η αμινοξική ακολουθία της πρωτεάσης βρέθηκε στις βάσεις δεδομένων που υπάρχουν στο διαδίκτυο και περιλαμβάνει δύο όμοιες πολυπεπτιδικές αλυσίδες με την αμινοξική ακολουθία:

PQITLWQRPLVTIKIGGQLKEALLDTGADDTVLEEMSLPGRWKPKMIGGIGGFIKVRQY
DQILIEICGHKAIGTVLVGPTPVNIIGRNLLTQIGCTLNF

Για τον έλεγχο των πιθανών αλληλεπιδράσεων της πρωτεΐνης με μόρια-προσδέτες θα πρέπει αρχικά να προσδιοριστεί η τριτοταγής δομή της πρωτεΐνης. Ανοίγουμε λοιπόν τον φυλλομετρητή, πληκτρολογούμε την διεύθυνση του προγράμματος και στο κατάλληλο "widget" πεδίου κειμένου πληκτρολογούμε την αμινοξική αλληλουχία της πρωτεΐνης (εφόσον η πρωτεΐνη μας αποτελείται από δύο πολυπεπτιδικές αλυσίδες κάνουμε click στο κουμπί υποβολής Feeling Lucky).

Αφού το πρόγραμμα τελειώσει την ομόλογη μοντελοποίηση ελέγχουμε την αξιοπιστία της διαδικασίας καθώς και το αρχείο που περιέχει την τρισδιάστατη δομή της πρωτεΐνης. την προκειμένη περίπτωση η δομή με τη μεγαλύτερη ομοιότητα είναι η πρωτεΐνη με την κωδική ονομασία 2avm (της PDB βάσης δεδομένων) με ποσοστό ομοιότητας 93% (ενώ ποσοστό παρόμοιας λειτουργείας αμινοξέων 97%), η οποία είναι μια πρωτέαση του ιού HIV-1 (Εικ 3-2). Έτσι, έχοντας υπόψη ότι το μοντέλο που παρήχθη από την ομόλογη μοντελοποίηση έχει μεγάλες πιθανότητες να είναι ακριβές μπορούμε να συνεχίσουμε την περαιτέρω μελέτη μας δοκιμάζοντας διάφορα μόρια-προσδέτες για να βρούμε το κατάλληλο που θα προσδένεται στην πρωτεάση αναστέλλοντας πιθανώς τη λειτουργία της.

```
_aln.pos          10        20        30        40        50        60
2avm       PQITLWKRPLVTIKIGGQLKEALIDTGADDTVIEEMSLPGRWKPKMIGGIGGFIKVRQYDQIIIEIAG
UnkP       PQITLWQRPLVTIKIGGQLKEALLDTGADDTVLEEMSLPGRWKPKMIGGIGGFIKVRQYDQILIEICG
 _consrvd  ****** ********************* ******** ****************************** *** *

 _aln.p    70        80        90       100       110       120       130
2avm       HKAIGTVLVGPTPVNIIGRNLLTQIGATLNF/PQITLWKRPLVTIKIGGQLKEALIDTGADDTVIEEM
UnkP       HKAIGTVLVGPTPVNIIGRNLLTQIGCTLNF/PQITLWQRPLVTIKIGGQLKEALLDTGADDTVLEEM
 _consrvd  ************************** *********** ********************* ******** ***

 _aln.pos   140       150       160       170       180       190
2avm       SLPGRWKPKMIGGIGGFIKVRQYDQIIIEIAGHKAIGTVLVGPTPVNIIGRNLLTQIGATLNF
UnkP       SLPGRWKPKMIGGIGGFIKVRQYDQILIEICGHKAIGTVLVGPTPVNIIGRNLLTQIGCTLNF
 _consrvd  ************************** *** ************************** ****
```

**Εικόνα 3-2**. Μέρος αρχείου που δείχνει την ομολογία των δύο πρωτεϊνών.

"Ανεβάσαμε" 3 πιθανά μόρια-προσδέτες, και όταν τελείωσε η διαδικασία του docking ελέγχουμε τα αρχεία εξόδου που παρήχθησαν. Διαπιστώθηκε ότι μόνο το μόριο-προσδέτης MK1 παρουσιάζει πιθανές θέσεις πρόσδεσης, όλες στην περιοχή ανάμεσα στα αμινοξέα ASP-25 και ASP-25'.

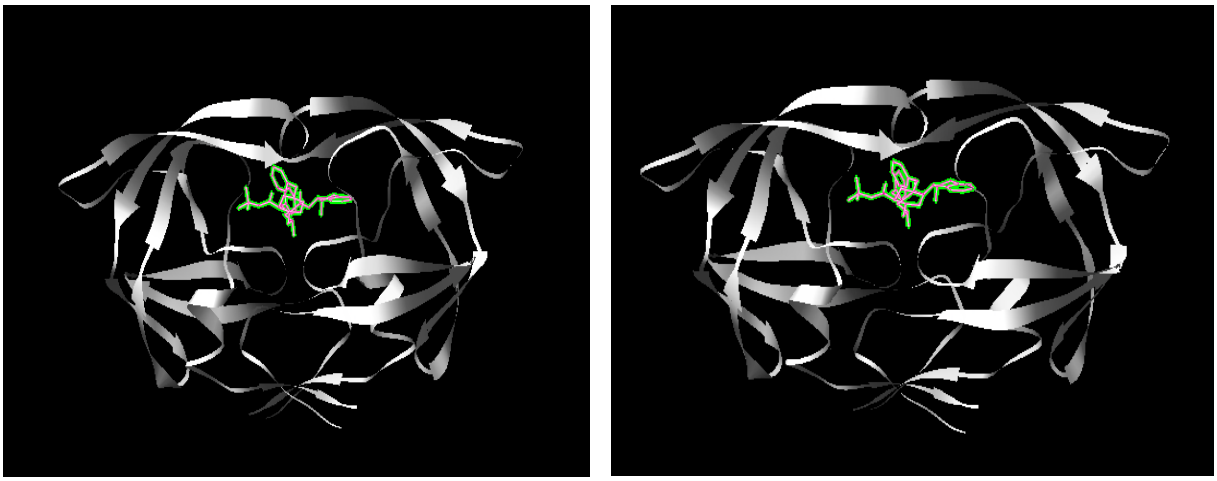
Αφού βρήκαμε ένα πιθανό μόριο-προσδέτη μέσω υπολογιστικών μεθόδων μπορούν να διεξαχθούν και διάφορες εργαστηριακές μελέτες για να επαληθευτούν οι ισχυρισμοί που προέκυψαν από την όλη διαδικασία.

## 3.3 Επαλήθευση Αποτελεσμάτων

Όλο το πείραμα διεξήχθη για να μπορέσουμε να επαληθεύσουμε εάν το πρόγραμμα λειτουργεί σωστά. Αρχικά θα μελετήσουμε τη δομή που δημιουργήθηκε από την ομόλογη μοντελοποίηση με την ήδη υπάρχουσα δομή της 1hsg που βρίσκεται στην PDB βάση δεδομένων. Από την εικόνα 3-3 διαπιστώνουμε ότι η ομόλογη μοντελοποίηση επιτελέστηκε επιτυχώς, μιας και οι δύο πρωτεΐνες είναι ολόιδιες.

**Εικόνα 3-3**. Αριστερά η πρωτεΐνη που δημιουργήθηκε από την ομόλογη μοντελοποίηση και δεξιά η πρωτεΐνη 1hsg.

Έπειτα μελετήθηκε και αν τα αποτελέσματα του docking θα συνέπιπταν με την τρισδιάστατη διαμόρφωση που υιοθετείται μεταξύ μορίου-δέκτη και μορίου-προσδέτη (Εικ 3-4). Με προσεκτική παρατήρηση διαπιστώνεται ότι δε συμπίπτουν εντελώς αν και μεγάλη ομοιότητα είναι προφανής. Αξίζει να τονιστεί ότι τα αποτελέσματα του docking βρίσκονται στο αρχείο με τη κατάληξη .dlg, το οποίο περιείχε 9 πιθανές θέσεις προσδέσεις όλες στην γενική "γειτονιά" των αμινοξέων ASP-25 και ASP-25' του μορίου-δέκτη. Για την μελέτη του dlg αρχείου χρησιμοποιήθηκαν τα AutoDock Tools, των οποίων το tutorial μπορεί να βρεθεί στον δικτυακό τόπο http://autodock.scripps.edu/.



**Εικόνα 3-4**. Αριστερά το αποτέλεσμα του docking που πραγματοποιήθηκε και δεξιά η ήδη υπάρχουσα στερεοδιαμόρφωση από τη PDB βάση δεδομένων.

# ΚΕΦΑΛΑΙΟ 4 – ΣΥΖΗΤΗΣΗ

Κάθε υπολογιστικό πρόγραμμα από τη στιγμή που θα δημιουργηθεί χρειάζεται συνεχείς βελτιώσεις για να μπορεί να ανταπεξέλθει στις εξελίξεις καθώς και τις επεκτάσεις ώστε να γίνει περισσότερο λειτουργικό.

Κύριο μέλημα ήταν η απλούστευση των διαδικασιών της ομόλογης μοντελοποίησης και του docking ώστε να μπορεί ο χρήστης εύκολα και απλά να τις χρησιμοποιεί. Χάριν αυτής της απλότητας όμως, χρειάστηκε να στερηθεί ο χρήστης κάποιες επιτρεπόμενες εξ' ορισμού μεταβολές της λειτουργίας αυτών των διεργασιών με το να χρησιμοποιεί το πρόγραμμα κάποιες σταθερές ορισμένες μεταβλητές.

Συγκεκριμένα κατά τη διαδικασία της ομόλογης μοντελοποίησης η μόνη μεταβλητή που μπορεί να θέσει ο χρήστης είναι η τιμή E-value χωρίς να μπορεί να έχει διαδραστικό ρόλο στο είδος των πρωτεϊνών που επιλέγονται από το πρόγραμμα για να γίνει η ομόλογη μοντελοποίηση. Κινδυνεύουν έτσι να χαθούν πολύτιμα ομόλογα μοντέλα που περιέχουν ένα πολύ υψηλό ποσοστό ομολογίας σε συγκεκριμένα συνεχόμενα αμινοξέα ενώ στα υπόλοιπα χαμηλό, και να χρησιμοποιηθούν ομόλογα μοντέλα που περιέχουν γενικά μέτριο ποσοστό ομολογίας. Μια πιθανή βελτίωση σε αυτόν τον τομέα θα ήταν ο χρήστης να μπορεί να επιλέγει αν το επιθυμεί τα ομόλογα μοντέλα με τα οποία θα τελεστεί η ομόλογη μοντελοποίηση.

Κατά τη διαδικασία του docking τα πράγματα έχουν απλουστευτεί ακόμα περισσότερο. Ο χρήστης δεν έχει την παραμικρή δυνατότητα να θέσει οποιαδήποτε μεταβλητή. Και για να γίνει ακόμα πιο περίεργο, ορισμένες θέσεις πρόσδεσης χρειάζονται διαφορετικές μεταβλητές από κάποιες άλλες. Εξάλλου, το όλο θέμα του docking υποστηρίζει τη δοκιμασία διαφορετικών μεταβλητών και αλλαγή των παραμέτρων ώστε να βρεθούν οι κατάλληλες θέσεις πρόσδεσης. Βέβαια, ο χρήστης θα πρέπει να είναι γνώστης του αντικειμένου, ξεφεύγοντας έτσι από την απλότητα που προσπαθήθηκε να επιτευχθεί. Μια πιθανή βελτίωση θα μπορούσε να είναι μια αναλυτικότερη και απλουστευμένη επεξήγηση των μεταβλητών με συγκεκριμένα παραδείγματα εφαρμογής της καθεμίας καθώς και την υπόδειξη της προτεινόμενης τιμής κάθε μεταβλητής από το πρόγραμμα.

Επίσης έχει αφαιρεθεί το δικαίωμα στον χρήστη να θέτει αμινοξέα του μορίου-δέκτη τα οποία να έχουν ελευθερία "κινήσεων" γύρω από τον άξονα των δεσμών τους, υιοθετώντας την στατική αντίληψη για όλα τα αμινοξέα. Κινδυνεύουν έτσι να χαθούν θέσεις πρόσδεσης, αφού κάθε πρωτεΐνη δεν είναι στατική αλλά αλλάζει συνεχώς (έστω και πολύ λίγο) στερεοδιαμόρφωση. Από την άλλη ορισμένα αμινοξέα μιας πρωτεΐνης συμπεριφέρονται μόνο ως στατικά και έτσι πιθανή προσθήκη δεσμικής ελευθερίας σε στατικά αμινοξέα να οδηγήσει σε λανθασμένα αποτελέσματα. Μια πιθανή βελτίωση θα ήταν η προσθήκη της δυνατότητας στον χρήστη να μπορεί δίνει δεσμική ελευθερία, παράλληλα όμως να έχει γίνει μια πιθανή μελέτη για τον προσδιορισμό αυτών των αμινοξέων.

Πρέπει να τονιστεί ότι και τα προγράμματα που χρησιμοποιήθηκαν εξελίσσονται συνεχώς, και έτσι για να είναι το HMDock πάντα επίκαιρο θα πρέπει να χρησιμοποιεί τις καινούργιες εκδόσεις και πιθανόν τις καινούργιες λειτουργίες αυτών.

Τέλος, πάντα υπάρχει η πιθανότητα βελτίωσης του κώδικα του προγράμματος ώστε να βελτιστοποιηθεί η ταχύτητα και η λειτουργία του.

Η τρέχουσα έκδοση του HMDock είναι η 1.00 και στην ιστοσελίδα utopia.duth.gr/~gk6624 θα παρέχονται μελλοντικά οι επόμενες εκδόσεις.

# ΠΑΡΑΡΤΗΜΑ – Α

Για την καλύτερη κατανόηση της λειτουργίας των προγραμμάτων BLAST, Modeller, και Autodock κρίνεται απαραίτητη η περαιτέρω ανάλυση του τρόπου λειτουργίας αυτών.

## BLAST

Το πρόγραμμα BLAST εισήγαγε διάφορες βελτιώσεις στις αναζητήσεις σε βάσεις δεδομένων, βελτιώνοντας γενικά την ταχύτητα αναζήτησης και τοποθετώντας αυτές τις αναζητήσεις σε ισχυρή στατιστική βάση. Μια καινοτομία που εισήγαγε το BLAST είναι η ιδέα των γειτονικών λέξεων. Αντί να υπάρχει η απαίτηση οι λέξεις να ταιριάζουν ακριβώς, ένα θετικό ταίριασμα για μια λέξη επιτυγχάνεται εάν η λέξη που λαμβάνεται από την υποκείμενη ακολουθία εμφανίζει βαθμολογία τουλάχιστον **T**, όταν γίνεται σύγκριση με μια λέξη από την ακολουθία επερώτησης χρησιμοποιώντας μια μήτρα αντικατάστασης. Αυτή ακριβώς η στρατηγική επιτρέπει το μέγεθος της λέξης (**W**) να κρατηθεί υψηλό (για την ταχύτητα), χωρίς να θυσιαστεί η ευαισθησία. Κατά συνέπεια το **T** γίνεται εκείνη η κρίσιμη παράμετρος η οποία καθορίζει την ταχύτητα και την ευαισθησία ενώ το **W** αλλάζει σπάνια. Αν η τιμή του **T** αυξηθεί, ο αριθμός των βασικών ταιριασμάτων λέξεων θα μειωθεί και το πρόγραμμα θα εκτελεσθεί γρηγορότερα. Η μείωση του **T** επιτρέπει να βρεθούν ακόμα πιο μακρινές σχέσεις [1].

Η εμφάνιση ενός ταιριάσματος λέξεων ακολουθείται από μια προσπάθεια να βρεθεί μια τοπικά βέλτιστη στοίχιση, της οποίας το αποτέλεσμα να είναι τουλάχιστον ίσο με ένα κατώφλι βαθμολογίας **S**. Αυτό ολοκληρώνεται με επαναληπτικές διαδικασίες που επεκτείνουν τη στοίχιση προς τα αριστερά και τα δεξιά, αθροίζοντας βαθμολογίες για τις αντιστοιχίες, τους λανθασμένους συνδυασμούς και την εισαγωγή κενών. Στην πράξη, είναι προτιμότερο να καθοριστεί ένα αναμενόμενο κατώφλι **E**, στο οποίο το πρόγραμμα εσωτερικά μετατρέπει στην κατάλληλη τιμή του **S**. Στις περιοχές όπου τα κατάλοιπα που ταιριάζουν είναι λιγοστά, η αθροιστική βαθμολογία θα αρχίσει να πέφτει. Καθώς οι ποινές για κατάλοιπα που δεν ταιριάζουν και για την εισαγωγή κενών αυξάνουν, είναι όλο και λιγότερο πιθανό ότι η βαθμολογία θα ανέβει και πάλι και θα φτάσει τελικά στη τιμή **S**. Η παρατήρηση αυτή αποτελεί τη βάση για μία πρόσθετη ευριστική ιδέα, σύμφωνα με την οποία η επέκταση ενός ταιριάσματος ολοκληρώνεται όταν η μείωση της βαθμολογίας (σχετικά με τη μέγιστη τιμή που συναντάται) υπερβαίνει την τιμή της ευαισθησίας του **X**. Χρησιμοποιώντας μικρότερη τιμή για το **X** βελτιώνεται η απόδοση, μειώνοντας το χρόνο που καταναλώνεται στις επεκτάσεις βαθμολογιών που δεν έχουν πολλές ελπίδες επιτυχίας, εις βάρος του να χαθούν, περιστασιακά, μερικές πραγματικές στοιχίσεις [1].

## Modeller

Αν και το Modeller έχει εργαλεία για στοίχιση ακολουθιών και εύρεση επιθυμητών στοιχείων από βάσεις δεδομένων, το αρχικό σημείο για την επιτέλεση της διεργασίας του homology modeling είναι ένα αρχείο πολλαπλής στοίχισης ακολουθιών μεταξύ της αλληλουχίας στόχου και του υποστρώματος (template) των πρωτεϊνικών αλληλουχιών. Το Modeller χρησιμοποιεί το υπόστρωμα δομών για να παράγει ένα σετ τοπικών περιορισμών που εφαρμόζονται στην αλληλουχία στόχος. Αυτοί οι περιορισμοί θέτουν όρια, όπως για παράδειγμα στην απόσταση μεταξύ δύο καταλοίπων στο μοντέλο που δημιουργείται, βασιζόμενοι στην απόσταση των δύο ομόλογων καταλοίπων του υποστρώματος δομής. Περιορισμοί τίθενται και σε διάφορες άλλες παραμέτρους όπως τις γωνίες των δεσμών, τις δίεδρες γωνίες κτλ. Εφαρμόζοντας αρκετούς τοπικούς περιορισμούς το Modeller μειώνει αποτελεσματικά τον αριθμό των στερεοδιαμορφώσεων που μπορεί να λάβει το μοντέλο [3].

Η ακριβής μορφή των περιορισμών βασίζεται στις στατιστικές αναλύσεις των διαφορετικών ζευγών των ομόλογων δομών. Αυτές οι αναλύσεις συντελούν στις ποσοτικές περιγραφές διάφορων ιδιοτήτων που μπορεί να ποικίλουν μεταξύ ομόλογων δομών. Το πλήθος της επιτρεπτής διαφοράς, για παράδειγμα μεταξύ της απόστασης δύο ισότιμων ανθράκων σε μια α-έλικα εκφράζεται ως PDF (Propability Dension Function, Συνάρτηση Πυκνότητας Πιθανότητας) [3].

Η χρήση των PDF περιορισμών, στην ομολογία μοντελοποίησης, επιτρέπει την κατασκευή της δομής χωρίς να την αντιγράφει αυτούσια από το υπόστρωμα δομής. Αντιθέτως, η δομή του μοντέλου μπορεί να παρεκκλίνει από το υπόστρωμα δομής κατά τρόπο που να είναι σύμφωνος με τις διαφορές μεταξύ των ομόλογων πρωτεϊνών γνωστής δομής. Για παράδειγμα, έστω ότι μια συγκεκριμένη δίεδρη γωνία έχει στο υπόστρωμα δομής την τιμή $60^o$, οι PDF περιορισμοί επιτρέπουν στην δίεδρη γωνία να λάβει την τιμή $60^o$ συν/πλην κάποιου αριθμού. Ο αριθμός αυτός προσδιορίζεται από την παρατήρηση των ομόλογων δομών και παίρνει την τιμή που είναι πιο πιθανή σύμφωνα με τις τιμές των αριθμών της συνάρτησης πυκνότητας πιθανότητας [3]. Οι τοπικοί περιορισμοί που είναι βασισμένοι στις ομολογίες δεν είναι και οι μόνοι περιορισμοί που τίθενται κατά τη διάρκεια του homology modeling. Εφαρμόζεται επίσης ένα ενεργειακό πεδίο που ελέγχει τη σωστή στερεοχημεία, ώστε να μην παραβιάζονται οι νόμοι της χημείας εις βάρος της ικανοποίησης των τοπικών περιορισμών που προέρχονται από το υπόστρωμα δομής. Όλοι οι χημικοί αλλά και οι τοπικοί περιορισμοί εφαρμόζονται στο μοντέλο κατά τη διάρκεια της μοντελοποίησης [3].

## Autodock

Η εκτίμηση της ελεύθερης ενέργειας επιτυγχάνεται με τον υπολογισμό των ατομικών δυνατοτήτων συγγένειας για κάθε τύπο ατόμου του μόριο-προσδέτη. Στη διαδικασία κατασκευής του χάρτη πλέγματος, η πρωτεΐνη εισέρχεται σε ένα τρισδιάστατο πλέγμα και ένα άτομο δείκτης τοποθετείται σε κάθε σημείο του πλέγματος. Η ενέργεια της αλληλεπίδρασης του συγκεκριμένου άτομου με την πρωτεΐνη προσδιορίζει το εκάστοτε σημείο πλέγματος. Υπολογίζεται έτσι ένα πλέγμα για κάθε τύπο ατόμου του μορίου-προσδέτη, συνήθως άνθρακας, οξυγόνο, άζωτο και υδρογόνο, καθώς και ένα πλέγμα ηλεκτροστατικών δυνατοτήτων. Οι τιμές ενέργειας για κάθε συγκεκριμένη διαμόρφωση του μορίου-προσδέτη υπολογίζονται από τον τριγραμμικό προσδιορισμό των τιμών συγγένειας των οχτώ σημείων πλέγματος που βρίσκονται γύρω από κάθε άτομό του. Οι ηλεκτροστατικές αλληλεπίδρασης υπολογίζονται με παρόμοιο τρόπο, προσδιορίζοντας τις ηλεκτροστατικές δυνατότητες και πολλαπλασιάζοντας αυτές με το φορτίο του ατόμου. Έτσι ο χρόνος υπολογισμού της ενέργειας με τη χρήση πλεγμάτων είναι ανάλογη με τον αριθμό των ατόμων του μορίου-προσδέτη, και ανεξάρτητη από τον αριθμό των ατόμων της πρωτεΐνης [19].

Η εξομοίωση του docking χρησιμοποιεί μια πληθώρα μεθόδων αναζήτησης. Κυρίως για ιστορικούς λόγους θα αναλυθεί η μέθοδος Metropolis που χρησιμοποιείται ακόμα, παράλληλα με καινούργιες μεθόδους αναζήτησης. Κατά τη διάρκεια της εξομοίωσης η πρωτεΐνη παραμένει στατική, ενώ το μόριο-προσδέτης παίρνει τυχαίες θέσεις γύρω από την πρωτεΐνη. Σε κάθε θέση κατά τη διάρκεια της εξομοίωσης υιοθετούνται κάποιοι βαθμοί ελευθερίας για το μόριο-προσδέτης (τοπικές μετατοπίσεις) που αφορούν το κέντρο βαρύτητας, τον προσανατολισμό και την τιμή των δίεδρων γωνιών. Αυτές οι τοπικές μετατοπίσεις ορίζουν μια νέα διαμόρφωση, η ενέργεια της οποίας υπολογίζεται σύμφωνα με τον τρόπο που περιγράφηκε παραπάνω. Η τιμή της νέας ενέργειας συγκρίνεται με την ενέργεια της προηγούμενης διαμόρφωσης, και αν είναι μικρότερη γίνεται αποδεκτή η νέα διαμόρφωση. Αν είναι μεγαλύτερη, η νέα διαμόρφωση γίνεται αποδεκτή ή απορρίπτεται ανάλογα με την παράμετρο **P** που βασίζεται στη θερμοκρασία. Η εξομοίωση συνεχίζεται με επαναλαμβανόμενους κύκλους για να προσδιοριστούν όλες οι πιθανές θέσεις πρόσδεσης [19].

# ΠΑΡΑΡΤΗΜΑ Β

Παρατίθεται στην τελική μορφή ολόκληρος ο κώδικας του προγράμματος.

```
#!/usr/bin/perl -w
use strict;
use CGI qw(:standard);
use Net::FTP;
use File::Basename;

my(
  $query_string,
  $trash,
  $QID,
  $trash2,
  $QIDtrue,
  $cur1,
  $action,
  $encoding,
  $Seq,
  $Begin,
  $Evalue,
  $cur3,
  @count,
  @dock_choices,
  $d_choice,
  @true,
  @check,
  $check,
  $pcheck,
  @pcheck,
  $hmm,
  $d,
  $dt,
  $t,
  $blastfile,
  $logfile,
  @line,
  $queryline,
  $nquery,
  $segment,
  $query,
  $k,
  $line_number,
  @multiqueryline,
  @nmultiquery,
  @multisegment,
  @multievalue,
  @multiquery,
  $ftp,
  $ftp_number,
  $i,
  $sequence,
```

```
  @date,
  $datenew,
  $cur2,
  @count1,
  @jmolscript,
  $Seq1,
  $Seq2,
  $testb,
  $cur4,
  @line1,
  $woot,
  $Start,
  $lig,
  @countone,
  @counttwo,
  @line2,
  @dock_zip,
  @wootty,
  $ligand,
  @wooty,
  $leest,
  @jmolscript,
  );

$query_string = $ENV{'QUERY_STRING'};
($trash, $QID, $trash2) = split (/=/, $query_string);
if (defined($QID)) {
  ($QIDtrue, $trash) = split (/&/, $QID);
  }
unless (defined($QIDtrue)) {
  html1();
  }
else {
  if (-e "/tmp/$QIDtrue/") {
  html2();
  }
  else {
  bail2("Wrong ID entered");
  }
}

sub html1 {
  print header, start_html("HMDoc"); print p("<IMG src='../banner.gif'>");
  print hr();
  ID_form();
  sequence_form();
  if (defined($Seq)) {
    if ($Seq ne '') {
      if (param) {
        if ($Begin eq 'Feeling Lucky') {
          $Seq =~ s/ //g;
          if ($Seq =~ /\//) {
            ID();
            Seq_writer("1");
```

```
            Seq_checker();
            blast_program("1");
            parse_blast("1");
            get_ftp("1");
            modeller_program_feeling_lucky("1");
            java_files();
            }
          else {
            ID();
            Seq_writer("2");
            Seq_checker();
            blast_program("2");
            parse_blast("2");
            get_ftp("1");
            modeller_program_feeling_lucky("2");
            java_files();
            }
          }
        else {
          ID();
          Seq_writer();
          Seq_checker();
          blast_program();
          parse_blast("3");
          get_ftp("2");
          modeller_program_multimodelling();
          java_files();
          }
        }
      }
    else {
      bail2 ("No sequence entered");
      }
    print end_html;
    }
  }

sub html2 {
  print header, start_html("HMDoc"); print p("<IMG src='../banner.gif'>");
  print hr();
  print p("<BODY bgcolor=#FDEEF4>");
  print p("MODELLER RESULTS");
  print p();
  if (-e "/tmp/$QIDtrue/blast.out") {
    java_script("1");
    }
  else {
    java_script("2");
    print p();
    java_script("3");
    }
  print p();
  java_script("4");
  print p();
```

```
  java();
  print p();
  ftp();
  print hr();
  upload_form();
  print hr();
  dock_results();
  print end_html;
  }

sub ID {
  system ("date > /tmp/DATE.txt");
  open DATE , "/tmp/DATE.txt" || bail("$!");
  @date = <DATE>;
  close DATE;
  for ( $i=0; $i <@date; $i++) {
    $datenew .= $date[$i];
    }
  $datenew =~ s/ //g;
  $datenew =~ s/://g;
  chop ($datenew);
  mkdir("/var/ftp/pub/$datenew/",0777);
  mkdir("/var/ftp/pub/$datenew/ligands/",0777);
  mkdir("/tmp/$datenew/",0777);
  }

sub Seq_writer {
  open PIRPROTEIN, "> /tmp/$datenew/UnkP.ali" || bail("$!");
  print PIRPROTEIN "\>P1;UnkP\n";
  print PIRPROTEIN "sequence:UnkP:::::::0.00: 0.00\n";
  print PIRPROTEIN "$Seq";
  print PIRPROTEIN "\*";
  close PIRPROTEIN;
  if ($_[0]=='1') {
    ($Seq1, $Seq2) = split (/\//, $Seq);
    open PROTEINONE, "> /tmp/$datenew/protein1.txt" || bail("$!");
    print PROTEINONE "$Seq1";
    close PROTEINONE;
    open PROTEINTWO, "> /tmp/$datenew/protein2.txt" || bail("$!");
    print PROTEINTWO "$Seq2";
    close PROTEINTWO;
    $Seq =~ s/\///g;
    open PROTEIN, "> /tmp/$datenew/protein.txt" || bail("$!");
    print PROTEIN "$Seq";
    close PROTEIN;
    }
  else {
    open PROTEIN, "> /tmp/$datenew/protein.txt" || bail("$!");
    print PROTEIN "$Seq";
    close PROTEIN;
    }
  }

sub Seq_checker {
```

```
  @true = qw(A C D E F G H I K L M N P Q R S T V W Y /);
  open PROTEIN, "/tmp/protein.txt"
  or die ("Filehandle problems");
  @check = <PROTEIN>;
  close PROTEIN;
  $check = join( '', @check);
  $check =~ s/\s//g;
  $pcheck = $check;
  @check = split( '', $check);
  @pcheck = split( '', $pcheck);
  $hmm='';
  for ( $dt=0; $dt<@check; $dt+=5) {
    $hmm = "$hmm" . " ";
    for ( $t=$dt; $t < $dt+5  && $t < @check; $t++) {
      $hmm= "$hmm" . "$pcheck[$t]";
      }
    }
    for ( $d=0; $d<@check; $d++ ) {
    if ( $check[$d] ne $true[0]  && $check[$d] ne $true[1]  && $check[$d] ne
$true[2]  && $check[$d] ne $true[3] && $check[$d] ne $true[4]
      && $check[$d] ne $true[5]  && $check[$d] ne $true[6]  && $check[$d] ne
$true[7] && $check[$d] ne $true[8]  && $check[$d] ne $true[9]
      && $check[$d] ne $true[10] && $check[$d] ne $true[11] && $check[$d] ne
$true[12] && $check[$d] ne $true[13] && $check[$d] ne $true[14]
      && $check[$d] ne $true[15] && $check[$d] ne $true[16] && $check[$d] ne
$true[17] && $check[$d] ne $true[18] && $check[$d] ne $true[19]) {
      $dt=$d+1;
        print "Sorry the character $check[$d] at place number $dt of the
query you have entered, shown below, is invalid\n";
        print p();
        print "$hmm";
        die ;
      }
    }
  }

sub blast_program {
  if ($_[0]=='1') {
    system("/usr/local/blast/bin/blastall -p blastp   -d
/usr/local/blast/data/pdbaa -i /tmp/$datenew/protein1.txt -o
/tmp/$datenew/blast1.out");
    system("/usr/local/blast/bin/blastall -p blastp -d
/usr/local/blast/data/pdbaa -i /tmp/$datenew/protein2.txt -o
/tmp/$datenew/blast2.out");
  }
  else {
    system("/usr/local/blast/bin/blastall -p blastp -d
/usr/local/blast/data/pdbaa -i /tmp/$datenew/protein.txt -o
/tmp/$datenew/blast.out");
    }
  }

sub parse_blast {
  if ($_[0]=='1') {
    open BLASTFILE, "/tmp/$datenew/blast1.out" || bail("$!");
```

```
    while (<BLASTFILE>) {
      chomp;
      if (/>pdb/) {
        @line=split(/ /,$_);
        $queryline=$line[0];
      ( $trash, $nquery, $segment) = split (/\|/, $queryline);
        last;
        }
      }
    close BLASTFILE;
    $testb=0;
    open BLASTFILETWO, "/tmp/$datenew/blast2.out" || bail("$!");
    while (<BLASTFILETWO>) {
      if (/$nquery/) {
        $testb=1;
      }
    }
    unless ($testb == '1') {
      bail2("Homology modelling cant be completed, not enough homologous
proteins");
      }
    close BLASTFILETWO;
    $nquery="\L$nquery";
    $query=$nquery.$segment;
    }
  elsif ($_[0]=='2') {
    open BLASTFILE, "/tmp/$datenew/blast.out" || bail("$!");
    while (<BLASTFILE>) {
      chomp;
      if (/>pdb/) {
        @line=split(/ /,$_);
        $queryline=$line[0];
      ( $trash, $nquery, $segment) = split (/\|/, $queryline);
        last;
        }
      }
    $nquery="\L$nquery";
    $query=$nquery.$segment;
    close BLASTFILE;
    }
  elsif ($_[0]=='3') {
    $k = 0;
    open BLASTFILE, "/tmp/$datenew/blast.out" || bail("$!");
    while (<BLASTFILE>) {
      chomp;
      if (/pdb\|/) {
        @line=split(/ /,$_);
        $line_number = @line;
        $line_number = $line_number - 1;
        $multiqueryline[$k]=$line[0];
        if ( $line[0] =~ /^>pdb/) {
          last;
          }
        else {
```

```
          ($trash, $nmultiquery[$k], $multisegment[$k]) = split (/\|/,
$multiqueryline[$k]);
           $multievalue[$k] = $line[$line_number];
           $k++;
           }
         }
       }
     for ( $k =0; $k < @nmultiquery; $k++) {
       $nmultiquery[$k] = "\L$nmultiquery[$k]";
       $multiquery[$k] = $nmultiquery[$k].$multisegment[$k];
       }
     close BLASTFILE;
     }
   }

sub get_ftp {
  if ($_[0]=='1') {
    unless (-e "/tmp/pdb$nquery\.ent") {
      $ftp = Net::FTP->new("ftp.rcsb.org", Debug => 0)
        || bail("Cannot connect to ftp.rcsb.org: $@");
      $ftp->login("anonymous",'-anonymous@')
        || bail("Cannot login ", $ftp->message);
      $ftp->cwd("/pub/pdb/data/structures/all/pdb/")
        || bail("Cannot change working directory ", $ftp->message);
      $ftp->get("pdb$nquery\.ent.Z","/tmp/pdb$nquery\.ent.Z")
        || bail("get failed ", $ftp->message);
      $ftp->quit;
      system("gunzip /tmp/pdb$nquery\.ent.Z");
      }
    system ("cp /tmp/pdb$nquery\.ent /tmp/$datenew/pdb$nquery\.ent");
    print p("The proccess of homology modeling has started");
    }
  elsif ($_[0]=='2') {
    $ftp_number = 0;
    for ($i=0; $i<@multievalue; $i++) {
      if ( $multievalue[$i] < $Evalue && $multievalue[$i] ne "") {
        $ftp_number++;
        }
      }
    if ( $ftp_number > 5 ) {
      $ftp_number = 5;
      }
    for ($i=0; $i<$ftp_number; $i++) {
      unless (-e "/tmp/pdb$nmultiquery[$i]\.ent") {
        $ftp = Net::FTP->new("ftp.rcsb.org", Debug => 0)
          || bail("Cannot connect to ftp.rcsb.org: $@");
        $ftp->login("anonymous",'-anonymous@')
          || bail("Cannot login ", $ftp->message);
        $ftp->cwd("/pub/pdb/data/structures/all/pdb/")
          || bail("Cannot change working directory ", $ftp->message);
        $ftp-
>get("pdb$nmultiquery[$i]\.ent.Z","/tmp/pdb$nmultiquery[$i]\.ent.Z")
          || bail("get failed ", $ftp->message);
        $ftp->quit;
```

```
        }
      system("gunzip /tmp/pdb$nmultiquery[$i]\.ent.Z");
      system ("cp /tmp/pdb$nmultiquery[$i]\.ent
/tmp/$datenew/pdb$nmultiquery[$i]\.ent");
      }
    print p("The proccess of homology modeling has started");
    }
  }

sub modeller_program_feeling_lucky {
  $sequence='UnkP';
  open  BUILD_PROFILE , ">/tmp/$datenew/build_profile\.py" || bail("$!");
  print BUILD_PROFILE "from modeller import *\n";
  print BUILD_PROFILE "log.verbose()\n";
  print BUILD_PROFILE "env = environ()\n";
  print BUILD_PROFILE "sdb = sequence_db(env)\n";
  print BUILD_PROFILE
"sdb.read(seq_database_file='/usr/local/blast/data/pdbaa.bin',
seq_database_format='BINARY',\n";
  print BUILD_PROFILE "chains_list='ALL')\n";
  print BUILD_PROFILE "aln = alignment(env)\n";
  print BUILD_PROFILE "aln.append(file='/tmp/$datenew/$sequence\.ali',
alignment_format='PIR', align_codes='ALL')\n";
  print BUILD_PROFILE "prf = aln.to_profile()\n";
  print BUILD_PROFILE "prf.build(sdb, matrix_offset=-450,
rr_file='\${LIB}/blosum62.sim.mat',\n";
  print BUILD_PROFILE "gap_penalties_1d=(-500, -50),
n_prof_iterations=1,\n";
  print BUILD_PROFILE "check_profile=False, max_aln_evalue=0.01)\n";
  print BUILD_PROFILE "prf.write(file='/tmp/$datenew/build_profile.prf')\n";
  print BUILD_PROFILE "aln = prf.to_alignment()\n";
  print BUILD_PROFILE "aln.write(file='/tmp/$datenew/build_profile.ali',
alignment_format='PIR')\n";
  print BUILD_PROFILE "\n";
  close BUILD_PROFILE;
  system("/usr/bin/mod9v1 /tmp/$datenew/build_profile.py");

  open  ALIGNTWOD , ">/tmp/$datenew/align2d\.py" || bail("$!");
  print ALIGNTWOD "from modeller import *\n";
  print ALIGNTWOD "env = environ()\n";
  print ALIGNTWOD "aln = alignment(env)\n";
  if ($_[0]=='1') {
    print ALIGNTWOD "mdl = model(env, file='/tmp/$datenew/pdb$nquery\.ent',
model_segment=('FIRST:A','LAST:B'))\n";
    print ALIGNTWOD "aln.append_model(mdl, align_codes='$nquery',
atom_files='/tmp/$datenew/pdb$nquery\.ent')\n";
    print ALIGNTWOD "aln.append(file='/tmp/$datenew/$sequence\.ali',
align_codes='$sequence')\n";
    print ALIGNTWOD "aln.align2d()\n";
    print ALIGNTWOD "aln.write(file='/tmp/$datenew/$sequence\-$nquery\.ali',
alignment_format='PIR')\n";
    print ALIGNTWOD "aln.write(file='/tmp/$datenew/$sequence\-$nquery\.pap',
alignment_format='PAP')\n";
    }
  elsif ($_[0]=='2') {
```

```
    print ALIGNTWOD "mdl = model(env, file='/tmp/$datenew/pdb$nquery\.ent',
model_segment=('FIRST:$segment','LAST:$segment'))\n";
    print ALIGNTWOD "aln.append_model(mdl, align_codes='$query',
atom_files='/tmp/$datenew/pdb$nquery\.ent')\n";
    print ALIGNTWOD "aln.append(file='/tmp/$datenew/$sequence\.ali',
align_codes='$sequence')\n";
    print ALIGNTWOD "aln.align2d()\n";
    print ALIGNTWOD "aln.write(file='/tmp/$datenew/$sequence\-$query\.ali',
alignment_format='PIR')\n";
    print ALIGNTWOD "aln.write(file='/tmp/$datenew/$sequence\-$query\.pap',
alignment_format='PAP')\n";
    }
  print ALIGNTWOD "\n";
  close ALIGNTWOD;
  system("/usr/bin/mod9v1 /tmp/$datenew/align2d.py");

  open  MODELSINGLE , ">/tmp/$datenew/model-single\.py" || bail("$!");
  print MODELSINGLE "from modeller import *\n";
  print MODELSINGLE "from modeller.automodel import *\n";
  print MODELSINGLE "env = environ()\n";
  if ($_[0]=='1') {
    print MODELSINGLE "a = automodel(env, alnfile='/tmp/$datenew/$sequence\-
$nquery\.ali',\n";
    print MODELSINGLE "knowns='$nquery\', sequence='$sequence')\n";
    }
  elsif ($_[0]=='2') {
    print MODELSINGLE "a = automodel(env, alnfile='/tmp/$datenew/$sequence\-
$query\.ali',\n";
    print MODELSINGLE "knowns='$query\', sequence='$sequence')\n";
    }
  print MODELSINGLE "a.starting_model = 1\n";
  print MODELSINGLE "a.ending_model = 1\n";
  print MODELSINGLE "a.make()\n";
  print MODELSINGLE "\n";
  close MODELSINGLE;
  system("(cd /tmp/$datenew/ ; /usr/bin/mod9v1 /tmp/$datenew/model-
single.py)");

  print p("The proccess of homology modeling has finished");
  system ("mv /tmp/$datenew/UnkP.B99990001.pdb /tmp/$datenew/UnkP.pdb");
  print " Your query ID is $datenew";
  }

sub modeller_program_multimodelling {
  $sequence='UnkP';
  open  SALIGN , ">/tmp/$datenew/salign.py" || bail("$!");
  print SALIGN "from modeller import *\n";
  print SALIGN "log.verbose()\n";
  print SALIGN "env = environ()\n";
  print SALIGN "env.io.atom_files_directory = './:../atom_files/'\n";
  print SALIGN "aln = alignment(env)\n";
  print SALIGN "for (code, chain) in (";
  for ( $i=0 ; $i<$ftp_number ; $i++) {
    if ( $i == $ftp_number-1) {
      print SALIGN "('$nmultiquery[$i]','$multisegment[$i]')";
```

```
      }
    else {
      print SALIGN "('$nmultiquery[$i]', '$multisegment[$i]'),";
      }
    }
  print SALIGN "):\n";
  print SALIGN "  mdl = model(env, file=code, model_segment=('FIRST:'+chain,
'LAST:'+chain))\n";
  print SALIGN "  aln.append_model(mdl, atom_files=code,
align_codes=code+chain)\n";
  print SALIGN "for (weights, write_fit, whole) in (((1., 0., 0., 0., 1.,
0.), False, True),\n";
  print SALIGN "                                    ((1., 0.5, 1., 1., 1.,
0.), False, True),\n";
  print SALIGN "                                    ((1., 1., 1., 1., 1., 0.),
True, False)):\n";
  print SALIGN "  aln.salign(rms_cutoffs=(3.5, 6., 60, 60, 15, 60, 60, 60,
60, 60, 60),\n";
  print SALIGN "             normalize_pp_scores=False,\n";
  print SALIGN "             rr_file='\$(LIB)/as1.sim.mat', overhang=30,\n";
  print SALIGN "             gap_penalties_1d=(-450, -50),\n";
  print SALIGN "             gap_penalties_3d=(0, 3), gap_gap_score=0,
gap_residue_score=0,\n";
  print SALIGN "
dendrogram_file='/tmp/$datenew/fm00495.tree',\n";
  print SALIGN "             alignment_type='tree',
  print SALIGN "             feature_weights=weights,
  print SALIGN "             improve_alignment=True, fit=True,
write_fit=write_fit,\n";
  print SALIGN "             write_whole_pdb=whole, output='ALIGNMENT
QUALITY')\n";
  print SALIGN "aln.write(file='/tmp/$datenew/fm00495.pap',
alignment_format='PAP')\n";
  print SALIGN "aln.write(file='/tmp/$datenew/fm00495.ali',
alignment_format='PIR')\n";
  print SALIGN "aln.salign(rms_cutoffs=(1.0, 6., 60, 60, 15, 60, 60, 60, 60,
60, 60),\n";
  print SALIGN "         normalize_pp_scores=False,
rr_file='\$(LIB)/as1.sim.mat', overhang=30,\n";
  print SALIGN "         gap_penalties_1d=(-450, -50), gap_penalties_3d=(0,
3),\n";
  print SALIGN "         gap_gap_score=0, gap_residue_score=0,
dendrogram_file='1is3A.tree',\n";
  print SALIGN "         alignment_type='progressive',
feature_weights=[0]*6,\n";
  print SALIGN "         improve_alignment=False, fit=False,
write_fit=True,\n";
  print SALIGN "         write_whole_pdb=False, output='QUALITY')\n";
  close SALIGN;
  system("(cd /tmp/$datenew/ ; /usr/bin/mod9v1 /tmp/$datenew/salign.py)");

  open  ALIGNTWODMULT , ">/tmp/$datenew/align2dmult.py" || bail("$!");
  print ALIGNTWODMULT "from modeller import *\n";
  print ALIGNTWODMULT "log.verbose()\n";
  print ALIGNTWODMULT "env = environ()\n";
  print ALIGNTWODMULT
```

```
"env.libs.topology.read(file='\$(LIB)/top_heav.lib')\n";
  print ALIGNTWODMULT "aln = alignment(env)\n";
  print ALIGNTWODMULT "aln.append(file='/tmp/$datenew/fm00495.ali',
align_codes='all')\n";
  print ALIGNTWODMULT "aln_block = len(aln)\n";
  print ALIGNTWODMULT "aln.append(file='/tmp/$datenew/$sequence\.ali',
align_codes='$sequence')\n";
  print ALIGNTWODMULT "aln.salign(output='', max_gap_length=20,\n";
  print ALIGNTWODMULT "           gap_function=True,\n";
  print ALIGNTWODMULT "           alignment_type='PAIRWISE',
align_block=aln_block,\n";
  print ALIGNTWODMULT "           feature_weights=(1., 0., 0., 0., 0., 0.),
overhang=0,\n";
  print ALIGNTWODMULT "           gap_penalties_1d=(-450, 0),\n";
  print ALIGNTWODMULT "           gap_penalties_2d=(0.35, 1.2, 0.9, 1.2,
0.6, 8.6, 1.2, 0., 0.),\n";
  print ALIGNTWODMULT "           similarity_flag=True)\n";
  print ALIGNTWODMULT "aln.write(file='/tmp/$datenew/$sequence\-mult.ali',
alignment_format='PIR')\n";
  print ALIGNTWODMULT "aln.write(file='/tmp/$datenew/$sequence\-mult.pap',
alignment_format='PAP')\n";
  close ALIGNTWODMULT;
  system ("(cd /tmp/$datenew/ ; /usr/bin/mod9v1
/tmp/$datenew/align2dmult.py)");

  open  MODELMULT , ">/tmp/$datenew/modelmult.py" || bail("$!");
  print MODELMULT "from modeller import *\n";
  print MODELMULT "from modeller.automodel import *\n";
  print MODELMULT "env = environ()\n";
  print MODELMULT "a = automodel(env, alnfile='$sequence\-mult.ali',\n";
  print MODELMULT "              knowns=(";
  for ( $i=0 ; $i<$ftp_number ; $i++) {
    if ( $i == $ftp_number-1) {
      print MODELMULT "'$multiquery[$i]'";
      }
    else {
      print MODELMULT "'$multiquery[$i]',";
      }
    }
  print MODELMULT "), sequence='$sequence')\n";
  print MODELMULT "a.starting_model = 1\n";
  print MODELMULT "a.ending_model = 1\n";
  print MODELMULT "a.make()\n";
  close MODELMULT;
  system ("(cd /tmp/$datenew/ ; /usr/bin/mod9v1
/tmp/$datenew/modelmult.py)");

  print p("The proccess of homology modeling has finished");
  system ("mv /tmp/$datenew/UnkP.B99990001.pdb /tmp/$datenew/UnkP.pdb");
  print " Your query ID is $datenew";
  }

sub java_files {
  open HTML, ">/tmp/$datenew/start.html" || bail("$!");
  print HTML "PROTEIN VIEWER
```

```
  <html>
   <head>
    <title>protein viewer</title>
    <script src=\"./jmol/Jmol.js\" type=\"text/javascript\"></script>
    </head>
   <body>
   <form>
      <script type=\"text/javascript\">
        jmolInitialize(\"./jmol\");
        jmolApplet(400, \"load ./jmol/structures/$datenew.pdb\");
    </script>
    </form>
    <CENTER>
    <A HREF=\"javascript:window.close()\">Close this window</A>
    </CENTER>
   </body>
  </html>";
  close HTML;
  open JMOLSCRIPT, ">/tmp/$datenew/jmolscript.txt" || bail("$!");
  print JMOLSCRIPT "<A HREF=\"javascript:void(0)\"

onclick=\"window.open('../$datenew.html','welcome','width=440,height=490')\"
>
    <font color=black>View the structure</font></A>";
  close JMOLSCRIPT;
  system ("mv /tmp/$datenew/start.html /var/www/html/$datenew.html");
  system ("cp /tmp/$datenew/UnkP.pdb
/var/www/html/jmol/structures/$datenew.pdb");
  system ("(cd /tmp/$datenew ; /usr/bin/zip -q -r modeller_results . -i
UnkP.* *.log blast*.out *.ali *.pap *.prf protein*.txt)");
  chmod (0666,"/tmp/$datenew/modeller_results.zip");
  system ("mv /tmp/$datenew/modeller_results.zip
/var/ftp/pub/$datenew/modeller_results.zip");
  }

sub java_script {
  if (-e "/tmp/$QIDtrue/blast.out") {
    open BLASTFILE, "/tmp/$QIDtrue/blast.out" || bail("$!");
    @count = <BLASTFILE>;
    close BLASTFILE;
    }
  else {
    open BLASTFILEONE, "/tmp/$QIDtrue/blast1.out" || bail("$!");
    @countone= <BLASTFILEONE>;
    close BLASTFILE;
    open BLASTFILETWO, "/tmp/$QIDtrue/blast2.out" || bail("$!");
    @counttwo= <BLASTFILETWO>;
    close BLASTFILETWO;
    }
  $leest = </tmp/$QIDtrue/*-*.pap>;
  if (defined($leest)) {
    open ALI, "$leest" || bail("$!");
    @count1 = <ALI>;
    close ALI;
```

```
    }
  print q(<script type="text/javascript">);
  print q(<!--
        function advance(obj) {
          var child;
          child = obj.firstChild;
          sibling = child.nextSibling;
          if(child.style.display != 'none') {
            child.style.display = 'none';
            sibling.style.display = 'inline';
            }
          else {
          child.style.display = 'inline';
          sibling.style.display = 'none';
          }
        }
        // -->);
  print q(</script>);
  if ($_[0]=='1') {
    print q(<table align="center" border="1" width="90%" cellpadding="3"
cellspacing="0">
            <tr><td ><b>Blast Output</b></td></tr><tr><td
onclick="advance(this)" class="quote"
            style="cursor: pointer;"><span style="display: none">);
            print q(<pre>);
            print ("@count");
            print q(</pre>);
    print q(</span><span>Click to expand...</span></td></tr></table>);
    }
  elsif ($_[0]=='2') {
    print q(<table align="center" border="1" width="90%" cellpadding="3"
cellspacing="0">
            <tr><td ><b>Blast Output for first chain</b></td></tr><tr><td
onclick="advance(this)" class="quote"
            style="cursor: pointer;"><span style="display: none">);
            print q(<pre>);
            print ("@countone");
            print q(</pre>);
    print q(</span><span>Click to expand...</span></td></tr></table>);
    }
  elsif ($_[0]=='3') {
    print q(<table align="center" border="1" width="90%" cellpadding="3"
cellspacing="0">
            <tr><td ><b>Blast Output for second chain</b></td></tr><tr><td
onclick="advance(this)" class="quote"
            style="cursor: pointer;"><span style="display: none">);
            print q(<pre>);
            print ("@counttwo");
            print q(</pre>);
    print q(</span><span>Click to expand...</span></td></tr></table>);
    }
  elsif ($_[0]=='4') {
    print q(<table align="center" border="1" width="90%" cellpadding="3"
cellspacing="0">
            <tr><td ><b>Alignment Output</b></td></tr><tr><td
```

```
onclick="advance(this)" class="quote"
            style="cursor: pointer;"><span style="display: none">);
            print q(<pre>);
            print ("@count1");
            print q(</pre>);
    print q(</span><span>Click to expand...</span></td></tr></table>);
    }
  }

sub java {
  open JMOLSCRIPT, "/tmp/$QIDtrue/jmolscript.txt" || bail("$!");
  @jmolscript=<JMOLSCRIPT>;
  close JMOLSCRIPT;
  print "@jmolscript";
  }

sub upload_form {
  $k=0;
  system ("(cd /var/ftp/pub/$QIDtrue/ligands ; /bin/ls >
list_of_files.txt)");
  open LIST, "/var/ftp/pub/$QIDtrue/ligands/list_of_files.txt";
  while (<LIST>) {
    if (/pdb/) {
      @line1=$_;
      $line_number = @line1;
      $line_number = $line_number - 1;
      $dock_choices[$k]=$line1[0];
      $k++;
      }
    }
    close LIST;
  my $upload_dir = "/var/ftp/pub/$QIDtrue/ligands";
  $cur4 = new CGI;
  print $cur4->startform(-method=>'POST',
                         -action=>$action,
                         -enctype=>'multipart/form-data');
  print p("Files to upload");
  print $cur4->filefield(-name=>'uplfile',
                         -default=>'starting value',
                         -size=>50,
                         -maxlength=>80);
  print p();
  print $cur4->submit(-name=>'start',
                      -value=>'Upload Files');
  print hr();
  print p ("DOCK AREA");
  print $cur4->start_form(-method=>'POST',
                          -action=>$action,
                          -enctype=>$encoding);
  print p("Perform docking with molecule");
  print $cur4->popup_menu('choice',\@dock_choices);
  print p();
  print $cur4->submit(-name=>'start',
                      -value=>'Start Docking');
```

```
  print $cur4->submit(-name=>'start',
                      -value=>'MultiDock');
  print $cur4->endform;
  $d_choice=(param('choice'));
  $Start=(param('start'));
  my $filename = (param('uplfile'));
  if (defined($Start)) {
    if (param) {
      if ($Start eq 'Upload Files' && $filename ne '') {
        my ( $name, $path, $extension ) = fileparse ( $filename, '\..*' );
        $filename = $name . $extension;
        $filename =~ tr/ /_/;
        my $upload_filehandle = (upload("uplfile"));
        open ( UPLOADFILE, ">$upload_dir/$filename", ) || bail("$!");
        chmod(0666,"/$upload_dir/$filename");
        binmode UPLOADFILE;
        while ( <$upload_filehandle> ) {
          print UPLOADFILE;
          }
        close UPLOADFILE;
        }
      elsif ($Start eq 'Start Docking') {
        dock("1");
        }
      elsif ($Start eq 'MultiDock') {
        dock("2");
        }
      }
    }
  }

sub dock {
  $k=0;
  system ("(cd /var/ftp/pub/$QIDtrue/ligands ; /bin/ls >
list_of_files.txt)");
  open LIST, "/var/ftp/pub/$QIDtrue/ligands/list_of_files.txt";
  while (<LIST>) {
    if (/pdb/) {
      @line1=$_;
      $line_number = @line1;
      $line_number = $line_number - 1;
      $dock_choices[$k]=$line1[0];
      $k++;
      }
    }
    close LIST;
  if ($_[0]=='1') {
    if ($d_choice ne '') {
    $lig = $d_choice;
    $lig =~ s/.pdb//;
    unless (-e "/tmp/$QIDtrue/UnkP_$lig/") {
      mkdir ("/tmp/$QIDtrue/UnkP_$lig/",0777);
      system ("cp /var/ftp/pub/$QIDtrue/ligands/$d_choice
/tmp/$QIDtrue/UnkP_$lig/$d_choice");
```

```
      system ("cp /tmp/$QIDtrue/UnkP.pdb /tmp/$QIDtrue/UnkP_$lig/UnkP.pdb");
      print hr();
      print p();
      print ("The autodock proccess with $lig ligand has started....");
      }
    unless (-e "/tmp/$QIDtrue/UnkP_$lig/$lig\.pdbqt") {
      system ("( cd /tmp/$QIDtrue/UnkP_$lig/ ; /usr/local/MGLTools-
1.4.6/share/bin/pythonsh /usr/local/MGLTools-
1.4.6/MGLToolsPckgs/AutoDockTools/Utilities24/prepare_ligand4.py -l
/tmp/$QIDtrue/UnkP_$lig/$d_choice)");
      }
    unless (-e "/tmp/$QIDtrue/UnkP_$lig/UnkP.pdbqt") {
      system ("( cd /tmp/$QIDtrue/UnkP_$lig/ ; /usr/local/MGLTools-
1.4.6/share/bin/pythonsh /usr/local/MGLTools-
1.4.6/MGLToolsPckgs/AutoDockTools/Utilities24/prepare_receptor4.py -r
/tmp/$QIDtrue/UnkP_$lig/UnkP.pdb)");
      }
    unless (-e "/tmp/$QIDtrue/UnkP_$lig/UnkP.gpf") {
      system ("( cd /tmp/$QIDtrue/UnkP_$lig/ ; /usr/local/MGLTools-
1.4.6/share/bin/pythonsh /usr/local/MGLTools-
1.4.6/MGLToolsPckgs/AutoDockTools/Utilities24/prepare_gpf4.py -l
/tmp/$QIDtrue/UnkP_$lig/$lig\.pdbqt -r
/tmp/$QIDtrue/UnkP_$lig/UnkP.pdbqt)");
      }
    unless (-e "/tmp/$QIDtrue/UnkP_$lig/$lig\_UnkP.dpf") {
      system ("( cd /tmp/$QIDtrue/UnkP_$lig/ ; /usr/local/MGLTools-
1.4.6/share/bin/pythonsh /usr/local/MGLTools-
1.4.6/MGLToolsPckgs/AutoDockTools/Utilities24/prepare_dpf4.py -l
/tmp/$QIDtrue/UnkP_$lig/$lig\.pdbqt -r
/tmp/$QIDtrue/UnkP_$lig/UnkP.pdbqt)");
      }
    unless (-e "/tmp/$QIDtrue/UnkP_$lig/UnkP.glg") {
      system ("( cd /tmp/$QIDtrue/UnkP_$lig/ ; /usr/local/autodock/autogrid4
-p /tmp/$QIDtrue/UnkP_$lig/UnkP.gpf -l /tmp/$QIDtrue/UnkP_$lig/UnkP.glg)");
      }
    unless (-e "/tmp/$QIDtrue/UnkP_$lig/$lig\_UnkP.dlg") {
      system ("( cd /tmp/$QIDtrue/UnkP_$lig/ ; /usr/local/autodock/autodock4
-p /tmp/$QIDtrue/UnkP_$lig/$lig\_UnkP.dpf -l
/tmp/$QIDtrue/UnkP_$lig/$lig\_UnkP.dlg)");
      system ("(cd /tmp/$QIDtrue ; /usr/bin/zip -q -r dock_$lig\_results . -
i UnkP*.map* *.glg *.gpf *.dpf *.dlg)");
      chmod (0666,"/tmp/$QIDtrue/dock_$lig\_results.zip");
      system ("mv /tmp/$QIDtrue/dock_$lig\_results.zip
/var/ftp/pub/$QIDtrue/dock_$lig\_results.zip");
      my $woot1='<A
HREF='.'"ftp://anonymous:@127.0.0.1/pub/'.$QIDtrue.'/dock_'.$lig.'_results.z
ip"<font color=black>Download dock resuslts</font></A>';
      print ("and finished, $woot1");
      }
    }
    else {
    bail2("No ligand chosen");
    }
  }
  elsif ($_[0]=='2') {
    if ($d_choice ne '') {
```

```
      print hr();
      for ( $i=0 ; $i < @dock_choices ; $i++) {
        $lig = $dock_choices[$i];
        $lig =~ s/.pdb//;
        chomp ($lig);
        unless (-e "/tmp/$QIDtrue/UnkP_$lig/") {
          mkdir ("/tmp/$QIDtrue/UnkP_$lig/",0777);
          system ("cp /var/ftp/pub/$QIDtrue/ligands/$lig\.pdb
/tmp/$QIDtrue/UnkP_$lig/$lig\.pdb");
          system ("cp /tmp/$QIDtrue/UnkP.pdb
/tmp/$QIDtrue/UnkP_$lig/UnkP.pdb");
          print p();
          print ("The autodock proccess with $lig ligand has started....");
          }
        unless (-e "/tmp/$QIDtrue/UnkP_$lig/$lig\.pdbqt") {
          system ("( cd /tmp/$QIDtrue/UnkP_$lig/ ; /usr/local/MGLTools-
1.4.6/share/bin/pythonsh /usr/local/MGLTools-
1.4.6/MGLToolsPckgs/AutoDockTools/Utilities24/prepare_ligand4.py -l
/tmp/$QIDtrue/UnkP_$lig/$dock_choices[$i])");
          }
        unless (-e "/tmp/$QIDtrue/UnkP_$lig/UnkP.pdbqt") {
          system ("( cd /tmp/$QIDtrue/UnkP_$lig/ ; /usr/local/MGLTools-
1.4.6/share/bin/pythonsh /usr/local/MGLTools-
1.4.6/MGLToolsPckgs/AutoDockTools/Utilities24/prepare_receptor4.py -r
/tmp/$QIDtrue/UnkP_$lig/UnkP.pdb)");
          }
        unless (-e "/tmp/$QIDtrue/UnkP_$lig/UnkP.gpf") {
          system ("( cd /tmp/$QIDtrue/UnkP_$lig/ ; /usr/local/MGLTools-
1.4.6/share/bin/pythonsh /usr/local/MGLTools-
1.4.6/MGLToolsPckgs/AutoDockTools/Utilities24/prepare_gpf4.py -l
/tmp/$QIDtrue/UnkP_$lig/$lig\.pdbqt -r
/tmp/$QIDtrue/UnkP_$lig/UnkP.pdbqt)");
          }
        unless (-e "/tmp/$QIDtrue/UnkP_$lig/$lig\_UnkP.dpf") {
          system ("( cd /tmp/$QIDtrue/UnkP_$lig/ ; /usr/local/MGLTools-
1.4.6/share/bin/pythonsh /usr/local/MGLTools-
1.4.6/MGLToolsPckgs/AutoDockTools/Utilities24/prepare_dpf4.py -l
/tmp/$QIDtrue/UnkP_$lig/$lig\.pdbqt -r
/tmp/$QIDtrue/UnkP_$lig/UnkP.pdbqt)");
          }
        unless (-e "/tmp/$QIDtrue/UnkP_$lig/UnkP.glg") {
          system ("( cd /tmp/$QIDtrue/UnkP_$lig/ ;
/usr/local/autodock/autogrid4 -p /tmp/$QIDtrue/UnkP_$lig/UnkP.gpf -l
/tmp/$QIDtrue/UnkP_$lig/UnkP.glg)");
          }
        unless (-e "/tmp/$QIDtrue/UnkP_$lig/$lig\_UnkP.dlg") {
          system ("( cd /tmp/$QIDtrue/UnkP_$lig/ ;
/usr/local/autodock/autodock4 -p /tmp/$QIDtrue/UnkP_$lig/$lig\_UnkP.dpf -l
/tmp/$QIDtrue/UnkP_$lig/$lig\_UnkP.dlg)");
          system ("(cd /tmp/$QIDtrue ; /usr/bin/zip -q -r dock_$lig\_results
. -i UnkP*.map* *.glg *.gpf *.dpf *.dlg)");
          chmod (0666,"/tmp/$QIDtrue/dock_$lig\_results.zip");
          system ("mv /tmp/$QIDtrue/dock_$lig\_results.zip
/var/ftp/pub/$QIDtrue/dock_$lig\_results.zip");
          $wooty[$i]='<A
HREF='.'"ftp://anonymous:@127.0.0.1/pub/'.$QIDtrue.'/dock_'.$lig.'_results.z
ip"<font color=black>Download dock resuslts</font></A>';
```

```
          print ("and finished, ");
          print ($wooty[$i]);
          }
        }
      }
    else {
      bail2("No ligands chosen");
      }
    }
  }

sub ftp {
  $woot='<A HREF='.'"ftp://anonymous:@127.0.0.1/pub/'.
$QIDtrue.'/modeller_results.zip"<font color=black>Download Modeller
resuslts</font></A>';
  print p($woot);
  }

sub dock_results {
  $k=0;
  system ("(cd /var/ftp/pub/$QIDtrue/ ; /bin/ls >
list_of_zipped_files.txt)");
  open LISTZIP, "/var/ftp/pub/$QIDtrue/list_of_zipped_files.txt";
  while (<LISTZIP>) {
    if (/dock/) {
      @line2=$_;
      $line_number = @line2;
      $line_number = $line_number - 1;
      $dock_zip[$k]=$line2[0];
      $k++;
      }
    }
  close LISTZIP;
  for ($i=0; $i < @dock_zip ; $i++) {
    chomp ($dock_zip[$i]);
    ($trash,$ligand,$trash2) = split (/_/, $dock_zip[$i]);
    $wootty[$i]='<A
HREF='.'"ftp://anonymous:@127.0.0.1/pub/'.$QIDtrue.'/'.$dock_zip[$i].'"<font
color=black>Download dock resuslts with '.$ligand.' ligand</font></A>';
    print ($wootty[$i]);
    print p();
    }
  }

sub bail {
  my $error = "@_";
  print h1 ("Error"), p($error), end_html;
  die $error;
  }

sub bail2 {
  if ($_[0] eq 'Wrong ID entered') {
    print header, start_html("Error");
    }
```

```
my $error2 = "@_";
print h1 ("Notification"), p($error2), end_html;
die $error2;
}
```

# ΒΙΒΛΙΟΓΡΑΦΙΑ


[1] Andreas D. Baxevanis & B. F. Francis Ouellette, *Bioinformatics: A Practical Guide to the Analysis of Genes and Proteins (Second Edition)*, Canada, John Wiley & Sons 2001.

[2] Pierre Baldi & Soren Brunek, *Bioinformatics: The Machine Learning Approach (Second Edition)*, USA, MIT Press 2001.

[3] Cynthia Gibas & Per Jambeck, *Developing Bioinformatics Computer Skills*, USA, O'Reilly & Associates 2001.

[4] Arthur M. Lesk, *Introduction to Bioinformatics*, Oxford, UK, Oxford University Press 2002.

[5] Jean-Michel Claverie & Cedric Notredame, *Bioinformatics for Dummies (Second Edition)*, USA, Wiley Publishing 2007.

[6] Andreas D. Baxevanis, Daniel B. Davison, Roderic D. M. Page, Gregory A. Petsko, Lincoln D. Stein & Gary D. Stormo, *Current Protocols in Bioinformatics*, USA, John Wiley & Sons 2002.

[7] Sandor Suhai, *Theoretical and Computational Methods in Genome Research*, USA, Plenum Press 1997.

[8] Oliver S. Jovanovic, *Introduction to Computational Biology*, Lecture: Columbia University, Department of Microbiology, Fall 2007.

[9] Zhongguo Chen, Ying Li, Elizabeth Chen, Dawn L. Hall, Paul L. Darke, Chris Culberson, Jules A. Shafer & Lawrence C. Kuo, *Crystal Structure at 1.9-A$^o$ Resolution of Human Immunodeficiency Virus (HIV) I1 Protease Gomplexed with L-735,524, an Orally Bioavailable Inhibitor of the HlV Proteases*, USA, The Journal of Biological Chemistry Vol.269 No.42 Issue of October 21 pp.26344-26348, The American Society for Biochemistry and Molecular Biology 1994.

[10] Randal L. Scwartz & Tom Christiansen, *Learning Perl (Second Edition)*, USA, O'Reilly & Associates 1997.

[11] Larry Wall, Tom Christiansen & Randal L. Scwartz, *Programming Perl (Second Edition)*, USA, O'Reilly & Associates 1996.

[12] James Tisdall, *Beggining Perl for Bioinformatics*, USA, O'Reilly & Associates 2001.

[13] James Tisdall, *Mastering Perl for Bioinformatics*, USA, O'Reilly & Associates 2003.

[14] Eugene Eric Kim, *CGI Developer's Guide*, USA, Sams.net Publishing 1996.

[15] Shishir Gundavaram, *CGI Programming on the World Wide Web*, USA, O'Reilly & Associates 1996.

[16] David A. Crowder & Andrew Bailey, *Creating Web Sites Bible (Second Edition)*, USA, Wiley Publishing 2004.

[17] Danny Goodman, *JavaScript Bible (Gold Edition)*, USA, Hungry Minds 2001.

[18] David Flannagan, *JavaScript: The Definitive Guide*, USA, O'Reilly & Associates 1997.

[19] Morris, G. M., Goodsell, D. S., Halliday, R.S., Huey, R., Hart, W. E., Belew, R. K. and Olson, A. J. (1998), *J. Computational Chemistry*, **19**: 1639-1662. "Automated Docking Using a Lamarckian Genetic Algorithm and and Empirical Binding Free Energy Function".

[20] Ruth Huey & Garret M. Morris, *Using Autodock 4 with AutoDock Tools: A tutorial*, USA, E-Book 2008.

[21] http://www.ncbi.nlm.nih.gov/blast/Blast.cgi (BLAST Program)

[22] http://salilab.org/modeller/ (Modeller Program)

[23] http://jmol.sourceforge.net/ (Jmol Program)

[24] http://autodock.scripps.edu/ (Autodock Program)

[25] http://www.rcsb.org/pdb/home/home.do (Protein DataBase)