Εθνικό Μετσόβιο Πολυτεχνείο
Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών
Τομέας Τεχνολογίας Πληροφορικής και Υπολογιστών

Εντοπισμός L-type RNA pseudoknot με τεχνικές συντακτικής αναγνώρισης προτύπων

Διπλωματική Εργασία

του

ΚΟΡΟΥΛΗ ΧΡΗΣΤΟΥ

Επιβλέπων: Παναγιώτης Τσανάκας, Καθηγητής Ε.Μ.Π.

Αθήνα, Απρίλιος 2023

Εθνικό Μετσόβιο Πολυτεχνείο
Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών
Τομέας Τεχνολογίας Πληροφορικής και Υπολογιστών

Εντοπισμός L-type RNA pseudoknot με τεχνικές συντακτικής αναγνώρισης προτύπων

Διπλωματική Εργασία

του

ΚΟΡΟΥΛΗ ΧΡΗΣΤΟΥ

Επιβλέπων: Παναγιώτης Τσανάκας, Καθηγητής Ε.Μ.Π.

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 5η Απριλίου 2023.

(Υπογραφή)	*(Υπογραφή)*	*(Υπογραφή)*
.....	.....	.....
Παναγιώτης Τσανάκας	Ανδρέας-Γ. Σταφυλοπάτης	Γεώργιος Ματσόπουλος
Καθηγητής Ε.Μ.Π.	Καθηγητής Ε.Μ.Π.	Καθηγητής Ε.Μ.Π.

Αθήνα, Απρίλιος 2023

Εθνικό Μετσόβιο Πολυτεχνείο
Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών
Τομέας Τεχνολογίας Πληροφορικής και Υπολογιστών

ΧΡΗΣΤΟΣ ΚΟΡΟΥΛΗΣ
Διπλωματούχος Ηλεκτρολόγος Μηχανικός και Μηχανικός Υπολογιστών Ε.Μ.Π.

(Υπογραφή)

.....
Χ. ΚΟΡΟΥΛΗΣ

Απρίλιος 2023

Περίληψη

Τα μόρια του RNA διαδραματίζουν σημαντικό ρόλο σε όλα τα κύτταρα των ζωντανών οργανισμών. Επιπλέον, το μεγαλύτερο μέρος του ανθρώπινου γονιδιώματος μετατρέπεται σε μη κωδικοποιητικό RNA του οποίου η λειτουργία είναι άγνωστη. Ωστόσο, για να αποκρυπτογραφήσουμε τη λειτουργία του RNA, είναι σημαντικό να διερευνήσουμε τη δομή του. Επιπροσθέτως, η παρουσία του ιού SARS-CoV-2, ο οποίος έπληξε μεγάλο μέρος του παγκόσμιου πληθυσμού, κατέστησε ακόμα πιο σημαντική τη γνώση της δομής του RNA. Ένα από τα βήματα στην αποσαφήνιση της δομής του RNA αποτελεί ο ακριβής υπολογισμός της δευτεροταγούς δομής του. Ιστορικά, διάφορες προσεγγίσεις πρόβλεψης της δευτεροταγούς δομής του έχουν αποκλείσει τη δομή του ψευδοκόμβου καθώς θεωρείται ότι το συνολικό πρόβλημα γίνεται ακόμα πιο σύνθετο και πιο χρονοβόρο ως προς τη λύση του. Ωστόσο, η παράλειψη αυτής της δομής από διάφορες προσεγγίσεις για το συγκεκριμένο πρόβλημα περιορίζει τη χρησιμότητα τους ως προς την αποδοτική επίλυση του.

Η παρούσα διπλωματική εργασία συνεισφέρει στη διερεύνηση της δευτεροταγούς δομής του RNA καθώς επιλύει το πρόβλημα της πρόβλεψης μίας από τις βασικές συνιστώσες της, του ψευδοκόμβου, και συγκεκριμένα του ψευδοκόμβου τύπου L. Το συνολικό πρόβλημα αντιμετωπίζεται αρχικά ως ένα πρόβλημα ανάλυσης της ακολουθίας του RNA και στη συνέχεια ως πρόβλημα βελτιστοποίησης βασιζόμενο στην ελαχιστοποίηση της ελεύθερης ενέργειας αναδίπλωσης της δομής και στη μεγιστοποίηση του αριθμού των ζευγαριών μεταξύ των βάσεων. Σε οποιαδήποτε ακολουθία εισόδου η δομή αυτή μπορεί να εντοπίζεται πολλές φορές γι' αυτό και μέσω αυτής της ανάλυσης προσπαθούμε να βρούμε τη μία εξ' αυτών η οποία θα αποτελεί βέλτιστη λύση. Χρησιμοποιούμε τόσο τη συντακτική αναγνώριση προτύπων εισάγοντας μία γραμματική χωρίς συμφραζόμενα με μεγάλη εκφραστική ικανότητα όσο και έναν άπληστο αλγόριθμο δυναμικού προγραμματισμού προκειμένου να επιλύσουμε το συγκεκριμένο πρόβλημα. Η συνολική διαδικασία η οποία υλοποιήθηκε παρουσιάζει πολυωνυμική πολυπλοκότητα. Ο σύνδεσμος για τον πλήρη κώδικα αυτής της υλοποίησης παρατίθεται στη βιβλιογραφία.[1].

Λέξεις Κλειδιά

ριβονουκλεϊκό οξύ, δευτεροταγής δομή, συντακτική αναγνώριση προτύπων, ψευδοκόμβος, γραμματικές χωρίς συμφραζόμενα

Abstract

The molecules of RNA play an important role for all of the cells of living organisms. Furthermore, the major part of the human genome is converted to non-coding RNA, whose function is unknown. However, to decode RNA's functions, it is important to explore its structure. Additionally, the presence of SARS-CoV-2 virus, which has infected a huge part of the planet's population, establishes even more the importance of RNA's structure. One step towards the clarification of RNA's structure, is the exact calculation of its secondary structure. Historically speaking, different predictive approaches of its secondary structure have excluded the structure of pseudoknot because it is considered the whole problem becomes even more complex and time consuming, towards finding a solution. Nevertheless, the omission of this structure from different approaches for this specific problem, limits their usefulness as to finding an efficient solution.

The presence of diplomacy has contributed to the investigation of RNA's secondary structure, as to solving the problem of predicting one of its basic components, the pseudoknot, and in particular the pseudoknot of type L. The whole problem is firstly confronted as an analytical problem of RNA's sequencing, and furthermore as an optimization problem based on the minimization of free energy of the structure and the maximization of the number of pairs between the bases. In any incoming stream, this structure can be often located, and therefore from this analysis, we have tried to find one of them which will constitute the optimal solution.We utilize the syntactic pattern recognition methodology by introducing a context free grammar with large expressive ability, as well as one greedy algorithm of dynamic programming in order to solve the specific problem. The entire process which was implemented, presents polynomial complexity. The link for the whole code of this implementation is cited in the bibliography[1].

Keywords

RNA , secondary structure, syntactic pattern recognition, pseudoknot, context-free grammars

Στην οικογένεια μου

Ευχαριστίες

Αρχικά θα ήθελα να ευχαριστήσω τον καθηγητή κ. Παναγιώτη Τσανάκα ο οποίος εμπιστεύτηκε σε μένα αυτό το πολύ ενδιαφέρον θέμα. Επιπλέον, θα ήθελα να ευχαριστήσω τον κ. Χρήστο Παυλάτο ,τον κ.Ευάγγελο Μακρή και τον κ.΄Αγγελο Κολαΐτη με τη βοήθεια των οποίων ολοκλήρωσα τη συγκεκριμένη εργασία.Τέλος, θα ήθελα να ευχαριστήσω τους γονείς μου, Γιώργο και Ελένη, την αδερφή μου, Πηνελόπη-Μαρία, και τους φίλους μου οι οποίοι ήταν δίπλα μου όλα αυτά τα χρόνια και χωρίς την παρουσία και τη στήριξη των οποίων δε θα είχα καταφέρει να ολοκληρώσω τις σπουδές μου.

Αθήνα, Απρίλιος 2023

ΧΡΗΣΤΟΣ ΚΟΡΟΥΛΗΣ

Περιεχόμενα

Κατάλογος σχημάτων

Κεφάλαιο 1

Εισαγωγή

Το μόριο του RNA έχει σημαντικό ρόλο σε πολλές βιολογικές διαδικασίες καθώς είναι η ενδιάμεση αναπαράσταση της ροής πληροφορίας από το DNA στις πρωτεΐνες.Η δομή του RNA συχνά απεικονίζεται ως μια διδιάστατη αναπαράσταση των νουκλεοτιδίων που ζευγαρώνουν μεταξύ τους (αδενίνη-ουρακίλη, κυτοσίνη-γουανίνη και γουανίνη-ουρακίλη) η οποία είναι γνωστή ως δευτεροταγής δομή και συμβάλλει στην τρισδιάστατη αναπαράσταση του που αποκαλείται τριτοταγής δομή. Ο σημαντικός ρόλος του RNA στην έκφραση των πρωτεϊνών μαζί με τη συνεισφορά του σε λειτουργίες όπως η ρύθμιση της γενετικής έκφρασης και η κατάλυση καθιστούν απαραίτητη την αποσαφήνιση της δομής του η οποία επίσης σχετίζεται με άλλες σημαντικές βιολογικές λειτουργίες. Η βιβλιογραφία περιέχει σημαντικό αριθμό από δημοσιεύσεις οι οποίες αφορούν την πρόβλεψη της δευτεροταγούς δομής του RNA. Οι δημοσιεύσεις αυτές βασίζονται σε χρήση μεθολογιών όπως ο δυναμικός προγραμματισμός, οι στοχαστικές γραμματικές χωρίς συμφραζόμενα και η μηχανική μάθηση.

Στην παρούσα διπλωματική, θα ασχοληθούμε με την πρόβλεψη μίας άλλης δομής στην ακολουθία του RNA, του ψευδοκόμβου τύπου L. Θα στηριχθούμε στη δημοσίευση του knotify[7],[8],[9], στην οποία έχει ήδη επιλυθεί το συγκεκριμένο πρόβλημα για τον ψευδοκόμβο τύπου H, καθώς και για πιο σύνθετες δομές που πέρα από τον ψευδοκόμβο τύπου H περιλαμβάνουν επίσης τις δομές των hairpins, των bulges και των internal loops και με κατάλληλες τροποποιήσεις στην επίλυση του προβλήματος για τον ψευδοκόμβο τύπου H, θα λύσουμε και το πρόβλημα πρόβλεψης του ψευδοκόμβου τύπου L στην ακολουθία του RNA. Πιο αναλυτικά, η διπλωματική εργασία χωρίζεται σε 6 κεφάλαια . Στο κεφάλαιο 2 θα δοθούν ορισμένες θεωρητικές πληροφορίες για το RNA ενώ θα γίνει και σύγκριση με το DNA, θα αναφερθούν οι κυριότερες κατηγορίες του RNA καθώς επίσης και οι βασικότερες δομές που εντοπίζονται στην ακολουθία του RNA. Στο κεφάλαιο 3 , θα παρατεθεί σχετική βιβλιογραφία καθώς και κάποιες θεωρητικές έννοιες σχετικές με την επίλυση του παραπάνω προβλήματος. Στο κεφάλαιο 4 παρουσιάζεται αναλυτικά η μεθοδολογία της επίλυσης του προβλήματος για τον ψευδοκόμβο τύπου L καθώς και τα κυριότερα σημεία του κώδικα που χρησιμοποιήθηκε. Στο κεφάλαιο 5 παρουσιάζονται παραδείγματα από την εκτέλεση του προγράμματος με χρήση διαφόρων παραμέτρων καθώς και η αντίστοιχη έξοδος που προκύπτει. Στο κεφάλαιο 6 γίνεται η σύγκριση της συγκεκριμένης μεθοδολογίας με κάποιες άλλες γνωστές με-

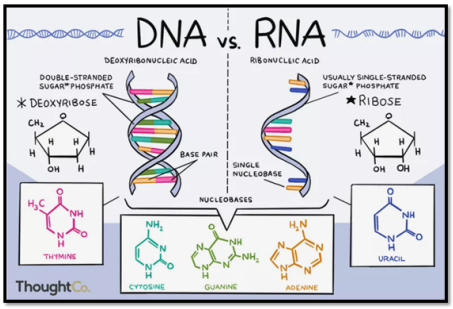
θόδους πρόβλεψης με τη χρήση κάποιων θεμελιωδών μετρικών. Τέλος, στο κεφάλαιο 7 αναφέρονται τα συμπεράσματα που προκύπτουν καθώς και μελλοντικές επεκτάσεις.

Κεφάλαιο 2

Θεωρητικό υπόβαθρο

2.1 RNA και σύγκριση με DNA

Το RNA ή ριβονουκλεϊκό οξύ [10] είναι μία από τις δύο κατηγορίες πολυμερών νουκλεϊκών οξέων στο κύτταρο. Αποτελείται από μονομερή νουκλεοτίδια τα οποία παίζουν σημαντικό ρόλο στη διαδικασία της μετάφρασης από την άλλη κατηγορία νουκλεϊκού οξέος, το DNA,σε πρωτεϊνικά προϊόντα. Ακόμη, το RNA χαρακτηρίζεται ως αγγελιαφόρος μεταξύ του DNA και των πρωτεϊνικών συμπλεγμάτων στο κυτταρόπλασμα του κυττάρου που είναι γνωστά ως ριβοσώματα καθώς είναι αυτό το οποίο καθορίζει τη σειρά με την οποία θα μεταφερθούν τα αμινοξέα στα ριβοσώματα[11], ενώ μαζί με το DNA αποτελούν το γενετικό υλικό των οργανισμών. Από χημικής άποψης, το RNA είναι παρόμοιο με το DNA. Αρχικά, τόσο το DNA όσο και το RNA είναι μακρομοριακές ενώσεις μεγάλου μοριακού βάρους. Επίσης, αμφότερα περιλαμβάνουν 4 τύπους νουκλεοτιδίων που συνδέονται μεταξύ τους με 3'-5' φωσφοδιεστερικούς δεσμούς. Ακόμη, από τα τέσσερα νουκλεοτίδια που περιέχονται στα δύο αυτά νουκλεϊκά οξέα τα τρία είναι κοινά και είναι η αδενίνη, η γουανίνη και η κυτοσίνη. Ωστόσο, υπάρχουν και αρκετές δομικές διαφορές ανάμεσα τους. Καταρχάς, το μόριο του RNA είναι μονόκλωνο σε αντίθεση με το μόριο του DNA το οποίο είναι δίκλωνο. Επιπροσθέτως, το σάκχαρο στα νουκλεοτίδια του RNA είναι η ριβόζη ενώ το αντίστοιχο σάκχαρο για το DNA είναι η δεοξυριβόζη. Επιπλέον το τέταρτο νουκλεοτίδιο στο RNA είναι η ουρακίλη ενώ στο DNA είναι η θυμίνη. Στην παρακάτω εικόνα φαίνονται οι προαναφερθείσες διαφορές στα χαρακτηριστικά των δύο μορίων.



Σχήμα 2.1: *Σύγκριση DNA και RNA [2]*

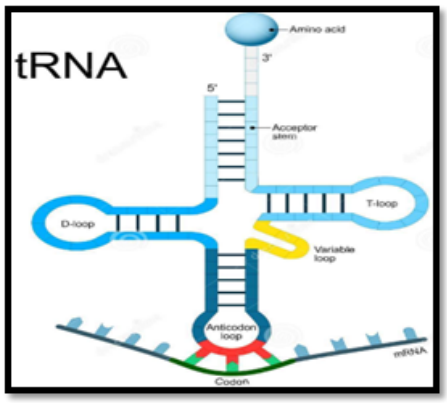
2.2 Κατηγορίες RNA

2.2.1 Αγγελιαφόρο RNA (mRNA)

Το μόριο του mRNA[12] είναι ένα μονόκλωνο μόριο RNA που αντιστοιχεί στη γενετική αλληλουχία ενός γονιδίου και διαβάζεται από ένα ριβόσωμα κατά τη διαδικασία σύνθεσης μίας πρωτεΐνης. Το mRNA δημιουργείται κατά τη διάρκεια της διαδικασίας μεταγραφής στην οποία ένα ένζυμο, η RNA πολυμεράση, μετατρέπει το γονίδιο σε ένα πρωτογενές μεταγράφημα RNA γνωστό και ως προ-mRNA.Η αποστολή του mRNA[11] είναι να μεταφέρει πληροφορίες του DNA στα άλλα δύο είδη RNA, μεταφορικό και ριβοσωμικό, με σκοπό τη μετάφραση και τη σύνθεση πρωτεϊνών. Αυτή η πληροφορία για τη σύνθεση των πρωτεϊνών αποτελείται από μια γραμμική αλληλουχία γενετικών λέξεων που βασίζονται στο αλφάβητο των τεσσάρων νουκλεοτιδίων A,C,G,U (αδενίνη, κυτοσίνη, γουανίνη, ουρακίλη). Κάθε μία από τις λέξεις αυτές που φέρει ένα mRNA έχει μήκος τριών νουκλεοτιδίων και αποτελεί ένα κωδικόνιο για το συγκεκριμένο αμινοξύ. Συνολικά υπάρχουν 64 κωδικόνια τα οποία αποτελούν το γενετικό κώδικα. Η αλληλουχία των νουκλεοτιδίων σε μια τριάδα καθορίζει το αμινοξύ που κωδικοποιείται. Για παράδειγμα, το κωδικόνιο AUG κωδικοποιεί το αμινοξύ μεθειονίνη.

2.2.2 Μεταφορικό RNA (tRNA)

Το μεταφορικό RNA[13] είναι ένας τύπος μη κωδικοποιητικού RNA με μήκος 74 έως 95 νουκλεοτίδια και ο ρόλος του είναι να μεταφέρει αμινοξέα σε μία επεκτεινόμενη πολυπεπτιδική αλυσίδα στο ριβόσωμα του κυττάρου με βάση τις οδηγίες του mRNA. Ένα μόριο tRNA έχει δύο θέσεις σύνδεσης[11]. Η πρώτη είναι ένα αντικωδικόνιο το οποίο είναι μία τριάδα νουκλεοτιδίων που ζευγαρώνουν με ένα κωδικόνιο του mRNA. Το ζευγάρωμα αυτό γίνεται με βάση τη συμπληρωματικότητα των βάσεων. Η άλλη θέση σύνδεσης συνδέεται με ένα αμινοξύ που καθορίζεται από το κωδικόνιο. Τα μεταφορικά RNA με διαφορετικά αντικωδικόνια φέρουν διαφορετικά αμινοξέα. Ένας τύπος tRNA αντιστοιχεί σε πολλά κωδικόνια αλλά σε ένα μόνο αμινοξύ. Αυτά τα tRNA μεταφέρουν αμινοξέα, το ένα μετά το άλλο, σε συνεργασία με το ριβόσωμα κατά τη μετάφραση ενός μορίου mRNA.



Σχήμα 2.2: *Δομή tRNA[3]*

Όπως φαίνεται και στην παραπάνω εικόνα, στο κάτω μέρος βρίσκεται το αντικωδικόνιο στο οποίο συνδέεται το αντίστοιχο κωδικόνιο από το mRNA προκειμένου να παραχθεί το κατάλληλο αμινοξύ. Η περιοχή D (D-arm)[14], η οποία βρίσκεται στο αριστερό μέρος, είναι χαρακτηριστική της τριτοταγούς δομής του tRNA και περιέχει τη βάση διυδρουριδίνη στην οποία οφείλεται και το όνομα της. Η περιοχή αυτή αποτελείται από μία κυκλική ακολουθία (D-loop) η οποία περιέχει και τη βάση διυδρουριδίνη και από δύο δεσμούς (D-stems). Ο ρόλος της κυκλικής περιοχής D-loop, παρόλο που δεν έχει επιβεβαιωθεί, είναι να αναγνωρίζει ένα ειδικό ένζυμο του tRNA, την τυροσίνη, ενώ αντίστοιχο αναγνωριστικό ρόλο πιστεύεται ότι έχουν και οι δύο δεσμοί D-stems χωρίς επίσης να έχει αποδειχτεί. Η περιοχή T(T arm)[15], στο δεξί μέρος της εικόνας, λειτουργεί ως αναγνωριστής για το ριβόσωμα προκειμένου αυτό να δημιουργήσει ριβοσωμικά συμπλέγματα κατά τη διάρκεια της σύνθεσης πρωτεϊνών ή της μετάφρασης.

2.2.3 Ριβοσωμικό RNA (rRNA)

Το ριβοσωμικό RNA [16] είναι ένας τύπος RNA των ριβοσωμάτων. Τα ευκαρυωτικά ριβοσώματα περιέχουν τέσσερα είδη μοριων RNA, τα 18S, 5.8S, 28S και 5S rRNA. Τα διαφορετικά είδη rRNA είναι πολυάριθμα στο κύτταρο και αποτελούν το 80% των ολικών RNA σε ένα τυπικό ευκαρυωτικό κύτταρο. Το ριβοσωμικό RNA είναι ένα παράδειγμα RNA με ενζυμική ενεργητικότητα[11] καθώς καταλύει το σχηματισμό ενός πεπτιδικού δεσμού μεταξύ των αμινοξέων. Καθώς τα αμινοξέα προστίθενται στην αλυσίδα το ένα μετά το άλλο, το ριβόσωμα τα συνδέει με πεπτιδικούς δεσμούς δημιουργώντας ένα νέο πολυπεπτίδιο. Έτσι, η σειρά των κωδικονίων σε ένα mRNA που αποτελεί το μήνυμα του DNA για τη σύνθεση πρωτεΐνης μεταφράζεται σε μία νέα πρωτεΐνη.

2.2.4 Μικρό πυρηνικό RNA (snRNA)

Το μικρό πυρηνικό RNA[17] είναι μία κλάση μικρών μορίων RNA τα οποία εντοπίζονται μέσα στις κυτταρικές δομές splicing speckles και Cajal bodies οι οποίες βρίσκονται στον πυρήνα του κυττάρου στα ευκαρυωτικά κύτταρα. Το snRNA μεταγράφεται από την RNA πολυμεράση II ή την RNA πολυμεράση III και το μήκος του είναι περίπου ίσο με

150 νουκλεοτίδια. Η κύρια λειτουργία του είναι η επεξεργασία του προ-αγγελιαφόρου RNA στο κύτταρο ενώ επίσης συμμετέχει στη ρύθμιση των παραγόντων της διαδικασίας της μεταγραφής.

2.2.5 Μικρό πυρηνισκικό RNA (snoRNA)

Το μικρό πυρηνισκικό RNA [18] είναι υπεύθυνο να καθοδηγεί τις χημικές μεταβολές στα άλλα είδη του RNA και κυρίως στο μεταφορικό, το ριβοσωμικό και το μικρό πυρηνικό RNA. Υπάρχουν 2 κατηγορίες μικρού πυρηνισκικού RNA. Η πρώτη είναι το C/D box snoRNA το οποίο ονομάζεται έτσι διότι περιέχει δύο σύντομα ακολουθιακά μοτίβα, το C (RUGAUGA) και το D (CUGA) τα οποία βρίσκονται κοντά στις άκρες 5' και 3' του μικρού πυρηνισκικού RNA αντίστοιχα. Η δεύτερη κατηγορία είναι το H/ACA box snoRNA του οποίου η δευτεροταγής δομή αποτελείται από δύο κυκλικές ακολουθίες, τα hairpins, και δύο μονές περιοχές και ονομάζεται hairpin-hinge-hairpin-tail.Επίσης το H/ACA box snoRNA περιέχει δύο ακολουθιακά μοτίβα, το H (ANANNA) και το ACA(ACA), τα οποία συνήθως περιέχονται στις μονές περιοχές της δευτεροταγούς δομής του.

2.2.6 Μικρο-RNA (miRNA)

Το μικρο-RNA [19] είναι μικρό μονόκλωνο μόριο μη κωδικοποιητικού RNA του οποίου το μήκος είναι μεταξύ 21 και 23 νουκλεοτιδίων. Το συγκεκριμένο είδος RNA εντοπίζεται στα ζώα στα φυτά και σε ορισμένους ιούς ενώ ο κύριος ρόλος του έγκειται στη συμμετοχή σε μία συγκεκριμένη διαδικασία στο RNA η οποία ονομάζεται RNA silencing και συμβάλλει στην άμυνα των οργανισμών απέναντι στους ιούς[20]. Επίσης, το μικρο-RNA συμμετέχει στη ρύθμιση της γενετικής έκφρασης μετά τη διαδικασία της μεταγραφής.

2.2.7 Μικρό παρεμβατικό RNA (siRNA)

Το μικρό παρεμβατικό RNA [21] είναι μία κατηγορία μη κωδικοποιητικού RNA το οποίο περιέχει περίπου 20 με 24 ζευγάρια βάσεων. Είναι παρόμοιο με το μικρο-RNA και συμμετέχει και αυτό στη διαδικασία του RNA silencing ενώ επίσης είναι υπεύθυνο για την έκφραση συγκεκριμένων γονιδίων με συμπληρωματικές ακολουθίες νουκλεοτιδίων η οποία επιτυγχάνεται μέσω της υποβάθμισης του αγγελιαφόρου RNA μετά τη μεταγραφή και έχει ως στόχο την πρόληψη της διαδικασίας της μετάφρασης.

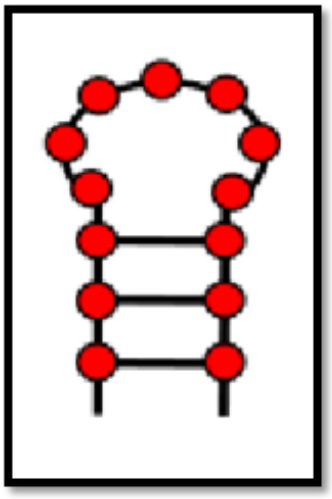
2.2.8 Μικρό υποθετικό RNA (scRNA)

Το scRNA [22] είναι ένα μικρό μόριο RNA με μήκος που συνήθως είναι λίγο μικρότερο από 100 νουκλεοτίδια και είναι επιφορτισμένο να αλληλεπιδρά με συγγενείς μοριακές εισόδους και να διαμορφώνεται ανάλογα προκειμένου να επιτύχει μεταγωγή σήματος τόσο μέσα σε ζωντανούς οργανισμούς (in vivo) όσο και εκτός αυτών (in vitro).

2.3 Κατηγορίες δομών στο RNA

2.3.1 Hairpin Loop

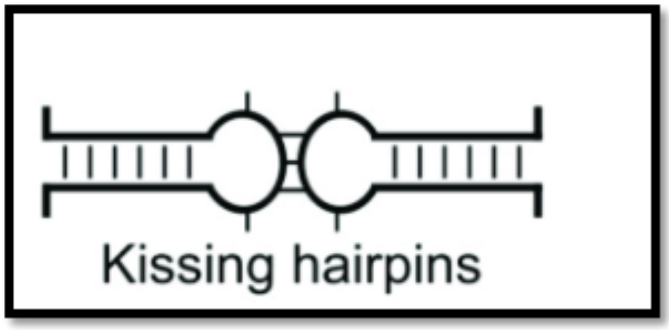
Η δομή hairpin loop [23] δημιουργείται σε μονόκλωνο RNA όταν δύο περιοχές της ίδιας αλυσίδας, συνήθως συμπληρωματικές ως προς τις βάσεις τους, ζευγαρώνουν και δημιουργούν μία διπλή έλικα η οποία καταλήγει σε ένα σύνολο αζευγάρωτων βάσεων οι οποίες σχηματίζουν έναν κύκλο. Η συγκεκριμένη δομή είναι μία από τις σημαντικότερες δευτεροταγείς δομές στο RNA και μπορεί να κατευθύνει το δίπλωμα της αλυσίδας του RNA. Επίσης συμβάλλει στη δομική σταθερότητα του αγγελιαφόρου RNA, λειτουργεί ως αναγνωριστής για ορισμένες πρωτεΐνες και λειτουργεί ως υπόστρωμα σε διάφορες ενζυμικές αντιδράσεις.



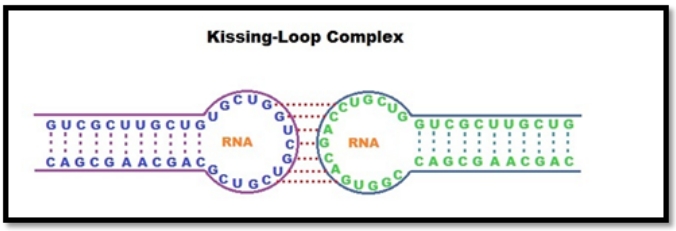
Σχήμα 2.3: *Hairpin Loop[4]*

2.3.2 Kissing Hairpin

Αυτή η δομή [24] δημιουργείται στο RNA όταν αζευγάρωτες βάσεις από ένα hairpin loop δημιουργούν ζευγάρια με αζευγάρωτες βάσεις από άλλο hairpin loop. Η αλληλεπίδραση αυτή μεταξύ των βάσεων είναι γνωστή ως loop-loop-pseudoknot. Αυτές οι ενδομοριακές αλληλεπιδράσεις συμβάλλουν σημαντικά στο να σχηματιστούν τριτοταγείς και τεταρτοταγείς δομές στο RNA. Όταν τα hairpin loops που αλληλοεπιδρούν βρίσκονται σε διαφορετικά μόρια RNA τότε αυτή η αλληλεπίδραση ονομάζεται kissing complex.



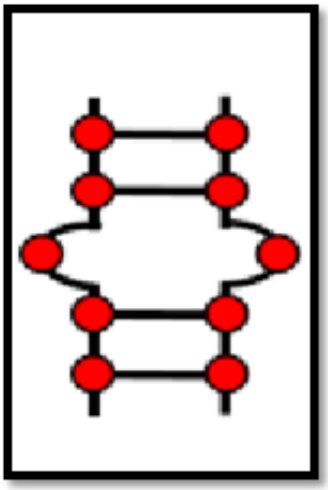
Σχήμα 2.4: *Kissing Hairpins[5]*



Σχήμα 2.5: *Kissing Loop Complex[6]*

2.3.3 Internal Loop

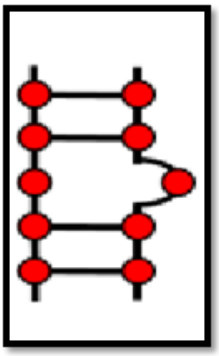
Η δομή αυτή [25] προκύπτει στο RNA όταν η αλυσίδα του χωρίζεται λόγω της μη ύπαρξης ζευγαριών Watson Crick ανάμεσα στις βάσεις. Η διαφορά αυτής της δομής με το hairpin loop είναι ότι το internal loop δημιουργείται στη μέση του τεντώματος της διπλής αλυσίδας ενώ το hairpin loop δημιουργείται στην άκρη του. Τα internal loops διακρίνονται στα συμμετρικά και στα μη συμμετρικά τα οποία είναι γνωστά ως bulges.Τέλος, η δομή του internal loop αποτελεί τη βάση για τη δημιουργία πολλών σύνθετων δομών στο RNA όπως το C-loop, το kink-turn και τα bulged-G motifs.



Σχήμα 2.6: *Internal Loop[4]*

2.3.4 Bulges

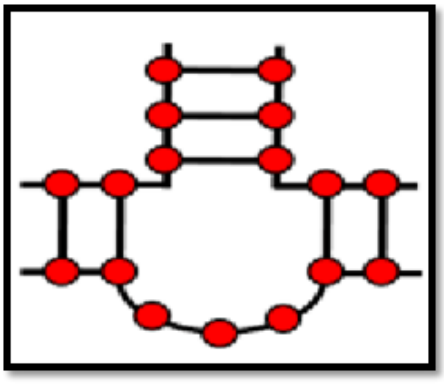
Τα bulges είναι ανοίγματα που δημιουργούνται στη μία πλευρά της αλυσίδας του RNA από βάσεις που δε ζευγαρώνουν με γειτονικές βάσεις που προκύπτουν κατά το δίπλωμα της μονής αλυσίδας του RNA.Αυτή η δομή [26] είναι από τις πιο κοινές non-Watson-Crick δομές που συναντώνται στο RNA. Τα bulges είναι σημαντικά για το σχηματισμό δευτεροταγών και τριτοταγών δομών ενώ μπορούν επίσης να αλληλεπιδράσουν με τις πρωτεΐνες.



Σχήμα 2.7: *Bulge στη δεξιά πλευρά της αλυσίδας λόγω της μη ύπαρξης ζευγαριού μεταξύ των βάσεων[4]*

2.3.5 Multiloop

Αυτή η δομή [27] η οποία επίσης ονομάζεται και junction ή multi-branched loop δημιουργείται όταν τρία ή περισσότερα σύνολα από ζευγάρια μεταξύ βάσεων συναντώνται δημιουργώντας έναν κύκλο. Αυτά τα ζευγάρια βάσεων μπορούν να περιέχουν ανάμεσα τους και βάσεις οι οποίες είναι αζευγάρωτες.

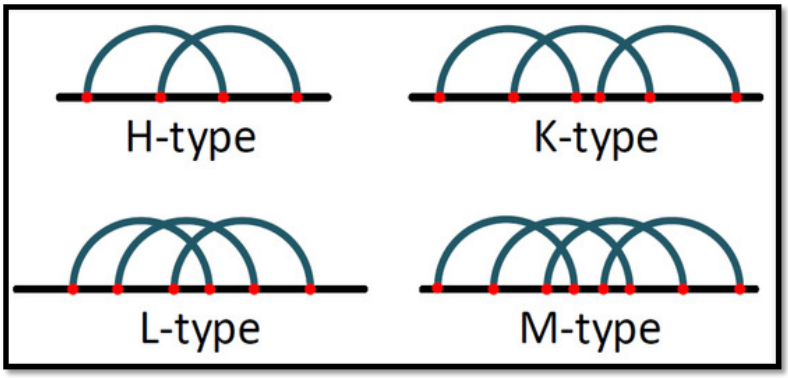


Σχήμα 2.8: *Multiloop[4]*

2.3.6 Pseudoknot

Ο ψευδοκόμβος [28] είναι μία δευτεροταγής δομή στα νουκλεϊκά οξέα η οποία περιέχει τουλάχιστον δύο hairpin loops και στην οποία το μισό του ενός loop παρεμβάλλεται ανάμεσα στα δύο μισά του άλλου loop. Είναι ένα από τα πιο συνηθισμένα μοτίβα που συναντώνται στο δίπλωμα του RNA και εντοπίστηκε για πρώτη φορά στον ιό Turnip Yellow Mosaic το 1982. Η λειτουργία των ψευδοκόμβων [29] συχνά σχετίζεται με την αλληλεπίδραση με τα ριβοσώματα. Επίσης, οι ψευδοκόμβοι χρησιμοποιούνται στους ιούς προκειμένου να γίνεται έλεγχος στη μετάφραση του RNA τους,στην αντιγραφή του RNA, καθώς και στην εναλλαγή ανάμεσα στις δύο αυτές λειτουργίες. Υπάρχουν τέσσερις βασικές κατηγορίες ψευδοκόμβων οι οποίες έχουν εντοπιστεί και είναι οι εξής:

1.Ψευδοκόμβος τύπου H
2. Ψευδοκόμβος τύπου K
3. Ψευδοκόμβος τύπου L
4. Ψευδοκόμβος τύπου M



Σχήμα 2.9: *Οι 4 τύποι ψευδοκόμβων[7]*

Από τους 4 παραπάνω τύπους ψευδοκόμβου ο επικρατέστερος [30] είναι ο τύπος H. Στην παρούσα διπλωματική θα επιλύσουμε το πρόβλημα εντοπισμού του ψευδοκόμβου τύπου L στην ακολουθία του RNA στηριζόμενοι στην υλοποίηση της δημοσίευσης knotify[7] για την επίλυση του προβλήματος για τον ψευδοκόμβο τύπου H.

Κεφάλαιο 3

Σχετική βιβλιογραφία και θεωρητικές έννοιες

3.1 Σχετική βιβλιογραφία

Έχει αποδειχτεί ότι το γενικό πρόβλημα της πρόβλεψης της δευτεροταγούς δομής του RNA μέσω της χρήσης θερμοδυναμικών μοντέλων ανήκει στην κατηγορία των NP-complete προβλημάτων όταν περιέχεται σε αυτήν και η δομή των ψευδοκόμβων[31]. Προκειμένου να αντιμετωπιστεί το συγκεκριμένο πρόβλημα, μία μεθοδολογία που χρησιμοποιείται ευρέως είναι αυτή του δυναμικού προγραμματισμού. Ένας από τους αλγορίθμους σε αυτήν την κατηγορία είναι ο αλγόριθμος των Eddy και Rivas[32], ο οποίος υπολογίζει τη δομή με την ελάχιστη ενέργεια λαμβάνοντας υπ'όψιν διάφορες θερμοδυναμικές παραμέτρους. Η χρονική πολυπλοκότητα του αλγορίθμου είναι $O(n^6)$ και η χωρική του πολυπλοκότητα είναι $O(n^4)$, αμφότερες στη χειρότερη περίπτωση.Ένας άλλος αλγόριθμος δυναμικού προγραμματισμού είναι αυτός του Dirks[33], ο οποίος υπολογίζει με τη βοήθεια μίας συνάρτησης διαμέρισης τη δομή με την ελάχιστη ενέργεια παρουσιάζοντας χρονική πολυπλοκότητα $O(n^5)$.

Επιπλέον, διάφοροι ευριστικοί αλγόριθμοι έχουν υλοποιηθεί προκειμένου να ξεπεράσουν το εμπόδιο της υπολογιστικής πολυπλοκότητας. Ένας από αυτούς είναι ο Knotty[34] ο οποίος προβλέπει τη δομή με την ελάχιστη ενέργεια και αποτελεί βελτίωση του αλγορίθμου CCJ αφού χρησιμοποιεί την τεχνική της αραιοποίησης μειώνοντας έτσι τη χωρική πολυπλοκότητα από $\Theta(n^4)$ σε $\Theta(n^3+Z)$, ενώ αμφότεροι παρουσιάζουν χρονική πολυπλοκότητα $\Theta(n^5)$. Ο IPknot[35], ο οποίος κάνει χρήση ακεραίου προγραμματισμού, αποσυνθέτει τη δομή με τους ψευδοκόμβους σε ένα σύνολο υποακολουθιών χωρίς ψευδοκόμβους και υπολογίζει προσεγγιστικά μία κατανομή πιθαοτήτων για τα ζευγαρώματα μεταξύ των βάσεων με χρήση τιμών κατωφλίων προβλέποντας τελικά τη δομή με τη μεγαλύτερη ακρίβεια. Ο Probknot[36] μεγιστοποιεί την ακρίβεια της τελικής δομής μέσω του υπολογισμού πιθανοτήτων για ζευγάρια βάσεων παρουσιάζοντας χρονική πολυπλοκότητα $O(n^2)$. Ο Threshknot[37] αποτελεί βελτίωση του προηγούμενου αλγορίθμου καθώς αποκλείει τα ζευγάρια των οποίων η πιθανότητα βρίσκεται κάτω από ένα όριο. Ο συγκεκριμένος αλγόριθμος είναι πολύ γρήγορος όσον αφορά την πρόβλεψη ενώ επίσης κλιμακώνει σχεδόν γραμμικά με το μήκος της ακολουθίας εισόδου.

Επιλέον, έχουν αναπτυχθεί και διάφοροι αλγόριθμοι οι οποίοι κάνουν χρήση στοχαστικών γραμματικών χωρίς συμφραζόμενα και η ακρίβεια τους εξαρτάται από την εκάστοτε γραμματική. Ο αλγόριθμος των Knudsen και Hein κάνει χρήση τέτοιων γραμματικών και υπολογίζει μία κατανομή πιθανοτήτων για διάφορες δομές του RNA ωστόσο είναι ιδιαίτερα αργός. Ο αλγόριθμος Pfold[38] βασίζεται στον αλγόριθμο των Knudsen και Hein ωστόσο είναι ταχύτερος από αυτόν ενώ επίσης παρουσιάζει μεγαλύτερη ανεκτικότητα σε σφάλματα. Η PPfold[39] αποτελεί μία έκδοση της Pfold η οποία χρησιμοποιεί πολλαπλά νήματα βελτιώνοντας έτσι το χρόνο εκτέλεσης για πολυπύρηνες μηχανές. Μία άλλη μέθοδος,η RNADecoder[40], λαμβάνει ως εισόδους πέρα από τη γραμματική, ένα σύνολο από ακολουθίες RNA μαζί με διάφορες περιοχές κωδικοποίησης πρωτεϊνών και σχετικά φυλογενετικά δέντρα και μπορεί να προβλέψει είτε τις πιθανότητες για διάφορα ζευγάρια βάσεων είτε συγκεκριμένα διπλώματα της ακολουθίας για το σχηματισμό δευτεροταγών δομών μαζί με την αξιοπιστία κάθε ενός από τα διπλώματα.

Τέλος, σύμφωνα με πρόσφατες έρευνες, έχει προταθεί η χρήση μεθόδων μηχανικής μάθησης για την πρόβλεψη της δευτεροταγούς δομής του RNA. Η DMfold[41] χρησιμοποιεί μεθόδους βαθιάς μηχανικής μάθησης προκειμένου να προβλέψει τη δομή του RNA. Πιο αναλυτικά, χρησιμοποιεί δύο μονάδες, τη μονάδα πρόβλεψης και τη μονάδα διόρθωσης. Η μονάδα πρόβλεψης χρησιμοποιεί ένα αμφίδρομο δίκτυο LSTM τριών στρωμάτων το οποίο λειτουργεί ως κωδικοποιητής και ένα πλήρως συνδεδεμένο δίκτυο τεσσάρων στρωμάτων (FFCL) ως αποκωδικοποιητή προκειμένου να προβλέψει από την ακολουθία του RNA την αναπαράσταση dot-bracket. Στη συνέχεια, η μονάδα διόρθωσης βρίσκει πιθανά λάθη πρόβλεψης που υπάρχουν από το προηγούμενο στάδιο και καταλήγει στη σωστή έξοδο χρησιμοποιώντας και την αρχή μεγιστοποίησης των ζευγαριών βάσεων (IBPMP). Μία άλλη μέθοδος, η 2dRNA[42], κάνει χρήση πάλι ενός δικτύου LSTM και ενός συνελικτικού νεωρωνικού δικτύου, του U-net, προκειμένου να πραγματοποιήσει την πρόβλεψη της δευτεροταγούς δομής. Τέλος, η μέθοδος ATTfold[43] είναι ένα μοντέλο βαθιάς μηχανικής μάθησης το οποίο χρησιμοποιεί έναν κωδικοποιητή, ένα μηχανιμό παρατήρησης και ένα συνελικτικό νευρωνικό δίκτυο ως αποκωδικοποιητή προκειμένου να παράξει έναν πίνακα με σκορ για τα ζευγάρια βάσεων.Στη συνέχεια, αυτός ο πίνακας συμμορφώνεται με κάποιους περιορισμούς σχετικούς με τη δομή του RNA και προκύπτει ένας πίνακας από 0 κι 1 όπου το 0 δηλώνει ότι οι βάσεις δε δημιουργούν ζευγάρι μεταξύ τους ενώ το 1 δηλώνει την ύπαρξη ζευγαριού ανάμεσα τους.

3.2 Θεωρητικές Έννοιες

3.2.1 Συντακτική Αναγνώριση Προτύπων

Η συντακτική αναγνώριση προτύπων[44] είναι μία διαδικασία κατά την οποία κάθε αντικείμενο μπορεί να αναπαρασταθεί σα μία ειδική μεταβλητή η οποία θα εκφράζει τα ιδιαίτερα χαρακτηριστικά του. Αυτό μας επιτρέπει να λάβουμε υπ' όψιν πιο σύνθετες σχέσεις μεταξύ των χαρακτηριστικών του αντικειμένου απ' ότι στην περίπτωση όπου απλά χρησιμοποιούμε διανύσματα χαρακτηριστικών σταθερών διαστάσεων, όπως στη στατιστική ταξινόμηση. Στη συντακτική αναγνώριση προτύπων, η γλώσσα ορίζεται σαν ένα σύνολο από συντακτικούς κανόνες[7] οι οποίοι κατασκευάζουν μία συμβολοσειρά η οποία ανήκει στη γλώσσα. Το συγκεκριμένο σύνολο από κανόνες είναι μέρος της γραμματικής της γλώσσας και καθορίζει τον τρόπο με τον οποίο παράγονται συγκεκριμένα σύνολα από σύμβολα, τα οποία αποτελούν συστατικά της. Οι συγκεκριμένες γραμματικές ανήκουν σε τέσσερις κατηγορίες οι οποίες έχουν χωριστεί από το Noam Chomsky και αποτελούν την ιεραρχία Chomsky. Η μία κατηγορία εξ' αυτών είναι οι γραμματικές χωρίς συμφραζόμενα και σε αυτήν την κατηγορία ανήκει και η γραμματική η οποία δημιουργήθηκε για την επίλυση του προβλήματος για τον ψευδοκόμβο τύπου L και θα αναλυθεί παρακάτω. Οι άλλες τρεις κατηγορίες γραμματικών είναι οι context-sensitive, οι recursively-enumerable και οι κανονικές γραμματικές (regular grammars).

3.2.2 Context Free Grammars

Στη θεωρία των τυπικών γλωσσών, σε μία γραμματική χωρίς συμφραζόμενα[45] οι κανόνες παραγωγής είναι της μορφής A->a όπου το A είναι ένα μονό μη τερματικό σύμβολο και το a είναι μία συμβολοσειρά από τερματικά και μη τερματικά σύμβολα. Μία γραμματική χωρίς συμφραζόμενα ορίζεται από την πλειάδα (V,Σ,R,S). Το V είναι ένα πεπερασμένο σύνολο το οποίο αναπαριστά όλους τους μη τερματικούς χαρακτήρες της γραμματικής. Το Σ είναι ένα πεπερασμένο σύνολο, το οποίο περιέχει όλα τα τερματικά σύμβολα της γραμματικής. Το R δηλώνει μία σχέση στο σύνολο V x (V U Σ)*, όπου το σύμβολο * είναι το άστρο Kleene, και αναπαριστά το σύνολο των κανόνων παραγωγής της γραμματικής ενώ το S αναπαριστά το σύμβολο από το οποίο εκκινεί η γραμματική και πρέπει να περιέχεται στο σύνολο V των μη τερματικών χαρακτήρων. Οι κεφαλαίοι λατινικοί χαρακτήρες [7] χρησιμοποιούνται για να εκφράσουν μη τερματικά σύμβολα ενώ οι μικροί λατινικοί χαρακτήρες εκφράζουν τερματικά σύμβολα. Οι ελληνικοί χαρακτήρες μπορούν να χρησιμοποιηθούν τόσο ως τερματικά όσο και ως μη τερματικά σύμβολα.

3.2.3 Αφηρημένα Συντακτικά Δέντρα

Στην επιστήμη των υπολογιστών, ένα αφηρημένο συντακτικό δέντρο [46] είναι μία δενδρική αναπαράσταση μίας αφηρημένης συντακτικής δομής ενός κειμένου το οποίο είναι γραμμένο σε μία τυπική γλώσσα. Κάθε κόμβος αυτού του δέντρου δηλώνει ένα

κατασκεύασμα που υπάρχει μέσα στο κείμενο. Η δομή είναι αφηρημένη υπό την έννοια ότι δεν απεικονίζει κάθε πληροφορία που εμφανίζεται στην πραγματική σύνταξη του κειμένου, αλλά πληροφορίες σχετικές με τη δομή ή το περιεχόμενό του. Επίσης, τα αφηρημένα συντακτικά δέντρα χρησιμοποιούνται ευρέως στους μεταγλωττιστές προκειμένου να αναπαραστήσουν τη δομή του προγράμματος. Ένα αφηρημένο συντακτικό δέντρο είναι συνήθως αποτέλεσμα της φάσης συντακτικής ανάλυσης του μεταγλωττιστή. Συχνά χρησιμοποιούνται ως μία ενδιάμεση αναπαράσταση του προγράμματος μέσα από τα πολλά στάδια ανάλυσης που απαιτείται να κάνει ο μεταγλωττιστής και έχουν σημαντική επίδραση στην τελική έξοδο του μεταγλωττιστή. Πέρα από τους μεταγλωττιστές, τα δέντρα αυτά χρησιμοποιούνται ευρέως και σε συστήματα ανάλυσης και μετατροπής προγραμμάτων.

3.2.4 CFG Parsers

Η ανάλυση(parse) [47] είναι η διαδικασία κατά την οποία ανατίθεται δομή σε προτάσεις ενός κειμένου η οποία προκύπτει από τη γραμματική περιγραφή της γλώσσας. Υπάρχουν διάφοροι αλγόριθμοι οι οποίοι έχουν αναπτυχθεί προκειμένου να πραγματοποιούν όσο το δυνατόν αποδοτικότερα αυτή τη διαδικασία. Δύο από τους πιο σημαντικούς αλγορίθμους οι οποίοι έχουν αναπτυχθεί είναι ο CYK από τους Cocke Younger και Kasami οι οποίοι τον δημιούργησαν και ο αλγόριθμος του Earley. Στη συγκεκριμένη υλοποίηση για την επίλυση του προβλήματος του ψευδοκόμβου τύπου L έχει χρησιμοποιηθεί ο αλγόριθμος του Earley λόγω της μεγάλης του ικανότητας στο χειρισμό διφορούμενων γραμματικών. Παρακάτω παρουσιάζονται πιο αναλυτικά οι δύο προαναφερθέντες αλγόριθμοι.

3.2.5 Αλγόριθμος CYK

Ο αλγόριθμος αυτός[48] χρησιμοποιεί τη μεθοδολογία του δυναμικού προγραμματισμού προκειμένου να αποφανθεί εάν μία συμβολοσειρά μπορεί να προκύψει από την εκάστοτε γραμματική. Προκειμένου να μπορέσει να χρησιμοποιηθεί ο συγκεκριμένος αλγόριθμος, θα πρέπει η γραμματική εισόδου να είναι σε CNF(Chomsky Normal Form).Αυτό συμβαίνει διότι ο αλγόριθμος ελέγχει κατά πόσο μπορεί να χωρίσει την υπακολουθία σε μικρότερα μέρη. Ο συγκεκριμένος αλγόριθμος παρουσιάζει πολυπλοκότητα χειρότερης περίπτωσης $O(n^3 * G)$ όπου n είναι το μήκος της συμβολοσειράς εισόδου και G είναι το μέγεθος της γραμματικής. Ο ψευδοκώδικας για το συγκεκριμένο αλγόριθμο παρουσιάζεται παρακάτω

1.Αλγόριθμος CYK

```
    let the input be a string I consisting of n characters: a1 ... an.
let the grammar contain r nonterminal symbols R1 ... Rr,
with start symbol R1.
let P[n,n,r] be an array of booleans. Initialize all
elements of P to false.
let back[n,n,r] be an array of lists of backpointing triples.
```

```
Initialize all elements of back to the empty list.

for each s = 1 to n
    for each unit production Rv → as
        set P[1,s,v] = true

for each l = 2 to n -- Length of span
    for each s = 1 to n−l+1 -- Start of span
        for each p = 1 to l−1 -- Partition of span
            for each production Ra →    Rb Rc
                if P[p,s,b] and P[l−p,s+p,c] then
                    set P[l,s,a] = true,
                    append <p,b,c> to back[l,s,a]

if P[n,1,1] is true then
    I is member of language
    return back -- by retracing the steps through back,
    one can easily construct all possible parse trees
    of the string.
else
    return "not a member of language"
```

Ο αλγόριθμος λαμβάνει υπ' όψιν κάθε υπακολουθία της συμβολοσειράς εισόδου και θεωρεί ότι το στοιχείο P[l,s,v] είναι true αν η υπακολουθία μήκους l η οποία ξεκινά από το s μπορεί να έχει παραχθεί από το μη τερματικό σύμβολο Rv. Ο αλγόριθμος εκτελείται για υπακολουθίες από μήκος 1 και προχωράει και σε μεγαλύτερα μήκη. Για μήκος μεγαλύτερο ή ίσο του 2, λαμβάνει υπ' όψιν κάθε διαμέριση της ακολουθίας σε δύο μέρη και ελέγχει αν υπάρχει κανόνας της μορφής A->BC όπου το B αντιστοιχεί στο πρώτο κομμάτι και το C αντιστοιχεί στο δεύτερο κομμάτι της διαμέρισης. Αν αυτό αληθεύει, τότε καταγράφει ότι το A ταιριάζει με τη συμβολοσειρά. Όταν η διαδικασία ολοκληρωθεί, η συμβολοσειρά εισόδου παράγεται από τη γραμματική εφόσον επιβεβαιωθεί ότι μπορεί να παραχθεί από το αρχικό σύμβολο της.

3.2.6 Αλγόριθμος του Earley

Ο συγκεκριμένος αλγόριθμος[49] έχει πάρει το όνομα του από τον εφευρέτη του Jay Earley και χρησιμοποιεί μία top-down μεθοδολογία δυναμικού προγραμματισμού. Στη γενική περίπτωση παρουσιάζει πολυπλοκότητα $O(n^3)$ όπου n είναι το μήκος της ακολουθίας εισόδου. Ακόμη, παρουσιάζει τετραγωνική πολυπλοκότητα για μη διφορούμενες γραμματικές ενώ για ντετερμινιστικές γραμματικές χωρίς συμφραζόμενα η πολυπλοκότητα του είναι γραμμική. Ακόμη ο αλγόριθμος είναι ιδιαίτερα αποδοτικός όταν οι κανόνες της γραμματικής είναι γραμμένοι με αναδρομή από αριστερά. Ο ψευδοκώδικας για το συγκεκριμένο αλγόριθμο παρουσιάζεται παρακάτω.

```
    DECLARE ARRAY S;

function INIT(words)
    S ← CREATE_ARRAY(LENGTH(words) + 1)
    for k ← from 0 to LENGTH(words) do
        S[k] ← EMPTY_ORDERED_SET

function EARLEY_PARSE(words, grammar)
    INIT(words)
    ADD_TO_SETγ(( →• S, 0), S[0])
    for k ← from 0 to LENGTH(words) do
        for each state in S[k] do  // S[k] can expand during this loop
            if not FINISHED(state) then
                if NEXT_ELEMENT_OF(state) is a nonterminal then
                    PREDICTOR(state, k, grammar) //non_terminal
                else do
                    SCANNER(state, k, words) // terminal
            else do
                COMPLETER(state, k)
        end
    end
    return chart

procedure PREDICTOR((A →α•β B, j), k, grammar)
    for each (B →γ ) in GRAMMAR_RULES_FOR(B, grammar) do
        ADD_TO_SET((B →•γ , k), S[k])
    end

procedure SCANNER((A →α•β a, j), k, words)
    if a ⊠ PARTS_OF_SPEECH(words[k]) then
        ADD_TO_SET((A →α•β a, j), S[k+1])
    end

procedure COMPLETER((B →γ• , x), k)
    for each (A →α•β B, j) in S[x] do
        ADD_TO_SET((A →α•β B, j), S[k])
    end
```

Στον παραπάνω αλγόριθμο τα α,β,γ αναπαριστούν συμβολοσειρές από τερματικά και μη τερματικά σύμβολα, τα X και Y αναπαριστούν μη τερματικά σύμβολα ενώ το a είναι ένα τερματικό σύμβολο. Η αναπαράσταση X->α•β δηλώνει ότι το α έχει ήδη αναλυθεί ενώ το β όχι. Για κάθε έναν από τους χαρακτήρες της συμβολοσειράς εισόδου ο αναλυτής παράγει ένα σύνολο καταστάσεων. Κάθε ένα από αυτά τα σύνολα είναι μία

πλειάδα της μορφής (X-> α•β, i) η οποία αποτελείται:

1)Από τον κανόνα παραγωγής(X → α β)

2)Τη θέση που βρισκόμαστε στον κανόνα η οποία δηλώνεται από την τελεία και

3)Τη θέση i της συμβολοσειράς εισόδου όπου άρχισε το ταίριασμα του κανόνα παραγωγής

Το σύνολο κατάστασης στη θέση της εισόδου k συμβολίζεται με S(k). Ο αναλυτής ξεκινά από το S(0) και εκτελεί επαναλαμβανόμενα τις τρεις παρακάτω λειτουργίες:

1)Prediction: Για κάθε κατάσταση στο S(k) της μορφής (X->α•Yβ,j) προσθέτει στο σύνολο κάθε κανόνα που έχει το Y στο αριστερό μέρος του

2)Scanning:Εάν το επόμενο σύμβολο είναι τερματικό τότε για κάθε κανόνα της μορφής (X->α•aβ,j) μεταφέρει την τελεία μία θέση δεξιά και προσθέτει τον παραγόμενο κανόνα στο επόμενο σύνολο S(k+1)

3)Completion:Για κάθε κατάσταση στο S(k) της μορφής (Y → γ •, j) βρίσκει όλες τις καταστάσεις της μορφής (X → α • Y β, i) στο S(j) και αφού μεταφέρει την τελεία μία θέση δεξιά προσθέτει τον κανόνα στο S(k).

Ο αλγόριθμος δεν προσθέτει διπλότυπα στα σύνολα καταστάσεων. Οι τρεις συναρτήσεις αυτές επαναλαμβάνονται από τον αλγόριθμο μέχρι να μη μπορούν να προστεθούν άλλες καταστάσεις στα σύνολα. Ο αλγόριθμος αποδέχεται τη συμβολοσειρά αν η κατάσταση (X → γ •, 0) καταλήγει στο S(n), όπου (X → γ) είναι ο πρώτος κανόνας της γραμματικής και n είναι το μήκος της ακολουθίας, αλλιώς την απορρίπτει.

3.2.7 YAEP

Ο yaep[50] είναι συντομογραφία του Yet Another Earley Parser και είναι μία από τις γρηγορότερες και πιο αποτελεσματικές υλοποιήσεις του αλγορίθμου του Earley και σε αυτόν στηριζόμαστε προκειμένου να επιλύσουμε το πρόβλημα και για τον ψευδοκόμβο τύπου L με τη μεθοδολογία της συντακτικής αναγνώρισης προτύπων. Ο κύριος στόχος του είναι να επιτύχει μεγάλη ταχύτητα καθώς και όσο το δυνατόν μικρότερες απαιτήσεις σε μνήμη που είναι απαραίτητα στοιχεία προκειμένου να χρησιμοποιηθεί σε μεταγλωττιστές και επεξεργαστές γλώσσας. Χρησιμοποιεί μνήμη περίπου 5Mb προκειμένου να αναλύσει 10 χιλιάδες γραμμές προγράμματος σε γλώσσα C. Εκτός από ανάλυση απλής σύνταξης και παραγωγή ενός αφηρημένου δέντρου μπορεί επίσης να αναλύσει εισόδους που προκύπτουν και από αμφίσημες γραμματικές παράγοντας ως αποτέλεσμα πολλαπλά αφηρημένα συντακτικά δέντρα.

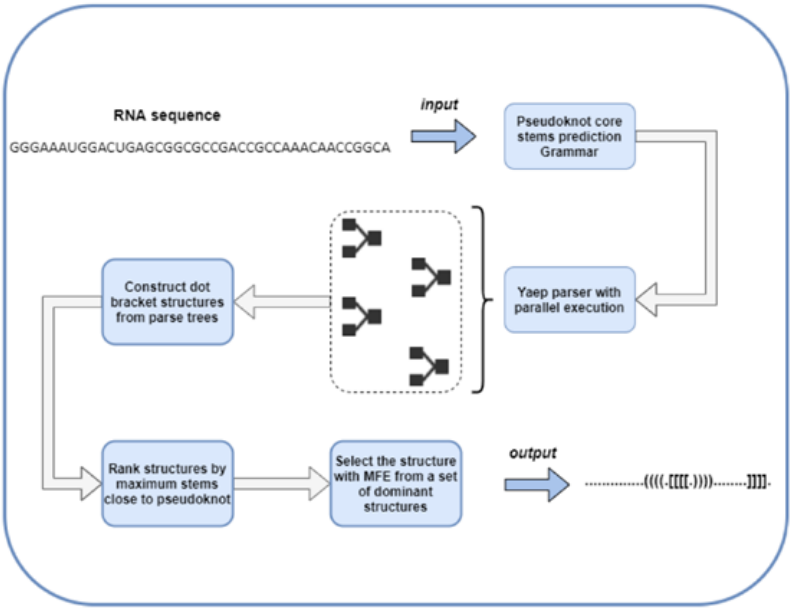
Κεφάλαιο 4

Επίλυση του προβλήματος για τον ψευδοκόμβο τύπου L

4.1 Μεθοδολογία επίλυσης του προβλήματος για τον ψευδοκόμβο τύπου L

Η μεθοδολογία που ακολουθήσαμε προκειμένου να επιλυθεί το πρόβλημα για τον ψευδοκόμβο τύπου L χωρίζεται σε τρία στάδια. Στο πρώτο στάδιο, γίνεται ανάλυση της ακολουθίας του RNA και εντοπίζονται όλες οι περιπτώσεις ύπαρξης του ψευδοκόμβου τύπου L μέσα σε αυτήν. Προκειμένου να φέρουμε εις πέρας αυτή τη δουλειά, έχουμε υλοποιήσει δύο διαφορετικούς αλγορίθμους στους οποίους, για ευκολία, αντιμετωπίζουμε την ακολουθία του RNA σα μία συμβολοσειρά της οποίας οι χαρακτήρες είναι οι 'A','U','G','C' για την αδενίνη, την ουρακίλη, τη γουανίνη και την κυτοσίνη αντίστοιχα. Ο πρώτος εκ των δύο αλγορίθμων κάνει χρήση της μεθοδολογίας συντακτικής αναγνώρισης προτύπων. Πιο αναλυτικά, αρχικά έχουμε δημιουργήσει μία γραμματική χωρίς συμφραζόμενα η οποία περιέχει όλους τους κανόνες μέσω των οποίων μπορεί να παραχθεί ένας ψευδοκόμβος τύπου L. Στη συνέχεια, μέσω της γραμματικής και με τη χρήση ενός συντακτικού αναλυτή παράγονται όλα τα δέντρα τα οποία περιέχουν έναν ψευδοκόμβο για την εκάστοτε ακολουθία RNA που αποτελεί την είσοδο. Τέλος, τα δέντρα αυτά διασχίζονται μέσω κατάλληλης συνάρτησης προκειμένου να προσδιορίσουμε ορισμένα βασικά μεγέθη για τον κάθε ψευδοκόμβο τα οποία μας βοηθούν τελικά να βρούμε τις θέσεις των βάσεων για τους κύριους δεσμούς που περιέχει ο καθένας εξ' αυτών. Ο δεύτερος αλγόριθμος ο οποίος χρησιμοποιείται είναι ένας άπληστος αλγόριθμος δυναμικού προγραμματισμού ο οποίος υπολογίζει τα ίδια βασικά μεγέθη με πριν στηριζόμενος μόνο σε διαδοχικές προσπελάσεις της συμβολοσειράς που αντικατοπτρίζει την ακολουθία του RNA. Όσον αφορά το συγκεκριμένο στάδιο της επίλυσης του προβλήματος, στην περίπτωση του ψευδοκόμβου τύπου H είχαν χρησιμοποιηθεί και οι δύο παραπάνω αλγόριθμοι ενώ για τη δομή των hairpins έγινε χρήση μόνο της συντακτικής αναγνώρισης προτύπων. Το επόμενο βήμα ήταν να υπολογίσουμε τους δεσμούς που δημιουργούνταν μεταξύ βάσεων γύρω από τους κύριους δεσμούς στην ακολουθία του RNA.Προκειμένου να ξεχωρίσουμε στην τελική αναπαράσταση της ακολουθίας του RNA τις βάσεις που δημιουργούσαν δεσμούς από τις υπόλοιπες έχουμε

επιλέξει να αναπαραστήσουμε τη συμβολοσειρά εισόδου σε μορφή dot-bracket. Συγκεκριμένα, τις βάσεις που δημιουργούν ζεύγη με άλλες τις αντικαθιστούμε με διάφορες αγκύλες (brackets) ενώ τις αζευγάρωτες βάσεις τις αντικαθιστούμε με τελίτσες (dots). Το τελευταίο στάδιο της όλης διαδικασίας αφορούσε τον εντοπισμό της βέλτιστης δομής από όλες όσες είχαν προκύψει στα προηγούμενα στάδια η οποία θα ήταν τελικά και η έξοδος του συνολικού προγράμματος. Τα δύο κριτήρια στα οποία στηρίχτηκε αυτή η επιλογή ήταν η ελαχιστοποίηση του ποσού της ελεύθερης ενέργειας αναδίπλωσης καθώς και η μεγιστοποίηση του αριθμού ζευγαριών μεταξύ βάσεων που υπήρχαν στη δομή. Στην παρακάτω εικόνα φαίνονται σχηματικά όλα τα βήματα επίλυσης του συνολικού προβλήματος.



Σχήμα 4.1: *Μεθοδολογία επίλυσης του προβλήματος εντοπισμού ψευδοκόμβων[7]*

4.2 Μεθοδολογία Συντακτικής Αναγνώρισης Προτύπων

4.2.1 Δημιουργία γραμματικής αναγνώρισης των ψευδοκόμβων

Προκειμένου να δημιουργήσουμε τη γραμματική μέσω της οποίας γίνεται η αναγνώριση της δομής μέσα στην ακολουθία του RNA, στηριχθήκαμε στα χαρακτηριστικά τα οποία παρουσιάζει ο συγκεκριμένος τύπος ψευδοκόμβου. Όπως φαίνεται στο σχήμα 2.9, οι κύριες βάσεις που αποτελούν τον συγκεκριμένο ψευδοκόμβο είναι 6 και δημιουργούν 3 ζευγάρια μεταξύ τους ανά δύο. Η πρώτη κύρια βάση δημιουργεί ζευγάρι με την τέταρτη, η δεύτερη με την πέμπτη και η τρίτη με την έκτη. Οπότε, λόγω των 3 κύριων ζευγαριών που υπάρχουν στον τύπο L, αλλά και των 6 τύπων ζευγαριών που συναντώνται συνολικά στο RNA (ζευγάρια αδενίνης-ουρακίλης, κυτοσίνης-γουανίνης, γουανίνης-ουρακίλης και τα αντίστροφα τους) ο συνολικός αριθμός κανόνων της γραμματικής για την αναγνώριση της δομής είναι ίσος με 6^3=216. Προκειμένου να δημιουργήσουμε όλους αυτούς τους κανόνες, έχουμε δημιουργήσει ένα μικρό κομμάτι κώδικα

σε python το οποίο παρατίθεται παρακάτω.

2.Κομμάτι κώδικα σε python για τη δημιουργία των κανόνων της γραμματικής για τον ψευδοκόμβο τύπου L

```
L=['a','u','g','c','g','u']
L2=['u','a','c','g','u','g']
L1=['' for _ in range(216)]
for i in range(216):
    L1[i]+=L[i//36]
for i in range(216):
    L1[i]+='L'
for i in range(0,216):
    L1[i]+=L[(i//6)%6]
for i in range(0,216):
    L1[i]+='L'
for i in range(216):
    L1[i]+=L[i%6]
for i in range(216):
    L1[i]+='D'
for j in range(216):
    L1[j]+=L2[j//36]
for j in range(216):
    L1[j]+='L'
for j in range(216):
    L1[j]+=L2[(j//6)%6]
for j in range(216):
    L1[j]+='L'
for j in range(0,216):
    L1[j]+=L2[j%6]
L3,L4=[],[]
for x in L1:
    if ((x[0]=='u' and x[6]=='g') or (x[2]=='u' and x[8]=='g') or
    (x[4]=='u' and x[10]=='g') or (x[6]=='u' and x[0]=='g') or
    (x[8]=='u' and x[2]=='g') or (x[10]=='u' and x[4]=='g')):
        L3.append(x)
    else:
        L4.append(x)
```

Αρχικά δημιουργούμε δύο λίστες, τις L και L2, οι οποίες περιέχουν τις βάσεις για τα 6 συνολικά ζευγάρια που συναντώνται στην ακολουθία του RNA. Για παράδειγμα, η πρώτη θέση της λίστας L περιέχει το χαρακτήρα 'a' που αντιστοιχεί στην αδενίνη ενώ στην αντίστοιχη θέση στη λίστα L2 υπάρχει ο χαρακτήρας 'u' που αντιστοιχεί στην ουρακίλη, η οποία είναι η συμπληρωματική βάση της αδενίνης. Αντίστοιχα συμπληρώνονται με χαρακτήρες και οι υπόλοιπες 5 θέσεις σε καθε μία από τις δύο λίστες για τα

υπόλοιπα 5 ζευγάρια βάσεων που συναντώνται στο RNA. Ακόμη, γίνεται αρχικοποίηση της λίστας L1 η οποία, μέσω αναφορών στις δύο προαναφερθείσες λίστες, θα περιέχει όλους τους κανόνες της γραμματικής για την αναγνώριση της δομής. Οι κανόνες αυτοί, για λόγο που θα αναφερθεί στη συνέχεια, χωρίζονται, στο τέλος του προγράμματος, σε δύο λίστες, τις L3 και L4, οι οποίες αντίστοιχα περιέχουν τους κανόνες για τους ψευδοκόμβους με δεσμό γουανίνης-ουρακίλης και τους ψευδοκόμβους χωρίς αυτόν. Οι κανόνες της γραμματικής ήταν της μορφής S->aLcLuDuLgLa, όπου οι μικροί λατινικοί χαρακτήρες αντιστοιχούν στις βάσεις που δημιουργούν τα κύρια ζευγάρια της δομής (εν προκειμένω η αδενίνη, η οποία εκφράζεται μέσω του χαρακτήρα 'a' στην πρώτη θέση του κανόνα, δημιουργεί ζευγάρι με την ουρακίλη, η οποία αναπαρίσταται μέσω του χαρακτήρα 'u' στην έβδομη θέση του κανόνα, η κυτοσίνη, η οποία αναπαρίσταται με το χαρακτήρα 'c' στην τρίτη θέση του κανόνα, δημιουργεί ζευγάρι με τη γουανίνη, η οποία αναπαρίσταται με το χαρακτήρα 'g' στην ένατη θέση του κανόνα, και η ουρακίλη, η οποία αναπαρίσταται μέσω του χαρακτήρα 'u' στην πέμπτη θέση του κανόνα, δημιουργεί ζευγάρι με την αδενίνη, η οποία εκφράζεται μέσω του χαρακτήρα 'a' στην τελευταία θέση του κανόνα), ενώ οι κεφαλαίοι χαρακτήρες στο δεξί μέρος του κανόνα εκφράζουν τα μήκη των ακολουθιών που υπάρχουν ανάμεσα στις κύριες βάσεις. Ο χαρακτήρας S, όπως αναφέρθηκε και προηγουμένως, αναπαριστά το σύμβολο εκκίνησης της γραμματικής. Οι χαρακτήρες L του κανόνα εκφράζουν τα κενά ανάμεσα σε όλες τις κύριες βάσεις πλην των δύο ενδιάμεσων, δηλαδή της τρίτης και της τέταρτης. Από το σχήμα 2.9 φαίνεται ότι κάθε τέτοιο κενό μεταξύ των βάσεων πρέπει να έχει μήκος τουλάχιστον ένα οπότε προκύπτει και ο παρακάτω κανόνας για το χαρακτήρα L.

3.Κανόνας για το σύμβολο L της γραμματικής

```
L : 'a' L # L1 (1)
  | 'u' L # L2 (1)
  | 'c' L # L3 (1)
  | 'g' L # L4 (1)
  | 'a'   # 0
  | 'u'   # 0
  | 'c'   # 0
  | 'g'   # 0
  ;
```

Το μήκος της ακολουθίας μεταξύ των δύο ενδιάμεσων βάσεων της δομής, το οποίο εκφράζεται από το χαρακτήρα D στη γραμματική, έχει ένα άνω όριο το οποίο καθορίζεται από τη μεταβλητή max_dd_size, της οποίας μπορεί ο χρήστης να ορίσει την τιμή κατά την εκτέλεση του προγράμματος ενώ η default τιμή της, αν αυτή δεν οριστεί από το χρήστη, είναι 2. Ο κανόνας για το σύμβολο D της γραμματικής φαίνεται παρακάτω.

4.Κανόνας για το σύμβολο D της γραμματικής

```
 D:{% for x in range(max_dd_size) %}D{{ x }} {% endfor %} # M1
({% for x in range(max_dd_size) %}{{ x }} {% endfor %})
{% for x in range(max_dd_size) %}
D{{ x }} : E # N{{ x }} (0)
```

```
{% endfor %}
```

Το σύμβολο E, που φαίνεται στον παραπάνω κανόνα, εκφράζει τη βάση η οποία μπορεί να υπάρχει σε κάθε μία από τις θέσεις που ορίζονται από το άνω όριο του μήκους της ακολουθίας (τη μεταβλητή max_dd_size) στον κανόνα για το χαρακτήρα D. Οπότε, λόγω της έλλειψης περιορισμού ως προς το ποια βάση μπορεί να υπάρχει σε κάθε θέση, προκύπτει ο κανόνας για το συμβολο E που φαίνεται παρακάτω.

5.Κανόνας για το σύμβολο E της γραμματικής

```
E : 'a' # 0
  | 'u' # 0
  | 'c' # 0
  | 'g' # 0
  | ;
```

Τέλος, έχει υλοποιηθεί μία συνάρτηση η οποία παράγει την τελική γραμματική στην οποία στηρίζεται ο συντακτικός αναλυτής προκειμένου να παράξει όλα τα πιθανά δέντρα που περιέχουν έναν ψευδοκόμβο τύπου L. Η συνάρτηση αυτή παίρνει 2 παραμέτρους οι οποίες καθορίζουν την τελική μορφή της γραμματικής. Η μία από αυτές είναι η max_dd_size η οποία, όπως αναφέραμε πριν, καθορίζει το άνω όριο που μπορεί να πάρει η τιμή του μήκους της ακολουθίας ανάμεσα στις δύο ενδιάμεσες κύριες βάσεις του ψευδοκόμβου. Η δεύτερη μεταβλητή είναι η δυαδική μεταβλητή allow-ug η οποία ανάλογα με την τιμή της (True ή False) καθορίζει αν θα περιλαμβάνονται οι ψευδοκόμβοι με δεσμούς γουανίνης και ουρακίλης (u-g) στην τελική γραμματική. Αυτός ήταν και ο λόγος για τον οποίο έγινε ο διαχωρισμός παραπάνω κατά τη δημιουργία όλων των κανόνων της γραμματικής. Η προαναφερθείσα συνάρτηση καθορισμού της τελικής μορφής της γραμματικής φαίνεται παρακάτω.

6.Συνάρτηση παραγωγής της γραμματικής με βάση τους παραπάνω κανόνες

```
def generate_grammar(allow_ug: bool, max_dd_size: int) -> str:
    """
    generate grammar for pseudoknot detection, based on parameters.
    """
    return (
        jinja2.Environment()
        .from_string(TEMPLATE)
        .render(
            allow_ug=allow_ug,
            max_dd_size=max_dd_size,
        )
)
```

4.2.2 Ανάλυση της γραμματικής και παραγωγή του συντακτικού δέντρου

Η διαδικασία αυτή επιτεύχθηκε μέσω της συνάρτησης parse[50] η οποία δεχόταν ως είσοδο τη γραμματική και επέστρεφε έναν κωδικό λάθους ο οποίος υποδείκνυε αν

υπήρξε κάποιο σφάλμα στη διαδικασία. Αν ο κωδικός αυτός ήταν μηδέν, τότε αυτό σήμαινε ότι η διαδικασία κύλησε ομαλά, οπότε η συνάρτηση επέστρεφε και τη ρίζα του δέντρου το οποίο περιείχε όλα τα πιθανά υποδέντρα που αναπαριστούσαν τους ψευδοκόμβους τύπου L στην ακολουθία του RNA. Επίσης η συνάρτηση δεχόταν ένα σημείο εισόδου το οποίο ήταν το σημείο εκκίνησης της υποακολουθίας της συμβολοσειράς για την οποία γινόταν η συντακτική ανάλυση. Ο κώδικας της συγκεκριμένης συνάρτησης παρατίθεται παρακάτω.

7.Συνάρτηση ανάλυσης της γραμματικής και παραγωγής του συντακτικού δέντρου

```
struct yaep_tree_node *parse(const char *description) {
  struct grammar *g;
  struct yaep_tree_node *root;
  int ambiguous_p;

  if ((g = yaep_create_grammar()) == NULL) {
    fprintf(stderr, "yaep_create_grammar: No memory\n");
    exit(1);
  }
  yaep_set_debug_level(g, 0);
  yaep_set_one_parse_flag(g, 0);
  yaep_set_cost_flag(g, 0);
  yaep_set_lookahead_level(g, 2);
  if (yaep_parse_grammar(g, TRUE, description) != 0) {
    fprintf(stderr, "%s\n", yaep_error_message(g));
    exit(1);
  }
  int parsed = yaep_parse(g, read_token_func, syntax_error_func,
                          parse_alloc_func, NULL, &root, &ambiguous_p);
  if (parsed) {
    printf("this should not be accepted\n");
    fprintf(stderr, "yaep_parse: %s\n", yaep_error_message(g));
  }
  return root;
}
```

Η πρώτη συνάρτηση που καλείται[50] είναι η συνάρτηση yaep_create_grammar, η οποία δημιουργεί έναν αναλυτή με απροσδιόριστη γραμματική. Στην περίπτωση όπου δεν υπάρχει μνήμη για την πραγματοποίηση της διαδικασίας της ανάλυσης, τότε η συνάρτηση αυτή επιστρέφει NULL. Η συνάρτηση yaep_set_debug_level δέχεται ως όρισμα έναν ακέραιο ο οποίος καθορίζει το κατά πόσο θα εμφανίζονται στην έξοδο πληροφορίες που αφορούν το δέντρο ανάλυσης, τη γραμματική καθώς και άλλα δεδομένα σχετικά με αυτήν τη διαδικασία. Αν η τιμή του ορίσματος εισόδου ισούται με μηδέν τότε δεν εμφανίζονται καθόλου αυτές οι πληροφορίες στην έξοδο. Η συνάρτηση yaep_set_one_parse_flag καθορίζει το εάν θα χτιστεί ένα δέντρο μετάφρασης της

γραμματικής το οποίο συμβαίνει για τιμή 0 της ακέραιας παραμέτρου ή πολλαπλά συντακτικά δέντρα για τις γραμματικές οι οποίες είναι αμφίδρομες. Αν η γραμματική δεν είναι αμφίδρομη τότε η τιμή της ακέραιας παραμέτρου δεν επηρεάζει το αποτέλεσμα. Η συνάρτηση yaep_set_cost_flag λαμβάνει επίσης ως όρισμα μία ακέραια μεταβλητή και αν η τιμή αυτής είναι 0 τότε παράγει μόνο ένα δέντρο μετάφρασης με ελάχιστο κόστος, εφόσον η γραμματική είναι αμφίδρομη, αλλιώς παράγει πολλαπλά δέντρα. Για μη αμφίδρομες γραμματικές, ομοίως με την προηγούμενη συνάρτηση, η τιμή της ακέραιας παραμέτρου δεν επηρεάζει το αποτέλεσμα. Η συνάρτηση yaep_set_lookahead_level καθορίζει, επίσης μέσω μίας ακέραιας παραμέτρου, το επίπεδο χρήσης lookaheads κατά τη διαδικασία ανάλυσης της γραμματικής. Η τιμή 0 δηλώνει ότι δε θα γίνει καθόλου χρήση lookaheads. Η τιμή 1 δηλώνει ότι τα lookaheads θα χρησιμοποιηθούν ανεξάρτητα από το σημείο εισόδου το οποίο δίνεται στην αρχική συνάρτηση και δίνει τα καλύτερα αποτελέσματα σε σχέση με τη μνήμη που δαπανάται συνολικά και την ταχύτητα με την οποία γίνεται η διαδικασία. Τέλος, η τιμή 2 δηλώνει ότι τα lookaheads θα χρησιμοποιηθούν δυναμικά που σημαίνει ότι θα λάβουν υπ' όψιν και το σημείο εισόδου. Η συνάρτηση yaep_parse_grammar συντονίζει τον αναλυτή που δημιουργήθηκε με τη γραμματική ενώ η συνάρτηση yaep_parse είναι η κύρια συνάρτηση που πραγματοποιεί τη διαδικασία ανάλυσης της γραμματικής και παραγωγής των δέντρων επιστρέφοντας και τον κωδικό λάθους. Το πρώτο όρισμα που δέχεται η συγκεκριμένη συνάρτηση είναι η γραμματική η οποία μέσω της συνάρτησης συντονισμού που αναφέραμε παραπάνω απεικονίζεται πλέον στη μεταβλητή g. Το δεύτερο όρισμα, η συνάρτηση read_token_func, επιστρέφει στην κύρια συνάρτηση το σημείο εισόδου της συμβολοσειράς μαζί με ένα χαρακτηριστικό του. Αν η συνάρτηση επιστρέψει αρνητική τιμή, τότε έχουμε διαβάσει όλα τα σημεία εισόδου οπότε έχει ολοκληρωθεί η ανάλυση. Το τρίτο όρισμα, η συνάρτηση syntax_error_func, καλείται εφόσον αναγνωριστεί ότι υπάρχει κάποιο συντακτικό λάθος στη διαδικασία. Η συνάρτηση parse_alloc_func κάνει την απαραίτητη δέσμευση μνήμης για να γίνει η διαδικασία ανάλυσης και δέχεται ως όρισμα έναν ακέραιο που αναπαριστά το μέγεθος της μνήμης που θα δεσμευτεί. Το επόμενο όρισμα είναι η συνάρτηση parse_free η οποία αφορά την απελευθέρωση της μνήμης που δεσμεύεται μέσω της συνάρτησης parse_alloc και στη συγκεκριμένη υλοποίηση αρχικοποιείται με NULL. Τέλος, η μεταβλητή ambiguous_p δείχνει εάν η γραμματική εισόδου είναι διφορούμενη ενώ η μεταβλητή root είναι αυτή η οποία εν τέλει επιστρέφεται από την αρχική συνάρτηση και αναπαριστά τη ρίζα του συντακτικού δέντρου.

4.2.3 Διάσχιση των δέντρων ανάλυσης για την αναγνώριση των ψευδοκόμβων τύπου L

Το επόμενο βήμα, μετά την ανάλυση της γραμματικής και τη δημιουργία του συντακτικού δέντρου ανάλυσης, ήταν η διάσχιση του δέντρου αυτού προκειμένου να εντοπισθούν ακριβώς όλοι οι ψευδοκόμβοι που υπήρχαν στην ακολουθία του RNA. Προκειμένου να επιτευχθεί αυτό, αρχικά δημιουργήθηκε ένα struct το οποίο αναπαριστούσε τη συνδεδεμένη λίστα που θα περιέχει όλους τους ψευδοκόμβους που θα προκύψουν από τη διάσχιση του δέντρου ανάλυσης και για κάθε ψευδοκόμβο της λίστας είχε τα βασικά μεγέθη που τον χαρακτήριζαν.Για τον τύπο L το struct περιείχε τις εξής 4 μεταβλητές:

1)Τη μεταβλητή left_left_loop_size, η οποία εκφράζει το μήκος της ακολουθίας ανάμεσα στην πρώτη και στη δεύτερη κύρια βάση

2)Τη μεταβλητή left_right_loop_size, η οποία εκφράζει το μήκος της ακολουθίας ανάμεσα στη δεύτερη και στην τρίτη κύρια βάση

3)Τη μεταβλητή dd_size, η οποία εκφράζει το μήκος της ακολουθίας ανάμεσα στις δύο ενδιάμεσες βάσεις του ψευδοκόμβου τύπου L, δηλαδή την τρίτη και την τέταρτη κύρια βάση

4)Τη μεταβλητή right_left_loop_size η οποία εκφράζει το μήκος της ακολουθίας ανάμεσα στην τέταρτη και στην πέμπτη κύρια βάση

Ακόμη, το struct περιείχε τη μεταβλητή next, η οποία ήταν ίδιου τύπου με το struct, και αναπαριστούσε τον επόμενο ψευδοκόμβο της λίστας. Παρακάτω φαίνεται το struct με τις προαναφερθείσες κύριες μεταβλητές για τον τύπο L.

8.Ορισμός της δομής του ψευδοκόμβου για τον τύπο L

```
struct pseudoknot {
  int left_left_loop_size;
  int left_right_loop_size;
  int right_left_loop_size;
  int dd_size;
  struct pseudoknot *next;
};
```

Προτού αναφερθούμε στον τρόπο με το οποίο διασχίστηκε το δέντρο ανάλυσης θα κάνουμε μία αναφορά στα είδη των κόμβων που περιέχονται στα συγκεκριμένα δέντρα καθώς και σε κάποια χαρακτηριστικά ορισμένων από τους κόμβους.

ΕΙΔΗ ΚΟΜΒΩΝ ΣΤΑ YAEP TREES

Οι κατηγορίες των κόμβων στα συγκεκριμένα δέντρα[50] περιέχονται στο enumeration yaep_tree_node_type που αναπαριστά όλους τους πιθανούς κόμβους του αφηρημένου συντακτικού δέντρου που προκύπτει από την ανάλυση της γραμματικής και είναι οι εξής:

1)YAEP_NIL:Αναπαριστά κόμβο με κενή μετάφραση

2)YAEP_ERROR:Αναπαριστά κόμβο με μετάφραση λάθους

3)YAEP_TERM:Αναπαριστά έναν τερματικό κόμβο

4)YAEP_ANODE:Αναπαριστά έναν αφηρημένο κόμβο

5)YAEP_ALT:Αυτός ο τύπος κόμβου μας δείχνει ότι υπάρχουν περισσότερες από μία πιθανές μεταφράσεις του συγκεκριμένου κόμβου.

Προκειμένου να γίνεται αναφορά στον κάθε κόμβο ανάλογα με το είδος του υπάρχει επίσης ένα union με το όνομα val το οποίο περιέχει 5 μέλη ,ένα για κάθε είδος κόμβου.Τα 5 αυτά μέλη είναι τα εξής:

1)nil για τον κόμβο YAEP_NIL

2)error για τον κόμβο YAEP_ERROR

3)term για τον κόμβο YAEP_TERM

4)anode για τον κόμβο YAEP_ANODE

5)alt για τον κόμβο YAEP_ALT

Κάποια από τα παραπάνω είδη κόμβων περιέχουν κάποιες εσωτερικές μεταβλητές οι οποίες εκφράζουν ορισμένα ιδιαίτερα χαρακτηριστικά τους. Ο κόμβος τύπου term περιέχει την ακέραια μεταβλητή code που δείχνει τον κωδικό του συγκεκριμένου τερματικού κόμβου καθώς και τη μεταβλητή attr τύπου *void η οποία είναι αναφορά σε ένα γνωρισμα του συγκεκριμένου κόμβου. Ο κόμβος τύπου alt περιέχει τη μεταβλητή node τύπου yaep_tree_node η οποία αναφέρεται στην πρώτη εναλλακτική μετάφραση του και τη μεταβλητή next η οποία είναι του ίδιου τύπου και αναφέρεται στην επόμενη εναλλακτική μετάφραση. Τέλος, ο κόμβος τύπου anode περιέχει τη μεταβλητή name τύπου const char που εκφράζει το όνομα του κόμβου όπως δίνεται στη μετάφραση του κανόνα, τη μεταβλητή children που είναι τύπου yaep_tree_node και αναφέρεται στους κόμβους που είναι μεταφράσεις των συμβόλων που περιέχονται στον κανόνα της γραμματικής μαζί με τον αφηρημενο κόμβο και ουσιαστικά αποτελούν τα παιδιά του κόμβου σύμφωνα με τον κανόνα, καθώς και την ακέραια μεταβλητή cost η οποία εκφράζει ανάλογα με την τιμή μιας μεταβλητής σημαίας είτε το κόστος του συγκεκριμένου κόμβου μαζί με το κόστος όλων των παιδιών του είτε μόνο το κόστος του συγκεκριμένου αφηρημένου κόμβου.

Σχήμα 4.2: *Κανόνας γραμματικής για τον ψευδοκόμβο τύπου L[1]*

Στον παραπάνω κανόνα της γραμματικής για τον ψευδοκόμβο τύπου L, το όνομα του αφηρημένου κόμβου είναι η συμβολοσειρά R01 ενώ η λίστα (1 3 5 7 9) είναι η λίστα που δείχνει τις θέσεις των παιδιών του κόμβου στον κανόνα.

Προκειμένου να διασχισθεί το δέντρο ανάλυσης, έχει υλοποιηθεί μία συνάρτηση μέσω της οποίας γίνεται έλεγχος για τον κάθε κόμβο που συναντάμε στο δέντρο και ο έλεγχος αυτός αφορά το είδος του κόμβου που συναντάμε κάθε φορά. Αν ο κόμβος είναι τύπου YAEP_NIL, YAEP_ERROR ή YAEP_TERM τότε η συνάρτηση επιστρέφει NULL καθώς δε μπορεί να κάνει κάποιου είδους διάσχιση σε αυτούς τους κόμβους, σύμφωνα και με τη γραμματική.Στην περίπτωση που ο κόμβος είναι τύπου YAEP_ALT, τότε εφαρμόζουμε αναδρομικά την ίδια συνάρτηση τόσο για την πρώτη μετάφραση του κόμβου

όσο και για την επόμενη μετάφραση του, αν αυτή υπάρχει, και προσθέτουμε τους ψευδοκόμβους που προκύπτουν από αυτές τις κλήσεις της συνάρτησης στην τελική λίστα. Τέλος, αν ο κόμβος ανήκει στην κατηγορία YAEP_ANODE, τότε αν ο πρώτος χαρακτήρας του ονόματος του δεν είναι 'R' λόγω εντοπισμού λανθασμένου κόμβου σύμφωνα με τη υλοποιημένη γραμματική επιστρέφεται NULL καθώς όλα τα ονόματα των κανόνων στη γραμματική ξεκινούν με το χαρακτήρα R. Αντίθετα, αν ο πρώτος χαρακτήρας του ονόματος του κόμβου είναι 'R', οπότε το όνομα του κόμβου είναι σωστό σύμφωνα με τη γραμματική, τότε υπολογίζουμε όλα τα μεγέθη που περιέχονται στο struct της δομής. Πιο αναλυτικά, αρχικά καλούμε τη συνάρτηση traverse_parse_tree_for_loop, η οποία θα αναλυθεί παρακάτω, για το πρώτο παιδί σύμφωνα με τον κανόνα δηλαδή το χαρακτήρα στη θέση 1, αφού η λίστα ήταν (1 3 5 7 9) όπως αναφέραμε παραπάνω, ο οποίος ήταν ο πρώτος χαρακτήρας L οπότε και υπολογιζόταν το μήκος της ακολουθίας χαρακτήρων ανάμεσα στην πρώτη και τη δεύτερη κύρια βάση. Έπειτα καλούσαμε την ίδια συνάρτηση για το δεύτερο παιδί του κόμβου σύμφωνα με τον κανόνα, δηλαδή το χαρακτήρα L στη θέση 3 οπότε και υπολογιζόταν το μήκος της ακολουθίας ανάμεσα στη δεύτερη και στην τρίτη κύρια βάση. Στη συνέχεια καλούσαμε τη συνάρτηση traverse_parse_tree_for_dd για το τρίτο παιδί του κανόνα, δηλαδή το χαρακτήρα D στη θέση 5, οπότε και βρίσκαμε το μήκος της ακολουθίας ανάμεσα στις δύο ενδιάμεσες βάσεις για τον τύπο L, την τρίτη και την τέταρτη. Τέλος, καλούσαμε πάλι τη συνάρτηση traverse_parse_tree_for_loop για το τέταρτο παιδί του κανόνα, δηλαδή το χαρακτήρα L στη θέση 7, οπότε έτσι υπολογίζαμε το μήκος της ακολουθίας ανάμεσα στην τέταρτη και την πέμπτη κύρια βάση. Η διαδικασία αυτή πραγματοποιήθηκε εντός της συνάρτησης traverse_parse_tree η οποία φαίνεται παρακάτω.

9.Συνάρτηση διάσχισης των δέντρων και εντοπισμού των ψευδοκόμβων στην ακολουθία του RNA

```
struct pseudoknot *traverse_parse_tree(struct yaep_tree_node *node) {
  int left_left_loop_size, left_right_loop_size, right_left_loop_size;
  struct size *dd_sizes;
  struct pseudoknot *pknots = NULL;
  struct pseudoknot *pseudoknots = NULL;
  if (!node) {
    return pseudoknots;
  }
  switch (node->type) {
  case YAEP_ERROR:
    return NULL;
  case YAEP_NIL:
    return NULL;
  case YAEP_TERM:
    return NULL;
  case YAEP_ANODE:
    if ((node->val.anode.name)[0] != 'R') {
```

```
      return NULL;
    }
    left_left_loop_size = traverse_parse_tree_for_loop(node->val.
    anode.children[0]);
    left_right_loop_size  = traverse_parse_tree_for_loop(node->val.
    anode.children[1]);
    dd_sizes = traverse_parse_tree_for_dd(node->val.
    anode.children[2]);
    right_left_loop_size = traverse_parse_tree_for_loop(node->val.
    anode.children[3]);

    while (dd_sizes != NULL) {
      pseudoknots = append_pseudoknot_if_not_exists(
          pseudoknots, create_pseudoknot(left_left_loop_size,
          left_right_loop_size, right_left_loop_size,
          dd_sizes->value));
      dd_sizes = dd_sizes->next;
    }
    return pseudoknots;
  case YAEP_ALT:
    pknots = traverse_parse_tree(
        node->val.alt.node);
    pseudoknots = NULL;
    if (node->val.alt.next != NULL) {
      pseudoknots = traverse_parse_tree(node->val.alt.next);
    }
    pseudoknots = concatenate_punique(pseudoknots, pknots);
    return pseudoknots;
  default:
    printf("I don't care! \n");
    return pseudoknots;
  }
}
```

Η συνάρτηση create_pseudoknot δημιουργεί έναν νέο ψευδοκόμβο εισάγοντας ως χαρακτηριστικά του τις τιμές που υπολογίζει προηγουμένως, η συνάρτηση append-pseudoknot-if-not-exists συγκρίνει τα χαρακτηριστικά του ψευδοκόμβου με τα χαρακτηριστικά όλων των ψευδοκόμβων που υπάρχουν ήδη στην τελική λίστα. Αν εντοπιστεί ένας ψευδοκόμβος που να έχει ακριβώς ίδιες όλες τις τιμές χαρακτηριστικών με τον ψευδοκόμβο που θέλουμε να προσθέσουμε στη λίστα τότε η λίστα των ψευδοκόμβων παραμένει όπως είναι, αλλιώς ο νέος ψευδοκόμβος προστίθεται στην αρχή της λίστας. Τέλος, η συνάρτηση concatenate_punique ενώνει σε μία λίστα τους ψευδοκόμβους που προκύπτουν από τις αναδρομικές κλήσεις της ίδιας συνάρτησης, για

τις διαφορετικές μεταφράσεις των κόμβων που είναι τύπου YAEP_ALT, δημιουργώντας έτσι μία ενιαία λίστα με όλους τους ψευδοκόμβους.

Η συνάρτηση traverse_parse_tree_for_loop, η οποία χρησιμοποιήθηκε προκειμένου να υπολογιστούν τα μήκη των ακολουθιών πίσω από τους χαρακτήρες L της γραμματικής, έκανε κατά βάθος διάσχιση των υποδέντρων που αφορούσαν τους κόμβους με L του συνολικού δέντρου. Πιο αναλυτικά, αν ο κόμβος στον οποίο βρισκόταν ήταν τύπου YAEP_ANODE, τότε καλούσε αναδρομικά τη συνάρτηση για το πρώτο παιδί σύμφωνα με τον κανόνα. Αν ο κανόνας είχε δύο χαρακτήρες ως παιδιά, τότε το πρώτο παιδί ήταν ο χαρακτήρας στη θέση 1, άρα ο επόμενος χαρακτήρας L, οπότε και συνεχιζόταν η κατά βάθος διάσχιση. Αντίθετα, αν ο κόμβος είχε μόνο ένα παιδί, τότε θα καλέσουμε τη συνάρτηση για το παιδί αυτό το οποίο θα είναι τερματικός χαρακτήρας οπότε και η συνάρτηση θα επιστρέψει την τιμή 1. Επίσης προσθέτουμε 1 στην τελική τιμή που επιστρέφεται από τη συνάρτηση αφού κάθε φορά πηγαίνουμε στην επόμενη θέση στο δέντρο. Η υλοποίηση της συνάρτησης αυτής φαίνεται παρακάτω.

10.Συνάρτηση κατά βάθος διάσχισης του δέντρου για τον υπολογισμό των ακολουθιών που αντικατοπτρίζονται από τους χαρακτήρες L των κανόνων της γραμματικής

```
int traverse_parse_tree_for_loop(struct yaep_tree_node *node) {
  switch (node->type) {
  case YAEP_ANODE:
    return traverse_parse_tree_for_loop(node->val.anode.children[0]) + 1;
    break;
  case YAEP_TERM:
    return 1;
    break;
  default:
    return -1;
    break;
  }
}
```

Η συνάρτηση traverse_parse_tree_for_dd επίσης έκανε έλεγχο για το είδος του κάθε κόμβου και εν τέλει επέστρεφε μία λίστα με όλες τις πιθανές τιμές για το μήκος της ακολουθίας ανάμεσα στις δύο ενδιάμεσες κύριες βάσεις που εκφραζόταν από το χαρακτήρα D της γραμματικής. Ο λόγος για τον οποίο η συνάρτηση επιστρέφει λίστα από τιμές, και όχι μόνο μία τιμή όπως η προηγούμενη συνάρτηση, αφορά τον τρόπο με τον οποίο ο parser δημιουργεί το δέντρο ανάλυσης. Αν ο κόμβος ήταν τύπου YAEP_ALT, τότε όμοια με πριν η συνάρτηση καλούνταν αναδρομικά και για τις δύο μεταφράσεις του κόμβου. Στην περίπτωση που ο κόμβος ήταν τύπου YAEP_ANODE, τότε, αν ο πρώτος χαρακτήρας του ονόματος του κανόνα ήταν 'M', τότε η συνάρτηση καλούνταν αναδρομικά για κάθε ένα από τα παιδιά του. Αν ο πρώτος χαρακτήρας ήταν 'N' τότε πάλι η συνάρτηση καλούνταν αναδρομικά για το μοναδικό παιδί το οποίο εκφραζόταν μέσω του κανόνα για το χαρακτήρα 'E' που αναλύθηκε παραπάνω. Αν ο κόμβος ήταν τύπου YAEP_TERM τότε η συνάρτηση δημιουργούσε ένα αντικείμενο τύπου size και του ανέ-

θετε την τιμή 1, λόγω της ύπαρξης κόμβου, ενώ για κόμβο τύπο YAEP_NIL ανέθετε την τιμή 0 για να δείξει αντίστοιχα τη μη ύπαρξη κόμβου σε εκείνη τη θέση. Η υλοποίηση της συνάρτησης αυτής φαίνεται παρακάτω.

11.Συνάρτηση υπολογισμού των μηκών μεταξύ των ενδιάμεσων βάσεων που αναπαρίστανται στους κανόνες της γραμματικής με το χαρακτήρα D

```
struct size *traverse_parse_tree_for_dd(struct yaep_tree_node *node) {
  switch (node->type) {
  case YAEP_ANODE:
    if ((node->val.anode.name)[0] == 'M') {
      struct size **childSizes =
      malloc(s_max_dd_size * sizeof(struct size *));
      for (int i = 0; i < s_max_dd_size; i++) {
        childSizes[i] = traverse_parse_tree_for_dd(node->
        val.anode.children[i]);
      }
      return multiCartesianProduct(childSizes, s_max_dd_size);
    } else if ((node->val.anode.name)[0] == 'N') {
      return traverse_parse_tree_for_dd(node->val.
      anode.children[0]);
    }
    break;
  case YAEP_TERM:
    return create_size(1);
    break;
  case YAEP_NIL:
    return create_size(0);
    break;
  case YAEP_ALT:;
    struct size *anode_dds = traverse_parse_tree_for_dd(node->val.alt.node);
    struct size *alt_dds = NULL;
    if (node->val.alt.next != NULL) {
      alt_dds = traverse_parse_tree_for_dd(node->val.alt.next);
    }
    struct size *new_list = concatenate_sunique(anode_dds, alt_dds);
    return new_list;
    break;
  default:
    printf("I think something went really wrong with node type \n");
  }
  return NULL;
}
```

Η συνάρτηση concatenate_sunique ενώνει σε μία λίστα τους ακεραίους που προέκυπταν ως αποτελέσματα από τις αναδρομικές κλήσεις της συνάρτησης για τις διαφορετικές μεταφράσεις σε περίπτωση που ο κόμβος είναι τύπου YAEP_ALT δημιουργώντας έτσι μία ενιαία λίστα με όλες τις πιθανές τιμές του συγκεκριμένου μήκους ακολουθίας.

Η παραπάνω διαδικασία ανάλυσης της συμβολοσειράς και εντοπισμού όλων των ψευδοκόμβων πραγματοποιείται εντός μίας διπλής επανάληψης μέσω της οποίας καθορίζεται για κάθε ψευδοκόμβο τόσο το σημείο εκκίνησης του, το οποίο δίνεται ως όρισμα στη συνάρτηση ανάλυσης της συμβολοσειράς, όσο και το συνολικό μέγεθος του. Πιο συγκεκριμένα, οι μεταβλητές left και right, οι οποίες υπάρχουν μέσα στη διπλή επανάληψη, καθορίζουν τα παράθυρα της ακολουθίας εισόδου στα οποία ο parser κάθε φορά θα αναζητά τους ψευδοκόμβους, εφόσον αυτοί υπάρχουν. Το γεγονός ότι γνωρίζουμε τα άκρα του κάθε ψευδοκόμβου, μέσω των μεταβλητών left και right της διπλής επανάληψης, μας απαλλάσσει από την ανάγκη να υπολογίσουμε το μήκος της ακολουθίας ανάμεσα στις δύο τελευταίες κύριες βάσεις για τον τύπο L, δηλαδή την πέμπτη και την έκτη. Γι' αυτό και δεν έχουμε συμπεριλάβει στο struct του ψευδοκόμβου, που αναφέραμε παραπάνω, κάποια μεταβλητή που να εκφράζει αυτήν την απόσταση. Έπειτα, για κάθε ψευδοκόμβο που εντοπίζεται, γίνεται έλεγχος της μεταβλητής dd_size, η οποία εκφράζει το μήκος της ακολουθίας μεταξύ των δύο ενδιάμεσων κύριων βάσεων. Αν η τιμή της είναι κάτω από ένα συγκεκριμένο όριο, το οποίο εκφράζεται μέσω της μεταβλητής min_dd_size, τότε ο συγκεκριμένος ψευδοκόμβος απορρίπτεται. Η μεταβλητή min_dd_size μπορεί να οριστεί από το χρήστη κατά τη διάρκεια εκτέλεσης του προγράμματος ενώ η αρχική τιμή της, αν ο χρήστης δεν την ορίσει, είναι 0. Οι υπόλοιποι ψευδοκόμβοι επιστρέφονται σε μία συνάρτηση η οποία είναι υλοποιημένη σε python και από την οποία γίνονται όλες οι κλήσεις των συναρτήσεων σε c προκειμένου να γίνει η ανάλυση της ακολουθίας του RNA. Το τμήμα κώδικα που υλοποιεί την παραπάνω διαδικασία είναι το παρακάτω:

12.Κώδικας για τη συνολική διαδικασία υπολογισμού των ψευδοκόμβων και επιστροφή των χαρακτηριστικών τους στο αρχικό πρόγραμμα της python

```
for (int right = len - 1; right >= min_window_size - 1; right--) {
    for (int left = right - min_window_size + 1;
         left > right - max_window_size && left >= 0; left--) {
      s_ntok = left;
      struct yaep_tree_node *root = parse(s_definition);
      struct pseudoknot *ps = traverse_parse_tree(root);

      for (struct pseudoknot *i = ps; i != NULL; i = i->next) {
        if (i->dd_size < s_min_dd_size) {
          continue;
        }
        cb(left, right - left + 1, i->left_left_loop_size,
        i->left_right_loop_size, i->right_left_loop_size,
```

```
          i->dd_size);
        }
      }
      s_input[right] = '\0';
    }
}
```

Μέσω της τελευταίας εντολής του παραπάνω κωδικά αφαιρούμε το τελευταίο χαρακτήρα από τη δεξιά μεριά της ακολουθίας του RNA προκειμένου να τροποποιούμε καθε φορά το δεξί όριο του παραθύρου αναζήτησης των ψευδοκόμβων. Μέσω της συνάρτησης cb επιστρέφονται για κάθε ψευδοκόμβο τα χαρακτηριστικά του στον κώδικα της python προκειμένου στη συνέχεια να γίνει η περαιτέρω επεξεργασία και να απεικονιστεί η δομή σε μορφή dot bracket.Η εντολή μέσω της οποίας γίνεται η κλήση της συνάρτησης σε c από τον αντίστοιχο κώδικα της python είναι η παρακάτω:

13.Εντολή κλήσης της συνάρτησης σε c από τη συνάρτηση σε python

```
for (i, j, left_left_loop_size, left_right_loop_size,
right_left_loop_size, dd_size) in parser(sequence):
```

Η μεταβλητή parser επίσης μπορεί να τροποποιηθεί από το χρήστη κατά τη διάρκεια εκτέλεσης του προγράμματος. Οι δύο πιθανές τιμές που μπορεί να λάβει είναι yaep και bruteforce οι οποίες καθορίζουν αντίστοιχα την επιλογή είτε του αλγορίθμου με χρήση συντακτικής αναγνώρισης προτύπων είτε του άπληστου αλγορίθμου δυναμικού προγραμματισμού για τη διαδικασία εντοπισμού των ψευδοκόμβων. Η αρχική τιμή της μεταβλητής, αν αυτή δεν οριστεί κατά την εκτέλεση του προγράμματος, είναι yaep οπότε και καλείται ο αλγόριθμος που κάνει χρήση συντακτικών δέντρων για την επίλυση του προβλήματος. Αντίθετα, με τη χρήση του ορίσματος --parser bruteforce κατά την εκτέλεση του προγράμματος θα εκτελεστεί ο άπληστος αλγόριθμος προκειμένου να επιλυθεί το πρόβλημα. Αφού ολοκληρώθηκε η παρουσίαση της μεθοδολογίας συντακτικής αναγνώρισης προτύπων για τον εντοπισμό όλων των ψευδοκόμβων στην ακολουθία του RNA, θα παρουσιάσουμε στη συνέχεια τον τρόπο επίλυσης του ίδιου προβλήματος με χρήση του άπληστου αλγορίθμου δυναμικού προγραμματισμού.

4.3 Άπληστος Αλγόριθμος Δυναμικού Προγραμματισμού

Στο συγκεκριμένο αλγόριθμο, αρχικά έχουν δημιουργηθεί δύο δομές προκειμένου να διατηρούνται πληροφορίες οι οποίες αφορούν τους δεσμούς που εντοπίζονται μεταξύ των βάσεων στην ακολουθία του RNA. Η πρώτη δομή, η οποία καλείται stem, περιέχει δύο ακέραιες μεταβλητές, τις left και right οι οποίες εκφράζουν τις θέσεις των δύο βάσεων, που δημιουργούν ζευγάρι κάθε φορά μεταξύ τους, στην ακολουθία του RNA. Η δεύτερη δομή καλείται core_stem και περιέχει έναν πίνακα τύπου stem ο οποίος είναι τριών θέσεων. Στο συγκεκριμένο πίνακα θα αποθηκεύονται οι θέσεις των βάσεων για κάθε έναν από τους τρεις κύριους δεσμούς οι οποίοι εκφράζουν τον ψευδοκόμβο τύπου L. Το πρώτο βήμα σε αυτή τη διαδικασία είναι να εντοπιστούν όλα τα ζευγάρια βάσεων που σχηματίζονται διασχίζοντας την ακολουθία του RNA. Οπότε, για κάθε χαρακτήρα της συμβολοσειράς που εκφράζει την αντίστοιχη βάση, ελέγχεται αν υπάρχει βάση σε θέση μεγαλύτερη από τη δική του με την οποία να δημιουργεί ζευγάρι. Επιπλέον έχει γίνει αρχικοποίηση ενός πίνακα μεγέθους ίσου με το τετράγωνο του μήκους της ακολουθίας προκειμένου να αποθηκεύονται όλα τα ζευγάρια που εντοπίζονται στην ακολουθία του RNA, καθώς και μίας ακέραιας μεταβλητής, της n_stems, η οποία αυξάνεται κάθε φορά που εντοπίζεται ένα ζευγάρι βάσεων. Ο κώδικας για τη διαδικασία αυτή φαίνεται παρακάτω.

14.Κώδικας μέσω του οποίου εντοπίζοντι όλα τα ζευγάρια βάσεων που σχηματίζονται στην ακολουθία του RNA

```
for (int i = 0; i < n; i++) {
    switch (sequence[i]) {
    case 'a':
      for (int j = i + 1; j < n; j++)
        if (sequence[j] == 'u') {
          cs_position->left = i;
          cs_position->right = j;
          cs_position++;
          n_stems++;
        }
      break;
    case 'u':
      for (int j = i + 1; j < n; j++)
        if ((sequence[j] == 'a') || ((sequence[j] == 'g') && s_allow_ug)) {
          cs_position->left = i;
          cs_position->right = j;
          cs_position++;
          n_stems++;
        }
      break;
    case 'c':
```

```
      for (int j = i + 1; j < n; j++)
        if (sequence[j] == 'g') {
          cs_position->left = i;
          cs_position->right = j;
          cs_position++;
          n_stems++;
        }
      break;
    case 'g':
      for (int j = i + 1; j < n; j++)
        if ((sequence[j] == 'c') || ((sequence[j] == 'u') && s_allow_ug)) {
          cs_position->left = i;
          cs_position->right = j;
          cs_position++;
          n_stems++;
        }
      break;
    default:
      printf("Invalid character\n");
    }
}
```

Όπως φαίνεται στην παραπάνω συνάρτηση, κάθε φορά που βρίσκουμε ένα ζευγάρι αποθηκεύουμε στη μεταβλητή left τη θέση της βάσης που βρίσκεται πιο αριστερά στο ζευγάρι, και στη μεταβλητή right τη θέση της βάσης που βρίσκεται πιο δεξιά στο ζευγάρι. Επίσης, πηγαίνουμε κάθε φορά στην επόμενη θέση του πίνακα αυξάνοντας τη μεταβλητή cs_position ενώ αυξάνουμε κατά 1 και τη μεταβλητή n_stems, στην οποία κρατάμε το συνολικό αριθμό ζευγαριών που υπάρχουν στη δομή.

Αφού βρούμε όλα τα ζευγάρια μεταξύ βάσεων που σχηματίζονται, τα ελέγχουμε ανά τριάδες για να διαπιστώσουμε αν η θέση των βάσεων είναι τέτοια ώστε να σχηματίζεται η δομή του ψευδοκόμβου τύπου L. Ο κώδικας στον οποίο πραγματοποιείται αυτή η διαδικασία φαίνεται παρακάτω.

15.Διαδικασία ελέγχου όλων των ζευγαριών ανά τριάδες και αποθήκευσης όλων των ψευδοκόμβων που σχηματίζονται

```
for (int i=0; i<n_stems; i++) {
    cs_position = all_stems + i;
    for (int j=i+1; j<n_stems; j++) {
      cs_position2 = all_stems+j;
      for (int k=j+1; k<n_stems; k++) {
        cs_position3 = all_stems + k;
        if ((cs_position->left<cs_position2->left) &&
            (cs_position2->left<cs_position3->left) &&
            (cs_position3->left < cs_position->right) &&
```

```
            (cs_position->right<cs_position2->right)  &&
            (cs_position2->right < cs_position3->right)) {
              ccs_position->cstem[0].left = cs_position->left;
              ccs_position->cstem[0].right = cs_position->right;
              ccs_position->cstem[1].left = cs_position2->left;
              ccs_position->cstem[1].right = cs_position2->right;
              ccs_position->cstem[2].left = cs_position3->left;
              ccs_position->cstem[2].right = cs_position3->right;
              ccs_position++;
              n_cstems++;
            }
        }
    }
}
```

Οι 3 μεταβλητές cs_position,cs_position1 και cs_position2 χρησιμοποιούνται προκειμένου να έχουμε πρόσβαση στη δομή όπου είναι αποθηκευμένα τα ζευγάρια από την προηγούμενη φάση του αλγορίθμου. Η τριπλή πρόσβαση στην ίδια δομή γίνεται διότι, λόγω του ελέγχου τριάδων από ζευγάρια η διαδικασία αυτή πραγματοποιείται με μία τριπλή επανάληψη for. Έπειτα, μέσω της εντολής if ελέγχουμε αν οι θέσεις των βάσεων στα 3 ζευγάρια είναι τέτοιες ώστε να έχουμε πράγματι ψευδοκόμβο τύπου L. Οι συνθήκες ελέγχου της παραπάνω συνάρτησης για τον ψευδοκόμβο τύπου L είναι οι εξής:

1)Η θέση της πρώτης βάσης του πρώτου ζευγαριού να είναι μικρότερη από τη θέση της πρώτης βάσης του δεύτερου ζευγαριού

2)Η θέση της πρώτης βάσης του δεύτερου ζευγαριού να είναι μικρότερη από τη θέση της πρώτης βάσης του τρίτου ζευγαριού

3)Η θέση της πρώτης βάσης του τρίτου ζευγαριού να είναι μικρότερη από τη θέση της δεύτερης βάσης του πρώτου ζευγαριού

4)Η θέση της δεύτερης βάσης του πρώτου ζευγαριού να είναι μικρότερη από τη θέση της δεύτερης βάσης του δεύτερου ζευγαριού και

5)Η θέση της δεύτερης βάσης του δεύτερου ζευγαριού να είναι μικρότερη από τη θέση της δεύτερης βάσης του τρίτου ζευγαριού

Εφόσον όλες αυτές οι συνθήκες είναι αληθείς, τότε καταχωρούμε την τριάδα αυτή των ζευγαριών στη δομή τύπου core_stem, που, όπως αναφέραμε παραπάνω, έχουμε αρχικοποιήσει προκειμένου να αποθηκεύουμε όλες τις τριάδες ζευγαριών που επιβεβαιώσαμε ότι μπορούν να σχηματίσουν τη δομή του ψευδοκόμβου τύπου L, και προχωράμε στην επόμενη θέση του πίνακα κάθε φορά προκειμένου να αποθηκεύσουμε την επόμενη κατάλληλη τριάδα ζευγαριών. Ακόμη, αυξάνουμε κάθε φορά κατά ένα μία ακέραια μεταβλητή, την nc_stems, στην οποία αποθηκεύουμε τον συνολικό αριθμό αυτών των τριάδων που βρίσκουμε.

Αφού εντοπίστηκαν και όλες οι δυνατές τριάδες βάσεων οι οποίες μπορούν να σχηματίσουν τη δομή του ψευδοκόμβου τύπου L,στη συνέχεια υπολογίστηκαν τα βασικά

μεγέθη που χαρακτήριζαν τον κάθε ψευδοκόμβο ως συνάρτηση των θέσεων των βάσεων οι οποίες είχαν υπολογιστεί προηγουμένως. Έπειτα, για κάθε ψευδοκόμβο πραγματοποιήθηκαν ορισμένοι έλεγχοι οι οποίοι αφορούσαν κάποια από τα βασικά μεγέθη του και, άμα αυτές οι συνθήκες ελέγχου ικανοποιούνταν, τότε οι ψευδοκόμβοι αυτοί γίνονταν αποδεκτοί και επιστρέφονταν στην προαναφερθείσα συνάρτηση της python από την οποία έγινε η κλήση. Ο κώδικας για το συγκεκριμένο στάδιο του αλγορίθμου παρατίθεται παρακάτω.

16.Κώδικας υπολογισμού των κύριων μεταβλητών του ψευδοκόμβου τύπου L μέσω των μεταβλητών που εκφράζουν τις θέσεις των κύριων βάσεων στους σχηματιζόμενους δεσμούς

```
for (int i = 0; i < n_cstems; i++) {
    int left = (ccs_position->cstem[0]).left;
    int size = (ccs_position->cstem[2]).right -
    ccs_position->cstem[0].left + 1;
    int left_left_loop_size =
        ccs_position->cstem[1].left - ccs_position->cstem[0].left - 1;
    int left_right_loop_size =ccs_position->cstem[2].left
    -ccs_position->cstem[1].left-1;
    int right_left_loop_size = ccs_position->cstem[1].right
    -ccs_position->cstem[0].right-1;
    int dd_size =
        ccs_position->cstem[0].right - ccs_position->cstem[2].left - 1;
    if (size < min_window_size || size > max_window_size ||
        dd_size < s_min_dd_size || dd_size > s_max_dd_size ||
        left_left_loop_size == 0 || left_right_loop_size == 0 ||
right_left_loop_size  == 0  ||
        (ccs_position->cstem[1].right == ccs_position->cstem[2].right - 1)) {
      ccs_position++;
      continue;
    }
    cb(left, size, left_left_loop_size, left_right_loop_size,
    right_left_loop_size,dd_size);
    ccs_position++;
}
```

Οι αναθέσεις στις κύριες μεταβλητές του ψευδοκόμβου τύπου L με βάση τις θέσεις των κύριων βάσεων έγιναν ως εξής:

1)Η μεταβλητή left η οποία εκφράζει τη θέση αρχής του ψευδοκόμβου έλαβε ως τιμή τη θέση της αριστερής βάσης του πρώτου ζευγαριού

2)Η μεταβλητή size, η οποία αναπαριστά το συνολικό μέγεθος του ψευδοκόμβου, έλαβε ως τιμή τη διαφορά των θέσεων της δεξιάς βάσης του τρίτου ζευγαριού και της αριστερής βάσης του πρώτου ζευγαριού αυξημένη κατά 1

3)Η μεταβλητή left_left_loop_size έλαβε ως τιμή τη διαφορά των θέσεων της αριστερής βάσης του δεύτερου ζευγαριού και της αριστερής βάσης του πρώτου ζευγαριού μειωμένη κατά 1

4)Η μεταβλητή left_right_loop_size έλαβε ως τιμή τη διαφορά των θέσεων της αριστερής βάσης του τρίτου ζευγαριού και της αριστερής βάσης του δεύτερου ζευγαριού μειωμένη κατά 1

5)Η μεταβλητή dd_size έλαβε ως τιμή τη διαφορά των θέσεων της δεξιάς βάσης του πρώτου ζευγαριού και της αριστερής βάσης του τρίτου ζευγαριού, που είναι οι δύο ενδιάμεσες βάσεις του ψευδοκόμβου τύπου L, μειωμένη κατά 1

6)Τέλος, η μεταβλητή right_left_loop_size έλαβε ως τιμή τη διαφορά των θέσεων μεταξύ της δεξιάς βάσης του δεύτερου ζευγαριού και της δεξιάς βάσης του πρώτου ζευγαριού επίσης μειωμένη κατά 1

Οι συνθήκες ελέγχου τις οποίες έπρεπε να ικανοποιεί ο κάθε ψευδοκόμβος προκειμένου να γίνει αποδεκτός και να επιστραφεί στο αρχικό πρόγραμμα ήταν οι εξής:

1)Όλες οι κύριες μεταβλητές του ψευδοκόμβου οι οποίες υπολογίστηκαν παραπάνω να έχουν τιμή μεγαλύτερη του μηδενός

2)Οι δύο τελευταίες βάσεις του κάθε ψευδοκόμβου να έχουν διαφορά θέσης μεγαλύτερη από ένα, δηλαδή να μην είναι η μία βάση ακριβώς δίπλα στην άλλη

3)Η μεταβλητή dd_size η οποία εκφράζει το κενό ανάμεσα στην τρίτη και στην τέταρτη κύρια βάση να βρίσκεται μεταξύ των ορίων που καθορίζονται από τις μεταβλητές min_dd_size και max_dd_size.

4)Η μεταβλητή size, η οποία εκφράζει το συνολικό μήκος του κάθε ψευδοκόμβου, να βρίσκεται μεταξύ των ορίων που καθορίζονται από τις μεταβλητές min_window_size και max_window_size. Αυτές οι δύο μεταβλητές μπορούν επίσης να οριστούν από το χρήστη κατά την εκτέλεση του προγράμματος ενώ οι default τιμές τους, αν δεν οριστούν εξ' αρχής, είναι 10 και 204 αντίστοιχα. Τέλος, όπως φαίνεται παραπάνω, η επιστροφή των κύριων μεγεθών του ψευδοκόμβου στη συνάρτηση της python από την οποία κλήθηκε γίνεται πάλι μέσω της συνάρτησης cb.

4.4 Εύρεση των ζευγαριών μεταξύ των βάσεων γύρω από τη δομή του ψευδοκόμβου

Η συνάρτηση μέσω της οποίας εντοπίστηκαν τα ζευγάρια βάσεων σε περιοχές γύρω από τα κύρια ζευγάρια του ψευδοκόμβου φαίνεται παρακάτω

17.Συνάρτηση υπολογισμού όλων των ζευγαριών που σχηματίζονται μεταξύ των βάσεων γύρω από τους κύριους δεσμούς του ψευδοκόμβου

```
void pairalign(char *sequence, int i, int j, int left_left_loop_size,
int left_right_loop_size, int right_left_loop_size,int dd_size,
              void (*cb)(char *, int, int, int)) {

  int L = i;
  int M = i + left_left_loop_size + 1;
  int R = i +left_left_loop_size +left_right_loop_size +2;
  int l = i + left_left_loop_size + left_right_loop_size + dd_size + 3;
  int m= l +right_left_loop_size+1;
  int r = i + j - 1;

  char *dot_bracket = strdup(sequence);
  int left_loop_stems = 0, middle_loop_stems=0,right_loop_stems = 0;


  int len = strlen(sequence);
  memset(dot_bracket, '.', len);
  dot_bracket[L] = '(';
  dot_bracket[l] = ')';
  dot_bracket[M] = '[';
  dot_bracket[m] = ']';
  dot_bracket[R] = '{';
  dot_bracket[r] = '}';


  left_loop_stems = 0;
  for (int a = L - 1, b = l + 1; a >= 0 && b <= m - 1; a--, b++) {
    if (!IS_PAIR(sequence[a], sequence[b])) {
      break;
    }
    dot_bracket[a] = '(';
    dot_bracket[b] = ')';
    left_loop_stems++;
  }

  middle_loop_stems=0;
  for (int a = M - 1, b = m + 1; a >= L + 1 && b <= r - 1; a--, b++) {
    if (!IS_PAIR(sequence[a], sequence[b])) {
      break;
    }
```

```
    dot_bracket[a] = '[';
    dot_bracket[b] = ']';
    left_loop_stems++;
  }


  right_loop_stems = 0;
  for (int a = R - 1, b = r + 1; a >= M + 1 && b <= len - 1; a--, b++) {
    if (!IS_PAIR(sequence[a], sequence[b])) {
      break;
    }
    dot_bracket[a] = '{';
    dot_bracket[b] = '}';
    right_loop_stems++;
  }
 cb(dot_bracket,left_loop_stems,middle_loop_stems,right_loop_stems);
  free(dot_bracket);
}
```

Αρχικά, μέσω των κύριων μεταβλητών που υπολογίσαμε προηγουμένως και τις οποίες περνάμε ως ορίσματα στη συνάρτηση, υπολογίζουμε ακριβώς τις θέσεις των κύριων βάσεων στην ακολουθία του RNA. Οι μεταβλητές L,M και R αναπαριστούν τις θέσεις της πρώτης, της δεύτερης και της τρίτης κύριας βάσης αντίστοιχα ενώ οι μεταβλητές l,m και r τις θέσεις της τέταρτης, της πέμπτης και της έκτης κύριας βάσης αντίστοιχα. Έπειτα μέσω της εντολής memset απεικονίζουμε την ακολουθία των χαρακτήρων ως ένα σύνολο από τελίτσες και στη συνέχεια τοποθετούμε διάφορες αγκύλες στις θέσεις των κύριων βάσεων προκειμένου να αναπαραστήσουμε την ακολουθία του RNA σε μορφή dot-bracket. Η αντικατάσταση των βάσεων με αγκύλες γίνεται ως εξής:

1)Στη θέση της πρώτης βάσης τοποθετείται το σύμβολο '('

2)Στη θέση της δεύτερης κύριας βάσης τοποθετείται το σύμβολο «[«

3)Στη θέση της τρίτης κύριας βάσης τοποθετείται το σύμβολο «{«

4)Η τέταρτη κύρια βάση αντικαθίσταται με το σύμβολο «)» το οποίο είναι το αντίθετο του συμβόλου της πρώτης κύριας βάσης προκειμένου να υποδείξουμε ότι αυτές οι δύο βάσεις σχηματίζουν δεσμό μεταξύ τους.

5) Η πέμπτη κύρια βάση αντικαθίσταται με το σύμβολο «]» το οποίο είναι το αντίθετο του συμβόλου της δεύτερης κύριας βάσης προκειμένου να υποδείξουμε ότι αυτές οι δύο βάσεις σχηματίζουν δεσμό μεταξύ τους.

6) Η έκτη κύρια βάση αντικαθίσταται με το σύμβολο «}» το οποίο είναι το αντίθετο του συμβόλου της τρίτης κύριας βάσης προκειμένου να υποδείξουμε ότι αυτές οι δύο βάσεις σχηματίζουν δεσμό μεταξύ τους.

Στη συνέχεια, υπολογίζουμε τα ζευγάρια μεταξύ των βάσεων που βρίσκονται γύρω από τα κύρια ζευγάρια του ψευδοκόμβου τύπου L. Για να κρατάμε τον αριθμό των ζευγαριών γύρω από κάθε κύριο ζευγάρι, αρχικοποιήσαμε 3 ακέραιες μεταβλητές με την

τιμή μηδέν. Η μεταβλητή left_loop_stems αφορούσε τα ζευγάρια γύρω από το πρώτο κύριο ζευγάρι, η μεταβλητή middle_loop_stems αφορούσε τα ζευγάρια γύρω από το δεύτερο κύριο ζευγάρι και η μεταβλητή right_loop_stems αφορούσε τα ζευγάρια γύρω από το τρίτο κύριο ζευγάρι. Τα πρώτα ζευγάρια σχηματίζονταν μεταξύ βάσεων που βρίσκονταν αριστερά από την πρώτη κύρια βάση και βάσεων που βρίσκονταν ανάμεσα στην τέταρτη και την πέμπτη κύρια βάση. Οπότε στην πρώτη επανάληψη for ελέγχουμε αν η τελευταία βάση της πρώτης περιοχής και η πρώτη βάση της δεύτερης περιοχής μπορούν να σχηματίσουν ζευγάρι μεταξύ τους. Αν όντως σχηματίζουν ζευγάρι, τότε πάμε στην επόμενη θέση και στις δύο περιοχές (προτελευταία και δεύτερη βάση αντίστοιχα) για να ελέγξουμε το επόμενο ζευγάρι και αυξάνουμε την ακέραια μεταβλητή, που κρατά το συνολικό αριθμό ζευγαριών, κατά 1. Αν όχι, τότε σταματάμε τη διαδικασία μέσω της εντολής break αφού δε μπορούμε να βρούμε άλλα ζευγάρια συνεχόμενα σε εκείνη την περιοχή. Όσα ζευγάρια βρίσκουμε τα αναπαριστούμε πάλι με αγκύλες και συγκεκριμένα με τα σύμβολα ’(‘ για τις βάσεις της πρώτης περιοχής και ‘)’ για τις βάσεις της δεύτερης περιοχής τα οποία είναι τα σύμβολα με τα οποία αναπαραστήσαμε και τις βάσεις του πρώτου κύριου ζευγαριού. Για να εντοπίσουμε τα ζευγάρια γύρω από το δεύτερο κύριο ζευγάρι πραγματοποιούμε την ίδια διαδικασία με πριν. Οι δύο περιοχές στις οποίες γίνεται η αναζήτηση αυτή τη φορά είναι αυτή ανάμεσα στην πρώτη και στη δεύτερη κύρια βάση και αυτή ανάμεσα στην πέμπτη και την έκτη κύρια βάση. Τα ζευγάρια που εντοπίζουμε σε αυτήν την περιοχή τα αναπαριστούμε με τα σύμβολα ‘[‘ και ‘]’ για την πρώτη και τη δεύτερη περιοχή αντίστοιχα τα οποία είναι ίδια με τα σύμβολα που αναπαραστήσαμε τις βάσεις του δεύτερου κύριου ζευγαριού. Τέλος, οι δύο περιοχές στις οποίες έγινε η αναζήτηση για το τρίτο σύνολο ζευγαριών είναι αυτή ανάμεσα στη δεύτερη και την τρίτη κύρια βάση και αυτή δεξιά από την έκτη κύρια βάση. Οι βάσεις των ζευγαριών απεικονίζονταν με τα σύμβολα ‘{‘ και ‘}’ που ήταν ίδια με τα σύμβολα αναπαράστασης των βάσεων του τρίτου κύριου ζευγαριού. Τέλος, αφού ολοκληρώσουμε την αναπαράσταση με dot-bracket μέσω της παραπάνω διαδικασίας την επιστρέφουμε στην αρχική συνάρτηση της python μαζί με τις τρεις ακέραιες μεταβλητές στις οποίες αποθηκευόταν ο αριθμός των ζευγαριών που δημιουργούνταν σε κάθε μία από τις τρεις περιοχές.

Ο έλεγχος αν δύο βάσεις αποτελούσαν ζευγάρι στη συγκεκριμένη συνάρτηση έγινε μέσω των βοηθητικών τμημάτων κώδικα που φαίνονται παρακάτω

18.Ορισμοί μέσω των οποίων ελέγχουμε αν οι βάσεις σχηματίζουν ζευγάρι μεταξύ τους

```
#define IS_PAIR(a, b) (IS_PAIR_ONEWAY(a, b) || IS_PAIR_ONEWAY(b, a))

#define IS_PAIR_ONEWAY(a, b)
  (((a) == 'a' && (b) == 'u') || ((a) == 'g' && ((b) == 'c' || (b) == 'u')))
```

Μέσω του κώδικα του ορισμού IS_PAIR_ONEWAY διαπιστώνουμε αν τα ζευγάρια που εντοπίζουμε στην ακολουθία του RNA είναι ένα από τα ζευγάρια αδενίνης-ουρακίλης, γουανίνης-κυτοσίνης ή γουανίνης-ουρακίλης. Αντίστοιχα, μέσω του ορισμού IS_PAIR ελέγχουμε αν το ζευγάρι των βάσεων ή το αντίστροφο του ζευγαριού ικανοποιεί τη

συνθήκη της πρώτης συνάρτησης.

4.5 Απαλοιφή ζευγαριών αδενίνης ουρακίλης στα άκρα της ακολουθίας του RNA

Το τελευταίο στάδιο της αναπαράστασης με dot-bracket της ακολουθίας του RNA αφορά την αφαίρεση των ζευγαριών αδενίνης-ουρακίλης τα οποία βρίσκονται στην άκρη της ακολουθίας του RNA. Η επιλογή αν θα πραγματοποιηθεί αυτή η διαδικασία καθορίζεται από μία δυαδική μεταβλητή η οποία μπορεί να καθορίσει ο χρήστης κατά την ώρα εκτέλεσης του προγράμματος ενώ ο λόγος υλοποίησης αυτής της συνάρτησης έγκειται σε φυσικά κριτήρια καθώς έχει διαπιστωθεί ότι η πραγματική δομή σε πολλές περιπτώσεις προκύπτει από τέτοια αφαίρεση δεσμών. Η υλοποίηση της συνάρτησης αυτής φαίνεται παρακάτω.

19.Συνάρτηση διαγραφής των δεσμών αδενίνης-ουρακίλης που βρίσκονται στα άκρα της ακολουθίας του RNA

```
void skip_final_au(char *sequence, char *dot_bracket, int left_loop_stems,
                   int right_loop_stems, void (*cb)(char *, int, int)) {
  int L = -1, l = -1, R = -1, r = -1;

  char *bracket = strdup(dot_bracket);

  for (int i = 0; i < strlen(sequence); i++) {
    if (bracket[i] == '(' && L == -1) {
      L = i;
    } else if (bracket[i] == '{' && R == -1) {
      R = i;
    }
else if (bracket[i] == ')') {
      l = i;
    }
else if (bracket[i] == '}') {
      r = i;
    }
  }

  int left_is_au = left_loop_stems > 0 && IS_AU(sequence[L], sequence[l]);
  int right_is_au = right_loop_stems > 0 && IS_AU(sequence[R], sequence[r]);

  if (left_is_au) {
    bracket[L] = bracket[l] = '.';
    cb(bracket, left_loop_stems - 1, right_loop_stems);
    bracket[L] = '(';
```

```
    bracket[l] = ')';
  }
  if (right_is_au) {
    bracket[R] = bracket[r] = '.';
    cb(bracket, left_loop_stems, right_loop_stems - 1);
    bracket[R] = '{';
    bracket[r] = '}';
  }
  if (left_is_au && right_is_au) {
    bracket[L] = bracket[l] = bracket[R] = bracket[r] = '.';
    cb(bracket, left_loop_stems - 1, right_loop_stems - 1);
  }

  free(bracket);
}
```

Αρχικά, βρίσκουμε τις θέσεις των δύο βάσεων που βρίσκονται στην άκρη της δομής του ψευδοκόμβου καθώς και τις δύο βάσεις με τις οποίες δημιουργούν ζευγάρια. Στη μεταβλητή L καταχωρούμε τη θέση της βάσης που βρίσκεται πιο αριστερά στη δομή του ψευδοκόμβου λαμβάνοντας υπ' όψιν την αναπαράσταση dot_bracket. Στη μεταβλητή l αποθηκεύουμε τη θέση της βάσης η οποία δημιουργεί ζευγάρι με την προηγούμενη. Στη μεταβλητή r αποθηκεύουμε τη θέση της βάσης που βρίσκεται πιο δεξιά στη δομή του ψευδοκόμβου με βάση την αναπαράσταση dot_bracket ενώ στη μεταβλητή R αποθηκεύουμε τη θέση της βάσης η οποία είναι ζευγάρι της προηγούμενης. Στη συνέχεια, μέσω της δυαδικής μεταβλητής left_is_au ελέγχουμε αν το σύνολο των ζευγαριών στο οποίο εντοπίσαμε το αριστερό ζευγάρι είναι μεγαλύτερο του μηδενός, δηλαδή αν όντως υπάρχουν αυτά τα ζευγάρια βάσεων γύρω από το πρώτο κύριο ζευγάρι του ψευδοκόμβου, και αν όντως το ζευγάρι που εντοπίσαμε είναι μεταξύ αδενίνης και ουρακίλης. Παρόμοια, μέσω της δυαδικής μεταβλητής right_is_au κάνουμε τους ίδιους ελέγχους για το ζευγάρι που περιέχει την πιο δεξιά βάση στην dot_bracket αναπαράσταση του RNA. Τέλος, αν κάποια από τις δύο προαναφερθείσες δυαδικές μεταβλητές έχει την τιμή True, τότε αντικαθιστούμε τις αγκύλες στις συγκεκριμένες θέσεις πάλι με τελίτσες, μειώνουμε την αντίστοιχη μεταβλητή που εκφράζει το πλήθος των ζευγαριών και επιστρέφουμε τη dot-bracket αναπαράσταση μαζί με τις δύο ακέραιες μεταβλητές για τα δύο σύνολα ζευγαριών στα οποία έγινε η παραπάνω ανάλυση στο πρόγραμμα της python. Αν και οι δύο μεταβλητές έχουν την τιμή True, τότε αντικαθιστούμε τις αγκύλες και των τεσσάρων βάσεων με τελίτσες, μειώνουμε και τις δύο μεταβλητές για τον αριθμό ζευγαριών κατά 1 και επιστρέφουμε τα ίδια στοιχεία στο αρχικό πρόγραμμα. Η μεταβλητή middle_loop_stems η οποία εξέφραζε τον αριθμό των ζευγαριών γύρω από το μεσαίο κύριο ζευγάρι δεν επηρεάζεται από αυτή τη συνάρτηση οπότε ούτε εισάγεται ως όρισμα στη συνάρτηση ούτε επιστρέφεται από αυτήν.

Όλοι αυτοί οι ψευδοκόμβοι μαζί με τη dot_bracket αναπαράσταση τους η οποία υπολογίστηκε και τροποποιήθηκε από αυτήν και την προηγούμενη συνάρτηση προ-

στίθενται στην τελική λίστα με όλες τις πιθανές περιπτώσεις ύπαρξης της δομής του ψευδοκόμβου τύπου L στην ακολουθία του RNA. Οι πληροφορίες που υπάρχουν για κάθε δομή στην τελική λίστα αναπαρίστανται ως ένα dictionary στο οποίο αποθηκεύεται η dot_bracket αναπαράσταση, το πλήθος των ζευγαριών στις 3 περιοχές γύρω από τούς βασικούς δεσμούς του ψευδοκόμβου μέσω των μεταβλητών left_loop_stems, middle_loop_stems και right_loop_stems καθώς και το μήκος της ακολουθίας ανάμεσα στις δύο ενδιάμεσες βάσεις το οποίο, όπως αναφέραμε και προηγουμένως, αποθηκεύεται στη μεταβλητή dd_size. Αν η τελική λίστα των ψευδοκόμβων είναι κενή, οπότε δεν έχει εντοπιστεί καμία ύπαρξη της δομής μέσα στην ακολουθία του RNA,τότε η τελική λίστα περιέχει μόνο ένα dictionary όπου η dot_bracket αναπαράσταση του είναι μία συμοβολοσειρά από τελίτσες και οι υπόλοιπες μεταβλητές έχουν την τιμή 0.

4.6 Επιλογή της βέλτιστης δομής

Το τελευταίο βήμα της όλης διαδικασίας αφορά την επιλογή από την τελική λίστα του βέλτιστου ψευδοκόμβου με κριτήρια το μικρότερο ποσό ελεύθερης ενέργειας αναδίπλωσης και το μεγαλύτερο αριθμό ζευγαριών βάσεων. Αρχικά, για καθε ψευδοκόμβο αθροίζουμε τα ζευγάρια βάσεων που δημιουργούνταν γύρω από τα κύρια ζευγάρια. Έπειτα, από όλους τους ψευδοκόμβους κρατούσαμε αυτούς για τους οποίους το παραπάνω άθροισμα βρισκόταν πάνω από ένα συγκεκριμένο όριο. Το όριο αυτό ήταν ίσο με διαφορά της μέγιστης τιμής αυτού του αθροίσματος (left-loop-stems+middle-loop-stems+right-loop-stems), που εντοπιζόταν σε όλους τους ψευδοκόμβους της λίστας, με μία μεταβλητή, τη max-stem-allow-smaller, της οποίας η τιμή μπορούσε να καθοριστεί από το χρήστη κατά τη διάρκεια εκτέλεσης του προγράμματος ενώ η default τιμή της ήταν 1. Έπειτα, για όλους αυτούς τους ψευδοκόμβους, οι οποίοι παρέμεναν στη λίστα έπειτα και από την εφαρμογή του προηγούμενου κριτηρίου, υπολογιζόταν η συνολική ενέργεια που εκλυόταν από αυτούς. Για τον υπολογισμό αυτό, υπάρχουν δύο διαφορετικοί τρόποι. Ο πρώτος εξ' αυτών είναι μέσω ενός πακέτου,του pkenergy, το οποίο είναι υλοποιημένο σε C++ και στηρίζεται στη μεθοδολογία Hotknots[51]. Στη συγκεκριμένη μεθοδολογία, η ενέργεια της δομής υπολογίζεται ως συνάρτηση τόσο του αριθμού των δεσμών μεταξύ των κύριων βάσεων όσο και του αριθμού των βάσεων οι οποίες παραμένουν αζευγάρωτες στη δομή. Ο δεύτερος τρόπος είναι με χρήση του πακέτου Vienna RNA και συγκεκριμένα μίας συνάρτησης της βιβλιοθήκης RNA της python, η οποία λαμβάνει ως εισόδους την ακολουθία του RNA μαζί με τη dot_bracket αναπαράσταση της και υπολογίζει την ενέργεια της δομής. Αφού υπολογιζόταν και η ενέργεια για κάθε μία από τις δομές, αυτές ταξινομούνταν έπειτα με βάση την ενέργεια (σε αύξουσα σειρά), το συνολικό αριθμό δεσμών (σε φθίνουσα σειρά) και την τιμή του μήκους ανάμεσα στις δύο ενδιάμεσες κύριες βάσεις (σε αύξουσα σειρά). Τέλος, από αυτήν την ταξινόμηση επιλέγουμε την πρώτη δομή την οποία και εμφανίζουμε στην έξοδο του προγράμματος.Ο κώδικας για τη διαδικασία ταξινόμησης όλων των ψευδοκόμβων, βάσει των προαναφερθέντων κριτηρίων, παρατίθεται παρακάτω.

20. Διαδικασία ταξινόμησης των ψευδοκόμβων με βάση την ενέργεια και τον αριθμό

δεσμών

```
def apply_free_energy_and_stems_criterion(
    data: pd.DataFrame,
    sequence: str,
    max_stem_allow_smaller: int,
    energy: BaseEnergy,
):
    data["stems"] = data["left_loop_stems"] + data["right_loop_stems"]
    + data["middle_loop_stems"]
    data = data[
        data["stems"] >= data["stems"].max()-max_stem_allow_smaller
    ].reset_index()

    data["energy"] = data["dot_bracket"].apply(lambda r:
    energy.eval(sequence, r))
    data.sort_values(
        ["energy", "stems", "dd"],
        ascending=(True, False, True),
        inplace=True
    )
    return data
```

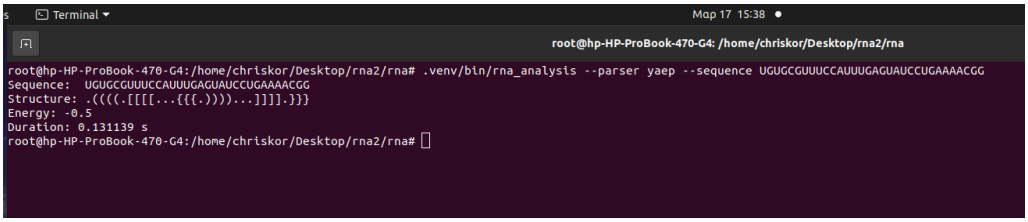
Στην έξοδο απ΄την κάθε εκτέλεση του προγράμματος εμφανίζονται αρχικά η συμβολοσειρά που αναπαριστά την ακολουθία του RNA. Ακόμη εμφανίζεται η dot_bracket αναπαράσταση, η ελεύθερη ενέργεια αναδίπλωσης της δομής καθώς και ο χρόνος εκτέλεσης του προγράμματος.

Κεφάλαιο 5

Επίδειξη λειτουργίας του συστήματος

5.1 Επιλογή αλγορίθμου εντοπισμού ψευδοκόμβων

Στις πρώτες δύο εικόνες φαίνεται η εκτέλεση του προγράμματος για μία τυχαία ακολουθία RNA χρησιμοποιώντας τον αλγόριθμο συντακτικής αναγνώρισης προτύπων στην πρώτη και τον άπληστο αλγόριθμο δυναμικού προγραμματισμού στη δεύτερη.



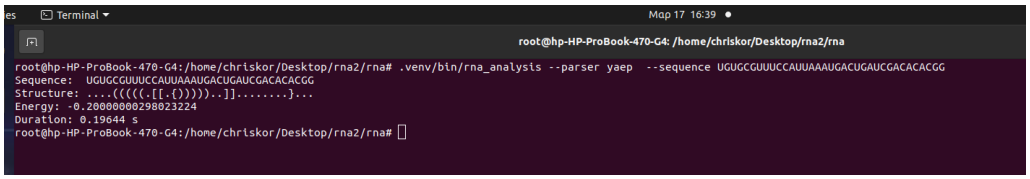
Σχήμα 5.1: *Εκτέλεση του προγράμματος με χρήση του αλγορίθμου συντακτικής αναγνώρισης προτύπων*

Σχήμα 5.2: *Εκτέλεση του προγράμματος με χρήση του άπληστου αλγορίθμου δυναμικού προγραμματισμού*

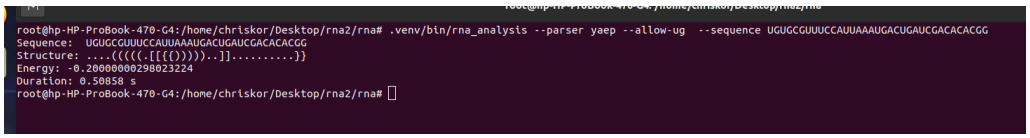
Παρατηρούμε ότι και οι δύο αλγόριθμοι δίνουν την ίδια αναπαράσταση dot_bracket, το οποίο είναι λογικό εφόσον δεν αλλάζει κάποιο κριτήριο στην τελική κατάταξη των ψευδοκόμβων, ενώ επίσης ότι ο άπληστος αλγόριθμος είναι πολύ πιο γρήγορος, όπως είναι αναμενόμενο, από τον αλγόριθμο συντακτικής αναγνώρισης προτύπων.

5.2 Έλεγχος για ψευδοκόμβους που περιέχουν το δεσμό γουανίνης ουρακίλης

Στις επόμενες δύο εικόνες φαίνεται η εκτέλεση του προγράμματος για μία άλλη ακολουθία του RNA, την πρώτη φορά χωρίς τη χρήση του ειδικού ορίσματος –allow-ug, το οποίο επιτρέπει τον εντοπισμό ψευδοκόμβων που περιέχουν το δεσμό γουανίνης ουρακίλης, και τη δεύτερη φορά με χρήση αυτού του ορίσματος ενώ και στις δύο περιπτώσεις χρησιμοποιείται ο αλγόριθμος συντακτικής αναγνώρισης προτύπων.



Σχήμα 5.3: *Εκτέλεση του προγράμματος με τον αλγόριθμο συντακτικής αναγνώρισης προτύπων*

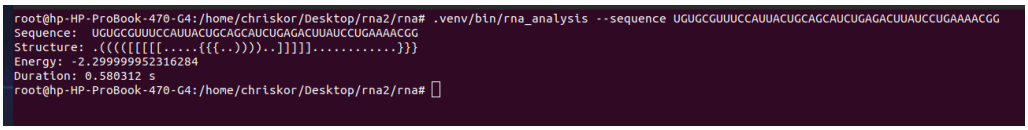


Σχήμα 5.4: *Εκτέλεση του προγράμματος με τον αλγόριθμο συντακτικής αναγνώρισης προτύπων με χρήση του ορίσματος –allow-ug*

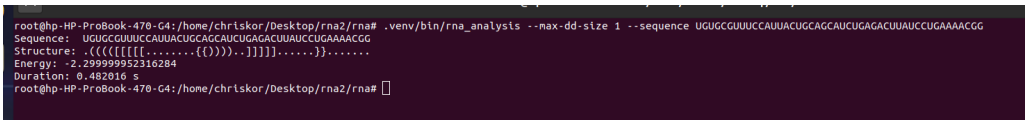
Στη dot_bracket αναπαράσταση της δεύτερης εικόνας παρατηρούμε ότι η τελευταία εμφάνιση του συμβόλου '{' αντιστοιχεί στο χαρακτήρα 'U' ενώ το πρώτο σύμβολο '}' που εμφανίζεται αντιστοιχεί στο χαρακτήρα 'G' της ακολουθίας του RNA, γεγονός που σημαίνει ότι εντοπίζεται ψευδοκόμβος που περιέχει το δεσμό γουανίνης-ουρακίλης.

5.3 Εισαγωγή άνω και κάτω ορίου στο μήκος της ακολουθίας ανάμεσα στις δύο ενδιάμεσες βάσεις

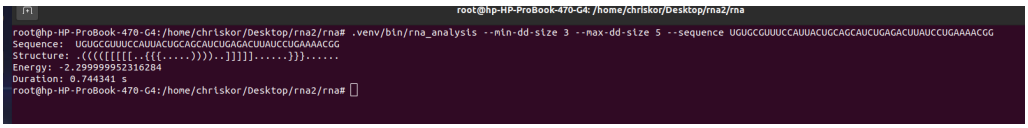
Στις επόμενες τριες εικόνες παρουσιάζεται η εκτέλεση του προγράμματος χρησιμοποιώντας και στις τρεις τον αλγόριθμο συντακτικής αναγνώρισης προτύπων, ενώ στη δεύτερη ορίζουμε ένα άνω όριο στο μήκος της ακολουθίας μεταξύ των ενδιάμεσων βασεων μέσω ορισμού τιμής στη μεταβλητή max_dd_size και στην τρίτη ορίζουμε ένα κάτω όριο στο συγκεκριμένο μήκος μέσω ορισμού τιμής στη μεταβλητή min_dd_size.



Σχήμα 5.5: *Εκτέλεση του προγράμματος με τον αλγόριθμο συντακτικής αναγνώρισης προτύπων*



Σχήμα 5.6: *Εκτέλεση του προγράμματος με τον αλγόριθμο συντακτικής αναγνώρισης προτύπων με ορισμό άνω ορίου στο μήκος της ακολουθίας μεταξύ των δύο ενδιάμεσων βάσεων*

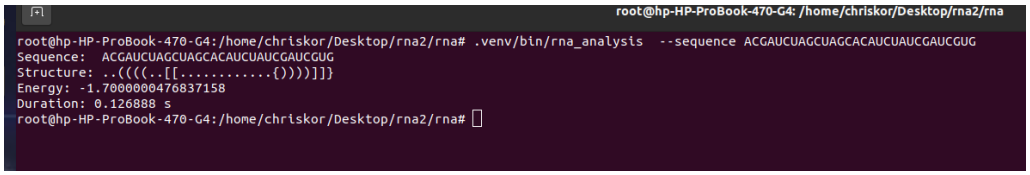


Σχήμα 5.7: *Εκτέλεση του προγράμματος με τον αλγόριθμο συντακτικής αναγνώρισης προτύπων με ορισμό κάτω ορίου στο μήκος της ακολουθίας μεταξύ των δύο ενδιάμεσων βάσεων*

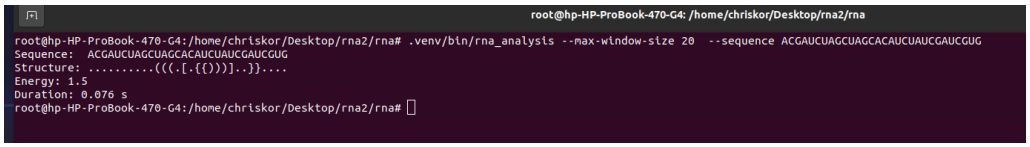
Στην πρώτη εκτέλεση του προγράμματος, όπου δεν έχει τεθεί κάποιο όριο στην τιμή της μεταβλητής dd_size, η τιμή της προκύπτει ίση με 2(απόσταση μεταξύ τελευταίας εμφάνισης του συμβόλου '{' και πρώτης εμφάνισης του συμβόλου ')'. Στη δεύτερη περίπτωση, οπου θέτουμε το άνω όριο για τη μεταβλητή ίσο με 1, η απόσταση αυτή είναι 0, ενώ στην τρίτη περίπτωση, όπου θέτουμε το κάτω όριο ίσο με 3, ενώ θέτουμε και το άνω όριο ίσο με 5 διότι με τη default τιμή του που είναι 2 το πρόγραμμα δε βγάζει έξοδο όπως είναι αναμενόμενο, η απόσταση αυτή προκύπτει ίση με 5.

5.4 Εισαγωγή άνω και κάτω ορίου για το συνολικό μήκος του ψευδοκόμβου

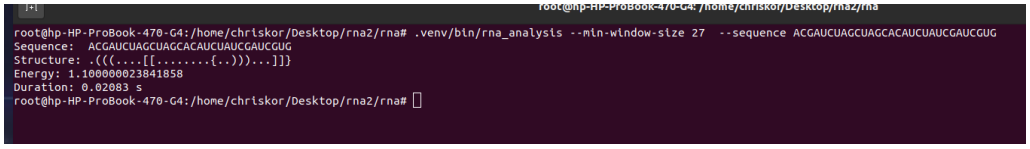
Στις επόμενες τρεις εικόνες παρουσιάζεται η εκτέλεση του προγράμματος χρησιμοποιώντας και στις τρεις τον αλγόριθμο συντακτικής αναγνώρισης προτύπων, ενώ στη δεύτερη ορίζουμε ένα άνω όριο στο μήκος του ψευδοκόμβου μέσω ορισμού τιμής στη μεταβλητή max_window_size και στην τρίτη ορίζουμε ένα κάτω όριο στο μήκος του ψευδοκόμβου μέσω ορισμού τιμής στη μεταβλητή min_window_size.



Σχήμα 5.8: *Εκτέλεση του προγράμματος με τον αλγόριθμο συντακτικής αναγνώρισης προτύπων*



Σχήμα 5.9: *Εκτέλεση του προγράμματος με τον αλγόριθμο συντακτικής αναγνώρισης προτύπων με ορισμό άνω ορίου στο συνολικό μήκος του ψευδοκόμβου*

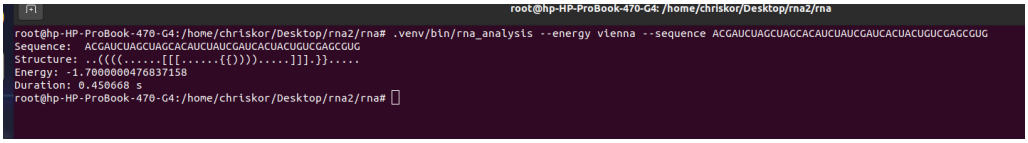


Σχήμα 5.10: *Εκτέλεση του προγράμματος με τον αλγόριθμο συντακτικής αναγνώρισης προτύπων με ορισμό κάτω ορίου στο συνολικό μήκος του ψευδοκόμβου*

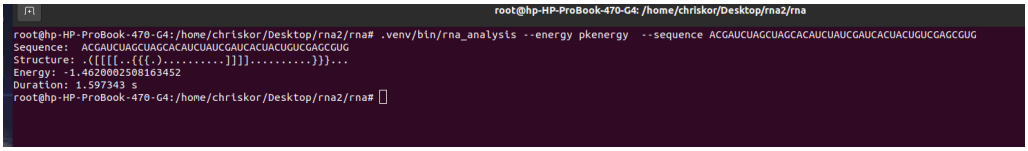
Στην πρώτη περίπτωση, όπου δε θέτουμε κάποιο όριο στο μήκος του ψευδοκόμβου, αυτό προκύπτει ίσο με 25(απόσταση μεταξύ τελευταίας εμφάνισης του συμβόλου '(' και πρώτης εμφάνισης του συμβόλου '}').Στη δεύτερη περίπτωση, όπου το άνω όριο στο μήκος είναι ίσο με 20, η συγκεκριμένη απόσταση είναι ίση με 13, ενώ στην τρίτη περίπτωση, όπου θέτουμε το κάτω όριο για το συγκεκριμένο μήκος ίσο με 27, η απόσταση προκύπτει και αυτή ίση με 27.

5.5 Τροποποίηση του πακέτου μέσω του οποίου θα υπολογιστεί η ενέργεια της κάθε δομής

Τέλος, στις επόμενες 2 εικόνες παρουσιάζεται η εκτέλεση του προγράμματος χρησιμοποιώντας τον αλγόριθμο συντακτικής αναγνώρισης προτύπων και στις δύο, ορίζοντας όμως ως τρόπο υπολογισμού της ενέργειας το πακέτο Vienna RNA στην πρώτη και το πακέτο pkenergy στη δεύτερη



Σχήμα 5.11: *Εκτέλεση του προγράμματος με τον αλγόριθμο συντακτικής αναγνώρισης προτύπων και με χρήση του πακέτου ViennaRNA*



Σχήμα 5.12: *Εκτέλεση του προγράμματος με τον αλγόριθμο συντακτικής αναγνώρισης προτύπων και με χρήση του πακέτου pkenergy*

Παρατηρούμε ότι η dot_bracket αναπαράσταση είναι διαφορετική στις δύο περιπτώσεις, γεγονός που οφείλεται στα διαφορετικά ενεργειακά κριτήρια κατάταξης των ψευδοκόμβων της τελικής λίστας.

Κεφάλαιο 6

Αξιλόγηση της επίδοσης του συστήματος

6.1 Σύγκριση εξόδων διαφόρων συστημάτων για μία συγκεκριμένη γνωστή ακολουθία RNA

Προκειμένου να αξιολογήσουμε την επίδοση του συστήματος μας, δώσαμε ως είσοδο σε αυτό μία ακολουθία RNA που αντιστοιχεί σε ένα γνωστό ψευδοκόμβο τύπου L[52]. Στην παρακάτω εικόνα φαίνεται η αληθινή dot_bracket αναπαράσταση της συγκεκριμένης ακολουθίας RNA όσο και οι dot_bracket αναπραστάσεις που προκύπτουν τόσο από το δικό μας σύστημα όσο και από άλλα δύο γνωστά συστήματα, τα IPknot και Knotty.

Platform	RNA/ Dot Bracket
	UGUGCGUUUUCCAUUUGAGUAUCCUGAAAACGGGCUUUUCAGCAUGGAACGUACAUAUUAAAUAGUAGGAGUGCAUAGUGGCCCGUAUAGCAGGCAUUAACAUUCCUGA
Ground truth	(((((((.(((((........[[[[....[[[[....{{{{.)))))))))))).....]]]].....]]]]...........}}}}
knotify	.......((((((.............[[[[[[...{{{{.))))))....................]]]]]].................}}}}
IPknot	.((((((.(((((.[[[[...((.......))((..]]]]))))))))))))).....(((((((...(((.(((.........))))))..))))))).
Knotty	(((((((.((((((((((...[[[[...[[[[[[[.)))))..)))))))))))).....(((((((..(((()]]]]]]]....]]]].)))))..))))))).

Σχήμα 6.1: *Πρόβλεψη dot_bracket αναπαράστασης για συγκεκριμένη ακολουθία εισόδου*

Παρατηρούμε ότι το δικό μας σύστημα προέβλεψε επιτυχημένα τους βασικούς δεσμούς για τον ψευδοκόμβο τύπου L, ενώ τα άλλα δύο συστήματα προέβλεψαν ψευδοκόμβους τύπου H οι οποίοι επίσης περιείχαν και hairpins.

Πέρα από την έξοδο των συστημάτων, χρησιμοποιήθηκαν και διάφορες μετρικές προκειμένου να γίνει η σύγκριση τους συστήματος μας με τα άλλα δύο προαναφερθέντα συστήματα. Οι μεταβλητές στις οποίες στηρίχτηκαν οι μετρικές που χρησιμοποιήθηκαν ήταν οι εξής:

1)True Positives(TP):Τα ζευγάρια βάσεων που προβλέψαμε ότι υπήρχαν στη δομή του RNA και πραγματικά υπήρχαν.

2)False Positives(FP):Τα ζευγάρια βάσεων που προβλέψαμε ότι υπήρχαν στη δομή του RNA ενώ στην πραγματικότητα δεν υπήρχαν.

3)True Negatives(TN):Τα ζευγάρια βάσεων που προβλέψαμε ότι δεν υπήρχαν στη δομή του RNA και δεν υπήρχαν και στην πραγματικότητα.

4)False Negatives(FN):Τα ζευγάρια βάσεων που προβλέψαμε ότι δεν υπήρχαν στη δομή του RNA ενώ στην πραγματικότητα υπήρχαν.

Οι 4 μετρικές λοιπόν με βάση τις οποίες έγινε η σύγκριση των συστημάτων είναι οι εξής:

1)Positive Predictive Value(PPV):Ισούται με το πηλίκο TP/(TP+FP)

2)Recall:Ισούται με το πηλίκο TP/(TP+FN)

3)F1-score:Ισούται με (2*PPV*Recall)/(PPV+Recall) (αρμονικός μέσος των PPV και Recall)

4)Matthews Correlation Coefficient(MCC):Ισούται με το κλάσμα που έχει ως αριθμητή τη διαφορά του γινομένου του TP με το TN με το γινόμενο του FP με το FN και ως παρονομαστή την ποσότητα:

X= $\sqrt{(TP+FP)(TP+FN)(TN+FP)(TN+FN)}$

Όσον αφορά την πρώτη μετρική, το σύστημα μας πέρασε κατά πολύ τις άλλες δύο μεθόδους έχοντας precision ίσο με 0.821, ενώ το Knotty είχε 0.5 και η IPknot 0.33. Όσον αφορά το F1-score, το σύστημα μας είχε σκορ 0.639 το οποίο ήταν λίγο χαμηλότερο από το 0.661 που σημείωσε το Knotty. Ανώτερο απ' όλα ήταν το σύστημα μας και στη μετρική MCC με σκορ ίσο με 0.499. Τέλος η πλατφόρμα Knotty σημείωσε πολύ υψηλό σκορ όσον αφορά το recall, παρουσιάζοντας μόνο ένα false negative, το οποίο ίσως οφείλεται στο γεγονός ότι οι μεγάλες ακολουθίες RNA συχνά περιέχουν πολλαπλές δομές οι οποίες δε σχετίζονται άμεσα με τον ψευδοκόμβο. Στην παρακάτω εικόνα φαίνονται αναλυτικά οι επιδόσεις που σημείωσε η κάθε μία πλατφόρμα.

Platform	tp	tn	fp	fn	Precision	Recall	F1-score	MCC
knotify	23	59	5	21	0.821	0.523	0.639	0.499
IPknot	22	32	44	10	0.333	0.688	0.449	0.102
Knotty	40	27	40	1	0.500	0.976	0.661	0.419

Σχήμα 6.2: *Οι επιδόσεις σε θεμελιώδεις μετρικές ανά πλατφόρμα*

Κεφάλαιο 7

Συμπεράσματα και Μελλοντικές επεκτάσεις

Η ανάγκη μελέτης και ανάλυσης του RNA είναι μεγάλη όπως αποδεικνύεται τόσο λόγω της πρόσφατης παρουσίας του Covid-19 όσο και λόγω της ανάγκης κατανόησης διαφόρων γεγονότων στη φύση. Γι' αυτό το σκοπό, η συγκεκριμένη έρευνα εισήγαγε μία ακριβή και αποδοτική μεθοδολογία πρόβλεψης μίας σπάνιας μορφής ψευδοκόμβου, του ψευδοκόμβου τύπου L. Η συγκεκριμένη μεθοδολογία στηρίχτηκε στον αλγόριθμο του Earley και παρήγαγε όλα τα πιθανά συντακτικά δέντρα, καθένα από τα οποία αναπαριστούσε μία πιθανή ύπαρξη της δομής στην ακολουθία του RNA. Ακομη, αναπτύχθηκε ένας άπληστος αλγόριθμος για την ίδια διαδιασία ο οποίος συνέβαλε σημαντικά προκειμένου να ενισχυθεί η απόδοση του συστήματος πρόβλεψης. Έπειτα, για την επιλογή της βέλτιστης δομής στηριχτήκαμε τόσο στη μεγιστοποίηση του αριθμού ζευγαριών βάσεων όσο και στην ελαχιστοποίηση της ελεύθερης ενέργειας αναδίπλωσης της δομής. Όσον αφορά τις μελλοντικές επεκτάσεις, θα μπορούσαμε να πραγματοποιήσουμε την ίδια διαδικασία πρόβλεψης και για άλλες δομές όπως ο ψευδοκόμβος τύπου M.

Βιβλιογραφία

[1] *Available online:*. https://github.com/chriskor1/rna.git, 2022.

[2] https://www.thoughtco.com/dna-versus-rna-608191.

[3] https://www.dreamstime.com/transfer-rna-genetic-code-trna-adaptor-molecule-composed-necessary-component-translation-biological-synthesis-image113257763.

[4] https://www.researchgate.net/figure/Different-types-of-RNA-Secondary-Structures-1-First-structure-represents-a-stem-with_fig1_261030242.

[5] https://www.researchgate.net/figure/Known-viral-RNA-structures-from-stem-loops-to-complex-tRNA-like-structures-A-The_fig1_322244039.

[6] https://commons.wikimedia.org/wiki/File:Kissing-loop-interaction.jpg.

[7] Christos Andrikos, Evangelos Makris, Angelos Kolaitis, Georgios Rassias, Christos Pavlatos και Panayiotis Tsanakas. *Knotify: An Efficient Parallel Platform for RNA Pseudoknot Prediction Using Syntactic Pattern Recognition*. *Methods and Protocols*, 5(1), 2022.

[8] Evangelos Makris, Angelos Kolaitis, Christos Andrikos, Vrettos Moulos, Panayiotis Tsanakas και Christos Pavlatos. *An Intelligent Grammar-Based Platform for RNA H-type Pseudoknot Prediction*. *Artificial Intelligence Applications and Innovations. AIAI 2022 IFIP WG 12.5 International Workshops: MHDW 2022, 5G-PINE 2022, AIBMG 2022, ML@ HC 2022, and AIBEI 2022, Hersonissos, Crete, Greece, June 17–20, 2022, Proceedings*, σελίδες 174–186. Springer, 2022.

[9] Evangelos Makris, Angelos Kolaitis, Christos Andrikos, Vrettos Moulos, Panayiotis Tsanakas και Christos Pavlatos. *Knotify+: Toward the Prediction of RNA H-Type Pseudoknots, Including Bulges and Internal Loops*. *Biomolecules*, 13(2):308, 2023.

[10] Βικιπαίδεια. *RNA — Βικιπαίδεια, Η Ελεύθερη Εγκυκλοπαίδεια*. https://el.wikipedia.org/wiki/RNA, 2023.

[11] Cecie Starr Christine A. Evers και Lisa Starr. *Βιολογία Βασικές Έννοιες και Αρχές*.

[12] Βικιπαίδεια. *Αγγελιαφόρο RNA — Βικιπαίδεια, Η Ελεύθερη Εγκυκλοπαίδεια*. https://el.wikipedia.org/wiki/%CE%91%CE%B3%CE%B3%CE%B5%CE%BB%CE%B9%CE%B1%CF%86%CF%8C%CF%81%CE%BF_RNA, 2022.

[13] Βικιπαίδεια. *Μεταφορικό RNA — Βικιπαίδεια, Η Ελεύθερη Εγκυκλοπαίδεια*. https://el.wikipedia.org/wiki/%CE%9C%CE%B5%CF%84%CE%B1%CF%86%CE%BF%CF%81%CE%B9%CE%BA%CF%8C_RNA#:~:text=%CE%A4%CE%BF%20%CE%BC%CE%B5%CF%84%CE%B1%CF%86%CE%BF%CF%81%CE%B9%CE%BA%CF%8C%20RNA%20%CE%AE%20tRNA,%CF%84%CE%B9%CF%82%20%CE%BF%CE%B4%CE%B7%CE%B3%CE%AF%CE%B5%CF%82%20%CF%84%CE%BF%CF%85%20%CE%B1%CE%B3%CE%B3%CE%B5%CE%BB%CE%B9%CE%B1%CF%86%CF%8C%CF%81%CE%BF%CF%85%20RNA., 2022.

[14] Wikipedia contributors. *D arm — Wikipedia, The Free Encyclopedia*. https://en.wikipedia.org/w/index.php?title=D_arm&oldid=950900873, 2020.

[15] Wikipedia contributors. *T arm — Wikipedia, The Free Encyclopedia*. https://en.wikipedia.org/w/index.php?title=T_arm&oldid=1083286316, 2022.

[16] Βικιπαίδεια. *Ριβοσωμικό RNA — Βικιπαίδεια, Η Ελεύθερη Εγκυκλοπαίδεια*. https://el.wikipedia.org/wiki/%CE%A1%CE%B9%CE%B2%CE%BF%CF%83%CF%89%CE%BC%CE%B9%CE%BA%CF%8C_RNA, 2021.

[17] Wikipedia contributors. *Small nuclear RNA — Wikipedia, The Free Encyclopedia*. https://en.wikipedia.org/w/index.php?title=Small_nuclear_RNA&oldid=1130879836, 2023.

[18] Wikipedia contributors. *Small nucleolar RNA — Wikipedia, The Free Encyclopedia*. https://en.wikipedia.org/w/index.php?title=Small_nucleolar_RNA&oldid=1128877426, 2022.

[19] Wikipedia contributors. *MicroRNA — Wikipedia, The Free Encyclopedia*. https://en.wikipedia.org/w/index.php?title=MicroRNA&oldid=1138025656, 2023.

[20] Mi Ri Park, Jang Kyun Seo και Kook Hyung Kim. *Viral and nonviral elements in potexvirus replication and movement and in antiviral responses*. *Advances in virus research*, 87:75–112, 2013.

[21] Wikipedia contributors. *Small interfering RNA — Wikipedia, The Free Encyclopedia*. https://en.wikipedia.org/w/index.php?title=Small_interfering_RNA&oldid=1136167737, 2023.

[22] Wikipedia contributors. *Small conditional RNA — Wikipedia, The Free Encyclopedia*. https://en.wikipedia.org/w/index.php?title=Small_conditional_RNA&oldid=985040485, 2020.

[23] Wikipedia contributors. *Stem-loop — Wikipedia, The Free Encyclopedia*. https://en.wikipedia.org/w/index.php?title=Stem-loop&oldid=1112992981, 2022.

[24] Wikipedia contributors. *Kissing stem-loop — Wikipedia, The Free Encyclopedia*. https://en.wikipedia.org/w/index.php?title=Kissing_stem-loop&oldid=1122308937, 2022.

[25] Wikipedia contributors. *Internal loop — Wikipedia, The Free Encyclopedia*. https://en.wikipedia.org/w/index.php?title=Internal_loop&oldid=1136264616, 2023.

[26] Xiaofeng Zheng and Philip C. Bevilacqua. *Straightening of bulged RNA by the double-stranded RNA-binding domain from the protein kinase PKR*. https://www.pnas.org/doi/10.1073/pnas.011355798#:~:text=Bulges%20are%20among%20the%20most,interactions%20with%20proteins%20(21)., 2000.

[27] Quasispecies. *Multiloop*. https://eternagame.fandom.com/wiki/Multiloop, 2012.

[28] Wikipedia contributors. *Pseudoknot — Wikipedia, The Free Encyclopedia*. https://en.wikipedia.org/w/index.php?title=Pseudoknot&oldid=1044438287, 2021.

[29] Ian Brierley, Simon Pennell and Robert J. C. Gilbert . *Viral RNA pseudoknots: versatile motifs in gene expression and replication*. https://www.nature.com/articles/nrmicro1704#citeas, 2007.

[30] David W Staple και Samuel E Butcher. *Pseudoknots: RNA structures with diverse functions*. *PLoS biology*, 3(6):e213, 2005.

[31] Jens Reeder και Robert Giegerich. *Design, implementation and evaluation of a practical pseudoknot folding algorithm based on thermodynamics*. *BMC bioinformatics*, 5:1–12, 2004.

[32] Elena Rivas και Sean R Eddy. *A dynamic programming algorithm for RNA structure prediction including pseudoknots*. *Journal of molecular biology*, 285(5):2053–2068, 1999.

[33] Robert M Dirks και Niles A Pierce. *A partition function algorithm for nucleic acid secondary structure including pseudoknots*. *Journal of computational chemistry*, 24(13):1664–1677, 2003.

[34] Hosna Jabbari, Ian Wark, Carlo Montemagno και Sebastian Will. *Knotty: efficient and accurate prediction of complex RNA pseudoknot structures*. *Bioinformatics*, 34(22):3849–3856, 2018.

[35] Kengo Sato, Yuki Kato, Michiaki Hamada, Tatsuya Akutsu και Kiyoshi Asai. *IPknot: fast and accurate prediction of RNA secondary structures with pseudoknots using integer programming*. *Bioinformatics*, 27(13):i85–i93, 2011.

[36] Stanislav Bellaousov και David H Mathews. *ProbKnot: fast prediction of RNA secondary structure including pseudoknots*. *Rna*, 16(10):1870–1880, 2010.

[37] Liang Zhang, He Zhang, David H Mathews και Liang Huang. *ThreshKnot: Thresholded ProbKnot for improved RNA secondary structure prediction*. *arXiv preprint arXiv:1912.12796*, 2019.

[38] Bjarne Knudsen και Jotun Hein. *Pfold: RNA secondary structure prediction using stochastic context-free grammars*. *Nucleic acids research*, 31(13):3423–3428, 2003.

[39] Zsuzsanna Sükösd, Bjarne Knudsen, Morten Værum, Jørgen Kjems και Ebbe S Andersen. *Multithreaded comparative RNA secondary structure prediction using stochastic context-free grammars*. *BMC bioinformatics*, 12:1–8, 2011.

[40] Jakob Skou Pedersen, Irmtraud Margret Meyer, Roald Forsberg, Peter Simmonds και Jotun Hein. *A comparative method for finding and folding RNA secondary structures within protein-coding regions*. *Nucleic acids research*, 32(16):4925–4936, 2004.

[41] Linyu Wang, Yuanning Liu, Xiaodan Zhong, Haiming Liu, Chao Lu, Cong Li και Hao Zhang. *DMFold: A novel method to predict RNA secondary structure with pseudoknots based on deep learning and improved base pair maximization principle*. *Frontiers in genetics*, 10:143, 2019.

[42] Kangkun Mao, Jun Wang και Yi Xiao. *Prediction of RNA secondary structure with pseudoknots using coupled deep neural networks*. *Biophysics Reports*, 6:146–154, 2020.

[43] Yili Wang, Yuanning Liu, Shuo Wang, Zhen Liu, Yubing Gao, Hao Zhang και Liyan Dong. *ATTfold: RNA secondary structure prediction with pseudoknots based on attention mechanism*. *Frontiers in Genetics*, 11:612086, 2020.

[44] Wikipedia contributors. *Syntactic pattern recognition — Wikipedia, The Free Encyclopedia*. https://en.wikipedia.org/wiki/Syntactic_pattern_recognition#:~:text=Syntactic%20pattern%20recognition%20or%20structural,set%20of%20symbolic%2C%20nominal%20features., 2019.

[45] Wikipedia contributors. *Context-free grammar — Wikipedia, The Free Encyclopedia*. https://en.wikipedia.org/wiki/Context-free_grammar, 2022. [Online; accessed 16-February-2023].

[46] Wikipedia contributors. *Abstract syntax tree — Wikipedia, The Free Encyclopedia*. https://en.wikipedia.org/wiki/Abstract_syntax_tree, 2023.

[47] Klaas Sikkel και Anton Nijholt. *Parsing of Context Free Languages*. https://link.springer.com/chapter/10.1007/978-3-662-07675-0_2#:~:text=Special%20subclasses%20of%20the%20context,developed%20for%20context%2Dfree%20grammars., 2013.

[48] Wikipedia contributors. *CYK algorithm — Wikipedia, The Free Encyclopedia*. https://en.wikipedia.org/wiki/CYK_algorithm, 2022. [Online; accessed 16-February-2023].

[49] Wikipedia contributors. *Earley parser — Wikipedia, The Free Encyclopedia*. https://en.wikipedia.org/wiki/Earley_parser, 2022.

[50] Vladimir Makarov. *YAEP (Yet Another Earley Parser) - C interface*. https://github.com/vnmakarov/yaep/blob/master/doc/yaep.pdf, 2015.

[51] Jihong Ren, Baharak Rastegari, Anne Condon και Holger H Hoos. *HotKnots: heuristic prediction of RNA secondary structures including pseudoknots*. *Rna*, 11(10):1494–1504, 2005.

[52] Michael Bon, Graziano Vernizzi, Henri Orland και Anthony Zee. *Topological classification of RNA structures*. *Journal of molecular biology*, 379(4):900–911, 2008.