Ανοικτό Πανεπιστήμιο Κύπρου

Σχολή Θετικών και Εφαρμοσμένων Επιστημών

Μεταπτυχιακό Πρόγραμμα Σπουδών *Ασφάλεια Υπολογιστών και Δικτύων*

Μεταπτυχιακή Διατριβή

Χρήση Γενετικών Αλγορίθμων Για Παραγωγή Κρυπτογραφικών Συναρτήσεων

Σπυρίδων Παπαϊωάννου

Επιβλέπων Καθηγητής
Δρ. Κωνσταντίνος Λιμνιώτης

Νοέμβριος 2023

Ανοικτό Πανεπιστήμιο Κύπρου

Σχολή Θετικών και Εφαρμοσμένων Επιστημών

Μεταπτυχιακό Πρόγραμμα Σπουδών *Ασφάλεια Υπολογιστών και Δικτύων*

Μεταπτυχιακή Διατριβή

Χρήση Γενετικών Αλγορίθμων Για Παραγωγή Κρυπτογραφικών Συναρτήσεων

Σπυρίδων Παπαϊωάννου

Επιβλέπων Καθηγητής
Δρ. Κωνσταντίνος Λιμνιώτης

Η παρούσα μεταπτυχιακή διατριβή υποβλήθηκε προς μερική εκπλήρωση των απαιτήσεων για απόκτηση μεταπτυχιακού τίτλου σπουδών στην Ασφάλεια Υπολογιστών και Δικτύων από τη Σχολή Θετικών και Εφαρμοσμένων Επιστημών του Ανοικτού Πανεπιστημίου Κύπρου.

Νοέμβριος 2023

ΛΕΥΚΗ ΣΕΛΙΔΑ

Περίληψη

Η ασφάλεια των κρυπτογραφικών συμμετρικών αλγορίθμων έγκειται, κατά τον μεγαλύτερο βαθμό στην εκάστοτε χρησιμοποιούμενη λογική συνάρτηση. Παρά τις πολλές γνωστές κατασκευές Boolean functions με αποδεδειγμένες καλές ιδιότητες, η δημιουργία μιας Boolean function που να επιτυγχάνει ταυτόχρονα όλες τις απαιτούμενες κρυπτογραφικές ιδιότητες παραμένει μια πρόκληση. Στην παρούσα διπλωματική διατριβή εξετάζεται η αναζήτηση/κατασκευή λογικών κρυπτογραφικών συναρτήσεων που να παρουσιάζουν, ταυτόχρονα, όσο αυτό είναι δυνατόν, καλά κρυπτογραφικά χαρακτηριστικά (αλγεβρικός βαθμός, αν η συνάρτηση είναι ισορροπημένη, ανθεκτικότητα σε αλγεβρικές επιθέσεις και μη γραμμικότητα). Δεδομένης και της ανόδου της τεχνητής νοημοσύνης τα τελευταία χρόνια, επιχειρείται ένας συνδυασμός των δύο επιστημών. Για αυτόν τον σκοπό, δημιουργήθηκε ένας γενετικός αλγόριθμος για αναζήτηση τέτοιων συναρτήσεων. Ο αλγόριθμος δημιουργήθηκε εξ΄ αρχής.

Η χρήση γενετικού αλγορίθμου για την αναζήτηση λογικών συναρτήσεων δεν είναι κάτι νέο. Η καινοτομία και η διαφορά από παρόμοιες προσπάθειες, έγκειται στην χρήση γνωστών κατηγοριών συναρτήσεων (Carlet-Feng και bent) με ήδη καλά χαρακτηριστικά, συνδυαστικά με τυχαίες συναρτήσεις.

Ο αλγόριθμος εκτελεί τρεις διαφορετικές μεθοδολογίες αναζήτησης. Μία έχοντας αποκλειστικά τυχαίες συναρτήσεις ως αρχικό πληθυσμό, μία έχοντας τυχαίες συναρτήσεις με Carlet-Feng και συναρτήσεων που παράγονται από αυτήν μέσω bit swapping της αρχικής, καθώς και μία όπου ο αρχικός πληθυσμός αποτελείται από συναρτήσεις bent και Carlet-Feng συναρτήσεις μαζί με τις παραγόμενες από το bit swapping.

Οι τελικές παραγόμενες συναρτήσεις βρίσκονται αρκετά κοντά στις αρχικές συναρτήσεις Carlet-Feng, όσον αφορά τα χαρακτηριστικά τους. Μάλιστα, για την περίπτωση της αναζήτησης με Carlet-Feng και τις παράγωγες συναρτήσεις τους συνδυαστικά με τυχαίες συναρτήσεις και αριθμό μεταβλητών n=8, ο αλγόριθμος επιτυγχάνει την ίδια μη γραμμικότητα με την Carlet-Feng συνάρτηση. Συνολικά, η μεθοδολογία που χρησιμοποιεί τυχαίες συναρτήσεις συνδυαστικά με Carlet-Feng και παραγόμενες από αυτές συναρτήσεις, έχει καλύτερη επίδοση από τις άλλες μεθοδολογίες, ενώ η μεθοδολογία συναρτήσεων bent με Carlet-Feng και τις παράγωγες δεν παρουσιάζει βελτίωση.

Summary

The security of symmetric cryptographic algorithms largely depends on the underlying logical function used. Despite the existence of many well-known Boolean functions with proven good properties, creating a Boolean function that simultaneously achieves all the required cryptographic properties remains a challenge. This dissertation explores the search/construction of logical cryptographic functions that exhibit, simultaneously, as much as possible, good cryptographic characteristics (algebraic degree, whether the function is balanced, resistance to algebraic attacks, and nonlinearity). Given the recent rise of artificial intelligence, an attempt is made to combine the two fields. For this purpose, a genetic algorithm was created to search for such functions. The algorithm was developed from scratch.

The use of a genetic algorithms for searching logical functions does not constitute a new approach. The innovation and difference from similar efforts lies in the use of known categories of functions (Carlet-Feng and bent) with already good characteristics, combined with random functions.

The algorithm executes three different search methodologies. One exclusively using random functions as the initial population, one using random functions with Carlet-Feng and functions generated from it through bit swapping of the original, and one where the initial population consists of bent functions and Carlet-Feng functions along with those generated from bit swapping.

The final generated functions are quite close to the initial Carlet-Feng functions in terms of their characteristics. In fact, for the case of searching with Carlet-Feng and their derived functions combined with random functions and the number of variables being n=8, the algorithm achieves the same nonlinearity as the original Carlet-Feng function. Overall, the methodology using random functions combined with Carlet-Feng and the functions generated from them, performs better than the other methodologies, while the methodology of bent functions with Carlet-Feng and their derivatives does not show improvement.

Ευχαριστίες

Θα ήθελα να ευχαριστήσω τους καθηγητές μου για την γνώση και τις κατευθύνσεις που μου παρείχαν, και ειδικότερα τον εξαιρετικό Δρ. Κωνσταντίνο Λιμνιώτη, χάρη στην συμβολή και καθοδήγηση του οποίου έγινε πράξη η παρούσα διπλωματική καθώς και για την άψογη συνεργασία μας. Θα ήθελα να ευχαριστήσω τις αδερφές μου Ελευθερία, Ερασμία και Θεοδώρα για την πολύτιμη βοήθειά και στήριξή τους καθώς και την σύντροφό μου, Φανή για την αμέριστη υπομονή και στήριξή της.

Τον φίλο μου, Παναγιώτη Αντωνίου, για την βοήθεια που προσέφερε.

Κυρίως όμως, θα ήθελα να ευχαριστήσω και να αφιερώσω την παρούσα διατριβή στους γονείς μου, Ευτυχία και Βασίλη, χάρη στην προσπάθεια, τους κόπους και την επιμονή των οποίων οφείλω την πορεία και τα επιτεύγματά μου.

Στην μνήμη του αγαπημένου μου πατέρα.

Περιεχόμενα

Κεφάλαιο 1
Εισαγωγή

Η κρυπτογραφία αποτελεί ένα από τα θεμέλια της σύγχρονης πληροφορικής και επικοινωνιών, αποσκοπώντας στη διατήρηση της εμπιστευτικότητας και της ακεραιότητας των δεδομένων και των επικοινωνιών. Η σημασία της κρυπτογραφίας εκδηλώνεται ευρέως στην καθημερινή ζωή, από τις ασφαλείς συναλλαγές στο διαδίκτυο μέχρι την προστασία των προσωπικών δεδομένων [1].

Πολλοί κρυπτογραφικοί αλγόριθμοι που χρησιμοποιούνται σήμερα βασίζονται σε μαθηματικές κατασκευές με εξαιρετικές ιδιότητες ασφάλειας και τυχαιότητας. Αυτές οι μαθηματικές κατασκευές συχνά βασίζονται στις λεγόμενες λογικές συναρτήσεις (Boolean functions). Αυτές οι συναρτήσεις εμφανίζονται ιδίως στους συμμετρικούς κρυπτογραφικούς αλγορίθμους, με έμφαση στους κρυπταλγορίθμους ροής (stream ciphers) αλλά και στους κρυπταλγορίθμους τμήματος (block ciphers), οι οποίοι αποτελούν σημαντική κατηγορία κρυπτογραφικών αλγορίθμων, ειδικά στο τα τελευταία χρόνια που πλέον προτείνονται για ασφάλεια και στο πλαίσιο των συσκευών Internet of Things (IoT) [2].

Παρά τις πολλές γνωστές κατασκευές Boolean functions με αποδεδειγμένες καλές ιδιότητες, η δημιουργία μιας Boolean function που να επιτυγχάνει ταυτόχρονα όλες τις απαιτούμενες κρυπτογραφικές ιδιότητες παραμένει μια πρόκληση. Ως αποτέλεσμα, ο χώρος αυτός παραμένει ανοικτός για ενδιαφέρουσες ερευνητικές εξελίξεις.

Μια εναλλακτική μέθοδος που έχει αναδειχθεί για τη δημιουργία Boolean functions που θα μπορούσαν να χρησιμοποιηθούν για κρυπτογραφικούς σκοπούς είναι η χρήση γενετικών αλγορίθμων [3], οι οποίοι αποτελούν μία ειδική περίπτωση αλγορίθμων τεχνητής νοημοσύνης. Αυτή η μέθοδος φαίνεται να είναι πολλά υποσχόμενη, καθώς επιτρέπει τη δημιουργία συναρτήσεων που πληρούν τις επιθυμητές κρυπτογραφικές

ιδιότητες. Το αντικείμενο της παρούσας διατριβής είναι να εξετάσει αυτές τις τεχνικές με μια διαφοροποίηση από τις προηγούμενες προσεγγίσεις. Συγκεκριμένα, στην παρούσα διατριβή θα επιχειρηθεί η χρήση γενετικών αλγορίθμων, με διάφορες παραμέτρους, οι οποίοι ως αφετηρία δεν θα ξεκινούν μόνο με τυχαίες συναρτήσεις - όπως είναι η τυπική μέχρι τώρα περίπτωση - αλλά και γνωστές συναρτήσεις που ήδη επιτυγχάνουν κάποιες από τις κρυπτογραφικές ιδιότητες που αναζητούνται.

Στο πλαίσιο αυτής της διατριβής, θα εξεταστεί η απόδοση αυτής της προσέγγισης και η εφαρμογή της στην δημιουργία κρυπτογραφικών Boolean functions. Θα αναλυθούν τα πλεονεκτήματα, οι περιορισμοί και οι δυνητικές εφαρμογές αυτής της νέας προσέγγισης στον τομέα της κρυπτογραφίας. Επιπλέον, θα διερευνηθεί πώς αυτή η μέθοδος μπορεί να συμβάλει στη βελτίωση της ασφάλειας των συστημάτων κρυπτογραφίας και την αντιμετώπιση των προκλήσεων που παρουσιάζονται στη σύγχρονη κρυπτογραφία.

1.1 Σκοπός Έρευνας

Σκοπός της παρούσας διατριβής είναι η εξερεύνηση της δυνατότητας κατασκευής κρυπτογραφικών συναρτήσεων με βέλτιστα χαρακτηριστικά για χρήση ιδίως σε αλγορίθμους συμμετρικής κρυπτογραφίας. Για την αναζήτηση και εύρεση αυτών, θα χρησιμοποιηθούν γενετικοί αλγόριθμοι, με μία διαφορετική προσέγγιση σε σχέση με τις υφιστάμενες, αναφορικά με το πώς θα επιλεγούν οι αρχικές λογικές συναρτήσεις που θα έχουν ρόλο «γονέα» στους γενετικούς αυτούς αλγορίθμους.

Ο απώτερος στόχος είναι η δημιουργία λογικών συναρτήσεων που να είναι ισοβαρείς, να έχουν υψηλό αλγεβρικό βαθμό, υψηλή μη γραμμικότητα και υψηλή ανθεκτικότητα σε αλγεβρικές επιθέσεις.

1.2 Ερευνητικά Ερωτήματα

• Μπορεί η τεχνητή νοημοσύνη να δημιουργήσει κρυπτογραφικές συναρτήσεις με καλές ιδιότητες;

Η πρώτη ερευνητική ερώτηση επικεντρώνεται στην ικανότητα της τεχνητής νοημοσύνης να δημιουργήσει ασφαλείς κρυπτογραφικές λύσεις. Η τεχνητή νοημοσύνη διαθέτει τη

δυνατότητα να εξερευνήσει πολύπλοκους χώρους λύσεων και να ανακαλύψει συναρτήσεις που ενδέχεται να είναι ανθεκτικές σε κρυπταναλυτικές επιθέσεις.

• Βελτιώνονται οι υπάρχουσες τεχνικές τεχνητής νοημοσύνησης για την κατασκευή κρυπτογραφικών λογικών συναρτήσεων εάν επιλέξουμε κατάλληλα τις αρχικές συναρτήσεις που θα αποτελέσουν τον «γονέα» σε έναν εξελικτικό/γενετικό αλγόριθμο;

Το δεύτερο ερευνητικό ερώτημα έγκειται στο να εξεταστεί μία νέα προσέγγιση αναφορικά με τη χρήση γενετικών αλγορίθμων, η οποία σχετίζεται με την κατάλληλη επιλογή όχι τυχαίων αλλά συγκεκριμένων λογικών συναρτήσεων ως αφετηρία, οι οποίες ήδη επιτυγχάνουν αποδεδειγμένα κάποιες καλές κρυπτογραφικές ιδιότητες. Θα διερευνηθεί κατά πόσον με αυτή την προσέγγιση οι γενετικοί αλγόριθμοι μπορούν να έχουν ως αποτέλεσμα κάποια ακόμα καλύτερη λογική συνάρτηση – και, ενδεχομένως, και πόσα γρήγορα μπορούν να ολοκληρώσουν τη διαδικασία οι εν λόγω αλγόριθμοι.

• Μπορεί η χρήση τεχνητής νοημοσύνης να βελτιώσει ή να ενισχύσει υπάρχοντες αλγόριθμους;

Η τρίτη ερευνητική ερώτηση ασχολείται με τη δυνατότητα βελτίωσης των υφιστάμενων κρυπτογραφικών λύσεων με τη χρήση τεχνητής νοημοσύνης. Η ανάπτυξη προηγμένων αλγορίθμων που βασίζονται σε συνεργασία ανθρώπου και μηχανής μπορεί να οδηγήσει σε πιο ασφαλή συστήματα κρυπτογραφίας. Ειδικότερα, στο πλαίσιο της διατριβής, θα εξεταστεί αν, έχοντας ως αφετηρία λογικές συναρτήσεις με καλές κρυπτογραφικές ιδιότητες, θα επιτύχουμε τη δημιουργία νέων λογικών συναρτήσεων με ακόμα καλύτερες ιδιότητες (και, άρα, υποσχόμενες ως προς την βελτίωση/ενίσχυση υπαρχόντων κρυπτογραφικών αλγορίθμων).

• Πόσο επηρεάζουν οι διάφορες παράμετροι ενός γενετικού αλγορίθμου την αποτελεσματικότητα αυτού;

Το εν λόγω ερευνητικό ερώτημα σχετίζεται με τη διερεύνηση της επίδρασης που έχουν οι διάφορες σχεδιαστικές παράμετροι στην απόδοση του αλγορίθμου (όπως, π.χ., η επιλογή της fitness function) – δηλαδή στο πόσο γρήγορα ολοκληρώνει τους υπολογισμούς του, καθώς και εάν το τελικό αποτέλεσμα είναι αποδεκτό ή όχι.

• Σε τί βαθμό μπορεί να συνεισφέρει η τεχνητή νοημοσύνη στην εύρεση κρυπτογραφικών συναρτήσεων με καλά χαρακτηριστικά;

Τέλος, η τέταρτη ερευνητική ερώτηση αναλύει τον βαθμό συνεισφοράς της τεχνητής νοημοσύνης στην εύρεση κρυπτογραφικών συναρτήσεων με επιθυμητά χαρακτηριστικά. Η αξιολόγηση αυτής της συνεισφοράς είναι κρίσιμη για την κατανόηση της αξίας της τεχνητής νοημοσύνης στον τομέα της κρυπτογραφίας.

1.3 Προτεινόμενη μεθοδολογία

Για την επίτευξη των στόχων της διατριβής, θα αναπτυχθούν προγράμματα σε γλώσσα προγραμματισμού Python τα οποία θα ενσωματώνουν τεχνικές τεχνητής νοημοσύνης. Αυτά τα προγράμματα θα χρησιμοποιηθούν για την δημιουργία και την αξιολόγηση κρυπτογραφικών συναρτήσεων βάσει διαφόρων κριτηρίων, όπως η μη γραμμικότητα, η ισορροπία, ο αλγεβρικός βαθμός και η αλγεβρική ανθεκτικότητα.

Περαιτέρω, προκειμένου να ελέγχονται οι κρυπτογραφικές ιδιότητες των διαφόρων λογικών συναρτήσεων, χρησιμοποιείται το γνωστό λογισμικό Sage [4] το οποίο είναι βασισμένο σε Python.

1.4 Επισκόπηση

Κεφάλαιο 1: Το παρόν κεφάλαιο, το οποίο εξυπηρετεί σαν μία πρώτη εισαγωγή στο αντικείμενο μελέτης της διπλωματικής διατριβής. Τονίζεται η σημασία της κρυπτογραφίας στην σημερινή εποχή και αναφέρονται οι cryptographic Boolean functions, το κύριο πεδίο εφαρμογής τους καθώς και σε τι αποσκοπεί η παρούσα μελέτη.

Κεφάλαιο 2: Αναλύονται περεταίρω οι Boolean functions στην κρυπτογραφία και τα χαρακτηριστικά τους. Εξηγούνται κάποιες βασικές επιθέσεις κρυπτανάλυσης σε αυτές και πως μπορεί να εξετασθεί η ανθεκτικότητα των συναρτήσεων απέναντι στις εν λόγω επιθέσεις.

Κεφάλαιο 3: Γενική εισαγωγή στους γενετικούς αλγορίθμους. Παρουσιάζεται η βασική δομή ενός γενετικού αλγορίθμου καθώς και οι τελεστές και πώς αυτοί συνδυάζονται

κατά την δόμησή του. Ακόμα, αναλύεται η λειτουργία κάποιων από τους γνωστότερους τελεστές.

Κεφάλαιο 4: Ανασκόπηση βιβλιογραφίας όσον αφορά την χρήση γενετικών αλγορίθμων για παραγωγή Boolean functions. Περιγραφή στις ήδη υπάρχουσες προσεγγίσεις που έχουν χρησιμοποιηθεί για το εν λόγω θέμα.

Κεφάλαιο 5: Περιγραφή και ανάλυση της προσέγγισης που ακολουθήθηκε στην εν λόγω διατριβή αλλά και του προγράμματος που αναπτύχθηκε. Πλήρης ανάλυση της φιλοσοφίας του προγράμματος και του πως υλοποιήθηκε κάθε τελεστής του γενετικού αλγορίθμου. Περιγραφή της δομής του γενετικού αλγορίθμου.

Κεφάλαιο 6: Παρουσίαση αποτελεσμάτων με διαγράμματα καθώς και των καλύτερων συναρτήσεων που παράχθηκαν.

Κεφάλαιο 7: Επίλογος

Κεφάλαιο 2
Λογικές συναρτήσεις στην κρυπτογραφία & ιδιότητές τους

Στους συμμετρικούς κρυπτογραφικούς αλγορίθμους χρησιμοποιούνται συχνά, ως βασικά δομικά συσταστικά, κρυπτογραφικές λογικές συναρτήσεις (Boolean functions). Αυτές επιλέγονται και χρησιμοποιούνται, για παράδειγμα, στην γεννήτρια κλειδοροής σε διάφορους κρυπτογραφικούς συμμετρικούς αλγόριθμους ροής (stream ciphers) προκειμένου οι παραγόμενες κλειδοροές να παρουσιάζουν καλά χαρακτηριστικά τυχαιότητας [5]. Άλλες χρήσεις τους στην κρυπτογραφία είναι τόσο για κρυπταλγορίθμους τμήματος (block ciphers) όσο και σε κρυπτογραφικές συναρτήσεις κατακερματισμού (hash functions).

Οι λογικές συναρτήσεις γενικότερα έχουν πληθώρα εφαρμογών, πέραν της κρυπτογραφίας. Αποτελούνται από n μεταβλητές και μία έξοδο, των οποίων οι τιμές ανήκουν στο σώμα {0,1}. Ο αριθμός n των μεταβλητών καθορίζει το μέγεθος τη συνάρτησης και, άρα, με το μέγεθος του λεγόμενο πίνακα αληθείας (truth table) αυτής, ο οποίος είναι ο πίνακας που περιγράφει,για κάθε πιθανή είσοδο της συνάρτησης, ποια είναι η έξοδός της. Ως εκ τούτου, το πλήθος των γραμμών ενός πίνακα αληθείας για μία συνάρτηση n μεταβλητών ισούται με 2n. Για παράδειγμα, ο πίνακας αληθείας μίας συνάρτησης n=10 μεταβλητών θα έχει μέγεθος 1024 (δηλαδή η έξοδός του, που περιγράφει πλήρως τη συνάρτηση, έχει μέγεθος 1024).

Οι πιο συνήθεις τρόποι αναπαράστασής τους είναι είτε μέσω πίνακα αληθείας είτε μέσω της κανονικής αλγεβρικής μορφής τους, η οποία αναπαριστά την συνάρτηση σαν άθροισμα XOR γινομένων μεταβλητών. Στον παρακάτω πίνακα περιγράφεται πίνακας αληθείας μίας τυχαία επιλεγμένης συνάρτησης τριών μεταβλητών, οπότε και έχει

μέγεθος 8 (προφανώς, το πλήθος όλων των πιθανών λογικών συναρτήσεων n μεταβλητών ισούται με 2^{2^n}.

X1	X2	X3	Έξοδος
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

Πίνακας 1. Πίνακας αληθείας λογικής συνάρτησης 3 μεταβλητών.

Ο παραπάνω πίνακας 1 αντιστοιχεί σε μία λογική συνάρτηση f η οποία δίνει τα εξής αποτελέσματα:

f(000) = 0, f(001)= 0, f(010) = 0, f(011) = 1, f(100) = 1, f(101) = 1, f(110) = 1 , f(111) = 0

Γνωρίζοντας μία από τις δύο απεικονίσεις της συνάρτησης, είναι εύκολο να μετατραπεί η πληροφορία στην αντίστοιχη άλλη μορφή. Δηλαδή, γνωρίζοντας τον πίνακα αληθείας, είναι εύκολο να βρεθεί η αλγεβρική κανονική μορφή και το αντίστροφο.

Για το ανωτέρω παράδειγμα δίνεται η αντίστοιχη αλγεβρική κανονική μορφή:

x1+x2x3

2.1 Χαρακτηριστικά και ιδιότητες συναρτήσεων

Όπως έχει αποδειχθεί, οι κρυπτογραφικές λογικές συναρτήσεις πρέπει να έχουν συγκεκριμένες ιδιότητες (κρυπτογραφικά κριτήρια) προκειμένου οι αντίστοιχοι κρυπτογραφικοί αλγόριθμοι να παρουσιάζουν ανθεκτικότητα σε συγκεκριμένες τεχνικές

κρυπτανάλυσης. Οι βασικοί ορισμοί και τα βασικά κρυπτογραφικά κριτήρια παρατίθενται σε αυτήν την ενότητα.

Γραμμικές συναρτήσεις (linear functions) ονομάζονται αυτές που στην αλγεβρική κανονική μορφή τους δεν έχουν κανένα γινόμενο μεταβλητών και αποτελούνται αποκλειστικά από πράξεις XOR. Αν υπάρχει έστω και ένα γινόμενο τότε η συνάρτηση θεωρείται μη γραμμική (nonlinear).

Για παράδειγμα η συνάρτηση $f(x1,x2,x3)=x1 \oplus x3$ θεωρείται γραμμική ενώ η $f(x1,x2,x3)=x1x2 \oplus x3$ είναι μη γραμμική.

Επιπλέον, κάθε συνάρτηση έχει συγκεκριμένα χαρακτηριστικά, ανάλογα με τις εισόδους και εξόδου του πίνακα αληθείας της. Από αυτά τα χαρακτηριστικά, η παρούσα μεταπτυχιακή διατριβή ασχολείται με :

Βάρος (weight) της συνάρτησης f (συμβολίζεται **ως wt(f)**) ορίζεται ως το πλήθος των εξόδων με τιμή «1». Αν το πλήθος των εξόδων είναι ισόποσα μοιρασμένο σε «1» και «0» τότε η συνάρτηση λέγεται ισοβαρής. Πρέπει δηλαδή να ισχύει $wt(f)=2^{n-1}$.

Αλγεβρικός βαθμός της συνάρτησης είναι το πλήθος των μεταβλητών που εμφανίζονται στο μεγαλύτερο γινόμενό της αλγεβρικής κανονικής μορφής της. Για παράδειγμα, στην συνάρτηση που μελετήθηκε παραπάνω ο αλγεβρικός βαθμός της είναι 2. Αν ο βαθμός μίας συνάρτησης ισούται με 1, τότε αυτή η συνάρτηση λέγεται γραμμική. Για μία ισοβαρή συνάρτηση n μεταβλητών, ο μέγιστος αλγεβρικός βαθμός που μπορεί να έχει είναι n-1.

Απόσταση Hamming δύο συναρτήσεων f και g ($wt(f \oplus g)$) ορίζεται ως το πλήθος θέσεων στο οποίο διαφέρουν οι πίνακες αληθείας των συναρτήσεων.
Μη Γραμμικότητα μιας συνάρτησης **(nl(f))** ισούται με το μικρότερο πλήθος στοιχείων του πίνακα εξόδου τα οποία αν μεταβληθούν μετατρέπουν την τρέχουσα συνάρτηση σε γραμμική.

2.2 Κρυπτογραφικές επιθέσεις

Τα ανωτέρω χαρακτηριστικά αποτελούν μερικά από τα σημαντικότερα κριτήρια για την αξιολόγηση της τυχαιότητας της παραγόμενης κλειδοροής καθώς και της ποιότητας της εκάστοτε συνάρτησης. Ωστόσο, υπάρχουν διάφορες τεχνικές επιθέσεων με τις οποίες ένας κρυπταναλυτής επιχειρεί να αποδυναμώσει ή να προβλέψει την ακολουθία των bits της συνάρτησης/κλειδοροής. Τα παραπάνω χαρακτηριστικά δεν εξασφαλίζουν τις κρυπτογραφικές ιδιότητες των συναρτήσεων απέναντι σε τέτοιες επιθέσεις. Πιο συγκεκριμένα, μερικοί τύποι επιθέσεων είναι:

2.2.1 Επιθέσεις συσχετίσεων (correlation attacks)

Το 1984 ο Siegenthaler [6] ανέδειξε πως αν η έξοδος κάποιας συνάρτησης ταυτίζεται με πιθανότητα $p > ½$ για τουλάχιστον μία εκ των εισόδων της, τότε εάν ένα ικανοποιητικό τμήμα της παραγόμενης κλειδοροής είναι γνωστό, τότε είναι δυνατό να βρεθεί η αρχική κατάσταση της εν λόγω γεννήτριας (υπό την έννοια ότι αποκτάται πληροφορία που επιτρέπει πολύ λιγότερους υπολογισμούς από ό,τι μία εξαντλητική αναζήτηση).

Για να αποφευχθεί αυτό, πρέπει για κάθε συνδυασμό τιμών για οποιεσδήποτε (μία ή περισσότερες) μεταβλητές εισόδου, οι τιμές εξόδου να είναι όσο γίνεται πιο ισομοιρασμένες μεταξύ των τιμών 0 και 1 (άρα να είναι όσο γίνεται η αντίστοιχη υπο-συνάρτηση ισοβαρής). Διαφορετικά, υπάρχει εμφανή συσχέτιση πως για δεδομένη τιμή εισόδου (ή συνδυασμό τιμών εισόδου) υπάρχει και δεδομένη τιμή εξόδου. Κάτι τέτοιο μειώνει την τυχαιότητα της συνάρτησης και δημιουργεί τρόπους να προβλεφθούν/ευρεθούν ζευγάρια τιμών εισόδου-εξόδου. Άρα λοιπόν, μία συνάρτηση που οι τιμές εισόδου των μεταβλητών της παρουσιάζουν άμεση συσχέτιση με την τιμή εξόδου, καθίσταται ευάλωτη σε αυτόν τον τύπο επιθέσεων. Κατά συνέπεια, καθίσταται ευάλωτος και ανασφαλής και ο ίδιος ο κρυπτογραφικός αλγόριθμος για τον οποίον χρησιμοποιείται η εκάστοτε συνάρτηση.

Το σχετικό χαρακτηριστικό το οποίο υποδεικνύει ότι μία συνάρτηση είναι ανθεκτική έναντι σε τέτοιου τύπου επιθέσεις ονομάζεται **ανθεκτικότητα σε συσχετίσεις (correlational immunity)**[7]. Συγκεκριμένα, για μία συνάρτηση με n μεταβλητές, θεωρείται ότι είναι ανθεκτική σε συσχετίσεις τάξεως k, όταν θεωρώντας σταθερές τις

τιμές σε οποιεσδήποτε k μεταβλητές εισόδου στον πίνακα, η υπό-συνάρτηση που απομένει είναι ισοβαρής.

Για παράδειγμα, έστω η συνάρτηση με πίνακα αληθείας:

X1	X2	X3	f
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

Πίνακας 2. Πίνακας αληθείας λογικής συνάρτησης 3 μεταβλητών.

Υπάρχουν 3 ζευγάρια μεταβλητών (x1,x2),(x2,x3) και (x1,x3).

Για τα ζευγάρια τιμών (x1,x2)=(0,0), (x1,x2)=(0,1), (x1,x2)=(1,0) και (x1,x2)=(1,1) οι πίνακες εξόδου είναι ισοβαρής. Ομοίως εφαρμόζεται και για τα ζεύγη (x1,x3) και (x2,x3).

Αν το παραπάνω ισχύει για όλα τα ζεύγη μεταβλητών, τότε η συνάρτηση είναι ανθεκτική σε συσχετίσεις 2ου βαθμού.

Ένα γνωστό αποτέλεσμα, που καταδεικνύει πόσο δύσκολο είναι να επιτευχθούν ταυτόχρονα όλα τα κρυπτογραφικά κριτήρια είναι το εξής: Για μία ισοβαρή συνάρτηση f με n μεταβλητές και με αλγεβρικό βαθμό d και η οποία είναι ανθεκτική σε συσχετίσεις τάξης k, όπου $1 \leq k \leq n-2$ ισχύει:

$$d \leq n - k - 1$$

το οποίο σημαίνει ότι αν θέλουμε μία ισοβαρή συνάρτηση να έχει τον μέγιστο βαθμό n-1 (κάτι που γενικά είναι επιθυμητό), τότε ουσιαστικά η συνάρτηση δεν μπορεί να είναι ανθεκτική σε συσχετίσεις.

2.2.2 Επιθέσεις προσεγγίσεων (Approximation attacks)

Έστω ότι μία μη γραμμική συνάρτηση f διαφέρει σε ελάχιστες θέσεις από μία γραμμική συνάρτηση g στον πίνακα αληθείας της και παράγει μία κλειδοροή k. Αν λοιπόν αντικατασταθεί η f με την g, τότε η κλειδοροή k' που θα παραχθεί από την g, θα διαφέρει επίσης μόλις λίγες θέσεις από την αρχική κλειδοροή k της f. Χρησιμοποιώντας τον αλγόριθμο Berlekamp-Massey [8] είναι δυνατό να προσδιοριστεί η γεννήτρια που παράγει την κλειδοροή k'. Γνωρίζοντας πλέον την k' γνωρίζουμε ταυτόχρονα και μεγάλο μέρος της κλειδοροής k, λόγω της εγγύτητας των δύο συναρτήσεων που τις παράγουν f και g. Άρα υπάρχει μία κλειδοροή k' η οποία προσεγγίζει ικανοποιητικά την αρχική ζητούμενη κλειδοροή k.

Για την αποτροπή τέτοιων επιθέσεων, θα πρέπει η μη γραμμικότητα της f να είναι μεγάλη και συνεπώς, να μην μπορεί να προσεγγιστεί ικανοποιητικά από γραμμικές συναρτήσεις.

Η βέλτιστη μη γραμμικότητα που μπορεί να επιτευχθεί από μία συνάρτηση n μεταβλητών, όπου το n είναι άρτιος, είναι $2^{n-1} - 2^{\frac{n}{2}-1}$ και οι συναρτήσεις που το επιτυγχάνουν αυτό ονομάζονται συναρτήσεις bent. Υπάρχουν διάφορες γνωστές τεχνικές κατασκευής συναρτήσεων bent – η πιο απλή μορφή συνάρτησης bent είναι, για οποιαδήποτε άρτια τιμή του n, η εξής:

$$f(x_1, x_2, \dots, x_n) = x_1x_2 + x_3x_4 + \cdots + x_{n-1}x_n$$

Ωστόσο οι bent συναρτήσεις δεν είναι ισοβαρείς. Από την άλλη πλευρά, δεν γνωρίζουμε τη μέγιστη δυνατή μη γραμμικότητα που μπορεί να επιτύχει μία ισοβαρής συνάρτηση n μεταβλητών (για n άρτιο αριθμό), ενώ ακόμα πιο πολλά ερωτήματα υπάρχουν όταν το n είναι περιττός αριθμός (όπου δεν γνωρίζουμε τη μέγιστη μη γραμμικότητα, ανεξάρτητα του αν είναι ισοβαρής ή όχι). Για σχετικά μικρές τιμές του n έχουν γίνει εξαντλητικοί υπολογισμοί προκειμένου να δούμε ποια είναι η μέγιστη μη γραμμικότητα που μπορεί να επιτευχθεί – οι τιμές αυτές αποτυπώνονται στη συνέχεια στον πίνακα 2.

Τιμή του n	Θεωρητική μέγιστη τιμή του nonlinearity	Μέγιστη τιμή του nonlinearity που έχει βρεθεί στην πράξη
7	58	56

8	120	116
9	244	240
10	496	492
11	1000	992
12	2016	2010
13	4050	4036
14	8128	8120
15	16292	16272

Πίνακας 3. Μέγιστες τιμές μη γραμμικότητας για συναρτήσεις ανά αριθμό μεταβλητών.

2.2.3 Αλγεβρικές επιθέσεις (Algebraic attacks)

Άλλη μία κατηγορία επιθέσεων είναι οι αλγεβρικές επιθέσεις. Στόχος τους είναι η απλοποίηση των μαθηματικών εκφράσεων που χρησιμοποιούνται και προτάθηκαν πρώτη φορά από τον Courtois το 2003 [9]. Έχει αποδειχθεί ότι ακόμα και αν μία συνάρτηση πληροί τα κριτήρια της ανθεκτικότητας σε συσχετίσεις και έχει υψηλή μη γραμμικότητα, δεν αρκεί για να προστατευθεί από τις αλγεβρικές επιθέσεις. Συνεπώς, σε αυτό το σημείο εισάγεται και ένα νέο επιθυμητό κριτήριο, αυτό της **ανθεκτικότητας σε αλγεβρικές επιθέσεις**.

Αναλυτικότερα, αν επικεντρωθούμε, για παράδειγμα, σε μία γεννήτρια μη γραμμικού φίλτρου, οι τελικές μαθηματικές εκφράσεις είναι εξίσου μη γραμμικές, και άρα, αρκετά σύνθετες. Για να βρεθεί το μυστικό κλειδί που χρησιμοποιήθηκε, πρέπει να λυθεί ένα πολύπλοκο σύστημα εξισώσεων στο οποίο, όσο μεγαλύτερος είναι ο βαθμός της f, τόσο πιο σύνθετες γίνονται και οι εξισώσεις. Στόχος των αλγεβρικών επιθέσεων είναι να απλοποιηθούν αυτές οι σύνθετες εξισώσεις και να μετασχηματιστούν σε χαμηλού βαθμού.

Για να απλοποιηθούν λοιπόν οι εν λόγω εξισώσεις, αρκεί να υπάρχει μία συνάρτηση g χαμηλού βαθμού που να ικανοποιεί το εξής κριτήριο:

$f*g=0$ ή $(f \oplus 1)*g=0$

Αυτό σημαίνει πως αν σε μία θέση του πίνακα αληθείας των f,g μία από τις δύο ισούται με 1, τότε η άλλη ισούται υποχρεωτικά με 0. Η g με τον μικρότερο δυνατό βαθμό που

ικανοποιεί αυτό το κριτήριο (εκτός της μηδενικής) καθορίζει και την αλγεβρική ανθεκτικότητα της f. Πρακτικά, ο βαθμός της g ισούται με την αλγεβρική ανθεκτικότητα (algebraic immunity) της f.

Άρα για να έχει υψηλή ανθεκτικότητα η f σε αλγεβρικές επιθέσεις πρέπει ο ελάχιστος βαθμός της g να είναι όσο το δυνατόν υψηλότερος.

Έχει αποδειχθεί ότι για μία συνάρτηση n μεταβλητών, η μέγιστη δυνατή αλγεβρική ανθεκτικότητα που μπορεί να επιτευχθεί ισούται με $\left\lceil \frac{n}{2} \right\rceil$. Υπάρχουν διάφορες γνωστές κατασκευές συναρτήσεων οι οποίες εγγυημένα επιτυγχάνουν τη μέγιστη δυνατή τιμή αλγεβρικής ανθεκτικότητας – για λίγες εξ αυτών όμως έχει αποδειχθεί ότι επιτυγχάνουν εγγυημένα και υψηλή μη γραμμικότητα. Μία εξ αυτών είναι η λεγόμενη συνάρτηση Carlet-Feng, που ορίζεται για οποιαδήποτε επιθυμητή τιμή του n [10].

2.3 Συμπεράσματα

Κατά τον σχεδιασμό κρυπτογραφικών αλγορίθμων ροής, η έμφαση δίνεται στην γεννήτρια κλειδοροής. Συνεπώς, για την χρήση μίας συνάρτησης σε κρυπτογραφικό αλγόριθμο, είναι αναγκαίο τα ανωτέρω χαρακτηριστικά της συνάρτησης να πετυχαίνουν μία επιθυμητή τιμή.

Η εκτέλεση κάποιας από τις παραπάνω επιθέσεις με επιτυχία οδηγεί αλληλένδετα στην αποσαφήνιση ή αποδυνάμωση του εκάστοτε κρυπτογραφικού αλγορίθμου στον οποίον χρησιμοποιείται η εν λόγω συνάρτηση. Στην συνέχεια, είναι προφανές πως ο αλγόριθμος καθίσταται άχρηστος για κρυπτογραφικές εφαρμογές, αφού αποτυγχάνει στην διασφάλιση της εμπιστευτικότητας των δεδομένων, το οποίο είναι και ο βασικός σκοπός του.

Καθώς αλλάζει το μέγεθος και οι τιμές του πίνακα αληθείας της συνάρτησης το ίδιο συμβαίνει και στα χαρακτηριστικά της. Μία μικρή αλλαγή πιθανώς να φέρει μία πολύ καλύτερη τιμή σε κάποιο χαρακτηριστικό ενώ όμως μπορεί να φέρει και μία πολύ χειρότερη σε ένα άλλο. Οπότε, η χειροκίνητη αναζήτηση μίας συνάρτησης με βέλτιστα

χαρακτηριστικά μπορεί γρήγορα να γίνει μία ανιαρή, χρονοβόρα, δύσκολη και πολύ πιθανόν, μη αποδοτική διαδικασία.

Έτσι, χρησιμοποιήθηκαν οι δυνατότητες των γενετικών αλγορίθμων προς αναζήτηση βέλτιστων ή τουλάχιστον ικανοποιητικών συναρτήσεων.

Στην παρούσα διπλωματική διατριβή χρησιμοποιήθηκαν:

- Τυχαίες συναρτήσεις δεδομένου μήκους.
- Συναρτήσεις Carlet-Feng και παράγωγές τους (βλ. υποενότητα 5.2.2 για το πώς παράγονται) οι οποίες έχουν καλή τιμή nonlinearity και είναι balanced.
- Συναρτήσεις bent οι οποίες έχουν υψηλό nonlinearity αλλά δεν ειναι balanced.

Κεφάλαιο 3
Γενετικοί Αλγόριθμοι

Επιγραμματικά, ο Δαρβίνος υποστήριξε πως το κάθε είδος κινείται από γενιά σε γενιά προς την εξέλιξη μέσω της φυσικής επιλογής. Η φυσική επιλογή σημαίνει ότι επιβιώνει και ευημερεί ο καταλληλότερος (survival of the fittest). Δηλαδή, συγκεκριμένα άτομα από κάθε πληθυσμό που έχουν καλύτερα χαρακτηριστικά ή έχουν προσαρμοστεί αποτελεσματικότερα στο περιβάλλον τους από άλλα, είναι πιο πιθανό να επιβιώσουν και να αναπαραχθούν, κληροδοτώντας αυτά τα χαρακτηριστικά και στις επόμενες γενιές. Έτσι, γενιά με την γενιά, ο πληθυσμός βελτιώνεται συνολικά, χωρίς ωστόσο να επικρατούν αποκλειστικά τα καλύτερα άτομα του πληθυσμού, δίνοντας έτσι την ευκαιρία να αναδυθούν και κάποια καλά χαρακτηριστικά κάποιου ατόμου που ίσως να μην ξεχωρίζει τόσο.

Οι γενετικοί αλγόριθμοι είναι μία κατηγορία προγραμμάτων τεχνητής νοημοσύνης και αντλούν την έμπνευσή τους από την θεωρία της εξέλιξης του Δαρβίνου. Επί της ουσίας, αναζητούν λύσεις σε προβλήματα βελτιστοποίησης.

3.1 Ορολογία & Στάδια Γενετικών Αλγορίθμων

Αρχικά, υπάρχει ένα σύνολο από άτομα/οντότητες τα οποία εκπροσωπούν πιθανές λύσεις στο εκάστοτε πρόβλημα. Το εκάστοτε σύνολο ατόμων ονομάζεται **πληθυσμός ή γενιά**. Αυτές οι λύσεις αναφέρονται με τους όρους **άτομα/λύσεις/χρωμοσώματα/γονείς**. Κάθε λύση απαρτίζεται από επιμέρους στοιχεία τα οποία ονομάζονται **γονίδια**.

Το συνολικό πεδίο στο οποίο ο αλγόριθμος αναζητά λύσεις/άτομα ονομάζεται **πεδίο αναζήτησης**.

Επιγραμματικά, τα στάδια που αποτελούν έναν γενετικό αλγόριθμο είναι τα εξής.

Τα παρακάτω στάδια συμβαίνουν **μία** φορά, κατά την εκκίνηση του αλγορίθμου:

- Εισαγωγή η δημιουργία του αρχικού πληθυσμού
- Αξιολόγηση του εκάστοτε ατόμου με την συνάρτηση καταλληλότητας

Τα παρακάτω στάδια επαναλαμβάνονται μέχρις ότου συναντηθεί η **συνθήκη τερματισμού** όπου και σταματά η εκτέλεση του αλγορίθμου:

- Επιλογή των γονέων που θα αναπαραχθούν
- Αναπαραγωγή/ Δημιουργία νέων απογόνων
- Μετάλλαξη/ Mutation
- Εκ νέου αξιολόγηση των απογόνων με την συνάρτηση καταλληλότητας.
- Επιλογή των γονέων που θα συνεχίσουν στην επόμενη γενιά (Steady-State Algorithms).

3.2 Steady-State Vs Generational Genetic Algorithms

Το τελευταίο βήμα συναντάται κυρίως στους steady-state αλγορίθμους. Αυτοί, τείνουν να αντικαθιστούν μόνο μερικά άτομα από τον αρχικό πληθυσμό τους για κάθε νέα γενιά απογόνων και όχι πλήρως [16], όπως συμβαίνει με τους απλούστερους (generational) γενετικούς αλγορίθμους. Στους απλούς γενετικούς αλγορίθμους συνήθως τα άτομα που προκύπτουν από τα στάδια της μετάλλαξης και της αναπαραγωγής αντικαθιστούν εξολοκλήρου την παλαιά γενιά. Συνεπώς, τα όρια και η έννοια των γενεών στους steady-state αλγόριθμους είναι πιο θολά ενώ στους generational πιο ευδιάκριτα. Οι steady-state δεν έχουν τόσο ξεκάθαρες γενιές, αφού όπως αναφέρθηκε για κάθε εκτέλεση του κύκλου εργασιών αντικαθίστανται κάποια και όχι όλα τα άτομα του πληθυσμού.

Η συνθήκη τερματισμού διαφέρει από πρόβλημα σε πρόβλημα. Για παράδειγμα, κάποια προβλήματα μπορεί να έχουν πεπερασμένη, συγκεκριμένη λύση και ο αλγόριθμος σταματά κατά την επίτευξή της. Σε άλλα πάλι, δεν υπάρχει τόσο σαφής, συγκεκριμένη

λύση οπότε ο αλγόριθμος μπορεί να τερματίζει κατά την επίτευξη μίας δεδομένης τιμής καταλληλότητας από έναν γονέα, ή την περάτωση συγκεκριμένου αριθμού γενεών.

3.3 Τελεστές Γενετικών Αλγορίθμων

Παρακάτω παρουσιάζονται αναλυτικότερα μερικές από τις πιο γνωστές τεχνικές κάθε σταδίου των γενετικών αλγορίθμων.

3.3.1 Κωδικοποίηση Ατόμων

Είναι σημαντικό το κάθε άτομο να μπορεί σαφώς να αναλυθεί σε μεμονωμένα γονίδια, ώστε να μπορούν να πραγματωθούν και οι απαραίτητες λειτουργίες πάνω τους κατά τα μετέπειτα στάδια. Οι διάφορες κωδικοποιήσεις των γονιδίων διαφέρουν ανάλογα με το πεδίο του προβλήματος. Οι πιο συνηθισμένες κωδικοποιήσεις [11] είναι η δυαδική (0,1), οκταδική (0-7) και δεκαεξαδική (0-9, A-F). Στις ανωτέρω κωδικοποιήσεις, ο κάθε γονέας αναπαρίσταται ως μία σειρά στοιχείων του εκάστοτε αριθμητικού συστήματος.

Chromosome A	10110010110010101110010 1
Chromosome B	11111110000011000001111 1

Πίνακας 4. Παράδειγμα δυαδικής κωδικοποίησης ατόμου.

Chromosome A	21415405606
Chromosome B	52015554771

Πίνακας 5. Παράδειγμα οκταδικής κωδικοποίησης ατόμου.

Chromosome A	827F749436
Chromosome B	4A31090238

Πίνακας 6. Παράδειγμα δεκαεξαδικής κωδικοποίησης ατόμου.

Άλλη γνωστή κωδικοποίηση είναι αυτή της μετάθεσης (permutation)[19]. Σε αυτήν, τα γονίδια αναπαριστούν διαδικασίες/καταστάσεις και οι γονείς/λύσεις την **σειρά με την οποία αυτές συμβαίνουν**. Συνεπώς, κάθε γονέας/λύση **αναπαριστά μία διαφορετική σειρά των ίδιων διαδικασιών/γονιδίων**. Χρησιμοποιείται κυρίως σε προβλήματα ταξινόμησης. Είναι ιδιαίτερα χρήσιμη για προβλήματα όπως το αυτό του πλανόδιου

πωλητή (Traveling Salesman Problem), τον προγραμματισμό εργασιών και άλλες συνδυαστικές βελτιστοποιήσεις. Κάθε στοιχείο εμφανίζεται ακριβώς μία φορά σε κάθε γονέα. Κατά την αναζήτηση λύσεων, η σειρά των γονιδίων κάθε ατόμου αλλάζει και άρα η κάθε λύση αντανακλά την διαφορετική σειρά με την οποία θα συμβούν οι διαδικασίες. Για παράδειγμα, στο TSP, κάθε γονίδιο μπορεί να αναπαριστά μια πόλη, και η μετάθεση θα αναπαριστούσε την αλλαγή της σειράς με την οποία ο πωλητής επισκέπτεται τις πόλεις.

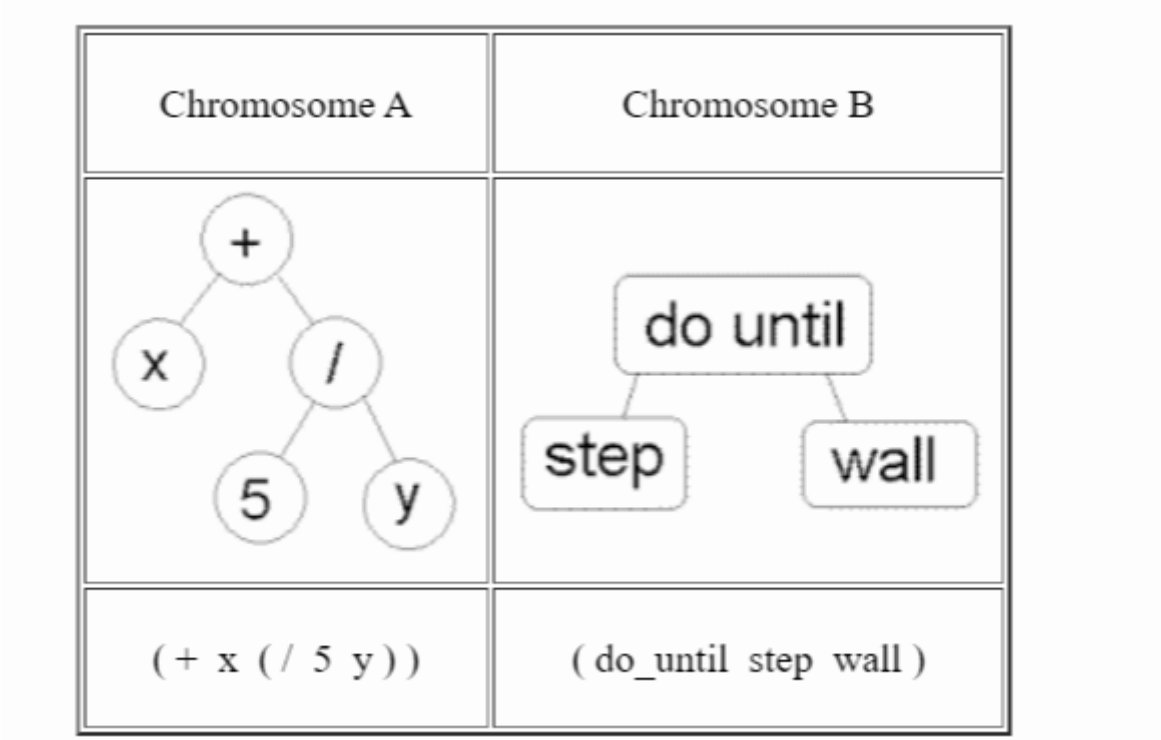
Chromosome A	1 5 3 2 6 4 7 9 8
Chromosome B	8 5 6 7 2 3 1 4 9

Πίνακας 7. Παράδειγμα κωδικοποίησης permutation ατόμου. [20]

Επιπλέον, υπάρχουν και οι κωδικοποιήσεις των τιμών (value-based) [19] και δέντρου [20]. Κατά την κωδικοποίηση τιμών, κάθε χρωμόσωμα είναι μία σειρά τιμών οι οποίες δεν μπορούν να αναπαρασταθούν αποτελεσματικά από τα ανωτέρω αριθμητικά συστήματα, όπως πραγματικοί αριθμοί ή άλλες αυθαίρετες, τιμές.
Στην κωδικοποίηση δέντρου, κάθε χρωμόσωμα συνήθως αποτελείται από ένα δέντρο από functions, εντολές ή λογικές εκφράσεις σε μία οποιαδήποτε προγραμματιστική γλώσσα.

Chromosome A	1.2324 5.3243 0.4556 2.3293 2.4545
Chromosome B	(back), (back), (right), (forward), (left)

Πίνακας 8. Παράδειγμα value-based κωδικοποίησης ατόμου. [20]



Εικόνα 1. Παράδειγμα κωδικοποίησης δέντρου. [20]

3.3.2 Τελεστές Επιλογής & Σύγκλιση

Η διαδικασία της επιλογής είναι από τις πιο σημαντικές σε έναν γενετικό αλγόριθμο. Πριν εισέλθει ο αλγόριθμος σε αυτό το στάδιο, τα άτομα αξιολογούνται βάση κάποιων χαρακτηριστικών και ανάλογα με τις ανάγκες της εκάστοτε εφαρμογής και βάση αυτών, τους προσδίδεται μία τιμή καταλληλότητας. Κατόπιν, λαμβάνει χώρα η επιλογή των ατόμων που θα αναπαραχθούν. Υπάρχουν διάφοροι τρόποι για να γίνει αυτό. Συνήθως, λαμβάνεται υπόψιν η καταλληλότητα των ατόμων, ενώ σημαντικό είναι να διατηρείται και ένα επίπεδο τυχαιότητας. Αυτή η προσέγγιση βοηθά να ευνοηθούν τα καλύτερα άτομα, χωρίς ωστόσο να αποκλείονται από την διαδικασία πιθανά καλά χαρακτηριστικά σε λιγότερο κατάλληλα άτομα.

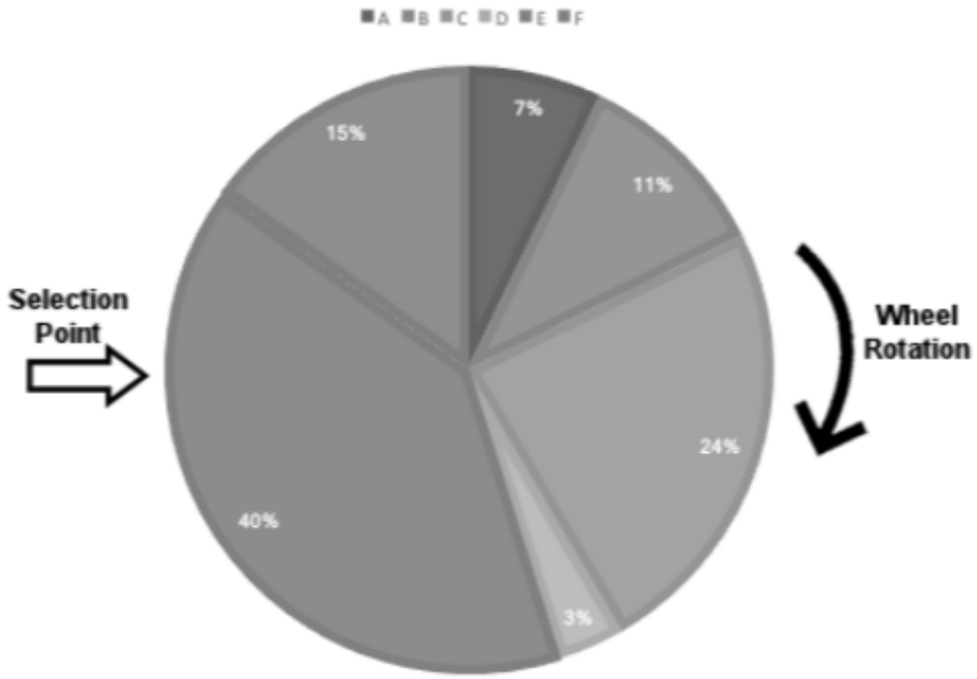
Οι τελεστές επιλογής [13] παίζουν μεγάλο ρόλο στην σύγκλιση του αλγορίθμου. Σύγκλιση [18] ονομάζεται όταν ο αλγόριθμος καταλήγει στην ουσία σε έναν σύνολο ατόμων που θεωρεί την βέλτιστη λύση του προβλήματος. Αν η διαδικασία της επιλογής είναι περιορισμένη (π.χ. μικρός αριθμός γονέων προς επιλογή, έλλειψη πιθανοτικού παράγοντα) τότε το πιθανότερο είναι ο αλγόριθμος να συγκλίνει πρόωρα, σε κάποιο τοπικό μέγιστο (local maximum –relative maximum). Βεβαίως ισχύει και το αντίθετο. Αν

υπάρχει μεγάλος αριθμός γονέων, οι περισσότεροι εξ αυτών έχοντας παρόμοιο βαθμό καταλληλότητας, ή μεγάλος πιθανοτικός παράγοντας, τότε ο αλγόριθμος θα αργήσει να συγκλίνει επειδή θα έχει υπερβολικά ευρύ πεδίο αναζήτησης και θα δεν θα αποκτήσει μία σαφή κατεύθυνση. Μάλιστα, είναι αρκετά πιθανό να εξερευνεί ξανά και ξανά ίδιες λύσεις, μην κάνοντας πρόοδο στην ουσία.

Όπως και στην κωδικοποίηση, υπάρχουν και εδώ διάφορες τεχνικές. Οι πιο γνωστές είναι οι τεχνικές της ρουλέτας, το τουρνουά, η επιλογή βάση κατάταξης (rank-based) και η στοχαστική καθολική δειγματοληψία (stochastic universal sampling).

3.3.3 Ρουλέτα

Η μέθοδος της ρουλέτας αντιστοιχίζει αναλογικά την καταλληλότητα του κάθε γονέα σε μία εικονική ρουλέτα. Όσο μεγαλύτερη η τιμή της καταλληλότητας, τόσο περισσότερο χώρο καταλαμβάνει ο γονέας στην ρουλέτα και άρα τόσο μεγαλύτερες οι πιθανότητές του να επιλεχθεί. Έπειτα παράγει έναν τυχαίο αριθμό μέσα στο πεδίο πιθανοτήτων που ορίζεται. Αυτή η κίνηση αντιστοιχεί στο «γύρισμα» της ρουλέτας. Σε όποιο «κομμάτι» ανήκει ο αριθμός επιλέγεται ο αντίστοιχος γονέας.



Εικόνα 2. Παράδειγμα ρουλέτας. [12]

3.3.4 Επιλογή Κατατάξεων (Rank-based)

Σε αυτήν την μέθοδο, τα άτομα του πληθυσμού ταξινομούνται με βάση την τιμή της καταλληλότητάς τους και τους ανατίθεται ένας αριθμός, ανάλογα με την κατάταξή τους. Ο τρόπος με τον οποίο επιλέγονται είναι παρόμοιος με την ρουλέτα, μόνο που χαρτογραφούνται πάνω σε αυτήν βάση του αριθμού της κατάταξής τους πλέον και όχι της τιμής καταλληλότητας. Αυτή η προσέγγιση εξυπηρετεί ώστε άτομα με πολύ μεγαλύτερη καταλληλότητα να μην καταλαμβάνουν υπερβολικά παραπάνω χώρο στην ρουλέτα σε σχέση με άλλα που έχουν μικρότερη τιμή καταλληλότητας. Έτσι, αυτά τα άτομα εξακολουθούν να έχουν μεγαλύτερες πιθανότητες επιλογής από τα άλλα. Όμως επειδή πλέον οι διαφορές στην καταλλήλότητά τους δεν είναι ποσοστιαίες (αφού το κάθε άτομο έχει μόλις 1 βαθμό μεγαλύτερο από το προηγούμενο) έχουν καλύτερες πιθανότητες να επιλεγούν και άτομα με μικρότερη καταληλλότητα. Συνεπώς, αυτή η προσέγγιση ευνοεί την επιλογή διαφορετικών ατόμων κατά την αναπαραγωγή. Δουλεύει υπέρ των ατόμων με μικρή καταλληλότητα και εξασφαλίζει σε μεγαλύτερο βαθμό ότι τα άτομα με υψηλή καταλληλότητα δεν μονοπωλούν την διαδικασία της αναπαραγωγής.

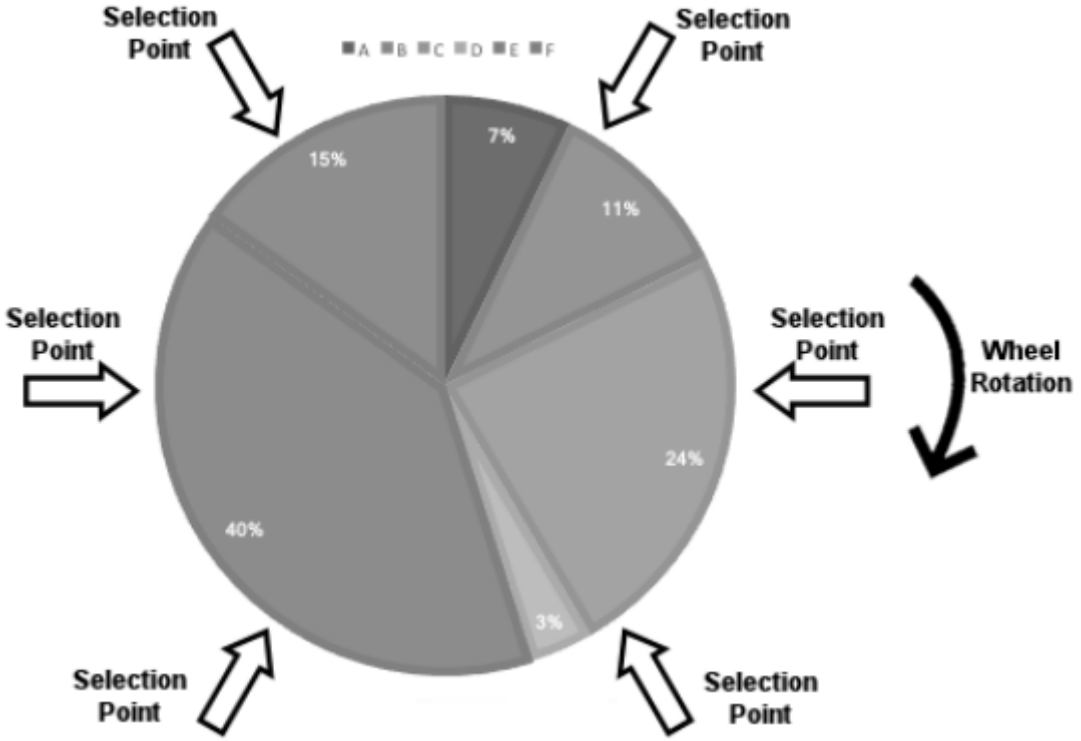
3.3.5 Τουρνουά

Η μέθοδος του τουρνουά προτάθηκε αρχικά από τον Bridle 1983. Σε αυτήν, δίνεται ένας προκαθορισμένος αριθμός, ο οποίος ονομάζεται μέγεθος τουρνουά. Επιλέγονται τυχαία άτομα από τον εκάστοτε πληθυσμό τόσα όσο είναι το μέγεθος τουρνουά. Η καταλληλότητα των επιλεγμένων ατόμων συγκρίνεται και αναδεικνύεται νικητής το άτομο με την μεγαλύτερη καταλληλότητα. Η διαδικασία επαναλαμβάνεται για όλους μέχρι να συγκριθούν όλα τα άτομα του πληθυσμού. Τα άτομα που επιλέγονται περνούν στην επόμενη γενιά. Άξιο αναφοράς είναι το γεγονός πως όσο μικρότερος ο αριθμός τουρνουά, τόσο περισσότερες πιθανότητες υπάρχουν να επιλεχθούν άτομα με μικρή καταλληλότητα.

3.3.6 Στοχαστική Καθολική Δειγματοληψία (Stochastic Universal Sampling)

Και αυτή η τεχνική βασίζεται στην ρουλέτα. Ωστόσο, η διαφορά της είναι ότι δεν υπάρχει πολλαπλή εκτέλεση της ρουλέτας ώστε να επιλεχθεί ξεχωριστά ο κάθε γονέας. Αντιθέτως, προεπιλέγεται δεδομένος αριθμός σημείων, ισόποσα μοιρασμένος στην

ρουλέτα, και αυτή «γυρνάει» μία μόνο φορά στην οποία και επιλέγονται όλοι οι γονείς ταυτόχρονα. Το κάθε σημείο επιλογής τυγχάνει μέσα στο πιθανοτικό πεδίο ενός γονέα και έτσι επιλέγονται οι γονείς.

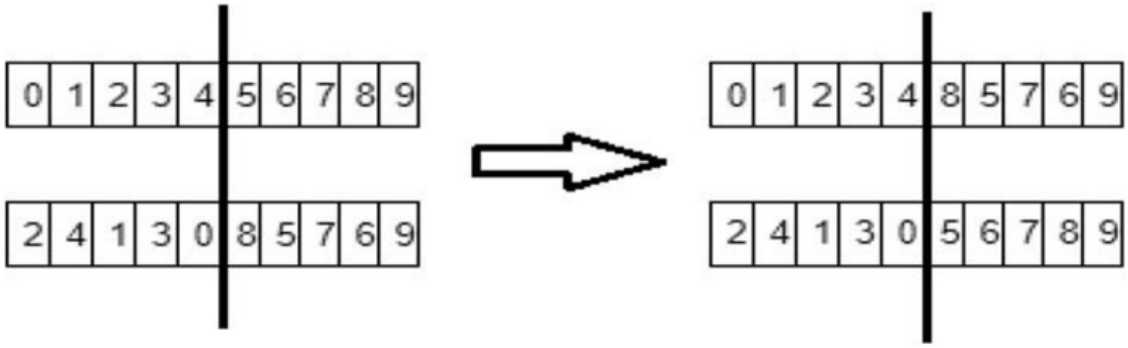


Εικόνα 3. Παράδειγμα στοχαστικής καθολικής δειγματοληψίας. [12]

3.3.7 Crossover (Αναπαραγωγή)

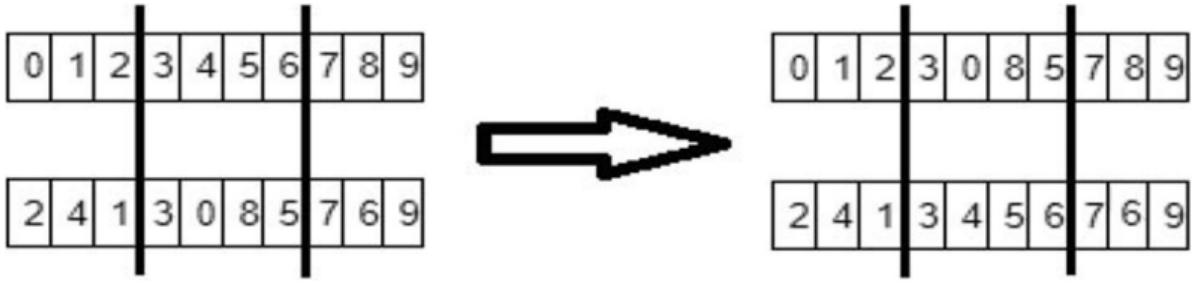
Τα επιλεγμένα άτομα από το προηγούμενο βήμα εισέρχονται στην διαδικασία της αναπαραγωγής. Εδώ τα γονίδια των επιλεγμένων λύσεων αναμιγνύονται ώστε να προκύψουν νέες λύσεις. Οι πιο γνωστές τεχνικές για αναπαραγωγή [12, 14] είναι οι single-point, two-point, k-point και uniform crossover.

Στην single-point crossover επιλέγεται ένα τυχαίο σημείο στους γονείς που προκύπτουν από το προηγούμενο στάδιο. Στην συνέχεια, χωρίζει τον κάθε γονέα στα δύο βάση αυτού του σημείου. Έτσι, για δύο γονείς, προκύπτουν τέσσερις διαφορετικές ακολουθίες γονιδίων, δύο από τον κάθε γονέα. Τέλος, ανταλλάζει ένα κομμάτι του πρώτου γονιού με ένα του δεύτερου ώστε να προκύπτουν 2 τελείως νέοι απόγονοι. Φυσικά, αυτό μπορεί να γίνει και με παραπάνω από δύο γονείς.



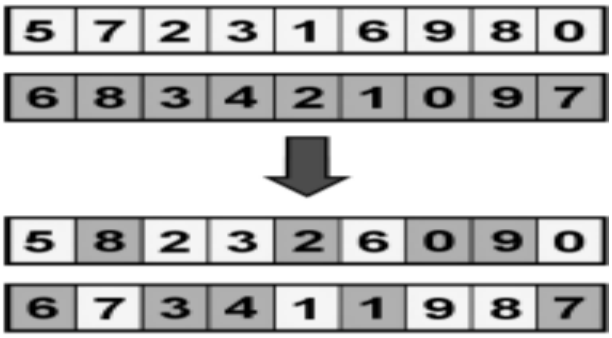
Εικόνα 4. Παράδειγμα single-point crossover. [11]

Η two-point crossover ακολουθεί την ίδια λογική, όμως τα γονίδια κάθε γονέα χωρίζονται σε δύο σημεία. Ομοίως, η k-point crossover είναι η υλοποίηση της παραπάνω μεθοδολογίας για k σημεία.



Εικόνα 5. Παράδειγμα two-point crossover. [11]

Στην uniform crossover, ο απόγονος δημιουργείται επιλέγοντας τυχαία γονίδια από τους γονείς για την κάθε θέση. Μπορεί να διατηρηθεί ένα ποσοστό διανομής γονιδίων 50% από τον κάθε γονέα ώστε ο καθένας να επηρεάζει εξίσου τους απογόνους.

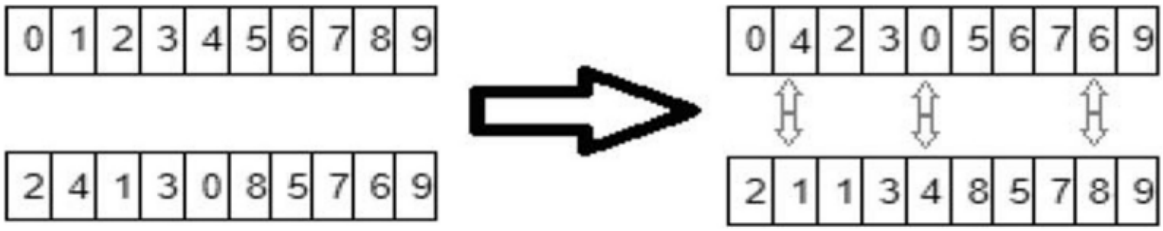


Εικόνα 6. Παράδειγμα uniform crossover. [12]

3.3.8 Μετάλλαξη (Mutation)

Στην συνέχεια, λαμβάνει χώρα η μετάλλαξη. Η γενικότερη ιδέα εδώ είναι πως κάποια γονίδια των απογόνων τροποποιούνται με μία μικρή πιθανότητα, τυχαία. Η αλλαγή που συμβαίνει θα πρέπει να είναι πολύ μικρή. Αυτή η τεχνική μιμείται τις τυχαίες μικρές μεταλλάξεις που συμβαίνουν σε κάποιους από τους απογόνους του πληθυσμού. Αυτός ο μηχανισμός βοηθά στην διατήρηση της ποικιλομορφίας του πληθυσμού καθώς και την εξερεύνηση νέων επιλογών στο πεδίο αναζήτησης.

Οι πιο γνωστές τεχνικές μετάλλαξης [12, 13] είναι η αντιμετάθεση, στην οποία αλλάζουν θέσεις μεταξύ τους 2 ή περισσότερα γονίδια (swap), η αντιστροφή bit (flip bit) (όταν οι γονείς αποτελούνται από ακολουθίες bits), η τυχαία ανακατάταξη των γονιδίων σε ένα υποσύνολο αυτών του ατόμου με οποιαδήποτε σειρά (scramble) και η αντιστροφή της σειράς των γονιδίων σε ένα υποσύνολο αυτών του ατόμου (inversion). Τα παραπάνω μπορούν να εφαρμοστούν σε οποιονδήποτε βαθμό. Δηλαδή, στις περιπτώσεις της αντιμετάθεσης και της ανακατάταξης γονιδίων, οι θέσεις που επιλέγονται μπορούν να είναι τυχαίες και το πλήθος των γονιδίων στο οποίο δρα ο μηχανισμός οποιοδήποτε εφόσον εξυπηρετεί τον σκοπό.



Εικόνα 7. Παράδειγμα μετάλλαξης με αντιμετάθεση γονιδίων (swap). [11]

Εικόνα 8. Παράδειγμα flip-bit mutation. [12]

Εικόνα 9. Παράδειγμα scramble mutation. [12]

Εικόνα 10. Παράδειγμα inversion mutation. [12]

3.3.9 Επιλογή Ατόμων (Προαιρετική – Steady-State Genetic Algorithms)

Υπάρχουν κάποιοι σχετικοί μηχανισμοί [16,17] οι οποίοι με συγκεκριμένα κριτήρια αποφασίζουν ποιος από τους απογόνους και ποιος από τους γονείς θα προαχθεί στην επόμενη γενιά. Μερικοί από αυτούς είναι η αντικατάσταση των χειρότερων ατόμων του τρέχοντα πληθυσμού (Replace Worst), η αντικατάσταση τυχαίων ατόμων (Replace Random) καθώς και ένας τύπος τουρνουά (όπως και κατά την επιλογή γονέων) μεταξύ γονέων και απογόνων όπου οι νικητές αντικαθιστούν τους χαμένους. Άλλη μία ενδιαφέρουσα τεχνική αξιοποιεί την χρήση ηλικίας, όπου τα άτομα έχουν μία «ηλικία» η οποία αυξάνει με το πέρασμα των γενεών. Τα γηραιότερα άτομα είναι πιο πιθανό να αντικατασταθούν.

3.3.10 Ελιτισμός

Τέλος, άλλη μία τεχνική είναι ο ελιτισμός [13, 15]. Ο ελιτισμός εφαρμόζεται προαιρετικά και στα δύο προαναφερθέντα είδη αλγορίθμων. Κατά την χρήση αυτής της τεχνικής, ένα μικρό ποσοστό ατόμων θεωρείται elites λόγω των πολύ καλών χαρακτηριστικών τους. Αυτά τα άτομα δεν αντικαθίστανται από άλλα και είναι παρόντα σε κάθε γενιά. Είναι σημαντικό τα άτομα που θεωρούνται elites να καταλαμβάνουν μικρό ποσοστό του πληθυσμού ώστε να αποφεύγεται η συνεχόμενη επιλογή τους κατά το αντίστοιχο στάδιο της επιλογής και να μην μονοπωλούν τον πληθυσμό.

3.4 Συμπεράσματα

Με μία ματιά στα παραπάνω, μπορεί κανείς να συμπεράνει πως οι γενετικοί αλγόριθμοι είναι μία σειρά διαδικασιών που «ισοδυναμούν» κατά μία έννοια με μία ένωση τελεστών, όπως ακριβώς και σε μία μαθηματική παράσταση. Υπάρχουν πολλοί τρόποι να υλοποιηθεί η κάθε λειτουργία τους και, όπως είναι προφανές, αρκετά κομμάτια μπορούν να χρησιμοποιηθούν ή να παραλειφθούν. Αυτό σημαίνει ότι υπάρχουν αρκετές επιλογές και παραλλαγές κατά την δημιουργία ενός γενετικού αλγορίθμου.

Κεφάλαιο 4
Χρήση Γενετικών Αλγορίθμων Για Παραγωγή Κρυπτογραφικών Λογικών Συναρτήσεων

Στον τομέα της κρυπτογραφίας χρησιμοποιούνται ήδη ευρέως εξελικτικοί αλγόριθμοι για εφαρμογές όπως εξέλιξη Boolean functions [23], S-boxes [29], επιθέσεις σε PUFs (Physical Unclonable Functions)[22], εντοπισμό Trojan στο hardware[37] και επιθέσεις παράπλευρου καναλιού [38]. Μάλιστα, οι εφαρμογές για εξέλιξη Boolean functions είναι από τους πιο επιτυχημένους τομείς.

Αναφορικά με την δημιουργία λογικών συναρτήσεων με ισχυρές κρυπτογραφικές ιδιότητες, η χρήση εξελικτικών αλγορίθμων έχει μια ιστορία διάρκειας 25 ετών. Όσον αφορά την πρώτη προσπάθεια γνωρίζουμε ότι πραγματοποιήθηκε το 1997. Σε αυτήν την αρχική προσπάθεια, οι συγγραφείς αξιοποίησαν γενετικούς αλγόριθμους για τη βελτίωση λογικών συναρτήσεων με υψηλή μη γραμμικότητα [31]. Στη συνέχεια, διάφοροι αλγόριθμοι υποβλήθηκαν σε δοκιμές προκειμένου να επιτευχθούν ακόμη καλύτερα αποτελέσματα. Για παράδειγμα, άλλοι ερευνητές(Millan κ.α.) συνδύασαν γενετικούς αλγόριθμους με τεχνικές hill-climbing και resetting step για τη διαμόρφωση λογικών συναρτήσεων έως 12 εισόδων [32]. Από την άλλη πλευρά, οι ερευνητές Clark και Jacob χρησιμοποίησαν την simulated annealing και την hill-climbing με μια συνάρτηση κόστους εμπνευσμένη από το θεώρημα Parseval για την ανεύρεση λογικών συναρτήσεων Boolean με υψηλή μη γραμμικότητα και χαμηλή αυτοσυσχέτιση [24]. Άλλοι ερευνητές

(Aguirre κ.α.) ήταν οι πρώτοι που εξέτασαν την βελτιστοποίηση με χρήση πολλαπλών στόχων για αυτό το πρόβλημα [21]. Οι συγγραφείς χρησιμοποίησαν έναν random bit climber για την εύρεση ισορροπημένων, υψηλά μη γραμμικών λογικών συναρτήσεων Boolean.

Οι Picek κ.α. εξέτασαν διάφορους εξελικτικούς αλγόριθμους (γενετικούς αλγόριθμους και γενετικό προγραμματισμό) για την εύρεση λογικών συναρτήσεων Boolean που πληρούν πολλαπλές κρυπτογραφικές ιδιότητες [35]. Οι Mariot και Leporati πραγματοποίησαν πειράματα με την τεχνική Particle Swarm Optimization (PSO) [26] για την ανεύρεση λογικών συναρτήσεων Boolean με καλή εξισορρόπηση μεταξύ διαφόρων κρυπτογραφικών ιδιοτήτων για συναρτήσεις από 7 έως 12 εισόδων [28]. Σε αυτήν την προσέγγιση, ο προτεινόμενος αλγόριθμος ενημερώνει τις θέσεις των σωματιδίων διατηρώντας τα βάρη Hamming, για να διασφαλίσει ότι οι προκύπτουσες συναρτήσεις είναι ισορροπημένες, και υιοθετεί τη μέθοδο Hill Climbing για να βελτιώσει περαιτέρω τη μη γραμμικότητά τους και την ανθεκτικότητα σε συσχετίσεις. Τα αποτελέσματά τους δείχνουν ότι αυτός ο νέος αλγόριθμος PSO βρίσκει συναρτήσεις με καλή εξισορρόπηση μεταξύ μη γραμμικότητας, ανθεκτικότητας σε συχετίσεις και ενός κρυπτογραφικού κριτηρίου που ονομάζεται «αυστηρό κριτήριο χιονοστιβάδας». (Strict Avalanche Criterion). Επιπλέον, οι Picek κ.α. διερεύνησαν την χρήση των ανοσολογικών (immunological) αλγορίθμων CLONALG και opt-IA για την εξέλιξη συναρτήσεων Boolean με υψηλή μη γραμμικότητα έως 16 εισόδων [36]. Τους συνέκριναν με δύο εξελικτικούς αλγορίθμους, τους γενετικούς και τους αλγόριθμους στρατηγικής εξέλιξης (evolution strategy). Χρησιμοποίησαν 4 διαφορετικές fitness functions, ποικίλων κριτηρίων, καταλήγοντας ότι σε γενικές γραμμές καλύτερη απόδοση επιτυγχάνεται με εξελικτικούς αλγορίθμους. Οι Clark κ.α.[25] πρωτοπόρησαν στη μέθοδο του «spectral inversion», εργαζόμενοι πάνω στο λεγόμενο Walsh μετασχηματισμό (φάσμα Walsh) των λογικών συναρτήσεων: ο στόχος ήταν, με αφετηρία μετασχηματισμούς Walsh, να βρεθεί ένας άλλος μετασχηματισμός Walsh ο οποίος να αντιστοιχεί σε πραγματική λογική συνάρτηση. Οι Mariot και Leporati [27] εξέτασαν περαιτέρω αυτήν την προσέγγιση προτείνοντας ένα γενετικό αλγόριθμο για την εξέλιξη των μετασχηματισμών. Ο αλγόριθμος που ανέπτυξαν κωδικοποιεί το χρωμόσωμα μιας υποψήφιας λύσης ως ένα permutation (περιστροφή) τριών τιμών του φάσματος Walsh. Επιπλέον, σχεδίασαν εξειδικευμένους τελεστές διασταύρωσης και μετάλλαξης, έτσι ώστε οι ανταλλαγές

θέσεων στα χρωμοσώματα των απογόνων να αντιστοιχούν σε διάφορες τιμές στα προκύπτοντα φάσματα Walsh. Ορισμένες δοκιμές που πραγματοποιήθηκαν στο σύνολο των ψευδο-Boolean συναρτήσεων με n = 6 και n = 7 μεταβλητές δείχνουν ότι στην πρώτη περίπτωση ο γενετικός αλγόριθμός τους υπερτερεί σε σχέση με τον αλγόριθμο προσομοιωμένης ανάβασης (simulated annealing) του Clark κ.α. όσον αφορά τον λόγο των παραγόμενων λογικών συναρτήσεων ανά αριθμό εκτελέσεων βελτιστοποίησης.

Στο [33], οι συγγραφείς στοχεύουν στην εύρεση bent συναρτήσεων για ένα μεγαλύτερο αριθμό μεταβλητών. Αντί να εξελίξουν απευθείας bent συναρτήσεις, όπως στο [40], οι συγγραφείς χρησιμοποίησαν τον γενετικό προγραμματισμό (GP) για να εξελίξουν μια δευτερογενή κατασκευή που μετατρέπει διάφορες bent συναρτήσεις εισόδου με n - 2 μεταβλητές σε μια bent συνάρτηση εξόδου με n μεταβλητές.. Χρησιμοποιώντας σαν αφετηρία bent συναρτήσεις μόνο με 4 μεταβλητές, οι συγγραφείς δημιούργησαν δευτερογενείς κατασκευές που παράγουν bent συναρτήσεις σε υψηλές διαστάσεις. Εκτός από αυτό, φαίνεται ότι οι εξελιγμένες κατασκευές είναι αρκετά γενικές, καθώς οι ίδιες κατασκευές κατάφεραν να δημιουργήσουν bent συναρτήσεις από 6 μέχρι 24 μεταβλητές με διαφορετικές συναρτήσεις εκκίνησης (με δύο μεταβλητές λιγότερες). Αυτό που παραμένει ασαφές από τα αποτελέσματα που προέκυψαν είναι εάν κάποιες από τις δημιουργηθείσες κατασκευές συναρτήσεων bent είναι πραγματικά νέες. Πράγματι, δεδομένου ότι οι μετα-ευρηστικοί αλγόριθμοι βρίσκουν πολλές λύσεις, θα πρέπει να ελεγχθούν όλες αυτές για να διαπιστωθεί εάν υπάρχουν νέες κατασκευές. Επιπλέον, καθώς δεν είναι δυνατό να γνωρίζουμε το μέγεθος της κατασκευής που θα προκύψει, αυτές συχνά παρουσιάζουν σημαντική αύξηση του μεγέθους που πρέπει να αναλυθεί και να αφαιρεθεί.

Μια παρόμοια προσέγγιση χρησιμοποιείται στο [39], αλλά αυτή τη φορά για την εύρεση ισορροπημένων συναρτήσεων με υψηλή μη γραμμικότητα. Η προσέγγιση αυτή βασίζεται και πάλι στην εύρεση δευτερογενών κατασκευών, οι οποίες και πάλι εξελίσσονται με τον γενετικό προγραμματισμό. Αυτή η προσέγγιση θεωρήθηκε, από τους εμπνευστές της, ενδιαφέρουσα, αφού υπήρχαν ελάχιστες τέτοιες γνωστές κατασκευές. Τα αποτελέσματα τους δείχνουν ότι ο γενετικός προγραμματισμός μπορεί να βρει κατασκευές που τείνουν να γενικεύουν καλά, δηλαδή οδηγούν σε ισορροπημένες υψηλά-μη-γραμμικές συναρτήσεις για διάφορα δοκιμασμένα μεγέθη και διάφορες ομάδες λογικών

συναρτήσεων εισόδου. Ενώ τα επίπεδα μη γραμμικότητας που επιτυγχάνονται σπανίως φτάνουν τις καλύτερες γνωστές τιμές, η απλούστερη λύση που βρέθηκε από τον γενετικό προγραμματισμό φαίνεται να αποτελεί μία συγκεκριμένη περίπτωση της γνωστής κατασκευής έμμεσης αθροιστικής(indirect sum construction).

Τα παραπάνω επιστημονικά άρθρα δεν αποτελούν εξαντλητική αναφορά στην έρευνα που έχει διεξαχθεί. Καταφέρνουν όμως να δείξουν πώς τα περισσότερα από αυτά τα έργα κατόρθωσαν να ανακαλύψουν λογικές συναρτήσεις Boolean υψηλής καταλληλότητας, ανεξάρτητα από τις εκάστοτε συγκεκριμένες απαιτούμενες ιδιότητες. Ωστόσο, παρέμεινε το πρόβλημα της χρήσης τέτοιων υπολογιστικά απαιτητικών προσεγγίσεων για λογικές συναρτήσεις με περισσότερες εισόδους ή κρυπτογραφικές ιδιότητες των οποίων ο υπολογισμός είναι υπολογιστικά ακριβός. Επιπλέον, αυτές οι προσεγγίσεις οδήγησαν στη δημιουργία συγκεκριμένων περιπτώσεων λογικών συναρτήσεων, απαιτώντας νέες αναζητήσεις για κάθε νέο απαιτούμενο μέγεθος.

Οι Picek και Jakobovic εξέτασαν μια προσέγγιση όπου, αντί να εξελίσσουν άμεσα τις λογικές συναρτήσεις, εξελίσσουν κατασκευές που οδηγούν σε λογικές συναρτήσεις με τις επιθυμητές ιδιότητες [33].Χρησιμοποίησαν γενετικό προγραμματισμό για την ανάπτυξη δευτερογενών αλγεβρικών κατασκευών που παράγουν bent Boolean functions (δηλαδή, μη ισορροπημένες αλλά μέγιστα μη γραμμικές). Οι Carlet κ.α. χρησιμοποίησαν γενετικό προγραμματισμό για τη βελτίωση λογικών συναρτήσεων που προέρχονται από αλγεβρικές κατασκευές [23]. Αυτή η προσέγγιση οδήγησε σε λογικές συναρτήσεις με υψηλότερη μη γραμμικότητα από τις μέχρι τώρα γνωστές. Τέλος, οι Mariot κ.α. [30] διερεύνησαν μια δευτερογενή κατασκευή βασισμένη σε πεπερασμένα αυτόματα(cellular automata CA), χρησιμοποιώντας εξελικτικές στρατηγικές για την αναζήτηση τοπικών κανόνων CA που οδηγούν σε bent και περίπου bent λογικές συναρτήσεις όταν ενσωματώνονται στην κατασκευή.

Για μια πιο λεπτομερή επισκόπηση των εξελικτικών αλγορίθμων και των λογικών συναρτήσεων στο πλαίσιο της κρυπτογραφίας, μπορεί κανείς να δει το [34].

Όλες οι μέχρι τώρα έρευνες δεν βασίζονται στο ότι οι αρχικές συναρτήσεις που θα τεθούν στην είσοδο του γενετικού αλγορίθμου θα πληρούν συγκεκριμένα κρυπτογραφικά

κριτήρια (με εξαίρεση κάποιες λίγες ειδικές περιπτώσεις, όπως αυτές που τέθηκαν συναρτήσεις bent στην είσοδο, με απώτερο σκοπό όμως να παραχθούν συναρτήσεις bent στην έξοδο). Επίσης, δεν φάνηκε να εξετάζεται η χρήση γενετικών αλγορίθμων για δημιουργία συναρτήσεων με υψηλή αλγεβρική ανθεκτικότητα. Οι εν λόγω προσεγγίσεις είναι και αυτές που διερευνά η παρούσα διατριβή.

Κεφάλαιο 5
Ανάλυση Λογισμικού

Για την εκπόνηση της παρούσας έρευνας, αναπτύχθηκε εξ αρχής – όπως προαναφέρθηκε – κατάλληλο πρόγραμμα λογισμικού. Το πρόγραμμα αναπτύχθηκε σε python 3.11.2 χρησιμοποιώντας σαν editor το notepad++ v8.3.3. Το λειτουργικό σύστημα στο οποίο έγιναν οι εκτελέσεις είναι Kali Linux 6.1.0-kali7-amd64 σε VMware Workstation 16 Player 16.2.4 build-20089737. Ο υπολογισμός των χαρακτηριστικών των συναρτήσεων έγινε με την χρήση του περιβάλλοντος sagemath 9.5. Οι πόροι που η μηχανή kali είχε στην διάθεσή της είναι 4 πυρήνες του επεξεργαστή AMD FX 8350 4 Ghz και 12 GB DDR3 RAM.

5.1 Αρχιτεκτονική προγράμματος

Το πρόγραμμα αναπτύχθηκε σαν ένα σύνολο βιβλιοθηκών και διαδικασιών που συνδυάζονται και συνδράμουν στην δημιουργία του τελικού λογισμικού. Η ανάπτυξή τους έγινε με την λογική να διατηρούνται όσο γίνεται ανεξάρτητα και αποσυνδεδεμένα μεταξύ τους τα εκάστοτε στοιχεία τους (decoupled). Επιπρόσθετα, δεν χρησιμοποιήθηκε κάποια έτοιμη βιβλιοθήκη γενετικών αλγορίθμων. Η ανωτέρω προσέγγιση έγινε με στόχο την συνολική ευελιξία του προγράμματος, την διαχειρισιμότητα του κώδικα καθώς και την αποφυγή πιθανών δεσμεύσεων που θα επέβαλλε η ανάπτυξη ενός ενιαίου, άκαμπτου κορμού κώδικα ή η χρήση τρίτων βιβλιοθηκών.

Κάθε ξεκάθαρη, μικρή, λογική διαδικασία υλοποιείται σαν ένα μεμονωμένο function. Κάθε στάδιο/βήμα του γενετικού αλγορίθμου επίσης υλοποιείται σαν ένα function. Κάθε function, ανήκει σε μία αντίστοιχη βιβλιοθήκη. Αυτές οι διαδικασίες συνεργάζονται και συνδυάζονται μεταξύ τους, επιτυγχάνοντας τις πιο σύνθετες λειτουργίες. Έτσι, σταδιακά, επιτυγχάνεται το τελικό αποτέλεσμα. Ακόμα, αξίζει να τονιστεί πως ακριβώς επειδή η κατεύθυνση είναι η ανάπτυξη μικρών μεμονωμένων κομματιών, δύναται αυτά τα κομμάτια να συνδυαστούν με περισσότερους από έναν τρόπους οι οποίοι προφανώς

και οδηγούν στην δυνατότητα δημιουργίας περισσότερων από ένα τελικών προϊόντων/αλγορίθμων. Άρα, λοιπόν, το εκάστοτε κύριο/τελικό πρόγραμμα δεν είναι παρά ένα «μωσαϊκό» των μικρότερων διαδικασιών που αναπτύχθηκαν.

Οι βιβλιοθήκες που αναπτύχθηκαν και χρησιμοποιούνται είναι οι εξής:

- cfeng.py - βιβλιοθήκη για την δημιουργία συναρτήσεων που μοιάζουν με Carlet-Feng, δεδομένης μίας αρχικής Carlet-Feng συνάρτησης.
- genesys_sage.py - βιβλιοθήκη για την δημιουργία τυχαίων κρυπτογραφικών συναρτήσεων.
- darwin.py - βιβλιοθήκη η οποία περιέχει τις υλοποιημένες λειτουργίες του κάθε σταδίου ενός γενετικού αλγορίθμου.
- bent.sage - βιβλιοθήκη για την δημιουργία συναρτήσεων bent.

Η δημιουργία του προγράμματος και των βιβλιοθηκών είχε τις εξής κατευθύνσεις:

- την δυνατότητα κλιμάκωσης και εύκολης παραμετροποίησης - το πρόγραμμα μπορεί να παράξει εύκολα διαφορετικό αριθμό γονέων, να επιλέξει ποικίλους γονείς για το crossover και να το εφαρμόσει με ποικίλους παραμέτρους, να λειτουργήσει με διαφορετικά ποσοστά μετάλλαξης καθώς και διαφορετικό ποσοστό γονέων που διατηρούνται στην επόμενη γενιά. Όλα αυτά είναι εύκολα ελέγξιμα κατά την κλήση του αντίστοιχου function. Αρκεί η αλλαγή ενός και μόνο αριθμού στις παραμέτρους του εκάστοτε function, ώστε να διαφοροποιηθεί η τελική συνολική διαδικασία. Για παράδειγμα, τα functions θα λειτουργήσουν το ίδιο είτε δοθούν 10 είτε 100 συναρτήσεις/γονείς, είτε επιλεχθούν 10 είτε 100 γονείς, είτε το ποσοστό μετάλλαξης είναι 5% είτε 50% κλπ.
- την ευελιξία – με την απομόνωση την διαδικασιών σε functions και βιβλιοθήκες, το αποτέλεσμα είναι το τελικό πρόγραμμα να αποτελείται στην πλειοψηφία του από κλήσεις σε functions. Επομένως, υπάρχει η δυνατότητα στο κυρίως πρόγραμμα να γίνουν εύκολα και γρήγορα αλλαγές, ακόμα και ριζικές, απλώς αντικαθιστώντας/προσθέτοντας/αφαιρώντας την αντίστοιχη επιθυμητή function.

- Την δυνατότητα επαναχρησιμοποίησης - όλες οι βιβλιοθήκες μπορούν να επαναχρησιμοποιηθούν και για άλλους σκοπούς/προγράμματα. Κυρίως η darwin.py, περιλαμβάνοντας υλοποιήσεις από όλα τα στάδια ενός γενετικού αλγορίθμου, μπορεί να χρησιμοποιηθεί σε έναν γενετικό αλγόριθμο εντελώς άσχετο από αυτόν της μεταπτυχιακής διατριβής με μικρές τροποποιήσεις/περιορισμούς. Βεβαίως, και οι cfeng.py, bent.py και genesys.py μπορούν να χρησιμοποιηθούν μεμονωμένα για την παραγωγή των αντίστοιχων συναρτήσεων.
- Την διαχειρισιμότητα του κώδικα - καθώς το πρόγραμμα και οι βιβλιοθήκες αυξάνουν σε γραμμές, γίνεται όλο και πιο δύσκολη η διαχείρισή τους. Πολλές φορές, ακόμα και μια μικρή αλλαγή σε ένα σημείο, σημαίνει ότι πρέπει κανείς να γυρίσει πίσω και να τροποποιήσει παλαιότερα κομμάτια. Η διάσπαση του κώδικα σε functions και βιβλιοθήκες επιτρέπει την καλύτερη διαχείριση του. Ακόμα, συνεισφέρει στην μείωση της ανάγκης για πολλαπλές ανασκοπήσεις καθώς και στην μείωση των πιθανοτήτων να χαλάσει κάτι μετά από ενδεχόμενες αλλαγές/διορθώσεις.

Για κάθε λειτουργία στην διαδικασία των γενετικών αλγορίθμων υπάρχουν διαφορετικές τεχνικές με τις οποίες μπορεί αυτή να πραγματωθεί. Στην παρούσα μεταπτυχιακή διατριβή δεν έχουν υλοποιηθεί όλες οι υπάρχουσες τεχνικές για την κάθε λειτουργία. Ωστόσο, για κάποιες από αυτές, έχουν υλοποιηθεί παραπάνω από μία παραλλαγές στην χρησιμοποιούμενη τεχνική. Παρακάτω αναλύεται ποιες από αυτές υλοποιήθηκαν και πώς λειτουργούν.

5.2 Functions – Κομμάτια κώδικα

Παρακάτω αναλύονται οι επιμέρους βιβλιοθήκες του συνολικού προγράμματος και περιγράφεται επί της ουσίας η λειτουργία του συνολικού προγράμματος.

5.2.1 genesys_sage.py

Η genesis_sage.py, όπως αναφέρθηκε και παραπάνω είναι η βιβλιοθήκη η οποία δημιουργεί τυχαίες κρυπτογραφικές συναρτήσεις δεδομένου μήκους. Αρχικοποιείται μία συγκεκριμένη class η οποία χρησιμοποιείται για την κατασκευή της εκάστοτε κρυπτογραφικής συνάρτησης και των χαρακτηριστικών της. Κάθε συνάρτηση και τα χαρακτηριστικά αυτής, λογίζονται ως object το οποίο κατασκευάζεται από αυτήν την class.

Στην συνέχεια, δημιουργείται μια λίστα με προκαθορισμένο μήκος από τυχαία bits. Η λίστα αυτή εξυπηρετεί στο να περιγράψει τον πίνακα αλήθειας της συνάρτησης. Για την δημιουργία αυτής της λίστας χρησιμοποιείται η βιβλιοθήκη random. Έπειτα, η προκύπτουσα λίστα τροφοδοτείται σαν συνάρτηση στο λογισμικό SageMath μέσω του function objectify. Εκεί, υπολογίζονται τα χαρακτηριστικά της συνάρτησης όπως αλγεβρικός βαθμός, αλγεβρική ανθεκτικότητα, μη γραμμικότητα και αν η συνάρτηση είναι ισορροπημένη. Αφού υπολογιστούν αυτά τα χαρακτηριστικά, υπολογίζεται και η καταλληλότητα (fitness) της συνάρτησης μέσω του fitness_function. Η συνάρτηση αξιολόγησης έχει γραφτεί ως weighted πολλαπλασιασμός των balancedness και nonlinearity του κάθε γονέα. Όλα τα ανωτέρω δημιουργούν ένα object το οποίο λαμβάνει στα χαρακτηριστικά του τις τιμές που υπολογίστηκαν από το sage.

Η παραπάνω διαδικασία επαναλαμβάνεται όσες φορές ορισθεί κατά την εκτέλεση του προγράμματος και τα άτομα προστίθενται σε μία λίστα. Έτσι, λοιπόν, με αυτόν τον τρόπο δημιουργείται ο αρχικός πληθυσμός ο οποίος τροφοδοτεί το κυρίως πρόγραμμα. Όλα τα παραπάνω στο κυρίως πρόγραμμα εκτελούνται με μία και μόνο εντολή.

Αξίζει να αναφερθεί πως το function objectify που περιγράφεται παραπάνω, συμμετέχει και σε άλλες βιβλιοθήκες αλλά και στο κυρίως πρόγραμμα. Δηλαδή, χρησιμοποιείται όπου υπάρχει η ανάγκη μετατροπής της εκάστοτε συνάρτησης από απλή λίστα που περιέχει τον πίνακα αληθείας της συνάρτησης σε ολοκληρωμένο object με τιμές για όλα τα χαρακτηριστικά της.

Για παράδειγμα, η παρακάτω εντολή δημιουργεί 10 διαφορετικές τυχαίες κρυπτογραφικές συναρτήσεις όπου ο πίνακα αληθείας της καθεμίας είναι 512 bits.

generation0(10,512)

5.2.2 cfeng.py

Η cfeng.py είναι η βιβλιοθήκη η οποία χρησιμοποιείται για την παραγωγή συναρτήσεων με παρόμοια χαρακτηριστικά με τις Carlet-Feng συναρτήσεις. Αρχικά, υπάρχει μια συνάρτηση Carlet-Feng για κάθε διαφορετικό μήκος πίνακα αληθείας (256, 1024, 2048 bits). Ο πίνακας αληθείας για κάθε μίας από αυτές βρίσκεται αποθηκευμένος σε ένα ξεχωριστό αρχείο txt. Μέσω της μεταβλητής original_feng, εισάγεται στο πρόγραμμα ο πίνακας αληθείας της εκάστοτε επιθυμητής Carlet-Feng. Ο πίνακας αληθείας έχει την μορφή λίστας από 0 και 1.

Εισαγωγή αρχικής συναρτήσεως Carlet-Feng για 256 bits
original_feng='truth_table_carlet_256.txt'
Εισαγωγή αρχικής συναρτήσεως Carlet-Feng για 1024 bits
original_feng='truth_table_carlet_1024.txt'

Με την αντιμετάθεση των bits κάποιων θέσεων, μπορούν να δημιουργηθούν εκ νέου συναρτήσεις οι οποίες φέρουν χαρακτηριστικά παρόμοια με την αρχική συνάρτηση. Γνωρίζοντας τον πίνακα αληθείας των αρχικών συναρτήσεων, χωρίζονται νοερά σε τμήματα συγκεκριμένου μήκους και έπειτα λαμβάνει χώρα η αντιμετάθεση των bits. Αναλυτικότερα:

- Ο υπάρχοντας πίνακας χωρίζεται νοερά σε blocks των 4 bit και για κάθε τέτοιο block, γίνεται swap το 2ο με το 3ο bit.
- Ο υπάρχοντας πίνακας χωρίζεται νοερά σε blocks των 8 bit και για κάθε τέτοιο block, γίνεται swap: i) το 2ο με το 5ο bit, ii) το 4ο με το 7ο bit.
- Ο υπάρχοντας πίνακας χωρίζεται νοερά σε blocks των 16 bit και για κάθε τέτοιο block, γίνεται swap: i) το 2ο με το 9ο bit, ii) το 4ο με το 11ο bit, iii) το 6ο με το 13ο bit, iv) το 8ο με το 15ο bit

Τα blocks ανά περίπτωση μετά ενώνονται εκ νέου. Το τελικό αποτέλεσμα είναι μια λίστα από λίστες όπου η κάθε εσωτερική λίστα αντιπροσωπεύει τον πίνακα αληθείας μίας συνάρτησης.

Τέλος, καλείται ένα κομμάτι της βιβλιοθήκης genesis_sage.py (objectify) το οποίο είναι υπεύθυνο για την τροφοδοσία των συναρτήσεων στο λογισμικό SageMath, τον υπολογισμό μέσω αυτού των χαρακτηριστικών της εκάστοτε συνάρτησης και την δημιουργία του αντικειμένου που αντιπροσωπεύει την συνάρτηση.

Αυτή η διαδικασία επαναλαμβάνεται για όλες τις περιπτώσεις, όπως αυτές παρουσιάστηκαν παραπάνω. Στο κυρίως πρόγραμμα επιστρέφονται όλες οι συναρτήσεις ως λίστα από objects, με τα χαρακτηριστικά τους και έτοιμες προς χρήση.

Όλα αυτά συμβαίνουν με την εκτέλεση της παρακάτω εντολής. Στο συγκεκριμένο παράδειγμα, δημιουργούνται όλες οι συναρτήσεις με μήκος πίνακα αληθείας 1024 bits.

```
fengmaker('truth_table_carlet_1024.txt' )
```

5.2.3 bent.sage

Το script bent.sage χρησιμοποιείται για την δημιουργία συναρτήσεων bent κατά την μέθοδο αναζήτησης συναρτήσεων με συναρτήσεις bent και Carlet-Feng και τις παράγωγές τους. Η δημιουργία και η επιλογή των συναρτήσεων bent συμβαίνει με την κλήση μίας εντολής.

Η εντολή δέχεται σαν παραμέτρους τον αριθμό μεταβλητών n των συναρτήσεων που θα παραχθούν (αντίστοιχη μεταβλητή στο πρόγραμμα: n_bent) και τον αριθμό των συναρτήσεων που θα επιστραφούν (αντίστοιχη μεταβλητή στο πρόγραμμα: bents_to_chose). Δεν επιστρέφονται όλες οι συναρτήσεις που παράγονται διότι ο αριθμός τους είναι πολύ μεγάλος (105 διαφορετικές συναρτήσεις για n=8) και συνεπώς δεν χρειάζεται να αξιοποιηθεί όλες σαν γονείς για τον αρχικό πληθυσμό.

Όλες οι bent συναρτήσεις έχουν την εξής μορφή, ανάλογα και τον αριθμό μεταβλητών τους:

Για n=6

x0*x1 + x2*x3 + x4*x5

Για n=8

x0*x1 + x2*x3 + x4*x5 + x6*x7

Βάση του αριθμού μεταβλητών n που δίνεται, δημιουργείται μία λίστα μεταβλητών από x0 εώς xn-1. Στην συνέχεια, δημιουργούνται όλα τα πιθανά ζευγάρια μεταβλητών από x0 εώς xn-1 και αποθηκεύονται σε μία λίστα. Αυτή η λίστα αντιπροσωπεύει όλα τα πιθανά γινομένα 2 μεταβλητών (x0*x1, x2*x3 κλπ). Έπειτα, τα ζεύγη μεταβλητών προσπευλαύνονται και δημιουργείται μία νέα λίστα, η οποία περιέχει τους πιθανούς συνδυασμούς των γινομένων. Κάθε συνδυασμός περιέχει n/2 γινόμενα-ζευγάρια.
Η παρακάτω εικόνα δείχνει την μορφή των ζευγών μεταβλητών καθώς και μερικούς πιθανούς συνδυασμούς τους για n=8.

```
C:\>python bent.py
('x0', 'x1')
('x0', 'x1')
('x0', 'x2')
('x0', 'x3')
('x0', 'x4')
('x0', 'x5')
(('x0', 'x1'), ('x0', 'x2'), ('x0', 'x3'), ('x0', 'x4'))
(('x0', 'x1'), ('x0', 'x2'), ('x0', 'x3'), ('x0', 'x5'))
(('x0', 'x1'), ('x0', 'x2'), ('x0', 'x3'), ('x0', 'x6'))
(('x0', 'x1'), ('x0', 'x2'), ('x0', 'x3'), ('x0', 'x7'))
(('x0', 'x1'), ('x0', 'x2'), ('x0', 'x3'), ('x1', 'x2'))
```

Εικόνα 11. Ζευγάρια μεταβλητών και πιθανοί συνδυασμοί τους για n=8

Ελέγχεται αν οι συνδυασμοί ζευγών μεταβλητών είναι μοναδικοί και για όσους είναι, εισάγονται τα “*” και “+” στον κάθε συνδυασμό ζευγών μεταβλητών. Έτσι, προκύπτουν οι ολοκληρωμένες συναρτήσεις bent, εκφραζόμενες στην κανονική αλγεβρική μορφή τους.

Στην συνέχεια, επιστρέφονται μερικές από αυτές, ίσες με τον αριθμό που δόθηκε σαν παράμετρος κατά την κλήση του function. Φυσικά, η ίδια συνάρτηση δεν μπορεί να επιστραφεί δύο φορές.

Για να ολοκληρωθούν και να μπορούν να χρησιμοποιηθούν οι συναρτήσεις με το υπόλοιπο πρόγραμμα, πρέπει να μετατραπούν σε objects, όπως κάθε άλλη συνάρτηση. Για να την επίτευξη αυτού, καλείται και πάλι για κάθε bent συνάρτηση το function objectify του genesis_sage.py. Όμως, αυτό το function δέχεται την συνάρτηση σαν πίνακα αληθείας και όχι σαν αλγεβρική κανονική μορφή. Οπότε, πρώτα χρησιμοποιείται το sage για να μετατραπεί η κάθε bent συνάρτηση από την αλγεβρική κανονική μορφή της στον αντίστοιχο πίνακα αληθείας.

Αποτέλεσμα των παραπάνω ενεργειών είναι ένας πίνακας από objects, όπου το κάθε object αναπαριστά μία bent συνάρτηση.

Όλα τα παραπάνω συμβαίνουν με την εκτέλεση μίας εντολής. Για παράδειγμα, η παρακάτω εντολή δημιουργεί συναρτήσεις bent 8 μεταβλητών και επιστρέφει 6 από αυτές.

bentmaker(8, 6)

Μεγάλη προσοχή πρέπει να δοθεί κατά την δήλωση του αριθμού μεταβλητών n των bent καθότι πρέπει να συμφωνεί με το μήκος του πίνακα αληθείας των συναρτήσεων Carlet-Feng και των παραγώγων τους, αφού όλες αυτές οι συναρτήσεις θα χρησιμοποιηθούν μαζί. Δηλαδή, αν χρησιμοποιηθούν Carlet-Feng και παράγωγες μήκους 256 bits, πρέπει να δοθεί σαν είσοδος n=8 αφού n^8=256 bits. Δεν μπορεί να δοθεί μήκος συναρτήσεων Carlet-Feng και των παραγώγων τους 256 bits και οι μεταβλητές των bent που θα δηλωθούν να είναι λιγότερες ή περισσότερες από 8. Για αυτό, έχει υλοποιηθεί ένας έλεγχος όταν χρησιμοποιούνται bent συναρτήσεις ο οποίος ειδοποιεί τον χρήστη αν έχει δηλώσει λάθος παραμέτρους.

5.2.4 darwin.py

Η βιβλιοθήκη darwin.py περιέχει όλες τις διαδικασίες ενός γενετικού αλγόριθμου που υλοποιήθηκαν και είναι μακράν η μεγαλύτερη βιβλιοθήκη του προγράμματος σε έκταση.

Παρακάτω αναλύεται η υλοποίηση και οι τεχνικές που χρησιμοποιήθηκαν.

- Μέθοδος επιλογής – ρουλέτα

Υλοποιήθηκαν 3 παραλλαγές της μεθόδου επιλογής ρουλέτας. Η πρώτη, η κλασική ρουλέτα, υπήρξε η αρχική και η βάση για τις υπόλοιπες.

Η μέθοδος αυτή προσομοιώνει μία ρουλέτα στην οποία οι καλύτεροι γονείς έχουν υψηλότερες πιθανότητες να επιλεχθούν αφήνοντας όμως και πιθανότητες για την επιλογή γονέων με λιγότερο καλά χαρακτηριστικά. Αυτό γιατί, όπως και στη φύση, κάποιο άτομο μπορεί να μην έχει συνολικά καλά χαρακτηριστικά αλλά να έχει ένα ή 2 χαρακτηριστικά τα οποία να είναι επιθυμητά. Η χρήση του για αναπαραγωγή μπορεί να συνεισφέρει θετικά στην δημιουργία ενός καλύτερου απογόνου. Αυτή η μέθοδος διατηρεί ευρύ το πεδίο αναζήτησης αντί να το περιορίζει στην συνεχόμενη αναπαραγωγή μεταξύ των καλύτερων και μόνο ατόμων.

Οι ρουλέτες ξεκινούν υπολογίζοντας την σωρευτική καταλληλότητα του κάθε γονέα. Η διαδικασία έχει ως εξής:

- Υπολογίζεται το άθροισμα της καταλληλότητας όλων των γονέων.
- Γίνεται μια διαίρεση της καταλληλότητας κάθε γονέα με τη συνολική καταλληλότητα από το προηγούμενο βήμα και το αποτέλεσμα αποθηκεύεται σε μια λίστα. Έτσι κανονικοποιούνται οι πιθανότητες επιλογής κάθε γονέα από το μηδέν μέχρι το ένα.
- Η παραπάνω λίστα προσπελαύνεται και σε μία νέα λίστα, προστίθεται το τρέχον άθροισμα των στοιχείων της. Οι αποστάσεις μεταξύ των στοιχείων αυτής της λίστας είναι στην ουσία το πεδίο πιθανοτήτων που έχει ο κάθε γονέας να επιλεγεί.
- Έπειτα επιλέγεται ένας τυχαίος, πραγματικός αριθμός από το μηδέν μέχρι το ένα και ανάλογα το πεδίο στο οποίο ανήκει, επιλέγεται ο αντίστοιχος γονέας. Δηλαδή, για κάθε στοιχείο/πιθανότητα i, οι τιμές ανάμεσα σε i και i+1 (επόμενο στοιχείο) καθορίζουν το πεδίο αριθμών για το οποίο επιλέγεται ο αντίστοιχος γονέας i. Για

παράδειγμα, αν ο τυχαίος αριθμός που θα παραχθεί είναι από 0 έως i τότε επιλέγεται ο πρώτος γονέας. Αν είναι από i έως i+1 τότε επιλέγεται ο δεύτερος γονέας και ούτω καθεξής. Αυτό αντιστοιχεί στον αναλογικό διαμοιρασμό των πιθανοτήτων σε μια ρουλέτα.

Οι άλλες 2 παραλλαγές δρουν ακριβώς με τον ίδιο τρόπο όμως δίνουν τη δυνατότητα επιλογής πολλαπλών γονέων (ο αριθμός των οποίων ορίζεται κατά την κλήση του function) σε αντίθεση με την κλασική υλοποίηση που επιστρέφει μόνο έναν. Η πρώτη παραλλαγή επιτρέπει την πολλαπλή επιλογή του ίδιου γονέα, ενώ η δεύτερη επιλέγει μοναδικούς γονείς.

- Μέθοδος crossover/δημιουργίας απογόνων.

Η μέθοδος δημιουργίας απογόνων που χρησιμοποιήθηκε είναι η k-point crossover. Αποτελεί μία πιο ευέλικτη/εξελιγμένη μορφή της single-point crossover.

Αρχικά, υλοποιήθηκε η single-point crossover η οποία επιλέγει ένα τυχαίο σημείο στους γονείς που προκύπτουν από το προηγούμενο στάδιο. Στην συνέχεια, χωρίζει τον κάθε γονέα στα δύο βάσει αυτού του σημείου. Έτσι προκύπτουν τέσσερις διαφορετικές ακολουθίες γονιδίων, δύο από τον κάθε γονέα. Τέλος, ανταλλάζει ένα κομμάτι του πρώτου γονιού με ένα του δεύτερου και έτσι προκύπτουν 2 τελείως νέοι απόγονοι-συναρτήσεις. Φυσικά, αυτό μπορεί να γίνει και με παραπάνω από δύο γονείς. Η μέθοδος αυτή δημιουργήθηκε σαν πρώτο βήμα, σαν βάση. Δεν υπήρχε εξαρχής η πρόθεση να χρησιμοποιηθεί στο τέλος διότι κρίθηκε κάπως περιοριστική.

Στην συνέχεια, υλοποιήθηκε η μέθοδος k-points crossover. Κατά την κλήση της μεθόδου, ορίζεται ο πληθυσμός των ατόμων και το σύνολο των σημείων για crossover.

Επιλέγονται τυχαία τόσα σημεία crossover όσα δηλώθηκαν κατά την κλήση του function, από το 0 μέχρι και το μέγιστο μήκος του πίνακα αληθείας των γονέων. Ο πίνακας αληθείας του εκάστοτε γονέα διασπάται σε μικρότερα κομμάτια, ανάλογα τα τυχαία σημεία που επιλέχθηκαν. Τα σημεία αυτά παραμένουν σταθερά κατά την διαδικασία. Δηλαδή όλοι οι γονείς χωρίζονται ακριβώς με τον ίδιο τρόπο ακριβώς στα ίδια σημεία. Έτσι, όλα τα πρώτα τμήματα των γονέων έχουν όμοιο μήκος όπως και τα δεύτερα και τα

τρίτα και ούτω καθεξής. Έπειτα, τα τυχαία αυτά τμήματα αναμειγνύονται μεταξύ τους παράγοντας νέους απογόνους. Για κάθε πρώτο τμήμα του κάθε γονέα, προσκολλάται τυχαία ένα από τα δεύτερα τμήματα, στη συνέχεια ένα από τα τρίτα και ούτω καθεξής. Οι προκύπτοντες πίνακες αληθείας των απογόνων φέρουν το ίδιο μήκος με αυτούς των αρχικών γονέων.

Τέλος, συγκρίνεται ο πίνακας αληθείας των απογόνων με αυτούς των αρχικών γονέων, ώστε να διασφαλιστεί πως κάθε απόγονος είναι διαφορετικός από κάθε γονέα. Αν εντοπιστεί πως κάποιος απόγονος φέρει ίδιο πίνακα αληθείας με κάποιον γονέα, τότε ο απόγονος απορρίπτεται. Οι υπόλοιποι απόγονοι επιστρέφονται σαν λίστα από λίστες στο κυρίως πρόγραμμα. Κάθε εσωτερική λίστα αντιπροσωπεύει τον πίνακα αληθείας ενός απογόνου. Όλα τα παραπάνω συμβαίνουν με την κλήση μίας εντολής στο κυρίως πρόγραμμα.

Για παράδειγμα, η παρακάτω εντολή εφαρμόζει την όλη διαδικασία στον αρχικό πληθυσμό (λίστα από objects) για 200 τυχαία crossover points. Προφανώς, όπως αναφέρθηκε και παραπάνω, ο αριθμός των τυχαίων crossover points δεν μπορεί να είναι μεγαλύτερος από το μήκος του πίνακα αληθείας των γονέων. Στην συγκεκριμένη περίπτωση, η εντολή θα λειτουργήσει για γονείς με μήκος πίνακα αληθείας από 256 και άνω (χρησιμοποιούνται μόνο δυνάμεις του 2) αλλά όχι για αυτούς με μήκος πίνακα αληθείας 128 ή λιγότερο.

kpc(generation0,200)

Ακολουθεί ακόμα ένα παράδειγμα για την κατανόηση της χρήσης και δημιουργίας των σημείων crossover. Έστω ότι δίνεται στο πρόγραμμα η παρακάτω εντολή για γονείς με μήκος πίνακα αληθείας 128:

kpc(generation0,5)

Οι 5 τυχαίοι αριθμοί που θα παραχθούν θα μπορούσαν να είναι οι 5,46,72,100,120.
Κάθε γονέας θα τεμαχιστεί σε μικρότερα κομμάτια/λίστες ως εξής:

- Στοιχεία 0-5
- Στοιχεία 6-46
- Στοιχεία 47-72
- Στοιχεία 73-100
- Στοιχεία 101-120
- Στοιχεία 121-128.

Τα κομμάτια αυτά «συρράβονται» μεταξύ τους τυχαία, όπως αναφέρθηκε παραπάνω και έτσι προκύπτουν οι απόγονοι.

Αξίζει να σημειωθεί ότι το μεγαλύτερο και πολυπλοκότερο κομμάτι προγραμματιστικά, ήταν η υλοποίηση του k-point crossover(περίπου 140 γραμμές κώδικα).

- Mutation(μετάλλαξη)

Ο μηχανισμός μετάλλαξης υλοποιήθηκε με 2 παραλλαγές. Αρχικά, σαν παράμετροι του function, ορίζεται ο πληθυσμός πάνω στον οποίο θα δράσει ο μηχανισμός μετάλλαξης και το ποσοστό επί τοις εκατό της πιθανότητας μετάλλαξης. Για κάθε άτομο μέσα σε αυτό τον πληθυσμό, δημιουργείται ένα ξεχωριστό αντίγραφό του έτσι ώστε να μην τροποποιηθεί το αρχικό. Με αυτόν τον τρόπο, διατηρείται ανέπαφος ο αρχικός πληθυσμός εισόδου αλλά και προστίθενται σε αυτόν τα μεταλλαγμένα άτομα. Στην συνέχεια ο μηχανισμός προσπελαύνει τον πίνακα αληθείας του ατόμου/συνάρτησης ανά bit και δημιουργεί έναν τυχαίο αριθμό από το μηδέν μέχρι το 100 για αυτό. Αν ο αριθμός αυτός είναι μικρότερος από το ποσοστό μετάλλαξης που δόθηκε κατά την κλήση του function, τότε το εκάστοτε bit αντιστρέφεται. Αν όχι, το bit παραμένει ως έχει. Τέλος, αφού η διαδικασία επαναληφθεί για όλα τα bits της δεδομένης συνάρτησης, συγκρίνεται αν η προκύπτουσα συνάρτηση είναι η ίδια με το αρχικό άτομο. Αν δεν είναι, και άρα υπάρχει όντως μετάλλαξη, αποθηκεύεται σε μια λίστα προς επιστροφή στο κυρίως πρόγραμμα. Η παραπάνω διαδικασία επαναλαμβάνεται για κάθε άτομο στον πληθυσμό που δόθηκε ως είσοδο στο function της μετάλλαξης. Οι μεταλλαγμένοι απόγονοι (αν υπάρχουν) επιστρέφονται σαν λίστα από λίστες στο κυρίως πρόγραμμα. Κάθε εσωτερική λίστα αντιπροσωπεύει τον πίνακα αληθείας ενός απογόνου.

Η δεύτερη παραλλαγή δρα ακριβώς όπως η πρώτη με τη διαφορά ότι είναι 2 επιπέδων. Κατά την κλήση του function, Υπάρχει μια παραπάνω παράμετρος η οποία ορίζει μία πιθανότητα μετάλλαξης επί τοις εκατό για τον εκάστοτε γονέα. Για κάθε άτομο, παράγεται ένας τυχαίος αριθμός και αν αυτός ο αριθμός είναι μικρότερος από την πιθανότητα μετάλλαξης του γονέα, τότε ξεκινάει η ίδια λειτουργία για το κάθε bit του, όπως και στην πρώτη περίπτωση. Αν όχι, τότε αφήνει τον γονέα ανέγγιχτο και πάει στον επόμενο.

Για παράδειγμα, η παρακάτω εντολή δέχεται έναν πληθυσμό γονέων:

mutation(population, 7)

Για κάθε bit κάθε ατόμου παράγεται ένας τυχαίος αριθμός από το 0 μέχρι το 100. Αν ο αριθμός είναι μικρότερος του 7, τότε το αντίστοιχο bit αντιστρέφεται,

mutation_2layer(population, 7, 15)

Αντίστοιχα εδώ, για κάθε γονέα παράγεται ένας αριθμός από το 0 μέχρι το 100. Αν ο αριθμός είναι μικρότερος του 15, τότε ο αλγόριθμος ακολουθεί για αυτόν τον γονέα την διαδικασία του mutation(population, 7). Αν όχι, τότε προχωρά στον επόμενο γονέα όπου η διαδικασία επαναλαμβάνεται εκ νέου.

- Cull_worst (Επιλογή ατόμων για την επόμενη γενιά)

Όσον αφορά τον μηχανισμό που επιλέγει τα άτομα για την επόμενη γενιά, χρησιμοποιήθηκε μία εκδοχή της αντικατάστασης των χειρότερων γονέων. Ο μηχανισμός αυτός καθαιρεί το δεδομένο ποσοστό των χειρότερων ατόμων του πληθυσμού στην εκάστοτε γενιά.

Το function δέχεται όλον τον αρχικό πληθυσμό μαζί με τους απογόνους και τις μεταλλάξεις, καθώς και το ποσοστό των ατόμων που θα καθαιρεθεί. Έτσι, στην επόμενη γενιά περνάει το εναπομένον ποσοστό των καλύτερων ατόμων. Για να αποφευχθεί η υπερβολική καθαίρεση, γεγονός που θα οδηγούσε σε προβλήματα στον αλγόριθμο κατά την κλήση των άλλων functions στην επόμενη γενιά, υλοποιείται η παρακάτω συνθήκη.

Μετά την καθαίρεση, αν το σύνολο των ατόμων που πρόκειται να προαχθεί στην επόμενη γενιά είναι μικρότερο των γονέων που επιλέγονται στο στάδιο της επιλογής πριν το crossover, τότε ο τρέχων πληθυσμός μένει απείραχτος και δεν συμβαίνει καμία καθαίρεση. Αν δεν υπήρχε αυτήν η συνθήκη, τότε για μεγάλα ποσοστά καθαίρεσης και μικρό αριθμό επιλεγόμενων γονέων για crossover, η επόμενη γενιά, δεν θα είχε τόσους γονείς όσους ζητούνταν από το στάδιο της επιλογής.

Για παράδειγμα:

culling=cull_worst(sorted_individuals, cull_percentage, no_of_parents_to_select)

Το function δέχεται τον ταξινομημένο κατά fitness πληθυσμό αφού έχουν προστεθεί σε αυτόν οι απόγονοι και οι μεταλλάξεις. Καθαιρεί το ποσοστό ίσο με το cull_percentage του πληθυσμού. Λαμβάνει όμως υπόψιν ακριβώς την ίδια μεταβλητή που χρησιμοποιείται για τον καθορισμό του αριθμού γονέων που χρησιμοποιούνται κατά το στάδιο της επιλογής.

Ας υποτεθεί πως ο πληθυσμός που εισάγεται στο function είναι 10 γονείς και κάθε φορά επιλέγονται 5 ενώ το ποσοστό γονέων που θα καθαιρεθούν είναι 70%. Εύκολα καταλαβαίνει κανείς πως στην επόμενη γενιά θα υπάρχει πρόβλημα, διότι το πρόγραμμα θα θέλει να επιλέξει 5 γονείς όμως θα υπάρχουν μόνο 3. Οπότε σε αυτήν την γενιά δεν θα συμβεί καθαίρεση και ο πληθυσμός θα συνεχίσει αυτούσιος στην επόμενη. Το παρόν παράδειγμα είναι ακραίο, αφού για σχετικά μικρό πληθυσμό υπάρχει πολύ μεγάλο ποσοστό καθαίρεσης. Εξυπηρετεί όμως στην επεξήγηση αυτής της δικλείδας ασφαλείας.

5.2.5 Automation.py

Από τα παραπάνω, εύκολα συμπεραίνει κανείς ότι υπάρχουν πολλοί παράμετροι και ένας τεράστιος όγκος πιθανόν συνδυασμών αυτών οι οποίοι μπορούν να εισαχθούν στο πρόγραμμα για μία εκτέλεση. Αυτό δημιουργεί προβληματισμό, και ταυτόχρονα ερευνητική πρόκληση, ως προς την επιλογή των τιμών των παραμέτρων στην εκάστοτε εκτέλεση, καθώς δεν υπάρχει προηγούμενη γνώση για το ποιός συνδυασμός τιμών θα παράξει ικανοποιητικά αποτελέσματα. Για την αντιμετώπιση αυτής της ανάγκης αναπτύχθηκε το script automation.py. Αυτό λύνει το πρόβλημα της επιλογής παραμέτρων και των πολλαπλών, χειροκίνητων εκτελέσεων για την συλλογή δεδομένων,

έχοντας ως είσοδο διαφορετικούς συνδυασμούς παραμέτρων κάθε φορά για πολλαπλές εκτελέσεις του κυρίως προγράμματος.

Η εργασία που επιτελεί το πρόγραμμα αυτό είναι η εξής. Με δεδομένα τις αρχικές τιμές μεταβλητών, το ρυθμό αύξησης και το όριο για την κάθε μία, δοκιμάζει όλα τα ενδεχόμενα πιθανά σενάρια. Οι δοθείσες τιμές περνούν στο κυρίως πρόγραμμα σαν arguments, όπως ακριβώς συμβαίνει και με ποικίλα προγράμματα που καλούνται από την γραμμή εντολών. Τέτοιες μεταβλητές περιλαμβάνουν τα εξής:

1) Ανά εκτέλεση, οι παρακάτω παράμετροι μεταβάλλονται, ανάλογα με τον δεδομένο ρυθμό αύξησής τους, μέχρι να φτάσουν το όριό τους:

- Το πλήθος των γονέων που δημιουργούνται ή επιλέγονται.
- Το πλήθος των σημείων για crossover.

2) Επιπλέον, στο πρόγραμμα περνάνε και ως σταθερές τιμές (χωρίς να μεταβάλλεται ο αριθμός τους ανάμεσα στις εκτελέσεις) τα εξής:

- Ο αριθμός των bits των συναρτήσεων (ουσιαστικά, το μέγεθος του πίνακα αληθείας τους).
- Η πιθανότητα μετάλλαξης.
- Το ποσοστό γονέων που θα «περάσει» στην επόμενη γενιά.
- Οι εκάστοτε αρχικές συναρτήσεις bent και Carlet-Feng με τις παράγωγές τους (οι οποίες όμως χρησιμοποιούνται μόνο εφόσον χρειάζονται, ανάλογα την μέθοδο αναζήτησης που ακολουθείται κάθε φορά).

Το main.sage μαζί με το automation.py έχουν την δυνατότητα αναζήτησης συναρτήσεων σε 3 διαφορετικά πεδία. Αυτά είναι:

- αποκλειστικά με τυχαίες συναρτήσεις ως πληθυσμό.
- τυχαίες συναρτήσεις με συναρτήσεις Carlet-Feng και τις παράγωγές τους ως αρχικό πληθυσμό.
- με συναρτήσεις bent και Carlet-Feng και τις παράγωγές τους ως αρχικό πληθυσμό.

1 ή 3 εμφωλευμένες εντολές επανάληψης «for» (ανάλογα την περίπτωση) εγγυώνται την πλήρη κάλυψη του εκάστοτε πεδίου αναζήτησης.

Μεταξύ των διαφορετικών πεδίων αναζήτησης, υπάρχουν κάποιες διαφορές κατά την εκτέλεση. Αυτές αναλύονται στην ενότητα 5.3.

Για την καλύτερη κατανόηση της λειτουργίας αυτού του τμήματος του προγράμματος, παρακάτω παρατίθεται ένα παράδειγμα.

Σε αυτό το παράδειγμα οι αρχικές τιμές, οι ρυθμοί αύξησης και τα όρια που τίθενται έχουν ως εξής:

Αρχικός αριθμός γονέων που παράγονται=10
Ανώτατος αριθμός γονέων που παράγονται = 51
Ρυθμός αύξησης πληθυσμού γονέων =10

Αριθμός bits των συναρτήσεων=256 (δηλαδή πρόκειται για συναρτήσεις των 8 μεταβλητών).

Αρχικός αριθμός γονέων που επιλέγονται=4
Το αρχικό ανώτατο όριο των γονέων που επιλέγονται=0 (Το οποίο αυξάνεται κατά 9 ανα δεκάδα γονέων στον πληθυσμό.)
Ρυθμός αύξησης γονέων που επιλέγονται=4
Ανώτατο όριο γονέων που επιλέγονται ανά δεκάδα γονέων=9

Αρχικό πλήθος σημείων crossover=50
Ρυθμός αύξησης σημείων crossover=50
Ανώτατο όριο σημείων crossover=201

Ποσοστό μετάλλαξης=7
Ποσοστό κάθαρσης επόμενης γενιάς=65

Οι ανωτέρω παράμετροι «μεταφράζονται» ως εξής:

1) Σε κάθε εκτέλεση του προγράμματος παράγονται 10 γονείς.

   Στις πρώτες εκτελέσεις επιλέγονται 4 εξ' αυτών και υπάρχουν 50 crossover points για την 1η, 100 για την 2η, 150 για την 3η και 200 για την 4η.(4 εκτελέσεις)

2) Το ίδιο επαναλαμβάνεται επιλέγοντας 8 γονείς αφού ο ρυθμός αύξησης των γονέων έχει τιμή 4. Στο επόμενο βήμα ο αλγόριθμος θα προσπαθήσει να επιλέξει 12(8+4), όμως 12>9 που έχει τεθέι σαν όριο (και φυσικά, 12>το σύνολο των 10 γονέων που υπάρχουν). Οπότε θα σταματήσει εκεί. (8 εκτελέσεις συνολικά).

3) Έπειτα, σε κάθε εκτέλεση του προγράμματος παράγονται 20 γονείς.

   Στις πρώτες εκτελέσεις με 20 γονείς επιλέγονται 4 εξ' αυτών και υπάρχουν 50 crossover points για την 1η, 100 για την 2η, 150 για την 3η και 200 για την 4η.(4 εκτελέσεις).

4) Το ίδιο επαναλαμβάνεται επιλέγοντας 8, 12 και 16 γονείς αντίστοιχα. Έπειτα, ο αλγόριθμος θα προσπαθήσει να κάνει το ίδιο για 16+4=20 γονείς. Όμως το όριο τώρα έχει τεθεί στο 18 (9 αρχικά +9 για κάθε δεκάδα γονέων. Έχουμε 2 δεκάδες, άρα 18). Οπότε, αφού 20>18, ο αλγόριθμος σταματάει. (16 εκτελέσεις συνολικά – 4 εκτελέσεις για την κάθε περίπτωση).

5) Η ίδια διαδικασία συνεχίζεται με τους 30,40 και 50 γονείς. Ανά δεκάδα γονέων, το πρόγραμμα ξεκινάει πάντα επιλέγοντας 4 γονείς, ενώ το μέγιστο όριο που μπορεί να επιλεγεί αυξάνει κατά 9. Ο αριθμός των γονέων που επιλέγονται αυξάνεται ανά 4, και τα σημεία crossover ανά 50 κάθε φορά, μέχρι τα 200.

Φυσικά, οι εν λόγω αριθμοί ορίων, αρχικών τιμών και ρυθμών αύξησης αποτελούν ένα παράδειγμα και μπορούν εύκολα να τροποποιηθούν. Με αυτόν τον τρόπο υπάρχει δυνατότητα για δοκιμές με πολλαπλούς συνδυασμούς παραμέτρων χωρίς να υπάρχει ανάγκη φυσικής παρουσίας στον υπολογιστή.

```
no_of_parents=51
no_of_bits=256
parents_to_select= 0
crossover_points=201
original_feng='truth_table_carlet_256.txt'
#method_to_follow = "fengs+gens"
#method_to_follow = "cfeng"
#method_to_follow = "randomgensonly"
method_to_follow = "bent+fengs"
n_bents = 8
bents_to_chose = 8

mutation_rate=7
elitism_percentage=65

print("Number of bits for each parent:",no_of_bits)
print("Mutation rate:",mutation_rate)
print("Elitism percentage:",elitism_percentage)
print("----------------------")

f = open("report.txt", "w")
f.write("Number of bits for each parent:"+str(no_of_bits)+"\n")
f.write("Mutation rate:"+str(mutation_rate)+"\n")
f.write("Elitism percentage:"+str(elitism_percentage)+"\n")
f.write("--------------------"+"\n")
f.close()
if method_to_follow == "fengs+gens" or method_to_follow == "randomgensonly":
    for i in range(10,no_of_parents,10):
        parents_to_select = parents_to_select + 9
        for x in range(4,parents_to_select,4):
            for j in range(50,crossover_points,50):
                # Execute the second script in a new shell environment
                subprocess.call(
                f'sage ultimate.sage {int(i)} {int(no_of_bits)} {int(x)} {int(j)}
                shell=True
                )
                time.sleep(5)
```

Εικόνα 12. Παράδειγμα των παραμέτρων που εισάγονται στην είσοδο του προγράμματος.

5.2.6 Main.sage

Το κυρίως πρόγραμμα συνδυάζει όλα τα παραπάνω επιμέρους στοιχεία/βιβλιοθήκες προκειμένου να εκτελέσει την αναζήτηση για κρυπτογραφικές συναρτήσεις με επιθυμητά χαρακτηριστικά.

Η αρχική ιδέα ήτανε η κάθε εκτέλεση του κυρίως προγράμματος να συνεχίζει για δεδομένο συνολικό αριθμό γενεών. Ωστόσο, λόγο υλικών περιορισμών και συγκεκριμένα της μνήμης RAM, αναγκαστικά, η συνθήκη τερματισμού εκτέλεσης του αλγορίθμου έπρεπε να τροποποιηθεί. Έτσι, το πρόγραμμα τροποποιήθηκε ώστε να σταματά η εκτέλεση του κατά την επίτευξη ενός δεδομένου ποσοστού χρήσης της RAM.

Ξεκινώντας, υπάρχει η εισαγωγή των επιμέρους στοιχείων/βιβλιοθηκών και το function που υπολογίζει το τρέχον ποσοστό χρήσης της μνήμης RAM. Έπειτα, εισάγονται οι απαραίτητες μεταβλητές όπως αναφέρθηκαν παραπάνω (αριθμός γονέων, γονέων που επιλέγονται κλπ.) κατά την κλήση του προγράμματος. Η κλήση γίνεται είτε από το automation.py (πολλαπλές εκτελέσεις) είτε από την γραμμή εντολών (μεμονωμένη εκτέλεση).

Όπως ειπώθηκε, το πρόγραμμα έχει 3 διαφορετικούς τρόπους λειτουργίας. Η λειτουργία που επιλέγεται κάθε φορά ορίζεται μέσω του ορίσματος που δέχεται το πρόγραμμα, “method_to_follow”. Η τιμή του ορίσματος περνάει σε έλεγχο συνθηκών if ο οποίος καθορίζει και την εκάστοτε δημιουργία του αρχικού πληθυσμού που θα χρησιμοποιηθεί.

Στην συνέχεια λαμβάνει χώρο η επιλογή και η αναπαραγωγή crossover των γονέων. Όπως αναφέρθηκε, οι απόγονοι που προκύπτουν από το στάδιο της αναπαραγωγής ελέγχονται αν είναι όμοιοι με κάποιον ήδη υπάρχοντα γονέα, πριν παραδοθούν στο κυρίως πρόγραμμα. Συνεπώς, υπάρχει μία πιθανότητα, κυρίως στους μικρούς πληθυσμούς με μικρούς πίνακες αληθείας, η αναπαραγωγή να μη γυρίσει κανέναν απόγονο στο κυρίως πρόγραμμα. Οπότε, πριν προχωρήσει η εκτέλεση του προγράμματος ελέγχεται αν όντως υπάρχουν απόγονοι.

Πάνω σε αυτούς τους απογόνους λαμβάνει χώρα ή μετάλλαξη. Και εδώ υπάρχει πιθανότητα να μην προκύψει κανένας μεταλλαγμένος απόγονος. Οπότε το function είναι πιθανό να μην γυρίσει τίποτα στο κυρίως πρόγραμμα και αυτό.

Σε αυτό το σημείο όλοι οι απόγονοι(κανονικοί & μεταλλαγμένοι) που υπάρχουν, δεν είναι παρά μια λίστα από μηδέν και ένα ο καθένας από αυτούς. Όλοι μαζί, βρίσκονται σε μία μεγαλύτερη, εξωτερική λίστα. Δεν έχουν μετατραπεί ακόμα σε προγραμματιστικό object

και δεν έχουνε καμία τιμή για κανένα χαρακτηριστικό ως κρυπτογραφικές συναρτήσεις. Εφόσον λοιπόν υφίστανται, τροφοδοτούνται στο sage και έτσι μετατρέπονται σε αντικείμενα με τις αντίστοιχες τιμές για το εκάστοτε χαρακτηριστικό, όπως ακριβώς και οι αρχικοί γονείς.

Έπειτα, οι απόγονοι-αντικείμενα προστίθενται στον τρέχοντα πληθυσμό. Στη συνέχεια λαμβάνει χώρα μια ταξινόμηση από το άτομο με την υψηλότερη καταλληλότητα προς αυτό με την χαμηλότερη.

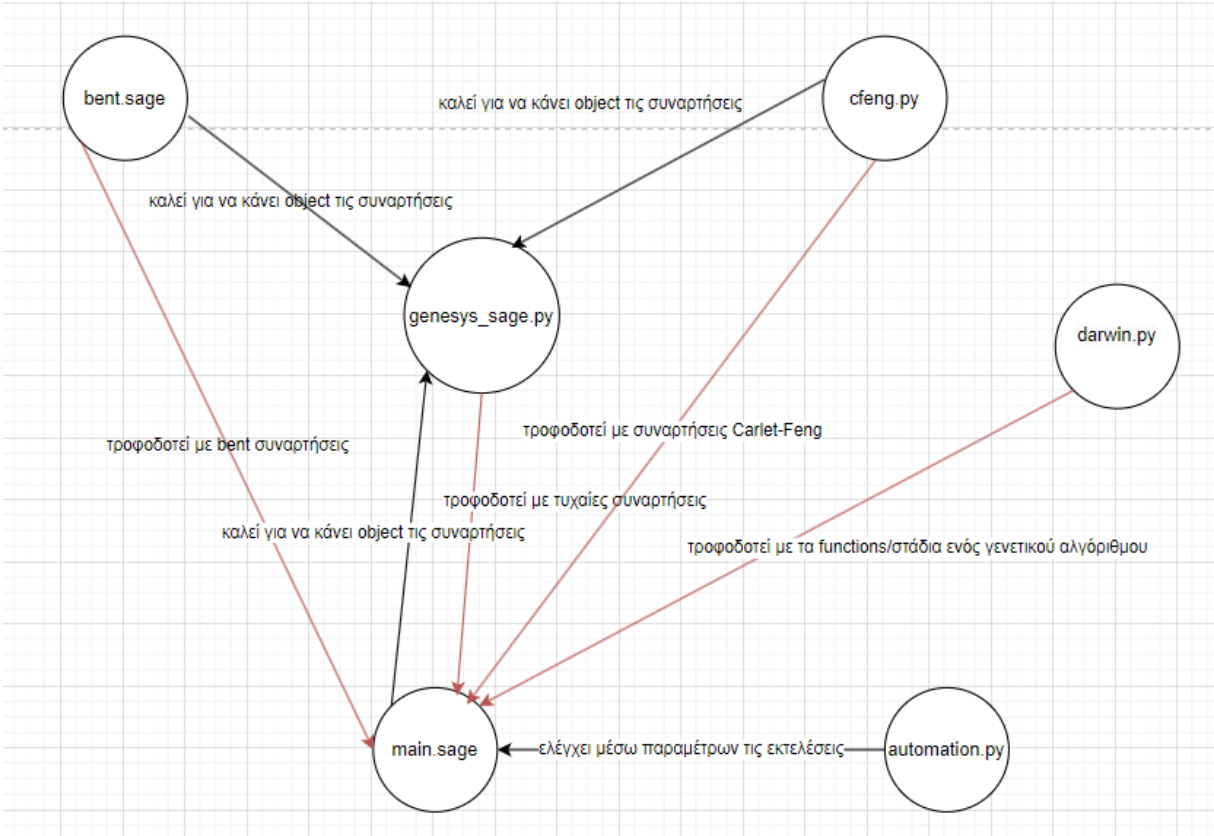
Τελευταίο βήμα που μένει είναι η δράση της επιλογή των ατόμων που θα περάσουν στην επόμενη γενιά. Το function λαμβάνει τον ταξινομημένο πλέον πληθυσμό και καθαιρεί το δεδομένο ποσοστό των χειρότερων ατόμων.

Η παραπάνω διαδικασία αποτελεί το πέρασμα μιας γενιάς. Στο τέλος κάθε γενιάς καταγράφεται το καλύτερο άτομο μαζί με όλα του τα χαρακτηριστικά. Στη συνέχεια ελέγχεται το ποσοστό της χρησιμοποιουμένης μνήμης RAM και, εφόσον δεν έχει γεμίσει, το πρόγραμμα συνεχίζει στην επόμενη γένια.

Στο τέλος κάθε εκτέλεσης γράφονται σε ένα αρχείο στοιχεία όπως αριθμός αρχικού πληθυσμού, γονέων που επιλέχθηκαν, πλήθος σημείων crossover, συνολικός αριθμός γενεών, χρόνος εκτέλεσης και λειτουργία που χρησιμοποιήθηκε.

Εφόσον το πρόγραμμα έχει κληθεί από το automation.py, ξεκινά η επόμενη εκτέλεσή του με τα εκ νέου στοιχεία όπως εξηγήθηκε ανωτέρω.

Παρακάτω παρουσιάζεται ένα διάγραμμα για την καλύτερη οπτικοποίηση της αλληλεπίδρασης των διάφορων κομματιών/βιβλιοθηκών μεταξύ τους.



Εικόνα 13. Διάγραμμα για την αλληλεπίδραση των διάφορων στοιχείων του προγράμματος.

5.3 Διαφορές μεταξύ των μεθόδων αναζήτησης

Οι κύριες διαφορές εντοπίζονται μεταξύ της μεθοδολογίας αναζητήσης με bent και Carlet-Feng και των παράγωγων συναρτήσεων με τις άλλες δύο μεθοδολογίες.

Αναλυτικότερα, οι μέθοδοι αναζήτησης αποκλειστικά τυχαίων συναρτήσεων και αναζήτησης έχοντας ως αρχικό πληθυσμό τυχαίες συναρτήσεις αλλά και Carlet-Feng και τις παράγωγες συναρτήσεις τους, έχουν μόνο μία διαφορά μεταξύ τους. Στην μέθοδο που συμμετέχουν οι συναρτήσεις Carlet-Feng και οι παράγωγες, αυτές προστίθενται εκ νέου στον πληθυσμό που επιλέγεται για crossover, μετά το στάδιο της επιλογής. Αυτό συμβαίνει για κάθε εκτέλεση, σε κάθε γενιά. Ο λόγος είναι ότι θέλουμε οι συναρτήσεις Carlet-Feng και οι παράγωγες να συμμετέχουν όσο πιο ενεργά γίνεται στην δημιουργία συναρτήσεων. Έτσι, για να εξασφαλιστεί ότι πάντα είναι παρούσες και δεν εξαλείφονται τελείως με το πέρασμα των γενεών, ακολουθήθηκε η παραπάνω διαδικασία. Αφενός, είναι μία μορφή ελιτισμού, αφετέρου αφήνει ανοιχτό το ενδεχόμενο κάποιες από αυτές

να τύχει να εμφανίζονται περισσότερες από μία φορές στην εκάστοτε γενιά. Επειδή όμως παράγεται μεγάλος αριθμός συναρτήσεων σε κάθε γενιά και λόγω της δομής και λειτουργίας της μεθόδου k-point crossover (βλ. κεφάλαιο 3), οι Carlet-Feng και οι παράγωγες συναρτήσεις δεν μπορούν να μονοπωλήσουν την συμμετοχή τους στην δημιουργία των απογόνων. Αντιθέτως, απλά συμμετέχουν πιο ενεργά στην δημιουργία των απογόνων, που είναι και το ζητούμενο.

Όσον αφορά την μεθοδολογία στην οποία ο αρχικός πληθυσμός αποτελείται από bent, Carlet-Feng και τις παράγωγες συναρτήσεις τους, η διαφορά με τις άλλες μεθόδους είναι ότι εδώ δεν υπάρχει στάδιο επιλογής στον αλγόριθμο. Αυτό συμβαίνει διότι είναι επιθυμητό να συμμετέχουν όλες στην αναπαραγωγή, ώστε να υπάρχει η μεγαλύτερη δυνατή ποικιλομορφία. Οπότε, στην ουσία, ο αρχικό πληθυσμός αναπαράγεται, οι απόγονοι και οι μεταλλάξεις προστίθενται σε αυτόν, και στο τέλος καθαιρούνται τα χειρότερα άτομα σε ποσοστό x% του πληθυσμού, όπως και στις άλλες μεθοδολογίες. Η ουσιαστική διαφορά δηλαδή, είναι ότι δεν υπάρχει στάδιο επιλογής καθώς και ότι ανάμεσα στις εκτελέσεις το μόνο που αλλάζει είναι το σύνολο των σημείων για crossover.

Επιπρόσθετα, πρέπει να σημειωθεί πως ανά μεθοδολογία και ανά εκτέλεση, οι συναρτήσεις έχουν όλες το ίδιο μήκος πίνακα αληθείας μεταξύ τους, ανεξάρτητα από την κατηγορία που ανήκουν. Δηλαδή, στην μεθοδολογία τυχαίων συναρτήσεων, όλες οι συναρτήσεις φέρουν το ίδιο μήκος πίνακα αλήθείας. Στην μεθολογία που χρησιμοποιούνται τυχαίες συναρτήσεις καθώς και Carlet-Fengs, πάλι έχουν όλες το ίδιο μήκος πίνακα αληθείας κλπ. Ειδικότερα για την μεθοδολογία όπου χρησιμοποιούνται συναρτήσεις bent με Carlet-Feng και τις παράγωγές τους, πρέπει να δοθεί μεγάλη προσοχή κατά την δήλωση του αριθμού μεταβλητών n των bent καθότι πρέπει να συμφωνεί με το μήκος του πίνακα αληθείας των συναρτήσεων Carlet-Feng και των παραγώγων τους, αφού όλες αυτές οι συναρτήσεις θα χρησιμοποιηθούν μαζί. Δηλαδή, αν χρησιμοποιηθούν Carlet-Feng και παράγωγες μήκους 256 bits, πρέπει να δοθεί σαν είσοδος n=8 αφού n^8=256 bits. Δεν μπορεί να δοθεί μήκος συναρτήσεων Carlet-Feng και των παραγώγων τους 256 bits και οι μεταβλητές των bent που θα δηλωθούν να είναι λιγότερες ή περισσότερες από 8. Για αυτό, έχει υλοποιηθεί ένας έλεγχος όταν χρησιμοποιούνται bent συναρτήσεις ο οποίος ειδοποιεί τον χρήστη αν έχει δηλώσει λάθος παραμέτρους. Ο παραπάνω έλεγχος βρίσκεται στο πρόγραμμα του automation.py.

Τέλος, όπως είναι κατανοητό, δεν συμμετέχουν όλοι οι παράμετροι που έχουν αναφερθεί ως τώρα (εικόνα 11) σε όλες τις μεθοδολογίες αναζήτησης. Για παράδειγμα, στην μεθοδολογία bent-Carlet-Feng, δεν συμμετέχουν τα εξής:

- Αρχικός αριθμός γονέων που παράγονται
- Ανώτατος αριθμός γονέων που παράγονται
- Ρυθμός αύξησης πληθυσμού γονέων

- Αρχικός αριθμός γονέων που επιλέγονται (ο οποίος τίθεται 0)
- Ρυθμός αύξησης γονέων που επιλέγονται
- Ανώτατο όριο γονέων που επιλέγονται

Είναι λογικό γιατί σε αυτήν την μεθοδολογία δεν παράγεται διαφορετικός αριθμός γονέων κάθε φορά αφού ο αριθμός των συναρτήσεων που παράγονται από τις Carlet-Feng για δεδομένο μήκος πίνακα αληθείας είναι σταθερός (7 παραγόμενες συναρτήσεις και μία η αρχική Carlet-Feng) ενώ ο αριθμός των συναρτήσεων bent που χρησιμοποιούνται (επίσης 8, ώστε να υπάρχει ίδιο πλήθος μεταξύ των δύο κατηγοριών) δηλώνεται μέσω της μεταβλητής bents_to_chose.

Επίσης, στις 2 μεθοδολογίες που δεν περιλαμβάνουν bent δεν χρησιμοποιούνται προφανώς οι μεταβλητές που σχετίζονται άμεσα με αυτές (n_bents, bents_to_chose).

Στην μεθοδολογία αναζήτησης αποκλειστικά με τυχαίες συναρτήσεις δεν χρησιμοποιείται και η εκάστοτε Carlet-Feng.

Είναι λογικό να μην συμμετέχουν όλες οι μεταβλητές παντού αφού όντως δεν χρησιμοποιούνται παντού το ίδιο.

Κεφάλαιο 6
Αποτελέσματα

Για την παραγωγή των αποτελεσμάτων ακολουθούνται οι διαδικασίες που περιεγράφηκαν στο κεφάλαιο 5, υποενότητες 5.2.5, 5.2.6 και ενότητα 5.3.

Γίνονται εκτελέσεις για συναρτήσεις μήκους 256 (n=8 μεταβλητές), 1024 (n=10 μεταβλητές), και 2048 (n=11 μεταβλητές) bits. Για κάθε διαφορετικό μήκος συναρτήσεων εκτελούνται και οι 3 διαφορετικές μεθοδολογίες που αναφέρονται στο κεφάλαιο 5. Ωστόσο, λόγω εγγενών περιορισμών που άπτονται του διαθέσιμου υλικού (βλ. κεφάλαιο 7 όπου επεξηγούνται οι περιορισμοί της έρευνας), έχουν ολοκληρωθεί, για την περίπτωση όπου n=11 μεταβλητές μερικώς μόνο οι εκτελέσεις που αφορούν αναζήτηση με αποκλειστικά με τυχαίες συναρτήσεις, ενώ δεν ήταν εφικτό να ολοκληρωθούν και οι εκτελέσεις συναρτήσεων bent με Carlet-Feng και τις παράγωγες συναρτήσεις.

Όπως αναφέρθηκε και στην προηγούμενη υποενότητα 5.2.6 («main.sage»), τα αποτελέσματα αποθηκεύονται σε ένα αρχείο τύπου csv με το καλύτερο άτομο και τα χαρακτηριστικά του από κάθε γενιά ανά εκτέλεση. Στο τέλος της εκάστοτε εκτέλεσης καταγράφονται και οι σχετικοί παράμετροι που δόθηκαν σαν είσοδοι στο πρόγραμμα ώστε να προκύψουν τα ανωτέρω αποτελέσματα.

Ο γενικός στόχος είναι να επιβεβαιωθεί πως με το πέρασμα των γενεών αυξάνεται η τιμή καταλληλότητας του καλύτερου ατόμου. Σε όποια μέθοδο συμμετέχουν οι συναρτήσεις Carlet-Feng και οι παράγωγές τους, στοχεύουμε να υπάρχει ίση ή καλύτερη τιμή καταλληλότητας από την αρχική Carlet-Feng συνάρτηση.

Αυτό το κεφάλαιο παραθέτει τους ελάχιστους και μέγιστους χρόνους μίας εκτέλεσης του προγράμματος ανά τον κάθε αριθμό μεταβλητών των συναρτήσεων καθώς και ανά την

εκάστοτε μεθοδολογία. Στην συνέχεια, αναγράφονται σε πίνακα οι συναρτήσεις Carlet-Feng και οι παράγωγές τους, όπως αυτές προκύπτουν μετά από την ανταλλαγή θέσεων bits, όπως περιεγράφηκε στο κεφάλαιο 5, υποενότητα 5.2.2, με τα αντίστοιχα χαρακτηριστικά της η κάθε μία. Ακολουθούν τα καλύτερα άτομα που παράχθηκαν ανά αριθμό μεταβλητών των συναρτήσεων καθώς και ανά μεθοδολογία. Τέλος, για την καλύτερη απεικόνιση των αποτελεσμάτων και τον σαφή προσδιορισμό των τάσεων που εμφανίζονται στα δεδομένα, χρησιμοποιήθηκαν διαγράμματα. Τα διαγράμματα είναι 2-plotted των τιμών των γενεών και των βαθμών καταλληλότητας του καλύτερου ατόμου στην εκάστοτε γενιά. Οι ακολουθίες των bits παραθέτονται ως hex στοιχεία, λόγω της μεγάλης έκτασης που θα καταλάμβαναν αν γραφόταν σαν ακολουθία bits.

6.1 Χρόνοι εκτελέσεων

Παρακάτω παρατίθενται ενδεικτικά οι ελάχιστοι, μέγιστοι και συνολικοί χρόνοι εκτελέσεων του προγράμματος για τα διάφορα μήκη πινάκων αληθείας, ανά την εκάστοτε μεθοδολογία.

	Τυχαίες συναρτήσεις/ Μέσος όρος ανά εκτέλεση	Carlet-Feng και παράγωγες με τυχαίες συναρτήσεις/ Μέσος όρος ανά εκτέλεση	Συναρτήσεις Bent με Carlet-Feng και παράγωγες/ Μέσος όρος ανά εκτέλεση	Συνολικός χρόνος εκτέλεσης για τυχαίες συναρτήσεις/ Πλήθος εκτελέσεων	Συνολικός χρόνος εκτέλεσης για Carlet-Feng και παράγωγες με τυχαίες συναρτήσεις/Πλήθος εκτελέσεων	Συνολικός χρόνος εκτέλεσης για συναρτήσεις Bent με Carlet-Feng και παράγωγες /Πλήθος εκτελέσεων
256 bits(n=8 μεταβλητές)	158.14 seconds - 254.46 seconds/ 2 minutes 51 seconds	156.36 seconds- 242.79 seconds/ 2 minutes 51 seconds	1425.23 seconds - 5264.18 seconds/ 46 minutes 54 seconds	16 hours 14 minutes/341 εκτελέσεις	16 hours 14 minutes/341 εκτελέσεις	7 hours 49 minutes/10 εκτελέσεις
1024 bits(n=10 μεταβλητές)	1908.16 seconds - 2675.17 seconds/ 35 minutes 11 seconds	2047.33 seconds - 2641.57 seconds/ 36 minutes 46 seconds	4569.3 seconds - 5905.53 seconds/ 88 minutes 46 seconds	140 hours 46 minutes/240 εκτελέσεις	148 hours 20 minutes/242 εκτελέσεις	13 hours 19 minutes/9 εκτελέσεις
2048 bits(n=11 μεταβλητές)	5028.7 seconds - 6804.35 seconds/92 minutes 48 seconds	4810.73 seconds - 8419.94 seconds/92 minutes 14 seconds		30 hours 56 minutes/20 εκτελέσεις - ελλιπής	287 hours 30 minutes/187 εκτελέσεις	

Πίνακας 9. Χρόνοι & αριθμός εκτελέσεων ανά κατηγορία.

Από τον παραπάνω πίνακα αμέσως παρατηρεί κανείς πως, όπως εξάλλου αναμενόταν, οι χρόνοι εκτέλεσης αυξάνονται κατακόρυφα όσο μεγαλώνει ο αριθμός bits των συναρτήσεων. Συνεπώς επαληθεύεται και με δεδομένα πλέον ότι αυτές οι δοκιμές είναι αρκετά χρονοβόρες.

6.2 Carlet-Feng και παράγωγες συναρτήσεις

Οι αρχικές αυτές συναρτήσεις είναι σταθερές και συγκεκριμένες καθόλη την διάρκεια των εκτελέσεων. Είναι δηλαδή γνωστές εκ των προτέρων και δεν είναι τυχαίες ή δεν είναι άτομα που προκύπτουν ως απόγονοι. Χρησιμοποιούνται σε δύο εκ των τριών μεθόδων αναζήτησης. Για κάθε δεδομένο n υπάρχουν συνολικά 8 συναρτήσεις.

	Algebraic Degree	Fitness	Balanced	Nonlinearity	Algebraic Immunity	Απόσταση Hamming από την αρχική συνάρτηση Carlet-Feng
256 bits(n=8 μεταβλητές)	7	16.2	TRUE	112	4	Πρωτότυπη
	7	16.2	TRUE	112	4	56
	7	15.8	TRUE	108	4	34
	7	15.8	TRUE	108	4	30
	7	16	TRUE	110	4	12
	7	15.6	TRUE	106	4	18
	7	15.8	TRUE	108	4	14
	7	15.4	TRUE	104	4	24
	Algebraic Degree	Fitness	Balanced	Nonlinearity	Algebraic Immunity	
1024 bits(n=10 μεταβλητές)	9	52.8	TRUE	478	5	Πρωτότυπη
	9	52.8	TRUE	478	5	250
	9	51.2	TRUE	462	5	118
	9	51.2	TRUE	462	5	136
	9	51.6	TRUE	466	5	56
	9	52.0	TRUE	470	5	76
	9	52.8	TRUE	468	5	68

	9	51.6	TRUE	466	5	60
2048 bits(n=11 μεταβλητές)	Algebraic Degree	Fitness	Balanced	Nonlinearity	Algebraic Immunity	
	10	103.0	TRUE	980	6	Πρωτότυπη
	10	103.0	TRUE	980	6	570
	10	99.8	TRUE	948	6	230
	10	99.8	TRUE	948	6	298
	10	100.8	TRUE	958	6	138
	10	99.8	TRUE	948	5	134
	10	100.4	TRUE	954	5	124
	10	100.2	TRUE	952	5	128

Πίνακας 10. Χαρακτηριστικά των αρχικών Carlet-Feng και των παραγώγων συναρτήσεών τους.

Αρχική Carlet-Feng 256 bits (n=8 μεταβλητών)
FEADD8A7B7859C3A9A2A8037C7F14F89C3C919CDD0014E7EE566BA4361AE8086

Πίνακας 11. Δεκαεξαδική αναπαράσταση της αρχικής Carlet-Feng συνάρτησης n=8 μεταβλητών.

Αρχική Carlet-Feng 1024 bits (n=10 μεταβλητών)
FEF9FAD7AF9CF62EC9EAD6B5AA2D0DECB5D2A9DCA268CB22D8C84CB351E6BDA 09E57A74D9C82A3E1DD1838D0B4CF1909F6C0B09531F5DF5E3602BC688BB29C5 1D6FC632E897E30A3C7E180098C5AAD07F6B306914FC4B6158E74F5EF068641D6 EE29E1149B10C2335E52AB76E2BF22E91A694558CEE1789094CB9A4C96A42602

Πίνακας 12. Δεκαεξαδική αναπαράσταση της αρχικής Carlet-Feng συνάρτησης n=10 μεταβλητών.

Αρχική Carlet-Feng 2048 bits (n=11 μεταβλητών)

ECE4FC65AEA42972D9A9DD255DD27A1CE692C9D3E3B65D2223A3A35C3EDD43B 4FC39861DB0D2A24ABD0F9B3D62E21D194C4ACD5B894A66B15BBDE2A2640BCE 71BFA14ED6C56C03B6CA11A258DD1C64CD8FF241EF978A0FA63919ED5C16B743 8621F435D9A0F336CAC58675D97879CF0223DBDAA6AD4DD94D783000DEA4B87 A078ABE884374FCF339A16769F0444B9F69B58D1302DC5D22D5A2E742A57D20A 4A7D4ABEA492503F9AEC36ED0C855EB9D684BD70792E8F737E04738CE7E310B8 13C0906BB754E63A6D6DC50BF0F0E2CF08CB426C4786A32F6D26FC12FD2E1AB5 449485FA68BB3DC883898A321B3E6C364B62F845B011445A6A9C8749E842B9851 7F

Πίνακας 13. Δεκαεξαδική αναπαράσταση της αρχικής Carlet-Feng συνάρτησης n=11 μεταβλητών.

Αξίζει να σημειωθεί ότι αυτή η οικογένεια συναρτήσεων έχει καλή τιμή nonlinearity και είναι πάντα balanced.

6.3 Συναρτήσεις καταλληλότητας

Η συνάρτηση καταλληλότητας (fitness function) που χρησιμοποιήθηκε είναι μια απλή πρόσθεση βεβαρημένων μεταβλητών. Ο συντελεστής βάρους της μη γραμμικότητας είναι πάντα 0.1.

Για τις μεθοδολογίες παραγωγής τυχαίων συναρτήσεων και τυχαίων συναρτήσεων με Carlet-Feng η τιμή του συντελεστή βάρους αν η συνάρτηση είναι balanced (balance_weight) είναι 4 ή 5. Στην ουσία, σε όσες συναρτήσεις είναι balanced δίνεται ένα μικρό προβάδισμα, τόσο ώστε ελαφρώς αλλά και σαφώς να προηγούνται από αυτές που δεν είναι. Όπως ειπώθηκε στο κεφάλαιο 2, είναι σημαντικό οι κρυπτογραφικές συναρτήσεις να είναι balanced. Δίνοντας ένα μικρό προβάδισμα σε αυτές, ο αλγόριθμος καταλαβαίνει ότι είναι προτιμώμενες, όμως δεν έχουν τόσο μεγάλη διαφορά στην τιμή καταλληλότητας από τις μη ισορροπημένες. Αυτό έχει ως αποτέλεσμα να συμμετέχουν και να επιλέγονται για την διαδικασία αναπαραγωγής και οι μη ισορροπημένες συναρτήσεις, εξασφαλίζοντας έτσι την γενετική ποικιλότητα του πληθυσμού(genetic

diversity). Συνεπώς, με αυτόν τον τρόπο δίνεται μία μικρή ώθηση στον αλγόριθμο προς τη ζητούμενη κατεύθυνση, χωρίς όμως να περιορίζεται στην επιλογή γονέων προς αναπαραγωγή.

balance_weight*is_balanced+0.1*nonlinearity

Για την μέθοδο με συναρτήσεις bent και Carlet-Feng η κατάσταση είναι λίγο διαφορετική. Οι συναρτήσεις bent είναι πάντα μη ισορροπημένες αλλά έχουν επίσης πάντα πολύ υψηλό nonlinearity έχουν και πολύ καλή τιμή καταλληλότητας. Στον αντίποδα αυτού, οι Carlet-Feng είναι πάντα balanced, αλλά η μη γραμμικότητά τους απέχει πολύ από αυτήν των bent.

Αποτέλεσμα των παραπάνω είναι η τιμή καταλληλότητας των bent να έχει τεράστια διαφορά από αυτήν των Carlet-Feng. Για να αντιμετωπιστεί αυτό, προσαρμόστηκε το balance_weight, ώστε πάλι να δώσει ένα μικρό, οριακό προβάδισμα στις ισορροπημένες συναρτήσεις έναντι των μη ισορροπημένων bent. Η τιμή προσαρμόζεται έτσι, ώστε οι ισορροπημένες συναρτήσεις να προηγούνται κατά λίγες μόλις μονάδες καταλληλότητας.

Ένα ακόμα εξίσου σημαντικό δεδομένο, είναι ότι η μη γραμμικότητα των bent παρουσιάζει μεγάλες διαφορές ανάλογα τον αριθμό μεταβλητών τους.

Αυτό σημαίνει ότι το balance_weight επίσης αναπροσαρμόζεται ανά αριθμό μεταβλητών, ώστε οι ισορροπημένες συναρτήσεις να εξακολουθούν να προηγούνται οριακά, όπως αναφέρθηκε.

Άρα στην ουσία, προκειμένου να επιτευχθεί η παραπάνω συνθήκη ανά κατηγορία αριθμού μεταβλητών(8,10,11), το balanced_weight προσαρμόζεται ως εξής:

Bent fitness - Carlet-Feng fitness = balanced_weight + την διαφορά που θέλουμε να υπάρχει.

6.4 Συναρτήσεις που παράχθηκαν

Οι συγκεκριμένες τιμές των παραμέτρων που δίνονται σε κάθε μεμονωμένη εκτέλεση βρίσκονται αναλυτικά στα παραρτήματα της παρούσας διατριβής.

Έπειτα από τις παραπάνω δοκιμές, παρουσιάζονται τα χαρακτηριστικά των καλύτερων συναρτήσεων που προέκυψαν ανά αριθμό μεταβλητών συνάρτησης καθώς και ανά μεθοδολογία.

Αξίζει να σημειωθεί πως η κορυφαία συνάρτηση σε κάθε κατηγορία δεν είναι μοναδική. Παρουσιάζονται πολλαπλές συναρτήσεις με τα ίδια ακριβώς χαρακτηριστικά.

6.4.1 256 bits μήκος πίνακα αληθείας (n=8 μεταβλητές)

Algebraic Degree	Fitness	Balanced	Nonlinearity	Algebraic Immunity
7	15,0	TRUE	110	4

Πίνακας 14. Καλύτερη συνάρτηση που παράχθηκε από χρήση αποκλειστικά τυχαίων συναρτήσεων.

Hex Representation
C3B39517F682AAA5BAAD7A1E6682C2D9A1E0B5DF5C4756F7440A828C63E9C90D

Πίνακας 15. Δεκαεξαδική αναπαράσταση της καλύτερης συνάρτησης με χρήση αποκλειστικά τυχαίων συναρτήσεων.

Algebraic Degree	Fitness	Balanced	Nonlinearity	Algebraic Immunity
7	16.2	TRUE	112	4

Πίνακας 16. Καλύτερη συνάρτηση που παράχθηκε από χρήση τυχαίων συναρτήσεων μαζί με Carlet-Feng.

Hex Representation
FEADD8A7B7859C3A9A638437C7F14F89C1C951CDD0014E7EE574BB4161A68086

Πίνακας 17. Δεκαεξαδική αναπαράσταση της καλύτερης συνάρτησης με χρήση τυχαίων συναρτήσεων μαζί με Carlet-Feng.

Algebraic Degree	Fitness	Balanced	Nonlinearity	Algebraic Immunity
7	195.2	TRUE	112	4

Πίνακας 18. Καλύτερη συνάρτηση που παράχθηκε από χρήση συναρτήσεων Bent με Carlet-Feng.

Hex Representation
FEADD8A7B7859C3A9A2A8037C7F14F89C3C919CDD0014E7EE566BA4361AE8086

Πίνακας 19. Δεκαεξαδική αναπαράσταση της καλύτερης συνάρτησης με χρήση συναρτήσεων Bent με Carlet-Feng.

6.4.2 1024 bits μήκος πίνακα αληθείας (n=10 μεταβλητές)

Algebraic Degree	Fitness	Balanced	Nonlinearity	Algebraic Immunity
9	51.0	TRUE	470	5

Πίνακας 20. Καλύτερη συνάρτηση που παράχθηκε από χρήση αποκλειστικά τυχαίων συναρτήσεων.

Hex Representation

BEC8295F9709491C4C5E524E96C2CA27F5A3882124FE4A353A9F2A3C768083BAE4
5540E909EEAE2B204552A455E7860446FAFA7E2B392A73D3B38249BF8A4711AEF
60594FDEF2A6AA23C99020E28494C8B6B7C8ABC63C9054548A3ED5969E65B66F9
580916FCF8ABD337EE41A563DC2BB7A5E24EEE2BEC4A7C9692A353F33BBE

Πίνακας 21. Δεκαεξαδική αναπαράσταση της καλύτερης συνάρτησης με χρήση αποκλειστικά τυχαίων συναρτήσεων.

Algebraic Degree	Fitness	Balanced	Nonlinearity	Algebraic Immunity
9	52.2	TRUE	472	5

Πίνακας 22. Καλύτερη συνάρτηση που παράχθηκε από χρήση τυχαίων συναρτήσεων μαζί με Carlet-Feng.

Hex Representation
FEF9FAFFAF1CF74ED9EA9EB5A24D45FCA512A9BAF068CB22D8A849B351E6FDA39
65F874D8CB2B3C1CF0A58D0B4ED0929FEC0B28531BDDF5E3622BC688B329C51B6
FC7126897632B1C7E188498C4AAD07B6B306814FC4B6158E54F56D06862196EE29
E1149B10A4575E3ABB76F2DF20E91A694548CEA9789494C9984C96A4220A

Πίνακας 23. Δεκαεξαδική αναπαράσταση της καλύτερης συνάρτησης με χρήση τυχαίων συναρτήσεων μαζί με Carlet-Feng.

Algebraic Degree	Fitness	Balanced	Nonlinearity	Algebraic Immunity
9	202.8	TRUE	478	5

Πίνακας 24. Καλύτερη συνάρτηση που παράχθηκε από χρήση συναρτήσεων Bent με Carlet-Feng.

Hex Representation

FEF9FAD7AF9CF62EC9EAD6B5AA2D0DECB5D2A9DCA268CB22D8C84CB351E6BDA0
9E57A74D9C82A3E1DD1838D0B4CF1909F6C0B09531F5DF5E3602BC688BB29C51D
6FC632E897E30A3C7E180098C5AAD07F6B306914FC4B6158E74F5EF068641D6EE2
9E1149B10C2335E52AB76E2BF22E91A694558CEE1789094CB9A4C96A42602

Πίνακας 25. Δεκαεξαδική αναπαράσταση της καλύτερης συνάρτησης με χρήση συναρτήσεων Bent με Carlet-Feng.

6.4.3 2048 bits μήκος πίνακα αληθείας (n=11 μεταβλητές)

Algebraic Degree	Fitness	Balanced	Nonlinearity	Algebraic Immunity
10	100.6	TRUE	956	5

Πίνακας 26. Καλύτερη συνάρτηση που παράχθηκε από χρήση αποκλειστικά τυχαίων συναρτήσεων.

Hex Representation
B1EABA629BBDE54E2491D4B88646F1EE61CD42DFD48B36DFC80B3846927376C93
01B212B2B24A3876F87B5DB1CAE4A6CAF57816C8C2F621DBC29E1754312379CF5
9A789837611538D8F32E3C767C21FA30D10EDF2800822449EFC125E43B1AC8FFED
805DDC5BBB8156E6CE31E8145BFD678B145896B549B59CE3F65AC38FB4F36285C
C8D4DABCEB975D1159F68C7E4141E517278DCA188A368386439D74ED340855040
B597406FD6D81001BBFA8A7E618BF0745451DC1DEA52B9CC3EB4078E82FB4F015
4356F8AC4041FB6079294D9F0E487F3F16B5B35491C9EAD79C5EFE760B258C5283
459C7A69D12BC7AE0A9F6CFAB994662047557ED2FC2D36DC3928E14

Πίνακας 27. Δεκαεξαδική αναπαράσταση της καλύτερης συνάρτησης με χρήση αποκλειστικά τυχαίων συναρτήσεων.

Algebraic Degree	Fitness	Balanced	Nonlinearity	Algebraic Immunity
10	101.2	TRUE	962	6

Πίνακας 28. Καλύτερη συνάρτηση που παράχθηκε από χρήση τυχαίων συναρτήσεων μαζί με Carlet-Feng.

Hex Representation
ECE4BE65BCA44972DBA19C275BD26AFCA69AC9D3E1B65C2227A3C33C2EBF4196F
C39861DB0D0E24AD71D9B3572C015516C4CCD7B810866911FBFE2C2240BCE39BF
A17E96C5EC05B6CA01A258CD3C64CD8BFC21EFD78E1726510BED5C16B5438621D
A255BA0D336CAC58E7D8B7879CF0A23DBDFA4A54DCD4D780040DCE4987807CA9
E884374FCF33BA16778E0064F9F69B58F4322DC5D22D5A2EF02A57D00A5B5F5AF
EE492311E9A6CD6E9AE83DE99D684BF70793F8F737600518CE7E312D913C0906B
B732E21B6D4DA5AB7070E2CF28CA6268CF86A32A6D06ED16BDAE1AB1C49495FA
789B3DE8830998125B3E6C364B62B865B011445E6A9D83C9A844A9A517F

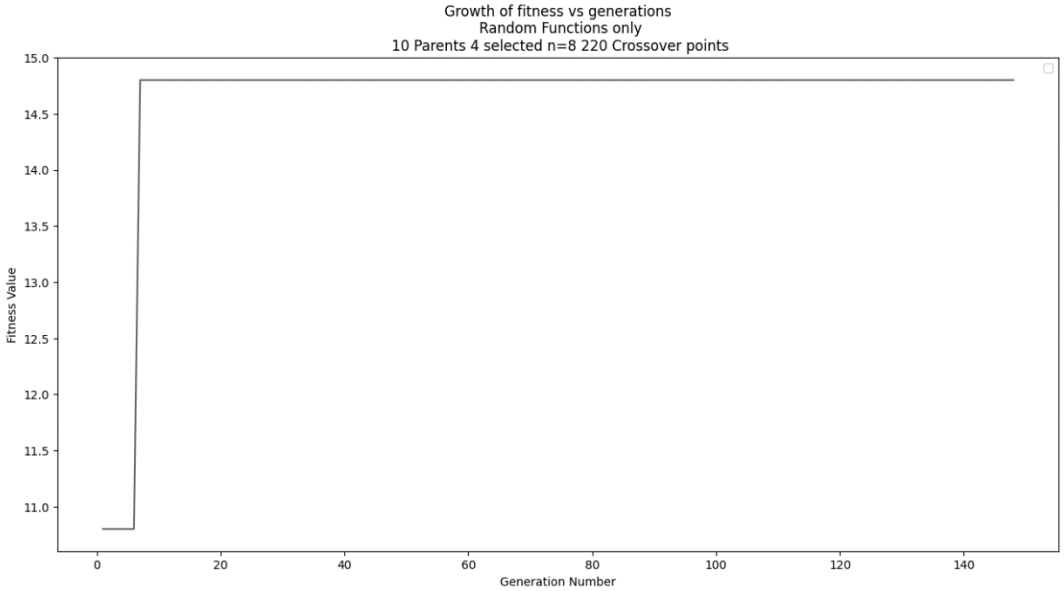
Πίνακας 29. Δεκαεξαδική αναπαράσταση της καλύτερης συνάρτησης με χρήση τυχαίων συναρτήσεων μαζί με Carlet-Feng.

6.5 Διαγράμματα

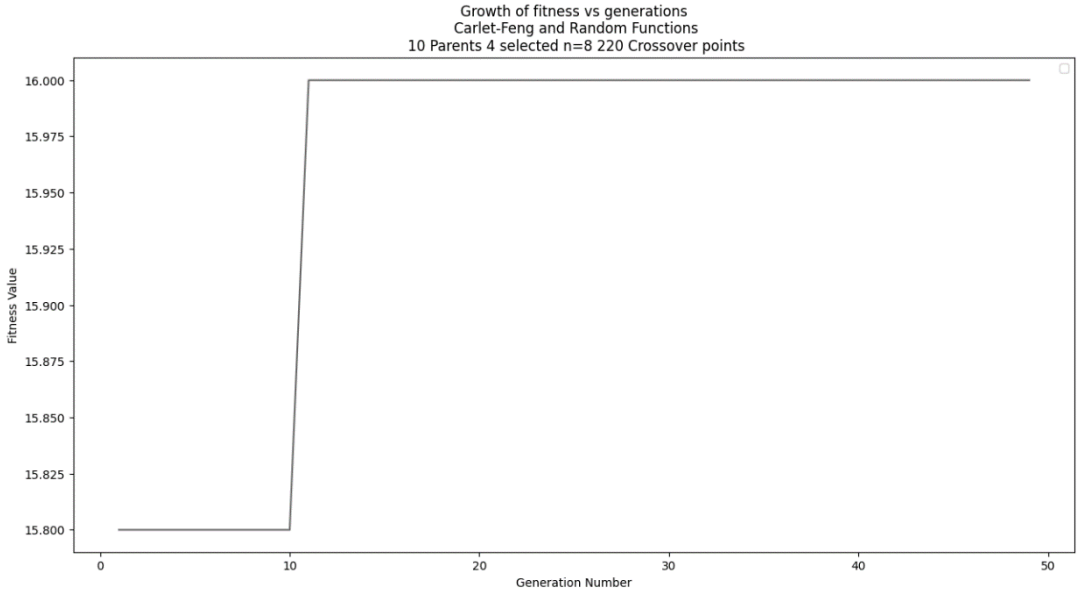
Τα παρακάτω διαγράμματα παρέχουν μία ξεκάθαρη εικόνα για την συνολική κίνηση του βαθμού καταλληλότητας στο πέρασμα των γενεών.

Για την παραγωγή των διαγραμμάτων αναπτύχθηκε ένα μικρό python script, το charter.py.

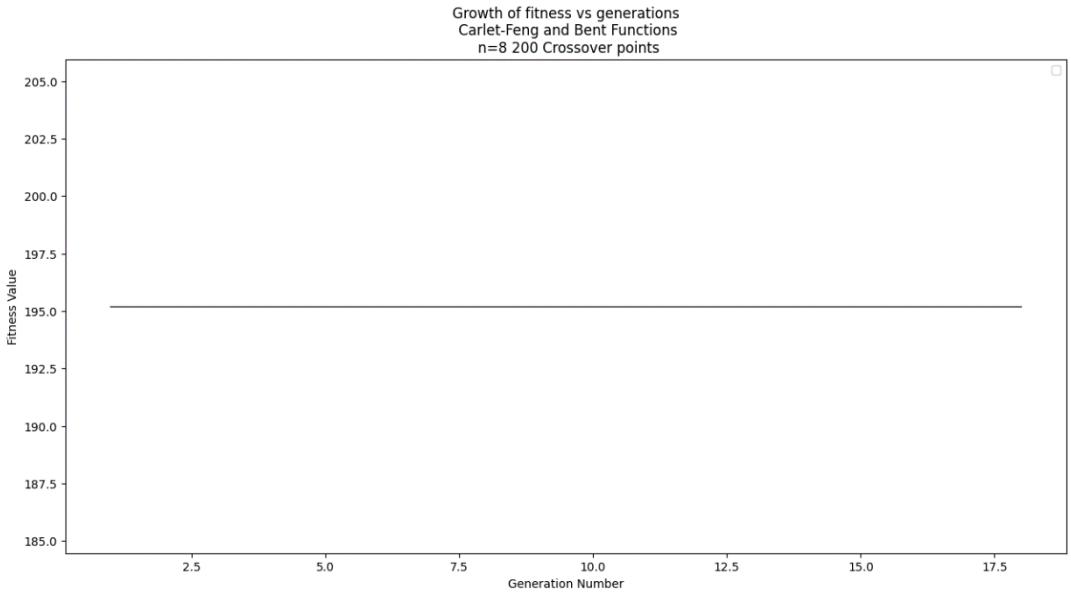
6.5.1 Διαγράμματα για 256 bits μήκος πίνακα αληθείας (n=8 μεταβλητές)



Διάγραμμα 1. Τυχαίες συναρτήσεις n=8 μεταβλητών, 10 γονέων και 220 σημείων crossover.

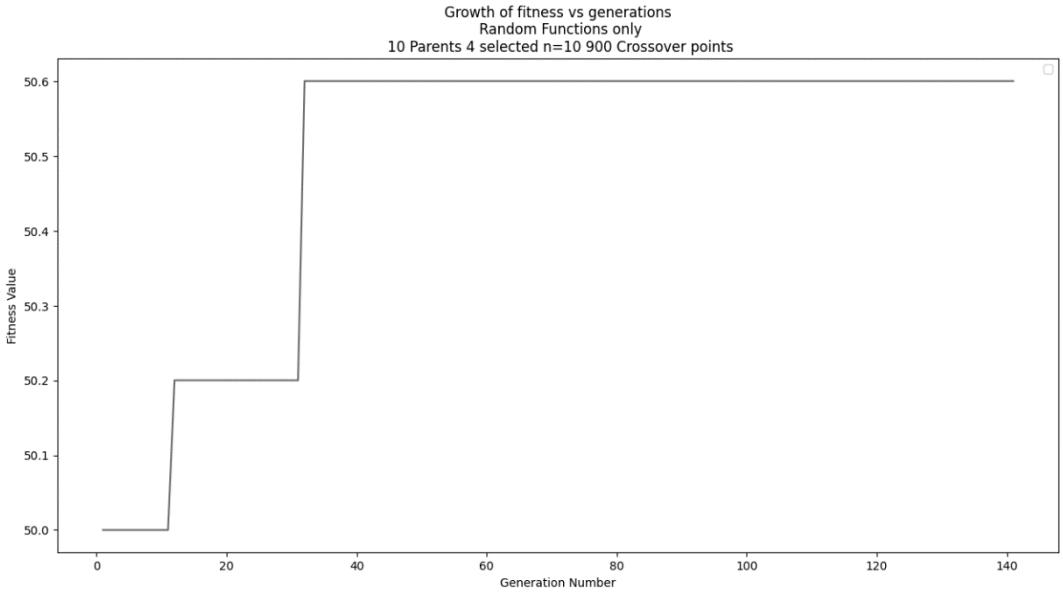


Διάγραμμα 2. Carlet-Feng με τυχαίες συναρτήσεις n=8 μεταβλητών, 10 γονέων και 220 σημείων crossover.

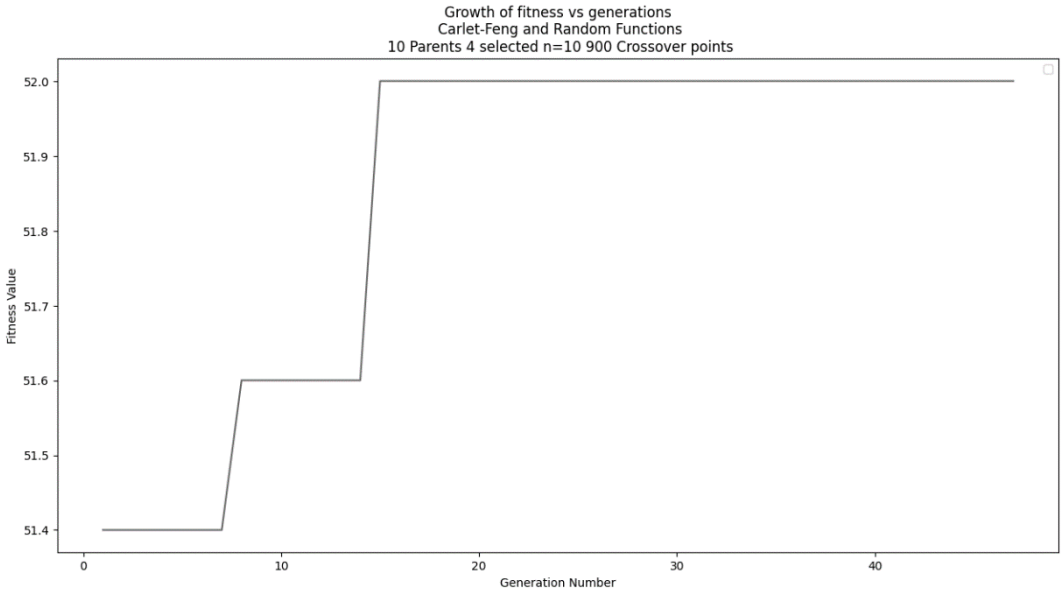


Διάγραμμα 3. Carlet-Feng με bent συναρτήσεις n=8 μεταβλητών και 200 σημείων crossover.

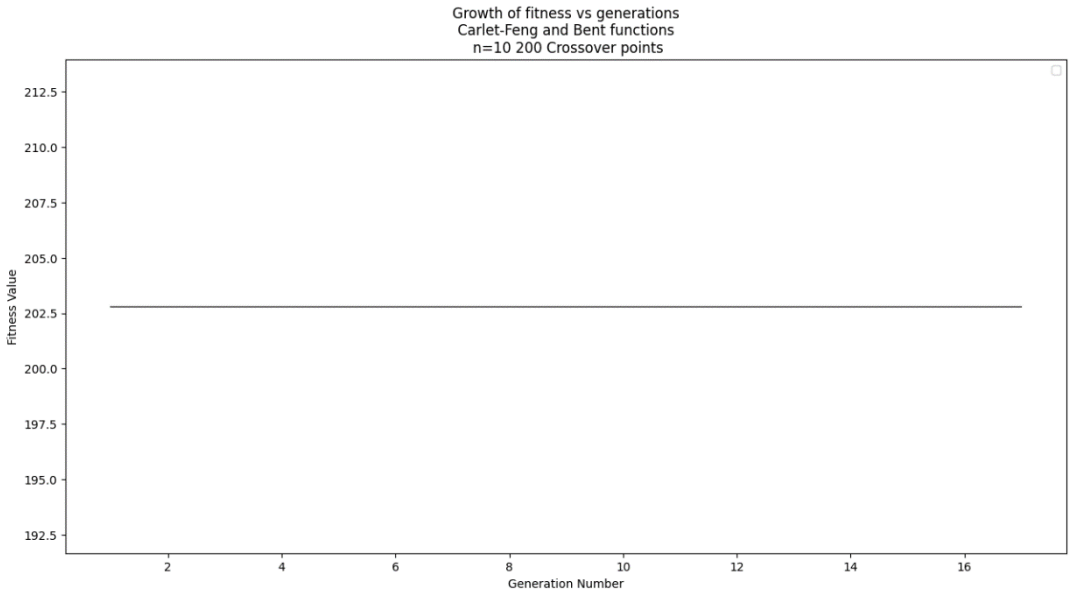
6.5.2 Διαγράμματα για 1024 bits μήκος πίνακα αληθείας (n=10 μεταβλητές)



Διάγραμμα 4. Τυχαίες συναρτήσεις n=10 μεταβλητών, 10 γονέων και 900 σημείων crossover.

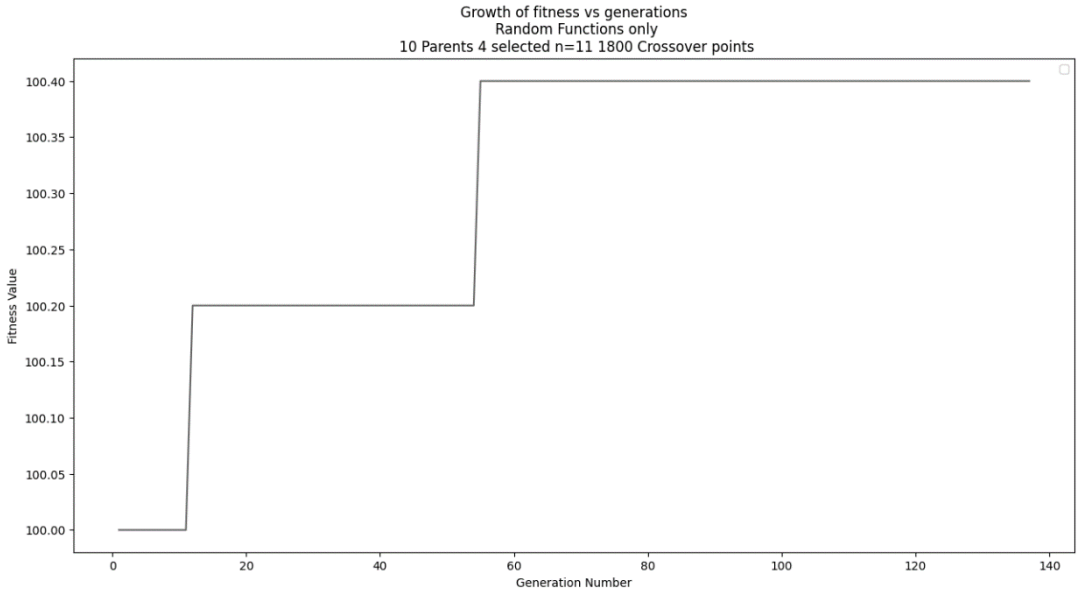


Διάγραμμα 5. Carlet-Feng με τυχαίες συναρτήσεις n=10 μεταβλητών, 10 γονέων και 900 σημείων crossover.

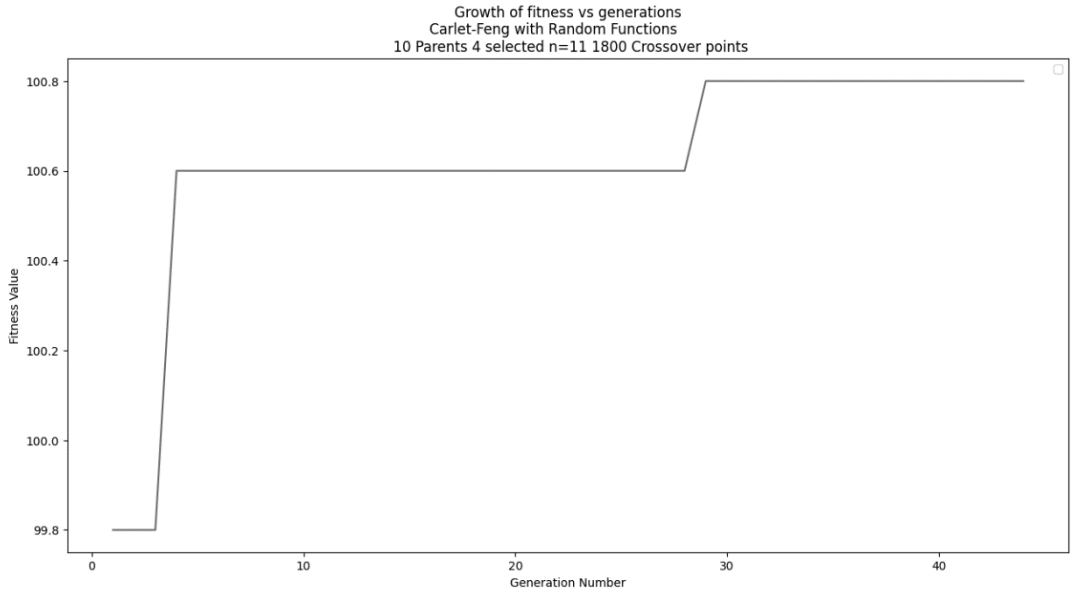


Διάγραμμα 6. Carlet-Feng με bent συναρτήσεις n=10 μεταβλητών και 200 σημείων crossover.

6.5.3 Διαγράμματα για 2048 bits μήκος πίνακα αληθείας (n=11 μεταβλητές)



Διάγραμμα 7. Τυχαίες συναρτήσεις n=11 μεταβλητών, 10 γονέων και 1800 σημείων crossover.



Διάγραμμα 8. Carlet-Feng με τυχαίες συναρτήσεις n=11 μεταβλητών, 10 γονέων και 1800 σημείων crossover.

6.6 Συμπεράσματα

Για τις συναρτήσεις n=8 μεταβλητών, καταλήγουμε σε αρκετά ικανοποιητικά αποτελέσματα. Βλέπουμε ότι τα χαρακτηριστικά των συναρτήσεων που προκύπτουν, είναι παρόμοια με αυτά της αρχικής Carlet-Feng του ίδιου αριθμού μεταβλητών. Επίσης, παρατηρούμε ότι στις μεθοδολογίες που συμμετέχουν οι Carlet-Feng και οι παράγωγες (διαγράμματα 2 & 3 και πίνακες 16 & 18) με τις παράγωγές συναρτήσεις τους, η μη γραμμικότητα που επιτυγχάνεται είναι ελαφρώς υψηλότερη από την μεθοδολογία όπου συμμετάσχουν μόνο τυχαίες συναρτήσεις στον αρχικό πληθυσμό (διάγραμμα 1 και πίνακας 14) και ίδια με αυτήν της αρχικής Carlet-Feng. Πρέπει να σημειωθεί πως η καλύτερη συνάρτηση που επιτυγχάνεται για n=8 και με μεθοδολογία Carlet-Feng και παραγώγων συνδυαστικά με τυχαίες συναρτήσεις είναι τελείως διαφορετική από την αρχική Carlet-Feng n=8 μεταβλητών (πίνακες 16 & 17) ενώ επιτυγχάνει το ίδιο καλά χαρακτηριστικά.

Ακόμα, παρατηρείται πως στις μεθοδολογίες που συμμετέχουν οι Carlet-Feng με τις παράγωγες, ο αριθμός των γενεών είναι αρκετά μικρός σε σχέση με την μεθοδολογία όπου χρησιμοποιούνται αποκλειστικά τυχαίες συναρτήσεις. Αυτό σημαίνει πως το κατώφλι της μνήμης RAM που έχει τεθεί σαν όριο για τον τερματισμό της εκάστοτε εκτέλεσης, επιτυγχάνεται γρηγορότερα, και άρα, αυτή η μεθοδολογία είναι πιο δαπανηρή. Ωστόσο, επειδή ακριβώς πετυχαίνει υψηλότερα αποτελέσματα με μικρότερο αριθμό γενεών, μπορούμε να συμπεράνουμε ότι η μεθοδολογία των Carlet-Feng και παραγώγων με τις τυχαίες συναρτήσεις είναι η πιο αποτελεσματική.

Ομοίως και για τα τις άλλες 2 κατηγορίες συναρτήσεων (n=10, n=11). Τα χαρακτηριστικά των προκύπτοντων συναρτήσεων είναι αρκετά κοντά σε αυτά των αρχικών Carlet-Feng. Η διαφορά για n=10 και n=11 είναι ότι οι καλύτερες συναρτήσεις που προκύπτουν δεν φτάνουν τα χαρακτηριστικά των αρχικών Carlet-Feng, όπως συμβαίνει με την εκτέλεση με Carlet-Feng και τις παράγωγες συναρτήσεις τους συνδυαστικά με τυχαίες συναρτήσεις για n=8 μεταβλητές. Επιπλέον, παρατηρούμε ότι για n=11 η διαφορά στην μη γραμμικότητα των καλύτερων συναρτήσεων που βρέθηκαν (962 – πίνακας 28) έναντι της μη γραμμικότητας της αρχικής Carlet-Feng (980) είναι 18. Η αντίστοιχη διαφορά για

συναρτήσεις n=10 μεταβλητών είναι 6. Οπότε, μπορούμε να υποθέσουμε ότι όσο αυξάνεται το n, με τα παρόντα δεδομένα και περιορισμούς (υλικό, αριθμός γενεών) αυξάνεται και η διαφορά μεταξύ της εκάστοτε καλύτερης συνάρτησης και της αρχικής Carlet-Feng για αυτές τις δύο μεθοδολογίες.

Όσον αφορά τις εκτελέσεις με την μεθοδολογία συναρτήσεων bent με Carlet-Feng και τις παράγωγές τους, βλέπουμε ότι δεν υπάρχει καμία βελτίωση (διαγράμματα 3 & 6). Επίσης παρατηρούμε ότι είναι και η πιο υπολογιστικά δαπανηρή, αφού ο αλγόριθμος σταματά μόλις λίγο μετά την 17η γενιά. Ακόμα, ο αλγόριθμος φαίνεται όχι μόνο να μην κατάφερε να βελτιώσει, αλλά και να μην τροποποιήσει συνολικά την συνάρτηση. Τα αντίστοιχα αποτελέσματα παρουσιάζουν μία και μόνο συνάρτηση, η οποία μάλιστα είναι πανομοιότυπη με την αρχική συνάρτηση Carlet-Feng τόσο στην περίπτωση n=8 μεταβλητών, όσο και στην περίπτωση n=10 μεταβλητών.

Παρατηρούμε, ότι με βάση τα διαγράμματα, η μεθοδολογία αποκλειστικά τυχαίων συναρτήσεων είναι η υπολογιστικά πιο φθηνή, ενώ η μεθοδολογία Carlet-Feng και παραγώμενων συναρτήσεων με bent είναι η πιο δαπανηρή.

Ακόμα, φαίνεται ότι οι μεθοδολογίες τυχαίων συναρτήσεων και τυχαίων συναρτήσεων συνδυαστικά με Carlet-Feng, παρουσιάζουν αυξητική τάση με το πέρασμα των γενεών. Αυτή είναι μία πολύ θετική ένδειξη. Το γεγονός ότι ανεξαρτήτου του αριθμού μεταβλητών και της μεθοδολογίας ο βαθμός καταλληλότητας συνεχίζει να αυξάνεται, δείχνει ότι η υλοποίηση του αλγορίθμου είναι αρκετά στιβαρή. Η γενικότερη αυξητική τάση υποδεικνύει ότι πολύ πιθανόν, η καταλληλότητα και η μη γραμμικότητα των συναρτήσεων-απογόνων, μπορούν να βελτιωθούν περεταίρω. Εξαίρεση αποτελεί η μεθοδολογία που περιλαμβάνει τις bent, αφού όπως αναφέρθηκε δεν υπάρχει κάποια βελτίωση της καταλληλότητας.

Κεφάλαιο 7
Επίλογος

Στην παρούσα διπλωματική διατριβή, αναπτύχθηκε εξ αρχής ένας steady-state γενετικός αλγόριθμος, ο οποίος στόχο έχει την εύρεση και εξέλιξη κρυπτογραφικών συναρτήσεων με καλά χαρακτηριστικά. Ο αλγόριθμος αξιοποιεί 3 διαφορετικούς τρόπους αναζήτησης για μεγαλύτερη κάλυψη του εύρους των πιθανών συναρτήσεων και τον πειραματισμό ως προς το ποια μεθοδολογία δίνει τα καλύτερα αποτελέσματα. Ο αλγόριθμος αναπτύχθηκε με τη λογική του να μπορεί να διερευνήσει κατά πόσον ένας εξελικτικός αλγόριθμος μπορεί να δώσει καλά αποτελέσματα αν στην είσοδό του τεθούν, πέρα από τυχαίες συναρτήσεις, και συναρτήσεις που έχουν αποδεδειγμένα καλά (κάποια) κρυπτογραφικά χαρακτηριστικά. Για αυτό, δύναται να δημιουργήσει τόσο εντελώς τυχαίες συναρτήσεις δεδομένου του μήκους πίνακα αληθείας τους όσο και πολλαπλές συναρτήσεις bent (που επιτυγχάνουν τη μέγιστη μη γραμμικότητα), δεδομένου του αριθμού n των μεταβλητών τους. Ακόμα, δεδομένης μίας αρχικής συνάρτησης Carlet-Feng, δύναται να δημιουργήσει παρόμοιες συναρτήσεις με παρόμοια κρυπτογραφικά χαρακτηριστικά, για το δεδομένο μήκος πίνακα αληθείας της αρχικής.

Επιγραμματικά, βλέπουμε ότι οι προκύπτουσες συναρτήσεις είναι αρκετά κοντά στην αρχική Carlet-Feng, όσον αφορά τα χαρακτηριστικά τους. Επιπλέον, η μεθοδολογία αναζήτησης Carlet-Feng και παράγωγων συναρτήσεων με τυχαίες συναρτήσεις φαίνεται να είναι η πιο αποδοτική, επιτυγχάνοντας υψηλότερη μη γραμμικότητα από αυτήν της αναζήτησης αποκλειστικά τυχαίων συναρτήσεων σε μικρότερο αριθμό γενεών. Όσον αφορά την μεθοδολογία με συναρτήσεις bent και Carlet-Feng και των παραγώμενων συναρτήσεων, φαίνεται να είναι αρκετά δαπανηρή σε σχέση με τις άλλες δύο, αφού τερματίζει την εκτέλεση σε με αρκετά μικρό αριθμό γενεών. Επίσης, δεν παρέχει και κάποια ουσιαστική βελτίωση και τροποποίηση στις υποκείμενες συναρτήσεις.

7.1 Περιορισμοί της έρευνας και αντιμετώπισή τους

Όπως αναφέρθηκε ήδη, η αρχική ιδέα ήτανε η κάθε εκτέλεση του κυρίως προγράμματος να συνεχίζει για δεδομένο συνολικό αριθμό γενεών. Ωστόσο, παρατηρήθηκε ότι η μνήμη ram του μηχανήματος στο οποίο εκτελέστηκαν οι υπολογισμοί «γέμιζε» σε αρκετές περιπτώσεις πολύ πριν επιτευχθεί ο δεδομένος αριθμός γενεών. Έγιναν κάποιες προσπάθειες ώστε να βελτιστοποιηθεί το πρόγραμμα, χωρίς όμως να υπάρχει αποτέλεσμα. Σε αυτές, περιλαμβάνεται η διαγραφή των sage objects μετά τον υπολογισμό των απαραίτητων χαρακτηριστικών και την δημιουργία των objects του προγράμματος της κάθε συνάρτησης, σε μία προσπάθεια απελευθέρωσης μνήμης RAM. Επίσης, έγινε προσπάθεια μέτρησης της κατανάλωσης μνήμης RAM με το εργαλείο memory-profiler στα functions objectify της genesis_sage.py και kpc της darwin.py. Συνεπώς, λόγω υλικών περιορισμών, αναγκαστικά, η συνθήκη τερματισμού εκτέλεσης του αλγορίθμου έπρεπε να τροποποιηθεί. Έτσι, το πρόγραμμα τροποποιήθηκε ώστε να σταματά η εκτέλεσή του κατά την επίτευξη ενός δεδομένου ποσοστού χρήσης της RAM.

Άξιο αναφοράς είναι και το γεγονός ότι τα προβλήματα με την μνήμη εμφανίστηκαν μόλις άρχισε να χρησιμοποιείται το λογισμικό SageMath για τον υπολογισμό των τιμών των χαρακτηριστικών των συναρτήσεων. Ως εκ τούτου εξάγεται το συμπέρασμα ότι πιθανότατα, η μνήμη RAM γέμιζε λόγω της χρήσης του εν λόγω λογισμικού. Αυτό, διότι το μεγαλύτερο και «βαρύτερο» μέρος του προγράμματος αναπτύχθηκε χωρίς την χρήστη του λογισμικού SageMath. Αρχικά δηλαδή, το πρόγραμμα δημιουργούσε τυχαίες συναρτήσεις και επιτελούσε τις ίδιες λειτουργίες της επιλογής, crossover και mutation (έχοντας βέβαια τυχαίες τιμές καταλληλότητας) πάνω στους πίνακες αληθείας των συναρτήσεων χωρίς την χρήση του SageMath. Για πολύ μεγάλο αριθμό παραμέτρων (δέκα χιλιάδες γονείς, bits, crossover points και επιλεγόμενους γονείς) το πρόγραμμα λειτούργησε κανονικά.

Μια άλλη σημαντική παράμετρος η οποία δεν υπήρξε σύμμαχος είναι ο χρόνος. Όπως είναι κατανοητό, με την χρήση του automation.py δύναται να παραχθεί τεράστια ποσότητα δεδομένων. Ωστόσο, από την στιγμή που άρχισαν να υπολογίζονται τα χαρακτηριστικά της εκάστοτε συνάρτησης (και άρα να παράγονται αξιοποιήσιμα δεδομένα) αυξήθηκε ο χρόνος υπολογισμού/κατασκευής της εκάστοτε συνάρτησης με

τα χαρακτηριστικά της. Μάλιστα, όσο μεγαλύτερο είναι το πλήθος των μεταβλητών (δηλαδή όσο περισσότερα bits αποτελούν τον πίνακα αληθείας της εκάστοτε συνάρτησης), τόσο περισσότερος χρόνος χρειάζεται για τον υπολογισμό των χαρακτηριστικών της. Φυσικά, οι χρόνοι κατασκευής δεν είναι ακριβώς ίδιοι ακόμα και όταν γίνεται λόγος για συναρτήσεις με ίδιο μήκος πίνακα αληθείας. Οι διαφορές μεταξύ τους ωστόσο, είναι αμελητέες.

Στο ανωτέρω πλαίσιο, έγινε μία καθολική προσπάθεια βελτιστοποίησης της αξιοποίησης του χρόνου. Δηλαδή, ξεκίνησε η αναζήτηση συναρτήσεων (με την μέθοδο αναζήτησης συναρτήσεων με τυχαίες αρχικές συναρτήσεις) μόλις γράφτηκαν τα απολύτως απαραίτητα κομμάτια κώδικα/functions, ενώ ταυτόχρονα συνεχιζόταν η ανάπτυξη functions για την αναζήτηση με τις άλλες μεθοδολογίες (Bent, Carlet-Feng).

Το ιδανικό θα ήταν για κάθε διαφορετικό μήκος πίνακα αληθείας και για κάθε διαφορετική μεθοδολογία, να υπήρχαν όλα τα δεδομένα από 10 εώς 50 γονείς, παραγόμενα με τον τρόπο που ορίζεται στην υποενότητα «automation.py».

Ωστόσο, επειδή όπως αναφέρθηκε, η κατασκευή των functions είναι χρονικά εξαιρετικά δαπανηρή, και ειδικότερα για την παραγωγή συναρτήσεων με μήκους πίνακα αληθείας 2048, αυτό δεν ήταν εφικτό σε όλες τις περιπτώσεις για κάθε δεδομένο μήκος πίνακα αληθείας. Πιο συγκεκριμένα, για την παραγωγή τυχαίων συναρτήσεων μήκους 2048 bits, ήταν πρακτικά ανέφικτο να ολοκληρωθούν όλες οι εκτελέσεις.

7.2 Μελλοντικές κατευθύνσεις

Όπως αναφέρθηκε στο κεφάλαιο 3, ένας γενετικός αλγόριθμος αποτελείται από διακριτά στάδια. Κάθε στάδιο αυτού μπορεί να υλοποιηθεί με παραπάνω από έναν τρόπους. Συνεπώς, υπάρχουν πολλαπλοί πιθανοί συνδυασμοί των υλοποιήσεων και άρα διαφορετικοί αλγόριθμοι που μπορεί να εξερευνηθούν. Στην παρούσα διπλωματική διατριβή χρησιμοποιείται μόνο ένας τρόπος σε κάθε στάδιο για την παραγωγή αποτελεσμάτων.

Αντίστοιχα, οι περεταίρω δοκιμές θα μπορούσαν να γίνουν και με διαφορετικές fitness functions καθώς και μεγαλύτερα ποσοστά μετάλλαξης. Οπότε, αυτό συνιστά μία πιθανή νέα κατεύθυνση αναζήτησης..

Στην συνέχεια, η μέθοδος αναζήτησης με bent και Carlet-Feng συναρτήσεις θα μπορούσε να υλοποιηθεί με διάφορες οικογένειες bent και μάλιστα, να δοκιμαστεί και μία μέθοδος αναζήτησης όπου ο αρχικός πληθυσμός αποτελείται από bent και τυχαίες συναρτήσεις.

Ακόμα, αξίζει να σημειωθεί πως η παρούσα υλοποίηση δίνει περισσότερη έμφαση στο ευρύ πεδίο αναζήτησης, στην εκτέλεση πολλαπλών εκτελέσεων με ποικίλες παραμέτρους. Υπάρχει η δυνατότητα να ακολουθηθεί και μία άλλη προσέγγιση, η οποία θα δοκιμάζει σαφώς λιγότερες διαφορετικές παραμέτρους αλλά που θα εμβαθύνει και θα πιέζει για ακόμα καλύτερους και υψηλότερους βαθμούς καταλληλότητας και μη γραμμικότητας, με ελάχιστες τροποποιήσεις.

Τέλος, θα μπορούσε να δοκιμαστεί και ένας αλγόριθμος με μεγαλύτερη κλίση προς την οικογένεια των generational αλγορίθμων, δηλαδή αυτών που δεν ενσωματώνουν μηχανισμό αντικατάστασης στο τελευταίο βήμα τους και αντί να αντικαθιστούν μέρος του υπάρχοντος πληθυσμού γενιά με την γενιά, αντικαθιστούν τον τρέχοντα πληθυσμό εξολοκλήρου με τους απογόνους και τις μεταλλάξεις της αντίστοιχης γενιάς.

Παράρτημα Α
Παράμετροι Εκτελέσεων & Αποτελέσματα

Παρακάτω παρουσιάζονται όλες οι παράμετροι που δόθηκαν κατά την διεξαγωγή των πειραμάτων.

Α.1 256 bits (n=8 μεταβλητές)

Εκτελέσεις που αφορούν συναρτήσεις μήκους 256 bits (n=8 μεταβλητές).

Α.1.1 Χρήση αποκλειστικά τυχαίων συναρτήσεων

256 εκτελέσεις τυχαίων συναρτήσεων.txt

256 τυχαίες συναρτήσεις.csv

Α.1.2 Χρήση τυχαίων συναρτήσεων μαζί με Carlet-Feng

256 εκτελέσεις Carlet Feng με τυχαίες συναρτήσεις.txt

256 Carlet Feng με τυχαίες συναρτήσεις.c

Α.1.3 Χρήση συναρτήσεων Bent με Carlet-Feng

256 εκτελέσεις Carlet Feng με Bent συναρτήσεις.txt

256 Carlet Feng με Bent συναρτήσεις.csv

Α.2 1024 bits (n=10 μεταβλητές)

Εκτελέσεις που αφορούν συναρτήσεις μήκους 1024 bits (n=10 μεταβλητές).

Α.2.1 Χρήση αποκλειστικά τυχαίων συναρτήσεων

1024 εκτελέσεις τυχαίων συναρτήσεων.txt

1024 τυχαίες συναρτήσεις.csv

A.2.2 Χρήση τυχαίων συναρτήσεων μαζί με Carlet-Feng


1024 εκτελέσεις Carlet Feng με τυχαίες συναρτήσεις.txt


1024 Carlet Feng με τυχαίες συναρτήσεις.c

A.2.3 Χρήση συναρτήσεων Bent με Carlet-Feng


1024 Carlet Feng με Bent συναρτήσεις.txt


1024 Carlet Feng με Bent συναρτήσεις.csv

A.3 2048 bits (n=11 μεταβλητές).

Εκτελέσεις που αφορούν συναρτήσεις μήκους 2048 bits (n=11 μεταβλητές).

A.3.1 Χρήση αποκλειστικά τυχαίων συναρτήσεων


2048 εκτελέσεις τυχαίων συναρτήσεων .txt


2048 τυχαίες συναρτήσεις.csv

A.3.2 Χρήση τυχαίων συναρτήσεων μαζί με Carlet-Feng


2048 εκτελέσεις Carlet Feng με τυχαίες συναρτήσεις.txt


2048 Carlet Fengs με τυχαίες συναρτήσεις.c

Παράρτημα Β
Κώδικας προγράμματος σε python

Παρακάτω παρουσιάζεται όλος ο κώδικας όπως αναπτύχθηκε και χρησιμοποιήθηκε, ανά βιβλιοθήκη.

Β.1 genesys_sage.py

genesys_sage.py

Β.2 cfeng.py

cfeng.py

Β.3 bent.sage

bent.sage

Β.4 darwin.py

darwin.py

Β.5 automation.py

automation.py

B.6 main.sage

main.sage

Βιβλιογραφία

[1] Mohammed, Abdalbasit & Varol, Nurhayat. (2019). A Review Paper on Cryptography. 1-6. 10.1109/ISDFS.2019.8757514.

[2] Noura, Hassan, et al. “LESCA: LightwEight Stream Cipher Algorithm for Emerging Systems.” Ad Hoc Networks, vol. 138, Elsevier BV, Jan. 2023, p. 102999. https://doi.org/10.1016/j.adhoc.2022.102999.

[3] Picek S, Carlet C, Guilley S, Miller JF, Jakobovic D. Evolutionary Algorithms for Boolean Functions in Diverse Domains of Cryptography. Evol Comput. 2016 Winter;24(4):667-694. doi: 10.1162/EVCO_a_00190. Epub 2016 Aug 2. PMID: 27482749.

[4] “SageMath Mathematical Software System - Sage.” SageMath Mathematical Software System, www.sagemath.org.

[5]https://www.researchgate.net/publication/329018296_Cryptographic_Boolean_Functions_and_Applications_Second_edition Advances in Cryptology-Eurocrypt. ’85, vol. 219, pp. 103-110, 1986.

[6] T. Siegenthaler, «Cryptanalysts representation of nonlinearly filtered m-sequences». Advances in Cryptology-Eurocrypt. ’85, vol. 219, pp. 103-110, 1986.

[7] P. Camion, C. Carlet, P. Charpin, N. Sendrier. «On correlation-immune functions. In: Advances in cryptology: Crypto ’91». Lecture notes in computer science, Springer, vol 576, pp 86– 100, 1991.

[8] Κ. Λιμνιώτης, Σημειώσεις μαθήματος «Διάλεξη 3- “Κρυπταλγόριθμοι ροής – Τυχαιότητα ακολουθιών & κρυπτογραφικές λογικές συναρτήσεις”»,Σελ.:25-34,2022-2023.

[9] N. Courtois. «Fast algebraic attacks on stream ciphers with linear feedback». Proceedings of CRYPTO 2003, Lecture notes in computer science, vol 2729, pp 177–194, 2003.

[10] Claude Carlet, Keqin Feng, An infinite class of balanced functions with optimal algebraic immunity, good immunity to fast algebraic attacks and good nonlinearity.

Advances in cryptology- ASIACRYPT 2008, 425-440, Lecture Notes in Comput. Sci.,5350, Springer, Berlin, 2008.

[11] Katoch, S., Chauhan, S.S. & Kumar, V. A review on genetic algorithm: past, present, and future. Multimed Tools Appl 80, 8091–8126 (2021). https://doi.org/10.1007/s11042-020-10139-6

[12] Wirsansky, Eyal. "Hands-On Genetic Algorithms With Python." Packt Publishing eBooks, 2020, international.scholarvox.com/catalog/book/88880128.

[13] Jebari K (2013) Selection methods for genetic algorithms. Abdelmalek Essaâdi University. International Journal of Emerging Sciences 3(4):333–344

[14] Soon GK, Guan TT, On CK, Alfred R, Anthony P (2013) "A comparison on the performance of crossover techniques in video game," 2013 IEEE international conference on control system. Computing and Engineering, Mindeb, pp 493–498

[15] Saini N (2017) Review of selection methods in genetic algorithms. International Journal of Engineering and Computer Science 6(12):22261–22263

[16] Lozano, Manuel & Herrera, Francisco & Cano, José. (1970). Replacement Strategies to Maintain Useful Diversity in Steady-State Genetic Algorithms. 10.1007/3-540-32400-3_7.

[17] Burkhardt, Micah & Yampolskiy, Roman. (2021). Death in Genetic Algorithms.

[18] Jing, Jiang & Lidong, Meng. (2012). The strategy of improving convergence of genetic algorithm. TELKOMNIKA Indonesian Journal of Electrical Engineering. 10. 10.11591/telkomnika.v10i8.1641.

[19] GeeksforGeeks. "Encoding Methods in Genetic Algorithm." GeeksforGeeks, Jan. 2023, www.geeksforgeeks.org/encoding-methods-in-genetic-algorithm.

[20] Marek Obitko, marek@obitko.com. Encoding - Introduction to Genetic Algorithms - Tutorial With Interactive Java Applets. www.obitko.com/tutorials/genetic-algorithms/encoding.php.

[21] H. Aguirre, H. Okazaki, and Y. Fuwa. An Evolutionary Multiobjective Approach to Design Highly Non-linear Boolean Functions. In Genetic and Evolutionary Computation Conference (GECCO), pages 749–756, 2007.

[22] G. T. Becker. The gap between promise and reality: On the insecurity of xor arbiter pufs. In T. G¨uneysu and H. Handschuh, editors, Cryptographic Hardware and Embedded Systems – CHES 2015, pages 535–555, Berlin, Heidelberg, 2015. Springer Berlin Heidelberg.

[23] C. Carlet, D. Jakobovic, and S. Picek. Evolutionary Algorithms-Assisted Construction of Cryptographic Boolean Functions, page 565–573. Association for Computing Machinery, New York, NY, USA, 2021.

[24] J. A. Clark and J. L. Jacob. Two-Stage Optimisation in the Design of Boolean Functions. In Information Security and Privacy, volume 1841 of LNCS, pages 242–254. Springer, 2000.

[25] J. A. Clark, J. L. Jacob, S. Maitra, and P. Stanica. Almost Boolean functions: The design of boolean functions by spectral inversion. Comput. Intell., 20(3):450–462, 2004.

[26] J. Kennedy and R. Eberhart. Particle swarm optimization. In Proceedings of ICNN'95 - International Conference on Neural Networks, volume 4, pages 1942–1948 vol.4, 1995.

[27] L. Mariot and A. Leporati. A genetic algorithm for evolving plateaued cryptographic boolean functions. In A. Dediu, L. Magdalena, and C. Mart´ın-Vide, editors, Theory and Practice of Natural Computing - Fourth International Conference, TPNC 2015, Mieres, Spain, December15-16, 2015. Proceedings, volume 9477 of Lecture Notes in Computer Science, pages 33–45. Springer, 2015.

[28] L. Mariot and A. Leporati. Heuristic Search by Particle Swarm Optimization of Boolean Functions for Cryptographic Applications. In Genetic and Evolutionary Computation Conference, GECCO, Companion Material Proceedings, pages 1425–1426, 2015.

[29] L. Mariot, S. Picek, A. Leporati, and D. Jakobovic. Cellular automata based s-boxes. Cryptography and Communications, 11(1):41–62, 2019.

[30] L. Mariot, M. Saletta, A. Leporati, and L. Manzoni. Heuristic search of (semi-)bent functions based on cellular automata. CoRR, abs/2111.13248, 2021.

[31] W. Millan, A. Clark, and E. Dawson. An Effective Genetic Algorithm for Finding Highly Nonlinear Boolean Functions. In First Int. Conference on Information and Communication Security, ICICS '97, pages 149–158. Springer, 1997.

[32] W. Millan, A. Clark, and E. Dawson. Heuristic design of cryptographically strong balanced Boolean functions. In Advances in Cryptology - EUROCRYPT '98, pages 489–499, 1998.

[33] S. Picek and D. Jakobovic. Evolving algebraic constructions for designing bent boolean functions. In T. Friedrich, F. Neumann, and A. M. Sutton, editors, Proceedings of the 2016 on Genetic and Evolutionary Computation Conference, Denver, CO, USA, July 20 - 24, 2016, pages 781–788. ACM, 2016.

[34] S. Picek and D. Jakobovic. Evolutionary computation and machine learning in cryptology. In Proceedings of the 2020 Genetic and Evolutionary Computation Conference Companion, GECCO '20, page 1147–1173, New York, NY, USA, 2020. Association for Computing Machinery.

[35] S. Picek, D. Jakobovic, and M. Golub. Evolving Cryptographically Sound Boolean Functions. In Genetic and Evolutionary Computation Conference (GECCO), GECCO '13 Companion, pages 191–192. ACM, 2013.

[36] S. Picek, D. Sisejkovic, and D. Jakobovic. Immunological algorithms paradigm for construction of boolean functions with good cryptographic properties. Eng. Appl. of AI, 62:320–330, 2017. Circuits Syst., 6(5):727–750, 1987.

[37] S. Saha, R. S. Chakraborty, S. S. Nuthakki, Anshul, and D. Mukhopadhyay. Improved test pattern generation for hardware trojan detection using genetic algorithm and boolean satisfiability. In Cryptographic Hardware and Embedded Systems - CHES 2015 - 17th International Workshop, Saint-Malo, France, September 13-16, 2015, Proceedings, pages 577–596, 2015.

[38] Z. Zhang, L. Wu, A. Wang, Z. Mu, and X. Zhang. A novel bit scalable leakage model based on genetic algorithm. Security and Communication Networks, 8(18):3896–3905, 2015.

[39] Carlet, C., Djurasevic, M., Jakobovic, D., Mariot, L., Picek, S.: Evolving constructions for balanced, highly nonlinear boolean functions. In Proceedings of the Genetic and Evolutionary Computation Conference, GECCO '22, page 1147–1155, New York, NY, USA. Association for Computing Machinery (2022)

[40] Hrbacek, R., Dvorak, V.: Bent function synthesis by means of cartesian genetic programming. In T. BartzBeielstein, J. Branke, B. Filipiˇc, and J. Smith, editors, Parallel Problem Solving from Nature - PPSN XIII, pages 414–423, Cham, 2014. Springer International Publishing (2014)