Ανοικτό Πανεπιστήμιο Κύπρου

Σχολή Θετικών και Εφαρμοσμένων Επιστημών

Μεταπτυχιακό Πρόγραμμα Σπουδών Στα *Πληροφοριακά Και Επικοινωνιακά Συστήματα*

Μεταπτυχιακή Διατριβή

Κατασκευές κρυπτογραφικών συναρτήσεων μέσω γενετικών αλγορίθμων

Ευάγγελος Λυκάκης

Επιβλέπων Καθηγητής
Δρ. Κωνσταντίνος Λιμνιώτης

Ιούνιος 2017

Ανοικτό Πανεπιστήμιο Κύπρου

Σχολή Θετικών και Εφαρμοσμένων Επιστημών

Μεταπτυχιακό Πρόγραμμα Σπουδών *Στα Πληροφοριακά*

Και Επικοινωνιακά Συστήματα

Μεταπτυχιακή Διατριβή

Κατασκευές κρυπτογραφικών συναρτήσεων μέσω γενετικών αλγορίθμων

Ευάγγελος Λυκάκης

Επιβλέπων Καθηγητής
Δρ. Κωνσταντίνος Λιμνιώτης

Η παρούσα μεταπτυχιακή διατριβή υποβλήθηκε προς μερική εκπλήρωση των απαιτήσεων για απόκτηση μεταπτυχιακού τίτλου σπουδών στα Πληροφοριακά και Επικοινωνιακά Συστήματα από τη Σχολή Θετικών και Εφαρμοσμένων Επιστημών του Ανοικτού Πανεπιστημίου Κύπρου.

Ιούνιος 2017

ΛΕΥΚΗ ΣΕΛΙΔΑ

Περίληψη

Η ασφάλεια των συμμετρικών κρυπτογραφικών αλγορίθμων βασίζεται σε μεγάλο βαθμό στις ιδιότητες των υποκείμενων συναρτήσεων, είτε αυτές είναι λογικές συναρτήσεις μίας εξόδου (Boolean functions) είτε διανυσματικές συναρτήσεις πολλών εξόδων (vectorial functions). Παρόλο που είναι γνωστό το σύνολο των ιδιοτήτων που πρέπει να πληροί μία συνάρτηση για να είναι ανθεκτική σε γνωστές επιθέσεις ασφαλείας, εν τούτοις δεν είναι γνωστές πολλές κατασκευές που εγγυημένα να ικανοποιούν όλες αυτές τις ιδιότητες. Περαιτέρω, όλες οι γνωστές τεχνικές βασίζονται σε κάποιες καλά ορισμένες μαθηματικές ιδιότητες συγκεκριμένης δομής (π.χ. στο πρότυπο κρυπτογράφησης AES, η μονάδα αντικατάστασης S-box έχει μία εξαιρετικά απλά μαθηματική περιγραφή) – γεγονός που πάντα αφήνει ανοικτό το ενδεχόμενο να αποδειχθούν κάποια στιγμή ευπαθείς σε αλγεβρικές επιθέσεις που εκμεταλλεύονται αυτήν τη μαθηματική δομή.

Στην παρούσα διατριβή μελετάται το ερευνητικό ζήτημα της κατασκευής λογικών συναρτήσεων που να ικανοποιούν – κατά το δυνατόν - το σύνολο των σημαντικών κρυπτογραφικών κριτηρίων (όπως το να είναι ισοβαρείς, να έχουν υψηλό αλγεβρικό βαθμό, υψηλή μη γραμμικότητα αλλά και ανθεκτικότητα σε κάθε είδους αλγεβρική επίθεση). Η προσέγγιση που ακολουθείται είναι διεπιστημονική, αφού για την κατασκευή των συναρτήσεων γίνεται χρήση εξελικτικών αλγορίθμων (evolution algorithms), ήτοι αλγορίθμων που προσομοιώνουν τη φυσική εξέλιξη της ανάπτυξης – δηλαδή μίας κατηγορίας αλγορίθμων διαφορετικού επιστημονικού πεδίου. Η προσέγγιση αυτή στα τελευταία χρόνια αρχίζει να φαίνεται ότι αποτελεί μία νέα εναλλακτική για τη δημιουργία ισχυρών κρυπτογραφικών συναρτήσεων.

Ειδικότερα, θα μελετηθούν οι μέχρι σήμερα γνωστές τεχνικές για την κατασκευή κρυπτογραφικών συναρτήσεων βάσει της κατηγορίας των γενετικών αλγορίθμων (μιας κατηγορίας εξελικτικού αλγόριθμου). Ακολούθως, αναπτύσσονται γενετικοί αλγόριθμοι κατάλληλα προσαρμοσμένοι για την κατασκευή ισχυρών κρυπτογραφικών συναρτήσεων, βάσει κατάλληλης επιλογής νέων σχεδιαστικών κριτηρίων. Ιδιαίτερη έμφαση δίνεται στην κατασκευή συναρτήσεων υψηλής μη γραμμικότητας, θέτοντας ως βάση αναφοράς τη μη γραμμικότητα μιας γνωστής οικογένειας κρυπτογραφικών συναρτήσεων (γνωστή ως συνάρτηση Carlet-Feng) η οποία ικανοποιεί το σύνολο των κρυπτογραφικών κριτηρίων. Τα αποτελέσματα καταδεικνύουν ότι οι γενετικοί αυτοί αλγόριθμοι μπορούν να οδηγήσουν στην εύρεση ισχυρών κρυπτογραφικών συναρτήσεων – σε κάποιες δε περιπτώσεις ισάξιες ή και καλύτερες από τη συνάρτηση Carlet-Feng ως προς τη μη γραμμικότητα, χωρίς να υστερούν και στα λοιπά κριτήρια.

Summary

The security of symmetric cryptographic algorithms is highly based on the properties of the underlying functions, which are either Boolean (i.e. single-output) functions or vectorial (i.e. multi-output) functions. Although all the properties that a function needs to satisfy in order to be resistant against known security attacks are known, it is a difficult task to provide a construction of functions that are bound to satisfy all these properties. Furthermore, all known and used techniques are based on some well-defined mathematical properties of a particular structure (e.g. in the AES cipher, the S-box has an extremely simple mathematical description for which certain mathematical properties are easily demonstrated), which in turn raises some concerns with regard to possible future attacks that may exploit this mathematical structure.

In this thesis, we focus on constructing new Boolean functions satisfying all main cryptographic criteria such as high algebraic degree, balancedness, high nonlinearity and resistance against any kind of algebraic attacks. To this goal, the adopted approach is inter-scientific, based on the use of evolution algorithms, which simulate the natural evolution. This approach has been followed in recent years as a new alternative for constructing powerful cryptographic functions.

More precisely, this thesis studies all the currently known such techniques for the construction of cryptographic functions. In addition, a class of genetic algorithms (a subclass of evolutionary algorithms) is being used for constructing new functions, which is appropriately modified via forcing specific input parameters. Special emphasis is given on achieving high nonlinearity, with respect to the nonlinearity of a known cryptographic function (called as Carlet-Feng function) which satisfies all the main cryptographic criteria. It is shown that functions with nonlinearity equal or even higher than the one of the Carlet-Feng function can be achieved, without sacrificing other cryptographic criteria.

Ευχαριστίες

Ως ελάχιστη ένδειξη της βαθιάς μου ευγνωμοσύνης, αισθάνομαι μέγιστη υποχρέωση μου να εκφράσω τις θερμές μου ευχαριστίες, σε όλους όσους συνέβαλαν έστω και ελάχιστα, άμεσα ή έμμεσα, στην ολοκλήρωση της παρούσας μεταπτυχιακής διατριβής.

Αρχικώς στον επιβλέποντα Καθηγητή μου κ. Λιμνιώτη Κωνσταντίνο, που αφιέρωσε άπλετο χρόνο και έδινε πάντα εύστοχες υποδείξεις και επισημάνσεις. Ο σπάνιος συνδυασμός γνώσεων και εμπειριών που τον διακρίνουν, διαδραμάτισαν καθοριστικό ρόλο στην διαδικασία εκπόνησης της μεταπτυχιακής μου διατριβής.

Τέλος θα ήθελα να ευχαριστήσω θερμά τον πατέρα μου Ιωάννη και την μητέρα μου Αθηνά, που με στήριξαν όλο το διάστημα των σπουδών μου.

στην μνήμη της
γιαγιάς μου Αικατερίνης

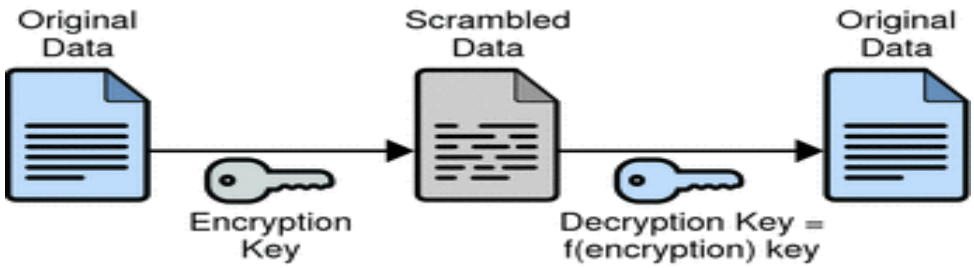
Περιεχόμενα

Κεφάλαιο 1

Εισαγωγή

Η κρυπτογραφία έχει μία πάρα πολύ μακρά ιστορία, έχοντας χρησιμοποιηθεί, μεταξύ άλλων για να διαφυλάξει στρατιωτικές και διπλωματικές επικοινωνίες. Χαρακτηριστικά ιστορικά παραδείγματα κρυπτογράφησης είναι ,οι Σπαρτιάτες το 400 π.Χ. που χρησιμοποιούσαν κρυπτογραφημένα μηνύματα για να μεταφέρουν μηνύματα στους στρατιώτες τους (Σπαρτιατική σκυτάλη), αυτή του Ιούλιου Καίσαρα που χρησιμοποίησε ένα αλγόριθμο αντικατάστασης για μεταφορά μηνυμάτων (Caesar cipher) καθώς επίσης και η μηχανή enigma που χρησιμοποιήθηκε στον Β΄παγκόσμιο πόλεμο[49].

Ο θεμελιώδης στόχος της κρυπτογραφίας είναι να επιτρέπει σε δύο άτομα να έχουν επικοινωνία μέσω ενός μη ασφαλούς καναλιού με τέτοιο τρόπο έτσι ώστε να μην μπορούν άλλα μη εξουσιοδοτημένα άτομα να διαβάσουν το μήνυμα. Με την κρυπτογράφηση λοιπόν μπορούμε να πετύχουμε εμπιστευτικότητα, αλλά επίσης και ταυτοποίηση και ακεραιότητα του μηνύματος. Η μετατροπή του αρχικού μηνύματος (plaintext) σε κρυπτογραφημένο μήνυμα ή κρυπτοκείμενο (ciphertext) το οποίο είναι ακατάληπτο σε οποιοδήποτε μη εξουσιοδοτημένο χρήστη, ονομάζεται κρυπτογράφηση. Ο αλγόριθμος κρυπτογράφησης δέχεται ως είσοδο το αρχικό μήνυμα και ένα κλειδί κρυπτογράφησης, και εξάγει το κρυπτοκείμενο. Η αντίστροφη διαδικασία ονομάζεται αποκρυπτογράφηση. Ο αλγόριθμος αποκρυπτογράφησης λαμβάνει σαν είσοδο το κρυπτοκείμενο και το κλειδί αποκρυπτογράφησης (Σχ. 1.1).



Σχήμα 1.1. κρυπτογράφηση –αποκρυπτογράφηση

Οι κρυπτογραφικοί αλγόριθμοι αποτελούν το μέσο για το μετασχηματισμό μηνυμάτων σε κρυπτογραφημένα κείμενα. Μια διαδικασία κρυπτογράφησης συμβολίζεται ως : $c = e_k(m)$, όπου m είναι το αρχικό κείμενο, e είναι ο αλγόριθμος κρυπτογράφησης, k είναι το μυστικό κλειδί και c είναι το κρυπτογραφημένο κείμενο. Αντίστοιχα, η διαδικασία της αποκρυπτογράφησης συμβολίζεται ως : m = d_k(c) όπου d ο αλγόριθμος αποκρυπτογράφησης. Για να αποφανθούμε για την ασφάλεια του κρυπτοσυστήματος λαμβάνεται υπόψη η λεγόμενη αρχή του Kerchoff . Βάσει αυτής της αρχής, όλοι οι κρυπτογραφικοί αλγόριθμοι, δεν θα πρέπει να είναι μυστικοί, αλλά θα πρέπει να είναι ευρέως γνωστοί και να δημοσιεύονται στο κοινό. Η ασφάλεια του κρυπτοσυστήματος έγκειται στην μυστικότητα του κλειδιού που χρησιμοποιείται τόσο στην κρυπτογράφηση όσο και στην αποκρυπτογράφηση. Το κλειδί θα πρέπει να το γνωρίζουν και, άρα, να το έχουν ανταλλάξει με ασφάλεια μόνο οι δύο συνδιαλεγόμενοι.

Με τον όρο κρυπτανάλυση αναφερόμαστε στη μελέτη μαθηματικών τεχνικών που έχουν στόχο να ακυρώσουν τις κρυπτογραφικές μεθόδους, προκειμένου να πληγεί η ασφάλεια[43]. Υπάρχουν τέσσερα είδη τεχνικών κρυπτανάλυσης, ανάλογα με το ποια γνώση έχει ο επίδοξος υποκλοπέας : α) γνωστού κρυπτοκειμένου (γνωρίζει μόνο το κρυπτοκείμενο) β) γνωστού μηνύματος (εκτός από το κρυπτοκείμενο γνωρίζει και μέρος του αρχικού μηνύματος) γ) επιλεγμένου μηνύματος (επιλέγει το μέρος από το αρχικό μήνυμα που θα μελετήσει , για το οποίο γνωρίζει το αντίστοιχο κρυπτοκείμενο) δ) επιλεγμένου κρυπτοκειμένου (επιλέγει μέρος του κρυπτοκειμένου που θα μελετήσει για το οποίο γνωρίζει το αντίστοιχο αρχικό μήνυμα) [50].

Δύο είναι οι κύριοι τύποι των κρυπτογραφικών συστημάτων είναι: ή συμμετρική κρυπτογραφία (βλ. Σχήμα 1) και η κρυπτογραφία δημόσιου κλειδιού (ή ασύμμετρη). Στην πράξη, η αρχή της συμμετρικής κρυπτογραφίας βασίζεται στην χρήση του ίδιου μυστικού κλειδιού μεταξύ του αποστολέα ενός μηνύματος και του αποδέκτη του. Είναι

η περίπτωση κρυπτογραφικών αλγόριθμων που θα εστιάσει η παρούσα διατριβή. Από την άλλη πλευρά, στην κρυπτογραφία δημόσιου κλειδιού υπάρχει μια διαφορετική λογική: το κλειδί κρυπτογράφησης είναι δημόσιο και διαφορετικό από το κλειδί αποκρυπτογράφησης, το οποίο το ξέρει μόνο ο παραλήπτης.

Η κρυπτογραφία δημόσιου κλειδιού εμφανίστηκε στη βιβλιογραφία στα τέλη της δεκαετίας του 1970[12]. Έχει πλεονέκτημα σε σχέση με τη συμμετρική κρυπτογραφία, ότι επιτρέπει σε δύο χρήστες να επικοινωνούν με ασφάλεια χωρίς να πρέπει να έχουν προ-ανταλλάξει ένα μυστικό κλειδί: κάθε πρόσωπο που επιθυμεί να λάβει μυστικά μηνύματα μπορεί να κρατήσει μυστικό ένα κλειδί αποκρυπτογράφησης και δημοσιεύει ένα κλειδί κρυπτογράφησης (κατάλληλα επιλεγμένα). Στην κρυπτογράφηση δημόσιου κλειδιού όταν n άτομα θέλουν να επικοινωνήσουν τότε χρειάζονται (n) κλειδιά κρυπτογράφησης και (n) κλειδιά αποκρυπτογράφησης, την στιγμή που στην συμβατική κρυπτογράφηση χρειάζονται $\binom{n}{2} = \frac{n(n-1)}{2}$ κλειδιά. Ωστόσο όλα τα γνωστά κρυπτογραφικά συστήματα δημόσιου κλειδιού είναι πολύ λιγότερο αποδοτικά από τους συμμετρικούς αλγόριθμους κρυπτογράφησης. Αυτός είναι ο λόγος που η συμμετρική κρυπτογράφηση εξακολουθεί να χρησιμοποιείται ευρέως στις μέρες μας ενώ οι κρυπταλγόριθμοι δημοσίου κλειδιού χρησιμοποιούνται για την ασφαλή ανταλλαγή του μυστικού συμμετρικού κλειδιού κρυπτογράφησης.

Οι συμμετρικοί αλγόριθμοι αναφέρονται και ως αλγόριθμοι μυστικού κλειδιού ή συμβατικής κρυπτογραφίας. Η συμμετρική κρυπτογραφία εγγυάται την εμπιστευτικότητα των δεδομένων αφού κρυπτογραφεί το μήνυμα με το μυστικό κλειδί. Το μήνυμα που παράγεται αποκρυπτογραφείται από τον παραλήπτη με τη βοήθεια του ίδιου κλειδιού, το οποίο πρέπει να μείνει μυστικό μεταξύ των δύο. Η ασφάλεια τους βασίζεται στην μυστικότητα του κλειδιού. Τα συμμετρικά κρυπτοσυστήματα προϋποθέτουν την ανταλλαγή του κλειδιού μέσα από ένα ασφαλές κανάλι επικοινωνίας ή μέσα από τη φυσική παρουσία των προσώπων.

Ο σχεδιασμός των συμμετρικών κρυπτογραφικών συστημάτων στηρίζεται σε δύο θεμελιώδεις αρχές που εισήγαγε ο Claude Shannon ,στη σύγχυση και την διάχυση. Η σύγχυση αποσκοπεί στην απόκρυψη κάθε αλγεβρικής δομής του συστήματος και είναι στενά συνδεδεμένο με την κρυπτογραφική πολυπλοκότητα των εμπλεκόμενων λογικών (Boolean) συναρτήσεων. Συγκεκριμένα θα πρέπει η σχέση μεταξύ του κρυπτοκειμένου και του μυστικού κλειδιού να είναι σύνθετη, έτσι ώστε ακόμα από τα

στατιστικά χαρακτηριστικά του κρυπτοκειμένου να μην είναι εφικτή η ανάκτηση του κλειδιού λόγω ακριβώς του σύνθετου τρόπου με τον οποίο επέδρασε το κλειδί κατά την παραγωγή του κρυπτοκειμένου. Η διάχυση συνίσταται στην εξάπλωση της επιρροής της κάθε ήσσονος σημασίας τροποποίησης των δεδομένων εισόδου ή του κλειδιού πάνω από όλες τις εξόδους. Συγκεκριμένα κάθε ψηφίο (bit) του αρχικού μηνύματος πρέπει να επηρεάζει όσο γίνεται περισσότερα ψηφία του κρυπτοκειμένου[50].

Οι συμμετρικοί αλγόριθμοι κρυπτογράφησης διακρίνονται σε δύο μεγάλες κατηγορίες. Στους Αλγόριθμους ροής (Stream ciphers) και τους Αλγόριθμους τμήματος (Block ciphers). Στους αλγόριθμους ροής το αρχικό μήνυμα κρυπτογραφείται χαρακτήρα-χαρακτήρα (ξεχωριστή κρυπτογράφηση για κάθε bit ή byte). Στους αλγόριθμους τμήματος το αρχικό μήνυμα χωρίζεται σε τμήματα (blocks) και το κάθε τμήμα κρυπτογραφείται ξεχωριστά.

1.1 Αλγόριθμοι ροής

Ένας αλγόριθμος ροής επεξεργάζεται κατά συνεχή τρόπο τα στοιχεία εισόδου και κάθε φορά παράγεται ως έξοδος ένα στοιχείο, με τη σειρά που καταφθάνουν τα δεδομένα. Η κρυπτογραφική λειτουργία περιγράφεται ως εξής: Προσθέτει με τη χρήση της λογικής πράξη της αποκλειστικής διάζευξης (XOR) το αρχικό κείμενο με μία τυχαία ακολουθία δεδομένων που αποκαλείται κλειδοροή (keystream) για να παράγει το κρυπτογραφημένο κείμενο. Ένας συμβολισμός του κρυπτογραφικού αλγόριθμου ροής θα μπορούσε να είναι :

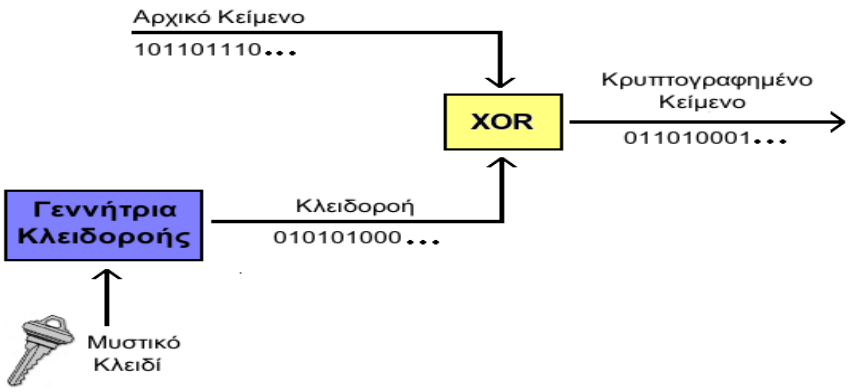
$$c_i = p_i \oplus k_i, \text{ για i} \geq 0,$$

όπου p0 ,p1 , ... είναι τα bits του αρχικού κειμένου, k0 , k1 , ... είναι τα bits της κλειδοροής και c0 ,c1 , ..είναι τα bits του κρυπτογραφημένου κειμένου. Το σύμβολο $\oplus$ συμβολίζει την πράξη της αποκλειστικής διάζευξης (xor) μεταξύ των bits. Αντίστοιχα για την αποκρυπτογράφηση θα ισχύει :

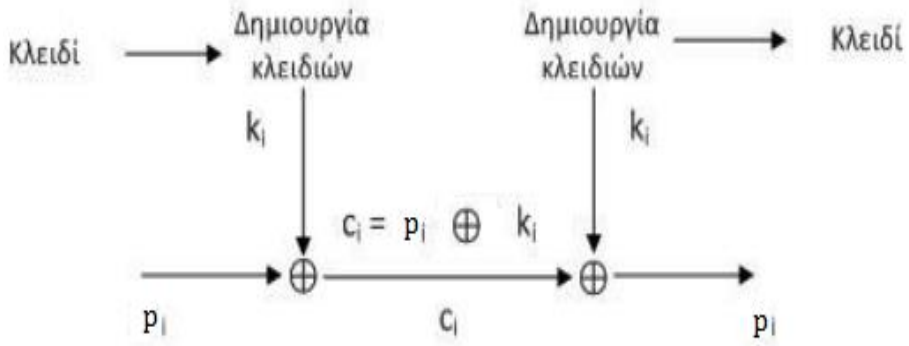
$$p_i = c_i \oplus k_i, \text{ για i} \geq 0$$

Δηλαδή , η αποκρυπτογράφηση στους αλγόριθμους ροής γίνεται με τον ίδιο τρόπο που γίνεται η κρυπτογράφηση (Σχ. 1.2,Σχ.1.3)[51]. Ένα παράδειγμα κρυπτογράφησης είναι το εξής:

Μήνυμα	:10010011
Κλειδοροή	:00010110
Κρυπτοκείμενο	:10000101



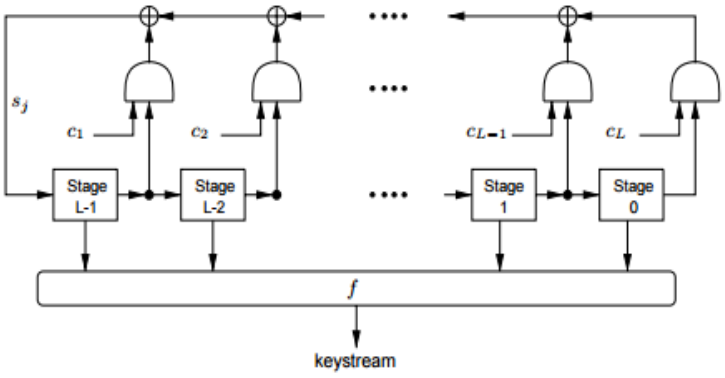
Σχήμα 1.2. Διάγραμμα λειτουργίας αλγορίθμων ροής



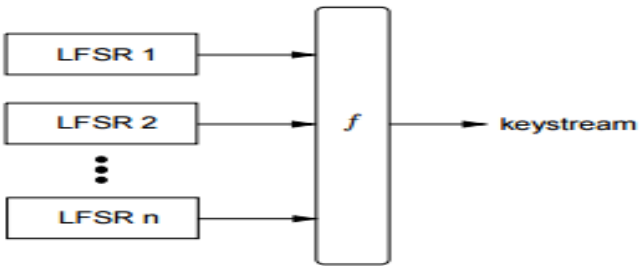
Σχήμα 1.3. Αλγόριθμος ροής

Για την παραγωγή της κλειδοροής χρησιμοποιούνται οι λεγόμενες γεννήτριες κλειδοροής (keystream generators), οι οποίες αρχικοποιούνται με μία αρχική κατάσταση και στη συνέχεια διατρέχουν μία σειρά καταστάσεων, παράγοντας κάθε φορά ένα bit εξόδου. Η αρχική κατάσταση της γεννήτριας προκύπτει από το κλειδί του αλγορίθμου (το οποίο, αν το γνωρίζουν και ο αποστολέας και ο παραλήπτης, είναι σε θέση να παράγουν την ίδια κλειδοροή). Η σχεδίαση ενός κρυπταλγορίθμου ροής ανάγεται στη σχεδίαση μίας γεννήτριας κλειδοροής, η οποία πρέπει να παράγει ακολουθίες με ποιοτικά κρυπτογραφικά χαρακτηριστικά έτσι ώστε να μην είναι προβλέψιμες. Μια γεννήτρια κλειδοροής με σχετικά ποιοτικά χαρακτηριστικά είναι οι

γραμμικοί καταχωρητές ολίσθησης με ανάδραση (LFSR). Οι γραμμικοί καταχωρητές ολίσθησης με ανάδραση (LFSR) υλοποιούνται εύκολα σε hardware, απαιτούν μικρή κατανάλωση ισχύος, επιτυγχάνουν υψηλές ταχύτητες λειτουργίας και έχουν στέρεο μαθηματικό υπόβαθρο. Ωστόσο παράγουν ακολουθίες χαμηλής γραμμικής πολυπλοκότητας. Για να παράγουν όμως ακολουθίες υψηλής γραμμικής πολυπλοκότητας συνδυάζονται με σύνθετες δομές (λογικές συναρτήσεις). Ορισμένα από αυτά είναι το μη γραμμικό φίλτρο (Σχ. 1.4) και ο μη γραμμικός συνδυαστής (Σχ.1.5)[47].



Σχήμα 1.4. Μη γραμμικό φίλτρο



Σχήμα 1.5. Μη γραμμικός συνδυαστής

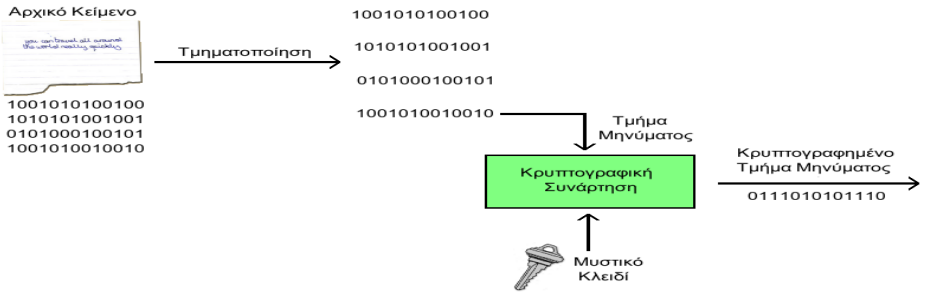
Οι αλγόριθμοι ροής χωρίζονται με την σειρά τους σε δύο κατηγορίες :τους σύγχρονους και τους ασύγχρονους. Στους σύγχρονους αλγόριθμους ροής η επόμενη κατάσταση που θα βρίσκεται το σύστημα (γεννήτρια κλειδοροής) είναι ανεξάρτητη τόσο από το μήνυμα που κρυπτογραφείται όσο και από και από το κρυπτογραφημένο κείμενο. Σε αυτή την περίπτωση, αν γίνει σφάλμα κατά την μετάδοση ενός bit, τότε δεν θα επηρεαστεί η αποκρυπτογράφηση των επόμενων bits. Στους ασύγχρονους αλγόριθμους ροής η κατάσταση στην οποία βρίσκεται το σύστημα (γεννήτρια κλειδοροής) είναι άμεσα εξαρτημένη από το κείμενο που έχει προηγουμένως αποκρυπτογραφηθεί. Αυτό έχει σαν αποτέλεσμα αν μεταδοθεί ένα bit λανθασμένα, τα επόμενα bits να μη μπορούν να αποκρυπτογραφηθούν σωστά. Για αυτό το λόγο, οι αλγόριθμοι αυτοί ανά τακτά

διαστήματα στέλνουν ειδικά μηνύματα επανασυγχρονισμού. Έτσι αν η τρέχουσα κατάσταση εξαρτάται από n προηγούμενες καταστάσεις, τότε σε n αποκρυπτογραφήσεις το πολύ, το λάθος θα γίνει αντιληπτό και θα γίνει επανασυγχρονισμός. Για το λόγο αυτό, οι αλγόριθμοι αυτοί ονομάζονται και αυτοσυγχρονιζόμενοι.

Λόγω της απλής λειτουργίας τους ,είναι κατάλληλοι για εφαρμογές τηλεπικοινωνιών όπου είναι επιθυμητή πολύ υψηλή ταχύτητα ή/και χαμηλή κατανάλωση ενέργειας. Έχοντας επίσης πολύ απλή κατασκευή, εφαρμόζονται εύκολα τόσο σε hardware και software. Οι πιο γνωστοί stream ciphers είναι οι RC4 ,Chacha20, Grain, A5/1(για το πρωτόκολλο GSM) και E0 (στο πρωτόκολλο bluetooth)[47].

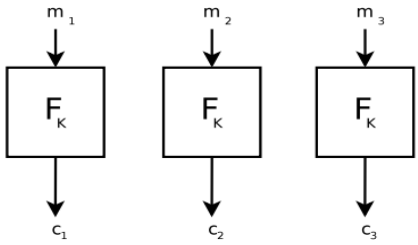
1.2 Αλγόριθμοι τμήματος

Ένας κρυπταλγόριθμος τμήματος επεξεργάζεται την είσοδο(block) ενός τμήματος (συνόλου bit) κάθε φορά, παράγοντας ένα τμήμα (block) εξόδου για κάθε συγκεκριμένο τμήμα εισόδου. Συνεπώς τα δεδομένα που πρόκειται να κρυπτογραφηθούν πρέπει να κατατμηθούν σε n τμήματα (block) δυαδικών ψηφίων (bits) (Σχ.1.6). Αρχικά είχε επιλεγεί ως ικανοποιητικό μέγεθος τμήματος τα 64-bit, ενώ σήμερα συνηθίζεται το 128-bit μέγεθος block. Το αρχικό κείμενο m κρυπτογραφείται ,ένα τμήμα κάθε φορά, με εφαρμογή της κρυπτογραφικής συνάρτησης e και του μυστικού κλειδιού k. Το αποτέλεσμα είναι μια ακολουθία τμημάτων του κρυπτογραφημένου κειμένου c. Σε έναν τέτοιο κρυπτογραφικό αλγόριθμο μπορούμε να αναπαραστήσουμε τη λειτουργία κρυπτογράφησης με τη σχέση: c= e_k (m), όπου m είναι η δέσμη αρχικού κειμένου, k είναι το μυστικό κλειδί και c είναι η δέσμη κρυπτογραφημένου κειμένου. Επιπρόσθετα, μπορούμε να αναπαραστήσουμε τη λειτουργία της αποκρυπτογράφησης d με τη σχέση: m = d_k (c) .

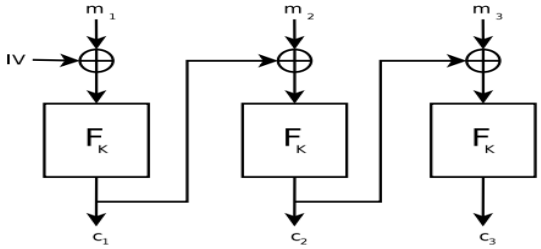


Σχήμα 1.6. Διάγραμμα τρόπου λειτουργίας αλγόριθμων τμήματος

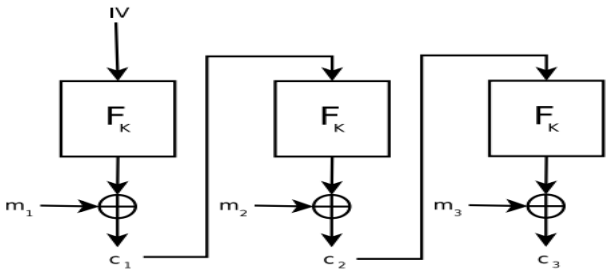
Οι αλγόριθμοι τμήματος ανάλογα με το πώς διαχειρίζονται τα τμήματα κατά την κρυπτογράφηση, μπορούν να χρησιμοποιούν διάφορους τρόπους λειτουργίας. Συγκεκριμένα έχουν τις εξής καταστάσεις λειτουργίας : Ηλεκτρονικό κωδικοβιβλίο (Electronic Codebook –ECB), Cipher Block Chaining (CBC), Cipher Feedback (CFB), Output Feedback (OFB), Counter (CTR)), όπως φαίνονται στα σχήματα 1.7-1.11 αντίστοιχα[52]. Κάθε τρόπος λειτουργία έχει τα δικά του πλεονεκτήματα και μειονεκτήματα έναντι των υπολοίπων.



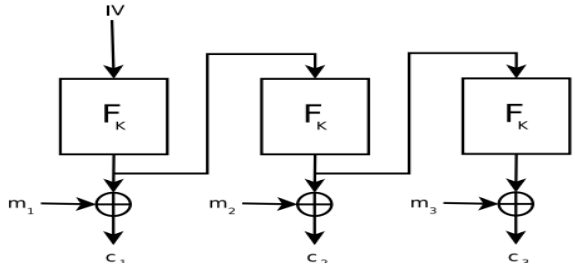
Σχήμα 1.7. Κατάσταση λειτουργίας ECB.



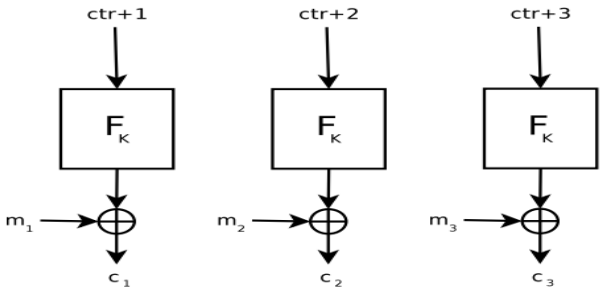
Σχήμα 1.8. Κατάσταση λειτουργίας CBC.



Σχήμα 1.9. Κατάσταση λειτουργίας CFB.



Σχήμα 1.10. Κατάσταση λειτουργίας OFB.



Σχήμα 1.11. Κατάσταση λειτουργίας CTR

Οι κρυπτογραφικοί αλγόριθμοι τμήματος εφαρμόζουν επαναληπτικά μια σύνθετη κρυπτογραφική λειτουργία, η οποία ονομάζεται γύρος κρυπτογράφησης (round function), και αντιστοιχίζει ένα τμήμα μήκους n bit σε ένα άλλο τμήμα μήκους πάλι n bit. Ο αριθμός των κρυπτογραφικών αυτών γύρων συμβολίζεται με r. Σε κάθε εφαρμογή της ενός γύρου χρησιμοποιείται ένα μέρος κλειδιού (sub key) ki ($1 \leq i \leq r$) , το οποίο εξάγεται με συγκεκριμένη διαδικασία από το κλειδί k. Για να γίνει δυνατή η αποκρυπτογράφηση, για κάθε υπο-κλειδί ο γύρος της συνάρτησης πρέπει να είναι αναστρέψιμος (invertible), δηλαδή να παρέχει μια αντίστροφη λειτουργία. Οπότε, κατά την αποκρυπτογράφηση εφαρμόζονται οι γύροι με ανάστροφη σειρά.

Στους αλγόριθμους τμήματος οι γύροι κρυπτογράφησης επιτελούν πολύ πιο σύνθετες πράξεις από ότι μια απλή XOR πρόσθεση που επιτελούν οι αλγόριθμοι ροής, με απώτερο στόχο του να επιτυγχάνονται οι ιδιότητες της σύγχυσης και της διάχυσης. Η σύγχυση επιτυγχάνεται με μη γραμμικούς μετασχηματισμούς με τη μορφή των ειδικών μονάδων (συναρτήσεων) αντικατάστασης (S-box). Η διάχυση επιτυγχάνεται με γραμμικούς μετασχηματισμούς αντιμεταθέσεων (P-box). Στη συντριπτική πλειοψηφία των κρυπταλγορίθμων τμήματος, οι γύροι κρυπτογράφησης περιέχουν διαφόρων τύπων S-box και P-box, τα οποία πρέπει να έχουν κατάλληλα ποιοτικά χαρακτηριστικά για να είναι κρυπρογραφικά ισχυρός ο αλγόριθμος στο σύνολό του. Ο πιο γνωστός

κρυπτογραφικός αλγόριθμος τμήματος είναι ο AES (Advanced Encryption Standard), που αποτελεί από το 2000 το πρότυπο κρυπτογράφησης. Ο AES αντικατέστησε το προηγούμενο πρότυπο κρυπτογράφησης DES (Data Encryption Standard), ο οποίος δεν χρησιμοποιείται πια – παρά μόνο η παραλλαγή του 3DES, που έχει μεγαλύτερο μέγεθος κλειδιού.

1.3 Λογικές συναρτήσεις

Οι λογικές (Boolean) συναρτήσεις παίζουν σημαντικό ρόλο στην κατασκευή και την ασφάλεια πολλών συμμετρικών- κρυπταλγόριθμων. Στους αλγόριθμους ροής χρησιμοποιούνται για την κατασκευή των γεννητριών κλειδοροής. Στους αλγόριθμους τμήματος τα s-boxes αποτελούν ουσιαστικά διανυσματικές λογικές συναρτήσεις (δηλαδή συναρτήσεις που η έξοδος δεν είναι ένα μόνο bit αλλά περισσότερα), και σχεδιάζονται με βάση τις ιδιότητες των υποκείμενων λογικών συναρτήσεων. Οι κρυπτογραφικές συναρτήσεις πρέπει να πληρούν διάφορα κριτήρια για να μην είναι ευάλωτες σε γνωστές επιθέσεις κρυπτανάλυσης. Πιο συγκεκριμένα, η αντίσταση των κρυπτογραφικών συναρτήσεων με τις γνωστές επιθέσεις μπορεί να ποσοτικοποιηθεί μέσω κάποιων θεμελιωδών χαρακτηριστικών (που είναι σχετικά με την σύγχυση και την διάχυση) των λογικών (Boolean) συναρτήσεων που χρησιμοποιούνται σε αυτά. Έτσι, υπάρχει ένα σύνολο κρυπτογραφικών κριτηρίων για τις λογικές ή/και διανυσματικές συναρτήσεις, τα οποία ιδανικά πρέπει ταυτόχρονα να ικανοποιούνται. Μερικά από αυτά τα χαρακτηριστικά είναι ο αλγεβρικός βαθμός, η μη γραμμικότητα, η ανθεκτικότητα σε συσχετίσεις, η αλγεβρική ανθεκτικότητα ,ισοζύγιο κ.τ.λ. (θα αναλυθούν στην συνέχεια). Δεδομένου ότι κάποια κριτήρια αντικρούουν σε κάποια άλλα, θα πρέπει να υπάρχει ένας κατά το δυνατόν βέλτιστος συμβιβασμός. Ως εκ τούτου, παραμένει ανοιχτό ερευνητικό πρόβλημα η κατασκευή λογικών συναρτήσεων με τη βέλτιστη συμπεριφορά ως προς το σύνολο των κρυπτογραφικών κριτηρίων.

1.4 Σκοπός και Δομή Μεταπτυχιακής Εργασίας

Ο κύριος στόχος της παρούσας εργασίας είναι η κατασκευή κρυπτογραφικών λογικών συναρτήσεων, με την βοήθεια των εξελικτικών αλγορίθμων, και να μελετηθούν αν πληρούν κάποια βασικά κρυπτογραφικά κριτήρια. Οι εξελικτικοί αλγόριθμοι ανήκουν

στην κατηγορία των αλγορίθμων βελτιστοποίησης που στόχο έχουν να βρίσκουν λύση σε δύσκολα και πολύπλοκα προβλήματα παντός τύπου σε διάφορους τομείς επιστημών. Τα τελευταία χρόνια έχει αρχίσει να χρησιμοποιείται αυτή η μέθοδος και στην κρυπτογραφία, ακριβώς για την κατασκευή κρυπτογραφικών συναρτήσεων με καλά κρυπτογραφικά κριτήρια. Το είδος του εξελικτικού αλγόριθμου που θα χρησιμοποιηθεί για την κατασκευή των κρυπτογραφικών αλγορίθμων είναι οι γενετικοί αλγόριθμοι. Βασικός στόχος για την κατασκευή των αλγορίθμων είναι οι λογικές συναρτήσεις που θα παράγονται να είναι ισοβαρείς και να έχουν υψηλή μη γραμμικότητα. Ως βάση σύγκρισης και αναφοράς θα τεθεί η μη γραμμικότητα μίας γνωστής κρυπτογραφικής λογικής συνάρτησης (που αποκαλείται ως συνάρτηση Carlet-Feng), για την οποία είναι γνωστό ότι ικανοποιεί όλα τα βασικά κρυπτογραφικά κριτήρια. Η υλοποίηση του γενετικού αλγορίθμου θα γίνει σε γλώσσα JAVA. Στο τέλος των υλοποιήσεων των συναρτήσεων θα μελετηθούν με διάφορα εργαλεία τα κρυπτογραφικά κριτήρια των συναρτήσεων και θα αναλυθούν τα αποτελέσματα.

Η σπουδαιότητα του συγκεκριμένου θέματος έγκειται ακριβώς στο γεγονός του ότι η κατασκευή συναρτήσεων που να ικανοποιούν το σύνολο των επιθυμητών κρυπτογραφικών κριτηρίων παραμένει ανοιχτό ερευνητικό πρόβλημα, ενώ παράλληλα η προσέγγιση που ακολουθείται στην παρούσα έρευνα για την αντιμετώπιση του ζητήματος είναι σχετικά πρόσφατη και δεν έχει μελετηθεί εκτενώς – δείχνει ωστόσο ότι είναι ελπιδοφόρα.

Η δομή της διατριβής, όπως διαρθρώνεται ανά κεφάλαιο είναι η ακόλουθη:

Στο **Κεφάλαιο 2** ορίζονται κατ' αρχάς οι λογικές και οι διανυσματικές συναρτήσεις, που αποτελούν βασικά δομικά συστατικά σε συμμετρικούς κρυπτογραφικούς αλγορίθμους. Στην συνέχεια αναλύονται κάποιες βασικές ιδιότητες των κρυπτογραφικών συναρτήσεων (ο αλγεβρικός βαθμός, η μη γραμμικότητα , η ανθεκτικότητα σε συσχετίσεις, η αλγεβρική ανθεκτικότητα και το ισοζύγιο) και γίνεται αναφορά σε συναρτήσεις που ικανοποιούν τα παραπάνω κριτήρια (Carlet –Feng συνάρτηση).

Στο **Κεφάλαιο 3** αρχικά γίνεται αναφορά στους εξελικτικούς αλγόριθμους και σε ποιους τομείς έχουν χρησιμοποιηθεί. Στην συνέχεια αναλύονται οι γενετικοί αλγόριθμοι (με περιγραφή των βασικών χαρακτηριστικών, των βασικών τελεστών, των πλεονεκτημάτων και μειονεκτημάτων της χρήσης γενετικών αλγορίθμων, αλλά και αναφορά στους τομείς που έχουν χρησιμοποιηθεί) και αναλύεται ο τρόπος λειτουργίας

τους. Στην συνέχεια γίνεται αναφορά σε τεχνικές που έχουν ήδη χρησιμοποιηθεί για την κατασκευή κρυπτογραφικών συναρτήσεων με την βοήθεια των γενετικών αλγορίθμων.

Στο **Κεφάλαιο 4** παρουσιάζονται γενετικοί αλγόριθμοι, κατάλληλα τροποποιημένοι, για την παραγωγή νέων κρυπτογραφικών συναρτήσεων. Στο πλαίσιο αυτό, αναπτύχθηκαν τέσσερα προγράμματα για τους γενετικούς αλγορίθμους που χρησιμοποιήθηκαν – ένα πρόγραμμα για κάθε διαφορετικό αλγόριθμο. Αρχικά γίνεται αναφορά στο είδος του γενετικού αλγόριθμου αλλά και στα είδη των τελεστών που χρησιμοποιήθηκαν για την κατασκευή λογικών συναρτήσεων. Ακολούθως γίνεται εκτενής αναφορά της μεθοδολογίας (επιλογή πληθυσμού, παραμέτρων κτλ.) αλλά των αποτελεσμάτων που λάβαμε – όπου ως σχεδιαστικός στόχος τέθηκε η κατασκευή συναρτήσεων με 5, 6, 7, 8 και 9 μεταβλητές που να είναι ισοβαρείς και να έχουν μη γραμμικότητα τουλάχιστον ίση με την αντίστοιχη της Carlet-Feng συνάρτησης. Για τα αποτελέσματα παρέχεται πλήρης περιγραφή, συμπεριλαμβανομένων στοιχείων όπως σε ποια γενιά βρέθηκε η βέλτιστη λύση και ποιος ήταν ο χρόνος εκτέλεσης. Στις συναρτήσεις που προέκυψαν πραγματοποιήθηκε εκ των υστέρων έλεγχος για άλλα κρυπτογραφικά κριτήρια – ήτοι για τον αλγεβρικό βαθμό και την ανθεκτικότητα στις γρήγορες αλγεβρικές επιθέσεις – και καταδεικνύουμε ότι με την τεχνική μας κατέστη εφικτό να κατασκευαστούν συναρτήσεις ισάξιες – ή, σε ορισμένες περιπτώσεις, και καλύτερες – από τη συνάρτηση Carlet-Feng.

Τέλος στο **Κεφάλαιο 5** γίνεται η σύνοψη των αποτελεσμάτων καθώς και συζήτηση για μελλοντική έρευνα αναφορικά με την κατασκευή κρυπτογραφικών συναρτήσεων με την βοήθεια των εξελικτικών αλγορίθμων.

Κεφάλαιο 2

Κρυπτογραφικές ιδιότητες συναρτήσεων

Οι κρυπτογραφικές συναρτήσεις, για να είναι αποτελεσματικές και να προσφέρουν ασφάλεια ώστε να μην είναι επιρρεπείς σε διάφορων ειδών επιθέσεις ,θα πρέπει να ικανοποιούν ταυτόχρονα διάφορα κριτήρια. Η σχεδιαστική πρόκληση έγκειται στο γεγονός ότι είναι δύσκολη η επίτευξη όλων ταυτόχρονα των επιθυμητών κρυπτογραφικών κριτηρίων. Στο συγκεκριμένο κεφάλαιο ορίζονται κατ' αρχάς οι λογικές και οι διανυσματικές συναρτήσεις, που αποτελούν βασικά δομικά συστατικά σε συμμετρικούς κρυπτογραφικούς αλγορίθμους. Στην συνέχεια αναλύονται κάποιες βασικές ιδιότητες των κρυπτογραφικών συναρτήσεων και γίνεται αναφορά σε συναρτήσεις που ικανοποιούν τα παραπάνω κριτήρια.

2.1 Λογικές και Διανυσματικές Συναρτήσεις

Για την κατασκευή κρυπτογραφικών αλγορίθμων, σημαντικό ρόλο έχουν οι λογικές συναρτήσεις και οι ιδιότητές τους. Στην παρούσα ενότητα θα δοθούν οι ορισμοί των λογικών και διανυσματικών συναρτήσεων και θα φανεί πώς συνδέονται οι λογικές με τις διανυσματικές συναρτήσεις.

2.1.1 Λογικές Συναρτήσεις (Boolean functions)

Οι Λογικές Συναρτήσεις έχουν σημαντικό ρόλο στην κατασκευή και την ασφάλεια πολλών συμμετρικών συναρτήσεων. Στους αλγόριθμους ροής (Stream ciphers) συνδυάζουν τις εξόδους σε διάφορους γραμμικούς (ή και μη γραμμικούς) καταχωρητές ολίσθησης ανάδρασης, ή φιλτράρουν και συνδυάζουν τα περιεχόμενα ενός μόνο καταχωρητή. Με άλλα λόγια, στους αλγόριθμους ροής υπεισέρχονται στην κατασκευή των γεννητριών κλειδοροής (keystream generators), οι οποίες στην έξοδο παράγουν μια ψευδό-τυχαία ακολουθία που προστίθεται bit-προς-bit με το αρχικό μήνυμα. Από την άλλη πλευρά, στους αλγόριθμους τμήματος (Block ciphers) τα s-box έχουν σχεδιαστεί με κατάλληλη σύνθεση μη γραμμικών Boolean συναρτήσεων.

Μια λογική (Boolean) συνάρτηση f (x) με n μεταβλητές είναι συνάρτηση από το σύνολο όλων των δυαδικών διανυσμάτων μήκους n της μορφής $X=(X_1,...X_n)$ στο διάστημα {0,1}. Ο αριθμός n των μεταβλητών (δηλαδή το πλήθος των βαθμίδων του καταχωρητή που τροφοδοτούν την συνάρτηση) είναι σπανίως μεγάλος, στην πράξη. Στην περίπτωση των αλγορίθμων ροής ήταν μέχρι πρόσφατα λιγότερο από 10 και τώρα είναι πιο συχνά μικρότερη από 20[35].

Μία λογική συνάρτηση n μεταβλητών μπορεί να αναπαρασταθεί από τον πίνακα αληθείας της (truth-table). Ο πίνακας αληθείας αποτυπώνει, για κάθε πιθανή n-άδα εισόδου της συνάρτησης, ποια θα είναι η έξοδός της.

Παρακάτω παρατίθεται πίνακας αληθείας μίας τυχαίας συνάρτησης f 3 μεταβλητών (Πιν.2.1).

X1	X2	X3	f(X)
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 2.1. Πίνακας τριών μεταβλητών Boolean συνάρτησης

Στον παραπάνω πίνακα παρατηρούμε ότι η συνάρτηση f(x) είναι ίση με 1 στις περιπτώσεις {(001),(010),(100),(111)}.

Ορισμός 2.1 :Βάρος (weight) μιας συνάρτησης wt(f) ονομάζεται το πλήθος των "1" που υπάρχουν στη στήλη εξόδου στον πίνακα αληθείας της[47].

Για παράδειγμα, στον Πίνακα 2.1 το βάρος της συνάρτησης wt(f)= 4 γιατί στην έξοδο της f(x) έχουμε τέσσερις άσσους.

Ορισμός 2.2 :Ισοβαρής (balanced) συνάρτηση λέγεται η συνάρτηση που στην έξοδο της έχει ίσο πλήθος «0» και «1»[47].

Για παράδειγμα, στον «Πίνακα 2.1» η συνάρτηση είναι ισοβαρής γιατί έχει τέσσερα «0» και τέσσερα «1».

Ορισμός 2.3 : Η μαθηματική αναπαράσταση των λογικών συναρτήσεων που χρησιμοποιείται συνήθως στην κρυπτογραφία είναι μια πολυωνυμική αναπαράσταση με το όνομα **Αλγεβρική κανονική μορφή (Algebraic Normal Form - ANF).** Η ANF υπάρχει και είναι μοναδική για κάθε συνάρτηση και έχει την παρακάτω μορφή:

$$f(x_1, x_2, \dots, x_n) = c_0 + c_1x_1 + \cdots + c_nx_n + c_{12}x_1x_2 + c_{13}x_1x_3 + \cdots . . + c_{12\dots n}x_1x_2 \dots x_n$$

όπου οι συντελεστές c_k λαμβάνουν τιμές είτε «0» είτε 1» (είναι δηλαδή στοιχεία του πεπερασμένου σώματος GF(2)={0,1}), ενώ η πρόσθεση γίνεται πάνω σε αυτό το πεπερασμένο σώμα (δηλαδή 0+0=1+1=0 και 0+1=1+0=1) – πρόκειται δηλαδή ουσιαστικά για πρόσθεση XOR. Ουσιαστικά λοιπόν, η ANF είναι μια αποκλειστική διάζευξη (xor) του αθροίσματος κάποιων γινομένων μεταβλητών της συνάρτησης. Οι μεταβλητές $X_1, X_2 \dots, X_n$ εμφανίζονται με εκθέτες μικρότερους ή ίσους με 1 γιατί αντιπροσωπεύουν bits[35].

Στον «Πιν. 2.1» η αλγεβρική κανονική μορφή της συνάρτησης f(x) είναι η : (1⊕x1)(1⊕x2)x3 ⊕ (1⊕x1)x2(1 ⊕ x3) ⊕ x1(1⊕x2) (1 ⊕x3) ⊕ x1 x2 x3= x1⊕x2⊕x3. Το (1⊕x1)(1⊕x2)x3 είναι το (0,0,1), το (1⊕x1)x2(1 ⊕ x3) είναι το (0,1,0) , το x1(1⊕x2) (1 ⊕x3) είναι το (1,0,0) και το x1 x2 x3 είναι το (1,1,1). Ουσιαστικά από τον πίνακα αληθείας, για το σχηματισμό της ANF υπεισέρχονται οι

γραμμές του πίνακα που έχουν την μονάδα ως έξοδο και αναπαριστώνται οι είσοδοί τους ως εξής : η τιμή 0 μιας μεταβλητής x_n ως $(1 \oplus x_n)$ και η τιμή 1 ως x_n.

Ορισμός 2.4 : Αλγεβρικός Βαθμός (Algebraic degree -deg (f)) μιας συνάρτησης f(x) είναι το πλήθος των μεταβλητών που εμφανίζονται στο μεγαλύτερο γινόμενο της Αλγεβρικής Κανονικής Μορφή της συνάρτησης f(x) [48] .

Στον «Πιν. 2.1» ο βαθμός της συνάρτησης είναι 1 γιατί η ANF είναι η $x1 \oplus x2 \oplus x3$ (δηλαδή υπάρχουν γινόμενα μόνο μιας μεταβλητής) . Ως άλλο παράδειγμα , αν θεωρήσουμε τη συνάρτηση $f(x)=x1x2 \oplus x1x2x3 \oplus x3$, ο βαθμός αυτής της συνάρτησης είναι deg (f)=3 διότι στην ANF υπάρχει γινόμενο τριών μεταβλητών.

Ορισμός 2.5 : Συναρτήσεις affine είναι οι συναρτήσεις με την απλούστερη μορφή ANF . Ουσιαστικά είναι συναρτήσεις με βαθμό ίσο με 0 ή 1) [35]. Αν ο σταθερός όρος c_0 στην Αλγεβρική Κανονική Μορφή μίας affine συνάρτησης f είναι ίσος με 1 τότε ονομάζονται **γραμμικές (linear) συναρτήσεις**.

Στον «Πιν. 2.1» η συνάρτηση με ANF $x1 \oplus x2 \oplus x3$ είναι γραμμική γιατί ο βαθμός της deg(f) =1και επιπλέον, ο σταθερός όρος στην ANF ισούται με 0. Η **συμπληρωματική** συνάρτηση της f, δηλαδή η συνάρτηση που διαφέρει από την f σε όλες τις εξόδους του πίνακα αληθείας, έχει προφανώς ANF $x1 \oplus x2 \oplus x3 \oplus 1$, οπότε αυτή η συνάρτηση είναι affine.

Ορισμός 2.6 : Απόσταση Hamming (Hamming distance) μεταξύ δύο συναρτήσεων f(x) και g(x) είναι το πλήθος των θέσεων στον πίνακα αληθείας στις οποίες $f(x) \neq g(x)$ [35]. Προφανώς, η απόσταση Hamming των f,g ισούται με $wt(f \oplus g)$.

Παράδειγμα 1:Έστω η f(x) του πίνακα 2.1 και g(x) της οποίας οι αντίστοιχες έξοδοι στον πίνακα αληθείας φαίνονται στον παρακάτω πίνακα (Πιν. 2.2):

f(x)	g(x)
0	0
1	0

1	1
0	1
1	0
0	1
0	0
1	1

Πίνακας 2.2. Έξοδοι συναρτήσεων f(x) ,g(x)

Τότε η απόσταση Hamming ισούται με 3 γιατί διαφέρουν στις εξόδους τους σε 3 σημεία. Είναι εύκολο να δει κανείς ότι wt(f⊕g)=3.

2.1.2 Διανυσματικές Συναρτήσεις (vectorial functions)

Οι λογικές συναρτήσεις (Boolean functions) διαδραματίζουν σημαντικό ρόλο όχι μόνο στους κρυπταλγορίθμους ροής αλλά και στους κρυπταλγόριθμους τμήματος (block ciphers). Μια πρώτη παρατήρηση είναι ότι ο αλγόριθμος κρυπτογράφησης δέχεται ως είσοδο ένα δυαδικό διάνυσμα {x1,..., xn} (ένα μπλοκ κειμένου) και εξάγει ένα δυαδικό διάνυσμα {Y1,.., Yn} (ένα μπλοκ κρυπτογραφημένης εξόδου). Ουσιαστικά, οι έξοδοι Y1,. . . , Yn μπορούν να εκληφθούν ως έξοδοι n λογικών συναρτήσεων (Boolean functions), οι οποίες διαφοροποιούνται ανάλογα με το μυστικό κλειδί, των οποίων η κοινή είσοδος είναι η {x1,..., xn} [06].

Στους κρυπταλγορίθμους τμήματος, όπου συναντώνται περιπτώσεις συναρτήσεων με πολλές εξόδους, αυτές ονομάζονται διανυσματικές συναρτήσεις, ως γενίκευση των λογικών συναρτήσεων που έχουν μόνο μία έξοδο. Συγκεκριμένα, στους κρυπταλγορίθμους τμήματος χρησιμοποιούνται τα s-box (μονάδες αντικατάστασης) ,που βασίζονται στις ιδιότητες των λογικών συναρτήσεων,, και χρησιμοποιούνται για να επιτευχθεί η κρυπτογραφική ιδιότητα της σύγχυσης. Επίσης στους κρυπταλγορίθμους τμήματος χρησιμοποιούνται και τα p-box (μονάδες αντιμετάθεσης) που με γραμμικούς σχηματισμούς μεταθέσεων επιτυγχάνουν την κρυπτογραφική ιδιότητα της διάχυσης. Οι διανυσματικές συναρτήσεις κι συγκεκριμένα τα s-box χρησιμοποιούνται επίσης και στις ψευδό-τυχαίες γεννήτριες των κρυπταλγόριθμων ροής. Συγκεκριμένα συνδυάζουν τις εξόδους από n γραμμικούς καταχωρητές ολίσθησης με ανάδραση(LFSR) ή φιλτράρουν το περιεχόμενο ενός, και στη συνέχεια

παράγουν m bits σε κάθε κύκλο αντί ενός, το οποίο αυξάνει την ταχύτητα του κρυπταλγόριθμου[07].

Ορισμός 2.7: Αν μια συνάρτηση F έχει n εισόδους και m εξόδους (όπου m>1) τότε η συνάρτηση λέγεται **διανυσματική λογική συνάρτηση (vectorial function)** [48] και συμβολίζεται με F(n,m). Ουσιαστικά αυτές οι συναρτήσεις έχουν σαν εξόδους περισσότερα από ένα bit. Κάθε τέτοια συνάρτηση μπορεί να θεωρηθεί ότι αποτελεί σύνολο m λογικών συναρτήσεων, των n μεταβλητών η κάθε μια. Έτσι, η διανυσματική συνάρτηση f μπορεί να παρουσιαστεί ως εξής:

$$F(X_1,\dots,X_n) = (F^1(X_1,\dots,X_n),\dots,F^m(X_1,\dots,X_n),$$

όπου οι λογικές συναρτήσεις $F^1,\dots,F^m$ ονομάζονται **συνιστώσες συναρτήσεις** (component functions)της συνάρτησης F [05].Παρακάτω ακολουθεί παράδειγμα με διανυσματική λογική συνάρτηση με τρεις εισόδους και δύο εξόδους (Πιν. 2.3):

Είσοδος				**Έξοδος**	
A/A	**X1**	**X2**	**X3**	**Y1**	**Y2**
1	0	0	0	0	1
2	0	0	1	1	1
3	0	1	0	1	0
4	0	1	1	0	1
5	1	0	0	0	0
6	1	0	1	1	1
7	1	1	0	1	0
8	1	1	1	0	1

Πίνακας 2.3. Διανυσματική λογική συνάρτηση με τρεις εισόδους και δύο εξόδους.

Τα S-box που χρησιμοποιούνται στους περισσότερους αλγόριθμους κρυπτογράφησης τμήματος είναι μια αλληλουχία από μικρότερου μεγέθους S-box των 8 μεταβλητών το πολύ. Οι n μεταβλητές είναι πολύ μεγάλος αριθμός όταν n>=6 και ο λόγος είναι γιατί αν για να επισκεφτείς μια n – μεταβλητή θέλει ένα νανοδευτερόλεπτο

(10^{-9}δευτερόλεπτα) τότε για n=6 θα θέλει δισεκατομμύρια ώρες για να επισκεφτούν όλες τις n- μεταβλητές , και όσο μεγαλώνει το n τόσο μεγαλώνουν και οι ώρες προσπέλασης στις n- μεταβλητές. Ωστόσο, έξυπνες υπολογιστικές έρευνες μπορούν να δημιουργήσουν τέτοιες ενδιαφέρουσες συναρτήσεις [35].

Ορισμός 2.8 : Αλγεβρική κανονική μορφή (ANF) μιας **διανυσματικής λογικής συνάρτησης F(n,m)** Η ANF υπάρχει και είναι μοναδική για κάθε συνάρτηση και έχει την παρακάτω μορφή:

$$f(x_1, x_2, \dots, x_n) = c_0 + c_1x_1 + \cdots + c_nx_n + c_{12}x_1x_2 + c_{13}x_1x_3 + \cdots .. + c_{12\dots n}x_1x_2 \dots x_n$$

όπου οι συντελεστές c_k λαμβάνουν τιμές στο πεπερασμένο σώμα GF(2^m) (πρόκειται δηλαδή για διανύσματα με m μεταβλητές), ενώ η πρόσθεση γίνεται πάνω σε αυτό το πεπερασμένο σώμα [07].

Ορισμός 2.9 : Ο αλγεβρικός βαθμός μιας διανυσματικής λογικής συνάρτησης F(n,m) είναι ο μέγιστος αλγεβρικός βαθμός από τις συνιστώσες συναρτήσεις (component functions) της συνάρτησης f [05].

Για τον Πίνακα 2.3 για να υπολογίσουμε την αλγεβρική κανονική μορφή (ANF) θα πρέπει να υπολογίσουμε πρώτα την αλγεβρική κανονική μορφή της συνάρτησης Y1 (όπως λογική συνάρτηση), στην συνέχεια της Y2: ο μεγαλύτερος βαθμός θα είναι και ο βαθμός της διανυσματικής συνάρτησης.

Ορισμός 2.10 Μη γραμμικός βαθμός μιας διανυσματικής λογικής συνάρτησης είναι ο ελάχιστος αλγεβρικός βαθμός από όλες τις μη μηδενικές συνιστώσες συναρτήσεις (component functions) της συνάρτησης f [28].

2.2 Ορισμός των Κρυπτογραφικών Ιδιοτήτων των Συναρτήσεων

Οι κρυπτογραφικές συναρτήσεις πρέπει να πληρούν διάφορα κριτήρια για να μην είναι ευάλωτες σε γνωστές επιθέσεις κρυπτανάλυσης. Πιο συγκεκριμένα, η αντίσταση των

κρυπτογραφικών συναρτήσεων έναντι γνωστών επιθέσεων μπορεί να ποσοτικοποιηθεί μέσω κάποιων θεμελιωδών χαρακτηριστικών (που είναι σχετικά με την σύγχυση και την διάχυση) των λογικών (Boolean) συναρτήσεων που χρησιμοποιούνται σε αυτά. Έτσι, υπάρχει ένα σύνολο κρυπτογραφικών κριτηρίων για τις λογικές ή/και διανυσματικές συναρτήσεις, τα οποία ιδανικά πρέπει να ικανοποιούνται όλα ταυτόχρονα. Τα πιο βασικά χαρακτηριστικά που θα μελετηθούν, θα οριστούν και θα αποτελέσουν πρότυπο για την εξέλιξη της έρευνας είναι: ο αλγεβρικός βαθμός, η μη γραμμικότητα, η ανθεκτικότητα σε συσχετίσεις, η αλγεβρική ανθεκτικότητα και το ισοζύγιο.

2.2.1 Αλγεβρικός Βαθμός (algebraic degree)

Ο αλγεβρικός βαθμός σε ένα μονώνυμο έχει οριστεί να είναι ίσο με το βάρος του μονώνυμου [23]. Ο αλγεβρικός βαθμός στις λογικές συναρτήσεις (Boolean functions), όπως είδαμε νωρίτερα, είναι το μέγιστο πλήθος μεταβλητών που εμφανίζεται σε κάποιο γινόμενο στην αλγεβρική κανονική μορφή(ANF) (βλέπε ορισμό 2.4). Στις διανυσματικές συναρτήσεις (vectorial functions), είναι ο μέγιστος αλγεβρικός βαθμός από τις συνιστώσες συναρτήσεις (component functions) της συνάρτησης f (βλέπε ορισμό 2.8).

Ο αλγεβρικός βαθμός μίας λογικής συνάρτησης αποτελεί ένα σημαντικό κρυπτογραφικό της χαρακτηριστικό. Για παράδειγμα, στην περίπτωση του συνδυαστικών συναρτήσεων, αν οι n γραμμικοί καταχωρητές ολίσθησης με ανάδραση (LFSR) έχει μήκη L1,..,Ln αντίστοιχα, και οι έξοδοί τους τροφοδοτούν τη συνάρτηση-συνδυαστή :

$$f(x_1, x_2, \dots, x_n) = c_0 + c_1x_1 + \cdots + c_nx_n + c_{12}x_1x_2 + c_{13}x_1x_3 + \cdots . . + c_{12\dots n}x_1x_2 \dots x_n$$

τότε η ακολουθία που παράγεται από την συνάρτηση f μπορεί να έχει γραμμική πολυπλοκότητα της τάξης του:

$$L \leq c_0 + c_1L_1 + \cdots + c_nL_n + c_{12}L_1L_2 + c_{13}L_1L_3 + \cdots . . + c_{12\dots n}L_1L_2 \dots L_n$$

.

Σε αυτή την περίπτωση, ο αλγεβρικός βαθμός της συνάρτησης f πρέπει να είναι υψηλός έτσι ώστε η γραμμική πολυπλοκότητα L (που αντιστοιχεί στο μέγεθος του μικρότερου

LFSR που μπορεί να παράγει την ακολουθία) να είναι υψηλή για να είναι ανθεκτικός ο αλγόριθμος στις επιθέσεις βάσει του αλγορίθμου Berlekamp-Massey, οι οποίες μπορούν να οδηγήσουν σε πρόβλεψη ολόκληρης της μυστικής κλειδοροής αν αυτή έχει χαμηλή γραμμική πολυπλοκότητα.

Και στην περίπτωση των κρυπταλγόριθμων τμήματος χρησιμοποιώντας λογικές συναρτήσεις με χαμηλό αλγεβρικό βαθμό είναι επιρρεπείς στις διαφορικές επιθέσεις. Συνεπώς κρυπτοσυσκευές που χρησιμοποιούν λογικές συναρτήσεις θα πρέπει να έχουν υψηλό αλγεβρικό βαθμό γιατί αλλιώς θα είναι ευάλωτες σε επιθέσεις [35].

2.2.2 Μη Γραμμικότητα (Nonlinearity)

Προκειμένου να παρασχεθεί η σύγχυση οι κρυπτογραφικές συναρτήσεις πρέπει να μη μπορούν να προσεγγιστούν ικανοποιητικά από κάποια συνάρτηση affine ή, με άλλα λόγια, να βρίσκονται σε μεγάλη απόσταση Hamming (βλέπε ορισμό 2.6) από όλες τις λειτουργίες affine (βλέπε ορισμό 2.5) [06]. Ισοδύναμα, η συγκεκριμένη κρυπτογραφική ιδιότητα μπορεί να περιγραφεί ως εξής: Δεδομένου ότι έξοδος κάθε λογικής συνάρτησης f έχει κάποια συσχέτιση με συγκεκριμένους γραμμικούς συνδυασμούς των εισόδων της. Αυτή η συσχέτιση θα πρέπει να είναι μικρή αλλιώς υπάρχει κίνδυνος για επιθέσεις σε τέτοια συστήματα [35].

Ορισμός 2.11 :Η **μη γραμμικότητα (nonlinearity)** μίας λογικής συνάρτησης nl(f) ισούται με το μικρότερο πλήθος θέσεων στον πίνακα αληθείας της οι οποίες, αν μεταβληθούν, θα μετατρέψουν τη συνάρτηση σε γραμμική ή affine (βαθμού 1) [48]. Από μαθηματικής άποψης είναι η εφαρμογή του τύπου nl(f)= min_g ($wt(f \oplus g)$) ,όπου το $wt(f \oplus g)$, είναι η απόσταση Hamming (βλέπε ορισμό 2.6) μεταξύ δύο συναρτήσεων f και g, για όλες τις γραμμικές συναρτήσεις g. Ουσιαστικά είναι η ελάχιστη απόσταση Hamming (βλέπε ορισμό 2.6) μεταξύ της f και μιας affine συνάρτησης [06].

Αντίστοιχα, η **μη γραμμικότητα τάξης r (r-th nonlinearity)** μίας συνάρτησης (συμβολίζεται με $nl_r(f)$) ισούται με το μικρότερο πλήθος θέσεων στον πίνακα αληθείας της οι οποίες, αν μεταβληθούν, θα μετατρέψουν τη συνάρτηση σε βαθμού το πολύ r. Δηλαδή, η τιμή της υπολογίζεται από τον τύπο min_g ($wt(f \oplus g)$), για όλες τις συναρτήσεις g, τέτοιες ώστε $deg(g) \leq r$. Από τους ορισμούς προκύπτει ότι $nl(f) \geq nl_2(f)$

≥ $nl_3(f)$… Σημειώνεται ότι η μη γραμμικότητα για τάξη μεγαλύτερη ή ίση του 2 δεν έχει μελετηθεί πολύ στη βιβλιογραφία, λόγω της εγγενούς δυσκολίας.

Ορισμός 2.12: Η **μη γραμμικότητα (nonlinearity)** nl(fF) μίας διανυσματικής λογικής συνάρτησης F(n,m) ισούται με την ελάχιστη μη γραμμικότητα όλων των συνιστωσών συναρτήσεων x ∈ f_2^n ⟼ u* f(x) ,u∈ f_2^m ,u≠0. Στην πραγματικότητα είναι η μικρότερη απόσταση Hamming (βλέπε ορισμό 2.6) μεταξύ όλων των συνιστωσών συναρτήσεων και όλων των affine συναρτήσεων στις n μεταβλητές [07].

Συναντάται ωστόσο, για τις διανυσματικές συναρτήσεις, και άλλος γενικότερος ορισμοί της μη γραμμικότητας. Συγκεκριμένα, κρυπτογραφική αξία έχει η ελάχιστη μη γραμμικότητα υπολογιζόμενη όχι μόνο στις συνιστώσες συναρτήσεις αλλά και σε όλους τους γραμμικούς συνδυασμούς αυτών.

Για να είναι ανθεκτικές οι κρυπτογραφικές συναρτήσεις σε επιθέσεις συσχέτισης(correlation attacks) ή επιθέσεις γραμμικής προσέγγισης (affine approximation attacks) θα πρέπει να είναι υψηλής μη γραμμικότητας[07]. Κυρίως έχει μελετηθεί η μη γραμμικότητα πρώτης τάξης (που λέγεται, απλά, μη γραμμικότητα) και λιγότερο η μη γραμμικότητα δεύτερης τάξης.

Η μέγιστη μη γραμμικότητα για n μεταβλητές κυμαίνεται μεταξύ του 2^{n-1} -$2^{\frac{n-1}{2}}$ και 2^{n-1} - $2^{\frac{n}{2}-1}$. Για n άρτιο έχουμε τη μέγιστη δυνατή μη γραμμικότητα ίση με 2^{n-1} - $2^{\frac{n}{2}-1}$ και οι συναρτήσεις που το επιτυγχάνουν ονομάζονται Bent. Για n περιττό δεν γνωρίζουμε ποια είναι η μέγιστη δυνατή τιμή της μη γραμμικότητας που μπορεί να επιτευχθεί: για n=1,3,5,7 η μέγιστη μη γραμμικότητα είναι ίση με 2^{n-1} -$2^{\frac{n-1}{2}}$, αλλά για n ≥15 έχουν κατασκευαστεί συναρτήσεις με μη γραμμικότητα μεγαλύτερη από την 2^{n-1} -$2^{\frac{n-1}{2}}$. Η τιμή 2^{n-1} -$2^{\frac{n-1}{2}}$ ονομάζεται τετραγωνικό όριο (quadratic bound) και ο λόγος είναι γιατί αυτή η μη γραμμικότητα επιτυγχάνεται με τις τετραγωνικές συναρτήσεις (quadratic functions) – δηλαδή τις συναρτήσεις με αλγεβρικό βαθμό 2 [06].

2.2.3 Ισοζύγιο (Balanceness)

Για να αποφευχθεί η στατιστική εξάρτηση μεταξύ της εισόδου και της εξόδου μιας λογικής συνάρτησης, η οποία μπορεί να χρησιμοποιηθεί για επιθέσεις, οι κρυπτογραφικές συναρτήσεις θα πρέπει να είναι ισοζυγισμένες. Με άλλα λόγια, θα πρέπει οι έξοδοι των κρυπτογραφικών συναρτήσεων να είναι ομοιόμορφα κατανεμημένες, δηλαδή να είναι ισοβαρείς (βλέπε ορισμό 2.2). Μια λογική συνάρτηση είναι ισοβαρής αν και μόνο αν έχει βάρος Hamming ίσο με 2^{n-1}[07].

Μια διανυσματική λογική συνάρτηση με n εισόδους και m εξόδους είναι ισοβαρής όταν κάθε πιθανή m-άδα από bits εμφανίζεται στην έξοδο του πίνακα αληθείας ακριβώς 2^{n-m} φορές. Μία διανυσματική λογική συνάρτηση είναι ισοβαρής αν κάθε μη μηδενικός γραμμικός συνδυασμός των συνιστωσών της συναρτήσεων είναι ισοβαρής. Τα s-box που χρησιμοποιούνται στους κρυπταλγόριθμους τμήματος και στους κρυπταλγόριθμους ροής προτιμούνται να είναι ισοβαρή [07].

2.2.4 Αλγεβρική Ανθεκτικότητα (Algebraic immunity)

Οι αλγεβρικές επιθέσεις εφαρμόζονται τόσο σε κρυπταλγόριθμους ροής όσο και σε κρυπταλγόριθμους τμήματος. Βασίζονται στη χρήση έξυπνων τεχνικών προκειμένου να απλοποιηθούν οι σύνθετες μαθηματικές εκφράσεις που περιγράφουν τα συμμετρικά κρυπτογραφικά συστήματα, κατά τρόπο τέτοιο ώστε να είναι υπολογιστικά εφικτή η ανάκτηση από αυτές του μυστικού κλειδιού. Από την μελέτη των αλγεβρικών επιθέσεων ανέκυψε το κρυπτογραφικό κριτήριο της αλγεβρικής ανθεκτικότητας (Algebraic immunity).

Ορισμός 2.14: **Αλγεβρική ανθεκτικότητα (AI)** μίας λογικής συνάρτησης f είναι ο ελάχιστος βαθμός που μπορεί να έχει μία μη μηδενική συνάρτηση g τέτοια ώστε είτε f*g=0 ή (f⊕1)g=0. Τέτοιες συναρτήσεις g είναι γνωστά ως εκμηδενιστές (annihilators) της f ή της συμπληρωματικής της f⊕1. Γι' αυτόν τον λόγο η αλγεβρική ανθεκτικότητα ονομάζεται από κάποιους ερευνητές και εκ-μηδενιστική ανθεκτικότητα (annihilator immunity). Η μέγιστη δυνατή αλγεβρική ανθεκτικότητα (AI) που μπορεί να επιτευχθεί από μία συνάρτηση n μεταβλητών έχει οριοθετηθεί στη τιμή $\left\lceil \frac{n}{2} \right\rceil$ [35].

Η αλγεβρική ανθεκτικότητα πρέπει να είναι υψηλή, αλλά δεν είναι ικανή συνθήκη για την αντιμετώπιση όλων των ειδών των αλγεβρικών επιθέσεων. Υπάρχει μια παραλλαγή των αλγεβρικών επιθέσεων οι οποίες είναι γνωστές ως γρήγορες αλγεβρικές επιθέσεις(fast algebraic attacks) και είναι ισχυρότερες από τις απλές αλγεβρικές επιθέσεις. Αυτό πρακτικά σημαίνει ότι αν μια κρυπτογραφική συνάρτηση έχει τη μέγιστη αλγεβρική ανθεκτικότητα, δεν είναι απαραίτητα ανθεκτική στις γρήγορες αλγεβρικές επιθέσεις. Αυτό είχε σαν αποτέλεσμα να ανακύψουν τα κρυπτογραφικά κριτήρια της γρήγορης αλγεβρικής αντίστασης (fast algebraic resistance) και της γρήγορης αλγεβρικής ανθεκτικότητας (fast algebraic immunity).

Ορισμός 2.15: Για 2 μεταβλητές d_g, d_h>d_g λέμε ότι η συνάρτηση f ικανοποιεί μία (d_g, d_h)-σχέση αν υπάρχουν για, δύο μη μηδενικές συναρτήσεις g, h τέτοιές ώστε f*g=h, deg(g)<deg(h) , όπου deg(g)= d_g και deg(h)= d_h. Η ανθεκτικότητα έναντι των γρήγορων αλγεβρικών επιθέσεων (**Γρήγορη αλγεβρική αντίσταση (FAR**)) αποτιμάται με την ελάχιστη τιμή $(d_g + d_h)$ για όλες τις (d_g, d_h)-σχέσεις της f. Δεδομένου ότι f*1 = f, είναι σαφές ότι οι τιμές των αθροισμάτων αυτών οροθετούνται πάνω από τον αλγεβρικό βαθμό της συνάρτησης f. Επίσης, αποδεικνύεται ότι για οποιαδήποτε (d_g, d_h)-σχέση , η μεταβλητή d_h είναι μεγαλύτερη η ίση με την αλγεβρική ανθεκτικότητα της συνάρτησης f(AI(f)).

Ορισμός 2.16: Μια προσπάθεια ενοποίησης των κριτηρίων της αλγεβρικής ανθεκτικότητας και της ανθεκτικότητας έναντι των γρήγορων αλγεβρικών επιθέσεων είναι η **Γρήγορη αλγεβρική ανθεκτικότητα (FAI)** όπου δίνεται από τον τύπο :

$$FAI(f)=\min\{2*AI(f), FAR(f)\}$$

Ωστόσο αυτό το κριτήριο θεωρείται ανεπαρκές και ένας λόγος είναι ο εξής: αν π.χ. για αλγεβρική ανθεκτικότητα AI(f)=6 ή AI(f)=7 και η γρήγορη αλγεβρική αντίσταση είναι FAR(f)=12 τότε, και για τις δύο τιμές της αλγεβρικής ανθεκτικότητας η γρήγορη αλγεβρική ανθεκτικότητα είναι ίση με FAI(f)=12 [23](δηλαδή, οι δύο συναρτήσεις αποτιμώνται το ίδιο, παρόλο που έχουν διαφορετική αλγεβρική ανθεκτικότητα).

Έχει αποδειχθεί ότι για κάθε συνάρτηση f με n μεταβλητές υπάρχουν συναρτήσεις g, h τέτοιες ώστε f*g=h, όπου $d_g + d_h = n$. Αυτό σημαίνει ότι η μέγιστη δυνατή τιμή που μπορεί να λάβει η γρήγορη αλγεβρική ανθεκτικότητα FAI είναι n. Περαιτέρω, έχει αποδειχθεί ότι αν μία συνάρτηση έχει τη μέγιστη δυνατή FAI (ίση με n), τότε υποχρεωτικά έχει και τη μέγιστη δυνατή AI ίση με $\left\lceil \frac{n}{2} \right\rceil$ [26]. Συνεπώς, αν γνωρίζουμε ότι μία συνάρτηση έχει μέγιστο FAI, δεν χρειάζεται να ελέγξουμε την τιμή του AI αφού θα είναι εγγυημένα μέγιστη (αυτό το αποτέλεσμα θα αξιοποιηθεί στην περαιτέρω ανάλυσή μας, αφού για τις συναρτήσεις που θα κατασκευάσουμε θα εξετάζουμε απευθείας τη FAI).

Ένα άλλο σημαντικό αποτέλεσμα είναι το ότι μία ισοβαρής συνάρτηση με n μεταβλητές μπορεί να έχει τη μέγιστη FAI (ίση με n) μόνο αν το n έχει τη μορφή $n=2^s+1$ για κάποιον ακέραιο s [22]. Άρα, η μέγιστη δυνατή FAI μπορεί να επιτευχθεί μόνο για συναρτήσεις με 5, 9, 17 κ.ο.κ. μεταβλητές. Για όλες τις υπόλοιπες συναρτήσεις, η καλύτερη δυνατή τιμή που μπορούμε να επιτύχουμε για τη FAI είναι n-1.

2.2.5 Ανθεκτικότητα Σε Συσχετίσεις (correlation immunity)

Ένα είδος επιθέσεων που εισήγαγε ο Siegenthaler το 1985 ήταν οι επιθέσεις συσχετίσεων. Σκοπός αυτών τον επιθέσεων, οι οποίες αρχικά εφαρμόστηκαν σε γεννήτριες κλειδοροής που αποτελούνται από μία συνάρτηση-συνδυαστή n μεταβλητών στις εισόδους της οποίας εφαρμόζονται οι έξοδοι n LFSR αντιστοίχως, ήταν η ανάκτηση της αρχικής κατάστασης του κάθε LFSR ξεχωριστά γνωρίζοντας κάποια bits της κλειδοροής. Ουσιαστικά ήταν ένα είδος επίθεσης που αναφερόταν σε όλες τις γεννήτριες κλειδοροής. Το συγκεκριμένο είδος επιθέσεων είχε σαν αποτέλεσμα να μελετηθεί ένα ακόμη κριτήριο για την κατασκευή κρυπτογραφικών αλγορίθμων, αυτό της ανθεκτικότητας σε συσχετίσεις[33].

Ορισμός 2.17: Ανθεκτικότητα σε συσχετίσεις (correlation immunity) μιας συνάρτησης f η μέγιστη τιμή m τέτοια ώστε, αν τηρούνται σταθερές m είσοδοι της συνάρτησης, τότε η υπο-συνάρτηση που προκύπτει είναι ισοβαρής [23].

Οι κρυπτογραφικές συναρτήσεις πρέπει να έχουν υψηλή ανθεκτικότητα σε συσχετίσεις για να αποφεύγονται οι αντίστοιχες επιθέσεις. Συναρτήσεις που έχουν ανθεκτικότητα

τάξης m σε συσχετίσεις και m-**ελαστικές(m-resilient)** συναρτήσεις [35]. Λογικές συναρτήσεις οι οποίες είναι m-ελαστικές για κάποια τιμή του m>0 αλλά δεν είναι ισοβαρείς ονομάζονται **m-th βαθμού ανθεκτικές σε συσχετίσεις** (m-th order correlation-immune)[35].

2.3 Συναρτήσεις που ικανοποιούν τα βασικά κρυπτογραφικά κριτήρια

Λόγω της σπουδαιότητας των ανωτέρω κρυπτογραφικών κριτηρίων, έχουν υπάρξει πολλές κατασκευές λογικών συναρτήσεων προσανατολισμένες στην ικανοποίηση κάποιων εκ των κριτηρίων αυτών. Ένα χαρακτηριστικό παράδειγμα είναι οι συναρτήσεις bent που επιτυγχάνουν, όπως είδαμε νωρίτερα, τη μέγιστη μη γραμμικότητα που είναι ίση με 2^{n-1} - $2^{\frac{n}{2}-1}$. Οι συναρτήσεις αυτές είναι ανθεκτικές στις επιθέσεις συσχετίσεων (correlation attacks) αλλά και γραμμικών προσεγγίσεων (affine approximation attacks). Ωστόσο οι συγκεκριμένες συναρτήσεις δεν είναι ισοβαρείς, πράγμα το οποίο δεν είναι κρυπτογραφικά επιθυμητό. Γενικότερα, αποτελεί σχεδιαστική πρόκληση η σχεδίαση συναρτήσεων που να ικανοποιούν ταυτόχρονα όλα τα βασικά κρυπτογραφικά κριτήρια.

Ένα είδος κρυπτογραφικών συναρτήσεων, που ικανοποιούν σχεδόν όλα τα βασικά χαρακτηριστικά που αναφέρθηκαν στην ενότητα 2.2, προτάθηκε από τους Claude Carlet και Keqin Feng [08]. Οι συγκεκριμένες κρυπτογραφικές συναρτήσεις (στο εξής θα ονομάζονται ως Carlet- Feng συναρτήσεις) είναι ανθεκτικές τόσο στις αλγεβρικές επιθέσεις όσο και στις γρήγορες αλγεβρικές επιθέσεις και μπορούν να χρησιμοποιηθούν στους κρυπταλγόριθμους ροής. Ουσιαστικά σχεδιάστηκαν ακριβώς με γνώμονα την αντιμετώπιση των αλγεβρικών επιθέσεων, αλλά ταυτόχρονα είναι ισοβαρείς, έχουν υψηλό αλγεβρικό βαθμό και υψηλή μη γραμμικότητα.

Οι συναρτήσεις αυτές περιγράφονται καλύτερα με την αναπαράσταση σε πεπερασμένα σώματα. Συγκεκριμένα, κάθε πιθανή n-άδα εισόδου της συνάρτησης στον πίνακα αληθείας εκλαμβάνεται ως στοιχείο του πεπερασμένου σώματος $GF(2^n)$ με 2^nστοιχεία.

Ορισμός 2.18: Πεπερασμένο σώμα είναι ένα σώμα που περιέχει πεπερασμένο αριθμό στοιχείων, και ο αριθμός αυτός ονομάζεται τάξη του σώματος. Τα πεπερασμένα σώματα ονομάζονται και σώματα Galois[54].

Είναι γνωστό ότι κάθε στοιχείο ενός πεπερασμένου σώματος GF(2^n) έχει δύο πιθανές αναπαραστάσεις: είτε μία n-άδα από bits είτε δύναμη της μορφής α^k, όπου α ένα πρωταρχικό στοιχείο του σώματος (δηλαδή ένα στοιχείο τέτοιο ώστε όλες οι δυνάμεις $\alpha^0, \alpha^1, \alpha^2, \dots, \alpha^{2^n-2}$ να είναι διαφορετικά στοιχεία του σώματος και, άρα, οι δυνάμεις του πρωταρχικού στοιχείου «διατρέχουν» όλα τα μη μηδενικά στοιχεία του σώματος).

Με την παραπάνω θεώρηση, μία συνάρτηση Carlet-Feng έχει έξοδο «1» στον πίνακα αληθείας της αν και μόνο αν η είσοδός της ανήκει στο σύνολο 0, $\alpha^0, \alpha^1, \alpha^2, \dots, \alpha^{2^{(n-1)}-2}$ (όπου α ένα οποιοδήποτε πρωταρχικό στοιχείο του σώματος και το n είναι ένας θετικός ακέραιος).

Κατ' αρχάς, όπως προκύπτει από τον ορισμό τους, οι συναρτήσεις αυτές είναι ισοβαρείς. Επίσης, αποδεικνύεται ότι έχουν πολύ υψηλό αλγεβρικό βαθμό, ίσο με n-1 (ουσιαστικά, αυτός είναι ο μέγιστος δυνατός αλγεβρικός βαθμός που μπορεί να επιτύχει μια ισοβαρής συνάρτηση n μεταβλητών). Επίσης, οι συγκεκριμένες συναρτήσεις (Carlet – Feng) έχουν βέλτιστη αλγεβρική ανθεκτικότητα ίση με $\left\lceil \frac{n}{2} \right\rceil$, γι' αυτό και είναι και ανθεκτικές στις αλγεβρικές επιθέσεις. Επίσης έχουν υψηλή μη γραμμικότητα. Οι Carlet-Feng έχουν αποδείξει ότι η μη γραμμικότητά τους ικανοποιεί τον τύπο:

$$nl(f) \geq 2^{n-1} - \frac{2ln2}{\pi} n2^{n/2}$$

Αν και στην πράξη παρατηρείται πολύ υψηλότερη μη γραμμικότητα από το ανωτέρω κάτω φράγμα (ενώ διάφορες βελτιώσεις αυτού του κάτω φράγματος έχουν έκτοτε αποδειχθεί).

Επίσης οι συναρτήσεις Carlet–Feng είναι ανθεκτικές στις γρήγορες αλγεβρικές επιθέσεις, αφού ισχύουν τα εξής:

α) Καμία μη μηδενική συνάρτηση g βαθμού το πολύ e και καμία συνάρτηση h βαθμού το πολύ d δεν υπάρχει τέτοια ώστε f*g=h ,όταν (e,d)=(1,n-2) για n περιττό και όταν (e,d)=(1,n-3) για n άρτιο.

β) Για e>1 ζευγάρια συναρτήσεων (g,h) βαθμών (e,d) αντίστοιχα τέτοια ώστε , e+d<n+1 ποτέ δεν έχουν παρατηρηθεί. Αυτό υποδηλώνει ότι η κατηγορία αυτή των λειτουργιών, ακόμα και αν δεν είναι πάντοτε η βέλτιστη κατά τις γρήγορες αλγεβρικές επιθέσεις έχουν μια πολύ καλή συμπεριφορά.
Η βέλτιση συμπεριφορά των συναρτήσεων Carlet-Feng ως προς τις γρήγορες αλγεβρικές επιθέσεις έχει αποδειχθεί μεταγενέστερα στο [22].

Παρακάτω παρατίθεται πίνακας για την συνάρτηση Carlet –Feng για n=4 έως n=11 , όπου παρατηρείται ότι οι συναρτήσεις αυτές έχουν υψηλό αλγεβρικό βαθμό (degree), είναι ισοβαρείς (βαθμός συνάρτησης n-1), έχουν υψηλή αλγεβρική ανθεκτικότητα (algebraic immunity)και καλή μη γραμμικότητα (nonlinearity) (Πιν. 2.4).

Συνάρτηση	**Βαθμός**	**Μη Γραμμικότητα**	**Αλγεβρική Ανθεκτικότητα**
n=4	3	4	2
n=5	4	10	3
n=6	5	24	3
n=7	6	54	4
n=8	7	112	4
n=9	8	232	5
n=10	9	484	5
n=11	10	980	6

Πίνακας 2.4. Κρυπτογραφικές ιδιότητες συναρτήσεων Carlet-Feng [54]

Με αφετηρία την πλούσια σε ιδιότητες αυτή οικογένεια συναρτήσεων, έχουν υπάρξει νέες κατασκευές που τροποποιούν τις συναρτήσεις Carlet-Feng χωρίς να μειώνεται η αλγεβρική ανθεκτικότητα [10] [20] [32][40]. Λόγω των συσσωρευμένων καλών κρυπτογραφικών ιδιοτήτων, οι συναρτήσεις Carlet-Feng θα αποτελέσουν την βάση για την κατασκευή νέων συναρτήσεων μέσω γενετικών αλγορίθμων που θα μελετηθούν και θα ελεγχθούν οι κρυπτογραφικές τους ιδιότητες (αναλυτικά στο κεφάλαιο 4).

Κεφάλαιο 3

Γενετικοί Αλγόριθμοι

Για την επίλυση δύσκολων προβλημάτων, ο χώρος της πληροφορικής οδηγήθηκε σε νέες τεχνικές αναζήτησης λύσεων και βελτιστοποίησης. Μια τεχνική που χρησιμοποιήθηκε σε πολλούς τομείς για να δίνονται λύσεις σε πολύπλοκα προβλήματα με την μέθοδο της βελτιστοποίησης, είναι οι εξελικτικοί αλγόριθμοι. Οι εξελικτικοί αλγόριθμοι χωρίζονται σε διάφορες κατηγορίες με βάση τον τρόπο υλοποίησης των αναζητήσεων. Μια από αυτές τις κατηγορίες είναι και οι γενετικοί αλγόριθμοι. Αξίζει να σημειωθεί ότι και στην κρυπτογραφία τα τελευταία χρόνια μελετάται η αξιοποίηση γενετικών αλγορίθμων για την κατασκευή κρυπτογραφικών συναρτήσεων με καλά κρυπτογραφικά χαρακτηριστικά. Στο κεφάλαιο αυτό αρχικά θα γίνει αναφορά στους εξελικτικούς αλγόριθμους και σε ποιους τομείς έχουν χρησιμοποιηθεί. Στην συνέχεια θα αναλυθούν οι γενετικοί αλγόριθμοι και θα περιγραφεί ο τρόπος λειτουργίας τους. Τέλος θα γίνει αναφορά σε εφαρμογές των γενετικών αλγορίθμων για την παραγωγή κρυπτογραφικών συναρτήσεων.

3.1 Εξελικτικοί Αλγόριθμοι (ΕΑ)

Οι εξελικτικοί αλγόριθμοι (Evolutionary Algorithms -ΕΑ) είναι στοχαστικές μέθοδοι αναζήτησης, οι οποίες έχουν εφαρμοστεί επιτυχώς σε πολλά προβλήματα αναζήτησης, βελτιστοποίησης και μηχανικής μάθησης. Σε αντίθεση με τις περισσότερες τεχνικές βελτιστοποίησης, οι ΕΑ χρησιμοποιούν έναν πιθανό πληθυσμό λύσεων, τον οποίο διαχειρίζονται με ανταγωνιστικό τρόπο, εφαρμόζοντας κάποιους τελεστές διαφοροποίησης, για την εύρεση μιας αρκετά ικανοποιητικής, αν και όχι συνολικά βέλτιστης, λύσης.

3.1.1 Αλγόριθμοι Βελτιστοποίησης - Μεθευρητικές Μέθοδοι

Όταν αναφερόμαστε στον όρο βελτιστοποίηση, εννοούμε την διαδικασία αναζήτησης μίας λύσης για κάποια προβλήματα που εκφράζονται με μία μορφή συνάρτησης f(x). Η αναζήτηση αυτής της λύσης μπορεί να οδηγήσει είτε στη μεγιστοποίηση είτε στην ελαχιστοποίηση της τιμής της συνάρτησης που περιγράφει το πρόβλημα. Συνήθως, αυτά τα προβλήματα υπόκεινται σε ορισμένους περιορισμούς. Ειδικότερα, τρεις είναι οι περιορισμοί που διαθέτουν τα προβλήματα βελτιστοποίησης και αυτοί είναι οι εξής :

1. Ένα σύνολο αγνώστων ή μεταβλητών του προβλήματος.

2. Ο τύπος της συνάρτησης που θέλουμε να βελτιστοποιήσουμε.

3. Και ένα σύνολο περιορισμών που θα πρέπει να πληροί η επιστρεφόμενη βέλτιστη λύση.

Οι αλγόριθμοι βελτιστοποίησης μπορούν να χωριστούν σε δύο κατηγορίες: στους ακριβείς αλγόριθμους (exact algorithms) και τους ευρετικούς αλγόριθμους (heuristic algorithms). Η κύρια διαφορά μεταξύ τους είναι ότι οι πρώτοι είναι σχεδιασμένοι με τέτοιο τρόπο ώστε να βρουν μια βέλτιστη λύση και απαιτείται χρόνος για να αποδείξουν ότι έχουν βρει την βέλτιστη λύση. Από την άλλη μεριά, οι ευρετικοί αλγόριθμοι δεν εγγυώνται κάτι τέτοιο [42].

Ορισμός 3.1 :Μεθευρετική μέθοδος είναι ένα υψηλού επιπέδου αλγοριθμικό πλαίσιο που παρέχει ένα πακέτο κατευθυντήριων γραμμών και στρατηγικών με στόχο να αναπτύξει έναν ευρετικό αλγόριθμο βελτιστοποίησης[42].

Ο όρος «μεθευρητική μέθοδος» επινοήθηκε από τον Fred Glover και αναλυτικά προέρχεται από το πρόθεμα «μετά» (πέρα από την έννοια του υψηλού επιπέδου) και την ελληνική λέξη «ευρετική» (ευρίσκω)[16]. Ουσιαστικά, οι μεθευρετικές μέθοδοι δεν είναι αλγόριθμοι, αλλά ένα συνεκτικό σύνολο ιδεών, εννοιών και λειτουργιών που μπορούν να χρησιμοποιηθούν για τη σχεδίαση ευρετικών αλγορίθμων βελτιστοποίησης. Οι μεθευρετικές μέθοδοι είναι μια πιο αφηρημένη διαδικασία που χρησιμοποιεί χαμηλού επιπέδου ευρετικούς αλγόριθμους ή αλγόριθμους αναζήτησης. Η αφορμή που οδήγησε στην ανάπτυξη των μεθευρετικών μεθόδων ήταν το ότι ένας ευρετικός αλγόριθμος δεν πρέπει απαραίτητα να εξαρτάται πλήρως από το πρόβλημα, αλλά το ότι θα μπορούσαν να αναπτυχθούν γενικές τεχνικές βελτιστοποίησης ώστε να είναι εφαρμόσιμες σε προβλήματα βελτιστοποίησης της καθημερινής ζωής.

3.1.2 Εισαγωγή στους εξελικτικούς αλγορίθμους

Έχοντας περιγράψει την έννοια της μεθευρητικής μεθόδου, οι εξελικτικοί αλγόριθμοι μπορούν να περιγραφούν ακριβέστερα ως μεθευρετικές μέθοδοι βελτιστοποίησης με βάση τον πληθυσμό, που έχουν προέλευση και έμπνευση από τον κόσμο της βιολογίας [01]. Βασίζονται στο μοντέλο φυσικής, βιολογικής εξέλιξης το οποίο προτάθηκε από τον Κάρολο Δαρβίνο στο έργο του "The Origin of Species". Η θεωρία της εξέλιξης του Δαρβίνου εξηγεί την προσαρμοστική αλλαγή των ειδών μέσω της αρχής της φυσικής επιλογής, η οποία ευνοεί την επιβίωση και την περαιτέρω εξέλιξη εκείνων των ειδών που είναι καλύτερα προσαρμοσμένα στις περιβαλλοντικές τους συνθήκες (η λεγόμενη "επιβίωση του ικανότερου") [11].

Οι εξελικτικοί αλγόριθμοι αποτελούνται από ένα σύνολο μεθόδων, η λειτουργία των οποίων βασίζεται σε διάφορους μηχανισμούς της φυσικής εξέλιξης, και οι οποίες χρησιμοποιούνται για να επιλύουν προβλήματα βελτιστοποίησης. Συγκεκριμένα μιμούνται τις φυσικές διαδικασίες της επιλογής (selection), αναπαραγωγής (reproduction), της μετάλλαξης (mutation) και της διασταύρωσης (crossover) ή αλλιώς του ανασυνδυασμού (recombination) και τις χρησιμοποιούν ως μηχανισμούς ή τελεστές αναζήτησης (search operators), ώστε να βρουν πιο γρήγορα καλύτερες λύσεις σε προβλήματα βελτιστοποίησης [42].

Οι πρώτες εφαρμογές των εξελικτικών αλγορίθμων στην επιστήμη των υπολογιστών εμφανίστηκαν στη δεκαετία του 1950. Στα μέσα της δεκαετίας του 1960 ο John Holland από το πανεπιστήμιο του Michigan εισήγαγε τις μεθόδους της βιολογικής αναπαραγωγής στους εξελικτικούς αλγόριθμους, διατυπώνοντας τους γενετικούς αλγόριθμους. Γι' αυτό και θεωρείται ο θεμελιωτής των εξελικτικών αλγορίθμων [27].

Οι εξελικτικοί αλγόριθμοι συχνά καταφέρνουν να επιφέρουν μια καλή κατά προσέγγιση λύση σε όλα τα είδη των προβλημάτων. Αυτό έχει σαν αποτέλεσμα να χρησιμοποιούνται σε διάφορους τομείς. Οι κυριότεροι τομείς είναι: η μηχανική, η τέχνη, η βιολογία, η οικονομία , το μαρκετινγκ, η γενετική, η επιχειρησιακή έρευνα, η ρομποτική, η τεχνητή νοημοσύνη, οι κοινωνικές επιστήμες, η φυσική, η πολιτική, η χημεία, ενώ τα τελευταία χρόνια μελετάται και η χρησιμοποίησή της στην κρυπτογραφία[37].

3.1.3 Ορολογία των εξελικτικών αλγορίθμων

Σε έναν ΕΑ ένας πληθυσμός ατόμων (population of individuals) είναι ένα υποσύνολο όλων των πιθανών λύσεων για το συγκεκριμένο πρόβλημα. Ο πληθυσμός οδηγείται (εξελίσσεται) προς τη λύση ενός προβλήματος βελτιστοποίησης με χρήση τελεστών (operators) στα άτομα του πληθυσμού. Οι τελεστές αυτοί τροποποιούν τα άτομα, εισάγουν νέα άτομα στον πληθυσμό ή αφαιρούν άτομα από αυτόν.

Ένα άτομο (individual) είναι η αναπαράσταση μιας πιθανής λύσης του προβλήματος βελτιστοποίησης, μια αναπαράσταση των μεταβλητών της λύσης αυτής. Το άτομο κατά συνέπεια περιέχει πληροφορίες για τη θέση της πιθανής λύσης στο χώρο λύσεων του προβλήματος. Επίσης περιέχει πληροφορίες για την καταλληλότητα της λύσης αυτής, κατά πόσο δηλαδή αυτή πλησιάζει τη βέλτιστη.

Οι τελεστές που επιδρούν στον πληθυσμό συχνά χωρίζονται σε δυο κατηγορίες. Στην πρώτη κατηγορία ανήκουν οι τελεστές οι οποίοι τροποποιούν τα άτομα του πληθυσμού και στη δεύτερη, εκείνοι που προσθέτουν ή αφαιρούν άτομα από τον πληθυσμό.

Στην πρώτη κατηγορία ανήκουν οι τελεστές μετάλλαξης (mutation) και διασταύρωσης (crossover). Οι τελεστές αυτής της κατηγορίας τροποποιούν τα άτομα, αυξάνοντας την ποικιλότητα του πληθυσμού και κατά συνέπεια δημιουργώντας νέες πιθανές λύσεις. Ο τελεστής της μετάλλαξης προσομοιώνει την αντίστοιχη φυσική διαδικασία - δηλαδή λαμβάνει ένα άτομο και τροποποιεί τις παραμέτρους του. Αντίστοιχα ο τελεστής της διασταύρωσης λαμβάνει δυο (ή και μερικές φορές περισσότερα) άτομα (γονείς) και δημιουργεί έναν ή περισσότερους απογόνους από αυτά. Οι απόγονοι λαμβάνουν κάποια από τα χαρακτηριστικά των γονέων.

Στην δεύτερη κατηγορία ανήκει ο τελεστής της επιλογής (selection) ή αναπαραγωγής (reproduction) όπως αλλιώς αναφέρεται. Σκοπός του είναι να επιλέξει τα καλύτερα άτομα του πληθυσμού τα οποία στη συνέχεια θα εισαχθούν στον τελεστή της διασταύρωσης.

Η συνάρτηση κόστους (pay-off fuction) ή καταλληλότητας (fitness) του ατόμου είναι το κριτήριο επιλογής ενός ατόμου από τον τελεστή επιλογής. Η συνάρτηση κόστους μετράει το πόσο κοντά βρίσκεται το κάθε άτομο του πληθυσμού στη βέλτιστη λύση.

Ο χώρος αναζήτησης σε έναν εξελικτικό αλγόριθμο ονομάζεται γενότυπο (genotype) και αποτελεί το σύνολο της γενετικής πληροφορίας .Ο χώρος αναζήτησης αναφέρεται

συχνά ως γονιδίωμα (genome). Κάθε γενότυπος αποτελείται από χρωμοσώματα (chromosomes) που αποτελούν λύσεις στο συγκεκριμένο πρόβλημα. Τα χρωμοσώματα με την σειρά τους αποτελούνται από γονίδια (genes), δηλαδή τα γονίδια αποτελούν ένα στοιχείο θέσης ενός χρωμοσώματος . Αλληλόμορφο (Allele) είναι η τιμή που παίρνει ένα γονίδιο για ένα συγκεκριμένο χρωμόσωμα. Το αποκωδικοποιημένο περιεχόμενο ενός χρωμοσώματος καλείται φαινότυπος (phenotype). Ουσιαστικά το φαινότυπο είναι η λύση του πληθυσμού για τον πραγματικό χώρο, στην οποία οι λύσεις εκπροσωπούνται με ένα τρόπο που εκπροσωπούνται σε καταστάσεις του πραγματικού κόσμου. Η αποκωδικοποίηση (decoding)είναι μια διαδικασία μετασχηματισμού μιας λύσης από το γονότυπο με το φαινότυπο. Αντίθετα η κωδικοποίηση (encoding) είναι μια διαδικασία μετασχηματισμού από τον φαινότυπο στο γονότυπο[13].

3.1.4 Ο βασικός κύκλος των Εξελικτικών Αλγορίθμων

Οι Εξελικτικοί Αλγόριθμοι λειτουργούν ως εξής:

1. Αρχικά, δημιουργείται ένας πληθυσμός (population) από άτομα (individuals). Κάθε άτομο λαμβάνει μια τιμή ικανότητας (fitness value) και αποκωδικοποιείται κατάλληλα για να παράγει μια υποψήφια λύση.

2. Η κάθε υποψήφια λύση αξιολογείται, υπολογίζοντας την αξία της μέσω μιας συνάρτησης ικανότητας (fitness function). Αυτή η αξιολόγηση μπορεί να περιλαμβάνει περίπλοκες προσομοιώσεις και υπολογισμούς.

3. Με βάση αυτήν την αξιολόγηση, τα άτομα του πληθυσμού με μεγαλύτερη ικανότητα επιλέγονται να γονιμοποιήσουν την επόμενη γενιά (γονείς). Έτσι, η διαδικασία της επιλογής γονέων απορρίπτει τις υποψήφιες λύσεις με χαμηλότερη ικανότητα και επιτρέπει στις υπόλοιπες να εισέλθουν στην «πισίνα ζευγαρώματος» (mating pool) με μεγαλύτερη πιθανότητα.

4. Στην φάση της αναπαραγωγής δημιουργούνται απόγονοι (offspring), μεταλλάσσοντας ή/και διασταυρώνοντας τους γονότυπους των επιλεγμένων ατόμων. Ο ανασυνδυασμός (recombination) ή αλλιώς διασταύρωση (crossover) είναι ένας τελεστής που εφαρμόζεται σε δύο ή περισσότερα επιλεγμένα άτομα (γονείς) και δημιουργεί μία ή περισσότερες υποψήφιες λύσεις (παιδιά). Από την άλλη πλευρά, ο

τελεστής της μετάλλαξης (mutation) εφαρμόζεται σε ένα άτομο και δημιουργεί μία υποψήφια λύση.

5. Οι απόγονοι εν συνεχεία ενσωματώνονται στον πληθυσμό. Συνεπώς, δημιουργείται ένας νέος πληθυσμός ατόμων που με τη σειρά τους ανταγωνίζονται για μια θέση στην επόμενη γενιά. Τα άτομα με μεγαλύτερη ικανότητα έχουν περισσότερες πιθανότητες να επιβιώσουν(επιβίωση του ικανότερου).

6. Ο εξελικτικός αλγόριθμος σταματάει όταν φτάσουμε σε κάποια συνθήκη τερματισμού (υπολογιστικό όριο που έχει τεθεί, κατάλληλη ικανότητα επιτευχθεί, κτλ), αλλιώς ακολουθεί τον επαναλαμβανόμενο κύκλο που ξεκινάει από το 2ο βήμα. Το μέγεθος του πληθυσμού διατηρείται σταθερό σε κάθε επανάληψη (Σχήμα 3.1)[42].

Αλγόριθμος: Εξελικτικός Αλγόριθμος

Αρχικοποίησε ένα πληθυσμό υποψήφιων λύσεων με τυχαίο τρόπο

Επανάλαβε

    Υπολόγισε την καταλληλότητα όλων των ατόμων του πληθυσμού

    Επίλεξε τα καταλληλότερα άτομα ως γονείς

    Δημιούργησε τους απογόνους από την αναπαραγωγή των γονέων

μέχρις ότου ικανοποιηθεί κάποια συνθήκη τερματισμού

Σχήμα 3.1 : Ψευδοκώδικας Εξελικτικού αλγόριθμου.

3.1.5 Μέθοδοι Εξελικτικών Αλγορίθμων

Υπάρχουν διάφοροι μέθοδοι εξελικτικών αλγορίθμων, με διάφορες υλοποιήσεις για την κάθε μία από αυτές, ωστόσο όλες έχουν ως κύριο χαρακτηριστικό την ύπαρξη πληθυσμού που αναπαριστούν τις πιθανές λύσεις ενός συγκεκριμένου προβλήματος βελτιστοποίησης. Οι βασικοί μέθοδοι είναι οι παρακάτω :

1.**Γενετικοί αλγόριθμοι**: Αυτό είναι το πιο δημοφιλές είδος των εξελικτικών αλγορίθμων. Επιδιώκει τη λύση ενός προβλήματος με τη μορφή αριθμών (παραδοσιακά δυαδική μορφοποίηση), εφαρμόζοντας τελεστές όπως του ανασυνδυασμού και της μετάλλαξης (μερικές φορές έναν, μερικές φορές και τους δύο). Αυτό το είδος της ΕΑ χρησιμοποιείται συχνά σε προβλήματα βελτιστοποίησης.

2.**Γενετικός προγραμματισμός:** Εδώ οι λύσεις είναι με τη μορφή προγραμμάτων ηλεκτρονικών υπολογιστών, και η καταλληλότητά τους καθορίζεται από την ικανότητά τους να λύσουν ένα υπολογιστικό πρόβλημα.

3.**Εξελικτικός προγραμματισμού:** Είναι παρόμοια με τον γενετικό προγραμματισμό, αλλά η δομή του προγράμματος είναι σταθερή και οι αριθμητικές παράμετροι της επιτρέπονται να εξελιχθούν.

4.**Γονιδιακή έκφραση προγραμματισμού**: Είναι όπως ο γενετικός προγραμματισμός, αλλά διερευνά ένα σύστημα γενότυπου-φαινότυπου, όπου τα προγράμματα υπολογιστών διαφορετικών μεγεθών κωδικοποιούνται σε γραμμικά χρωμοσώματα σταθερού μήκους.

5.**Εξελικτικές στρατηγικές**: Λειτουργεί με πραγματικούς αριθμούς ως αναπαραστάσεις των λύσεων, και συνήθως χρησιμοποιεί αυτο-προσαρμοζόμενα ποσοστά μετάλλαξης.

6.**Διαφορική εξέλιξη :**Είναι κατάλληλη για προβλήματα αριθμητικής βελτιστοποίησης.

7.**Neuroevolution:** Παρόμοια με τον γενετικό προγραμματισμό, αλλά τα γονιδιώματα αντιπροσωπεύουν τεχνητά νευρωνικά δίκτυα, περιγράφοντας τη δομή και τα βάρη σύνδεσης . Το γονιδίωμα που κωδικοποιεί μπορεί να είναι άμεσο ή έμμεσο[37].

8.**Μαθαίνοντας σύστημα ταξινομητή:** Είναι ένα παράδειγμα των μεθόδων μηχανικής μάθησης με βάση κανόνες που συνδυάζουν ένα συστατικό ανακάλυψης (π.χ. τυπικά ένα γενετικό αλγόριθμο) με ένα συστατικό μάθησης (εκτέλεση είτε εποπτευόμενης εκμάθησης, μάθησης οπλισμού, ή χωρίς επίβλεψη μάθηση). Επιδιώκουν να προσδιορίσουν ένα σύνολο μεταβλητών κανόνων που συλλογικά αποθηκεύουν και εφαρμόζουν τις γνώσεις τμηματικά, προκειμένου να γίνουν προβλέψεις (π.χ. μοντελοποίηση της συμπεριφοράς, κατάταξης, εξόρυξης δεδομένων, παλινδρόμησης, προσέγγιση συνάρτησης, ή στρατηγική παιχνίδι). Αυτή η προσέγγιση επιτρέπει

περίπλοκα τμήματα λύσεων που πρόκειται να διασπαστούν σε μικρότερα, απλούστερα μέρη[39].

3.2 Γενετικοί Αλγόριθμοι (ΓΑ)

Οι Γενετικοί Αλγόριθμοι (ΓΑ) είναι στοχαστικοί αλγόριθμοι (αναζητούν το βέλτιστο σημείο μέσω τυχαίων μεταβάσεων) των οποίων η δημιουργία εμπνεύστηκε από τους μηχανισμούς επιλογής και αναπαραγωγής που υπάρχουν στην φύση και μελετώνται στην γενετική επιστήμη. Γι' αυτόν τον λόγο και κρατούνται αυτούσιοι κάποιοι όροι των επιστημών της βιολογίας και της γενετικής, κατά την περιγραφή της μεθοδολογίας και των χαρακτηριστικών της.

3.2.1 Εισαγωγικά Στοιχεία για τους Γενετικούς Αλγόριθμους (ΓΑ)

Η πρώτη εμφάνιση των γενετικών αλγορίθμων χρονολογείται στις αρχές του 1950, όταν διάφοροι βιολόγοι επιστήμονες αποφάσισαν να χρησιμοποιήσουν υπολογιστές στην προσπάθειά τους να προσομοιώσουν πολύπλοκα βιολογικά συστήματα. Η συστηματική τους ανάπτυξη, όμως, που οδήγησε στη μορφή με την οποία είναι γνωστοί και σήμερα, πραγματοποιήθηκε στις αρχές του 1970 από τον John Holland και τους συνεργάτες του στο Πανεπιστήμιο του Michigan [53].

Ο μηχανισμός της φυσικής επιλογής φάνηκε ιδιαίτερα ελκυστικός στον John Holland, στις αρχές της δεκαετίας του '70. Ο Holland επινόησε ότι κάποιες ιδέες και λειτουργίες που εφαρμόζει η φύση στα συστήματά της θα μπορούσαν να έχουν αποτελέσματα αν ενσωματώνονταν σε αλγόριθμους για υπολογιστές, ώστε να προκύψουν αποδοτικές τεχνικές επίλυσης δύσκολων προβλημάτων. Στο βιβλίο «Adaptation in Natural and Artificial Systems»[18], που δημοσίευσε το 1975, παρουσίασε το γενετικό αλγόριθμο ως μια αφαίρεση της βιολογικής εξέλιξης και έδωσε ένα θεωρητικό πλαίσιο προσαρμογής με γενετικούς αλγορίθμους. Οι γενετικοί αλγόριθμοι, όπου πρωτοπόρος τους είναι ο John Holland, αποτελούν σχετικά μια καινούργια εξελισσόμενη και πολλά υποσχόμενη τεχνική αναζήτησης και βελτιστοποίησης [53].

Η βασική ιδέα που κρύβεται πίσω από τους γενετικούς αλγόριθμους (ΓΑ) είναι η μίμηση των μηχανισμών της φύσης. Προσπαθούν να επιλύσουν ένα πρόβλημα όχι με

μαθηματικό αλλά με βιολογικό τρόπο, γεγονός που τους δίνει μεγάλη ενδογενή ευελιξία. Φτάνουν στη βέλτιστη (ή σχεδόν στη βέλτιστη) λύση ανεξάρτητα από την πολυπλοκότητα του προβλήματος.

Η μίμηση της φυσικής εξελικτικής διαδικασίας στους γενετικούς αλγόριθμους πραγματοποιείται με την εφαρμογή τριών βασικών τελεστών: του τελεστή επιλογής, του τελεστή διασταύρωσης και του τελεστή μετάλλαξης, οι οποίοι ήδη αναφέρθηκαν ανωτέρω και θα επεξηγηθούν περαιτέρω στη συνέχεια κατά την περιγραφή της λειτουργίας των γενετικών αλγορίθμων.

3.2.2 Κύρια Χαρακτηριστικά Γενετικών Αλγορίθμων

Οι ΓΑ πλεονεκτούν αισθητά στη λύση προβλημάτων αναζήτησης και βελτιστοποίησης από τις παραδοσιακές μεθόδους. Αυτό συμβαίνει, διότι διαφέρουν θεμελιωδώς από αυτές. Τα κυριότερα νέα χαρακτηριστικά που τους διαφοροποιούν, αλλά και τους δίνουν υπεροχή είναι, σύμφωνα με τον D. Goldberg , είναι τα εξής:

1. Οι ΓΑ δουλεύουν με μια κωδικοποίηση ενός συνόλου τιμών που μπορούν να λάβουν οι μεταβλητές και όχι με τις ίδιες τις μεταβλητές του προβλήματος: Στους ΓΑ οι μεταβλητές κωδικοποιούνται με μεταβλητές πεπερασμένου μήκους κάνοντας χρήση ενός πεπερασμένου αλφαβήτου. Χαρακτηριστικό παράδειγμα είναι η δυαδική κωδικοποίηση των μεταβλητών.

2. Οι ΓΑ κάνουν αναζήτηση σε πολλά σημεία ταυτόχρονα και όχι μόνο σε ένα: Σε πολλές μεθόδους βελτιστοποίησης, η επεξεργασία γίνεται βήμα προς βήμα, πηγαίνοντας προσεκτικά από σημείο σε σημείο του πεδίου ορισμού του προβλήματος. Αυτή η βήμα προς βήμα προσέγγιση ενέχει αρκετούς κινδύνους, ο κυριότερος από τους οποίους είναι να περιοριστεί η αναζήτηση σε μια περιοχή τοπικού – και όχι συνολικού - ακρότατου, Οι ΓΑ εξαλείφουν αυτόν τον κίνδυνο ενεργώντας ταυτόχρονα πάνω σε ένα ευρύ σύνολο σημείων (σύνολο από συμβολοσειρές).

3. Οι ΓΑ χρησιμοποιούν μόνο την αντικειμενική συνάρτηση και καμία επιπρόσθετη πληροφορία: Πολλές μέθοδοι αναζήτησης απαιτούν αρκετές βοηθητικές πληροφορίες για τη συνάρτηση που επεξεργάζονται. Τέτοιου είδους πληροφορίες δεν προαπαιτούνται από τους ΓΑ. Το ψάξιμό τους είναι κατά κάποιο τρόπο «τυφλό», με την

έννοια ότι αξιοποιούν μόνο όση πληροφορία περιέχεται στην αντικειμενική συνάρτηση. Αυτό τους κάνει να είναι και πιο ευέλικτοι.

4. Οι ΓΑ χρησιμοποιούν πιθανοθεωρητικούς κανόνες αναζήτησης και όχι ντετερμινιστικούς: Η χρήση πιθανοθεωρητικών κανόνων αναζήτησης είναι κυρίαρχο γνώρισμα των ΓΑ . Το στοιχείο της τύχης σε ένα ΓΑ ,που εφαρμόζεται μέσω των γενετικών τελεστών, χρησιμοποιείται ως οδηγός για αναζήτηση σε περιοχές που αναμένεται να δώσουν καλά αποτελέσματα [53].

3.2.3 Είδη Γενετικών Αλγορίθμων (ΓΑ)

Generational GA

Σε ένα παραδοσιακό γενετικό αλγόριθμο η εξέλιξη επιτυγχάνεται μέσα από μία αλληλουχία διακεκριμένων γενιών που δεν αλληλεπικαλύπτονται. Ένα άτομο υπάρχει μόνο σε μία γενιά και μπορεί να επηρεάσει τις επόμενες μόνο διαμέσου των παιδιών του. Συνεπώς σε κάθε αναπαραγωγικό κύκλο ο αλγόριθμος παράγει μία εντελώς νέα γενιά παιδιών η οποία και αντικαθιστά εξ ολοκλήρου την προηγούμενη.

Steady-state GA

Ένα δεύτερο είδος είναι ο Steady-state γενετικός αλγόριθμος στον οποίο μόνο ένα μέρος της τρέχουσας γενιάς αντικαθίσταται από παιδιά του αναπαραγωγικού κύκλου. Το γεγονός αυτό έχει ως αποτέλεσμα τη δημιουργία μιας γενιάς η οποία προσωρινά έχει μεγαλύτερο μέγεθος οπότε επιλέγονται και απομακρύνονται άτομα από τον προσωρινό αυτό πληθυσμό μέχρις ότου να μειωθεί στα κανονικά επίπεδα. Αυτό βεβαιώνει ότι η γενιά παραμένει σε ένα σταθερό μέγεθος. Το ποσοστό του πληθυσμού που θα αντικατασταθεί καθορίζεται από το χρήστη. Οι Steady-state γενετικοί αλγόριθμοι είναι πολύ δημοφιλείς καθώς πετυχαίνουν γρηγορότερη σύγκλιση σε πολλές εφαρμογές[45].

3.2.4 Βασικά Στοιχεία των Γενετικών Αλγορίθμων (ΓΑ)

Ένας τυπικός ΓΑ περιλαμβάνει απλές λειτουργίες, που κρύβουν όμως μέσα τους μεγάλη ισχύ. Αυτός ο συνδυασμός απλοϊκότητας και ισχύος είναι το μεγαλύτερο θέλγητρο της

τεχνικής τους. Αρχικά σε έναν ΓΑ πρέπει να υπάρχουν στοιχεία που θα τον συνδέουν με το πρόβλημα που επιλύει. Η κωδικοποίηση και η αντικειμενική συνάρτηση επιτελούν αυτό το σκοπό και είναι απαραίτητα συστατικά για έναν ΓΑ.

Η Κωδικοποίηση

Η κωδικοποίηση αφορά ένα σύνολο πιθανών λύσεων ενός προβλήματος. Η αναπαράσταση των λύσεων πρέπει να γίνει με ένα μαθηματικό τρόπο, ώστε να είναι δυνατή η επεξεργασία από τον υπολογιστή. Κύριος στόχος της κωδικοποίησης είναι να αναπαριστά με ικανοποιητικό τρόπο τα επιμέρους χαρακτηριστικά των λύσεων, ώστε να διευκολύνει τις επόμενες λειτουργίες του αλγορίθμου (κυρίως την επιλογή). Αποτέλεσμα της κωδικοποίησης πρέπει να είναι η ύπαρξη ομοιοτήτων ανάμεσα στα άτομα με σκοπό την κατάλληλη αξιοποίησή τους, διότι οι ομοιότητες βοηθούν την κατεύθυνση του ψαξίματος. Έχουν αναφερθεί ποικίλες μορφές κωδικοποιήσεων, που καθεμία εξαρτάται από το υπό εξέταση πρόβλημα. Η καταλληλότητα της κωδικοποίησης εξαρτάται σε μεγάλο βαθμό από τη διαίσθηση και την πείρα του σχεδιαστή. Η πιο απλή μορφή κωδικοποίησης που προτείνεται σε προβλήματα αριθμητικής βελτιστοποίησης , είναι η κωδικοποίηση με δυαδικά ψηφία (bits). Δηλαδή, κάθε λύση αναπαρίσταται από μια δυαδική συμβολοσειρά (binary string) καθορισμένου μήκους. Αντίθετα σε προβλήματα συνδυαστικής βελτιστοποίησης προτιμάται η κωδικοποίηση ακεραίων.

Η Αντικειμενική Συνάρτηση

Το δεύτερο βασικό στοιχείο σύνδεσης ενός ΓΑ με το πρόβλημα που λύνει, είναι η αντικειμενική συνάρτηση (fitness function) ή συνάρτηση καταλληλότητας. Παίρνει ως είσοδο μια αποκωδικοποιημένη συμβολοσειρά και επιστρέφει μια τιμή (συνήθως πραγματική), που είναι ανάλογη του πόσο καλά λύνει το πρόβλημα η συγκεκριμένη συμβολοσειρά. Η τιμή αυτή αποτελεί και τον καθοριστικό παράγοντα επιβίωσης και πολλαπλασιασμού ή όχι του ατόμου. Ουσιαστικά, αποτελεί τη μόνη πληροφορία που δέχεται ο αλγόριθμος για το πρόβλημα που πρέπει να λύσει. Δηλαδή σε κάθε πιθανή τιμή της μεταβλητής , αντιστοιχεί μια τιμή ικανότητας ή απόδοσης (fitness ή score). Αυτή η τιμή αξιολογεί το πόσο καλή είναι η λύση και που, για τη μεγιστοποίηση της συνάρτησης [53].

Συνθήκη τερματισμού

Οι Γενετικοί αλγόριθμοι μπορούν να συνεχίσουν να εξελίσσουν τις νέες υποψήφιες λύσεις για όσο καιρό είναι απαραίτητο. Ανάλογα με τη φύση του προβλήματος, ένας γενετικός αλγόριθμος μπορεί να τρέξει οπουδήποτε από μερικά δευτερόλεπτα μέχρι πολλά χρόνια! Καλούμε την κατάσταση στην οποία ένας γενετικός αλγόριθμος τελειώνει την αναζήτηση ως κατάστασης λήξη του. Μερικές τυπικές προϋποθέσεις τερματισμού της αναζήτησης στον γενετικό αλγόριθμο είναι οι εξής:

α) Έχει επιτευχθεί ο μέγιστος αριθμός των γενεών.

β) Η λύση έχει βρεθεί και πληροί τα απαιτούμενα κριτήρια.

γ) Για μεγάλο πλήθος γενεών δεν αλλάζει περεταίρω η εξέλιξη του . Δηλαδή ο αλγόριθμος έχει φτάσει σε σημείο στασιμότητας.

δ) Βάζουμε ανώτατο χρονικό περιθώριο για να βρεθεί η βέλτιστη λύση[19].

3.2.5 Βασικοί Τελεστές Γενετικών Αλγορίθμων (ΓΑ)

Όπως έχει προαναφερθεί και παραπάνω υπάρχουν τρείς βασικοί τελεστές στους γενετικούς αλγόριθμους. Ο πρώτος είναι ο τελεστής της επιλογής, ο δεύτερος της διασταύρωσης και ο τρίτος της μετάλλαξης.

Τελεστής Επιλογής

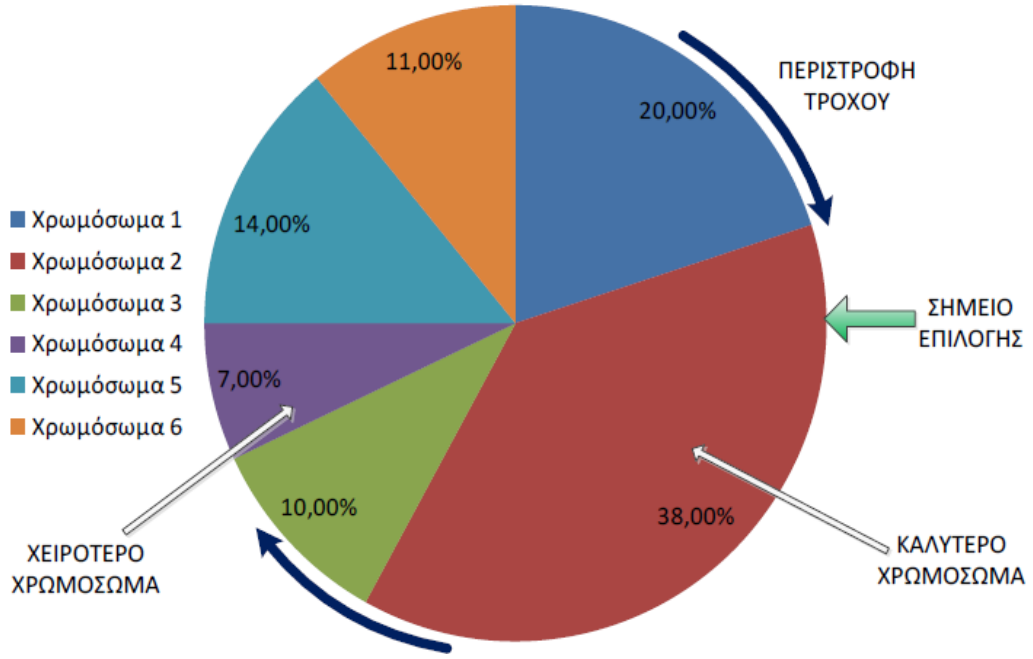
Με τον τελεστή της επιλογής, βρίσκει εφαρμογή στα πλαίσια του αλγορίθμου, ο νόμος της επιβίωσης του ικανότερου. Μέσω αυτής της διαδικασίας, καθορίζεται ποια άτομα από τον υπάρχοντα πληθυσμό θα έχουν την ευκαιρία να λάβουν μέρος στην αναπαραγωγή και να κληροδοτήσουν στην επόμενη γενιά μέρος ή το σύνολο των χαρακτηριστικών τους. Στόχος της λειτουργίας της επιλογής είναι να επιτρέπει εκθετική αύξηση των ικανοτέρων ατόμων και τελικά, μετά από αναπαραγωγή αρκετών γενεών, την επικράτησή τους [53]. Στη βιβλιογραφία έχουν αναφερθεί αρκετές μέθοδοι επιλογής [03] [04] [17]και συνεχώς προστίθενται νέες.

Η πιο κλασική μέθοδος επιλογής , είναι η επιλογή τροχού ρουλέτας. Το βασικό χαρακτηριστικό της συγκεκριμένης μεθόδου είναι ότι κάθε άτομο του πληθυσμού επιλέγεται με μια πιθανότητα P_i η οποία είναι ανάλογη της καταλληλότητας του. Η πιθανότητα P_i περιγράφεται από την παρακάτω σχέση:

$$P_i = \frac{K_i}{\sum_{j=1}^{n} K_j}$$

όπου K_i είναι η καταλληλότητα του ι-οστού ατόμου.

Στην συγκεκριμένη μέθοδο επιλογής σε κάθε άτομο του πληθυσμού ανατίθεται ένα μικρό τμήμα της ρουλέτας (Σχήμα 3.2). Το μέγεθος του αντίστοιχου τμήματος είναι ανάλογο με την καταλληλότητά του, όπως έχει υπολογιστεί από την αντικειμενική συνάρτηση του προβλήματος. Ο τροχός γυρίζει και κάθε άτομο του πληθυσμού που επιλέγεται με βάση το σημείο επιλογής του τροχού αποτελεί και τον επόμενο γονέα. Η διαδικασία είναι επαναληπτική μέχρι να συμπληρωθεί ο απαιτούμενος αριθμός των γονέων που θα δημιουργήσουν την επόμενη γενιά. Με την εφαρμογή αυτής της μεθόδου επιτυγχάνεται και η διαδικασία « ο πιο ικανός επιβιώνει» [27].



Σχήμα 3.2 : Επιλογή Τροχού Ρουλέτας

Η δεύτερη πιο δημοφιλής μέθοδος επιλογής είναι αυτή της «επιλογής τουρνουά» (tournament selection). Σε αυτή την μέθοδο, Ν άτομα επιλέγονται τυχαία από τον πληθυσμό. Η επιλογή πραγματοποιείται διαλέγοντας έναν αριθμό από το ένα έως τον αριθμό των ατόμων του πληθυσμού. Ο αριθμός αυτός καθορίζει το χρωμόσωμα που θα πάρει μέρος στο τουρνουά και η διαδικασία συνεχίζει έως ότου να συμπληρώσουμε τα

Ν άτομα της ομάδας. Για κάθε μέλος της ομάδας αυτής εν συνεχεία υπολογίζεται η τιμή της συνάρτησης κόστους και το χρωμόσωμα με τη μεγαλύτερη τιμή κερδίζει το τουρνουά και γίνεται ο επίλεκτος. Τα υπόλοιπα μέλη της ομάδας επιστρέφουν στον αρχικό πληθυσμό και η διαδικασία επαναλαμβάνεται. Η πιο συνηθισμένη μορφή τουρνουά είναι αυτή όπου το Ν είναι ίσο με δύο δηλαδή η κάθε ομάδα αποτελείται από δύο χρωμοσώματα. Η ταξινόμηση της ομάδας με βάση τη τιμή της συνάρτησης κόστους, μας βοηθάει να βρούμε το άτομο με την υψηλότερη τιμή και να διαλέξουμε τον γονέα [41]. Αυτού του είδους η επιλογή που παρουσιάστηκε παραπάνω είναι η ντετερμινιστική προσέγγιση και είναι γνωστή στη διεθνή βιβλιογραφία με τον όρο «marriage tournament». Υπάρχει επίσης και η στοχαστική προσέγγιση και η μέθοδος που το υποδηλώνει είναι η μέθοδος τουρνουά Boltzmann. Σε αυτή την προσέγγιση τα k επιλεγμένα χρωμοσώματα κάθε ομάδας τοποθετούνται σε k/2 ζεύγη και τα μέλη κάθε ζεύγους αναμετρώνται με πιθανότητα να κερδίσει το ικανότερο ίσο με p_i [55].

Τελεστής Διασταύρωσης

Ο προσωρινός πληθυσμός που προέκυψε από τη διαδικασία της επιλογής πρέπει να περάσει από τη διαδικασία ζευγαρώματος για να πραγματοποιηθεί ένα είδος γονιμοποίησης, όπως συμβαίνει και στη φύση. Η νέα, λοιπόν, ομάδα ατόμων που προέκυψε από την επιλογή σχηματίζει με τυχαίο τρόπο ομάδες των δύο. Σε κάθε ομάδα, τα δύο μέλη παίρνουν μέρος σε μια απλή λειτουργία ανταλλαγής γενετικού υλικού που ονομάζεται διασταύρωση. Η διασταύρωση είναι μια απαραίτητη λειτουργία που συμβάλει αποφασιστικά στην επίδοση ενός ΓΑ. Εξ' αιτίας αυτής της σπουδαιότητας, έχει γίνει αρκετή έρευνα και έχουν επινοηθεί πολλοί τρόποι υλοποίησής του. Στην πράξη η διασταύρωση υλοποιείται στο ΓΑ με μία πιθανότητα, την λεγόμενη πιθανότητα διασταύρωσης (crossover probability) P_c, που καθορίζεται από το σχεδιαστή του ΓΑ. Συνήθως, αυτή η πιθανότητα ποικίλει από πρόβλημα σε πρόβλημα, ενώ είναι δυνατό και να αλλάζει κατά τον χρόνο εκτέλεσης [53].

Υπάρχουν αρκετές υλοποίησης για την εφαρμογή της διασταύρωσης: οι πιο διαδεδομένες είναι οι διασταυρώσεις μονού σημείου, δύο σημείων, η πολυσημειακή διασταύρωση και η ομοιόμορφη διασταύρωση.

Στην διασταύρωση μονού σημείου επιλέγεται τυχαία ένα σημείο στα χρωμοσώματα που πρόκειται να διασταυρωθούν και ανταλλάσσονται τα τμήματα των 2 χρωμοσωμάτων μετά από αυτό το σημείο (Παρ. 1).

Παράδειγμα 1

Έστω ότι έχουμε τους παρακάτω γονείς και θέλουμε να διασταυρωθούν στην θέση 6.

1	2	3	4	5	6	7	8
1	1	1	1	0	1	1	0

Πίνακας 3.1 : Γονέας 1

1	2	3	4	5	6	7	8
1	0	1	0	0	1	0	0

Πίνακας 3.2 : Γονέας 2

Μετά την διασταύρωση θα δημιουργηθούν οι παρακάτω απόγονοι :

1	2	4	4	5	6	7	8
1	1	1	1	0	1	0	0

Πίνακας 3.3 : Απόγονος 1

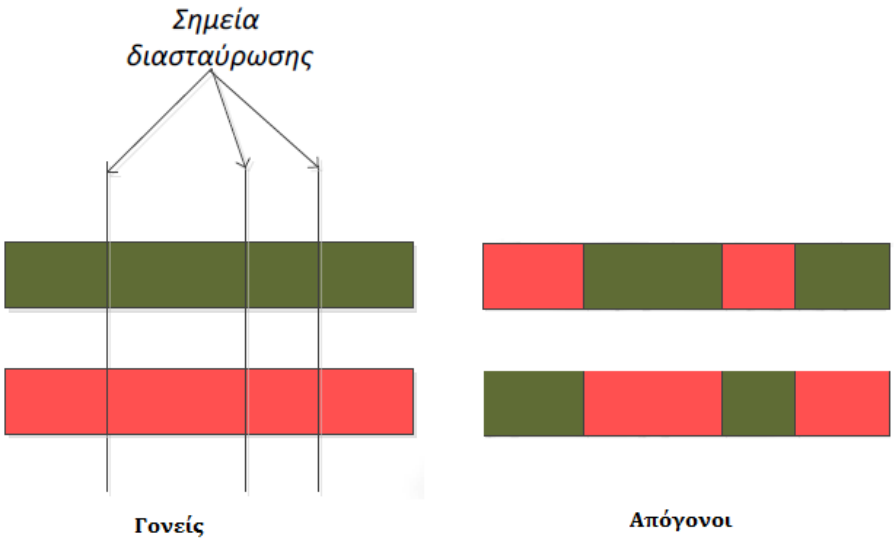
1	2	3	4	5	6	7	8
1	0	1	0	0	1	1	0

Πίνακας 3.4 : Απόγονος 2

Στην διασταύρωση δύο σημείων γίνεται η ίδια διαδικασία με την διασταύρωση ενός σημείου με την μόνη διαφορά ότι τα σημεία διασταύρωσης είναι δύο αντί για ένα. Στο «σχήμα 3.3» που ακολουθεί παρακάτω έχουμε διασταύρωση 2 σημείων για τις θέσεις 5 και 9 των γονέων.

Σχήμα 3.3 : Διασταύρωση 2 σημείων

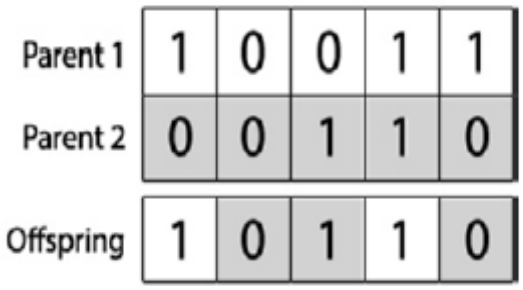
Στην πολυσημειακή διασταύρωση επιλέγονται n σημεία διασταύρωσης (με n <l όπου l το μήκος των χρωμοσωμάτων-γονέων και n, l θετικοί ακέραιοι) και ανταλλάσσονται τα τμήματα μεταξύ των διαδοχικών σημείων διασταύρωσης (Σχήμα 3.4).



Σχήμα 3.4: Πολυσημειακή Διασταύρωση

Στην ομοιόμορφη διασταύρωση χρησιμοποιείται μια σταθερή αναλογία ανταλλαγής γονιδίων μεταξύ των δύο γονέων. Με την συγκεκριμένη διασταύρωση κάθε γονίδιο αξιολογείται ως υποψήφιο για ανταλλαγή με μια πιθανότητα ίση συνήθως με 0,5.

Επίσης χρησιμοποιείται μια μάσκα διασταύρωσης που καθορίζει από ποιον γονέα θα κληρονομήσει ο απόγονος το κάθε γονίδιο (Σχήμα 3.5) [27].



Σχήμα 3.5:Ομοιόμορφη Διασταύρωση

Τελεστής Μετάλλαξης

Τελευταία στον κύκλο αναπαραγωγικής διαδικασίας και, ίσως, λιγότερο σημαντική, αλλά πάντως χρήσιμη, είναι η μετάλλαξη. Είναι μια λειτουργία που όταν συμβαίνει αραιά στη φύση δρα βελτιωτικά για τους οργανισμούς και γενικά για την εξέλιξη της ζωής. Ενεργεί σε ένα μόνο οργανισμό κάθε φορά. Καθώς αντιγράφονται δυαδικά ψηφία από τον γονέα στον απόγονο, επιλέγεται τυχαία με μικρή πιθανότητα, τη λεγόμενη πιθανότητα μετάλλαξης (mutation probability) P_m, ένα ψηφίο και αντιστρέφεται (από 0 σε 1 ή το αντίστροφο) (Παράδειγμα 2). Η μετάλλαξη λειτουργεί ως ασφαλιστική δικλείδα για τις περιπτώσεις κατά τις οποίες η επιλογή και η διασταύρωση, ενδεχομένως, χάσουν κάποιες πολύτιμες γενετικές πληροφορίες. Όταν συμβαίνει, επιφέρει ποικιλία στον πληθυσμό, ανακατευθύνει την αναζήτηση και εξασφαλίζει ότι κανένα σημείο του χώρου αναζήτησης δεν αποκλείεται από τη διαδικασία του ψαξίματος [53]. Υπάρχουν διάφορες τεχνικές μετάλλαξης, η πιο διαδεδομένη είναι η μετάλλαξη ενός bit string. Σε αυτή την τεχνική έχουμε μετάλλαξη σε μία τυχαία θέση ενός bit string [38]. Η τιμή της πιθανότητας της μετάλλαξης ενός bit είναι αντιστρόφως ανάλογη του μήκους N των χρωμοσωμάτων. Δηλαδή $P_m=1/N$ [02].

Μία εξίσου σημαντική τεχνική μετάλλαξης είναι αυτό της αναστροφής ενός bit (flit bit). Η συγκεκριμένη τεχνική επιλέγει ένα ή περισσότερα bits του χρωμοσώματος και τα αναστρέφει . Δηλαδή εάν επιλεγμένο bit του χρωμοσώματος είναι 1 το μετατρέπει σε 0 και αντίστροφα.

Ένα άλλο είδος τεχνικής μετάλλαξης είναι αυτό της ομοιόμορφης μετάλλαξης. Ο τελεστής αυτός χρησιμοποιεί ένα μόνο χρωμόσωμα-γονιό x και παράγει ένα παιδί x'. Επιλέγει τυχαία από το διάνυσμα $x = (x_1, ..., x_k, ..., x_q)$, όπου q είναι το πλήθος των μεταβλητών του διανύσματος , μία συνιστώσα $k \in (1, ..., q)$ και παράγει το διάνυσμα $x' = (x_1, ..., x'_k, ..., x_q)$ όπου το x'_k αποτελεί μια τυχαία τιμή στο διάστημα ορισμού της παραμέτρου x_k [41].

Παράδειγμα 2

Έστω ότι έχουμε τον παρακάτω γονέα και θέλουμε να γίνει μετάλλαξη στην θέση 4 .

1	2	3	4	5	6	7	8
1	0	1	0	0	1	1	0

Πίνακας 3.5 : Γονέας

Θα δημιουργηθεί ο παρακάτω απόγονος μετά την διαδικασία της μετάλλαξης:

1	2	3	4	5	6	7	8
1	0	1	1	0	1	1	0

Πίνακας 3.6 : Απόγονος

3.2.6 Μέθοδος του Βασικού Γενετικού Αλγόριθμου (ΓΑ)

Για την λειτουργία ενός βασικού γενετικού αλγόριθμου θα πρέπει πρώτα να καθοριστούν έξι βασικοί παράγοντες. Αυτοί είναι οι παρακάτω :

1. Πώς θα γίνει η αναπαράσταση της λύσης (solution representation) .

2. Ποια θα είναι η συνάρτηση επιλογής των χρωμοσωμάτων για την αναπαραγωγή (selection function) .

3. Ποιοι γενετικοί τελεστές θα χρησιμοποιηθούν για την αναπαραγωγή (genetic operators)

4. Πως θα επιλεγεί ο αρχικός πληθυσμός (initial population).

5. Ποια θα είναι τα κριτήρια τερματισμού (termination criteria)

6. Ποια θα είναι η συνάρτηση αξιολόγησης (evaluation function)

Τα βασικά βήματα ενός απλού γενετικού αλγορίθμου είναι τα παρακάτω :

1. Ξεκινάει με έναν τυχαία παραγόμενο πληθυσμό από N l-bit χρωμοσώματα – δηλ. δυαδικές ακολουθίες μεγέθους l bit - που αποτελούν τις υποψήφιες λύσεις για το εκάστοτε πρόβλημα.

2. Υπολογίζει την καταλληλότητα (fitness) κάθε χρωμοσώματος i στον πληθυσμό.

3. Επαναλαμβάνει τα παρακάτω βήματα μέχρι να δημιουργηθούν N απόγονοι :

(α) Επιλέγει ένα ζευγάρι χρωμοσωμάτων από τον τρέχοντα πληθυσμό που θα αποτελέσει τους γονείς, με πιθανότητα επιλογής που είναι ανάλογη της καταλληλότητας. Η επιλογή γίνεται με αντικατάσταση, δηλαδή το ίδιο χρωμόσωμα μπορεί να επιλεχτεί περισσότερο από μία φορά για να γίνει γονέας.

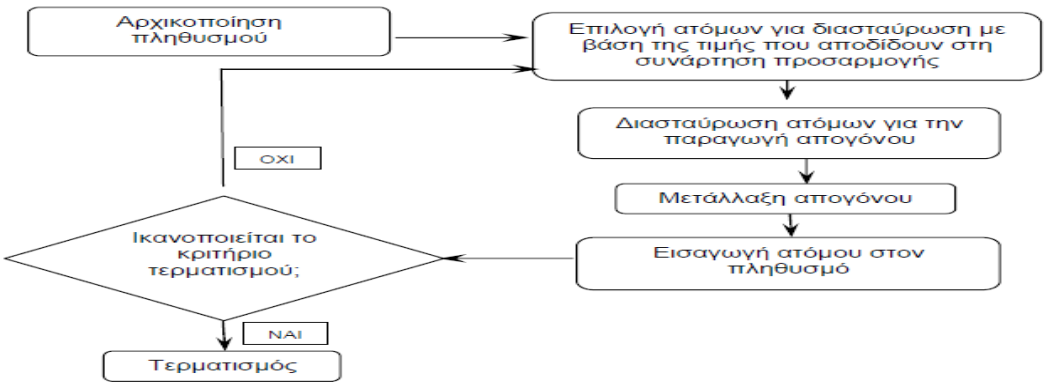
(β) Με πιθανότητα P_c (πιθανότητα διασταύρωσης), διασταυρώνει το ζευγάρι σε ένα τυχαία επιλεγμένο σημείο για να διαμορφώσει δύο νέους απογόνους. Εάν δεν πραγματοποιηθεί η διασταύρωση, οι δύο νέοι απόγονοι θα είναι τα ακριβή αντίγραφα των γονέων.

(γ) Μεταλλάσσει κάθε σημείο των δύο απογόνων με πιθανότητα P_m (πιθανότητα μετάλλαξης), και τοποθετεί τα προκύπτοντα χρωμοσώματα στο νέο πληθυσμό. Εάντο N είναι περιττός αριθμός, τότε μπορεί να απορριφθεί ένα μέλος του νέουπληθυσμού με τυχαίο τρόπο.

4. Αντικαθιστά τον τρέχοντα πληθυσμό με το νέο πληθυσμό που δημιουργήθηκε.

5. Επαναλαμβάνει τα βήματα 2 έως 5 μέχρι να ικανοποιηθεί η συνθήκη τερματισμού [19].

Στο «Σχήμα 17» που ακολουθεί παρακάτω δίνεται το διάγραμμα ροής του βασικού ΓΑ. Αντίστοιχα στο « Σχήμα 18» παρουσιάζεται ο Ψευδοκώδικας του ΓΑ .



Σχήμα 3.6 : Διάγραμμα Ροής Γενετικού Αλγορίθμου[46]

```
1: generation = 0;
2: population[generation] = initializePopulation(populationSize);
3: evaluatePopulation(population[generation]);
3: While isTerminationConditionMet() == false do
4:     parents = selectParents(population[generation]);
5:    population[generation+1] = crossover(parents);
6:   population[generation+1] = mutate(population[generation+1]);
7:    evaluatePopulation(population[generation]);
8:     generation++;
9: End loop;
```

Σχήμα 3.7 : Ψευδοκώδικας Γενετικού Αλγορίθμου

3.3 Πλεονεκτήματα και Μειονεκτήματα Χρήσης Γενετικών Αλγορίθμων

Όπως όλοι οι αλγόριθμοι έχουν και θετικά και αρνητικά στοιχεία από την χρήση τους, έτσι έχουν και οι γενετικοί αλγόριθμοι. Παρακάτω παρατίθενται τα πλεονεκτήματα και τα μειονεκτήματα της χρήσης των γενετικών αλγορίθμων.

3.3.1 Πλεονεκτήματα Χρήσης Γενετικών Αλγορίθμων

Μερικά από τα σημαντικότερα πλεονεκτήματα που έχει η χρήση εξελικτικών και γενετικών αλγορίθμων για την επίλυση προβλημάτων είναι τα εξής:

Ένα βασικό πλεονέκτημά τους είναι το γεγονός ότι έχουν τη δυνατότητα να επιλύουν δύσκολα προβλήματα γρήγορα και αξιόπιστα. Ένας από τους σημαντικούς λόγους χρήσης των αλγορίθμων αυτών είναι η μεγάλη τους αποδοτικότητα. Τόσο η θεωρία, όσο και η πράξη έχουν δείξει ότι προβλήματα που έχουν πολλές, δύσκολα προσδιορισμένες λύσεις μπορούν να αντιμετωπιστούν καλύτερα από Γ.Α. Είναι δε αξιοσημείωτο ότι συναρτήσεις που παρουσιάζουν μεγάλες διακυμάνσεις και καθιστούν ανεπαρκείς άλλες μεθόδους στην εύρεση των ακρότατών τους, για τους αλγορίθμους δεν αποτελούν σημεία δυσχέρειας.

Οι γενετικοί αλγόριθμοι μπορούν επίσης να συνεργαστούν με τα υπάρχοντα μοντέλα και συστήματα. Οι εξελικτικοί και γενετικοί αλγόριθμοι προσφέρουν το σημαντικό πλεονέκτημα της χρήσης τους με προσθετικό τρόπο στα μοντέλα που χρησιμοποιούνται σήμερα, μη απαιτώντας την επανασχεδίαση τους. Μπορούν εύκολα να συνεργαστούν με τον υπάρχοντα κώδικα, χωρίς μεγάλο κόπο. Αυτό συμβαίνει, διότι χρησιμοποιούν μόνο πληροφορίες της διαδικασίας ή συνάρτησης που πρόκειται να βελτιστοποιήσουν, δίχως να ενδιαφέρει άμεσα ο ρόλος της μέσα στο σύστημα ή η όλη δομή του συστήματος.

Ένα επίσης σημαντικό πλεονέκτημα των γενετικών αλγορίθμων, είναι το γεγονός πως είναι εύκολα επεκτάσιμοι και εξελίξιμοι. Πιο συγκεκριμένα, οι αλγόριθμοι αυτοί δεν αντιστέκονται σε αλλαγές, επεκτάσεις και μετεξελίξεις, ανάλογα με την κρίση του σχεδιαστή. Σε πολλές εφαρμογές, έχουν αναφερθεί λειτουργίες των αλγορίθμων που δεν είναι δανεισμένες από τη φύση ή που έχουν υποστεί σημαντικές αλλαγές, πάντα

προς όφελος της απόδοσης. Παραλλαγές στο βασικό σχήμα δεν είναι απλά αναγκαίες, αλλά σε ορισμένες περιπτώσεις επιβάλλονται.

Επιπροσθέτως, οι γενετικοί αλγόριθμοι έχουν την ιδιότητα ότι μπορούν να συμμετέχουν σε υβριδικές μορφές με άλλες μεθόδους. Αν και η ισχύς των αλγορίθμων αυτών είναι μεγάλη, σε μερικές ειδικές περιπτώσεις προβλημάτων, όπου άλλες μέθοδοι συμβαίνει να έχουν πολύ υψηλή αποδοτικότητα, λόγω εξειδίκευσης, υπάρχει η δυνατότητα χρησιμοποίησης ενός υβριδικού σχήματος αλγορίθμων με άλλη μέθοδο. Αυτό είναι αποτέλεσμα της μεγάλης ευελιξίας των γενετικών αλγορίθμων.

Σημαντικό επίσης πλεονέκτημα των αλγορίθμων αυτών είναι ότι εφαρμόζονται σε πολύ περισσότερα πεδία από κάθε άλλη μέθοδο. Το χαρακτηριστικό που τους εξασφαλίζει αυτό το πλεονέκτημα είναι η ελευθερία επιλογής των κριτηρίων που καθορίζουν την επιλογή μέσα στο τεχνικό περιβάλλον. Έτσι, οι αλγόριθμοι αυτοί μπορούν να χρησιμοποιηθούν στην οικονομία, στο σχεδιασμό μηχανών, στην επίλυση μαθηματικών εξισώσεων, στην εκπαίδευση Νευρωνικών Δικτύων και σε πολλούς άλλους τομείς. Αλλά πέρα από αυτό, είναι εξίσου σημαντικό το ότι δεν απαιτούν περιορισμούς στις συναρτήσεις που επεξεργάζονται. Ο κύριος λόγος που καθιστά τις παραδοσιακές μεθόδους δύσκαμπτες και ακατάλληλες για πολλά προβλήματα είναι η απαίτησή τους για ύπαρξη περιορισμών, όπως ύπαρξη παραγώγων, συνέχεια, όχι "θορυβώδεις" συναρτήσεις κ.τ.λ. Τέτοιου είδους ιδιότητες είναι αδιάφορες για τους αλγορίθμους αυτούς, πράγμα που τους κάνει κατάλληλους για μεγάλο φάσμα προβλημάτων.

Σημαντική και εξίσου χρήσιμη ιδιότητα των αλγορίθμων, είναι το γεγονός ότι δεν ενδιαφέρει η σημασία της υπό εξέταση πληροφορίας όταν χρησιμοποιούνται. Η μόνη "επικοινωνία" του γενετικού αλγορίθμου με το περιβάλλον του είναι η αντικειμενική συνάρτηση. Αυτό εγγυάται την επιτυχία του ανεξάρτητα από την σημασία του προβλήματος. Βέβαια, δεν σημαίνει ότι δεν υπάρχουν άλυτα προβλήματα για τους αλγορίθμους. Όπου όμως δεν τα καταφέρνουν, η αιτία είναι η φύση του χώρου που ερευνούν και όχι το πληροφοριακό περιεχόμενο του προβλήματος.

Οι γενετικοί αλγόριθμοι έχουν από τη φύση τους το στοιχείο του παραλληλισμού. Οι αλγόριθμοι σε κάθε τους βήμα επεξεργάζονται μεγάλες ποσότητες πληροφορίας, αφού κάθε άτομο θεωρείται αντιπρόσωπος πολλών άλλων, και μάλιστα έχει αποδειχτεί πως μπορούν να καλύψουν με αποδοτικό ψάξιμο μεγάλους χώρους σε μικρούς χρόνους. Παράλληλα, αποτελούν ουσιαστικά μια μέθοδο που κάνει ταυτόχρονα εξερεύνηση του

χώρου αναζήτησης και εκμετάλλευση της ήδη επεξεργασμένης πληροφορίας. Ο συνδυασμός αυτός σπάνια συναντάται σε οποιαδήποτε άλλη μέθοδο. Με το τυχαίο ψάξιμο γίνεται καλή εξερεύνηση του χώρου, αλλά δεν γίνεται εκμετάλλευση της πληροφορίας. Αντίθετα, με τον αλγόριθμο hill climbing (Είναι ένας επαναληπτικός αλγόριθμος μαθηματικής βελτιστοποίησης που ανήκει στην κατηγορία της τοπικής αναζήτησης. Ξεκινάει με μια αυθαίρετη λύση σε ένα πρόβλημα, στη συνέχεια προσπαθεί να βρει μια καλύτερη λύση αλλάζοντας σταδιακά ένα μόνο στοιχείο της λύσης.) γίνεται καλή εκμετάλλευση της πληροφορίας, αλλά όχι καλή εξερεύνηση. Συνήθως τα δύο αυτά χαρακτηριστικά είναι ανταγωνιστικά και το επιθυμητό είναι να συνυπάρχουν και τα δύο προς όφελος της όλης διαδικασίας. Οι γενετικοί αλγόριθμοι επιτυγχάνουν το βέλτιστο συνδυασμό εξερεύνησης και εκμετάλλευσης, πράγμα που τους κάνει ιδιαίτερα αποδοτικούς και ελκυστικούς.

Τέλος, οι γενετικοί αλγόριθμοι επιδέχονται παράλληλη υλοποίηση, και αυτό είναι ένα εξίσου σημαντικό τους πλεονέκτημα. Οι εξελικτικοί αλγόριθμοι μπορούν να εκμεταλλευτούν τα πλεονεκτήματα των παράλληλων μηχανών, αφού λόγω της φύσης τους, εύκολα μπορούν να δεχτούν παράλληλη υλοποίηση. Το χαρακτηριστικό αυτό αυξάνει ακόμη περισσότερο την απόδοσή τους, ενώ σπάνια συναντάται σε ανταγωνιστικές μεθόδους.

3.3.2 Μειονεκτήματα Χρήσης Γενετικών Αλγορίθμων

Παρά την πληθώρα πλεονεκτημάτων των γενετικών αλγορίθμων, υπάρχουν και κάποιες παράμετροι που κάνουν τους χρήστες τους επιφυλακτικούς. Οι βασικότεροι λόγοι για την δυσπιστία αυτή συνοψίζονται σε αυτήν την ενότητα.

Η χρήση των αλγορίθμων αυτών παρουσιάζει κάποια σημαντικά προβλήματα εξοικείωσης με τη Γενετική. Για τους περισσότερους, που ασχολούνται με την Επιστήμη των Υπολογιστών, οι έννοιες της Εξέλιξης και της Φυσικής Επιλογής μπορεί να μην ηχούν παράξενα, αλλά δεν είναι και από τις πιο οικείες. Η Βιολογία δεν έχει άμεση σχέση με τους υπολογιστές, γι' αυτό και οι γνώσεις σχεδόν όλων είναι σε πολύ γενικό επίπεδο. Εντούτοις, δεν απαιτούνται γνώσεις Γενετικής και Βιολογίας. Εκείνο που συμβαίνει με τους αλγορίθμους αυτούς είναι ότι μιμούνται με αφαιρετικό τρόπο κάποιες διαδικασίες που παρατηρούνται στη φύση, χωρίς να ενδιαφέρει σε μεγάλο βαθμό λεπτομέρειας η λειτουργία τους και χωρίς να είναι απαραίτητο το γνωστικό υπόβαθρο που έχουν οι

βιολόγοι για να μελετήσουν αυτά τα φαινόμενα. Οι όροι είναι δανεισμένοι από τη βιολογία με σκοπό την καλύτερη εισαγωγή και κατανόηση του θέματος και όχι την παραπομπή του μελετητή στα άγνωστα πεδία μιας ξένης επιστήμης και, τελικά, τη σύγχυσή του. Θα μπορούσε ίσως, να παραληφθεί η αναφορά στη Γενετική και να γίνει μια παρουσίαση των αλγορίθμων ως "προσωπικές διαδικασίες για αναζήτηση και βελτιστοποίηση", αυτό όμως θα έκανε τα πράγματα δυσκολότερα. Εξάλλου, είναι συνηθισμένο το φαινόμενο θεωρίες που είναι δανεισμένες από άλλες επιστήμες να διατηρούν την αυθεντική τους ορολογία.

Ένα ακόμα σημαντικό πρόβλημα, είναι αυτό του χρόνου. Στη φύση όπως είναι γνωστό, η εξέλιξη λειτουργεί με ρυθμούς πολύ αργούς. Χρειάζονται να περάσουν χιλιάδες γενιές, άρα και αρκετός χρόνος, για να αλλάξουν τα χαρακτηριστικά των ειδών και να διαφοροποιηθούν οι ικανότητες και η συμπεριφορά τους. Θέτουν έτσι ορισμένοι το ερώτημα: πώς είναι δυνατό ένα μοντέλο αναζήτησης λύσεων να έχει καλές επιδόσεις χρόνου, όταν είναι εμπνευσμένο από μια φυσική διαδικασία που εξελίσσεται με ρυθμούς απίστευτα αργούς; Η απάντηση εδώ είναι απλή. Κατ' αρχήν, ακόμη και στη φύση, η εξέλιξη δεν είναι από μόνη της μια αργή διαδικασία. Εξέλιξη των ειδών συμβαίνει όταν αλλάζει τα περιβάλλον τους και πρέπει να προσαρμοστούν στα καινούργια δεδομένα, ώστε να επιβιώσουν. Αλλαγές όμως του περιβάλλοντος γίνονται με πολύ αργούς ρυθμούς και κατά συνέπεια και η εξέλιξη ακολουθεί αυτούς τους ρυθμούς. Αν οι αλλαγές του περιβάλλοντος γίνονται με γρηγορότερο τρόπο, τότε επιταχύνεται και η εξέλιξη. Αυτό άλλωστε παρατηρείται και στα βιολογικά εργαστήρια, όπου μικροοργανισμοί αλλάζουν την συμπεριφορά τους αμέσως, όταν τοποθετούνται σε νέες συνθήκες. Επιπλέον, στο πεδίο των υπολογιστών τα άτομα κωδικοποιούνται συνήθως ως συμβολοσειρές και οι συνθήκες του περιβάλλοντος μοντελοποιούνται με απλές μαθηματικές σχέσεις. Έτσι, το μοντέλο με το οποίο δουλεύει ο υπολογιστής δεν παρουσιάζει ιδιαίτερο υπολογιστικό φόρτο, συγκρινόμενο πάντα με αντίστοιχες μεθόδους[50].

3.4 Εφαρμογές Γενετικών Αλγορίθμων

Παρακάτω παρουσιάζονται μερικές αντιπροσωπευτικές εφαρμογές των γενετικών αλγορίθμων.

I. Εύρεση μέγιστης τιμής αριθμητικών συναρτήσεων

Πρόκειται για την πιο καλά μελετημένη εφαρμογή των γενετικών αλγορίθμων. Η εύρεση του μέγιστου μιας συνάρτησης δεν είναι καθόλου εύκολη υπόθεση για συναρτήσεις πολλών μεταβλητών, οι οποίες εμφανίζουν ασυνέχειες, θόρυβο, κλπ. Το πλεονέκτημα που εμφανίζει η εφαρμογή τους σε αυτά τα προβλήματα είναι ότι η συνάρτηση καταλληλότητας είναι δεδομένη.

II. Επεξεργασία εικόνων

Οι γενετικοί αλγόριθμοι χρησιμοποιούνται για την αναγνώριση προτύπων, όπως ακμές, επιφάνειες, ακόμη και αντικείμενα, σε ψηφιοποιημένες εικόνες. Το αποτέλεσμα αυτής της επεξεργασίας μπορεί να αποτελέσει τη βάση για την ψηφιακή όραση.

III. Σχεδίαση

Οι γενετικοί αλγόριθμοι μπορούν να χρησιμοποιηθούν στη σχεδίαση κατασκευών και εξαρτημάτων, όπως π.χ. γέφυρες, μηχανολογικά εξαρτήματα, όπου ζητούμενο μπορεί να είναι τόσο η εύρεση μιας λύσης, όσο και η βελτιστοποίηση της. Οι αλγόριθμοι μπορούν να δοκιμάσουν συνδυασμούς και ιδέες που ο ανθρώπινος νους δε θα δοκίμαζε ποτέ, δίνοντας ενίοτε πρωτότυπα αποτελέσματα.

IV. Μηχανική μάθηση

Στα συστήματα μηχανικής μάθησης οι γενετικοί αλγόριθμοι μπορούν να χρησιμοποιηθούν για την ανακάλυψη κανόνων if…then... Η πιο γνωστή εφαρμογή είναι αυτή των συστημάτων κατηγοριοποίησης (classified systems), ωστόσο οι γενετικοί αλγόριθμοι έχουν χρησιμοποιηθεί και σε παιχνίδια, επίλυση λαβυρίνθων, καθώς και για πολιτικές και οικονομικές αναλύσεις.

V. Συνδυαστική βελτιστοποίηση

Πρόκειται για το κλασσικό πρόβλημα κατανομής πόρων σε δραστηριότητες, με σκοπό τη μεγιστοποίηση του οφέλους ή την ελάττωση του κόστους. Τα προβλήματα αυτής της

κατηγορίας παρουσιάζουν συνδυαστική έκρηξη του χώρου αναζήτησης, ως προς το μέγεθος του προβλήματος, με αποτέλεσμα ο έλεγχος όλων των υποψήφιων λύσεων να είναι αδύνατος. Το πιο γνωστό πρόβλημα αυτής της κατηγορίας είναι αυτό του πλανόδιου πωλητή, όπου στόχος είναι η εύρεση της συντομότερης διαδρομής για την επίσκεψη ενός συνόλου πόλεων.

Οι γενετικοί αλγόριθμοι μπορούν να δώσουν σε αυτό το πρόβλημα πολλές λύσεις κοντά στη βέλτιστη. Ένα άλλο πρόβλημα είναι η αποθήκευση κιβωτίων (bin packing) και αφορά την εύρεση του βέλτιστου τρόπου αποθήκευσης ενός αριθμού κιβωτίων σε περιορισμένο χώρο και έχει μεγάλη πρακτική σημασία στη βιομηχανία.

Τέλος στην κατηγορία αυτών των εφαρμογών εντάσσονται και τα προβλήματα καταμερισμού – χρονοπρογραμματισμού εργασιών (Job shop & Flow shop scheduling).

Γίνεται φανερό λοιπόν ότι οι γενετικοί αλγόριθμοι έχουν εφαρμοστεί σε διάφορα προβλήματα της Τεχνητής Νοημοσύνης και ιδιαίτερα σε προβλήματα βελτιστοποίησης. Αποτελούν λοιπόν έναν εύκολο τρόπο επίλυσης προβλημάτων με μεγάλη δυνατότητα προσαρμογής [43].

3.5 Εφαρμογές Γενετικών Αλγορίθμων στην Κρυπτογραφία

Τεχνικές που λαμβάνονται από το χώρο της Τεχνητής Νοημοσύνης, όπως οι εξελικτικοί αλγόριθμοι (γενετικοί αλγόριθμοι και γενετικός προγραμματισμός κ.τ.λ.) κάνουν σταθερά όλο και πιο αισθητή την παρουσία στον τομέα της ασφάλειας των υπολογιστών, τόσο στο δίκτυο όσο και στον πολύ απαιτητικό χώρο της κρυπτογραφίας. Ιδιαίτερα οι γενετικοί αλγόριθμοι έχουν χρησιμοποιηθεί σε πάρα πολλούς τομείς για την λύση δύσκολων προβλημάτων, χρησιμοποιώντας την τεχνική της φυσικής εξέλιξης της ανάπτυξης. Τα τελευταία χρόνια έχουν αρχίσει να χρησιμοποιούνται γενετικοί αλγόριθμοι για την κατασκευή κρυπτογραφικών συναρτήσεων με άμεσο στόχο τα καλά κρυπτογραφικά κριτήρια. Ειδικότερα, μία περιοχή όπου οι ΓΑ έχουν δείξει καλά αποτελέσματα είναι η παραγωγή λογικών συναρτήσεων που είναι κατάλληλες για την κρυπτογραφία. Ωστόσο, δεδομένου ότι υπάρχουν πολλά πιθανά κριτήρια σχεδιασμού για λογικές συναρτήσεις στην

κρυπτογραφία, αυτό το πεδίο είναι ακόμα ανοιχτό για την έρευνα. Στην συγκεκριμένη ενότητα θα αναφερθούν μελέτες που έχουν γίνει , για την κατασκευή κρυπτογραφικών συναρτήσεων με την βοήθεια των γενετικών αλγορίθμων.

Η πρώτη εφαρμογή των γενετικών αλγορίθμων για την εξέλιξη των κρυπτογραφικά κατάλληλων λογικών συναρτήσεων έγινε το 1997. Στο συγκεκριμένο πείραμα οι συγγραφείς εξέλιξαν λογικές συναρτήσεις με υψηλή μη γραμμικότητα [24].

Το 1998 στην διδακτορική του διατριβή ο Clark σε μια από τις εφαρμογές που δημιούργησε ήταν η εξέλιξη λογικών συναρτήσεων υψηλή μη γραμμικότητα χρησιμοποιώντας τιε τεχνικές των Γενετικών Αλγορίθμων και τους αλγόριθμων Hill Climbing [09] (είναι μια τεχνική μαθηματικής βελτιστοποίησης, η οποία ανήκει στην οικογένεια της τοπικής αναζήτησης). Την ίδια χρονολογία χρησιμοποιήθηκε η τεχνική των Γενετικών Αλγορίθμων σε συνδυασμό με τους αλγόριθμους Hill Climbing με βήμα επαναφοράς, για να εξελίξουν λογικές συναρτήσεις με υψηλή μη γραμμικότητα, για μεγέθη έως 12 εισόδους. Στο συγκεκριμένο πείραμα μόνο με την χρησιμοποίηση του Γενετικού Αλγορίθμου βρέθηκαν ισοβαρείς συναρτήσεις οκτώ εισόδων με μη γραμμικότητα ίση με 112 και ανθεκτικότητα στις συσχετίσεις ίση με 1. Όταν συνδυάστηκε ο Γενετικός Αλγόριθμος με τον αλγόριθμο Hill Climbing, βρέθηκαν ισοβαρείς συναρτήσεις οκτώ εισόδων με μη γραμμικότητα ίση με 116 [25].

Το 2005 οι A. Tragha, F. Omary και A. Mouloudi, στο έργο τους “Genetic Algorithms Inspired Cryptography” παρουσίασαν το σύστημα ICIGA, όπου αποτελεί βελτιωμένη μορφή του συστήματος GAIC που είχαν δημιουργήσει οι ίδιοι. Το ICIGA είναι ένα σύστημα κρυπτογράφησης τμήματος όπου το μυστικό κλειδί δημιουργείται κατά τη διάρκεια κάθε συνεδρίας μέσω τυχαίας διαδικασίας. Για την κρυπτογράφηση χρησιμοποιείται η παρακάτω διαδικασία. Αρχικά, διασπάει το δυαδικό κωδικοποιημένο κείμενο σε τμήματα ίσου μεγέθους με βάση το μήκος του κλειδιού, και στη συνέχεια τα διασπάει αυτά τα τμήματα σε τμήματα του ίδιου μεγέθους. Στην συνέχεια γίνονται πράξεις με την βοήθεια των γενετικών αλγορίθμων και ειδικότερα με τις τεχνικές της διασταύρωσης και της μετάλλαξης. Οι διαδικασίες αυτές λειτουργούν σε τυχαία επιλεγμένα τμήματα και θέσεις σε αυτά τα τμήματα. Με βάση το ίχνος των πράξεων αυτών δημιουργείται ένα μυστικό κλειδί. Στη συνέχεια, καλύπτουν την θέση των φορέων γενετικού αλγορίθμου με την εφαρμογή αριστερής μετατόπισης. Έπειτα, εφαρμόζεται αριστερή μετατόπιση σε ολόκληρο τμήμα για να καλύψουν την κατανομή των τμημάτων. Τέλος, εφαρμόζονται τα ίδια βήματα για το υπόλοιπο του κειμένου(

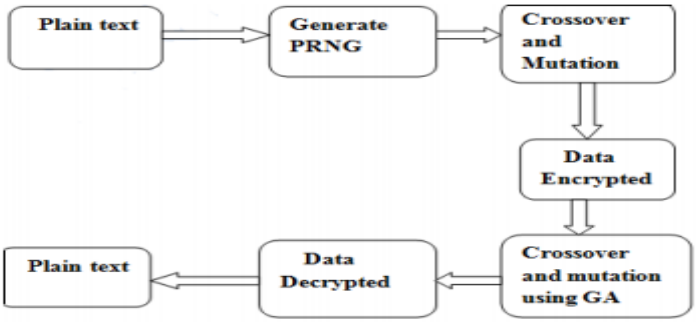
plaintext) αλλά χρησιμοποιούν το μυστικό κλειδί για να επιλέξουν λειτουργίες και τις θέσεις αντί του γενετικού αλγορίθμου. Οι εργασίες στη διαδικασία της αποκρυπτογράφησης γίνονται με αντίστροφη σειρά. Η τεχνική της διασταύρωσης γίνεται με μετάθεση των bits ανάμεσα σε δύο μπλοκ, και της μετάλλαξης γίνεται με την εφαρμογή λογικής άρνησης επί των bits[36].

Το 2008 προτάθηκε η δημιουργία νέων συναρτήσεων κατακερματισμού μέσω της δημιουργίας νέων αλγορίθμων τμήματος με βοήθεια των γενετικών αλγορίθμων. Συγκεκριμένα διάλεξαν έναν αλγόριθμο τμήματος με υψηλή μη γραμμικότητα, και μέσο του γενετικού αλγορίθμου (ρυθμίζοντας την αντικειμενική συνάρτηση του γενετικού αλγορίθμου), δημιούργησαν τον νέο αλγόριθμο τμήματος που ονομάστηκε Wheedham [14].

Το 2014 ο Sindhuja K. και ο Pramela Devi S. χρησιμοποίησαν τους Γενετικούς Αλγορίθμους για την παραγωγή συμμετρικών κλειδιών κρυπτογράφησης. Συγκεκριμένα χρησιμοποίησαν τους τελεστές της διασταύρωσης και της μετάλλαξης των γενετικών αλγορίθμων , για να δημιουργήσουν συναρτήσεις κρυπτογράφησης και αποκρυπτογράφησης. Για την κρυπτογράφηση, παρήγαγαν μέσω κάποιων τεχνικών ένα ενδιάμεσο κρυπτογράφημα και αυτό το περνούσαν αρχικά στον τελεστή της διασταύρωσης. Η τεχνική που χρησιμοποιήθηκε στον συγκεκριμένο τελεστή ήταν αυτή της διασταύρωσης δύο σημείων. Επιλέγονταν δηλαδή τυχαία 2 σημεία από τους γονείς και τα ενδιάμεσα σημεία των 2 γονέων ανταλλάσσονταν και παράγονταν οι απόγονοι. Στην συνέχεια γινόταν η μετάλλαξη των παραγόμενων απογόνων .Για τον τελεστή της μετάλλαξης χρησιμοποίησαν την τεχνική της αναστροφής bit (flipping bit). Μετά την μετάλλαξη παράγονταν το τελικό κρυπτογράφημα σε δεκαεξαδική μορφή. Για την αποκρυπτογράφηση γινόταν η αντίστροφη διαδικασία της κρυπτογράφησης [34].

Μία αντίστοιχη πρόταση έγινε από τον Pushpa B. R. το 2015. Στην συγκεκριμένη μελέτη αρχικά χρησιμοποίησε την ψευδο-γεννήτρια τυχαίων αριθμών.Η ψευδο-γεννήτρια τυχαίων αριθμών είναι ένας αλγόριθμος για την παραγωγή μιας ακολουθίας αριθμών που έχουν τις ιδιότητες των τυχαίων αριθμών. Στη συνέχεια μέσω των γενετικών αλγορίθμων χρησιμοποιήθηκαν οι τελεστές της διασταύρωσης και της μετάλλαξης για να δημιουργηθούν συναρτήσεις κρυπτογράφησης-αποκρυπτογράφησης. Στον τελεστή της διασταύρωσης χρησιμοποιήθηκε η τεχνική της διασταύρωσης μονού σημείου. Για τον τελεστή της μετάλλαξης χρησιμοποιήθηκε η

τεχνική της μετάλλαξη ενός bit string. Στο παρακάτω σχήμα (Σχήμα 3.8) ακολουθεί το διάγραμμα ροής της μεθόδου που προτάθηκε από τον Pushpa B. R. [31].



Σχήμα 3.8 : Διάγραμμα Ροής Μεθόδου «Pushpa B. R».

Το 2015 έγινε μια μελέτη από τους ερευνητές και μελέτησαν διάφορες μεθόδους εξελικτικών αλγορίθμων για την εξέλιξη των λογικών συναρτήσεων. Συγκεκριμένα έγιναν πειράματα με τις μεθόδους των γενετικών αλγορίθμων, του γενετικού προγραμματισμού και του καρτεσιανού γενετικού προγραμματισμού (είναι είδος γενετικού προγραμματισμού που αντιπροσωπεύει τα προγράμματα ως γραφήματα στα οποία υπάρχουν μη κωδικοποιημένα γονίδια). Πιο αναλυτικά για τους γενετικούς αλγόριθμους επιλέχτηκε ο απλός γενετικός αλγόριθμος με τελεστή επιλογής την «επιλογή τουρνουά » (είδος τελεστή επιλογής όπου Ν άτομα επιλέγονται τυχαία από τον πληθυσμό) εξάλειψης με μέγεθος Ν ίσο με τρία. Για τον τελεστή της διασταύρωσης επιλέχτηκε η ομοιόμορφη διασταύρωση. Ο πληθυσμός που διαλέχτηκε ήταν 50, 100, 200, 500 και 1000 και οι πιθανότητες μετάλλαξης 0.1, 0.3, 0.5, 0.7 και 0.9 αντίστοιχα. Βασικός στόχος των συναρτήσεων ήταν τα καλά κρυπτογραφικά κριτήρια, με ειδική βαρύτητα στην ανθεκτικότητας στις συσχετίσεις. Έτσι έκαναν δύο πειράματα για τον ορισμό της αντικειμενικής συνάρτησης. Στο πρώτο έβαλαν σαν στόχο την ανθεκτικότητα στις συσχετίσεις μεταξύ των τιμών 2 έως 5. Η τιμή ικανότητας , για την αντικειμενική συνάρτηση, δίνεται από τον παρακάτω τύπο :

$$fittness_1 = Bal + t + \delta_{t,x}(Nl(f) + AI + \deg)$$

Όπου bal παίρνει την τιμή 0 αν η συνάρτηση είναι ισοβαρής, αλλιώς παίρνει την αρνητική τιμή που εκφράζουν την διαφορά των μηδενικών από τους άσσους .Το χ είναι ο επιθυμητός στόχος στην ανθεκτικότητα στις συσχετίσεις (παίρνει τιμές από 2 έως 5) το $\delta_{t,x}$ είναι η συνάρτηση δέλτα του Kronecker και παίρνει την τιμή 1 εάν t=x αλλιώς παίρνει την τιμή 0. Το t είναι η ανθεκτικότητα στις συσχετίσεις, το $Nl(f)$ είναι η μη γραμμικότητα, το AI είναι η αλγεβρική ανθεκτικότητα και το deg ο βαθμός μιας λογικής συνάρτησης. Στο δεύτερο έβαλαν στόχο τις μη ισοβαρείς συναρτήσεις με ανθεκτικότητα στις συσχετίσεις (παίρνει τιμές από 1 έως 5) και το ελάχιστο βάρος του

πίνακα αληθείας της λογικής συνάρτησης. Η τιμή ικανότητας ,της αντικειμενικής συνάρτησης, δίνεται από τον τύπο :

$$fittness_2 = t + \delta_{t,x}(2^n - HW(TT)), \varepsilon\alpha\nu t = x$$

Αλλιώς,

$$fittness_2 = 2^n - HW(TT) - 2^n \left| t - x \right| ,$$

όπου το $HW(TT)$ είναι το βάρος του πίνακα αληθείας της λογικής συνάρτησης. Κατά το πείραμα τους με την χρήση των Γενετικών Αλγορίθμων κατασκεύασαν συναρτήσεις με ανθεκτικότητα στις συσχετίσεις ίση με ένα [29].

Το 2016 οι ίδιοι ερευνητές, χρησιμοποίησαν την ίδια μέθοδο για τους γενετικούς αλγόριθμους με κάποιες αλλαγές στους τύπους υπολογισμού της αντικειμενικής συνάρτησης. Συγκεκριμένα ο πρώτος τύπος που χρησιμοποιήθηκε στην αντικειμενική συνάρτηση για την τιμή ικανότητας ήταν ο παρακάτω:

$$fittness_1 = Bal + \delta_{bal,o}(Nl(f) + \deg + ANF\min imize + CI) ,$$

όπου Bal παίρνει την τιμή 0 αν η συνάρτηση είναι ισοβαρής αλλιώς παίρνει την αρνητική τιμή που εκφράζουν την διαφορά των μηδενικών από τους άσσους. Το $\delta_{bal,o}$ είναι η συνάρτηση δέλτα του Kronecker και παίρνει την τιμή 1 εάν t=x αλλιώς παίρνει την τιμή 0. Το $Nl(f)$ είναι η μη γραμμικότητα ,το deg ο βαθμός , το CI η ανθεκτικότητα στις συσχετίσεις μιας λογικής συνάρτησης. Το $ANF\min imize$ δίνεται από τον τύπο:

$$ANF\min imize = \frac{2^n - HW(ANF)}{HW(ANF)}$$

Ο δεύτερος τύπος που χρησιμοποιήθηκε στην αντικειμενική συνάρτηση για την τιμή ικανότητας ήταν η εξής :

$$fittness_2 = (MAX\_HW - \sup) - MAX\_HW * \left| CI - \mathrm{T}\,ARGET\_CI \right| ,$$

όπου MAX_HW το βάρος μιας λογικής συνάρτησης. Το sup αντιπροσωπεύει το πλήθος των γραμμών του πίνακα αληθείας της λογικής συνάρτησης των οποίων η έξοδος είναι 1. Το CI είναι η ανθεκτικότητα στης συσχετίσεις μιας λογικής συνάρτησης και τοTARGET_CI είναι η τάξη της ανθεκτικότητας στις συσχετίσεις που θέλουμε να αποκτήσει η συνάρτηση [30].

Κεφάλαιο 4

Κατασκευή Κρυπτογραφικών Συναρτήσεων Μέσω Γενετικών Αλγορίθμων

Για για την δημιουργία κρυπτογραφικών συναρτήσεων με την βοήθεια των γενετικών αλγορίθμων, κατασκευάστηκαν τέσσερα διαφορετικά προγράμματα γενετικών αλγορίθμων με τις ιδιότητες που θα παρουσιαστούν παρακάτω για το κάθε πρόγραμμα ξεχωριστά . Η γλώσσα προγραμματισμού που χρησιμοποιήθηκε ήταν η γλώσσα java στην πλατφόρμα Netbeans IDE 8.2.

Απώτερος στόχος μας ήταν η δημιουργία ισοβαρών συναρτήσεων (δηλαδή με βέλτιστο ισοζύγιο), υψηλής μη γραμμικότητας (με βάση αναφοράς τη μη γραμμικότητα της συνάρτησης Carlet-Feng) και υψηλής ανθεκτικότητας στις (γρήγορες) αλγεβρικές επιθέσεις (δηλαδή υψηλό FAI). Δεν εστιάσαμε στην ανθεκτικότητα στις συσχετίσεις - κάτι που κάνουν άλλοι ερευνητές – λόγω του ότι έχει σχετικά προσφάτως αποδειχθεί ότι αν μία συνάρτηση έχει καλή ανθεκτικότητα στις γρήγορες αλγεβρικές επιθέσεις τότε υποχρεωτικά δεν μπορεί να συμπεριφέρεται ιδανικά ως προς τις επιθέσεις συσχέτισης[21].

4.1 Κρυπτογραφικές Συναρτήσεις Μέσω Γενετικών Αλγορίθμων και Αποτελέσματα Εξόδων

Το είδος του γενετικού αλγόριθμου που χρησιμοποιήθηκε και στα τέσσερα προγράμματα είναι ο παραδοσιακός «Generational GA» όπως έχει αναφερθεί στο Κεφάλαιο 3 και συγκεκριμένα χρησιμοποιήθηκε ο γενετικός αλγόριθμος γενεών με

ελιτισμό «elitism Generational GA ». Η διαδικασία επιλογής των καλυτέρων για την επόμενη γενιά καλείται ελιτισμός, βάσει του οποίου ένας προκαθορισμένος αριθμός χρωμοσωμάτων που κρίνονται ως τα «πλέον κατάλληλα», επαναλαμβάνονται αυτούσια στο νέο πληθυσμό. Με τον τρόπο αυτό εξασφαλίζεται ότι κάποια «πλέον κατάλληλα», χρωμοσώματα θα συμπεριληφθούν στον νέο πληθυσμό και δε θα εξαφανιστούν ως αποτέλεσμα της εφαρμογής των υπόλοιπων τελεστών. Ο αριθμός τον «ελίτ» μελών πρέπει να είναι αρκετά μικρότερος από το μέγεθος του πληθυσμού για να είναι πιο αποτελεσματικός και πιο γρήγορος στην αναζήτηση ο γενετικός αλγόριθμος [44].

Για την υλοποίηση και των τεσσάρων προγραμμάτων, η κωδικοποίηση που χρησιμοποιήθηκε ήταν η δυαδική κωδικοποίηση. Επιχειρήθηκε κατασκευή λογικών συναρτήσεων με πλήθος μεταβλητών από n=5 έως n=9, δηλαδή για εξόδους 32 έως 512 bit αντίστοιχα. Οι πληθυσμοί που τέθηκαν ήταν οι παρακάτω: 500, 1000, 1500, 2000 με πιθανότητα μετάλλαξης 0.3 ,0.5, 0.7, 0.9 αντίστοιχα. Η επιλογή της πιθανότητας μετάλλαξης για κάθε τιμή του πληθυσμού βασίστηκε σε αντίστοιχες προσεγγίσεις που έχουν ακολουθεί κατά το παρελθόν, σε μια πρώτη προσπάθεια αποτίμησης της αποτελεσματικότητας του εν λόγω γενετικού αλγορίθμου για το πρόβλημα που μελετάται εδώ. Η πιθανότητα διασταύρωσης τέθηκε ίση με 0.99. Για τον υπολογισμό της συνάρτησης καταλληλότητας των προγραμμάτων τέθηκε σαν στόχος η έξοδος να είναι ισοβαρής και να έχει υψηλή μη γραμμικότητα. Ο λόγος που τέθηκαν μόνο αυτοί οι σχεδιαστικοί στόχοι είναι γιατί πρέπει η συνάρτηση καταλληλότητας να πραγματοποιεί γρήγορους υπολογισμούς. Για τον ελιτισμό στο πρώτο πρόγραμμα τέθηκε η μεταβλητή elitsmcount=2 ενώ για τα υπόλοιπα προγράμματα τέθηκε ο ελιτισμός σαν αληθής (true) , όπου έθετε την μεταβλητή elitsmcount=1. Η αναπαραγωγή των ατόμων του πληθυσμού γίνεται με τυχαίο τρόπο. Συγκεκριμένα:

α) Στο πρώτο πρόγραμμα, για την αναπαραγωγή ενός ατόμου χρησιμοποιείται η συνάρτηση «individual» της ομώνυμης κλάσης. Για κάθε byte για το μέγεθος του ατόμου παράγεται τυχαία ένας αριθμός αν αυτός ο αριθμός είναι μικρότερος από το 0.5 τότε το byte παίρνει την τιμή 0 αλλιώς παίρνει την τιμή 1.

β) Στα υπόλοιπα προγράμματα, για την αναπαραγωγή κάθε byte ενός ατόμου χρησιμοποιείται η εντολή «(byte) Math.round(Math.random());» στην συνάρτηση «generateIndividual» της κλάσης «individual», όπου παράγεται τυχαία ένας αριθμός ο οποίος στη συνέχεια στρογγυλοποιείται και μετατρέπεται σε byte.

Στα προγράμματα δόθηκε ανώτατο χρονικό όριο αναζήτησης της βέλτιστης λύσης. Για τα τρία πρώτα προγράμματα δόθηκε ανώτατο χρονικό περιθώριο τα 172800 δευτερόλεπτα (48 ώρες), ενώ για το τέταρτο πρόγραμμα τα 7200 δευτερόλεπτα (2 ώρες).

Σε όλα τα προγράμματα παράγονται δύο αρχεία μετά την εκτέλεσή τους. Το πρώτο αρχείο περιέχει την τελική λογική συνάρτηση που μας δίνει ο Γενετικός Αλγόριθμος, ενώ το δεύτερο περιέχει διάφορες ιδιότητες της κάθε εξόδου του Γενετικού Αλγορίθμου. Συγκεκριμένα παρέχει της εξής πληροφορίες: σε ποια γενιά παρήχθη η βέλτιστη λύση , σε πόση ώρα, και το ποσοστό επιτυχίας της βέλτιστης λύσης με βάσει τη συνάρτηση καταλληλότητας που είχε τεθεί, τη μη γραμμικότητά της, καθώς επίσης και αν είναι ισοβαρής ή όχι.

Στο πρώτο πρόγραμμα για τον τελεστή επιλογής επιλέχτηκε η επιλογή τροχού ρουλέτας όπως έχει αναφερθεί στο Κεφάλαιο 3. Για τον τελεστή της διασταύρωσης επιλέχτηκε η ομοιόμορφη διασταύρωση. Για τον τελεστή μετάλλαξης χρησιμοποιήθηκε η μετάλλαξη bit-flip όπως έχει αναφερθεί στο Κεφάλαιο 3. Ο μαθηματικός τύπος που τέθηκε για την συνάρτηση καταλληλότητας ήταν ο: Fitness= balance + nonlinearity.

Η μεταβλητή balance παίρνει την τιμή 0 όταν το άτομο που ελέγχεται είναι ισοβαρές, αλλιώς παίρνει την αρνητική τιμή που εκφράζουν την διαφορά των μηδενικών από τους άσσους. Επίσης υπολογίζεται και η τιμή της μη γραμμικότητας του ατόμου. Συνεπώς, για τις ισοβαρείς συναρτήσεις η τιμή fitness ισούται με την τιμή της μη γραμμικότητας. Στην συνθήκη τερματισμού τέθηκε σαν στόχος η μη γραμμικότητα της λογικής συνάρτησης Carlet -Feng όπως έχει αναφερθεί στο Κεφάλαιο 2. Συγκριμένα για n=5 έως n=9 δόθηκε η συνθήκη individual.getFitness()>= από (10,24,54,112,232) αντίστοιχα, οι οποίες είναι οι τιμές της συνάρτησης Carlet-Feng.

Στο δεύτερο πρόγραμμα για τον τελεστή επιλογής επιλέχτηκε η επιλογή τουρνουά “marriage tournament” όπως έχει αναφερθεί στο Κεφάλαιο 3, με μέγεθος τουρνουά ίσο με 2 και 3 αντίστοιχα. Για τον τελεστή της διασταύρωσης επιλέχτηκε η πολυσημιακή διασταύρωση όπως έχει αναφερθεί στο Κεφάλαιο 3. Τέλος για τον τελεστή μετάλλαξης χρησιμοποιήθηκε η ομοιόμορφη μετάλλαξη όπως έχει αναφερθεί στο Κεφάλαιο 3. Για την συνάρτηση καταλληλότητας χρησιμοποιήθηκε ίδιος μαθηματικός τύπος με αυτόν του πρώτου προγράμματος. Στην συνθήκη τερματισμού τέθηκε σαν κύριος στόχος η

μη γραμμικότητα της λογικής συνάρτησης Carlet-Feng όπως έχει αναφερθεί στο Κεφάλαιο 2, ενώ σε κάποιες περιπτώσεις τέθηκε επαυξημένη κατά δύο μονάδες. Πιο συγκεριμένα: για n=5 και n=6 τέθηκαν, ως επιθυμητές τιμές της μη γραμμικότητας, οι τιμές της συνάρτησης Carlet-Feng επαυξημένες κατά δύο μονάδες δηλαδή12 και 26 αντίστοιχα. Για n=7 έως n=9 τέθηκε ως συνθήκη τερματισμού η μη γραμμικότητα της συνάρτησης Carlet-Feng ,δηλαδή 54,112,232 αντίστοιχα.

Για την κατασκευή του τρίτου προγράμματος χρησιμοποιήθηκαν οι ίδιες κλάσεις και η ίδια μεθοδολογία με αυτές του δεύτερου προγράμματος με την μόνη διαφορά ότι έγινε αλλαγή στην μέθοδο της μετάλλαξης και στον τρόπο υπολογισμού της συνάρτησης καταλληλότητας. Η μέθοδος μετάλλαξης που χρησιμοποιήθηκε ήταν η bit flip ενός ή πολλαπλών σημείων μετάλλαξης όπως έχει αναφερθεί στο κεφάλαιο 3. Συγκεκριμένα με την βοήθεια της συνάρτησης «random_range» επιλέγεται τυχαία μία ελάχιστη τιμή «min» όσο είναι το μέγεθος του ατόμου. Δηλαδή, αν το άτομο είναι μεγέθους 32 bit (το οποίο σημαίνει ότι n=5) θα επιλεχτεί τυχαία μια τιμή από την θέση 0 ως την θέση 31. Στην συνέχεια πάλι με την βοήθεια της συνάρτησης « random_range» επιλέγεται τυχαία μία μέγιστη τιμή «max» από το εύρος τιμών της ελάχιστης τιμής ως το μέγεθος του ατόμου. Τέλος αν ικανοποιείται η συνθήκη μετάλλαξης, δηλαδή αν το «mutation_rate» είναι μεγαλύτερο μιας τιμής που παράγεται τυχαία, τότε για τις θέσεις από την ελάχιστή τιμή ως την μέγιστη, ελέγχεται κάθε bit του ατόμου και αν είναι 1 γίνεται 0 και το αντίθετο. Για την συνάρτηση καταλληλότητας εφαρμόστηκε ο τύπος :

fitness=balance+nonlinearity-difference,

όπου difference= nonlinearity_target – nonlinearity. Δηλαδή αν ο στόχος που έχουμε βάλει είναι η μη γραμμικότητα 12 και το άτομο έχει μη γραμμικότητα 10 τότε η μεταβλητή difference=12-10=2.

Για την κατασκευή του τέταρτου προγράμματος χρησιμοποιήθηκαν οι ίδιες κλάσεις και η ίδια μεθοδολογία με αυτές του τρίτου προγράμματος με την μόνη διαφορά ότι έγινε αλλαγή στην μέθοδο της μετάλλαξης και σαν στόχος της συνάρτησης καταλληλότητας ήταν η μη γραμμικότητα της συνάρτησης Carlet –Feng αυξημένη κατά 2 μονάδες για τις εισόδους από n=5 έως n=9. Η μέθοδος μετάλλαξης που χρησιμοποιήθηκε ήταν η bit string όπως έχει αναφερθεί στο κεφάλαιο 3. Συγκεκριμένα με την βοήθεια της συνάρτησης « random_range» επιλέγεται τυχαία μία τιμή «min» όσο είναι το μέγεθος του ατόμου. Δηλαδή, αν το άτομο είναι μεγέθους 32 bit θα επιλεχτεί τυχαία μια τιμή

από την θέση 0 ως την θέση 31. Στην συνέχεια αν ικανοποιείται η συνθήκη μετάλλαξης , δηλαδή αν το «mutation_rate» είναι μεγαλύτερο μιας τιμής που παράγεται τυχαία, τότε για την θέση της τυχαίας τιμής, ελέγχεται στην συγκεκριμένη θέση το bit του ατόμου και αν είναι 1 γίνεται 0 και το αντίθετο. Για την συνάρτηση καταλληλότητας για τις εισόδους από n=5 έως n=9 δόθηκαν οι τιμές 12,26,56,114 και 234 αντίστοιχα.

Για τα προγράμματα που κατασκευάστηκαν θα παρουσιαστούν 2 πίνακες. Ο πρώτος πίνακας περιέχει γενικές πληροφορίες για τις εξόδους (γενιά, χρόνος εκτέλεσης) που παράγονται από το πρόγραμμα. Στο δεύτερο πίνακα εξετάζεται αν οι έξοδοι του εκάστοτε προγράμματος (αλγορίθμου) ικανοποιούν κάποια λοιπά βασικά κρυπτογραφικά κριτήρια ώστε να αποτελούν καλές κρυπτογραφικές συναρτήσεις. Τα κριτήρια που εξετάζονται είναι τα εξής : αν είναι ισοβαρείς(BALANCED), η μη γραμμικότητα (NONLINEARITY), ο βαθμός (DEG) και ανθεκτικότητα στις γρήγορες αλγεβρικές επιθέσεις(FAI). Εξετάζοντας την ανθεκτικότητα στις γρήγορες αλγεβρικές επιθέσεις καλύπτουμε σε μεγάλο βαθμό και την περίπτωση των απλών αλγεβρικών επιθέσεων (AI). Τα δύο πρώτα κριτήρια (αν δηλαδή η έξοδος είναι ισοβαρής συνάρτηση αλλά και η τιμή της μη γραμμικότητάς της) εξετάζονται από το πρόγραμμα, αφού ουσιαστικά υπεισέρχονται στον προσδιορισμό της συνάρτησης καταλληλότητας και, τελικά, της βέλτιστης λύσης. Τα υπόλοιπα κριτήρια εξετάστηκαν εκ των υστέρων, πάνω στις συναρτήσεις που ο γενετικός αλγόριθμος υπολόγισε ως βέλτιστες, από το πρόγραμμα «FAA Equation Finder Version 1.0» όπου έχει υλοποιηθεί από τον Simon Fisher και ελέγχει αν μία συνάρτηση είναι ανθεκτική στις γρήγορες αλγεβρικές επιθέσεις και παράγει ένα αρχείο με τον βαθμό της συνάρτησης και την αλγεβρική κανονική της μορφή[15]. Για κάθε πληθυσμό του κάθε προγράμματος μελετήθηκαν πέντε έξοδοι, όπου με βάση τα κρυπτογραφικά κριτήρια κάποιες θα απορριφτούν και κάποιες θα αποτελούν καλές λύσεις για κρυπτογραφικές συναρτήσεις.

4.1.1 Πρώτο Πρόγραμμα Γενετικού Αλγορίθμου-Αποτελέσματα

Για την κατασκευή του αλγόριθμου χρησιμοποιήθηκαν οι εξής κλάσεις :

α) Boolean_ga : Περιέχει την εκτέλεση του προγράμματος.

β) GeneticAlgorithm: Περιέχει την βασική δομή του γενετικού αλγορίθμου και περιέχει τις συναρτήσεις της επιλογής, της διασταύρωσης, της μετάλλαξης, της συνθήκης

τερματισμού, του υπολογισμού της καταλληλότητας, και της αρχικοποίησης του πληθυσμού.

γ) Individual: Περιέχει την μέθοδο της τυχαίας δημιουργίας ατόμου και διάφορες μεθόδους set –get που παρέχουν πληροφορίες για το άτομο (μέγεθος, καταλληλότητα, γονίδιο) καθώς και η μέθοδος toString.

δ) Nonlinearity: Περιέχει την μέθοδο του υπολογισμού της μη γραμμικότητας ,τον υπολογισμό αν το τελικό αποτέλεσμα είναι ισοβαρής συνάρτηση, την αποθήκευση των τελικών αποτελεσμάτων σε 2 αρχεία (όπως εξηγήθηκαν στο Κεφ . 4.1) και τον υπολογισμό του ποσοστού επιτυχίας του γενετικού αλγορίθμου σε σχέση με τον στόχο που είχε τεθεί.

ε) Population: Περιέχει την μέθοδο της δημιουργίας πληθυσμού καθώς και μεθόδους set –get που παρέχουν πληροφορίες για τον πληθυσμό (μέγεθος, πιο ικανού του πληθυσμού, άτομο του πληθυσμού) καθώς και την μέθοδο της ανάμιξης του πληθυσμού.

Παρακάτω ακολουθούν οι πίνακες με τα αποτελέσματα του προγράμματος. Στους παρακάτω πίνακες όπου η μη γραμμικότητα είναι με μπλε χρώμα τότε σημαίνει ότι επιτεύχθηκε μη γραμμικότητα μεγαλύτερη της αναμενόμενης τιμής, ενώ με κόκκινο χρώμα είναι ελάχιστα μικρότερη της αναμενόμενης τιμής.

Για n=5 θα έχουμε έξοδο 2^n =32 bit

1. Αρχικοποίηση προγράμματος για εκτέλεση για πληθυσμό =500 :

Θέτουμε πληθυσμό ίσο με 500, πιθανότητα μετάλλαξης ίση με 0.3, πιθανότητα διασταύρωσης ίση με 0.99 , elitsmcount ίση με 2, ενώ ο αλγόριθμος τερματίζει όταν η συνάρτηση καταλληλότητας λάβει τιμή μεγαλύτερη ή ίση με 10 (που είναι η τιμή της μη γραμμικότητας της Carlet-Feng συνάρτησης για n=5).

A/A	**GENERATION**	**ΧΡΟΝΟΣ ΕΚΤΕΛΕΣΗΣ (SECONDS)**
1	1	0.04
2	1	0.02

3	1	0.02
4	1	0.01
5	1	0.01
Μέσος όρος		0.02

Πίνακας 4.1. Στατιστικά στοιχεία εξόδων συνάρτησης.

Α/Α	ΒΑΘΜΟΣ	BALANCED	NONLINEARITY	FAI
1	3	TRUE	12	-
2	3	TRUE	12	-
3	4	TRUE	10	-
4	4	TRUE	10	optimal
5	4	TRUE	10	-

Πίνακας 4.2. Κρυπτογραφικές ιδιότητες εξόδων συνάρτησης.

2. Αρχικοποίηση προγράμματος για εκτέλεση για πληθυσμό =1000 :

Θέτουμε πληθυσμό ίσο με 1000, πιθανότητα μετάλλαξης ίση με 0.5, πιθανότητα διασταύρωσης ίση με 0.99 , elitsmcount ίση με 2, ενώ ο αλγόριθμος τερματίζει όταν η συνάρτηση καταλληλότητας λάβει τιμή μεγαλύτερη ή ίση με 10 (που είναι η τιμή της μη γραμμικότητας της Carlet-Feng συνάρτησης για n=5).

Α/Α	GENERATION	ΧΡΟΝΟΣ ΕΚΤΕΛΕΣΗΣ (SECONDS)
1	1	0.01
2	1	0.02
3	1	0.02
4	1	0.02
5	1	0.01
Μέσος όρος		0.016

Πίνακας 4.3. Στατιστικά στοιχεία εξόδων συνάρτησης.

Α/Α	ΒΑΘΜΟΣ	BALANCED	NONLINEARITY	FAI
1	4	TRUE	10	-
2	4	TRUE	10	optimal

3	4	TRUE	12	-
4	3	TRUE	12	-
5	3	TRUE	12	-

Πίνακας 4.4. Κρυπτογραφικές ιδιότητες εξόδων συνάρτησης.

3. Αρχικοποίηση προγράμματος για εκτέλεση για πληθυσμό=1500 :

Θέτουμε πληθυσμό ίσο με 1500, πιθανότητα μετάλλαξης ίση με 0.7, πιθανότητα διασταύρωσης ίση με 0.99 , elitsmcount ίση με 2, ενώ ο αλγόριθμος τερματίζει όταν η συνάρτηση καταλληλότητας λάβει τιμή μεγαλύτερη ή ίση με 10 (που είναι η τιμή της μη γραμμικότητας της Carlet-Feng συνάρτησης για n=5).

Α/Α	**GENERATION**	**ΧΡΟΝΟΣ ΕΚΤΕΛΕΣΗΣ (SECONDS)**
1	1	0.02
2	1	0.02
3	1	0.02
4	1	0.02
5	1	0.02
Μέσος όρος		0.02

Πίνακας 4.5. Στατιστικά στοιχεία εξόδων συνάρτησης.

Α/Α	**ΒΑΘΜΟΣ**	**BALANCED**	**NONLINEARITY**	**FAI**
1	3	TRUE	12	-
2	4	TRUE	12	-
3	3	TRUE	12	-
4	4	TRUE	10	-
5	3	TRUE	12	-

Πίνακας 4.6. Κρυπτογραφικές ιδιότητες εξόδων συνάρτησης.

4. Αρχικοποίηση προγράμματος για εκτέλεση για πληθυσμό=2000:

Θέτουμε πληθυσμό ίσο με 2000, πιθανότητα μετάλλαξης ίση με 0.9, πιθανότητα διασταύρωσης ίση με 0.99 , elitsmcount ίση με 2, ενώ ο αλγόριθμος τερματίζει όταν η

συνάρτηση καταλληλότητας λάβει τιμή μεγαλύτερη ή ίση με 10 (που είναι η τιμή της μη γραμμικότητας της Carlet-Feng συνάρτησης για n=5).

A/A	GENERATION	ΧΡΟΝΟΣ ΕΚΤΕΛΕΣΗΣ (SECONDS)
1	1	0.02
2	1	0.03
3	1	0.03
4	1	0.02
5	1	0.02
Μέσος όρος		0.024

Πίνακας 4.7. Στατιστικά στοιχεία εξόδων συνάρτησης.

A/A	ΒΑΘΜΟΣ	BALANCED	NONLINEARITY	FAI
1	3	TRUE	12	-
2	3	TRUE	12	-
3	3	TRUE	12	-
4	3	TRUE	12	-
5	4	TRUE	12	-

Πίνακας 4.8. Κρυπτογραφικές ιδιότητες εξόδων συνάρτησης.

Για n=6 θα έχουμε έξοδο 2^n =64 bit

1. Αρχικοποίηση προγράμματος για εκτέλεση για πληθυσμό =500:

Θέτουμε πληθυσμό ίσο με 500, πιθανότητα μετάλλαξης ίση με 0.3, πιθανότητα διασταύρωσης ίση με 0.99 , elitsmcount ίση με 2, ενώ ο αλγόριθμος τερματίζει όταν η συνάρτηση καταλληλότητας λάβει τιμή μεγαλύτερη ή ίση με 24 (που είναι η τιμή της μη γραμμικότητας της Carlet-Feng συνάρτησης για n=6).

A/A	GENERATION	ΧΡΟΝΟΣ ΕΚΤΕΛΕΣΗΣ (SECONDS)
1	1	0.02

2	1	0.02
3	1	0.01
4	1	0.01
5	1	0.02
Μέσος όρος		0.018

Πίνακας 4.9. Στατιστικά στοιχεία εξόδων συνάρτησης.

Α/Α	ΒΑΘΜΟΣ	BALANCED	NONLINEARITY	FAI
1	5	TRUE	24	optimal
2	5	TRUE	24	optimal
3	5	TRUE	24	optimal
4	5	TRUE	24	optimal
5	5	TRUE	24	optimal

Πίνακας 4.10. Κρυπτογραφικές ιδιότητες εξόδων συνάρτησης.

2. Αρχικοποίηση προγράμματος για εκτέλεση για πληθυσμό =1000:

Θέτουμε πληθυσμό ίσο με 1000, πιθανότητα μετάλλαξης ίση με 0.5, πιθανότητα διασταύρωσης ίση με 0.99 , elitsmcount ίση με 2, ενώ ο αλγόριθμος τερματίζει όταν η συνάρτηση καταλληλότητας λάβει τιμή μεγαλύτερη ή ίση με 24 (που είναι η τιμή της μη γραμμικότητας της Carlet-Feng συνάρτησης για n=6).

Α/Α	GENERATION	ΧΡΟΝΟΣ ΕΚΤΕΛΕΣΗΣ (SECONDS)
1	1	0.02
2	1	0.02
3	1	0.02
4	1	0.02
5	1	0.02
Μέσος όρος		0.02

Πίνακας 4.11. Στατιστικά στοιχεία εξόδων συνάρτησης.

Α/Α	ΒΑΘΜΟΣ	BALANCED	NONLINEARITY	FAI
1	5	TRUE	24	optimal

2	5	TRUE	24	optimal
3	5	TRUE	24	optimal
4	5	TRUE	24	optimal
5	5	TRUE	24	optimal

Πίνακας 4.12. Κρυπτογραφικές ιδιότητες εξόδων συνάρτησης.

3. Αρχικοποίηση προγράμματος για εκτέλεση για πληθυσμό=1500:

Θέτουμε πληθυσμό ίσο με 1500, πιθανότητα μετάλλαξης ίση με 0.7, πιθανότητα διασταύρωσης ίση με 0.99 , elitsmcount ίση με 2, ενώ ο αλγόριθμος τερματίζει όταν η συνάρτηση καταλληλότητας λάβει τιμή μεγαλύτερη ή ίση με 24 (που είναι η τιμή της μη γραμμικότητας της Carlet-Feng συνάρτησης για n=6).

Α/Α	**GENERATION**	**ΧΡΟΝΟΣ ΕΚΤΕΛΕΣΗΣ (SECONDS)**
1	1	0.03
2	1	0.03
3	1	0.03
4	1	0.03
5	1	0.03
Μέσος όρος		0.03

Πίνακας 4.13. Στατιστικά στοιχεία εξόδων συνάρτησης.

Α/Α	**ΒΑΘΜΟΣ**	**BALANCED**	**NONLINEARITY**	**FAI**
1	5	TRUE	24	optimal
2	5	TRUE	24	optimal
3	5	TRUE	24	optimal
4	5	TRUE	24	optimal
5	5	TRUE	24	optimal

Πίνακας 4.14. Κρυπτογραφικές ιδιότητες εξόδων συνάρτησης.

4. Αρχικοποίηση προγράμματος για εκτέλεση για πληθυσμό=2000:

Θέτουμε πληθυσμό ίσο με 2000, πιθανότητα μετάλλαξης ίση με 0.9, πιθανότητα διασταύρωσης ίση με 0.99 , elitsmcount ίση με 2, ενώ ο αλγόριθμος τερματίζει όταν η συνάρτηση καταλληλότητας λάβει τιμή μεγαλύτερη ή ίση με 24 (που είναι η τιμή της μη γραμμικότητας της Carlet-Feng συνάρτησης για n=6).

A/A	GENERATION	ΧΡΟΝΟΣ ΕΚΤΕΛΕΣΗΣ (SECONDS)
1	1	0.04
2	1	0.03
3	1	0.03
4	1	0.03
5	1	0.04
Μέσος όρος		0,034

Πίνακας 4.15. Στατιστικά στοιχεία εξόδων συνάρτησης.

A/A	ΒΑΘΜΟΣ	BALANCED	NONLINEARITY	FAI
1	5	TRUE	26	optimal
2	5	TRUE	24	optimal
3	5	TRUE	24	optimal
4	5	TRUE	24	optimal
5	5	TRUE	24	optimal

Πίνακας 4.16. Κρυπτογραφικές ιδιότητες εξόδων συνάρτησης.

Για n=7 θα έχουμε έξοδο 2^n =128 bit

1. Αρχικοποίηση προγράμματος για εκτέλεση για πληθυσμό =500 :

Θέτουμε πληθυσμό ίσο με 500, πιθανότητα μετάλλαξης ίση με 0.3, πιθανότητα διασταύρωσης ίση με 0.99 , elitsmcount ίση με 2, ενώ ο αλγόριθμος τερματίζει όταν η

συνάρτηση καταλληλότητας λάβει τιμή μεγαλύτερη ή ίση με 54 (που είναι η τιμή της μη γραμμικότητας της Carlet-Feng συνάρτησης για n=7).

Α/Α	GENERATION	ΧΡΟΝΟΣ ΕΚΤΕΛΕΣΗΣ (SECONDS)
1	250	2.75
2	1614	15.68
3	4662	44.36
4	1878	19.06
5	5139	18.42
Μέσος όρος		20.054

Πίνακας 4.17. Στατιστικά στοιχεία εξόδων συνάρτησης.

Α/Α	ΒΑΘΜΟΣ	BALANCED	NONLINEARITY	FAI
1	6	TRUE	54	optimal
2	6	TRUE	54	optimal
3	6	TRUE	54	optimal
4	6	TRUE	54	optimal
5	6	TRUE	54	optimal

Πίνακας 4.18. Κρυπτογραφικές ιδιότητες εξόδων συνάρτησης.

2. Αρχικοποίηση προγράμματος για εκτέλεση για πληθυσμό =1000:

Θέτουμε πληθυσμό ίσο με 1000, πιθανότητα μετάλλαξης ίση με 0.5, πιθανότητα διασταύρωσης ίση με 0.99 , elitsmcount ίση με 2, ενώ ο αλγόριθμος τερματίζει όταν η συνάρτηση καταλληλότητας λάβει τιμή μεγαλύτερη ή ίση με 54 (που είναι η τιμή της μη γραμμικότητας της Carlet-Feng συνάρτησης για n=7).

Α/Α	GENERATION	ΧΡΟΝΟΣ ΕΚΤΕΛΕΣΗΣ (SECONDS)
1	1030	21.76

2	303	6.54
3	519	11.04
4	1491	31.44
5	1105	23.22
Μέσος όρος		18.8

Πίνακας 4.19. Στατιστικά στοιχεία εξόδων συνάρτησης.

Α/Α	ΒΑΘΜΟΣ	BALANCED	NONLINEARITY	FAI
1	6	TRUE	54	optimal
2	6	TRUE	54	optimal
3	6	TRUE	54	optimal
4	6	TRUE	54	optimal
5	6	TRUE	54	optimal

Πίνακας 4.20. Κρυπτογραφικές ιδιότητες εξόδων συνάρτησης.

3. Αρχικοποίηση προγράμματος για εκτέλεση για πληθυσμό=1500 :

Θέτουμε πληθυσμό ίσο με 1500, πιθανότητα μετάλλαξης ίση με 0.7, πιθανότητα διασταύρωσης ίση με 0.99 , elitsmcount ίση με 2, ενώ ο αλγόριθμος τερματίζει όταν η συνάρτηση καταλληλότητας λάβει τιμή μεγαλύτερη ή ίση με 54 (που είναι η τιμή της μη γραμμικότητας της Carlet-Feng συνάρτησης για n=7).

Α/Α	GENERATION	ΧΡΟΝΟΣ ΕΚΤΕΛΕΣΗΣ (SECONDS)
1	248	8.86
2	1561	54.76
3	481	17.65
4	324	11.72
5	3851	134.61
Μέσος όρος		45.52

Πίνακας 4.21. Στατιστικά στοιχεία εξόδων συνάρτησης.

Α/Α	ΒΑΘΜΟΣ	BALANCED	NONLINEARITY	FAI
1	6	TRUE	54	optimal
2	6	TRUE	54	optimal
3	6	TRUE	54	optimal
4	6	TRUE	54	optimal
5	6	TRUE	54	optimal

Πίνακας 4.22. Κρυπτογραφικές ιδιότητες εξόδων συνάρτησης.

4. Αρχικοποίηση προγράμματος για εκτέλεση για πληθυσμό=2000 :

Θέτουμε πληθυσμό ίσο με 2000, πιθανότητα μετάλλαξης ίση με 0.9, πιθανότητα διασταύρωσης ίση με 0.99 , elitsmcount ίση με 2, ενώ ο αλγόριθμος τερματίζει όταν η συνάρτηση καταλληλότητας λάβει τιμή μεγαλύτερη ή ίση με 54 (που είναι η τιμή της μη γραμμικότητας της Carlet-Feng συνάρτησης για n=7).

Α/Α	GENERATION	ΧΡΟΝΟΣ ΕΚΤΕΛΕΣΗΣ (SECONDS)
1	807	45.97
2	418	25.56
3	44	2.93
4	750	48.12
5	9	0.69
Μέσος όρος		24.654

Πίνακας 4.23. Στατιστικά στοιχεία εξόδων συνάρτησης.

Α/Α	ΒΑΘΜΟΣ	BALANCED	NONLINEARITY	FAI
1	6	TRUE	54	optimal
2	6	TRUE	54	optimal
3	6	TRUE	54	optimal
4	6	TRUE	54	optimal
5	6	TRUE	54	optimal

Πίνακας 4.24. Κρυπτογραφικές ιδιότητες εξόδων συνάρτησης.

Για n=8 θα έχουμε έξοδο 2^n =256 bit

1. Αρχικοποίηση προγράμματος για εκτέλεση για πληθυσμό =500 :

Θέτουμε πληθυσμό ίσο με 500, πιθανότητα μετάλλαξης ίση με 0.3, πιθανότητα διασταύρωσης ίση με 0.99 , elitsmcount ίση με 2, ενώ ο αλγόριθμος τερματίζει όταν η συνάρτηση καταλληλότητας λάβει τιμή μεγαλύτερη ή ίση με 112 (που είναι η τιμή της μη γραμμικότητας της Carlet-Feng συνάρτησης για n=8).

Α/Α	**GENERATION**	**ΧΡΟΝΟΣ ΕΚΤΕΛΕΣΗΣ (SECONDS)**
1	16643	288.77
2	1226	21.7
3	5507	96.63
4	7145	135.74
5	13987	258.17
Μέσος όρος		160.202

Πίνακας 4.25. Στατιστικά στοιχεία εξόδων συνάρτησης.

Α/Α	**ΒΑΘΜΟΣ**	**BALANCED**	**NONLINEARITY**	**FAI**
1	7	TRUE	112	optimal
2	7	TRUE	112	optimal
3	6	TRUE	112	optimal
4	7	TRUE	112	optimal
5	7	TRUE	112	optimal

Πίνακας 4.26. Κρυπτογραφικές ιδιότητες εξόδων συνάρτησης.

2. Αρχικοποίηση προγράμματος για εκτέλεση για πληθυσμό =1000 :

Θέτουμε πληθυσμό ίσο με 1000, πιθανότητα μετάλλαξης ίση με 0.5, πιθανότητα διασταύρωσης ίση με 0.99 , elitsmcount ίση με 2, ενώ ο αλγόριθμος τερματίζει όταν η συνάρτηση καταλληλότητας λάβει τιμή μεγαλύτερη ή ίση με 112 (που είναι η τιμή της μη γραμμικότητας της Carlet-Feng συνάρτησης για n=8).

A/A	**GENERATION**	**ΧΡΟΝΟΣ ΕΚΤΕΛΕΣΗΣ (SECONDS)**
1	723	27.19
2	1391	56.38
3	11217	427.75
4	1582	61.63
5	13619	524.84
Μέσος όρος		219.558

Πίνακας 4.27. Στατιστικά στοιχεία εξόδων συνάρτησης.

A/A	**ΒΑΘΜΟΣ**	**BALANCED**	**NONLINEARITY**	**FAI**
1	7	TRUE	112	optimal
2	7	TRUE	112	optimal
3	6	TRUE	112	optimal
4	6	TRUE	112	optimal
5	7	TRUE	112	optimal

Πίνακας 4.28. Κρυπτογραφικές ιδιότητες εξόδων συνάρτησης.

3. Αρχικοποίηση προγράμματος για εκτέλεση για πληθυσμό=1500 :

Θέτουμε πληθυσμό ίσο με 1500, πιθανότητα μετάλλαξης ίση με 0.7, πιθανότητα διασταύρωσης ίση με 0.99 , elitsmcount ίση με 2, ενώ ο αλγόριθμος τερματίζει όταν η συνάρτηση καταλληλότητας λάβει τιμή μεγαλύτερη ή ίση με 112 (που είναι η τιμή της μη γραμμικότητας της Carlet-Feng συνάρτησης για n=8).

A/A	**GENERATION**	**ΧΡΟΝΟΣ ΕΚΤΕΛΕΣΗΣ**

		(SECONDS)
1	859	53.31
2	343	22.8
3	6685	427.73
4	945	59.46
5	3051	188.1
Μέσος όρος		150.28

Πίνακας 4.29. Στατιστικά στοιχεία εξόδων συνάρτησης.

Α/Α	ΒΑΘΜΟΣ	BALANCED	NONLINEARITY	FAI
1	7	TRUE	112	optimal
2	7	TRUE	112	optimal
3	7	TRUE	112	optimal
4	7	TRUE	112	optimal
5	7	TRUE	112	optimal

Πίνακας 4.30. Κρυπτογραφικές ιδιότητες εξόδων συνάρτησης.

4. Αρχικοποίηση προγράμματος για εκτέλεση για πληθυσμό=2000 :

Θέτουμε πληθυσμό ίσο με 2000, πιθανότητα μετάλλαξης ίση με 0.9, πιθανότητα διασταύρωσης ίση με 0.99 , elitsmcount ίση με 2, ενώ ο αλγόριθμος τερματίζει όταν η συνάρτηση καταλληλότητας λάβει τιμή μεγαλύτερη ή ίση με 112 (που είναι η τιμή της μη γραμμικότητας της Carlet-Feng συνάρτησης για n=8).

Α/Α	GENERATION	ΧΡΟΝΟΣ ΕΚΤΕΛΕΣΗΣ (SECONDS)
1	2199	221.21
2	864	88.13
3	1174	119.23
4	2610	279.13
5	540	59.65
Μέσος όρος		153.47

Πίνακας 4.31. Στατιστικά στοιχεία εξόδων συνάρτησης.

Α/Α	ΒΑΘΜΟΣ	BALANCED	NONLINEARITY	FAI
1	6	TRUE	112	optimal
2	7	TRUE	112	optimal
3	7	TRUE	112	optimal
4	7	TRUE	112	optimal
5	7	TRUE	112	optimal

Πίνακας 4.32. Κρυπτογραφικές ιδιότητες εξόδων συνάρτησης.

Για n=9 θα έχουμε έξοδο 2^n =512 bit

1. Αρχικοποίηση προγράμματος για εκτέλεση για πληθυσμό =500 :

Θέτουμε πληθυσμό ίσο με 500, πιθανότητα μετάλλαξης ίση με 0.3, πιθανότητα διασταύρωσης ίση με 0.99 , elitsmcount ίση με 2, ενώ ο αλγόριθμος τερματίζει όταν η συνάρτηση καταλληλότητας λάβει τιμή μεγαλύτερη ή ίση με 232 (που είναι η τιμή της μη γραμμικότητας της Carlet-Feng συνάρτησης για n=9).

Α/Α	GENERATION	ΧΡΟΝΟΣ ΕΚΤΕΛΕΣΗΣ (SECONDS)
1	4900224	172800.00
2	4446360	172800.00
3	4492920	172800.01
4	4491120	172800.01
5	4491744	172800.01
Μέσος όρος		172800.006

Πίνακας 4.33. Στατιστικά στοιχεία εξόδων συνάρτησης.

Α/Α	ΒΑΘΜΟΣ	BALANCED	NONLINEARITY	FAI
1	8	TRUE	230	-
2	8	TRUE	230	-
3	8	TRUE	230	-
4	8	TRUE	230	-

5	8	TRUE	230	optimal

Πίνακας 4.34. Κρυπτογραφικές ιδιότητες εξόδων συνάρτησης.

Όπως υποδηλώνει το κόκκινο χρώμα στις τιμές της μη γραμμικότητας, δεν κατέστη εφικτή η εύρεση συνάρτησης με την επιθυμητή μη γραμμικότητα.

2. Αρχικοποίηση προγράμματος για εκτέλεση για πληθυσμό =1000 :

Θέτουμε πληθυσμό ίσο με 1000, πιθανότητα μετάλλαξης ίση με 0.5, πιθανότητα διασταύρωσης ίση με 0.99 , elitsmcount ίση με 2, ενώ ο αλγόριθμος τερματίζει όταν η συνάρτηση καταλληλότητας λάβει τιμή μεγαλύτερη ή ίση με 232 (που είναι η τιμή της μη γραμμικότητας της Carlet-Feng συνάρτησης για n=9).

Α/Α	**GENERATION**	**ΧΡΟΝΟΣ ΕΚΤΕΛΕΣΗΣ (SECONDS)**
1	2318424	172800.05
2	2397144	172800.01
3	2400072	172800.03
4	2391264	172800.01
5	2389272	172800.05
Μέσος όρος		172800.03

Πίνακας 4.35. Στατιστικά στοιχεία εξόδων συνάρτησης.

Α/Α	**ΒΑΘΜΟΣ**	**BALANCED**	**NONLINEARITY**	**FAI**
1	8	TRUE	230	-
2	8	TRUE	230	-
3	8	TRUE	230	optimal
4	8	TRUE	230	-
5	8	TRUE	230	-

Πίνακας 4.36. Κρυπτογραφικές ιδιότητες εξόδων συνάρτησης.

Όπως υποδηλώνει το κόκκινο χρώμα στις τιμές της μη γραμμικότητας, δεν κατέστη εφικτή η εύρεση συνάρτησης με την επιθυμητή μη γραμμικότητα.

3. Αρχικοποίηση προγράμματος για εκτέλεση για πληθυσμό=1500 :

Θέτουμε πληθυσμό ίσο με 1500, πιθανότητα μετάλλαξης ίση με 0.7, πιθανότητα διασταύρωσης ίση με 0.99 , elitsmcount ίση με 2, ενώ ο αλγόριθμος τερματίζει όταν η συνάρτηση καταλληλότητας λάβει τιμή μεγαλύτερη ή ίση με 232 (που είναι η τιμή της μη γραμμικότητας της Carlet-Feng συνάρτησης για n=9).

Α/Α	GENERATION	ΧΡΟΝΟΣ ΕΚΤΕΛΕΣΗΣ (SECONDS)
1	742584	172800.18
2	783648	172800.03
3	773328	172800.20
4	768936	172800.14
5	767016	172800.09
Μέσος όρος		172800.128

Πίνακας 4.37. Στατιστικά στοιχεία εξόδων συνάρτησης.

Α/Α	ΒΑΘΜΟΣ	BALANCED	NONLINEARITY	FAI
1	8	TRUE	230	-
2	8	TRUE	230	-
3	8	TRUE	230	-
4	8	TRUE	230	-
5	8	TRUE	230	-

Πίνακας 4.38. Κρυπτογραφικές ιδιότητες εξόδων συνάρτησης.

Όπως υποδηλώνει το κόκκινο χρώμα στις τιμές της μη γραμμικότητας, δεν κατέστη εφικτή η εύρεση συνάρτησης με την επιθυμητή μη γραμμικότητα.

4. Αρχικοποίηση προγράμματος για εκτέλεση για πληθυσμό=2000 :

Θέτουμε πληθυσμό ίσο με 2000, πιθανότητα μετάλλαξης ίση με 0.9, πιθανότητα διασταύρωσης ίση με 0.99 , elitsmcount ίση με 2, ενώ ο αλγόριθμος τερματίζει όταν η

συνάρτηση καταλληλότητας λάβει τιμή μεγαλύτερη ή ίση με 232 (που είναι η τιμή της μη γραμμικότητας της Carlet-Feng συνάρτησης για n=9).

Α/Α	GENERATION	ΧΡΟΝΟΣ ΕΚΤΕΛΕΣΗΣ (SECONDS)
1	1000080	172800.06
2	1006152	172800.15
3	1005000	172800.08
4	1003344	172800.03
5	1003320	172800.15
Μέσος όρος		172800.09

Πίνακας 4.39. Στατιστικά στοιχεία εξόδων συνάρτησης

Α/Α	ΒΑΘΜΟΣ	BALANCED	NONLINEARITY	FAI
1	8	TRUE	230	-
2	8	TRUE	230	optimal
3	8	TRUE	230	-
4	8	TRUE	230	-
5	8	TRUE	230	-

Πίνακας 4.40. Κρυπτογραφικές ιδιότητες εξόδων συνάρτησης.

Όπως υποδηλώνει το κόκκινο χρώμα στις τιμές της μη γραμμικότητας, δεν κατέστη εφικτή η εύρεση συνάρτησης με την επιθυμητή μη γραμμικότητα.

4.1.2 Δεύτερο Πρόγραμμα Γενετικού Αλγορίθμου-Αποτελέσματα

Για την κατασκευή του αλγόριθμου χρησιμοποιήθηκαν οι εξής κλάσεις :

α) Boolean_GA_2: Περιέχει την εκτέλεση του προγράμματος.

β) Algorithm: Αποτελεί την βασική δομή του γενετικού αλγορίθμου και περιέχει τις συναρτήσεις της επιλογής, της διασταύρωσης, της μετάλλαξης και της διαδικασίας της εξέλιξης του πληθυσμού.

γ) Individual: Περιέχει την μέθοδο της τυχαίας δημιουργίας ατόμου και τις μεθόδους set –get που παρέχουν πληροφορίες για το άτομο (μέγεθος, καταλληλότητα, γονίδιο) καθώς και την μέθοδο toString.

δ) Nonlinearity: Περιέχει την μέθοδο του υπολογισμού της μη γραμμικότητας ,τον υπολογισμό αν το τελικό αποτέλεσμα είναι ισοβαρής συνάρτηση, την αποθήκευση των τελικών αποτελεσμάτων σε 2 αρχεία (όπως εξηγήθηκαν στο Κεφ . 4.1) και τον υπολογισμό του ποσοστού επιτυχίας του γενετικού αλγορίθμου σε σχέση με τον στόχο που είχε τεθεί.

ε) Population: Περιέχει την μέθοδο της δημιουργίας πληθυσμού καθώς και μεθόδους get που παρέχουν πληροφορίες για τον πληθυσμό (μέγεθος, πιο ικανού του πληθυσμού, άτομο του πληθυσμού) καθώς και την μέθοδο της αποθήκευσης του πληθυσμού.

στ)Fitnesscalc: Περιέχει την συνάρτηση καταλληλότητας και την συνθήκη τερματισμού.

Παρακάτω ακολουθούν οι πίνακες με τα αποτελέσματα του προγράμματος. Στους παρακάτω πίνακες όπου υπάρχει T=2 και T=3 εννοείται ότι το tournamentsize για την μέθοδος επιλογής τουρνουά ήταν ίσο με 2 και 3 αντίστοιχα. Όπου η μη γραμμικότητα είναι με κόκκινο χρώμα στους παρακάτω πίνακες, είναι ελάχιστα μικρότερη της αναμενόμενης τιμής.

Για n=5 θα έχουμε έξοδο 2^n =32bit

1. Αρχικοποίηση προγράμματος για εκτέλεση για πληθυσμό =500 :

Θέτουμε πληθυσμό ίσο με 500, πιθανότητα μετάλλαξης ίση με 0.3, πιθανότητα διασταύρωσης ίση με 0.99 , elitsm ίσο με true, tournamentsize ίσο με 2 και 3 αντίστοιχα, ενώ ο αλγόριθμος τερματίζει όταν η συνάρτηση καταλληλότητας λάβει τιμή ίση με 12 (που είναι η τιμή της μη γραμμικότητας της Carlet-Feng συνάρτησης για n=5 επαυξημένη κατά 2 μονάδες).

Α/Α T=2	**GENERATION**	**ΧΡΟΝΟΣ ΕΚΤΕΛΕΣΗΣ (SECONDS)**
1	1	0.03
2	1	0.04
3	1	0.01

4	7	0.12
5	1	0.01
A/A T=3		
1	1	0.02
2	1	0.01
3	3	0.05
4	2	0.03
5	1	0.01
Μέσος όρος T2-T3		0.042-0.024

Πίνακας 4.41. Στατιστικά στοιχεία εξόδων συνάρτησης.

A/A T=2	ΒΑΘΜΟΣ	BALANCED	NONLINEARITY	FAI
1	4	TRUE	12	-
2	3	TRUE	12	-
3	4	TRUE	12	-
4	3	TRUE	12	-
5	3	TRUE	12	-
A/A T=3				
1	3	TRUE	12	-
2	3	TRUE	12	-
3	3	TRUE	12	-
4	4	TRUE	12	-
5	4	TRUE	12	-

Πίνακας 4.42. Κρυπτογραφικές ιδιότητες εξόδων συνάρτησης.

2. Αρχικοποίηση προγράμματος για εκτέλεση για πληθυσμό =1000 :

Θέτουμε πληθυσμό ίσο με 1000, πιθανότητα μετάλλαξης ίση με 0.5, πιθανότητα διασταύρωσης ίση με 0.99 , elitsm ίσο με true, tournamentsize ίσο με 2 και 3 αντίστοιχα , ενώ ο αλγόριθμος τερματίζει όταν η συνάρτηση καταλληλότητας λάβει τιμή ίση με 12 (που είναι η τιμή της μη γραμμικότητας της Carlet-Feng συνάρτησης για n=5 επαυξημένη κατά 2 μονάδες).

A/A T=2	GENERATION	ΧΡΟΝΟΣ ΕΚΤΕΛΕΣΗΣ (SECONDS)
1	1	0.03
2	1	0.02
3	1	0.01
4	1	0.02

5	1	0.01
A/A T=3		
1	1	0.03
2	2	0.06
3	2	0.04
4	1	0.01
5	1	0.01
Μέσος όρος T2-T3		0.018-0.03

Πίνακας 4.43. Στατιστικά στοιχεία εξόδων συνάρτησης.

A/A T=2	**ΒΑΘΜΟΣ**	**BALANCED**	**NONLINEARITY**	**FAI**
1	3	TRUE	12	-
2	3	TRUE	12	-
3	4	TRUE	12	-
4	3	TRUE	12	-
5	3	TRUE	12	-
A/A T=3				
1	3	TRUE	12	-
2	3	TRUE	12	-
3	3	TRUE	12	-
4	3	TRUE	12	-
5	3	TRUE	12	-

Πίνακας 4.44. Κρυπτογραφικές ιδιότητες εξόδων συνάρτησης.

3. Αρχικοποίηση προγράμματος για εκτέλεση για πληθυσμό=1500 :

Θέτουμε πληθυσμό ίσο με 1500, πιθανότητα μετάλλαξης ίση με 0.7, πιθανότητα διασταύρωσης ίση με 0.99 , elitsm ίσο με true, tournamentsize ίσο με 2 και 3 αντίστοιχα , ενώ ο αλγόριθμος τερματίζει όταν η συνάρτηση καταλληλότητας λάβει τιμή ίση με 12 (που είναι η τιμή της μη γραμμικότητας της Carlet-Feng συνάρτησης για n=5 επαυξημένη κατά 2 μονάδες).

A/A T=2	**GENERATION**	**ΧΡΟΝΟΣ ΕΚΤΕΛΕΣΗΣ (SECONDS)**
1	1	0.05
2	1	0.02
3	1	0.01
4	1	0.02
5	1	0.02

A/A T=3		
1	1	0.04
2	1	0.03
3	1	0.02
4	1	0.01
5	1	0.01
Μέσος όρος T2-T3		0.024-0.022

Πίνακας 4.45. Στατιστικά στοιχεία εξόδων συνάρτησης.

A/A T=2	**ΒΑΘΜΟΣ**	**BALANCED**	**NONLINEARITY**	**FAI**
1	3	TRUE	12	-
2	3	TRUE	12	-
3	4	TRUE	12	-
4	4	TRUE	12	-
5	3	TRUE	12	-
A/A T=3				
1	3	TRUE	12	-
2	3	TRUE	12	-
3	3	TRUE	12	-
4	3	TRUE	12	-
5	3	TRUE	12	-

Πίνακας 4.46. Κρυπτογραφικές ιδιότητες εξόδων συνάρτησης.

4. Αρχικοποίηση προγράμματος για εκτέλεση για πληθυσμό=2000 :

Θέτουμε πληθυσμό ίσο με 2000, πιθανότητα μετάλλαξης ίση με 0.9, πιθανότητα διασταύρωσης ίση με 0.99 , elitsm ίσο με true, tournamentsize ίσο με 2 και 3 αντίστοιχα, ενώ ο αλγόριθμος τερματίζει όταν η συνάρτηση καταλληλότητας λάβει τιμή ίση με 12 (που είναι η τιμή της μη γραμμικότητας της Carlet-Feng συνάρτησης για n=5 επαυξημένη κατά 2 μονάδες).

A/A T=2	**GENERATION**	**ΧΡΟΝΟΣ ΕΚΤΕΛΕΣΗΣ (SECONDS)**
1	1	0.04
2	1	0.03
3	1	0.02
4	1	0.02
5	1	0.02
A/A T=3		

1	1	0.06
2	1	0.03
3	1	0.02
4	1	0.02
5	1	0.02
Μέσος όρος Τ2-Τ3		0.026-0.03

Πίνακας 4.47. Στατιστικά στοιχεία εξόδων συνάρτησης.

Α/Α Τ=2	**ΒΑΘΜΟΣ**	**BALANCED**	**NONLINEARITY**	**FAI**
1	3	TRUE	12	-
2	3	TRUE	12	-
3	4	TRUE	12	-
4	3	TRUE	12	-
5	3	TRUE	12	-
Α/Α Τ=3				
1	4	TRUE	12	-
2	3	TRUE	12	-
3	4	TRUE	12	-
4	4	TRUE	12	-
5	3	TRUE	12	-

Πίνακας 4.48. Κρυπτογραφικές ιδιότητες εξόδων συνάρτησης.

Για n=6 θα έχουμε έξοδο 2^n =64 bit

1. Αρχικοποίηση προγράμματος για εκτέλεση για πληθυσμό =500 :

Θέτουμε πληθυσμό ίσο με 500, πιθανότητα μετάλλαξης ίση με 0.3, πιθανότητα διασταύρωσης ίση με 0.99 , elitsm ίσο με true, tournamentsize ίσο με 2 και 3 αντίστοιχα, ενώ ο αλγόριθμος τερματίζει όταν η συνάρτηση καταλληλότητας λάβει τιμή ίση με 26 (που είναι η τιμή της μη γραμμικότητας της Carlet-Feng συνάρτησης για n=6 επαυξημένη κατά 2 μονάδες).

Α/Α Τ=2	**GENERATION**	**ΧΡΟΝΟΣ ΕΚΤΕΛΕΣΗΣ (SECONDS)**
1	876	15.48
2	1	0.01
3	6562	113.8
4	1360	25.35
5	7351	152.74

Α/Α Τ=3		
1	418	9.43
2	1812	40.46
3	333	7.65
4	5198	116.15
5	3559	78.98
Μέσος όρος Τ2-Τ3		61.48-50.53

Πίνακας 4.49. Στατιστικά στοιχεία εξόδων συνάρτησης.

Α/Α Τ=2	**ΒΑΘΜΟΣ**	**BALANCED**	**NONLINEARITY**	**FAI**
1	5	TRUE	26	optimal
2	5	TRUE	26	optimal
3	5	TRUE	26	optimal
4	5	TRUE	26	optimal
5	5	TRUE	26	optimal
Α/Α Τ=3				
1	5	TRUE	26	optimal
2	5	TRUE	26	optimal
3	5	TRUE	26	optimal
4	5	TRUE	26	optimal
5	5	TRUE	26	Optimal

Πίνακας 4.50. Κρυπτογραφικές ιδιότητες εξόδων συνάρτησης.

2. Αρχικοποίηση προγράμματος για εκτέλεση για πληθυσμό =1000 :

Θέτουμε πληθυσμό ίσο με 1000, πιθανότητα μετάλλαξης ίση με 0.5, πιθανότητα διασταύρωσης ίση με 0.99 , elitsm ίσο με true, tournamentsize ίσο με 2 και 3 αντίστοιχα , ενώ ο αλγόριθμος τερματίζει όταν η συνάρτηση καταλληλότητας λάβει τιμή ίση με 26 (που είναι η τιμή της μη γραμμικότητας της Carlet-Feng συνάρτησης για n=6 επαυξημένη κατά 2 μονάδες).

Α/Α Τ=2	**GENERATION**	**ΧΡΟΝΟΣ ΕΚΤΕΛΕΣΗΣ (SECONDS)**
1	166	6.58
2	3486	134.14
3	917	35.75
4	399	15.42
5	1045	40.21
Α/Α		

T=3		
1	1090	53.23
2	1917	96.04
3	231	12.44
4	643	32.49
5	631	30.5
Μέσος όρος T2-T3		46.42-44.94

Πίνακας 4.51. Στατιστικά στοιχεία εξόδων συνάρτησης.

A/A T=2	ΒΑΘΜΟΣ	BALANCED	NONLINEARITY	FAI
1	5	TRUE	26	optimal
2	5	TRUE	26	optimal
3	5	TRUE	26	optimal
4	5	TRUE	26	optimal
5	5	TRUE	26	optimal
A/A T=3				
1	5	TRUE	26	optimal
2	5	TRUE	26	optimal
3	5	TRUE	26	optimal
4	5	TRUE	26	optimal
5	5	TRUE	26	optimal

Πίνακας 4.52. Κρυπτογραφικές ιδιότητες εξόδων συνάρτησης.

3. Αρχικοποίηση προγράμματος για εκτέλεση για πληθυσμό=1500 :

Θέτουμε πληθυσμό ίσο με 1500, πιθανότητα μετάλλαξης ίση με 0.7, πιθανότητα διασταύρωσης ίση με 0.99 , elitsm ίσο με true, tournamentsize ίσο με 2 και 3 αντίστοιχα , ενώ ο αλγόριθμος τερματίζει όταν η συνάρτηση καταλληλότητας λάβει τιμή ίση με 26 (που είναι η τιμή της μη γραμμικότητας της Carlet-Feng συνάρτησης για n=6 επαυξημένη κατά 2 μονάδες).

A/A T=2	GENERATION	ΧΡΟΝΟΣ ΕΚΤΕΛΕΣΗΣ (SECONDS)
1	930	55.0
2	329	19.39
3	117	6.87
4	805	47.18
5	1303	77.63
A/A T=3		

1	3088	236.3
2	190	14.46
3	134	10.16
4	180	13.64
5	610	47.53
Μέσος όρος Τ2-Τ3		41.21-64.42

Πίνακας 4.53. Στατιστικά στοιχεία εξόδων συνάρτησης.

Α/Α Τ=2	**ΒΑΘΜΟΣ**	**BALANCED**	**NONLINEARITY**	**FAI**
1	5	TRUE	26	optimal
2	5	TRUE	26	optimal
3	5	TRUE	26	optimal
4	5	TRUE	26	optimal
5	5	TRUE	26	optimal
Α/Α Τ=3				
1	5	TRUE	26	optimal
2	5	TRUE	26	optimal
3	5	TRUE	26	optimal
4	5	TRUE	26	optimal
5	5	TRUE	26	optimal

Πίνακας 4.54. Κρυπτογραφικές ιδιότητες εξόδων συνάρτησης.

4. Αρχικοποίηση προγράμματος για εκτέλεση για πληθυσμό=2000 :

Θέτουμε πληθυσμό ίσο με 2000, πιθανότητα μετάλλαξης ίση με 0.9, πιθανότητα διασταύρωσης ίση με 0.99 , elitsm ίσο με true, tournamentsize ίσο με 2 και 3 αντίστοιχα , ενώ ο αλγόριθμος τερματίζει όταν η συνάρτηση καταλληλότητας λάβει τιμή ίση με 26 (που είναι η τιμή της μη γραμμικότητας της Carlet-Feng συνάρτησης για n=6 επαυξημένη κατά 2 μονάδες).

Α/Α Τ=2	**GENERATION**	**ΧΡΟΝΟΣ ΕΚΤΕΛΕΣΗΣ (SECONDS)**
1	597	48.57
2	1409	107.66
3	566	43.64
4	4	0.26
5	213	16.46
Α/Α Τ=3		
1	136	13.5
2	409	40.08

3	899	88.42
4	513	58.72
5	133	13.88
Μέσος όρος Τ2-Τ3		43.32-42.92

Πίνακας 4.55. Στατιστικά στοιχεία εξόδων συνάρτησης.

Α/Α Τ=2	**ΒΑΘΜΟΣ**	**BALANCED**	**NONLINEARITY**	**FAI**
1	5	TRUE	26	optimal
2	5	TRUE	26	optimal
3	5	TRUE	26	optimal
4	5	TRUE	26	optimal
5	5	TRUE	26	optimal
Α/Α Τ=3				
1	5	TRUE	26	optimal
2	5	TRUE	26	optimal
3	5	TRUE	26	optimal
4	5	TRUE	26	optimal
5	5	TRUE	26	optimal

Πίνακας 4.56. Κρυπτογραφικές ιδιότητες εξόδων συνάρτησης.

Για n=7 θα έχουμε έξοδο 2^n =128 bit

1. Αρχικοποίηση προγράμματος για εκτέλεση για πληθυσμό =500 :

Θέτουμε πληθυσμό ίσο με 500, πιθανότητα μετάλλαξης ίση με 0.3, πιθανότητα διασταύρωσης ίση με 0.99 , elitsm ίσο με true, tournamentsize ίσο με 2 και 3 αντίστοιχα , ενώ ο αλγόριθμος τερματίζει όταν η συνάρτηση καταλληλότητας λάβει τιμή ίση με 54 (που είναι η τιμή της μη γραμμικότητας της Carlet-Feng συνάρτησης για n=7).

Α/Α Τ=2	**GENERATION**	**ΧΡΟΝΟΣ ΕΚΤΕΛΕΣΗΣ (SECONDS)**
1	415	15.37
2	1534	55.11
3	1417	50.91
4	4385	157.63
5	4374	157.34
Α/Α Τ=3		
1	8971	424.66
2	1773	82.33
3	7350	333.96
4	7010	319.44

5	2638	119.93
Μέσος όρος Τ2-Τ3		87.27-256.06

Πίνακας 4.57. Στατιστικά στοιχεία εξόδων συνάρτησης.

Α/Α Τ=2	**ΒΑΘΜΟΣ**	**BALANCED**	**NONLINEARITY**	**FAI**
1	6	TRUE	54	optimal
2	6	TRUE	54	optimal
3	6	TRUE	54	optimal
4	6	TRUE	54	optimal
5	6	TRUE	54	optimal
Α/Α Τ=3				
1	6	TRUE	54	optimal
2	6	TRUE	54	optimal
3	6	TRUE	54	optimal
4	6	TRUE	54	optimal
5	6	TRUE	54	optimal

Πίνακας 4.58. Κρυπτογραφικές ιδιότητες εξόδων συνάρτησης.

2. Αρχικοποίηση προγράμματος για εκτέλεση για πληθυσμό =1000 :

Θέτουμε πληθυσμό ίσο με 1000, πιθανότητα μετάλλαξης ίση με 0.5, πιθανότητα διασταύρωσης ίση με 0.99 , elitsm ίσο με true, tournamentsize ίσο με 2 και 3 αντίστοιχα, ενώ ο αλγόριθμος τερματίζει όταν η συνάρτηση καταλληλότητας λάβει τιμή ίση με 54 (που είναι η τιμή της μη γραμμικότητας της Carlet-Feng συνάρτησης για n=7).

Α/Α Τ=2	**GENERATION**	**ΧΡΟΝΟΣ ΕΚΤΕΛΕΣΗΣ (SECONDS)**
1	711	54.99
2	1746	138.97
3	941	75.69
4	1271	101.88
5	594	46.69
Α/Α Τ=3		
1	510	47.29
2	231	22.1
3	4232	411.84
4	573	54.54
5	106	10.83
Μέσος όρος Τ2-Τ3		83.64-109.32

Πίνακας 4.59. Στατιστικά στοιχεία εξόδων συνάρτησης.

A/A T=2	ΒΑΘΜΟΣ	BALANCED	NONLINEARITY	FAI
1	6	TRUE	54	optimal
2	6	TRUE	54	optimal
3	6	TRUE	54	optimal
4	6	TRUE	54	optimal
5	6	TRUE	54	optimal
A/A T=3				
1	6	TRUE	54	optimal
2	6	TRUE	54	optimal
3	6	TRUE	54	optimal
4	6	TRUE	54	optimal
5	6	TRUE	54	optimal

Πίνακας 4.60. Κρυπτογραφικές ιδιότητες εξόδων συνάρτησης.

3. Αρχικοποίηση προγράμματος για εκτέλεση για πληθυσμό=1500 :

Θέτουμε πληθυσμό ίσο με 1500, πιθανότητα μετάλλαξης ίση με 0.7, πιθανότητα διασταύρωσης ίση με 0.99 , elitsm ίσο με true, tournamentsize ίσο με 2 και 3 αντίστοιχα , ενώ ο αλγόριθμος τερματίζει όταν η συνάρτηση καταλληλότητας λάβει τιμή ίση με 54 (που είναι η τιμή της μη γραμμικότητας της Carlet-Feng συνάρτησης για n=7).

A/A T=2	GENERATION	ΧΡΟΝΟΣ ΕΚΤΕΛΕΣΗΣ (SECONDS)
1	2371	286.1
2	1290	160.85
3	27	4.02
4	891	113.77
5	544	65.21
A/A T=3		
1	2012	290.03
2	289	41.6
3	709	102.92
4	1001	144.09
5	392	56.47
Μέσος όρος T2-T3		125.99-115.73

Πίνακας 4.61. Στατιστικά στοιχεία εξόδων συνάρτησης.

Α/Α Τ=2	ΒΑΘΜΟΣ	BALANCED	NONLINEARITY	FAI
1	6	TRUE	54	optimal
2	6	TRUE	54	optimal
3	6	TRUE	54	optimal
4	6	TRUE	54	optimal
5	6	TRUE	54	optimal
Α/Α Τ=3				
1	6	TRUE	54	optimal
2	6	TRUE	54	optimal
3	6	TRUE	54	optimal
4	6	TRUE	54	optimal
5	6	TRUE	54	optimal

Πίνακας 4.62. Κρυπτογραφικές ιδιότητες εξόδων συνάρτησης.

4. Αρχικοποίηση προγράμματος για εκτέλεση για πληθυσμό=2000 :

Θέτουμε πληθυσμό ίσο με 2000, πιθανότητα μετάλλαξης ίση με 0.9, πιθανότητα διασταύρωσης ίση με 0.99 , elitsm ίσο με true, tournamentsize ίσο με 2 και 3 αντίστοιχα , ενώ ο αλγόριθμος τερματίζει όταν η συνάρτηση καταλληλότητας λάβει τιμή ίση με 54 (που είναι η τιμή της μη γραμμικότητας της Carlet-Feng συνάρτησης για n=7).

Α/Α Τ=2	GENERATION	ΧΡΟΝΟΣ ΕΚΤΕΛΕΣΗΣ (SECONDS)
1	270	41.68
2	1185	187.49
3	832	132.34
4	481	73.6
5	528	80.22
Α/Α Τ=3		
1	144	33.23
2	269	60.89
3	36	8.78
4	139	31.55
5	299	74.81
Μέσος όρος Τ2-Τ3		103.06-41.85

Πίνακας 4.63. Στατιστικά στοιχεία εξόδων συνάρτησης.

Α/Α Τ=2	ΒΑΘΜΟΣ	BALANCED	NONLINEARITY	FAI
1	6	TRUE	54	optimal
2	6	TRUE	54	optimal

3	6	TRUE	54	optimal
4	6	TRUE	54	optimal
5	6	TRUE	54	optimal
A/A T=3				
1	6	TRUE	54	optimal
2	6	TRUE	54	optimal
3	6	TRUE	54	optimal
4	6	TRUE	54	optimal
5	6	TRUE	54	optimal

Πίνακας 4.64. Κρυπτογραφικές ιδιότητες εξόδων συνάρτησης.

Για n=8 θα έχουμε έξοδο 2^n =256 bit

1. Αρχικοποίηση προγράμματος για εκτέλεση για πληθυσμό =500:

Θέτουμε πληθυσμό ίσο με 500, πιθανότητα μετάλλαξης ίση με 0.3, πιθανότητα διασταύρωσης ίση με 0.99 , elitsm ίσο με true, tournamentsize ίσο με 2 και 3 αντίστοιχα , ενώ ο αλγόριθμος τερματίζει όταν η συνάρτηση καταλληλότητας λάβει τιμή ίση με 112 (που είναι η τιμή της μη γραμμικότητας της Carlet-Feng συνάρτησης για n=8).

A/A T=2	**GENERATION**	**ΧΡΟΝΟΣ ΕΚΤΕΛΕΣΗΣ (SECONDS)**
1	60222	4195.35
2	29561	2090.94
3	217	86.39
4	23936	1687.61
5	41060	2857.25
A/A T=3		
1	18956	1670.59
2	87428	7685.67
3	14774	1295.21
4	12095	1059.7
5	44733	3925.49
Μέσος όρος T2-T3		1612.06-3127.42

Πίνακας 4.65. Στατιστικά στοιχεία εξόδων συνάρτησης.

A/A T=2	**ΒΑΘΜΟΣ**	**BALANCED**	**NONLINEARITY**	**FAI**
1	7	TRUE	112	optimal
2	7	TRUE	112	optimal
3	7	TRUE	112	optimal
4	6	TRUE	112	optimal

5	7	TRUE	112	optimal
A/A T=3				
1	7	TRUE	112	optimal
2	7	TRUE	112	optimal
3	7	TRUE	112	optimal
4	6	TRUE	112	optimal
5	7	TRUE	112	optimal

Πίνακας 4.66. Κρυπτογραφικές ιδιότητες εξόδων συνάρτησης.

2. Αρχικοποίηση προγράμματος για εκτέλεση για πληθυσμό =1000:

Θέτουμε πληθυσμό ίσο με 1000, πιθανότητα μετάλλαξης ίση με 0.5, πιθανότητα διασταύρωσης ίση με 0.99 , elitsm ίσο με true, tournamentsize ίσο με 2 και 3 αντίστοιχα , ενώ ο αλγόριθμος τερματίζει όταν η συνάρτηση καταλληλότητας λάβει τιμή ίση με 112 (που είναι η τιμή της μη γραμμικότητας της Carlet-Feng συνάρτησης για n=8).

A/A T=2	**GENERATION**	**ΧΡΟΝΟΣ ΕΚΤΕΛΕΣΗΣ (SECONDS)**
1	5063	763.27
2	76	11.99
3	732	110.25
4	9727	1463.76
5	2980	448.17
A/A T=3		
1	1871	347.21
2	15312	2839.09
3	2127	394.27
4	1399	260.29
5	2738	507.45
Μέσος όρος T2-T3		557.09-869.66

Πίνακας 4.67. Στατιστικά στοιχεία εξόδων συνάρτησης.

A/A T=2	**ΒΑΘΜΟΣ**	**BALANCED**	**NONLINEARITY**	**FAI**
1	6	TRUE	112	optimal
2	7	TRUE	112	optimal
3	7	TRUE	112	optimal
4	7	TRUE	112	optimal
5	6	TRUE	112	optimal
A/A				

T=3				
1	6	TRUE	112	optimal
2	7	TRUE	112	optimal
3	7	TRUE	112	optimal
4	7	TRUE	112	optimal
5	6	TRUE	112	optimal

Πίνακας 4.68. Κρυπτογραφικές ιδιότητες εξόδων συνάρτησης.

3. Αρχικοποίηση προγράμματος για εκτέλεση για πληθυσμό=1500:

Θέτουμε πληθυσμό ίσο με 1500, πιθανότητα μετάλλαξης ίση με 0.7, πιθανότητα διασταύρωσης ίση με 0.99 , elitsm ίσο με true, tournamentsize ίσο με 2 και 3 αντίστοιχα , ενώ ο αλγόριθμος τερματίζει όταν η συνάρτηση καταλληλότητας λάβει τιμή ίση με 112 (που είναι η τιμή της μη γραμμικότητας της Carlet-Feng συνάρτησης για n=8).

Α/Α T=2	GENERATION	ΧΡΟΝΟΣ ΕΚΤΕΛΕΣΗΣ (SECONDS)
1	3614	834.65
2	1557	361.42
3	1866	433.67
4	2732	634.4
5	18253	4239.76
Α/Α T=3		
1	6299	2279.31
2	4394	1556.48
3	2920	932.92
4	12387	3599.02
5	7707	2236.31
Μέσος όρος T2-T3		1300.78-2120.81

Πίνακας 4.69. Στατιστικά στοιχεία εξόδων συνάρτησης.

Α/Α T=2	ΒΑΘΜΟΣ	BALANCED	NONLINEARITY	FAI
1	7	TRUE	112	optimal
2	7	TRUE	112	optimal
3	7	TRUE	112	optimal
4	7	TRUE	112	optimal
5	7	TRUE	112	optimal
Α/Α T=3				
1	7	TRUE	112	optimal

2	7	TRUE	112	optimal
3	7	TRUE	112	optimal
4	7	TRUE	112	optimal
5	7	TRUE	112	optimal

Πίνακας 4.70. Κρυπτογραφικές ιδιότητες εξόδων συνάρτησης.

4. Αρχικοποίηση προγράμματος για εκτέλεση για πληθυσμό=2000 :

Θέτουμε πληθυσμό ίσο με 2000, πιθανότητα μετάλλαξης ίση με 0.9, πιθανότητα διασταύρωσης ίση με 0.99 , elitsm ίσο με true, tournamentsize ίσο με 2 και 3 αντίστοιχα , ενώ ο αλγόριθμος τερματίζει όταν η συνάρτηση καταλληλότητας λάβει τιμή ίση με 112 (που είναι η τιμή της μη γραμμικότητας της Carlet-Feng συνάρτησης για n=8).

A/A T=2	**GENERATION**	**ΧΡΟΝΟΣ ΕΚΤΕΛΕΣΗΣ (SECONDS)**
1	3492	1087.11
2	1172	361.78
3	2370	726.78
4	2946	900.0
5	787	243.25
A/A T=3		
1	4719	1971.3
2	3500	1491.67
3	2170	846.41
4	9225	3590.16
5	2751	1069.88
Μέσος όρος T2-T3		3124.32-1793.88

Πίνακας 4.71. Στατιστικά στοιχεία εξόδων συνάρτησης.

A/A T=2	**ΒΑΘΜΟΣ**	**BALANCED**	**NONLINEARITY**	**FAI**
1	6	TRUE	112	optimal
2	7	TRUE	112	optimal
3	7	TRUE	112	optimal
4	6	TRUE	112	optimal
5	7	TRUE	112	optimal
A/A T=3				
1	7	TRUE	112	optimal
2	7	TRUE	112	optimal
3	6	TRUE	112	optimal
4	7	TRUE	112	optimal

5	7	TRUE	112	optimal

Πίνακας 4.72. Κρυπτογραφικές ιδιότητες εξόδων συνάρτησης.

Για n=9 θα έχουμε έξοδο 2^n =512 bit

1. Αρχικοποίηση προγράμματος για εκτέλεση για πληθυσμό =500 :

Θέτουμε πληθυσμό ίσο με 500, πιθανότητα μετάλλαξης ίση με 0.3, πιθανότητα διασταύρωσης ίση με 0.99 , elitsm ίσο με true, tournamentsize ίσο με 2 και 3 αντίστοιχα , ενώ ο αλγόριθμος τερματίζει όταν η συνάρτηση καταλληλότητας λάβει τιμή ίση με 232(που είναι η τιμή της μη γραμμικότητας της Carlet-Feng συνάρτησης για n=9).

A/A T=2	**GENERATION**	**ΧΡΟΝΟΣ ΕΚΤΕΛΕΣΗΣ (SECONDS)**
1	847536	172800.15
2	854424	172800.22
3	859080	172800.11
4	865344	172800.17
5	866232	172800.12
A/A T=3		
1	667584	172800.08
2	670008	172800.31
3	664512	172800.17
4	666264	172800.27
5	663312	172800.30
Μέσος όρος T2-T3		172800.15-172800.5

Πίνακας 4.73. Στατιστικά στοιχεία εξόδων συνάρτησης.

A/A T=2	**ΒΑΘΜΟΣ**	**BALANCED**	**NONLINEARITY**	**FAI**
1	8	TRUE	230	-
2	8	TRUE	230	-
3	8	TRUE	230	-
4	8	TRUE	230	-
5	8	TRUE	230	-
A/A T=3				
1	8	TRUE	230	-
2	8	TRUE	230	-

3	8	TRUE	230	-
4	9	FALSE	230	-
5	8	TRUE	230	-

Πίνακας 4.74. Κρυπτογραφικές ιδιότητες εξόδων συνάρτησης.

Όπως υποδηλώνει το κόκκινο χρώμα στις τιμές της μη γραμμικότητας, δεν κατέστη εφικτή η εύρεση συνάρτησης με την επιθυμητή μη γραμμικότητα.

2. Αρχικοποίηση προγράμματος για εκτέλεση για πληθυσμό =1000 :

Θέτουμε πληθυσμό ίσο με 1000, πιθανότητα μετάλλαξης ίση με 0.5, πιθανότητα διασταύρωσης ίση με 0.99 , elitsm ίσο με true, tournamentsize ίσο με 2 και 3 αντίστοιχα , ενώ ο αλγόριθμος τερματίζει όταν η συνάρτηση καταλληλότητας λάβει τιμή ίση με 232(που είναι η τιμή της μη γραμμικότητας της Carlet-Feng συνάρτησης για n=9).

A/A T=2	**GENERATION**	**ΧΡΟΝΟΣ ΕΚΤΕΛΕΣΗΣ (SECONDS)**
1	420144	172800.34
2	421392	172800.48
3	423816	172800.27
4	422880	172800.26
5	419712	172800.20
A/A T=3		
1	333612	172800.59
2	336336	172800.57
3	333984	172800.29
4	333986	172800.32
5	332616	172800.52
Μέσος όρος T2-T3		172800.31-172800.46

Πίνακας 4.75. Στατιστικά στοιχεία εξόδων συνάρτησης.

A/A T=2	**ΒΑΘΜΟΣ**	**BALANCED**	**NONLINEARITY**	**FAI**
1	8	TRUE	230	-
2	8	TRUE	230	-
3	8	TRUE	230	-
4	8	TRUE	230	-
5	8	TRUE	230	-
A/A T=3				
1	8	TRUE	230	-
2	8	TRUE	230	-

3	8	TRUE	230	optimal
4	8	TRUE	230	-
5	8	TRUE	230	-

Πίνακας 4.76. Κρυπτογραφικές ιδιότητες εξόδων συνάρτησης.

Όπως υποδηλώνει το κόκκινο χρώμα στις τιμές της μη γραμμικότητας, δεν κατέστη εφικτή η εύρεση συνάρτησης με την επιθυμητή μη γραμμικότητα.

3. Αρχικοποίηση προγράμματος για εκτέλεση για πληθυσμό=1500 :

Θέτουμε πληθυσμό ίσο με 1500, πιθανότητα μετάλλαξης ίση με 0.7, πιθανότητα διασταύρωσης ίση με 0.99 , elitsm ίσο με true, tournamentsize ίσο με 2 και 3 αντίστοιχα , ενώ ο αλγόριθμος τερματίζει όταν η συνάρτηση καταλληλότητας λάβει τιμή ίση με 232(που είναι η τιμή της μη γραμμικότητας της Carlet-Feng συνάρτησης για n=9).

Α/Α Τ=2	**GENERATION**	**ΧΡΟΝΟΣ ΕΚΤΕΛΕΣΗΣ (SECONDS)**
1	281088	172800.48
2	281136	172800.73
3	281616	172800.40
4	277968	172800.37
5	281808	172800.74
Α/Α Τ=3		
1	271608	172800.20
2	273120	172800.49
3	273648	172800.83
4	272688	172800.73
5	273360	172800.36
Μέσος όρος Τ2-Τ3		172800.54-172800.52

Πίνακας 4.77. Στατιστικά στοιχεία εξόδων συνάρτησης.

Α/Α Τ=2	**ΒΑΘΜΟΣ**	**BALANCED**	**NONLINEARITY**	**FAI**
1	8	TRUE	230	-
2	8	TRUE	230	-
3	8	TRUE	230	-
4	8	TRUE	230	-
5	8	TRUE	230	-
Α/Α Τ=3				
1	8	TRUE	230	-
2	8	TRUE	230	-
3	8	TRUE	230	optimal

4	8	TRUE	230	-
5	8	TRUE	230	-

Πίνακας 4.78. Κρυπτογραφικές ιδιότητες εξόδων συνάρτησης.

Όπως υποδηλώνει το κόκκινο χρώμα στις τιμές της μη γραμμικότητας, δεν κατέστη εφικτή η εύρεση συνάρτησης με την επιθυμητή μη γραμμικότητα.

4. Αρχικοποίηση προγράμματος για εκτέλεση για πληθυσμό=2000 :

Θέτουμε πληθυσμό ίσο με 2000, πιθανότητα μετάλλαξης ίση με 0.9, πιθανότητα διασταύρωσης ίση με 0.99 , elitsm ίσο με true, tournamentsize ίσο με 2 και 3 αντίστοιχα , ενώ ο αλγόριθμος τερματίζει όταν η συνάρτηση καταλληλότητας λάβει τιμή ίση με 232(που είναι η τιμή της μη γραμμικότητας της Carlet-Feng συνάρτησης για n=9).

A/A T=2	**GENERATION**	**ΧΡΟΝΟΣ ΕΚΤΕΛΕΣΗΣ (SECONDS)**
1	204648	172800.06
2	206160	172800.60
3	205992	172800.40
4	206376	172800.35
5	206378	172800.72
A/A T=3		
1	164616	172800.72
2	163920	172800.02
3	163776	172800.80
4	162624	172800.03
5	162960	172800.34
Μέσος όρος T2-T3		172800.43-172800.38

Πίνακας 4.79. Στατιστικά στοιχεία εξόδων συνάρτησης.

A/A T=2	**ΒΑΘΜΟΣ**	**BALANCED**	**NONLINEARITY**	**FAI**
1	8	TRUE	230	optimal
2	8	TRUE	230	-
3	8	TRUE	230	-
4	8	TRUE	230	-
5	8	TRUE	230	-
A/A T=3				
1	8	TRUE	230	optimal

2	8	TRUE	230	-
3	8	TRUE	230	-
4	8	TRUE	230	-
5	8	TRUE	230	-

Πίνακας 4.80. Κρυπτογραφικές ιδιότητες εξόδων συνάρτησης.

Όπως υποδηλώνει το κόκκινο χρώμα στις τιμές της μη γραμμικότητας, δεν κατέστη εφικτή η εύρεση συνάρτησης με την επιθυμητή μη γραμμικότητα.

4.1.3 Τρίτο Πρόγραμμα Γενετικού Αλγορίθμου-Αποτελέσματα

Παρακάτω ακολουθούν οι πίνακες με τα αποτελέσματα του προγράμματος. Στους παρακάτω πίνακες όπου υπάρχει T=2 και T=3 εννοείται ότι το tournamentsize για την μέθοδος επιλογής τουρνουά ήταν ίσο με 2 και 3 αντίστοιχα.

Για n=5 θα έχουμε έξοδο 2^n =32 bit

1. Αρχικοποίηση προγράμματος για εκτέλεση για πληθυσμό =500 :

Θέτουμε πληθυσμό ίσο με 500, πιθανότητα μετάλλαξης ίση με 0.3, πιθανότητα διασταύρωσης ίση με 0.99 , elitsm ίσο με true, tournamentsize ίσο με 2 και 3 αντίστοιχα , ενώ ο αλγόριθμος τερματίζει όταν η συνάρτηση καταλληλότητας λάβει τιμή ίση με 12 (που είναι η τιμή της μη γραμμικότητας της Carlet-Feng συνάρτησης για n=5 επαυξημένη κατά 2 μονάδες).

A/A T=2	**GENERATION**	**ΧΡΟΝΟΣ ΕΚΤΕΛΕΣΗΣ (SECONDS)**
1	1	0.02
2	1	0.02
3	1	0.02
4	2	0.03
5	3	0.04
A/A T=3		
1	1	0.03
2	1	0.02
3	1	0.01
4	1	0.01
5	2	0.04
Μέσος όρος T2-T3		0.026-0.022

Πίνακας 4.81. Στατιστικά στοιχεία εξόδων συνάρτησης.

A/A T=2	ΒΑΘΜΟΣ	BALANCED	NONLINEARITY	FAI
1	3	TRUE	12	-
2	4	TRUE	12	-
3	3	TRUE	12	-
4	3	TRUE	12	-
5	3	TRUE	12	-
A/A T=3				
1	3	TRUE	12	-
2	3	TRUE	12	-
3	3	TRUE	12	-
4	4	TRUE	12	-
5	4	TRUE	12	-

Πίνακας 4.82. Κρυπτογραφικές ιδιότητες εξόδων συνάρτησης.

2. Αρχικοποίηση προγράμματος για εκτέλεση για πληθυσμό =1000 :

Θέτουμε πληθυσμό ίσο με 1000, πιθανότητα μετάλλαξης ίση με 0.5, πιθανότητα διασταύρωσης ίση με 0.99 , elitsm ίσο με true, tournamentsize ίσο με 2 και 3 αντίστοιχα , ενώ ο αλγόριθμος τερματίζει όταν η συνάρτηση καταλληλότητας λάβει τιμή ίση με 12 (που είναι η τιμή της μη γραμμικότητας της Carlet-Feng συνάρτησης για n=5 επαυξημένη κατά 2 μονάδες).

A/A T=2	GENERATION	ΧΡΟΝΟΣ ΕΚΤΕΛΕΣΗΣ (SECONDS)
1	1	0.03
2	1	0.02
3	1	0.01
4	1	0.01
5	1	0.01
A/A T=3		
1	1	0.03
2	1	0.02
3	1	0.02
4	1	0.01
5	1	0.01
Μέσος όρος T2-T3		0.016-0.018

Πίνακας 4.83. Στατιστικά στοιχεία εξόδων συνάρτησης.

A/A T=2	ΒΑΘΜΟΣ	BALANCED	NONLINEARITY	FAI
1	3	TRUE	12	-
2	3	TRUE	12	-
3	4	TRUE	12	-
4	4	TRUE	12	-
5	4	TRUE	12	-
A/A T=3				
1	3	TRUE	12	-
2	4	TRUE	12	-
3	4	TRUE	12	-
4	3	TRUE	12	-
5	4	TRUE	12	-

Πίνακας 4.84. Κρυπτογραφικές ιδιότητες εξόδων συνάρτησης.

3. Αρχικοποίηση προγράμματος για εκτέλεση για πληθυσμό=1500 :

Θέτουμε πληθυσμό ίσο με 1500, πιθανότητα μετάλλαξης ίση με 0.7, πιθανότητα διασταύρωσης ίση με 0.99 , elitsm ίσο με true, tournamentsize ίσο με 2 και 3 αντίστοιχα , ενώ ο αλγόριθμος τερματίζει όταν η συνάρτηση καταλληλότητας λάβει τιμή ίση με 12 (που είναι η τιμή της μη γραμμικότητας της Carlet-Feng συνάρτησης για n=5 επαυξημένη κατά 2 μονάδες).

A/A T=2	GENERATION	ΧΡΟΝΟΣ ΕΚΤΕΛΕΣΗΣ (SECONDS)
1	1	0.03
2	1	0.02
3	1	0.02
4	1	0.01
5	1	0.01
A/A T=3		
1	1	0.03
2	1	0.03
3	2	0.07
4	1	0.02
5	1	0.02
Μέσος όρος T2-T3		0.018-0.034

Πίνακας 4.85. Στατιστικά στοιχεία εξόδων συνάρτησης.

A/A T=2	ΒΑΘΜΟΣ	BALANCED	NONLINEARITY	FAI
1	3	TRUE	12	-
2	3	TRUE	12	-
3	4	TRUE	12	-
4	3	TRUE	12	-
5	3	TRUE	12	-
A/A T=3				
1	3	TRUE	12	-
2	3	TRUE	12	-
3	4	TRUE	12	-
4	3	TRUE	12	-
5	3	TRUE	12	-

Πίνακας 4.86. Κρυπτογραφικές ιδιότητες εξόδων συνάρτησης.

4. Αρχικοποίηση προγράμματος για εκτέλεση για πληθυσμό=2000 :

Θέτουμε πληθυσμό ίσο με 2000, πιθανότητα μετάλλαξης ίση με 0.9, πιθανότητα διασταύρωσης ίση με 0.99 , elitsm ίσο με true, tournamentsize ίσο με 2 και 3 αντίστοιχα , ενώ ο αλγόριθμος τερματίζει όταν η συνάρτηση καταλληλότητας λάβει τιμή ίση με 12 (που είναι η τιμή της μη γραμμικότητας της Carlet-Feng συνάρτησης για n=5 επαυξημένη κατά 2 μονάδες).

A/A T=2	GENERATION	ΧΡΟΝΟΣ ΕΚΤΕΛΕΣΗΣ (SECONDS)
1	1	0.07
2	1	0.03
3	1	0.02
4	1	0.03
5	2	0.07
A/A T=3		
1	1	0.06
2	1	0.02
3	1	0.02
4	1	0.02
5	1	0.02
Μέσος όρος T2-T3		0.044-0.028

Πίνακας 4.87. Στατιστικά στοιχεία εξόδων συνάρτησης.

A/A T=2	ΒΑΘΜΟΣ	BALANCED	NONLINEARITY	FAI
1	3	TRUE	12	-
2	3	TRUE	12	-
3	3	TRUE	12	-
4	3	TRUE	12	-
5	4	TRUE	12	-
A/A T=3				
1	3	TRUE	12	-
2	4	TRUE	12	-
3	3	TRUE	12	-
4	3	TRUE	12	-
5	3	TRUE	12	-

Πίνακας 4.88. Κρυπτογραφικές ιδιότητες εξόδων συνάρτησης.

Για n=6 θα έχουμε έξοδο 2^n =64 bit

1. Αρχικοποίηση προγράμματος για εκτέλεση για πληθυσμό =500 :

Θέτουμε πληθυσμό ίσο με 500, πιθανότητα μετάλλαξης ίση με 0.3, πιθανότητα διασταύρωσης ίση με 0.99 , elitsm ίσο με true, tournamentsize ίσο με 2 και 3 αντίστοιχα , ενώ ο αλγόριθμος τερματίζει όταν η συνάρτηση καταλληλότητας λάβει τιμή ίση με 26 (που είναι η τιμή της μη γραμμικότητας της Carlet-Feng συνάρτησης για n=6 επαυξημένη κατά 2 μονάδες).

A/A T=2	GENERATION	ΧΡΟΝΟΣ ΕΚΤΕΛΕΣΗΣ (SECONDS)
1	322	5.9
2	162	2.97
3	716	12.75
4	408	7.27
5	182	3.27
A/A T=3		
1	19	0.52
2	61	1.49
3	19	0.44
4	630	15.05
5	169	4.07
Μέσος όρος T2-T3		6.43-4.31

Πίνακας 4.89. Στατιστικά στοιχεία εξόδων συνάρτησης.

A/A T=2	ΒΑΘΜΟΣ	BALANCED	NONLINEARITY	FAI
1	5	TRUE	26	optimal
2	5	TRUE	26	optimal
3	5	TRUE	26	optimal
4	5	TRUE	26	optimal
5	5	TRUE	26	optimal
A/A T=3				
1	5	TRUE	26	optimal
2	5	TRUE	26	optimal
3	5	TRUE	26	optimal
4	5	TRUE	26	optimal
5	5	TRUE	26	optimal

Πίνακας 4.90. Κρυπτογραφικές ιδιότητες εξόδων συνάρτησης.

2. Αρχικοποίηση προγράμματος για εκτέλεση για πληθυσμό =1000 :

Θέτουμε πληθυσμό ίσο με 1000, πιθανότητα μετάλλαξης ίση με 0.5, πιθανότητα διασταύρωσης ίση με 0.99 , elitsm ίσο με true, tournamentsize ίσο με 2 και 3 αντίστοιχα, ενώ ο αλγόριθμος τερματίζει όταν η συνάρτηση καταλληλότητας λάβει τιμή ίση με 26 (που είναι η τιμή της μη γραμμικότητας της Carlet-Feng συνάρτησης για n=6 επαυξημένη κατά 2 μονάδες).

A/A T=2	GENERATION	ΧΡΟΝΟΣ ΕΚΤΕΛΕΣΗΣ (SECONDS)
1	257	9.9
2	93	3.56
3	113	4.32
4	105	3.96
5	53	2.02
A/A T=3		
1	130	6.61
2	35	1.73
3	211	10.53
4	69	3.45
5	93	4.67
Μέσος όρος T2-T3		4.75-5.4

Πίνακας 4.91. Στατιστικά στοιχεία εξόδων συνάρτησης.

A/A T=2	ΒΑΘΜΟΣ	BALANCED	NONLINEARITY	FAI
1	5	TRUE	26	optimal
2	5	TRUE	26	optimal
3	5	TRUE	26	optimal
4	5	TRUE	26	optimal
5	5	TRUE	26	optimal
A/A T=3				
1	5	TRUE	26	optimal
2	5	TRUE	26	optimal
3	5	TRUE	26	optimal
4	5	TRUE	26	optimal
5	5	TRUE	26	optimal

Πίνακας 4.92. Κρυπτογραφικές ιδιότητες εξόδων συνάρτησης.

3. Αρχικοποίηση προγράμματος για εκτέλεση για πληθυσμό=1500 :

Θέτουμε πληθυσμό ίσο με 1500, πιθανότητα μετάλλαξης ίση με 0.7, πιθανότητα διασταύρωσης ίση με 0.99 , elitsm ίσο με true, tournamentsize ίσο με 2 και 3 αντίστοιχα , ενώ ο αλγόριθμος τερματίζει όταν η συνάρτηση καταλληλότητας λάβει τιμή ίση με 26 (που είναι η τιμή της μη γραμμικότητας της Carlet-Feng συνάρτησης για n=6 επαυξημένη κατά 2 μονάδες).

A/A T=2	GENERATION	ΧΡΟΝΟΣ ΕΚΤΕΛΕΣΗΣ (SECONDS)
1	352	20.43
2	4	0.21
3	65	3.79
4	74	4.36
5	234	13.62
A/A T=3		
1	44	3.39
2	10	0.71
3	90	6.78
4	65	4.87
5	132	10.0
Μέσος όρος T2-T3		8.48-5.15

Πίνακας 4.93. Στατιστικά στοιχεία εξόδων συνάρτησης.

A/A T=2	ΒΑΘΜΟΣ	BALANCED	NONLINEARITY	FAI
1	5	TRUE	26	optimal
2	5	TRUE	26	optimal
3	5	TRUE	26	optimal
4	5	TRUE	26	optimal
5	5	TRUE	26	optimal
A/A T=3				
1	5	TRUE	26	optimal
2	5	TRUE	26	optimal
3	5	TRUE	26	optimal
4	5	TRUE	26	optimal
5	5	TRUE	26	optimal

Πίνακας 4.94. Κρυπτογραφικές ιδιότητες εξόδων συνάρτησης.

4. Αρχικοποίηση προγράμματος για εκτέλεση για πληθυσμό=2000 :

Θέτουμε πληθυσμό ίσο με 2000, πιθανότητα μετάλλαξης ίση με 0.9, πιθανότητα διασταύρωσης ίση με 0.99 , elitsm ίσο με true, tournamentsize ίσο με 2 και 3 αντίστοιχα , ενώ ο αλγόριθμος τερματίζει όταν η συνάρτηση καταλληλότητας λάβει τιμή ίση με 26 (που είναι η τιμή της μη γραμμικότητας της Carlet-Feng συνάρτησης για n=6 επαυξημένη κατά 2 μονάδες).

A/A T=2	GENERATION	ΧΡΟΝΟΣ ΕΚΤΕΛΕΣΗΣ (SECONDS)
1	63	5.32
2	108	8.6
3	22	1.8
4	184	14.53
5	458	36.15
A/A T=3		
1	75	7.59
2	4	0.33
3	66	6.47
4	186	18.8
5	68	6.74
Μέσος όρος T2-T3		13.28-7.99

Πίνακας 4.95. Στατιστικά στοιχεία εξόδων συνάρτησης.

A/A T=2	ΒΑΘΜΟΣ	BALANCED	NONLINEARITY	FAI
1	5	TRUE	26	optimal
2	5	TRUE	26	optimal
3	5	TRUE	26	optimal
4	5	TRUE	26	optimal
5	5	TRUE	26	optimal
A/A T=3				
1	5	TRUE	26	optimal
2	5	TRUE	26	optimal
3	5	TRUE	26	optimal
4	5	TRUE	26	optimal
5	5	TRUE	26	optimal

Πίνακας 4.96. Κρυπτογραφικές ιδιότητες εξόδων συνάρτησης.

Για n=7 θα έχουμε έξοδο 2^n =128 bit

1. Αρχικοποίηση προγράμματος για εκτέλεση για πληθυσμό =500 :

Θέτουμε πληθυσμό ίσο με 500, πιθανότητα μετάλλαξης ίση με 0.3, πιθανότητα διασταύρωσης ίση με 0.99 , elitsm ίσο με true, tournamentsize ίσο με 2 και 3 αντίστοιχα , ενώ ο αλγόριθμος τερματίζει όταν η συνάρτηση καταλληλότητας λάβει τιμή ίση με 54 (που είναι η τιμή της μη γραμμικότητας της Carlet-Feng συνάρτησης για n=7).

A/A T=2	GENERATION	ΧΡΟΝΟΣ ΕΚΤΕΛΕΣΗΣ (SECONDS)
1	334	11.74
2	348	11.66
3	124	4.17
4	523	17.84
5	34	1.15
A/A T=3		
1	38	1.74
2	77	3.36
3	251	10.78
4	158	6.77
5	540	23.2
Μέσος όρος T2-T3		9.31-9.17

Πίνακας 4.97. Στατιστικά στοιχεία εξόδων συνάρτησης.

A/A T=2	ΒΑΘΜΟΣ	BALANCED	NONLINEARITY	FAI
1	6	TRUE	54	optimal
2	6	TRUE	54	optimal
3	6	TRUE	54	optimal
4	6	TRUE	54	optimal
5	6	TRUE	54	optimal
A/A T=3				
1	6	TRUE	54	optimal
2	6	TRUE	54	optimal
3	6	TRUE	54	optimal
4	6	TRUE	54	optimal
5	6	TRUE	54	optimal

Πίνακας 4.98. Κρυπτογραφικές ιδιότητες εξόδων συνάρτησης.

2. Αρχικοποίηση προγράμματος για εκτέλεση για πληθυσμό =1000:

Θέτουμε πληθυσμό ίσο με 1000, πιθανότητα μετάλλαξης ίση με 0.5, πιθανότητα διασταύρωσης ίση με 0.99 , elitsm ίσο με true, tournamentsize ίσο με 2 και 3 αντίστοιχα , ενώ ο αλγόριθμος τερματίζει όταν η συνάρτηση καταλληλότητας λάβει τιμή ίση με 54 (που είναι η τιμή της μη γραμμικότητας της Carlet-Feng συνάρτησης για n=7).

A/A T=2	GENERATION	ΧΡΟΝΟΣ ΕΚΤΕΛΕΣΗΣ (SECONDS)
1	74	5.83
2	41	3.13
3	84	9.38
4	263	20.61
5	116	8.82
A/A T=3		
1	87	7.95
2	31	2.81
3	18	1.6
4	64	5.8
5	120	10.85
Μέσος όρος T2-T3		9.55-5.8

Πίνακας 4.99. Στατιστικά στοιχεία εξόδων συνάρτησης.

A/A T=2	ΒΑΘΜΟΣ	BALANCED	NONLINEARITY	FAI
1	6	TRUE	54	optimal
2	6	TRUE	54	optimal

3	6	TRUE	54	optimal
4	6	TRUE	54	optimal
5	6	TRUE	54	optimal
A/A T=3				
1	6	TRUE	54	optimal
2	6	TRUE	54	optimal
3	6	TRUE	54	optimal
4	6	TRUE	54	optimal
5	6	TRUE	54	optimal

Πίνακας 4.100. Κρυπτογραφικές ιδιότητες εξόδων συνάρτησης.

3. Αρχικοποίηση προγράμματος για εκτέλεση για πληθυσμό=1500 :

Θέτουμε πληθυσμό ίσο με 1500, πιθανότητα μετάλλαξης ίση με 0.7, πιθανότητα διασταύρωσης ίση με 0.99 , elitsm ίσο με true, tournamentsize ίσο με 2 και 3 αντίστοιχα, ενώ ο αλγόριθμος τερματίζει όταν η συνάρτηση καταλληλότητας λάβει τιμή ίση με 54 (που είναι η τιμή της μη γραμμικότητας της Carlet-Feng συνάρτησης για n=7).

A/A T=2	**GENERATION**	**ΧΡΟΝΟΣ ΕΚΤΕΛΕΣΗΣ (SECONDS)**
1	9	1.03
2	234	26.59
3	551	61.63
4	77	8.34
5	131	14.24
A/A T=3		
1	30	4.24
2	63	9.06
3	48	6.78
4	68	9.66
5	38	5.37
Μέσος όρος T2-T3		22.37-7.02

Πίνακας 4.101. Στατιστικά στοιχεία εξόδων συνάρτησης.

A/A T=2	**ΒΑΘΜΟΣ**	**BALANCED**	**NONLINEARITY**	**FAI**
1	6	TRUE	54	optimal
2	6	TRUE	54	optimal
3	6	TRUE	54	optimal
4	6	TRUE	54	optimal
5	6	TRUE	54	optimal

A/A T=3				
1	6	TRUE	54	optimal
2	6	TRUE	54	optimal
3	6	TRUE	54	optimal
4	6	TRUE	54	optimal
5	6	TRUE	54	optimal

Πίνακας 4.102. Κρυπτογραφικές ιδιότητες εξόδων συνάρτησης.

4. Αρχικοποίηση προγράμματος για εκτέλεση για πληθυσμό=2000:

Θέτουμε πληθυσμό ίσο με 2000, πιθανότητα μετάλλαξης ίση με 0.9, πιθανότητα διασταύρωσης ίση με 0.99 , elitsm ίσο με true, tournamentsize ίσο με 2 και 3 αντίστοιχα , ενώ ο αλγόριθμος τερματίζει όταν η συνάρτηση καταλληλότητας λάβει τιμή ίση με 54 (που είναι η τιμή της μη γραμμικότητας της Carlet-Feng συνάρτησης για n=7).

A/A T=2	GENERATION	ΧΡΟΝΟΣ ΕΚΤΕΛΕΣΗΣ (SECONDS)
1	95	13.96
2	56	8.13
3	283	41.36
4	15	2.13
5	28	4.02
A/A T=3		
1	16	3.0
2	77	14.53
3	27	4.99
4	53	9.97
5	278	53.14
Μέσος όρος T2-T3		13.92-17.13

Πίνακας 4.103. Στατιστικά στοιχεία εξόδων συνάρτησης.

A/A T=2	ΒΑΘΜΟΣ	BALANCED	NONLINEARITY	FAI
1	6	TRUE	54	optimal
2	6	TRUE	54	optimal
3	6	TRUE	54	optimal
4	6	TRUE	54	optimal
5	6	TRUE	54	optimal
A/A T=3				
1	6	TRUE	54	optimal
2	6	TRUE	54	optimal

3	6	TRUE	54	optimal
4	6	TRUE	54	optimal
5	6	TRUE	54	optimal

Πίνακας 4.104. Κρυπτογραφικές ιδιότητες εξόδων συνάρτησης.

Για n=8 θα έχουμε έξοδο 2^n =256 bit

1. Αρχικοποίηση προγράμματος για εκτέλεση για πληθυσμό =500 :

Θέτουμε πληθυσμό ίσο με 500, πιθανότητα μετάλλαξης ίση με 0.3, πιθανότητα διασταύρωσης ίση με 0.99 , elitsm ίσο με true, tournamentsize ίσο με 2 και 3 αντίστοιχα , ενώ ο αλγόριθμος τερματίζει όταν η συνάρτηση καταλληλότητας λάβει τιμή ίση με 112 (που είναι η τιμή της μη γραμμικότητας της Carlet-Feng συνάρτησης για n=8).

A/A T=2	**GENERATION**	**ΧΡΟΝΟΣ ΕΚΤΕΛΕΣΗΣ (SECONDS)**
1	18	1.41
2	812	60.76
3	112	8.07
4	236	16.74
5	1628	124.31
A/A T=3		
1	840	78.75
2	146	13.08
3	127	11.2
4	192	17.35
5	673	60.98
Μέσος όρος T2-T3		42.26-36.24

Πίνακας 4.105. Στατιστικά στοιχεία εξόδων συνάρτησης.

A/A T=2	**ΒΑΘΜΟΣ**	**BALANCED**	**NONLINEARITY**	**FAI**
1	6	TRUE	112	optimal
2	7	TRUE	112	optimal
3	7	TRUE	112	optimal
4	7	TRUE	112	optimal
5	7	TRUE	112	optimal
A/A T=3				
1	7	TRUE	112	optimal
2	7	TRUE	112	optimal

3	7	TRUE	112	optimal
4	7	TRUE	112	optimal
5	7	TRUE	112	optimal

Πίνακας 4.106. Κρυπτογραφικές ιδιότητες εξόδων συνάρτησης.

2. Αρχικοποίηση προγράμματος για εκτέλεση για πληθυσμό =1000:

Θέτουμε πληθυσμό ίσο με 1000, πιθανότητα μετάλλαξης ίση με 0.5, πιθανότητα διασταύρωσης ίση με 0.99 , elitsm ίσο με true, tournamentsize ίσο με 2 και 3 αντίστοιχα , ενώ ο αλγόριθμος τερματίζει όταν η συνάρτηση καταλληλότητας λάβει τιμή ίση με 112 (που είναι η τιμή της μη γραμμικότητας της Carlet-Feng συνάρτησης για n=8).

Α/Α T=2	**GENERATION**	**ΧΡΟΝΟΣ ΕΚΤΕΛΕΣΗΣ (SECONDS)**
1	36	5.62
2	104	15.66
3	80	12.44
4	198	31.33
5	176	28.54
Α/Α T=3		
1	42	8.06
2	173	32.23
3	136	25.75
4	47	8.84
5	8	1.44
Μέσος όρος Τ2-Τ3		18.72-15.26

Πίνακας 4.107. Στατιστικά στοιχεία εξόδων συνάρτησης.

Α/Α T=2	**ΒΑΘΜΟΣ**	**BALANCED**	**NONLINEARITY**	**FAI**
1	7	TRUE	112	optimal
2	6	TRUE	112	optimal
3	7	TRUE	112	optimal
4	7	TRUE	112	optimal
5	7	TRUE	112	optimal
Α/Α T=3				
1	7	TRUE	112	optimal
2	7	TRUE	112	optimal
3	7	TRUE	112	optimal
4	6	TRUE	112	optimal

5	6	TRUE	112	optimal

Πίνακας 4.108. Κρυπτογραφικές ιδιότητες εξόδων συνάρτησης.

3. Αρχικοποίηση προγράμματος για εκτέλεση για πληθυσμό=1500:

Θέτουμε πληθυσμό ίσο με 1500, πιθανότητα μετάλλαξης ίση με 0.7, πιθανότητα διασταύρωσης ίση με 0.99 , elitsm ίσο με true, tournamentsize ίσο με 2 και 3 αντίστοιχα , ενώ ο αλγόριθμος τερματίζει όταν η συνάρτηση καταλληλότητας λάβει τιμή ίση με 112 (που είναι η τιμή της μη γραμμικότητας της Carlet-Feng συνάρτησης για n=8).

Α/Α Τ=2	**GENERATION**	**ΧΡΟΝΟΣ ΕΚΤΕΛΕΣΗΣ (SECONDS)**
1	23	5.52
2	1374	321.22
3	619	143.62
4	300	72.77
5	124	29.93
Α/Α Τ=3		
1	72	20.92
2	67	19.35
3	322	94.17
4	51	14.72
5	44	12.69
Μέσος όρος Τ2-Τ3		114.61-32.37

Πίνακας 4.109. Στατιστικά στοιχεία εξόδων συνάρτησης.

Α/Α Τ=2	**ΒΑΘΜΟΣ**	**BALANCED**	**NONLINEARITY**	**FAI**
1	7	TRUE	112	optimal
2	7	TRUE	112	optimal
3	6	TRUE	112	optimal
4	7	TRUE	112	optimal
5	7	TRUE	112	optimal
Α/Α Τ=3				
1	6	TRUE	112	optimal
2	6	TRUE	112	optimal
3	7	TRUE	112	optimal
4	6	TRUE	112	optimal
5	7	TRUE	112	optimal

Πίνακας 4.110. Κρυπτογραφικές ιδιότητες εξόδων συνάρτησης.

4. Αρχικοποίηση προγράμματος για εκτέλεση για πληθυσμό=2000:

Θέτουμε πληθυσμό ίσο με 2000, πιθανότητα μετάλλαξης ίση με 0.9, πιθανότητα διασταύρωσης ίση με 0.99 , elitsm ίσο με true, tournamentsize ίσο με 2 και 3 αντίστοιχα , ενώ ο αλγόριθμος τερματίζει όταν η συνάρτηση καταλληλότητας λάβει τιμή ίση με 112 (που είναι η τιμή της μη γραμμικότητας της Carlet-Feng συνάρτησης για n=8).

A/A T=2	**GENERATION**	**ΧΡΟΝΟΣ ΕΚΤΕΛΕΣΗΣ (SECONDS)**
1	483	142.75
2	713	210.58
3	21	6.07
4	2244	662.07
5	80	23.51
A/A T=3		
1	264	107.4
2	550	228.56
3	210	88.71
4	25	11.59
5	1601	662.2
Μέσος όρος T2-T3		209-219.69

Πίνακας 4.111. Στατιστικά στοιχεία εξόδων συνάρτησης.

A/A T=2	**ΒΑΘΜΟΣ**	**BALANCED**	**NONLINEARITY**	**FAI**
1	7	TRUE	112	optimal
2	7	TRUE	112	optimal
3	6	TRUE	112	optimal
4	7	TRUE	112	optimal
5	7	TRUE	112	optimal
A/A T=3				
1	7	TRUE	112	optimal
2	7	TRUE	112	optimal
3	7	TRUE	112	optimal
4	6	TRUE	112	optimal
5	6	TRUE	112	optimal

Πίνακας 4.112. Κρυπτογραφικές ιδιότητες εξόδων συνάρτησης.

Για n=9 θα έχουμε έξοδο 2^n =512 bit

1. Αρχικοποίηση προγράμματος για εκτέλεση για πληθυσμό =500:

Θέτουμε πληθυσμό ίσο με 500, πιθανότητα μετάλλαξης ίση με 0.3, πιθανότητα διασταύρωσης ίση με 0.99 , elitsm ίσο με true, tournamentsize ίσο με 2 και 3 αντίστοιχα , ενώ ο αλγόριθμος τερματίζει όταν η συνάρτηση καταλληλότητας λάβει τιμή ίση με 232 (που είναι η τιμή της μη γραμμικότητας της Carlet-Feng συνάρτησης για n=9).

A/A T=2	**GENERATION**	**ΧΡΟΝΟΣ ΕΚΤΕΛΕΣΗΣ (SECONDS)**
1	2409	461.22
2	20282	3897.61
3	13137	2542.66
4	6395	1235.85
5	29122	5692.76
A/A T=3		
1	186	35.37
2	1729	327.68
3	1696	319.05
4	2913	548.15
5	658	123.19
Μέσος όρος T2-T3		2766.02-270.69

Πίνακας 4.113. Στατιστικά στοιχεία εξόδων συνάρτησης.

A/A T=2	**ΒΑΘΜΟΣ**	**BALANCED**	**NONLINEARITY**	**FAI**
1	8	TRUE	232	-
2	7	TRUE	232	-
3	7	TRUE	232	-
4	8	TRUE	232	-
5	7	TRUE	232	-
A/A T=3				
1	8	TRUE	232	-
2	8	TRUE	232	-
3	8	TRUE	232	-
4	8	TRUE	232	-
5	8	TRUE	232	-

Πίνακας 4.114. Κρυπτογραφικές ιδιότητες εξόδων συνάρτησης.

2. Αρχικοποίηση προγράμματος για εκτέλεση για πληθυσμό =1000:

Θέτουμε πληθυσμό ίσο με 1000, πιθανότητα μετάλλαξης ίση με 0.5, πιθανότητα διασταύρωσης ίση με 0.99 , elitsm ίσο με true, tournamentsize ίσο με 2 και 3 αντίστοιχα , ενώ ο αλγόριθμος τερματίζει όταν η συνάρτηση καταλληλότητας λάβει τιμή ίση με 232(που είναι η τιμή της μη γραμμικότητας της Carlet-Feng συνάρτησης για n=9).

A/A T=2	**GENERATION**	**ΧΡΟΝΟΣ ΕΚΤΕΛΕΣΗΣ (SECONDS)**
1	422	167.33
2	1021	396.84
3	7945	3133.34
4	170	66.7
5	495	192.77
A/A T=3		
1	312	118.14
2	2143	779.16
3	1941	715.54
4	168	61.48
5	1561	566.22
Μέσος όρος T2-T3		791.4-448.11

Πίνακας 4.115. Στατιστικά στοιχεία εξόδων συνάρτησης.

A/A T=2	**ΒΑΘΜΟΣ**	**BALANCED**	**NONLINEARITY**	**FAI**
1	8	TRUE	232	-
2	8	TRUE	232	-
3	8	TRUE	232	-
4	8	TRUE	232	-
5	8	TRUE	232	-
A/A T=3				
1	8	TRUE	232	-
2	8	TRUE	232	-
3	8	TRUE	232	-
4	7	TRUE	232	-
5	8	TRUE	232	-

Πίνακας 4.116. Κρυπτογραφικές ιδιότητες εξόδων συνάρτησης.

3. Αρχικοποίηση προγράμματος για εκτέλεση για πληθυσμό=1500:

Θέτουμε πληθυσμό ίσο με 1500, πιθανότητα μετάλλαξης ίση με 0.7, πιθανότητα διασταύρωσης ίση με 0.99 , elitsm ίσο με true, tournamentsize ίσο με 2 και 3 αντίστοιχα , ενώ ο αλγόριθμος τερματίζει όταν η συνάρτηση καταλληλότητας λάβει τιμή ίση με 232(που είναι η τιμή της μη γραμμικότητας της Carlet-Feng συνάρτησης για n=9).

Α/Α Τ=2	**GENERATION**	**ΧΡΟΝΟΣ ΕΚΤΕΛΕΣΗΣ (SECONDS)**
1	295752	172800.78
2	290784	172800.36
3	292920	172800.48
4	295584	172800.73
5	293592	172800.28
Α/Α Τ=3		
1	2464	1418.97
2	1702	966.24
3	2182	1233.8
4	3265	1845.2
5	1287	730.22
Μέσος όρος Τ2-Τ3		172800.5-1238.89

Πίνακας 4.117. Στατιστικά στοιχεία εξόδων συνάρτησης.

Α/Α Τ=2	**ΒΑΘΜΟΣ**	**BALANCED**	**NONLINEARITY**	**FAI**
1	8	TRUE	230	-
2	8	TRUE	230	-
3	8	TRUE	230	-
4	8	TRUE	230	-
5	8	TRUE	230	-
Α/Α Τ=3				
1	8	TRUE	232	-
2	8	TRUE	232	-
3	8	TRUE	232	-
4	8	TRUE	232	-
5	8	TRUE	232	-

Πίνακας 4.118. Κρυπτογραφικές ιδιότητες εξόδων συνάρτησης.

Όπως υποδηλώνει το κόκκινο χρώμα στις τιμές της μη γραμμικότητας, δεν κατέστη εφικτή για Τ=2 η εύρεση συνάρτησης με την επιθυμητή μη γραμμικότητα.

4. Αρχικοποίηση προγράμματος για εκτέλεση για πληθυσμό=2000:

Θέτουμε πληθυσμό ίσο με 2000, πιθανότητα μετάλλαξης ίση με 0.9, πιθανότητα διασταύρωσης ίση με 0.99 , elitsm ίσο με true, tournamentsize ίσο με 2 και 3 αντίστοιχα , ενώ ο αλγόριθμος τερματίζει όταν η συνάρτηση καταλληλότητας λάβει τιμή ίση με 232(που είναι η τιμή της μη γραμμικότητας της Carlet-Feng συνάρτησης για n=9).

A/A T=2	**GENERATION**	**ΧΡΟΝΟΣ ΕΚΤΕΛΕΣΗΣ (SECONDS)**
1	220344	172800.39
2	221064	172800.78
3	219720	172800.93
4	219240	172800.09
5	219888	172800.94
A/A T=3		
1	583	478.05
2	4314	3423.55
3	3994	3088.31
4	8711	7121.33
5	6334	5205.58
Μέσος όρος T2-T3		172800.6-3863.36

Πίνακας 4.119. Στατιστικά στοιχεία εξόδων συνάρτησης.

A/A T=2	**ΒΑΘΜΟΣ**	**BALANCED**	**NONLINEARITY**	**FAI**
1	8	TRUE	230	-
2	8	TRUE	230	-
3	8	TRUE	230	-
4	8	TRUE	230	-
5	8	TRUE	230	-
A/A T=3				
1	8	TRUE	232	-
2	8	TRUE	232	-
3	8	TRUE	232	-
4	8	TRUE	232	-
5	8	TRUE	232	-

Πίνακας 4.120. Κρυπτογραφικές ιδιότητες εξόδων συνάρτησης.

Όπως υποδηλώνει το κόκκινο χρώμα στις τιμές της μη γραμμικότητας, δεν κατέστη εφικτή για T=2 η εύρεση συνάρτησης με την επιθυμητή μη γραμμικότητα.

4.1.4 Τέταρτο Πρόγραμμα Γενετικού Αλγορίθμου-Αποτελέσματα

Παρακάτω ακολουθούν οι πίνακες με τα αποτελέσματα του προγράμματος. Στους παρακάτω πίνακες όπου υπάρχει T=2 και T=3 εννοείται ότι το tournamentsize για την μέθοδος επιλογής τουρνουά ήταν ίσο με 2 και 3 αντίστοιχα. Όπου η μη γραμμικότητα είναι με κόκκινο χρώμα στους παρακάτω πίνακες, είναι ελάχιστα μικρότερη της αναμενόμενης τιμής.

Για n=5 θα έχουμε έξοδο 2^n =32 bit

1. Αρχικοποίηση προγράμματος για εκτέλεση για πληθυσμό =500:

Θέτουμε πληθυσμό ίσο με 500, πιθανότητα μετάλλαξης ίση με 0.3, πιθανότητα διασταύρωσης ίση με 0.99 , elitsm ίσο με true, tournamentsize ίσο με 2 και 3 αντίστοιχα, ενώ ο αλγόριθμος τερματίζει όταν η συνάρτηση καταλληλότητας λάβει τιμή ίση με 12 (που είναι η τιμή της μη γραμμικότητας της Carlet-Feng συνάρτησης για n=5 επαυξημένη κατά 2 μονάδες).

A/A T=2	**GENERATION**	**ΧΡΟΝΟΣ ΕΚΤΕΛΕΣΗΣ (SECONDS)**
1	1	0.02
2	1	0.01
3	3	0.04
4	3	0.03
5	1	0.01
A/A T=3		
1	3	0.06
2	2	0.02
3	3	0.03
4	1	0.01
5	1	0.01
Μέσος όρος T2-T3		0.022-0.026

Πίνακας 4.121. Στατιστικά στοιχεία εξόδων συνάρτησης.

A/A T=2	**ΒΑΘΜΟΣ**	**BALANCED**	**NONLINEARITY**	**FAI**
1	3	TRUE	12	-
2	3	TRUE	12	-
3	3	TRUE	12	-

4	3	TRUE	12	-
5	3	TRUE	12	-
A/A T=3				
1	3	TRUE	12	-
2	3	TRUE	12	-
3	3	TRUE	12	-
4	3	TRUE	12	-
5	4	TRUE	12	-

Πίνακας 4.122. Κρυπτογραφικές ιδιότητες εξόδων συνάρτησης.

2. Αρχικοποίηση προγράμματος για εκτέλεση για πληθυσμό =1000:

Θέτουμε πληθυσμό ίσο με 1000, πιθανότητα μετάλλαξης ίση με 0.5, πιθανότητα διασταύρωσης ίση με 0.99 , elitsm ίσο με true, tournamentsize ίσο με 2 και 3 αντίστοιχα , ενώ ο αλγόριθμος τερματίζει όταν η συνάρτηση καταλληλότητας λάβει τιμή ίση με 12 (που είναι η τιμή της μη γραμμικότητας της Carlet-Feng συνάρτησης για n=5 επαυξημένη κατά 2 μονάδες).

A/A T=2	**GENERATION**	**ΧΡΟΝΟΣ ΕΚΤΕΛΕΣΗΣ (SECONDS)**
1	1	0.05
2	1	0.02
3	1	0.01
4	2	0.05
5	1	0.02
A/A T=3		
1	1	0.03
2	2	0.06
3	2	0.04
4	1	0.01
5	1	0.01
Μέσος όρος T2-T3		0.03-0.03

Πίνακας 4.123. Στατιστικά στοιχεία εξόδων συνάρτησης.

A/A T=2	**ΒΑΘΜΟΣ**	**BALANCED**	**NONLINEARITY**	**FAI**
1	3	TRUE	12	-
2	4	TRUE	12	-
3	3	TRUE	12	-
4	3	TRUE	12	-

5	3	TRUE	12	-
A/A T=3				
1	4	TRUE	12	-
2	3	TRUE	12	-
3	3	TRUE	12	-
4	3	TRUE	12	-
5	3	TRUE	12	-

Πίνακας 4.124. Κρυπτογραφικές ιδιότητες εξόδων συνάρτησης.

3. Αρχικοποίηση προγράμματος για εκτέλεση για πληθυσμό=1500 :

Θέτουμε πληθυσμό ίσο με 1500, πιθανότητα μετάλλαξης ίση με 0.7, πιθανότητα διασταύρωσης ίση με 0.99 , elitsm ίσο με true, tournamentsize ίσο με 2 και 3 αντίστοιχα , ενώ ο αλγόριθμος τερματίζει όταν η συνάρτηση καταλληλότητας λάβει τιμή ίση με 12 (που είναι η τιμή της μη γραμμικότητας της Carlet-Feng συνάρτησης για n=5 αυξημένης κατά 2 μονάδες).

A/A T=2	**GENERATION**	**ΧΡΟΝΟΣ ΕΚΤΕΛΕΣΗΣ (SECONDS)**
1	1	0.03
2	1	0.03
3	1	0.02
4	1	0.02
5	1	0.02
A/A T=3		
1	1	0.04
2	1	0.02
3	1	0.02
4	1	0.02
5	1	0.01
Μέσος όρος T2-T3		0.024-0.022

Πίνακας 4.125. Στατιστικά στοιχεία εξόδων συνάρτησης.

A/A T=2	**ΒΑΘΜΟΣ**	**BALANCED**	**NONLINEARITY**	**FAI**
1	4	TRUE	12	-
2	3	TRUE	12	-

3	3	TRUE	12	-
4	3	TRUE	12	-
5	4	TRUE	12	-
A/A T=3				
1	3	TRUE	12	-
2	3	TRUE	12	-
3	3	TRUE	12	-
4	3	TRUE	12	-
5	3	TRUE	12	-

Πίνακας 4.126. Κρυπτογραφικές ιδιότητες εξόδων συνάρτησης.

4. Αρχικοποίηση προγράμματος για εκτέλεση για πληθυσμό=2000 :

Θέτουμε πληθυσμό ίσο με 500, πιθανότητα μετάλλαξης ίση με 0.3, πιθανότητα διασταύρωσης ίση με 0.99 , elitsm ίσο με true, tournamentsize ίσο με 2 και 3 αντίστοιχα , ενώ ο αλγόριθμος τερματίζει όταν η συνάρτηση καταλληλότητας λάβει τιμή ίση με 12 (που είναι η τιμή της μη γραμμικότητας της Carlet-Feng συνάρτησης για n=5 επαυξημένη κατά 2 μονάδες).

A/A T=2	**GENERATION**	**ΧΡΟΝΟΣ ΕΚΤΕΛΕΣΗΣ (SECONDS)**
1	1	0.04
2	1	0.03
3	1	0.02
4	1	0.02
5	1	0.02
A/A T=3		
1	1	0.05
2	1	0.02
3	1	0.02
4	1	0.02
5	1	0.02
Μέσος όρος T2-T3		0.026-0.028

Πίνακας 4.127. Στατιστικά στοιχεία εξόδων συνάρτησης.

A/A T=2	**ΒΑΘΜΟΣ**	**BALANCED**	**NONLINEARITY**	**FAI**
1	3	TRUE	12	-
2	4	TRUE	12	-

3	4	TRUE	12	-
4	3	TRUE	12	-
5	3	TRUE	12	-
A/A T=3				
1	3	TRUE	12	-
2	3	TRUE	12	-
3	3	TRUE	12	-
4	3	TRUE	12	-
5	3	TRUE	12	-

Πίνακας 4.128. Κρυπτογραφικές ιδιότητες εξόδων συνάρτησης.

Για n=6 θα έχουμε έξοδο 2^n =64 bit

1. Αρχικοποίηση προγράμματος για εκτέλεση για πληθυσμό =500:

Θέτουμε πληθυσμό ίσο με 500, πιθανότητα μετάλλαξης ίση με 0.3, πιθανότητα διασταύρωσης ίση με 0.99 , elitsm ίσο με true, tournamentsize ίσο με 2 και 3 αντίστοιχα , ενώ ο αλγόριθμος τερματίζει όταν η συνάρτηση καταλληλότητας λάβει τιμή ίση με 26 (που είναι η τιμή της μη γραμμικότητας της Carlet-Feng συνάρτησης για n=6 επαυξημένη κατά 2 μονάδες).

A/A T=2	**GENERATION**	**ΧΡΟΝΟΣ ΕΚΤΕΛΕΣΗΣ (SECONDS)**
1	52	0.99
2	30	0.54
3	19	0.4
4	56	1.08
5	26	0.46
A/A T=3		
1	22	0.62
2	33	0.81
3	47	1.18
4	15	0.36
5	10	0.23
Μέσος όρος T2-T3		0.69-0.64

Πίνακας 4.129. Στατιστικά στοιχεία εξόδων συνάρτησης.

A/A T=2	**ΒΑΘΜΟΣ**	**BALANCED**	**NONLINEARITY**	**FAI**
1	5	TRUE	26	optimal
2	5	TRUE	26	optimal

3	5	TRUE	26	-
4	5	TRUE	26	-
5	5	TRUE	26	optimal
A/A T=3				
1	5	TRUE	26	optimal
2	5	TRUE	26	optimal
3	5	TRUE	26	optimal
4	5	TRUE	26	optimal
5	5	TRUE	26	optimal

Πίνακας 4.130. Κρυπτογραφικές ιδιότητες εξόδων συνάρτησης.

2. Αρχικοποίηση προγράμματος για εκτέλεση για πληθυσμό =1000:

Θέτουμε πληθυσμό ίσο με 1000, πιθανότητα μετάλλαξης ίση με 0.5, πιθανότητα διασταύρωσης ίση με 0.99 , elitsm ίσο με true, tournamentsize ίσο με 2 και 3 αντίστοιχα , ενώ ο αλγόριθμος τερματίζει όταν η συνάρτηση καταλληλότητας λάβει τιμή ίση με 26 (που είναι η τιμή της μη γραμμικότητας της Carlet-Feng συνάρτησης για n=6 επαυξημένη κατά 2 μονάδες).

A/A T=2	**GENERATION**	**ΧΡΟΝΟΣ ΕΚΤΕΛΕΣΗΣ (SECONDS)**
1	38	1.5
2	11	0.4
3	22	0.86
4	24	0.92
5	13	0.47
A/A T=3		
1	19	1.02
2	18	0.87
3	7	0.34
4	20	0.99
5	14	0.69
Μέσος όρος T2-T3		0.83-0.78

Πίνακας 4.131. Στατιστικά στοιχεία εξόδων συνάρτησης.

A/A T=2	**ΒΑΘΜΟΣ**	**BALANCED**	**NONLINEARITY**	**FAI**
1	5	TRUE	26	optimal
2	5	TRUE	26	optimal
3	5	TRUE	26	optimal

4	5	TRUE	26	optimal
5	5	TRUE	26	optimal
A/A T=3				
1	5	TRUE	26	optimal
2	5	TRUE	26	optimal
3	5	TRUE	26	optimal
4	5	TRUE	26	optimal
5	5	TRUE	26	optimal

Πίνακας 4.132. Κρυπτογραφικές ιδιότητες εξόδων συνάρτησης.

3. Αρχικοποίηση προγράμματος για εκτέλεση για πληθυσμό=1500:

Θέτουμε πληθυσμό ίσο με 1500, πιθανότητα μετάλλαξης ίση με 0.7, πιθανότητα διασταύρωσης ίση με 0.99 , elitsm ίσο με true, tournamentsize ίσο με 2 και 3 αντίστοιχα , ενώ ο αλγόριθμος τερματίζει όταν η συνάρτηση καταλληλότητας λάβει τιμή ίση με 26 (που είναι η τιμή της μη γραμμικότητας της Carlet-Feng συνάρτησης για n=6 επαυξημένη κατά 2 μονάδες).

A/A T=2	**GENERATION**	**ΧΡΟΝΟΣ ΕΚΤΕΛΕΣΗΣ (SECONDS)**
1	18	1.12
2	10	0.57
3	19	1.13
4	33	1.99
5	4	0.21
A/A T=3		
1	18	1.42
2	6	0.41
3	5	0.33
4	15	1.14
5	6	0.41
Μέσος όρος T2--T3		1.00-0.74

Πίνακας 4.133. Στατιστικά στοιχεία εξόδων συνάρτησης.

A/A T=2	**ΒΑΘΜΟΣ**	**BALANCED**	**NONLINEARITY**	**FAI**
1	5	TRUE	26	optimal
2	5	TRUE	26	optimal
3	5	TRUE	26	optimal
4	5	TRUE	26	optimal
5	5	TRUE	26	optimal

A/A T=3				
1	5	TRUE	26	optimal
2	5	TRUE	26	optimal
3	5	TRUE	26	optimal
4	5	TRUE	26	optimal
5	5	TRUE	26	optimal

Πίνακας 4.134. Κρυπτογραφικές ιδιότητες εξόδων συνάρτησης.

4. Αρχικοποίηση προγράμματος για εκτέλεση για πληθυσμό=2000 :

Θέτουμε πληθυσμό ίσο με 2000, πιθανότητα μετάλλαξης ίση με 0.9, πιθανότητα διασταύρωσης ίση με 0.99 , elitsm ίσο με true, tournamentsize ίσο με 2 και 3 αντίστοιχα, ενώ ο αλγόριθμος τερματίζει όταν η συνάρτηση καταλληλότητας λάβει τιμή ίση με 26 (που είναι η τιμή της μη γραμμικότητας της Carlet-Feng συνάρτησης για n=6 επαυξημένη κατά 2 μονάδες).

A/A T=2	GENERATION	ΧΡΟΝΟΣ ΕΚΤΕΛΕΣΗΣ (SECONDS)
1	17	1.4
2	25	2.0
3	16	1.26
4	20	1.57
5	20	1.56
A/A T=3		
1	8	0.84
2	7	0.66
3	10	1.04
4	8	0.79
5	6	0.56
Μέσος όρος T2-T3		1.56-0.78

Πίνακας 4.135. Στατιστικά στοιχεία εξόδων συνάρτησης.

A/A T=2	ΒΑΘΜΟΣ	BALANCED	NONLINEARITY	FAI
1	5	TRUE	26	optimal
2	5	TRUE	26	optimal
3	5	TRUE	26	optimal
4	5	TRUE	26	optimal
5	5	TRUE	26	optimal

A/A T=3				
1	5	TRUE	26	optimal
2	5	TRUE	26	optimal
3	5	TRUE	26	optimal
4	5	TRUE	26	optimal
5	5	TRUE	26	optimal

Πίνακας 4.136. Κρυπτογραφικές ιδιότητες εξόδων συνάρτησης.

Για n=7 θα έχουμε έξοδο 2^n =128 bit

1. Αρχικοποίηση προγράμματος για εκτέλεση για πληθυσμό =500 :

Θέτουμε πληθυσμό ίσο με 500, πιθανότητα μετάλλαξης ίση με 0.3, πιθανότητα διασταύρωσης ίση με 0.99 , elitsm ίσο με true, tournamentsize ίσο με 2 και 3 αντίστοιχα, ενώ ο αλγόριθμος τερματίζει όταν η συνάρτηση καταλληλότητας λάβει τιμή ίση με 56 (που είναι η τιμή της μη γραμμικότητας της Carlet-Feng συνάρτησης για n=7 επαυξημένη κατά 2 μονάδες).

A/A T=2	GENERATION	ΧΡΟΝΟΣ ΕΚΤΕΛΕΣΗΣ (SECONDS)
1	4880	158.63
2	6942	225.89
3	13331	440.48
4	1073	34.97
5	534	17.39
A/A T=3		
1	5035	207.47
2	10670	440.85
3	16184	667.75
4	2279	94.24
5	2875	118.47
Μέσος όρος T2-T3		175.47-305.76

Πίνακας 4.137. Στατιστικά στοιχεία εξόδων συνάρτησης.

A/A T=2	ΒΑΘΜΟΣ	BALANCED	NONLINEARITY	FAI
1	5	TRUE	56	optimal
2	5	TRUE	56	optimal
3	5	TRUE	56	optimal
4	5	TRUE	56	optimal
5	5	TRUE	56	optimal

A/A T=3				
1	5	TRUE	56	optimal
2	5	TRUE	56	optimal
3	5	TRUE	56	optimal
4	5	TRUE	56	optimal
5	5	TRUE	56	optimal

Πίνακας 4.138. Κρυπτογραφικές ιδιότητες εξόδων συνάρτησης.

2. Αρχικοποίηση προγράμματος για εκτέλεση για πληθυσμό =1000:

Θέτουμε πληθυσμό ίσο με 1000, πιθανότητα μετάλλαξης ίση με 0.5, πιθανότητα διασταύρωσης ίση με 0.99 , elitsm ίσο με true, tournamentsize ίσο με 2 και 3, ενώ ο αλγόριθμος τερματίζει όταν η συνάρτηση καταλληλότητας λάβει τιμή ίση με 56 (που είναι η τιμή της μη γραμμικότητας της Carlet-Feng συνάρτησης για n=7 επαυξημένη κατά 2 μονάδες).

A/A T=2	**GENERATION**	**ΧΡΟΝΟΣ ΕΚΤΕΛΕΣΗΣ (SECONDS)**
1	315	22.95
2	70	5.21
3	144	10.36
4	199	14.31
5	160	11.47
A/A T=3		
1	86	7.9
2	365	31.19
3	160	13.78
4	1553	139.36
5	2615	223.2
Μέσος όρος T2-T3		12.86-83.09

Πίνακας 4.139. Στατιστικά στοιχεία εξόδων συνάρτησης.

A/A T=2	**ΒΑΘΜΟΣ**	**BALANCED**	**NONLINEARITY**	**FAI**
1	5	TRUE	56	optimal
2	5	TRUE	56	optimal
3	5	TRUE	56	optimal
4	5	TRUE	56	optimal
5	5	TRUE	56	optimal
A/A T=3				

1	5	TRUE	56	optimal
2	5	TRUE	56	optimal
3	5	TRUE	56	optimal
4	5	TRUE	56	optimal
5	5	TRUE	56	optimal

Πίνακας 4.140. Κρυπτογραφικές ιδιότητες εξόδων συνάρτησης.

3. Αρχικοποίηση προγράμματος για εκτέλεση για πληθυσμό=1500:

Θέτουμε πληθυσμό ίσο με 1500, πιθανότητα μετάλλαξης ίση με 0.7, πιθανότητα διασταύρωσης ίση με 0.99 , elitsm ίσο με true, tournamentsize ίσο με 2 και 3 αντίστοιχα , ενώ ο αλγόριθμος τερματίζει όταν η συνάρτηση καταλληλότητας λάβει τιμή ίση με 56 (που είναι η τιμή της μη γραμμικότητας της Carlet-Feng συνάρτησης για n=7 επαυξημένη κατά 2 μονάδες).

Α/Α Τ=2	**GENERATION**	**ΧΡΟΝΟΣ ΕΚΤΕΛΕΣΗΣ (SECONDS)**
1	1702	194.0
2	3949	445.7
3	2707	305.65
4	2842	349.53
5	3183	343.57
Α/Α Τ=3		
1	26	3.89
2	41	5.82
3	49	7.04
4	88	13.51
5	143	20.54
Μέσος όρος Τ2-Τ3		327.69-10.16

Πίνακας 4.141. Στατιστικά στοιχεία εξόδων συνάρτησης.

Α/Α Τ=2	**ΒΑΘΜΟΣ**	**BALANCED**	**NONLINEARITY**	**FAI**
1	5	TRUE	56	optimal
2	5	TRUE	56	optimal
3	5	TRUE	56	optimal
4	5	TRUE	56	optimal
5	5	TRUE	56	optimal
Α/Α				

T=3				
1	5	TRUE	56	optimal
2	5	TRUE	56	optimal
3	5	TRUE	56	optimal
4	5	TRUE	56	optimal
5	5	TRUE	56	optimal

Πίνακας 4.142. Κρυπτογραφικές ιδιότητες εξόδων συνάρτησης.

4. Αρχικοποίηση προγράμματος για εκτέλεση για πληθυσμό=2000:

Θέτουμε πληθυσμό ίσο με 2000, πιθανότητα μετάλλαξης ίση με 0.9, πιθανότητα διασταύρωσης ίση με 0.99 , elitsm ίσο με true, tournamentsize ίσο με 2 και 3 αντίστοιχα , ενώ ο αλγόριθμος τερματίζει όταν η συνάρτηση καταλληλότητας λάβει τιμή ίση με 56 (που είναι η τιμή της μη γραμμικότητας της Carlet-Feng συνάρτησης για n=7 επαυξημένη κατά 2 μονάδες).

Α/Α T=2	GENERATION	ΧΡΟΝΟΣ ΕΚΤΕΛΕΣΗΣ (SECONDS)
1	9008	1368.79
2	929	141.02
3	888	133.76
4	11167	1687.07
5	4697	708.95
Α/Α T=3		
1	2507	469.31
2	697	130.58
3	114	21.28
4	846	157.9
5	338	63.54
Μέσος όρος T2-T3		807.92-168.52

Πίνακας 4.143. Στατιστικά στοιχεία εξόδων συνάρτησης.

Α/Α T=2	ΒΑΘΜΟΣ	BALANCED	NONLINEARITY	FAI
1	5	TRUE	56	optimal
2	5	TRUE	56	optimal
3	5	TRUE	56	optimal
4	5	TRUE	56	optimal
5	5	TRUE	56	optimal
Α/Α T=3				
1	5	TRUE	56	optimal

2	5	TRUE	56	optimal
3	5	TRUE	56	optimal
4	5	TRUE	56	optimal
5	5	TRUE	56	optimal

Πίνακας 4.144. Κρυπτογραφικές ιδιότητες εξόδων συνάρτησης.

Για n=8 θα έχουμε έξοδο 2^n =256 bit

1. Αρχικοποίηση προγράμματος για εκτέλεση για πληθυσμό =500:

Θέτουμε πληθυσμό ίσο με 500, πιθανότητα μετάλλαξης ίση με 0.3, πιθανότητα διασταύρωσης ίση με 0.99 , elitsm ίσο με true, tournamentsize ίσο με 2 και 3 αντίστοιχα , ενώ ο αλγόριθμος τερματίζει όταν η συνάρτηση καταλληλότητας λάβει τιμή ίση με 114 (που είναι η τιμή της μη γραμμικότητας της Carlet-Feng συνάρτησης για n=8 επαυξημένη κατά 2 μονάδες).

Α/Α Τ=2	**GENERATION**	**ΧΡΟΝΟΣ ΕΚΤΕΛΕΣΗΣ (SECONDS)**
1	618	42.3
2	945	64.46
3	420	28.31
4	6815	453.52
5	381	25.54
Α/Α Τ=3		
1	3521	300.39
2	1814	154.49
3	1241	109.57
4	1557	133.05
5	710	62.1
Μέσος όρος Τ2-Τ3		122.83-151.92

Πίνακας 4.145. Στατιστικά στοιχεία εξόδων συνάρτησης.

Α/Α Τ=2	**ΒΑΘΜΟΣ**	**BALANCED**	**NONLINEARITY**	**FAI**
1	7	TRUE	114	optimal
2	7	TRUE	114	optimal
3	7	TRUE	114	optimal
4	7	TRUE	114	optimal
5	7	TRUE	114	optimal
Α/Α				

T=3				
1	7	TRUE	114	optimal
2	7	TRUE	114	optimal
3	7	TRUE	114	optimal
4	7	TRUE	114	optimal
5	7	TRUE	114	optimal

Πίνακας 4.146. Κρυπτογραφικές ιδιότητες εξόδων συνάρτησης.

2. Αρχικοποίηση προγράμματος για εκτέλεση για πληθυσμό =1000:

Θέτουμε πληθυσμό ίσο με 1000, πιθανότητα μετάλλαξης ίση με 0.5, πιθανότητα διασταύρωσης ίση με 0.99 , elitsm ίσο με true, tournamentsize ίσο με 2 και 3 αντίστοιχα , ενώ ο αλγόριθμος τερματίζει όταν η συνάρτηση καταλληλότητας λάβει τιμή ίση με 114 (που είναι η τιμή της μη γραμμικότητας της Carlet-Feng συνάρτησης για n=8 επαυξημένη κατά 2 μονάδες).

A/A T=2	GENERATION	ΧΡΟΝΟΣ ΕΚΤΕΛΕΣΗΣ (SECONDS)
1	287	40.91
2	86	12.14
3	151	21.43
4	170	24.02
5	250	35.16
A/A T=3		
1	144	25.65
2	303	52.57
3	293	51.5
4	2876	493.37
5	418	73.32
Μέσος όρος T2-T3		26.73-139.28

Πίνακας 4.147. Στατιστικά στοιχεία εξόδων συνάρτησης.

A/A T=2	ΒΑΘΜΟΣ	BALANCED	NONLINEARITY	FAI
1	7	TRUE	114	optimal
2	7	TRUE	114	optimal
3	7	TRUE	114	optimal
4	7	TRUE	114	optimal
5	7	TRUE	114	optimal
A/A T=3				

1	7	TRUE	114	optimal
2	7	TRUE	114	optimal
3	7	TRUE	114	optimal
4	7	TRUE	114	optimal
5	7	TRUE	114	optimal

Πίνακας 4.148. Κρυπτογραφικές ιδιότητες εξόδων συνάρτησης.

3. Αρχικοποίηση προγράμματος για εκτέλεση για πληθυσμό=1500 :

Θέτουμε πληθυσμό ίσο με 1500, πιθανότητα μετάλλαξης ίση με 0.7, πιθανότητα διασταύρωσης ίση με 0.99 , elitsm ίσο με true, tournamentsize ίσο με 2 και 3 αντίστοιχα , ενώ ο αλγόριθμος τερματίζει όταν η συνάρτηση καταλληλότητας λάβει τιμή ίση με 114 (που είναι η τιμή της μη γραμμικότητας της Carlet-Feng συνάρτησης για n=8 επαυξημένη κατά 2 μονάδες).

Α/Α Τ=2	**GENERATION**	**ΧΡΟΝΟΣ ΕΚΤΕΛΕΣΗΣ (SECONDS)**
1	245	53.59
2	405	88.63
3	352	76.93
4	68	14.84
5	554	124.83
Α/Α Τ=3		
1	59	16.13
2	78	21.26
3	77	21.19
4	66	17.96
5	85	23.39
Μέσος όρος Τ2-Τ3		71.76-19.99

Πίνακας 4.149. Στατιστικά στοιχεία εξόδων συνάρτησης.

Α/Α Τ=2	**ΒΑΘΜΟΣ**	**BALANCED**	**NONLINEARITY**	**FAI**
1	7	TRUE	114	optimal
2	7	TRUE	114	optimal
3	7	TRUE	114	optimal
4	7	TRUE	114	optimal
5	7	TRUE	114	optimal
Α/Α Τ=3				
1	7	TRUE	114	optimal
2	7	TRUE	114	optimal

3	7	TRUE	114	optimal
4	7	TRUE	114	optimal
5	7	TRUE	114	optimal

Πίνακας 4.150. Κρυπτογραφικές ιδιότητες εξόδων συνάρτησης.

4. Αρχικοποίηση προγράμματος για εκτέλεση για πληθυσμό=2000:

Θέτουμε πληθυσμό ίσο με 2000, πιθανότητα μετάλλαξης ίση με 0.9, πιθανότητα διασταύρωσης ίση με 0.99 , elitsm ίσο με true, tournamentsize ίσο με 2 και 3 αντίστοιχα , ενώ ο αλγόριθμος τερματίζει όταν η συνάρτηση καταλληλότητας λάβει τιμή ίση με 114 (που είναι η τιμή της μη γραμμικότητας της Carlet-Feng συνάρτησης για n=8 επαυξημένη κατά 2 μονάδες).

Α/Α T=2	**GENERATION**	**ΧΡΟΝΟΣ ΕΚΤΕΛΕΣΗΣ (SECONDS)**
1	6137	1803.2
2	6212	1826.58
3	2837	831.83
4	2842	833.51
5	4333	1268.37
Α/Α T=3		
1	141	55.91
2	258	101.83
3	73	28.69
4	70	27.64
5	77	30.05
Μέσος όρος T2-T3		1145.93-48.82

Πίνακας 4.151. Στατιστικά στοιχεία εξόδων συνάρτησης.

Α/Α T=2	**ΒΑΘΜΟΣ**	**BALANCED**	**NONLINEARITY**	**FAI**
1	7	TRUE	114	optimal
2	7	TRUE	114	optimal
3	7	TRUE	114	optimal
4	7	TRUE	114	optimal
5	7	TRUE	114	optimal
Α/Α T=3				
1	7	TRUE	114	optimal
2	7	TRUE	114	optimal
3	7	TRUE	114	optimal
4	7	TRUE	114	optimal

5	7	TRUE	114	optimal

Πίνακας 4.152. Κρυπτογραφικές ιδιότητες εξόδων συνάρτησης.

Για n=9 θα έχουμε έξοδο 2^n =512 bit

1. Αρχικοποίηση προγράμματος για εκτέλεση για πληθυσμό =500 :

Θέτουμε πληθυσμό ίσο με 500, πιθανότητα μετάλλαξης ίση με 0.3, πιθανότητα διασταύρωσης ίση με 0.99 , elitsm ίσο με true, tournamentsize ίσο με 2 και 3 αντίστοιχα , ενώ ο αλγόριθμος τερματίζει όταν η συνάρτηση καταλληλότητας λάβει τιμή ίση με 234(που είναι η τιμή της μη γραμμικότητας της Carlet-Feng συνάρτησης για n=9 επαυξημένη κατά 2 μονάδες).

A/A T=2	**GENERATION**	**ΧΡΟΝΟΣ ΕΚΤΕΛΕΣΗΣ (SECONDS)**
1	5981	888.52
2	4985	728.7
3	741	107.57
4	249	36.05
5	8684	1239.89
A/A T=3		
1	2255	416.36
2	10790	1966.23
3	1296	239.16
4	2848	521.79
5	4224	772.72
Μέσος όρος T2-T3		600.15-783.25

Πίνακας 4.153. Στατιστικά στοιχεία εξόδων συνάρτησης.

A/A T=2	**ΒΑΘΜΟΣ**	**BALANCED**	**NONLINEARITY**	**FAI**
1	8	TRUE	234	optimal
2	8	TRUE	234	-
3	8	TRUE	234	optimal
4	8	TRUE	234	-
5	8	TRUE	234	optimal
A/A T=3				
1	8	TRUE	234	-
2	8	TRUE	234	-
3	8	TRUE	234	-

4	8	TRUE	234	-
5	8	TRUE	234	optimal

Πίνακας 4.154. Κρυπτογραφικές ιδιότητες εξόδων συνάρτησης.

2. Αρχικοποίηση προγράμματος για εκτέλεση για πληθυσμό =1000:

Θέτουμε πληθυσμό ίσο με 1000, πιθανότητα μετάλλαξης ίση με 0.5, πιθανότητα διασταύρωσης ίση με 0.99 , elitsm ίσο με true, tournamentsize ίσο με 2 και 3 αντίστοιχα , ενώ ο αλγόριθμος τερματίζει όταν η συνάρτηση καταλληλότητας λάβει τιμή ίση με 234(που είναι η τιμή της μη γραμμικότητας της Carlet-Feng συνάρτησης για n=9 επαυξημένη κατά 2 μονάδες).

A/A T=2	**GENERATION**	**ΧΡΟΝΟΣ ΕΚΤΕΛΕΣΗΣ (SECONDS)**
1	129	40.43
2	191	66.32
3	101	38.59
4	188	64.69
5	212	68.52
A/A T=3		
1	1239	464.57
2	166	62.18
3	262	100.76
4	3423	1268.59
5	9977	3500.17
Μέσος όρος T2-T3		55.71-1079.25

Πίνακας 4.155. Στατιστικά στοιχεία εξόδων συνάρτησης.

A/A T=2	**ΒΑΘΜΟΣ**	**BALANCED**	**NONLINEARITY**	**FAI**
1	8	TRUE	234	optimal
2	8	TRUE	234	-
3	8	TRUE	234	-
4	8	TRUE	234	-
5	8	TRUE	234	optimal
A/A T=3				
1	8	TRUE	234	-
2	8	TRUE	234	-

3	8	TRUE	234	-
4	8	TRUE	234	-
5	8	TRUE	234	-

Πίνακας 4.156. Κρυπτογραφικές ιδιότητες εξόδων συνάρτησης.

3. Αρχικοποίηση προγράμματος για εκτέλεση για πληθυσμό=1500:

Θέτουμε πληθυσμό ίσο με 1500, πιθανότητα μετάλλαξης ίση με 0.7, πιθανότητα διασταύρωσης ίση με 0.99 , elitsm ίσο με true, tournamentsize ίσο με 2 και 3 αντίστοιχα , ενώ ο αλγόριθμος τερματίζει όταν η συνάρτηση καταλληλότητας λάβει τιμή ίση με 234(που είναι η τιμή της μη γραμμικότητας της Carlet-Feng συνάρτησης για n=9 επαυξημένη κατά 2 μονάδες).

Α/Α Τ=2	**GENERATION**	**ΧΡΟΝΟΣ ΕΚΤΕΛΕΣΗΣ (SECONDS)**
1	175	82.47
2	178	83.5
3	136	63.42
4	185	86.49
5	187	87.44
Α/Α Τ=3		
1	72	45.95
2	101	61.8
3	127	77.81
4	94	61.62
5	75	46.24
Μέσος όρος Τ2--Τ3		80.66-47.68

Πίνακας 4.157. Στατιστικά στοιχεία εξόδων συνάρτησης.

Α/Α Τ=2	**ΒΑΘΜΟΣ**	**BALANCED**	**NONLINEARITY**	**FAI**
1	8	TRUE	234	-
2	8	TRUE	234	-
3	8	TRUE	234	-
4	8	TRUE	234	-
5	8	TRUE	234	-
Α/Α Τ=3				

1	8	TRUE	234	-
2	8	TRUE	234	-
3	8	TRUE	234	-
4	8	TRUE	234	-
5	8	TRUE	234	-

Πίνακας 4.158. Κρυπτογραφικές ιδιότητες εξόδων συνάρτησης.

4. Αρχικοποίηση προγράμματος για εκτέλεση για πληθυσμό=2000:

Θέτουμε πληθυσμό ίσο με 2000, πιθανότητα μετάλλαξης ίση με 0.9, πιθανότητα διασταύρωσης ίση με 0.99 , elitsm ίσο με true, tournamentsize ίσο με 2 και 3 αντίστοιχα , ενώ ο αλγόριθμος τερματίζει όταν η συνάρτηση καταλληλότητας λάβει τιμή ίση με 234(που είναι η τιμή της μη γραμμικότητας της Carlet-Feng συνάρτησης για n=9 αυξημένης κατά 2 μονάδες).

Α/Α Τ=2	**GENERATION**	**ΧΡΟΝΟΣ ΕΚΤΕΛΕΣΗΣ (SECONDS)**
1	410	261.22
2	486	309.8
3	287	182.59
4	448	285.9
5	166	106.23
Α/Α Τ=3		
1	85	69.72
2	91	77.33
3	74	62.17
4	67	59.28
5	70	58.9
Μέσος όρος Τ2-Τ3		229.15-65.5

Πίνακας 4.159. Στατιστικά στοιχεία εξόδων συνάρτησης.

Α/Α Τ=2	**ΒΑΘΜΟΣ**	**BALANCED**	**NONLINEARITY**	**FAI**
1	8	TRUE	234	-
2	8	TRUE	234	-
3	8	TRUE	234	-
4	8	TRUE	234	-
5	8	TRUE	234	-
Α/Α				

T=3				
1	8	TRUE	234	-
2	8	TRUE	234	-
3	8	TRUE	234	-
4	8	TRUE	234	-
5	8	TRUE	234	-

Πίνακας 4.160. Κρυπτογραφικές ιδιότητες εξόδων συνάρτησης.

4.2 Συμπεράσματα Αποτελεσμάτων

Από τα αποτελέσματα της έρευνας και για τα πέντε προγράμματα που δημιουργήθηκαν σαν γενικό συμπέρασμα είναι ότι σε όλα τα προγράμματα επιτεύχθηκε ο αρχικός στόχος οι έξοδοι να έχουν υψηλή μη γραμμικότητα (σε ελάχιστες μόνο περιπτώσεις δεν κατέστη εφικτό, υπό το χρονικό πλαίσιο που τέθηκε για την εκτέλεση των προγραμμάτων, να λάβει η μη γραμμικότητα την επιθυμητή τιμή, όπως αυτές προσδιορίζονται ανωτέρω) και είναι και ισοβαρείς συναρτήσεις. Για τα υπόλοιπα κρυπτογραφικά κριτήρια ακολουθούν συμπεράσματα για το κάθε είδος εξόδου κάθε προγράμματος.

4.2.1 Συμπεράσματα Πρώτου Προγράμματος

Για n=5 (έξοδος 32 bit) για τους πληθυσμούς 500,1000 με πιθανότητα μετάλλαξης 0.3 και 0.5 αντίστοιχα, υπήρχαν έξοδοι που κάλυπταν και τα τέσσερα κριτήρια. Δηλαδή οι συναρτήσεις που υπολογίστηκαν είχαν τον μέγιστο υψηλό βαθμό «4» , ήταν ισοβαρείς, είχαν καλή μη γραμμικότητα 10 και ήταν και ανθεκτικές στις γρήγορες αλγεβρικές επιθέσεις. Ωστόσο υπήρχαν και έξοδοι που δεν κάλυπταν όλα τα κρυπτογραφικά κριτήρια. Συγκεκριμένα υπήρχαν έξοδοι που είχαν τον μέγιστο βαθμό «4», ήταν ισοβαρείς ,είχαν υψηλή μη γραμμικότητα 10 ή 12 αλλά δεν ήταν ανθεκτικές στις γρήγορες αλγεβρικές επιθέσεις. Επίσης υπήρχαν έξοδοι που είχαν υψηλό βαθμό αλλά όχι τον μέγιστο δυνατό (3), ήταν ισοβαρείς, είχαν υψηλή μη γραμμικότητα 12, αλλά ήταν ευπαθείς στις γρήγορες αλγεβρικές επιθέσεις.

Για n=6 (έξοδος 64 bit) για όλους τους πληθυσμούς 500, 1000, 1500, 2000 και για όλες τις πιθανότητες μετάλλαξης 0.3, 0.5, 0.7, 0.9 αντίστοιχα όλες οι έξοδοι κάλυπταν όλα τα

κρυπτογραφικά κριτήρια. Πιο συγκεκριμένα είχαν τον μέγιστο υψηλό βαθμό «5», ήταν ισοβαρείς, είχαν υψηλή μη γραμμικότητα (24 ή26) και ήταν ανθεκτικές στις γρήγορες αλγεβρικές επιθέσεις. Πιο συγκεκριμένα για πληθυσμό 2000 με πιθανότητα μετάλλαξης 0.9 υπήρχε έξοδος που είχε μη γραμμικότητα υψηλότερη του στόχου (με στόχο μη γραμμικότητα 24 υπήρχε έξοδος με μη γραμμικότητα 26).

Για n=7 (έξοδος 128 bit) για όλους τους πληθυσμούς 500, 1000, 1500, 2000 και για όλες τις πιθανότητες μετάλλαξης 0.3, 0.5, 0.7, 0.9 αντίστοιχα όλες οι έξοδοι κάλυπταν όλα τα κρυπτογραφικά κριτήρια. Πιο συγκεκριμένα είχαν τον μέγιστο υψηλό βαθμό 6, ήταν ισοβαρείς, είχαν υψηλή μη γραμμικότητα 54 και ήταν ανθεκτικές στις γρήγορες αλγεβρικές επιθέσεις.

Για n=8 (έξοδος 256 bit) για όλους τους πληθυσμούς 500, 1000, 1500, 2000 και για όλες τις πιθανότητες μετάλλαξης 0.3, 0.5, 0.7, 0.9 αντίστοιχα όλες οι έξοδοι κάλυπταν όλα τα κρυπτογραφικά κριτήρια. Πιο συγκεκριμένα οι περισσότερες έξοδοι είχαν τον μέγιστο υψηλό βαθμό 7, ήταν ισοβαρείς, είχαν υψηλή μη γραμμικότητα 112 και ήταν ανθεκτικές στις γρήγορες αλγεβρικές επιθέσεις. Ωστόσο υπήρχαν και έξοδοι που είχαν υψηλό βαθμό αλλά όχι τον μέγιστο δυνατό (6), οι οποίες παρόλα αυτά ήταν ισοβαρείς, είχαν υψηλή μη γραμμικότητα 112 και ήταν ανθεκτικές στις γρήγορες αλγεβρικές επιθέσεις.

Για n=9 (έξοδος 512 bit) για τους πληθυσμούς 500, 1000, 2000 με πιθανότητες μετάλλαξης 0.3, 0.5, 0.9 αντίστοιχα υπήρχαν έξοδοι που ικανοποιούσαν όλα τα κρυπτογραφικά κριτήρια. Πιο συγκεκριμένα είχαν τον μέγιστο υψηλό βαθμό 8, ήταν ισοβαρείς , είχαν υψηλή μη γραμμικότητα αλλά όχι την μη γραμμικότητα του στόχου που είχε δοθεί (230) και ήταν ανθεκτικές στις γρήγορες αλγεβρικές επιθέσεις. Υπήρχαν όμως και έξοδοι που κάλυπταν τα τρία πρώτα στοιχεία αλλά ήταν ευπαθείς στις γρήγορες αλγεβρικές επιθέσεις. Αξίζει να σημειωθεί ότι δεν επιτεύχθηκε πλήρως ο στόχος της μη γραμμικότητας γιατί ο γενετικός αλγόριθμος σταμάτησε την αναζήτηση φτάνοντας στο μέγιστο χρονικό περιθώριο που είχε δοθεί στο πρόγραμμα (172800 δευτερόλεπτα).

4.2.2 Συμπεράσματα Δεύτερου Προγράμματος

Για n=5(έξοδος 32 bit) για όλους τους πληθυσμούς και για όλες τις πιθανότητες μετάλλαξης δεν υπήρχε καμία έξοδος που να ικανοποιεί όλα τα κρυπτογραφικά κριτήρια. Συγκεκριμένα υπήρχαν έξοδοι που είχαν τον μέγιστο υψηλό βαθμό 4, ήταν ισοβαρείς, είχαν υψηλή μη γραμμικότητα 12 , αλλά ήταν ευπαθείς στις γρήγορες αλγεβρικές επιθέσεις. Επίσης υπήρχαν έξοδοι που είχαν υψηλό βαθμό αλλά όχι τον μέγιστο δυνατό 3 ,ήταν ισοβαρείς, είχαν υψηλή μη γραμμικότητα 12, αλλά ήταν ευπαθείς στις γρήγορες αλγεβρικές επιθέσεις. Εδώ να σημειωθεί ότι επιτεύχθηκε αύξηση της μη γραμμικότητας όλων των εξόδων κατά 2 μονάδες σε σχέση με την συνάρτηση Carlet-Feng.

Για n=6 (έξοδος 64 bit) για όλους τους πληθυσμούς 500, 1000, 1500, 2000 και για όλες τις πιθανότητες μετάλλαξης 0.3, 0.5, 0.7, 0.9 αντίστοιχα όλες οι έξοδοι κάλυπταν όλα τα κρυπτογραφικά κριτήρια. Πιο συγκεκριμένα είχαν τον μέγιστο υψηλό βαθμό 5, ήταν ισοβαρείς, είχαν υψηλή μη γραμμικότητα 26 και ήταν ανθεκτικές στις γρήγορες αλγεβρικές επιθέσεις. Εδώ να σημειωθεί ότι επιτεύχθηκε αύξηση της μη γραμμικότητας όλων των εξόδων κατά 2 μονάδες σε σχέση με την συνάρτηση Carlet Feng.

Για n=7 (έξοδος 128 bit) για όλους τους πληθυσμούς 500, 1000, 1500, 2000 και για όλες τις πιθανότητες μετάλλαξης 0.3, 0.5, 0.7, 0.9 αντίστοιχα όλες οι έξοδοι κάλυπταν όλα τα κρυπτογραφικά κριτήρια. Πιο συγκεκριμένα είχαν τον μέγιστο υψηλό βαθμό 6, ήταν ισοβαρείς, είχαν υψηλή μη γραμμικότητα 54 και ήταν ανθεκτικές στις γρήγορες αλγεβρικές επιθέσεις.

Για n=8 (έξοδος 256 bit) για όλους τους πληθυσμούς 500, 1000, 1500, 2000 και για όλες τις πιθανότητες μετάλλαξης 0.3, 0.5, 0.7, 0.9 αντίστοιχα όλες οι έξοδοι κάλυπταν όλα τα κρυπτογραφικά κριτήρια. Συγκεκριμένα οι περισσότερες έξοδοι είχαν τον μέγιστο υψηλό βαθμό 7, ήταν ισοβαρείς, είχαν υψηλή μη γραμμικότητα 112 και ήταν ανθεκτικές στις γρήγορες αλγεβρικές επιθέσεις. Υπήρχαν και έξοδοι που είχαν υψηλό βαθμό αλλά όχι τον μέγιστο δυνατό 6, ήταν ισοβαρείς, είχαν υψηλή μη γραμμικότητα 112 και ήταν ανθεκτικές στις γρήγορες αλγεβρικές επιθέσεις. Να σημειωθεί ότι για τον πληθυσμό 1500 με πιθανότητα μετάλλαξης 0.7, για την μέθοδο της επιλογής τουρνουά μεγέθους N=2 και για N=3 «tournament_size=2 ,tournament_size=3» στην μέθοδο επιλογής τουρνουά όλες οι έξοδοι είχαν τον μέγιστο δυνατό υψηλό βαθμό 7, σε αντίθεση με τους άλλους πληθυσμούς που δεν είχαν όλες οι έξοδοι τον μέγιστο δυνατό υψηλό βαθμό.

Για n=9 (έξοδος 512 bit) για τους πληθυσμό 1000, 1500, 2000 με πιθανότητα μετάλλαξης 0.5, 0.7, 0.9 αντίστοιχα και για την μέθοδο της επιλογής τουρνουά μεγέθους N=3 «tournament_size=3» υπήρχαν έξοδοι που ικανοποιούσαν όλα τα κρυπτογραφικά κριτήρια. Επίσης για πληθυσμό 2000 με πιθανότητα μετάλλαξης 0.9 και για την μέθοδο της επιλογής τουρνουά μεγέθους N=2 «tournament_size=2» υπήρχε έξοδος που ικανοποιούσε όλα τα κρυπτογραφικά κριτήρια. Συγκεκριμένα αυτοί οι έξοδοι είχαν τον μέγιστο υψηλό βαθμό 8, ήταν ισοβαρείς, είχαν υψηλή μη γραμμικότητα 230 και ήταν ανθεκτικές στις γρήγορες αλγεβρικές επιθέσεις. Υπήρχαν όμως και συναρτήσεις που δεν ικανοποιούσαν πλήρως τα κρυπτογραφικά κριτήρια. Οι περισσότερες έξοδοι είχαν τον μέγιστο υψηλό βαθμό 8, ήταν ισοβαρείς, είχαν υψηλή μη γραμμικότητα 230 αλλά ήταν ευπαθείς στις γρήγορες αλγεβρικές επιθέσεις. Επίσης για πληθυσμό 500 με πιθανότητα μετάλλαξης 0.3 για την μέθοδο της επιλογής τουρνουά μεγέθους N=3 «tournament_size=3» επιτεύχθηκε σε έξοδο μη ισοβαρής συνάρτηση με βαθμό ίσο με n=9. Αξίζει να σημειωθεί ότι δεν επιτεύχθηκε πλήρως ο στόχος της μη γραμμικότητας γιατί ο γενετικός αλγόριθμος σταμάτησε την αναζήτηση φτάνοντας στο μέγιστο χρονικό περιθώριο που είχε δοθεί στο πρόγραμμα (172800 δευτερόλεπτα).

4.2.3 Συμπεράσματα Τρίτου Προγράμματος

Για n=5 (έξοδος 32 bit) για όλους τους πληθυσμούς και για όλες τις πιθανότητες μετάλλαξης δεν υπήρχε καμία έξοδος που να ικανοποιεί όλα τα κρυπτογραφικά κριτήρια. Συγκεκριμένα υπήρχαν έξοδοι που είχαν τον μέγιστο υψηλό βαθμό 4, ήταν ισοβαρείς, είχαν υψηλή μη γραμμικότητα 12 , αλλά ήταν ευπαθείς στις γρήγορες αλγεβρικές επιθέσεις. Επίσης υπήρχαν έξοδοι που είχαν υψηλό βαθμό αλλά όχι τον μέγιστο δυνατό (3) , ήταν ισοβαρείς, είχαν υψηλή μη γραμμικότητα 12, αλλά ήταν ευπαθείς στις γρήγορες αλγεβρικές επιθέσεις. Εδώ να σημειωθεί ότι επιτεύχθηκε αύξηση της μη γραμμικότητας όλων των εξόδων κατά 2 μονάδες σε σχέση με την συνάρτηση Carlet-Feng.

Για n=6 (έξοδος 64 bit) για όλους τους πληθυσμούς 500, 1000, 1500, 2000 και για όλες τις πιθανότητες μετάλλαξης 0.3, 0.5, 0.7, 0.9 αντίστοιχα όλες οι έξοδοι κάλυπταν όλα τα κρυπτογραφικά κριτήρια. Πιο συγκεκριμένα είχαν τον μέγιστο υψηλό βαθμό 5, ήταν ισοβαρείς, είχαν υψηλή μη γραμμικότητα 26 και ήταν ανθεκτικές στις γρήγορες

αλγεβρικές επιθέσεις. Εδώ να σημειωθεί ότι επιτεύχθηκε αύξηση της μη γραμμικότητας όλων των εξόδων κατά 2 μονάδες σε σχέση με την συνάρτηση Carlet- Feng.

Για n=7 (έξοδος 128 bit) για όλους τους πληθυσμούς 500, 1000, 1500, 2000 και για όλες τις πιθανότητες μετάλλαξης 0.3, 0.5, 0.7, 0.9 αντίστοιχα όλες οι έξοδοι κάλυπταν όλα τα κρυπτογραφικά κριτήρια. Συγκεκριμένα είχαν τον μέγιστο υψηλό βαθμό 6, ήταν ισοβαρείς, είχαν υψηλή μη γραμμικότητα 54 και ήταν ανθεκτικές στις γρήγορες αλγεβρικές επιθέσεις.

Για n=8 (έξοδος 256 bit) για όλους τους πληθυσμούς 500, 1000, 1500, 2000 και για όλες τις πιθανότητες μετάλλαξης 0.3, 0.5, 0.7, 0.9 αντίστοιχα όλες οι έξοδοι κάλυπταν όλα τα κρυπτογραφικά κριτήρια. Συγκεκριμένα οι περισσότερες έξοδοι είχαν τον μέγιστο υψηλό βαθμό 7, ήταν ισοβαρείς, είχαν υψηλή μη γραμμικότητα 112 και ήταν ανθεκτικές στις γρήγορες αλγεβρικές επιθέσεις. Υπήρχαν και έξοδοι που είχαν υψηλό βαθμό αλλά όχι τον μέγιστο δυνατό (6), ήταν ισοβαρείς, είχαν υψηλή μη γραμμικότητα 112 και ήταν ανθεκτικές στις γρήγορες αλγεβρικές επιθέσεις.

Για n=9 (έξοδος 512 bit) για όλους τους πληθυσμούς δεν υπήρξε καμία έξοδος που να ικανοποιεί όλα τα κρυπτογραφικά κριτήρια. Συγκεκριμένα οι περισσότερες έξοδοι είχαν τον μέγιστο υψηλό βαθμό 8, ήταν ισοβαρείς, είχαν υψηλή μη γραμμικότητα 232 αλλά ήταν ευπαθείς στις γρήγορες αλγεβρικές επιθέσεις. Υπήρχαν έξοδοι που δεν είχαν τον μέγιστο δυνατό βαθμό (7) ,ήταν ισοβαρείς, είχαν υψηλή μη γραμμικότητα 232 αλλά ήταν ευπαθείς στις γρήγορες αλγεβρικές επιθέσεις. Για πληθυσμό ίσο με 1500 και πιθανότητα μετάλλαξης 0.7 και για την μέθοδο της επιλογής τουρνουά μεγέθους N=2 «tournament_size=2» αξίζει να σημειωθεί, ότι δεν επιτεύχθηκε πλήρως ο στόχος της μη γραμμικότητας (230), γιατί ο γενετικός αλγόριθμος σταμάτησε την αναζήτηση φτάνοντας στο μέγιστο χρονικό περιθώριο που είχε δοθεί στο πρόγραμμα (172800 δευτερόλεπτα).

4.2.4 Συμπεράσματα Τέταρτου Προγράμματος

Για n=5 (έξοδος 32 bit) για όλους τους πληθυσμούς και για όλες τις πιθανότητες μετάλλαξης δεν υπήρχε καμία έξοδος που να ικανοποιεί όλα τα κρυπτογραφικά κριτήρια. Συγκεκριμένα υπήρχαν έξοδοι που είχαν τον μέγιστο υψηλό βαθμό 4, ήταν ισοβαρείς, είχαν υψηλή μη γραμμικότητα 12 , αλλά ήταν ευπαθείς στις γρήγορες

αλγεβρικές επιθέσεις. Επίσης υπήρχαν έξοδοι που είχαν υψηλό βαθμό αλλά όχι τον μέγιστο δυνατό 3 , ήταν ισοβαρείς, είχαν υψηλή μη γραμμικότητα 12, αλλά ήταν ευπαθείς στις γρήγορες αλγεβρικές επιθέσεις. Εδώ να σημειωθεί ότι επιτεύχθηκε αύξηση της μη γραμμικότητας όλων των εξόδων κατά 2 μονάδες σε σχέση με την συνάρτηση Carlet-Feng.

Για n=6 (έξοδος 64 bit) για όλους τους πληθυσμούς 500, 1000, 1500, 2000 και για όλες τις πιθανότητες μετάλλαξης 0.3, 0.5, 0.7, 0.9 αντίστοιχα η πλειοψηφία των εξόδων κάλυπτε όλα τα κρυπτογραφικά κριτήρια. Συγκεκριμένα οι περισσότερες έξοδοι είχαν τον μέγιστο υψηλό βαθμό 5, ήταν ισοβαρείς, είχαν υψηλή μη γραμμικότητα 26 και ήταν ανθεκτικές στις γρήγορες αλγεβρικές επιθέσεις. Για πληθυσμό ίσο με 500 με πιθανότητα μετάλλαξης 0.3 και για την μέθοδο της επιλογής τουρνουά μεγέθους N=2 «tournament_size=2» υπήρχαν δύο έξοδοι που ήταν ευπαθείς στης γρήγορες αλγεβρικές επιθέσεις αλλά είχαν μέγιστο βαθμό, ήταν ισοβαρείς και είχαν υψηλή μη γραμμικότητα 26. Εδώ να σημειωθεί ότι επιτεύχθηκε αύξηση της μη γραμμικότητας όλων των εξόδων κατά 2 μονάδες σε σχέση με την συνάρτηση Carlet- Feng.

Για n=7 (έξοδος 128 bit) για όλους τους πληθυσμούς 500, 1000, 1500, 2000 και για όλες τις πιθανότητες μετάλλαξης 0.3, 0.5, 0.7, 0.9 αντίστοιχα όλες οι έξοδοι κάλυπταν όλα τα κρυπτογραφικά κριτήρια. Συγκεκριμένα όλες οι έξοδοι είχαν υψηλό βαθμό αλλά όχι τον μέγιστο δυνατό (5), ήταν ισοβαρείς, είχαν υψηλή μη γραμμικότητα 56 και ήταν ανθεκτικές στις γρήγορες αλγεβρικές επιθέσεις. Εδώ να σημειωθεί ότι επιτεύχθηκε αύξηση της μη γραμμικότητας όλων των εξόδων κατά 2 μονάδες σε σχέση με την συνάρτηση Carlet-Feng. Αυξάνοντας την μη γραμμικότητα κατά 2 μονάδες παρατηρήθηκε και μείωση του βαθμού κατά μία μονάδα σε σχέση με εξόδους με μη γραμμικότητα ίση με 54. Δηλαδή για μη γραμμικότητα 54 είχαμε βαθμό «6», ενώ για μη γραμμικότητα ίση με 56 είχαμε βαθμό «5».

Για n=8 (έξοδος 256 bit) για όλους τους πληθυσμούς 500, 1000, 1500, 2000 και για όλες τις πιθανότητες μετάλλαξης 0.3, 0.5, 0.7, 0.9 αντίστοιχα όλες οι έξοδοι κάλυπταν όλα τα κρυπτογραφικά κριτήρια. Συγκεκριμένα όλες οι έξοδοι είχαν τον μέγιστο υψηλό βαθμό 7, ήταν ισοβαρείς, είχαν υψηλή μη γραμμικότητα 114 και ήταν ανθεκτικές στις γρήγορες αλγεβρικές επιθέσεις. Εδώ να σημειωθεί ότι επιτεύχθηκε αύξηση της μη γραμμικότητας όλων των εξόδων κατά 2 μονάδες σε σχέση με την συνάρτηση Carlet-Feng.

Για n=9 (έξοδος 512 bit) για τους πληθυσμούς 500, 1000 με πιθανότητες μετάλλαξης 0.3, 0.5 αντίστοιχα και για την μέθοδο της επιλογής τουρνουά μεγέθους N=2 «tournament_size=2» όπως και για πληθυσμό 500 με πιθανότητα μετάλλαξης 0.3 για την μέθοδο της επιλογής τουρνουά μεγέθους N=3 «tournament_size=3» υπήρχαν έξοδοι που ικανοποιούσαν πλήρως τα κρυπτογραφικά κριτήρια. Πιο αναλυτικά οι έξοδοι αυτοί είχαν τον μέγιστο δυνατό βαθμό 8, ήταν ισοβαρείς , είχαν υψηλή μη γραμμικότητα 234 και ήταν ανθεκτικές στις γρήγορες αλγεβρικές επιθέσεις. Υπήρχαν αρκετές έξοδοι που δεν ικανοποιούσαν πλήρως τα κρυπτογραφικά κριτήρια. Συγκεκριμένα είχαν τον μέγιστό δυνατό βαθμό 8, ήταν ισοβαρείς είχαν υψηλή μη γραμμικότητα 234, αλλά ήταν ευπαθείς στις γρήγορες αλγεβρικές επιθέσεις. Εδώ να σημειωθεί ότι επιτεύχθηκε αύξηση της μη γραμμικότητας όλων των εξόδων κατά 2 μονάδες σε σχέση με την συνάρτηση Carlet-Feng.

4.3 Οι «βέλτιστες» συναρτήσεις

Όπως αναφέρθηκε στο Κεφάλαιο 2, αν μία συνάρτηση έχει τη μέγιστη δυνατή ανθεκτικότητα στις γρήγορες αλγεβρικές επιθέσεις FAI (η οποία ισούται με n), τότε έχει και τη μέγιστη ανθεκτικότητα στις κλασικές αλγεβρικές επιθέσεις, δηλαδή τη μέγιστη αλγεβρική ανθεκτικότητα. Από την άλλη μεριά, όπως επίσης επισημάναμε στο Κεφάλαιο 2, μία συνάρτηση με n μεταβλητές έχει τη μέγιστη δυνατή ανθεκτικότητα στις γρήγορες αλγεβρικές επιθέσεις μόνο αν το n είναι της μορφής $n=2^s+1$ για κάποιο s.

Στα αποτελέσματά μας, στις περιπτώσεις n=6, 7 και 8 αντίστοιχα, όπου αναφέρουμε ότι βρέθηκε η μέγιστη δυνατή ανθεκτικότητα στις γρήγορες αλγεβρικές επιθέσεις FAI, εννοούμε τη μέγιστη δυνατή που «επιτρέπει» η τιμή του n (η οποία, λόγω του προαναφερθέντος αποτελέσματος, δεν είναι η μέγιστη θεωρητικά δυνατή n αλλά είναι n-1). Μόνο στις περιπτώσεις n=5 και n=9 η ένδειξη «μέγιστη FAI» αντιστοιχεί στην τιμή FAI=n, το οποίο σημαίνει ότι αυτές οι συναρτήσεις έχουν και τη μέγιστη αλγεβρική ανθεκτικότητα (κάτι που ικανοποιεί και η Carlet-Feng συνάρτηση). Συνεπώς, όλες οι συναρτήσεις 5 ή 9 μεταβλητών που βρήκαμε, οι οποίες είναι ισοβαρείς, έχουν μη γραμμικότητα ίση ή μεγαλύτερη από τη συνάρτηση Carlet-Feng και έχουν μέγιστο FAI, είναι συναρτήσεις που είναι «ισοδύναμες» ή και «καλύτερες» (αναλόγως την τιμή της μη γραμμικότητας) από την ισχυρή συνάρτηση Carlet-Feng όσον αφορά τα εν λόγω κρυπτογραφικά κριτήρια. Οι συναρτήσεις για n=9 προέκυψαν από τον τέταρτο

αλγόριθμο για πληθυσμούς: 500 με μέγεθος τουρνουά ίσο με 2 και 3 αντίστοιχα, 1000 με μέγεθος τουρνουά ίσο με 2 και εμφανίζονται στο Παράρτημα Β.

Προφανώς, και για τις άλλες τιμές του n οι συναρτήσεις που βρήκαμε ενδέχεται να είναι «ισοδύναμες» ή και «καλύτερες» από τη συνάρτηση Carlet-Feng, εάν υπολογιστεί ότι έχουν και τη μέγιστη αλγεβρική ανθεκτικότητα: αυτό μπορεί να ελεγχθεί σε επόμενο ερευνητικό βήμα.

Κεφάλαιο 5

Επίλογος

Ο κύριος στόχος της έρευνας ήταν η κατασκευή νέων κρυπτογραφικών συναρτήσεων με καλά κρυπτογραφικά κριτήρια, με την βοήθεια των Εξελικτικών Αλγορίθμων. Συγκεκριμένα στόχος ήταν η κατασκευή λογικών συναρτήσεων που να έχουν υψηλή μη γραμμικότητα, να είναι ισοβαρείς , να έχουν υψηλό βαθμό και να είναι ανθεκτικές στης αλγεβρικές επιθέσεις και στις γρήγορες αλγεβρικές επιθέσεις.

Για την επίτευξη αυτού, το είδος του εξελικτικού αλγόριθμου που χρησιμοποιήθηκε είναι ο γενετικός αλγόριθμος. Συγκεκριμένα χρησιμοποιήθηκε ο γενετικός αλγόριθμος γενεών με ελιτισμό «elitism Generational GA». Έτσι κατασκευάστηκαν τέσσερα προγράμματα γενετικών αλγορίθμων όπου συνδυάστηκαν διάφορα είδη τελεστών (επιλογής, διασταύρωσης και μετάλλαξης) και υπολογισμών της συνάρτησης καταλληλότητας.

Ως βάση σύγκρισης χρησιμοποιήσαμε μία γνωστή καλή κρυπτογραφική συνάρτηση, τη συνάρτηση Carlet-Feng ,που ικανοποιεί το σύνολο των κρυπτογραφικών κριτηρίων. Συγκεκριμένα τέθηκε σαν αρχικός στόχος για τον υπολογισμό της συνάρτηση καταλληλότητας του γενετικού αλγορίθμου, η μη γραμμικότητα της συνάρτησης Carlet –Feng και η συνάρτηση να είναι ισοβαρής. Για κάθε πρόγραμμα έγινε εξαγωγή ενδεικτικά πέντε εξόδων όπου με την βοήθεια του εργαλείου “FAA equation Finder” και των προγραμμάτων που κατασκευάστηκαν μελετήθηκαν οι κρυπτογραφικές τους ιδιότητες.

Οι συναρτήσεις που μελετήθηκαν ήταν για εισόδους από n=5 έως n=9, δηλαδή έξοδοι (32,64,128,256,512 bit). Τα αποτελέσματα των συναρτήσεων που μελετήθηκαν είναι ενθαρρυντικά γιατί κατέστη εφικτή η κατασκευή συναρτήσεων με καλά κρυπτογραφικά χαρακτηριστικά. Σε κάποιες περιπτώσεις τα κρυπτογραφικά κριτήρια

ήταν ίδια με αυτά της συνάρτησης Carlet-Fent και σε κάποιες άλλες ήταν καλύτερες (ως προς την μη γραμμικότητα που επιτεύχθηκε επαύξηση της κατά 2 μονάδες).

Από τα τέσσερα προγράμματα που κατασκευάστηκαν το πρόγραμμα που είχε καλύτερη απόδοση ήταν το τέταρτο, γιατί επιτεύχθηκε για όλα τα είδη εξόδων (32,64,128,256,512 bit) αύξηση της μη γραμμικότητας, σε σχέση με αυτή της συνάρτησης Carlet-Feng, κατά 2 μονάδες (12,26,56,114,234). Περαιτέρω, για το συγκεκριμένο πρόγραμμα για n=9 (512 bit) για πληθυσμό ίσο με 500 με μέγεθος επιλογής τουρνουά 2 και 3 αντίστοιχα και για πληθυσμό ίσο με 1000 με μέγεθος επιλογής τουρνουά 2 επιτεύχθηκαν οι καλύτερες λύσεις, γιατί για τους συγκεκριμένους πληθυσμούς βρέθηκαν λύσεις που ικανοποιούσαν όλα τα κρυπτογραφικά κριτήρια που είχε τεθεί ως στόχος (μέγιστος βαθμός, μέγιστη αλγεβρική ανθεκτικότητα, μέγιστη γρήγορη αλγεβρική ανθεκτικότητα, υψηλή μη γραμμικότητα, ισοβαρής).

Παρόλο που επιτεύχθηκε ο βασικός στόχος που είχε τεθεί, δηλαδή κατασκευή καλών κρυπτογραφικών συναρτήσεων με την βοήθεια των γενετικών αλγορίθμων, υπάρχει περαιτέρω δυνατότητα μετεξέλιξης των συγκεκριμένων αλγορίθμων που κατασκευάστηκαν. Συγκεκριμένα μελλοντικός στόχος για τις συγκεκριμένες συναρτήσεις είναι υπολογιστεί για n=6 έως n=8 που βρέθηκαν ότι είναι ανθεκτικές στις γρήγορες αλγεβρικές επιθέσεις αν έχουν και τη μέγιστη αλγεβρική ανθεκτικότητα. Επιπλέον θα μπορούν να συνδυαστούν μεταξύ τους και διάφορα άλλα είδη τελεστών μεταξύ τους που δεν μελετήθηκαν (π.χ. επιλογή πληθυσμού με την μέθοδο της ρουλέτας με διασταύρωση ενός σημείου και μετάλλαξη ενός σημείου κ.τ.λ.). Επίσης, με βάση τις εξόδους των παραπάνω προγραμμάτων θα μπορούν να χρησιμοποιηθούν για την δημιουργία νέων συναρτήσεων μέσω γενετικών αλγορίθμων με προκαθορισμένο πληθυσμό, αντί της τυχαίας δημιουργίας πληθυσμού. Με αυτόν τον τρόπο θα μελετηθεί αν μπορούν να εξελιχθούν περισσότερο οι υπάρχουσες έξοδοι που βγάζουν τα συγκεκριμένα προγράμματα. Τέλος θα μπορεί να μελετηθεί και ένα άλλο είδος εξελικτικού αλγορίθμου, συγκεκριμένα του γενετικού προγραμματισμού.

Παράρτημα Α
Λογισμικά Υλοποίησης Έρευνας

Α.1 Πρώτο Πρόγραμμα

Α.1.1 Κλάση Boolean_ga

```
public class Boolean_ga {

    /**

   * @param args the command line arguments

   */

  public static void main(String[] args) {

     for (int j=1; j<=5; j++){

     final long startTime = System.nanoTime();

     Individual solution =new Individual();

     Nonlinearity no = new Nonlinearity();

     int d[] =new int []{};

     double seconds2=0;

// Create GA object

GeneticAlgorithm ga = new GeneticAlgorithm(500, 0.3, 0.99, 2);

// Initialize population

Population population = ga.initPopulation(512);

ga.evalPopulation(population);
```

```
int generation = 1;
while (ga.isTerminationConditionMet(population) == false) {
  if (seconds2>=172800.00){break;}
// Print fittest individual  from population
        System.out.println("Best solution: " + population.getFittest(0).toString());
//crossover
        population = ga.crossoverPopulation(population);
//mutation
        population = ga.mutatePopulation(population);
// Evaluate population
        ga.evalPopulation(population);
// Increment the current generation
        generation++;
        final long duration2 = System.nanoTime() - startTime;
        seconds2 =(double) duration2 / 1000000000.0;
        seconds2=Math.round(seconds2*100);
        seconds2=seconds2/100;
}
solution= population.getFittest(0);
System.out.println("Found solution in " + generation + " generations");
System.out.println("Best solution: " + solution);
final long duration = System.nanoTime() - startTime;
double seconds =(double) duration / 1000000000.0;
```

```
seconds=Math.round(seconds*100);

seconds=seconds/100;

int c[]=new int [solution.getChromosomeLength()];

      for (int i = 0; i < solution.getChromosomeLength();  i++) {

       c[i]= solution.getGene(i);

      }

     boolean balance= no.Balanced(c);

     no.write_genes_to_data(c);

     d=no.FWT(c);

     int e=no.nonLinearity(d);

     double persent=no.success_persent(e);

     no.write_criteria_to_data(generation,balance,e,seconds,persent);

     System.out.println("Balanced:" +balance);

     System.out.println("Nonlinearity:" +  e);

     System.out.println("Success Persent:" + persent +"%");

     System.out.println("Time:" +seconds +" seconds");}}}
```

A.1.2 Κλάση GeneticAlgorithm

```
package boolean_ga;


/**
 *
 * @author likakis
 */
public class GeneticAlgorithm {
```

```
private int populationSize;

private double mutationRate;

private double crossoverRate;

private int elitismCount;

//constractor

public GeneticAlgorithm(int populationSize, double mutationRate, double
crossoverRate, int elitismCount) {

this.populationSize = populationSize;

this.mutationRate = mutationRate;

this.crossoverRate = crossoverRate;

this.elitismCount = elitismCount;

}

//initialize population

public Population initPopulation(int chromosomeLength) {

Population population = new Population(this.populationSize,chromosomeLength);

return population;

}

// Calculate fitness

public double calcFitness(Individual individual) {


int a=0;

int b=0;

int fitness=0;

int nonlinearity=0;

int balance=0;

int d[]=new int []{};

int c[]=new int [individual.getChromosomeLength()];

for (int i = 0; i < individual.getChromosomeLength();i++) {
```

```
c[i]= individual.getGene(i);
if (individual.getGene(i) == 1)
{a ++;}

else if (individual.getGene(i) == 0)
{ b++;}
}
      if (a==b)
      {balance=0;}
      else if (a>b){ balance= -(a-b);}
      else if (a<b){ balance= a-b;}
      Nonlinearity nonlinear= new Nonlinearity();
      d=nonlinear.FWT(c);
      nonlinearity=nonlinear.nonLinearity(d);

fitness=balance+nonlinearity;
individual.setFitness(fitness);
return fitness;
}
//evaluate Population
public void evalPopulation(Population population) {
double populationFitness = 0;
for (Individual individual : population.getIndividuals()) {
populationFitness += calcFitness(individual);
}
population.setPopulationFitness(populationFitness);
}
```

```
//terminate condition
public boolean isTerminationConditionMet(Population population) {
for (Individual individual : population.getIndividuals()) {
if (individual.getFitness() >=232) {
return true;
}
}
return false;
}

//roulete wheel selectParent
public Individual selectParent(Population population) {
// Get individuals
Individual individuals[] = population.getIndividuals();
// Spin roulette wheel
double populationFitness = population.getPopulationFitness();
double rouletteWheelPosition = Math.random() * populationFitness;

// Find parent
double spinWheel = 0;
for (Individual individual : individuals) {
spinWheel += individual.getFitness();
if (spinWheel >= rouletteWheelPosition) {
return individual;
}
}
return individuals[population.size() - 1];
}
```

```
//crossover
public Population crossoverPopulation(Population population) {
// Create new population
Population newPopulation = new Population(population.size());
// Loop over current population by fitness
for (int populationIndex = 0; populationIndex < population.size(); populationIndex++) {
Individual parent1 = population.getFittest(populationIndex);
// Apply crossover to this individual?
if (this.crossoverRate > Math.random() && populationIndex >this.elitismCount) {
// Initialize offspring
Individual offspring = new Individual(parent1.getChromosomeLength());
// Find second parent
Individual parent2 = selectParent(population);
// Loop over genome
for (int geneIndex = 0; geneIndex < parent1.getChromosomeLength(); geneIndex++) {
// Use half of parent1's genes and half ofparent2's genes
if (0.5 > Math.random()) {
offspring.setGene(geneIndex,parent1.getGene(geneIndex));
} else {
offspring.setGene(geneIndex,parent2.getGene(geneIndex));
}
}
// Add offspring to new population
newPopulation.setIndividual(populationIndex,offspring);
} else {
// Add individual to new population without applyingcrossover
newPopulation.setIndividual
```

```
(populationIndex, parent1);
}
}
return newPopulation;
}
//mutation
public Population mutatePopulation(Population population) {
// Initialize new population
Population newPopulation = new Population(this.populationSize);
// Loop over current population by fitness
for (int populationIndex = 0; populationIndex < population.size();populationIndex++) {
Individual individual = population.getFittest(populationIndex);
// Loop over individual's genes
for (int geneIndex = 0; geneIndex < individual.getChromosomeLength(); geneIndex++) {
// Skip mutation if this is an elite individual
if (populationIndex >= this.elitismCount) {
// Does this gene need mutation?
if (this.mutationRate > Math.random()) {
// Get new gene
int newGene = 1;
if (individual.getGene(geneIndex) == 1) {
newGene = 0;
}
// Mutate gene
individual.setGene(geneIndex, newGene);
}
}
}
```

```
// Add individual to population
newPopulation.setIndividual(populationIndex, individual);
}
// Return mutated population
return newPopulation;}}
```

A.1.3 Κλάση Individual

```
package boolean_ga;


/**
 *
 * @author likakis
 */
public class Individual {
private int[] chromosome;
private double fitness = -1;
// Create individual chromosome
public Individual (){}
public Individual(int[] chromosome) {
this.chromosome = chromosome;
}
//Get Chromosome
public int[] getChromosome() {
return this.chromosome;
```

```
}

//Get chromosome length

public int getChromosomeLength() {

return this.chromosome.length;

}

//Set Gene

public void setGene(int offset, int gene) {

this.chromosome[offset] = gene;

}

//Get Gene

public int getGene(int offset) {

return this.chromosome[offset];

}

//Set fitness

public void setFitness(double fitness) {

this.fitness = fitness;

}

//Get fitness

public double getFitness() {

return this.fitness;

}

//Generate Individual

public Individual(int chromosomeLength) {
```

```
this.chromosome = new int[chromosomeLength];

for (int gene = 0; gene < chromosomeLength; gene++) {

if (0.5 < Math.random())

{this.setGene(gene, 1);}

else {this.setGene(gene, 0);}

}

}

//toString

public String toString() {

String output = "";

for (int gene = 0; gene < this.chromosome.length; gene++) {

output += this.chromosome[gene];

}

return output;}}
```

A.1.4 Κλάση Nonlinearity

```
package boolean_ga;

import java.io.BufferedWriter;

import java.io.File;

import java.io.FileWriter;

import java.io.IOException;

import java.lang.Object;

import java.util.Random;
```

```
import java.math.BigInteger;
import java.util.Arrays;
/**
 *
 * @author likakis
 */
public class Nonlinearity extends Object {
    public static int data_size;
    public static int data_sizemo;
    public static int data_sizeo2;
    public static int straddle_width;
    public static int pair;
    public static int left_index;
    public static int right_index;
    public static int a;
    public static int b;
    public static int block;
    public static int blockstart;
    public static int max;
    public static int average;
    public static int min;
    public static int application = 0;
    public static int debug = 0;
```

```
    public static int i;
    public static int ret_val;
    public static int temp;
    public static int difference;
    public static int data[];

//checks if the output is balanced
public boolean Balanced (int [] balance){
   boolean balanced=false;
   int length;
   int assos=0;
   int mhden=0;
   for (int i = 0; i < balance.length;  i++) {
      if(balance[i]==1){assos+=1;}
      else if (balance[i]==0){mhden+=1;}
  }

   if (assos==mhden){balanced=true;}
   else  balanced=false;
   return balanced;
   }

//helpfull method for the method nonlinearity
```

```
public static int[] FWT(int[] data) {
        data_size   = data.length;
        data_sizemo = data_size - 1;
        data_sizeo2 = data_size >>> 1;
          straddle_width = 1;
          blockstart = data_sizemo;
          do {
            left_index = 0;
            blockstart = blockstart >>> 1;
            for (block = blockstart; block >= 0; block--) {
              right_index = left_index + straddle_width;
              for (pair = 0; pair < straddle_width; pair++) {
                a = data[left_index];
                b = data[right_index];
                data[left_index] = a + b;
                data[right_index] = a - b;
                left_index++;
                right_index++;
              }
              left_index = right_index;
            }
            straddle_width = (straddle_width << 1) & data_sizemo;
          } while (straddle_width != 0);
```

```
        data[0] = data_sizeo2 - data[0];
        return data;}

//calculates the nonlinearity of the output
public int nonLinearity(int [] data) {
       data_size   = data.length;
       data_sizeo2 = data_size >>> 1;
       max = 0;
       for (i = 0; i < data_size; i++) {
         temp = data[i];
         temp = (temp >= 0) ? temp : 0 - temp;
         if (temp > max)
            max =temp;
       }
       return data_sizeo2 - max;
    }

//writes the output of the genetic algorithm in text
public void write_genes_to_data (int [] genes){
try{
     String a=Arrays.toString(genes);
          File file =new File("C://Users//likakis//Desktop//pr1_n9_p500.txt");
          if(!file.exists()){
```

```
            file.createNewFile();
          }
          FileWriter fw = new FileWriter(file,true);
      BufferedWriter bw = new BufferedWriter(fw);
      bw.newLine();
      bw.write(a);
          bw.close();
          System.out.println("Data successfully appended at the end of file");

    }catch(IOException ioe){
       System.out.println("Exception occurred:");
           ioe.printStackTrace();
    }
}

//writes some criteria on a text file
public void write_criteria_to_data (int generation,boolean balanced,int nonlinearity,double time,double persent ){
try{
     String a=Integer.toString(generation);
     String b=Boolean.toString(balanced);
     String c=Integer.toString(nonlinearity);
     String d=Double.toString(time);
     String e=Double.toString(persent);
```

```
            File file =new File("C://Users//likakis//Desktop//pr1_n9_p500_2.txt");
            if(!file.exists()){
              file.createNewFile();
            }
            FileWriter fw = new FileWriter(file,true);
        BufferedWriter bw = new BufferedWriter(fw);
        bw.newLine();
        bw.write("Generation:"+ a +  "  Balanced: " +b  + "  Nonlinearity: "  + c  + " Built Time:
" +  d + "  Success Persent:" +e+ "%");
            bw.close();
        System.out.println("Data successfully appended at the end of file");
      }catch(IOException ioe){
         System.out.println("Exception occurred:");
             ioe.printStackTrace();
      }
}
//calculates the success of the output
public double success_persent(int a){
   double b=0;
   double nonlinearity=232;
 b=(double)a/nonlinearity;
 b=Math.round(b*100);
 return b;} }
```

A.1.5 Κλάση Population

```
package boolean_ga;

import java.util.Arrays;

import java.util.Comparator;

import java.util.Random;

/**

 *

 * @author likakis

 */

public class Population {

private Individual population[];

private double populationFitness = -1;

//constractor 1

public Population(int populationSize) {

this.population = new Individual[populationSize];

}

//constractor 2

public Population(int populationSize, int chromosomeLength) {

this.population = new Individual[populationSize];

for (int individualCount = 0; individualCount <populationSize; individualCount++) {

Individual individual = new Individual(chromosomeLength);

this.population[individualCount] = individual;
```

```
}}

//getIndividual

public Individual[] getIndividuals() {

return this.population;}

//getFitness

public Individual getFittest(int offset) {

Arrays.sort(this.population, new Comparator<Individual>() {

@Override

public int compare(Individual o1, Individual o2) {

if (o1.getFitness() > o2.getFitness()) {

return -1;

} else if (o1.getFitness() < o2.getFitness()) {

return 1;

}

return 0;}});

return this.population[offset];

}

//setPopulationFitness

public void setPopulationFitness(double fitness) {

this.populationFitness = fitness;

}

//getPopulationFitness

public double getPopulationFitness() {
```

```
return this.populationFitness;}

//populationsize

public int size() {

return this.population.length;}

//setIndividual

public Individual setIndividual(int offset, Individual individual) {

return population[offset] = individual;}

//getIndividual

public Individual getIndividual(int offset) {

return population[offset];}

//shuffle

public void shuffle() {

Random rnd = new Random();

for (int i = population.length - 1; i > 0; i--) {

int index = rnd.nextInt(i + 1);

Individual a = population[index];

population[index] = population[i];

population[i] = a;}}}
```
[19]

Α.2 Δεύτερο Πρόγραμμα

Α.2.1 Κλάση Algorithm

```
package boolean_ga_2;
```

```
/**
 *
 * @author likakis
 */
public class Algorithm {
 /* GA parameters */
   private static final double uniformRate =0.99;
   private static final double mutationRate =0.3;
   private static final int tournamentSize =2;
   private static final boolean elitism = true;
   /* Public methods */
   // Evolve a population
   public static Population evolvePopulation(Population pop) {
      Population newPopulation = new Population(pop.size(), false);
      // Keep our best individual
      if (elitism) {
         newPopulation.saveIndividual(0, pop.getFittest());
      }
      // Crossover population
      int elitismcount;
      if (elitism) {
         elitismcount = 1;
      } else {
```

```
        elitismcount = 0;}
    // Loop over the population size and create new individuals with
    // crossover
    for (int i = elitismcount; i < pop.size(); i++) {
        Individual indiv1 = tournamentSelection(pop);
        Individual indiv2 = tournamentSelection(pop);
        Individual newIndiv = crossover(indiv1, indiv2);
        newPopulation.saveIndividual(i, newIndiv);
    }
   // Mutate population
    for (int i = elitismcount; i < newPopulation.size(); i++) {
        mutate(newPopulation.getIndividual(i));  }
    return newPopulation;}


// Crossover individuals
private static Individual crossover(Individual indiv1, Individual indiv2) {
    Individual newSol = new Individual();
    double a=Math.random();
    for (int i = 0; i < indiv1.size(); i++) {
        // Crossover
        if (a <= uniformRate) {
            newSol.setGene(i, indiv1.getGene(i));
        } else {
```

```
            newSol.setGene(i, indiv2.getGene(i));}}
    return newSol;}

// Mutate an individual
private static void mutate(Individual indiv) {
    double a=Math.random();
    for (int i = 0; i < indiv.size(); i++) {
        if (a <= mutationRate) {
            // Create random gene
            byte gene = (byte) Math.round(Math.random());
            indiv.setGene(i, gene);
        }
    }
}

// Select individuals for crossover
private static Individual tournamentSelection(Population pop) {
    // Create a tournament population
    Population tournament = new Population(tournamentSize, false);
    // For each place in the tournament get a random individual
    for (int i = 0; i < tournamentSize; i++) {
        int randomId = (int) (Math.random() * pop.size());
        tournament.saveIndividual(i, pop.getIndividual(randomId));
```

```
        }
        // Get the fittest
        Individual fittest = tournament.getFittest();
        return fittest;} }
```

A.2.2 Κλάση Boolean_GA_2

```
package boolean_ga_2;


/**
 *
 * @author likakis
 */
public class Boolean_GA_2  {


   /**
    * @param args the command line arguments
    */
   public static void main(String[] args) {
      // TODO code application logic here
     for (int j=1; j<=5; j++){
      final long startTime = System.nanoTime();
      double  fitness;
      Individual solution =new Individual();
      Nonlinearity no = new Nonlinearity();
      int d[] =new int []{};
```

```
        // Create an initial population
        Population myPop = new Population(500, true);
        int generationCount = 1;
        double seconds2=0;
        while (myPop.getFittest().getFitness()<=FitnessCalc.getMaxFitness()) {
          if (seconds2>=172800.00){break;}
          generationCount++;
          System.out.println("Generation: " + generationCount + " Fittest: " +
myPop.getFittest().getFitness());
          myPop = Algorithm.evolvePopulation(myPop);
          final long duration2 = System.nanoTime() - startTime;
          seconds2 =(double) duration2 / 1000000000.0;
          seconds2=Math.round(seconds2*100);
          seconds2=seconds2/100;
        }
        System.out.println("Solution found!");
        System.out.println("Generation: " + generationCount);
        fitness= myPop.getFittest().getFitness();
        System.out.println("Fitness: " + fitness );
        System.out.println("Genes:");
        solution=myPop.getFittest();
        System.out.println(solution);
        final long duration = System.nanoTime() - startTime;
        double seconds =(double) duration / 1000000000.0;
        seconds=Math.round(seconds*100);
        seconds=seconds/100;
        int c[]=new int [solution.size()];
        for (int i = 0; i < solution.size();  i++) {
```

```
 c[i]= solution.getGene(i);
}
boolean balance= no.Balanced(c);
no.write_genes_to_data(c);
d=no.FWT(c);
int e=no.nonLinearity(d);
double persent=no.success_persent(e);
no.write_criteria_to_data(generationCount,balance,e,seconds,persent);
System.out.println("Balanced:" +balance);
System.out.println("Nonlinearity:" +  e);
System.out.println("Success Persent:" + persent +"%");
System.out.println("Time:" +seconds +" seconds");   }}}
```

A.2.3 Κλάση FitnessCalc

```
package boolean_ga_2;

/**
 *
 * @author likakis
 */
public class FitnessCalc {

   // Calculate inidividuals fittness
static int getFit(Individual individual) {

      int fitness = 0;
      int balance= 0;
      int nonlinearity=0;
```

```
    int a=0;
    int b=0;
    int d[]=new int []{};
    int c[]=new int [individual.size()];

  for (int i = 0; i < individual.size();  i++) {
     c[i]= individual.getGene(i);
      if (individual.getGene(i) == 1) {
          a++;
       }
       else  if (individual.getGene(i) == 0) {
          b++;
       }}
    if (a==b){  balance=0; }
    else if (a>b){ balance= -(a-b);}
    else if (a<b){ balance= a-b;}
   Nonlinearity nonlinear= new Nonlinearity();
    d=nonlinear.FWT(c);
    nonlinearity=nonlinear.nonLinearity(d);
    fitness=balance + nonlinearity;
    return fitness; }
// Get optimum fitness
static int getMaxFitness() {
    int nonlinearity=232;
    int maxFitness =nonlinearity-1;
    return maxFitness; }  }
```

A.2.4 Κλάση Individual

```
package boolean_ga_2;


/**
 *
 * @author likakis
 */
public class Individual {
 static int defaultGeneLength =512;
   private byte[] genes = new byte[defaultGeneLength];
   // Cache
   private int fitness = 0;

   // Create a random individual
   public void generateIndividual() {
      for (int i = 0; i < size(); i++) {
         byte gene = (byte) Math.round(Math.random());
         genes[i] = gene;
      }
   }

   /* Getters and setters */
   // Use this if you want to create individuals with different gene lengths
   public static void setDefaultGeneLength(int length) {
      defaultGeneLength = length;
   }

   public byte getGene(int index) {
```

```
        return genes[index];
    }

    public void setGene(int index, byte value) {
        genes[index] = value;
        fitness = 0;
    }
    public int size() {
        return genes.length;
    }
  public double getFitness() {
      fitness = FitnessCalc.getFit(this);
      return fitness;
    }
    @Override
    public String toString() {
        String geneString = "";
        for (int i = 0; i < size(); i++) {
            geneString += getGene(i);
        }
        return geneString;} }
```

A.2.5 Κλάση Nonlinearity

```
package boolean_ga_2;
import java.io.BufferedWriter;
import java.io.File;
import java.io.FileWriter;
import java.io.IOException;
```

```
import java.lang.Object;
import java.util.Random;
import java.math.BigInteger;
import java.util.Arrays;
/**
 *
 * @author likakis
 */
public class Nonlinearity extends Object {
    public static int data_size;
    public static int data_sizemo;
    public static int data_sizeo2;
    public static int straddle_width;
    public static int pair;
    public static int left_index;
    public static int right_index;
    public static int a;
    public static int b;
    public static int block;
    public static int blockstart;
    public static int max;
    public static int average;
    public static int min;
    public static int application = 0;
    public static int debug = 0;
    public static int i;
    public static int ret_val;
    public static int temp;
```

```
     public static int difference;
     public static int data[];

//checks if the output is balanced
public boolean Balanced (int [] balance){
    boolean balanced=false;
    int length;
    int assos=0;
    int mhden=0;
    for (int i = 0; i < balance.length;  i++) {
       if(balance[i]==1){assos+=1;}
       else if (balance[i]==0){mhden+=1;}
   }

    if (assos==mhden){balanced=true;}
    else  balanced=false;
    return balanced;
    }

//helpfull method for the method nonlinearity
public static int[] FWT(int[] data) {
        data_size   = data.length;
        data_sizemo = data_size - 1;
        data_sizeo2 = data_size >>> 1;
          straddle_width = 1;
          blockstart = data_sizemo;
          do {
            left_index = 0;
```

```
            blockstart = blockstart >>> 1;
            for (block = blockstart; block >= 0; block--) {
              right_index = left_index + straddle_width;
              for (pair = 0; pair < straddle_width; pair++) {
                a = data[left_index];
                b = data[right_index];
                data[left_index] = a + b;
                data[right_index] = a - b;
                left_index++;
                right_index++;
              }
              left_index = right_index;
            }
            straddle_width = (straddle_width << 1) & data_sizemo;
          } while (straddle_width != 0);
          data[0] = data_sizeo2 - data[0];
          return data;}

//calculates the nonlinearity of the output
public int nonLinearity(int [] data) {
        data_size   = data.length;
        data_sizeo2 = data_size >>> 1;
        max = 0;
        for (i = 0; i < data_size; i++) {
          temp = data[i];
          temp = (temp >= 0) ? temp : 0 - temp;
          if (temp > max)
            max =temp;
```

```
        }
        return data_sizeo2 - max;
    }

//writes the output of the genetic algorithm in text
public void write_genes_to_data (int [] genes){
try{
     String a=Arrays.toString(genes);
          File file =new File("C://Users//likakis//Desktop//pr2_n9_p500_t2.txt");
          if(!file.exists()){
            file.createNewFile();
          }
          FileWriter fw = new FileWriter(file,true);
      BufferedWriter bw = new BufferedWriter(fw);
      bw.newLine();
      bw.write(a);
          bw.close();

          System.out.println("Data successfully appended at the end of file");

    }catch(IOException ioe){
       System.out.println("Exception occurred:");
           ioe.printStackTrace();
     }
}

//writes some criteria on a text file
```

```
public void write_criteria_to_data (int generation,boolean balanced,int
nonlinearity,double time,double persent ){
try{
     String a=Integer.toString(generation);
     String b=Boolean.toString(balanced);
     String c=Integer.toString(nonlinearity);
     String d=Double.toString(time);
     String e=Double.toString(persent);
         File file =new File("C://Users//likakis//Desktop//pr2_n9_p500_t2_2.txt");
         if(!file.exists()){
           file.createNewFile();
         }
         FileWriter fw = new FileWriter(file,true);
      BufferedWriter bw = new BufferedWriter(fw);
      bw.newLine();
      bw.write("Generation:"+ a +  "  Balanced: " +b  + "  Nonlinearity: "  + c  + " Built Time:
" +  d + "  Success Persent:" +e+ "%");
         bw.close();
      System.out.println("Data successfully appended at the end of file");
    }catch(IOException ioe){
       System.out.println("Exception occurred:");
          ioe.printStackTrace();
     }
}

//calculates the success of the output
public double success_persent(int a){
   double b=0;
```

```
   double nonlinearity=232;
 b=(double)a/nonlinearity;
 b=Math.round(b*100);
 return b;} }
```

A.2.6 Κλάση Population

```
package boolean_ga_2;


/**
 *
 * @author likakis
 */
public class Population {
  Individual[] individuals;


    /*
     * Constructors
     */
    // Create a population
    public Population(int populationSize, boolean initialise) {
        individuals = new Individual[populationSize];
        // Initialise population
        if (initialise) {
            // Loop and create individuals
```

```
        for (int i = 0; i < size(); i++) {
            Individual newIndividual = new Individual();
            newIndividual.generateIndividual();
            saveIndividual(i, newIndividual);}}}
    /* Getters */
    public Individual getIndividual(int index) {
       return individuals[index];
    }
    public Individual getFittest() {
       Individual fittest = individuals[0];
       // Loop through individuals to find fittest
       for (int i = 0; i < size(); i++) {
          if (fittest.getFitness() <= getIndividual(i).getFitness()) {
   fittest = getIndividual(i);
          }
       }
       return fittest;    }

    /* Public methods */
    // Get population size
    public int size() {
       return individuals.length;
    }
```

```
    // Save individual
    public void saveIndividual(int index, Individual indiv) {
        individuals[index] = indiv; }  }
```

Α.3 Τρίτο Πρόγραμμα

Α.3.1 Κλάση Algorithm

```
package boolean_ga_3;


import java.util.Random;


/**
 *
 * @author likakis
 */
public class Algorithm {
/* GA parameters */
    private static final double uniformRate = 0.99;
    private static final double mutationRate =0.9;
    private static final int tournamentSize = 3;
    private static final boolean elitism = true;


    /* Public methods */
```

```
// Evolve a population
public static Population evolvePopulation(Population pop) {
   Population newPopulation = new Population(pop.size(), false);

   // Keep our best individual
   if (elitism) {
      newPopulation.saveIndividual(0, pop.getFittest());
   }

   // Crossover population
   int elitismcount;
   if (elitism) {
      elitismcount = 1;
   } else {
      elitismcount = 0;
   }
   // Loop over the population size and create new individuals with
   // crossover
   for (int i = elitismcount; i < pop.size(); i++) {
      Individual indiv1 = tournamentSelection(pop);
      Individual indiv2 = tournamentSelection(pop);
      Individual newIndiv = crossover(indiv1, indiv2);
      newPopulation.saveIndividual(i, newIndiv);
```

```
    }

    // Mutate population
    for (int i = elitismcount; i < newPopulation.size(); i++) {
        mutate(newPopulation.getIndividual(i));
    }
    return newPopulation;
}

// Crossover individuals
private static Individual crossover(Individual indiv1, Individual indiv2) {
    Individual newSol = new Individual();
    double a=Math.random();
    for (int i = 0; i < indiv1.size(); i++) {
        // Crossover
        if (a <= uniformRate) {
            newSol.setGene(i, indiv1.getGene(i));
        } else {
            newSol.setGene(i, indiv2.getGene(i));
        }
    }
    return newSol;
}
```

```
//generates a number by a random range
public static int random_range(int min,int max){
Random rnd=new Random();
int n =min + rnd.nextInt(max +1 - min);
return n;}
   // Mutate an individual
   private static void mutate(Individual indiv) {
      double a=Math.random();
      int min=random_range(0,511);
      int max= random_range(min,511);
         if (a <= mutationRate) {
          for (int j=min; j<=max; j++){
            if (indiv.getGene(j) == 1) {
            byte gene=0;
            indiv.setGene(j,gene);
         }
         else  if (indiv.getGene(j) == 0) {
            byte gene2=1;
            indiv.setGene(j,gene2);  }}}}
   // Select individuals for crossover
   private static Individual tournamentSelection(Population pop) {
      // Create a tournament population
      Population tournament = new Population(tournamentSize, false);
```

```
        // For each place in the tournament get a random individual
        for (int i = 0; i < tournamentSize; i++) {
            int randomId = (int) (Math.random() * pop.size());
            tournament.saveIndividual(i, pop.getIndividual(randomId));}
        // Get the fittest
        Individual fittest = tournament.getFittest();
        return fittest;}}
```

A.3.2 Κλάση Boolean_ga_3

```
package boolean_ga_3;


/**
 *
 * @author likakis
 */
public class Boolean_ga_3 {


    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        // TODO code application logic here
      for (int j=0; j<5; j++){
      final long startTime = System.nanoTime();
```

```
 double  fitness;
Individual solution =new Individual();
Nonlinearity no = new Nonlinearity();
int d[] =new int []{};

// Create an initial population
Population myPop = new Population(2000, true);
// Evolve our population until we reach an optimum solution
int generationCount = 1;
double seconds2=0;
while (myPop.getFittest().getFitness()<=FitnessCalc.getMaxFitness()) {
  if (seconds2>=172800.00){break;}
   generationCount++;
   System.out.println("Generation: " + generationCount + " Fittest: " +
myPop.getFittest().getFitness());
   myPop = Algorithm.evolvePopulation(myPop);
   final long duration2 = System.nanoTime() - startTime;
   seconds2 =(double) duration2 / 1000000000.0;
   seconds2=Math.round(seconds2*100);
   seconds2=seconds2/100;
}
System.out.println("Solution found!");
System.out.println("Generation: " + generationCount);
fitness= myPop.getFittest().getFitness();
```

```
System.out.println("Fitness: " + fitness );

System.out.println("Genes:");

solution=myPop.getFittest();

System.out.println(solution);

final long duration = System.nanoTime() - startTime;

double seconds =(double) duration / 1000000000.0;

seconds=Math.round(seconds*100);

seconds=seconds/100;

int c[]=new int [solution.size()];

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

 c[i]= solution.getGene(i);

}

boolean balance= no.Balanced(c);

no.write_genes_to_data(c);

d=no.FWT(c);

int e=no.nonLinearity(d);

double persent=no.success_persent(e);

no.write_criteria_to_data(generationCount,balance,e,seconds,persent);

System.out.println("Balanced:" +balance);

System.out.println("Nonlinearity:" +  e);

System.out.println("Success Persent:" + persent +"%");

System.out.println("Time:" +seconds +" seconds");}}}
```

A.3.3 Κλάση FitnessCalc

```
package boolean_ga_3;


/**
 *
 * @author likakis
 */
public class FitnessCalc {
static int getFit(Individual individual) {


      int fitness = 0;
      int balance= 0;
      int nonlinearity=0;
      int nonlinearity_target=232;
      int difference=0;
      int a=0;
      int b=0;
      int d[]=new int []{};
      int c[]=new int [individual.size()];


    for (int i = 0; i < individual.size();  i++) {
       c[i]= individual.getGene(i);
        if (individual.getGene(i) == 1) {
```

```
            a++;
        }
        else  if (individual.getGene(i) == 0) {
            b++;
        }

      }
    if (a==b){
    balance=0;
    }
    else if (a>b){ balance= -(a-b);}
    else if (a<b){ balance= a-b;}

    Nonlinearity nonlinear= new Nonlinearity();
    d=nonlinear.FWT(c);
    nonlinearity=nonlinear.nonLinearity(d);
    difference= nonlinearity_target-nonlinearity ;
    fitness=balance + nonlinearity -difference;

    return fitness;
}
// Get optimum fitness
static int getMaxFitness() {
```

```
        int nonlinearity=232;
        int maxFitness=nonlinearity-1;
        return maxFitness; } }
```

A.3.4 Κλάση Individual

```
package boolean_ga_3;


/**
 *
 * @author likakis
 */
public class Individual {
static int defaultGeneLength =512;
   private byte[] genes = new byte[defaultGeneLength];
   // Cache
   private int fitness = 0;


   // Create a random individual
   public void generateIndividual() {
      for (int i = 0; i < size(); i++) {
         byte gene = (byte) Math.round(Math.random());
         genes[i] = gene;
      }
```

```
    }

    /* Getters and setters */

    public static void setDefaultGeneLength(int length) {
        defaultGeneLength = length;
    }

    public byte getGene(int index) {
        return genes[index];
    }

    public void setGene(int index, byte value) {
        genes[index] = value;
        fitness = 0;
    }

    /* Public methods */
    public int size() {
        return genes.length;
    }

public double getFitness() {
```

```
    fitness = FitnessCalc.getFit(this);
    return fitness;
  }



  @Override
  public String toString() {
     String geneString = "";
     for (int i = 0; i < size(); i++) {
        geneString += getGene(i);
     }
     return geneString;} }
```

A.3.5 Κλάση Nonlinearity

```
package boolean_ga_3;


import java.io.BufferedWriter;
import java.io.File;
import java.io.FileWriter;
import java.io.IOException;
import java.util.Arrays;


/**
```

```
 *
 * @author likakis
 */
public class Nonlinearity {
 public static int data_size;
     public static int data_sizemo;
     public static int data_sizeo2;
     public static int straddle_width;
     public static int pair;
     public static int left_index;
     public static int right_index;
     public static int a;
     public static int b;
     public static int block;
     public static int blockstart;
     public static int max;
     public static int average;
     public static int min;
     public static int application = 0;
     public static int debug = 0;
     public static int i;
     public static int ret_val;
     public static int temp;
```

```
    public static int difference;
    public static int data[];

//checks if the output is balanced
public boolean Balanced (int [] balance){
   boolean balanced=false;
   int length;
   int assos=0;
   int mhden=0;
   for (int i = 0; i < balance.length;  i++) {
      if(balance[i]==1){assos+=1;}
      else if (balance[i]==0){mhden+=1;}
  }
   if (assos==mhden){balanced=true;}
   else  balanced=false;
   return balanced;
   }

//helpfull method for the method nonlinearity
public static int[] FWT(int[] data) {
      data_size   = data.length;
      data_sizemo = data_size - 1;
      data_sizeo2 = data_size >>> 1;
```

```
        straddle_width = 1;
        blockstart = data_sizemo;
        do {
          left_index = 0;
          blockstart = blockstart >>> 1;
          for (block = blockstart; block >= 0; block--) {
            right_index = left_index + straddle_width;
            for (pair = 0; pair < straddle_width; pair++) {
              a = data[left_index];
              b = data[right_index];
              data[left_index] = a + b;
              data[right_index] = a - b;
              left_index++;
              right_index++;
            }
            left_index = right_index;
          }
          straddle_width = (straddle_width << 1) & data_sizemo;
        } while (straddle_width != 0);
        data[0] = data_sizeo2 - data[0];
        return data;}

//calculates the nonlinearity of the output
```

```
public int nonLinearity(int [] data) {
        data_size   = data.length;
        data_sizeo2 = data_size >>> 1;
        max = 0;
        for (i = 0; i < data_size; i++) {
          temp = data[i];
          temp = (temp >= 0) ? temp : 0 - temp;
          if (temp > max)
            max =temp;
        }
        return data_sizeo2 - max;
      }


//writes the output of the genetic algorithm in text
public void write_genes_to_data (int [] genes){
    try{
     String a=Arrays.toString(genes);
          File file =new File("C://Users//likakis//Desktop//pr3_n9_p2000_t3.txt");
          if(!file.exists()){
            file.createNewFile();
          }
          FileWriter fw = new FileWriter(file,true);
       BufferedWriter bw = new BufferedWriter(fw);
```

```
        bw.newLine();
        bw.write(a);
            bw.close();
            System.out.println("Data successfully appended at the end of file");
       }catch(IOException ioe){
         System.out.println("Exception occurred:");
              ioe.printStackTrace();
       }
}

//writes some criteria on a text file
public void write_criteria_to_data (int generation,boolean balanced,int nonlinearity,double time,double persent ){
try{
      String a=Integer.toString(generation);
      String b=Boolean.toString(balanced);
      String c=Integer.toString(nonlinearity);
      String d=Double.toString(time);
      String e=Double.toString(persent);
            File file =new File("C://Users//likakis//Desktop//pr3_n9_p2000_t3_2.txt");
            if(!file.exists()){
              file.createNewFile();
            }
            FileWriter fw = new FileWriter(file,true);
```

```
        BufferedWriter bw = new BufferedWriter(fw);
        bw.newLine();
        bw.write("Generation:"+ a +  "  Balanced: " +b  + "  Nonlinearity: "  + c  + " Built Time:
" +  d + "  Success Persent:" +e+ "%");
            bw.close();
        System.out.println("Data successfully appended at the end of file");
      }catch(IOException ioe){
        System.out.println("Exception occurred:");
            ioe.printStackTrace();
      }
}

//calculates the success of the output
public double success_persent(int a){
   double b=0;
   double nonlinearity=232;
 b=(double)a/nonlinearity;
 b=Math.round(b*100);
 return b;} }
```

A.3.6 Κλάση Population

```
package boolean_ga_3;

/**
 *
```

```
 * @author likakis
 */
public class Population {
Individual[] individuals;

  /*
   * Constructors
   */
  // Create a population
  public Population(int populationSize, boolean initialise) {
     individuals = new Individual[populationSize];
     // Initialise population
     if (initialise) {
        // Loop and create individuals
        for (int i = 0; i < size(); i++) {
           Individual newIndividual = new Individual();
           newIndividual.generateIndividual();
           saveIndividual(i, newIndividual);
        }
     }
  }

  /* Getters */
```

```
public Individual getIndividual(int index) {
   return individuals[index];
}

public Individual getFittest() {
   Individual fittest = individuals[0];
   // Loop through individuals to find fittest
   for (int i = 0; i < size(); i++) {
      if (fittest.getFitness() <= getIndividual(i).getFitness()) {
      fittest = getIndividual(i);
      }
   }
   return fittest;
}

/* Public methods */
// Get population size
public int size() {
   return individuals.length;
}

// Save individual
public void saveIndividual(int index, Individual indiv) {
```

```
        individuals[index] = indiv;
    }
}
```

Α.4 Τέταρτο Πρόγραμμα

Α.4.1 Κλάση Algorithm

```
package boolean_ga_4;


import java.util.Random;


/**
 *
 * @author likakis
 */
public class Algorithm {
/* GA parameters */
    private static final double uniformRate = 0.99;
    private static final double mutationRate =0.7;
    private static final int tournamentSize = 2;
    private static final boolean elitism = true;


    /* Public methods */


    // Evolve a population
```

```
public static Population evolvePopulation(Population pop) {
    Population newPopulation = new Population(pop.size(), false);

    // Keep our best individual
    if (elitism) {
        newPopulation.saveIndividual(0, pop.getFittest());
    }

    // Crossover population
    int elitismcount;
    if (elitism) {
        elitismcount = 1;
    } else {
        elitismcount = 0;
    }
    // Loop over the population size and create new individuals with
    // crossover
    for (int i = elitismcount; i < pop.size(); i++) {
        Individual indiv1 = tournamentSelection(pop);
        Individual indiv2 = tournamentSelection(pop);
        Individual newIndiv = crossover(indiv1, indiv2);
        newPopulation.saveIndividual(i, newIndiv);
    }
```

```
    // Mutate population
    for (int i = elitismcount; i < newPopulation.size(); i++) {
        mutate(newPopulation.getIndividual(i));
    }

    return newPopulation;
}

// Crossover individuals
private static Individual crossover(Individual indiv1, Individual indiv2) {
    Individual newSol = new Individual();
     double a=Math.random();
    for (int i = 0; i < indiv1.size(); i++) {
        // Crossover
        if (a <= uniformRate) {
            newSol.setGene(i, indiv1.getGene(i));
        } else {
            newSol.setGene(i, indiv2.getGene(i));
        }
    }
    return newSol;
}
```

```
//random number with spesific  range
public static int random_range(int min,int max){
Random rnd=new Random();
int n =min + rnd.nextInt(max +1 - min);
return n;
}
   // Mutate an individual
   private static void mutate(Individual indiv) {
      int min=random_range(0,511);
      double a=Math.random();
      for (int i = 0; i < indiv.size(); i++) {
         if (a <= mutationRate) {
          if(i==min){
            if (indiv.getGene(i) == 1) {
               byte gene=0;
               indiv.setGene(i,gene);
            }
            else  if (indiv.getGene(i) == 0) {
                byte gene2=1;
                indiv.setGene(i,gene2);   }}}}}


   // Select individuals for crossover
   private static Individual tournamentSelection(Population pop) {
```

```
    // Create a tournament population
    Population tournament = new Population(tournamentSize, false);
    // For each place in the tournament get a random individual
    for (int i = 0; i < tournamentSize; i++) {
      int randomId = (int) (Math.random() * pop.size());
      tournament.saveIndividual(i, pop.getIndividual(randomId)); }
    // Get the fittest
    Individual fittest = tournament.getFittest();
    return fittest; } }
```

Α.4.2 Κλάση Boolean_ga4

```
package boolean_ga_4;


/**
 *
 * @author likakis
 */
public class Boolean_ga4 {


  /**
   * @param args the command line arguments
   */
  public static void main(String[] args) {
    // TODO code application logic here
```

```
for (int j=0; j<5; j++){
final long startTime = System.nanoTime();
double  fitness;
double seconds2=0;
Individual solution =new Individual();
Nonlinearity no = new Nonlinearity();
int d[] =new int []{};

// Create an initial population
Population myPop = new Population(1500, true);

// Evolve our population until we reach an optimum solution
int generationCount = 1;
while (myPop.getFittest().getFitness()<=FitnessCalc.getMaxFitness()) {
  if (seconds2>=7200.00){break;}
  generationCount++;
  System.out.println("Generation: " + generationCount + " Fittest: " +
myPop.getFittest().getFitness());
  myPop = Algorithm.evolvePopulation(myPop);
  final long duration2 = System.nanoTime() - startTime;
  seconds2 =(double) duration2 / 1000000000.0;
  seconds2=Math.round(seconds2*100);
  seconds2=seconds2/100;
}
```

```
System.out.println("Solution found!");
System.out.println("Generation: " + generationCount);
fitness= myPop.getFittest().getFitness();
System.out.println("Fitness: " + fitness );
System.out.println("Genes:");
solution=myPop.getFittest();
System.out.println(solution);
final long duration = System.nanoTime() - startTime;
double seconds =(double) duration / 1000000000.0;
seconds=Math.round(seconds*100);
seconds=seconds/100;
int c[]=new int [solution.size()];
for (int i = 0; i < solution.size();  i++) {
 c[i]= solution.getGene(i);
}
boolean balance= no.Balanced(c);
no.write_genes_to_data(c);
d=no.FWT(c);
int e=no.nonLinearity(d);
double persent=no.success_persent(e);
no.write_criteria_to_data(generationCount,balance,e,seconds,persent);
System.out.println("Balanced:" +balance);
System.out.println("Nonlinearity:" +  e);
```

```
System.out.println("Success Persent:" + persent +"%");

System.out.println("Time:" +seconds +" seconds");}}}
```

A.4.3 Κλάση FitnessCalc

```
package boolean_ga_4;


/**

 *

 * @author likakis

 */

public class FitnessCalc {

  // Calculate inidividuals fittness by comparing it to our candidate solution

static int getFit(Individual individual) {


      int fitness = 0;

      int balance= 0;

      int nonlinearity=0;

      int nonlinearity_target=234;

      int difference=0;

      int a=0;

      int b=0;

      int d[]=new int []{};

      int c[]=new int [individual.size()];
```

```
for (int i = 0; i < individual.size();  i++) {
  c[i]= individual.getGene(i);
   if (individual.getGene(i) == 1) {
       a++;
    }
    else  if (individual.getGene(i) == 0) {
       b++;
    }

   }
 if (a==b){
 balance=0;
 }
 else if (a>b){ balance= -(a-b);}
 else if (a<b){ balance= a-b;}

 Nonlinearity nonlinear= new Nonlinearity();
 d=nonlinear.FWT(c);
 nonlinearity=nonlinear.nonLinearity(d);
 difference= nonlinearity_target-nonlinearity ;
 fitness=balance + nonlinearity - difference;
 return fitness;
```

```
    }

    // Get optimum fitness
    static int getMaxFitness() {
        int nonlinearity=234;
        int maxFitness=nonlinearity-1;
        return maxFitness;}}
```

A.4.4 Κλάση Individual

```
package boolean_ga_4;

/**
 *
 * @author likakis
 */
public class Individual {
static int defaultGeneLength =512;
    private byte[] genes = new byte[defaultGeneLength];
    // Cache
    private int fitness = 0;

    // Create a random individual
    public void generateIndividual() {
        for (int i = 0; i < size(); i++) {
```

```
            byte gene = (byte) Math.round(Math.random());
            genes[i] = gene;
        }
    }

    /* Getters and setters */

    public byte getGene(int index) {
        return genes[index];
    }

    public void setGene(int index, byte value) {
        genes[index] = value;
        fitness = 0;
    }

    /* Public methods */
    public int size() {
        return genes.length;
    }

    public double getFitness() {
        fitness = FitnessCalc.getFit(this);
```

```
    return fitness;
  }



  @Override
  public String toString() {
     String geneString = "";
     for (int i = 0; i < size(); i++) {
        geneString += getGene(i);
     }
     return geneString;}}
```

Α.4.5 Κλάση Nonlinearity

```
package boolean_ga_4;
import java.io.File;
import java.io.FileWriter;
import java.io.BufferedWriter;
import java.io.IOException;
import java.util.Arrays;
/**
 *
 * @author likakis
 */
```

```
public class Nonlinearity {

    public static int data_size;

    public static int data_sizemo;

    public static int data_sizeo2;

    public static int straddle_width;

    public static int pair;

    public static int left_index;

    public static int right_index;

    public static int a;

    public static int b;

    public static int block;

    public static int blockstart;

    public static int max;

    public static int average;

    public static int min;

    public static int application = 0;

    public static int debug = 0;

    public static int i;

    public static int ret_val;

    public static int temp;

    public static int difference;

    public static int data[];
```

```
//checks if the output is balanced
public boolean Balanced (int [] balance){
    boolean balanced=false;
    int length;
    int assos=0;
    int mhden=0;



    for (int i = 0; i < balance.length;  i++) {
       if(balance[i]==1){assos+=1;}
       else if (balance[i]==0){mhden+=1;}
   }


    if (assos==mhden){balanced=true;}
    else  balanced=false;
    return balanced;
    }


//helpfull method for the method nonlinearity
public static int[] FWT(int[] data) {
        data_size   = data.length;
        data_sizemo = data_size - 1;
```

```
data_sizeo2 = data_size >>> 1;
  straddle_width = 1;
  blockstart = data_sizemo;
  do {
    left_index = 0;
    blockstart = blockstart >>> 1;
    for (block = blockstart; block >= 0; block--) {
      right_index = left_index + straddle_width;
      for (pair = 0; pair < straddle_width; pair++) {
        a = data[left_index];
        b = data[right_index];
        data[left_index] = a + b;
        data[right_index] = a - b;
        left_index++;
        right_index++;
      }
      left_index = right_index;
    }
    straddle_width = (straddle_width << 1) & data_sizemo;
  } while (straddle_width != 0);
  data[0] = data_sizeo2 - data[0];
  return data;}
```

```
//calculates the nonlinearity of the output
public int nonLinearity(int [] data) {
        data_size   = data.length;
        data_sizeo2 = data_size >>> 1;
        max = 0;
        for (i = 0; i < data_size; i++) {
          temp = data[i];
          temp = (temp >= 0) ? temp : 0 - temp;
          if (temp > max)
            max =temp;
        }
        return data_sizeo2 - max;
    }


//writes the output of the genetic algorithm in text
public void write_genes_to_data (int [] genes){
try{
     String a=Arrays.toString(genes);
          File file =new File("C://Users//likakis//Desktop//pr4_n9_p1500_t2.txt");
          if(!file.exists()){
            file.createNewFile();
          }
          FileWriter fw = new FileWriter(file,true);
```

```
        BufferedWriter bw = new BufferedWriter(fw);
        bw.newLine();
        bw.write(a);
            bw.close();

            System.out.println("Data successfully appended at the end of file");

       }catch(IOException ioe){
         System.out.println("Exception occurred:");
             ioe.printStackTrace();
       }
}

//writes some criteria on a text file
public void write_criteria_to_data (int generation,boolean balanced,int nonlinearity,double time,double persent ){
try{
      String a=Integer.toString(generation);
      String b=Boolean.toString(balanced);
      String c=Integer.toString(nonlinearity);
      String d=Double.toString(time);
      String e=Double.toString(persent);
            File file =new File("C://Users//likakis//Desktop//pr4_n9_p1500_t2_2.txt");
            if(!file.exists()){
```

```
            file.createNewFile();
          }
          FileWriter fw = new FileWriter(file,true);
      BufferedWriter bw = new BufferedWriter(fw);
      bw.newLine();
      bw.write("Generation:"+ a +  "  Balanced: " +b  + "  Nonlinearity: "  + c  + " Built Time:
" +  d + "  Success Persent:" +e+ "%");
          bw.close();
      System.out.println("Data successfully appended at the end of file");
    }catch(IOException ioe){
       System.out.println("Exception occurred:");
           ioe.printStackTrace();
     }
}


//calculates the success of the output
public double success_persent(int a){
   double b=0;
   double nonlinearity=234;
 b=(double)a/nonlinearity;
 b=Math.round(b*100);
 return b;}   }
```

A.4.6 Κλάση Population

```
package boolean_ga_4;
```

```
/**
 *
 * @author likakis
 */
public class Population {
Individual[] individuals;


   /*
    * Constructors
    */
   // Create a population
   public Population(int populationSize, boolean initialise) {
      individuals = new Individual[populationSize];
      // Initialise population
      if (initialise) {
         // Loop and create individuals
         for (int i = 0; i < size(); i++) {
            Individual newIndividual = new Individual();
            newIndividual.generateIndividual();
            saveIndividual(i, newIndividual);
         }
      }
```

```
}

/* Getters */
public Individual getIndividual(int index) {
   return individuals[index];
}

public Individual getFittest() {
   Individual fittest = individuals[0];
   // Loop through individuals to find fittest
   for (int i = 0; i < size(); i++) {
      if (fittest.getFitness() <= getIndividual(i).getFitness()) {
     fittest = getIndividual(i);
      }
   }
   return fittest; }
/* Public methods */
// Get population size
public int size() {
   return individuals.length;}
// Save individual
public void saveIndividual(int index, Individual indiv) {
   individuals[index] = indiv;}}
```

Παράρτημα Β
Αλγεβρική Κανονική Μορφή Εξόδων Τέταρτου Προγράμματος

Στο Παράρτημα αυτό δίνεται η Αλγεβρική Κανονική Μορφή των βέλτιστων συναρτήσεων που κατασκευάστηκαν από τους αλγορίθμους που μελετήσαμε.

Β.1 ANF Για πληθυσμο 500

Β.1.1 Μέγεθος τουρνουά 2

α)f = 1 + x1 + x1 x3 + x2 x3 + x1 x2 x3 + x1 x2 x4 + x1 x5 + x2 x5 + x2 x3 x5 + x1 x2 x4 x5 + x3 x4 x5 + x1 x3 x4 x5 + x1 x2 x3 x4 x5 + x1 x6 + x2 x6 + x1 x2 x6 + x3 x6 + x1 x3 x6 + x4 x6 + x1 x4 x6 + x2 x4 x6 + x1 x2 x4 x6 + x3 x4 x6 + x1 x3 x4 x6 + x2 x3 x4 x6 + x5 x6 + x4 x5 x6 + x1 x2 x4 x5 x6 + x3 x4 x5 x6 + x2 x3 x4 x5 x6 + x1 x2 x3 x4 x5 x6 + x7 + x1 x7 + x1 x2 x7 + x1 x3 x7 + x2 x3 x7 + x4 x7 + x1 x2 x4 x7 + x2 x3 x4 x7 + x1 x2 x3 x4 x7 + x1 x5 x7 + x1 x3 x5 x7 + x2 x3 x5 x7 + x4 x5 x7 + x1 x4 x5 x7 + x2 x4 x5 x7 + x1 x2 x4 x5 x7 + x2 x3 x4 x5 x7 + x1 x2 x3 x4 x5 x7 + x3 x6 x7 + x1 x3 x6 x7 + x2 x3 x6 x7 + x1 x2 x3 x6 x7 + x1 x4 x6 x7 + x3 x4 x6 x7 + x1 x2 x3 x4 x6 x7 + x1 x2 x3 x5 x6 x7 + x1 x2 x4 x5 x6 x7 + x2 x3 x4 x5 x6 x7 + x8 + x1 x8 + x2 x8 + x1 x2 x8 + x3 x8 + x1 x3 x8 + x2 x3 x8 + x1 x2 x3 x8 + x1 x4 x8 + x1 x2 x4 x8 + x3 x4 x8 + x1 x3 x4 x8 + x2 x3 x4 x8 + x1 x2 x3 x4 x8 + x1 x5 x8 + x3 x5 x8 + x1 x3 x5 x8 + x2 x3 x5 x8 + x1 x2 x3 x5 x8 + x2 x4 x5 x8 + x1 x2 x4 x5 x8 + x1 x3 x4 x5 x8 + x1 x2 x3 x4 x5 x8 + x6 x8 + x1 x6 x8 + x2 x6 x8 + x3 x6 x8 + x1 x3 x6 x8 + x1 x2 x3 x6 x8 + x4 x6 x8 + x1 x4 x6 x8 + x2 x4 x6 x8 + x1 x2 x4 x6 x8 + x3 x4 x6 x8 + x5 x6 x8 + x1 x5 x6 x8 + x2 x5 x6 x8 + x1 x2 x5 x6 x8 + x3 x5 x6 x8 + x1 x3 x5 x6 x8 + x1 x2 x3 x5 x6 x8 + x4 x5 x6 x8 + x2 x4 x5 x6 x8 + x1 x3 x4 x5 x6 x8 + x7 x8 + x1 x7 x8 + x1 x2 x7 x8 + x2 x4 x7 x8 + x1 x2 x4 x7 x8 + x3 x4 x7 x8 + x1 x3 x4 x7 x8 + x2 x3 x4 x7 x8 + x1 x2 x3 x4 x7 x8 + x5 x7 x8 + x1 x2 x5 x7 x8 + x4 x5 x7 x8 + x1 x4 x5 x7 x8 + x2 x3 x4 x5 x7 x8 + x1 x2 x3 x4 x5 x7 x8 + x1 x6 x7 x8 + x3 x6 x7 x8 + x1 x2 x3 x6 x7 x8 + x1 x4 x6 x7 x8 + x2 x4 x6 x7 x8 + x1 x3 x4 x6 x7 x8 + x1 x2 x3 x4 x6 x7 x8 + x3 x5 x6 x7 x8 + x1 x3 x5 x6 x7 x8 + x2 x3 x5 x6 x7 x8 + x4 x5 x6 x7 x8 + x2 x4 x5 x6 x7 x8 + x3 x4 x5 x6 x7 x8 + x1 x2 x3 x4 x5 x6 x7 x8 + x1 x9 + x3 x9 + x1 x3 x9 + x1 x2 x3 x9 + x1 x4 x9 + x2 x4 x9 + x3 x4 x9 + x1 x3 x4 x9 + x2 x3 x4 x9 + x1 x2 x3 x4 x9 + x1 x2 x5 x9 + x1 x3 x5 x9 + x1 x2 x3 x5 x9 + x1 x4 x5 x9 + x1 x2 x4 x5 x9 + x2 x3 x4 x5 x9 + x6 x9 + x2 x6 x9 + x2 x3 x6 x9 + x1 x2 x3 x6 x9 + x1 x4 x6 x9 + x2 x4 x6 x9 + x3 x4 x6 x9 + x1 x3 x4 x6 x9 + x1 x5 x6 x9 + x2 x5 x6 x9 + x1 x2 x5 x6 x9 + x3 x5 x6 x9 + x1 x2 x3 x5 x6 x9 + x1 x4 x5 x6 x9 + x1 x2 x4 x5 x6 x9 + x3 x4 x5 x6 x9 + x1 x2 x3 x4 x5 x6 x9 + x7 x9 + x1 x3 x7 x9 + x2 x3 x7 x9 + x1 x2 x3 x7 x9 + x4 x7 x9 + x2 x4 x7 x9 + x1 x2 x4 x7 x9 + x1 x2 x3 x4 x7 x9 + x1 x2 x5 x7 x9 + x3 x5 x7 x9 + x1 x3 x5 x7 x9 + x1 x2 x4 x5 x7 x9 + x3 x4 x5 x7 x9 + x1 x3 x4 x5 x7 x9 + x2 x3 x4 x5 x7 x9 + x1 x2 x3 x4 x5 x7 x9 + x1 x2 x6 x7 x9 + x3 x6 x7 x9 + x1 x3 x6 x7 x9 + x1 x2 x3 x6 x7 x9 + x4 x6 x7 x9 + x1 x4 x6 x7 x9 + x1 x2 x4 x6 x7 x9 + x3

x4 x6 x7 x9 + x1 x3 x4 x6 x7 x9 + x2 x3 x4 x6 x7 x9 + x1 x2 x3 x4 x6 x7 x9 + x1 x5 x6 x7 x9 + x1 x2 x5 x6 x7 x9 + x3 x5 x6 x7 x9 + x1 x3 x5 x6 x7 x9 + x2 x3 x5 x6 x7 x9 + x1 x4 x5 x6 x7 x9 + x2 x4 x5 x6 x7 x9 + x1 x2 x4 x5 x6 x7 x9 + x8 x9 + x1 x8 x9 + x2 x8 x9 + x1 x2 x8 x9 + x1 x3 x8 x9 + x1 x2 x3 x8 x9 + x1 x4 x8 x9 + x2 x4 x8 x9 + x1 x2 x4 x8 x9 + x1 x3 x4 x8 x9 + x2 x3 x4 x8 x9 + x5 x8 x9 + x2 x5 x8 x9 + x1 x2 x3 x5 x8 x9 + x4 x5 x8 x9 + x1 x4 x5 x8 x9 + x3 x4 x5 x8 x9 + x2 x3 x4 x5 x8 x9 + x6 x8 x9 + x3 x6 x8 x9 + x4 x6 x8 x9 + x1 x4 x6 x8 x9 + x1 x2 x4 x6 x8 x9 + x3 x4 x6 x8 x9 + x1 x3 x4 x6 x8 x9 + x1 x2 x3 x4 x6 x8 x9 + x1 x5 x6 x8 x9 + x2 x5 x6 x8 x9 + x1 x2 x5 x6 x8 x9 + x3 x5 x6 x8 x9 + x1 x3 x5 x6 x8 x9 + x2 x3 x5 x6 x8 x9 + x1 x2 x4 x5 x6 x8 x9 + x3 x4 x5 x6 x8 x9 + x1 x3 x4 x5 x6 x8 x9 + x2 x3 x4 x5 x6 x8 x9 + x1 x2 x7 x8 x9 + x3 x7 x8 x9 + x1 x3 x7 x8 x9 + x1 x2 x3 x7 x8 x9 + x1 x2 x4 x7 x8 x9 + x1 x3 x4 x7 x8 x9 + x5 x7 x8 x9 + x1 x5 x7 x8 x9 + x1 x2 x5 x7 x8 x9 + x3 x5 x7 x8 x9 + x2 x3 x5 x7 x8 x9 + x1 x4 x5 x7 x8 x9 + x2 x4 x5 x7 x8 x9 + x1 x2 x4 x5 x7 x8 x9 + x2 x3 x4 x5 x7 x8 x9 + x1 x2 x3 x4 x5 x7 x8 x9 + x1 x6 x7 x8 x9 + x2 x6 x7 x8 x9 + x1 x2 x6 x7 x8 x9 + x3 x6 x7 x8 x9 + x4 x6 x7 x8 x9 + x1 x4 x6 x7 x8 x9 + x2 x4 x6 x7 x8 x9 + x1 x2 x4 x6 x7 x8 x9 + x3 x4 x6 x7 x8 x9 + x1 x2 x3 x4 x6 x7 x8 x9 + x5 x6 x7 x8 x9 + x3 x5 x6 x7 x8 x9 + x2 x3 x5 x6 x7 x8 x9 + x1 x2 x3 x5 x6 x7 x8 x9 + x1 x4 x5 x6 x7 x8 x9 + x1 x2 x4 x5 x6 x7 x8 x9 + x1 x3 x4 x5 x6 x7 x8 x9 + x2 x3 x4 x5 x6 x7 x8 x9

β) f = 1 + x2 + x1 x2 + x3 + x2 x3 + x1 x4 + x1 x2 x4 + x2 x3 x4 + x5 + x2 x5 + x2 x3 x5 + x4 x5 + x1 x2 x4 x5 + x2 x3 x4 x5 + x1 x2 x3 x4 x5 + x1 x6 + x2 x6 + x3 x6 + x1 x3 x6 + x2 x3 x6 + x1 x2 x3 x6 + x4 x6 + x1 x4 x6 + x2 x3 x4 x6 + x5 x6 + x1 x5 x6 + x2 x5 x6 + x3 x5 x6 + x1 x3 x5 x6 + x1 x2 x3 x5 x6 + x1 x4 x5 x6 + x2 x4 x5 x6 + x2 x3 x4 x5 x6 + x1 x2 x3 x4 x5 x6 + x3 x7 + x2 x3 x7 + x2 x4 x7 + x1 x2 x4 x7 + x3 x4 x7 + x1 x2 x3 x4 x7 + x5 x7 + x1 x5 x7 + x2 x5 x7 + x1 x3 x5 x7 + x2 x3 x5 x7 + x4 x5 x7 + x1 x4 x5 x7 + x1 x2 x4 x5 x7 + x1 x3 x4 x5 x7 + x2 x3 x4 x5 x7 + x1 x2 x3 x4 x5 x7 + x6 x7 + x1 x6 x7 + x3 x6 x7 + x1 x3 x6 x7 + x2 x3 x6 x7 + x4 x6 x7 + x1 x2 x4 x6 x7 + x3 x4 x6 x7 + x1 x3 x4 x6 x7 + x2 x3 x4 x6 x7 + x5 x6 x7 + x1 x5 x6 x7 + x1 x2 x5 x6 x7 + x2 x3 x5 x6 x7 + x1 x2 x3 x5 x6 x7 + x3 x4 x5 x6 x7 + x2 x8 + x1 x3 x8 + x1 x4 x8 + x1 x2 x4 x8 + x2 x5 x8 + x3 x5 x8 + x2 x3 x5 x8 + x2 x4 x5 x8 + x1 x2 x4 x5 x8 + x1 x2 x3 x4 x5 x8 + x1 x6 x8 + x1 x2 x6 x8 + x2 x3 x6 x8 + x1 x2 x3 x6 x8 + x4 x6 x8 + x1 x4 x6 x8 + x3 x4 x6 x8 + x2 x3 x4 x6 x8 + x1 x2 x3 x4 x6 x8 + x5 x6 x8 + x1 x2 x5 x6 x8 + x1 x2 x3 x5 x6 x8 + x4 x5 x6 x8 + x1 x2 x4 x5 x6 x8 + x3 x4 x5 x6 x8 + x1 x3 x4 x5 x6 x8 + x2 x3 x4 x5 x6 x8 + x1 x2 x3 x4 x5 x6 x8 + x2 x7 x8 + x2 x3 x7 x8 + x4 x7 x8 + x2 x4 x7 x8 + x1 x2 x4 x7 x8 + x3 x4 x7 x8 + x2 x3 x4 x7 x8 + x5 x7 x8 + x1 x3 x5 x7 x8 + x2 x3 x5 x7 x8 + x4 x5 x7 x8 + x1 x4 x5 x7 x8 + x1 x2 x4 x5 x7 x8 + x3 x4 x5 x7 x8 + x1 x3 x4 x5 x7 x8 + x2 x3 x4 x5 x7 x8 + x1 x6 x7 x8 + x1 x2 x6 x7 x8 + x3 x6 x7 x8 + x1 x3 x6 x7 x8 + x2 x3 x6 x7 x8 + x1 x2 x4 x6 x7 x8 + x1 x2 x3 x4 x6 x7 x8 + x1 x4 x5 x6 x7 x8 + x1 x2 x4 x5 x6 x7 x8 + x3 x4 x5 x6 x7 x8 + x2 x3 x4 x5 x6 x7 x8 + x1 x2 x3 x4 x5 x6 x7 x8 + x1 x9 + x2 x9 + x3 x9 + x1 x3 x9 + x2 x3 x9 + x1 x2 x3 x9 + x1 x4 x9 + x3 x4 x9 + x1 x3 x4 x9 + x5 x9 + x2 x5 x9 + x3 x5 x9 + x2 x3 x5 x9 + x1 x2 x3 x5 x9 + x2 x4 x5 x9 + x1 x2 x4 x5 x9 + x3 x4 x5 x9 + x1 x6 x9 + x3 x6 x9 + x1 x3 x6 x9 + x1 x2 x3 x6 x9 + x1 x4 x6 x9 + x2 x4 x6 x9 + x3 x4 x6 x9 + x2 x3 x4 x6 x9 + x1 x5 x6 x9 + x2 x5 x6 x9 + x1 x3 x5 x6 x9 + x1 x2 x3 x5 x6 x9 + x1 x2 x4 x5 x6 x9 + x3 x4 x5 x6 x9 + x1 x3 x4 x5 x6 x9 + x2 x3 x4 x5 x6 x9 + x1 x2 x3 x4 x5 x6 x9 + x7 x9 + x1 x7 x9 + x3 x7 x9 + x1 x3 x7 x9 + x1 x2 x3 x7 x9 + x4 x7 x9 + x1 x4 x7 x9 + x2 x4 x7 x9 + x2 x3 x4 x7 x9 + x5 x7 x9 + x1 x5 x7 x9 + x1 x2 x5 x7 x9 + x1 x3 x5 x7 x9 + x2 x3 x5 x7 x9 + x1 x4 x5 x7 x9 + x2 x4

x5 x7 x9 + x1 x2 x4 x5 x7 x9 + x3 x4 x5 x7 x9 + x1 x3 x4 x5 x7 x9 + x1 x2 x3 x4 x5 x7 x9 + x6 x7 x9 + x2 x6 x7 x9 + x3 x6 x7 x9 + x2 x3 x6 x7 x9 + x4 x6 x7 x9 + x2 x4 x6 x7 x9 + x2 x3 x4 x6 x7 x9 + x1 x5 x6 x7 x9 + x2 x5 x6 x7 x9 + x1 x2 x5 x6 x7 x9 + x1 x3 x5 x6 x7 x9 + x2 x3 x5 x6 x7 x9 + x2 x4 x5 x6 x7 x9 + x1 x2 x4 x5 x6 x7 x9 + x3 x4 x5 x6 x7 x9 + x1 x3 x4 x5 x6 x7 x9 + x2 x8 x9 + x3 x8 x9 + x1 x3 x8 x9 + x1 x2 x3 x8 x9 + x4 x8 x9 + x2 x4 x8 x9 + x1 x2 x4 x8 x9 + x3 x4 x8 x9 + x1 x3 x4 x8 x9 + x1 x2 x3 x4 x8 x9 + x2 x5 x8 x9 + x1 x2 x5 x8 x9 + x1 x3 x5 x8 x9 + x4 x5 x8 x9 + x1 x2 x4 x5 x8 x9 + x3 x4 x5 x8 x9 + x1 x2 x3 x4 x5 x8 x9 + x6 x8 x9 + x1 x2 x6 x8 x9 + x1 x3 x6 x8 x9 + x1 x2 x3 x6 x8 x9 + x1 x4 x6 x8 x9 + x2 x4 x6 x8 x9 + x3 x4 x6 x8 x9 + x2 x3 x4 x6 x8 x9 + x5 x6 x8 x9 + x1 x5 x6 x8 x9 + x1 x2 x5 x6 x8 x9 + x3 x5 x6 x8 x9 + x1 x3 x5 x6 x8 x9 + x2 x3 x5 x6 x8 x9 + x1 x4 x5 x6 x8 x9 + x1 x3 x4 x5 x6 x8 x9 + x2 x3 x4 x5 x6 x8 x9 + x1 x2 x3 x4 x5 x6 x8 x9 + x1 x3 x7 x8 x9 + x2 x3 x7 x8 x9 + x1 x4 x7 x8 x9 + x2 x4 x7 x8 x9 + x1 x2 x4 x7 x8 x9 + x1 x5 x7 x8 x9 + x2 x5 x7 x8 x9 + x1 x2 x5 x7 x8 x9 + x3 x5 x7 x8 x9 + x1 x3 x5 x7 x8 x9 + x2 x3 x5 x7 x8 x9 + x1 x2 x3 x5 x7 x8 x9 + x2 x4 x5 x7 x8 x9 + x2 x3 x4 x5 x7 x8 x9 + x6 x7 x8 x9 + x3 x6 x7 x8 x9 + x2 x3 x6 x7 x8 x9 + x1 x2 x3 x6 x7 x8 x9 + x4 x6 x7 x8 x9 + x1 x4 x6 x7 x8 x9 + x3 x4 x6 x7 x8 x9 + x1 x3 x4 x6 x7 x8 x9 + x1 x2 x3 x4 x6 x7 x8 x9 + x5 x6 x7 x8 x9 + x2 x3 x5 x6 x7 x8 x9 + x1 x2 x3 x5 x6 x7 x8 x9 + x1 x4 x5 x6 x7 x8 x9 + x1 x3 x4 x5 x6 x7 x8 x9

γ)f = x2 + x1 x2 + x1 x3 + x2 x3 + x2 x4 + x2 x3 x4 + x5 + x1 x5 + x1 x2 x5 + x1 x3 x5 + x1 x4 x5 + x3 x4 x5 + x2 x3 x4 x5 + x1 x2 x3 x4 x5 + x6 + x1 x6 + x3 x6 + x1 x3 x6 + x2 x3 x6 + x1 x4 x6 + x1 x2 x4 x6 + x1 x3 x4 x6 + x2 x3 x4 x6 + x1 x2 x3 x4 x6 + x5 x6 + x2 x5 x6 + x1 x3 x5 x6 + x2 x4 x5 x6 + x3 x4 x5 x6 + x2 x3 x4 x5 x6 + x7 + x3 x7 + x4 x7 + x1 x4 x7 + x2 x4 x7 + x1 x2 x4 x7 + x3 x4 x7 + x1 x3 x4 x7 + x2 x3 x4 x7 + x1 x5 x7 + x1 x2 x5 x7 + x1 x2 x3 x5 x7 + x1 x4 x5 x7 + x2 x4 x5 x7 + x1 x2 x4 x5 x7 + x1 x3 x4 x5 x7 + x1 x2 x3 x4 x5 x7 + x1 x3 x6 x7 + x4 x6 x7 + x1 x2 x4 x6 x7 + x2 x3 x4 x6 x7 + x1 x2 x3 x4 x6 x7 + x2 x5 x6 x7 + x3 x5 x6 x7 + x1 x3 x5 x6 x7 + x1 x2 x3 x5 x6 x7 + x1 x4 x5 x6 x7 + x1 x2 x4 x5 x6 x7 + x2 x3 x4 x5 x6 x7 + x1 x2 x3 x4 x5 x6 x7 + x1 x8 + x1 x2 x3 x8 + x4 x8 + x1 x2 x4 x8 + x1 x3 x4 x8 + x2 x3 x4 x8 + x1 x2 x3 x4 x8 + x5 x8 + x1 x5 x8 + x2 x5 x8 + x4 x5 x8 + x1 x4 x5 x8 + x2 x4 x5 x8 + x3 x4 x5 x8 + x1 x3 x4 x5 x8 + x1 x6 x8 + x2 x6 x8 + x1 x2 x6 x8 + x3 x6 x8 + x1 x3 x6 x8 + x2 x3 x6 x8 + x4 x6 x8 + x1 x4 x6 x8 + x2 x4 x6 x8 + x3 x4 x6 x8 + x1 x5 x6 x8 + x2 x5 x6 x8 + x1 x2 x5 x6 x8 + x3 x5 x6 x8 + x1 x3 x5 x6 x8 + x1 x2 x3 x5 x6 x8 + x1 x4 x5 x6 x8 + x3 x4 x5 x6 x8 + x2 x3 x4 x5 x6 x8 + x7 x8 + x1 x7 x8 + x2 x7 x8 + x3 x7 x8 + x1 x3 x7 x8 + x2 x3 x7 x8 + x1 x2 x3 x7 x8 + x1 x4 x7 x8 + x2 x4 x7 x8 + x1 x2 x4 x7 x8 + x3 x4 x7 x8 + x2 x3 x4 x7 x8 + x1 x5 x7 x8 + x2 x5 x7 x8 + x3 x5 x7 x8 + x4 x5 x7 x8 + x1 x4 x5 x7 x8 + x1 x2 x4 x5 x7 x8 + x1 x6 x7 x8 + x1 x3 x6 x7 x8 + x2 x3 x6 x7 x8 + x1 x2 x3 x6 x7 x8 + x4 x6 x7 x8 + x2 x3 x4 x6 x7 x8 + x1 x5 x6 x7 x8 + x1 x3 x5 x6 x7 x8 + x1 x2 x3 x5 x6 x7 x8 + x4 x5 x6 x7 x8 + x3 x4 x5 x6 x7 x8 + x9 + x2 x9 + x1 x2 x9 + x3 x9 + x1 x3 x9 + x1 x2 x3 x9 + x1 x4 x9 + x1 x2 x4 x9 + x3 x4 x9 + x1 x3 x4 x9 + x1 x5 x9 + x2 x5 x9 + x1 x2 x5 x9 + x3 x5 x9 + x1 x3 x5 x9 + x1 x2 x3 x5 x9 + x4 x5 x9 + x1 x2 x4 x5 x9 + x2 x3 x4 x5 x9 + x1 x2 x3 x4 x5 x9 + x2 x6 x9 + x1 x2 x6 x9 + x3 x6 x9 + x2 x3 x6 x9 + x1 x2 x3 x6 x9 + x4 x6 x9 + x1 x2 x4 x6 x9 + x1 x3 x4 x6 x9 + x1 x2 x3 x4 x6 x9 + x1 x2 x5

x6 x9 + x1 x2 x3 x5 x6 x9 + x4 x5 x6 x9 + x1 x4 x5 x6 x9 + x2 x4 x5 x6 x9 + x3 x4 x5 x6 x9 + x1 x3 x4 x5 x6 x9 + x1 x2 x7 x9 + x1 x2 x3 x7 x9 + x4 x7 x9 + x1 x2 x4 x7 x9 + x1 x3 x4 x7 x9 + x5 x7 x9 + x1 x5 x7 x9 + x1 x2 x5 x7 x9 + x3 x5 x7 x9 + x2 x4 x5 x7 x9 + x1 x2 x4 x5 x7 x9 + x3 x4 x5 x7 x9 + x1 x2 x3 x4 x5 x7 x9 + x6 x7 x9 + x1 x2 x6 x7 x9 + x1 x4 x6 x7 x9 + x2 x4 x6 x7 x9 + x1 x2 x4 x6 x7 x9 + x3 x4 x6 x7 x9 + x1 x3 x4 x6 x7 x9 + x2 x5 x6 x7 x9 + x2 x3 x5 x6 x7 x9 + x1 x2 x3 x5 x6 x7 x9 + x2 x4 x5 x6 x7 x9 + x1 x3 x4 x5 x6 x7 x9 + x1 x8 x9 + x2 x8 x9 + x1 x2 x8 x9 + x3 x8 x9 + x1 x3 x8 x9 + x2 x3 x8 x9 + x1 x2 x3 x8 x9 + x2 x3 x4 x8 x9 + x5 x8 x9 + x2 x5 x8 x9 + x1 x2 x5 x8 x9 + x1 x3 x5 x8 x9 + x1 x2 x3 x5 x8 x9 + x1 x4 x5 x8 x9 + x1 x3 x4 x5 x8 x9 + x2 x3 x4 x5 x8 x9 + x1 x2 x3 x4 x5 x8 x9 + x6 x8 x9 + x3 x6 x8 x9 + x1 x4 x6 x8 x9 + x2 x4 x6 x8 x9 + x1 x3 x4 x6 x8 x9 + x5 x6 x8 x9 + x1 x5 x6 x8 x9 + x2 x5 x6 x8 x9 + x2 x3 x5 x6 x8 x9 + x1 x2 x3 x5 x6 x8 x9 + x4 x5 x6 x8 x9 + x7 x8 x9 + x1 x7 x8 x9 + x2 x7 x8 x9 + x1 x2 x7 x8 x9 + x2 x3 x7 x8 x9 + x4 x7 x8 x9 + x2 x4 x7 x8 x9 + x3 x4 x7 x8 x9 + x2 x3 x5 x7 x8 x9 + x4 x5 x7 x8 x9 + x1 x4 x5 x7 x8 x9 + x2 x4 x5 x7 x8 x9 + x1 x2 x4 x5 x7 x8 x9 + x1 x3 x4 x5 x7 x8 x9 + x2 x3 x4 x5 x7 x8 x9 + x6 x7 x8 x9 + x1 x6 x7 x8 x9 + x2 x6 x7 x8 x9 + x1 x2 x6 x7 x8 x9 + x1 x2 x4 x6 x7 x8 x9 + x1 x2 x3 x4 x6 x7 x8 x9 + x2 x5 x6 x7 x8 x9 + x3 x5 x6 x7 x8 x9 + x1 x3 x5 x6 x7 x8 x9 + x1 x2 x3 x5 x6 x7 x8 x9 + x4 x5 x6 x7 x8 x9 + x1 x4 x5 x6 x7 x8 x9 + x2 x4 x5 x6 x7 x8 x9 + x1 x2 x4 x5 x6 x7 x8 x9 + x1 x3 x4 x5 x6 x7 x8 x9

Β.1.2 Μέγεθος τουρνουά 3

α)f = x1 + x2 x3 + x1 x2 x3 + x1 x4 + x1 x2 x4 + x3 x4 + x1 x3 x4 + x2 x3 x4 + x1 x2 x3 x4 + x5 + x3 x5 + x2 x3 x5 + x4 x5 + x2 x4 x5 + x3 x4 x5 + x1 x3 x4 x5 + x2 x3 x4 x5 + x1 x6 + x2 x6 + x1 x3 x6 + x2 x3 x6 + x1 x4 x6 + x2 x4 x6 + x1 x3 x4 x6 + x2 x3 x4 x6 + x1 x2 x3 x4 x6 + x1 x5 x6 + x2 x5 x6 + x1 x2 x5 x6 + x3 x5 x6 + x4 x5 x6 + x1 x2 x4 x5 x6 + x3 x4 x5 x6 + x2 x3 x4 x5 x6 + x7 + x1 x7 + x3 x7 + x1 x2 x3 x7 + x1 x4 x7 + x2 x4 x7 + x1 x2 x4 x7 + x1 x3 x4 x7 + x1 x2 x3 x4 x7 + x1 x5 x7 + x1 x2 x5 x7 + x1 x4 x5 x7 + x2 x4 x5 x7 + x3 x4 x5 x7 + x1 x3 x4 x5 x7 + x2 x3 x4 x5 x7 + x1 x2 x3 x4 x5 x7 + x6 x7 + x1 x6 x7 + x1 x2 x3 x6 x7 + x1 x4 x6 x7 + x2 x4 x6 x7 + x1 x2 x4 x6 x7 + x3 x4 x6 x7 + x5 x6 x7 + x2 x5 x6 x7 + x2 x3 x5 x6 x7 + x1 x2 x3 x5 x6 x7 + x4 x5 x6 x7 + x2 x4 x5 x6 x7 + x1 x2 x4 x5 x6 x7 + x2 x3 x4 x5 x6 x7 + x1 x2 x3 x4 x5 x6 x7 + x8 + x2 x8 + x1 x2 x8 + x3 x8 + x1 x3 x8 + x2 x4 x8 + x1 x2 x4 x8 + x2 x3 x4 x8 + x1 x2 x3 x4 x8 + x1 x3 x5 x8 + x2 x3 x5 x8 + x1 x2 x3 x5 x8 + x2 x4 x5 x8 + x1 x3 x4 x5 x8 + x1 x6 x8 + x2 x6 x8 + x1 x2 x6 x8 + x1 x3 x6 x8 + x2 x3 x6 x8 + x4 x6 x8 + x1 x4 x6 x8 + x1 x2 x4 x6 x8 + x2 x3 x4 x6 x8 + x1 x2 x3 x4 x6 x8 + x5 x6 x8 + x1 x2 x5 x6 x8 + x1 x3 x5 x6 x8 + x2 x3 x5 x6 x8 + x1 x2 x3 x5 x6 x8 + x4 x5 x6 x8 + x2 x4 x5 x6 x8 + x1 x2 x4 x5 x6 x8 + x1 x3 x4 x5 x6 x8 + x2 x3 x4 x5 x6 x8 + x2 x7 x8 + x1 x2 x7 x8 + x3 x7 x8 + x1 x2 x3 x7 x8 + x4 x7 x8 + x2 x4 x7 x8 + x1 x2 x3 x4 x7 x8 + x5 x7 x8 + x2 x5 x7 x8 + x1

x2 x5 x7 x8 + x1 x3 x5 x7 x8 + x2 x3 x5 x7 x8 + x1 x2 x3 x5 x7 x8 + x4 x5 x7 x8 + x1 x4 x5 x7 x8 + x2 x4 x5 x7 x8 + x1 x2 x4 x5 x7 x8 + x3 x4 x5 x7 x8 + x2 x3 x4 x5 x7 x8 + x1 x2 x3 x4 x5 x7 x8 + x2 x6 x7 x8 + x1 x2 x3 x6 x7 x8 + x4 x6 x7 x8 + x1 x4 x6 x7 x8 + x3 x4 x6 x7 x8 + x1 x3 x4 x6 x7 x8 + x1 x2 x3 x4 x6 x7 x8 + x1 x5 x6 x7 x8 + x2 x5 x6 x7 x8 + x1 x2 x5 x6 x7 x8 + x3 x5 x6 x7 x8 + x2 x3 x5 x6 x7 x8 + x4 x5 x6 x7 x8 + x1 x2 x4 x5 x6 x7 x8 + x3 x4 x5 x6 x7 x8 + x1 x2 x3 x4 x5 x6 x7 x8 + x9 + x2 x9 + x1 x2 x9 + x1 x3 x9 + x1 x2 x3 x9 + x4 x9 + x2 x4 x9 + x2 x3 x4 x9 + x5 x9 + x1 x3 x5 x9 + x2 x3 x5 x9 + x1 x2 x3 x5 x9 + x1 x4 x5 x9 + x3 x4 x5 x9 + x1 x3 x4 x5 x9 + x2 x3 x4 x5 x9 + x1 x2 x3 x4 x5 x9 + x1 x6 x9 + x1 x2 x6 x9 + x2 x3 x6 x9 + x1 x4 x6 x9 + x2 x4 x6 x9 + x1 x3 x4 x6 x9 + x2 x3 x4 x6 x9 + x1 x2 x3 x4 x6 x9 + x1 x5 x6 x9 + x2 x5 x6 x9 + x3 x5 x6 x9 + x1 x2 x3 x5 x6 x9 + x1 x4 x5 x6 x9 + x3 x4 x5 x6 x9 + x1 x3 x4 x5 x6 x9 + x2 x3 x4 x5 x6 x9 + x1 x2 x3 x4 x5 x6 x9 + x2 x7 x9 + x1 x2 x7 x9 + x3 x7 x9 + x1 x4 x7 x9 + x2 x4 x7 x9 + x1 x3 x4 x7 x9 + x2 x3 x4 x7 x9 + x1 x2 x3 x4 x7 x9 + x5 x7 x9 + x1 x5 x7 x9 + x1 x2 x5 x7 x9 + x1 x3 x5 x7 x9 + x2 x3 x5 x7 x9 + x1 x2 x3 x5 x7 x9 + x4 x5 x7 x9 + x2 x3 x4 x5 x7 x9 + x1 x6 x7 x9 + x2 x3 x6 x7 x9 + x1 x4 x6 x7 x9 + x1 x3 x4 x6 x7 x9 + x2 x3 x4 x6 x7 x9 + x1 x5 x6 x7 x9 + x2 x5 x6 x7 x9 + x2 x3 x5 x6 x7 x9 + x1 x2 x3 x5 x6 x7 x9 + x1 x4 x5 x6 x7 x9 + x8 x9 + x2 x8 x9 + x3 x8 x9 + x1 x3 x8 x9 + x4 x8 x9 + x2 x4 x8 x9 + x1 x3 x4 x8 x9 + x3 x5 x8 x9 + x1 x3 x5 x8 x9 + x2 x3 x5 x8 x9 + x1 x2 x3 x5 x8 x9 + x2 x4 x5 x8 x9 + x3 x4 x5 x8 x9 + x1 x3 x4 x5 x8 x9 + x1 x2 x3 x4 x5 x8 x9 + x6 x8 x9 + x1 x6 x8 x9 + x2 x6 x8 x9 + x1 x2 x6 x8 x9 + x3 x6 x8 x9 + x2 x3 x6 x8 x9 + x1 x2 x3 x6 x8 x9 + x4 x6 x8 x9 + x1 x4 x6 x8 x9 + x2 x4 x6 x8 x9 + x1 x2 x4 x6 x8 x9 + x3 x4 x6 x8 x9 + x1 x3 x4 x6 x8 x9 + x2 x3 x4 x6 x8 x9 + x5 x6 x8 x9 + x3 x5 x6 x8 x9 + x2 x3 x5 x6 x8 x9 + x4 x5 x6 x8 x9 + x1 x4 x5 x6 x8 x9 + x2 x4 x5 x6 x8 x9 + x1 x3 x4 x5 x6 x8 x9 + x1 x2 x3 x4 x5 x6 x8 x9 + x2 x7 x8 x9 + x1 x2 x7 x8 x9 + x1 x3 x7 x8 x9 + x1 x2 x4 x7 x8 x9 + x1 x5 x7 x8 x9 + x2 x5 x7 x8 x9 + x1 x2 x5 x7 x8 x9 + x2 x3 x5 x7 x8 x9 + x1 x2 x3 x5 x7 x8 x9 + x4 x5 x7 x8 x9 + x1 x4 x5 x7 x8 x9 + x1 x2 x4 x5 x7 x8 x9 + x3 x4 x5 x7 x8 x9 + x1 x3 x4 x5 x7 x8 x9 + x2 x3 x4 x5 x7 x8 x9 + x6 x7 x8 x9 + x1 x2 x6 x7 x8 x9 + x4 x6 x7 x8 x9 + x1 x4 x6 x7 x8 x9 + x2 x5 x6 x7 x8 x9 + x3 x5 x6 x7 x8 x9 + x1 x3 x5 x6 x7 x8 x9 + x2 x3 x5 x6 x7 x8 x9 + x4 x5 x6 x7 x8 x9 + x1 x4 x5 x6 x7 x8 x9 + x2 x4 x5 x6 x7 x8 x9 + x3 x4 x5 x6 x7 x8 x9 + x1 x3 x4 x5 x6 x7 x8 x9 + x2 x3 x4 x5 x6 x7 x8 x9

B.2 ANF Για πληθυσμο 1000

α)f = 1 + x1 + x2 + x1 x2 x3 + x4 + x1 x4 + x2 x4 + x1 x2 x4 + x2 x3 x4 + x1 x2 x3 x4 + x1 x3 x5 + x2 x3 x5 + x1 x2 x3 x5 + x4 x5 + x2 x4 x5 + x3 x4 x5 + x1 x3 x4 x5 + x2 x3 x4 x5 + x1 x2 x3 x4 x5 + x1 x6 + x2 x6 + x1 x3 x6 + x2 x3 x6 + x1 x2 x3 x6 + x4 x6 + x1 x4 x6 + x1 x3 x4 x6 + x2 x3 x4 x6 + x5 x6 + x3 x5 x6 + x1 x3 x5 x6 + x2 x3 x5 x6 + x4 x5 x6 + x2 x4 x5 x6 + x1 x2 x4 x5 x6 + x2 x3 x4 x5 x6 + x1 x2 x3 x4 x5 x6 + x7 + x1 x2 x7 + x3 x7 + x1 x3 x7 + x1 x4 x7 + x2 x4 x7 + x1 x2 x4 x7 + x1 x3 x4 x7 + x2 x3 x4 x7 + x1 x2 x3 x4 x7 + x5 x7 + x2 x5 x7 + x1 x2 x5 x7 + x3 x5 x7 + x1 x3 x5 x7 + x1 x2 x3 x5 x7 + x4 x5 x7 + x2 x3 x4 x5 x7 + x6 x7 + x2 x6 x7 + x3 x6 x7 + x2 x3 x6 x7 + x1 x2 x3 x6 x7 + x3 x4 x6 x7 + x1 x3 x4 x6 x7 + x2 x3 x4 x6 x7 + x1 x2 x3 x4 x6 x7 + x1 x5 x6 x7 + x3 x5 x6 x7 + x2 x3 x5 x6 x7 + x4 x5 x6 x7 + x1 x4 x5 x6 x7 + x2 x4 x5 x6 x7 + x1 x3 x4 x5 x6 x7 + x2 x3 x4 x5 x6 x7 + x1 x2 x8 + x2 x3 x8 + x4 x8 + x1 x2 x4 x8 + x3 x4 x8 + x2 x3 x4 x8 + x2 x5 x8 + x1 x2 x5 x8 + x3 x5 x8 + x2 x3 x5 x8 + x4 x5 x8 + x1 x4 x5 x8 + x2 x4 x5 x8 + x3 x4 x5 x8 + x1 x6 x8 + x3 x6 x8 + x1 x3 x6 x8 + x2 x3 x6 x8 + x1 x2 x3 x6 x8 + x3 x4 x6 x8 + x2 x3 x4 x6 x8 + x5 x6 x8 + x1 x2 x5 x6 x8 + x1 x3 x5 x6 x8 + x2 x3 x5 x6 x8 + x2 x4 x5 x6 x8 + x3 x4 x5 x6 x8 + x1 x3 x4 x5 x6 x8 + x2 x3 x4 x5 x6 x8 + x1 x2 x3 x4 x5 x6 x8 + x1 x3 x7 x8 + x2 x3 x7 x8 + x4 x7 x8 + x3 x4 x7 x8 + x1 x3 x4 x7 x8 + x2 x3 x4 x7 x8 + x1 x2 x3 x4 x7 x8 + x1 x5 x7 x8 + x2 x5 x7 x8 + x2 x3 x5 x7 x8 + x1 x2 x4 x5 x7 x8 + x1 x3 x4 x5 x7 x8 + x1 x2 x3 x4 x5 x7 x8 + x6 x7 x8 + x1 x6 x7 x8 + x2 x6 x7 x8 + x2 x3 x6 x7 x8 + x1 x2 x3 x6 x7 x8 + x4 x6 x7 x8 + x2 x4 x6 x7 x8 + x3 x4 x6 x7 x8 + x1 x3 x4 x6 x7 x8 + x2 x3 x4 x6 x7 x8 + x1 x2 x3 x4 x6 x7 x8 + x5 x6 x7 x8 + x1 x5 x6 x7 x8 + x2 x5 x6 x7 x8 + x1 x3 x5 x6 x7 x8 + x4 x5 x6 x7 x8 + x2 x4 x5 x6 x7 x8 + x1 x3 x4 x5 x6 x7 x8 + x9 + x1 x9 + x2 x9 + x1 x2 x9 + x1 x3 x9 + x4 x9 + x1 x4 x9 + x2 x4 x9 + x1 x3 x4 x9 + x1 x2 x3 x4 x9 + x1 x2 x5 x9 + x4 x5 x9 + x1 x4 x5 x9 + x2 x4 x5 x9 + x1 x2 x4 x5 x9 + x3 x4 x5 x9 + x2 x3 x4 x5 x9 + x1 x2 x6 x9 + x1 x3 x6 x9 + x2 x3 x6 x9 + x1 x2 x3 x6 x9 + x1 x4 x6 x9 + x2 x4 x6 x9 + x1 x2 x3 x4 x6 x9 + x5 x6 x9 + x3 x5 x6 x9 + x1 x3 x5 x6 x9 + x2 x3 x5 x6 x9 + x1 x2 x3 x5 x6 x9 + x4 x5 x6 x9 + x1 x4 x5 x6 x9 + x2 x4 x5 x6 x9 + x1 x2 x4 x5 x6 x9 + x1 x3 x4 x5 x6 x9 + x2 x3 x4 x5 x6 x9 + x1 x7 x9 + x2 x7 x9 + x1 x2 x7 x9 + x3 x7 x9 + x2 x3 x7 x9 + x1 x2 x3 x7 x9 + x1 x2 x4 x7 x9 + x1 x5 x7 x9 + x2 x5 x7 x9 + x2 x3 x5 x7 x9 + x1 x2 x3 x5 x7 x9 +

x1 x4 x5 x7 x9 + x2 x4 x5 x7 x9 + x1 x3 x4 x5 x7 x9 + x1 x2 x3 x4 x5 x7 x9 + x1 x6 x7 x9 + x3 x6 x7 x9 + x1 x3 x6 x7 x9 + x1 x2 x3 x6 x7 x9 + x4 x6 x7 x9 + x1 x4 x6 x7 x9 + x2 x4 x6 x7 x9 + x1 x2 x3 x4 x6 x7 x9 + x1 x5 x6 x7 x9 + x1 x2 x5 x6 x7 x9 + x3 x5 x6 x7 x9 + x2 x3 x5 x6 x7 x9 + x4 x5 x6 x7 x9 + x1 x4 x5 x6 x7 x9 + x3 x4 x5 x6 x7 x9 + x2 x3 x4 x5 x6 x7 x9 + x1 x2 x8 x9 + x3 x8 x9 + x2 x3 x8 x9 + x1 x2 x3 x8 x9 + x2 x4 x8 x9 + x2 x5 x8 x9 + x1 x2 x5 x8 x9 + x3 x5 x8 x9 + x1 x3 x5 x8 x9 + x1 x2 x3 x5 x8 x9 + x1 x4 x5 x8 x9 + x1 x2 x4 x5 x8 x9 + x2 x3 x4 x5 x8 x9 + x2 x3 x6 x8 x9 + x1 x2 x3 x6 x8 x9 + x1 x2 x4 x6 x8 x9 + x1 x3 x4 x6 x8 x9 + x2 x5 x6 x8 x9 + x1 x2 x5 x6 x8 x9 + x3 x5 x6 x8 x9 + x2 x3 x5 x6 x8 x9 + x4 x5 x6 x8 x9 + x2 x4 x5 x6 x8 x9 + x3 x4 x5 x6 x8 x9 + x1 x3 x4 x5 x6 x8 x9 + x2 x3 x4 x5 x6 x8 x9 + x7 x8 x9 + x1 x7 x8 x9 + x2 x7 x8 x9 + x1 x2 x7 x8 x9 + x3 x7 x8 x9 + x1 x4 x7 x8 x9 + x1 x2 x4 x7 x8 x9 + x1 x2 x3 x4 x7 x8 x9 + x1 x2 x3 x5 x7 x8 x9 + x4 x5 x7 x8 x9 + x1 x2 x4 x5 x7 x8 x9 + x3 x4 x5 x7 x8 x9 + x1 x3 x4 x5 x7 x8 x9 + x1 x2 x3 x4 x5 x7 x8 x9 + x6 x7 x8 x9 + x1 x6 x7 x8 x9 + x2 x6 x7 x8 x9 + x1 x2 x6 x7 x8 x9 + x4 x6 x7 x8 x9 + x2 x4 x6 x7 x8 x9 + x3 x4 x6 x7 x8 x9 + x1 x3 x4 x6 x7 x8 x9 + x2 x3 x4 x6 x7 x8 x9 + x1 x2 x5 x6 x7 x8 x9 + x3 x5 x6 x7 x8 x9 + x1 x3 x5 x6 x7 x8 x9 + x2 x3 x5 x6 x7 x8 x9 + x4 x5 x6 x7 x8 x9 + x2 x4 x5 x6 x7 x8 x9 + x3 x4 x5 x6 x7 x8 x9 + x1 x3 x4 x5 x6 x7 x8 x9 + x2 x3 x4 x5 x6 x7 x8 x9

β) f = 1 + x1 + x2 x3 + x1 x2 x3 + x4 + x1 x4 + x2 x4 + x3 x4 + x1 x3 x4 + x2 x3 x4 + x5 + x2 x5 + x1 x2 x5 + x2 x3 x5 + x1 x4 x5 + x2 x4 x5 + x3 x4 x5 + x1 x3 x4 x5 + x2 x3 x4 x5 + x1 x2 x3 x4 x5 + x1 x6 + x2 x6 + x2 x3 x6 + x1 x4 x6 + x2 x4 x6 + x1 x2 x4 x6 + x1 x2 x3 x4 x6 + x2 x5 x6 + x3 x5 x6 + x1 x4 x5 x6 + x3 x4 x5 x6 + x7 + x1 x2 x7 + x1 x3 x7 + x2 x3 x7 + x1 x2 x3 x7 + x1 x4 x7 + x1 x3 x4 x7 + x2 x3 x4 x7 + x1 x2 x3 x4 x7 + x1 x5 x7 + x2 x5 x7 + x3 x5 x7 + x1 x2 x3 x5 x7 + x4 x5 x7 + x1 x4 x5 x7 + x2 x4 x5 x7 + x1 x3 x4 x5 x7 + x2 x3 x4 x5 x7 + x6 x7 + x1 x6 x7 + x2 x6 x7 + x3 x6 x7 + x1 x3 x6 x7 + x1 x2 x3 x6 x7 + x1 x4 x6 x7 + x2 x4 x6 x7 + x3 x4 x6 x7 + x2 x3 x4 x6 x7 + x1 x2 x3 x4 x6 x7 + x5 x6 x7 + x3 x5 x6 x7 + x1 x3 x5 x6 x7 + x1 x2 x3 x5 x6 x7 + x3 x4 x5 x6 x7 + x2 x3 x4 x5 x6 x7 + x1 x8 + x1 x2 x8 + x3 x8 + x1 x3 x8 + x2 x3 x8 + x1 x2 x3 x8 + x2 x3 x4 x8 + x5 x8 + x1 x5 x8 + x1 x2 x5 x8 + x3 x5 x8 + x2 x3 x5 x8 + x2 x4 x5 x8 + x1 x2 x4 x5 x8 + x2 x3 x4 x5 x8 + x6 x8 + x1 x6 x8 + x4 x6 x8 + x1 x4 x6 x8 + x2 x4 x6 x8 + x3 x4 x6 x8 + x2 x3 x4 x6 x8 + x1 x5 x6 x8 + x1 x2 x5 x6 x8 + x3 x5 x6 x8 + x1 x2 x3 x5 x6 x8 + x1 x2 x3 x4 x5 x6 x8 + x1 x2 x7 x8 + x3 x7 x8 + x1 x3 x7 x8 + x2 x4 x7 x8 + x1 x2 x4 x7 x8 + x2 x5 x7 x8 + x1 x2 x5 x7 x8 + x1 x3 x5 x7 x8 + x2 x3 x5 x7 x8 + x1 x2 x3 x5 x7 x8 + x2 x4 x5 x7 x8 + x1 x2 x4 x5 x7 x8 + x3 x4 x5 x7 x8 + x2 x3 x4 x5 x7 x8 + x1 x2 x3 x4 x5 x7 x8 + x6 x7 x8 + x2 x6 x7 x8 + x1 x2 x6 x7 x8 + x3 x6 x7

x8 + x2 x3 x6 x7 x8 + x4 x6 x7 x8 + x1 x4 x6 x7 x8 + x2 x4 x6 x7 x8 + x1 x2 x4 x6 x7 x8 + x1 x3 x4 x6 x7 x8 + x1 x2 x3 x4 x6 x7 x8 + x5 x6 x7 x8 + x1 x5 x6 x7 x8 + x2 x5 x6 x7 x8 + x1 x2 x5 x6 x7 x8 + x3 x5 x6 x7 x8 + x1 x3 x5 x6 x7 x8 + x2 x3 x5 x6 x7 x8 + x1 x2 x3 x5 x6 x7 x8 + x2 x4 x5 x6 x7 x8 + x1 x2 x4 x5 x6 x7 x8 + x3 x4 x5 x6 x7 x8 + x9 + x2 x9 + x1 x2 x3 x9 + x4 x9 + x1 x2 x4 x9 + x2 x3 x4 x9 + x2 x5 x9 + x1 x2 x5 x9 + x3 x5 x9 + x2 x3 x5 x9 + x4 x5 x9 + x3 x4 x5 x9 + x1 x2 x3 x4 x5 x9 + x3 x6 x9 + x1 x3 x6 x9 + x1 x2 x3 x6 x9 + x4 x6 x9 + x3 x4 x6 x9 + x1 x3 x4 x6 x9 + x1 x2 x3 x4 x6 x9 + x5 x6 x9 + x1 x5 x6 x9 + x2 x5 x6 x9 + x1 x2 x5 x6 x9 + x4 x5 x6 x9 + x1 x4 x5 x6 x9 + x2 x4 x5 x6 x9 + x3 x4 x5 x6 x9 + x1 x3 x4 x5 x6 x9 + x2 x3 x4 x5 x6 x9 + x7 x9 + x1 x7 x9 + x1 x2 x7 x9 + x3 x7 x9 + x1 x3 x7 x9 + x2 x3 x7 x9 + x1 x4 x7 x9 + x1 x2 x4 x7 x9 + x2 x3 x4 x7 x9 + x5 x7 x9 + x1 x5 x7 x9 + x2 x5 x7 x9 + x3 x5 x7 x9 + x1 x2 x3 x5 x7 x9 + x2 x4 x5 x7 x9 + x3 x4 x5 x7 x9 + x1 x3 x4 x5 x7 x9 + x6 x7 x9 + x1 x2 x6 x7 x9 + x3 x6 x7 x9 + x1 x3 x6 x7 x9 + x4 x6 x7 x9 + x1 x2 x4 x6 x7 x9 + x3 x4 x6 x7 x9 + x1 x3 x4 x6 x7 x9 + x1 x2 x3 x4 x6 x7 x9 + x1 x5 x6 x7 x9 + x3 x5 x6 x7 x9 + x1 x3 x5 x6 x7 x9 + x2 x4 x5 x6 x7 x9 + x1 x2 x4 x5 x6 x7 x9 + x1 x3 x4 x5 x6 x7 x9 + x1 x2 x3 x4 x5 x6 x7 x9 + x1 x2 x8 x9 + x3 x8 x9 + x2 x3 x8 x9 + x1 x4 x8 x9 + x2 x4 x8 x9 + x1 x2 x4 x8 x9 + x5 x8 x9 + x1 x5 x8 x9 + x1 x2 x5 x8 x9 + x3 x5 x8 x9 + x1 x3 x5 x8 x9 + x4 x5 x8 x9 + x1 x4 x5 x8 x9 + x3 x4 x5 x8 x9 + x2 x3 x4 x5 x8 x9 + x1 x2 x3 x4 x5 x8 x9 + x1 x2 x6 x8 x9 + x1 x3 x6 x8 x9 + x1 x2 x3 x6 x8 x9 + x4 x6 x8 x9 + x3 x4 x6 x8 x9 + x1 x3 x4 x6 x8 x9 + x2 x3 x4 x6 x8 x9 + x1 x2 x3 x4 x6 x8 x9 + x5 x6 x8 x9 + x1 x5 x6 x8 x9 + x2 x5 x6 x8 x9 + x1 x2 x5 x6 x8 x9 + x3 x5 x6 x8 x9 + x1 x3 x5 x6 x8 x9 + x2 x3 x5 x6 x8 x9 + x1 x2 x3 x5 x6 x8 x9 + x4 x5 x6 x8 x9 + x1 x4 x5 x6 x8 x9 + x1 x2 x4 x5 x6 x8 x9 + x3 x4 x5 x6 x8 x9 + x1 x3 x4 x5 x6 x8 x9 + x2 x3 x4 x5 x6 x8 x9 + x1 x2 x3 x4 x5 x6 x8 x9 + x7 x8 x9 + x1 x7 x8 x9 + x2 x7 x8 x9 + x1 x2 x7 x8 x9 + x1 x2 x3 x7 x8 x9 + x4 x7 x8 x9 + x1 x2 x4 x7 x8 x9 + x3 x4 x7 x8 x9 + x1 x2 x3 x4 x7 x8 x9 + x5 x7 x8 x9 + x2 x5 x7 x8 x9 + x1 x3 x5 x7 x8 x9 + x1 x2 x3 x5 x7 x8 x9 + x1 x4 x5 x7 x8 x9 + x2 x4 x5 x7 x8 x9 + x1 x2 x4 x5 x7 x8 x9 + x2 x3 x4 x5 x7 x8 x9 + x1 x2 x3 x4 x5 x7 x8 x9 + x6 x7 x8 x9 + x1 x2 x6 x7 x8 x9 + x3 x6 x7 x8 x9 + x4 x6 x7 x8 x9 + x1 x4 x6 x7 x8 x9 + x1 x2 x4 x6 x7 x8 x9 + x1 x3 x4 x6 x7 x8 x9 + x1 x5 x6 x7 x8 x9 + x2 x5 x6 x7 x8 x9 + x1 x3 x5 x6 x7 x8 x9 + x4 x5 x6 x7 x8 x9 + x1 x4 x5 x6 x7 x8 x9 + x2 x4 x5 x6 x7 x8 x9 + x3 x4 x5 x6 x7 x8 x9 + x2 x3 x4 x5 x6 x7 x8 x9

Βιβλιογραφία

[01] T. Bäck , "Evolutionary Algorithms in Theory and Practice: Evolution Strategies, Evolutionary Programming, Genetic Algorithms", Oxford University Press US,1996 available online : https://books.google.gr/books?hl=el&lr=&id=htJHI1UrL7IC&oi=fnd&pg=PR9&dq=Thomas+B%D3%93ck.+Evolutionary+Algorithms+in+Theory+and+Practice:+Evolution+Strategies,+Evolutionary+Programming,+Genetic+Algorithms.&ots=fztW0SWzeU&sig=1BJUHwuKM0-S4hckGc4vfsujbPg&redir_esc=y#v=onepage&q&f=false [30-11-2016].

[02]T. Baeck, "Optimal mutation rates in genetic search", Proceeding of the Fifth International Conference on Genetic Algorithms , pp.2-8,1993.

[03] J.E. Baker, "Reducing bias and inefficiency in the selection algorithm", Proceedings of the Second International Conference on Genetic Algorithms and their application, pp.14-21,1987.

[04] T.Blickle, and L.Thiele, "A Comparison of Selection Schemes used in Genetic Algorithms", Computer Engineering and Communication Networks Lab (TIK),TIK-Report Nr.11 version 2,Swiss Federal institute of Technology (ETH),Zurich Switzerland,1995.

[05] L. Budaghyan, A. Pott, "On differential uniformity and nonlinearity of functions", available online : http://www.sciencedirect.com/science/article/pii/S0012365X07010102,Siencedirect ,Vol 309 issue 2, pp. 371-384,2008.

[06] C. Carlet, "Boolean Functions for Cryptography and Error Correcting Codes", Cambridge University Press, available online : https://www.researchgate.net/profile/Claude_Carlet/publication/228720083_Boolean_Functions_for_Cryptography_and_Error_Correcting_Codes/links/5405a6e10cf23d9765a71b9c.pdf , Σελ. : 38,41,43,2007 [25-10-2016].

[07] C. Carlet, "Vectorial Boolean Functions for Cryptography", Cambridge University Press, available online: http://www.math.univ-paris13.fr/∼carlet/chap-vectorial-fcts-corr.pdf , Σελ. : 2,4-8 ,40,2208 [25-10-2016].

[08] C. Carlet,and K. Feng, "An infinite class of balanced functions with optimal algebraic immunity, good immunity to fast algebraic attacks and good nonlinearity",Asiacrypt 2008 (Lecture Notes in Computer Science, Springer), vol. 5350, pp. 425—440,2208.

[09] A. J. Clark, PhD thesis "Optimisation heuristics for cryptology", Queensland University of Technology,1998.

[10] J.H.Chung, P.Stanica, C.H. Tan, Q. Wang, "A construction of Boolean functions with good cryptographic properties", International Journal of Computer Mathematics, Vol 92 issue 4,pp.700-711,2014

[11] C. Darwin, " On the Origin of Species", London - John Murray,1859 Available Online: http://www.gutenberg.org/files/1228/1228-h/1228-h.htm [30-11-2016].

[12] W. Diffie, M. Hellman, "New directions in cryptography", , IEEE Trans. on Inf. Theory, vol. 22, pp. 644-654,1976.

[13] S. Eiben, "Chapter 2 - What is evolution algorithm", Σελ. 4-7,2016.

[14] J. M. Estevez – Tapiador, J. C. Hernandez – Castro, P. z Peris-Lope, and D A. Ribagorda,"Automated Design of Cryptographic Hash Schemes by Evolving Highly-Nonlinear Functions," Journal of Information Science and Engineering, vol. 24, pp. 1485-1504,2008.

[15]S. Fisher,"FAA Equation Finder Version 1.0 ",2008 online: http://www.simonfischer.ch/science/faa.html [27-04-17].

[16] F. Glover, "Future paths for integer programming and links to artificial intelligence", Computers and Operations Research, pp: 533-549,1986.

[17] D.Goldberg, K. Deb, "A comparative analysis of selection schemes used in genetic algorithms" in Foundation of Genetic Algorithms, Morgan Kaufmann,pp.69-93,1991.

[18] J. Holland, "Adaptation in Natural and Artificial Systems",1975.

[19] L.Jacobson, and B. Kanber, “Genetic Algorithms in Java Basics”, Apress , pp.9-44,2015.

[20] K. Limniotis ,N. Kolokotronis, and N. Kalouptsidis, “Secondary constructions of Boolean functions with maximum algebraic immunity”, Cryptogr. Comm., Springer, vol. 5, pp. 179–199,2013.

[21] K. Limniotis, "Algebraic attacks on stream ciphers: Recent developments and new results", Journal of Applied Mathematics and Bioinformatics (Special Issue: Cryptography and its Applications in the Armed Forces), Scienpress Ltd., vol. 3, n. 1, pp. 57—81,2013.

[22] M. Liu, Y. Zhang, and D. Lin, “Perfect Algebraic Immune functions”, ASIACRYPT 2012, Springer, pp. 172-189,2012.

[23] J. McLaughlin, and J. A. Clark, “Evolving balanced Boolean functions with optimal resistance to algebraic and fast algebraic attacks, maximal algebraic degree, and very high nonlinearity”, IACR Cryptology ePrint Archive , available online : https://pdfs.semanticscholar.org/2d7b/620ffe8c4971f28899f64a56f50bfd0d3992.pdf Σελ.: 3,4,2015 [25-10-2016].

[24] W. Millan, A. Clark, and E. Dawson, “An Effective Genetic Algorithm for Finding Highly Nonlinear Boolean Functions”. In Proceedings of the First International Conference on Information and Communication Security, ICICS ’97, London, UK, UK. Springer-Verlag ,p.p: 149–158,1997.

[25] W. Millan, A. Clark, and E. Dawson, “Heuristic design of cryptographically strong balanced Boolean functions”, In Advances in Cryptology - EUROCRYPT ’98, pp. 489–499,1998.

[26] E. Pasalic, “Almost Fully Optimized Infinite Classes of Boolean Functions Resistant to (Fast) Algebraic Cryptanalysis”, ICIS 2008, pp. 399 – 414,2008.

[27]Ι. Petikas,“Υβριδικοί εξελικτικοί αλγόριθμοι βελτιστοποίησης και εφαρμογές σε προβλήματα συνδυαστικής Βελτιστοποίησης”, Πανεπιστήμιο Πειραιά, Σελ :30,39,40,48-50,2012.

[28] S. Picel, “Applications of Evolutionary Computation to Cryptology”, Radboud University Nijmegen, available online :

http://repository.ubn.ru.nl/bitstream/handle/2066/141872/141872.pdf?sequence=1, Σελ. : 51,2015 [25-10-2016].

[29] S. Picek , C. Carlet, D. Jakobovic, J. F. Miller, ,L. Batina, "Correlation Immunity of Boolean Functions: An Evolutionary Algorithms Perspective". In Proc. Of Genetic and Evolutionary Computation Conference (GECCO) 2015, pp. 1095-1102,2015.

[30] S. Picek , C. Carlet, D. Jakobovic, J. F. Miller, ,L. Batina, "Evolutionary Algorithms for Boolean Functions in Diverse Domains of Cryptography", Massachusetts Institute of Technology, MIT press journals, Vol. 24, No. 4, pp. 667-694,2016.

[31] B.R. Pushpa,"Data Encryption Technique Using Genetic Algorithm and Random Number Generator " , International Journal of Innovative Research in Engineering & Science, issue 4 volume 4 ,ISSN: 2319-5665,2015.

[32]P. Rizomiliotis, "On the resistance of Boolean functions against algebraic attacks using univariate polynomial representation", IEEE Trans. Inform. Theory, vol. 56, pp. 4014-4024,2010.

[33] T. Siegenthaler, "Correlation-immunity of nonlinear combining functions for cryptographic applications ". IEEE Trans Inf Theory IT-30(5),pp.776–780,1985.

[34]K. Sindhuja, and S. Pramela Devi, "A Symmetric Key Encryption Technique Using Genetic Algorithm ", International Journal of Computer Science and Information Technologies, Vol. 5 (1),ISSN:0975-9646,pp. 414-416,2014.

[35] H.C.A. Tilborg, and S. Jajodia, "Encyclopedia of Cryptography and Security" , Springer ,Σελ : 31, 162 -164, 224,2011.

[36]A. Tragha, F. Omary, and A. Mouloudi,"Genetic Algorithms Inspired Cryptography", A.M.S.E Association for the Advancement of Modeling & Simulation Techniques in Enterprises, Series D: Computer Science and Statistics,2005.

[37]Wikipedia "Evolutionary algorithm", Wikipedia-the free encyclopedia, Available Online : https://en.wikipedia.org/wiki/Evolutionary_algorithm,2016 [26-11-2016].

[38]Wikipedia ,"Mutation (genetic algorithm)", Wikipedia-the free encyclopedia, Available Online:https://en.wikipedia.org/wiki/Mutation_(genetic_algorithm),2016 [26-11-2016].

[39]Wikipedia "Learning classifier system", Wikipedia-the free encyclopedia, Available Online: https://en.wikipedia.org/wiki/Learning_classifier_system,2017 [29-03-2017].

[40]X. Zeng, C. Carlet, J. Shan, and L. Hu, "More balanced Boolean functions with optimal algebraic immunity and good nonlinearity and resistance to fast algebraic attacks", IEEE Trans. Inform. Theory, vol. 57, pp. 6310—6320,2011.

[41] Γ. Βερυκάκη και Χ. Παπαγιάννη, «Βελτιστοποίηση Σχεδίασης Ευφυούς Κεραίας Με Τη Χρήση Γενετικών Αλγορίθμων», ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ-ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ- ΤΟΜΕΑΣ ΣΥΣΤΗΜΑΤΩΝ ΜΕΤΑΔΟΣΗΣ ΠΛΗΡΟΦΟΡΙΑΣ ΚΑΙ ΤΕΧΝΟΛΟΓΙΑΣ ΥΛΙΚΩΝ, Σελ. 26,27,36,2003.

[42]Α. ΓΚΟΥΝΤΗ, «ΕΞΕΛΙΚΤΙΚΟΙ ΑΛΓΟΡΙΘΜΟΙ:Ο ΑΛΓΟΡΙΘΜΟΣ ΤΗΣ ΔΙΑΦΟΡΙΚΗΣ ΕΞΕΛΙΞΗΣ», ΑΡΙΣΤΟΤΕΛΕΙΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΟΝΙΚΗΣ , Σελ. 13, 14,19-21,2013.

[43]Ι. Βλαχάβας, Π. Κεφάλας, Ν. Βασιλειάδης, Ι. Ρεφανίδης ,Φ. Κόκκορας, και Η. Σακελαρίου, «Κεφ.7 -Τεχνητή Νοημοσύνη-Β' Έκδοση», Θεσσαλλονίκη, online: http://aibook.csd.auth.gr/include/slides/Chap07.pdf ,Σελ.20,21,2002 [20-12-16].

[44]Β. Καμπουρλάζος, «Κεφάλαιο 3: Εξελικτικός Υπολογισμός»online: https://repository.kallipos.gr/bitstream/11419/3446/1/Chapter_3.pdf, Σελ.3-10,2015 [14-04-17].

[45] Δ. Καρνάβας, «Μελέτη εξελικτικών αλγορίθμων εκπαίδευσης για Ασαφή Γνωστικά Δίκτυα», ΠΑΝΕΠΙΣΤΗΜΙΟ ΣΤΕΡΕΑΣ ΕΛΛΑΔΟΣ ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΜΕ ΕΦΑΡΜΟΓΕΣ ΣΤΗ ΒΙΟΪΑΤΡΙΚΗ, Σελ 35,2012

[46]Ε. Κονσολάκη, «ΕΞΕΛΙΚΤΙΚΟΙ ΑΛΓΟΡΙΘΜΟΙ ΚΑΙ ΕΦΑΡΜΟΓΗ ΣΤΗΝ

ΑΠΟΤΥΠΩΣΗ ΧΡΗΜΑΤΙΣΤΗΡΙΑΚΩΝ ΔΕΙΚΤΩΝ» Πολυτεχνείο Κρήτης ,Χανιά,Σελ.23,2008

[47]Λιμνιώτης, Κ. Σημειώσεις μαθήματος «Διάλεξη 5- "Κρυπταλγόριθμοι ροής – Τεχνικές κατασκευής"» ,Σελ.:6,8,10,16,27,2014.

[48]Κ. Λιμνιώτης, Σημειώσεις μαθήματος «Διάλεξη 6- "Κρυπταλγόριθμοι τμήματος– Αλγόριθμοι DES και 3DES"» ,Σελ.:39,2014.

[49]Κ. Λιμνιώτης, Κ. Σημειώσεις μαθήματος «Διάλεξη 1- "Κλασικοί αλγόριθμοι κρυπτογράφησης"» ,Σελ.:6,10,12-17,2014.

[50]Κ. Λιμνιώτης, Σημειώσεις μαθήματος «Διάλεξη 3- "Ασφάλεια κρυπτογραφικών αλγορίθμων"» ,Σελ.:8,10-12,2014.

[51]Κ. Λιμνιώτης, Σημειώσεις μαθήματος «Διάλεξη 4- "Κρυπταλγόριθμοι ροής:

Βασικά χαρακτηριστικά – Τυχαιότητα ακολουθιών"» ,Σελ.:3,4,2014.

[52]Κ. Λιμνιώτης, Σημειώσεις μαθήματος «Διάλεξη 7- "Κρυπταλγόριθμοι τμήματος: Αλγόριθμος AES – Τρόποι λειτουργίας"» ,Σελ.:17-31,2014.

[53]Σ. Λυκοθανάσης, «Γενετικοί Αλγόριθμοι και Εφαρμογές», Θεματική ενότητα «ΤΕΧΝΗΤΗ ΝΟΗΜΟΣΥΝΗ ΚΑΙ ΕΦΑΡΜΟΓΕΣ» τόμος Γ΄,ΕΑΠ, Σελ.:17,18,25-29,35-38,41,43-45,2001.

[54]Α. Μαγκούτης, «Σύγχρονες Επιθέσεις σε Κρυπταλγορίθμους Ροής:Κρυπτογραφικές Ιδιότητες Συναρτήσεων», Ανοικτό Πανεπιστήμιο Κύπρου, Σελ.: 43,46,2014.

[55]Ε. Τόπακα, «Εφαρμογή Γενετικών αλγορίθμων και άλλων μεθόδων επιλογής χαρακτηριστικών για την υποστήριξη λήψης κλινικής απόφασης στη διάγνωση του καρκίνου του τραχήλου της μήτρας »,ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ-ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ- ΤΟΜΕΑΣ ΣΥΣΤΗΜΑΤΩΝ ΜΕΤΑΔΟΣΗΣ ΠΛΗΡΟΦΟΡΙΑΣ ΚΑΙ ΤΕΧΝΟΛΟΓΙΑΣ ΥΛΙΚΩΝ,Σελ.49,2016.