Πανεπιστήμιο Πειραιώς – Τμήμα Πληροφορικής

Πρόγραμμα Μεταπτυχιακών Σπουδών

«Προηγμένα Συστήματα Πληροφορικής»

Μεταπτυχιακή Διατριβή

Τίτλος Διατριβής	**Αλγόριθμοι Αυτοματοποιημένων Συναλλαγών Βασισμένοι σε Τεχνικές Μηχανικής Μάθησης** **Machine Learning-based Automated Trading Algorithms**
Ονοματεπώνυμο Φοιτητή	**Στεργίου Θεόφιλος**
Πατρώνυμο	**Στεργίου Παναγιώτης**
Αριθμός Μητρώου	**ΜΠΣΠ/16032**
Επιβλέπων	**Διονύσιος Σωτηρόπουλος, Επίκουρος Καθηγητής**

Ημερομηνία Παράδοσης **Οκτώβριος 2021**

Τριμελής Εξεταστική Επιτροπή

(υπογραφή)	(υπογραφή)	(υπογραφή)
Διονύσιος Σωτηρόπουλος	Γεώργιος Τσιχριντζής	Ευάγγελος Σακκόπουλος
Επίκουρος Καθηγητής	Καθηγητής	Επίκουρος Καθηγητής

Περιεχόμενα

ΠΕΡΙΛΗΨΗ

Μια σχετικά νέα προσέγγιση στις συναλλαγές κρυπτονομισμάτων είναι η χρήση αλγορίθμων μηχανικής εκμάθησης για την πρόβλεψη της αύξησης και της πτώσης των τιμών πριν εμφανιστούν. Ένας βέλτιστος έμπορος κρυπτονομισμάτων θα αγόραζε πριν ανέβει η τιμή και θα πουλούσε πριν υποχωρήσει η αξία του.

Αυτή η εργασία παρουσιάζει μια προσέγγιση που συνδυάζει ένα μοντέλο επαναλαμβανόμενης μεθόδου ενίσχυσης της μάθησης (RRL) μέσω NeuroEvolution of Augmenting Topologies (NEAT) για να δημιουργήσει ένα σήμα συναλλαγών - νευρωνικό δίκτυο ικανό να επιτύχει υψηλές αποδόσεις σε κρυπτονομίσματα με χαμηλό σχετικό κίνδυνο. Στόχος μας είναι να τροφοδοτούμε το δίκτυό μας με ένα διάνυσμα εισόδου κάθε φορά χρησιμοποιώντας την προηγούμενη έξοδο του νευρωνικού μας δικτύου. Για να δημιουργήσουμε το διάνυσμα εισόδου, χρησιμοποιούμε ένα μοντέλο χρονικών σειρών απόδοσης επενδύσεων, τη θέση του εμπόρου μας που είναι μπορεί να είναι short, long και neutral και έναν συντελεστή ισοδύναμο με την μονάδα.

Η προτεινόμενη προσέγγιση έχει δοκιμαστεί με πραγματικές τιμές καθημερινών δεδομένων κρυπτονομίσματος όπου και σαν αναφερόμενο κρυπτονόμισμα πήραμε το γνωστό σε όλους πλέον BTC(Bitcoin). Ο λόγος Sharpe είναι η μετρική που χρησιμοποιήσαμε στο έργο μας για την αξιολόγηση του μοντέλου μας. Ο παραπάνω αλγόριθμος προσπαθεί να μεγιστοποιήσει το λόγο Sharpe. Τα αποτελέσματα που επιτεύχθηκαν δείχνουν ότι ο λόγος Sharpe αυξάνεται εντός της περιόδου προπόνησης άρα και στο διάστημα του testing παίρνουμε εξίσου καλά αποτελέσματα όπως και κατά την εκπαίδευση .

ABSTRACT

One relatively new approach in crypto currency trading is to use machine learning algorithms to predict the rise and fall of prices before they occur. An optimal crypto currency trader would buy before the price rises and sell before the value declines.

This thesis presents an approach combining a recurrent reinforcement learning method (RRL) model with the NeuroEvolution of Augmenting Topologies (NEAT) to generate a trading signal – neural network capable of achieving high returns in crypto currency with low associated risk. Our goal is to feed our network with an input vector each time using the previous output of our neural network. To create our input vector, we are using a time series model of investment returns, the position of our trader which is short, long, and neutral plus a weighted factor which is equal with one.

The proposed approach has been tested with real daily data of daily cryptocurrency data, where we have selected the well-known cryptocurrency BTC (Bitcoin). Sharpe ratio is the metric that we have used in our Project to evaluate our model. The above-mentioned algorithm attempts to maximize Sharpe ratio. The results achieved show that Sharpe ratio increases within training period therefore we are getting good results also during testing period.

1 ΤΕΧΝΗΤΑ ΝΕΥΡΩΝΙΚΑ ΔΙΚΤΥΑ

1.1 Εισαγωγή

Στο συγκεκριμένο Project θα παρουσιάσουμε την δυνατότητα πραγματοποίησης συναλλαγών σε κρυπτονομίσματα (cryptocurrency trading) με την βοήθεια νευρωνικών δικτύων. Μία σχετικά νέα προσέγγιση στις χρηματοοικονομικές συναλλαγές (trading) είναι η χρήση αλγορίθμων μηχανικής μάθησης για την πρόβλεψη της ανόδου ή και της πτώσης στις τιμές περιουσιακών στοιχείων προτού συμβούν. Η τιμή ενός bitcoin που θα είναι και το νόμισμα που θα χρησιμοποιήσουμε σαν αναφορά για αυτή την εργασία καθορίζεται από την προσφορά και τη ζήτηση ,όταν η ζήτηση για bitcoin αυξάνεται, η τιμή αυξάνεται και όταν η ζήτηση πέφτει, η τιμή πέφτει κάποιες ωστόσο έχουμε παρατηρήσει ότι αυτό δεν είναι πάγιο.

Ένας <<καλός>> έμπορος (trader) λοιπόν θα αγόραζε ένα περιουσιακό στοιχείο (κβάντο- bitcoin) πριν από την αύξηση της τιμής και θα πουλούσε το περιουσιακό στοιχείο πριν η αξία του μειωθεί. Η προσέγγιση της μηχανικής μάθησης στη χρηματοπιστωτική αγορά βασίζεται στη μάθηση από την εκπαίδευση και έπειτα την εμπειρία. Οι αλγόριθμοι μηχανικής μάθησης είναι ικανοί να επεξεργαστούν μεγάλο αριθμό προηγούμενων οικονομικών δεδομένων (καθώς και άλλες πληροφορίες που επηρεάζουν την αγορά, όπως νέες ειδήσεις και τάσεις αγορών η δεικτών) που μπορεί να φαίνονται ασύνδετα με το ανθρώπινο μάτι, για τον εντοπισμό μοτίβων και την πρόβλεψη μελλοντικών αποτελεσμάτων της αγοράς με στόχο τη μεγιστοποίηση της απόδοσης/κέρδους. Αυτή η προσέγγιση διευκολύνει την γρήγορη αντίδραση σε γεγονότα που επηρεάζουν την αγορά και έτσι η χρήση μηχανικής μάθησης

μπορεί να είναι ανταγωνιστικό πλεονέκτημα για τον προσδιορισμό των ιδανικών σημείων εισόδου και εξόδου(entry – exit points). Πολλές από τις μεθόδους μηχανικής μάθησης που εφαρμόζονται στις χρηματοπιστωτικές αγορές χρησιμοποιούν ως εισροές (inputs) , τεχνικούς δείκτες (indicators) αντί για οικονομικά δεδομένα καθώς οι δείκτες αυτοί δεν είναι τόσο θορυβώδεις όσο τα ακατέργαστα οικονομικά δεδομένα και αν επιλεγεί κατάλληλα, μπορεί να διευκολύνει στην ανακάλυψη προτύπων και αγορών και τάσεων που μπορούν μετέπειτα να χρησιμοποιηθούν από τον αλγόριθμο για να προβλεφθούν μελλοντικά αποτελέσματα.

Στο συγκεκριμένο Project, θα προσομοιώσουμε έναν έμπορο περιουσιακών στοιχείων (bitcoin trader) χρησιμοποιώντας την μέθοδο επαναλαμβανόμενης εκμάθησης ενίσχυσης (RRL-recurrent reinforcement learning) κάνοντας χρήση του γνωστού αλγορίθμου NEAT. Στην εκμάθηση με ενίσχυση ο υπολογιστής χρησιμοποιεί την δοκιμή και το σφάλμα για να βρει μια λύση στο πρόβλημα. Πιο συγκεκριμένα το δίκτυο δέχεται ανταμοιβές ή κυρώσεις για τις ενέργειες που εκτελεί όπου έχει σαν κύριο στόχο να μεγιστοποιήσει τη συνολική ανταμοιβή. Παρόλο που εμείς καθορίζουμε την πολιτική ανταμοιβής , δεν μπορούμε να δώσουμε σαφείς οδηγίες για την επίλυση του προβλήματος . Έτσι, το μοντέλο, αναλαμβάνει την εκτέλεση της εργασίας αυτής και ξεκινώντας από εντελώς τυχαίες - στοχαστικές δοκιμές , καταλήγει να χρησιμοποιεί εξελιγμένες τακτικές βασισμένες σε πολύπλοκα μαθηματικά μοντέλα. Η τεχνική αυτή απαιτεί σημαντική υπολογιστική ισχύ.

Το μοντέλο και οι παράμετροι που χρησιμοποιήθηκαν έχουν αναπτυχθεί και βασίζονται στην επιστημονική δημοσίευση των Moody και Saffell που έγινε στο Stanford[1]. Πιο συγκεκριμένα θα χρησιμοποιηθεί ένας αλγόριθμος ανάβασης κλίσης που επιχειρεί να

μεγιστοποιήσει την συνάρτηση χρησιμότητας , ως μέθοδος ανταμοιβής θα χρησιμοποιηθεί η αναλογία Sharpe (Sharpe ratio). Επίσης στο πρόβλημα αυτό προσπαθούμε να επωφεληθούμε από τις αλλαγές των τιμών έτσι ώστε να επιλέξουμε μια βέλτιστη παράμετρο w (weight) για τον trader.

Μερικές διαφορετικές προσεγγίσεις επίλυσης του προβλήματος πρόβλεψης χρονοσειρών - για πρόγνωση της τιμής είτε σε κρυπτονομίσματα είτε σε καθαρό συνάλλαγμα είναι η Long short-term memory (LSTM) αρχιτεκτονική νευρωνικού δικτύου, η οποία είναι τύπου RNN(Recurrent neural network) και τα πολυέπιπεδα δίκτυα πρόσθιας τροφοδότησης (MLP – Multilayer Perceptron) Back Propagation μέθοδοι ωστόσο αυτές οι μέθοδοι αφορούν καθαρά το prediction και όχι το prediction σε συνδυασμό με απόφαση όπως το μοντέλο που χρησιμοποιήσαμε εμείς. Μια ίσως κοντινή προσέγγιση στο δικό μας πρόβλημα θα ήταν η χρήση ενός Deep Neural Network (DNN), Support Vector Machine (SVM), Random Forest (RF) και Logistic Regression (LR)[2] βασιζόμενο στην λήψη αποφάσεων για την κατεύθυνση της αγοράς (Afan Hasan, 2020) ή μιας Deep Q – Learning τεχνικής που συμπίπτει στην κατηγορία του Reinforcement Learning[3].

1.2 Γενική Περιγραφή

Τα Τεχνητά Νευρωνικά Δίκτυα (ANNs) **Artificial neural networks** είναι εργαλεία υπολογιστικής μοντελοποίησης που βρίσκουν εκτεταμένη αποδοχή σε πολλούς τομείς για τη μοντελοποίηση πολύπλοκων πραγματικών προβλημάτων. Τα ANNs ορίζονται ως δομές που αποτελούνται από πυκνά διασυνδεδεμένα προσαρμοστικά στοιχεία απλής επεξεργασίας (τεχνητοί νευρώνες ή κόμβοι) που είναι σε θέση να πραγματοποιούν σωρευτικά παράλληλους υπολογισμούς για την

επεξεργασία δεδομένων και την εκπροσώπηση γνώσης. Παρόλο που τα ANNs είναι δραστικές αφαιρέσεις των βιολογικών ομολόγων, η ιδέα των ANNS δεν είναι να αναπαράγει τη λειτουργία των βιολογικών συστημάτων, αλλά να κάνει χρήση της λειτουργικότητας των βιολογικών δικτύων για την επίλυση σύνθετων προβλημάτων. Η ελκυστικότητα των ANN προέρχεται από τα αξιοσημείωτα χαρακτηριστικά επεξεργασίας της πληροφορίας του βιολογικού συστήματος όπως (I.A. Basheer, 2000):

- **Προσαρμοστικότητα:** Τα νευρωνικά δίκτυα έχουν μπορούν να προσαρμόζουν τα συναπτικά τους βάρη σύμφωνα με τις αλλαγές που γίνονται στο περιβάλλον τους. Ένα ΤΝΔ εκπαιδευμένο να λειτουργεί σε συγκεκριμένο περιβάλλον, μπορεί εύκολα να γίνει retrain ώστε να μπορεί να χειρίζεται τις μεταβολές στο νέο περιβάλλον λειτουργίας του.
- **Ομοιομορφία ανάλυσης και σχεδίασης:** Οι νευρώνες αντιπροσωπεύουν ένα συστατικό κοινό σε όλα τα νευρωνικά δίκτυα και αυτό καθιστά εφικτή τη χρήση των ίδιων θεωριών και αλγορίθμων μάθησης σε διαφορετικές εφαρμογές των νευρωνικών δικτύων.
- **Χαρτογράφηση εισόδου εξόδου:** ένα δημοφιλές παράδειγμα μάθησης, η μάθηση με εκπαιδευτή ή επιβλεπόμενη μάθηση, συνιστάται στην τροποποίηση των συναπτικών βαρών ενός νευρωνικού δικτύου εφαρμόζοντας ένα σύνολο παραδειγμάτων εκπαίδευσης. Κάθε παράδειγμα αποτελείται από ένα μοναδικό σήμα εισόδου και μια αντίστοιχη επιθυμητή απόκριση. Τα συναπτικά βάρη τροποποιούνται προκειμένου να ελαχιστοποιηθεί η διαφορά μεταξύ της

επιθυμητής απόκρισης και της πραγματικής απόκρισης του δικτύου. Τότε το δίκτυο φτάνει σε μια ευσταθή κατάσταση όπου δεν υπάρχουν άλλες αλλαγές βαρών. Αν και τα ΤΝΔ δεν είναι τα μόνα συστήματα με ικανότητα μάθησης μέσω παραδειγμάτων, ,εντούτοις διακρίνονται για την ικανότητά τους να οργανώνουν την πληροφορία των δεδομένων εισόδου σε χρήσιμες μορφές. Αυτές οι μορφές αποτελούν στην ουσία ένα μοντέλο που αναπαριστά τη σχέση που ισχύει μεταξύ των δεδομένων εισόδου και εξόδου.

- **Παράλληλος τρόπος λειτουργίας:** Τα νευρωνικά δίκτυα λειτουργούν με παράλληλο τρόπο γιατί μια εργασία διαμοιράζεται στα διάφορα τμήματα του δικτύου, δηλαδή σε όλους τους επιμέρους νευρώνες. Έτσι τα νευρωνικά δίκτυα είναι συστήματα παράλληλων κατανεμημένων διεργασιών. Αυτό παρέχει μεγάλες ταχύτητες γιατί είναι σαν να έχουμε ταυτόχρονα πολλούς επεξεργαστές.
- **Μη γραμμικότητα:** ένας τεχνητός νευρώνας μπορεί να μη γραμμικός. Ένα ΤΝΔ που αποτελείται από διασυνδεδεμένους μη γραμμικούς νευρώνες είναι μη γραμμικό. Η συγκεκριμένη ιδιότητα είναι πολύ σημαντική, κυρίως αν ο υποκείμενος φυσικός μηχανισμός που παράγει το σήμα εισόδου είναι εκ φύσεως μη γραμμικός. Η ιδιότητα αυτή είναι σημαντική, ιδιαίτερα στην περίπτωση που τα σήματα εισόδου έχουν τέτοια χαρακτηριστικά (πχ. ομιλία)
- **Ανοχή σε σφάλματα:** Τα νευρωνικά δίκτυα είναι ανεκτικά σε μικρές αλλαγές στην είσοδό τους και σε περίπτωση που είναι υλοποιημένο σε hardware είναι ανθεκτικά σε βλάβες. Υπό την έννοια ότι σε περίπτωση που νευρώνας ή σύνδεση πάθουν βλάβη η ποιότητα της εξόδου μειώνεται σταδιακά, καθότι η πληροφορία που

αποθηκεύεται στο δίκτυο είναι κατανεμημένη σε όλη την δομή του. Συνεπώς το μέγεθος του σφάλματος είναι ανάλογο του ποσού των κατεστραμμένων συνδέσεων

- **Αναλογία με την νευροφυσιολογία του εγκεφάλου:** Η σχεδίαση ενός νευρωνικού δικτύου δανείζεται στοιχεία από τη λειτουργία του ανθρώπινου εγκεφάλου, ο οποίος είναι η ζωντανή απόδειξη ότι η εύρωστη, παράλληλη επεξεργασία δεν είναι μόνο φυσικά εφικτή, αλλά επίσης γρήγορη και ισχυρή. Τα ΤΝΔ, όπως και τα βιολογικά, έχουν μεγάλη ανοχή σε δομικά σφάλματα. Αυτό σημαίνει ότι η κακή λειτουργία ή η καταστροφή ενός νευρώνα ή κάποιων συνδέσεων δεν είναι ικανή να διαταράξει σημαντικά τη λειτουργία τους καθώς η πληροφορία που εσωκλείουν δεν είναι εντοπισμένη σε συγκεκριμένο σημείο αλλά διάχυτη σε όλο το δίκτυο. Γενικά, το μέγεθος σφάλματος λόγω δομικών αστοχιών είναι ανάλογο του ποσοστού των κατεστραμμένων συνδέσεων.
- **Η δυνατότητα θεώρησής τους ως κατανεμημένη μνήμη (Distributed memory) και ως μνήμη συσχέτισης (Associative memory):** Ο χαρακτηρισμός των ΤΝΔ ως κατανεμημένη μνήμη, προκύπτει από το ότι η κωδικοποίηση που δημιουργούν είναι κατανεμημένη σε όλα τα βάρη της συνδεσμολογίας τους. Για τον ίδιο λόγο τα ΤΝΔ χαρακτηρίζονται και ως μνήμες συσχέτισης. Μια μνήμη συσχέτισης αποθηκεύει πληροφορία μέσω κατάλληλων συσχετίσεων που δημιουργεί από τα δεδομένα εκπαίδευσης. Η ανάκληση της πληροφορίας γίνεται με βάση το περιεχόμενο και ότι τη διεύθυνση, όπως δηλαδή συμβαίνει και με τον ανθρώπινο εγκέφαλο. Η παραπάνω οργάνωση κάνει ορισμένα είδη ΤΝΔ να είναι πολύ

ανεκτικά σε μικρές αλλαγές στα σήματα εισόδου, δηλαδή να είναι σε θέση να παράγουν τη σωστή έξοδο ακόμη και αν τα δεδομένα εισόδου είναι λίγο διαφορετικά (για παράδειγμα, λόγω θορύβου) ή και ελλιπή.

- **Η ικανότητά τους για αναγνώριση προτύπων (pattern recognition):** Τα ΤΝΔ έχουν εξαιρετική ικανότητα αναγνώρισης προτύπως καθώς δεν επηρεάζονται από ελλιπή ή και με θόρυβο δεδομένα. Από τη στιγμή που ένα ΤΝΔ εκπαιδευτεί στο να αναγνωρίζει συνθήκες και καταστάσεις, απαιτείται ένας μόνο κύκλος λειτουργίας τους για να προσδιορίσουν μια συγκεκριμένη κατάσταση. Η τελευταία ιδιότητα κάνει ένα ΤΝΔ ιδανικό για χρήση σε αυτοματισμούς που θα λειτουργήσουν σε αντίξοες συνθήκες.

Βασικός στόχος των υπολογισμών που βασίζονται στο ANN είναι να αναπτυχθούν μαθηματικοί αλγόριθμοι που θα επιτρέψουν στην ANN με τη μίμηση του ανθρώπινου εγκέφαλου την επεξεργασία πληροφοριών και την απόκτηση γνώσεων. Τα μοντέλα που βασίζονται στο ANN είναι εμπειρικά, αλλά μπορούν να παράσχουν πρακτικά ακριβείς λύσεις για ακριβή ή ανακριβώς διατυπωμένα προβλήματα και για φαινόμενα που κατανοούνται μόνο μέσω πειραματικών δεδομένων και παρατηρήσεων. Στη μικροβιολογία, τα ANNs έχουν χρησιμοποιηθεί σε ποικίλες εφαρμογές που κυμαίνονται από τη μοντελοποίηση, την ταξινόμηση, την αναγνώριση προτύπων και την ανάλυση πολλών μεταβλητών δεδομένων. Οι εφαρμογές δειγμάτων περιλαμβάνουν :

- την ερμηνεία της φασματομετρίας πυρόλυσης μάζας, GC και HPLC αναγνώριση προτύπου DNA, RNA, πρωτεϊνικής δομής και μικροσκοπικών εικόνων

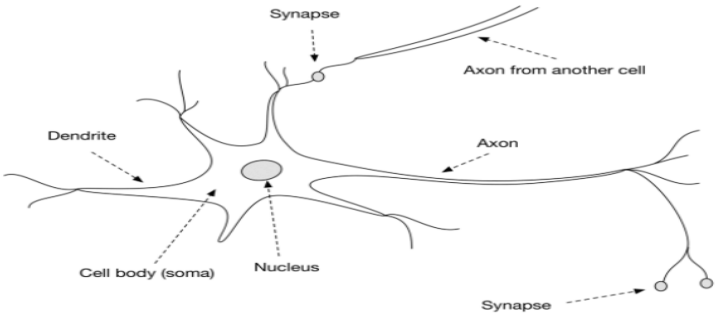
- πρόβλεψη μικροβιακής ανάπτυξης βιομάζας
- ταυτοποίηση μικροοργανισμών και μορίων

Η ιδέα των ΤΝΔ ξεκίνησε από τη συνεργασία των McCulloch και Pitts το 1943, οι οποίοι συνδύασαν τις γνώσεις τους στον τομέα της νευρολογίας και των μαθηματικών αντίστοιχα. Στο επίσημο μοντέλο τους, υποστήριξαν ότι ένα δίκτυο που αποτελείται από επαρκή αριθμό απλών υπολογιστικών μονάδων, κατάλληλα συνδεδεμένων, μπορεί να εκτελέσει οποιαδήποτε πράξη και διαδικασία. Η δουλειά τους αυτή, σε συνδυασμό με την ιδέα ότι ο ανθρώπινος εγκέφαλος λειτουργεί με έναν εντελώς διαφορετικό τρόπο από τους τότε συμβατικούς ψηφιακούς υπολογιστές, αποτέλεσε τη βάση για μια πληθώρα από εργασίες και έρευνες. Στη γενική του μορφή, ένα ΤΝΔ αποτελεί μία μηχανή που είναι σχεδιασμένη να προσομοιώνει τον τρόπο με τον οποίο ο εγκέφαλος εκτελεί μία εργασία ή μία πράξη ενδιαφέροντος. Τα δίκτυα συνήθως υλοποιούνται χρησιμοποιώντας ηλεκτρονικά στοιχεία ή προσομοιώνοντας τη λειτουργίας τους με χρήση λογισμικού σε έναν ηλεκτρονικό υπολογιστή. Για να πετύχουν την επιθυμητή απόδοση, τα ΤΝΔ χρησιμοποιούν ένα μεγάλο αριθμό διασυνδέσεων μεταξύ των υπολογιστικών στοιχείων.

Χαρακτηριστικό των συστημάτων αυτών είναι η μάθηση. Η διαδικασία αυτή, γνωστή και ως αλγόριθμος εκπαίδευσης, έχει σκοπό τη ρύθμιση και μεταβολή των διασυνδέσεων των υπολογιστικών μονάδων, ώστε να επιτευχθεί ο επιθυμητός στόχος. Σε κάποιες περιπτώσεις μάλιστα, ο αλγόριθμος εκπαίδευσης μπορεί να επιφέρει αλλαγές και στην τοπολογία του δικτύου, ακολουθώντας το παράδειγμα του εγκεφάλου, όπου κάποιοι νευρώνες μπορεί να καταστραφούν ή καινούργιες διασυνδέσεις να δημιουργηθούν.

1.3 Μοντέλο Νευρώνα

Ένα ΤΝΔ αποτελεί ένα δίκτυο διασυνδεδεμένων υπολογιστικών μονάδων, γνωστών και ως **νευρώνων**. Ο νευρώνας, αποτελεί το κύριο δομικό στοιχείο των ΤΝΔ, η δομή του οποίου είναι εμπνευσμένη από τους νευρώνες του εγκέφαλου (Εικόνα 1) και έχει τα εξής χαρακτηριστικά:



Εικόνα 1: Βιολογικός νευρώνας

- Ένα σύνολο από συνδέσεις, τις λεγόμενες **συνάψεις** κάθε μία από τις οποίες χαρακτηρίζεται από ένα **βάρος(weight)**. Οι συνάψεις αυτές αποτελούν την είσοδο των διαφόρων σημάτων στον νευρώνα και μπορεί να προέρχονται από άλλους νευρώνες ή από το περιβάλλον του δικτύου. Τα βάρη των συνάψεων λειτουργούν πολλαπλασιαστικά στα σήματα και μπορούν να έχουν οποιαδήποτε πραγματική τιμή.
- Έναν **αθροιστή** για την άθροιση των σημάτων εισόδου, πολλαπλασιασμένων με τα βάρη των συνάψεων, και την πόλωση.

$$\mathbf{u_k} = \sum_{\mathbf{i=0}}^{\mathbf{m}} \mathbf{w_{ki} x_i} + \mathbf{b_k}$$

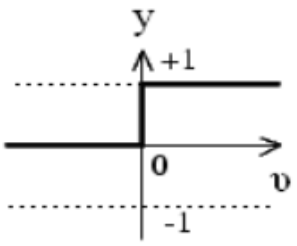
- Και τέλος, μία συνάρτηση ενεργοποίησης, yk. Η συνάρτηση αυτή δέχεται ως είσοδο την έξοδο του αθροιστή και το αποτέλεσμα αποτελεί την έξοδο του ίδιου του νευρώνα στο υπόλοιπο σύστημα ή το εξωτερικό περιβάλλον.

$$\boldsymbol{y_k = \varphi(u_k)}$$

Τρεις από τις συναρτήσεις ενεργοποίησης που συναντώνται στη βιβλιογραφία παρουσιάζονται παρακάτω:

- **Συνάρτηση κατωφλιού (Heaviside)**

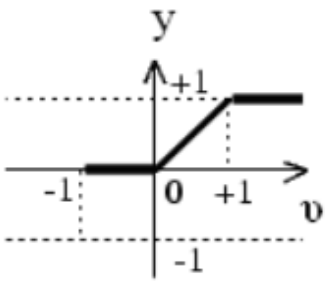
$$\boldsymbol{\varphi(u) = \begin{cases} 0, & u < 0 \\ 1, & u \geq 0 \end{cases}}$$



Εικόνα 3: Συνάρτηση ενεργοποίησης Κατωφλιού

- **Κατά-τμήματα γραμμική**

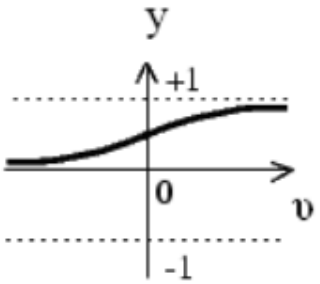
$$\varphi(u) = \begin{cases} 1 & u \geq c_2 \\ a\,u + b & c_2 \geq u > c_1 \\ 0 & u \leq c_1 \end{cases}$$



Εικόνα 2: Συνάρτηση ενεργοποίησης κατά τμήματα γραμμική

- **Σιγμοειδής**

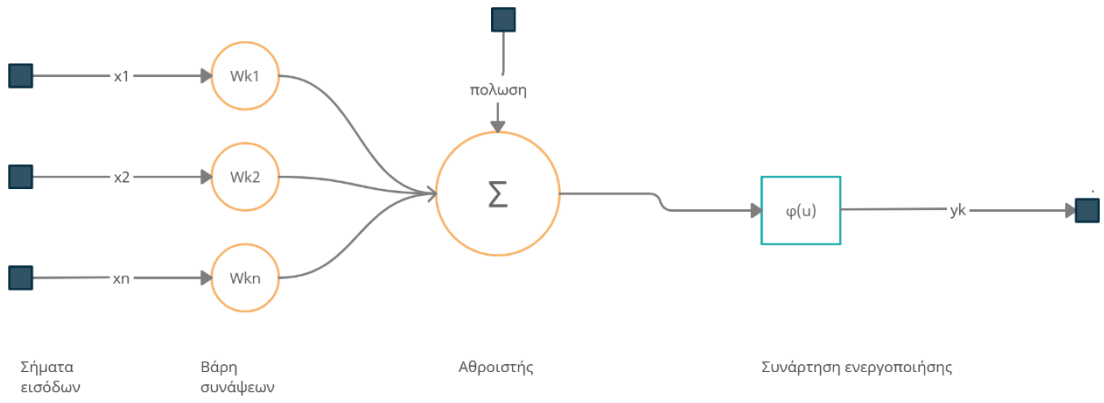
$$\varphi(v) = \frac{1}{1 + e^{-au}}$$



Εικόνα 3: Συνάρτηση ενεργοποίησης σιγμοειδής

Οι παραπάνω συναρτήσεις χρησιμοποιούνται όταν ο σχεδιαστής προτιμάει τιμές εξόδου στο εύρος [0,1]. Σε ορισμένες περιπτώσεις, η έξοδος σχεδιάζεται έτσι ώστε να βγάζει τιμές στο εύρος [-1,1]. Οι συναρτήσεις ενεργοποίησης είναι παρόμοιες όσον αφορά τη δομή, με αντίθεση τη σιγμοειδή, όπου γίνεται χρήση της υπερβολικής εφαπτομένης.

$$\boldsymbol{\varphi(u) = \tanh(u)}$$



Εικόνα 4:Μοντέλο τεχνητού νευρώνα

Αξίζει να αναφερθεί ότι το παραπάνω μοντέλο ακολουθεί ντετερμινιστικό σκεπτικό όσον αφορά την έξοδο ενός νευρώνα. Σε ορισμένες εφαρμογές είναι επιθυμητό, η ενεργοποίηση να ακολουθεί στοχαστικές διαδικασίες. Στην περίπτωση αυτή, η έξοδος λαμβάνει μόνο δύο καταστάσεις βάσει πιθανότητας *P(u)* . Η πιθανότητα αυτή εξαρτάται και πάλι από τις εισόδους του νευρώνα μέσω των συνάψεων, που στην περίπτωση αυτή παίζουν το ρόλο θορύβου.

1.4 Συνήθεις Αρχιτεκτονικές

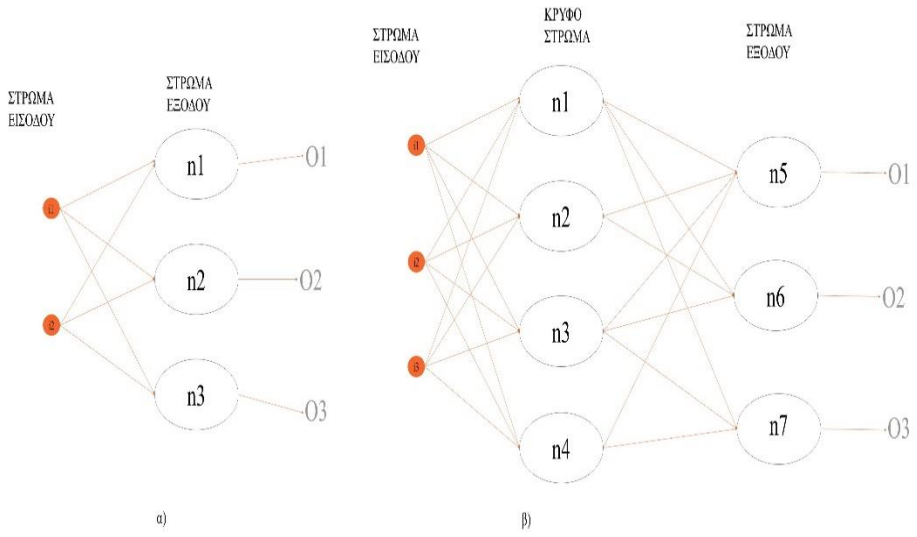
Ένα ΤΝΔ μπορεί να αναπαρασταθεί από έναν κατευθυνόμενο γράφο. Κάθε κόμβος του γράφου αντιστοιχίζεται σε ένα νευρώνα του ΤΝΔ και κάθε ακμή σε μία σύναψη. Εξαίρεση αποτελούν οι κόμβοι εισόδου ή πηγής, οι οποίοι δεν αντιστοιχούν σε νευρώνες άλλα ρόλος τους είναι η εισαγωγή σημάτων στο δίκτυο. Τα βάρη των συνάψεων αντιστοιχίζονται στα βάρη των ακμών. Συνήθης τακτική είναι να χωρίζονται οι κόμβοι σε ομάδες ανάλογα με τη θέση τους στο δίκτυο και τον αριθμό των βημάτων που χρειάζεται για να φτάσει ένα σήμα στον κόμβο αυτό. Οι ομάδες αυτές ονομάζονται στρώματα. Υπάρχουν τρείς κατηγορίες στρωμάτων. Το στρώμα εισόδου, στο οποίο ανήκουν οι κόμβοι εισόδου, το στρώμα εξόδου, που αποτελείται από κόμβους που ορίζουν την απόκριση του δικτύου και τα κρυφά στρώματα, στα οποία ανήκουν οι υπόλοιποι κόμβοι. Ο αριθμός των στρωμάτων και ο τρόπος με τον οποίο αυτά συνδέονται ορίζει την αρχιτεκτονική του δικτύου. Παρακάτω παρουσιάζονται οι τρείς βασικές κατηγορίες

1.4.1 Μονού Στρώματος Πρόσθιας Τροφοδότησης

Η πιο απλουστευμένη μορφή ΤΝΔ είναι αυτή του **Μονού Στρώματος Πρόσθιας Τροφοδότησης** (Single-Layer Feed Forward). Σε αυτού του τύπου την αρχιτεκτονική, το ΤΝΔ αποτελείται από input και ένα output layer. Ονομάζεται μονού στρώματος γιατί παραβλέπεται το στρώμα εισόδου, καθώς σε αυτό δεν εκτελούνται πράξεις. Ο όρος πρόσθιας τροφοδότησης δηλώνει ότι κάθε κόμβος ενός στρώματος μπορεί να τροφοδοτεί έναν ή περισσότερους κόμβους που ανήκουν στο επόμενο στρώμα. Δεν υπάρχουν δηλαδή ανακυκλώσεις ή μεταφορά σήματος σε κόμβο του ίδιου στρώματος.

1.4.2 Πολλαπλών Στρωμάτων Πρόσθιας Τροφοδότησης

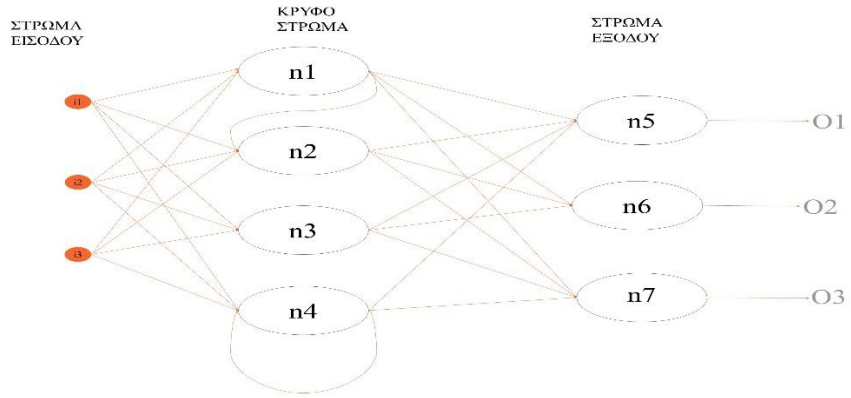
Η δεύτερη κατηγορία αρχιτεκτονικών ΤΝΔ είναι αυτή των **Πολλαπλών Στρωμάτων Πρόσθιας Τροφοδότησης**. Αυτό που διαφοροποιεί αυτή την κατηγορία από την πρώτη είναι η εισαγωγή ενός ή περισσότερων κρυφών στρωμάτων στο δίκτυο. Το πρώτο κρυφό στρώμα τροφοδοτείται από το στρώμα εισόδου και στη συνέχεια οι έξοδοι των κόμβων τροφοδοτούν το επόμενο στρώμα. Η διαδικασία επαναλαμβάνεται μέχρι να φτάσουμε στο στρώμα εξόδου. Η αρχιτεκτονική αυτής της μορφής επιτρέπει τη δημιουργία πιο πολύπλοκων δικτύων που έχουν την δυνατότητα επίλυσης προβλημάτων ανωτέρας τάξης.



Εικόνα 5:Δίκτυα πρόσθιας τροφοδότησης (α) μονού στρώματος (β) με κρυφό στρώμα

1.4.3 Δίκτυο με ανάδραση

Η τρίτη και τελευταία κατηγορία αρχιτεκτονικής ενός ΤΝΔ είναι αυτή των δικτύων με ανάδραση. Όπως περιγράφεται και από το όνομα, ένα δίκτυο ανήκει στην κατηγορία αυτή αν περιέχει έστω και μία ανάδραση. Ανάδραση παρατηρείται όταν η έξοδος ενός κόμβου τροφοδοτεί έναν κόμβο που ανήκει στο ίδιο ή προηγούμενο στρώμα, όπως φαίνεται στην Εικόνα 7. Η παρουσία αναδράσεων σε ένα ΤΝΔ έχει μεγάλη επίπτωση στις δυνατότητες μάθησης του δικτύου όπως και στην επίδοσή του. Μπορούμε να πούμε ότι στο δίκτυο εισάγεται με κάποιο τρόπο η έννοια της μνήμης καθώς η νέα απόκριση του δικτύου εξαρτάται και από στοιχεία της προηγούμενης.



Εικόνα 6:Δίκτυο με ανάδραση

2 ΕΚΠΑΙΔΕΥΣΗ ΤΝΔ

2.1 Τύποι Τεχνικών Νευρωνικών Δικτυων

Μία από τις βασικότερες δυνατότητες ενός ΤΝΔ είναι ότι μπορεί μαθαίνει από το περιβάλλον και μέσα από τη διαδικασία αυτή να βελτιώνει την απόδοσή του. Όπως αναφέρθηκε, η διαδικασία μάθησης ενός ΤΝΔ ονομάζεται Αλγόριθμος Εκπαίδευσης. Στη βιβλιογραφία υπάρχουν αρκετοί τέτοιοι αλγόριθμοι. Όπως για κάθε πρόβλημα που έχει πολλές λύσεις, κάποιος μπορεί να σκεφτεί τους αλγόριθμους αυτούς ως ένα κουτί από εργαλεία. Ανάλογα με τη φύση του προβλήματος που καλείται να αντιμετωπίσει το ΤΝΔ, μπορεί να επιλεχθεί και ο κατάλληλος αλγόριθμος.

Σε γενικές γραμμές, υπάρχουν δύο προσεγγίσεις όσο αφορά τους αλγόριθμους εκπαίδευσης: α) εκπαίδευση με «δάσκαλο» και β) εκπαίδευση χωρίς «δάσκαλο». Η εκπαίδευση με δάσκαλο, γνωστή και ως **Εκπαίδευση με** Επίβλεψη (ΕμΕ, Supervised Learning), βασίζεται στην εκπαίδευση μέσα από παραδείγματα και περιλαμβάνει έναν «δάσκαλο» που παρέχει στο ΤΝΔ την επιθυμητή έξοδο για δεδομένη είσοδο, ώστε το νευρωνικό δίκτυο να μπορεί να συγκρίνει την απόκρισή του σε σχέση με τη σωστή. Οι αλγόριθμοι που ανήκουν στην κατηγορία αυτή βασίζονται σε κανόνες μάθησης, όπως της διόρθωσης σφάλματος (Error-correction) και βασιζόμενης στη μνήμη (Memory Based). Χαρακτηριστικά παραδείγματα τέτοιων αλγορίθμων είναι ο Back-Propagation και ο LQV.

Στη εκπαίδευση χωρίς δάσκαλο το δίκτυο μαθαίνει χωρίς εξωτερική βοήθεια. Η κύρια ιδέα των αλγορίθμων αυτής της κατηγορίας είναι το ΤΝΔ να μάθει από μόνο του έναν πιο

αποτελεσματικό τρόπο αντιπροσώπευσης των δεδομένων. Η διαδικασία της εκπαίδευσης είναι συνυφασμένη συνήθως με κανόνες βασισμένους στη νευροβιολογία, όπως Hebbian, ή στοχαστικούς κανόνες, όπως του Boltzmann. Η κατηγορία αυτή μπορεί να χωριστεί σε δύο περαιτέρω υπόκατηγορίες: β1) την Ενισχυτική Εκπαίδευση (ΕΕ, Reinforcement Learning - RL) και β2) την Εκπαίδευση Χωρίς Επίβλεψη (ΕΧΕ, Unsupervised). Στην περίπτωση της ΕΕ, η διαδικασία της μάθησης γίνεται μέσα από συνεχή αλληλεπίδραση του ΤΝΔ με το περιβάλλον εκπαίδευσης, με σκοπό τη βελτιστοποίηση ενός δείκτη ποιότητας. Παραδείγματα αλγορίθμων ΕΕ αποτελούν και οι Εξελικτικοί Αλγόριθμοι (ΕΑ), ένας από τους οποίους αποτέλεσε και βάση για τη λύση που παρουσιάζεται σε αυτή τη διπλωματική. Στην ΕΧΕ, όπως και στην ΕΕ, δεν υπάρχει κάποιος κριτής που να επιβλέπει την διαδικασία μάθησης, εν αντιθέσει όμως με την ΕΕ, το μέτρο ποιότητας απόκρισης του δικτύου είναι συνήθως ανεξάρτητο από την ίδια την πράξη και οι ελεύθερες παράμετροι του ΤΝΔ βελτιστοποιούνται βάσει αυτού του μέτρου. Παραδείγματα τέτοιων αλγορίθμων είναι οι Self-Organised Maps (SOM) και οι ανταγωνιστικοί (competitive).

2.2 Εξελικτικοί αλγόριθμοι

Οι ΕΑ συνθέτουν μία ευρεία κλάση στοχαστικών μεθόδων βελτιστοποίησης και αποτελούν σήμερα ένα δυνατό εργαλείο για την επίλυση και αντιμετώπιση ποικίλων προβλημάτων. Η έμπνευση τους προέρχεται από την βιολογία και συγκεκριμένα από εκείνες τις διεργασίες που επιτρέπουν πληθυσμούς από οργανισμούς να προσαρμόζονται στο περιβάλλον τους, όπως η γενετική κληρονομικότητα και η επιβίωση του καταλληλότερου, ιδέες που εμφανίστηκαν στα μέσα του 19ου αιώνα.

2.3 Γενική Περιγραφή ΕΑ

Συνοπτικά, ο τρόπος που λειτουργεί ένας ΕΑ είναι να ενημερώνει επαναληπτικά έναν **πληθυσμό** από μέλη που αποτελούν τις πιθανές λύσεις στο πρόβλημα που μας αναφοράς. Τα μέλη του πληθυσμού, που συνήθως ονομάζονται **οργανισμοί**, κωδικοποιούνται σε δομές που ονομάζονται **χρωμοσώματα/γονιδιώματα**. Σε κάθε επανάληψη, που ονομάζεται **γενιά**, οι οργανισμοί-λύσεις του πληθυσμού αξιολογούνται βάσει μίας συνάρτησης καταλληλότητας και στη συνέχεια δημιουργούνται οι **απόγονοι** (νέες λύσεις) των οργανισμών που είχαν καλή απόδοση. Κατά τη διαδικασία της γέννησης νέων απογόνων, τα **γονίδια**, στοιχεία που συνθέτουν μία λύση, μεταβάλλονται με χρήση γενετικών πράξεων όπως **μετάλλαξη** ή **διασταύρωση**. Η κεντρική ιδέα είναι ότι δομές που σχετίζονται ενίοτε με καλές λύσεις μπορούν να μεταλλαχθούν ή να διασταυρωθούν και να συνθέσουν καλύτερες λύσεις σε επόμενες γενιές. Παρακάτω παρουσιάζεται ένας ΕΑ στη βασική του μορφή. Στη βιβλιογραφία συναντώνται διάφοροι τύποι ΕΑ που περιλαμβάνουν Γενετικούς Αλγόριθμους Εξελικτικό Προγραμματισμό Γενετικό Προγραμματισμό και Εξελικτικές Στρατηγικές.

Representation of Evolution Algorithm (Αναπαράσταση Εξελικτικού αλγορίθμου)

```
Start
        t= 0;
        Initialize Population P(t) ;
        Evaluate genomes inside P(t) ;
        While termination clause is not true continue:
        Start
```

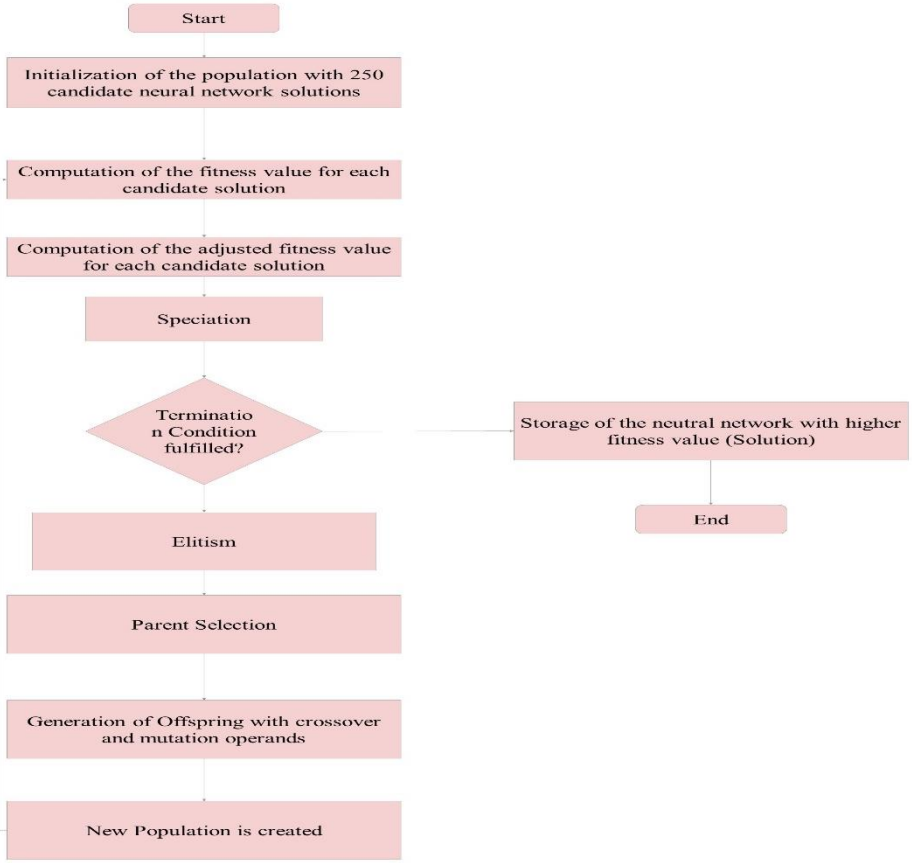
```
        t = t+1 ;
        Select P(t) from P(t-1) ;
        Change genomes in P(t) ;
        Evaluate genomes in P(t) ;
    End
End
```

Στον παραπάνω αλγόριθμο βλέπουμε μια επαναληπτική διαδικασία που τροποποιεί επανειλημμένα τον πληθυσμό σε κάθε γενιά και εξελίσσεται προκειμένου να επιτύχει μια καλύτερη απόδοση. Στην αρχή του αλγορίθμου δημιουργείται ένας αρχικός τυχαίος πληθυσμός και σε κάθε γενιά γίνονται τα ακόλουθα βήματα:

1. Αξιολόγηση - Κάθε γονιδίωμα του πληθυσμού παίρνει μια τιμή καταλληλότητας ανάλογα με τις επιδόσεις του στην επιθυμητή εργασία, που προκύπτει από μια fitness function.
2. Επιλογή - Αφού παίρνει μια τιμή καταλληλότητας, επιλέγονται τα άτομα για αναπαραγωγή, όπου εκείνα που ταιριάζουν περισσότερο στο περιβάλλον είναι πιο πιθανό να επιλεγούν, μιμούμενοι την αρχή της «επιβίωσης του ισχυρότερου».
3. Διασταύρωση συνδυάζει τα επιλεγμένα (genomes). Μία πιθανή υποψήφια λύση προκύπτει μέσω ενός συνδυασμού μεταξύ των γονιδίων δύο υπαρχόντων υποψηφίων λύσεων (γονέων). Ο σκοπός του crossover είναι να δημιουργήσει νέα άτομα που μπορούν να συνδυάσουν τα καλύτερα χαρακτηριστικά των γονέων και, ως εκ τούτου, να δημιουργήσουν μια καλύτερη απόδοση.

4. Μετάλλαξη - Ο τελεστής διασταύρωσης μπορεί να δημιουργήσει απογόνους που μοιάζουν πολύ με τους γονείς, προκαλώντας χαμηλή ποικιλομορφία στον πληθυσμό. Προκειμένου να έχουμε γενετική ποικιλομορφία στον πληθυσμό και να αποφευχθεί να έχουμε μόνο μια περιοχή χώρου αναζήτησης, υπάρχει μικρή πιθανότητα (αν ήταν πολύ μεγάλη ο αλγόριθμος θα έχει την τάση να εξελίσσεται τυχαία παρά με κατευθυνόμενη αναζήτηση) το παιδί του crossover να μεταλλαχθεί αλλάζοντας τυχαία κάποια γονίδια του γονιδιώματός του.

Ο εξελικτικός αλγόριθμος τερματίζεται όταν πληρείται η προϋπόθεση τερματισμού η οποία μπορεί και να ορίζεται είτε από έναν αριθμό γενεών για την εκτέλεση του αλγορίθμου ή όταν ο αλγόριθμος φτάσει μια συγκεκριμένη τιμή καταλληλότητας.



Εικόνα 7:Αναπαράσταση ΕΑ

2.4 Το Πρόβλημα της Σχεδίασης

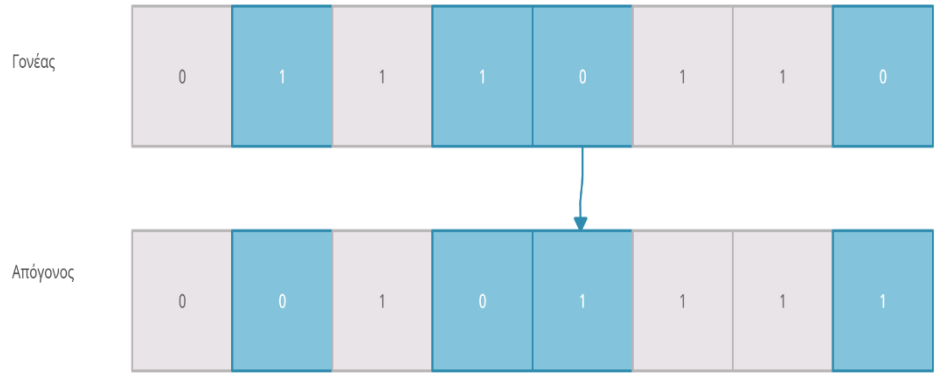
Οι ΕΑ έχουν βρει εφαρμογή σε διάφορους τύπους προβλημάτων, όπως βελτιστοποίηση αριθμητικών συναρτήσεων, προσαρμοσμένος έλεγχος και μηχανική μάθηση. Ένας από τους βασικούς λόγους που κάνει τους ΕΑ διαδεδομένους είναι ότι έχουν σχετικά μικρές απαιτήσεις σε παραμέτρους για να μπορούν να εφαρμοστούν, όπως:

- επιλογή κατάλληλης χαρτογράφησης του χώρου των λύσεων στο χώρο των χρωμοσωμάτων
- επιλογή σωστής Συνάρτησης Καταλληλότητας

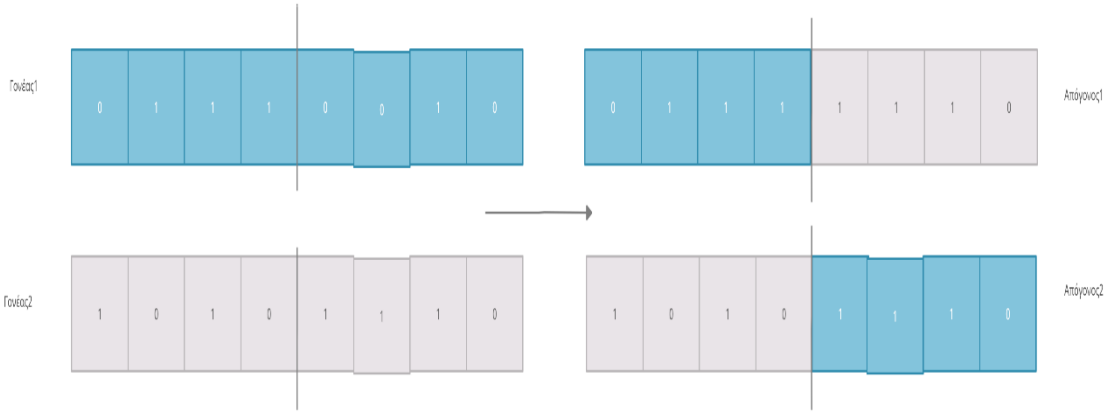
Για παράδειγμα, στην περίπτωση βελτιστοποίησης παραμέτρων, είναι συνήθης η παρουσίαση των λύσεων ως ένα διάνυσμα από πραγματικούς αριθμούς που κωδικοποιούν τις παραμέτρους αυτές. Όποια από τις κωδικοποιήσεις και αν επιλεχθεί, οι πράξεις της μετάλλαξης (Εικόνα 10) ή της διασταύρωσης (Εικόνα 11) μπορούν να γίνουν με άμεσο τρόπο ώστε να παραχθεί η γενετική παραλλαγή που απαιτείται. Στην περίπτωση αυτή, ο χρήστης πρέπει να αποφασίσει για έναν αριθμό άλλων παραμέτρων ελέγχου του ΕΑ, όπως το μέγεθος του πληθυσμού, την πιθανότητα με τη οποία γίνεται μετάλλαξη ή διασταύρωση των γονιδίων και τον κανόνα με τον οποίο επιλέγονται οι οργανισμοί που θα παίξουν το ρόλο του γονέα. Υπάρχουν αρκετές μελέτες που υποστηρίζουν ότι οι ΕΑ παρουσιάζουν σχετική ελαστικότητα ως προς το συνδυασμό των παραμέτρων ελέγχου, κάνοντας την εφαρμογή τους σε μία σειρά από προβλήματα, πολύ ελκυστική.

Σε άλλες κατηγορίες εφαρμογών, είναι απαραίτητο ο ΕΑ να προσαρμοστεί στο συγκεκριμένο πρόβλημα. Ένα σημαντικό ζήτημα που έχει να αντιμετωπίσει ο σχεδιαστής του ΕΑ είναι η κατάλληλη επιλογή του τρόπου με τον οποίο θα γίνει η αντιστοίχιση του χώρου των πιθανών λύσεων (**φαινότυπος**) στο πεδίο των χρωμοσωμάτων, καθώς από αυτό εξαρτάται και η ποιότητα της απόδοσης του αλγόριθμου. Μία γενικά αποδεκτή αρχή είναι ότι δύο λύσεις που παρουσιάζουν λειτουργικές ομοιότητες στο χώρο των λύσεων πρέπει να παρουσιάζουν και δομικές ομοιότητες στο πεδίο των χρωμοσωμάτων. Πολύ συχνά είναι απαραίτητος ο εκ νέου σχεδιασμός των γενετικών πράξεων ώστε αυτές να συμβαδίζουν με τη νέα αναπαράσταση. Οι

παραπάνω σχεδιαστικές επιλογές παίζουν το ρόλο της πόλωσης στο χώρο αναζήτησης της λύσης. Δεδομένης της κατάλληλης πόλωσης, ένας ΕΑ μπορεί να οδηγηθεί στη γρήγορη αναγνώριση αποτελεσματικών δομών και στη σύγκλισή του σε περιοχές λύσεων με υψηλότερη απόδοση.



Εικόνα 8:Πράξη μετάλλαξης σε έναν ΕΑ



Εικόνα 9:Πράξη διασταύρωσης σε έναν ΕΑ

3 ΠΕΡΙΓΡΑΦΗ ΜΕΘΟΔΩΝ ΕΚΠΑΙΔΕΥΣΗΣ

3.1 Νευροεξέλιξη

Η Νευροεξέλιξη, η εκπαίδευση δηλαδή ΤΝΔ με χρήση ΕΑ, έχει αναδειχθεί σε μία ισχυρή προσέγγιση για την υλοποίηση πολύπλοκων συμπεριφορών με τη χρήση πρακτόρων λογισμικού. Τέτοια δίκτυα έχουν τη δυνατότητα να δέχονται ως είσοδο έναν μεγάλο αριθμό ερεθισμάτων και να υλοποιούν μη γραμμικές χαρτογραφήσεις εξόδων που αντιπροσωπεύουν ενέργειες όπως κίνηση, χρήση εργαλείων ή όπλων κ.α. Η ανάδραση σε ΤΝΔ δίνει τη δυνατότητα αφομοίωσης πληροφορίας με την πάροδο του χρόνου, υλοποιώντας έτσι μίας μορφής μνήμη, η οποία εισάγει ευρωστία στην λήψη αποφάσεων ακόμα και σε περιβάλλοντα με ελλιπή πληροφόρηση. Τα παραπάνω χαρακτηριστικά των δικτύων καθιστούν την Νευρο-εξέλιξη ένα χρήσιμο εργαλείο για την υλοποίηση έξυπνων πρακτόρων σε περιβάλλοντα όπως βίντεο-παιχνίδια, επιτραπέζια παιχνίδια, κινούμενα ρομπότ και αυτόνομα οχήματα.[6]

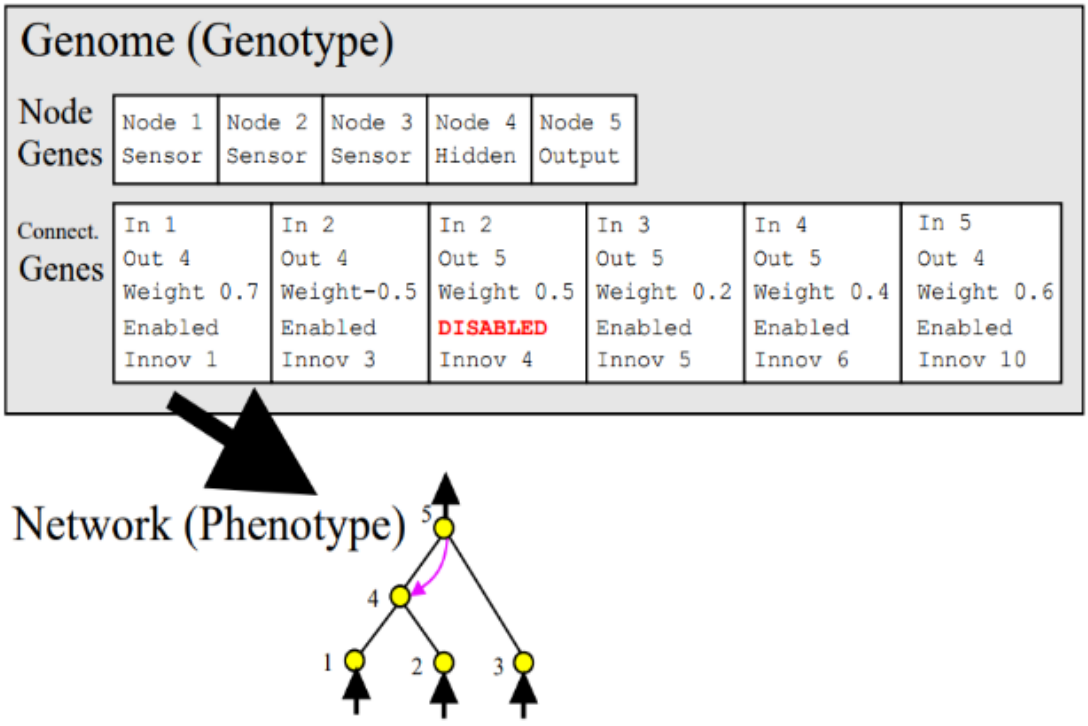
Στο πεδίο της Νευροεξέλιξης έχει γίνει μεγάλη πρόοδος όσον αφορά την εξέλιξη ευφυών συμπεριφορών ατομικών πρακτόρων σε πολύπλοκα περιβάλλοντα. Τέτοιες συμπεριφορές έχουν υλοποιηθεί με μεγάλη επιτυχία. Η επιτυχία σε τέτοια περιβάλλοντα απαιτεί την εξέλιξη μηχανισμών για την αλληλεπίδραση των πρακτόρων. Η εξέλιξη ανά πληθυσμούς παρέχει ένα φυσικό ομαδικό πλαίσιο και τα ΤΝΔ επιτρέπουν την υλοποίηση ομαδικής ανίχνευσης με φυσικό τρόπο.

3.2 TWEANNs - Neuro Evolution of Augmenting Topologies (NEAT)

Στην βιβλιογραφία υπάρχουν δύο τύποι Εξελικτικών Αλγόριθμών, πιο συγκεκριμένα είναι αυτοί που συνδυάζουν την εξεύρεση βέλτιστων βαρών σε συνδυασμό με τη βέλτιστη τοπολογία, οι οποίοι ανήκουν στην κατηγορία **Topology and Weight Evolution of Artificial Neural Networks** (TWEANNs) [8] και αυτοί που εξελίσσουν μόνο τα βάρη του νευρωνικού δικτύου (Conventional NeuroEvolution). Παρακάτω γίνεται περιγραφή του **NEAT**, ενός αλγορίθμου που ανήκει στην κατηγορία των TWEANNs. Ο NEAT συνδυάζει την αναζήτηση των κατάλληλων βαρών και της κατάλληλης τοπολογίας ξεκινώντας από απλά δίκτυα και στη συνέχεια εξελίσσοντάς τα σε κάθε γενιά. Ο ακριβής μηχανισμός περιγράφεται παρακάτω.

3.3 Γενετική Κωδικοποίηση μέσω NEAT

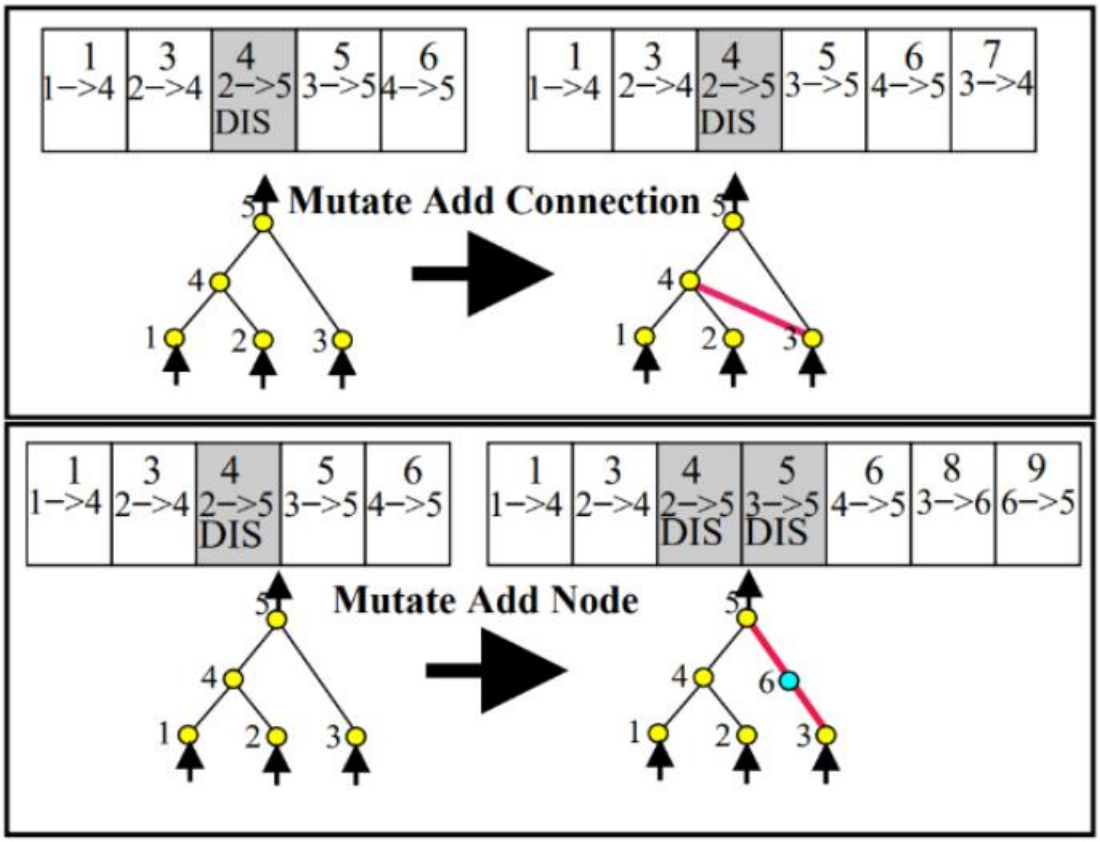
Ο NEAT χρησιμοποιεί ένα σύστημα γενετικής κωδικοποίησης . Κάθε γονότυπος αποτελείται από μία λίστα που περιέχει τα γονίδια κόμβων και μία λίστα με τα γονίδια ένωσης (Εικόνα 12). Κάθε γονίδιο κόμβου αποτελείται από έναν αριθμό καινοτομίας και το είδος του κόμβου (είσοδος, έξοδος, κρυφός). Κάθε γονίδιο ένωσης αποτελείται επίσης από έναν αριθμό καινοτομίας, δύο δείκτες που αναφέρονται στα γονίδια κόμβους στα οποία γίνεται η ένωση, το βάρος της ένωσης και τέλος ένα δυφίο ενεργοποίησης που δείχνει αν η ένωση είναι ενεργοποιημένη.



Εικόνα 10:Γενετική κωδικοποίηση στον αλγόριθμο NEAT [8]

Η διαδικασία της μετάλλαξης στον NEAT έχει δύο μορφές [4]. Η πρώτη έχει να κάνει με τα βάρη των ενώσεων που είτε αλλάζουν είτε όχι. Η δεύτερη μορφή μετάλλαξης είναι δομική. Η δομική μετάλλαξη γίνεται είτε με την πρόσθεση νέας ένωσης, προσθέτοντας ένα νέο γονίδιο ένωσης στο γονότυπο, είτε με την πρόσθεση νέου κόμβου, όπου μία υπάρχουσα ένωση χωρίζεται και στη θέση της εμφανίζεται ένας νέος κόμβος, προσθέτοντας το αντίστοιχο γονίδιο κόμβο στον γονότυπο. Το γονίδιο της αρχικής ένωσης απενεργοποιείται και προστίθενται δύο νέα στο γονότυπο. Η ένωση μεταξύ του πρώτου κόμβου και του νέου παίρνει τιμή βάρους ίση

με ένα (1) και η ένωση μεταξύ του νέου και του τελευταίου παίρνει την τιμή βάρους που είχε η αρχική ένωση.



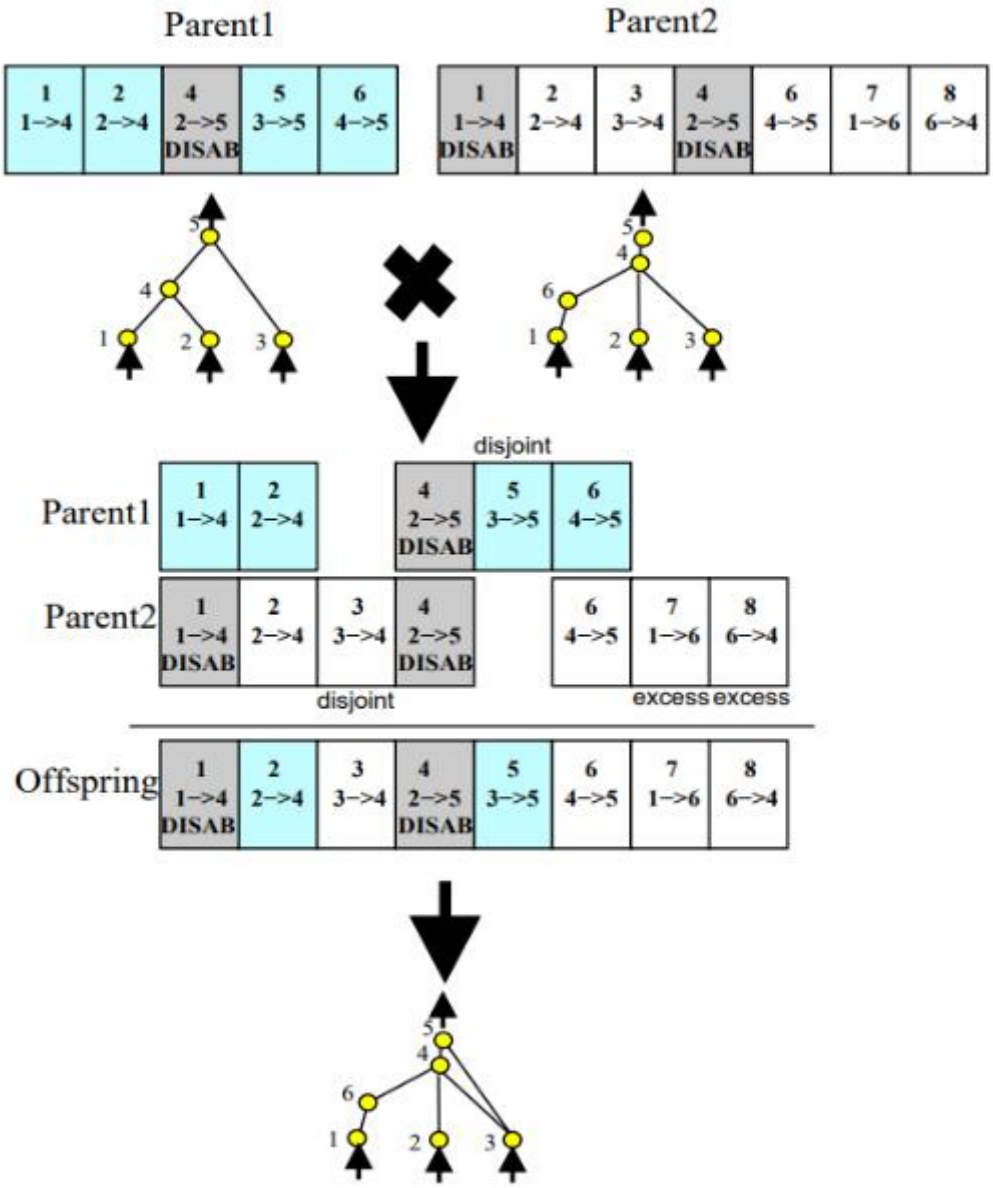
Εικόνα 11:Διαδικασίες μετάλλαξης στον αλγόριθμο NEAT [8]

3.4 Παρακολούθηση Γονιδίων με χρήση Ιστορικής Καταγραφής

Η τοπολογική εξέλιξη που χαρακτηρίζει τον αλγόριθμο εισάγει το πρόβλημα της αναγνώρισης πότε δύο ανόμοια δίκτυα μπορούν να διασταυρωθούν. Λύση στο παραπάνω δίνεται με τη χρήση της **Ιστορικής Καταγραφής** των γονιδίων. Δύο γονίδια με την ίδια προϊστορία παρουσιάζουν ίδια δομή και οι γονότυποι που τα περιέχουν μπορεί πιθανώς να διασταυρωθούν.

Το ρόλο της Ιστορικής Καταγραφής έχει ο αριθμός καινοτομίας των γονιδίων. Κάθε φορά που δημιουργείται ένα νέο γονίδιο, αυξάνεται ένας μετρητής. Μάλιστα εξασφαλίζεται ότι στην περίπτωση που έχει προκύψει η ίδια δομική μετάλλαξη στην ίδια γενιά, σε ανεξάρτητες μεταξύ τους μεταλλάξεις, τότε δίνεται ο ίδιος αριθμός καινοτομίας. Με αυτόν τον τρόπο είναι δυνατή η διασταύρωση δύο γονότυπων.

Κατά τη διασταύρωση, τα γονίδια κάθε γονότυπου τοποθετούνται κατά αύξουσα σειρά με βάση τον αριθμό καινοτομίας. Τα βάρη των γονιδίων ένωσης με τον ίδιο αριθμό και στους δύο γονείς είτε υιοθετούνται τυχαία από έναν από τους δύο γονείς είτε υπολογίζεται ο μέσος όρος. Τα υπόλοιπα γονίδια κληρονομούνται από τον γονέα με την καλύτερη επίδοση ή και από τους δύο στην περίπτωση ίσης επίδοσης. Τέλος, στα ανενεργά γονίδια δίνεται πιθανότητα ενεργοποίησης.



Εικόνα 12: Συνδυασμός διαφορετικών τοπολογιών με χρήση ιστορικής καταγραφής [8]

3.5 Προστασία Καινοτομιών

Τα μικρότερα ΝΔ θέλουν λιγότερο χρόνο μέχρι να βελτιστοποιηθούν σε σχέση με τα μεγαλύτερα. Αυτό το χαρακτηριστικό καθιστά αδύναμα τα δίκτυα που έχουν προσφάτως αυξηθεί τοπολογικά και, επομένως, ευάλωτα σε εκτοπισμό από τον πληθυσμό σε μικρό αριθμό γενεών. Ο αλγόριθμος NEAT προσφέρει προστασία από αυτό το φαινόμενο με τη χρήση της **ειδογένεσης**. Ακολουθώντας την φιλοσοφία ότι νέες ιδέες πρέπει να έχουν αρκετό χρόνο ώστε να δείξουν τις πλήρεις δυνατότητές τους, ο NEAT διαχωρίζει τον πληθυσμό σε είδη(species) ώστε ο ανταγωνισμός των μελών να γίνεται μόνο εντός κάθε ομάδας. Με αυτόν τον τρόπο προστατεύονται οι τοπολογικές καινοτομίες και αποτρέπεται η ανούσια αντικατάσταση μικρότερων δικτύων από μεγαλύτερα.

Ο διαχωρισμός των γονιδιωμάτων σε είδη γίνεται με χρήση του δείκτη ομοιότητας δ σύμφωνα με την παρακάτω εξίσωση [8]:

$$\boldsymbol{\delta = \frac{c_1 E}{N} + \frac{c_2 \Gamma}{N} + c_3 \overline{W}}$$

όπου E ο αριθμός επιπλέον γονιδίων, Γ ο αριθμός γονιδίων που αποτελούν ασυνέχεια, W η μέση διαφορά στα βάρη, N ο αριθμός γονιδίων του μεγαλύτερου γονιδιώματος και C1,C2,C3 οι δείκτες που ορίζουν την σημαντικότητα κάθε παράγοντα.

Κατά την ταξινόμηση ενός γονιδιώματος, επιλέγεται τυχαία ένα υπάρχον μέλος ενός είδους και γίνεται η σύγκριση. Αν ο δείκτης είναι μεγαλύτερος από ένα κατώφλι., τότε το γονιδίωμα κατατάσσεται στο είδος αυτό. Στην περίπτωση που δεν πληρείται η παραπάνω προϋπόθεση, τότε ο έλεγχος συνεχίζεται και στα υπόλοιπα είδη του πληθυσμού. Αν δεν

υπάρχει ομοιότητα με κανένα είδος της προηγούμενης γενιάς, τότε το γονιδίωμα αποτελεί ένα νέο είδος.

Για τον μηχανισμό της αναπαραγωγής, χρησιμοποιείται η μέθοδος της *ρητής κοινής καταλληλότητας* (explicit fitness sharing), σύμφωνα με την οποία όλοι οι οργανισμοί που ανήκουν σε ένα είδος μοιράζονται το βαθμό καταλληλότητας στην ειδίκευσή τους. Ο βαθμός καταλληλότητας του είδους είναι και αυτός που καθορίζει τον αριθμό των απογόνων που επιτρέπεται να έχει το είδος αυτό.

Για να γίνει αυτό, στην αρχή υπολογίζεται ο προσαρμοσμένος βαθμός καταλληλότητας *fi'* για κάθε οργανισμό *i* σύμφωνα με τον παρακάτω τύπο [8].

$$\boldsymbol{fi'} = \frac{\boldsymbol{f_i}}{\sum_{\boldsymbol{j=1}}^{\boldsymbol{n}} \boldsymbol{sh}(\boldsymbol{\delta}(\boldsymbol{i},\boldsymbol{j}))}$$

Όπως φαίνεται από την σχέση αυτή, η καταλληλότητα ενός οργανισμού συνδέεται άμεσα με το πλήθος των μελών ενός είδους. Στη συνέχεια υπολογίζεται ο μέσος βαθμός καταλληλότητας ενός είδους και ο αθροιστικός βαθμός καταλληλότητας του πληθυσμού

$$\overline{\boldsymbol{F}}_{\boldsymbol{tot}} = \sum_{\boldsymbol{k}} \overline{\boldsymbol{F}}_{\boldsymbol{k}}$$

Τέλος, ο αριθμός των απογόνων κάθε είδους δίνεται από τη σχέση:

$$\boldsymbol{n_k} = \frac{\overline{\boldsymbol{F}}_{\boldsymbol{k}}}{\overline{\boldsymbol{F}}_{\boldsymbol{tot}}}\ \boldsymbol{P}$$

όπου P ο συνολικός αριθμός των οργανισμών του πληθυσμού.

3.6 Διατήρηση Μικρού Μεγέθους

Όπως αναφέρθηκε, ο NEAT εφαρμόζει την εξέλιξη ΝΔ με την παράλληλη αύξηση της διάστασής τους. Ο αλγόριθμος αυτός σκοπεύει στην εύρεση των βέλτιστων βαρών συντηρώντας παράλληλα και την ελάχιστη δυνατή τοπολογία. Αυτό το καταφέρνει, κάνοντας initialize τον πληθυσμό με ΤΝΔ χωρίς κρυφά στρώματα και με τυχαία βάρη, τα οποία στη συνέχεια αυξάνει.

3.7 Real-time Neuro Evolution of Augmenting Topologies (rtNEAT)

Οι περισσότεροι εξελικτικοί αλγόριθμοι είναι κατασκευασμένοι ώστε να τρέχουν offline, πράγμα που σημαίνει ότι άνθρωπος-χειριστής δεν έχει καμία αλληλεπίδραση με τους πράκτορες κατά τη διαδικασία της εξέλιξης. Οι Stanley παρουσιάσαν τον αλγόριθμο rtNEAT βασιζόμενοι στην ιδέα ότι σε ένα παιχνίδι πραγματικού χρόνου μπορούν να δοκιμάζονται όλοι οι πράκτορες την ίδια στιγμή και έτσι να συλλέγονται δεδομένα ως προς την επίδοσή τους και να εξελίσσονται συνεχώς.

Η εξέλιξη σε πραγματικό χρόνο εισάγει περιορισμό ως προς την δημιουργία νέων απογόνων. Η ταυτόχρονη αντικατάσταση ολόκληρου του πληθυσμού θα φαινόταν περίεργη στον χρήστη καθώς οι πράκτορες θα άλλαζαν συμπεριφορά απότομα. Η λύση που προτείνεται είναι η αντικατάσταση ενός πράκτορα κάθε φορά, αυτού με την χειρότερη απόδοση. Όπως φανερώνει και το όνομα του αλγορίθμου, ο rtNEAT βασίζεται στον NEAT, που περιγράφηκε παραπάνω, υιοθετώντας τα ισχυρά στοιχεία του (χρήση ειδογένεσης και σταδιακή αύξηση τοπολογίας) και επεκτείνοντας τον στο πεδίο της εξέλιξης σε πραγματικό χρόνο. Κάθε *n* παλμούς του ρολογιού εισάγεται ένας νέος πράκτορας στον πληθυσμό, αντικαθιστώντας έναν

αδύναμο. Η ειδογένεση διατηρείται διαλέγοντας βάσει πιθανοτήτων τους γονείς του νέου πράκτορα και προσεκτικά επιλέγοντας αυτόν που θα αντικατασταθεί.

4 ΣΥΝΑΛΛΑΓΕΣ ΜΕ ΕΚΜΑΘΗΣΗ ΕΝΙΣΧΥΣΗΣ - TRADING WITH REINFORCEMENT LEARNING

4.1 Συνάρτηση Χρησιμότητας – Utility function: Sharpe‘s Ratio

Μια μετρική που χρησιμοποιείται συνήθως στον χρηματοοικονομικό τομέα και πήρε το όνομα της από τον Αμερικανό οικονομολόγο, William Sharpe είναι το Sharpe Ratio (ή Sharpe Index ή Modified Sharpe Ratio) όπου είναι ένα πολύπλοκο μέτρο που χρησιμοποιεί την τυπική απόκλιση του stock ή του χαρτοφυλακίου για τη μέτρηση της μεταβλητότητας. Αυτός ο υπολογισμός μετρά την αυξητική ανταμοιβή του αυξητικού κινδύνου. Όσο υψηλότερη είναι η αναλογία Sharpe, τόσο μεγαλύτερη είναι η πιθανή απόδοση της επένδυσης σε σχέση με το ποσό του κινδύνου που αναλαμβάνεται και, επομένως, τόσο καλύτερη είναι η επένδυση. Η αναλογία μπορεί να χρησιμοποιηθεί για την αξιολόγηση μιας μετοχής ή μιας επένδυσης ή ενός ολόκληρου χαρτοφυλακίου. Ο τύπος Sharpe Ratio = (συνολική απόδοση μείον το ποσοστό απόδοσης χωρίς κίνδυνο) διαιρούμενο με την τυπική απόκλιση του χαρτοφυλακίου.

Για μια χρονική σειρά επενδυτικών αποδόσεων, η αναλογία Sharpe μπορεί να υπολογιστεί ως [9]:

$$S_T = \frac{\boldsymbol{Average(Rt)}}{\boldsymbol{Standard\ Deviation\ (Rt)}} \quad \boldsymbol{for\ interval\ t} = \mathbf{1}, \dots, \boldsymbol{T}$$

όπου Rt είναι η απόδοση της επένδυσης για την περίοδο διαπραγμάτευσης t. Διαισθητικά, η αναλογία Sharpe επιβραβεύει τις επενδυτικές στρατηγικές που βασίζονται σε λιγότερο ασταθείς τάσεις έτσι ώστε να προκύψει κέρδος.

Ο trader θα προσπαθήσει να μεγιστοποιήσει την αναλογία Sharpe για μια δεδομένη χρονική σειρά τιμών. Για αυτό το Project , η συνάρτηση trader παίρνει τη μορφή ενός νευρώνα:

Trader function – Activation function evaluate output neuron: [9]

$$\boldsymbol{F_t = tanh\,(w^T x_t)}$$

Όπου M είναι ο αριθμός εισόδου χρονοσειρών: [9]

$$\boldsymbol{W \in R^{M+2}}$$

Το διάνυσμα εισόδου (Input vector) : [9]

$$\boldsymbol{x_t = [1, r_t, \ldots., r_{t-M}, F_{t-1}]}$$

Και η απόδοση μεταξύ γειτονικών χρονικών στιγμών (return): [9]

$$\boldsymbol{r_t = p_t - p_{t-1}}$$

Σημειώστε ότι rt είναι η διαφορά στην αξία του περιουσιακού στοιχείου μεταξύ της τρέχουσας περιόδου t και της προηγούμενης περιόδου. Ως εκ τούτου, $\mathbf{r_t}$ είναι η απόδοση ενός μεριδίου του περιουσιακού στοιχείου που αγοράστηκε τη στιγμή t - 1. Επίσης, η συνάρτηση $\boldsymbol{F_t}$ παίρνει τιμές στο διάστημα [-1,1] και αντιπροσωπεύει τη θέση διαπραγμάτευσης στο χρονοδιάγραμμα.

4.2 Trading Positions – Θέσεις συναλλαγής

Υπάρχουν τρεις τύποι θέσεων που μπορεί ο trader να πάρει:

- **Θέση Long (Long position)**

Long position παίρνουμε όταν η trader function είναι θετική άρα το $\boldsymbol{F_t} > 0$. Σε αυτήν την περίπτωση, ο έμπορος (trader) αγοράζει ένα περιουσιακό στοιχείο στην τιμή $\mathbf{p_t}$ και ελπίζει ότι θα εκτιμηθεί έως την περίοδο t + 1.

- **Θέση Short (Short Position)**

Short Position παίρνουμε όταν η trader function είναι αρνητική άρα το $\boldsymbol{F_t} < 0$. Σε αυτήν την περίπτωση, ο έμπορος πουλάει ένα περιουσιακό στοιχείο που δεν κατέχει στην τιμή $\mathbf{p_t}$ με την προσδοκία της αύξησης των μετοχών του κατά την περίοδο t + 1. Εάν η τιμή την χρονική στιγμή t + 1 είναι υψηλότερη, τότε ο έμπορος(trader) αναγκάζεται να αγοράσει στην υψηλότερη τιμή t + 1 για την εκπλήρωση της σύμβασης (συμβολαίου). Ωστόσο εάν η τιμή την χρονική στιγμή t + 1 είναι χαμηλότερη, τότε ο έμπορος έχει κέρδος.

- **Θέση Neutral (Neutral Position)**

Η ουδέτερη θέση είναι όταν η trader function είναι μηδενική άρα το $\boldsymbol{F_t} = 0$. Σε αυτήν την περίπτωση, το αποτέλεσμα τη χρονική στιγμή t+1 δεν επηρεάζει τα κέρδη του εμπόρου, συνεπώς δεν θα υπάρχει ούτε κέρδος ούτε ζημία.

Έτσι λοιπόν η trader function $\boldsymbol{F_t}$ αντιπροσωπεύει τις εκμεταλλεύσεις στην περίοδο t. Δηλαδή, $\boldsymbol{n_t}$ = μ * $\boldsymbol{F_t}$ μετοχές που αγοράστηκαν (long position) ή πουλήθηκαν (short position), όπου μ είναι ο μέγιστος δυνατός αριθμός μετοχών ανά συναλλαγή. Η απόδοση το return στο

χρόνο t, λαμβάνοντας υπόψη την απόφαση της προηγούμενη χρονικής στιγμής Ft-1 υπολογίζεται από τον παρακάτω τύπο [9]:

$$\boldsymbol{R_t = \mu\,(F_{t-1}\cdot r_t - \delta\,|F_t - F_{t-1}|)}$$

Όπου δ είναι το κόστος του transaction την περίοδο t.

Εάν το $\boldsymbol{F_t}$ = $\boldsymbol{F_t}$ -1 (δηλαδή καμία αλλαγή στην επένδυσή μας σε αυτή την περίοδο) τότε δεν θα υπάρξει ποινή συναλλαγής. Διαφορετικά, η ποινή είναι ανάλογη με τη διαφορά στις μετοχές που κατέχονται. Ο πρώτος όρος (μ* $\boldsymbol{F_{t-1}}$ *$\mathbf{r_t}$) είναι η απόδοση που προκύπτει από την επενδυτική απόφαση από την περίοδο t - 1. Για παράδειγμα, αν μ = 20 μετοχές, η απόφαση ήταν να αγοράσετε το μισό μέγιστο επιτρεπόμενο F_{t-1}=0.5 και κάθε μετοχή αυξήθηκε κατά 8 μονάδες άρα rt = 8, αυτός ο όρος θα είναι 80, το συνολικό κέρδος απόδοσης (αγνοώντας τις κυρώσεις συναλλαγών που προέκυψαν κατά τη διάρκεια περίοδος t).

4.3 Implementation

Στο Project μας υλοποιούμε ένα έμπορο περιουσιακών στοιχείων (πράκτορας) χρησιμοποιώντας επαναλαμβανόμενη μάθηση ενίσχυσης (RRL). Ως συνάρτηση ανταμοιβής θα χρησιμοποιήσουμε την μετρική του sharpe όπου και θα προσπαθήσουμε να την μεγιστοποιήσουμε.

Η λογική της συγκεκριμένης εργασίας είναι ότι αναλόγως με το αν σε κάποια χρονική στιγμή αγοράζεις και σε κάποια άλλη χρονική στιγμή πουλάς μπορεί ενδεχομένως να προκύψει κάποιο κέρδος, μπορούμε δηλαδή αυτήν την λογική του να εκτελεστούν αυτές οι αγοροπωλησίες να την υλοποιήσουμε μέσω μηχανικής μάθησης όπου το fitness function θα

τρέχει σειριακά αύτη την διαδικασία, με τα διαφορετικά Input vectors και λαμβάνοντας υπόψιν τις ενέργειες που έχει βγάλει το ίδιο το νευρωνικό δίκτυο για την προηγούμενη χρονική στιγμή θα φτιάξει το διάνυσμα των Returns R.

Το source dataset που έχουμε στην διάθεση μας αποτελείται από Trade transactions ανά timestamp οπως φαίνεται και στην παρακάτω εικόνα. Το συγκεκρίμενο dataset έχει γίνει download μέσω ενός API (Binance). Το Binance API είναι μια μέθοδος που επιτρέπει να συνδεθείς με τους διακομιστές Binance μέσω Python ή αρκετών άλλων γλωσσών προγραμματισμού και να αντλήσεις δεδομένα προηγούμενων μηνών και όχι μόνο σου παρέχει ακόμα και την δυνατότητα να αυτοματοποιήσεις συναλλαγές καθώς διαθέτει ένα RESTful API που χρησιμοποιεί αιτήματα HTTP για αποστολή και λήψη δεδομένων.

price,isbuyermaker,quantity,quoteqty,trade_time,api_trade_ts
30670.29,0,0.001084,33.246593,1611055402194,1611055598440
30670.29,0,0.001086,33.307934,1611055402196,1611055598440
30668.59,1,0.000799,24.504204,1611055402312,1611055598440
30669.51,0,0.003129,95.9649,1611055402369,1611055598440
30669.74,0,0.002555,78.36118,1611055402369,1611055598440
30670.1,0,0.000799,24.50541,1611055402369,1611055598440
30671.13,0,0.004166,127.775925,1611055402369,161105559844(
30669.51,0,0.00312,95.68887,1611055402548,1611055598440
30669.51,0,9e-06,0.2760256,1611055402637,1611055598440
30671.13,0,0.001261,38.676296,1611055402637,1611055598440
30671.13,0,1.8e-05,0.55208033,1611055402734,1611055598440
30671.15,0,0.00544,166.85106,1611055402798,1611055598440
30671.15,0,5e-06,0.15335575,1611055402805,1611055598440
30671.17,0,0.00544,166.85117,1611055402853,1611055598440
30671.17,0,7e-06,0.2146982,1611055402862,1611055598440
30671.51,0,0.00409,125.44647,1611055402909,1611055598440
30671.51,0,9e-06,0.2760436,1611055403070,1611055598440
30671.61,0,0.0008,24.537289,1611055403086,1611055598440
30673.12,0,0.0008,24.538496,1611055403121,1611055598440
30673.99,0,0.00228,69.9367,1611055403133,1611055598440

Εικόνα 13:Source Data

Τα source data βάσει του μοντέλου που θα αναλυθεί παρακάτω θα μετασχηματιστούν με την δομή που φαίνεται στην παρακάτω εικόνα:

```
In [18]: import pickle
         from IPython.display import display
         import pandas as pd
         pd.get_option("display.max_columns")

Out[18]: 20

In [25]: # settings to display all columns
         pd.set_option("display.max_columns", 25)
         Input_vector = open('10000_200_Training_protocol_4','rb')
         Input_vector_load=pickle.load(Input_vector)

In [26]: display(Input_vector_load)
```

	0	1	2	3	4	5	6	7	8	9	10	11	...	190	191	192	193	194	195	196	197	198	199	200	201
0	1	0.00	-1.70	0.92	0.23	0.36	1.03	-1.62	0.00	1.62	0.00	0.02	...	1.51	1.51	1.86	0.00	0.99	0.17	-1.71	0.00	-1.31	0.01	0.01	0
1	1	-1.70	0.92	0.23	0.36	1.03	-1.62	0.00	1.62	0.00	0.02	0.00	...	1.51	1.86	0.00	0.99	0.17	-1.71	0.00	-1.31	0.01	0.01	0.01	Ft
2	1	0.92	0.23	0.36	1.03	-1.62	0.00	1.62	0.00	0.02	0.00	0.02	...	1.86	0.00	0.99	0.17	-1.71	0.00	-1.31	0.01	0.01	0.01	-1.53	Ft
3	1	0.23	0.36	1.03	-1.62	0.00	1.62	0.00	0.02	0.00	0.02	0.00	...	0.00	0.99	0.17	-1.71	0.00	-1.31	0.01	0.01	0.01	-1.53	0.50	Ft
4	1	0.36	1.03	-1.62	0.00	1.62	0.00	0.02	0.00	0.02	0.00	0.34	...	0.99	0.17	-1.71	0.00	-1.31	0.01	0.01	0.01	-1.53	0.50	0.60	Ft
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
9795	1	-0.54	-0.46	-0.14	-0.86	-0.05	-0.95	-0.56	-0.38	-0.01	-1.03	0.75	...	-1.27	-0.24	-0.13	-0.23	-0.70	-0.45	-0.35	-0.90	1.28	-0.79	0.00	Ft
9796	1	-0.46	-0.14	-0.86	-0.05	-0.95	-0.56	-0.38	-0.01	-1.03	0.75	2.10	...	-0.24	-0.13	-0.23	-0.70	-0.45	-0.35	-0.90	1.28	-0.79	0.00	0.00	Ft
9797	1	-0.14	-0.86	-0.05	-0.95	-0.56	-0.38	-0.01	-1.03	0.75	2.10	-0.50	...	-0.13	-0.23	-0.70	-0.45	-0.35	-0.90	1.28	-0.79	0.00	0.00	0.00	Ft
9798	1	-0.86	-0.05	-0.95	-0.56	-0.38	-0.01	-1.03	0.75	2.10	-0.50	-0.60	...	-0.23	-0.70	-0.45	-0.35	-0.90	1.28	-0.79	0.00	0.00	0.00	0.00	Ft
9799	1	-0.05	-0.95	-0.56	-0.38	-0.01	-1.03	0.75	2.10	-0.50	-0.60	-0.70	...	-0.70	-0.45	-0.35	-0.90	1.28	-0.79	0.00	0.00	0.00	0.00	2.27	Ft

9800 rows × 202 columns

Εικόνα 14:Πίνακας διανυσμάτων εισόδου για 10000 timestamps και time window 200 (Input Dataset)

Για τις απαιτήσεις της εργασίας υλοποιήσαμε μια function σε python που κάνει evaluate και καθορίζει την εξέλιξη αλλά και το πόσο καλά τα πάει ο αλγόριθμος μας. Στο συγκεκριμένο Project δοκιμάστηκαν αρκετές πειραματικές δοκιμές έτσι ώστε να δούμε την συμπεριφορά του Neat δεδομένου των εισόδων του. Ωστόσο λόγω της έλλειψης από resources(CPU/RAM) καθότι

το συγκεκριμένο πρόβλημα προϋποθέτει την χρήση ενός καλού μηχανήματος αφού απαιτεί την εκτέλεση/προσπέλαση εκατοντάδων χιλιάδων εγγραφών, προκειμένου να κάνουμε την ανάλυση μας χρησιμοποιήσαμε data sets των 1000 & 10.000 χρονικά στιγμών . Ας γίνουμε όμως λίγο πιο αναλυτικοί με την διαδικασία.

Προετοιμασία data (Data preparation):

Αφού ανακτήθηκαν σε Excel τιμές κλεισίματος (trades) (βλέπε Εικόνα 15) , μέσω του παρακάτω κώδικα μπορέσαμε να φτιάξουμε τα διανύσματα εισόδου με βάση την περίοδο (T) και το βήμα(time window M) , ουσιαστικά συμφώνα και με τις παραπάνω πληροφορίες προσπαθήσαμε να φτιάξουμε ένα διάνυσμα εισόδου το όποιο αποτελείται

1. από την μονάδα,
2. τις διαφορές μεταξύ γειτονικών χρονικών στιγμών
3. και την ενέργεια της προηγουμένης χρονικής στιγμής

Time	price	difference
t0	30670,29	
t1	30670,29	0
t2	30668,59	-1,7
t3	30669,51	0,92
t4	30669,74	0,23
t5	30670,1	0,36
t6	30671,13	1,03
t7	30669,51	-1,62
t8	30669,51	0
t9	30671,13	1,62
t10	30671,13	0
t11	30671,15	0,02
t12	30671,15	0
t13	30671,17	0,02
t14	30671,17	0
t15	30671,51	0,34
t16	30671,51	0
t17	30671,61	0,1
t18	30673,12	1,51

Εικόνα 15:Αρχικό Dataset με τιμές συναλλαγών/ διαφορές τιμών μεταξύ γειτονικών χρονικών στιγμών

Για να γίνει αντιληπτό όπως προαναφέραμε χρησιμοποιήσαμε μια περίοδο T=1000 όπου 1000 αντιστοιχεί σε 1000 χρονικά υστερημένες στιγμές (από το παρελθόν με κατεύθυνση προς το μέλλον).Ο κώδικας μας ο οποίος είναι υλοποιημένος σε **python** αρχικά διαβάζει τα source data μας από excel και τα κάνει transform βάσει του μοντέλου μας (Εικόνα 16) - μετέπειτα αποθηκεύει την final μορφή των data μας με την οποία θα κάνουμε feed το νευρωνικό μας δίκτυο σε ένα data frame το οποίο γίνεται serialize και αποθηκεύεται σε byte μορφή μέσω της βιβλιοθήκης Pickle ώπου και κρατάμε την τελική μορφή του διανύσματος εισόδου.

In [1]:
```
import pickle
from IPython.display import display
import pandas as pd
pd.get_option("display.max_columns")
```

Out[1]: 20

In [2]:
```
# settings to display all columns
pd.set_option("display.max_columns", 25)
Input_vector = open('1000_10_Training_protocol_4','rb')
Input_vector_load=pickle.load(Input_vector)
```

In [3]:
```
Input_vector_load
```

Out[3]:

	0	1	2	3	4	5	6	7	8	9	10	11
0	1	0.00	-1.70	0.92	0.23	0.36	1.03	-1.62	0.00	1.62	0.00	0
1	1	-1.70	0.92	0.23	0.36	1.03	-1.62	0.00	1.62	0.00	0.02	Ft
2	1	0.92	0.23	0.36	1.03	-1.62	0.00	1.62	0.00	0.02	0.00	Ft
3	1	0.23	0.36	1.03	-1.62	0.00	1.62	0.00	0.02	0.00	0.02	Ft
4	1	0.36	1.03	-1.62	0.00	1.62	0.00	0.02	0.00	0.02	0.00	Ft
...	...	...	...	...	...	...	...	...	...	...	...	...
985	1	0.21	1.04	1.24	1.51	1.51	0.79	0.50	0.02	-1.93	0.00	Ft
986	1	1.04	1.24	1.51	1.51	0.79	0.50	0.02	-1.93	0.00	-0.56	Ft
987	1	1.24	1.51	1.51	0.79	0.50	0.02	-1.93	0.00	-0.56	-0.33	Ft
988	1	1.51	1.51	0.79	0.50	0.02	-1.93	0.00	-0.56	-0.33	-1.51	Ft
989	1	1.51	0.79	0.50	0.02	-1.93	0.00	-0.56	-0.33	-1.51	-1.51	Ft

990 rows × 12 columns

Εικόνα 17:Πίνακας διανυσμάτων εισόδου στην final μορφή τους για 1000 timestamps και time window 10

Ο παραπάνω πίνακας παριστάνει την μορφή των μονοδιάστατων εισόδων του νευρωνικού μας για Period 1000 και βήμα 10 χρονικές στιγμές. Πιο συγκεκριμένα για t0 = 0 το διάνυσμα εισόδου είναι το ακόλουθο:

1	0.00	-1.70	0.92	0.23	0.36	1.03	-1.62	0.00	1.62	0.00	0

Εικόνα 18:Input Vector t0=0

το όποιο έχει μέγεθος M+2 όπου M είναι το time step (βήμα) των χρονικά υστερημένων χρονικών στιγμών. Το διάνυσμα εισόδου λοιπόν αποτελείται από την μονάδα στην θέση (1,1) η θέση (1,2) υπολογίζεται από την διαφορά της τιμής μεταξύ των γειτονικών χρονικών στιγμών $\boldsymbol{t_0}$ και $\boldsymbol{t_1}$ που στην προκειμένη περίπτωση είναι 0 καθότι όπως μπορείτε να δείτε και στην **Εικόνα 17** για $\boldsymbol{t_0}$ έχουμε trade value 30670,29 και για $\boldsymbol{t_1}$ έχουμε trade value 30670,29 άρα η διαφορά τους είναι 0, στην τελευταία θέση του διανύσματος εισόδου έχουμε την ενέργεια της προηγούμενης χρονικής στιγμής:

Γενικότερα τα διανύσματα εισόδου έχουν την παρακάτω μορφή

$$[\mathbf{1},\quad \boldsymbol{r_1 = p_{t1} - p_{t0}, r_2 = p_{t2} - p_{t1} \ldots r_M = p_M - p_{M-1}, F_0}]$$

$$[\mathbf{1},\ \boldsymbol{r_2 = p_{t2} - p_{t1}, r_3 = p_{t3} - p_{t2} \ldots r_{M+1} = p_{M+1} - p_{M-1+1}, F_1}]$$

.

.

.

$$[\mathbf{1},\quad \boldsymbol{r_{T-M} = p_{T-M} - p_{T-M-1}} \quad \ldots \quad \boldsymbol{r_T = p_T - p_{T-1}, F_{T-M-1}}]$$

(Επεξεργασία δεδομένων) Data processing:

Ο NEAT σταθμίζει το επίπεδο εισόδου με το αντίστοιχο βάρος όπου και περνάει από μια Sigmoid Transfer Function F(t). Σκοπός μας είναι να επανατροφοδοτούμε το input layer κάθε φορά με διανύσματα εισόδου μεγέθους M+2 χρησιμοποιώντας κάθε φορά την έξοδο του νευρωνικού που προκύπτει από την ενέργεια της προηγούμενη χρονικής στιγμής. Όλη αυτή η διαδικασία θα μας δώσει μια σειρά από ενέργειες $\boldsymbol{F_0}$, $\boldsymbol{F_1}$, $\boldsymbol{F_2}$… $\boldsymbol{F_{T-M-1}}$. Βασιζόμενοι στο paper[9] το ποια θα πρέπει να είναι η σωστή εκμετάλλευση των ενεργειών την συσχετίζουμε με μια συνάρτηση κέρδους $\boldsymbol{R_t}$ που μου λέει ποσό κερδίζω σε κάθε χρονική στιγμή με μ να θεωρούμε το κβαντο της ποσότητας που πουλάς η αγοράζεις σε κάθε χρονική στιγμή. Άρα η απόφαση καθορίζεται κάθε φορά από την έξοδο μας (συνάρτηση μεταφοράς F(t)) και το πόσο κερδίζεις ή χάνεις κάθε φορά (return $\boldsymbol{R_t}$). Συνεπώς καταλήγουμε με μια ακολουθία συνεχών τιμών του F(t) αλλά δεν ξέρουμε ποια θα είναι η σωστή έκβαση ,για να βγει το επιδιωκόμενο κέρδος. Αυτό λοιπόν που θα προσπαθήσουμε να μεγιστοποιήσουμε είναι το $\boldsymbol{R_t}$:return αφού αυτή η σχέση μεταφράζει τις αποφάσεις σε κέρδος. Επίσης η οπτική μας για αυτό το Project είναι να δίνουμε βάση στην μεταβολή της τιμής :price difference και όχι στο price αυτό καθ' αυτό.

Ξεφεύγοντας λοιπόν από την δομή του Paper, το Input layer μας δεν θα αλλάξει, η συνάρτηση μεταφοράς στο τελευταίο επίπεδο θα είναι **Sigmoid tanh** έτσι ώστε να αναγκάσουμε την έξοδο μας να ανήκει μεταξύ του διαστήματος **[-1 ,1]** και από εκεί και ύστερα ο NEAT θα αποφασίσει την κατάλληλη για αυτόν δομή και θα αναπτυχθεί με έναν εξελικτικό τρόπο, αυτή λοιπόν την εξέλιξη θα την καθορίσει η Objective function: $\boldsymbol{R_t}$. Ο NEAT αναπτύσσει

διάφορα νευρωνικά όπου το καθένα δίνει από ένα Ft & Rt ανά νευρωνικό, η λογική με τα οποία εκπαιδεύονται είναι να αναγκάσεις τον NEAT να κάνει mutation/crossover ώστε να αλλάξει τις δομές των νευρωνικών, η αξιολόγηση του θα πρέπει να γίνεται με βάση την καταγραφή της συμπεριφοράς μετα από T period άρα η αποτίμηση γίνεται ανά T ενέργειες . Για παράδειγμα σε ένα παιχνίδι που θα έπρεπε ο ΕΑ να βρει τον δρόμο σε έναν λαβύρινθο θα ξεκινούσαμε με έναν πληθυσμό και θα αφήναμε τον πράκτορα να κάνει 10 ενέργειες προς το μέλλον, μετά και την $10^{η}$ χρονική στιγμή βλέπεις πόσο καλά τα έχει πάει. Άρα ξεκινάμε από μια γενιά 0 με μια αυθαίρετη αρχιτεκτονική και αφήνουμε να γίνουν 10 ενέργειες, έπειτα συλλέγουμε το σύνολο του fitness και βάσει αυτού γίνονται evolve, έπειτα ξανά αφήνουμε να κάνουν πάλι 10 ενέργειες και ούτω καθεξής. Συνεπώς η αποτίμηση κάθε δομής γίνεται μετά από T period , ο αλγόριθμος μας κάνει react μετά από κάθε περίοδο και όχι σε κάθε ένα μεμονωμένο timestamp.

Στην περίπτωση μας το fitness function γίνεται evaluate στο τέλος της περιόδου T , προχωράμε ανά M όπου και παίρνω κάθε φορά ανά M την εκάστοτε απόφαση. Συμβατικά θεωρούμε ότι η πρώτη μας απόφαση έχει τιμή 0 (Neutral position) δηλάδη ο πράκτορας(trader) δεν κάνει κάποια κίνηση. Ο NEAT θα εξελίσσει μια γενιά από νευρωνικα τα οποία έχουν διαφορετική συμπεριφορά το καθένα. Παραμετροποιούμε το δίκτυο μας ώστε να είναι fully connected από το Input στο μοναδικό Output. (T period : Σε πόσο χρονικό διάστημα αποτιμούμε το fitness). Μετά από κάθε time step υπολογίζω το return R(t) που μου λέει και πόσο καλά τα πάει ο αλγόριθμος μου. Στα run που κάναμε θέσαμε το αρχικό μας πληθυσμό (population) ίσο με 150 και τρέξαμε τον αλγόριθμο για 150 γενιές (generations). Σε κάθε γενιά (generation) ο κώδικας θα διάβασει όλο το dataset για T -περίοδο ανά βήμα M 150 φορές και

για κάθε γενιά θα προκύπτει ένα fitness , η προσδοκία μας θα είναι να μεγαλώσει η αναμενομένη τιμή του R(t) η όποια και αξιολογείται βάσει του sharpe ratio, άρα η επιβράβευση όταν ο NEAT κερδίζει δηλάδη Rt >0 είτε όταν χάνει δηλαδή Rt<0 γίνεται μέσω του sharpe ratio.

Μετά το πέρας όλων των γενεών , βασιζόμαστε στην καλύτερη ή καλύτερες εκδοχές των νευρωνικών που αναπτύχθηκαν κατά την εκπαίδευση. Από τα runs που πραγματοποιήσαμε παρατηρήσαμε ότι το fitness μεγαλώνει εντός του training. .Θα μπορούσαμε να εξαναγκάσουμε τον NEAT να περιορίζεται σε ένα συγκεκριμένο όγκο χρημάτων για ένα πιο ρεαλιστικό σενάριο σε κάθε χρονική στιγμή για κάθε return ωστόσο θεωρήσαμε ότι έχουμε ένα άπειρο portfolio προκειμένου να δούμε την πλήρη συμπεριφορά του αλγορίθμου. Παρατηρήσαμε έπειτα και από την εκτέλεση της εκπαίδευσης και του testing ότι καταλήγουμε σε μονοστρωματικό δίκτυο όπου το Sharpe ratio δείχνει ανοδικές τάσεις για περισσότερους χρόνους εκτέλεσης/γενιές.

4.4 Συμπεράσματα

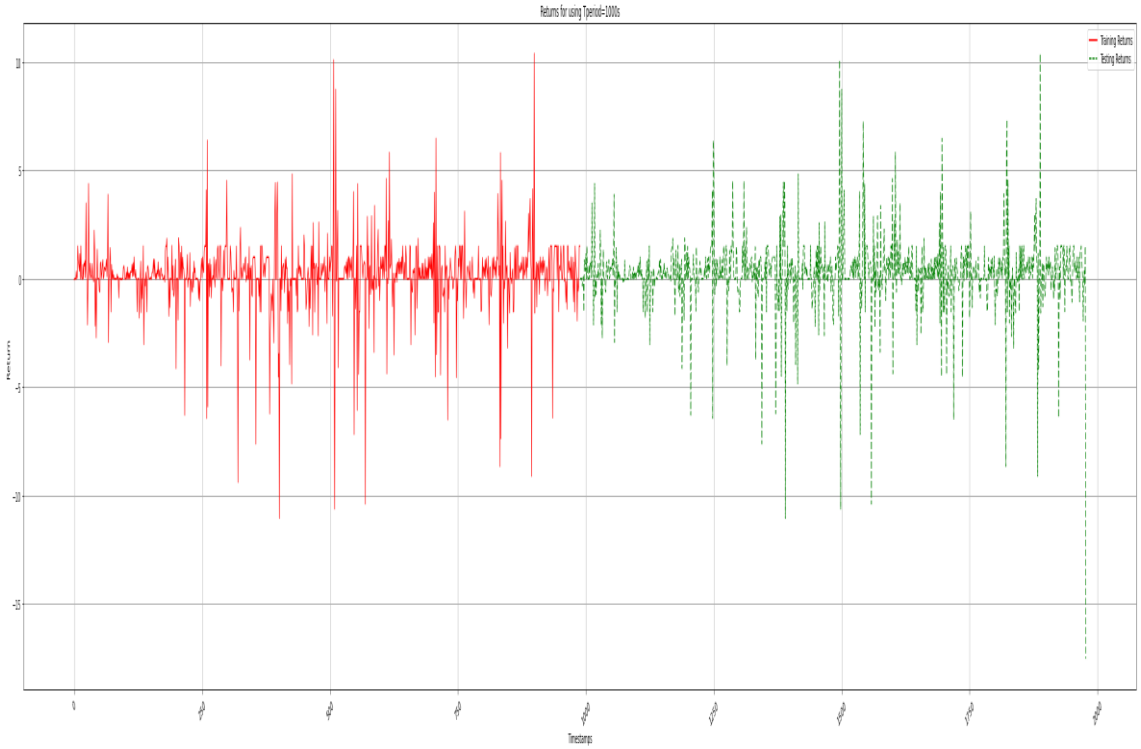
Στην συγκεκριμένη υλοποίηση θεωρήσαμε ότι έχουμε ένα άπειρο Portfolio προκειμένου να δούμε την πλήρη συμπεριφορά του NEAT. Παρατηρήσαμε έπειτα και από την ολοκλήρωση της εκπαίδευσης(training) και του testing ότι καταλήγουμε σε **μονοστρωματικό δίκτυο (one layer architecture)** όπου το sharpe ratio δείχνει ανοδικές τάσεις για μεγαλύτερο πλήθος γενεών αλλά και για μεγαλύτερο data set.

Σε μια μεταγενέστερη μελέτη θα μπορούσαμε απλά να εξαναγκάζουμε τον NEAT να παίζει με συγκεκριμένο πλήθος χρήματων και ουσιαστικά να υπάρχει ένα φυσικό Termination του παιχνιδιού) , δηλαδή κατά την εκπαίδευση να μπει σαν παράμετρος και το Portfolio οπότε

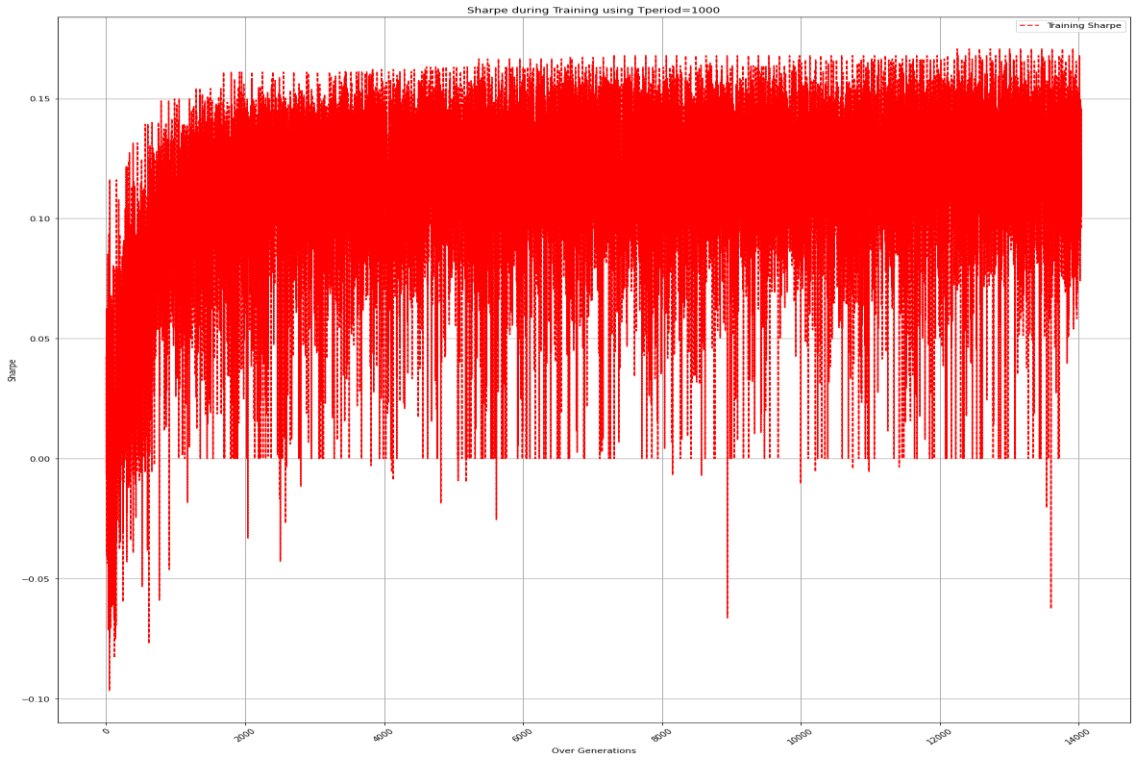
να εξαναγκάσουμε τον NEAT να μαθαίνει να παίρνει αποφάσεις βάσει των χρημάτων κάθε φορά που έχει στην κατοχή του.

Παρακάτω παρουσιάζονται μερικές από τις γραφικές απεικονίσεις που προέκυψαν από το μοντέλο που υλοποιήσαμε.

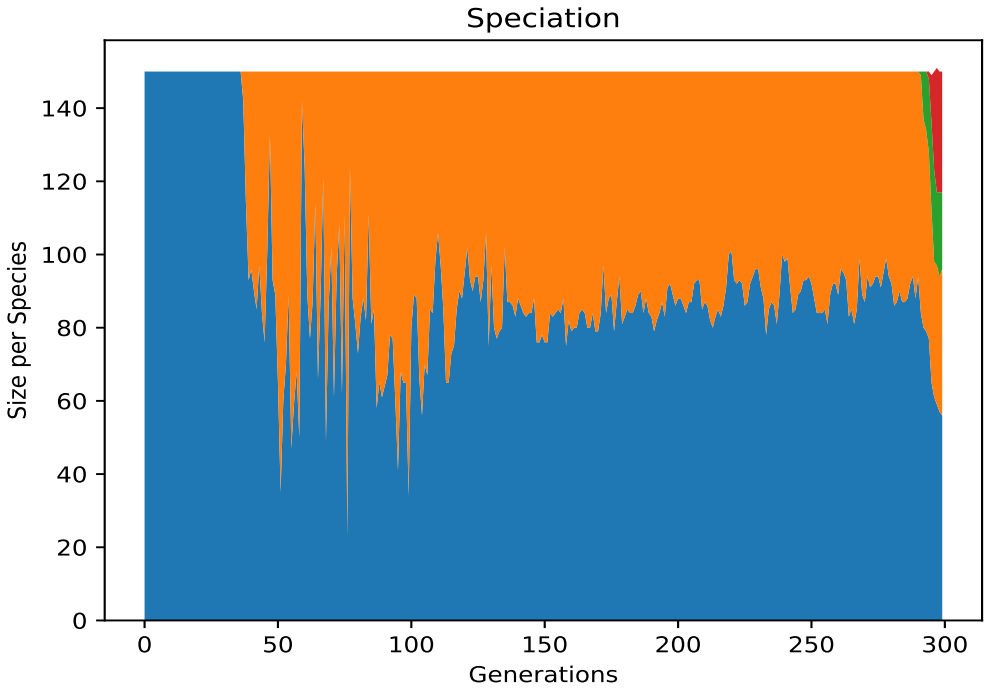
Διαγράμματα Tperiod 1000 με time window 10:



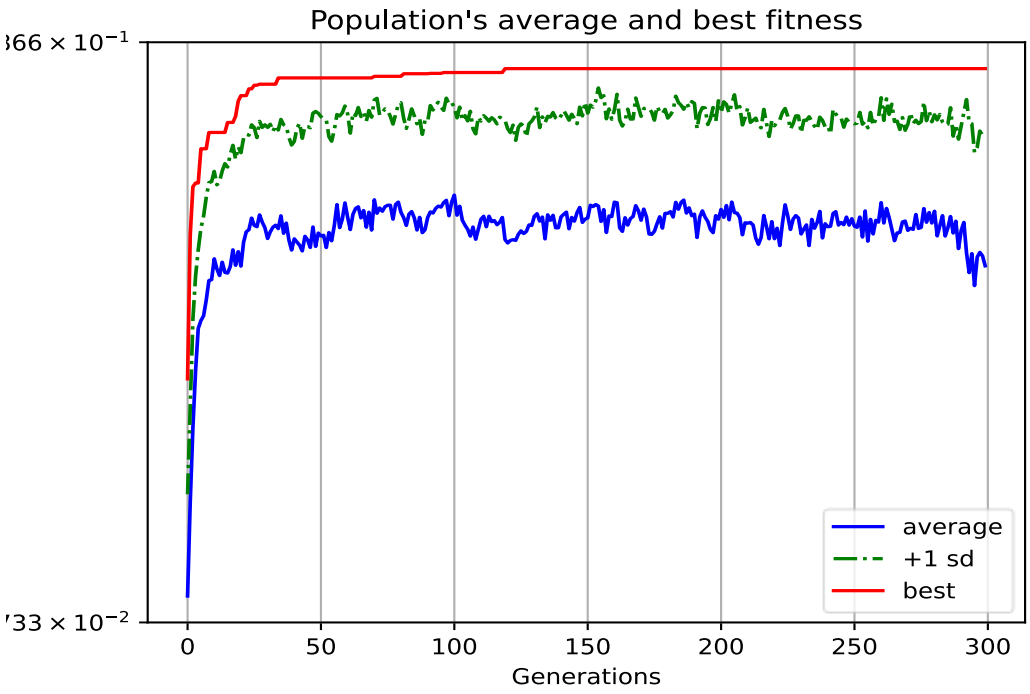
Εικόνα 19:Returns over Training Period/Testing Period



Εικόνα 20:Sharpe Ratio during Training



Εικόνα 21:Speciation during Training



Εικόνα 22:fitness over Training Period

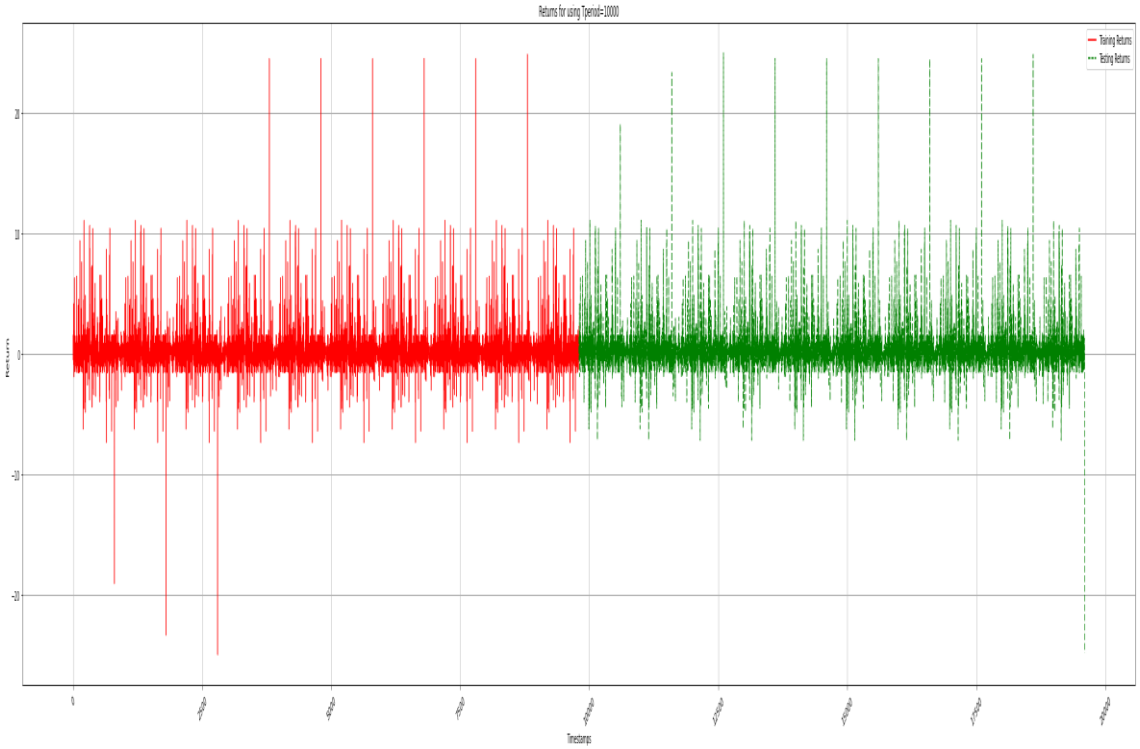
```
Best genome:
Key: 17526
Fitness: 0.1773921716716076
Nodes:
        0 DefaultNodeGene(key=0, bias=0.6453207123228475, response=1.0, activation=tanh, aggregation=sum)
Connections:
        DefaultConnectionGene(key=(-12, 0), weight=2.07957431327263, enabled=True)
        DefaultConnectionGene(key=(-10, 0), weight=7.768603309915276, enabled=True)
        DefaultConnectionGene(key=(-8, 0), weight=8.53488242580295, enabled=True)
        DefaultConnectionGene(key=(-7, 0), weight=7.510621012845153, enabled=True)
        DefaultConnectionGene(key=(-5, 0), weight=3.7613766524558288, enabled=True)
        DefaultConnectionGene(key=(-4, 0), weight=-0.011101575506193367, enabled=True)
        DefaultConnectionGene(key=(-3, 0), weight=-3.7554153212388877, enabled=True)
```

Εικόνα 23:Structure of Best Genome on Training Period

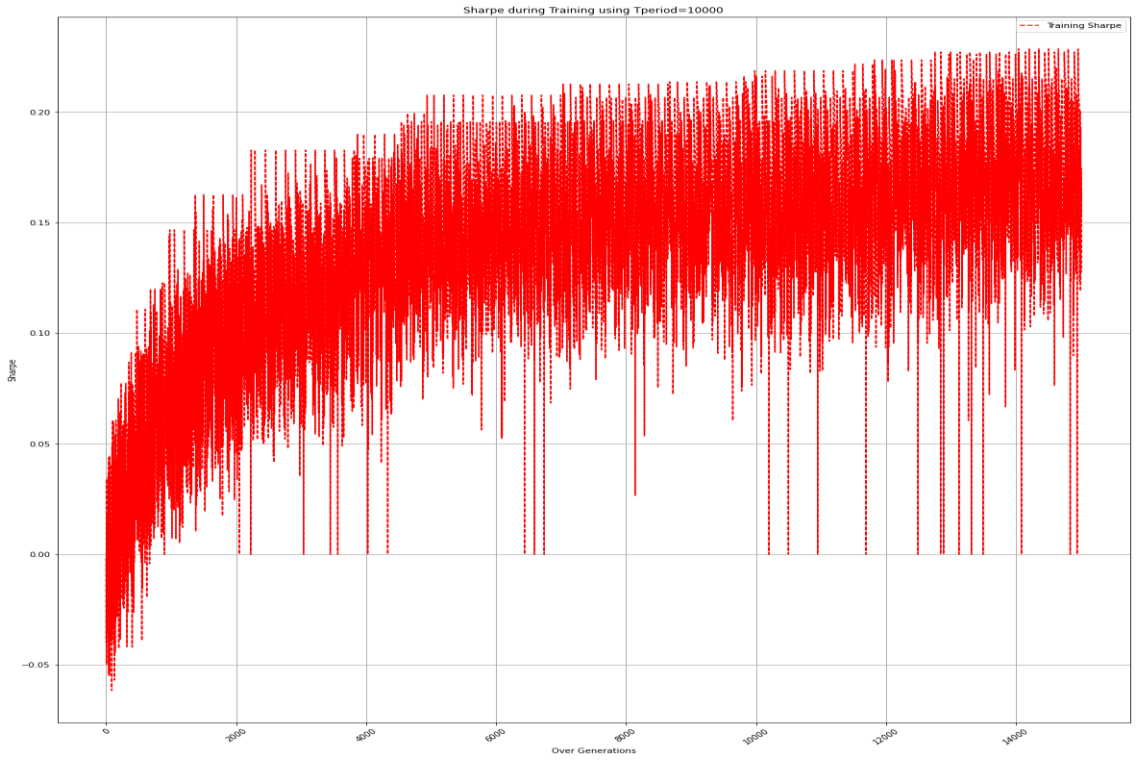
```
Best genome:
Key: 62
Fitness: 0.08709300735094654
Nodes:
        0 DefaultNodeGene(key=0, bias=0.9404237566544985, response=1.0, activation=tanh, aggregation=sum)
Connections:
        DefaultConnectionGene(key=(-12, 0), weight=0.7460169926828051, enabled=True)
        DefaultConnectionGene(key=(-11, 0), weight=-0.5282562794032284, enabled=True)
        DefaultConnectionGene(key=(-10, 0), weight=0.41640965501479044, enabled=True)
        DefaultConnectionGene(key=(-9, 0), weight=-0.5794978580880826, enabled=True)
        DefaultConnectionGene(key=(-8, 0), weight=0.04006944117766861, enabled=True)
        DefaultConnectionGene(key=(-7, 0), weight=1.9603280727713623, enabled=True)
        DefaultConnectionGene(key=(-6, 0), weight=0.35950805550493475, enabled=True)
        DefaultConnectionGene(key=(-5, 0), weight=1.543742036125031, enabled=True)
        DefaultConnectionGene(key=(-4, 0), weight=-0.21579027179197177, enabled=True)
        DefaultConnectionGene(key=(-3, 0), weight=0.8953312621331602, enabled=True)
        DefaultConnectionGene(key=(-2, 0), weight=1.0498507811397524, enabled=True)
        DefaultConnectionGene(key=(-1, 0), weight=-0.7071719871785809, enabled=True)
```

Εικόνα 24:Structure of Best Genome on Testing Period

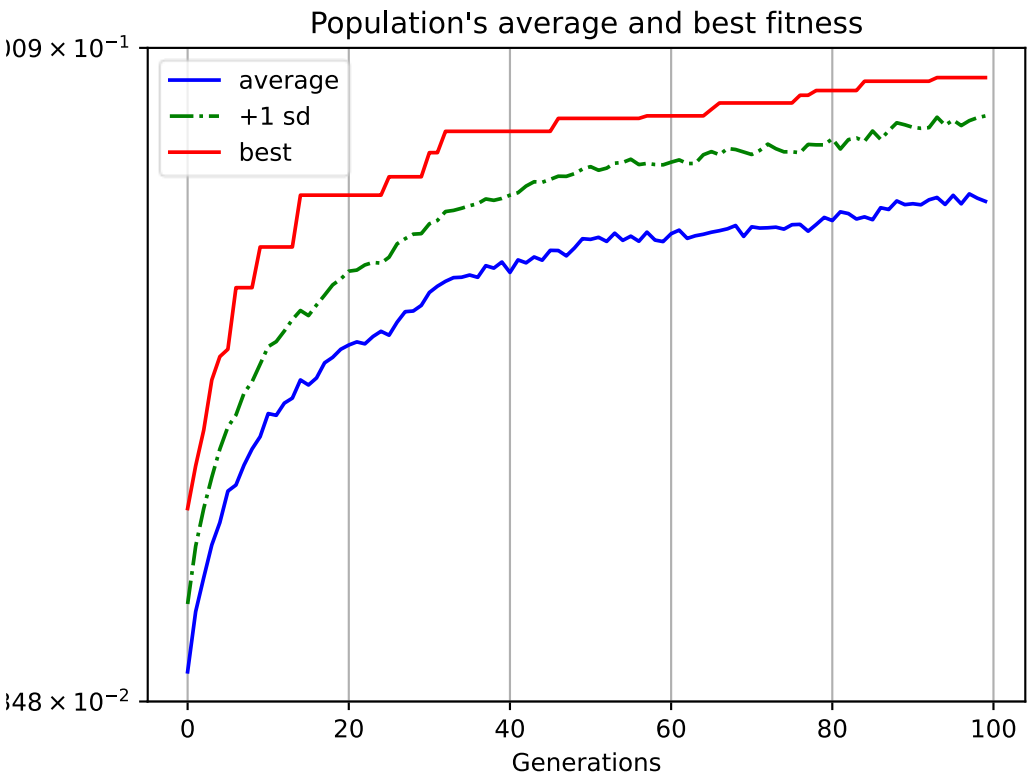
Διαγράμματα Tperiod 10000 με time window 200:



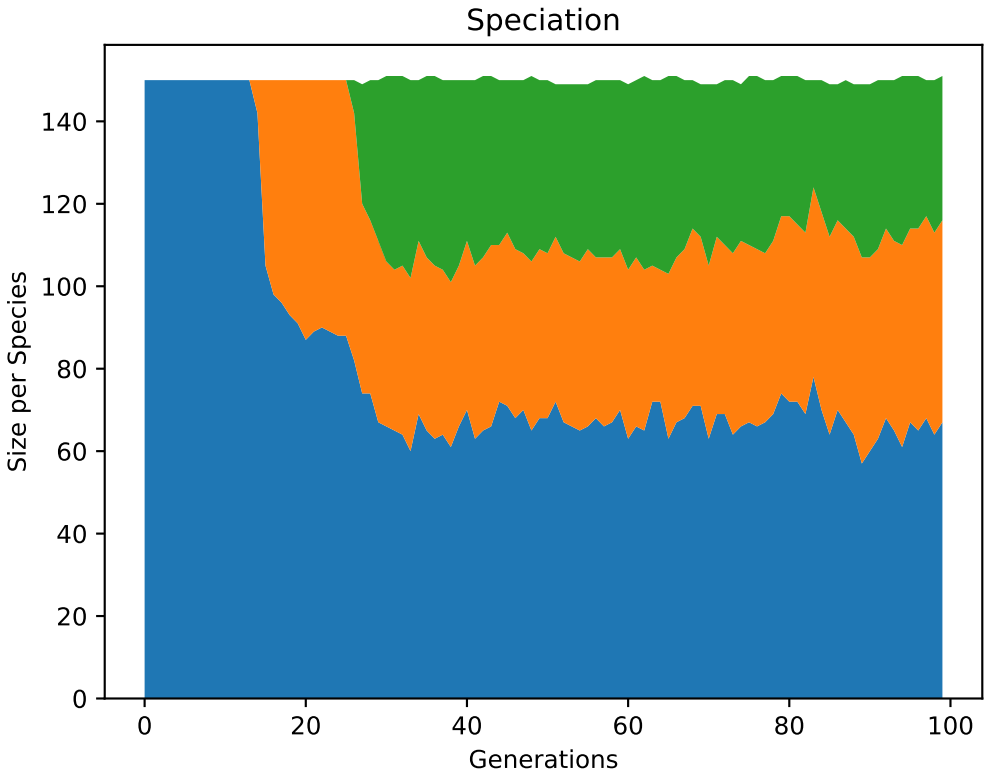
Εικόνα 25:Returns over Training Period/Testing Period



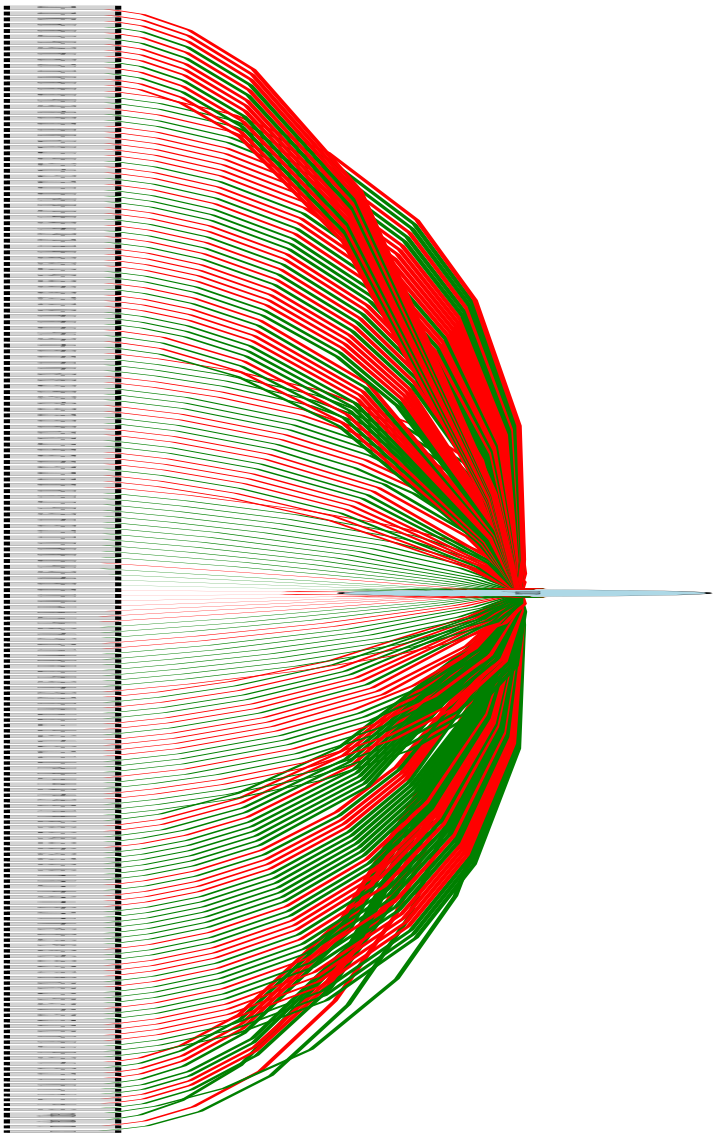
Εικόνα 26:Sharpe Ratio during Training



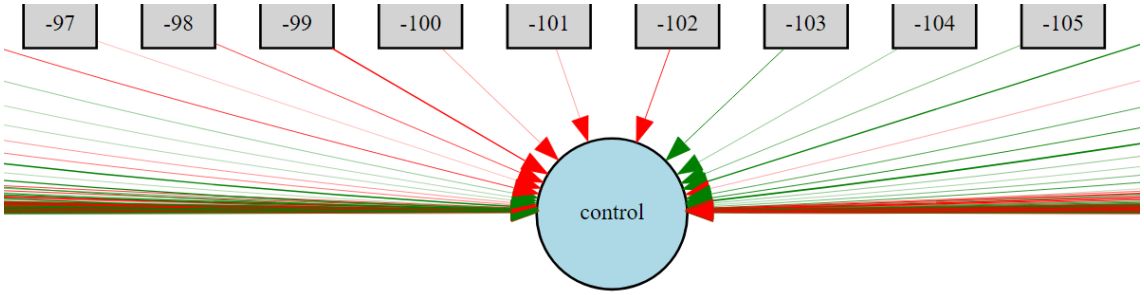
Εικόνα 27:Fitness over Training



Εικόνα 28:Speciation over Training period



Εικόνα 29:Winner neural Network for T=10.000 and time window 200



Εικόνα 30:Winner neural Network for T=10.000 and time window 200

Best genome:

Key: **9302**

Fitn ess: **0.22851595096792313**

Nodes:

0 DefaultNodeGene(key=**0**, bias=-**2.8456249454603215**, response=**1.0**, activation=tanh, aggregation=sum)

668 DefaultNodeGene(key=**668**, bias=-**0.41615046318363813**, response=**1.0**, activation=tanh, aggregation=sum)

1789 DefaultNodeGene(key=**1789**, bias=-**0.7752243109397667**, response=**1.0**, activation=tanh, aggregation=sum)

Connections:

DefaultConnectionGene(key=(-**202**, **0**), weight=-**2.8183801633766437**, enabled=False)

DefaultConnectionGene(key=(-**201**, **0**), weight=**0.4372126462275907**, enabled=True)

DefaultConnectionGene(key=(-**200**, **0**), weight=**2.364276200007936**, enabled=True)

DefaultConnectionGene(key=(-**199**, **0**), weight=-**0.9687857278639302**, enabled=True)

DefaultConnectionGene(key=(-**198**, **0**), weight=**1.498046221828562**, enabled=True)

DefaultConnectionGene(key=(-**197**, **0**), weight=**0.381144905729107**, enabled=True)

DefaultConnectionGene(key=(-**196**, **0**), weight=**0.7475559222591479**, enabled=True)

DefaultConnectionGene(key=(-**195**, **0**), weight=**1.8532148257746364**, enabled=True)

DefaultConnectionGene(key=(-**194**, **0**), weight=-**1.0585653920384872**, enabled=False)

DefaultConnectionGene(key=(-**193**, **0**), weight=**0.2822033006275268**, enabled=True)

DefaultConnectionGene(key=(-**193**, **668**), weight=**4.003057217981979**, enabled=True)

DefaultConnectionGene(key=(-**192**, **0**), weight=-**0.09611395637445863**, enabled=True)

DefaultConnectionGene(key=(-**191**, **0**), weight=**3.491406679229016**, enabled=True)

DefaultConnectionGene(key=(-**190**, **0**), weight=**0.5091821262180842**, enabled=True)

DefaultConnectionGene(key=(-**189**, **0**), weight=-**3.9151109370525843**, enabled=True)

DefaultConnectionGene(key=(-**188**, **0**), weight=-**1.1226099408852837**, enabled=True)

DefaultConnectionGene(key=(-**187**, **0**), weight=-**1.206653776374363**, enabled=False)

DefaultConnectionGene(key=(-**186**, **0**), weight=**1.1886044089149868**, enabled=True)

DefaultConnectionGene(key=(-**185**, **0**), weight=-**0.07037377831627956**, enabled=True)

DefaultConnectionGene(key=(-**184**, **0**), weight=-**3.0181679419339016**, enabled=True)

DefaultConnectionGene(key=(-**183**, **0**), weight=**0.46119617621602416**, enabled=False)

DefaultConnectionGene(key=(-**182**, **0**), weight=**1.0710366218332334**, enabled=True)

DefaultConnectionGene(key=(-**181**, **0**), weight=**0.32993787801965035**, enabled=True)

DefaultConnectionGene(key=(-**180**, **0**), weight=**3.6617912609663086**, enabled=True)

DefaultConnectionGene(key=(-**179**, **0**), weight=**0.9903195947467686**, enabled=True)

DefaultConnectionGene(key=(-**178**, **0**), weight=**2.1719387088779176**, enabled=True)

DefaultConnectionGene(key=(-**177**, **0**), weight=**1.1228542657137484**, enabled=True)

DefaultConnectionGene(key=(-**176**, **0**), weight=**0.6962371963886137**, enabled=False)

DefaultConnectionGene(key=(-**175**, **0**), weight=-**0.7156837571347832**, enabled=False)

DefaultConnectionGene(key=(-**174**, **0**), weight=**0.05837411735606823**, enabled=True)

DefaultConnectionGene(key=(-**173**, **0**), weight=**0.7211862835389699**, enabled=True)

DefaultConnectionGene(key=(-**171**, **0**), weight=-**2.2398476488601955**, enabled=True)

DefaultConnectionGene(key=(-**170**, **0**), weight=-**1.8378580230298844**, enabled=True)

DefaultConnectionGene(key=(-**169**, **0**), weight=**0.5737195667909524**, enabled=True)

DefaultConnectionGene(key=(-**168**, **0**), weight=**3.630331849288183**, enabled=True)

DefaultConnectionGene(key=(-**166**, **0**), weight=-**4.236664632961027**, enabled=True)

DefaultConnectionGene(key=(-**165**, **0**), weight=**0.620722783126362**, enabled=True)

DefaultConnectionGene(key=(-**164**, **0**), weight=**0.37782818904663584**, enabled=True)

DefaultConnectionGene(key=(-**163**, **0**), weight=**1.5451636755112033**, enabled=True)

DefaultConnectionGene(key=(-**162**, **0**), weight=**2.2937063651035743**, enabled=True)

DefaultConnectionGene(key=(-**161**, **0**), weight=-**1.2418947710623531**, enabled=True)

DefaultConnectionGene(key=(-**160**, **0**), weight=**2.068074216308118**, enabled=True)

DefaultConnectionGene(key=(-**159**, **0**), weight=-**0.978619117970355**, enabled=True)

DefaultConnectionGene(key=(-**158**, **0**), weight=**2.3290858852697878**, enabled=False)

DefaultConnectionGene(key=(-**157**, **0**), weight=-**0.5355864430280451**, enabled=True)

DefaultConnectionGene(key=(-**156**, **0**), weight=-**1.163920704551914**, enabled=True)

DefaultConnectionGene(key=(-**155**, **0**), weight=-**0.985013261713393**, enabled=True)

DefaultConnectionGene(key=(-**154**, **0**), weight=**0.9056408676246503**, enabled=True)

DefaultConnectionGene(key=(-**153**, **0**), weight=**4.416966099242115**, enabled=True)

DefaultConnectionGene(key=(-**152**, **0**), weight=-**0.7004269459223922**, enabled=True)

DefaultConnectionGene(key=(-**151**, **0**), weight=**1.239404821796537**, enabled=True)

DefaultConnectionGene(key=(-**150**, **0**), weight=-**0.24147374879711214**, enabled=False)

DefaultConnectionGene(key=(-**149**, **0**), weight=**1.124193501714438**, enabled=True)

DefaultConnectionGene(key=(-**148**, **0**), weight=**1.2426595089573271**, enabled=True)

DefaultConnectionGene(key=(-**147**, **0**), weight=-**6.453222640061434**, enabled=True)

DefaultConnectionGene(key=(-**146**, **0**), weight=-**1.6908695641615727**, enabled=True)

DefaultConnectionGene(key=(-**144**, **0**), weight=**0.28141342515577694**, enabled=True)

DefaultConnectionGene(key=(-**143**, **0**), weight=**0.14075359542274815**, enabled=True)

DefaultConnectionGene(key=(-**142**, **0**), weight=-**1.2992066614743412**, enabled=True)

DefaultConnectionGene(key=(-**141**, **0**), weight=**4.644889469389257**, enabled=True)

DefaultConnectionGene(key=(-**140**, **0**), weight=-**0.029331560421688496**, enabled=True)

DefaultConnectionGene(key=(-**139**, **0**), weight=**4.254822195000223**, enabled=True)

DefaultConnectionGene(key=(-**138**, **0**), weight=-**1.6978907658622775**, enabled=False)

DefaultConnectionGene(key=(-**137**, **0**), weight=-**0.05201319883338595**, enabled=True)

DefaultConnectionGene(key=(-**136**, **0**), weight=**0.9507945449799975**, enabled=True)

DefaultConnectionGene(key=(-**135**, **0**), weight=**0.610448865138174**, enabled=True)

DefaultConnectionGene(key=(-**134**, **0**), weight=**0.5409488874639272**, enabled=False)

DefaultConnectionGene(key=(-**133**, **0**), weight=-**1.1972239998527563**, enabled=True)

DefaultConnectionGene(key=(-**132**, **0**), weight=-**1.4688522522362655**, enabled=False)

DefaultConnectionGene(key=(-**131**, **0**), weight=**0.466102952095293**, enabled=True)

DefaultConnectionGene(key=(-**130**, **0**), weight=-**0.22686811446920646**, enabled=True)

DefaultConnectionGene(key=(-**129**, **0**), weight=**0.5941535366509959**, enabled=True)

DefaultConnectionGene(key=(-**128**, **0**), weight=-**1.3482890267326817**, enabled=True)

DefaultConnectionGene(key=(-**127**, **0**), weight=**0.7770047830044735**, enabled=True)

DefaultConnectionGene(key=(-**126**, **0**), weight=-**1.0205955266994107**, enabled=True)

DefaultConnectionGene(key=(-**126**, **1789**), weight=**0.7199925426515255**, enabled=True)

DefaultConnectionGene(key=(-**125**, **0**), weight=**1.1039065656945863**, enabled=True)

DefaultConnectionGene(key=(-**124**, **0**), weight=**2.9599233896652812**, enabled=True)

DefaultConnectionGene(key=(-**123**, **0**), weight=-**0.9599090159556753**, enabled=True)

DefaultConnectionGene(key=(-**121**, **0**), weight=**2.430141751954168**, enabled=True)

DefaultConnectionGene(key=(-**120**, **0**), weight=**0.4437016064663344**, enabled=True)

DefaultConnectionGene(key=(-**119**, **0**), weight=**0.9739757685774326**, enabled=True)

DefaultConnectionGene(key=(-**118**, **0**), weight=**0.01692917595312693**, enabled=True)

DefaultConnectionGene(key=(-**117**, **0**), weight=-**0.6117976977390012**, enabled=True)

DefaultConnectionGene(key=(-**117**, **1789**), weight=**1.1854070222644544**, enabled=True)

DefaultConnectionGene(key=(-**116**, **0**), weight=-**1.3355081306061976**, enabled=True)

DefaultConnectionGene(key=(-**115**, **0**), weight=**0.6569071709575676**, enabled=True)

DefaultConnectionGene(key=(-**114**, **0**), weight=**0.22298633672813922**, enabled=True)

DefaultConnectionGene(key=(-**113**, **0**), weight=-**2.5739808665508566**, enabled=True)

DefaultConnectionGene(key=(-**112**, **0**), weight=**0.1981171699450872**, enabled=True)

DefaultConnectionGene(key=(-**111**, **0**), weight=**0.47189354784509097**, enabled=True)

DefaultConnectionGene(key=(-**110**, **0**), weight=-**0.6675556149338688**, enabled=True)

DefaultConnectionGene(key=(-**109**, **0**), weight=-**2.1252927258985124**, enabled=True)

DefaultConnectionGene(key=(-**108**, **0**), weight=**1.3020135739735308**, enabled=False)

DefaultConnectionGene(key=(-**107**, **0**), weight=**2.3018706820905472**, enabled=True)

DefaultConnectionGene(key=(-**105**, **0**), weight=-**0.5762064147478113**, enabled=False)

DefaultConnectionGene(key=(-**104**, **0**), weight=-**0.014170507432976626**, enabled=True)

DefaultConnectionGene(key=(-**103**, **0**), weight=-**0.2660889045293542**, enabled=True)

DefaultConnectionGene(key=(-**102**, **0**), weight=**2.9509996179687303**, enabled=True)

DefaultConnectionGene(key=(-**101**, **0**), weight=-**0.4883181870193275**, enabled=False)

DefaultConnectionGene(key=(-**100**, **0**), weight=**2.377211735986182**, enabled=True)

DefaultConnectionGene(key=(-**99**, **0**), weight=-**2.8665559030550942**, enabled=True)

DefaultConnectionGene(key=(-**98**, **0**), weight=**0.04466311138834355**, enabled=True)

DefaultConnectionGene(key=(-**97**, **0**), weight=**0.3302028067927481**, enabled=True)

DefaultConnectionGene(key=(-**96**, **0**), weight=-**1.4852827151603214**, enabled=True)

DefaultConnectionGene(key=(-**95**, **0**), weight=-**0.6704239780724246**, enabled=True)

DefaultConnectionGene(key=(-**94**, **0**), weight=-**1.0122847777738977**, enabled=True)

DefaultConnectionGene(key=(-**93**, **0**), weight=**2.8362092597652997**, enabled=True)

DefaultConnectionGene(key=(-**92**, **0**), weight=-**1.9969397394292594**, enabled=True)

DefaultConnectionGene(key=(-**91**, **0**), weight=**1.4500962634971353**, enabled=True)

DefaultConnectionGene(key=(-**90**, **0**), weight=**0.14443088898952902**, enabled=True)

DefaultConnectionGene(key=(-**90**, **668**), weight=-**1.7854996269220103**, enabled=True)

DefaultConnectionGene(key=(-**89**, **0**), weight=**2.685841821883416**, enabled=False)

DefaultConnectionGene(key=(-**88**, **0**), weight=-**0.6680280837638156**, enabled=True)

DefaultConnectionGene(key=(-**87**, **0**), weight=**1.0609350096028953**, enabled=True)

DefaultConnectionGene(key=(-**86**, **0**), weight=**0.3092711725361897**, enabled=True)

DefaultConnectionGene(key=(-**85**, **0**), weight=-**0.6211857677181055**, enabled=True)

DefaultConnectionGene(key=(-**84**, **0**), weight=**2.2460683507139656**, enabled=True)

DefaultConnectionGene(key=(-**83**, **0**), weight=-**1.7625610284240358**, enabled=True)

DefaultConnectionGene(key=(-**82**, **0**), weight=**0.7276871241661483**, enabled=False)

DefaultConnectionGene(key=(-**81**, **0**), weight=**2.8396198831359567**, enabled=True)

DefaultConnectionGene(key=(-**80**, **0**), weight=**2.1204296670200637**, enabled=True)

DefaultConnectionGene(key=(-**78**, **0**), weight=-**1.44671352008542**, enabled=True)

DefaultConnectionGene(key=(-**77**, **0**), weight=**2.0646730823489587**, enabled=True)

DefaultConnectionGene(key=(-**76**, **0**), weight=-**2.03547331599191**, enabled=True)

DefaultConnectionGene(key=(-**75**, **0**), weight=**0.5384559333277844**, enabled=True)

DefaultConnectionGene(key=(-**74**, **0**), weight=**0.36993539169625833**, enabled=True)

DefaultConnectionGene(key=(-**73**, **0**), weight=**2.3408526431071017**, enabled=True)

DefaultConnectionGene(key=(-**72**, **0**), weight=**2.0235528656542656**, enabled=True)

DefaultConnectionGene(key=(-**71**, **0**), weight=-**3.574180775908805**, enabled=True)

DefaultConnectionGene(key=(-**70**, **0**), weight=**1.010631014069867**, enabled=True)

DefaultConnectionGene(key=(-**69**, **0**), weight=-**0.4419829846990262**, enabled=True)

DefaultConnectionGene(key=(-**68**, **0**), weight=**1.0233725412387156**, enabled=True)

DefaultConnectionGene(key=(-**67**, **0**), weight=**1.6665693764994065**, enabled=True)

DefaultConnectionGene(key=(-**66**, **0**), weight=**0.4594987327470717**, enabled=True)

DefaultConnectionGene(key=(-**65**, **0**), weight=**1.1345568527890704**, enabled=True)

DefaultConnectionGene(key=(-**65**, **668**), weight=-**1.7103392522645249**, enabled=True)

DefaultConnectionGene(key=(-**64**, **0**), weight=**0.5457545745933953**, enabled=True)

DefaultConnectionGene(key=(-**63**, **0**), weight=**1.974552026049754**, enabled=True)

DefaultConnectionGene(key=(-**62**, **0**), weight=-**0.5074017042751705**, enabled=True)

DefaultConnectionGene(key=(-**61**, **0**), weight=**2.439640284665462**, enabled=True)

DefaultConnectionGene(key=(-**60**, **0**), weight=**2.271789177451643**, enabled=False)

DefaultConnectionGene(key=(-**59**, **0**), weight=-**0.7193030308970572**, enabled=True)

DefaultConnectionGene(key=(-**58**, **0**), weight=**2.9156663430233385**, enabled=True)

DefaultConnectionGene(key=(-**57**, **0**), weight=-**1.5379803629274276**, enabled=True)

DefaultConnectionGene(key=(-**56**, **0**), weight=**0.3957258845861789**, enabled=True)

DefaultConnectionGene(key=(-**55**, **0**), weight=**1.1625677247235786**, enabled=True)

DefaultConnectionGene(key=(-**54**, **0**), weight=-**3.1498765848303596**, enabled=True)

DefaultConnectionGene(key=(-**53**, **0**), weight=**1.9900184129321437**, enabled=True)

DefaultConnectionGene(key=(-**52**, **0**), weight=**0.8509556820009624**, enabled=True)

DefaultConnectionGene(key=(-**51**, **0**), weight=**0.7906170896908888**, enabled=False)

DefaultConnectionGene(key=(-**50**, **0**), weight=**2.450418785664569**, enabled=True)

DefaultConnectionGene(key=(-**49**, **0**), weight=**2.1878759337189537**, enabled=True)

DefaultConnectionGene(key=(-**48**, **0**), weight=**8.013991948865009**, enabled=True)

DefaultConnectionGene(key=(-**47**, **0**), weight=-**0.3872516396255078**, enabled=True)

DefaultConnectionGene(key=(-**46**, **0**), weight=**3.299695343476172**, enabled=True)

DefaultConnectionGene(key=(-**45**, **0**), weight=**2.0536069534395445**, enabled=True)

DefaultConnectionGene(key=(-**44**, **0**), weight=**0.8682491900415573**, enabled=True)

DefaultConnectionGene(key=(-**43**, **0**), weight=**1.5469445672040611**, enabled=False)

DefaultConnectionGene(key=(-**42**, **0**), weight=-**1.0971461863834218**, enabled=True)

DefaultConnectionGene(key=(-**41**, **0**), weight=**0.8503721926785197**, enabled=True)

DefaultConnectionGene(key=(-**40**, **0**), weight=-**0.16471497790227874**, enabled=True)

DefaultConnectionGene(key=(-**39**, **0**), weight=**0.45103800976217334**, enabled=True)

DefaultConnectionGene(key=(-**38**, **0**), weight=-**0.030023784445974533**, enabled=True)

DefaultConnectionGene(key=(-**37**, **0**), weight=**1.5240356824318053**, enabled=True)

DefaultConnectionGene(key=(-**36**, **0**), weight=**1.183591373253214**, enabled=True)

DefaultConnectionGene(key=(-**35**, **0**), weight=-**1.9373468022465021**, enabled=True)

DefaultConnectionGene(key=(-**34**, **0**), weight=-**2.193491676028111**, enabled=True)

DefaultConnectionGene(key=(-**33**, **0**), weight=-**0.0966476328142784**, enabled=True)

DefaultConnectionGene(key=(-**32**, **0**), weight=-**1.0056851189992984**, enabled=False)

DefaultConnectionGene(key=(-**31**, **0**), weight=-**1.7491641595543779**, enabled=False)

DefaultConnectionGene(key=(-**30**, **0**), weight=-**2.125364445818521**, enabled=True)

DefaultConnectionGene(key=(-**29**, **0**), weight=-**0.836875067832068**, enabled=False)

DefaultConnectionGene(key=(-**28**, **0**), weight=-**0.8346434897953428**, enabled=True)

DefaultConnectionGene(key=(-**26**, **0**), weight=-**1.1887754924190936**, enabled=True)

DefaultConnectionGene(key=(-**25**, **0**), weight=-**4.129332732005795**, enabled=True)

DefaultConnectionGene(key=(-**24**, **0**), weight=**0.1461612723007733**, enabled=True)

DefaultConnectionGene(key=(-**23**, **0**), weight=**0.23277245597545393**, enabled=True)

DefaultConnectionGene(key=(-**22**, **0**), weight=**0.4357154812216515**, enabled=True)

DefaultConnectionGene(key=(-**21**, **0**), weight=-**4.708569363112581**, enabled=True)

DefaultConnectionGene(key=(-**20**, **0**), weight=**0.6746188491293945**, enabled=True)

DefaultConnectionGene(key=(-**19**, **0**), weight=-**2.5722291947277585**, enabled=True)

DefaultConnectionGene(key=(-**18**, **0**), weight=-**4.911831976103354**, enabled=True)

DefaultConnectionGene(key=(-**17**, **0**), weight=**2.300338008214563**, enabled=True)

DefaultConnectionGene(key=(-**16**, **0**), weight=-**0.4508061101873491**, enabled=True)

DefaultConnectionGene(key=(-**15**, **0**), weight=**2.525682468668415**, enabled=True)

DefaultConnectionGene(key=(-**14**, **0**), weight=**1.4352029750071607**, enabled=True)

DefaultConnectionGene(key=(-**13**, **0**), weight=**0.2048687623549486**, enabled=True)

DefaultConnectionGene(key=(-**12**, **0**), weight=-**0.06086410385025953**, enabled=True)

DefaultConnectionGene(key=(-**11**, **0**), weight=**0.1775840846922427**, enabled=True)

DefaultConnectionGene(key=(-**10**, **0**), weight=**2.558564848781472**, enabled=True)

DefaultConnectionGene(key=(-**9**, **0**), weight=-**0.026537770274122074**, enabled=True)

DefaultConnectionGene(key=(-**8**, **0**), weight=-**0.15344426277569**, enabled=True)

DefaultConnectionGene(key=(-**7**, **0**), weight=**3.8513779386356326**, enabled=True)

DefaultConnectionGene(key=(-**6**, **0**), weight=**0.7607128719349814**, enabled=True)

DefaultConnectionGene(key=(-**5**, **0**), weight=**1.4393792087645398**, enabled=True)

DefaultConnectionGene(key=(-**4**, **0**), weight=-**2.3666838001232566**, enabled=True)

DefaultConnectionGene(key=(-**3**, **0**), weight=-**2.8240081937047896**, enabled=True)

DefaultConnectionGene(key=(-**1**, **0**), weight=-**0.5934113680272534**, enabled=True)

DefaultConnectionGene(key=(**668**, **0**), weight=-**0.14998428654530757**, enabled=True)

DefaultConnectionGene(key=(**1789**, **0**), weight=**0.9334999092503151**, enabled=True)

4.5 Αντικειμενικές δυσκολίες υλοποίησης

Αντικειμενικές δυσκολίες στην υλοποίηση του συγκεκριμένου προβλήματος:

1. Data set preparation (Ο χρόνος προετοιμασίας του input dataset γίνεται σε διαφορετικό χρόνο από αυτόν του χρόνου εκτέλεσης του EA)
2. Έλλειψη από compute resources (Όσον αφορά το συγκεκριμένο πρόβλημα , επιχειρήσαμε να κάνουμε χρήση του Colaboratory). Εν συντομία το «Colab», είναι προϊόν της Google Research, το οποίο επιτρέπει σε οποιονδήποτε να γράψει και να εκτελέσει αυθαίρετο κώδικα python μέσα από το browser(cloud based) και είναι ιδιαίτερα κατάλληλο για μηχανική μάθηση, ανάλυση δεδομένων και εκπαίδευση. Το Colab εμπεριέχει την υπηρεσία Jupyter, ενώ παρέχει πρόσβαση σε υπολογιστικούς πόρους (VM), συμπεριλαμβανομένων των GPU με πληρωμή.
3. Huge Data Processing για μεγαλύτερα time series. Παρόλο που το Colab δίνει την δυνατότητα scaling των resources αντιμετωπίζαμε πολλά time out errors μετά από κάποιες ώρες λειτουργίας του προγράμματος μας, ακόμα και αν τοποθετήσαμε save points η δυνατότητα του να τρέξει ο συγκεκριμένος EA για μεγάλο διάστημα χρονικά υστερημένων χρονικών στιγμών ήταν αδύνατο χωρίς subscription. Το συγκεκριμένο issue αποτελεί μέρος μελλοντικής διερεύνησης καθότι με περισσότερα compute resources ο συγκεκριμένος EA δείχνει να έχει την τάση όπως προαναφέραμε να αυξάνει το sharpe ratio άρα τα αποτελέσματα δείχνουν να είναι θετικά.

4. Parallel Execution: Καταφέραμε να τρέξουμε παράλληλα την αλγοριθμική διαδικασία του NEAT όσον αφορά του να τρέχουν πολλά generations παράλληλα ωστόσο ενδέχεται σε μια μεταγενέστερη μελέτη να γίνει optimization σε parallel αντί για sequential ο κώδικας ανάγνωσης του input dataset.

ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] J Moody,M Saffell (2001). Neural Networks: Learning to Trade Via Direct Reinforcement

[2] Afan Hasan, O. K. (2020). Modeling Traders' Behavior with Deep Learning and Machine Learning Methods: Evidence from BIST 100 Index.

[3] Park, H., Sim, M., & Choi, D. (2019). An intelligent financial portfolio trading strategy.

[4] Stanley, Kenneth O. Massachusetts Institute of Technology. (2002). Evolving neural networks through augmenting topologies.

[5] Simon Haykin (2005). Neural Networks: A Comprehensive Foundation 2nd edition, Prentice Hall.

[6] Stanley, Kenneth O. Massachusetts Institute of Technology (2002). Evolving neural networks through augmenting topologies.

[7] Stanley, Kenneth O., Bryant, B.D. and Miikkulainen, R., 2005. Real-time neuroevolution in the NERO video game. IEEE transactions on evolutionary computation.

[8] Kenneth O. Stanley and Risto Miikkulainen. Efficient Evolution of Neural Network Topologies.

[9] Gabriel Molina. Stock Trading with Recurrent Reinforcement Learning (RRL) CS229 Application Project, SUID 5055783.