ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ – ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

ΠΜΣ «Προηγμένα Συστήματα Πληροφορικής – Ανάπτυξη Λογισμικού και Τεχνητής Νοημοσύνης»

Μεταπτυχιακή Διατριβή

Τίτλος Διατριβής	**Εξομοίωση και Σύγκριση Αλγορίθμων Ενισχυτικής Μάθησης** **Simulation and Comparison of Reinforcement Learning Algorithms**
Ονοματεπώνυμο Φοιτητή	**Κωνσταντίνος Σπυρόπουλος**
Πατρώνυμο	**Μιλτιάδης**
Αριθμός Μητρώου	**ΜΠΣΠ18023**
Επιβλέπων	**Διονύσιος Σωτηρόπουλος, Επίκουρος Καθηγητής**

Ημ. Παράδοσης: Σεπτέμβριος 2022

Τριμελής Εξεταστική Επιτροπή

Δ. Σωτηρόπουλος	**Γ. Τσιχριντζής**	**Ε. Σακκόπουλος**
Επίκουρος Καθηγητής	**Καθηγητής**	**Αναπληρωτής Καθηγητής**

Περιεχόμενα

Λίστα Εικόνων

Λίστα Πινάκων

Περίληψη

Η Τεχνητή Νοημοσύνη στη σύγχρονη εποχή αποτελεί ένα σημαντικό εργαλείο για την υλοποίηση καινοτόμων ιδεών και την ανάπτυξη πολλών τεχνολογικών εφαρμογών. Ειδικότερα, το κομμάτι της μηχανικής μάθησης προσφέρει μια πληθώρα τεχνικών και αλγορίθμων για την επίλυση προβλημάτων που αντιμετωπίζονται δύσκολα κάνοντας χρήση του κλασσικού προγραμματισμού. Αντίστοιχο πρόβλημα παρουσιάζεται και στην παρούσα εργασία, η οποία αποτελεί μια θεωρητική και πρακτική προσέγγιση στον τομέα της μηχανικής μάθησης και στην εύρεση κατάλληλων τεχνικών για την επίλυσή σύνθετων προβλημάτων.

Η δημιουργία ευφυών τεχνητών πρακτόρων (agents) για την λύση πολύπλοκων προβλημάτων που σχετίζονται με τον άνθρωπο αποτελεί μια μεγάλη πρόκληση για την κοινότητα της τεχνητής νοημοσύνης. Ιδιαίτερα σημαντική είναι η κατανόηση του δυναμικού περιβάλλοντος στο οποίο δρουν και η αλληλεπίδραση τους με αυτό όπως και ο άνθρωπος στον φυσικό κόσμο. Ο τομέας που ειδικεύεται στην δημιουργία τέτοιων agents ονομάζεται Reinforcement Learning. Με τον όρο Reinforcement Learning, αναφερόμαστε στις μεθόδους μέσω των οποίων ένα σύστημα αλγορίθμων ‘μαθαίνει’ να αλληλεπιδρά μέσα σε ένα περιβάλλον δομημένο γύρω από κάποιους ορισμένους κανόνες μετά από δοκιμές και σφάλματα. Η εκμάθηση αυτή γίνεται μέσω της εξερεύνησης του περιβάλλοντος μέσω των πρακτόρων, των ενεργειών και των αντίστοιχων επιβραβεύσεων που δίνονται από αυτό με σκοπό να επιτευχθεί ένας στόχος με την βέλτιστη προσπάθεια.

Η έρευνα αυτή αναπτύχθηκε πάνω στην ιδέα του απλού παιχνιδιού κυνηγητού, με σκοπό την υλοποίηση ενός περιβάλλοντος και των πρακτόρων που θα παίζουν. Στο παιχνίδι αυτό, οι κυνηγοί καλούνται να πιάσουν τους αντίπαλους, ενώ αντίστοιχα οι κυνηγημένοι προσπαθούν να παραμείνουν ελεύθεροι. Στόχος της εργασίας είναι, πέρα από την κατανόηση των κανόνων και την λήψη των σωστών αποφάσεων για την επίτευξη της νίκης, η εύρεση νέων στρατηγικών που θα οδηγήσουν σε βέλτιστα αποτελέσματα αλλά και η σύγκριση δημοφιλών αλγορίθμων ανάλογα με το πρόβλημα.

Οι προτεινόμενες προσεγγίσεις έχουν επιλεγεί μετά από ανάλυση ενός μεγάλου εύρους αλγορίθμων σε Reinforcement Learning σε ένα γραφικό περιβάλλον εξομοίωσης. Τα αποτελέσματα που επιτεύχθηκαν παρουσιάζουν την βέλτιστη τεχνική υλοποίησης, παράλληλα όμως επισημαίνονται και προοπτικές βελτίωσης όσο στον τρόπο λήψεων αποφάσεων αλλά και στην αντιμετώπιση πολυπλοκότερου περιβάλλοντος και κανονισμών του παιχνιδιού.

Abstract

The field of Artificial intelligence in modern era is an important tool in the implementation of modern ideas and the development of many useful applications. Machine learning provides a variety of techniques and algorithms for problem solving where classical programming is limited. A similar problem is presented in this work, which is a theoretical and practical approach in the field of machine learning aiming in finding suitable techniques for solving complex problems.

Creating intelligent artificial agents to solve human-related problems is a major challenge for the artificial intelligence community. Particularly important is the understanding of the dynamic environment in which they operate and their interaction with it, just like humans do in the physical world. The field that specializes in creating such agents is called Reinforcement Learning. With the term Reinforcement Learning, we refer to the methods through which a system of algorithms 'learns' to interact within a structured environment after trial and error. This learning is done by exploring the environment through actions and rewards given by the environment to achieve a goal with optimal effort.

This research was developed on the idea of the simple chase game, which aims to create an environment and the agents that play in it. In this game, hunting agents are tasked with catching opponents, while the hunted agents try to stay free. The goal of the work is, apart from understanding the rules and getting the right decisions to achieve victory, finding new strategies that will lead to optimal results and comparing the most valuable algorithms.

The proposed approach has been chosen after analysing a wide range of Reinforcement Learning algorithms in a graphical emulation environment. The results achieved show the best implementation technique, but at the same time prospective improvements are also highlighted in the way of getting the right decisions but also in dealing with the more complex environment and rules of the game.

1. Εισαγωγή

1.1 Τι είναι Reinforcement Learning?

Η μάθηση στη φύση προκύπτει μέσω της διάδρασης των ζωντανών οργανισμών με το περιβάλλον τους. Με την αλληλεπίδραση αυτή, οι οργανισμοί εξελίσσονται, μαθαίνουν από τις ενέργειες τους και τις επιπτώσεις τους και προσαρμόζονται στον χώρο στον οποίο ζούν με σκοπό την επίτευξη στόχων που έχουν άμεση σύνδεση με την επιβίωση τους. Σε αυτή την ιδέα είναι βασισμένη και η λογική του reinforcement learning. Αποτελεί έναν τομέα της μηχανικής μάθησης που αντιμετωπίζει προβλήματα λήψης αποφάσεων αποκτώντας εμπειρία μέσα σε ένα περιβάλλον.

Πιο συγκεκριμένα, μέσα σε ένα σύστημα κανόνων, ο agent δρα στον χώρο και λαμβάνει ανταμοιβή (reward) η οποία αξιολογεί αυτή την πράξη του ανάλογα με το πόσο αυτή η πράξη συνέβαλε στην επίτευξη ενός στόχου. Τελικός σκοπός είναι η επιλογή των σωστών αποφάσεων ώστε να μεγιστοποιηθεί η αθροιστική ανταμοιβή.

Αυτό που ξεχωρίζει τη μεθοδολογία του reinforcement learning από άλλες μεθόδους machine learning (π.χ. supervised learning) είναι ότι δίνεται μόνο μερική ανατροφοδότηση στον agent από τις προβλέψεις του. Παράλληλα, οι προβλέψεις αυτές μπορεί να έχουν μακρυπρόθεσμα μεγάλη επιρροή στις μελλοντικές καταστάσεις του agent. Άρα, αποδεικνύεται η σημαντικότητα του ρόλου που έχει ο χρόνος στην μάθηση του agent, καθώς κάθε απόφαση επηρεάζει άμεσα και έμμεσα όλες τις καταστάσεις του συστήματος μέχρι την περάτωση του στόχου.

Σημαντική ακόμα παράμετρος του RL είναι το γεγονός ότι η εκπαίδευση γίνεται μέσω της μεθόδου trial and error. Ο agent δεν έχει την πλήρη εικόνα ή τον έλεγχο του περιβάλλοντος στο οποίο υπάρχει, συνεπώς χρειάζεται να συλλέξει πληροφορίες οι οποίες θα καθορίσουν και την συμπεριφορά του. Σημειώνεται πως μέσω αυτής της τακτικής προκύπτει το δίλημμα exploration/exploitation στο οποίο θα γίνει εκτενέστερη αναφορά και στα επόμενα κεφάλαια.

1.2 Ιστορική Αναδρομή

Η έννοια του reinforcement learning δεν είναι νέα, αλλά έχει τροποποιηθεί και αναπτυχθεί κατά την διάρκεια των τελευταίων 70 χρόνων. Η επιστημονική κοινότητα ακολούθησε ταυτόχρονα τρεις διαφορετικές νοοτροπίες ενισχυμένης μάθησης (trial and error, optimal control, temporal difference), ώσπου εν τέλει ενώθηκαν στην έρευνα της δεκαετίας του 1990 όπως την γνωρίζουμε και σήμερα.

- Trial and Error:

  Η μεθοδολογία αυτή ενσωματώθηκε στη μηχανική μάθηση μετά την έρευνα του Minsky (1954) και την χρήση των SNARCs (Stochastic Neural-Analogue Reinforcement Calculators). Με την έρευνα των Clark και Farley (1955) στην αναγνώριση μοτίβων αλλά και του Rosenblatt (1962) όπου και θεμελιώθηκε η θεωρία των νευρωνικών δικτύων και η ενημέρωση συνδεδεμένων νευρώνων, θεωρήθηκε πως δεν υπάρχει διάκριση μεταξύ reinforcement και supervised learning. Το 1961, επίσης από τον Minsky, τέθηκε το 'Credit Assignment Problem', η δυσκολία δηλαδή με την οποία ορίζουμε ποιές ενέργειες είναι αυτές οι οποίες συνέβαλαν περισσότερο στην εκπλήρωση του τελικού στόχου. Τελικά, ο John Andreae (1963-1977) με το STELLA το οποίο μάθαινε μέσω της αλληλεπίδρασης του με το περιβάλλον, οριστικοποιήθηκε ο διαχωρισμός των δυο πεδίων.

- Optimal Control

  Η έρευνα ξεκίνησε τη δεκαετία του 1950 και ορίζεται ως '*ένας ελεγκτής που ελαχιστοποιεί το μέτρο ενός δυναμικού συστήματος συμπεριφοράς του στο χρόνο*' (Sutton και Burto 2018). Ο Bellman (1957) υλοποίησε την ευρέως γνωστή συνάρτηση Bellman, η οποία ορίζει δυναμικά μια συναρτησιακή εξίσωση χρησιμοποιώντας την κατάσταση ενός δυναμικού συστήματος, και επιστρέφει μια βέλτιστη συνάρτηση τιμής. Στη συνέχεια εισήγαγε το MDP (Markovian Decision Process) την οποία ορίζει ως *'μια διακριτή στοχαστική έκδοση του optimal control'*.

- Temporal Difference

  Εμπνευσμένο από τον διαφορικό λογισμό, η μάθηση μέσω temporal difference στοχεύει στην πρόβλεψη μέσω ενός σετ γνωστών μεταβλητών της παρούσας εκτίμησης της συνάρτησης τιμής, μια μεθοδολογία παρόμοια με τη μέθοδο Monte Carlo (Hammersley 1964). Αναπτύχθηκε κυρίως μετά την έρευνα των Minsky (1954) και Samuel (1959).

Τελικά, οι τρεις αυτές παραδοχές ενωποιήθηκαν με την έρευνα του Watkins (1989) όπου και προτάθηκε η ιδέα του Q-Learning. Τις επόμενες δεκαετίες, οι ιδέες αυτές εξελίχθηκαν και εφαρμόστηκαν πάνω σε πολλούς τομείς, ξεκινώντας με τα επιτραπέζια παιχνίδια όπως το τάβλι, σκάκι και Go (Tesauro 1994, Baxter 2000, DeepBlue IBM 1997, Google AlphaGo 2016). Ακολούθησαν εφαρμογές σε ηλεκτρονικά παιχνίδια Atari, καθώς μπορούσε να υλοποιηθεί εύκολα λόγω των απλών χειρισμών των παιχνιδιών και του γραφικού περιβάλλοντος που αποτελείται από pixels. Μέχρι και σήμερα, το reinforcement learning εφαρμόζεται σε διάφορους τομείς όπως έξυπνα αυτοκίνητα, διαφημίσεις, συστήματα web, ρομποτικές εφαρμογές, ακόμα και στην χημεία.

1.3 Στόχος της εργασίας και εφαρμογες

Στην εργασία αυτή υλοποιήθηκε ένα ολοκληρωμένο σύστημα εκπαίδευσης agent μέσα σε ένα γραφικό περιβάλλον κανόνων. Βασικός στόχος της υλοποίησης είναι η εφαρμογή διαφόρων τεχνικών και αλγορίθμων και σύγκριση αυτών για την εύρεση της πιο αποτελεσματικής μεθοδολογίας. Παράλληλα, μέσω των πειραμάτων, έγινε εξαγωγή συμπερασμάτων ως προς τον τρόπο με τον οποίο η εκάστοτε μεθοδολογία βοηθά σε συγκεκριμένες περιπτώσεις του προβλήματος, αλλά εξίσου σημαντική είναι και η σύγκριση των αποφάσεων και των ενεργειών των agent σε σχέση με τον άνθρωπο και οι διαφορετικές λογικές με τις οποίες ενεργούν.

Το παρόν πείραμα, αλλά και πειράματα παρόμοιας φύσης που επιλύονται μέσω του reinforcement learning, μπορούν να εφαρμοστούν και σε προβλήματα στον πραγματικό κόσμο. Παραδείγματα τέτοιων χρήσεων συναντώνται συχνά όλο και περισσότερο σε ερευνητικές και εμπορικές εφαρμογές, ιδιαίτερα τα τελευταία χρόνια και με την ραγδαία βελτίωση της επεξεργαστικής ισχύς των υπολογιστικών συστημάτων.

Αυτο-οδηγούμενα Οχήματα

Όλο και περισσότερες δημοσιεύσεις αφορούν την τεχνολογία των αυτοκινούμενων οχημάτων και την βελτίωσης χαρακτηριστικών τους, όπως για παράδειγμα η εναλλαγή λωρίδας στον δρόμο, αποφυγή εμποδίων, διατήρηση της ταχύτητας και αυτόματο παρκάρισμα. Ακόμα, παρόμοια μέθοδος μπορεί να εφαρμοστεί για την βελτιστοποίηση του ελέγχου των φαναριών με σκοπό την μείωσης της κίνησης και αποφυγής ατυχημάτων. Πολλές από αυτές τις εφαρμογές έχουν αναπτυχθεί με τη βοήθεια του Q-Learning αλγορίθμου που αναλύεται και στο επόμενο κεφάλαιο.

Ρομποτικές Εφαρμογές

Όλο και συχνότερα σε γραμμές παραγωγής χρησιμοποιούνται ρομποτικά συστήματα, καθώς εκτελούν ενέργειες γρηγορότερα και σε περιβάλλοντα υψηλού κινδύνου για τον άνθρωπο. Εκπαιδεύοντας το ρομπότ να εκτελεί συγκεκριμένες διαδικασίες με τον βέλτιστο τρόπο, μειώνεται ο χρόνος εκτέλεσης αλλά και η κατανάλωση ενέργειας του έως και 40%. Αυτό επιτυγχάνεται επίσης με βαθιά δίκτυα Q-Learning (QT-Opt) καθώς υποστηρίζει συνεχής κινήσεις στο χώρο.

Video Games

Όσο αφορά το συνεχώς αυξανόμενο μέτωπο της gaming βιομηχανίας, κάτι στο οποίο ειδικεύεται και η παρούσα εργασία. Συνεχώς όλο και περισσότερες εταιρίες ασχολούνται με το αντικείμενο προσπαθώντας όχι μόνο να υπερβούν τις ανθρώπινες δυνατότητες αλλά και να επιτύχουν όσο το δυνατό περισσότερο ρεαλιστική εμπειρία στον παίχτη.

Marketing και Διαφήμιση

Μεγάλη ανάπτυξη του τομέα βρίσκεται και στον χώρο της διαφήμισης. Συγκεκριμένα, η διαφήμιση προϊόντων σε συγκεκριμένες ομάδες ανθρώπων είναι πολύ σημαντική, καθώς η εύρεση της σωστής ομάδας αποσκοπεί σε μεγαλύτερα ποσά επένδυσης. Τέλος, με την άνοδο της χρήσης των κοινωνικών δικτύων, συναντώνται όλο και περισσότερες εφαρμογές εκπαίδευσης για την εμφάνιση διαφημίσεων σε σχέση με το προφίλ του κάθε χρήστη, τα ενδιαφέροντα και τις προτιμήσεις του, ακόμα με τα άτομα με τα οποία συνομιλεί.

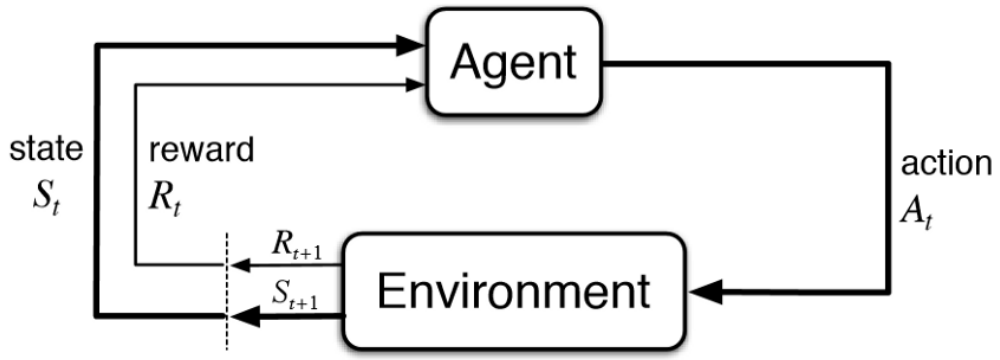
2. Βασικοί Ορισμοί και Μεθοδολογία

2.1 Δομή και ορισμοί

Η βασική δομή ενός Reinforcement Learning συστήματος αποτελείται από δυο οντότητες, τον agent και το περιβάλλον. Το περιβάλλον αποτελεί τον χώρο στον οποίο ο agent ενεργεί, στέλνοντάς του την κατάσταση για κάθε χρονική στιγμή t, ενώ ο agent, που αντιπροσωπεύει τον ιδιο τον RL αλγόριθμο, κάνει μια ενέργεια βάση της κατάστασης στην οποία βρίσκεται. Στην αμέσως επόμενη χρονική στιγμή, το περιβάλλον μαζί με την νέα κατάσταση, στέλνει και το reward στον agent. Ο agent με την σειρά του, αναθεωρεί την γνώση την οποία έχει με το reward αυτό και έτσι διαμορφώνει τις επόμενες ενέργειές του. Η επανάληψη αυτή συνεχίζεται έως ότου το περιβάλλον στείλει μια τερματική κατάσταση στον agent, όπου τελειώνει και το επεισόδιο.

Ως επεισόδιο ορίζεται η παραπάνω επανάληψη μέχρι την τερματική κατάσταση του περιβάλλοντος. Τερματική κατάσταση μπορεί να θεωρηθεί όταν ο agent εκπληρώσει τον στόχο του ή όταν υπερβεί κάποιο threshold κινήσεων στον χώρο προς αποφυγή ατέρμωνου βρόγχου.

Η λογική αυτή περιγράφεται και στη Εικόνα 1.



Εικόνα 1: Βασική δομή RL συστήματος

Η πλειοψηφία των RL αλγορίθμων ακολουθούν την παραπάνω τακτική. Βασικοί όροι που χρησιμοποιούνται είναι:

- Agent: η οντότητα που ενεργεί στο περιβάλλον και δέχεται reward για τις ενέργειες αυτές
- Περιβάλλον (e): Ο χώρος στον οποίο δρά ο agent.
- State (S): Η παρούσα κατάσταση την οποία επιστρέφει το περιβάλλον.
- Action (A): Όλες οι πιθανές ενέργειες που μπορεί να κάνει ο agent σε κάθε κατάσταση.
- Reward (R): Η ανταμοιβή που δίνεται στον agent για κάθε ενέργεια.

Ένα σύστημα Reinforcement Learning επίσης χαρακτηρίζεται και από τρία σημαντικά στοιχεία που χαρακτηρίζουν τον τρόπο συμπεριφοράς του agent, την τακτική (policy), την συνάρτηση τιμής (value function) και το μοντέλο του συστήματος (model).

Policy

Το policy ορίζει την συμπεριφορά του agent άνα κάποια χρονική στιγμή t. Είναι ο πυρήνας του agent καθώς είναι από μόνο του αρκετό για να ορίσει την εκάστοτε συμπεριφορά. Πιο συγκεκριμένα, το policy αντιστοιχίζει το state στο οποίο βρίσκεται ο agent με το action το οποίο θα προβεί. Σε πολλές περιπτώσεις μπορεί να είναι μια απλή συνάρτηση ή ένας πίνακας τιμών, υπάρχουν όμως και περιπτώσεις που μπορεί η λογική του να είναι αρκετά πιο πολύπλοκη εμπεριέχοντας εκτεταμένη υπολογιστική διαδικασία όπως έναν αλγόριθμο αναζήτησης. Γενικότερα, τα policies είναι στοχαστικές διαδικασίες με συγκεκριμένες πιθανότητες για κάθε ενέργεια.

Value Function

Αντίστοιχα, το value function αναφέρεται στο πόσο σωστή είναι μια ενέργεια μακρυπρόθεσμα. Η τιμή του κάθε state είναι το σύνολο του reward που μπορεί να λάβει ο agent στο μέλλον, ξεκινώντας από το παρόν state, διαφοροποιώντας το απο το reward που εκφράζει τη άμεση επιθυμητή ανταμοιβή του state. Γίνεται λοιπόν κατανοητό πως, η εκτίμηση του value function είναι αρκετά δυσκολότερο να βρεθεί από το reward, έχει όμως βασικό ρόλο κατά την διάρκεια της εκπαίδευσης.

Model

Τέλος, το μοντέλο μιμείται την συμπεριφορά του περιβάλλοντος. Χρησιμοποιείται για τον σχεδιασμό και την πρόβλεψη των επομένων καταστάσεων πριν ο agent βρεθεί σε αυτές, μαθαίνοντας την πιθανότητα μετάβασης από μια κατάσταση s_0 σε μια άλλη s με βάση μια ενέργεια α.

$$P(s_1 \mid (s_0, a))$$

Model-based και Model-free

Στα προβλήματα ενισχυμένης μάθησης, τα συστήματα κατηγοριοποιούνται σε model-based και model-free μεθόδους, ανάλογα με το αν τα μοντέλα αυτά σχεδιάζονται εξαρχής ή αν δημιουργούνται στην πορεία μέσω της διαδικασίας του trial and error. Συχνά χρησιμοποιούνται τα model-free συστήματα καθώς δεν απαιτούν χώρο για την αποθήκευση όλων των συνδιασμών state-action και επιπλέον προσφέρουν μια πιο δυναμική προσέγγιση σε μεταβαλλόμενα περιβάλλοντα.

Παρακάτω, γίνεται μια εκτενής αναφορά σε Reinforcement Learning αλγορίθμους, σημαντικούς για την εξέλιξη και τη δημιουργία του τελικού αλγορίθμου που χρησιμοποιήθηκε. Αναφορικά, γίνεται ανάλυση της Μαρκοβιανής Διαδικασίας Αποφάσεων (MDP), του αλγορίθμου Q-Learning και Deep Q-Learning, του NEAT (Neuro-Evolution of Augmenting Topologies) αλλά και λοιπών στοχαστικών αλγορίθμων.

2.2 Markov Decision Process (MDP)

2.2.1 Εισαγωγή

Η μαρκοβιανή διαδικασία αποφάσεων (MDP) είναι μια μαθηματική στοχαστική διαδικασία διακριτού χρόνου πολύ σημαντική για την μελέτη προβλημάτων βελτιστοποίησης μέσω δυναμικού προγραμματισμού. Σχεδόν όλα τα προβήματα Reinforcement Learning μπορούν να επιλυθούν με την χρήση MDP ή να προσαρμοστούν σε αυτό. Η βασική ιδέα είναι ότι ο agent αποφασίζει την επιλογή της καλύτερης ενέργειας με βάση την παρούσα κατάσταση του. Παρακάτω ακολουθεί μια περιγραφή των σημαντικότερων ιδιοτήτων του MDP καθώς και μια σύντομη περιγραφή για την επίλυση ενός προβλήματος σε ένα πλήρως επιβλεπόμενο περιβάλλον.

2.2.2 Μαθηματική περιγραφή του MDP

Πιο αναλυτικά, σύμφωνα με την ιδιότητα Markov, μια κατάσταση S_t εξαρτάται μόνο από την τελευταία κατάσταση και αγνοώντας όλες τις παρελθοντικές καταστάσεις, δηλαδή:

$$P[S_{t+1} \mid S_t] = P[S_{t+1} \mid S_1, ..., S_t] \tag{1}$$

όπου S είναι οι καταστάσεις του περιβάλλοντος.

Επομένως, συμπεραίνουμε πως:

$$P_{ss'} = P[S_{t+1} = s' \mid S_t = s] \tag{2}$$

όπου $P_{ss'}$ είναι η πιθανότητα μετάβασης από την παρούσα κατάσταση στην επόμενη. Η εξίσωση (2) ορίζεται ως αλυσίδα Markov και αποτελείται από μια σειρά από καταστάσεις S_1, S_2, … που όλες υπακούν στην ιδιότητα Markov.

Η Μαρκοβιανή Διαδικασία Ανταμοιβής (Markov Reward Process) ορίζεται ως το εκτιμώμενο reward από όλες τις πιθανές καταστάσεις που μπορεί ο agent να μεταβεί από μια κατάσταση s. Πιο συγκεκριμένα:

$$R_s = \mathbb{E}[R_{t+1} \mid S_t = s] \tag{3}$$

Καταλήγοντας, η μαρκοβιανή διαδικασία αποφάσεων (MDP), συναρτήσει και (1), (2) και (3), η πιθανότητα μετάβασης καταστάσεως και ανταμοιβής ορίζονται ως:

$$R_s^a = \mathbb{E}[R_{t+1} \mid S_t = s, A_t = a] \tag{4}$$

$$P_{ss'}^a = \mathbb{P}[S_t + 1 = s' \mid S_t = s, A_t = a] \tag{5}$$

Καθώς το reward είναι προσωρινό, μπορεί μετά από μια ενέργεια να έχουμε μεγάλο reward αλλά μακροπρόθεσμα να επιτυγχάνεται μικρότερο από το επιθυμητό. Το μακρυπρόθεσμο αυτό reward ορίζεται ως Return, το οποίο στην πράξη υπολογίζεται με την βοήθεια ενός discount (γ). Η λογική πίσω από την χρήση του discount είναι ότι δεν μπορούμε να είμαστε 100% σίγουροι για το μέλλον. Συνεπώς, είναι σημαντικό να θέτουμε υπ' όψιν μας τα μελλοντικά rewards αλλά και να ελαχιστοποιήσουμε την συνεισφορά τους μέσω του discount.

Πιο συγκεκριμένα:

$$G_t = R_{t+1} + \gamma R_{t+2} + ... = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \tag{6}$$

όπου γ ανήκει στο [0, 1].

Η εκτίμηση του 'πόσο καλή' είναι η επιλογή μιας ενέργειας από τον agent σε μια συγκεκριμένη κατάσταση ορίζεται από το policy, που περιγράφει την συμπεριφορά του ως προς την λήψη αποφάσεων μέσω χαρτογράφησης των καταστάσεων που βρίσκεται με τις πιθανότητες των πιθανών ενεργειών. Η εκτίμηση αυτών των επιλογών γίνεται από τις συναρτήσεις τιμής (value function) για την ενέργεια που θα προβεί άλλα και την κατάσταση που βρίσκεται (action-value and state-value functions). Οι συγκεκριμένες συναρτήσεις εκτιμούνται μέσω τις εμπειρίας του agent και είναι σημαντικό να βελτιστοποιηθεί κατάλληλα.

Αναλυτικότερα, το policy είναι πιθανοτική κατανομή ενεργειών α βάση της παρούσας κατάστασης s.

$$\pi(a \mid s) = \mathbb{P}[A_t = a \mid S_t = s] \tag{7}$$

Η συνάρτηση τιμής (value function) εμφανίζει τη μακρυπρόθεσμη τιμή της κατάστασης s, δηλαδή:

$$v(s) = \mathbb{E}[G_t \mid S_t = s] \tag{8}$$

Αντίστοιχα, και σύμφωνα με την (8), τα state-value και action-value functions που υπόκεινται σε ένα policy π ορίζονται ως:

$$v_\pi(s) = \mathbb{E}_\pi[G_t \mid S_t = s] \tag{9}$$

$$q_\pi(s, a) = \mathbb{E}_\pi[G_t \mid S_t = s, A_t = a] \tag{10}$$

Η συνάρτηση Bellman δίνει μια απεικόνιση της συνάρτησης τιμής. Αποτελείται από δύο επιμέρους στοιχεία:

- Το άμεσο reward
- Την τιμή των μελλοντικών καταστάσεων μετά το discount

$$\begin{aligned} v(s) &= \mathbb{E}[G_t \mid S_t = s] \\ &= \mathbb{E}[R_{t+1} + \gamma v(S_{t+1}) \mid S_t = s] \end{aligned} \tag{11}$$

Πιο συγκεκριμένα, ένας agent μπορεί να μεταβεί από μια κατάσταση s σε μια κατάσταση s’. Η state value function υπολογίζει την προσδοκώμενη τιμή του return από όλες τις επόμενες καταστάσεις s’. Χρησιμοποιώντας επαναλλαμβανόμενα τον ίδιο ορισμό για κάθε επόμενη κατάσταση s’, οδηγούμαστε στην συνάρτηση (11).

Συμπεραίνοντας, καταλήγουμε στην παρακάτω συνάρτηση Bellman για την βέλτιστη τιμή:

$$v_*(s) = R(s) + \max_a \gamma \sum_{s'} P_{sa}(s') v_*(s') \tag{12}$$

όπου, παίρνοντας τις τιμές για κάθε κατάσταση ενός MDP για όλα τα policies, επιλέγουμε το policy με τη μεγαλύτερη τιμή:

$$v_*(s) = \max_\pi v_\pi(s) \tag{12}$$

Αντίστοιχα, και για κάθε ενέργεια, επιλέγουμε το μέγιστο policy:

$$q_*(s, a) = \max_\pi q_\pi(s, a) \tag{13}$$

2.3 Q-Learning και Deep Q-Learning

2.3.1 Βασική ιδέα και διαφορές

Με τον όρο Q-Learning αναφερόμαστε σε έναν αλγόριθμο Reinforcement Learning με τα εξής χαρακτηριστικά:

- model-free: Κάνει εκτίμηση του βέλτιστου policy χωρίς να απαιτεί μοντέλο του περιβάλλοντος βάση των συναρτήσεων μετάβασης και των rewards.
- off-policy: Εκτιμά το reward για κάθε μελλοντική ενέργεια και ανανεώνει το policy, χωρις να ακολουθεί κάποια συγκεκριμένη συμπεριφορά.

Ο agent χρησιμοποιεί τα rewards του περιβάλλοντος για να εκτελέσει την καλύτερη δυνατή ενέργεια για την παρούσα κατάσταση. Στον απλό Q-Learning αλγόριθμο, η απόφαση αυτή λαμβάνεται σύμφωνα με έναν πίνακα Q ο οποίος αποτελείται απο Q-values όπου αντιστοιχίζονται οι ενέργειες με τις πιθανές καταστάσεις (state-action pairs). Κατά την εκπαίδευση ο πίνακας αυτός ανανεώνεται, βελτιώνοντας τα Q-values που αποφέρουν μεγαλύτερο reward.

Γίνεται αντιληπτό πως, όσο μεγαλύτερος είναι ο χώρος ή οι πιθανές ενέργειες στις οποίες μπορεί να προβεί ο agent, τόσο μεγαλώνει και ο πίνακας Q, άρα κατα συνέπεια εκπαιδεύεται και συντηρείται δυσκολότερα. Ακόμα περισσότερο εντείνεται το πρόβλημα εάν οι πιθανές καταστάσεις του agent βρίσκονται σε συνεχή χώρο ή είναι υπερπληθής.

Ο Deep Q-Learning αποτελεί μια λύση σε αυτό το πρόβλημα αντικαθιστώντας τον πίνακα με ένα νευρωνικό δίκτυο, το οποίο δέχεται σαν είσοδο την πιθανή κατάσταση και σαν έξοδο τα Q-values που αντιστοιχίζονται με την κάθε ενέργεια που μπορεί να εκτελεστεί.

2.3.2 Περιγραφή του Q-Learning

Για κάθε πεπερασμένη Μαρκοβιανή διαδικασία αποφάσεων, ο Q-Learning βρίσκει το βέλτιστο policy μεγιστοποιώντας την τιμή της συνολικής ανταμοιβής για όλες τις επόμενες καταστάσεις, ξεκινώντας από την παρούσα. Κανει χρήση του Temporal Differences (TD) για να εκτιμηθεί το $Q^*(s,a)$, το εκτιμώμενο discounted reward δηλαδή, και συνεπώς μαθαίνει μέσω της εμπειρίας του χωρίς γνώση του περιβάλλοντος.

Αρχικά, ορίζεται ο πίνακας Q, ο οποίος είναι και η δομή που χρησιμοποιείται για να υπολογιστεί η μέγιστη εκτιμώμενη μελλοντική ανταμοιβή για την ενέργεια της κάθε κατάστασης. Η αρχικοποίηση γίνεται με μηδενικές τιμές με διαστάσεις πίνακα MxN, με M τον αριθμό των καταστάσεων s και N τον αριθμό των ενεργειών α.

Στη συνέχεια, ο agent επιλέγει μια ενέργεια. Εδώ τίθενται σε εφαρμογή η στρατηγική epsilon-greedy. Αναλυτικά, η επιλογή της ενέργειας αρχικά γίνεται με τυχαίο τρόπο. Σταδιακά, και ενώ ο agent αποκτά καλύτερη εικόνα για το περιβάλλον, το epsilon μειώνεται, άρα ο agent γίνεται πιο άπληστος, επιλέγοντας μόνο ενέργειες που θα του αποφέρουν σίγουρα μεγάλη ανταμοιβή, άρα συνεπάγεται μικρότερη εξερεύνηση του χώρου (exploration-exploitation dilemma). Η ισορροπία ανάμεσα στο exploitation και exploration είναι πολύ σημαντική. Με μεγάλο exploration, ο agent θα περιφέρεται άσκοπα στον χώρο ακολουθώντας τακτικές που δεν τον βοηθούν στην επίτευξη του τελικού στόχου, ενώ με μεγάλο exploitation, κινδυνεύει ο agent να εγκλωβιστεί σε τοπικό ελάχιστο.

Μετά την εφαρμογή της ενέργειας, ο agent δέχεται το reward από το περιβάλλον και ενημερώνει τον πίνακα Q.

Οι τιμές του πίνακα Q ενημερώνονται μέσω της Q-function, η οποία χρησιμοποιεί την συνάρτηση Bellman παίρνοντας σαν εισόδους την κατάσταση (s) και την ενέργεια (α).

$$Q^{\pi}(s_t, a_t) = \mathbb{E}[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \ \ldots \ \mid s_t, a_t] \tag{15}$$

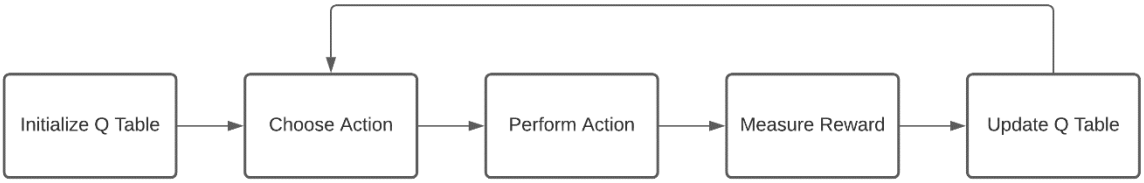
Από την (15), για την εύρεση της βέλτιστης τιμής, συνεπάγεται:

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha \left[R_{t+1} + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t) \right] \tag{16}$$

όπου α είναι η υπερπαράμετρος που εκφράζει τον βαθμό εκπαίδευσης του αλγορίθμου (learning rate).

Η συνάρτηση (16) περιγράφει την ενημέρωση της νέας τιμής Q προσθέτοντας την παλιά τιμή συν τη νέα εκτιμώμενη τιμή ανταμοιβής.

Η διαδικασία αυτή επαναλαμβάνεται μέχρι να σταματήσει η εκπαίδευση. Παρακάτω παρουσιάζεται ένα διάγραμμα δίνοντας μια σύντομη περιγραφή του αλγορίθμου.



Εικόνα 2: Διάγραμμα Αλγορίθμου Q-Learning

2.3.3 Περιγραφή του Deep Q-Learning

Ο Deep Q-Learning αλγόριθμος βασίζεται στην ίδια λογική με τον Q, με την διαφορά ότι χρησιμοποιούμε νευρωνικά δίκτυα αντί για έναν πίνακα Q για να εκτιμηθούν οι Q-values τιμές. Πιο συγκεκριμένα, σε ένα τέτοιο δίκτυο η είσοδος είναι η αντίστοιχη παρούσα κατάσταση και έξοδος οι τιμές Q-values για κάθε μια από τις ενέργειες.

Για την υλοποίηση ενός τέτοιου δικτύου, το πρώτο βασικό βήμα είναι η αποθήκευση στην μνήμη την προηγούμενη εμπειρία του agent. Η διαδικασία αυτή ονομάζεται replay memory:

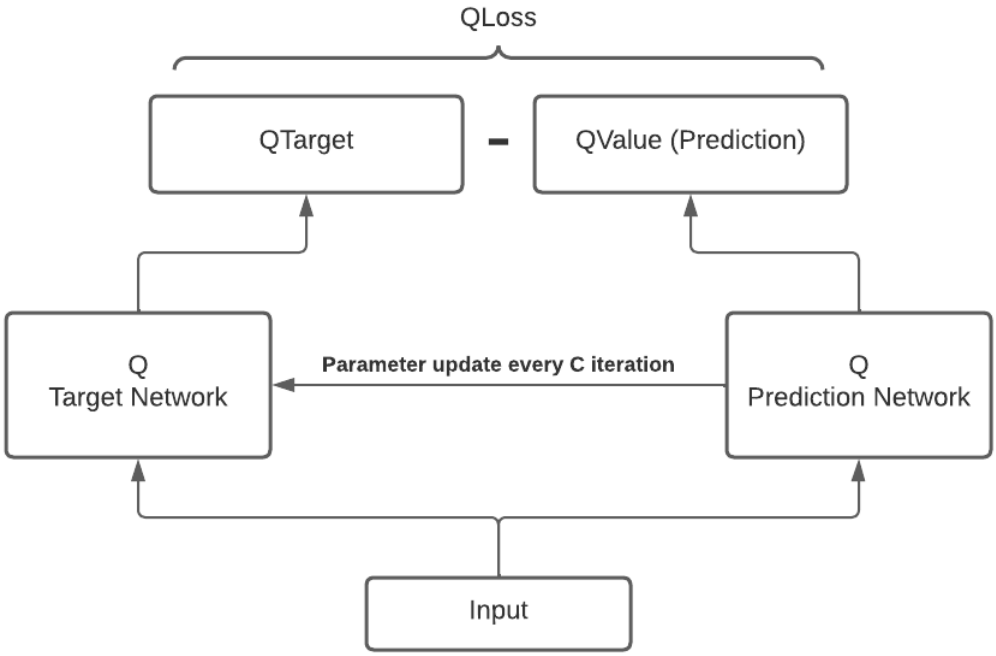
$$e_t = (s_t, a_t, r_t, s_{t+1}) \tag{17}$$

Ο λόγος που χρησιμοποιείται η replay memory είναι αρχικά για να γίνει πιο αποδοτική η εμπειρία του agent καθώς επαναχρησιμοποιείται κατά την διάρκεια της εκπαίδευσης. Αυτό επιτρέπει στο σύστημα να μαθαίνει από συγκεκριμένα παραδείγματα πολλαπλές φορές. Επιπλέον, βοηθά στο να μην ‘ξεχνά’ προηγούμενες εμπειρίες του αλλά και στη μείωση της συσχέτισης μεταξύ παρόμοιων καταστάσεων.

Το επόμενο βήμα αφορά την εκπαίδευση του δικτύου. Σύμφωνα με την εξίσωση (16), μπορεί να υπολογιστεί η loss function του δικτύου, βρίσκοντας την διαφορά μεταξύ του Q value και του επιθυμητού Q. Αναλυτικότερα, χρησιμοποιώντας την εξίσωση Bellman (12), γίνεται εκτίμηση του επιθυμητού Q-value (Q-Target):

$$\begin{aligned} Q_{target} &= R_{t+1} + \gamma max_a Q(S_{t+1}, a) \\ Q_{loss} &= R_{t+1} + \gamma max_a Q(S_{t+1}, a) - Q(S_t, A_t) \end{aligned} \tag{18}$$

Μια σημαντική δυσκολία που παρουσιάζεται ακολουθώντας την παραπάνω λογική είναι ότι το σύστημα χρησιμοποιεί τις ίδιες παραμέτρους για την εκτίμηση του Q-target αλλά και του Q-value, με αποτέλεσμα σε κάθε βήμα της εκπαίδευσης, και οι δυο τιμές να μετακινούνται. Μια λύση είναι η υλοποίηση διαφορετικών νευρωνικών δικτύων, με το δίκτυο εκτίμησης του Q-Target να έχει σταθερές τιμές παραμέτρων, και να ανανεώνεται από το Q-value δίκτυο μετά από C βήματα.



Εικόνα 3: Διάγραμμα Υλοποίησης διαφορετικών δικτύων για τον υπολογισμό του QLoss

Αξίζει να σημειωθεί πως κατά τη διάρκεια του training κάθε μια προβλεπόμενη ενέργεια ακολουθεί, όπως και στο Q-Learning, μια epsilon-greedy policy για να αυξηθεί ο βαθμός exploration σε κάθε ενέργεια της εκπαίδευσης. Αμέσως μετά τον υπολογισμό του Q-Loss, εκτελείται ο gradient descent με σκοπό την μείωση του σφάλματος. Μέτα την ολοκλήρωση ενός συγκεκριμένου αριθμού επαναλήψεων C, τα βάρη του δικτύου μεταφέρονται στο target δίκτυο. Η όλη διαδικασία επαναλαμβάνεται για ένα σταθερό αριθμό επεισοδίων.

2.4 NeuroEvolution of Augmenting Topologies (NEAT)

2.4.1 Εισαγωγη στην Νευροεξέλιξη

Με τον όρο Νευροεξέλιξη (Neuroevolution) ορίζεται η μέθοδος της Μηχανικής Μάθησης με την οποία εφαρμόζονται εξελικτικοί αλγόριθμοι για την κατασκευή τεχνητών νευρωνικών δικτύων. Εφαρμογές με την μέθοδο αυτή πάνω σε προβλήματα ενισχυμένης μάθησης παρουσιάζουν υψηλά επίπεδα αποτελεσματικότητας, καθώς σε σύγκριση με παραδοσιακές μεθόδους μάθησης με στατικά νευρωνικά δίκτυα, εμφανίζουν μεγάλη γενίκευση που επιτρέπει την μάθηση χωρίς σαφείς στόχους και με μικρή ανατροφοδότηση.

Ένα μεγάλο ερώτημα στην Νευροεξέλιξη, είναι η ταυτόχρονη εξέλιξη της τοπολογίας των νευρωνικών δικτύων με την μάθηση των βαρών των συνδεδεμένων νευρώνων. Στις ανάγκες επίλυσης αυτού του προβλήματος αναπτύχθηκε ο αλγόριθμος NEAT (NeuroEvolution of Augmenting Topologies) που ανήκει στην ομάδα των TWEANNs (Topology and Weight Evolving Artificial Neural Networks). Ο NEAT δείχνει να υπερτερεί των εξελικτικών αλγορίθμων με συγκεκριμένη τοπολογία, δηλαδή μια εξαρχής δομημένη μορφή νευρωνικού δικτύου, η οποία εξελίσει μόνο τις τιμές των βαρών.

Σύμφωνα με τον Kenneth O. Stanley και Risto Miikkulainen στην έρευνά τους το 2002 (Evolving Neural Networks through Augmenting Topologies), η βασική ιδέα του NEAT περιστρέφεται γύρω από την αντιμετώπιση των προβλημάτων που αντιμετωπίζουν τα TWEANNs και στον τρόπο με τον οποίο σχεδιάστηκε ώστε να τα αντιμετωπίζει ο αλγόριθμος. Αυτές οι τεχνικές δυσκολίες που αναδύονται είναι οι εξής:

1. Η εύρεση μιας γενετικής αναπαράστασης των δικτύων που να επιτρέπει σε διαφορετικές τοπολογίες να εφαρμόσουν crossover μεταξύ τους.
2. Η προστασία της τοπολογίας, η οποία χρειάζεται λίγες γενιές για να βελτιστοποιηθεί, ώστε να μην εξαφανιστεί πρόωρα από τον πληθυσμό, καθώς απότερος σκοπός είναι η εύρεση μιας απλής τοπολογίας που να μπορεί να επωφεληθεί μονο από προσθήκη πολυπλοκότητας.
3. Η ελαχιστοποίηση των τοπολογιών κατά τη διάρκεια της εξέλιξης χωρίς την χρήση μια fitness function που να υπολογίζει και την πολυπλοκότητα.

2.4.2 Περιγραφή αλγορίθμου NEAT

Για την επίλυση του πρώτου προβλήματος, τα εκάστοτε δίκτυα ανάγονται σε μια αναπαράσταση που ονομάζεται genome και η αντίστοιχη έκφραση του δικτύου σε μορφή γράφου που ονομάζεται phenotype. Κάθε ένα από τα genome περιλαμβάνει μια λίστα από κόμβους σύνδεσης οι οποίοι περιγράφουν:

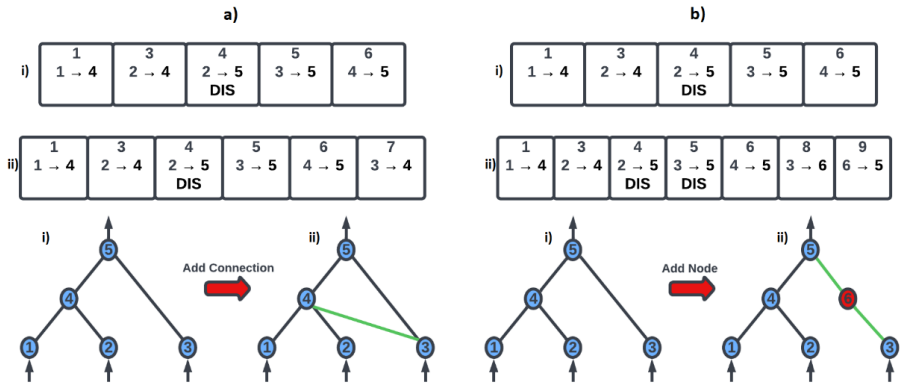
- Τον κόμβο εισόδου (in-node)
- Τον κόμβο εξόδου (out-node)
- Το βάρος (weight)
- Την ύπαρξη του κόμβου στο δίκτυο (enable bit)
- Έναν μοναδικό αριθμό καινοτομίας (innovation number)

Το innovation number είναι ιδιαίτερα μεγάλης σημασίας για την δημιουργία νέων δικτύων. Για κάθε ένα νέο κόμβο σύνδεσης (gene), θέτουμε ένα innovation number το οποίο είναι μοναδικό σε όλο το φάσμα της εκπαίδευσης. Η δημιουργία νέων δικτύων μπορεί να γίνει μέσω μιας από τις παρακάτω μεθοδολογίες.

Μετάλλαξη (Mutation)

Η μετάλλαξη των δικτύων μπορεί να γίνει είτε σε ένα υπάρχον δίκτυο, είτε προσθέτοντας νέα δομή σε αυτό, αλλάζοντας ταυτόχρονα την δομή του δικτύου αλλά και τα βάρη των συνδέσεων. Οι δομικές μεταλλάξεις υλοποιούνται με δυο διαφορετικούς τρόπους:

a) με add connection όπου γίνεται η ένωση δυο υπαρκτών κόμβων
b) με add node όπου δημιουργείται ένας νέος κόμβων και οι δυο αντίστοιχες συνδέσεις της εισόδου και της εξόδου του



Εικόνα 4: Παραδείγματα genotype και phenotype δικτύου κατα το mutation

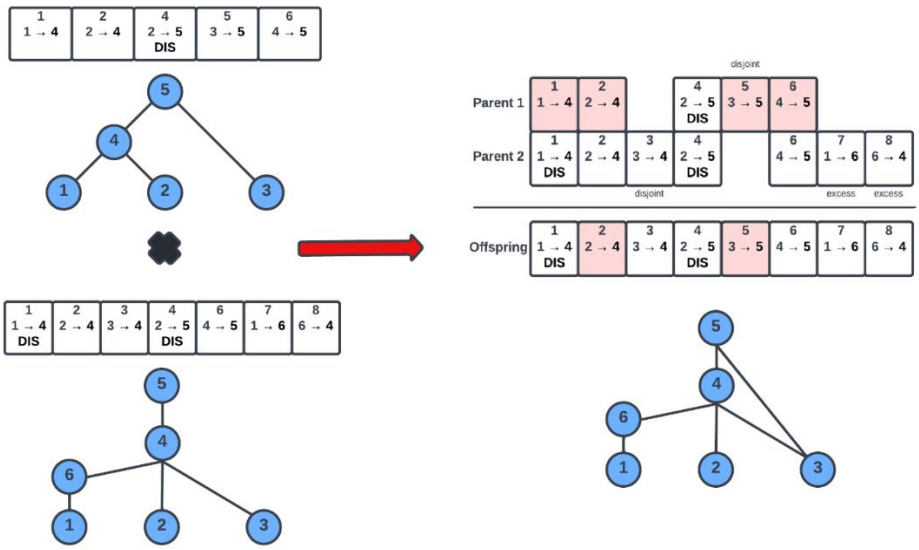
Μέσω της μετάλλαξης, δημιουργείται μια πληθώρα διαφορετικών δομών δικτύων ορισμένα στις ίδιες θέσεις θέτοντας υπόψιν και το innovation number.

Διασταύρωση (Crossover) με χρήση του innovation number

Ένα σημαντικό πρόβλημα με το crossover μεταξύ των δικτύων είναι ότι μπορεί να δημιουργήσει πολύ πολύπλοκα ή ακόμα και μη λειτουργικά δίκτυα λόγω εντελώς διαφορετικής δομής και μεγέθους. Στον αλγόριθμο NEAT αυτό επιλύεται εύκολα κάνοντας χρήση του innovation number, ενός ιστορικού αριθμού που δίνεται μετά από κάθε μια νέα εξέλιξη.

Αρχικά, για να εφαρμοστεί το crossover μεταξύ δυο genomes, ο αλγόριθμος χρειάζεται να γνωρίζει ποιά genomes μέσα από τον πληθυσμό προέρχονται από κοινό πρόγονο. Αυτά μπορούν να βρεθούν εύκολα χρησιμοποιώντας το innovation number του κάθε gene τους, το οποίο είναι μοναδικό. Έτσι, η προέλευση του κάθε gene στο σύστημα είναι γνωστή και βρίσκονται εύκολα τα ζευγάρια που μπορούν να ενωθούν, ενώ τα genes που δεν ταιριάζουν αποσυνδέονται ή προστίθενται.

Πιο συγκεκριμένα, τα genes ανάμεσα σε δύο genomes τα οποία είναι ίδια τότε παραμένουν και ευθυγραμμίζονται μεταξύ τους. Αυτά που δεν ταιριάζουν, άρα έχουν και διαφορετικά innovation numbers, κληρονομούνται από τον καλύτερο πρόγονο ή τον καλύτερο τύχαιο πρόγονο εάν και οι δύο είναι εξίσου καλοί.



Εικόνα 5: : Παραδείγματα genotype και phenotype δικτύου κατα το crossover

Με αυτόν τον τρόπο ο αλγόριθμος NEAT εφαρμόζει crossover μεταξύ δυο διαφορετικών δικτύων χωρίς να προσθέτει αχρείαστη πολυπλοκότητα στο δίκτυο ή να το επιβαρύνει με υπολογιστικά κοστοβόρα τοπολογική ανάλυση. Παράλληλα, μπορούν να ενωθούν οποιεσδήποτε δομές δικτύων χωρίς να χρειαστεί περαιτέρω ανάλυση και με αρκετά απλούστερο τρόπο.

Speciation

Ένα μεγάλο μειονέκτημα που παρουσιάζεται κατά την διάρκεια της εξέλιξης είναι το γεγονός ότι οι νέες δομές στην δημιουργία τους έχουν αυξημένο σφάλμα. Ειδικότερα, προσθέτοντας μια νέα σύνδεση ή έναν νέο κόμβο χωρίς να έχει γίνει κάποια βελτιστοποίηση στα βάρη του δικτύου οδηγεί στο να μην έχει καλή επίδοση άρα μεγάλο μειονέκτημα απέναντι στα υπόλοιπα δίκτυα του πληθυσμού. Ένας τρόπος αντιμετώπισης αυτού του προβλήματος είναι η προστασία αυτών των νέων δικτύων κατηγοριοποιώντας τον πληθυσμό σε είδη με βάση την τοπολογική τους ομοιότητα (Speciation). Με αυτή την μέθοδο, τα καινούργια δίκτυα προλαβαίνουν να βελτιστοποιήσουν τα βάρη τους πριν χρειαστεί να συγκριθούν με τα υπόλοιπα του πληθυσμού.

Ο διαχωρισμός των δικτύων υλοποιείται με τη βοήθεια των innovation numbers. Όσο μεγαλύτερος είναι ο αριθμός των μη όμοιων genes μεταξύ δύο genomes τόσο πιο ανόμοια αυτά χαρακτηρίζονται. Όσο μικρότερη είναι η απόσταση συμβατότητας δ δύο διαφορετικών δομών τόσο πιο όμοιες είναι οι δομές αυτές. Αυτό εκφράζεται ως:

$$\delta = \frac{C_1 E}{N} + \frac{C_2 D}{N} + C_3 \cdot W \tag{19}$$

όπου:

- E: ο αριθμός των excess genes
- D: ο αριθμός των disjoint genes
- W: η μέση διαφορά βαρών μεταξύ των δυο genomes
- N: ο αριθμός των genes του μεγαλύτερου genome
- c_1, c_2, c_3: η σημαντικότητα των παραπάνω παραμέτρων

Τα genomes που ανήκουν στην ίδια κατηγορία αντιπροσωπεύονται από ένα τυχαίο genome από αυτή την κατηγορία της προηγούμενης γενιάς. Εάν ένα genome δεν ανήκει σε καμία κατηγορία, τότε το genome αυτό δημιουργεί μια νέα κατηγορία με το ίδιο σαν αντιπρόσωπο.

Επίσης, οι δομές της ίδιας κατηγορίας μοιράζονται τα ίδια βάρη μεταξύ τους, αποτρέποντας από την κατηγορία να υπερβεί του συνολικού πληθυσμού, προστατεύοντας έτσι τις νέες δομές κατά τη διιάρκεια της εξέλιξης. Αυτό συμβαίνει με τη χρήση της τροποποιημένης fitness function (*f'i*) για μια δομή *i* που υπολογίζεται συναρτήσει της απόστασης *δ* από όλες τις άλλες δομές *j*.

Πιο συγκεκριμένα:

$$f_i' = \frac{f_i}{\sum_{j=1}^{n} sh(\delta(i,j))} \tag{20}$$

Η συνάρτηση sh είναι ίση με 0 όταν το δ(i, j) είναι πάνω από ένα όριο δ_t το οποίο ορίζουμε εμείς, αλλιώς είναι ίσο με 1.

Ελαχιστοποίηση Διαστάσεων (Minimizing Dimensionality)

Βασιζόμενοι στην γενικότερη ιδέα των TWEANNs, ο πληθυσμός αρχικοποιείται με διάφορες τυχαίες τοπολογίες ώστε να υπάρχει από την αρχή η διαφορετικότητα. Αντίθετα, ο NEAT ξεκινά από ένα χώρο μικρής διάστασης χωρίς hidden κόμβους, και σταδιακά αυξάνει την πολυπλοκότητα των δομών του, μειώνοντας έτσι τον χρόνο εκπαίδευσης και αυξάνοντας την απόδοση του συγκριτικά με τα TWEANNs.

2.5 Εναλλακτικές Μεθοδολογίες

2.5.1 Μέθοδοι

Παράλληλα, στην επίλυση προβλημάτων reinforcement learning, χρησιμοποιείται μια πληθώρα αλγορίθμων και παραλλαγές αυτών. Η επιλογή αυτών γίνεται ανάλογα με τη φύση του ίδιου του προβλήματος αλλά και άλλες παραμέτρους, όπως την πολυπλοκότητα του προβλήματος, την διάθεση ή μη δεδομένων επίλυσης, την επεξεργαστική ισχύ του συστήματος μας και άλλα. Μερικές από τις επικρατέστερες μεθόδους περιγράφονται παρακάτω.

2.5.2 Monte Carlo

Σε αντίθεση με τις προηγούμενες μεθόδους, στην μεθοδολογία Monte Carlo δεν απαιτείται επίγνωση του περιβάλλοντος αλλά μόνο εμπειρία από την αλληλεπίδραση με το περιβάλλον. Η βασική διαφορά με την μεθοδολογία του Markov είναι ότι σε αυτή την περίπτωση υπάρχουν πολλαπλές καταστάσεις οι οποίες όλες συμπεριφέρονται σαν διαφορετικά προβλήματα που συγγενεύουν μεταξύ τους.

Πιο συγκεκριμένα, η βάση των Monte Carlo μεθόδων υπόκεινται στην ιδέα της μέσης τιμής των αποτελεσμάτων πολλών πειραμάτων. Ομοίως λοιπόν και σε reinforcement learning προβλήματα, θέτοντας μια state-value function και για ένα συγκεκριμένο policy, μετά από πολλά πειράματα καταλήγουμε σε πολλά returns. Για όλα τα returns αυτά, καθώς εκφράζουν το μακροπρόθεσμο reward για κάθε ένα από τα πειράματα, βρίσκουμε την μέση τιμή τους και το αποτέλεσμα θα πρέπει να πλησιάζει την προσδοκώμενη τιμή.

Η παραπάνω λογική αναλύεται περαιτέρω και στον παρακάτω αλγόριθμο.

```
Initialize:
        π, V, Returns(s)
Loop:
        Create episode with policy π
        For s in episode:
                G ← return after first s
                Append G to Return(s)
                V(s) ← Average(Return(s))
```

2.5.3 Γενετικοί Αλγόριθμοι

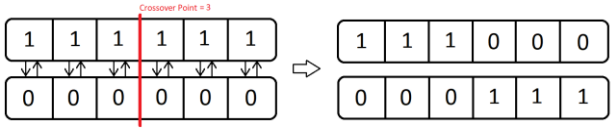
Οι γενετικοί αλγόριθμοι (Genetic Algorithms) σχεδιάστηκαν από τον John Holland την δεκαετία του 70’ και βασίζονται στις ιδέες του Δαρβίνου περί βιολογικής εξέλιξης μέσω της θεωρίας της φυσικής επιλογής. Ανήκουν στην ευρύτερη ομάδα των εξελικτικών αλγορίθμων και χρησιμοποιούνται για να την παραγωγή υψηλής ποιότητας λύσεων σε προβλήματα βελτιστοποίησης ή αναζήτησης.

Η διαδικασία ξεκινά με την δημιουργία μιας τυχαίας ομάδας λύσεων που ονομάζεται πληθυσμός (Population). Κάθε μια από αυτές τις λύσεις χαρακτηρίζεται από ένα σετ παραμέτρων που ονομάζεται γονίδια (Genes) και η ένωση αυτών ονομάζεται χρωμόσωμα όπου αποτελεί και την κάθε μια από τις λύσεις.

Η εύρεση των επικρατέστερων λύσεων γίνεται με την δημιουργία της fitness function, όπου βρίσκει το fitness score για κάθε λύση, και η πιθανότητα να επιλεγεί η λύση για αναπαραγωγή βασίζεται από αυτό το score. Κατά τη διάρκεια της αναπαραγωγής, τα γονίδια που θα περαστούν στην επόμενη γενιά παράγονται με τους παρακάτω τρόπους:

- Crossover

  Η σημαντικότερη από τις μεθόδους αναπαραγωγής, έχει ως σκοπό την επιλογή δύο λύσεων και την ένωση μερών αυτών για την δημιουργία ενός νέου απογόνου. Η επιλογή των μερών αυτών από τους γονείς γίνεται με την κατάλληλη επιλογή της παραμέτρου crossover point.

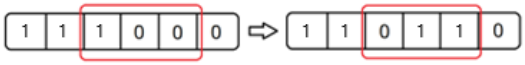


Εικόνα 6: Παραδειγμα Crossover

  Οι απόγονοι αυτής της αναπαραγωγής προστίθενται στον πληθυσμό.

- Mutation

  Σε αυτή την μέθοδο αναπαραγωγής εφαρμόζονται αλλαγές σε μια λύση-πρόγονο με τυχαίο τρόπο, το αποτέλεσμα του οποίου αποτελεί τον απόγονο που προστίθεται επίσης στον πληθυσμό. Η μέθοδος αυτή χρησιμοποιείται για να διατηρηθεί η ποικιλομοργία εντός του πληθυσμού αλλά και για να αποτραπεί η πρόωρη γενίκευση του αποτελέσματος.



Εικόνα 7: Παράδειγμα Mutation

- Elitism

  Μια συχνή τακτική σημιουργίας απογόνων είναι η μέθοδος του ελιτισμού (Elitism), καθώς επιτρέπει να περάσουν στην νέα γενιά οι καλύτερες από τις λύσεις χωρίς να υποστούν καμία επεξεργασία. Με την μέθοδο αυτή εξασφαλίζεται η διατήρηση της ποιότητας των λύσεων από την μια γενιά στην άλλη.

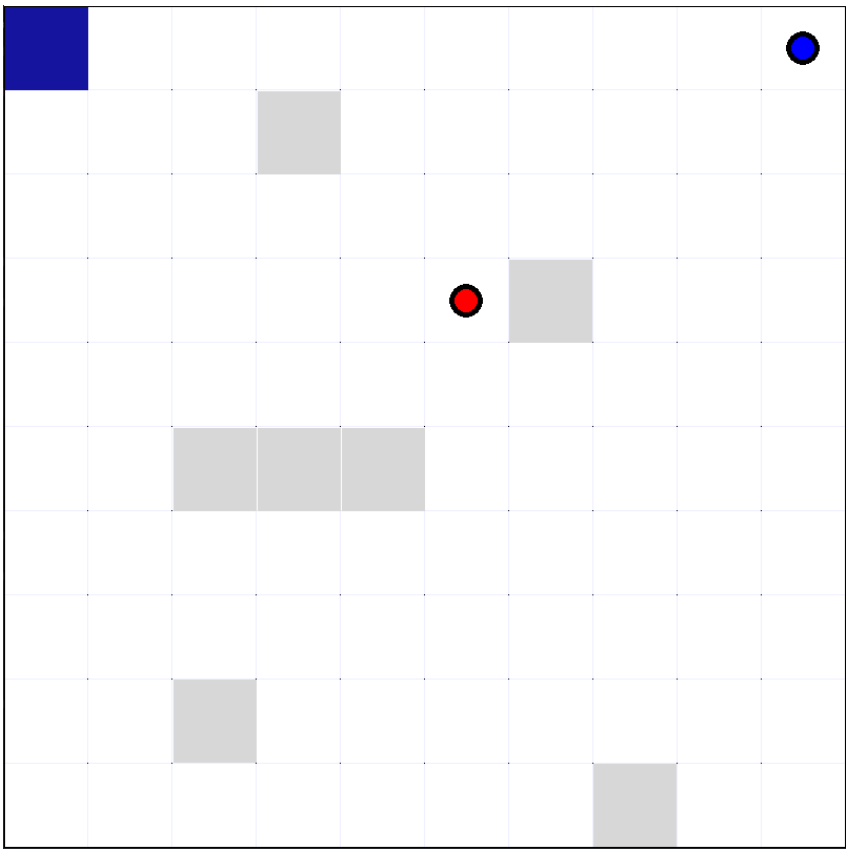
Οι γενετικοί αλγόριθμοι χρησιμοποιούνται πολύ συχνά και στην πλειοψηφία των εφαρμογών τους έχουν πολύ καλά αποτελέσματα. Παρόλαυτα, στην παρούσα εργασία δεν χρησιμοποιήθηκαν αυτούσια, καθώς ένα από τα βασικά μειονεκτήματά τους είναι η γενίκευση στα τοπικά ακρότατα, ιδίως σε προβλήματα μεγάλης πολυπλοκότητας. Αυτό σημαίνει πως δεν καταφέρνουν να αξιολογίσουν την σημαντικότητα των μακρυπρόθεσμων αποτελεσμάτων σε σχέση με τα βραχυπρόθεσμα, κάτι που καταφέρνει σε αντίθεση ο Q-Learning. Παρολαυτά, χρησιμοποιούνται ως ένα βαθμό όχι για την εύρεση της βέλτιστης λύσης του προβλήματος, αλλά την εύρεση της βέλτιστης δομής μέσω της οποίας θα βρεθεί η βέλτιστη λύση, κάτι το οποίο αναλύθηκε και στο προηγούμενο κεφάλαιο με την περιγραφή του NEAT αλγορίθμου.

Επίσης, η τελική λύση των γενετικών αλγορίθμων είναι καλύτερη μόνο σε σύγκριση με τις προηγούμενες, άρα συνεπώς δεν είμαστε σε θέση να γνωρίζουμε εάν η καλύτερη λύση είναι και η βέλτιστη.

3. Κανονισμοί Εξομοίωσης και Περιβάλλον

3.1 Κανονισμοί Παιχνιδιού

Στην παρούσα εργασία έχει υλοποιηθεί ένα περιβάλλον εξομοίωσης κυνηγητού μεταξύ δύο ομάδων agents. Ειδικότερα, η μπλε ομάδα έχει ένα συγκεκριμένο αριθμό κινήσεων, ανάλογα με τις διαστάσεις του χώρου στον οποίο δρουν, έως ώστε να πιάσει την αντίπαλη κόκκινη ομάδα. Αντίστοιχα, η κόκκινη ομάδα στο ίδιο διάστημα πρέπει να αποφύγει τον μπλε.



Εικόνα 8: Παράδειγμα Περιβάλλοντος Εξομοίωσης 10x10 με κινητά εμπόδια

Ο χώρος στον οποίο δρουν οι agents είναι ένα grid διαφόρων διαστάσεων. Εσωτερικά, το περιβάλλον μπορεί να περιέχει εμπόδια είτε σε μορφή τοίχου, είτε καλύπτωντας ολόκληρα κελιά. Τα πειράματα αυτά υλοποιήθηκαν σε τρεις διαφορετικούς χώρους για λόγους ποικιλομορφίας των πειραμάτων, διαστάσεων 5x5, 10x10 και 10x10 με εμπόδια.

3.2 Υλοποίηση Περιβάλλοντος Εξομοίωσης

Η δημιουργία του περιβάλλοντος εξομοίωσης έγινε με την χρήση της γλώσσας προγραμματισμού Python 3.9. Για την δημιουργία γραφικών χρησιμοποιήθηκε η βιβλιοθήκη pygame, ενώ για την δομή του περιβάλλοντος η βιβλιοθήκη Gym της OpenAI.

Το OpenAI Gym είναι ένα περιβάλλον υλοποίησης εκπαιδευόμενων agents. Ο λόγος που επιλέχθηκε για τη δημιουργία του περιβάλλοντος είναι η εύκολη ενσωμάτωση με μια μεγάλη ποικιλία RL αλγορίθμων, καθώς και η ουσιαστική σύγκρισή τους χωρίς μεγάλες τροποποιήσεις στον κώδικα από υλοποίηση σε υλοποίηση. Η συγκεκριμένη βιβλιοθήκη έχει χρησιμοποιηθεί σε πολλά projects που αφορούν RL implementations όπως τα Atari και MuJoCo environments, αλλά και projects που αφορούν Natural Language Processing, Simple Classification Problems, Game Theory κτλ.

Εκτός από τα υπάρχοντα περιβάλλοντα που παρέχει το Gym, δίνει τη δυνατότητα δημιουργίας custom περιβαλλόντων και σύνδεση τους με το interface του. Μερικές από τις βασικές μεθόδους που χρησιμοποιεί είναι:

- gym.make(env_name): Δημιουργία περιβάλλοντος από τα ήδη υπάρχοντα περιβάλλοντα. Σε αυτή την υλοποίηση το περιβάλλον έχει στηθεί σε άλλο python script με τη χρήση pygame και καλείται μέσω της κλάσης MazeView2D.
- env.reset(): Επαναφέρει το περιβάλλον στην αρχική του κατάσταση.
- env.render(): Σχεδιασμός και εμφάνιση του περιβάλλοντος.
- env.step(): Εκτελεί μια ενέργεια σε κάθε βήμα. Αποτελεί την σημαντικότερη μέθοδο καθώς μέσα σε αυτή εκτελείται ο αλγόριθμος εκμάθησης και η λογική απόδοσης των rewards.

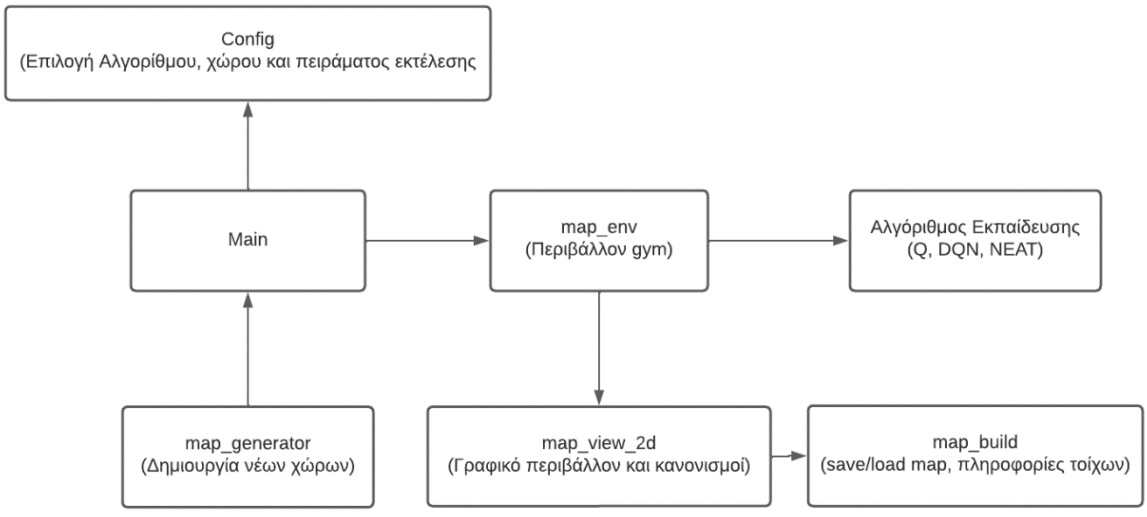
Με το πέρας της εκτέλεσης της env.step(), επιστρέφονται τέσσερις παράμετροι:

- observation: η κατάσταση του περιβάλλοντος εκείνη τη στιγμή
- reward: το reward που αποκτήθηκε με βάση την προηγούμενη ενέργεια.
- done: Boolean τιμή που προσδιορίζει το τέλος του επεισοδίου
- info: διαγνωστική πληροφορία που βοηθά στο debugging του κώδικα.

Σε κάθε step, ο agent επιλέγει μια action, και η env.step επιστρέφει το observation και το reward. Η παραπάνω υλοποίηση έχει αναπτυχθεί για διαφορετικό αριθμό πειραμάτων και διαφορετικών μεθοδολογιών που αναπτύχθηκαν μέσα στην class MapEnv(). Αντίστοιχα, ο σχεδιασμός του γραφικού περιβάλλοντος σε pygame και οι κανονισμοί του παιχνιδιού αναπτύχθηκαν στην class MapView2D.

Τέλος, έχει αναπτυχθεί και ο κώδικας δημιουργίας νέων χώρων map_generator.py για να πραγματοποιηθούν νέα πειράματα, αλλά και ένα config.py αρχείο όπου καταγράφονται οι επιθυμητοί παράμετροι για την φάση των train και test, δηλάδη αλγόριθμος μάθησης, επιλογή χώρου, στόχος πειράματος, ενεργοποιήση multi-processing κτλ. Αναλυτικό structure του project και λεπτομέρειες του κώδικα παρατίθονται και στο παράρτημα της εργασίας.

Συνολικά, τα επιμέρους τμήματα συνδέονται σύμφωνα με το παρακάτω διάγραμμα.



Εικόνα 9: Διάγραμμα Υλοποίησης Κώδικα Εξομοίωσης

4. Υλοποίηση

4.1 Επιλογή Αλγορίθμων και Φάσεις Πειραμάτων

Ανάμεσα στους προαναφερθέντες αλγόριθμους Ενισχυμένης Μάθησης, η επιλογή για την εξαγωγή των πειραμάτων έγινε με βάση τη φύση του προβλήματος και την πολυπλοκότητα των διεργασιών που καλούνται να εκτελέσουν οι agents, όλα αυτά συναρτήσει της επεξεργαστικής ισχύς την οποία διαθέτουμε. Οι αλγόριθμοι που επιλέχθηκαν και υλοποιήθηκαν σε βάθος είναι οι Q-Learning, Deep Q-Learning και NEAT.

Για κάθε έναν από τους παραπάνω αλγορίθμους, εκτελέστηκε μια σειρά πειραμάτων αυξημένης πολυπλοκότητας για την διεξαγωγή συμπερασμάτων ως προς την σύγκριση τους σε κάθε φάση της εκπαίδευσης. Οι φάσεις των πειραμάτων είναι οι εξής:

Φάσεις πειραμάτων:

- Static: Εκπαίδευση της μπλε ομάδας να πιάσει την κόκκινη, η οποία αλλάζει τυχαίες θέσεις και μένει ακίνητη σε κάθε παιχνίδι.
- Random Movement: Εκπαίδευση της μπλε ομάδας να πιάσει την κόκκινη. Η κόκκινη ομάδα κάνει τυχαία βήματα σε κάθε step του επεισοδίου.
- 2Players: Ταυτόχρονη εκπαίδευση των δυο ομάδων

4.1.1 Είσοδοι – Έξοδοι

Η μορφοποίηση των χώρων εισόδου και εξόδου σε κάθε αλγόριθμο υπόκειται σε μια προεπεξεργασία. Αρχικά, ο χώρος εισόδου εκφράζεται με τις θέσεις των δύο ομάδων μετά το απαραίτητο preprocessing. Στην φάση των πειραμάτων η είσοδος δινόταν κωδικοποιημένη, είτε σε one-hot encoding είτε normalized στο 1 οι τιμές των συντεταγμένων για την κάθε ομάδα. Μετά την φάση των πειραμάτων δεν παρουσιάστηκε ιδιαίτερη αλλαγή σε σχέση με τις κωδικοποιήσεις, οπότε η είσοδος στο σύστημα δόθηκε ως έχει. Αναλυτικότερα, οι καταστάσεις του agent και οι ενέργειες που εκτελεί ανά βήμα εκφράζεται ως εξής:

Είσοδος:

- CHASER observation = [x_chaser, y_chaser, x_runner, y_runner]
- RUNNER observation = [x_runner, y_runner, x_chaser, y_chaser]

Έξοδος:

- CHASER action = ["N", "S", "E", "W"]
- RUNNER action = ["N", "S", "E", "W"]

Στις πρώτες φάσεις των πειραμάτων χρησιμοποιήθηκαν μόνο η είσοδος και η έξοδος της μπλέ ομάδας (Chaser). Στα πειράματα εκπαίδευσης με τις δύο ομάδες agents, αρχικός σκοπός ήταν να εκπαιδευτούν και οι δύο ομάδες ταυτόχρονα και αργότερα να εκπαιδευτούν διαδοχικά.

Μια πρόκληση που χρειάστηκε να αντιμετωπιστεί είναι η συμπεριφορά των agents απέναντι στα εμπόδια. Μια σκέψη σχετικά με την αναγνώριση και την αντιμετώπιση των εμποδίων είναι η προσθήκη των θέσεων των εμποδίων στον πίνακα εισόδου, όσο αφορά τα εμπόδια που καταλαμβάνουν ένα ολόκληρο κελί. Και οι δύο ομάδες χρειαζόταν παραπάνω input features για να μπορούν να αλληλεπιδράσουν μαζί τους, μεγαλώνοντας έτσι το χώρο και τον χρόνο εκπαίδευσης.

Για τις ανάγκες λοιπόν των παραπάνω προϋποθέσεων, το observation state και το action state άλλαξαν. Αυτό επηρέασε κατά πολύ την πολυπλοκότητα των εκάστοτε αλγορίθμων, ιδιαίτερα του Q καθώς μεγενθύνεται κατά πολύ και το μέγεθος του πίνακα.

4.1.2 Rewards

Σημαντική παράμετρος για την διεξαγωγή των πειραμάτων είναι η διατήρηση της reward function για κάθε φάση του προβλήματος. Το reward το οποίο δίνεται στο σύστημα για κάθε ενέργεια του και επόμενη κατάσταση του είναι το ίδιο οριζόντια σε κάθε διαφορετική μεθοδολογία.

Η μέθοδος reward_logic() εκτελούταν σε κάθε ένα βήμα της εκπαίδευσης και σε όλες τις φάσεις των πειραμάτων η λογική ήταν αρκετά απλή καθώς δινόταν reward=100 στην μπλε ομάδα εάν έπιανε την κόκκινη, reward=-1 για κάθε έγκυρο βήμα και reward=-5 για κάθε άκυρο βήμα (βήμα προς τοίχο ή εμπόδιο που οδηγεί σε μη αλλαγή της καταστασής του).

Η διαφοροποίηση του αρνητικού reward ως προς την εγκυρότητα του βήματος έγινε για το λόγο πως αρκετές φορές παρατηρήθηκαν μη έγκυρες κινήσεις του agent, με αποτέλεσμα να καθυστερεί το exploration του χώρου και ταυτόχρονα μεγάλωνε και ο χρόνος εκπαίδευσης. Επίσης, η κινήσεις αυτές θεωρήθηκαν άσκοπες και απαραίτητες για την αποφυγή τους.

Αντίστοιχα, στην εκπαίδευση της κόκκινης ομάδας, έπαιρνε reward=10 για κάθε έγκυρο βήμα, reward=2 για κάθε άκυρο βήμα και reward=-100 κάθε φορά που πιανόταν από την μπλε ομάδα.

4.2 Διεξαγωγή Πειραμάτων με Q-Learning

Ως πρώτος αλγόριθμος εκμάθησης επιλέχθηκε ο Q-Learning. Ένας από τους βασικούς λόγους επιλογής του ήταν ο πεπερασμένος και σχετικά μικρός, ανάλογα και το πείραμα, χώρος του περιβάλλοντος. Κατα συνέπεια, ο πίνακας των Q-values που θα δημιουργηθεί θα είναι μικρός, επομένως μειώνεται και ο χρόνος εκπαίδευσης.

Επομένως, στην πρώτη φάση των πειραμάτων και για έναν 5x5 χώρο, ο πίνακας Q που αρχικοποιείται είναι διαστάσεων 625x4.

4.2.1 Υπερπαράμετροι

Παρακάτω παρουσιάζονται αναλυτικά οι υπερπαράμετροι οι οποίοι χρησιμοποιήθηκαν για όλες τις φάσεις των πειραμάτων:

Αριθμός Επεισοδίων – (NUM_EPISODES)	30000
Μέγιστος Αριθμός Βημάτων ανά Επεισόδιο – (MAX_T)	maze_size * 4
Αριθμός Βημάτων για Επιτυχία (Streak) – (SOLVED_T)	(maze_size * 4) / 2
Αριθμός Επιτυχιών (Streaks) για Τέλος Εκπαίδευσης – (STREAK_TO_END)	100
Ρυθμός Εκπαίδευσης (learning rate)	0.2
Ρυθμός Εξερεύνησης (explore rate)	0.001
Discount Factor	0.99

Πίνακας 1: Υπερπαράμετροι μεθόδου Q

Η επιλογή των υπερπαραμέτρων έγινε μετά από πολλές δοκιμές και εναλλαγές, ιδιαίτερα μεταξύ MAX_T και SOLVED_T. Πιο αναλυτικά, σε κάθε ένα επεισόδιο, ο agent έχει να κάνει MAX_T βήματα πριν τελειώσει το επεισόδιο, και SOLVED_T βήματα ώστε το επεισόδιο να θεωρηθεί επιτυχές. Αυτός είναι ένας τρόπος αντίληψης ότι το σύστημα είναι σωστά εκπαιδευμένο, καθώς εάν α agent επιτύχει 100 σερί επεισόδια, βρεί δηλαδή τον στόχο πριν τα SOLVED_T βήματα, τότε η εκπαίδευση σταματά και ο πίνακας Q θεωρείται σωστά ανανεωμένος.

4.2.2 Βασική δομή αλγορίθμου

Η βασική δομή του αλγορίθμου είναι η ακόλουθη:

```
For episode in NUM_EPISODES:
        env.reset()
        For t in MAX_T:
                action ← select action from Q table
                observation, reward, done = env.step(action)
                Update Q table
                If STREAK_TO_END:
                        Save Q table and end training
```

Η παραπάνω λογική υλοποιήθηκε στις δύο πρώτες φάσεις των πειραμάτων.

Στα πειράματα που ενεργούσαν και οι δύο ομάδες (2Players), η ίδια λογική υλοποιήθηκε σε κάθε step για κάθε έναν από τους agents. Κατά τη διάρκεια της εκπαίδευσης, όποιος agent κατάφερε να επιτύχει το απαραίτητο σερί επιτυχιών, αποθηκευόταν ο πίνακάς του και η εκπαίδευση συνεχιζόταν. Τελικά, δημιουργήθηκαν δύο πίνακες Q οι οποίοι στην φάση του testing ενεργούσαν εναλλάξ.

4.3 Διεξαγωγή Πειραμάτων με Deep Q-Learning

Κατά τη διάρκεια των πειραμάτων με τον Q-Learning, παρατηρήθηκαν πολλές αδυναμίες στην εκπαίδευση, κυρίως λόγω της εκθετικής αύξησης του χρόνου αλλά και του μεγέθους του πίνακα Q όσο η πολυπλοκότητα των πειραμάτων αυξανόταν. Η λύση σε αυτές τις αδυναμίες ήταν η μετατροπή του Q σε Deep Q. Η αντικατάσταση του πίνακα με ένα Dense νευρωνικό δίκτυο μείωσε κατα πολύ τον χώρο μάθησης και το πλήθος των βαρών προς εκπαίδευση. Για την διεξαγωγή των πειραμάτων, ο κώδικας εκπαίδευσης αναπτύχθηκε με χρήση της βιβλιοθήκης rl.agents της keras.

4.3.1 Υπερπαράμετροι

Παρακάτω παρουσιάζονται αναλυτικά οι υπερπαράμετροι οι οποίοι χρησιμοποιήθηκαν για όλες τις φάσεις των πειραμάτων:

Αριθμός Επεισοδίων – (num_episodes)	100-200
Μέγιστος Αριθμός Βημάτων ανά Επεισόδιο – (num_steps)	maze_size * 3
Warmup Steps	500-5000
Replay Memory	50000-200000
Ρυθμός Εκπαίδευσης (learning rate)	0.0001
Epsilon	0.1
Gamma	0.95
Loss Function	mse

Πίνακας 2: Υπερπαράμετροι μεθόδου Deep Q

4.3.2 Βασική Δομή Αλγορίθμου

Η βασική λογική που υλοποιήθηκε είναι η ακόλουθη:

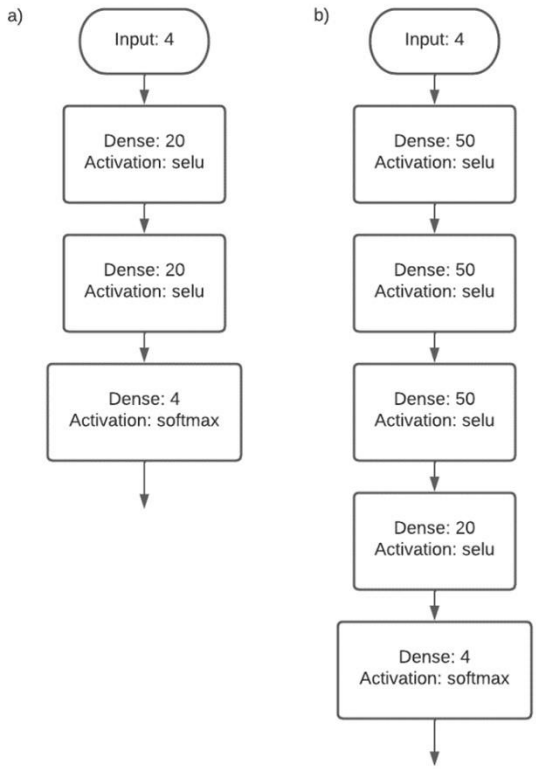
```
Initialize replay_memory, Q model weights
For episode in num_episodes:
        env.reset()
        For t in num_steps:
                If epsilon: action ← random, else: action ← select action from model
                observation, reward, done = env.step(action)
                Store in replay memory
                Sample random mini-batch from replay memory
                Gradient Descent to update weights
        Copy Prediction Network to Target Network
```

Όσο αφορα τη φάση των πειραμάτων 2Players, δημιουργήθηκαν δυο υποστάσεις της παραπάνω υλοποίησης οι οποίες εκπαιδεύτηκαν σε διαφορετικούς χρόνους. Πρώτα έγινε η εκπαίδευση της μπλε ομάδας και ακολούθως της κόκκινης, με την μπλε να κινείται βάση του μοντέλου που εκπαιδεύτηκε προηγουμένως. Η μάθηση δεν έγινε παράλληλα όπως αναφέρθηκε και στην Q μεθοδολογία. Ο λογος για τον οποίο επιλέχθηκαν να γίνουν μεμονωμένα οι εκπαιδεύσεις είναι το προφανές προβάδισμα της κόκκινης ομάδας κατά τη διάρκεια της ταυτόχρονης εκπαίδευσης, με αποτέλεσμα η μπλε ομάδα να μην καταφέρνει να γενικεύσει σωστά.

4.3.3 Δομές δικτύων

Οι βασικές διαφορές μεταξύ των στατικών δομών των δικτύων δεν επηρεάστηκαν από το μέγεθος του χώρου. Κατά κύριο λόγο η πολυπλοκότητα του δικτύου επηρεαζόταν άμεσα με τον στόχο του προβλήματος, κάτι το οποίο βοήθησε στην μείωση της ποσότητας των δικτύων, ανάλογα το πείραμα προς διεκπεραίωση.

Αναλυτικότερα, παρακάτω παρουσιάζονται ενδεικτικά οι δομές των δικτύων για την μπλε ομάδα.



Εικόνα 10: Δομές Νευρωνικών δικτύων μπλε ομάδας a) Static, b)Random + 2Players

Ένα σημαντικό πρόβλημα που αναδεικνύεται με το πέρας των πειραμάτων είναι η δυσκολία εύρεσης των κατάλληλων δικτύων αλλά και αναγκή επανατροποποίησής τους μέχρι την λήψη των βέλτιστων αποτελεσμάτων, σε συνδιασμό βέβαια με την επιλογή των κατάλληλων υπερπαραμέτρων.

4.4 Διεξαγωγή Πειραμάτων με NEAT

Η τελευταία μεθοδολογία που υλοποιήθηκε ήταν ο αλγόριθμος NEAT. Θεωρητικά, ο NEAT αποτελεί τον περισσότερα υποσχόμενο αλγόριθμο μάθησης από τους τρεις. Αυτό συμβαίνει γιατί υπερτερεί σε ταχύτητα και όγκο από τον Q αλλά επίσης επιτρέπει μέσω της νευροεξέλιξης την ανάπτυξη της βέλτιστης, ή κοντά στη βέλτιστη, τοπολογίας νευρωνικού δικτύου. Ο δεύτερος λόγος αποτελεί και ένα μεγάλο πλεονέκτημα σε σχέση με τον Deep Q, που η βασική σύσταση του δικτύου του αφορά στατική δομή η οποία βρέθηκε μετά από πολλές δοκιμές και πειράματα διαφορετικών τοπολογιών. Η ανάπτυξη του κώδικα και του αλγορίθμου μάθησης έγινε χρησιμοποιώντας την βιβλιοθήκη neat-python, σχεδιασμένη από τους CodeReclaimers.

4.4.1 Υπερπαράμετροι

Στα πλαίσια του NEAT έγινε η χρήση των παρακάτω υπερπαραμέτρων για όλες τις φάσεις των πειραμάτων.

Αριθμός Γενιών – (generations)	<10000
Αριθμός Δικτύων ανά Επεισόδιο – (runs_per_net)	maze_size * 3
Fitness Threshold	Μπλε:95, Κόκκινη:220
Replay Memory	50000-200000
Ρυθμός Εκπαίδευσης (learning rate)	0.0001
Epsilon	0.1
Gamma	0.95
Loss Function	mse

Πίνακας 3: Υπερπαράμετροι μεθόδου NEAT

4.4.2 Βασική Δομή Αλγορίθμου

Ο βασικός αλγόριθμος που υλοποιείται μέσω της βιβλιοθήκης neat-python είναι ο ακόλουθος.

```
Initialize population, fitness_function
Initialize genome from inputs,outputs
Initialize feedforward_net
For runs in runs_per_net:
        env.reset()
        observation, reward, done = env.step(action)
        next_action = net.activate(observation)
        calculate fitness
        evolve population through Crossover, Mutation
        divide population into species
```

Όπως και στις προηγούμενες υλοποιήσεις, στα πειράματα 2Players, οι εκπαιδεύσεις έγιναν και ξεχωριστά και ταυτόχρονα. Τελικά, επιλέχθηκε η ξεχωριστή εκπαίδευση για τους ίδιους λόγους με την υλοποίηση του Deep Q.

Ιδιαίτερη σημασία κατά την υλοποίηση πρέπει να δοθεί και στο configuration file του NEAT. Το configuration file περιέχει όλη τη σημαντική πληροφορία για την εκπαίδευση των agent.

Παρακάτω γίνεται μια παρουσίαση των σημαντικότερων παραμέτρων του configuration και η σύντομη περιγραφή τους.

NEAT		
fitness_criterion	mean	Κατά την εξέλιξη, η fitness_fuction υπολογίζεται από το μέσο όρο των fitnesses ανα generation
fitness_threshold	95	Ο στόχος της fitness_function κατά την εξέλιξη
pop_size	200	Πληθυσμός genome ανα population
reset_on_extinction	True	Reset του αλγορίθμου στη περίπτωση εξαφάνισης όλων των ειδών

DefaultStagnation		
species_fitness_func	max	Μέθοδος υπολογισμού fitness στα είδη
max_stagnation	20	Μέγιστος αριθμός γενιών για ένα είδος να εξαφανιστεί εφόσον δεν βελτιωθεί το fitness του
species_elitism	2	Πλήθος προστατευόμενων ειδών

DefaultSpeciesSet		
compatibility_threshold	3	Εκτίμηση ίδιων ειδών βάση της genomic distance με όριο την τιμή compatibility_threshold

DefaultReproduction		
elitism	1	Προστατευόμενα genome απο γενιά σε γενιά
survival_threshold	0.2	Ποσοστό αναπαραγωγής ειδών

DefaultGenome		
num_inputs	4	Αριθμός εισόδων ανά δίκτυο
num_hidden	0	Αριθμός hidden node
num_outputs	4	Αριθμός εξόδων ανά δίκτυο
activation_default	clamped	Βασική activation function
activation_options	sigmoid	Διαθέσιμες activation function κατά το mutation
activation_mutate_rate	0.2	Ποσοστό εμφάνισης διαθέσιμων activation functions

Παραμετροποίηση Αρχικοποίησης Weights και Biases		
bias_init_mean	0	
bias_init_stdev	1	
bias_replace_rate	0.1	
bias_mutate_rate	0.7	
bias_mutate_power	0.5	
bias_max_value	30	
bias_min_value	-30	
weight_max_value	30	
weight_min_value	-30	
weight_init_mean	0	
weight_init_stdev	1	
weight_init_type	gaussian	
weight_mutate_rate	0.8	
weight_replace_rate	0.1	
weight_mutate_power	0.5	
aggregation_default	sum	
aggregation_options	sum	
aggregation_mutate_rate	0.01	
compatibility_disjoint_coefficient	1	
compatibility_weight_coefficient	0.5	

Ρυθμίσεις νέων συνδέσεων και nodes		
conn_add_prob	0.5	Πιθανότητα προσθήκης σύνδεσης
conn_delete_prob	0.5	Πιθανότητα διαγραφής σύνδεσης
node_add_prob	0.2	Πιθανότητα προσθήκης node
node_delete_prob	0.2	Πιθανότητα διαγραφής node
enabled_default	True	Ενεργοποίηση νέων συνδέσεων
enabled_mutate_rate	0.01	Πιθανότητα mutation ενεργοποίησης ή απενεργοποίησης σύνδεσης

Παραμετροποίηση αρχικών εξόδων από το δίκτυο		
response_init_mean	1	
response_init_stdev	0	
response_init_type	gaussian	
response_replace_rate	0	
response_mutate_rate	0.01	
response_mutate_power	0	
response_max_value	30	
response_min_value	-30	

Τρόποι σύνδεσης		
initial_connection	full	Τρόπος σύνδεσης nodes
feed_forward	True	Τρόπος εμπροσθοδιάδοσης

Πίνακας 4: Παραμετροποίηση config file

5. Αποτελέσματα – Συμπεράσματα

5.1 Αποτελέσματα Q

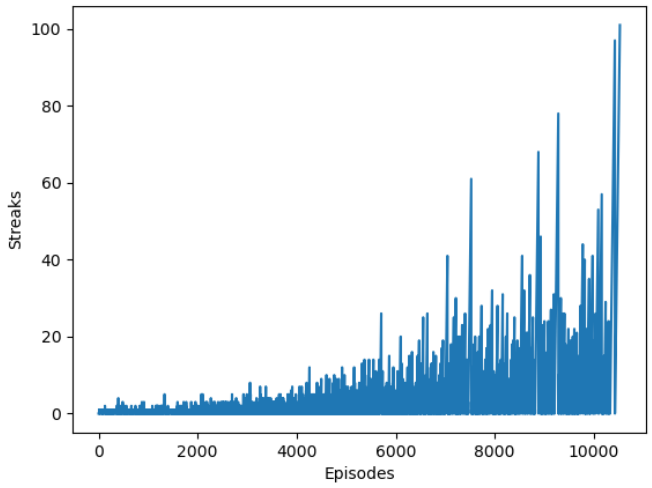
Το πρώτο συμπέρασμα που προέκυψε από την διεκπεραίωση όλων των πειραμάτων ήταν η αναποτελεσματικότητα του απλού Q σε πολύπλοκα προβλήματα. Παρότι ο χρόνος εκπαίδευσης του σε μικρής πολυπλοκότητας προβλήματα ήταν αρκετά μικρός, όσο πιο πολύπλοκο γινόταν το πρόβλημα τόσο δυσκολευόταν να γενικεύσει ως προς την λύση του προβλήματος.

Αυτό γίνεται αρκετά φανερό και στον παρακάτω πίνακα. Οι χρόνοι στα προβλήματα 2 ταυτόχρονων εκπαιδεύσεων εκφράζουν το πέρας της πρώτης αποθήκευσης της πρώτης επιτυχούς εκπαίδευσης.

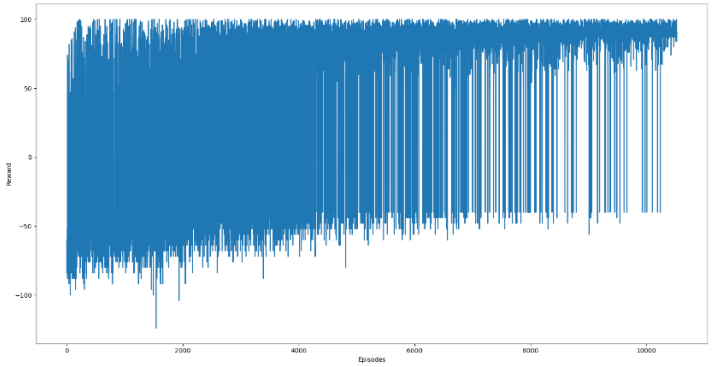
Map	Time (hh:mm:ss)	Episodes		Problem
5x5_empty	00:00:41	226	100	Static
	00:08:23	935	95	Random
	00:45:23	14586	80/20	2Players (Chaser/Runner)
10x10_empty	00:28:38	3532	85	Static
	07:12:01	25454	75	Random
	-	-	-	2Players
10x10_obst	00:31:15	3951	85	Static
	08:18:45	25810	70	Random
	-	-	-	2Players

Πίνακας 5: Αποτελέσματα Πειραμάτων Q

Στο τέλος κάθε εκτέλεσης, ο αλγόριθμος έτρεχε 20 test επεισόδια για τον υπολογισμό του accuracy. Ένα επεισόδιο θεωρείται ολοκληρωμένο επιτυχώς εάν ο agent πραγματοποιήσει το στόχο του σε συγκεκριμένο αριθμό βημάτων. Ο αριθμός αυτός διαφέρει ανάμεσα στους agents, καθώς ο καθένας ενεργεί βάση του δικού του reward. Συγκεκριμένα, η μπλε ομάδα ολοκληρώνει επιτυχώς εάν επιτύχει το στόχο του σε λιγότερο από (maze_size*2) βήματα, ενώ η κόκκινη εάν δεν πιαστεί από την μπλέ για πάνω από (maze_size*maze_size) βήματα.



Εικόνα 11: Αριθμός συνεχόμενων επιτυχημένων προσπαθειών (Streaks) ανά επεισόδιο



Εικόνα 12: Reward ανα επεισόδιο

Σε μικρούς χώρους όπως ο 5x5, είχε αρκετά καλά αποτελέσματα. Όσο ο χώρος μεγάλωνε, αυξανόταν εκθετικά και ο χρόνος εκπαίδευσης. Με την αύξηση της πολυπλοκότητας και του μεγάθους του πίνακα, ο αλγόριθμος δεν κατάφερνε να γενικεύσει στην λύση του προβλήματος.

Για παράδειγμα, στον χώρο 10x10 χωρις εμπόδια για το πείραμα 2Players, το μέγιστο observation state για τον κάθε agent είναι [10, 10, 10, 10] και το action [4]. Καταλήγουμε λοιπόν σε εκπαίδευση δυο πινάκων διαστάσεων 10000x4. Στο συγκεκριμένο παράδειγμα ο αλγόριθμος δεν κατάφερε να βρεί λύση μέχρι και τα 30000 επεισόδια. Αυτό μας οδηγεί στο συμπέρασμα ότι ακόμα και να αυξήσουμε τον αριθμό των επεισοδίων και να επιτύχουμε την επιθυμητή λύση, δύσκολα θα καταφέρει να γενικεύσει σε πειράματα αυξημένης πολυπλοκότητας ή μεγαλύτερων χώρων.

Επιπλέον, για την καταγραφή σχετικών πειραμάτων, απαιτείται συστήματα αυξημένης υπολογιστικής ισχύος καθώς η εκπαίδευση ενός τέτοιου agent μπορεί να διαρκέσει ακόμα και μέρες.

Καταλήγοντας, ο Q είναι ένας γρήγορος και αποτελεσματικός αλγόριθμος για την επίλυση απλών δομημένων προβλημάτων. Με την άνοδο της πολυπλοκότητας όμως, η εύρεση άλλων μεθοδολογιών είναι απαραίτητη.

5.2 Αποτελέσματα Deep Q

Σχετικά με τον Deep Q, τα πειράματα έδειξαν σίγουρα καλύτερη εικόνα και δικαιώνει την επιλογή του ως μια βελτιωμένη εκδοχή του απλού Q-Learning αλγορίθμου. Συγκεκριμένα, με την άνοδο της πολυπλοκότητας, τα πειράματα έτρεχαν σε εύλογο χρόνο και συγκριτικά πολύ μικρότερο από τον Q. Σημαντικός παράγοντας αποτέλεσε το γεγονός ότι δεν γινόταν εκπαίδευση σε πίνακα αλλά σε δίκτυο. Το πλήθος των weights των είναι πληθυσμιακά λιγότερα από την έκταση ενός Q πίνακα, άρα καθιστά σαφέστατα πιο έμπιστη γενίκευση στο testing.

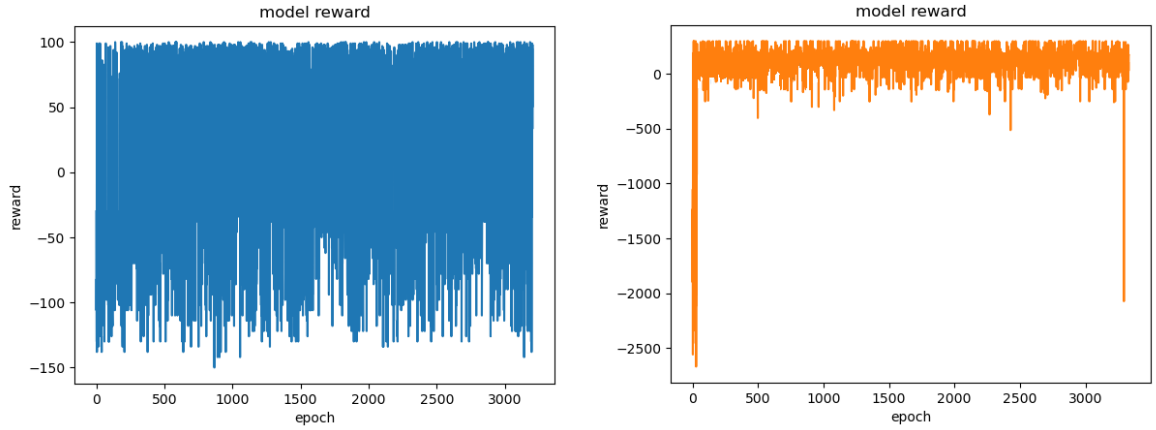
Ενδεικτικά, παρουσιάζονται τα αποτελέσματα των πειραμάτων, καθώς και οι χρόνοι εκπαίδευσης τους και το μέγιστο reward ανά επεισόδιο.

Map	Time (hh:mm:ss)	Episodes	Max Episode Reward	Problem
5x5_empty	00:02:34	7	98	Static
	00:18:56	13	89	Random
	00:28:21	12	90	2Players Chaser
	00:28:21	17	212	2Players Runner
10x10_empty	00:07:15	10	95	Static
	00:27:55	26	86	Random
	00:45:16	14	89	2Players Chaser
	00:45:16	89	157	2Players Runner
10x10_obst	00:15:10	20	94	Static
	00:58:13	29	76	Random
	02:56:17	26	79	2Players Chaser
	02:56:17	54	221	2Players Runner

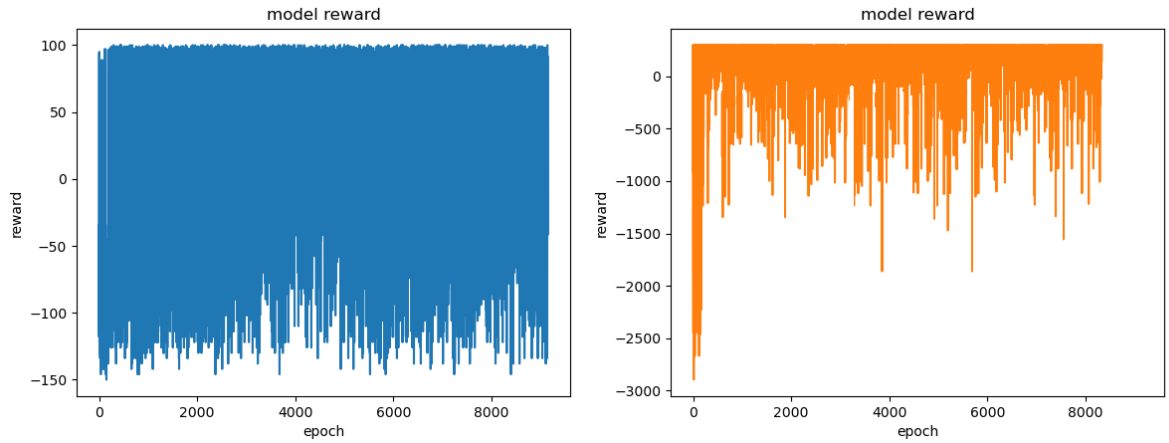
Πίνακας 6: Αποτελέσματα πειραμάτων Deep Q

Στον παραπάνω πίνακα, το Max Episode Reward εκφράζει το μέγιστο reward που επιτυγχάνει ο μέσος agent ανα επεισόδιο. Σύμφωνα με τα rewards που έχουν δοθεί, το μεγιστο πιθανό reward για την μπλε ομάδα είναι το 100, ενώ για την κόκκινη ομάδα είναι το 250.

Στα πειράματα παρατηρήθηκε μια μικρή υπεροχή του Chaser, της μπλε ομάδας. Αυτό ίσως οφείλεται στην χρήση του ίδιας δομής δικτύου και για τους δυο agents. Το έργο της κόκκινης ομάδας φέρει μεγαλύτερη πολυπλοκότητα καθώς τίθεται να αντιμετωπίσει ένα ήδη εκπαιδευμένο σύστημα, άρα ένα πιο σύνθετο μοντέλο θα είχε καλύτερα αποτελέσματα.



Εικόνα 13: Model Reward Chaser - Runner κατά την εκπαίδευση σε 10x10 χώρο



Εικόνα 14: Model Reward Chaser - Runner κατά την εκπαίδευση σε 10x10 χώρο με εμπόδια

Επιπροσθέτως, στις παραπάνω γραφικές φαίνεται το mean reward ανα epoch, το οποίο εκφράζει steps μέχρι την επίτευξη του στόχου ή την υπέρβαση του ορίου βημάτων. Παρατηρείται ότι τα μοντέλα στην πραγματικότητα γενικεύουν αρκετά γρηγορότερα σε σχέση με τις εποχές τις οποίες έτρεξαν. Αυτό μπορεί να οφείλεται και στην τυχαιότητα των παραδειγμάτων. Μια δικλείδα ασφαλείας για την εύρεση του βέλτιστου μοντέλου είναι ενός callback implementation, μιας διεργασίας δηλαδή που έχει ενσωματωθεί στην fit συνάρτηση της εκπαίδευσης και αποθηκεύει το καλύτερο μοντέλο μετά από κάθε επεισόδιο μέχρι εκείνη τη στιγμή.

5.3 Αποτελέσματα NEAT

Συγκριτικά με τους προηγούμενους αλγορίθμους, ο NEAT παρουσίασε πιο ικανοποιητικά αποτελέσματα. Η εύρεση της κατάλληλης δομής δικτύου αποτέλεσε σίγουρα προβάδισμα έναντι του Deep Q που εκπαιδεύτηκε σε σταθερή δομή. Συνολικά πάντως, ο χρόνος εκπαίδευσης ήταν φανερά μεγαλύτερος σε σχέση με τους υπόλοιπους, όχι όμως απαγορευτικός.

Στον παρακάτω πίνακα παρουσιάζονται τα αποτελέσματα των πειραμάτων.

Map	Time (hh:mm:ss)	Generations	Fitness Result	Problem
5x5_empty	00:00:09	10	98.2	Static
	00:08:59	752	93.7	Random
	00:14:45	761	92.7	2Players Chaser
	00:14:45	520	215	2Players Runner
10x10_empty	00:11:02	851	94.2	Static
	01:28:56	1121	87.6	Random
	02:10:12	1242	86.2	2Players Chaser
	02:10:12	810	180	2Players Runner
10x10_obst	00:32:04	2584	85	Static
	03:45:13	3650	82.4	Random
	04:56:30	3820	81.2	2Players Chaser
	04:56:30	2110	204	2Players Runner

Πίνακας 7: Αποτελέσματα NEAT

Τα παραπάνω αποτελέσματα επιτεύχθηκαν μετά από πολλές εναλλαγές υπερπαραμέτρων και δοκιμών. Σημειώνεται πως για το Fitness Result της μπλε ομάδας, για κάθε πρόβλημα δηλαδή εκτός του '2Players Runner', το μέγιστο fitness που μπορεί να επιτευχθεί είναι το 100. Αντίστοιχα, για την κόκκινη ομάδα και το '2Players Runner', το μέγιστο fitness είναι το 250 καθώς παίρνει 10 reward για κάθε έγκυρη κίνηση σε μέγιστο αριθμό κινησεων 25.

Παρατηρείται πως όσο δυσκολεύει ο χώρος, το fitness του chaser μειώνεται. Αντιθέτως, ο runner έχει αρκετά καλά αποτελέσματα, κάτι το οποίο είναι προφανές καθώς ο chaser δεν έχει εκπαιδευτεί επαρκώς ώστε να πιάνει σε κάθε επεισόδιο τον runner. Επιπλέον, ο runner φαίνεται να έχει ένα προβάδισμα με την προσθήκη των εμποδίων.

Κατά τη διάρκεια των πειραμάτων, παρατηρήθηκε ότι τα περισσότερα nodes από το δίκτυο που δημιουργείται έχουν sigmoid activation function. Έτσι στο config file του NEAT αυξήθηκε το ποσοστό επιλογής της σιγμοειδούς συνάρτησης σε σχέση με την clamped από 0.2 σε 0.5.

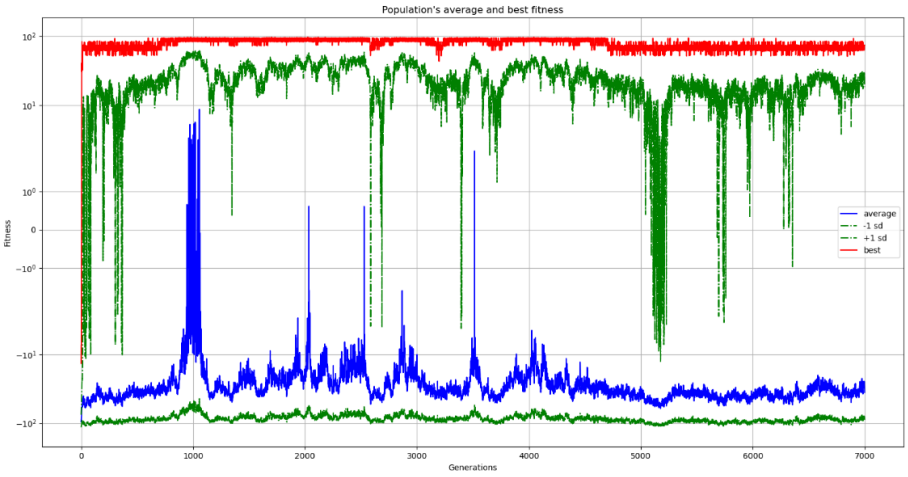
Η αποτελεσματικότητα του NEAT ιδιαίτερα στον chaser οδήγησε σε προσπάθεια βελτίωσης των αποτελεσμάτων και του runner, αναζητώντας τη βέλτιστη παραμετροποίηση. Μια πολύ σημαντική αλλαγή αποτέλεσε η διαφοροποίηση του τρόπου ανάθεσης του reward. Αναλυτικότερα, για τον runner, έγιναν δοκιμές με ανάθεση reward=-100 όταν πιάνεται από τον chaser και reward=4 για οποιαδήποτε άλλη κίνηση. Με αυτόν τον τρόπο δίνεται περισσότερη έμφαση στην αποφυγή του αντιπάλου ακόμα και αν η κίνηση που κάνει ο agent δεν είναι έγκυρη. Παράλληλα, το μέγιστο fitness result του runner αλλάζει από 250 σε 100, καθώς πλέον επιτυγχάνεται reward=4 για μέγιστο αριθμό βημάτων 25.

Map	Time (hh:mm:ss)	Generations	Fitness Result	Problem
10x10_obst	04:21:40	1081	89.5	2Players Chaser
	04:21:40	2412	85	2Players Runner

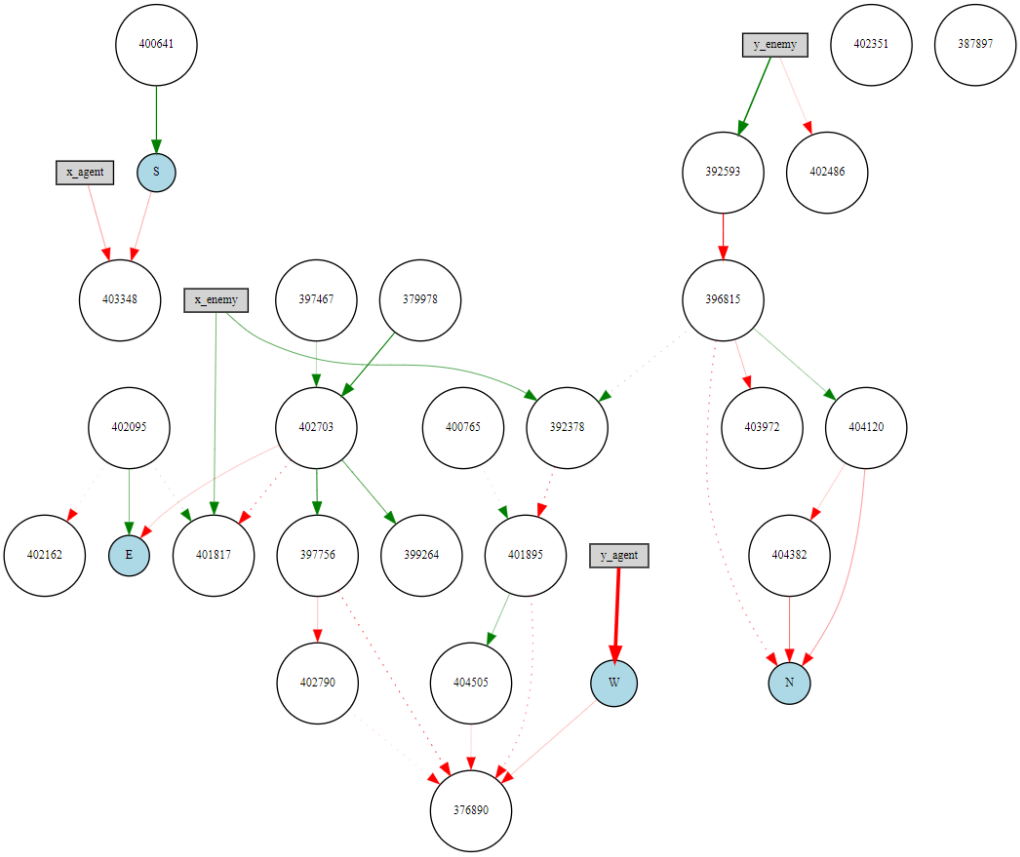
Πίνακας 8: Αποτελέσματα NEAT μετά την παραμετροποίηση

Έτσι, επιτεύχθηκε βελτίωση του fitness result κατά 8% στον chaser και 4% στον runner.

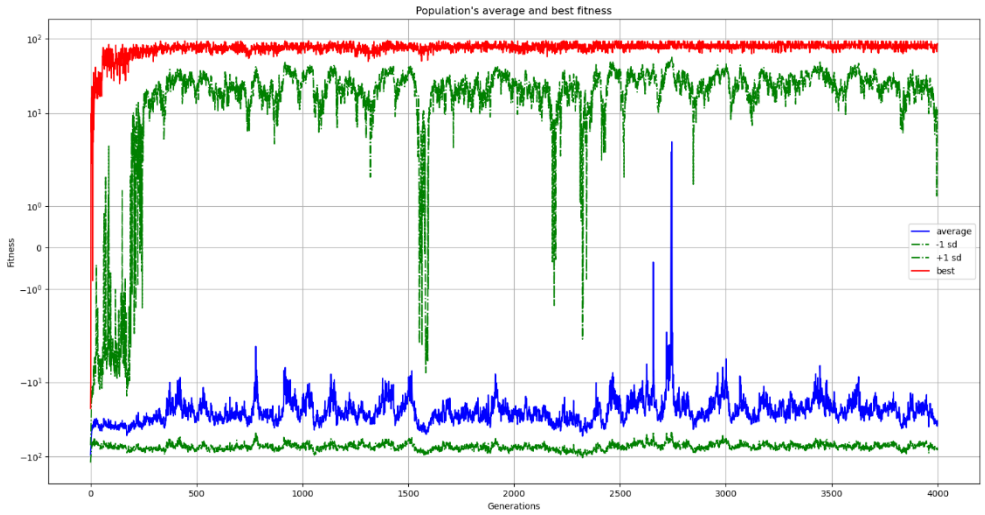
Παρακάτω παρουσιάζονται ενδεικτικά οι δομές των δικτύων για τα δυό τελευταία πειράματα, (2Players Chaser, 2Players Runner) καθώς και οι αντίστοιχες γραφικές παραστάσεις:



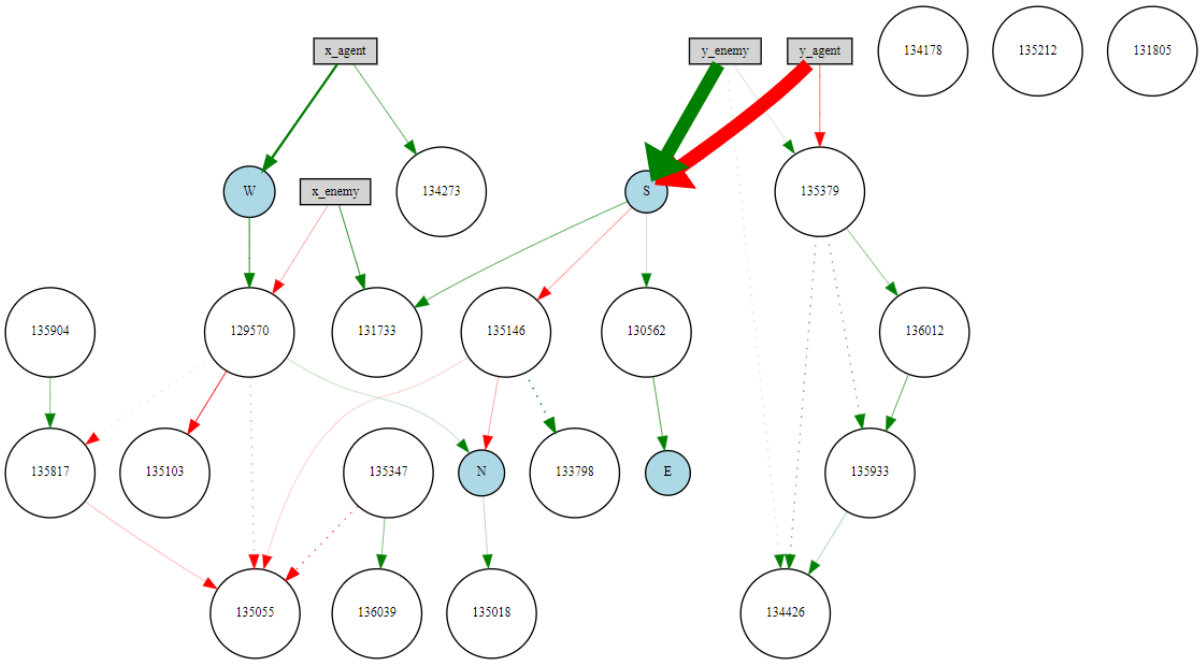
Εικόνα 15: Μέσο και μέγιστο reward κατά την εκπαίδευση του Chaser



Εικόνα 16: Δομή νευρωνικού δικτύου NEAT Chaser



Εικόνα 17: Μέσο και μέγιστο reward κατά την εκπαίδευση του Runner



Εικόνα 18: Δομή νευρωνικού δικτύου NEAT Runner

5.4 Συμπεράσματα και Μελλοντικές Βελτιώσεις

Η γενική εικόνα που παρουσίασαν τα πειράματα δείχνουν την υπεροχή του NEAT απέναντι στους υπόλοιπους αλγορίθμους σχετικά με το συγκεκριμένο πρόβλημα. Παρ' ολα αυτά, η χαμηλή επεξεργαστική ισχύ στην οποία δοκιμάστηκαν τα πειράματα, σε συνδιασμό με την συνθετότητα των πειραμάτων καθιστά δύσκολη την σύνθεση πολυπλοκότερων δικτύων τα οποία δημιουργούνται μέσω της νευροεξελικτικής ακολουθίας. Επομένως, υλοποιώντας τα πειράματα σε συστήματα μεγαλύτερης επεξεργαστικής δυνατότητας, θα οδηγούμασταν σε καλύτερα αποτελέσματα και συνεπώς σε μικρότερους χρόνους εκπαίδευσης, ακόμα και στα πειράματα υλοποίησης όπως του αλγορίθμου εκμάθησης Q.

Επιπλέον, στην πλειοψηφία των πειραμάτων υλοποιήθηκαν feedforward δίκτυα. Μια αλλαγή που θα μπορούσε να αποφέρει βελτιωμένα αποτελέσματα είναι τα convolutional νευρωνικά δίκτυα. Ειδικότερα, στην μεθοδολογία Deep Q-Learning, η αντικατάσταση των Dense δικτύων με Conv είναι μια μέθοδος που έχει δοκιμαστεί κατά καιρούς, με χαρακτηριστικό παράδειγμα τη δημιουργία ενός ολοκληρωμένου συστήματος reinforcement learning σε παιχνίδια Atari. Καθώς τα convolutional δίκτυα δέχονται 2d εισόδους, θα πρέπει να γίνει το απαραίτητο encoding στην είσοδο, μετατροπή τους δηλαδή σε ένα δισδιάστατο χώρο είτε σε μορφή εικόνας του περιβάλλοντος ανά διαφορετική κατάσταση (pixels), είτε να τροφοδοτείται το δίκτυο με μια απεικόνιση του χώρου ανα κελί (10x10 2d array με διαφορετική τιμή ανάλογα με τη κατάσταση του κάθε κελιού για τον 10x10 χώρο).

Σημαντική αλλαγή θα μπορούσε να επιφέρει και η αλλαγή του encoding για τις ήδη υπάρχουσες δομές δικτύων. Στην παρούσα έρευνα δοκιμάστηκαν διαφορετικά encodings για την εισοδο του observation στο δίκτυο. Ένα από αυτά είναι το one-hot encoding, το οποίο μετατρέπει την είσοδο του δικτύου σε ένα μονοδιάστατο πίνακα με 0 και σε ένα από αυτά έχουμε 1. Η κάθε μια από αυτές τις κωδικοποιήσεις είναι μοναδική. Τα συμπεράσματα από αυτή την κωδικοποίηση είναι η καθυστέρηση της γενίκευσης του στόχου, τα αποτελέσματα όμως ήταν αρκετά όμοια με αυτά που αναλύθηκαν και στην προηγούμενη ενότητα, θέτωντας τις κατάλληλες προοπτικές για βελτίωσή τους. Τίθεται λοιπόν το ζήτημα εύρεσης του κατάλληλου encoding που θα βελτιώσει τα παρόντα αποτελέσματα.

Η παρούσα εργασία ήταν μια προσέγγιση μάθησης χρησιμοποιώντας τρείς διαφορετικές αλγοριθμικές λογικές. Τα αποτελέσματα που αναδυκνύονται είναι επιθυμητά, η σχετική βιβλιογραφία όμως που αφορά παρόμοιες τεχνικές είναι πολύ μεγάλη. Σε μελλοντική έρευνα θα μπορούσαν να χρησιμοποιηθούν και άλλοι σχετικοί τρόποι εκμάθησης που μπορεί να αποφέρουν τα βέλτιστα αποτελέσματα.

Βιβλιογραφία

Andrew G.Barto, Richard S. Sutton. 2018. *Reinforcement Learning - An Introduction.*

Andrew Ng, Daishi Harada, Stuart Russell. χ.χ. «Policy invariance under reward transformations: Theory and application to reward shaping.»

Banzhaf, Wolfgang. 1998. «Genetic programming : an introduction on the automatic evolution of computer programs and its applications.»

Beakcheol Jang, Myeonghwi Kim, Gaspard Harerimana, Jong Wook Kim. 2019. «Q-Learning Algorithms: A Comprehensive.»

Bonate, Peter L. 2001. «A Brief Introduction to Monte Carlo Simulation.»

Bowen Baker, Ingmar Kanitscheider, Todor Markov, Yi Wu, Glenn Powell, Bob McGrew, Igor Mordatch. 2019. «Emergent Tool Use From Multi-Agent Autocurricula.»

Chan, Matthew. χ.χ. «gym-maze.»

Choudhary, Ankit. 2019. *A Hands-On Introduction to Deep Q-Learning using OpenAI Gym in Python.*

CodeReclaimers. 2015-2019. *NEAT Overview.*

Dimitri P. Bertsekas, John N. Tsitsiklis. 1996. «Neuro-Dynamic Programming.»

Foy, Peter. 2021. *Deep Reinforcement Learning: Guide to Deep Q-Learning.*

Kenneth O. Stanley, Risto Miikkulainen. χ.χ. «Evolving Neural Networks through.»

Kung-Hsiang, Huang (Steeve). 2018. *Introduction to Various Reinforcement Learning Algorithms. Part I (Q-Learning, SARSA, DQN, DDPG).*

Mahoney, Chris. χ.χ. *Reinforcement Learning.*

Miikkulainen, Kenneth O. Stanley and Risto. χ.χ. «Efficient Evolution of Neural Network Topologies.»

Nicholson, Chris. χ.χ. *A Beginner's Guide to Deep Reinforcement Learning.*

Schmidhuber, Jurgen. 2014. «Deep Learning in Neural Networks: An Overview.»

Scholz, Jan. 2019. «Genetic Algorithms and the Traveling Salesman Problem a historical Review.»

Sebastian Lang, Tobias Reggelin, Johann Schmidt, Marcel Müller. 2021. «NeuroEvolution of Augmenting Topologies for Solving a Two-Stage Hybrid Flow Shop Scheduling Problem: A Comparison of Different Solution Strategies.»

Simonini, Thomas. χ.χ. *Deep Q-Learning with Space Invaders.*

Szepesv´ari, Csaba. 2009. «Algorithms for Reinforcement Learning.»

Tensorflow. 2021. *Introduction to RL and Deep Q Networks.*

Vincent François-Lavet, Peter Henderson, Riashat Islam, Marc G. Bellemare, Joelle Pineau. 2018. «An Introduction to Deep Reinforcement Learning.»

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou. χ.χ. «Playing Atari with Deep Reinforcement Learning.»

Zaremba, Greg Brockman and Vicki Cheung and Ludwig Pettersson and Jonas Schneider and John Schulman and Jie Tang and Wojciech. 2016. «OpenAI Gym.»