Εθνικο Μετσοβιο Πολυτεχνειο
Σχολη Ηλεκτρολογων Μηχανικων και Μηχανικων Υπολογιστων
Τομεας Τεχνολογιας Πληροφορικης και Υπολογιστων

Χρήση βαθιάς ενισχυτικής μάθησης για την επίλυση του **Rubik's Cube**

Διπλωματικη Εργασια

ΣΤΥΛΙΑΝΟΣ ΑΝΔΡΕΑΣ ΣΤΑΥΡΟΥΛΑΚΗΣ

Επιβλέπων: Ανδρέας-Γεώργιος Σταφυλοπάτης
Καθηγητής ΕΜΠ

Αθήνα, Ιούλιος 2019

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

Χρήση βαθιάς ενισχυτικής μάθησης για την επίλυση του **Rubik's Cube**

Διπλωματικη Εργασια

ΣΤΥΛΙΑΝΟΣ ΑΝΔΡΕΑΣ ΣΤΑΥΡΟΥΛΑΚΗΣ

Επιβλέπων: Ανδρέας-Γεώργιος Σταφυλοπάτης
Καθηγητής ΕΜΠ

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 15η Ιουλίου 2019.

(Υπογραφή)	*(Υπογραφή)*	*(Υπογραφή)*
.....	.....	.....
Ανδρέας - Γεώργιος Σταφυλοπάτης	Γιώργος Στάμου	Νικόλαος Παπασπύρου
Καθηγητής ΕΜΠ	Αναπληρωτής Καθηγητής	Αναπληρωτής Καθηγητής

Αθήνα, Ιούλιος 2019

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

(Υπογραφή)

.....
Στυλιανός-Ανδρέας Σταυρουλάκης
Διπλωματούχος Ηλεκτρολόγος Μηχανικός και Μηχανικός Υπολογιστών Ε.Μ.Π.

Περίληψη

Ένας από τους κλάδους της μηχανικής μάθησης, ο οποίος παρουσιάζει μεγάλη πρόοδο τα τελευταία χρόνια, είναι η βαθιά ενισχυτική μάθηση. Προγράμματα που στοχεύουν στη μεγιστοποίηση κάποιας σωρευτικής ανταμοιβής αποδεικνύονται ελπιδοφόρα στην προσπάθεια να επεκταθεί το εύρος των επιλύσιμων προβλημάτων μεγάλης πολυπλοκότητας. Ωστόσο, το εκάστοτε περιβάλλον με το οποίο αλληλεπιδρά κάποιος πράκτορας βαθιάς ενισχυτικής μάθησης, ενδέχεται να περιέχει μια ευρύτερη ποικιλία πιθανών σημάτων ανταμοιβής κατά την εκπαίδευση.

Ένα παράδειγμα συνδυαστικού προβλήματος με αραιές ανταμοιβές αποτελεί και ο φημισμένος κύβος του Ρούμπικ. Η αντιμετώπιση των αραιών ανταμοιβών καθώς και το γεγονός ότι δεν υπάρχει εγγύηση τερματισμού του προβλήματος, είναι μια από τις μεγαλύτερες προκλήσεις στην βαθιά ενισχυτική μάθηση.

Στόχος της διπλωματικής εργασίας είναι η εισαγωγή της έννοιας της εντροπίας του κύβου του Ρούμπικ, και η μερική χρήση του σαν οδηγό από έναν πράκτορα, ο οποίος χρησιμοποιεί έναν αλγόριθμο αυτοδιδασκόμενης επανάληψης κατά την εκπαίδευση του, με απώτερο σκοπό την αποδοτικότερη λύση του κύβου.

Λέξεις Κλειδιά

Βαθιά Ενισχυτική Μάθηση, Αυτοδιδασκόμενη Επανάληψη, Ελαχιστοποίηση Εντροπίας, Αναζήτηση Δέντρων Μόντε Κάρλο, Νευρωνικά Δίκτυα, Κύβος Ρούμπικ

Abstract

One of the most promising types of Machine Learning called Deep Reinforcement Learning has advanced to the point where we're seeing computers do things that would have been considered science fiction just a few years ago. Deep Reinforcement Learning agents have achieved state-of-the-art results by directly minimizing cumulative reward. However, many environments contain a plethora of possible training signals.

In particular, for many combinatorial optimization environments (such as the famous Rubik's Cube), rewards are sparse and episodes are not guaranteed to terminate. Learning goal-directed behavior in environments with sparse feedback is a major challenge for reinforcement learning algorithms.

The goal of the thesis is to introduce the concept of cube entropy and let the agent take advantage of entropy minimization, while using an autodidactic iteration algorithm, in its pursuit of finding God's Numnber.

Keywords

Deep Reinforcement Learning, Auto-didactic Iteration, Entropy Minimization, Monte Carlo Tree Search, Neural Networks, Rubik's Cube

στους γονείς μου

Ευχαριστίες

Θα ήθελα καταρχήν να ευχαριστήσω τον καθηγητή κ. Ανδρέα - Γεώργιο Σταφυλοπάτη για την επίβλεψη αυτής της διπλωματικής εργασίας και για την ευκαιρία που μου έδωσε να πραγματοποιήσω την εκπόνηση αυτής στο Εργαστήριο Ευφυών Συστημάτων.

Εξαιρετικής σημασίας ήταν και η βοήθεια των υποψηφίων διδακτόρων Θάνου Τασάκου και Τάσου Παπαγιάννη, οι οποίοι μου έδωσαν χρήσιμες συμβουλές και κατευθύνσεις καθ' όλη τη διάρκεια μέχρι την περάτωσή της.

Τέλος, ευχαριστώ τους γονείς, τα αδέρφια και τους φίλους μου για όλη την υποστήριξη που μου παρείχαν κατά τη διάρκεια των σπουδών μου. Ιδιαίτερα ευχαριστώ θερμά τους Βασίλη Σταυρουλάκη, Αδάμ Ραπτάκη, Γιώργο Χαλκιόπουλο (ε είσαι φίλος) και Αντώνη Νικολαΐδη για την έμπρακτη συμπαράστασή τους όλα αυτά τα χρόνια.

Περιεχόμενα

Κατάλογος Σχημάτων

Κεφάλαιο 1

Εισαγωγή

Το ταίριασμα ενισχυτικής και βαθιάς μάθησης είναι γνωστή ως βαθιά ενισχυτική μάθηση (Deep Reinforcement Learning ή DRL. Η βαθιά ενισχυτική μάθηση στοχεύει στη δημιουργία ευφυών πρακτόρων οι οποίοι προσαρμόζονται σε κάποιο περιβάλλον μέσω ανάλυσης των εμπειριών τους κατά την αλληλεπίδραση τους μ΄ αυτό. Οι πράκτορες ενισχυτικής μάθησης έχουν ήδη γίνει καλύτεροι από τον άνθρωπο σε ένα μεγάλο εύρος παιχνιδιών [19]. Η πρόσφατη επιτυχία των πρακτόρων είναι προϊόν κατάλληλης συνεργασίας της κλασικής ενισχυτικής μάθησης, της βαθιάς μάθησης και της δενδρικής αναζήτησης Monte Carlo (MCTS)[6].

Παρόλο που οι αλγόριθμοι βαθιάς ενισχυτικής μάθησης έχουν δείξει εντυπωσιακά αποτελέσματα σε πολλούς διαφορετικούς τομείς, τα αποτελέσματα αυτά βασίζονται σε μεγάλο βαθμό στη δυνατότητα του πράκτορα να λαμβάνει περιοδικά ανταμοιβές από μία αρχικά τυχαία πολιτική.

Οι σύγχρονοι αλγόριθμοι DRL δυσκολεύονται σε περιβάλλοντα με μεγάλο αριθμό καταστάσεων και μικρό αριθμό ανταμοιβών όπως στα παιχνίδια Sokoban και Starcraft. Άλλα περιβάλλοντα, όπως προβλήματα εξετάσεων σύντομης απάντησης, παραγοντοποίηση πρώτων αριθμών, καθώς και ο κύβος του Ρούμπικ έχουν ένα πολύ μεγάλο χώρο καταστάσεων και μόνο μία τελική κατάσταση στην οποία υπάρχει αμοιβή.

Συγκεκριμένα, το σύνολο των διαφορετικών καταστάσεων στις οποίες μπορεί να βρεθεί ο κύβος του Ρούμπικ είναι:

$$S \approx 4 * 10^{19} \tag{1.1}$$

Ωστόσο, από όλες αυτές τις καταστάσεις, μία μόνο έχει το σήμα ανταμοιβής, η κατάσταση στην οποία ο κύβος θεωρείται λυμένος. Συνεπώς, ξεκινώντας από τυχαίες καταστάσεις και εφαρμόζοντας γνωστούς αλγορίθμους βαθιάς ενισχυτικής μάθησης που δουλεύουν καλά σε άλλα περιβάλλοντα (όπως ο αλγόριθμος A3C) ενδεχομένως να μην έλυναν ποτέ τον κύβο και, συνεπώς, να ήταν αδύνατον να λάβουν σήμα ανταμοιβής.

Για την αντιμετώπιση του προβλήματος της αραιής ανταμοιβής, χρησιμοποιείται η αυτοδιδασκούμενη επανάληψη [17] για τη δημιουργία αυθαίρετης ποσότητας δειγμάτων εκπαίδευσης.

Ο αλγόριθμος αυτός, εμπνευσμένος από τεχνικές αναζήτησης βέλτιστης πολιτικής, υπολογίζει ένα policy και value μέσω μίας επαναληπτικής διαδικασίας εποπτευόμενης μάθησης. Στη συνέχεια, γίνεται χρήση της αναζήτησης Monte Carlo για την αποτελεσματική επίλυση του κύβου. Ως επιπρόσθετη πληροφορία κατά την αναζήτηση, εισάγεται η έννοια της εντροπίας και δίνεται κίνητρο στον πράκτορα να κινηθεί προς χαμηλότερη εντροπία για την αποδοτικότερη λύση του κύβου.

1.1 Οργάνωση του τόμου

Η διπλωματική εργασία είναι οργανωμένη σε έξι κεφάλαια.

Στο Κεφάλαιο 2 δίνεται το θεωρητικό υπόβαθρο των βασικών τεχνικών που χρησιμοποιούνται στη διπλωματική αυτή. Περιγράφεται η θεωρία της ενισχυτικής μάθησης καθώς και η χρήση των νευρωνικών δικτύων στη βαθιά ενισχυτική μάθηση.

Στο Κεφάλαιο 3 γίνεται εισαγωγή των ιδιαιτεροτήτων του κύβου του Ρούμπικ και η ανάλυσή του από την σκοπιά της επιστήμης υπολογιστών. Περιγράφονται γνωστοί αλγόριθμοι επίλυσης καθώς και το πρόβλημα αναζήτησης του "αλγορίθμου του Θεού", ένα ανοιχτό ακόμη πρόβλημα.

Στο Κεφάλαιο 4 γίνεται η ανάλυση της πειραματικής διαδικασίας που χρησιμοποιήθηκε για την επίλυση του κύβου με μία νέα μέθοδος αυτοδιδασκόμενης επανάληψης. [17]

Στο Κεφάλαιο 5 εισάγεται η έννοια της εντροπίας του κύβου και παρουσιάζεται η ιδέα χρήσης της εντροπίας για εύρεση αποδοτικότερων λύσεων του κύβου.

Στο Κεφάλαιο 6 γίνεται η παρουσίαση των συμπερασμάτων και εξετάζονται κατευθύνσεις για περαιτέρω βελτίωση του αλγορίθμου.

Κεφάλαιο 2

Θεωρητικό Υπόβαθρο

Στο κεφάλαιο αυτό παρουσιάζεται αναλυτικά η θεωρία της βαθιάς ενισχυτικής μάθησης, η οποία αποτελεί και τον πυρήνα του αλγορίθμου αυτοδιδασκόμενης επανάληψης.

2.1 Εισαγωγή

Ένας από τους πιο ενδιαφέρον κλάδους της τεχνητής νοημοσύνης, "παντρεύει" τη δύναμη και τη δυνατότητα των βαθιών νευρωνικών δικτύων για αναπαράσταση και κατανόηση του περιβάλλοντος με τη δυνατότητα ενέργειας πάνω σε αυτήν την κατανόηση.

Η βαθιά ενισχυτική μάθηση χρησιμοποιεί τη δύναμη της βαθιάς μάθησης η οποία βρίσκεται στη δυνατότητα κωδικοποίησης της αναπαράστασης ενός περιβάλλοντος με έναν τρόπο που επιτρέπει την δυνατότητα λήψης αποφάσεων μέσω κατάλληλης δειγματοληψίας και επίβλεψης/προσαρμογής της συμπεριφοράς του δικτύου στο περιβάλλον.

2.2 Βαθιά Ενισχυτική Μάθηση

2.2.1 Ορισμός Βαθιάς Ενισχυτικής Μάθησης

Με μία πρόταση, μπορούμε να ορίσουμε το reinforcement learning ως μία χαρτογράφηση καταστάσεων σε αποφάσεις με κίνητρο την μεγιστοποίηση κάποιας αριθμητικής ανταμοιβής. Στη διαδικασία εκμάθησης δεν παρέχεται από την αρχή η πληροφορία και η γνώση για το ποιες αποφάσεις είναι προτιμότερες. Ο πράκτορας (όπως θα ορίσουμε και αργότερα) οφείλει να ανακαλύψει μόνος του μέσω αλληλεπίδρασης με το περιβάλλον ποιες είναι οι καλύτερες ενέργειες. Στις πιο ενδιαφέρουσες και δύσκολες περιπτώσεις, μάλιστα, οι ενέργειες αυτές ενδεχομένως να επηρεάζουν όχι μόνο την άμεση ανταμοιβή αλλά και ανταμοιβές στο μέλλον. Οι δύο αυτές έννοιες (αναζήτηση trial-and-error και η καθυστερημένη ανταμοιβή) χαρακτηρίζουν το reinforcement learning.

Ο φορμαλισμός του προβλήματος του reinforcement learning κληρονομεί ιδέες που υπάρχουν στη θεωρία δυναμικών συστημάτων. Πιο συγκεκριμένα, η μαθηματική θεωρία που χρησιμοποιείται κατά κόρον είναι ο βέλτιστος έλεγχος μιας μη-γνωστής (από την αρχή) διαδικασίας Markov, και αυτό γιατί οι διαδικασίες Markov συμπεριλαμβάνουν τις τρεις πιο σημαντικές πτυχές του reinforcement learning - αίσθηση, δράση, στόχος - στις απλούστερες δυνατές μορφές τους. Οποιαδήποτε μέθοδος κατάλληλη για την επίλυση προβλημάτων τέτοιας φύσης ονομάζουμε μέθοδο επίλυσης προβλημάτων reinforcement learning.

2.2.2 Στοιχεία Βαθιάς Ενισχυτικής Μάθησης

Πέρα από τον πράκτορα και το περιβάλλον, εντοπίζονται τέσσερα βασικά στοιχεία ενός συστήματος ενισχυτικής μάθησης [13].

- Πολιτική - Policy

  Η πολιτική ορίζει τον τρόπο εκμάθησης του πράκτορα οποιαδήποτε χρονική στιγμή. Σε γενικές γραμμές, η πολιτική είναι μία χαρτογράφηση καταστάσεων του περιβάλλοντος σε κατάλληλες ενέργειες που δρουν συναρτήσει αυτών των καταστάσεων. Αντιστοιχεί σε μία αλληλουχία ταιριασμένων ερεθισμάτων και δράσεων στα ερεθίσματα, όπως εμφανίζεται και στον κλάδο της ψυχολογίας.

  Σε ορισμένες περιπτώσεις, η πολιτική ενδέχεται να είναι μία απλή συνάρτηση ή πίνακας αναζήτησης, ενώ σε άλλες ενδεχομένως να είναι αρκετά πολύπλοκη με σημαντικές υπολογιστικές απαιτήσεις. Συμπερασματικά λοιπόν, η πολιτική είναι ο πυρήνας του πράκτορα RL με την έννοια ότι είναι αρκετή από μόνη της να ορίσει την συμπεριφορά του πράκτορα. Οι περισσότερες από τις πολιτικές σήμερα που παρουσιάζουν ερευνητικό ενδιαφέρον είναι στοχαστικές.

- Σήμα Επιβράβευσης/Ανταμοιβής - Reward Signal

  Το σήμα επιβράβευσης (κάποια αριθμητική τιμή / μονόμετρο μέγεθος) ορίζει το στόχο σε ένα πρόβλημα RL. Σε κάθε χρονική στιγμή, το περιβάλλον στέλνει στον πράκτορα μία ποσότητα ανταμοιβής για την εκάστοτε ενέργειά του. Έτσι ο πράκτορας σχηματίζει μία ιδέα για την αξιολόγηση γεγονότων βάση της αλληλεπίδρασής του με το περιβάλλον. Υπάρχει αναλογία του σήματος επιβράβευσης με τα βιολογικά συστήματα καθώς μπορούμε να θεωρήσουμε τα σήματα αυτά ως εμπειρίες ευχάριστες ή επίπονες.

  Βλέπουμε λοιπόν ότι το reward signal αποτελεί την κύρια βάση για την τροποποίηση της πολιτικής. Αν η επιλογή ενέργειας από την πολιτική οδηγήσει σε χαμηλό reward signal, η πολιτική ενδεχομένως να αλλάξει. Η αλλαγή αυτή γίνεται με σκοπό την αναζήτηση πιο ελπιδοφόρας κίνησης/ενέργειας, πάντα με οδηγό την μεγιστοποίηση του συνολικού σήματος επιβράβευσης. Όπως και οι πολιτικές, έτσι και τα σήματα επιβράβευσης ενδεχομένως να είναι στοχαστικές συναρτήσεις των καταστάσεων του περιβάλλοντος και των ενεργειών σε κάθε κατάσταση.

- Συνάρτηση Αξίας - Value Function

  Το reward signal όπως αναφέρθηκε παραπάνω, πληροφορεί τον πράκτορα για την ποιότητα της άμεσης ενέργειάς του. Η συνάρτηση αξίας υποδεικνύει την μακροπρόθεσμη αξία της άμεσης ενέργειας του πράκτορα.

  Γενικά, η αξία μιας κατάστασης είναι το συνολικό ποσό ανταμοιβής που ένας πράκτορας εκτιμά ότι θα συσσωρεύσει στο μέλλον ξεκινώντας από τη δοθείσα κατάσταση. Τα σήματα επιβράβευσης καθορίζουν ποιες είναι οι άμεσες, εγγενείς επιθυμητές καταστάσεις του περιβάλλοντος. Οι αξίες σε κάθε κατάσταση ορίζουν ποιες καταστάσεις είναι μακροπρόθεσμα επιθυμητές, αφού ληφθούν υπόψη οι καταστάσεις που θα ακολουθήσουν καθώς και η συνολική επιβράβευση που λαμβάνεται από τον πράκτορα ακολουθώντας το συγκεκριμένο μονοπάτι.

  Παραδείγματος χάρη, μία κατάσταση ενδεχομένως πάντα να αποφέρει μικρή ανταμοιβή στον πράκτορα αλλά να έχει υψηλή αξία διότι συνήθως ακολουθούν καταστάσεις που κρύβουν δώρα για τον πράκτορα (large accumulated reward). Θα μπορούσε φυσικά να ισχύει και το αντίστροφο. Κάνοντας και εδώ την αναλογία στα βιολογικά συστήματα, η αξία που προσδίδεται σε κάθε κατάσταση αντιστοιχεί σε μία πιο εκλεπτυσμένη και ολοκληρωμένη μακροπρόθεσμη εκτίμηση της συνολικής αξίας του να βρίσκεται ο πράκτορας σε αυτήν την κατάσταση.

  Η σωστή εκτίμηση του value function είναι το κεντρικό μας μέλημα κατά την εκπαίδευση διότι οι επιλογές των ενεργειών γίνονται βάση της εκτίμησης της συνάρτησης αξίας. Επιδιώκουμε αποφάσεις που μας οδηγούν σταδιακά σε καταστάσεις με υψηλότερη αξία διότι οι αποφάσεις αυτές μακροπρόθεσμα μας επιφέρουν τη μεγαλύτερη δυνατή ανταμοιβή. Δυστυχώς, είναι δύσκολη καμιά φορά η σωστή εκτίμηση της αξίας της κάθε κατάστασης. Οι αξίες κάθε κατάστασης ανανεώνονται και υπολογίζονται ξανά κάθε φορά που εκτελούμε μία αλληλουχία ενεργειών. Στην πραγματικότητα, ο πιο σημαντικός στόχος σε σχεδόν ολόκληρο το πεδίο του RL είναι η εύρεση σωστής μεθόδου για την αποτελεσματική εκτίμηση του value function.

  Ένα βασικό ερώτημα που τίθεται κατά τον υπολογισμό του value function είναι αν υπάρχει πεπερασμένος ορίζοντας ή άπειρος ορίζοντας για τη λήψη αποφάσεων. Πεπερασμένος ορίζοντας σημαίνει ότι υπάρχει ένας σταθερός χρόνος N μετά τον οποίο δεν λαμβάνουμε τίποτα υπόψιν. Συνεπώς,

  $$V([s_0, s_1, \dots s_{N+k}]) = V([s_0, s_1, \dots, s_N]) \forall k > 0 \tag{2.1}$$

  Αξίζει να σημειωθεί ότι υπάρχει πιθανότητα μεταβολής με το χρόνο της βέλτιστης ενέργειας συναρτήσει της κάθε κατάστασης, σε αντιδιαστολή με τις πολιτικές άπειρου ορίζοντα κατά την οποία ο πράκτορας συμπεριφέρεται με τον ίδιο τρόπο στην ίδια κατάσταση.

  Αν υποθέσουμε ότι οι προτιμήσεις ενός πράκτορα είναι στάσιμες, υπάρχει μαθηματική απόδειξη ότι υπάρχουν αποκλειστικά δύο τρόποι ορισμού της αμοιβής.

Additve Rewards: Η αξία μιας ακολουθίας καταστάσεων είναι:

$$V([s_0, s_1, s_2, ...]) = R(s_0) + R(s_1) + \dots \tag{2.2}$$

Discounted Rewards: Η αξία μιας ακολουθίας καταστάσεων είναι:

$$V([s_0, s_1, s_2, ...]) = R(s_0) + gR(s_1) + g^2R(s_2) + \dots \tag{2.3}$$

όπου ο συντελεστής γ ορίζεται ως ο παράγοντας ελάττωσης. Ο αριθμός γ έχει πεδίο τιμών μεταξύ 0 και 1 και περιγράφει το ισοζύγιο μεταξύ προτίμησης βραχυπρόθεσμων ή μακροπρόθεσμων ανταμοιβών.

- Μοντέλο Περιβάλλοντος - Environment Model

  Το τέταρτο και τελευταίο στοιχείο ενός συστήματος RL είναι το μοντέλο του περιβάλλοντος. Ως το μόνο προαιρετικό από τα τέσσερα στοιχεία, το μοντέλο ενός περιβάλλοντος μιμείται τη συμπεριφορά του περιβάλλοντος, η αλλιώς επιτρέπει την εξαγωγή συμπερασμάτων για τη συμπεριφορά του. Για παράδειγμα, δεδομένης της κατάστασης και ενέργειας, το μοντέλο μπορεί να προβλέψει την προκύπτουσα κατάσταση και ανταμοιβή. Μέθοδοι που περιέχουν μοντέλα του περιβάλλοντος και χρησιμοποιούνται για την επίλυση προβλημάτων RL λέγονται μέθοδοι με χρήση μοντέλου και έρχονται σε αντιδιαστολή με απλούστερα "model free" μοντέλα τα οποία λειτουργούν καθαρά με trial-and-error.

2.2.3 Διεπαφή Πράκτορα - Περιβάλλοντος

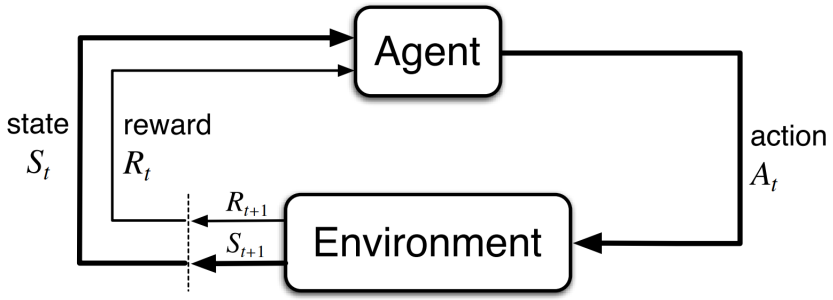
Όπως αναφέρθηκε και προηγουμένως, ο μαθητευόμενος στο πρόβλημα της ενισχυτικής μάθησης ονομάζεται πράκτορας. Η αλληλεπίδραση του πράκτορα γίνεται με το περιβάλλον. Ο πράκτορας και το περιβάλλον αλληλεπιδρούν συνεχώς, καθώς ο πράκτορας επιλέγει τις ενέργειες και το περιβάλλον ανταποκρίνεται στις ενέργειες αυτές και παρουσιάζει νέες καταστάσεις στον πράκτορα.

Το περιβάλλον επίσης, παράγει τις ανταμοιβές (rewards) που τροφοδοτεί στον πράκτορα, ειδικές τιμές που προσπαθεί ο πράκτορας να μεγιστοποιήσει.

Ο πράκτορας και το περιβάλλον αλληλεπιδρούν σε κάθε μία από τις διακριτές χρονικές μεταβάσεις, $t = 0, 1, 2, ...$. Περιοριζόμαστε στο διακριτό χρόνο για απλούστευση, πολλές από τις ιδέες της ενισχυτικής μάθησης μπορούν να γενικευτούν και σε συνεχή χρόνο [4].

Σε κάθε χρονική στιγμή t, ο πράκτορας δέχεται κάποια αναπαράσταση της κατάστασης του περιβάλλοντος, $S_t \in S$, όπου S το σύνολο των δυνατών καταστάσεων, και επιλέγει μία ενέργεια, $A_t \in A(St)$, όπου A(St) το σύνολο των ενεργειών διαθέσιμο στην κατάσταση S_t.

Την επόμενη χρονική στιγμή, ως αποτέλεσμα της προηγούμενης ενέργειας, ο πράκτορας λαμβάνει την ανταμοιβή του με τη μορφή ενός αριθμού R_{t+1} και πηγαίνει σε μία νέα κατάσταση S_{t+1}. Το παρακάτω σχήμα παρουσιάζει την αλληλεπίδραση περιβάλλοντος και πράκτορα:



Σχήμα 2.1: Τρόπος αλληλεπίδρασης πράκτορα και περιβάλλοντος. Τα βέλη δείχνουν τη ροή πληροφορίας μεταξύ τους [13]

Σε κάθε χρονική στιγμή, ο πράκτορας προσπαθεί να χαρτογραφήσει καταστάσεις σε κατανομές πιθανών κινήσεων επιλέγοντας κάθε δυνατή κίνηση. Αυτή η χαρτογράφηση ονομάζεται πολιτική και συμβολίζεται με π_t, όπου $\pi_t(s|a)$ ορίζεται ως η πιθανότητα $\mathrm{A}_t = a$ για $\mathrm{S}_t = s$. Οι μέθοδοι ενισχυτικής μάθησης ορίζουν το πως ο πράκτορας τροποποιεί την πολιτική του βασιζόμενος στην αλληλεπίδραση με το περιβάλλον. Ένας πετυχημένος πράκτορας μεγιστοποιεί τη συνολική ανταμοιβή που δέχεται στο βάθος του χρόνου.

Το πλαίσιο αυτό, ως αφηρημένο και ευέλικτο, έχει τη δυνατότητα εφαρμογής σε πολλά διαφορετικά παραδείγματα και με διαφορετικούς τρόπους. Για παράδειγμα, τα χρονικά βήματα δεν είναι αναγκαστικό να αντιστοιχούν σε σταθερά βήματα πραγματικού χρόνου. Ενδεχομένως να αναφέρονται σε αυθαίρετα διαδοχικά στάδια λήψης αποφάσεων και ενεργειών. Ενέργειες μπορούν να θεωρηθούν χειρισμοί από χαμηλού επιπέδου, όπως π.χ. τάσεις που εφαρμόζονται σε κινητήρες ρομπότ, μέχρι αποφάσεις υψηλού επιπέδου, όπως π.χ. η εγγραφή σε μεταπτυχιακό πρόγραμμα. Όμοια, οι καταστάσεις εμφανίζονται σε μία ποικιλία διαφορετικών μορφών. Ενδεχομένως να προσδιορίζονται από χαμηλού επιπέδου αισθήσεις, όπως άμεσες ενδείξεις αισθητήρων, μέχρι υψηλού επιπέδου αφηρημένες έννοιες, όπως για παράδειγμα συμβολικές αναπαραστάσεις αντικειμένων σε ένα δωμάτιο. Ορισμένες από τις οντότητες που ορίζουν μία κατάσταση ενδεχομένως να λαμβάνουν υπόψη στοιχεία από το παρελθόν. Σε γενικές γραμμές λοιπόν, οι ενέργειες αντιπροσωπεύουν οποιαδήποτε απόφαση επιτρέπεται να πάρουμε και οι καταστάσεις αντιπροσωπεύουν οποιοδήποτε ερέθισμα λαμβάνεται έτσι ώστε να βοηθήσει στη λήψη αποφάσεων.

Συγκεκριμένα, το όριο μεταξύ πράκτορα και περιβάλλοντος δεν ταυτίζεται συνήθως με το φυσικό όριο του ρομπότ. Συνήθως, το όριο μετατοπίζεται πιο κοντά στον πράκτορα. Για το παράδειγμα του ρομπότ, οι κινητήρες και οι μηχανικοί σύνδεσμοι θα πρέπει συνήθως να θεωρούνται τμήματα του περιβάλλοντος και όχι τμήματα του πράκτορα. Οι ανταμοιβές παρόμοια, υπολογίζονται εντός των φυσικών ή τεχνητών σωμάτων των συστημάτων μάθησης, αλλά θεωρούνται εξωτερικά ως προς τον πράκτορα. Ο γενικός κανόνας που ακολουθείται στην οριοθέτηση μεταξύ πράκτορα και περιβάλλοντος είναι η ακόλουθη: "Οτιδήποτε δεν ελέγχεται άμεσα από πράκτορα θεωρείται περιβάλλον".

Δεν υποθέτουμε ότι το περιβάλλον είναι άγνωστο στον πράκτορα καθώς ο πράκτορας

συχνά γνωρίζει αρκετά καλά πως υπολογίζονται οι ανταμοιβές του ως συνάρτηση των ενεργειών του και των καταστάσεων στις οποίες εκτελούνται οι ενέργειες αυτές. Παρ' όλα αυτά, θεωρούμε πάντα ότι ο υπολογισμός της ανταμοιβής είναι εξωτερικός για τον πράκτορα. Ουσιαστικά ο υπολογισμός αυτός ορίζει το πρόβλημα που καλείται ο πράκτορας να λύσει και απαγορεύεται να αλλάξει αυθαίρετα από τον πράκτορα. Σε ορισμένες περιπτώσεις, όπως και στη διπλωματική αυτή εργασία, ο πράκτορας γνωρίζει όλες τις λεπτομέρειες για το πως λειτουργεί το περιβάλλον αλλά και πάλι έρχεται αντιμέτωπος με ένα αρκετά δύσκολο πρόβλημα ενισχυτικής μάθησης. Ο πράκτορας γνωρίζει ακριβώς πως λειτουργεί ο κύβος του Ρούμπικ αλλά αντιμετωπίζει μεγάλο πρόβλημα κατά την επίλυση του το οποίο θα αναλυθεί αρκετά στα επόμενα κεφάλαια. Το σύνορο πράκτορα-περιβάλλοντος αλλάζει φυσικά ανάλογα με τη χρήση. Σε ένα περίπλοκο ρομπότ συναντάμε πολλούς διαφορετικούς πράκτορας οι οποίοι λειτουργούν σε διαφορετικά αφαιρετικά επίπεδα.

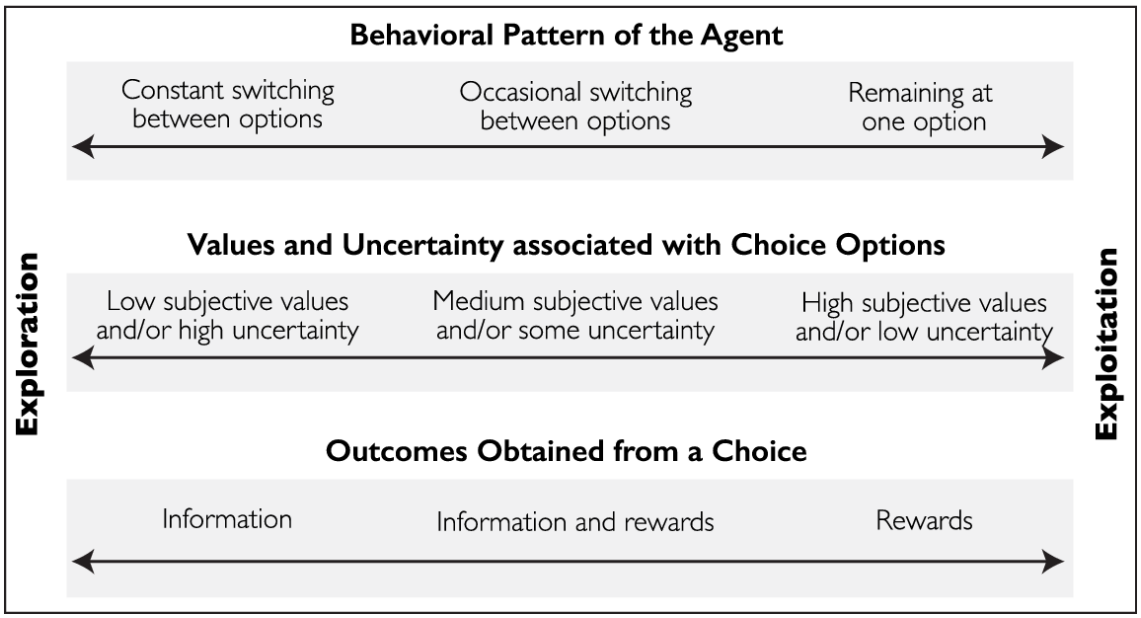
2.2.4 Διαφορες με Supervised Learning

Το reinforcement learning διαφοροποιείται από την εποπτευόμενη μάθηση που αποτελεί πιο διαδεδομένη περιοχή μηχανικής μάθησης. Η εποπτευόμενη μάθηση ορίζεται ως η εκμάθηση από ένα σύνολο επισυναπτόμενων παραδειγμάτων που παρέχονται από έναν εξειδικευμένο εξωτερικό επόπτη. Κάθε παράδειγμα αποτελεί έναν συνδυασμό μιας κατάστασης και μιας προδιαγραφής (ταμπέλας) η οποία υποδεικνύει τη σωστή ενέργεια στην εκάστοτε περίπτωση. Ο σκοπός αυτής της διαδικασίας είναι να γίνει γενίκευση των ενεργειών έτσι ώστε να ενεργεί σωστά ακόμα και σε καταστάσεις που δεν έχει ξανασυναντήσει κατά τη διαδικασία εκπαίδευσης. Το reinforcement learning συνιστά ένα τρίτο παράδειγμα μηχανικής μάθησης, παράλληλα με την εποπτευόμενη και μη-εποπτευόμενη μάθηση.

2.2.5 Προκλήσεις του Reinforcement Learning

Μία από τις προκλήσεις που υπάρχουν στο Reinforcement Learning είναι το trade-off μεταξύ exploration και exploitation.

Η ιδέα αυτή εξηγείται παρακάτω. Ένας πράκτορας, κατά την αλληλεπίδραση με το περιβάλλον του, θέλει να μεγιστοποιήσει την ανταμοιβή του για τις πράξεις που εκτελεί. Κατά τη διαδικασία αυτή είναι απαραίτητο να προτιμήσει ενέργειες που έχει δοκιμάσει στο παρελθόν και βρέθηκαν αποτελεσματικές. Για να ανακαλύψει παρόλα αυτά τέτοιες ενέργειες, οφείλει να δοκιμάσει ενέργειες που δεν έχει επιλέξει στο παρελθόν. Το δίλημμα είναι ότι ούτε η εξερεύνηση ούτε η εκμετάλλευση μπορούν να λειτουργήσουν αποκλειστικά χωρίς να αποτύχουν. Ο πράκτορας οφείλει να δοκιμάσει μία πληθώρα ενεργειών και σταδιακά να προτιμάει αυτές που φαίνονται καλύτερες. Σε μία στοχαστική διαδικασία, κάθε ενέργεια πρέπει να δοκιμαστεί αρκετές φορές έτσι ώστε να διαμορφωθεί μία αξιόπιστη εκτίμηση της αναμενόμενης ανταμοιβής της. Το exploration-exploitation trade-off έχει μελετηθεί έντονα από μαθηματικούς για πολλές δεκαετίες και παραμένει άλυτο.



Σχήμα 2.2: Εικονογράφηση τριών εννοιολογικών διαστάσεων του exploration - exploitation. Συμπεριφορικά πρότυπα του πράκτορα, τιμές και στοχαστικότητα η οποία συσχετίζεται με το σύνολο επιλογών και τα λαμβανόμενα αποτελέσματα

2.2.6 Ιδιαιτερότητες του Reinforcement Learning

Ένα άλλο βασικό χαρακτηριστικό του reinforcement learning το οποίο έρχεται σε αντίθεση με άλλες προσεγγίσεις μηχανικής μάθησης είναι ότι λαμβάνει υπόψιν όλο το πρόβλημα ενός πράκτορα που έχει σαν στόχο τη μεγιστοποίηση κάποιο σήματος επιβράβευσης αλληλεπιδρώντας με ένα αβέβαιο περιβάλλον. Για παράδειγμα, παρόλο που υπάρχουν εφαρμογές εποπτευόμενης μάθησης ευρέως γνωστές, δεν είναι ξεκάθαρο πως η εκμάθηση αυτή θα είναι χρήσιμη σε μεγαλύτερα και πιο ολοκληρωμένα προβλήματα. Παρόλο που οι εφαρμογές του supervised/unsupervised learning έχουν αποφέρει πολλά χρήσιμα αποτελαματα, η εστίαση σε μία απομονωμένη εφαρμογή αποτελεί σημαντικό περιορισμό.

Το Reinforcement Learning έχει διαφορετική φιλοσοφία. Όλοι οι Reinforcement Learning (RL) πράκτορες έχουν καλώς ορισμένους στόχους, αισθάνονται το περιβάλλον με το οποίο αλληλεπιδρούν και επιλέγουν ενέργειες οι οποίες επηρεάζουν το περιβάλλον. Επιπλέον, θεωρείται συνήθως δεδομένο ότι ο πράκτορας θα λειτουργεί με υψηλή την αίσθηση αβεβαιότητας για το περιβάλλον με το οποίο αλληλεπιδρά. Ένας πλήρης, διαδραστικός πράκτορας που θέτει στόχους και τους επιτυγχάνει, ενδέχεται να είναι κομμάτι κάποιου άλλου μεγαλύτερου συμπεριφορικού συστήματος. Ένας πράκτορας που παρακολουθεί το επίπεδο φόρτισης μιας μπαταρίας και στέλνει εντολές στην αρχιτεκτονική ελέγχου ενός ρομπότ είναι ένα τέτοιο παράδειγμα.

Μία από τις πιο συναρπαστικές πτυχές του RL είναι οι καρποφόρες αλληλεπιδράσεις του με διαφορετικούς επιστημονικούς κλάδους όπως η στατιστική, η βελτιστοποίηση, ο βέλτιστος

έλεγχος και άλλα πεδία μαθηματικών. Για παράδειγμα, η ικανότητα κάποιων αλγορίθμων RL να μαθαίνουν με διαδοχικές προσεγγίσεις, συνεισφέρει στην κλασσική θεωρία ελέγχου με το να αντιμετωπίζει προβλήματα που πάσχουν από το γνωστό "curse of dimensionality". Ακόμα, η ενισχυτική μάθηση εμπλέκεται έντονα σε θέματα ψυχολογίας και νευροεπιστήμης με σημαντικά αμφίδρομα οφέλη. Από όλες τις μορφές μηχανικής μάθησης, αυτή που πλησιάζει περισσότερο την ανθρώπινη διαδικασία εκμάθησης είναι το Reinforcement Learning, και αυτό γιατί πολλοί από τους βασικούς αλγορίθμους RL αρχικά εμπνεύστηκαν από τα συστήματα βιολογικής μάθησης.

Τέλος, η ενισχυτική μάθηση αποτελεί μέρος μιας ευρύτερης τάσης στην τεχνητή νοημοσύνη να βασιστεί σε γενικότερες αρχές. Μέχρι τη δεκαετία του 1960 πολλοί ερευνητές πίστευαν ότι δεν υπάρχουν γενικές αρχές πίσω από την έννοια της γνώσης και ότι εναλλακτικά η γνώση είναι απλά μία άφθονη συλλογή μεμονωμένης πληροφορίας. Υπήρχε η πεποίθηση ότι αν μπορούσε κανείς να συλλέξει αρκετή πληροφορία, θα γινόταν ευφυής. Μάλιστα, μέθοδοι που βασίζονταν πάνω σε γενικές αρχές θεωρούνταν "ασθενείς μέθοδοι" ενώ εκείνες που βασίζονταν σε πιο συγκεκριμένες γνώσεις ονομάστηκαν "ισχυρές μέθοδοι". Η σύγχρονη τεχνητή νοημοσύνη βασίζεται σε γενικές αρχές για εκμάθηση, αναζήτηση και λήψη αποφάσεων σε συνδυασμό με την προσπάθεια ενσωμάτωσης τεράστιων ποσοτήτων πληροφορίας.

2.2.7 Μέθοδοι υπολογισμού Action - Value

Ξεκινάμε εξετάζοντας προσεκτικά ορισμένες απλές μεθόδους για την εκτίμηση της αξίας κάθε δυνατής ενέργειας, και για τη χρήση των εκτιμήσεων αυτών για τη λήψη αποφάσεων. Σαν σύμβαση, αναπαριστούμε την αληθινή αξία μιας κίνησης/ενέργειας ως q(a) και της εκτιμώμενης αξίας τη χρονική στιγμή t ως $Q_t(a)$. Θυμίζουμε ότι η πραγματική αξία μιας ενέργειας είναι η μέση ανταμοιβή που λαμβάνεται όταν επιλέγουμε αυτή η ενέργεια. Ένας τρόπος να εκτιμηθεί η αξία είναι κάνοντας χρήση του μέσου όρου όλων των ανταμοιβών. Με άλλα λόγια, συμβολίζουμε με $N_t(a)$ το πλήθος των χρήσεων της κίνησης α μέχρι τη χρονική στιγμή t. Οι ανταμοιβές μέχρι το t είναι $R_1, R_2, \ldots, R_{Nt}(a)$ οπότε η συνολική αξία υπολογίζεται ως:

$$Qt(a) = (R_1 + R_2 + ... + R_{Nt}(a))/N_t(a) \tag{2.4}$$

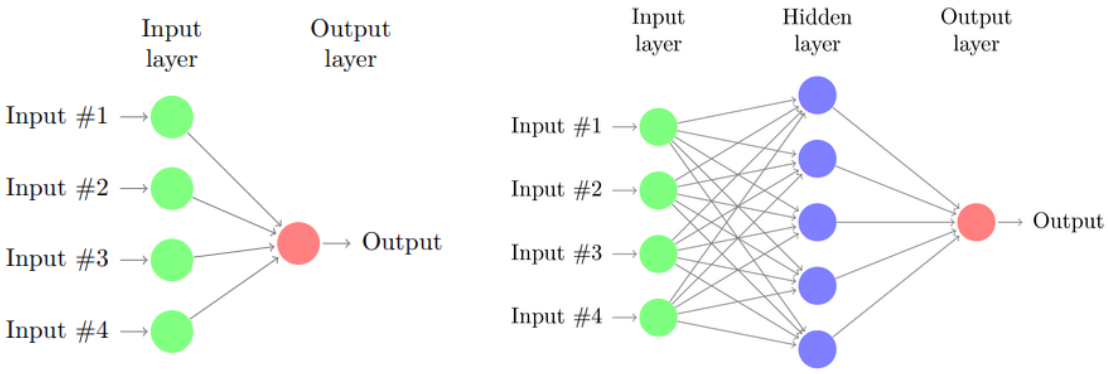
Στην περίπτωση που $N_t(a) = 0$, αρχικοποιούμε την $Q_t(a)$ με μία τιμή (π.χ. 0) Καθώς το $N_t(a)$ß και σύμφωνα με τη θεωρία μεγάλων αριθμών, η $Q_t(a)$ συγκλίνει στην q(t) [3].

2.3 Χρήση Νευρωνικών Δικτύων στο RL

2.3.1 Τεχνητά Νευρωνικά Δίκτυα

Τα τεχνητά νευρωνικά δίκτυα είναι υπολογιστικά μοντέλα που προσπαθούν να προσομοιάσουν την συμπεριφορά και τα προσαρμοστικά χαρακτηριστικά των βιολογικών νευρωνικών δικτύων. Αυτά τα μοντέλα υλοποιούνται είτε με την χρήση λογισμικού, είτε χρησιμοποιώντας ηλεκτρονικά στοιχεία (hardware). Τα τεχνητά νευρωνικά δίκτυα χρησιμοποιούνται συχνά για την επίλυση προβλημάτων όπου είναι δύσκολο ή αδύνατο να δοθεί μια αναλυτική λύση.

Ένα τεχνητό νευρωνικό δίκτυο αποτελείται από επιμέρους μονάδες που καλούνται νευρώνες, οι οποίοι είναι διασυνδεδεμένοι μεταξύ τους με σταθμισμένες συνδέσεις. Οι νευρώνες είναι συνήθως χωρισμένοι σε επίπεδα και ο αριθμός των επιπέδων εξαρτάται από την αρχιτεκτονική του δικτύου.



Σχήμα 2.3: Γενική μορφή τεχνητών νευρωνικών δικτύων.

Τα επίπεδα μπορούν να χωριστούν σε επίπεδο εισόδου, κρυφά επίπεδα και επίπεδο εξόδου. Οι νευρώνες στο επίπεδο εισόδου δέχονται πληροφορίες από το περιβάλλον, ενώ οι νευρώνες στο επίπεδο εξόδου παράγουν ένα σήμα εξόδου. Οι ενδιάμεσοι νευρώνες δεν αποτελούν μέρος ούτε της εισόδου, ούτε της εξόδου, για αυτό χαρακτηρίζονται ως κρυφοί. Τα περισσότερα τεχνητά νευρωνικά δίκτυα έχουν ένα ή περισσότερα κρυφά επίπεδα, ενώ είναι δυνατή και η ύπαρξη δικτύου χωρίς κανένα κρυφό επίπεδο. Παραπάνω παρουσιάζονται δύο γενικευμένες μορφές δικτύων πρόσθιας τροφοδότησης. Ένα χωρίς κρυφό επίπεδο, και ένα με ένα κρυφό επίπεδο.

2.3.2 Χαρακτηριστικά Νευρωνικών Δικτύων

Τα νευρωνικά δίκτυα που χρησιμοποιήθηκαν έχουν τα εξής χαρακτηριστικά

- Ένα σύνολο συνδέσεων (συνάψεις), κάθε μία από τις οποίες χαρακτηρίζεται από ένα βάρος. Οι συνάψεις αποτελούν την είσοδο των διαφόρων σημάτων στον νευρώνα και μπορεί να προέρχονται από άλλους νευρώνες ή 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.

- Μία συνάρτηση μεταφοράς. Αυτή είναι μία μαθηματική συνάρτηση (γραμμική, κατωφλίου, σιγμοειδής κ.α.) που δέχεται ως είσοδο την έξοδο του αθροιστή και το αποτέλεσμα αποτελεί την έξοδο του νευρώνα στο υπόλοιπο σύστημα ή στο εξωτερικό Περιβάλλον.

2.3.3 Αρχιτεκτονική Νευρωνικών Δικτύων

Η αρχιτεκτονική των τεχνητών νευρωνικών δικτύων επηρεάζει την λειτουργικότητά τους. Οι πιο συνήθεις αρχιτεκτονικές είναι τα δίκτυα πρόσθιας τροφοδότησης, και τα δίκτυα με ανάδραση.

1. Τα δίκτυα πρόσθιας τροφοδότησης είναι χωρισμένα σε επίπεδα όπου η ροή της πληροφορίας ρέει μόνο προς την μία κατεύθυνση. Ο κάθε νευρώνας τροφοδοτεί μόνο νευρώνες που ανήκουν στο επόμενο επίπεδο. Αυτή είναι η απλούστερη μορφή ΤΝΝ.

2. Τα δίκτυα με ανάδραση μπορεί να έχουν συνδέσεις μεταξύ των νευρώνων του ιδίου επιπέδου, καθώς και προηγούμενων. Η παρουσία αναδράσεων έχει μεγάλη επίπτωση στις δυνατότητες μάθησης του δικτύου καθώς και στην επίδοση του.

Η έξοδος του νευρωνικού δικτύου όταν δέχεται πληροφορίες από το περιβάλλον εξαρτάται από την συνάρτηση μεταφοράς των νευρώνων, τις συνδέσεις μεταξύ αυτών και τα βάρη των συνδέσεων.

Μία από τις βασικότερες δυνατότητες ενός ΤΝΝ είναι ότι μπορεί να μαθαίνει από το περιβάλλον και μέσα από την διαδικασία αυτή να βελτιώνει την απόδοσή του. Χρησιμοποιώντας μεθόδους εκμάθησης μπορούν να προσαρμοστούν οι συνδέσεις μεταξύ των νευρώνων έτσι ώστε να βελτιωθεί η απόδοση του και έτσι να "μάθει". Οι περισσότερες μέθοδοι εκπαίδευσης ΤΝΝ χρησιμοποιούν παραδείγματα και μπορούν να χωριστούν σε εκπαίδευση με επίβλεψη - supervised learning και εκπαίδευση χωρίς επίβλεψη - unsupervised learning.

Οι μέθοδοι εκπαίδευσης με επίβλεψη, όπως για παράδειγμα ο αλγόριθμος backpropagation [15], βασίζονται στην σύγκριση της εξόδου του ΤΝΝ με μία επιθυμητή έξοδο, για έναν δοσμένο σετ από εισόδους. Στην συνέχεια η διαφορά των δύο εξόδων (επιθυμητής και πραγματικής) θεωρείται η τιμή σφάλματος, και αυτό το σφάλμα διαδίδεται προς τα πίσω μέσα από το δίκτυο προσαρμόζοντας στην πορεία τα βάρη των συνάψεων βάση του κανόνα εκμάθησης που χρησιμοποιεί.

2.4 Μαρκοβιανές Αλυσίδες Αποφάσεων

2.4.1 Αλυσίδες Markov

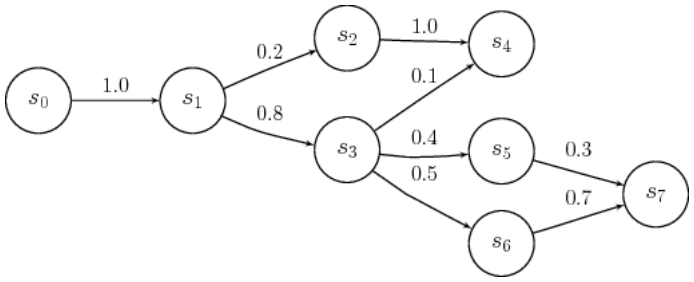
Η Μαρκοβιανή αλυσίδα είναι μια στοχαστική διαδικασία με τη μαρκοβιανή ιδιότητα για ένα πεπερασμένο ή μετρήσιμο χώρο καταστάσεων. Ο όρος Ἠμαρκοβιανή αλυσίδα' αναφέρεται στην αλληλουχία (ή αλυσίδα) των καταστάσεων μέσω των οποίων κινείται μια τέτοια διαδικασία. Συνήθως μια Μαρκοβιανή αλυσίδα ορίζεται για μια διακριτή συλλογή χρόνων (Μαρκοβιανή αλυσίδα διακριτών χρόνων), παρόλο που μερικοί συγγραφείς χρησιμοποιούν την ίδια ορολογία για να αναφερθούν σε Μαρκοβιανή αλυσίδα συνεχούς χρόνου. Η χρήση του όρου στη μεθοδολογία Μαρκοβιανής αλυσίδας Μόντε Κάρλο καλύπτει περιπτώσεις όπου η διαδικασία βρίσκεται σε διακριτό χρόνο (διακριτά αλγοριθμικά βήματα) με ένα συνεχή χώρο καταστάσεων. Οι πληροφορίες που ακολουθούν επικεντρώνονται στην περίπτωση διακριτού χρόνου και διακριτού χώρου καταστάσεων.

Μια τυχαία διαδικασία διακριτού χρόνου περιλαμβάνει ένα σύστημα που βρίσκεται σε μια συγκεκριμένη κατάσταση σε κάθε βήμα, με την κατάσταση να μεταβάλλεται τυχαία μεταξύ των βημάτων. Τα βήματα συχνά θεωρούνται ως στιγμές στο χρόνο αλλά μπορούν εξίσου να αναφέρονται σε φυσική απόσταση ή οποιαδήποτε άλλη διακριτή μέτρηση: τυπικά τα βήματα είναι οι ακέραιοι ή φυσικοί αριθμοί και η τυχαία διαδικασία είναι η χαρτογράφησή τους σε καταστάσεις. Η Μαρκοβιανή ιδιότητα δηλώνει ότι η δεσμευμένη πιθανότητα κατανομής του συστήματος στο επόμενο βήμα (και κατά βάση, σε όλα τα μελλοντικά βήματα) εξαρτάται μόνο από την παρούσα κατάσταση του συστήματος και όχι αθροιστικά από την κατάσταση του συστήματος σε προηγούμενα βήματα.

Πιο αυστηρά, μια Μαρκοβιανή Αλυσίδα είναι μια ακολουθία τυχαίων μεταβλητών X1, X2, X3, ... με τη Μαρκοβιανή Ιδιότητα, δηλαδή με δεδομένη την παρούσα κατάσταση, οι παλαιότερες και οι μελλοντικές καταστάσεις είναι ανεξάρτητες. Ορίζουμε

$$Pr(X_{n+1} = x | X_1 = x_1, X_2 = x_2, ..., X_n = x_n) \tag{2.5}$$

Οι πιθανές τιμές των Xi σχηματίζουν ένα αριθμήσιμο σύνολο Σ που ονομάζουμε χώρο-καταστάσεων της αλυσίδας



Σχήμα 2.4: Παράδειγμα κατευθυνόμενου γραφήματος Markov.

Οι Μαρκοβιανές Αλυσίδες συχνά περιγράφονται από ένα κατευθυνόμενο γράφημα που οι άκρες του περιγράφουν τις πιθανότητες μετάβασης από τη μια κατάσταση στις άλλες.

Καθώς το σύστημα μεταβάλλεται τυχαία, είναι γενικά αδύνατο να προβλεφθεί με βεβαιότητα η κατάσταση μιας Μαρκοβιανής αλυσίδας σε ένα δεδομένο μελλοντικό σημείο. Παρ΄ όλα αυτά, οι στατιστικές ιδιότητες του μέλλοντος του συστήματος μπορούν να προβλεφθούν. Σε πολλές εφαρμογές, είναι αυτές οι στατιστικές ιδιότητες που είναι σημαντικές.

Οι αλλαγές κατάστασης του συστήματος ονομάζονται μεταβάσεις και οι πιθανότητες που σχετίζονται με τις διάφορες μεταβατικές καταστάσεις ονομάζονται πιθανότητες μετάβασης. Η διαδικασία χαρακτηρίζεται από ένα χώρο καταστάσεων, μια μήτρα μετάβασης που περιγράφει τις πιθανότητες μιας συγκεκριμένης μετάβασης και μια αρχική κατάσταση ή αρχική κατανομή στο χώρο καταστάσεων. Κατά συνθήκη, θεωρούμε ότι όλες οι δυνατές καταστάσεις και μεταβάσεις έχουν συμπεριληφθεί στον ορισμό των διαδικασιών, ώστε υπάρχει πάντα μια επόμενη κατάσταση και η διαδικασία συνεχίζεται για πάντα.

Αυτό που διαφοροποιεί μία MDP με μία αλυσίδα Markov είναι η πρόσθεση κινήσεων και ανταμοιβών. Σε μία MDP, κάθε μετάβαση κατάστασης παράγει μία ανταμοιβή για το σύστημα (το οποίο μπορεί να χρησιμοποιηθεί για έλεγχο). Η ανταμοιβή αυτή ως αριθμητική τιμή ενδεχομένως να είναι θετική, αρνητική ή και μηδέν. Μερικές ή ακόμα και όλες οι καταστάσεις είναι καταστάσεις στις οποίες παίρνουμε αποφάσεις και τα αντίστοιχα χρονικά σημεία τους ονομάζονται “decision epochs”. Οποτεδήποτε ένα σύστημα βρίσκεται σε ένα σημείο που πρέπει να πάρει μία απόφαση, ο πράκτορας οφείλει να αποφασίσει μεταξύ των διαθέσιμων επιλογών. Όλες οι δυνατές επιλογές/κινήσεις ονομάζονται “χώρος επιλογών” ή αλλιώς “action-space” της MDP. Η επιλεγμένη ενέργεια θα επηρεάσει την πιθανοτική κατανομή του συστήματος στην επόμενη κατάσταση. Μια διαδικασία Markov λοιπόν με διακριτό χώρο καταστάσεων ονομάζεται αλυσίδα Markov.

Αυστηρά, μία Μαρκοβιανή Διαδικασία MDP αποτελείται από:

1. Σύνολο Καταστάσεων - State-space

   S (επιτρέπεται να συμπεριλαμβάνονται αρχικές και τελικές καταστάσεις)

2. Σύνολο Ενεργειών - Action-space

   $A \equiv \bigcup_{s \in S} A(s)$

3. Συνάρτηση Μετάβασης - Transition Function

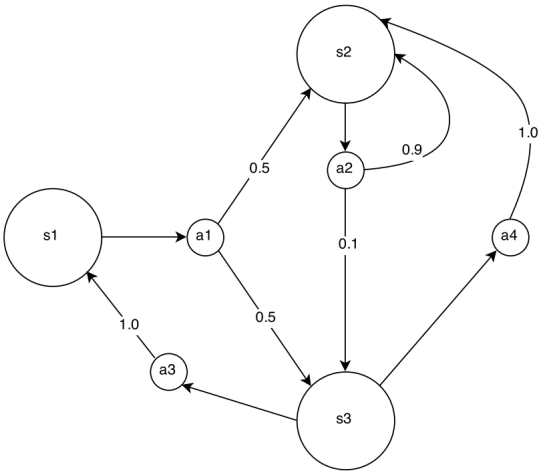
   $T(s, a, s') = Pr(S_{t+1} = s' | S_t = s, A_t = a)$

4. Συνάρτηση Ανταμοιβής - Reward Function

   $R(s, a, s')$

- s,s'$\in S$
- a$\in A(s)$

- S_t, τυχαία μεταβλητή που δηλώνει την κατάσταση S την χρονική στιγμή t
- A_t, τυχαία μεταβλητή που δηλώνει την ενέργεια A την χρονική στιγμή t



Σχήμα 2.5: Μία απλή Μαρκοβιανή Διαδικασία με 3 καταστάσεις και 4 διαφορετικές ενέργειες. Η αρίθμηση των ακμών αντιπροσωπεύει τις πιθανότητες μετάβασης. Οι ανταμοιβές δεν φαίνονται στο παραπάνω σχήμα.

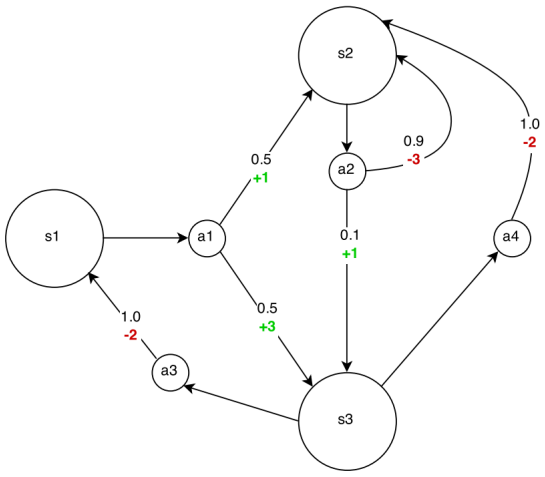
Οι ανταμοιβές είναι πιθανοθεωρητικές. Είναι σημαντικό να προσδιοριστεί αν ευθύνεται για αυτή τη στοχαστικότητα η στοχαστική φύση της υλοποίησης ή το γεγονός ότι οι μεταβάσεις είναι πιθανοτικές αλλά οι ανταμοιβές εξαρτώνται ντετερμινιστικά από αυτές. Συνήθως για απλοποίηση θεωρούμε ότι ισχύει το δεύτερο. Με άλλα λόγια, για μία μετάβαση (s,a,s') η οποία συμβαίνει τη χρονική στιγμή t, η άμεση ανταμοιβή ορίζεται ως: $R : S_t * A_t * S_{t+1}$. Επομένως, κανείς αρκεί να λάβει υπόψη του τις κατανομές μετάβασης των καταστάσεων διότι η κατανομή της ανταμοιβής (συναρτήσει των ενεργειών) εξαρτάται μόνο από αυτές. Θεωρείται επίσης ότι οι πιθανότητες μετάβασης των καταστάσεων δεν μεταβάλλονται με το χρόνο.

Η ιδιότητα Markov μπορεί να εκφραστεί ως εξής:

$Pr(S_{t+1} = s' | S_t = s_t, A_t = a_t, S_{t1} = s_{t1}, A_{t1} = a_{t1}, ..., S_0 = s_0, A_0 = a_0)$
$\equiv Pr(S_{t+1} = s' | S_t = s_t, A_t = a_t)$

υπονοώντας έτσι ότι ο τρόπος με τον οποίον καταλήξαμε στην κατάσταση που βρισκόμαστε δεν επηρεάζει την πιθανοτική κατανομή των επόμενων καταστάσεων.

Ως αποτέλεσμα της διαδικασίας Markov, η περιγραφή της παρούσας κατάστασης (η οποία ουσιαστικά είναι η κατάσταση στην οποία ανήκει το σύστημά μας) οφείλει να είναι τόσο λεπτομερής έτσι ώστε να επιτρέπει την MDP να μοντελοποιεί την εξέλιξη της πραγματικής διαδικασίας. Με άλλα λόγια, η περιγραφή της κατάστασης οφείλει να περιέχει όλες τις πτυχές του κόσμου που μπορούν να επηρεάσουν τον προσδιορισμό της έκβασης μιας ενέργειας, αλλιώς το μοντέλο μας δεν έχει τη δυνατότητα να μοντελοποιήσει το πραγματικό περιβάλλον.



Σχήμα 2.6: Έχοντας υποθέσει ότι η άμεση ανταμοιβή κάθε μετάβασης είναι σταθερή, οι ανταμοιβές μπορούν τώρα να αναπαρασταθούν στο γράφημα του προηγούμενου σχήματος.

2.4.2 Το πρόβλημα απόφασης Markov

Η πολιτική μιας MDP είναι μία αλληλουχία συναρτήσεων: $\pi = m_0, m_1, ...$ όπου τα m_t αναφέρονται στις χρονικές στιγμές t και αποτελούν μία αντιστοιχία ζευγών "κατάστασης-ενέργειας" και κατανομής πιθανοτήτων των ενεργειών που καλείται να επιλέξει ο πράκτορας σε κάθε κατάσταση. mt : S x A $\rightarrow [0, 1]$. Μία πολιτική ονομάζεται στατική αν δεν εξαρτάται από το χρόνο: π = m,m,... και στην περίπτωση αυτή, η προαναφερθείσα χαρτογράφηση μπορεί να θεωρηθεί πολιτική: π : S x A $\rightarrow [0, 1]$ και π(s,a) = $\Pr(A_t = a|S_t = s), \forall s \in S, \forall a \in A(s)$.

Με άλλα λόγια, σε μία στατική πολιτική, η κατανομή των ενεργειών εξαρτάται μόνο από την παρούσα κατάσταση. Επιπλέον, στην περίπτωση που η στατική πολιτική είναι και ντετερμινιστική, τότε η αντιστοιχία από το σύνολο καταστάσεων στο σύνολο ενεργειών είναι:

$$\pi : S \rightarrow A, \pi(s) \in A(s), \forall s \in S \tag{2.6}$$

Επομένως, μία ντετερμινιστική και στατική πολιτική ορίζει ποιες ενέργειες να επιλεγούν σε κάθε κατάσταση. Η συνάρτηση επιστροφής λειτουργεί ως μία μετρική της απόδοσης του πράκτορα έτσι ώστε να διαφοροποιηθούν οι καλές από τις κακές πολιτικές. Συνήθως μία τέτοια μαθηματική συνάρτηση συνυπολογίζει τις άμεσες ανταμοιβές που έχει συλλέξει σε ένα πεπερασμένο ή και άπειρο βάθος χρόνου. Αυτή ακριβώς αποτελεί και η μαθηματική έκφραση του στόχου του πράκτορα. Το πρόβλημα της διαδικασίας Markov ορίζεται ως το πρόβλημα εύρεσης πολιτικής η οποία βελτιστοποιεί τον προκαθορισμένο αυτό στόχο, δοθείσας μιας MDP.

Η συνάρτηση αυτή δίνει τη δυνατότητα στον πράκτορα να ενσωματώσει όλες τις πληροφορίες που θεωρεί χρήσιμες και να αντιμετωπίσει τα πολλαπλά και πολλές φορές αντικρουόμενα κριτήρια για την εύρεση βέλτιστης λύσης.

Η συνάρτηση τιμής/αξίας (value function) υπολογίζει την βέλτιστη αξία για την συνάρτηση στόχου ή οποία γράφεται συναρτήσει κάθε κατάστασης, δηλαδή, $V^*(s)$ είναι η αναμενόμενη συνολική αξία, ξεκινώντας από την κατάσταση s $\in S$ δεδομένης βέλτιστης συμπεριφοράς από την κατάσταση s και ύστερα.

Όμοια, η συνάρτηση Q-value είναι η βέλτιστη συνάρτηση αξίας της αντικειμενικής μας συνάρτησης, γραμμένη ως μια συνάρτηση των Q-states. Ορίζουμε ως Q*(s,a) την αναμενόμενη συνολική αξία, ξεκινώντας από την κατάσταση s και επιλέγοντας κάποια από τις διαθέσιμες ενέργειες a $\in A(s)$, δεδομένης βέλτιστης συμπεριφοράς από την κατάσταση s και ύστερα. Προκύπτει από τους 2 προηγούμενους ορισμούς ότι:

$$V^*(s) = \max_{a \in A(s)} Q^*(s,a) \tag{2.7}$$

$$Q^*(s,a) = \sum_{s' \in S} T(s,a,s')[R(s,a,s') + \gamma V^*(s')] \tag{2.8}$$

Η αντικατάσταση του ενός όρου στον άλλον δίνει μία αναδρομική έκφραση της συνάρτησης αξίας γνωστή και στη βιβλιογραφία ως Bellman Optimality Equation[3]:

$$V^*(s) = \max_{a \in A(s)} \sum_{s' \in S} T(s,a,s')[R(s,a,s') + \gamma V^*(s')] \tag{2.9}$$

καθώς και την αναδρομική έκφραση της συνάρτησης Q-value

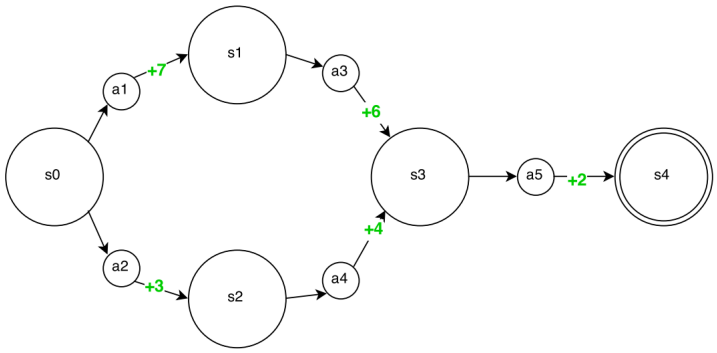
$$Q^*(s,a) = \sum_{s' \in S} T(s,a,s')[R(s,a,s') + \gamma \max_{a' \in A(s)} Q^*(s',a')] \tag{2.10}$$

Τέλος, η συνάρτηση πολιτικής είναι η βέλτιστη ενέργεια συναρτήσει της κάθε κατάστασης:

$$\pi^*(s) = \operatorname*{argmax}_{a \in A(s)} Q^*(s,a) \tag{2.11}$$

Αξίζει εδώ να σημειωθεί η δυϊκότητα μεταξύ του χώρου των συναρτήσεων αξίας και πολιτικής. Αναλυτικότερα, ακολουθώντας μία σταθερή πολιτική εντός κάποιας MDP παράγεται μία αλληλουχία ενεργειών οι οποίες παρέχουν ανταμοιβές. Προφανώς, ο στόχος της βελτιστοποίησης της συνάρτησης στόχου οδηγεί σε προτιμήσεις όσον αφορά τις αλληλουχίες των ενεργειών. Για παράδειγμα, στο παρακάτω σχήμα, ο πράκτορας ο οποίος έχει σαν στόχο τη μεγιστοποίηση του αθροίσματος των ανταμοιβών σε κάθε κίνηση θα προτιμούσε μία πολιτική η οποία θα του έδινε (7,6,2) σε σύγκριση με (3,4,2)

Ως εκ τούτου, οι προτιμήσεις σε σχέση με τις πολιτικές μπορούν να θεωρηθούν ως προτιμήσεις σε σχέση με τις αλληλουχίες ανταμοιβών. Τυπικά, είναι χρονικά αμετάβλητες αν για οποιεσδήποτε δύο περιοδικές ροές ανταμοιβών: (r1,r2,...,rT) και (r1, r2',...rT')



Σχήμα 2.7: MDP με 5 διαφορετικές ενέργειες. Υπάρχουν μόνο δύο ντετερμινιστικές πολιτικές, οι οποίες αντιστοιχούν σε δύο ενέργειες διαθέσιμες στην κατάσταση s_0.

Επίσης, ισχύει ότι

$$(r1, r2, ..., rT) \geqslant (r1, r'2, ..., r'T) \iff (r2, ..., rT) \geqslant (r'2, ..., r'T) \tag{2.12}$$

Ισοδύναμα, συναρτήσει του utility function, παραμένουν σταθερές με το χρόνο αν για οποιεσδήποτε δύο περιοδικές ροές ανταμοιβών ισχύει:

$$U(r1, r2, ..., rT) \geqslant U(r1, r'2, ..., r'T) \iff U(r2, ..., rT) \geqslant U(r'2, ..., r'T) \tag{2.13}$$

Διαισθητικά, η ακινητοποίηση σημαίνει ότι εάν τα πρώτα n στοιχεία των δύο συνολικών ανταμοιβών συσσωρευτικά είναι ίδια, τότε η διαφορά στη χρησιμότητά τους είναι δυνατόν να προσδιοριστεί από τις υπόλοιπες ανταμοιβές. Μία συνέπεια της στατικότητας αυτής είναι ότι:

$$Ut(r1, r2, ..., rT) = U(r1, r2, ..., rT) = V(r1, U(r2, ..., rT)) \tag{2.14}$$

που σημαίνει ότι η ταξινόμηση προτίμησης γίνεται ανεξάρτητα με το χρόνο. Ως συνέπεια αυτού, οι επιλογές είναι επίσης χρονικά ανεξάρτητες. Με απλά λόγια, αν ο πράκτορας προτιμά μία πολιτική π1 έναντι μιας άλλης π2, τότε θα την προτιμά πάντα (ανεξάρτητα από το πέρασμα του χρόνου).

Όπως συζητήθηκε και πριν, εάν κάθε φορά που επισκέπτεται ένας κόμβος και βρισκόμαστε σε κάποια κατάσταση, η πολιτική καθορίζει ντετερμινιστικά μία επιλεγόμενη ενέργεια, τότε πρόκειται για ντετερμινιστική και στατική πολιτική.

Αν από την άλλη, κάθε φορά που βρισκόμαστε σε μία κατάσταση η πολιτική προσδιορίζει την πιθανοτική κατανομή όλων των διαθέσιμων ενεργειών, τότε πρόκειται για στατική αλλά στοχαστική πολιτική.

Οι στατικές πολιτικές είναι σημαντικοί για τους ακόλουθους λόγους: Ο David Blackwell έδειξε ότι αν οι ίδιες ενέργειες είναι διαθέσιμες σε κάθε κατάσταση, τότε υπάρχει μία βέλτιστη

ντετερμινιστική στατική πολιτική [5]. Επίσης, στη διδακτορικη του διατριβή ο Ralph Strauch έδειξε ότι σε οποιοδήποτε πρόβλημα δυναμικού προγραμματισμού με αρνητική ή μειούμενη ανταμοιβή, εάν υπάρχει βέλτιστη πολιτική τότε υπάρχει και βέλτιστη πολιτική η οποία είναι στατική.

Ο Backwell στη συνέχεια έκανε γενίκευση αυτής της ιδέας για όλα τα προβλήματα δυναμικού προγραμματισμού. Ο Sheldon Ross στη συνέχεια απέδειξε ότι για κάθε πρόβλημα MDP υπάρχει μία στατική βέλτιστη πολιτική [14].

Το τελευταίο αυτό αποτέλεσμα πηγάζει από το γεγονός ότι, λόγω της ικανοποίησης της ιδιότητας Markov, η περιγραφή της παρούσας κατάστασης στην οποία βρίσκεται ο πράκτορας εμπεριέχει όλη την απαραίτητη πληροφορία για την λήψη αποφάσεων. Επομένως, δοθείσας της κατάστασης, υπάρχει πάντα μία χρονικά ανεξάρτητη βέλτιστη ενέργεια. Εφόσον αυτή η ιδιότητα ισχύει για όλες τις καταστάσεις, η βέλτιστη πολιτική θα είναι πάντα η ίδια.

Η σημασία του παραπάνω συμπεράσματος είναι εμφανής κατά την εύρεση λύσης στο πρόβλημα απόφασης Markov (Markov Decision Problem). Η αναζήτηση βέλτιστης πολιτικής η οποία είναι περιορισμένη σε ένα σύνολο στατικών πολιτικών δεν βλάπτει τη γενικότητα. Ως εκ τούτου, είναι εύκολο να επιλέξουμε μία συνάρτηση η οποία οδηγεί στη στασιμότητα των προτιμήσεων με την πάροδο του χρόνου. Δύο ειδών συναρτήσεις υποστηρίζουν στατικές προτιμήσεις:

- Additive utility: $U(r0, r1, r2, ..., rn) = r0 + r1 + r2 + ... + rn$
- Discounted utility: $U(r0, r1, r2, ..., rn) = r0 + \gamma * r1 + \gamma^2 * r2 + ... + \gamma^n * rn$

Ο λόγος που η δεύτερη εξίσωση οδηγεί σε στατικές προτιμήσεις είναι επειδή η καθυστέρηση έχει το ίδιο αντίκτυπο στη χρησιμότητα ανεξάρτητα από το σημείο εκκίνησης. Η πρώτη εξίσωση είναι ειδική περίπτωση της δεύτερης για γ=1. Γενικά το discounted utility αυξάνεται με επιβραδυνόμενο ρυθμό και επομένως είναι φραγμένη συνάρτηση. Ως μοντέλο συμπεριφέρεται "καλύτερα" σε σχέση με τη μη-φραγμένη αθροιστική χρησιμότητα (utility).

Θεωρώντας μία αρχική κατάσταση s, ενέργεια a και θεωρώντας ότι ακολουθείται η πολιτική π σε έναν άπειρο χρονικό ορίζοντα, χρησιμοποιούνται δύο μετρικές όσον αφορά την απόδοση του utility function:

1. Αναμενόμενο άθροισμα discounted rewards

$$\psi_\pi(s) = \lim_{l\to\infty} E[\sum_{k=1}^{l} \gamma^{k-1} r(x_k), \pi(x_k), x_{k+1}|_{x_1=s}] \tag{2.15}$$

2. Αναμενόμενος μέσος όρος ανταμοιβής:

$$\rho_\pi(s) = \lim_{l\to\infty} E[\sum_{k=1}^{l} \gamma^{k-1} r(x_k), \pi(x_k), x_{k+1}|_{x_1=s}]/l \tag{2.16}$$

όπου:

$0 < \gamma < 1$ είναι ο συντελεστής έκπτωσης/μείωσης

x_k είναι η κατάσταση πριν την κ-οστή μετάβαση

Ε είναι ο τελεστής αναμενόμενης τιμής

Επομένως, το λεγόμενο "Markov Decision Problem" μεταφράζεται στην εύρεση της κατάλληλης πολιτικής π* έτσι ώστε:

$$\psi_\pi^*(s) \geqslant \psi_\pi(s), \forall s \in S, \forall \pi \tag{2.17}$$

και

$$\rho_\pi^*(s) \geqslant, \forall s \in S, \forall \pi \tag{2.18}$$

Οι μετρικές υπολογισμού μέσου όρου ανταμοιβών σταθμίζουν εξίσου την κάθε ανταμοιβή ανεξάρτητα από το πότε εμφανίστηκαν. Οι μετρικές υπολογισμού ανταμοιβής που φθίνουν με το χρόνο θεωρούν τις βραχυπρόθεσμες ανταμοιβές σημαντικότερες από της μακροπρόθεσμες. Αυτό προέρχεται από την χρηματοοικονομική έννοια της απόδοσης και στηρίζεται στην ιδέα ότι η κατοχή χρήματος στο μέλλον αξίζει λιγότερο από το ίδιο ποσό χρήματος αυτήν την στιγμή, λόγω του κόστους ευκαιρίας. Συγκεκριμένα,ο συντελεστής γ υποδηλώνει τον παράγοντα με τον οποίον πολλαπλασιάζεται μία μελλοντική τιμή για τον προσδιορισμό της τιμής της.[7]

2.4.3 Value Iteration

Η επαναληπτική μέθοδος υπολογισμού αξίας υπολογίζει την βέλτιστη συνάρτηση αξίας βελτιώνοντας την εκτίμηση του V(s) με μία επαναληπτική μέθοδο. Ο αλγόριθμος αρχικοποιεί την V(s) με αυθαίρετες τυχαίες τιμές. Επαναλαμβάνει την ανανέωση των Q(s,a) και V(s) μέχρι να συγκλίνουν οι τιμές τους. Ο αλγόριθμος αυτός εμφανίζεται στον παρακάτω ψευδοκώδικα:

Initialize $V(s)$ to arbitrary values
Repeat
  For all $s \in S$
    For all $a \in \mathcal{A}$
      $Q(s,a) \leftarrow E[r|s,a] + \gamma \sum_{s' \in S} P(s'|s,a)V(s')$
    $V(s) \leftarrow \max_a Q(s,a)$
Until $V(s)$ converge

Σχήμα 2.8: Ψευδοκώδικας αλγορίθμου Value Iteration. [1]

2.4.4 Policy Iteration

Η επαναληπτική μέθοδος υπολογισμού αξίας συνεχίζει να βελτιώνει τη συνάρτηση τιμών σε κάθε επανάληψη μέχρι να επέλθει σύγκλιση. Ο πράκτορας, παρόλα αυτά, ενδιαφέρεται αποκλειστικά για την εύρεση βέλτιστης πολιτικής η οποία ενδεχομένως να συγκλίνει πριν την σύγκλιση της συνάρτησης αξίας. Συνεπώς, ο αλγόριθμος αναζήτησης βέλτιστης πολιτικής αντί να επαναλαμβάνει τη βελτίωση της εκτίμησης της συνάρτησης αξίας, επαναπροσδιορίζει την πολιτική σε κάθε βήμα και υπολογίζει την αξία σύμφωνα με τη νέα πολιτική έως ότου επέλθει η σύγκλιση. Η επαναληπτική μέθοδος υπολογισμού πολιτικής εγγυάται σύγκλιση και απαιτεί συχνά λιγότερες επαναλήψεις από την επαναληπτική μέθοδο υπολογισμού συνάρτησης αξίας. Ο αλγόριθμος αυτός εμφανίζεται στον παρακάτω ψευδοκώδικα:

Initialize a policy π' arbitrarily
Repeat
    $\pi \leftarrow \pi'$
    Compute the values using π by
        solving the linear equations
            $V^{\pi}(s) = E[r|s, \pi(s)] + \gamma \sum_{s' \in S} P(s'|s, \pi(s)) V^{\pi}(s')$
    Improve the policy at each state
        $\pi'(s) \leftarrow \arg\max_a (E[r|s, a] + \gamma \sum_{s' \in S} P(s'|s, a) V^{\pi}(s'))$
Until $\pi = \pi'$

Σχήμα 2.9: Ψευδοκώδικας αλγορίθμου Policy Iteration. [1]

2.4.5 Temporal Difference Learning

Η διαδικασία μάθησης TD συνδυάζει ιδέες από Monte Carlo και δυναμικό προγραμματισμό. Όπως και στις μεθόδους Monte Carlo, οι μέθοδοι Temportal Difference μαθαίνουν απευθείας από εμπειρία χωρίς την ανάγκη μοντέλου της δυναμικής του περιβάλλοντος. Όπως και στον δυναμικό προγραμματισμό, οι μέθοδοι TD ανανεώνουν τις εκτιμήσεις τους βασιζόμενοι εν μέρη σε άλλες εκτιμήσεις χωρίς να περιμένουν το τελικό αποτέλεσμα. Η σχέση μεταξύ μεθόδων TD, DP και Monte Carlo καθώς και η σύγκριση μεταξύ τους γίνεται συχνά στη θεωρία του Reinforcement Learning. Συγκεκριμένα, ο αλγόριθμος TD(λ) συνδιάζει τα τρία αυτά μεταξύ τους. Ως συνήθως, επικεντρωνόμαστε στο policy evaluation ή αλλιώς στο πρόβλημα πρόβλεψης, εκείνου του προβλήματος δηλαδή της εκτίμησης της συνάρτησης αξίας u_π για μία δεδομένη πολιτική π. Για το πρόβλημα ελέγχου (δηλαδή εύρεσης βέλτιστης πολιτικής), οι μέθοδοι DP, TD και Monte Carlo όλες χρησιμοποιούν κάποια παραλλαγή της γενικής επαναληπτικής μεθόδου υπολογισμού πολιτικής (generalized policy iteration). Οι διαφορές στις μεθόδους είναι κυρίως διαφορές στις προσεγγίσεις τους στην πρόβλεψη.

Εμειρία χρησιμοποιείται και από τις δύο μεθόδους (TD και Monte Carlo) για την επίλυση του προβλήματος της πρόβλεψης. Δεδομένης κάποιας εμπειρίας και ακολουθώντας την πολιτική π, οι μέθοδοι TD και Monte Carlo ανανεώνουν τις εκτιμήσεις υ της u_π τις μη-τελικές

Input: the policy π to be evaluated
Initialize $V(s)$ arbitrarily (e.g., $V(s) = 0, \forall s \in \mathcal{S}^+$)
Repeat (for each episode):
  Initialize S
  Repeat (for each step of episode):
    $A \leftarrow$ action given by π for S
    Take action A; observe reward, R, and next state, S'
    $V(S) \leftarrow V(S) + \alpha\big[R + \gamma V(S') - V(S)\big]$
    $S \leftarrow S'$
  until S is terminal

Σχήμα 2.10: Ψευδοκώδικας TD για τον υπολογισμό u_π

καταστάσεις S_t που εμφανίζονται σε αυτήν την εμπειρία. Σε γενικές γραμμές, οι μέθοδοι Monte Carlo περιμένουν έως ότου γίνει γνωστή η επιστροφή μετά την επίσκεψη γίνει γνωστή και στη συνέχεια χρησιμοποιούν αυτήν την επιστροφή ως στόχο της $V(S_t)$. Μία πολύ απλή 'every-visit' μέθοδος Monte Carlo κατάλληλη για ένα δυναμικό περιβάλλον είναι:

$$V(S_t) \leftarrow V(S_t) + a[G_t - V(S_t)] \tag{2.19}$$

όπου G_t είναι η πραγματική επιστροφή μετά τη χρονική στιγμή t, a είναι μία σταθερά της παραμέτρου step-size. Ας αποκαλέσουμε αυτή τη μέθοδο 'constant-a' MC [13]. Ενώ οι μέθοδοι Monte Carlo πρέπει να περιμένουν μέχρι το τέλος του επεισοδίου για να αποφανθούν για την μεταβολή στο $V(S_t)$ (μόνο δηλαδή όταν είναι γνωστό το G_t), οι μέθοδοι TD χρειάζεται να να περιμένουν μόνο μέχρι το επόμενο βήμα. Τη χρονική στιγμή $t+1$ σχηματίζουν αμέσως έναν στόχο και πραγματοποιείται ενημέρωση χρησιμοποιώντας την παρατηρούμενη ανταμοιβή R_{t+1} και τις εκτιμήσεις $V(S_{t+1})$. Οι πιο απλή μέθοδος TD, γνωστή και ως $TD(0)$ ορίζεται:

$$V(S_t) \leftarrow V(S_t) + a[R_{t+1} + \gamma V(S_{t+1}) - V(S_t)][13]. \tag{2.20}$$

Ο στόχος λοιπόν στο Monte Carlo είναι G_t, ενώ ο στόχος στη TD είναι $R_{t+1} + \gamma V(S_{t+1})$.

Λόγω του ότι η μέθοδος ΤΔ βασίζει την αναβάθμιση της πρόβλεψης μερικώς στην παρούσα κατάσταση, λέμε ότι είναι μία μέθοδος bootstrapping, όπως και στον δυναμικό προγραμματισμό. Παρακάτω παρουσιάζεται ο ψευδοκώδικας του αλγορίθμου TD(0):

Κεφάλαιο 3

Ο Κύβος του Ρούμπικ

3.1 Εισαγωγή

Ο κύβος του Ρούμπικ (Rubik's Cube) είναι ένα τρισδιάστατο παζλ που εφευρέθηκε το 1974 από τον Ούγγρο γλύπτη και καθηγητή Erno Rubik. Μέχρι το 2009 έχουν πουληθεί παγκοσμίως 350 εκατομμύρια κύβοι και ως εκ τούτου, θεωρείται ευρέως το κορυφαίο παιχνίδι σε πωλήσεις παγκοσμίως.

Στον αρχικό κλασικό κύβο του Ρούμπικ, κάθε μια από τις 6 έδρες καλύπτεται από εννέα αυτοκόλλητα με τα εξής χρώματα: λευκό, κόκκινο, μπλε, πορτοκαλί, πράσινο και κίτρινο. Η τρέχουσα έκδοση του κύβου έχει πάντα αντιδιαμετρικά τα χρώματα που γειτνιάζουν στο ορατό φάσμα. Ένας εσωτερικός μηχανισμός περιστροφής επιτρέπει σε κάθε πλευρά να περιστρέφεται ανεξάρτητα, αναμιγνύοντας έτσι τα χρώματα. Για να θεωρείται λυμένο το παζλ, κάθε επιφάνεια θα πρέπει να έχει αυτοκόλλητα μόνο ενός χρώματος. Λόγω της διαχρονικότητας του ως παζλ, ο παγκόσμιος οργανισμός του κύβου του Ρούμπικ (World Cube Association), ο οποίος θεωρείται το διεθνές κυβερνητικό σώμα για σχετικά θέματα, εξακολουθεί να διοργανώνει διαγωνισμούς και αναγνωρίζει παγκόσμια ρεκόρ.

Ο κύβος του Ρούμπικ, αν και φαινομενικά απλός ως προς τη μηχανική του και τον στόχο (να ταιριάξουμε κάθε πλευρά με τα σωστά χρώματα) κρύβει παγίδες ως προς την αντιμετώπισή του στον τομέα του reinforcement learning. Αξίζει να τονιστεί ξανά ότι ο χώρος καταστάσεων στον οποίο κινούμαστε είναι σχεδόν άπειρος, οπότε δεν υπάρχει ελπίδα εύρεσης λύσεων με brute force.

Υπάρχουν αρκετοί κλάδοι μαθηματικών που εξετάζουν τέτοιου είδους προβλήματα. Τα αφηρημένα μαθηματικά και η άλγεβρα εξετάζουν αφηρημένα σύνολα αντικειμένων πάνω στα οποία δρουν τελεστές και μετασχηματισμοί. Ο κύβος λοιπόν του Ρούμπικ αποτελεί ένα παράδειγμα θεωρίας ομάδων. Άλλες γνωστές αλγεβρικές δομές, όπως οι δακτύλιοι, τα σώματα, και οι διανυσματικοί χώροι, μπορούν να αντιμετωπιστούν σαν ομάδες που έχουν εφοδιαστεί με επιπρόσθετες πράξεις και αξιώματα. Οι ομάδες συναντώνται επανειλημμένα σε όλο το φάσμα των μαθηματικών, και οι μέθοδοι της θεωρίας ομάδων έχουν επηρεάσει πολλούς τομείς της

άλγεβρας.

Το πρόβλημα με τον κύβο δεν είναι οι διαδοχικοί μετασχηματισμοί καταστάσεων αλλά ότι ο στόχος είναι να φέρουμε τον κύβο στην αρχική (λυμένη) διάταξη, ανεξαρτήτως αρχικών συνθηκών. Προβλήματα τέτοιου τύπου μελετώνται στον κλάδο μαθηματικών που λέγεται συνδυαστική βελτιστοποίηση. Η Συνδυαστική βελτιστοποίηση είναι η μελέτη της βελτιστοποίησης σε διακριτά και συνδυαστικά αντικείμενα. Ξεκίνησε ως ένα μέρος της Συνδυαστικής και της Θεωρίας Γραφημάτων, αλλά θεωρείται πλέον ως κλάδος των εφαρμοσμένων μαθηματικών και της επιστήμης των υπολογιστών που σχετίζεται με την επιχειρησιακή έρευνα, τη θεωρία αλγορίθμων και τη θεωρία πολυπλοκότητας. Εφαρμόζεται σε αρκετά διάσημα προβλήματα όπως

- Το πρόβλημα του πλανόδιου πωλητή
- Προσομοίωση αναδίπλωσης πρωτεϊνών
- Προβλήματα κατανομής πόρων - Αλυσίδες εφοδιασμού

Κοινός παρονομαστής όλων αυτών των προβλημάτων είναι ο τεράστιος χώρος καταστάσεων που καθιστά σχεδόν αδύνατο τον έλεγχο όλων των συνδυασμών για την εύρεση βέλτιστης λύσης. Ο κύβος του Ρούμπικ ανήκει στην ίδια αυτή κατηγορία.

3.2 Μαθηματική Ανάλυση

3.2.1 Διατάξεις

Ο κλασικός 3x3x3 κύβος του Ρούμπικ έχει οκτώ γωνίες και δώδεκα πλευρές. Ως σύμβαση θα ονομάζουμε τα γωνιακά κομμάτια ως corner-cubelets και τα κομμάτια στις ακμές του κύβου ως edge-cubelets. Υπάρχουν λοιπόν 8! τρόποι να διατάξουμε τα corner-cubelets. Κάθε γωνία έχει τρεις πιθανούς προσανατολισμούς στο χώρο. Παρ'όλα αυτά, μόνο τα 7 από αυτές μπορούν να προσανατολιστούν ανεξάρτητα. Ο προσανατολισμός της 8ης γωνίας προκύπτει συναρτήσει των άλλων προσανατολισμών. Έτσι, έχουμε 2187 διαφορετικούς συνδυασμούς προσανατολισμών. Υπάρχουν επίσης, 12!/2 (239,500,800) τρόποι να τοποθετήσουμε τα corner-cubelets. Περνώντας στα edge-cubelets, οι έντεκα από τους δώδεκα προσανατολισμούς ορίζονται ανεξάρτητα και η τελευταία προκύπτει συναρτήσει των άλλων έντεκα. Μας δίνεται λοιπόν η ελευθερία για την αντιμετώπιση 2048 περιπτώσεων. Το σύνολο λοιπόν των δυνατών διατάξεων συνδυάζοντας τα προηγούμενα είναι:

$8! * 3^7 * (12!/2) * 2^{11} = 43.252.003.274.489.856.000$

Για να γίνει η σύλληψη του μεγέθους του αριθμού αυτού, αν υπήρχε ένας πραγματικός κύβος για κάθε δυνατή διάταξη, θα μπορούσαν συνολικά να καλύψουν την επιφάνεια της γης 275 φορές. Αξίζει εδώ να σημειωθεί ότι ο προηγούμενος αριθμός αντιπροσωπεύει όλες τις δυνατές διατάξεις που μπορούν να αναπαρασταθούν από διαδοχικές περιστροφές του κύβου

ξεκινώντας από την λυμένη κατάσταση. Αν ληφθούν υπόψη οι διατάξεις που μπορούν να κατασκευαστούν μέσω αποσυναρμολόγησης του κύβου, ο αριθμός αυτός γίνεται περίπου 12 φορές μεγαλύτερος.

$8! * 3^8 * 12! * 2^{12} = 519.024.039.293.878.272.000$

Αυτό σημαίνει ότι μόνο 1/12 από αυτές τις διατάξεις είναι επιλύσιμες. Αυτό συμβαίνει επειδή δεν υπάρχει αλληλουχία κινήσεων η οποία μπορεί να αλλάξει τον προσανατολισμό μονάχα ενός cubelet. Υπάρχουν λοιπόν δώδεκα πιθανά σύνολα προσβάσιμων διατάξεων, αποκαλούμενα μερικές φορές "universes" ή "orbits" μέσα στα οποία μπορεί να τοποθετηθεί ο κύβος μέσω αποσυναρμολόγησης και επανασυναρμολόγησης. Τέλος, σημειώνεται ότι οι προηγούμενοι αριθμοί προκύπτουν ύστερα από υπόθεση ότι τα κεντρικά cubelets δεν αλλάζουν θέση λόγω κατασκευαστικής ιδιότητας του κύβου. Στην περίπτωση άρσης αυτού του περιορισμού, οι δυνατές διατάξεις πολλαπλασιάζονται επί 24.

3.2.2 Συσχετισμός με αλγορίθμους

Στη γλώσσα των ανθρώπων που ασχολούνται με την επίλυση του κύβου, μία ακολουθία κινήσεων η οποία μας οδηγεί σε κάποιο επιθυμητό αποτέλεσμα ονομάζεται αλγόριθμος επίλυσης. Η ορολογία αυτή προέρχεται από τη χρήση της έννοιας του αλγορίθμου στα μαθηματικά η οποία ορίζεται ως μία λίστα από καλώς ορισμένων οδηγιών οι οποίες αν εκτελεστούν σε κάποια αρχική κατάσταση, οδηγούν σε μία επιθυμητή τελική κατάσταση. Κάθε μέθοδος επίλυσης του κύβου του Ρούμπικ χρησιμοποιεί το δικό του σύνολο αλγορίθμων οι οποίοι έχουν τη δυνατότητα να οδηγήσουν τον κύβο προς τη λύση του. Οι περισσότεροι αλγόριθμοι που χρησιμοποιούνται από ανθρώπους έχουν την ιδιότητα να αλλάζουν μερικές υποδιατάξεις σε κάποιο υποσύνολο του κύβου, αφήνοντας τον υπόλοιπο κύβο ανεπηρέαστο. Αυτό πραγματοποιείται με το σκεπτικό να εφαρμόζονται διαδοχικά αλγόριθμοι σε διαφορετικά μέρη του κύβου έτσι ώστε στο τέλος να λυθεί ολόκληρος.

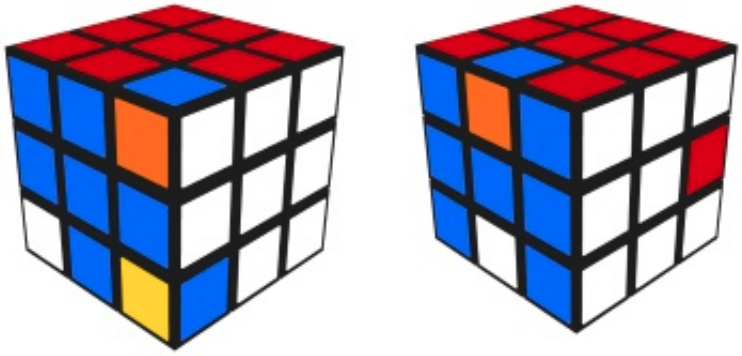
Μερικοί αλγόριθμοι έχουν μαζί με την επιθυμητή επίδραση στον κύβο (π.χ. αλλαγή θέσης γωνίας) και κάποιες παρενέργειες. Οι αλγόριθμοι αυτοί είναι συνήθως πιο απλοί από αυτούς που δεν έχουν παρενέργειες και χρησιμοποιούνται συνήθως κατά τα αρχικά στάδια της επίλυσης του κύβου καθώς εφόσον το μεγαλύτερο μέρος του παζλ είναι άλυτο οπότε οι παρενέργειες δεν δημιουργούν πρόβλημα. Προς το τέλος της επίλυσης χρησιμοποιούνται συνήθως πιο εξειδικευμένοι αλγόριθμοι.

3.2.3 Συσχετισμός με τη θεωρία ομάδων

Ο κύβος του Ρούμπικ αποτελεί μία εφαρμογή της θεωρίας ομάδων η οποία αξιοποιείται στην έκφραση ορισμένων αλγορίθμων, συγκεκριμένα σε αλγορίθμους της μορφής X Y X' Y', (commutators) και X Y X' (conjugate structures). Τα και αποτελούν αλληλουχίες κινήσεων και τα ' ' οι αντίστοιχες αντίστροφες αλληλουχίες.

1. Three Cycle - Commutator

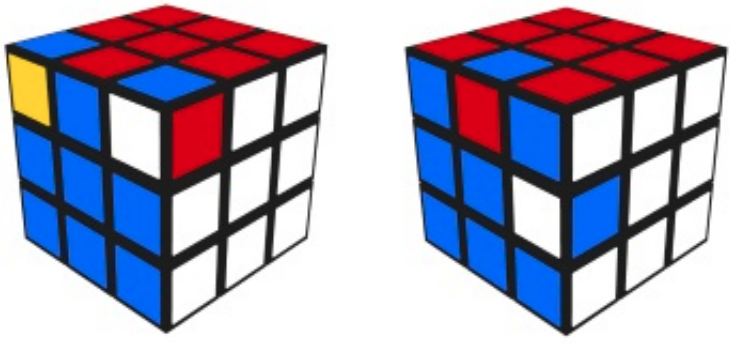
   Ο αλγόριθμος three-cycle μετακινεί κυκλικά 3 corner-cubelet ή edge-cubelet. Προϋποθέτει τα 2 από τα 3 cubelet να βρίσκονται στο ίδιο επίπεδο και το τρίτο cubelet να μην βρίσκεται στο επίπεδο που ορίζεται από τα άλλα 2 cubelets. Ονομάζουμε με X το σύνολο κινήσεων που χρειαζόμαστε για να εισάγουμε το cubelet που δεν ανήκει στο επίπεδο αναφοράς στη σωστή του θέση. Ο αλγόριθμος απαγορεύεται να αλλάξει οτιδήποτε πέρα από την αλλαγή αυτή στο επίπεδο αναφοράς. Με ονομάζουμε την αλληλουχία κινήσεων οι οποίες μετακινούν το το cubelet που δεν ανήκει στο επίπεδο αναφοράς στην υποδοχή που μόλις δημιουργήθηκε από την μετακίνηση του σωστού cubelet στο επίπεδο αναφοράς.



Σχήμα 3.1: Παρουσίαση cubelets που επηρεάζονται κατά την κυκλική μετακίνηση τους

2. Orientation Swap - Commutator

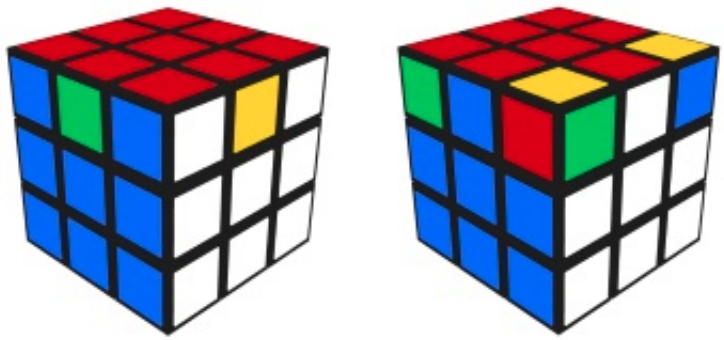
   Η αλλαγή προσανατολισμού ακριβώς δύο cubelet κατατάσσεται στους αλγορίθμους orientation swap. Προϋποθέτει και τα δύο cubelets να βρίσκονται στο ίδιο επίπεδο. Το πρώτο κομμάτι (ο αλγόριθμος) παίρνει ένα cubelet και το τοποθετεί ξανά στην ίδια θέση αλλά με σωστό προσανατολισμό. Υπενθυμίζεται ξανά ότι απαγορεύεται η μεταβολή θέσεων και προσανατολισμών άλλων cubelets στο ίδιο επίπεδο. Η αλληλουχία Y μετακινεί τη το δεύτερο cubelet στη σωστή του θέση.



Σχήμα 3.2: Παρουσίαση cubelets που επηρεάζονται κατά την αλλαγή προσανατολισμού

3. Pair Swap - Commutator

   Αυτού του είδους οι αλγόριθμοι αντιμεταθέτει δύο ζεύγη cubelet. Δεν είναι τόσο χρήσιμοι καθώς σε αρκετές περιπτώσεις αρκεί ένα πλήθος three-cycle κινήσεων για την επίλυση μεγάλου μέρους του κύβου. Η μόνη προϋπόθεση για την εκτέλεσή τους είναι ότι και τα τέσσερα cubelets οφείλουν να βρίσκονται στο ίδιο επίπεδο.



Σχήμα 3.3: Παρουσιάση cubelets που επηρεάζονται κατά την αλλαγή προσανατολισμού

3.2.4 Η Ομάδα του Κύβου

Η θεωρία ομάδων είναι το πεδίο των μαθηματικών που μελετά τις αλγεβρικές δομές γνωστές ως ομάδες.

Μια ομάδα είναι μια αλγεβρική δομή (G,•) που αποτελείται από ένα σύνολο στοιχείων G μαζί με μια συνάρτηση •, που συνδυάζει οποιαδήποτε δύο στοιχεία ώστε να σχηματιστεί ένα τρίτο στοιχείο. Μια ομάδα πρέπει να ικανοποιεί τις εξής συνθήκες:

- Κλειστότητα, για κάθε a, b στο G το στοιχείο ab που προκύπτει από την πράξη a • b πρέπει να ανήκει στο G.
- Προσεταιριστική ιδιότητα, για κάθε a, b, c στο G ισχύει (a • b) • c = a • (b • c).
- Ουδέτερο στοιχείο, υπάρχει μοναδικό στοιχείο e ώστε e • a = a • e = a.
- Αντίστροφο στοιχείο, για κάθε στοιχείο a υπάρχει αντίστοιχο στοιχείο b στο G ώστε a • b = b • a = e

Μια από τις κατηγορίες ομάδων είναι αυτή των κυκλικών μεταθέσεων στην οποία ανήκει ο κύβος του Ρούμπικ. Η ομάδα του κύβου αποτελείται από το ζεύγος (G,•), όπου G είναι το σύνολο όλων των κινήσεων/περιστροφών που επιτρέπονται στον κύβο και η συνάρτηση • είναι η σύνθεση των κινήσεων του κύβου και αντιστοιχεί στην κατάσταση που παράγεται με την σειριακή εκτέλεσή τους. Με αυτό τον τρόπο μπορούμε να αναπαραστήσουμε κάθε κατάσταση ενός κύβου ως μια αλληλουχία κινήσεων σε σχέση με την αρχική του κατάσταση, περιγράφοντας τις περιστροφές που πρέπει να εφαρμοστούν ώστε να επιλυθεί ο κύβος.

3.3 Ντετερμινιστικοί Αλγόριθμοι Επίλυσης

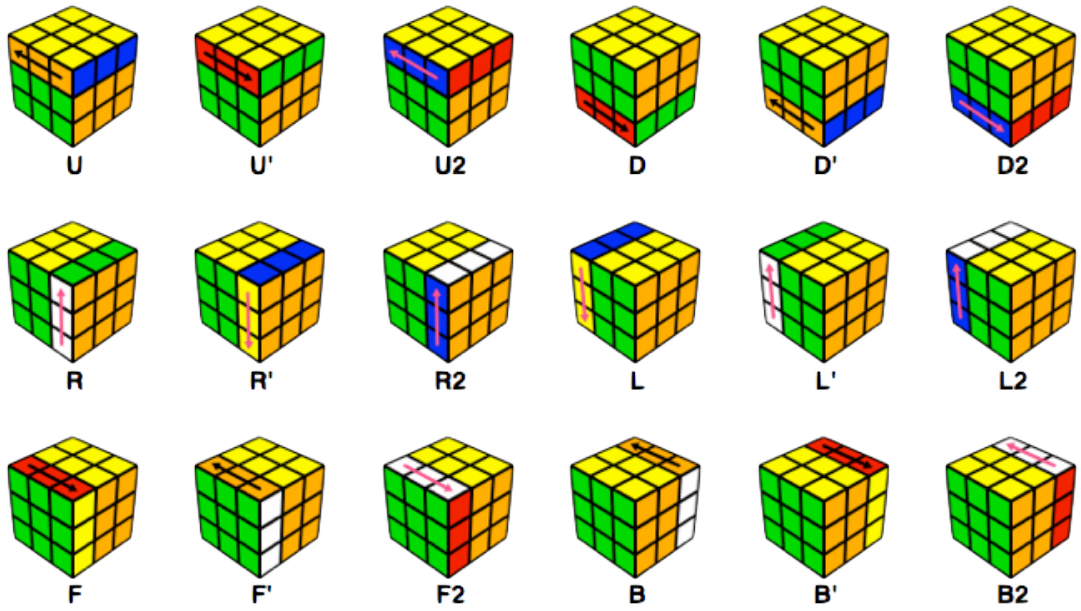
Στην παράγραφο αυτή εξετάζονται διάφοροι γνωστοί αλγόριθμοι επίλυσης του κύβου. Προκειμένου να αναπαρασταθεί μία αλληλουχία περιστροφών και για να εκφραστεί κάθε αλγόριθμος λεκτικά, υπάρχει μία συγκεκριμένη σημειογραφία κινήσεων που χρησιμοποιείται.

3.3.1 Σημειογραφία Κινήσεων

Η σημειογραφία που χρησιμοποιείται κατά κόρων για την αναπαράσταση των κινήσεων στον κλασικό 3x3x3 κύβο του Ρούμπικ αναπτύχθηκε από τον David Singmaster και αναφέρεται στη βιβλιογραφία ως "σημειογραφία Singmaster"[10]. Η τρόπος γραφής των κινήσεων επιτρέπει την εκτέλεση κινήσεων ανεξάρτητα από τη στερεοδιάταξη του κύβου στο χώρο.

- F (Front): the side currently facing the solver
- B (Back): the side opposite the front
- U (Up): the side above or on top of the front side
- D (Down): the side opposite the top, underneath the Cube
- L (Left): the side directly to the left of the front
- R (Right): the side directly to the right of the front
- f (Front two layers): the side facing the solver and the corresponding middle layer
- b (Back two layers): the side opposite the front and the corresponding middle layer
- u (Up two layers): the top side and the corresponding middle layer
- d (Down two layers): the bottom layer and the corresponding middle layer
- l (Left two layers): the side to the left of the front and the corresponding middle layer
- r (Right two layers): the side to the right of the front and the corresponding middle layer
- x (rotate): rotate the entire Cube on R
- y (rotate): rotate the entire Cube on U
- z (rotate): rotate the entire Cube on F

Ο συμβολισμός του τόνου ενδέχεται να υπάρχει αριστερά του βασικού συμβόλου (π.χ. F') και υποδηλώνει αντιωρολογιακή φορά. Η έλλειψη τόνου υπονοεί ωρολογιακή φορά περιστροφής. Το σύμβολο 2 που ενδεχομένως να υπάρχει δεξιά του συμβολισμού υποδεικνύει



Σχήμα 3.4: Σχηματική παρουσίαση της σημειογραφίας Singmaster

περιστροφή κατά 180 μοίρες, κάτι που κάνει περιττή τη σήμανση ωρολογιακής ή αντιωρολογιακής φοράς. Τα x, y, z χρησιμοποιούνται να υποδείξουν περιστροφή ολόκληρου του κύβου κατά τα R, U, και F αντίστοιχα. Το αντίστοιχο σύμβολο στο τετράγωνο υποδεικνύει περιστροφή κατά 180 μοίρες.

3.3.2 Μέθοδος Αρχαρίου & Jessica Fridrich

Σήμερα, υπάρχουν πολλοί διαφορετικοί τρόποι επίλυσης του κύβου του Ρούμπικ. Μερικά από αυτά είναι η μέθοδος για αρχάριους, η μέθοδος της Jessica Fridrich (για πιο προχωρημένους) κ.α. Όλες οι μέθοδοι απαιτούν διαφορετικό αριθμό βημάτων μέχρι την επίλυση. Η μέθοδος των αρχαρίων απαιτεί κατά μέσο όρο 100-120 κινήσεις, χρησιμοποιώντας μία αλληλουχία 7 διαφορετικών αλγορίθμων οι οποίοι έχουν ως χαρακτηριστικό να αλλάζουν ένα μέρος του κύβου, διατηρώντας ένα άλλο πάντα σταθερό. Όσο περισσότερο κάποιος ασχολείται με την επίλυση του κύβου του Ρούμπικ, τόσους περισσότερους αλγορίθμους απομνημονεύει, και τόσο περισσότερο ενισχύει το "οπλοστάσιό" του και την αποδοτικότητα ως προς την επίλυση. Η μέθοδος της Jessica Fridrich η οποία απευθύνεται σε άτομα που ασχολούνται χρόνια με την ταχεία επίλυση κύβων απαιτεί περίπου 50 κινήσεις. Το παγκόσμιο ρεκόρ το 2016 κατείχε ο Feliks Zemdegs ο οποίος μέσα σε 35 κινήσεις έλυσε τον κύβο σε 4.74 sec.

Σαφώς, το εύλογο ερώτημα είναι ποια είναι η μικρότερη ακολουθία κινήσεων για την επίλυση οποιουδήποτε κύβου. Μετά από 54 χρόνια ενασχόλησης από την επιστημονική κοινότητα δεν έχουμε ακόμα ξεκάθαρη απάντηση. Ενώ λοιπόν γνωρίζουμε τον ελάχιστο αριθμό κινήσεων, η εύρεση της λύσης δεν είναι τετριμμένη.

3.4 Θεωρητική Βέλτιστη Λύση

Οι βέλτιστες λύσεις για τον κύβο του Rubik αναφέρονται σε λύσεις που είναι ελάχιστες σε αλληλουχία κινήσεων.

Υπάρχουν δύο τρόποι μέτρησης του βέλτιστου μήκους μιας λύσης. Ο πρώτος λογαριάζει κάθε περιστροφή της εξωτερικής πλευράς κατά 90 μοίρες. Ο δεύτερος λογαριάζει κάθε περιστροφή των εξωτερικών στρωμάτων χωρίς να είναι διαφοροποιείται μία περιστροφή 90 ή 180 μοιρών. Αυτές οι περιστροφές ονομάζονται "Face Turns" στη διεθνή ορολογία. Εάν λοιπόν περιστραφεί κάποια εξωτερική επιφάνεια κατά 180 μοίρες, λογαριάζεται ως δύο κινήσεις στην μετρική QTM "Quarter-Turn Metric" αλλά ως μία κίνηση στην μετρική HTM "Half Turn Metric". Η τελευταία επίσης ονομάζεται και OBTM "Outer Block Turn Metric"

Ο μέγιστος αριθμός κινήσεων σύμφωνα με τις μετρικές HTM και QTM είναι 20 και 26 αντίστοιχα. Οι αριθμοί αυτοί αποτελούν και διαμέτρους του αντίστοιχοι γράφου Cayley της μαθηματικής ομάδας του κύβου του Ρούμπικ.

3.4.1 Γράφημα Cayley του Κύβου

Έστω G η προηγούμενη ομάδα που παράγεται από τα στοιχεία $K = g1, \ldots, gn$:

$G =< g_1, g_2, ..., g_n >$ υποομάδα S_X

Το γράφημα Cayley του G συσχετιζόμενο με το είναι το γράφημα (V,E) του οποίου οι κορυφές V είναι τα στοιχεία του G και οι ακμές καθορίζονται από την ακόλουθη συνθήκη:

Αν x και y ανήκουν στο V=G τότε υπάρχει ακμή από το x στο y (και από το y στο x) αν και μόνο αν $y = g_i * y$ για κάποιο $i = 1, 2, ..., n$

3.4.2 Αριθμός του Θεού

Έστω τώρα G η ομάδα του κύβου του Ρούμπικ και v μία κορυφή στο γράφημα Cayley της G. Αναζητούμε έναν αλγόριθμο για τον προσδιορισμό μιας διαδρομής μεταξύ δύο κόμβων (κορυφών) v και v_0 με μήκος διαδρομής ίσο με την εξ'ορισμού αφηρημένη έννοια της απόστασης των δύο κόμβων. Εάν ισχύει αυτός ο αλγόριθμος, ονομάζεται αλγόριθμος του Θεού και μας επιτρέπει την βέλτιστη μεταπήδηση μεταξύ καταστάσεων στον κύβο. Το μέγιστο μήκος ακολουθίας το οποίο φέρνει οποιαδήποτε από τις 43,252,003,274,489,856,000 διατάξεις του κύβου στην αρχική κατάσταση ορίζεται ως η διάμετρός του. Ως αρχική κατάσταση του κύβου ορίζουμε φυσικά την κατάσταση στην οποία όλες οι 6 έδρες έχουν το ίδιο χρώμα.

Παρατηρούμε ότι αν μας δοθεί ο κύβος λυμένος και ξεκινήσουμε να εκτελούμε τυχαίες κινήσεις, η διάμετρος αυξάνεται. Παρακάτω ακολουθεί μία ιστορική αναδρομή αναζήτησης και μαθηματικής απόδειξης για την αναζήτηση του ορίου αυτού.

- Εξέλιξη κάτω φράγματος

  Στα τέλη της δεκαετίας του ′70, ο βρετανός μαθηματικός David Singmaster (1939-) απέδειξε ότι επαρκούν 277 κινήσεις για την επίλυση του κύβου. Αυτό σημαίνει ότι το 277 είναι ένα άνω φράγμα. Χρησιμοποιώντας 277 κινήσεις, μπορεί να λυθεί οποιοσδήποτε κύβος. Από την δεκαετία του ‘70 και μετά, οι μαθηματικοί έχουν καταφέρει με συστηματική προσπάθεια να βρουν μικρότερα άνω φράγματα. Κατά τις αρχές της δεκαετίας του ‘80 ο Morwen Thistlethwaite διαπίστωσε ότι 52 κινήσεις ήταν αρκετές. Για μεγάλο χρονικό διάστημα ήταν αβέβαιο αν μπορεί να μειωθεί περαιτέρω.

- Εξέλιξη άνω φράγματος

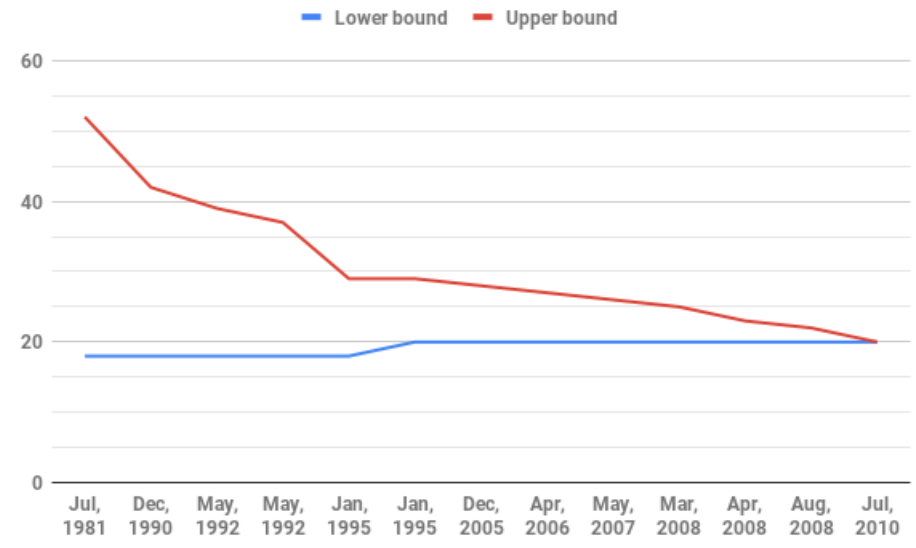
  Σε αντιδιαστολή με την αναζήτηση του μικρότερου δυνατού άνω φράγματος, υπήρξε αντίστοιχη μελέτη για την εύρεση του μεγαλύτερου κάτω φράγματος. Ήταν γνωστό ότι ορισμένες διατάξεις απαιτούσαν τουλάχιστον 18 κινήσεις για την επίλυσή τους. Παρ’όλα αυτά δεν ήταν σαφές αν ο αριθμός αυτός μπορούσε να είναι μεγαλύτερος. Στα μέσα της δεκαετίας του ‘90 βρέθηκε μία ειδική διάταξη του κύβου από τον Michael Reid, σχεδιασμένη ειδικά για την χρήση της ως παράδειγμα για την αύξηση του κάτω φράγματος. Η διάταξη ονομαζόταν superflip και δείχνεται στο σχήμα. Απαιτούσε 20 κινήσεις η επίλυση και ήταν το νέο κάτω όριο το οποίο θα συναντούσε το κατώτερο άνω όριο σε 15 χρόνια.

Σχήμα 3.5: Διάταξη superflip, μία από τις διατάξεις με μέγιστη απόσταση από τη λύση

- Σύγκλιση άνω και κάτω φράγματος

  Το 2010 κατόπιν έρευνας από τον Tomas Rokicki από το Palo Alto της Καλιφόρνιας και βοήθεια από την Google η οποία του παρείχε την απαραίτητη υπολογιστική ισχύ, απέδειξε ότι το κατώτερο άνω όριο ήταν επίσης 20 κινήσεις. Συνοπτικά παρουσιάζεται ο παρακάτω πίνακα με όλες τις σημαντικές ημερομηνίες καθώς και το αντίστοιχο χρονοδιάγραμμα.

Date	LB	UB	Gap	Notes and Links
July, 1981	18	52	34	Morwen Thistlethwaite proves 52 moves suffice.
December, 1990	18	42	24	Hans Kloosterman improves this to 42 moves.
May, 1992	18	39	21	Michael Reid shows 39 moves is always sufficient.
May, 1992	18	37	19	Dik Winter lowers this to 37 moves just one day later!
January, 1995	18	29	11	Michael Reid cuts the upper bound to 29 moves by analyzing Kociemba's two-phase algorithm.
January, 1995	20	29	9	Michael Reid proves that the "superflip" position (corners correct, edges placed but flipped) requires 20 moves.
December, 2005	20	28	8	Silviu Radu shows that 28 moves is always enough.
April, 2006	20	27	7	Silviu Radu improves his bound to 27 moves.
May, 2007	20	26	6	Dan Kunkle and Gene Cooperman prove 26 moves suffice.
March, 2008	20	25	5	Tomas Rokicki cuts the upper bound to 25 moves.
April, 2008	20	23	3	Tomas Rokicki and John Welborn reduce it to only 23 moves.
August, 2008	20	22	2	Tomas Rokicki and John Welborn continue down to 22 moves.
July, 2010	20	20	0	Tomas Rokicki, Herbert Kociemba, Morley Davidson, and John Dethridge prove that God's Number for the Cube is exactly 20.



Σχήμα 3.6: Γραφική αναπαράσταση χρονοδιαγράμματος σύγκλισης άνω και κάτω ορίου

3.5 Μη Ντετερμινιστική Προσέγγιση

Στην προσπάθεια να βρεθεί ο αλγόριθμος του Θεού, καταφεύγουμε σε μη-ντετερμινιστικές προσεγγίσεις επίλυσης του κύβου, κάνοντας χρήση βαθιάς ενισχυτικής μάθησης. Κατά γενική ομολογία, το Deep Reinforcement Learning είναι ένα εργαλείο το οποίο εφαρμόζεται στο gaming. Ο λόγος για τον οποίον υπάρχει αυτή η τάση είναι ότι ιστορικά, η πρώτη επιτυχία σε αυτόν τον τομέα έγινε στην ομάδα παιχνιδιών Atari από την Deep Mind το 2015 και το Atari Benchmark Suite. Το Atari Benchmark Suite αποδείχτηκε πολύ επιτυχημένη εφαρμογή για τα προβλήματα RL. Ακόμα και σήμερα, πολλές ερευνητικές ομάδες το χρησιμοποιούν αναδεικνύοντας έτσι την αποδοτικότητα του. Καθώς το πεδίο του RL εξελίσσεται, τα κλασσικά 53 αρχικά παιχνίδια Atari ολοένα γίνονται ευκολότερα για τους μοντέρνους πράκτορες RL (ήδη πάνω από τα μισά επιλύονται με σκορ μεγαλύτερο από τον μέσο άνθρωπο) και οι έρευνες στρέφονται σε πιο περίπλοκα παιχνίδια όσον αφορά:

- Χώρο καταστάσεων - state-space
- Πλήθος πρακτόρων - multi-agent
- Πλήθος παικτών - multiple players
- Ανακριβής πληροφορία - imperfect information
- Χώρο ενεργειών - action-space
- Καθυστέρηση στις επιβραβεύσεις - delayed credit assignment

Στα πλαίσια της διπλωματικής αυτής εργασίας, ακολουθεί μία εμπεριστατωμένη μελέτη και υλοποίηση σε PyTorch μιας πρόσφατης εφαρμογής του RL σε ένα πεδίο συνδυαστικής βελτιστοποίησης όπως είναι ο κύβος του Ρούμπικ.

Κεφάλαιο 4

Πειραματικό Μέρος

Η πειραματική διαδικασία που παρουσιάζεται σε αυτό το κεφάλαιο βασίζεται σε μία καινούργια προσέγγιση επίλυσης από ερευνητές στο πανεπιστήμιο της Καλιφόρνιας στο Irvine η οποία χρησιμοποιεί τεχνικές που χρησιμοποιήθηκαν πρόσφατα και από την Google DeepMind για την επίλυση προβλημάτων όπως το σκάκι και το Go.

4.1 Εισαγωγή

Μέχρι πριν την εφαρμογή μηχανικής μάθησης σε παιχνίδια, οι κύριοι τρόποι επίλυσης του κύβου ήταν:

- Χρήση αποτελεσμάτων θεωρίας ομάδων μέσω των οποίων υπάρχει τρόπος σημαντικής ελάττωσης του χώρου καταστάσεων του κύβου. Ένα από τα πιο διάσημα προγράμματα επίλυσης κύβων βασίζεται στον αλγόριθμο του Kociemba, ο οποίος αναλύθηκε στο προηγούμενο κεφάλαιο.

- Χρήση brute-force σε συνδυασμό με περιπτωσιολογικούς χειρισμούς οι οποίοι οδηγούν την αναζήτηση σε μία πιο ελπιδοφόρα υποπεριοχή του τεράστιου αυτού δέντρου. Προκύπτει από την εφαρμογή του αλγορίθμου του Korth.

Η επιστημονική έρευνα των ερευνητών βασίζεται στην εκπαίδευση ενός νευρωνικού δικτύου πάνω σε πολλούς τυχαία ανακατεμένους κύβους. Η διαδικασία αυτή επιτρέπει τη συστηματική σύγκλιση μιας πολιτικής του πράκτορα η οποία μέσα σε αυτή τη θάλασσα των δυνατών διατάξεων, δείχνει το δρόμο προς τη λυμένη διάταξη. Το σημαντικό σε αυτή τη μέθοδο είναι ότι η εκπαίδευση του νευρωνικού δικτύου δεν προαπαιτεί γνώση του αντικειμένου. Ο πράκτορας μαθαίνει μέσω αλληλεπίδρασης να λύνει τον κύβο χωρίς να έχει διδαχθεί τεχνικές επίλυσης που χρησιμοποιεί ο άνθρωπος. Το μόνο που χρειάζεται ο πράκτορας είναι μία αναπαράσταση του κύβου, δηλαδή το περιβάλλον του, με το οποίο θα αλληλεπιδράσει και θα αποκτήσει γνώση.

4.2 Αναπαράσταση Δεδομένων

Όπως αναφέρθηκε στα εισαγωγικά κεφάλαια για την ενισχυτική μάθηση, οφείλουμε να αναπαραστήσουμε με κάποιο τρόπο οντότητες όπως οι ενέργειες που κάνουμε και οι καταστάσεις στις οποίες βρισκόμαστε.

4.2.1 Ενέργειες

Οι ενέργειες που μπορούμε να κάνουμε σε έναν κύβο είναι περιστροφές των πλευρών του είτε ωρολογιακά (+90 μοίρες) είτε αντιωρολογιακά (-90 μοίρες). Διαθέτουμε 6 πλευρές άρα συνολικά 12 βαθμούς ελευθερίας, 12 δυνατές ενέργειες σε κάθε κατάσταση. Υπάρχει καθιερωμένη ονομασία των 12 αυτών ενεργειών. Η ονομασία προκύπτει από την πλευρά που περιστρέφεται κάθε φορά, θεωρώντας τον προσανατολισμού του κύβου στο χώρο δεδομένο και αμετάβλητο. Οι 12 ενέργειες που μπορούμε να κάνουμε είναι

1. Left - [L]
2. Right - [R]
3. Top - [T]
4. Bottom - [B]
5. Front - [F]
6. Back - [B]

Οι 6 πρώτες ενέργειες θεωρούνται ότι εκτελούνται ωρολογιακά, κοιτάζοντας πάντα κάθετα την πλευρά που περιστρέφεται. Για τις 6 αντιωρολογιακές ενέργειες έχουμε:

1. Left - [l]
2. Right - [r]
3. Top - [t]
4. Bottom - [b]
5. Front - [f]
6. Back - [b]

Σε άλλες αναπαραστάσεις, χρησιμοποιείται η σημειογραφία του τόνου για την υπόδειξη αντιωρολογιακής φοράς, δηλαδή [L', R', T', B', F', B']. Για την αναπαράσταση στον υπολογιστή, δεν είναι βολική η χρήση τόνων οπότε στην υλοποίησή μας χρησιμοποιήθηκαν μικρά γράμματα για την αντιωρολογιακή φορά.

Έχουμε λοιπόν μία λίστα από ενέργειες [L, R, T, B, F, B, l, r, t, b, f, b] οι οποίες αντιστοιχούν σε 12 αριθμούς από το 0 μέχρι το 11, διευθυνσιοδοτόντας τα 12 στοιχεία της λίστας μας. Στον κώδικα, η αναπαράσταση του χώρου ενεργειών γίνεται με τη χρήση enum της γλώσσας προγραμματισμού python, όπου κάθε ενέργεια χαρτογραφείται σε μία ακέραια αριθμητική τιμή.

4.2.2 Καταστάσεις

Κατάσταση ορίζεται ως μία συγκεκριμένη μετακίνηση των μικρών κύβων στο χώρο και ως επέκταση αυτού, μία συγκεκριμένη διάταξη χρωμάτων σε κάθε πλευρά. Έχουμε αναφέρει αρκετές φορές πόσο μεγάλος είναι αυτός ο χώρος ($4.33 * 10^{19}$ διαφορετικές καταστάσεις). Ο μεγάλος αυτός αριθμός δυστυχώς δεν είναι η μόνη δυσκολία που έχουμε να αντιμετωπίσουμε. Όταν επιλέγουμε μία συγκεκριμένη αναπαράσταση καταστάσεων έχουμε συγκεκριμένους στόχους:

1. Αποφυγή πλεονασμών

   Στη χειρότερη δυνατή περίπτωση, έχουμε τη δυνατότητα να αναπαραστήσουμε τον κύβο παρακολουθώντας μόνο τα αυτοκόλλητα του κύβου. Υπολογίζοντας τον αριθμό των συνδυασμών αυτών, καταλήγουμε σε ένα νούμερο της τάξης του 10^{37}. Έχουμε 6 πλευρές του κύβου, κάθε πλευρά έχει 8 αυτοκόλλητα οπότε έχουμε 48 αυτοκόλλητα συνολικά. Κάθε αυτοκόλλητο μπορεί να πάρει 6 χρώματα, οπότε συνολικά έχουμε $6^{68} = 2.25 * 10^{37}$, αρκετά μεγαλύτερο από τον χώρο καταστάσεων του κύβου. Συμπεραίνουμε λοιπόν ότι η αναπαράσταση εμπεριέχει πλεονάζουσα πληροφορία η οποία δεν μας χρειάζεται. Για να χαρακτηρίσουμε αμφιμονοσήμαντα μία κατάσταση του κύβου, η πληροφορία που χρειάζεται είναι σημαντικά μικρότερη. Η αναπαράσταση με τα χρώματα δεν είναι καθόλου αποδοτική καθώς επιτρέπει π.χ. μία γωνία να έχει 3 ίδια χρώματα, και άλλες περίεργες αδύνατες διατάξεις.

2. Αποδοτικότητα στη μνήμη

   Όπως θα δούμε αργότερα, κατά τη διάρκεια της εκπαίδευσης και εφαρμογής, θα χρειαστεί να κρατήσουμε στη μνήμη μεγάλο πλήθος διαφορετικών καταστάσεων του κύβου που ενδεχομένως να επηρεάσουν την απόδοση της διαδικασίας. Έτσι, θα θέλαμε να είναι όσο το δυνατόν πιο συμπαγής η αναπαράσταση.

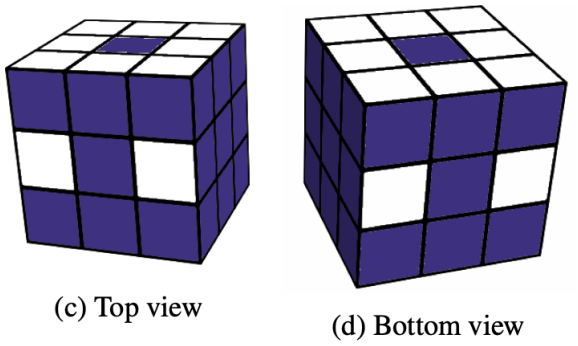
3. Αποδοτικότητα μετασχηματισμών

   Οφείλουμε επίσης να προσέξουμε το πόσο συμπαγής είναι η αναπαράσταση διότι χρειάζεται να εκτελούμε όλες τις ενέργειες πάνω στις καταστάσεις. Αυτές οι ενέργειες οφείλουν να εκτελούνται γρήγορα. Εάν η αναπαράστασή μας είναι πολύ συμπαγής όσον αφορά τη μνήμη (για παράδειγμα bit-κωδικοποίηση) αλλά απαιτεί χρονοβόρα διαδικασία αποσυμπίεσης για κάθε περιστροφή του κύβου, η εκπαίδευση θα γίνει πάρα πολύ αργή.

4. Συμβατότητα με αρχιτεκτονική νευρωνικού δικτύου

Δεν είναι όλες οι αναπαραστάσεις δεδομένων ίδιες ως προς τη συμβατότητα χρήσης με νευρωνικά δίκτυα. Αυτό ισχύει στη μηχανική μάθηση γενικότερα. Για παράδειγμα, στην επεξεργασία φυσικής γλώσσας (NLP) είναι σύνηθες να χρησιμοποιούνται batches λέξεων. Στην επεξεργασία εικόνας, αποκωδικοποιούνται οι jpeg φωτογραφίες σε raw pixels. Τα τυχαία δάση απαιτούν αρκετή τροποποίηση δεδομένων. Στην περίπτωσή μας, κάθε κατάσταση αναπαριστάται από ένα 20 x 24 τανιστή χρησιμοποιώντας κωδικοποίηση one-hot[9]. (Στην περίπτωση του 2x2 κύβου, η είσοδος είναι ένας 8x24 τανιστής)

Για την κατανόηση του μοντέλου αναπαράστασης, παραθέτουμε τη φωτογραφία από το paper[17]:

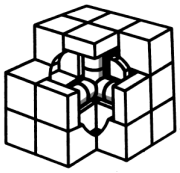


Σχήμα 4.1: (c) και (d) φαίνεται η διαστατική ελάττωση του κύβου για είσοδο στο νευρωνικό δίκτυο. Τα αυτοκόλλητα που χρησιμοποιούνται έχουν άσπρο χρώμα ενώ τα αυτοκόλλητα που δεν λαμβάνονται υπόψιν χρωματίζονται μπλε. [17]

Βλέπουμε με λευκό χρώμα τα αυτοκόλλητα που είναι οριακά απαραίτητα έτσι ώστε να μην έχουμε πλεονάζουσα πληροφορία στην αναπαράστασή μας. Τα μπλε αυτοκόλλητα δεν χρειάζεται να τα παρακολουθούμε γιατί προκύπτουν άμεσα από τα λευκά.

Ο κύβος του Ρούμπικ έχει 3 κατηγορίες μικρότερων κύβων που (ονομάζονται cubelets). 8 κυβάκια "γωνίες" που έχουν 3 αυτοκόλλητα, 12 κυβάκια "πλευρές" που έχουν 2 αυτοκόλλητα και 6 κεντρικά κυβάκια με 1 αυτοκόλλητο.

Αν και δεν μοιάζει προφανές από την αρχή, τα κεντρικά αυτοκόλλητα δεν κινούνται λόγω της εσωτερικής δομής του κύβου. Δεν μπορούν να αλλάξουν σχετική θέση μεταξύ τους. Η εσωτερική δομή του κύβου αποτελείται από 3 σταθερούς άξονες (x,y,z) οι οποίοι καταλήγουν στα κεντρικά κυβάκια όπως παρουσιάζει και το παρακάτω σχήμα:



Σχήμα 4.2: Παρουσίαση εσωτερικής δομής του κύβου. Λόγω σύνδεσης των κεντρικών cubelets στους άξονες, δεν αλλάζουν σχετική θέση κατά την περιστροφή των εδρών

Η τοποθεσία των κεντρικών κύβων ορίζει μία συγκεκριμένη στερεοδιάταξη του κύβου στο χώρο οπότε συμφωνούμε σε μία σταθερή (Λευκά TOP, Κόκκινα FRONT). Συνεπώς, δεν παρακολουθούμε τα κεντρικά κυβάκια, μένουν σταθερά και αμετάβλητα στο χώρο.

Για τα υπόλοιπα κυβάκια πάλι συμπεραίνουμε ότι χρειαζόμαστε να παρακολουθούμε μόνο ένα αυτοκόλλητο από τα 3 αν βρίσκεται στη γωνία ή 2 αν βρίσκεται σε κάποια πλευρά. Έστω η μοναδική γωνία που έχει συνδυασμούς χρωμάτων: κόκκινο/λευκό/πράσινο. Λόγω γνωστής σχετικής θέσης μεταξύ τους, αν γνωρίζουμε τη μία θέση, εξάγουμε έμμεσα τα άλλα 2 αυτοκόλλητα πάνω στο κυβάκι. Ας φανταστούμε ένα ορθοκανονικό σύστημα (x,y,z) το οποίο τοποθετούμε σε μία γωνία συγκεκριμένη και γνωρίζουμε μόνο την κατεύθυνση του x άξονα. Εύκολα λόγω της σχετικής θέσης του x με y και z μπορούμε να εξάγουμε που βρίσκεται ο y άξονας και που ο z. Το ίδιο ισχύει για τα πλευρικά κυβάκια τα οποία έχουν μόνο 2 χρώματα. Αν ξέρουμε το ζευγάρι χρωμάτων και που βρίσκεται το 1 από αυτά, μπορούμε να συμπεράνουμε που βρίσκεται και το άλλο. Για να βρούμε τη θέση ενός από τα κυβάκια, αρκεί να γνωρίζουμε τη θέση ενός από τα αυτοκόλλητά του. Η επιλογή αυτή είναι εντελώς αυθαίρετη, αλλά μόλις οριστεί, οφείλουμε να πειθαρχήσουμε σ'αυτήν. Συνοψίζοντας λοιπόν, οφείλουμε να παρακολουθήσουμε 8 αυτοκόλλητα από την πάνω πλευρά, 8 από την κάτω, και άλλα 4 από την μεσαία λωρίδα. Συνολικά λοιπόν παρακολουθούμε 20 αυτοκόλλητα.

Η είσοδος στο νευρωνικό μας δίκτυο είναι ένας τανυστής $20 * 24$. Ακολουθεί η ερμηνεία της δεύτερης αυτής διάστασης (24). Συνολικά έχουμε 20 διαφορετικά αυτοκόλλητα να παρακολουθήσουμε, τα οποία μπορούν να εμφανιστούν σε διαφορετικές θέσεις ανάλογα με το είδος του cublet που παρακολουθούμε. Ας θεωρήσουμε λοιπόν τα cubelet-γωνίες. Έχουμε 8 πιθανές γωνίες και κάθε γωνία έχει 3 δυνατές θέσεις. Συνολικά λοιπόν, ένα αυτοκόλλητο μπορεί να εμφανιστεί στο γινόμενο $3 * 8 = 24$ δυνατές θέσεις.

Αντίστοιχα, ένα cubelet-πλευρά μπορεί να εμφανιστεί σε 12 πλευρές, και η κάθε πλευρά έχει δύο διαθέσιμες θέσεις. Συνολικά λοιπόν και κατά σύμπτωση, το αυτοκόλλητο μπορεί να εμφανιστεί σε 24 διαφορετικές θέσεις.

4.3 Κωδικοποίηση Δεδομένων

Μία πολύ διάσημη μέθοδος κωδικοποίησης δεδομένων είναι η κωδικοποίηση one-hot[9]. Η κωδικοποίηση one-hot μας διαμορφώνει λοιπόν την είσοδο στο νευρωνικό δίκτυο ως έναν τανυστή $20 * 24$.

- Θετικά στοιχεία αναπαράστασης

  Οι δυνατές διατάξεις με την αναπαράστασή μας είναι 24^{20} που είναι περίπου $4.02 * 10^{27}$. Ο αριθμός των διατάξεων ξεπερνάει τον πραγματικό δυνατό αριθμό αναπαραστάσεων (που είναι περίπου 10^{19}). Παρόλα αυτά, η προσέγγισή μας είναι πολύ πιο αποδοτική από την απλή παρακολούθηση μόνο των stickers στον κύβο.

- Αρνητικά στοιχειά αναπαράστασης

  Η αναπαράσταση του κύβου ως τανυστή $20 * 24$ δεν είναι η καλύτερη όσον αφορά τη δέσμευση μνήμης. Αυτό συμβαίνει γιατί δεσμεύουμε κάθε φορά στη μνήμη έναν τανυστή κινητής υποδιαστολής διαστάσεων $20 * 24$, ξοδεύοντας έτσι $4 * 20 * 24 = 1920$ byte δεδομένων. Ενώ το νούμερο αυτό δεν είναι μεγάλο, θα χρειαστεί να αποθηκεύσουμε χιλιάδες κύβους στη μνήμη κατά την εκπαίδευση του νευρωνικού μας δικτύου.

Δεν είναι όμως απαραίτητο να υπάρχει μία μοναδική αναπαράσταση του κύβου στη μνήμη. Εναλλακτικά, μπορούμε να χρησιμοποιήσουμε άλλη μία αναπαράσταση του κύβου προκειμένου να εκμεταλλευτούμε τα πλεονεκτήματα κάθε μίας. Η νέα αναπαράσταση που προτείνεται είναι πολύ πιο συμπαγής έτσι ώστε να δεσμεύεται λιγότερη μνήμη. Ειδικότερα, η αποθήκευση του κύβου γίνεται με τη μορφή λίστας, η οποία περιέχει τις θέσεις των cublets και τους προσανατολισμούς τους στο χώρο. Λόγω αυτής τη κωδικοποίησης, ο κάθε κύβος δεσμεύει 160 byte μνήμης με αποτέλεσμα η συνολική δέσμευση μνήμης να είναι πολύ μικρότερη. Επομένως, η πρώτη αναπαράσταση είναι ιδανική ωστε να λειτουργήσει ως είσοδος σε νευρωνικό δίκτυο και η δεύτερη, είναι ιδανική για τη διαχείριση μνήμης.

```
10  # environment API
11  State = collections.namedtuple("State", field_names=['corner_pos', 'side_pos', 'corner_ort', 'side_ort'])
12
```

Σχήμα 4.3: Compact State Representation

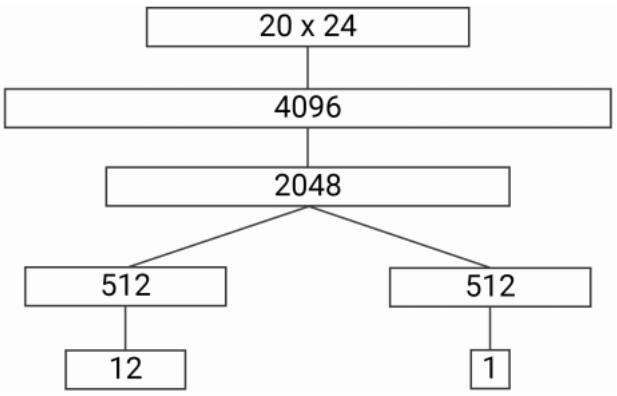
```
230 def encode_inplace(target, state):
231     """
232     Encode cude into existig zeroed numpy array
233     Follows encoding described in paper https://arxiv.org/abs/1805.07470
234     :param target: numpy array
235     :param state: state to be encoded
236     """
237     assert isinstance(state, State)
238
239     # handle corner cubelets: find their permuted position
240     for corner_idx in range(8):
241         perm_pos = state.corner_pos.index(corner_idx)
242         corn_ort = state.corner_ort[perm_pos]
243         target[corner_idx, perm_pos * 3 + corn_ort] = 1
244
245     # handle side cubelets
246     for side_idx in range(12):
247         perm_pos = state.side_pos.index(side_idx)
248         side_ort = state.side_ort[perm_pos]
249         target[8 + side_idx, perm_pos * 2 + side_ort] = 1
```

Σχήμα 4.4: Neural-friendly State Representation

4.4 Εκπαίδευση Νευρωνικού Δικτύου

4.4.1 Αρχιτεκτονική Νευρωνικού Δικτύου

Η αρχιτεκτονική του νευρωνικού δικτύου ακολουθεί την αρχιτεκτονική που παρουσιάζεται στο άρθρο[17]



Σχήμα 4.5: Επίπεδα νευρωνικού δικτύου σε μπλοκ. Ως είδοσο δέχεται την αναπαράσταση του κύβου ως $20 * 24$ τανυστή. Στην έξοδο υπολογίζεται η κατανομή πολιτικής και η αξία της κάθε κατάστασης

Το νευρωνικό μας δίκτυο δέχεται σαν είσοδο τον $20 * 24$ τανυστή όπως αναφέρθηκε παραπάνω και παράγει 2 εξόδους:

- Policy

  Η πολιτική είναι ένα διάνυσμα 12 διαστάσεων που αντιπροσωπεύει την κατανομή πιθανότητας των ενεργειών μας

- Value

  Διάνυσμα μιας διάστασης από το οποίο αντλούμε πληροφορία για την αναμενόμενη αξία της κατάστασης που βρίσκεται ο κύβος.

Το νευρωνικό δίκτυο που παρουσιάζεται [17] αποτελείται από δύο πλήρως συνδεδεμένα κρυμμένα επίπεδα υπολογιστικών νευρώνων. Η συνάρτηση ενεργοποίησης των νευρώνων του δικτύου είναι η ELU (εκθετική γραμμική μονάδα) η οποία είναι:

$$f(\alpha, x) = \begin{cases} \alpha(e^x - 1) & \text{for } x \leq 0 \\ x & \text{for } x > 0 \end{cases}$$

Στον κώδικα η αρχιτεκτονική του νευρωνικού δικτύου ακολουθεί πιστά την αρχιτεκτονική των συγγραφέων στο άρθρο παρουσίασης της αυτοδιδασκόμενης επανάληψης [17]

```
11
12 class Net(nn.Module):
13     def __init__(self, input_shape, actions_count):
14         super(Net, self).__init__()
15
16         self.input_size = int(np.prod(input_shape))
17         self.body = nn.Sequential(
18             nn.Linear(self.input_size, 4096),
19             nn.ELU(),
20             nn.Linear(4096, 2048),
21             nn.ELU()
22         )
23         self.policy = nn.Sequential(
24             nn.Linear(2048, 512),
25             nn.ELU(),
26             nn.Linear(512, actions_count)
27         )
28         self.value = nn.Sequential(
29             nn.Linear(2048, 512),
30             nn.ELU(),
31             nn.Linear(512, 1)
32         )
```

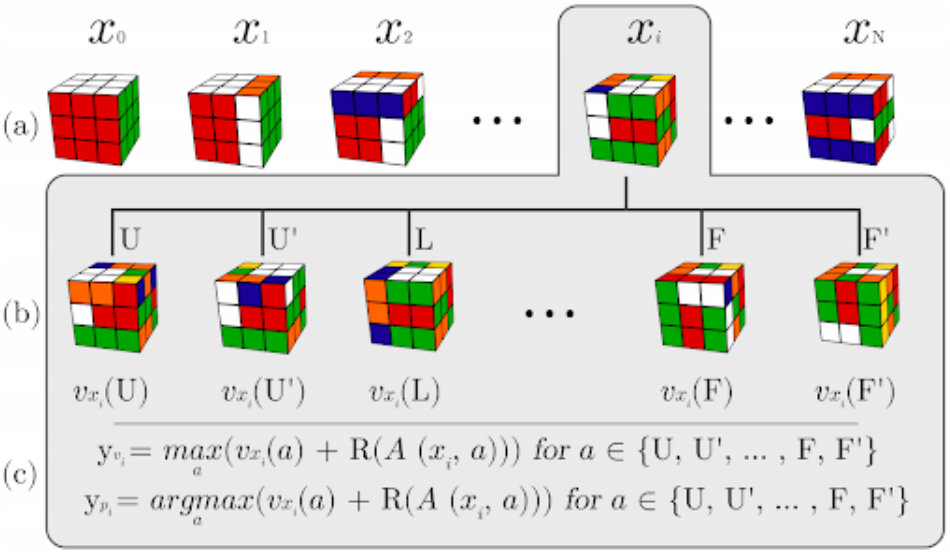
Σχήμα 4.6: Υλοποίηση του νευρωνικού δικτύου σε python όπως δείχνει το σχήμα 4.5

4.4.2 Αυτοδιδασκόμενη Επανάληψη

Το νευρωνικό δίκτυο είναι απλό στη δομή του. Το μεγάλο ερώτημα είναι το πώς δίνεται η σωστή κατεύθυνση ως προς την εκπαίδευσή του. Η εκπαίδευση του νευρωνικού δικτύου γίνεται μέσω μιας νέας μεθόδου που ονομάζεται "Auto-Didactic Iteration" [17]. Η μέθοδος θυμίζει αναδρομικό αλγόριθμο καθώς ακολουθεί αντίστροφη πορεία διάσχισης του διαγράμματος καταστάσεων του κύβου. Ξεκινώντας από την αρχική κατάσταση (λυμένη), επιλέγουμε τυχαίες N κινήσεις και ανακατεύουμε τον κύβο. Σε κάθε μία από τις καταστάσεις που δημιουργούνται εφαρμόζουμε τα ακόλουθα:

- Εφαρμόζουμε όλες τις δυνατές κινήσεις (12) στην κατάσταση s
- Τροφοδοτούμε όλες τις 12 νέες καταστάσεις στο νευρωνικό δίκτυο και ζητάμε την τιμή της αξίας για κάθε κατάσταση. Έτσι παίρνουμε 12 τιμές για όλες τις νέες καταστάσεις που προκύπτουν
- Η επιθυμητή τιμή της αξίας της κατάστασης υπολογίζεται ως εξής: $y_{u_i} = \max_a(u_s(s) + R(A(s,a)))$ όπου $A(s,a)$ είναι η κατάσταση μετά την κίνηση a στην κατάσταση s και $R(s)$ είναι η αμοιβή του πράκτορα η οποία είναι $+1$ άμα ο κύβος είναι λυμένος και -1 αν δεν είναι
- Η επιθυμητή τιμή της πολιτικής υπολογίζεται ως εξής: $y_{p_i} = \text{argmax}_a(u_s(s)+R(A(s,a)))$. Αυτό σημαίνει ότι η επιθυμητή κίνηση παίρνει την τιμή 1 ενώ οι υπόλοιπες 11 κινήσεις παίρνουν την τιμή 0

Αυτη η διαδικασία φαίνεται και στο παρακάτω σχήμα. Η αλληλουχία κινήσεων $x_0, x_1, ...$ δημιουργείται και φαίνονται τα 4 παραπάνω βήματα στην κατάσταση x_i.



Σχήμα 4.7: Ο αλγόριθμος αυτοδιδασκόμενης επανάληψης [17]

4.5 Κατάλληλη Μέθοδος Διάσχισης Γράφου

Μία αναμενόμενη και λογική διαδικασία που θα ακολουθούσε κανείς προκειμένου να ελέγξει την ικανότητα του νευρωνικού δικτύου για την επίλυση του κύβου, αφού έχει προηγηθεί η εκπαίδευσή του θα ήταν:

1. Παρουσίαση στο νευρωνικό δίκτυο μιας κατάστασης του κύβου
2. Επιλογή κατάλληλης κίνησης που προκύπτει από την κατανομή πολιτικής στην έξοδο
3. Εκτέλεση κίνησης στον κύβο
4. Επανάληψη της παραπάνω διαδικασίας μέχρις ότου να λυθεί ο κύβος

Η διαδικασία αυτή εκ πρώτης όψεως φαίνεται να έχει βάση και να είναι λειτουργική. Παρ' όλα αυτά, η φύση των νευρωνικών δικτύων σε συνδυασμό με την πολυπλοκότητα του προβλήματος παρακωλύουν την αποδοτική λειτουργία του πράκτορα. Πιο συγκεκριμένα, η εκπαίδευση ενός νευρωνικού δικτύου σε τέτοιο βαθμό ώστε να επιστρέφει την θεωρητικά ιδανική αλληλουχία κινήσεων για την επίλυση κάθε κύβου, είναι σχεδόν αδύνατη. Εναλλακτικά, το μοντέλο μας σε κάθε κατάσταση προτείνει μία κίνηση η οποία ενδέχεται να οδηγήσει τον κύβο στην αρχική του κατάσταση. Οι κινήσεις αυτές όμως είναι πιθανό να οδηγήσουν τον κύβο προς κάποια λανθασμένη κατεύθυνση, κάτι το οποίο δικαιολογείται από την πληθώρα των καταστάσεων που υπάρχουν και τις οποίες δεν είναι εφικτό να έχει συναντήσει στη διαδικασία

εκπαίδευσης. Υπενθυμίζεται ότι οι δυνατές καταστάσεις είναι $4.33 * 10^{19}$. Αν υπήρχε κάποια GPU που συναντά εκατοντάδες χιλιάδες καταστάσεις του κύβου το δευτερόλεπτο θα έχει συναντήσει μόνο το 0.0000005% των πιθανών καταστάσεων, ύστερα από εκπαίδευση ενός μήνα.

4.5.1 Monte Carlo Tree Search

Η εξαγωγή της μέγιστης δυνατής πληροφορίας από το δέντρο μας χρήζει επιλογής της κατάλληλης μεθόδου διάσχισης γράφου. Η μέθοδος που κρίνεται καταλληλότερη για την επίτευξη του στόχου αυτού, είναι η Monte Carlo Tree Search (MCTS). Αυτή έρχεται να ισορροπήσει τις πλέον διαδεδομένες brute force μεθόδους Breadth-First Search και Depth-First Search, όπου πραγματοποιείται εξονυχιστική αναζήτηση του δέντρου περνώντας τελικά από όλες τις δυνατές καταστάσεις.

- DFS

  Ο αλγόριθμος DFS ξεκινάει από τη ρίζα του δέντρου και κάνει κάθετη αναζήτηση μέχρι να φτάσει σε κάποιο φύλλο πριν αναζητήσει άλλους κλάδους

- BFS

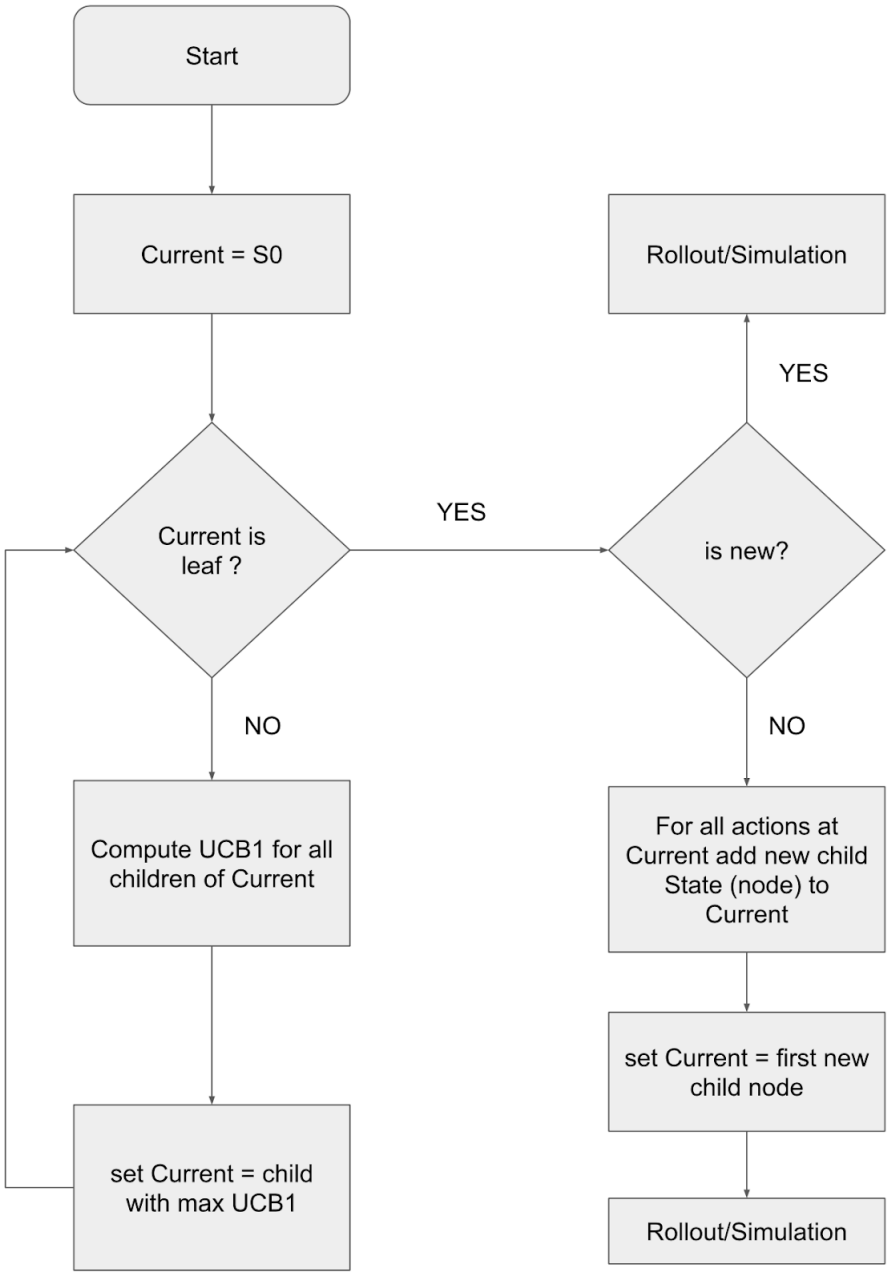
  Ο αλγόριθμος BFS ξεκινάει από τη ρίζα του δέντρου αλλά κάνει τοπική αναζήτηση (παράλληλη) πριν ξεκινήσει την κάθετη αναζήτηση σε διαφορετικό επίπεδο. Ο αλγόριθμος BFS κάνει αναζήτηση κόμβων με βάση την απόσταση (ύψος) από την κορυφή.

Ο αλγόριθμος Monte Carlo Tree Search αξιοποιεί τους 2 αυτούς αλγορίθμους και τους συνδυάζει "βέλτιστα" για να κάνει διάσχιση του δέντρου έχοντας 2 χαρακτηριστικά που οφείλουμε σαν σχεδιαστές να ισορροπήσουμε. Το γνωστό exploration-exploitation trade-off το οποίο παρουσιάζεται συχνά σε θέματα reinforcement learning.

Για να πετύχουμε αυτή την ισορροπία, ακολουθούμε τα τέσσερα βήματα του αλγορίθμου αναζήτησης Monte Carlo που χωρίζουν τον αλγόριθμο σε τέσσερα μέρη.

1. Μέρος 1 & 2 - Επιλογή Επέκταση (Selection & Expansion)

   Ο αλγόριθμος ξεκινάει από τη ρίζα (R) και κινείται κάθετα στο δέντρο επιλέγοντας βέλτιστα παιδιά μέχρι να φτάσει σε κάποιο φύλλο (L). Η επιλογή παιδιού γίνεται βάση της εξίσωσης "Upper Confidence Bounds" $u_i + C * \sqrt{\ln(N)/n_i}$ όπου u_i είναι η τιμή της κατάστασης s_i, n_i είναι ο αριθμός των επισκέψεων στον κόμβο/κατάσταση s_i, N είναι ο συνολικός αριθμός επισκέψεων στον κόμβο-πατέρα, C είναι μία σταθερά η οποία χρησιμοποιείται για μικρο-αλλαγές/τελειοποίηση. Αφού υπολογιστεί το UCB1 για κάθε παιδί της κάθε κατάστασης, επιλέγεται το παιδί με την μεγαλύτερη τιμή.

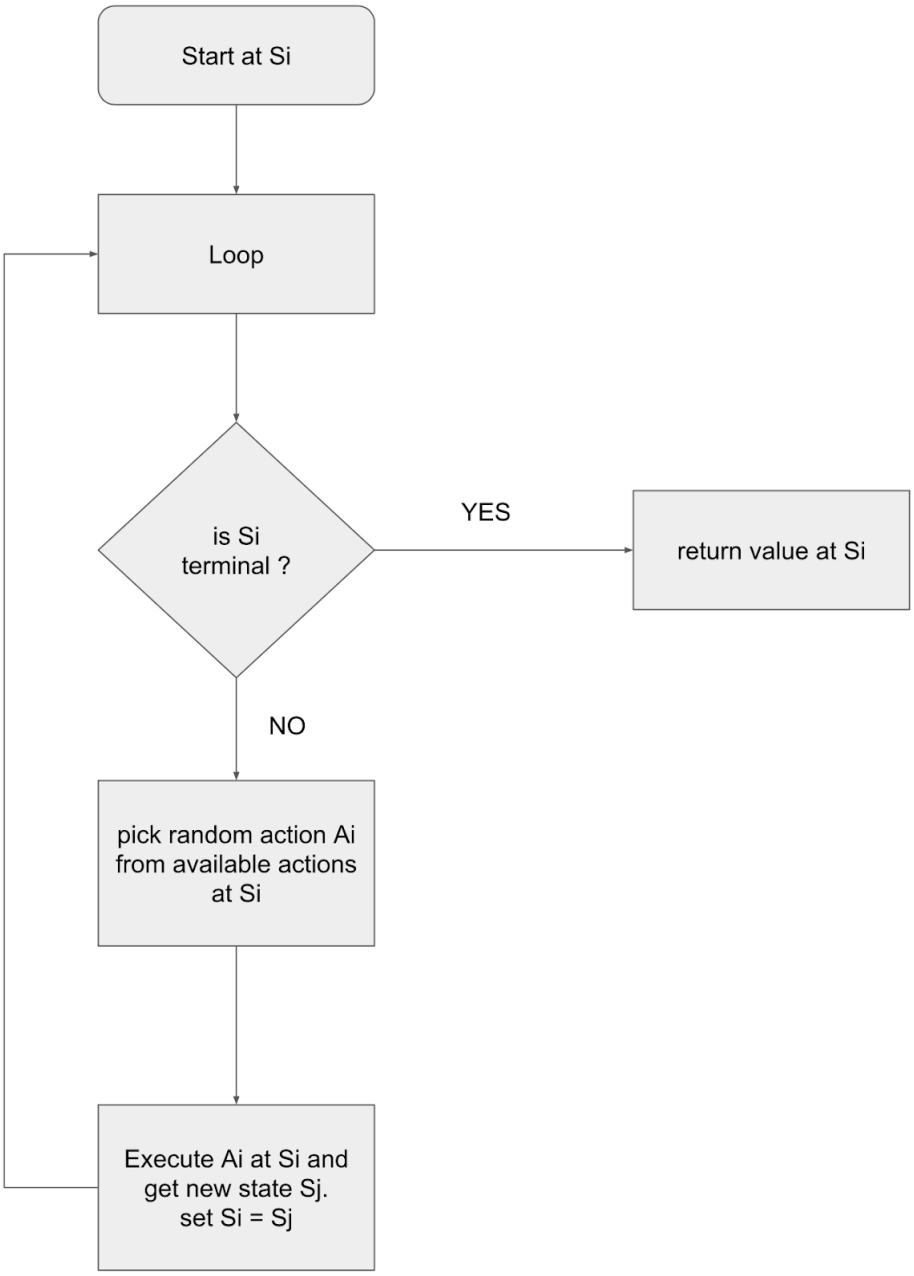


Σχήμα 4.8: Λογικό διάγραμμα αναζήτησης Monte Carlo - (Selection & Expansion)

Αν ο επιλεγόμενος κόμβος δεν έχει επισκεφθεί ξανά, καλείται η διαδικασία προσομοίωσης για να βρεθεί μία τελική κατάσταση με τιμή. Διαφορετικά, αν ο κόμβος που συναντιέται έχει επιλεγεί και στο παρελθόν, ακολουθεί η δημιουργία όλων των δυνατών παιδιών, για όλες τις επιτρεπτές κινήσεις (στην περίπτωσή μας 12).

2. Μέρος 2 - Προσομοίωση (Simulation)

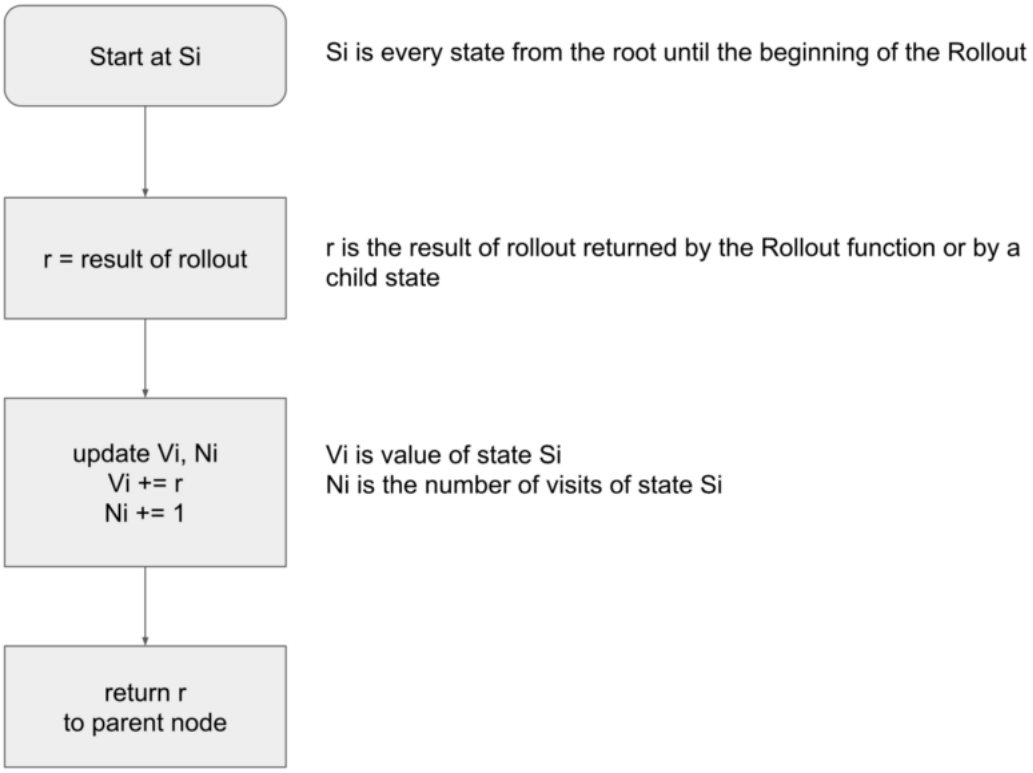
   Εκτελείται ένα τυχαίο "roll-out" από τον κόμβο M μέχρι μία τερματική κατάσταση. Η τερματική κατάσταση περιέχει κάποια τιμή η οποία διαδίδεται προς τα πίσω στο τελευταίο μέρος του αλγορίθμου μέσω backpropagation



Σχήμα 4.9: Λογικό διάγραμμα αναζήτησης Monte Carlo - (Simulation)

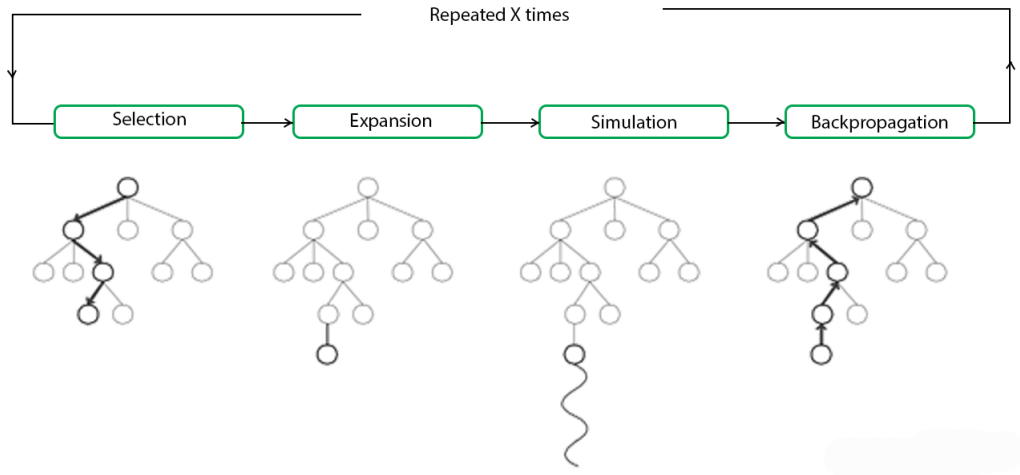
3. Μέρος 3 - Διάδοση προς τα πίσω (Backpropagation)

   Ύστερα από την περάτωση της προσομοίωσης, μία τιμή επιστρέφεται. Όλοι οι κόμβοι από τον κόμβο M μέχρι την ρίζα P ανανεώνονται αθροίζοντας το αποτέλεσμα στην τιμή τους και αυξάνουν τον αριθμό επίσκεψης στον κόμβο κατά 1.



Σχήμα 4.10: Λογικό διάγραμμα αναζήτησης Monte Carlo - (Backpropagation)

Παρακάτω παρουσιάζονται συνοπτικά όλα τα βήματα της αναζήτησης Monte Carlo



Σχήμα 4.11: Πλήρης αναπαράσταση αναζήτησης Monte Carlo

Πλεονεκτήματα MCTS

- Ευκολία στην υλοποίηση
- Χειριστικός αλγόριθμος. Η Δενδρική Αναζήτηση Monte Carlo μπορεί να λειτουργήσει αποτελεσματικά χωρίς καμία γνώση του τομέα
- Ύπαρξη δυνατότητας αποθήκευσης οποιασδήποτε ενδιάμεσης κατάστασης η οποία ενδέχεται να χρησιμοποιηθεί σε μελλοντικές περιπτώσεις όποτε χρειαστεί
- Υποστηρίζεται η ασύμμετρη επέκταση του δέντρου αναζήτησης βάση των συνθηκών λειτουργίας

Μειονεκτήματα MCTS

- Καθώς μεγαλώνει το δέντρο μετά από μερικές επαναλήψεις, έχει μεγάλες απαιτήσεις μνήμης
- Υπάρχει ζήτημα αξιοπιστίας σε περιπτώσεις αναζήτησης δέντρων με πολλούς κόμβους καθώς ο κάθε κόμβος δεν επισκέπτεται αρκετές φορές ώστε να κατανοήσει την έκβαση μακροπρόθεσμα
- Απαιτείται τεράστιος αριθμός επαναλήψεων για αποτελεσματική λήψη αποφάσεων. Δεν έχει δηλαδή τόσο γρήγορη ταχύτητα σύγκλισης

4.5.2 MCTS σε συνδυασμό με ADI

Όπως προαναφέρθηκε, το Monte Carlo Tree Search είναι μία οικογένεια μεθόδων οι οποίες διαφοροποιούνται σε λεπτομέρειες και χαρακτηριστικά. Στην έρευνα που εξετάζουμε, χρησιμοποιήθηκε η μέθοδος Upper Confidence Bound 1. Η μέθοδος αυτή εφαρμόζεται στο δέντρο όπου οι κόμβοι είναι οι καταστάσεις του κύβου και οι ακμές είναι οι 12 επιτρεπτές περιστροφές του κύβου οι οποίες μας οδηγούν σε διαφορετικές καταστάσεις.

Σκοπός εδώ δεν είναι η κατασκευή ολόκληρου του δέντρο αλλά μόνο ενός μικρού υποδέντρου. Αρχικά, κατασκευάζεται ένα δέντρο το οποίο αποτελείται από έναν κόμβο που αντιπροσωπεύει την παρούσα κατάσταση του κύβου. Σε κάθε βήμα του MCTS εκτελούνται κινήσεις και ανακαλύπτονται νέοι κόμβοι οι οποίοι ενσωματώνονται στο παραγόμενο δέντρο. Αρχικά το δέντρο που αποτελείται μόνο από έναν κόμβο, την παρούσα κατάσταση του κύβου που δόθηκε στον αλγόριθμο κατά την εκκίνηση. Σε κάθε βήμα του MCTS γίνεται κάποια διάσχιση του δέντρου και υπάρχουν δύο περιπτώσεις.

1. Ο κόμβος είναι φύλλο και δεν έχει παιδιά
2. Ο κόμβος έχει παιδιά

Στην περίπτωση συνάντησης φύλλου, πραγματοποιείται η επέκταση που αναφέρθηκε προηγουμένως, εφαρμόζοντας τις 12 επιτρεπτές κινήσεις στον κύβο. Όλες οι παραγόμενες καταστάσεις ελέγχονται με την ελπίδα εύρεσης της αρχικής κατάστασης (εάν ο στόχος μας επιτευχθεί τερματίζει η αναζήτηση). Η κατάσταση φύλλου περνάει από το μοντέλο μας και οι παραγόμενες έξοδοι πολιτικής και αξίας αποθηκεύονται για μελλοντική χρήση.

Στην περίπτωση συνάντησης κατάστασης η οποία δεν είναι φύλλο, υπάρχει γνώση των παιδιών της καθώς και γνώση των κατανομών πολιτικής και αξίας εφόσον πέρασαν από το νευρωνικό δίκτυο. Υποχρέωση του αλγορίθμου εδώ είναι η επιλογή καταλληλότερης απόφασης για την εύρεση της αρχικής κατάστασης. Παρόλο που υπάρχει συσσωρευμένη γνώση από τις εξόδους του νευρωνικού δικτύου, δεν υπάρχει κάποια εγγύηση ότι ο κύβος πάντα κατευθύνεται προς τη σωστή κατεύθυνση. Αυτή η ανησυχία θα μπορούσε να εξαλειφθεί εάν υπήρχε δυνατότητα άπειρων προσπελάσεων και διασχίσεων στο γράφο (καμία πραγματική αξία).

Για την εύρεση ισορροπίας, για κάθε κατάσταση αποθηκεύουμε τον μετρητή επιλογής της κάθε ενέργειας ο οποίος επιλέχθηκε για την αναζήτηση. Για να δοθεί κίνητρο αναζήτησης νέου μονοπατιού, το πλήθος επιλογής της ενέργειας a και η πιθανότητα επιλογής της ενέργειας a στο μέλλον είναι αντιστρόφως ανάλογα ποσά. Επί προσθέτως, η αξία που υπολογίζεται από το μοντέλο χρησιμοποιείται για τη λήψη αποφάσεων. Αποθηκεύεται η μέγιστη αξία κατόπιν σύγκρισης της αξίας της κατάστασης με την μέγιστη αξία των παιδιών της κατάστασης. Το γεγονός αυτό επιτρέπει τις πιο ελπιδοφόρες πορείες (σύμφωνα πάντα με το μοντέλο) να είναι ορατές από τα parent states.

Συνοψίζοντας, η σειρά κινήσεων που ακολουθείται από μία κατάσταση η οποία δεν είναι φύλλο επιλέγεται από την παρακάτω φόρμουλα:

$$A_t = \underset{a}{\operatorname{argmax}}(U_{st}(a) + W_{st}(a)) \tag{4.1}$$

όπου $U_{s_t}(a) = cP_{s_t}(a)\frac{\sqrt{\sum_{a'} N_{s_t}(a')}}{(1+N_{s_t}(a))}$

- $N_{s_t}(a)$ εκφράζει το πλήθος τον φορών που επιλέχθηκε η ενέργεια a στην κατάσταση s_t
- $P_{s_t}(a)$ η κατανομή πολιτικής του μοντέλου για την κατάσταση s_t
- $W_{s_t}(a)$ η μέγιστη τιμή που επιστρέφει το μοντέλο για όλα τα παιδιά της κατάστασης s_t δεδομένου ότι επιλέχθηκε η ενέργεια a

Η διαδικασία αυτή επαναλαμβάνεται ωσότου είτε (α) βρεθεί η λύση είτε (β) εξαντληθεί το όριο χρόνου ή δυνατών κινήσεων κατά τη διάσχιση του δέντρου η οποία ορίζεται ως υπερπαράμετρος (–MAX-STEPS / –MAX-TIME). Για επιτάχυνση της διαδικασίας, συνήθως η MCTS υλοποιείται κάνοντας χρήση παράλληλης επεξεργασίας (εφόσον κάθε αναζήτηση μπορεί να πραγματοποιηθεί ανεξάρτητα) και πολλές αναζητήσεις πραγματοποιούνται ταυτόχρονα με πολλαπλά νήματα.

```
def is_initial(state):
    return state.corner_pos == initial_state.corner_pos and \
           state.side_pos == initial_state.side_pos and \
           state.corner_ort == initial_state.corner_ort and \
           state.side_ort == initial_state.side_ort
```

Έλεγχος παρούσας κατάστασης και σύγκρισης με την αρχική. Επιστρέφει "True" εάν είναι η αρχική και "False εάν όχι.

Το τελευταίο κομμάτι στη διαδικασία επίλυσης είναι η εξαγωγή της αλληλουχίας των κινήσεων που οδήγησαν στη λύση.

- Naive: Μόλις βρεθεί η λύση, χρησιμοποιούμε το μονοπάτι από την αρχική κατάσταση στην τελική.
- BFS: Μόλις βρεθεί η λύση, εκτελούμε τον αλγόριθμο BFS στο δέντρο για να βρούμε την κοντινότερη διαδρομή από την αρχική ανακατεμένη κατάσταση στην τελική λυμένη

Προφανώς, η δεύτερη μέθοδος βρίσκει μικρότερα σε μήκος μονοπάτια από την πρώτη, καθώς η στοχαστική φύση του MCTS ενδεχομένως να εισάγει κυκλικές μεταβάσεις στο δέντρο.

4.6 Λεπτομέρειες Υλοποίησης σε Python

Αρχικά, αξίζει να σημειωθεί ότι όσον αφορά τη ρύθμιση των υπερπαραμέτρων του Deep Cube[17] δυστυχώς υπάρχει ελάχιστη διαθέσιμη πληροφορία, οπότε αρκετός πειραματισμός. Ο κώδικας αποτελείται από τέσσερα κομμάτια.

4.6.1 Περιβάλλον

Το πρώτο κομμάτι του κώδικα ασχολείται με το περιβάλλον με το οποίο αλληλεπιδρά ο πράκτορας. Ορίζει το observation space και την ακριβή αναπαράσταση της κατάστασης στην είσοδο του νευρωνικού δικτύου. Μερικές σημαντικές συναρτήσεις που περιέχει είναι:

```
class Action(enum.Enum):
    R = 0
    L = 1
    T = 2
    D = 3
    F = 4
    B = 5
    r = 6
    l = 7
    t = 8
    d = 9
    f = 10
    b = 11
```

Οι διαθέσιμες ενέργειες του πράκτορα. Κεφαλαία γράμματα υποδεικνύουν δεξιόστροφη περιστροφή.

```
def transform(state, action):
    assert isinstance(state, State)
    assert isinstance(action, Action)
    global _transform_map

    is_inv = action not in _transform_map
    if is_inv:
        action = inverse_action(action)
    c_map, s_map, c_rot, s_flp = _transform_map[action]
    corner_pos = _common._permute(state.corner_pos, c_map, is_inv)
    corner_ort = _common._permute(state.corner_ort, c_map, is_inv)
    corner_ort = _common._rotate(corner_ort, c_rot)
    side_pos = _common._permute(state.side_pos, s_map, is_inv)
    side_ort = state.side_ort
    if s_flp:
        side_ort = _common._permute(side_ort, s_map, is_inv)
        side_ort = _flip(side_ort, s_flp)
    return State(corner_pos=tuple(corner_pos), corner_ort=tuple(corner_ort),
                 side_pos=tuple(side_pos), side_ort=tuple(side_ort))
```

Η συνάρτηση αλλαγής κατάστασης δεδομένης κάποιας προηγούμενης κατάστασης και κάποιας ενέργειας.

```
def render(state):
    assert isinstance(state, State)
    global corner_colors, corner_maps, side_colors, side_maps

    sides = _init_sides()

    for corner, orient, maps in zip(state.corner_pos, state.corner_ort, corner_maps):
        cols = corner_colors[corner]
        cols = _common._map_orient(cols, orient)
        for (arr_idx, index), col in zip(maps, cols):
            sides[arr_idx][index] = col

    for side, orient, maps in zip(state.side_pos, state.side_ort, side_maps):
        cols = side_colors[side]
        cols = cols if orient == 0 else (cols[1], cols[0])
        for (arr_idx, index), col in zip(maps, cols):
            sides[arr_idx][index] = col

    return RenderedState(top=sides[0], left=sides[1], back=sides[2], front=sides[3],
                         right=sides[4], bottom=sides[5])
```

Rendering κατάστασης σε μορφή κατάλληλη για τον άνθρωπο.

```
encoded_shape = (20, 24)
```

Χρήσιμη λεπτομέρια, η κωδικοποίηση της κατάστασης του κύβου έχει διαστάσεις $20 * 24$.

Υπάρχει δυνατότητα εκτέλεσης του ADI και σε άλλα περιβάλλοντα αρκεί να πληρούνται ορισμένες προϋποθέσεις:

1. Οι καταστάσεις οφείλουν να είναι παρατηρήσιμες και η παρατήρηση των καταστάσεων οφείλει να διαφοροποιεί καταστάσεις μεταξύ τους. Στην περίπτωση παιχνιδιών με ελλειπή πληροφορία (όπως το poker) δεν μπορεί να εφαρμοστεί η μέθοδος χωρίς τροποποίηση του κώδικα.

2. Το πλήθος των ενεργειών οφείλει να είναι διακριτοποιήσιμο και πεπερασμένο. Στην περίπτωση που για παράδειγμα δώσουμε τη δυνατότητα στον πράκτορα να κινήσει κάποια πλευρά κατά [-90°, 90°] τότε αλλάζει ριζικά το πρόβλημα.

3. Είναι προϋπόθεση σωστής λειτουργίας του ADI και του MCTS να γνωρίζουμε το αποτέλεσμα της ενέργειας a στην κατάσταση s_t.

4.6.2 Εκπαίδευση Νευρωνικού Δικτύου

Όλες οι υπερπαράμετροι που χρησιμοποιήθηκαν για την εκπαίδευση παρουσιάζονται παρακάτω:

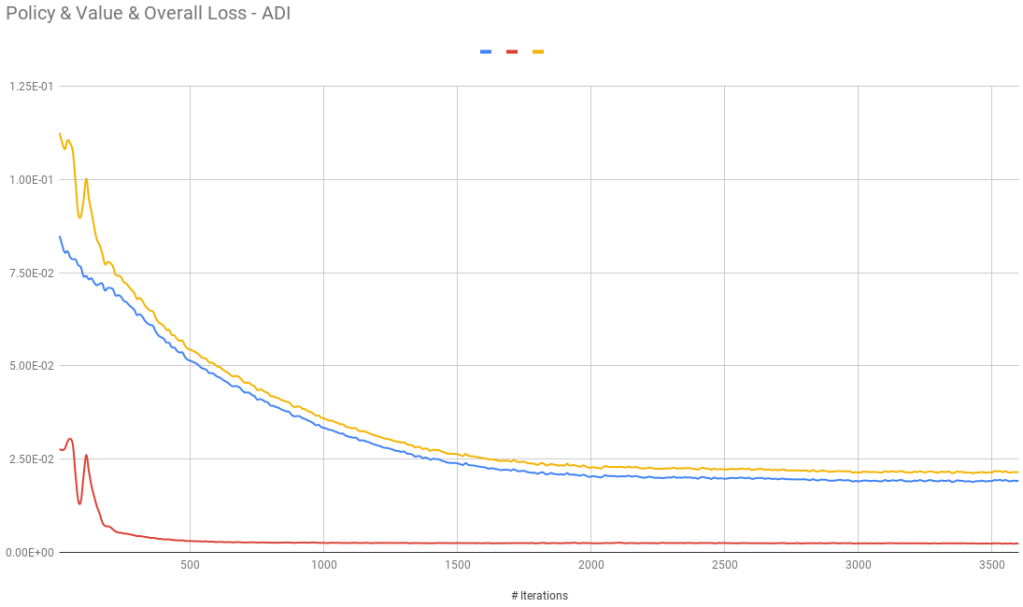
```
; Τρόπος υπολογισμού target τιμών
value_targets_method=zero_goal_value

; Όριο των batches για εκπαίδευση (train iterations)
max_batches=40000

; Χρήση cuda
cuda=True

; Learning rate
lr=1e-5

; Αριθμός κύβων σε κάθε batch
batch_size=10000

; Βάθος ανακατέματος
scramble_depth=200

; Συχνότητα ενημέρωσης κατά την εκπαίδευση
report_batches=10

; Συχνότητα αποθήκευσης μοντέλου
; checkpoint_batches=100
; enables LR decay
lr_decay=True

; LR decay γ
lr_decay_gamma=0.95

; Interval μεταξύ decays
lr_decay_batches=200
```

Οι παράμετροι του νευρωνικού δικτύου όπως αρχικοποιήθηκαν στην εκπαίδευση φαίνονται παρακάτω

```
Net(
  (body): Sequential(
    (0): Linear(in_features=480, out_features=4096, bias=True)
    (1): ELU(alpha=1.0)
    (2): Linear(in_features=4096, out_features=2048, bias=True)
    (3): ELU(alpha=1.0)
  )
  (policy): Sequential(
    (0): Linear(in_features=2048, out_features=512, bias=True)
    (1): ELU(alpha=1.0)
    (2): Linear(in_features=512, out_features=12, bias=True)
  )
  (value): Sequential(
    (0): Linear(in_features=2048, out_features=512, bias=True)
    (1): ELU(alpha=1.0)
    (2): Linear(in_features=512, out_features=1, bias=True)
  )
)
```

Κατά τη διαδικασία εκπαίδευσης έγινε στενή παρακολούθηση του policy & value loss και σταματήσαμε την εκπαίδευση μόλις σταθεροποιήθηκαν οι αποκλίσεις και επήλθε σύγκλιση.

```
Pushed new data in scramble buffer, new size = 80400
LR decrease to 7.737809375e-06
610: p_loss=4.690e-02, v_loss=2.773e-03, loss=4.967e-02, speed=4092.3 cubes/s
620: p_loss=4.636e-02, v_loss=2.744e-03, loss=4.910e-02, speed=4817.1 cubes/s
630: p_loss=4.592e-02, v_loss=2.691e-03, loss=4.861e-02, speed=4785.7 cubes/s
640: p_loss=4.552e-02, v_loss=2.717e-03, loss=4.823e-02, speed=4789.6 cubes/s
650: p_loss=4.487e-02, v_loss=2.717e-03, loss=4.759e-02, speed=4779.8 cubes/s
660: p_loss=4.453e-02, v_loss=2.623e-03, loss=4.715e-02, speed=4779.0 cubes/s
670: p_loss=4.461e-02, v_loss=2.713e-03, loss=4.732e-02, speed=4769.0 cubes/s
680: p_loss=4.443e-02, v_loss=2.715e-03, loss=4.715e-02, speed=4825.3 cubes/s
690: p_loss=4.379e-02, v_loss=2.654e-03, loss=4.645e-02, speed=4785.2 cubes/s
700: p_loss=4.302e-02, v_loss=2.622e-03, loss=4.564e-02, speed=4788.1 cubes/s
Pushed new data in scramble buffer, new size = 90450
710: p_loss=4.287e-02, v_loss=2.610e-03, loss=4.548e-02, speed=4090.3 cubes/s
720: p_loss=4.284e-02, v_loss=2.653e-03, loss=4.549e-02, speed=4754.1 cubes/s
730: p_loss=4.221e-02, v_loss=2.654e-03, loss=4.486e-02, speed=4776.6 cubes/s
740: p_loss=4.190e-02, v_loss=2.651e-03, loss=4.456e-02, speed=4747.9 cubes/s
750: p_loss=4.096e-02, v_loss=2.569e-03, loss=4.353e-02, speed=4768.3 cubes/s
760: p_loss=4.111e-02, v_loss=2.611e-03, loss=4.372e-02, speed=4803.0 cubes/s
770: p_loss=4.088e-02, v_loss=2.609e-03, loss=4.349e-02, speed=4827.6 cubes/s
780: p_loss=4.039e-02, v_loss=2.607e-03, loss=4.300e-02, speed=4799.5 cubes/s
790: p_loss=4.021e-02, v_loss=2.567e-03, loss=4.277e-02, speed=4773.3 cubes/s
800: p_loss=3.935e-02, v_loss=2.573e-03, loss=4.192e-02, speed=4781.1 cubes/s
```



Σχήμα 4.12: Policy - Value - Overall Loss, ADI

4.6.3 Solver

Η διαδικασία εκπαίδευσης δημιουργεί ένα model file με τα διαμορφωμένα βάρη του νευρωνικού δικτύου. Μετά από πολλές ώρες εκπαίδευσης στο Google Colaboratory και κατόπιν σταθεροποίησης τιμών, χρησιμοποιείται η δενδρική αναζήτηση Monte Carlo για την επίλυση του κύβου. Ο solver έχει τις εξής δηνατότητες:

- Επίλυση κάποιου κύβου του οποίου το ανακάτεμα προήλθε από τον άνθρωπο. Το πέρασμα γίνεται με την μορφή λίστας (βλ. class Action) για παράδειγμα [2,5,1,2,4] το οποίο μεταφράζεται σε κινήσεις που γίνονται σε κάποιον λυμένο κύβο πριν αναλάβει την επίλυση ο solver. Απαιτείται η χρήση της εντολής -p ως argument πριν την εκκίνηση του προγράμματος
- Ανάγνωση του scramble από αρχείο txt με την εντολή -i στο όρισμα.
- Παραγωγή κύβου με τυχαίο βάθος ανακατέματος και προσπάθεια επίλυσης από τον solver
- Παραγωγή γραφικών παραστάσεων με αυξανόμενο βάθος ανακατέματος και παραγωγή αρχείων csv με το αποτέλεσμα.

Η παρουσίαση των αποτελεσμάτων θα γίνει στο επόμενο κεφάλαιο καθώς και η σύγκριση αυτών με μία νέα πρόταση χρήσης κάποιας μετρικής της εντροπίας του κύβου για αποδοτικότερη αναζήτηση της λύσης στο MCTS.

Κεφάλαιο 5

Εντροπία του Rubik's Cube

Στο κεφάλαιο αυτό εισάγεται η έννοια της εντροπίας του κύβου του Ρούμπικ. Ο λόγος που εισάγεται η έννοια αυτή προέρχεται από την ανάγκη έκφρασης της απόστασης από τη λύση με κάποιο τρόπο ώστε να βελτιώσουμε το MCTS. Η βελτίωση αυτή βασίζεται στην γενικότερη παραδοχή ότι ο λυμένος κύβος έχει ελάχιστη εντροπία και ότι όσο ανακατεύουμε τον κύβο, τόσο αυξάνεται η εντροπία του (μέχρι κάποιο βαθμό).

5.1 Πολυπλοκότητα Kolmogorov

Στην αλγοριθμική θεωρία πληροφορίας (υποπεδίο της επιστήμης υπολογιστών και των μαθηματικών) η πολυπλοκότητα Kolmogorov ενός αντικειμένου, όπως ένα κομμάτι κειμένου, ή μιας λίστας, ή ενός τανυστή (ο κύβος του Ρούμπικ στην περίπτωσή μας) ορίζεται ως το μήκος του μικρότερου δυνατού προγράμματος (σε μία προκαθορισμένη γλώσσα προγραμματισμού Η/Υ) που παράγει το αντικείμενο ως έξοδο. Αποτελεί μία μετρική απαραίτητων υπολογιστικών πόρων για τον ακριβή καθορισμό του αντικειμένου, γνωστή ως περιγραφική πολυπλοκότητα, πολυπλοκότητα Kolmogorov-Chaitin, αλγοριθμική πολυπλοκότητα, αλγοριθμική εντροπία ή πολυπλοκότητα μεγέθους προγράμματος. Ονομάστηκε προς τιμήν του Andrey Kolmogorov ο οποίος έκανε την πρώτη σχετική δημοσίευση το 1963.[2]

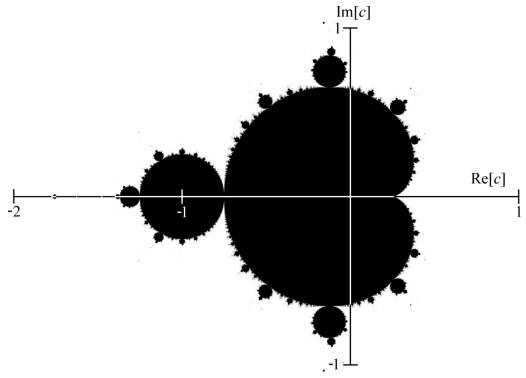
Παράδειγμα

Έστω οι δύο παρακάτω αλφαριθμητικές σειρές:

Παράδειγμα 1: abababababababababababababababab

Παράδειγμα 2: 4c1j5b2p0cv4w1x8rx2y39umgw5q85s7

Η πρώτη σειρά έχει μία σύντομη γλωσσική περιγραφή: "Το αβ 16 φορές" το οποίο αποτελείται από 11 χαρακτήρες. Η δεύτερη δεν έχει καμία προφανή απλή περιγραφή άλλη από την ίδια την καταγραφή της ίδιας της συμβολοσειράς, η οποία αποτελείται από 32 χαρακτήρες. Κάθε σειρά έχει τουλάχιστον μία περιγραφή. Για παράδειγμα, η δεύτερη σειρά είναι έξοδος του παρακάτω προγράμματος:



Σχήμα 5.1: Mandelbrot Set Fractal. [11]

function GenerateExample2String() $\rightarrow return : 4c1j5b2p0cv4w1x8rx2y39umgw5q85s7$

Η πρώτη σειρά είναι έξοδος ενός πιο σύντομου ψευδοπρογράμματος:

function GenerateExample1String() $\rightarrow return : (ab) * 16$

Εάν η περιγραφή $d(s)$ μιας σειράς είναι ελάχιστου μήκους, καλείται ελάχιστη περιγραφή της s. Συνεπώς, το μήκος της $d(s)$ ονομάζεται πολυπλοκότητα Kolmogorov της σειράς s και γράφεται $K(s)$. Συμβολικά:

$K(S) = |d(s)|$

Η έννοια της πολυπλοκότητας Kolmogorov μπορεί να χρησιμοποιηθεί για τη διατύπωση και την απόδειξη παρόμοιων αποτελεσμάτων με το διαγώνιο επιχείρημα του Cantor, τη θεωρία μη-πληρότητας του Gödel και το πρόβλημα τερματισμού του Turing. Συγκεκριμένα, για σχεδόν όλα τα αντικείμενα (συμπεριλαμβανομένης και οποιασδήποτε αναπαράστασης του κύβου του Ρούμπικ) δεν είναι δυνατόν να υπολογίσουμε την πολυπλοκότητα Kolmogorov. Μάλιστα, δεν είναι ακόμα δυνατόν να υπολογίσουμε ούτε κάποιο κατώτατο όριο για την πολυπλοκότητα Kolmogorov ενός αντικειμένου. Αν και αποτελεί μία από τις πιο αξιόπιστες μετρικές έκφρασης της οργάνωσης του κύβου, η έλλειψη υπολογισμού κλειστού τύπου για κάθε διάταξη του κύβου οδηγεί στην έκφραση μίας διαφορετικής μαθηματικά αυστηρής έννοιας της πολυπλοκότητας η οποία να επιτρέπει τον ακριβή υπολογισμό πολυπλοκότητας του κύβου του Ρούμπικ.

5.2 Εντροπία Shannon

Επικεντρωνόμαστε στη μελέτη τυχαίων φαινομένων που περιγράφονται από πιθανοτικά μοντέλα. Ένα σημαντικό χαρακτηριστικό των μοντέλων αυτών είναι η προσπάθεια σύλληψης της εγγενής αβεβαιότητας ενός συστήματος. Το ερώτημα που δημιουργείται εδώ είναι πως μπορεί να μετρηθεί η "αβεβαιότητα" του κύβου του Ρούμπικ. Η μαθηματική θεωρία της πληροφορίας αναπτύχθηκε στο πλαίσιο της θεωρίας τηλεπικοινωνιών για να απαντήσει δύο θεμελιώδη ερωτήματα.

1. Ποια είναι η απόλυτη δυνατή συμπίεση δεδομένων;

2. Ποιος είναι ο απόλυτος ρυθμός μετάδοσης πληροφορίας;

Ο Claude Shannon το 1948 έθεσε τα θεμέλια της θεωρίας της πληροφορίας με τη δημοσίευσή του.[16]

Ανέπτυξε την εντροπία ως μέτρο αβεβαιότητας. Προηγούμενες έρευνες είχαν πραγματοποιηθεί από τον Nyquist το 1924 και τον Hartley το 1928. Παρόλο που ο Shannon ανέπτυξε την έννοια της εντροπίας στα πλαίσια τηλεπικοινωνιακών συστημάτων, υπήρξε ευρεία εφαρμογή της έννοιας αυτής σε διαφορετικούς κλάδους μαθηματικών, φυσικής, βιολογίας και κοινωνικών επιστημών. Ο λόγος της αξιοσημείωτης αυτής επιτυχίας βασίστηκε στο γεγονός ότι υπήρξε η δυνατότητα ποσοτικοποίησης της αβεβαιότητας που υπάρχει σε πιθανοκρατικά προβλήματα. Στη μαθηματική θεωρία πληροφορίας, υπάρχει αρκετό ενδιαφέρον μέτρησης του περιεχομένου της μεταδιδόμενης πληροφορίας από μία γνωστή ομάδα πιθανών μηνυμάτων. Ο Hartley περιέγραψε ένα λογαριθμικό μέτρο για το περιεχόμενο πληροφορίας μηνυμάτων με ομοιόμορφη κατανομή. Ο Shannon έδωσε μία απάντηση για την μέτρηση αβεβαιότητας σε μη-ισοπίθανα μηνύματα γνωστή ως εντροπία Shannon, η οποία έχει την ίδια μαθηματική έκφραση με την εντροπία που ορίζεται στη στατιστική.

5.2.1 Παράδειγμα Υπολογισμού εντροπίας μίας σειράς αριθμών

Έστω η σειρά: abababcd

Υπολογίζουμε την εντροπία H(X) σύμφωνα με την παρακάτω σχέση

$$H(X) = \sum_{i=1}^{n} p(x_i) I(x_i) = \sum_{i=1}^{n} p(x_i) \log_b \frac{1}{p(x_i)} = -\sum_{i=1}^{n} p(x_i) \log_b p(x_i) \tag{5.1}$$

Τα σύμβολα που υπάρχουν στη λίστα είναι: [a,b,c,d]

Οι συχνότητες εμφάνισης των συμβόλων είναι:

- a: 0.375
- b: 0.375
- c: 0.125
- d: 0.125

Η εντροπία Shanon υπολογίζεται ως εξής:

$H(X) = -[(0.375 \log(20.375)) + (0.375 \log(20.375)) + (0.125 \log(20.125)) + (0.125 \log(20.125))]$

$H(X) = -[(-0.531) + (-0.531) + (-0.375) + (-0.375)]$

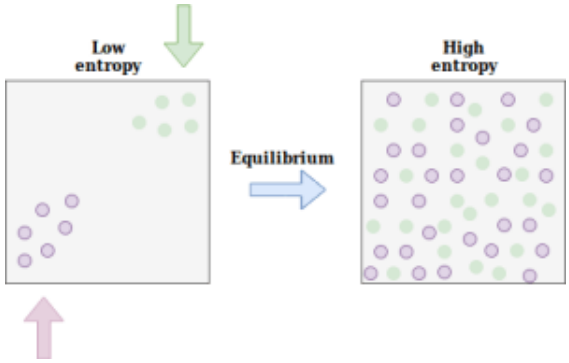
$$H(X) = -[-1.81128]$$

$$H(X) = 1.81128$$

Η πληροφορία που λαμβάνεται από την παραπάνω τιμή ταυτίζεται με τον ελάχιστο αριθμό bits/symbol που χρειάζεται για να κωδικοποιηθεί η πληροφορία σε δυαδική μορφή (με 2 ως βάση του λογαρίθμου). Με δεδομένη την παραπάνω εντροπία Shannon στρογγυλοποιημένη προς τα πάνω, κάθε σύμβολο κωδικοποιείται χρησιμοποιώντας 2 bit και απαιτούνται συνολικά 16 bit για τη βέλτιστη κωδικοποίηση της παραπάνω αλφαριθμητικής σειράς

5.2.2 Πρόβλημα Εφαρμογής Εντροπίας Shannon στο Rubik's Cube

Η ελάττωση εντροπίας θα είχε νόημα σε εφαρμογές πιθανοθεωρητικών προβλημάτων όπου η διάταξη των σωματιδίων είναι αδιάφορη (όπως το φημισμένο παράδειγμα της διάχυσης)



Σχήμα 5.2: Αύξηση της εντροπίας ενός συστήματος [20]

Στην περίπτωση του κύβου του Ρούμπικ, ο διαχωρισμός των χρωμάτων δεν γίνεται ανεξάρτητα καθώς ένα cubelet έχει πάντα δύο ή ακόμα και τρία αυτοκόλλητα και η μετακίνησή του συμπαρασέρνει την μετακίνηση πολλών χρωμάτων ταυτόχρονα. Καταφεύγουμε σε μία απλούστερη έκφραση της εντροπίας του κύβου για την υλοποίησή του στην γλώσσα προγραμματισμού Python. Η υλοποίηση αυτή μετράει το πλήθος των σωστά τοποθετημένων cubelets.

```
def cube_entropy(cube_env, state):
    assert isinstance(cube_env, cubes.CubeEnv)
    init_render = cube_env.render(cube_env.initial_state)
    render = cube_env.render(state)
    total = 0
    match = 0
    for s1, s2 in zip(init_render, render):
        for v1, v2 in zip(s1, s2):
            if v1 == v2:
                match += 1
            total += 1
    return (total-match)/total
```

Κώδικας έκφρασης εντροπίας του κύβου. Υπολογίζεται ο λόγος των κελιών που είναι σωστά τοποθετημένα προς τον συνολικό αριθμό θέσεων. Έχει σύνολο τιμών: $[0, 1]$

5.3 Χρήση εντροπίας στο Monte Carlo Tree Search

Η εντροπία του κύβου θα χρησιμοποιηθεί στο στάδιο της αναζήτησης Monte Carlo και συγκεκριμένα στο σημείο όπου δίνεται ο κύβος στον agent και προσπαθεί από τη ρίζα να βρει την κατάλληλη πορεία μέχρι να φτάσει σε κάποιο φύλλο. Υπενθυμίζουμε από τη θεωρία ότι:

$A_t = \text{argmax}_a(U_{st}(a) + W_{st}(a))$, όπου $U_{s_t}(a) = cP_{s_t}(a)\frac{\sqrt{\sum_{a'} N_{s_t}(a')}}{1+N_{s_t}(a)}$

- $N_{s_t}(a)$ εκφράζει το πλήθος τον φορών που επιλέχθηκε η ενέργεια a από την κατάσταση s_t
- $P_{s_t}(a)$ η κατανομή πολιτικής του μοντέλου για την κατάσταση s_t
- $W_{s_t}(a)$ η μέγιστη τιμή που επιστρέφει το μοντέλο για όλα τα παιδιά της κατάστασης s_t δεδομένου ότι επιλέχθηκε η ενέργεια a

Για κάθε κατάσταση s_t όπως υπολογίζουμε το distribution policy με τη βοήθεια του νευρωνικού δικτύου, υπολογίζουμε επίσης και την κατανομή της εντροπίας δοκιμάζοντας τις 12 δυνατές ενέργειες και υπολογίζοντας κάθε φορά την εντροπία του κύβου μέσω της συνάρτησης $\text{cube}_entropy$.

Στη συνέχεια αλλάζουμε (ελαφρώς) το U_{st} αθροίζοντας την κατανομή της εντροπίας:

$$U_{s_t}(a) = C(P_{s_t} - \frac{V_{s_t}E_{s_t}}{1+N_{s_t}(a)})\frac{\sqrt{\sum a' N_{s_t}(a')}}{1+N_{s_t}(a)} \tag{5.2}$$

5.3.1 Αιτιολόγηση προσθετικού όρου $-\frac{V_{s_t}E_{s_t}}{1+N_{s_t}(a)}$

Πρόθεσή μας είναι η υπόδειξη στον πράκτορα κάποιας ελπιδοφόρας κατεύθυνσης μικρότερης εντροπίας. Ο πράκτορας ήδη έχει κάποια κατανομή πολιτικής (P_{s_t}) από το εκπαιδευμένο νευρωνικό δίκτυο η οποία του προτείνει μία κίνηση.

Ο όρος $-\frac{V_{s_t}E_{s_t}}{1+N_{s_t}(a)}$ τροποποιεί την κατανομή πολιτικής και ουσιαστικά εμφανίζει στον πράκτορα μία νέα κατανομή πολιτικής:

$P'_{s_t} = P_{s_t} - \frac{V_{s_t}E_{s_t}}{1+N_{s_t}(a)}$

η οποία αποτελείται πλέον από δύο μέρη.

- P_{s_t} - Κατανομή πολιτικής από ADI
- $-\frac{V_{s_t}E_{s_t}}{1+N_{s_t}(a)}$ - Κατανομή εντροπιών για υποβοήθηση στο decision making

Ο όρος E_{s_t} είναι η κατανομή εντροπίας που δημιουργείται αφού δοκιμαστούν οι 12 δυνατές κινήσεις στην κατάσταση s_t

Ο αρνητικός όρος υπάρχει γιατί κινούμαστε προς αρνητικότερη εντροπία.

Ο όρος V_{s_t} υπάρχει γιατί όσο πιο μακρυά βρισκόμαστε από τη λύση του κύβου, τόσο λιγότερο λαμβάνουμε υπόψιν την εντροπία του κύβου

Ο όρος $(1 + N_{s_t}(a))$ υπάρχει για την σταδιακή ελάττωση της συνεισφοράς του όρου στην πολιτική του πράκτορα, κάτι το οποίο γίνεται για λόγους εγγύησης της σύγκλισης προς τη λύση στο άπειρο.

5.4 Αξιολόγηση Απόδοσης

Παρακάτω παραθέτουμε τα διαγράμματα κατά την αναζήτηση λύσης κάνοντας χρήση του Monte Carlo Tree Search.

Γίνεται χρήση του αλγορίθμου στους κύβους:

- 2x2
- 3x3

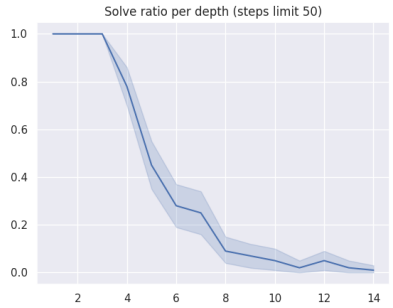
και για καθένα από τους κύβους πραγματοποιείται η αναζήτηση:

- Με υποβοήθηση εντροπίας
- Χωρίς υποβοήθηση εντροπίας

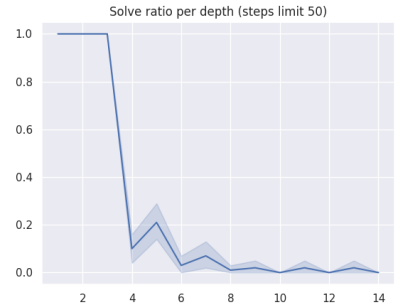
Ως μεταβλητή επιλέξαμε το όριο των μεταβάσεων στην αναζήτηση Monte Carlo πριν αποφανθούμε για την ικανότητα του πράκτορα να λύνει τον κύβο. Η μεταβλητή max-steps είναι ο μέγιστος αριθμός μεταβάσεων στο δέντρο και πήρε τις εξής τιμές:

- 50
- 150
- 300
- 500
- 1000 (Η απόδοση φτάνει σε κορεσμό για max-steps > 1000)

5.4.1 Κύβος 2x2

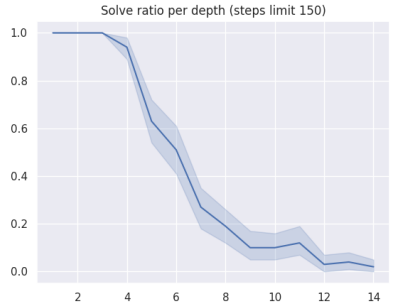


(α′) Classic

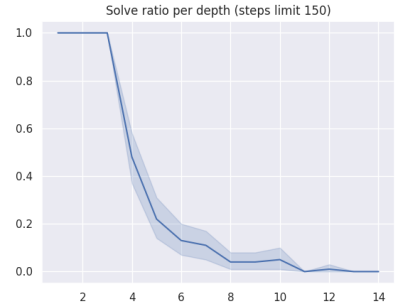


(β′) Classic + Entropy

Σχήμα 5.3: cube:2x2, max-steps: 50

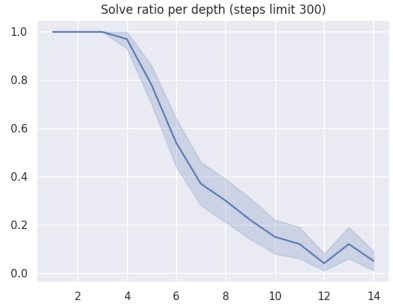


(α′) Classic

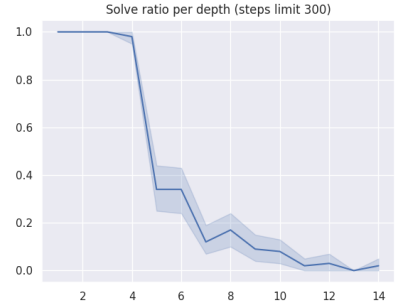


(β′) Classic + Entropy

Σχήμα 5.4: cube:2x2, max-steps: 150

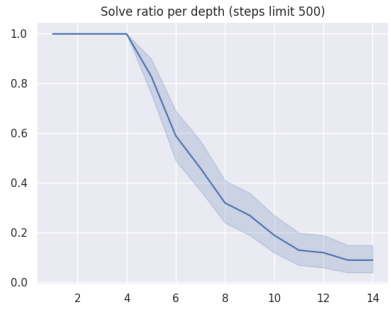


(α′) Classic

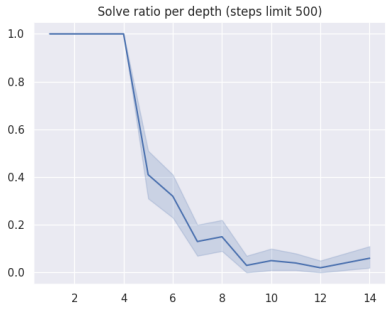


(β′) Classic + Entropy

Σχήμα 5.5: cube:2x2, max-steps: 300

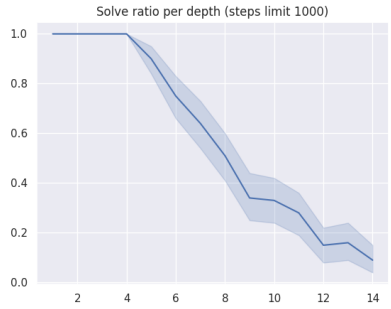


(α′) Classic

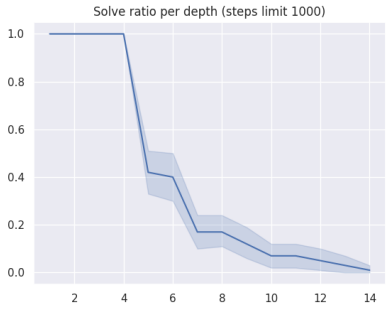


(β′) Classic + Entropy

Σχήμα 5.6: cube:2x2, max-steps: 500



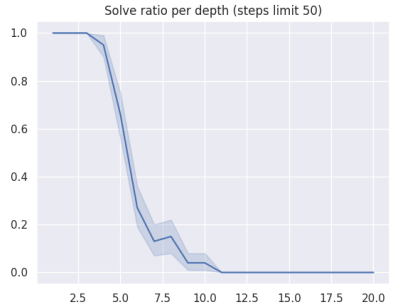
(α′) Classic



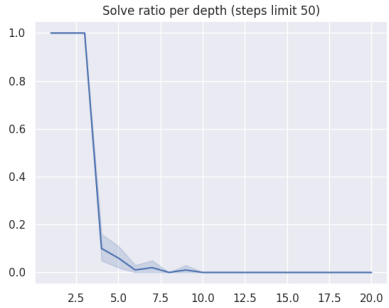
(β′) Classic + Entropy

Σχήμα 5.7: cube:2x2, max-steps: 1000

5.4.2 Κύβος 3x3

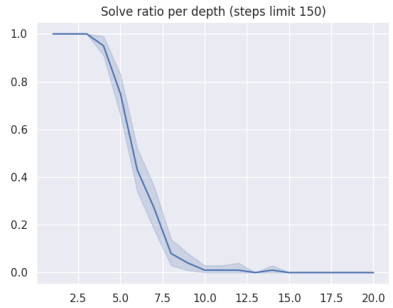


(α′) Classic

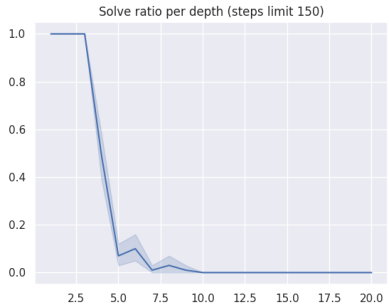


(β′) Classic + Entropy

Σχήμα 5.8: cube:3x3, max-steps: 50

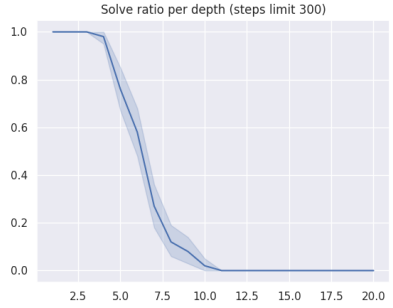


(α′) Classic

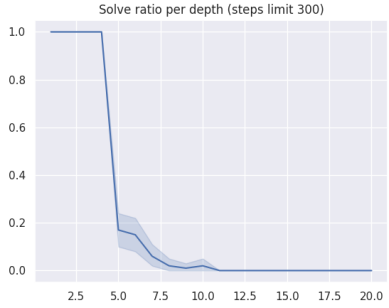


(β′) Classic + Entropy

Σχήμα 5.9: cube:3x3, max-steps: 150

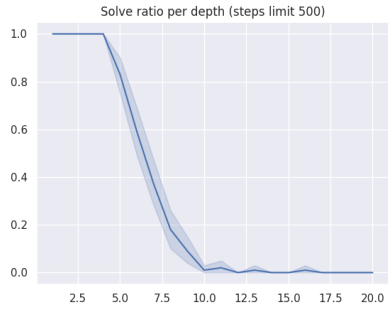


(α′) Classic

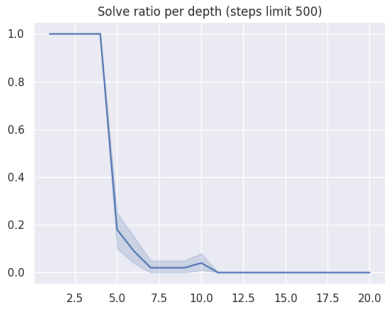


(β′) Classic + Entropy

Σχήμα 5.10: cube:3x3, max-steps: 300

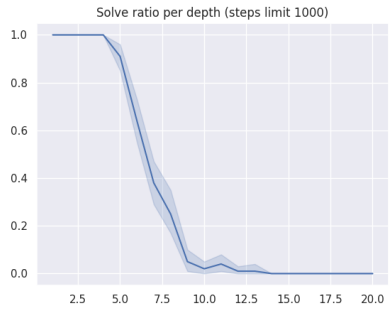


(α′) Classic

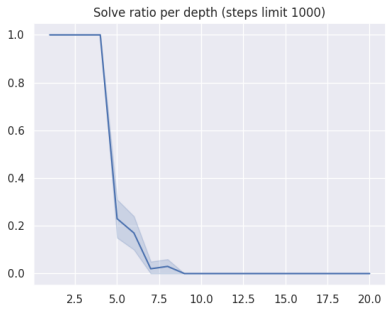


(β′) Classic + Entropy

Σχήμα 5.11: cube:3x3, max-steps: 500



(α′) Classic



(β′) Classic + Entropy

Σχήμα 5.12: cube:3x3, max-steps: 1000

5.5 Ανάλυση Αποτελεσμάτων

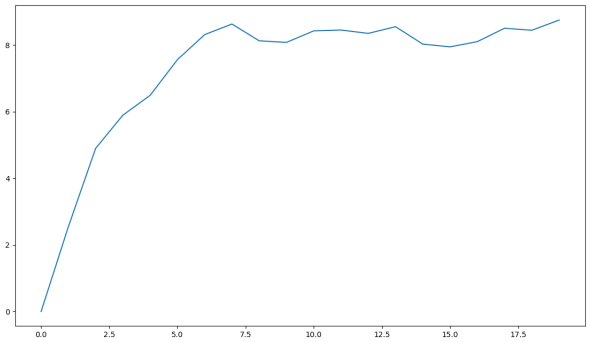
Συγκρίνουμε την απόδοση των δύο μεθοδολογιών κατασκευάζοντας γραφήματα βάθους ανακατέματος και ποσοστό επίλυσης δοκιμάζοντας 100 κύβους σε κάθε βάθος. Αριστερά παρουσιάζονται οι καμπύλες απόδοσης της αμιγώς Monte Carlo μεθόδου και δεξιά οι καμπύλες απόδοσης αναζήτησης με κίνητρο ελάττωσης της εντροπίας. Κάνουμε τις εξής δύο παρατηρήσεις:

- Η αύξηση των max-steps βελτιώνει τα ποσοστά λύσης του κύβου μέχρι ένα σημείο. Πειραματικά διαπιστώθηκε ότι η απόδοση φτάνει σε κορεσμό μετά από max-steps 1000.

- Η συγκεκριμένη έκφραση της εντροπίας ενδεχομένως να μην είναι αρκετά αποτελεσματική γιατί ο αριθμός των σωστών cubelets δεν είναι ανάλογος της πραγματικής απόστασης από τη λύση του κύβου. Αυτό μεταφράζεται σε παροχή ωφέλιμης πληροφορίας μόνο σε μικρές αποστάσεις από τη λύση όπου η αλλαγή της εντροπίας που θεωρήσαμε είναι δραστική.

5.5.1 Πρόβλημα Διακύμανσης της Εντροπίας

Κατά τη διεξαγωγή των πειραμάτων παρατηρήσαμε κάτι πολύ σημαντικό. Η εντροπία του κύβου καθώς εκτελούμε μία γνωστή ιδανική πορεία προς τη λύση δεν είναι μονότονη συνάρτηση. Επίσης, η μετρική της εντροπίας που χρησιμοποιήθηκε δυστυχώς δεν αλλάζει γραμμικά και (ακόμα χειρότερα) σταθεροποιείται μετά από περίπου 7 κινήσεις βάθος. Αυτό σημαίνει ότι η συνεισφορά της σε μεγάλα βάθη είναι αμελητέα, αν όχι ανεπιθύμητη. Ακόμα, η συμπεριφορά της εντροπίας αλλάζει ανάλογα με τον εκάστοτε ορισμό εντροπίας του κύβου. Το πιο ανησυχητικό φαινόμενο που παρατηρήθηκε είναι ότι η εντροπία του κύβου σε ένα τυχαίο ανακάτεμα 20 κινήσεων ενώ στην αρχή αυξάνεται, παραμένει σταθερή μετά από μερικές κινήσεις.



Σχήμα 5.13: Μέσος όρος εντροπίας συναρτήσει του βάθους ανακατέματος για 100 τυχαία ανακατέματα του κύβου.

Λόγω της φύσης του κύβου και εκτελώντας ειδικές αναγκαίες κινήσεις οι οποίες κάνουν κάπου είδους reshuffling των cubelets όπως αναφέρθηκαν και στο Κεφάλαιο 3, είναι αναγκαία η έντονη διακύμανση της εντροπίας προκειμένου να οδηγηθεί ο κύβος στη λυμένη του κατάσταση με βέλτιστο τρόπο. Συμπεραίνουμε ότι δεν αρκεί η βραχυπρόθεσμη απαίτηση άμεσης ελαχιστοποίησης της εντροπίας για την εύρεση της αποδοτικότερης λύσης του κύβου.

Κεφάλαιο 6

Κατευθύνσεις Μελλοντικής Έρευνας

Σαν κατακλείδα, γίνεται μία αναφορά διαφορετικών τρόπος αντιμετώπισης του προβλήματος καθώς και αναφορά διάφορων ιδεών και προτάσεων για βελτίωση στο μέλλον.

6.1 Προτάσεις Βελτίωσης Αλγορίθμου ADI με MCTS

Υπάρχουν πολλές κατευθύνσεις που αξίζει να δοκιμαστούν για ενδεχόμενη βελτίωση των αποτελεσμάτων.

- Η πολυπλοκότητα του κύβου ενδεχομένως να επωφελείται από συνελίξεις. Θα είχε νόημα λοιπόν να τροποποιηθεί η αρχιτεκτονική του δικτύου από Dense σε CNN

- Ταλαντώσεις και αστάθεια κατά την εκπαίδευση ενδεχομένως να οφείλονται στο πρόβλημα που παρουσιάζουν μερικοί αλγόριθμοι RL σε συνδυαστικά προβλήματα. Μία λύση που προβλέπεται για αποθορυβοποίηση σε αντίστοιχες περιπτώσεις είναι η χρήση target network.

- Ένας διαφορετικός ορισμός εντροπίας ο οποίος θα έκανε πιο αξιόπιστη την εκτίμηση απόστασης από τη ρίζα σίγουρα θα έφερνε θετικότερα αποτελέσματα.

- Χρήση Hierarchical Reinforcement Learning [18] που θα βοηθήσει ενδεχομένως στην εκμάθηση ισχυρών πολιτικών, κάνοντας χρήση intrinsic motivation

- Χρήση Macro Operators [8] οι οποίοι θα συνδυάζουν ένα πλήθος κινήσεων μαζί για να κτίσουν μία μακρο-εντολή η οποία ενδεχομένως να αποτελείται από πολλές κινήσεις. Μοιάζουν φιλοσοφικά με τα commutators που αναφέρθηκαν στο Κεφάλαιο 3

6.2 Ιδέες για επίλυση του Rubik's Cube με άλλους τρόπους

- Recurrent Neural Networks including LSTM-like Networks

  Η χρήση LSTM επιτρέπει στα δίκτυα να απομνημονεύουν καλύτερα την κατάσταση που παρατήρησαν. Παρόλο που αυξάνει τον χρόνο εκπαίδευσης, δεν αυξάνει δραστικά τον χρόνο αναζήτησης Monte Carlo.

- Διαφοροποίηση του αλγόριθμου αναζήτησης σε (A*)

  Ενδεχομένως να υπάρχουν ελπιδοφόρες παραλλαγές οι οποίες δεν θα χρησιμοποιούν ρητή γνώση ομαδο-θεωρίας του κύβου.

6.3 Ιδέες Αντιμετώπισης των Sparse Rewards

Ο κύβος του Ρούμπικ, από την σκοπιά της βαθιάς ενισχυτικής μάθησης, αποτελεί μία κλασσική περίπτωση "sparse reward problem". Αυτό σημαίνει ότι η ανταμοιβή που παίρνει κάποιος πράκτορας κατά την εκπαίδευση του είναι αραιή. Κάνοντας την κατάσταση ακόμα χειρότερη, ενδεχομένως να μην πάρει ποτέ θετική ανταμοιβή εάν ξεκινήσει από έναν τυχαίο κύβο και συμπεριφερθεί τυχαία. Τρόποι αντιμετώπισης της αραιής ανταμοιβής παρουσιάζονται παρακάτω.

6.3.1 Unsupervised Auxiliary Tasks

Οι σύγχρονοι πράκτορες βαθιάς ενισχυτικής μάθησης που έχουν πετύχει αποτελέσματα state-of-the-art μέσω μεγιστοποίησης κάποιας συσσωρευτικής ανταμοιβής. Παρόλα αυτά, περιβάλλοντα όπως οι κύβος του Ρούμπικ, ο οποίος έχει μία λυμένη κατάσταση μέσα σε $4 * 10^{19}$ άλυτες, δεν διευκολύνει το ήδη δύσκολο έργο του πράκτορα. Θα είχε ενδιαφέρον να οριστούν ενδιάμεσες ανταμοιβές κατά την εκπαίδευση πέρα από το ¨ναι' ή ¨όχι' του περιβάλλοντος, όσον αφορά την επίλυση. Στην περίπτωση ορισμού ενδιάμεσων ανταμοιβών, ένας πράκτορας θα μπορούσε να μεγιστοποιήσει επίσης πολλές άλλες ψευδο-ανταμοιβές ταυτόχρονα μέσω ενισχυτικής μάθησης. [12]

6.3.2 Credit Assignment Problem

Σε πολλά πραγματικά σενάρια, οι ανταμοιβές του πράκτορα είναι εξαιρετικά αραιές και ενδεχομένως να απουσιάζουν και εντελώς, όπως είδαμε στον κύβο του Ρούμπικ. Σε τέτοιες περιπτώσεις, η περιέργεια μπορεί να χρησιμεύσει ως εγγενές σήμα ανταμοιβής για να επιτρέψει στον πράκτορα να εξερευνήσει το περιβάλλον του και να μάθει δεξιότητες που μπορεί να είναι χρήσιμες αργότερα κατά την εκμάθησή του. Η περιέργεια ορίζεται ως το σφάλμα στην ικανότητα ενός πράκτορα να προβλέψει την συνέπεια των δικών του ενεργειών.

Βιβλιογραφία

[1] Ethem Alpaydin. Introduction to machine learning 3rd edition. 2015.

[2] Shiryayev A.N. On tables of random numbers. in: Shiryayev a.n. (eds) selected works of a. n. kolmogorov. mathematics and its applications (soviet series). *Springer, Dordrecht*, σελίδες 176–183, 1993.

[3] Richard Bellman. Dynamic programming. *Princeton University Press*, 1957.

[4] DP Bertsekas. Convex optimization algorithms. *A Scientific*, 2015.

[5] David Blackwell. Discounted dynamic programming. *Ann. Math. Statist.*, σελίδα 226–235, χ.χ.

[6] Rémi Coulom. Efficient selectivity and backup operators in monte-carlo tree search. *Springer*, σελίδες 72–83, 2006.

[7] John R. Doyle. Survey of time preference, delay discounting models. *Judgment and Decision Making*, 2013.

[8] Richard E. Korf. Macro-operators: A weak method for learning. *Artificial Intelligence*, 26:35–77, 1985.

[9] David Harris και Sarah Harris. Digital design and computer architecture (2nd ed.). *Morgan Kaufmann*, σελίδα 129, 2018.

[10] David Joyner. Adventures in group theory: Rubik's cube, merlin's machine, and other mathematical toys. 2002.

[11] Benoit Mandelbrot. Fractal aspects of the iteration of z -¿ lz(1-z) for complex l,z. *Annals of the New York Academy of Sciences*, 2004.

[12] Wojciech Marian Czarnecki Tom Schaul Joel Z Leibo David Silver Koray Kavukcuoglu Max Jaderberg, Volodymyr Mnih. Reinforcement learning with unsupervised auxiliary tasks. 2017.

[13] Andrew G. Barto Richard S. Sutton. Introduction to reinforcement learning. 2018.

[14] Sheldon M. Ross. Introduction to stochastic dynamic programming: Probability and mathematical. *Academic Press, Inc.*, 1983.

[15] Hinton Geoffrey E. Williams Ronald J. Rumelhart, David E. Learning representations by back-propagating errors. *Nature*, 1986.

[16] C. E. SHANNON. A mathematical theory of communication. *The Bell System Technical Journal*, 1948.

[17] Alexander Shmakov Pierre Baldi Stephen McAleer, Forest Agostinelli. Solving the rubik's cube without human knowledge. *NIPS*, 2018.

[18] Ardavan Saeedi Joshua B. Tenenbaum Tejas D. Kulkarni, Karthik R. Narasimhan. Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation. 2016.

[19] David Silver Andrei A Rusu Joel Veness Marc G Bellemare Alex Graves Martin Riedmiller Andreas K Fidjeland Georg Ostrovski et al Volodymyr Mnih, Koray Kavukcuoglu. Human-level control through deep reinforcement learning. *Nature*, 2015.

[20] S. Wagner. Comparing clusterings – an overview. *Technical report, Karlsruher Institut fur Technologie*, 2007.

Συντομογραφίες - Αρκτικόλεξα - - Ακρωνύμια

ΕΜΠ	Εθνικό Μετσόβιο Πολυτεχνείο
DRL	Deep Reinforcement Learning
RL	Reinforcement Learning
MDP	Markov Decision Process
BOT	Bellman Optimality Equation
QTM	Quarter Turn Metric
HTM	Half Turn Metric
OBTM	Outer Block Turn Metric
NLP	Natural Language Processing
JPEG	Joint Photographic Experts Group
MCTS	Monte Carlo Tree Search
ADI	Auto-didactic Iteration
CNN	Convolutional Neural Network

Απόδοση ξενόγλωσσων όρων

Απόδοση	Ξενόγλωσσος όρος
δοκιμή-και-σφάλμα	trial-and-error
μηχανική μάθηση	reinforcement learning
επιβλεπούμενη μάθηση	supervised learning
συμβιβασμός	trade-off
εξερεύνησης	exploration
εκμετάλλευσης	exploitation
κατάρα διαστασιμότητας	curse of dimensionality
μεγάλη συσσωρευμένη ανταμοιβή	large accumulated reward
πεπερασμένος ορίζοντας	finite horizon
άπειρος ορίζοντας	ifinite horizon
παράγοντας ελάττωσης	discount factor
μέθοδοι με χρήση μοντέλου	model-based methods
μέθοδοι χωρίς τη χρήση μοντέλου	model-free methods
πίσω-διάδοση	backpropagation
εποχές λήψης αποφάσεων	decision epochs
καταστάσεις Q	Q-states
συνάρτηση Bellman	Bellman optimality equation
μειωμένη χρησιμότητα	discounted utility
επαναληπτική μέθοδος υπολογισμού αξίας	value iteration
επαναληπτική μέθοδος υπολογισμού πολιτικής	policy iteration
μεταγωγέας	commutator
αλλαγή προσανατολισμού	orientation swap
κυβάκι	cubelet
ομάδες	groups
θεωρία ομάδων	group theory
ομάδα μεταθέσεων	permutation group
τρόπος υπολογισμού περιστροφής πλευράς	face turns

μετρική υπολογισμού περιστροφής 90°	quarter-turn metric
μετρική υπολογισμού περιστροφής 180°	half-turn metric
εξαντλητική δοκιμή και επανάληψη	brute force
τεμάχια	batches
ακατέργαστα πίξελ	raw pixels
αυτοκόλλητα	stickers
αταστάσεις γονέα	parent states
χώρος παρατήρησης	observation space
πόκερ	poker
Διαδικτυακή Πλατφρότμα Ενοικίασης Υπολογιστικής Ισχύος	Google Colaboratory
όρισμα	argument
ανακάτεμα	scramble
κατανομή πολιτικής	distribution policy
λήψη απόφασης	decision making
αξιολόγηση πολιτικής	policy evaluation
μέγεθος βήματος	step-size
κάθε επίσκεψης	every-visit
αραιών ανταμοιβών	sparse rewards
δίκτυο - στόχος	target network
πυκνό	dense
εγγενών κινήτρων	intrinsic motivation