ΕΘΝΙΚΟ ΚΑΙ ΚΑΠΟΔΙΣΤΡΙΑΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ
ΤΜΗΜΑ ΜΑΘΗΜΑΤΙΚΩΝ

ΜΕΤΑΠΤΥΧΙΑΚΟ ΠΡΟΓΡΑΜΜΑ ΣΠΟΥΔΩΝ
ΣΤΑΤΙΣΤΙΚΗΣ ΚΑΙ ΕΠΙΧΕΙΡΗΣΙΑΚΗΣ ΕΡΕΥΝΑΣ

ΕΦΑΡΜΟΓΕΣ ΝΕΥΡΩΝΙΚΩΝ ΔΙΚΤΥΩΝ ΣΕ ΠΡΟΒΛΗΜΑΤΑ ΔΥΝΑΜΙΚΗΣ ΒΕΛΤΙΣΤΟΠΟΙΗΣΗΣ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ του
ΑΝΤΩΝΟΠΟΥΛΟΥ Κ. ΓΕΩΡΓΙΟΥ

ΕΠΙΒΛΕΠΩΝ ΚΑΘΗΓΗΤΗΣ
ΜΠΟΥΡΝΕΤΑΣ ΑΠΟΣΤΟΛΟΣ

ΑΘΗΝΑ, ΣΕΠΤΕΜΒΡΙΟΣ 2014

Αφιερώνεται στον Βασίλειο Ζούκο

Ευχαριστώ πολύ τον επιβλέποντα καθηγητή μου κ. Απ. Μπουρέτα,
όπως και όλους τους καθηγητές του μεταπτυχιακού.

Τριμελής επιτροπή

Απόστολος	Αντώνης	Κωστής
Μπουρνέτας	Οικονόμου	Μηλολιδάκης

ΕΦΑΡΜΟΓΕΣ ΝΕΥΡΩΝΙΚΩΝ ΔΙΚΤΥΩΝ ΣΕ ΠΡΟΒΛΗΜΑΤΑ ΔΥΝΑΜΙΚΗΣ ΒΕΛΤΙΣΤΟΠΟΙΗΣΗΣ

ΠΕΡΙΕΧΟΜΕΝΑ

Α΄ ΜΕΡΟΣ _____ ΝΕΥΡΩΝΙΚΑ ΔΙΚΤΥΑ

ΚΕΦΑΛΑΙΟ 1ο ΠΩΣ ΠΡΟΕΚΥΨΕ ΤΟ ΤΕΧΝΗΤΟ ΝΕΥΡΩΝΙΟ

1.1 *ΕΥΡΥΤΕΡΟ ΠΛΑΙΣΙΟ*

1.2 *ΦΥΣΙΟΛΟΓΙΑ – ΚΕΝΤΡΙΚΟ ΝΕΥΡΙΚΟ ΣΥΣΤΗΜΑ*

1.3 *ΓΕΝΙΚΑ ΠΕΡΙ ΤΕΧΝΗΤΩΝ ΝΕΥΡΩΝΙΚΩΝ ΔΙΚΤΥΩΝ*

1.4 *ΤΕΧΝΗΤΟ ΝΕΥΡΩΝΙΟ (στατικό)*

ΚΕΦΑΛΑΙΟ 2ο ΝΕΥΡΩΝΙΚΟ ΔΙΚΤΥΟ

2.1 *ΣΥΝΔΕΣΜΟΛΟΓΙΑ – ΑΡΧΙΤΕΚΤΟΝΙΚΗ ΝΕΥΡΩΝΙΚΩΝ ΔΙΚΤΥΩΝ*

2.2 *ΜΑΘΗΣΗ – ΕΚΠΑΙΔΕΥΣΗ ΝΕΥΡΩΝΙΚΩΝ ΔΙΚΤΥΩΝ*

2.3 *ΓΕΝΙΚΟΤΕΡΟΙ ΜΕΘΟΔΟΙ ΜΑΘΗΣΗΣ*

2.4 *ΧΡΗΣΗ ΤΕΧΝΗΤΩΝ ΝΕΥΡΩΝΙΩΝ*

Β΄ΜΕΡΟΣ _____ ΕΝΙΣΧΥΤΙΚΗ ΜΑΘΗΣΗ

ΚΕΦΑΛΑΙΟ 3ο ΕΝΙΣΧΥΤΙΚΗ ΜΑΘΗΣΗ

3.1 *ΜΑΘΗΣΗ , ΣΧΕΣΗ ΑΙΤΙΟΥ ΑΠΟΤΕΛΕΣΜΑΤΟΣ*

3.2 ΕΝΙΣΧΥΤΙΚΗ ΜΑΘΗΣΗ: ΤΟ ΠΡΟΒΛΗΜΑ

3.3 ΛΟΓΙΚΗ ΤΗΣ ΕΝΙΣΧΥΤΙΚΗΣ ΜΑΘΗΣΗΣ

3.4 EXPLORATION-EXPLOITATION

3.5 ΔΙΑΔΟΧΙΚΗ ΑΝΑΝΕΩΣΗ ΤΙΜΩΝ

Γ΄ ΜΕΡΟΣ _____ ΤΡΟΠΟΙ ΕΠΙΛΥΣΗΣ ΤΟΥ ΠΡΟΒΛΗΜΑΤΟΣ ΕΝΙΣΧΥΤΙΚΗ ΜΑΘΗΣΗ

Εισαγωγή

ΚΕΦΑΛΑΙΟ 4ο ΔΥΝΑΜΙΚΟΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ

4.1 *Η ΜΕΘΟΔΟΣ ΕΠΑΝΑΛΗΠΤΙΚΗΣ ΑΠΟΤΙΜΗΣΗΣ ΠΟΛΙΤΙΚΗΣ*

4.2 ΒΕΛΤΙΩΣΗ ΠΟΛΙΤΙΚΗΣ

4.3 POLICY ITERATION

4.4 ΥΠΕΡ ΚΑΙ ΚΑΤΑ ΤΟΥ ΔΥΝΑΜΙΚΟΥ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ

ΚΕΦΑΛΑΙΟ 5ο ΜΕΘΟΔΟΣ MONTE CARLO

5.1 ΕΚΤΙΜΗΣΗ MONTE CARLO

5.2 BOOTSTRAPPING

5.3 Η ΣΥΝΑΡΤΗΣΗ ΑΞΙΟΛΟΓΗΣΗΣ ΚΑΤΑΣΤΑΣΕΩΝ-ΑΠΟΦΑΣΕΩΝ $Q^{\pi}(s,a)$

5.4 ΜΕΘΟΔΟΣ ON POLICY– ε-greedy ΠΟΛΙΤΙΚΗ

5.5 ΜΕΘΟΔΟΣ OFF POLICY

5.6 ΠΛΕΟΝΕΚΤΗΜΑΤΑ-ΜΕΙΟΝΕΚΤΗΜΑΤΑ ΤΗΣ MONTE CARLO

5.7 Η ΚΑΤΑΡΑ ΤΗΣ ΔΙΑΣΤΑΣΗΣ

ΚΕΦΑΛΑΙΟ 6ο ΜΑΘΗΣΗ ΧΡΟΝΙΚΩΝ ΔΙΑΦΟΡΩΝ ΕΝΟΣ ΒΗΜΑΤΟΣ , TD(0)-Learning

6.1 *ΑΛΓΟΡΙΘΜΟΣ TD(0)-Learning*

6.2 *ΑΛΓΟΡΙΘΜΟΣ SARSA (TD ενός βήματος / on policy)*

6.3 *ΑΛΓΟΡΙΘΜΟΣ Q (TD ενός βήματος / off policy)*

6.4 *ΑΛΓΟΡΙΘΜΟΣ R (TD, απείρου χρονικού ορίζοντα χωρίς εκπτωτικό παράγοντα)*

ΚΕΦΑΛΑΙΟ 7ο ΜΑΘΗΣΗ ΧΡΟΝΙΚΩΝ ΔΙΑΦΟΡΩΝ, TD(λ)-Learning - ELIGIBILITY TRACES

7.1 *ΧΡΟΝΙΚΕΣ ΔΙΑΦΟΡΕΣ n ΒΗΜΑΤΩΝ, n-step TD*

7.2 *ΑΛΓΟΡΙΘΜΟΣ TD(λ) "Forward View"*

7.3 ΑΛΓΟΡΙΘΜΟΣ TD(λ) "Backward View"

7.4 *SARSA (λ) ΑΛΓΟΡΙΘΜΟΣ*

7.5 *Q(λ) ΑΛΓΟΡΙΘΜΟΣ ή Watkins's Q(λ)*

Δ΄ ΜΕΡΟΣ _____ ΕΦΑΡΜΟΓΕΣ

ΚΕΦΑΛΑΙΟ 8ο ΕΦΑΡΜΟΓΕΣ

8.1 ΈΛΕΓΧΟΣ ΑΠΟΘΕΜΑΤΩΝ (ΈΝΑ ΠΡΟΪΟΝ)

8.2 ΈΛΕΓΧΟΣ ΑΠΟΘΕΜΑΤΩΝ (ΔΥΟ ΠΡΟΪΟΝΤΑ)

ΠΑΡΑΡΤΗΜΑ

ΕΙΣΑΓΩΓΗ

Τα νευρωνικά δίκτυα είναι ένα εργαλείο με πολλές δυνατότητες και εφαρμογές. Στις θετικές επιστήμες έχει βρει πληθώρα εφαρμογών, είτε ως προσέγγιση συναρτήσεων, είτε ως αναγνώριση εικόνας και ήχου ακόμα και εφαρμογή στην τεχνητή νοημοσύνη. Για τα νευρωνικά δίκτυα υπάρχουν διάφοροι τρόποι διάταξης (σύνδεσης των απλών νευρωνίων), όπως και διάφοροι αλγόριθμοι για την υλοποίηση των νευρωνικών δικτύων.

Στην παρούσα εργασία θα κάνουμε μια εισαγωγή-αναφορά στα νευρωνικά δίκτυα όπως και μια αναφορά στους τρόπους διασύνδεσης που υπάρχουν, αλλά και τους τρόπους εκπαίδευσης αυτών. Ακόμα, θα αναλύσουμε κάποιους αλγορίθμους από αυτούς που εφαρμόζονται στην εκπαίδευση των νευρωνικών δικτύων, και θα εφαρμόσουμε αυτούς τους αλγόριθμους σε προβλήματα δυναμικής βελτιστοποίησης.

Η εργασία χωρίζεται σε 4 μέρη. Στο Α΄ μέρος αναφερόμαστε στα νευρωνικά δίκτυα, το Β΄μέρος αφορά την Ενισχυτική Μάθηση, έναν τρόπο εκπαίδευσης των νευρωνικών δικτύων, που είναι και ένα γενικότερο πρόβλημα, ενώ στο Γ΄ μέρος αναφέρονται τρεις διαφορετικοί τρόποι επίλυσης του προβλήματος της Ενισχυτικής Μάθησης, ο Δυναμικός Προγραμματισμός, η *Monte Carlo* Μέθοδος και η μέθοδος Χρονικών Διαφορών. Η μέθοδος Χρονικών Διαφορών χωρίζεται σε αυτήν ενός βήματος και πολλών βημάτων, αλλά και οι δύο είναι συνδυασμός των *Monte Carlo* και Δυναμικού Προγραμματισμού. Επίσης, αναλύεται και η μέθοδος Χρονικών Διαφορών (πολλών βημάτων), που είναι ένας συνδυασμός της *Monte Carlo* και της Μεθόδου χρονικών διαφορών ενός βήματος. Τέλος, στο Δ΄ μέρος γίνονται εφαρμογές των αλγορίθμων σε δύο προβλήματα. Το πρώτο είναι έλεγχος αποθεμάτων για ένα προϊόν, όπου η λύση του είναι γνωστή, και συγκρίνονται τα αποτελέσματα των αλγορίθμων με αυτήν. Ενώ το δεύτερο πρόβλημα είναι έλεγχος αποθεμάτων για δύο προϊόντα, όπου είναι μία νέα εφαρμογή και δίνουμε τα αποτελέσματα των αλγορίθμων.

ΚΕΦΑΛΑΙΟ 1^{ο}
ΠΩΣ ΠΡΟΕΚΥΨΕ ΤΟ ΤΕΧΝΗΤΟ ΝΕΥΡΩΝΙΟ

Στο κεφάλαιο αυτό, αρχικώς, αναφέρουμε πώς ξεκίνησε η χρήση του τεχνητού νευρωνίου και πώς εμπλέκεται στο ευρύτερο πεδίο της υπολογιστικής νοημοσύνης. Επίσης, παρουσιάζουμε την ιδέα λειτουργίας του τεχνητού νευρωνίου, και πως προέκυψε αυτό. Το τεχνητό νευρώνιο είναι η βασική μονάδα σύνθεσης των νευρωνικών δικτύων τα οποία χρησιμοποιούνται στην τεχνητή νοημοσύνη και αναφέρουμε κάποιες από τις εφαρμογές τους, όπως και μερικούς τύπους νευρωνίων (συγκεκριμένα του στατικού νευρώνα).

Η ανάλυση του κεφαλαίου αυτού, όπως και του δευτέρου κεφαλαίου, βασίζεται στο βιβλίο «Υπολογιστική Νοημοσύνη» του κ. Σπ. Τζαφέστα (2002), καθηγητή του Ε.Μ.Π. και σε πηγές από το διαδίκτυο.

1.1 ***ΕΥΡΥΤΕΡΟ ΠΛΑΙΣΙΟ***

Η ανάπτυξη των Τεχνητών Νευρωνικών Δικτύων έγινε σε συνδυασμό με την ανάπτυξη της «Τεχνητής Νοημοσύνης», καθώς έγινε προσπάθεια να προσεγγισθεί η λειτουργία του ανθρωπίνου εγκεφάλου. Γνωρίζοντας την φυσιολογία του Κεντρικού Νευρικού Συστήματος **και την απλότητα αυτού, όσων αφορά στην δομή του,** έγινε προσπάθεια αντιγραφής του απλού νευρώνα του Νευρικού Συστήματος από τους McCulloch & Pitts σε μια εργασία τους το 1943. Η απλότητα του μοντέλου ενός νευρώνα αυτής της εργασίας, που καθιερώθηκε ως «νευρώνιο McCulloch & Pitts» σε συνδυασμό με τις παραλλαγές του και τους ποικίλους τρόπους σύνδεσης των απλών αυτών νευρωνίων, όπως και η δυνατότητα εφαρμογής τους σε διαφόρους επιστημονικούς κλάδους, προκάλεσε την ανάπτυξη της έρευνας στο πεδίο της Τεχνητής Νοημοσύνης.

Ένα τεχνητό νευρώνιο εκ κατασκευής έχει την δυνατότητα καθορισμού κάποιων παραμέτρων, οι οποίες ορίζουν την λειτουργία του (ή την λειτουργία ολόκληρου του Τεχνητού Νευρωνικού Δικτύου). Αυτό μπορεί να γίνει εξαρχής (από τον σχεδιαστή του Τεχνητού Νευρωνίου/Ν.Δικτύου), οπότε και το Τ.Ν.Δ. είναι απλώς ένα είδος προγραμματισμού (με τον προγραμματιστή να γνωρίζει πλήρως τον τρόπο λειτουργίας του Τ.Ν.Δ.) ή μπορεί να καθορισθούν, οι τιμές των παραμέτρων του, εμμέσως (χωρίς να γνωρίζει ο σχεδιαστής-κατασκευαστής τις τιμές αυτές), αρκεί να εκτελεί το Τ.Ν.Δ. μια επιθυμητή προκαθορισμένη διεργασία. Στην δεύτερη περίπτωση έχουμε την εμπλοκή των Νευρωνικών Δικτύων στον τομέα της Υπολογιστικής Νοημοσύνης.

Η Υπολογιστική Νοημοσύνη μπορούμε να πούμε πως βρίσκεται στο σταυροδρόμι της Τεχνητής Νοημοσύνης και της Θεωρίας Συστημάτων και Βελτιστοποίησης. Περιλαμβάνει τα Ασαφή Συστήματα - Ασαφή Λογική, τους Γενετικούς-Εξελικτικούς Αλγορίθμους και τα Νευρωνικά Δίκτυα.

Τα Ασαφή Συστήματα-Ασαφής Λογική (fuzzy sets-fuzzy logic) έχουν την ικανότητα να βγάζουν συμπεράσματα και αποφάσεις (σε επίπεδο γλώσσας) όταν υπάγονται σε μια κατάσταση αβεβαιότητας. Αυτό το πετυχαίνουν μεταφράζοντας μια κατάσταση όχι σε μαθηματικό επίπεδο αλλά σε επίπεδο γλώσσας (π.χ. βαθμολογία 8 ή 12 ενός μαθητή μεταφράζεται ως «αρκετά χαμηλή» ή «μέτρια», ενώ η απόφαση μπορεί να είναι μία εκ των «πολύ διάβασμα & βοήθεια» ή «βοήθεια-επίβλεψη»). Έχουν την λειτουργία-συμπεριφορά παρόμοια με αυτή ενός εμπειρογνώμονα και πηγάζει η λογική τους από την "αρχή αβεβαιότητας του Heisenberg" καθώς ορίζουν πάνω στα δυνατά ενδεχόμενα μια κατανομή πιθανότητας. Κάποιες επιτυχείς εφαρμογές των ασαφών συστημάτων υπάρχουν στην ιατρική (διάγνωση), στην ρομποτική, στην πρόβλεψη σεισμών κ.α. Τα ασαφή συστήματα προσπαθούν να μοντελοποιήσουν την αβεβαιότητα όσον αφορά τα δεδομένα αλλά και τα συμπεράσματα ενός συστήματος αλλά και τον «μηχανισμό» της δυνατότητας να δοθεί συγκεκριμένη απάντηση σε ένα πρόβλημα, έστω και αν αυτή δεν είναι ντετερμενιστική (συμπεριφορά εμπειρογνώμονα).

Οι Γενετικοί-Εξελικτικοί Αλγόριθμοι είναι κατηγορία αλγορίθμων βελτιστοποίησης ή λήψης αποφάσεων που βασίζονται στην ιδέα πως μέσω τυχαίων αλλαγών («μεταλλάξεων»), αναπροσαρμόζονται σε ένα πρόβλημα/κατάσταση στο οποίο έχουν επέλθει κάποιες αλλαγές, και έτσι να μπορεί να επικρατεί η λύση εκείνη, η οποία έχει την καλύτερη συμπεριφορά-επίλυση πάνω στο συγκεκριμένο πρόβλημα. Οι γενετικοί αλγόριθμοι προσπαθούν να αντιγράψουν τον «μηχανισμό» των τυχαίων μεταλλάξεων και της επικράτησης των «καλύτερων» ζωικών και φυτικών οργανισμών στον φυσικό κόσμο.

Τα Νευρωνικά Δίκτυα, είναι κατηγορία αλγορίθμων τεχνητής νοημοσύνης που έχουν την δυνατότητα να εκτελούν μια διεργασία γρήγορα (ακόμα και αν αυτή είναι πολύπλοκη), όπως και να αποκρίνονται σε καταστάσεις/δεδομένα τα οποία αντιμετωπίζουν για πρώτη φορά. Η αρχική προσέγγιση της λειτουργίας του νευρικού συστήματος ενός οργανισμού, ήταν η αντιγραφή της λειτουργίας ενός απλού νευρώνα και έπειτα η πολύπλοκη σύνδεση αυτών των νευρώνων μεταξύ τους. Η ποικιλία στην συνδεσμολογία των νευρώνων δίνει την δυνατότητα χρήσης των Τεχνητών Νευρωνικών Δικτύων σε πολλά διαφορετικά προβλήματα, όπως στατιστική, τεχνητή όραση-ακοή, συσχετιστικές μνήμες, αυτόματο έλεγχο, προσέγγιση συναρτήσεων , βελτιστοποίηση κ.α.

Η Υπολογιστική Νοημοσύνη εφαρμόζεται σε συστήματα και διαδικασίες οι οποίες απαιτούν μία ή περισσότερες από τις παρακάτω λειτουργίες:

- Προσέγγιση συναρτήσεων / απεικονίσεων
- Μοντελοποίηση στατικών / δυναμικών συστημάτων
- Αναγνώριση και ταξινόμηση προτύπων / διαδικασιών (π.χ. εικόνα, ήχος, γραφικός χαρακτήρας)
- Αναγνώριση της δομής και των παραμέτρων φυσικών και τεχνολογικών συστημάτων
- Εκτίμηση και πρόβλεψη χρονοσειρών και ανελίξεων
- Βελτιστοποίηση διεργασιών / συστημάτων
- Επεξεργασία σημάτων και δεδομένων

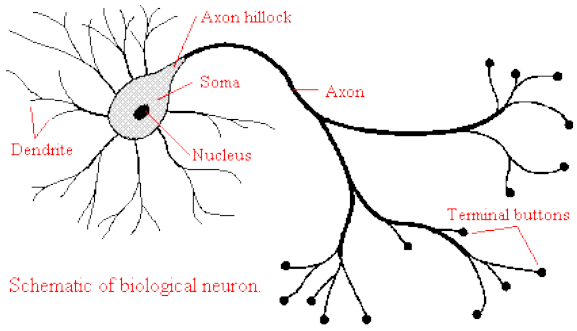
- Εντοπισμός και αναγνώριση βλαβών και ανωμαλιών
- Επιλογή λύσεων και λήψη αποφάσεων (ανανέωση/συντήρηση εξοπλισμού, δίκτυο διανομής)
- Ρύθμιση και έλεγχος βιομηχανικών και μη-βιομηχανικών συστημάτων

Ο συνδυασμός των Τεχνητών Νευρωνικών Δικτύων, των Γενετικών Αλγορίθμων και των Ασαφών Συστημάτων δίνει μεγάλη αυτονομία σε τέτοια συστήματα, με ελάχιστη παρέμβαση του ανθρώπου.

Θα κάνουμε παρακάτω μια αναφορά στα Τεχνητά Νευρωνικά Δίκτυα, ξεκινώντας από το απλό τεχνητό νευρώνιο, όπως και στον τρόπο σύνδεσης των απλών τεχνητών νευρωνίων, ενώ τέλος θα αναφέρουμε και τους τρείς τρόπους μάθησης/εκπαίδευσης των Τεχνητών Νευρωνικών Δικτύων. Από τους τρεις αυτούς τρόπους θα μας απασχολήσει στην συνέχεια της εργασίας η ***ΕΝΙΣΧΥΤΙΚΗ ΜΑΘΗΣΗ***.

1.2 *ΦΥΣΙΟΛΟΓΙΑ – ΚΕΝΤΡΙΚΟ ΝΕΥΡΙΚΟ ΣΥΣΤΗΜΑ*

Παρ'όλη την πολυπλοκότητα του ανθρωπίνου εγκεφάλου και των λειτουργιών που εκτελεί αυτός, η απλότητα των νευρωνικών κυττάρων του εγκεφάλου είναι αξιοσημείωτη. Όλοι οι νευρώνες του εγκεφάλου ανεξαρτήτως μεγέθους, σχήματος και είδους, αποτελούνται από τα ίδια βασικά μέρη: το **κυτταρικό σώμα**, τους **δενδρίτες** και τον **άξονα** ο οποίος καταλήγει στις **συνάψεις**. Οι δενδρίτες μεταφέρουν πληροφορίες από άλλους νευρώνες στο σώμα του νευρώνα, μέσω του άξονα μεταφέρεται ηλεκτροχημικό σήμα, βάσει της καταστάσεως του κυττάρου, και καθώς ο άξονας διακλαδίζεται στις συνάψεις καταλήγει σε συναπτικούς βολβούς κοντά σε δενδρίτες άλλων νευρώνων. Εκεί το μεταφερόμενο σήμα αναπαράγεται, βάσει συνθηκών, από τους επόμενους νευρώνες και με αυτόν τον τρόπο «ταξιδεύουν» πληροφορίες και επεξεργάζονται μέσα στον εγκέφαλο. Καθώς υπάρχουν περίπου $1.5 \cdot 10^{10}$ νευρωνικά κύτταρα διαφόρων ειδών στον ανθρώπινο εγκέφαλο, με κάθε νευρωνικό κύτταρο να διαθέτει 10^4 συνδέσεις με άλλα κύτταρα μέσω των δενδριτών που διαθέτει, αντιλαμβανόμαστε την πολυπλοκότητα του εγκεφάλου αλλά συγχρόνως και την απλότητα της λειτουργίας κάθε νευρωνικού κυττάρου του. Εν ολίγοις, ένας βιολογικός νευρώνας λαμβάνει πληροφορία (δενδρίτες), την επεξεργάζεται (σώμα) και τέλος μεταδίδει κατάλληλη πληροφορία σε επόμενο/ους νευρώνα/ες (συνάψεις). Αυτήν την απλότητα προσπάθησαν να αντιγράψουν με το τεχνητό νευρώνιο.



Schematic of biological neuron.

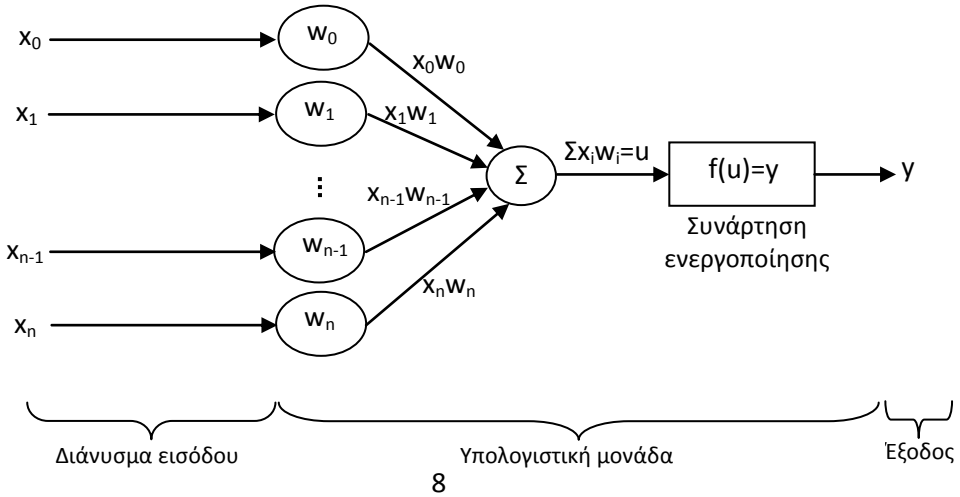
1.3 *ΓΕΝΙΚΑ ΠΕΡΙ ΤΕΧΝΗΤΩΝ ΝΕΥΡΩΝΙΚΩΝ ΔΙΚΤΥΩΝ*

Τα Τεχνητά Νευρωνικά Δίκτυα αποτελούνται από τεχνητά νευρώνια σε κατάλληλη διάταξη-σύνδεση μεταξύ τους. Κάθε νευρώνιο αποτελείται από ένα σύνολο κλάδων διασύνδεσης εισόδου (πλήθους n), και από έναν τουλάχιστον κλάδο διασύνδεσης εξόδου. Οι κλάδοι μπορούν να συνδέονται είτε με άλλα νευρώνια είτε με το περιβάλλον, όπου περιβάλλον θεωρούμε οτιδήποτε εκτός του Νευρωνικού Δικτύου, όπως μια μεταβλητή εισόδου και μια μεταβλητή εξόδου. Ένα νευρώνιο π.χ. μπορεί να δέχεται μέσω των ***n*** διασυνδέσεων εισόδου του ένα διάνυσμα $\bar{x} \in \mathbb{R}^n$, και να δίνει ως έξοδο μία τιμή $y \in \mathbb{R}$. Η έξοδος του νευρωνίου μπορεί να είναι πλήρως ή μερικώς καθορισμένη από τον σχεδιαστή του νευρωνίου (νευρωνικού δικτύου). Πλήρως καθορισμένη είναι όταν προγραμματίζουμε ένα νευρώνιο γνωρίζοντας ακριβώς την εργασία που θέλουμε να εκτελέσει (και τους νόμους που διέπουν την εργασία αυτή) και βέβαια είναι εύκολα πραγματοποιήσιμη μέσω του νευρωνίου, ενώ μερικώς καθορισμένη είναι, είτε όταν δεν γνωρίζουμε επακριβώς τους νόμους που διέπουν την εργασία για την οποία προορίζεται να εξυπηρετήσει το νευρώνιο, είτε όταν δεν είναι εύκολα πραγματοποιήσιμη. Σε αυτήν την περίπτωση μέσω του νευρωνίου επιθυμούμε την κατά προσέγγιση εκτέλεση της εργασίας, χωρίς να μας ενδιαφέρει η εσωτερική λειτουργία του νευρωνίου.

Γενικά ένα Νευρωνικό Δίκτυο μπορεί να είναι είτε αλγόριθμος είτε υλοποιήσιμο σύστημα (hardware). Χρησιμοποιείται σε πάρα πολλές και διαφορετικές περιπτώσεις, με σημαντικότερα πλεονεκτήματα την ταχύτητα απόκρισης και την λειτουργία/απόδοση, ικανοποιητικής προσέγγισης, ακόμα και σε πολύπλοκα μη γραμμικά συστήματα, όπως επίσης και το πλεονέκτημα της ταχύτητας απόκρισης/λειτουργίας ακόμα και σε δεδομένα εισόδου τα οποία δεν έχει ξανασυναντήσει/μάθει. Το τελευταίο είναι και από τα πιο σημαντικά πλεονεκτήματα ενός τεχνητού νευρωνικού δικτύου σε κάποιες εφαρμογές.

1.4 *ΤΕΧΝΗΤΟ ΝΕΥΡΩΝΙΟ (στατικό)*

Ένα Τεχνητό Νευρώνιο, είναι ένα μοντέλο υπολογιστικής μονάδας κατ' αντιστοιχία με τον βιολογικό **νευρώνα,** του οποίου και προσπαθεί να προσεγγίσει την λειτουργία. Έχει ένα πλήθος εισόδων σήματος (στον ρόλο των δενδριτών), μια υπολογιστική μονάδα (στο ρόλο του κυτταρικού σώματος), και μια έξοδο σήματος (στο ρόλο του άξονα/συνάψεων). Το βασικό μοντέλο του Τεχνητού Νευρωνίου σχηματικά είναι:



Εδώ, όπως και στο βιολογικό νευρώνα, το «κύτταρο» δέχεται ερέθισμα/είσοδο από κάθε μία συντεταγμένη του διανύσματος εισόδου σήματος $(x_0, x_1, x_2, ..., x_n)$, και ενισχύει/υποβαθμίζει κάθε επί μέρους είσοδο x_i πολλαπλασιάζοντάς την με αντίστοιχο βάρος w_i. Αθροίζοντας τα σήματα αυτά σε έναν κόμβο άθροισης, προκύπτει το σήμα u, ως πραγματική μεταβλητή για την συνάρτηση ενεργοποίησης (f). Αξίζει να σημειωθεί πως στο διάνυσμα εισόδου η πρώτη συντεταγμένη (x_0), καθώς και το πρώτο βάρος (w_0), χρησιμοποιούνται για τον υπολογισμό του κατωφλίου θ<0 στην περίπτωση όπου η f είναι η συνάρτηση προσήμου, ενώ στις περιπτώσεις όπου δεν θέλουμε να υπάρχει κατώφλι μπορούμε να θέτουμε $w_0 = 0$ και έτσι το διάνυσμα εισόδου να έχει *n* συντεταγμένες.

Συγκεκριμένα, έχουμε την τιμή $u = \sum_{i=0}^{n} w_i x_i$, ενώ η συνάρτηση προσήμου *f* χωρίς κατώφλι είναι $f_0(u) = \begin{cases} -1, & \alpha\nu\ u < 0 \\ 1, & \alpha\nu\ u \geq 0 \end{cases}$ και με κατώφλι είναι $f_\theta(u) = \begin{cases} -1, & \alpha\nu\ u < \theta \\ 1, & \alpha\nu\ u \geq \theta \end{cases}$.

Έτσι από την σχέση $u = \sum_{i=0}^{n} w_i x_i = w_0 x_0 + \sum_{i=1}^{n} w_i x_i$,

θέτοντας $w_0 x_0 = 0, \text{ορίζοντας } w_0 = 0$

ή $w_0 x_0 = \theta, \text{ορίζοντας } w_0 = \theta \text{ και } x_0 = 1$

έχουμε την επιθυμητή απόκριση της συνάρτησης *f* βάσει των *n* συντεταγμένων (1,2,…,n), από τις *n+1* του διανύσματος εισόδου.
Γενικότερα, αν θέλουμε να χρησιμοποιήσουμε για μια συνάρτηση ενεργοποίησης έναν μετασχηματισμό θέσης της μεταβλητής της u, κάνουμε χρήση των x_0 και w_0.

Το νευρώνιο που περιγράφτηκε παραπάνω είναι **στατικό**, καθώς το σήμα μετά την άθροιση περνάει αμέσως στην συνάρτηση ενεργοποίησης. Εάν όμως το σήμα αυτό περνούσε από ένα δυναμικό σύστημα (ανατροφοδότησης) τότε, μέσω της συνάρτησης μεταφοράς αυτού του συστήματος (με την βοήθεια μετασχηματισμού Laplace) το σήμα θα μετατρεπόταν σε u' οπότε και θα είχαμε **δυναμικό** νευρώνιο, με το οποίο δεν θα ασχοληθούμε ιδιαίτερα.

Τέλος, για την συνάρτηση ενεργοποίησης έχουμε διάφορους τύπους, μη-γραμμικές, γραμμικές, παραγωγίσιμες κτλ. Ιδιαίτερο ρόλο παίζουν οι παραγωγίσιμες συναρτήσεις και μάλιστα οι «σιγμοειδείς» (οι οποίες έχουν πάρει το όνομά τους από το σχήμα της γραφικής τους παράστασης).

Παραθέτονται οι σπουδαιότερες :

(a) Συνάρτηση κατωφλίου (ή λογική, ή Boolean, ή προσήμου) (τιμές : {-1, +1})

(b) Συνάρτηση κατωφλίου (ή λογική, ή Boolean) (τιμές : {0,1})

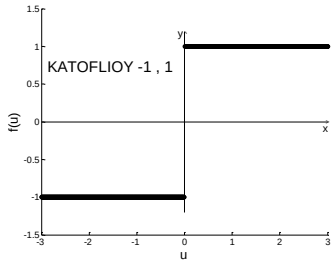
(c) Συνάρτηση κατά τμήματα γραμμική (τιμές : στο διάστημα [-1, 1])

(d) Συνάρτηση κατά τμήματα γραμμική (τιμές : στο διάστημα [0, 1])

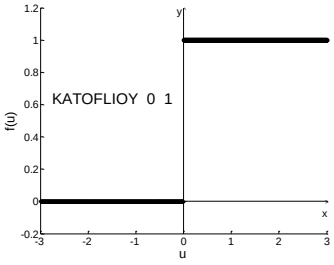
(e) Συνάρτηση σιγμοειδής (τιμές : στο διάστημα (-1,1))

(f) Συνάρτηση σιγμοειδής (τιμές : στο διάστημα (0,1))

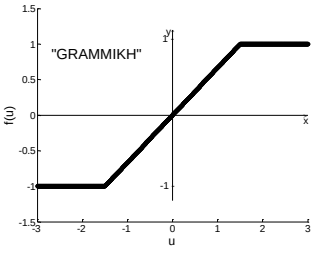
(a)



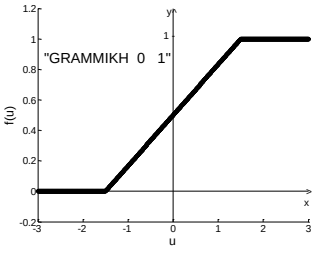
(b)



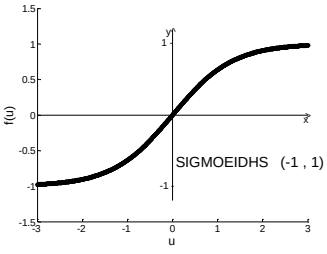
(c)



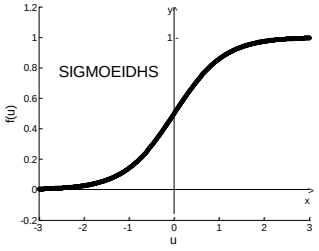
(d)



(e)



(f)



Οι δύο πρώτες έχουν προφανή τύπο, ενώ οι 3η και 4η , κατά τμήματα γραμμικές με $c>0$

έχουν τύπο: $f(u)=\begin{cases} -1 & ,\alpha\nu\ u<-c \\ \dfrac{1}{c}u & ,\alpha\nu\ u\in[-c,c] \\ 1 & ,\ \alpha\nu \quad u<c \end{cases}$ και $f(u)=\begin{cases} 0 & ,\alpha\nu\ u<-c \\ \dfrac{1}{2c}u+\dfrac{1}{2} & ,\alpha\nu\ u\in[-c,c] \\ 1 & ,\ \alpha\nu\ u<c \end{cases}$ αντιστοίχως.

Η 5η και 6η , οι οποίες είναι και παραγωγίσιμες, είναι οι συναρτήσεις:
υπερβολικής εφαπτομένης και ***logistic* (λογιστική συνάρτηση)**, οι οποίες έχουν τύπο

$$f(u)=\frac{1-e^{-cu}}{1+e^{-cu}}$$ και $$f(u)=\frac{1}{1+e^{-cu}}$$ (για $c>0$ και $u\in\mathbb{R}$) αντιστοίχως.

Γενικά οι δύο πρώτες, (a) και (b), χρησιμοποιούνται για να μοντελοποιήσουν την ύπαρξη ή όχι μιας ιδιότητας, με την (b) να δηλώνει με 1 την ύπαρξη και με 0 την μη ύπαρξη της ιδιότητας (ουδετερότητα), ενώ η (a) με 1 την ύπαρξη και με -1 την ύπαρξη της αντίθετης ιδιότητας. Μπορούν να χρησιμοποιηθούν για την ύπαρξη ενός στοιχείου σε ένα σύνολο ή όχι. Οι επόμενες δύο συναρτήσεις (c) και (d), είναι συνεχείς και σε κάθε τους κλάδο είναι γραμμικές, παραγωγίσιμες στο $u\in(-\infty,-c)\cup(-c,c)\cup(c,+\infty)$, αλλά όχι στο $\mathbb{R}$. Τέλος οι σιγμοειδείς συναρτήσεις ενεργοποίησης (f) και (e) είναι παραγωγίσιμες στο $\mathbb{R}$, κάτι που είναι πολύ χρήσιμο στα νευρωνικά δίκτυα. Υπάρχουν και άλλες συναρτήσεις ενεργοποίησης, όπως και άλλες μορφές νευρωνίων, όπου οι κλάδοι εισόδου (x_i) έχουν εκτός από το βάρος w_i και μια συνάρτηση $R_i(x_i)$, από όπου το γινόμενο $w_i\cdot R_i(x_i)$ δίνει μια τοπική προσέγγιση της τελικής συνάρτησης εξόδου. Η $R_i(x_i)$ ονομάζεται ακτινική συνάρτηση βάσεως, και είναι συνήθως μια Γκαουσιανή με δεδομένο κέντρο και διακύμανση. Αυτά τα δίκτυα χρησιμοποιούνται για την προσέγγιση συναρτήσεων μέσω παρεμβολής.

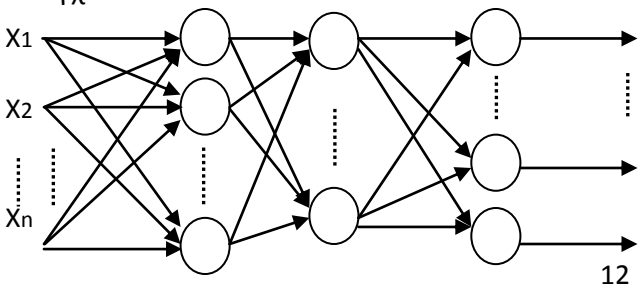
ΚΕΦΑΛΑΙΟ 2^{ο}
ΝΕΥΡΩΝΙΚΟ ΔΙΚΤΥΟ

2.1 *ΣΥΝΔΕΣΜΟΛΟΓΙΑ – ΑΡΧΙΤΕΚΤΟΝΙΚΗ ΝΕΥΡΩΝΙΚΩΝ ΔΙΚΤΥΩΝ*

Όπως και στους βιολογικούς οργανισμούς, όπου ένας νευρώνας δεν έχει και μεγάλες δυνατότητες, αλλά ο συνδυασμός πολλών νευρώνων μπορεί να εκτελέσει εξαιρετικά πολύπλοκες διεργασίες, έτσι και στα τεχνητά νευρωνικά δίκτυα η αποτελεσματικότητά τους οφείλεται στον τρόπο σύνδεσης των νευρωνίων και στο κατάλληλο πλήθος νευρωνίων, πάντα βέβαια σχεδιαζόμενα για το πρόβλημα για το οποίο προορίζονται. Το πλήθος των νευρωνίων, ο τρόπος διασύνδεσής τους και ο τύπος της συνάρτησης ενεργοποίησής τους (συνήθως είναι ίδιος για όλα τα νευρώνια του δικτύου) καλείται αρχιτεκτονική των νευρωνικών δικτύων. Υπάρχουν δύο βασικές κατηγορίες νευρωνικών δικτύων: τα **Νευρωνικά Δίκτυα Προσοτροφοδότησης** και **Νευρωνικά Δίκτυα Ανατροφοδότησης ή (Αναδρομικά/Δυναμικά Νευρωνικά Δίκτυα)**.

Γενικά, πολλά νευρώνια όταν συνδέονται μεταξύ τους μπορούν να είναι σε παράλληλη σύνδεση, οπότε και το σύνολο αυτών των n νευρωνίων ονομάζεται στρώμα, όπου κάθε ένα από τα νευρώνια δέχεται ένα διάνυσμα εισόδου (μπορεί αυτό να είναι κοινό για όλα τα νευρώνια του στρώματος), και δίνει μία έξοδο y_i . Το σύνολο των εξόδων y_i του στρώματος αποτελεί ένα διάνυσμα $y=(y_1,...,y_i,...,y_n)$ το οποίο πλέον μπορεί να είναι το τελικό διάνυσμα εξόδου του νευρωνικού δικτύου, ή με την σειρά του να αποτελεί ένα νέο διάνυσμα εισόδου για άλλα νευρόνια. Μπορούν διάφορα στρώματα νευρωνίων να διαδέχεται το ένα το άλλο, με το διάνυσμα εξόδου του ενός να είναι διάνυσμα εισόδου του επομένου, οπότε και λέμε πως έχουμε ένα **νευρωνικό δίκτυο προσοτροφοδότησης** καθώς το σήμα μεταφέρεται από την αρχική είσοδο προς την τελική έξοδο του νευρωνικού δικτύου, χωρίς να διέρχεται από νευρώνια/στρώμα για δεύτερη φορά. Εδώ έχουμε και το στατικό μοντέλο νευρώνίου που έχουμε προαναφέρει. Τέτοια νευρωνικά δίκτυα διακρίνονται περαιτέρω σε μονοστρωματικά, πολυστρωματικά, πλήρως ή μερικώς συνδεδεμένα (με διάκριση ως προς το αν νευρώνια του ίδιου στρώματος έχουν όλα τις ίδιες εισόδους), με πλάγιες συνδέσεις (σύνδεση μεταξύ νευρωνίων του ίδιου στρώματος). Ακόμα σε ένα πολυστρωματικό νευρωνικό δίκτυο έχουμε την διάκριση του πρώτου στρώματος, το οποίο λαμβάνει την αρχική είσοδο σήματος και του τελικού στρώματος, το οποίο δίνει και την έξοδο του νευρωνικού δικτύου, από το υπόλοιπα στρώματα τα οποία και ονομάζονται **κρυμμένα/κρυφά στρώματα** του νευρωνικού δικτύου.

Αν κάποιο ή κάποια νευρώνια δέχονται ως είσοδο (έστω και μία συνιστώσα του διανύσματος εισόδου τους), σήμα εξόδου του ιδίου ή επομένου στρώματος, τότε το νευρωνικό δίκτυο ονομάζεται αναδρομικό ή **Νευρωνικό Δίκτυο Ανατροφοδότησης** και πλέον έχουμε ένα δυναμικό σύστημα. Με τέτοια νευρωνικά δίκτυα δεν θα ασχοληθούμε στην παρούσα εργασία, απλά αναφέρουμε μερικές εφαρμογές: αναδρομικό δίκτυο Hopfield, μηχανή Boltzmann (προσομοιωμένη ανόπτηση), θεωρία/συστήματα αυτομάτου ελέγχου.



Παράδειγμα νευρωνικού δικτύου προσοτροφοδότησης, με ένα κρυφό στρώμα, με n εισόδους και m εξόδους.

2.2 *ΜΑΘΗΣΗ – ΕΚΠΑΙΔΕΥΣΗ ΝΕΥΡΩΝΙΚΩΝ ΔΙΚΤΥΩΝ*

Κάθε νευρωνικό δίκτυο σχεδιάζεται για να εκτελεί μια εργασία, και η αρχιτεκτονική του είναι σημαντική για την απόδοση του νευρωνικού δικτύου. Εξίσου σημαντικός είναι και ο καθορισμός των συναπτικών βαρών του νευρωνικού δικτύου. Κάθε νευρώνιο διαθέτει συναπτικά βάρη w_i τα οποία πρέπει να εκτιμηθούν/υπολογιστούν και αυτό γίνεται μέσω μιας **επαναληπτικής διαδικασίας** κατά την οποία **ανανεώνονται τα συναπτικά βάρη** έως ότου επιτευχθεί η επιθυμητή συμπεριφορά του νευρωνικού δικτύου. Ο τρόπος (αλγόριθμος) με τον οποίο γίνεται αυτή η ανανέωση, λέγεται μάθηση ή εκπαίδευση των νευρωνικών δικτύων. Κάθε μάθηση έχει πλεονεκτήματα και μειονεκτήματα έναντι άλλης, αλλά κάποιες εξαρτώνται και από την αρχιτεκτονική του δικτύου. Οι τέσσερις βασικοί τρόποι μάθησης, βάσει και της αρχιτεκτονικής των δικτύων είναι: η **Μάθηση Διόρθωσης Σφάλματος,** η **Μάθηση Hebb,** η **Ανταγωνιστική Μάθηση** και η **Μάθηση Boltzmann.**

Για την **Μάθηση Διόρθωσης Σφάλματος**, απαραίτητο είναι να γνωρίζουμε την επιθυμητή έξοδο του δικτύου για κάποιο πλήθος παραδειγμάτων ελέγχου. Για τις εισόδους του γνωστού παραδείγματος, θα λάβουμε την έξοδο του δικτύου και συγκρίνοντάς την με την επιθυμητή (γνωστή) έξοδο, ανανεώνουμε τα συναπτικά βάρη (τα διορθώνουμε), με τέτοιον τρόπο, ώστε να μειωθεί το σφάλμα επιθυμητής-πραγματικής εξόδου του νευρωνικού δικτύου, ή και να επιτευχθεί ακριβώς η επιθυμητή έξοδος, αναλόγως και με την συνάρτηση ενεργοποίησης. Συγκεκριμένα για την Μάθηση Διόρθωσης Σφάλματος, πρέπει να έχουμε για δεδομένα διανύσματα εισόδου, γνωστή την επιθυμητή/προσδοκόμενη έξοδο του νευρωνικού δικτύου, έτσι όταν εφαρμόσουμε την δεδομένη είσοδο στο δίκτυο θα λάβουμε μια έξοδο την οποία μπορούμε να συγκρίνουμε με την επιθυμητή έξοδο.

Ένα παράδειγμα για την Μάθηση Διόρθωσης Σφάλματος δίδεται παρακάτω (§***2.4***).

Η **Μάθηση Hebb**, εφαρμόζεται για να ενισχύει (αυξάνει συγχρόνως) συναπτικά βάρη διαφορετικών νευρωνίων, τα οποία όμως έχουν παρόμοια αντίδραση σε ίδιες εισόδους, και αντιστρόφως, να αυξάνει σε ένα ενώ μειώνει σε δεύτερο τα συναπτικά βάρη, όταν η αντίδρασή τους είναι αντίθετη. Έχει σκοπό να «ταιριάξει» τα νευρώνια που «δουλεύουν» για τα ίδια δεδομένα εισόδου, και να «ξεχωρίσει» τα νευρώνια που δουλεύουν για διαφορετικά δεδομένα.

Η **Ανταγωνιστική Μάθηση**, έχει ως σκοπό την αντιστοίχιση ενός διανύσματος εισόδου σε ένα και μόνο νευρώνιο του στρώματος εξόδου. Βάσει ενός κριτηρίου ομοιότητας (π.χ. ελάχιστη ευκλείδεια απόσταση), «ταιριάζει» το διάνυσμα εισόδου σε ένα μόνο νευρώνιο και ανανεώνει τα συναπτικά βάρη αυτού του νευρωνίου μόνο, με τρόπο ώστε να ταιριάζει ακόμα περισσότερο στην συγκεκριμένη είσοδο (αλγόριθμος των κ-μέσων).

Η **Μάθηση Boltzmann**, είναι μια μορφή στοχαστικής μάθησης η οποία αναπτύχθηκε από τον Boltzmann κάνοντας χρήση μεθόδων θερμοδυναμικής. Χρησιμοποιείται στην μηχανή Boltzmann η οποία είναι ένα αναδρομικό νευρωνικό δίκτυο, με νευρώνια που έχουν δύο καταστάσεις, “ON” και “OFF” και συμμετρικά συναπτικά βάρη. Κάνει χρήση της προσομοιωμένης ανόπτησης (simulation annealing) και δεν έχει ως αναγκαία συνθήκη να είναι η συνάρτηση ενεργοποίησης των νευρωνίων παραγωγίσιμη, αλλά μέσω τυχαίας προσπέλασης νέων συναπτικών βαρών έχει πιθανότητα να «βρει» τέτοια βάρη, ώστε να έχει μικρότερη τιμή για την αντικειμενική συνάρτηση (της οποίας προσπαθεί να βρει το ολικό ελάχιστο).

2.3 *ΓΕΝΙΚΟΤΕΡΟΙ ΜΕΘΟΔΟΙ ΜΑΘΗΣΗΣ*

Επιβλεπόμενη Μάθηση

Η επιβλεπόμενη μάθηση έχει ως χαρακτηριστικό την ύπαρξη ενός εκπαιδευτή/δασκάλου του νευρωνικού δικτύου, ο οποίος δεν είναι στοιχείο του δικτύου αλλά εξωτερικός παράγοντας (μπορεί να είναι ο άνθρωπος που σχεδίασε το νευρωνικό δίκτυο). Ο δάσκαλος βάσει της γνώσης και της εμπειρίας του, θα πρέπει να είναι σε θέση να βρει και να επιλέξει τα κατάλληλα παραδείγματα/ζεύγη εισόδων-εξόδων για το νευρωνικό δίκτυο. Αυτά τα ζεύγη θα πρέπει να επαληθεύονται από το νευρωνικό δίκτυο και μετά το πέρας της μάθησης. Η κατάλληλη επιλογή από τον δάσκαλο αυτών των παραδειγμάτων, έχει σκοπό την επίτευξη της καλύτερης απόκρισης/συμπεριφοράς του νευρωνικού δικτύου μετά την μάθηση, στις πιθανές εισόδους που θα εφαρμοστούν σε αυτό. Δηλαδή, μετά την μάθηση θα πρέπει το νευρωνικό δίκτυο να εκτελεί την διεργασία για την οποία σχεδιάστηκε όσο καλύτερα γίνεται, όπως επίσης να μπορεί να αποκρίνεται και να δίνει αποτέλεσμα για δεδομένα τα οποία δεν έχει διδαχθεί ποτέ. Ένα πλεονέκτημα αυτής της μάθησης είναι ότι καθώς βρίσκονται νέα ζεύγη γνωστών εισόδων-εξόδων (ανεξάρτητα του δικτύου), η μάθηση μπορεί να συνεχιστεί και να ανανεωθούν τα συναπτικά βάρη από τις τελευταίες τιμές που υπήρχαν. Η μάθηση σταματάει όταν το δίκτυο αποκρίνεται «ικανοποιητικά» στις επιθυμητές εξόδους, είτε αυτό σημαίνει να έχει ακριβώς την επιθυμητή έξοδο, είτε να έχει κατά προσέγγιση την επιθυμητή έξοδο (για ένα πλήθος παραδειγμάτων).

Συνδυάζεται συνήθως με την **Μάθηση Διόρθωσης Σφάλματος,** καθώς είναι γνωστές οι επιθυμητές αποκρίσεις του δικτύου και επομένως και το αντίστοιχο σφάλμα.

Ας θεωρήσουμε ένα νευρωνικό δίκτυο με ένα στρώμα και k παράλληλα συνδεδεμένα νευρώνια, οπότε θα έχουμε και k σήματα εξόδου. Έστω, $X=(x_1,x_2,...,x_n)$ το διάνυσμα εισόδου για το νευρωνικό δίκτυο, $Y=(y_1,y_2,...,y_k)$ το διάνυσμα εξόδου, ενώ $D=(d_1,d_2,...,d_k)$ το επιθυμητό διάνυσμα εξόδου για το δεδομένο διάνυσμα εισόδου X. Προφανώς τα ζεύγη $(X,d_i),i=1,...,k$ είναι παράδειγμα στο οποίο πρέπει να ταυτιστεί κάθε νευρώνιο του νευρωνικού δικτύου. Όσο δεν έχει ταυτιστεί, θα υπάρχουν τα αντίστοιχα σφάλματα απόκρισης $e_k=d_k-y_k$. Με βάση αυτά τα σφάλματα ορίζεται το κριτήριο μέσου τετραγωνικού σφάλματος: $J=J(\bar{w})=\frac{1}{2}\sum_{i=1}^{k}e_i^2$, όπου $\bar{w}$ είναι το διάνυσμα που περιέχει τα βάρη του νευρωνικού δικτύου, και ζητούμενο είναι να αλλαχθούν τα βάρη του δικτύου έτσι ώστε να επηρεαστούν οι έξοδοι και να ελαχιστοποιηθεί το μέσο τετραγωνικό σφάλμα.

Για την ελαχιστοποίηση του $J(\bar{w})$ παίρνουμε :

$$\frac{\partial J(\bar{w})}{\partial w_{ij}}=e_i\cdot\frac{\partial e_i}{\partial w_{ij}}=e_i\cdot\frac{\partial\left(d_i-w_i\cdot X\right)}{\partial w_{ij}}=e_i\cdot\left(-x_j\right)=-e_i\cdot x_j$$

και βάσει αυτού έχουμε την ανανέωση των βαρών του δικτύου:

$$w_{ij}^{new}\leftarrow w_{ij}^{old}+\gamma\cdot e_i\cdot x_j$$

Μη επιβλεπόμενη Μάθηση

Σε αυτή την μέθοδο μάθησης δεν χρησιμοποιούνται ούτε εξωτερικός δάσκαλος, ούτε γνωστά επιθυμητά ζεύγη εισόδων-εξόδων. Κάθε νευρωνικό δίκτυο βάσει κάποιων κανόνων/συνάρτησης ανανεώνει τα συναπτικά βάρη του, με τέτοιο τρόπο ώστε να «ταιριάζει» τα διανύσματα εισόδου που λαμβάνει (αυτό γίνεται εκ κατασκευής). Όταν θα έχει εφαρμοστεί το σύνολο των διανυσμάτων εισόδου αρκετές φορές ώστε να μην επέρχεται μεταβολή των συναπτικών βαρών, η μάθηση έχει επιτευχθεί και το νευρωνικό δίκτυο έχει «συντονιστεί» όπως λέγεται στις στατιστικές κατανομές των δεδομένων εισόδου. Χρησιμοποιείται ένα τέτοιο νευρωνικό δίκτυο για αναγνώριση προτύπων (εικόνας/ήχου με εισαχθέντα «θόρυβο», αναγνώρισης κειμένου), «κβαντισμό διανύσματος» δηλαδή εύρεση χαρακτηριστικού αντιπροσώπου για παρόμοια διανύσματα εισόδου, ανάλυσης δεδομένων σε κύριες συνιστώσες, Cluster Analysis, PCA κ.α. Εφαρμόζεται αρκετά σε προβλήματα στατιστικής και δεν θα αναφερθούμε περαιτέρω σε αυτά.

Ενισχυτική Μάθηση

Η ενισχυτική μάθηση μοιάζει περισσότερο με την επιβλεπόμενη μάθηση, αλλά δεν υπάρχουν δεδομένες επιθυμητές έξοδοι για τα διανύσματα εισόδου. Στηρίζεται όχι σε αρχές και κανόνες στατιστικής (μη ενισχυτική μάθηση), αλλά στα αποτελέσματα/εξόδους που προκύπτουν από το ίδιο το νευρωνικό δίκτυο, ώστε να ανανεώσει τα συναπτικά βάρη του. Υπάρχει, μια γενικότερη **βαθμολογία/αποτίμηση** της συμπεριφοράς/απόδοσης του νευρωνικού δικτύου, σχετιζόμενη με την εργασία που πρέπει να φέρει εις πέρας το νευρωνικό δίκτυο. Μέσω αυτής της αποτίμησης γίνεται «επιβράβευση» ή «επίπληξη» της συμπεριφοράς του νευρωνικού δικτύου, οπότε και ανανεώνονται καταλλήλως τα συναπτικά βάρη του. Για την ενισχυτική μάθηση χρειάζεται, λοιπόν, μία **συνάρτηση αξιολόγησης** V, η οποία θα αξιολογεί την συμπεριφορά του νευρωνικού δικτύου (στο σύνολό του και όχι μεμονωμένα για κάποια ζεύγη εισόδων-εξόδων). Με βάση αυτήν την συνάρτηση ενισχύονται τα συναπτικά βάρη του νευρωνικού δικτύου, τα οποία οδήγησαν σε καλύτερη τιμή της συνάρτησης αξιολόγησης (άρα και σε καλύτερη συμπεριφορά του νευρωνικού δικτύου) και «τιμωρούνται» συναπτικά βάρη, τα οποία οδήγησαν σε χειρότερη τιμή της συνάρτησης αξιολόγησης (άρα και σε χειρότερη συμπεριφορά του νευρωνικού δικτύου). Υπάρχει δηλαδή, μια αλληλεπίδραση μεταξύ των αλλαγών των συναπτικών βαρών του νευρωνικού δικτύου, και των μεταβολών της τιμής της συνάρτησης αξιολόγησης.

Ο όρος **ΕΝΙΣΧΥΤΙΚΗ ΜΑΘΗΣΗ** καλύπτει πεδίο ευρύτερο του πεδίου των νευρωνικών δικτύων. Πρώτη φορά αναφέρθηκε από τον Minski το 1961 στο πεδίο της τεχνητής νοημοσύνης, και συγχρόνως το ίδιο έτος από τους Waltz και Fu στο πεδίο της θεωρίας ελέγχου. Έχει ως γενική αρχή την ***αρχή της ενίσχυσης***, η οποία αναφέρεται στην λειτουργία ενός συστήματος και την «τάση» να αυξηθεί η πιθανότητα να λάβουμε την ίδια απόκριση από το σύστημα, υπό παρόμοιες συνθήκες λειτουργίας του συστήματος. Σήμερα η

ενισχυτική μάθηση θεωρείται μια κατηγορία μεθόδων και αλγορίθμων μάθησης που διέπονται από την παραπάνω *αρχή της ενίσχυσης.* Θα συνεχίσουμε την παρούσα εργασία μετά το Α' Μέρος ασχολούμενοι με την ενισχυτική μάθηση την σχέση της με τον Δυναμικό Προγραμματισμό και την Monte Carlo μέθοδο, όπως και με διαφόρους ενδιαφέροντες αλγορίθμους υλοποίησης/επίλυσης αντιστοίχων προβλημάτων.

Μια διαδεδομένη μορφή συνάρτησης αξιολόγησης είναι η εξής:

$$V(w) = E\left[\sum_{k=0}^{\infty} \gamma^k \cdot r(k+1) | w_0 = w\right]$$

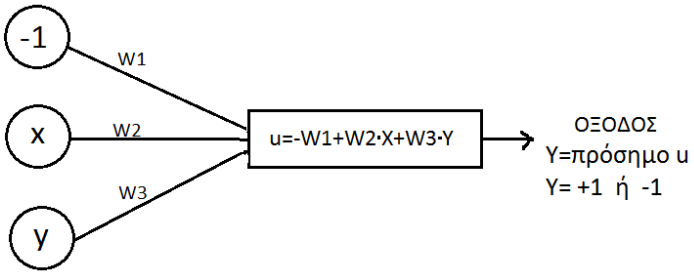
Όπου w το διάνυσμα των βαρών του δικτύου, $r(k+1)$ η «αμοιβή» (θετική ή αρνητική) μετά από την $(k+1)οστή$ ενέργεια (ενέργεια μπορεί να είναι είτε η αλλαγή των συναπτικών βαρών, είτε κάποια απόφαση γενικότερα, η οποία όμως έχει επίπτωση στην τιμή της συνάρτησης αξιολόγησης και επομένως έμμεση επίπτωση και στα συναπτικά βάρη). Ενώ $\gamma \in [0,1]$ είναι μια παράμετρος έκπτωσης ρυθμού και δίνει μικρότερη βαρύτητα στις αμοιβές από ενέργειες που συνέβησαν μακρύτερα στο παρελθόν. Με Ε[...] έχουμε την μέση τιμή του αντίστοιχου αθροίσματος. Η Ενισχυτική μάθηση διατυπώνεται συνήθως και ως πρόβλημα βελτιστοποίησης (βέλτιστη επιλογή των συναπτικών βαρών, ώστε να προκύπτουν καλύτερες τιμές για την συνάρτηση αξιολόγησης).

2.4 *ΧΡΗΣΗ ΤΕΧΝΗΤΩΝ ΝΕΥΡΩΝΙΩΝ*

Θα δώσουμε ένα παράδειγμα χρήσης ενός νευρωνίου για τον διαχωρισμό ενός επιπέδου σε δύο ημιεπίπεδα. Προφανώς ζητούμε να βρεθεί η ευθεία που διαχωρίζει το επίπεδο, και μπορούμε να δώσουμε στο νευρωνικό δίκτυο (εδώ αρκεί ένα νευρώνιο) ένα παράδειγμα ώστε να εκτελέσει αυτό που ζητάμε. Πρώτο ζητούμενο είναι πόσες τιμές εισόδου θα έχει το νευρώνιο και δεύτερον τί θα είναι το παράδειγμα που θα δώσουμε.

Για την οριοθέτηση των δύο ημιεπιπέδων χρησιμοποιούμε μία ευθεία (άγνωστη προς το παρόν). Αυτή θα είναι της μορφής $\mathrm{A} \cdot x + \mathrm{B} \cdot y = \Gamma$ ή $-\Gamma + \mathrm{A} \cdot x + \mathrm{B} \cdot y = 0$, η δεύτερη μορφή μας ενδιαφέρει καθώς μπορούμε να χρησιμοποιήσουμε μία συνάρτηση προσήμου ως συνάρτηση απόκρισης για το νευρώνιο. Από την μορφή της εξίσωσης φαίνεται πως θέλουμε τρεις συντελεστές να υπολογίσουμε, τους Γ,Α και Β, οπότε θέλουμε ένα διάνυσμα εισόδου με 3 συντεταγμένες. Τα αντίστοιχα βάρη θα είναι οι ζητούμενοι αριθμοί Γ,Α και Β. Οπότε, έχουμε $\bar{w} = \left(w_1 = \Gamma, w_2 = \mathrm{A}, w_3 = \mathrm{B}\right)$ τα βάρη του νευρωνίου, ενώ το διάνυσμα εισόδου θα είναι το $X = \left(x_1 = -1, x_2 = x, x_3 = y\right)$ για κάθε σημείο του επιπέδου όπου η $2^{η}$ και $3^{η}$ συντεταγμένη του είναι οι 2 συντεταγμένες του σημείου. Ενώ τελικά η ευθεία θα είναι :

$$y = \left(-\frac{w_2}{w_3}\right) \cdot x + \left(\frac{w_1}{w_3}\right) = a \cdot x + b$$



Για την έξοδο του νευρωνίου έστω πως θα ισχύει +1 για το 1^{ο} ημιεπίπεδο και -1 για το 2^{ο} ημιεπίπεδο. Επίσης, πρέπει να ορίσουμε και κάποιο παράδειγμα για την οριοθέτηση της ζητούμενης ευθείας. Για αυτήν αρκούν 3 σημεία κατάλληλα επιλεγμένα αρκετά "κοντά" στην επιθυμητή ευθεία, όπως επίσης και να ορίσουμε την επιθυμητή έξοδο του νευρωνίου (διάνυσμα *d*).

Για παράδειγμα αν θέλουμε να βρει το νευρώνιο τον διαχωρισμό του επιπέδου από την ευθεία $y=3x$ (***α=3 και β=0***) μπορούμε να επιλέξουμε τα σημεία $\mathrm{K}(-1,-3.2), \Lambda(3,8.5)$ και $\mathrm{M}(1.2,4.5)$ κοντά στην ευθεία αλλά τα δυο πρώτα να ανήκουν στο ίδιο ημιεπίπεδο, οπότε και ορίζουμε $d=(1,1,-1)$ ως επιθυμητές εξόδους. Ζητούμενο είναι να έχει το νευρώνιο τέτοια βάρη ώστε να καταχωρεί και τα τρία σημεία στο σωστό ημιεπίπεδο, π.χ. αν τα βάρη είναι (4 ,2 ,-2) για το σημείο Κ θα είναι $y=sign\left[(-1,-1,-3.2)\begin{pmatrix}4\\2\\-2\end{pmatrix}\right]=sign\left[-4-2+6,4\right]=sign\left[+0,4\right]=+1$ (σωστό) για το σημείο Λ θα είναι $y=sign\left[(-1,3,8.5)\begin{pmatrix}4\\2\\-2\end{pmatrix}\right]=sign\left[-4+6-17\right]=sign\left[-15\right]=-1$ (λάθος) και για το Μ θα είναι $y=sign\left[(-1,1.2,4.5)\begin{pmatrix}4\\2\\-2\end{pmatrix}\right]=sign\left[-4+2.4-9\right]=sign\left[-10.6\right]=-1$ (σωστό).

Για το σημείο Λ (είσοδος $x_2=(-1,3,8.5)$) θα είχαμε την διόρθωση : $\bar{w}=\bar{w}+\gamma\cdot\left[d_2-y_2\right]\cdot x_2$

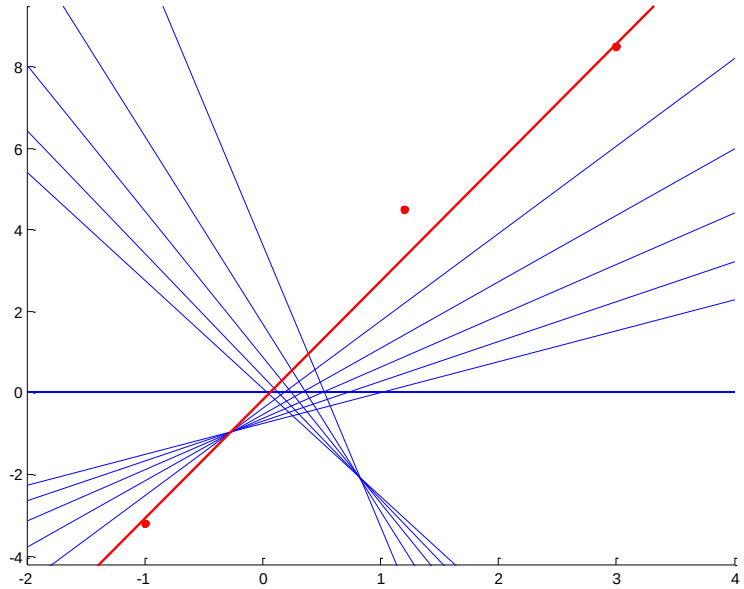
$$w_1=w_1+\gamma\cdot\left[d_2-y_2\right]\cdot x_{2,1}=4+\gamma\cdot\left[1-(-1)\right]\cdot(-1)=4-2\cdot\gamma$$

Ή αναλυτικά : $w_2=w_2+\gamma\cdot\left[d_2-y_2\right]\cdot x_{2,2}=2+\gamma\cdot\left[1-(-1)\right]\cdot(3)=2+6\cdot\gamma$ και αν *γ=0.*1 θα

$$w_3=w_3+\gamma\cdot\left[d_2-y_2\right]\cdot x_{2,3}=-2+\gamma\cdot\left[1-(-1)\right]\cdot(8.5)=-2+17\cdot\gamma$$

είναι τα νέα βάρη $w_1=3.8$, $w_2=2.6$, $w_3=-0.3$ από όπου έχουμε την νέα έξοδο για το Λ $y=sign(1.45)=+1$ (σωστό), για το Κ $y=sign(-5.44)=-1$ (λάθος) και για το Μ $y=sign(-2.03)=-1$ (σωστό). Με τις διαδοχικές διορθώσεις φτιάχνουμε κάποιες εξόδους αλλά ενδεχομένως χαλάμε κάποιες άλλες, αλλά τελικά όλες οι έξοδοι για τα δεδομένα εισόδου ταυτίζονται με τις επιθυμητές εξόδους για αυτά. Για το παράδειγμα που δώσαμε ο αντίστοιχος αλγόριθμος που σχηματίζει και τις ευθείες δίνει το παρακάτω διάγραμμα, όπου με μπλε χρώμα φαίνονται οι αποτυχημένες ευθείες και με κόκκινο η τελική ευθεία που διαχωρίζει σωστά τα 3 σημεία. Πρέπει να σημειωθεί πως μοναδική λύση δεν υπάρχει,

καθώς υπάρχουν άπειρες ευθείες οι οποίες μπορούν να διέρχονται μεταξύ των 3 σημείων, με πολύ μικρές διαφορές βέβαια. Εδώ ο αλγόριθμος έδωσε τελικά α=2.9113 και β=-0,1538, αντί των αρχικών ***α=3 και β=0***. Ο αλγόριθμος υπάρχει στο **παράρτημα 1**.



<u>Νευρωνικό δίκτυο</u>

Παράδειγμα νευρωνικού δικτύου θα μπορούσε εύκολα να χρησιμοποιηθεί με την χρήση του παραπάνω νευρωνίου. Για την καταχώρηση ενός σημείου σε μέρος του επιπέδου, π.χ. σε κυρτό χωρίο ενός επιπέδου, θα μπορούσαμε να χρησιμοποιήσουμε πολλά νευρώνια όπως το παραπάνω, που θα έχουν ως έξοδο την απάντηση «να ανήκει το σημείο σε ένα ημιεπίπεδο», οπότε αν ισχύει για όλα τα νευρώνια καταφατική απάντηση, τότε θα έχουμε απάντηση από το νευρωνικό δίκτυο πως το σημείο ανήκει στο κυρτό χωρίο.

Τα νευρωνικά δίκτυα τέτοιου τύπου μπορούν να έχουν ένα κρυφό στρώμα, το οποίο θα αποτελείται από τα απλά νευρώνια όπως το παράδειγμα που δώσαμε, και ένα νευρώνιο που θα δέχεται ως είσοδο τις εξόδους των νευρωνίων του κρυφού στρώματος και να έχει συνάρτηση κατωφλίου με σταθερά το πλήθος των νευρωνίων του στρώματος.

ΚΕΦΑΛΑΙΟ 3ο
ΕΝΙΣΧΥΤΙΚΗ ΜΑΘΗΣΗ

Στο κεφάλαιο αυτό παρουσιάζεται το πρόβλημα της ενισχυτικής μάθησης και πως αυτό δημιουργήθηκε και απασχόλησε τομείς από διαφορετικές επιστήμες, όπως η ψυχολογία και εν συνεχεία οι θετικές επιστήμες. Όπως προαναφέρθηκε είναι ένας από τους τρόπους εκπαίδευσης των νευρωνικών δικτύων. Παρουσιάζεται, παρ' όλα αυτά, ως αυτόνομο πρόβλημα, όπως παρουσιάζονται και διάφορα προβλήματα που συναντιόνται, όπως το πρόβλημα προσπέλασης όλων των καταστάσεων, αλλά και η τεχνική ανανέωσης τιμών (για την αποφυγή χρήσης μεγάλης μνήμης). Στην συνέχεια της εργασίας παρουσιάζεται η Ενισχυτική Μάθηση ενώ στο Γ' μέρος αυτής παρουσιάζονται οι τρόποι επίλυσής του.

Η ανάλυση βασίστηκε στο βιβλίο των *Richard S. Sutton and Andrew G. Barto: "Reinforcement Learning: An Introduction" (1998)*, ενώ το παράδειγμα του ταξιδιού προς το θησαυρό είναι από του βιβλίο του κ. *Σπ.Τζαφέστα "Υπολογιστική Νοημοσύνη" (2002).*

3.1 ΜΑΘΗΣΗ , ΣΧΕΣΗ ΑΙΤΙΟΥ ΑΠΟΤΕΛΕΣΜΑΤΟΣ

Η ιδέα της ενισχυτικής μάθησης έχει ξεκινήσει από ερευνητές ψυχολόγους, καθώς ερευνούσαν την συμπεριφορά ζώων και ανθρώπων, με βάση σχέσεις αιτίου αποτελέσματος. Το άτομο που τίθεται υπό έρευνα (ο μαθητής), μπορεί να κάνει συσχετίσεις ενός γεγονότος (***αποτέλεσμα*** - όπως μια ενέργεια/αντίδραση, αίσθηση, αμοιβή που εισπράττει), με κάποια ενέργεια/δράση (***αίτιο***), η οποία προηγήθηκε του αποτελέσματος. Συσχετίζοντας λοιπόν, διαδοχικές παρατηρήσεις ή δράσεις και αντιδράσεις, όσο τυχαία και αν είναι μια τέτοια διαδοχική σχέση, ο μαθητής έχει την τάση να συνδέει τα δύο γεγονότα (δράση-αντίδραση), με τρόπο ώστε το πρώτο να είναι το αίτιο και το δεύτερο το αποτέλεσμα. Αυτή η ταύτιση των δύο γεγονότων με τα αίτιο και αποτέλεσμα είναι βασικό στοιχείο της μάθησης και της ψυχολογίας, αρκεί το ζεύγος «αίτιο» - «αποτέλεσμα» να επαναλαμβάνεται πολλές φορές, ώστε το άτομο-μαθητής να συσχετίσει αυτά τα δύο. Βέβαια, ενδέχεται ο μαθητής να ελέγξει αυτήν την σχέση με κάποιο τρόπο ή και όχι. Μπορεί δηλαδή ο μαθητής αφού κάνει την συσχέτιση να αναλύσει τα γεγονότα και να επιβεβαιώσει αν όντως υπάρχει αυτή η σχέση, ή να έχει ως δεδομένη αυτήν την σχέση, απλά και μόνο, λόγω παρατήρησης της συνεχούς διαδοχής των δύο γεγονότων. Έχει, επομένως, ο μαθητής την ικανότητα και την τάση, όταν παρατηρεί κάποιο αποτέλεσμα μετά από μια συγκεκριμένη ενέργεια να συνδέει την ενέργεια αυτή με το αποτέλεσμα και εάν εισπράττει ως «καλό» ή «κακό» το αποτέλεσμα να ωθείται στην αξιολόγηση της ενέργειας ως «καλής» ή «κακής»!

Χαρακτηριστικό παράδειγμα είναι αυτό του "**σκύλου του Pavlov**", όπου η ενέργεια "χτύπημα κουδουνιού", χαρακτηρίστηκε από τον σκύλο ως αίτιο και αξιολογήθηκε ως πολύ καλό, απλά και μόνο γιατί μετά από κάθε κουδούνισμα ακολουθούσε φαγητό για τον σκύλο. Αυτή η επαναλαμβανόμενη διαδικασία (κουδούνισμα-προσφορά φαγητού),

δημιούργησε μια εξαρτημένη σχέση κουδουνιού-φαγητού στον σκύλο, όπου ακόμα και χωρίς να του προσφέρεται τελικώς φαγητό, κάθε κουδούνισμα του προκαλούσε έκκριση σιέλου και τον έκανε να περιμένει φαγητό. Μάθαινε, δηλαδή, να έχει παρόμοιες αντιδράσεις, είτε με το κουδούνισμα, είτε με το αν έβλεπε ή αν μύριζε το φαγητό. Ένα ανακλαστικό το οποίο ο Pavlov το ονόμασε «εξαρτημένο ανακλαστικό». Η ιδέα της μάθησης-εκπαίδευσης, λοιπόν, μέσω αξιολόγησης της σχέσης μιας ενέργειας ή απόφασης (αίτιο) και ενός παρατηρούμενου γεγονότος (αποτελέσματος), το οποίο και ακολουθεί το αίτιο, είναι απλή και συνήθης. Το αν πραγματικά υπάρχει αίτιο και αν είναι σωστή η σχέση αυτή ή ακόμα και αν είναι ηθική ("σκύλος του Pavlov") δεν τίθεται ως ζήτημα στο πλαίσιο που μελετάμε.

Υπάρχουν περιβάλλοντα όπου κάποιος ("μαθητής") πρέπει να λάβει αποφάσεις, να ενεργήσει με έναν συγκεκριμένο τρόπο, όταν βρίσκεται σε μια δεδομένη κατάσταση. Επίσης, δεδομένης της απόφασής του, επιστρέφεται στον μαθητή μια αμοιβή/απολαβή (ή έτσι θέλει να πιστεύει ο "μαθητής"). Θεωρώντας πάντα πως ο μαθητής επιθυμεί την καλύτερη δυνατή απολαβή η οποία θα είναι και συνάρτηση της απόφασης που πρέπει να λάβει, τίθεται το ερώτημα: Ποιά απόφαση πρέπει να λάβει ο μαθητής; Η απολαβή, από την άλλη πλευρά, του μαθητή δεν είναι πάντα μια απλή συνάρτηση της απόφασής του. Μπορεί να εξαρτάται και από άλλες παραμέτρους (γνωστές ή και άγνωστες στον μαθητή), ή ακόμα και να μην είναι ντετερμινιστική, να μην είναι γνωστός ο τρόπος με τον οποίο επιστρέφεται η αμοιβή/απολαβή αυτή, μετά από την απόφαση που λαμβάνει ο μαθητής. Ακόμα πιο σημαντικό και ενδιαφέρον είναι να υπάρχουν διαδοχικές αποφάσεις του μαθητή και απολαβές αυτού, οι οποίες να εναλλάσσονται οι μεν τις δε, και ο μαθητής να έχει ως στόχο όχι μόνο να λάβει την καλύτερη άμεση αμοιβή, αλλά και την καλύτερη αμοιβή αθροιστικά σε βάθος χρόνου. Αρχικά ο μαθητής δεν έχει είτε την εμπειρία είτε τις απαραίτητες γνώσεις για να λάβει τις «σωστές» αποφάσεις, αλλά έχει την δυνατότητα να παρατηρεί τις αμοιβές που διαδέχονται κάθε απόφασή του, να συγκρίνει τις αποφάσεις με τις αμοιβές που ακολουθούν, να αλλάζει αποφάσεις όταν βρίσκεται σε παρόμοιες (ή και ίδιες) καταστάσεις και να επανεκτιμά την συσχέτιση απόφασης-αμοιβής. Μπορεί λοιπόν, να βελτιώνει με αυτήν την διαδικασία τον τρόπο με τον οποίο λαμβάνει τις καλύτερες γι'αυτόν αποφάσεις. Λέμε ότι η διαδικασία αυτή είναι μια **διαδικασία μάθησης**, δηλαδή ο μαθητής μαθαίνει να λαμβάνει τις καλύτερες αποφάσεις για κάθε κατάσταση στην οποία βρίσκεται, βάσει των αμοιβών που λαμβάνει κάθε φορά σε παρόμοιες ή και ίδιες καταστάσεις.

Σημαντικό είναι πως ο μαθητής μπορεί να λαμβάνει αποφάσεις αναλόγως της κατάστασης στην οποία βρίσκεται, και φυσικά αυτή η απόφαση που λαμβάνει είναι δική του επιλογή, ενώ το αποτέλεσμα-γεγονός που ακολουθεί την ενέργειά του δεν καθορίζεται από τον ίδιο, απλά ο μαθητής το παρατηρεί. Το γεγονός που ακολουθεί την απόφαση-ενέργεια του μαθητή, και το οποίο ο ίδιος το θεωρεί άμεση συνέπεια της δικής του απόφασης και ενέργειας, συμβαίνει/πραγματοποιείται από εξωτερικούς παράγοντες όσων αφορά τον μαθητή και οφείλονται στο περιβάλλον του μαθητή, χωρίς ο ίδιος να γνωρίζει επακριβώς τι μπορεί να συμβεί και κατά πόσο είναι βέβαιο αυτό. Μόνο μετά από πολλές

επαναλήψεις και «συμπτώσεις» μπορεί να συνδέσει ο μαθητής το αίτιο με το αποτέλεσμα και να καταγράψει μια ισχυρή ή όχι συσχέτιση μεταξύ τους. Επίσης, ενδέχεται το αποτέλεσμα, όπως το θεωρεί ο μαθητής, να επηρεάζεται περισσότερο από την κατάσταση στην οποία βρισκόταν ο μαθητής και λιγότερο από την απόφαση την οποία πήρε ο μαθητής. Το τελευταίο πάλι είναι κάτι που μπορεί να συνυπολογιστεί από τον μαθητή ή και όχι, καθώς η βασική αρχή του είναι να συνδέσει το αποτέλεσμα (αυτό που παρατηρεί και έπεται) με το αίτιο (αυτό που προηγείται και είναι δικής του επιλογής). Υπάρχει, δηλαδή, μια αλληλεπίδραση μεταξύ του μαθητή και του περιβάλλοντός του και μέσω της αλληλεπίδρασης αυτής, ενισχύεται η «σωστή» απόφαση, δηλαδή όποια απόφαση απέφερε “καλύτερη αμοιβή-αποτέλεσμα”.

Μπορούμε να θεωρήσουμε πως ο μαθητής μπορεί να είναι ένα νευρωνικό δίκτυο (με αποφάσεις να είναι οι τιμές των βαρών του), ένα σύστημα λήψης αποφάσεων, ένα ρομπότ το οποίο μαθαίνει να στέκεται ή να πιάνει ένα αντικείμενο, ένα ζώο που μαθαίνει να κυνηγάει ή ένα παιδί που μαθαίνει να κρύβεται ή να ζωγραφίζει, ένας παίχτης στο σκάκι (διαφέρει από την θεωρία παιγνίων στο ότι ο $2^{ος}$ παίχτης δεν κάνει απαραιτήτως την βέλτιστη κίνηση) κ.α. Σε όλα τα παραπάνω, ο μαθητής έχει να λάβει κάποιες αποφάσεις/ενέργειες όταν βρίσκεται σε μία κατάσταση, και λαμβάνει αντιδράσεις από το περιβάλλον. Οι αντιδράσεις αυτές μπορεί να του δείχνουν κατά πόσο έχει επιτύχει τον στόχο του ή κατά πόσο απομακρύνθηκε ή πλησίασε σε αυτόν, όπως και τι προέκυψε μετά την ενέργειά του, δηλαδή ενδεχομένως μια καλύτερη ή χειρότερη κατάσταση για αυτόν. Στο πρόβλημα της ενισχυτικής μάθησης η αντίδραση από το περιβάλλον, περιλαμβάνει την νέα κατάσταση και μια άμεση αμοιβή. Π.χ. μια κίνηση στο σκάκι επιστρέφει μέσω του περιβάλλοντος το πάρσιμο ενός πιονιού του αντιπάλου (αμοιβή), και τις νέες θέσεις του συνόλου των πιονιών στην σκακιέρα (νέα κατάσταση). Σκοπός του μαθητή είναι όχι η άμεση αμοιβή αλλά η βέλτιστη συσσωρευτική αμοιβή έως το “τέλος”. Συγκεκριμένα δεν ενδιαφέρεται ο παίχτης στο σκάκι μόνο να “πάρει” ένα πιόνι, αλλά να κερδίσει τελικώς το παιχνίδι (αυτό μπορεί να γίνει αν ορίζεται διαφορετική αμοιβή για κάθε πιόνι του αντιπάλου, ή για κατάληψη θέσης στην σκακιέρα, και πολύ μεγαλύτερη αμοιβή για την νίκη, ή εναλλακτικά καμία αμοιβή, εκτός της νίκης η οποία θα έχει κάποια αμοιβή).

Στην παρούσα εργασία θα θεωρήσουμε πως ο μαθητής είναι μια αφηρημένη οντότητα ή ένας αλγόριθμος, που μπορεί να παίρνει αποφάσεις σύμφωνα με μια πολιτική, όπως θα ορίσουμε παρακάτω, έτσι ώστε να φέρνει εις πέρας με τον καλύτερο τρόπο μια διαδικασία. Αυτή η διαδικασία μπορεί να είναι π.χ. ένα πρόβλημα βελτιστοποίησης, μάθηση κίνησης ενός μηχανικού βραχίονα κ.α.

3.2 ΕΝΙΣΧΥΤΙΚΗ ΜΑΘΗΣΗ: ΤΟ ΠΡΟΒΛΗΜΑ

Ενισχυτική μάθηση λοιπόν, είναι η μάθηση/βελτίωση του τρόπου με τον οποίο λαμβάνονται αποφάσεις από κάποιον, όταν μέσω αλληλεπίδρασης λαμβάνει αμοιβές, ενώ ο απώτερος σκοπός είναι η επίτευξη ενός στόχου ή η βελτίωση μιας συσσωρευτικής αμοιβής. Έτσι έχουμε τους παρακάτω **ορισμούς**:

ΣΥΣΤΗΜΑ: *είναι ότι μπορεί να ορισθεί και να συμπεριληφθεί σε ένα πρόβλημα μάθησης μέσω αλληλεπίδρασης, ώστε να επιτευχθεί ένας στόχος*. Περιλαμβάνει το ζεύγος αλληλεπίδρασης **μαθητής-περιβάλλον**.

Το σύστημα παρατηρείται σε διαδοχικές **χρονικές στιγμές - βήματα** (ή στην αρχή διαδοχικών διαστημάτων χρόνου, όπου κανένα χρονικό διάστημα δεν διαρκεί άπειρο χρόνο). Το πλήθος των χρονικών στιγμών μπορεί να είναι πεπερασμένο ή αριθμήσιμο, αναλόγως του προβλήματος. Συνήθως ως αρχική χρονική στιγμή θεωρούμε την $t_1 = 1$, και ακολουθούν $t_2 = 2$, $t_3 = 3$,… χωρίς απαραιτήτως να αντιστοιχούν ίσα χρονικά διαστήματα μεταξύ τους. Υπάρχουν προβλήματα ενισχυτικής μάθησης τα οποία τερματίζονται μετά από συγκεκριμένο πλήθος βημάτων, έστω $N \in \mathbb{N}$, ή τερματίζονται σε άγνωστο μεν αλλά πεπερασμένο πλήθος βημάτων. Το τελευταίο συμβαίνει όταν υπάρχουν κάποιες καταστάσεις στις οποίες άπαξ και βρεθεί ο μαθητής, το πρόβλημα αυτομάτως τερματίζεται. Επομένως, και σε αυτήν την περίπτωση θα υπάρχει κάποια χρονική στιγμή η οποία δηλώνει το τέλος του προβλήματος. Θα δηλώνουμε και για τις δύο περιπτώσεις, στις οποίες έχουμε πεπερασμένο πλήθος βημάτων, την τερματική χρονική στιγμή με $t_{τελικό}$. Τέλος, υπάρχουν προβλήματα ενισχυτικής μάθησης στα οποία οι χρονικές στιγμές είναι άπειρες, δηλαδή δεν έχουμε τερματισμό της διαδικασίας, αλλά μια συνεχή αλληλεπίδραση μαθητή-περιβάλλοντος, με αυτό να σημαίνει πως ο μαθητής συνεχώς ανανεώνει τις εκτιμήσεις του για καλύτερες αποφάσεις. Δηλαδή, υπάρχουν τρεις περιπτώσεις προβλημάτων ενισχυτικής μάθησης, ως προς τον χρόνο, και δύο σύνολα χρονικών στιγμών. Οπότε, έχουμε για πεπερασμένο πλήθος βημάτων $t \in \{1, 2, \ldots, t_{τελικό}\}$ ενώ για αριθμήσιμο $t \in \{1, 2, \ldots\} = \mathbb{N}^*$. Θα συμβολίζουμε στο εξής το σύνολο των χρονικών στιγμών ως T_m.

Υπάρχουν διακριτές **καταστάσεις** στις οποίες μπορεί να βρίσκεται το σύστημα (και ειδικότερα ο μαθητής, ο οποίος και παρατηρεί το σύστημα). Το σύνολο των καταστάσεων συμβολίζεται S, και για το πλήθος των στοιχείων του θα είναι $|S| \in \mathbb{N}$, δηλαδή πεπερασμένο σύνολο καταστάσεων. Για κάθε χρονική στιγμή ορίζεται η αντίστοιχη κατάσταση $s \in S$ ως s_t. Ακόμα, υπάρχει και ένα πεπερασμένο **σύνολο αποφάσεων** το οποίο αφορά τον μαθητή και συμβολίζεται A. Για κάθε χρονική στιγμή s_t, ορίζεται η αντίστοιχη απόφαση ως a_t. Τέλος, υπάρχουν και κάποιες αμοιβές $r_t \in \mathbb{R}$ για κάθε χρονική στιγμή t που αφορούν τον μαθητή, αλλά δεν ελέγχονται από αυτόν.

Ορίζουμε, επίσης, και την **ιστορία του συστήματος** $H_n, n \in T_m$ **έως την χρονική στιγμή** $n, n \in T_m$, να είναι όλη η ακολουθία των καταστάσεων και αποφάσεων από την αρχική στιγμή $t_1 = 1$ έως και την στιγμή $t_n = n$ και πριν ληφθεί η $n-οστή$ απόφαση από τον μαθητή:

$$H_n = (s_1, a_1, s_2, a_2, \ldots, s_{n-1}, a_{n-1}, s_n), n \in T_m$$

ΜΑΘΗΤΗΣ (agent/learner/decision maker): είναι αυτός που λαμβάνει τις αποφάσεις και μαθαίνει μέσω της αλληλεπίδρασης. Κάθε χρονική στιγμή ο μαθητής μπορεί να γνωρίζει μέσω παρατήρησης την κατάσταση $s_n = s, n \in T_m$ στην οποία βρίσκεται, όπως επίσης και την ιστορία του συστήματος H_n, δηλαδή την ακολουθία των προηγουμένων καταστάσεων και αποφάσεων. Έχει την δυνατότητα να παίρνει μια απόφαση a από ένα σύνολο αποφάσεων $A(s_n)$ δεδομένης της κατάστασης s στην οποία βρίσκεται, υποσύνολο βέβαια του A (δηλαδή $A(s_n) \subseteq A$). Στην γενική περίπτωση, η απόφαση του μαθητή ενδεχομένως να εξαρτάται και από την ιστορία του συστήματος, δηλαδή να είναι $a \in A(H_n) \subseteq A$. Εδώ αξίζει να σημειωθεί πως ο τρόπος που λαμβάνει τις αποφάσεις του ο μαθητής ενδέχεται να μην είναι ντετερμινιστικός. Ντετερμινιστική είναι μια απόφαση όταν από το σύνολο των δυνατών επιλογών $A(H_n)$, λαμβάνεται μόνο μία με βεβαιότητα (με πιθανότητα 1). Μη ντετερμινιστική ή στοχαστική απόφαση έχουμε όταν μπορεί ο μαθητής να επιλέξει μεταξύ κάποιων αποφάσεων από το δυνατό σύνολο $A(H_n)$ βάσει κάποιας κατανομής πιθανότητας που έχει ορίσει στο σύνολο αυτό. Μετά από κάθε απόφασή του, ο μαθητής είναι σε θέση να παρατηρήσει την αμοιβή $r_n, n \in T_m$ την οποία δέχθηκε, όπως και την νέα κατάσταση s_{n+1} στην οποία βρίσκεται/μεταπηδά την επόμενη χρονική στιγμή. Για την περίπτωση όπου $n = t_{τελικό}$ δεν έχουμε από τον μαθητή κάποια απόφαση (θα μπορούσαμε να ορίσουμε ίσες πιθανότητες για όλες τις δυνατές αποφάσεις χωρίς βλάβη της γενικότητας), έχουμε όμως μια τερματική αμοιβή $r_{t_{τελικό}}$ και μπορούμε να ορίσουμε την επόμενη κατάσταση πως παραμένει αυτή της τελευταίας κατάστασης $s_{n+1} = s_n$, καθώς το πρόβλημα σταματά.

ΠΟΛΙΤΙΚΗ (π): είναι ο τρόπος με τον οποίο ο μαθητής έχει κανονίσει και παίρνει τις αποφάσεις του, ώστε να επιτύχει τον στόχο του. Σκοπός του μαθητή είναι να παίρνει την βέλτιστη απόφαση. Έτσι, για κάθε κατάσταση $s_n \in S, n \in T_m$ ορίζεται μια απόφαση $a \in \mathrm{A}(H_n)$, ή γενικότερα για κάθε χρονική στιγμή $n \in T_m$ και για κάθε $s_n \in S, n \in T_m$ ορίζεται μια κατανομή πιθανότητας $\pi_n(H_n)$, πάνω στο σύνολο αποφάσεων $\mathrm{A}(H_n)$

$$\text{με } \pi_n(a_n = a \mid H_n) = P(a_n = a \mid H_n),\ \forall\, a \in A \ και \ \forall\, H_n \ και \ \forall\, n \in T_m,$$
$$\text{με } \sum_{a \in A} P(a_n = a \mid H_n) = 1.$$

Το σύνολο των δεσμευμένων κατανομών πάνω στο A, $\pi(. \mid H_n)$ για κάθε $H_n \, και \, n \in T_m$ συμβολίζεται με π και ονομάζεται πολιτική.

ΠΕΡΙΒΑΛΛΟΝ: είναι οτιδήποτε εκτός του μαθητή και εντός του συστήματος. Είναι αυτό με το οποίο ο μαθητής αλληλεπιδρά. Το περιβάλλον μετά από την απόφαση $a_t, t \in T_m$ του μαθητή σε μία χρονική στιγμή t αποδίδει την αμοιβή r_t και συγχρόνως ορίζει την επόμενη κατάσταση $s_{t+1} \in S$ στην οποία μεταβαίνει το σύστημα. Η ***αμοιβή*** (***reward***) r_t αυτή είναι πραγματικός αριθμός και ορίζεται από το περιβάλλον, μέσω μιας συνάρτησης αμοιβής η

οποία αποτιμά την απόφαση του μαθητή. Για την ακρίβεια μετατρέπει την επιτυχία ή αποτυχία του μαθητή ως προς την επίτευξη του στόχου του, ή επιμέρους στόχων του, σε πραγματικούς αριθμούς. Μπορεί η αμοιβή να παίρνει μηδενική ή και αρνητική τιμή σε περιπτώσεις που η απόφαση του μαθητή δεν οδηγεί σε «καλή» κατάσταση ή δεν είναι «καλή» απόφαση. Η συνάρτηση αμοιβής εκ των προτέρων μπορεί να είναι είτε γνωστή είτε άγνωστη στον μαθητή, μπορεί να εξαρτάται από την κατάσταση και την απόφαση, να είναι ή να μην είναι ντετερμινιστική ή γενικότερα να εξαρτάται από κάποιο σύνολο προηγουμένων καταστάσεων και αποφάσεων, οπότε και στην τελευταία περίπτωση έχουμε πρόβλημα καθυστερημένης αμοιβής. Σε κάθε περίπτωση όμως ο μαθητής δεν είναι απαραίτητο να γνωρίζει εκ των προτέρων την αμοιβή αυτή, αλλά τουλάχιστον να την παρατηρεί μέσω της αλληλεπίδρασής του με το περιβάλλον. Επίσης, ο μαθητής μπορεί να παρατηρεί και την επόμενη κατάσταση, χωρίς να είναι απαραίτητο να γνωρίζει τον τρόπο μετάβασης. Ο τρόπος μετάβασης από μία κατάσταση στην επομένη του συστήματος δίδεται από μία κατανομή πιθανοτήτων μετάβασης από την κατάσταση $s_t = s$ προς όλες τις δυνατές καταστάσεις στις οποίες μπορεί να μεταβεί το σύστημα. Οι ***πιθανότητες μετάβασης*** από την κατάσταση s στην κατάσταση $s_{t+1} = s' \in S$ δεδομένης της ιστορίας του συστήματος H_t συμβολίζονται $P^a_{H_t,s'} = P\left(s_{t+1} = s' \mid H_t, a_t = a\right)$. Οι δύο τελευταίες ιδιότητες του περιβάλλοντος, δηλαδή η αμοιβή και ο τρόπος μετάβασης στην επόμενη κατάσταση, χαρακτηρίζεται ως ***ΜΟΝΤΕΛΟ ΤΟΥ ΠΕΡΙΒΑΛΛΟΝΤΟΣ.*** Αξίζει να σημειωθεί εδώ, πως ο μαθητής μέσω της αλληλεπίδρασής του με το περιβάλλον (ακόμα και αγνώστου μοντέλου), μπορεί να κάνει συσχέτιση των αμοιβών και των καταστάσεων (πριν και μετά την αμοιβή), και έτσι να «μαθαίνει/προσδιορίζει» ένα μέρος της λειτουργίας του περιβάλλοντος. Αυτό είναι πολύ ενδιαφέρον, ειδικά σε παιχνίδια όπου ο δεύτερος παίχτης θεωρείται μέρος του περιβάλλοντος, και ο μαθητής ουσιαστικά μαθαίνει τον τρόπο παιχνιδιού του 2^{ου} παίχτη, έχοντας έτσι επί πλέον δεδομένα για να βελτιώσει τις αντιδράσεις-αποφάσεις στο παιχνίδι του. Το τελευταίο δεν θα μας απασχολήσει στην παρούσα εργασία.

ΜΑΡΚΟΒΙΑΝΗ ΙΔΙΟΤΗΤΑ

Μια πολύ σημαντική και χρήσιμη ιδιότητα είναι η Μαρκοβιανή ιδιότητα, η οποία όταν ισχύει απλουστεύει αρκετά τους συμβολισμούς και γενικότερα διευκολύνει την επίλυση του προβλήματος. Έχει παρατηρηθεί, πως σε κάποια προβλήματα ακόμα και να μην ισχύει η Μαρκοβιανή ιδιότητα, μπορεί μέσω της ενισχυτικής μάθησης (θεωρώντας πως ισχύει η Μαρκοβιανή ιδιότητα) να επιλυθεί το πρόβλημα. Ένα σύστημα έχει την Μαρκοβιανή ιδιότητα όταν για την περιγραφή του δεν χρειάζεται η γνώση της ιστορίας του συστήματος, αλλά μόνο η γνώση της τελευταίας κατάστασής του. Δηλαδή, για οτιδήποτε αφορά το σύστημα, η γνώση της τελευταίας κατάστασης είναι αρκετή και δίνει όλη την πληροφορία η οποία είναι απαραίτητη για το σύστημα. Για το πρόβλημα της Ενισχυτικής Μάθησης όταν ισχύει η Μαρκοβιανή ιδιότητα, ισχύει πως το πρόβλημα, και συγκεκριμένα η ακολουθία των καταστάσεων – αποφάσεων – αμοιβών, ορίζει μια ***Μαρκοβιανή Διαδικασία***

Αποφάσεων (***MDP , Markov Decision Process***). Για βασικά μοντέλα και αποτελέσματα παραπέμπουμε *MDP* παραπέμπουμε σε *Ross* (1981), *Peterman* (1994) κτλ.

Επί πλέον, εάν ισχύει πως οι μεταβάσεις σε επόμενες καταστάσεις, οι αποφάσεις και οι αμοιβές δεν εξαρτώνται από την χρονική στιγμή στην οποία βρίσκεται το σύστημα, λέμε πως το πρόβλημα/σύστημα είναι στάσιμο ως προς τον χρόνο. Για παράδειγμα για μια στάσιμη πολιτική θα είναι : $\pi_n(a \mid H_n) = \pi(a \mid H_n)$, όπου δεν εξαρτάται η πολιτική από την χρονική στιγμή n, δηλαδή δεν αλλάζει η πολιτική σε σχέση με τον χρόνο.

Οι ορισμοί που έχουμε δώσει έως τώρα ισχύοντας η Μαρκοβιανή Ιδιότητα και η στασιμότητα γίνονται:

- $H_n = (s_1, a_1, s_2, a_2, \ldots, s_{n-1}, a_{n-1}, s_n = s) \underset{\sigma\tau\alpha\sigma\iota\mu\eta}{\overset{Markov}{=}} s_n = s \quad , n \in T_m$
- $A(H_n) \underset{\sigma\tau\alpha\sigma\iota\mu\eta}{\overset{Markov}{=}} A(s_n) = A(s)$
- Για κάθε $s_n = s, n \in T_m$ ο μαθητής επιλέγει απόφαση: $a_n = a, \ a \in A(s)$
- Για την πολιτική του μαθητή: $\pi_n(a \mid H_n) = \pi(a \mid s) = P(a \mid s), \quad \forall\, a \in A \quad \kappa\alpha\iota \quad \forall\, s \in S$
  με $\sum_{a \in A} P(a \mid s) = 1, \ \forall\, s \in S$.
- Ενώ για τις πιθανότητες μετάβασης $P^a_{H_n, s'} = P(s_{n+1} = s' \mid H_n, a_n = a)$ θα είναι :
  $P(s_{n+1} = s' \mid H_n, a_n = a) = P(s_{n+1} = s' \mid \cancel{H_{n-1}}, s_n = s, a_n = a) = P(s_{n+1} = s' \mid s_n = s, a_n = a)$
  και απλά θα συμβολίζεται $P^a_{s,s'} = P(s_{n+1} = s' \mid s_n = s, a_n = a)$.

Στην συνέχεια της παρούσας εργασίας θα θεωρούμε πως ισχύει η Μαρκοβιανή ιδιότητα και πως το σύστημα είναι ανεξάρτητο του χρόνου όπως δόθηκε παραπάνω.

ΣΥΝΑΡΤΗΣΗ ΑΞΙΟΛΟΓΗΣΗΣ: Ορίζεται σε κάθε περίπτωση μια πραγματική συνάρτηση αξιολόγησης η οποία είναι ουσιώδης στην μάθηση. Έχει σκοπό να αξιολογεί το κατά πόσο είναι καλό ή κακό να βρίσκεται το σύστημα σε μια δεδομένη κατάσταση (και αυτό για όλες τις καταστάσεις) όταν ο μαθητής ακολουθεί μία δεδομένη πολιτική κατά την επίλυση του προβλήματος. Ορίζουμε δύο τέτοιες συναρτήσεις αξιολόγησης. Η πρώτη για την περίπτωση όπου μας ενδιαφέρει άμεσα η αξιολόγηση των καταστάσεων του συστήματος, οπότε για συγκεκριμένη πολιτική π την οποία ακολουθεί ο μαθητής και για κάθε $s \in S$, ορίζεται η ***συνάρτηση αξιολόγησης καταστάσεων*** (***state-value function***) $V^\pi : S \to \mathbb{R}$, ενώ στην περίπτωση όπου μας ενδιαφέρει η απευθείας αξιολόγηση των αποφάσεων σε μια δεδομένη κατάσταση, και για συγκεκριμένη πολιτική π την οποία ακολουθεί ο μαθητής, για κάθε δυνατό ζεύγος κατάστασης-απόφασης $(s, a) \in S \times A$, ορίζεται η αντίστοιχη ***συνάρτηση αξιολόγησης καταστάσεων-αποφάσεων*** (***action-value function***) $Q^\pi : S \times A \to \mathbb{R}$. Οι δύο συναρτήσεις έχουν άμεση σχέση μεταξύ τους όπως θα δείξουμε στα επόμενα κεφάλαια. Ακόμα, ορίζονται σε άμεση σχέση με τις αμοιβές r_t οι οποίες ακολουθούν κάθε

απόφαση. Οι συναρτήσεις αυτές συνδέουν τις καταστάσεις, τις αποφάσεις και τις αμοιβές και είναι ένας τρόπος αξιολόγησης/αποτίμησης των καταστάσεων και αποφάσεων βάσει των αμοιβών που αποφέρουν αυτές.

Ορίζεται η ***αθροιστική*** ή αλλιώς ***συσσωρευτική αμοιβή*** $R_n, n \in T_m$ για κάποια χρονική στιγμή $n \in T_m$ ως το άθροισμα των αμοιβών από την χρονική στιγμή n και έως το τέλος του χρονικού ορίζοντα, ανηγμένο σε αξία την χρονική στιγμή n με την βοήθεια συντελεστή αποπληθωρισμού γ (όπου $\gamma \in (0,1]$ ή $\gamma \in (0,1)$ για πεπερασμένο ή άπειρο ορίζοντα αντιστοίχως).

Έτσι, έχουμε:

για πεπερασμένο ορίζοντα $R_n = r_n + \gamma \cdot r_{n+1} + \gamma^2 \cdot r_{n+2} + \ldots + \gamma^{t_{τελικό}-n} \cdot r_{t_{τελικό}}$

ή για άπειρο ορίζοντα $R_n = r_n + \gamma \cdot r_{n+1} + \gamma^2 \cdot r_{n+2} + \ldots$

ή συνοπτικά $R_n = \sum_{i=0}^{T} \gamma^i \cdot r_{n+i}, \quad όπου\ T = t_{τελικό-n} \quad ή \quad T = \infty$

Συγκεκριμένα, οι συναρτήσεις αξιολόγησης αξιολογούν/εκτιμούν (δίνουν ένα μέτρο) για το πόσο καλό είναι να βρίσκεται ο μαθητής στην κατάσταση s ή πόσο καλό είναι να πάρει την συγκεκριμένη απόφαση σε μία κατάσταση (s,a). Δηλαδή, αξιολογούν την σχέση που φαίνεται να έχουν οι αποφάσεις στις αντίστοιχες καταστάσεις, με την συσσωρευτική/αθροιστική αμοιβή $R_n, n \in T_m$.

Για την συνάρτηση *αξιολόγησης καταστάσεων* (*state-value function)* και δεδομένου πως βρίσκεται ο μαθητής στο βήμα n, στην κατάσταση $s_n = s$ και ακολουθεί δεδομένη πολιτική π ορίζεται:

$$V^{\pi}(s) = E^{\pi}\left[R_n \middle| s_n = s\right] = E^{\pi}\left[\sum_{i=0}^{T} \gamma^i \cdot r_{n+i} \middle| s_n = s\right] \quad , \forall s \in S.$$

Για την συνάρτηση *αξιολόγησης καταστάσεων-αποφάσεων* (*action-value function)* και δεδομένου πως βρίσκεται ο μαθητής στο βήμα n, στην κατάσταση s_n, παίρνει απόφαση $a_n = a$ και από το επόμενο βήμα και έως το τέλος ακολουθεί δεδομένη πολιτική π, ορίζεται:

$$Q^{\pi}(s,a) = E^{\pi}\left[R_n \middle| s_n = s, a_n = a\right] = E^{\pi}\left[\sum_{i=0}^{T} \gamma^i \cdot r_{n+i} \middle| s_n = s, a_n = a\right] \quad , \forall s \in S\ και\ \forall a \in A(s).$$

<u>ΒΕΛΤΙΣΤΗ ΠΟΛΙΤΙΚΗ</u>

Ορίζεται για δύο διαφορετικές πολιτικές π, π' πως η μία είναι καλύτερη από την άλλη αν:

$$\pi \le \pi' \Leftrightarrow V^{\pi}(s) \le V^{\pi'}(s) \quad \forall s \in S$$

Ο μαθητής πρέπει να προσδιορίσει μια πολιτική, μέσω της οποίας βελτιστοποιείται η τιμή της συνάρτησης αξιολόγησης (είτε της *state-value function* είτε της *action-value function*) για κάθε κατάσταση. Η ***βέλτιστη συνάρτηση αξιολόγησης καταστάσεων*** ορίζεται ως:

$$\boxed{V^{*}(s) = \sup_{\pi}\{V^{\pi}(s)\}, \quad \forall s \in S}$$

Επίσης η ***βέλτιστη συνάρτηση αξιολόγησης καταστάσεων-αποφάσεων*** ορίζεται ως:

$$\boxed{Q^{*}(s,a) = \sup_{\pi}\{Q^{\pi}(s,a)\}, \quad \forall s \in S \ \kappa\alpha\iota \ \forall a \in A(s)}$$

Η σχέση που υπάρχει μεταξύ των δύο συναρτήσεων είναι:

$$Q^{*}(s,a) = E\left[r_n + \gamma \cdot V^{*}(s_{n+1}) \middle| s_n = s, a_n = a \right]$$

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

Βέλτιστη πολιτική ορίζεται η πολιτική π^{*} για την οποία έχουμε :

$$V^{\pi^{*}}(s) = V^{*}(s) = \sup_{\pi}\{V^{\pi}(s)\}, \quad \forall s \in S$$

$$Q^{\pi^{*}}(s,a) = Q^{*}(s,a) = \sup_{\pi}\{Q^{\pi}(s,a)\}, \quad \forall s \in S \ \kappa\alpha\iota \ \forall a \in A(s).$$

Συνοπτικά ένα πρόβλημα ενισχυτικής μάθησης με Μαρκοβιανή ιδιότητα αντιστοιχεί σε μια επαναλαμβανόμενη διαδικασία MDP (Μαρκοβιανή Διαδικασία Αποφάσεων) αλληλεπίδρασης με το περιβάλλον, κατά την οποία ο **μαθητής** μαθαίνει/επανεκτιμά τις αποφάσεις που πρέπει να πάρει σε κάθε κατάσταση.

Υπάρχουν διακριτές διαδοχικές χρονικές στιγμές, $t \in T_m$, στις οποίες ο μαθητής μπορεί να παρατηρήσει το σύστημα. Το πλήθος των διαδοχικών χρονικών στιγμών, μπορεί να είναι: ***(α)*** πεπερασμένο, δηλαδή η διαδικασία της αλληλεπίδρασης του μαθητή με το περιβάλλον συντελείται σε πεπερασμένο πλήθος βημάτων, $T_m = \{1, 2, \ldots, t_{\tau\varepsilon\lambda\iota\kappa\acute{o}}\}$, (***<u>πεπερασμένος χρονικός ορίζοντας</u>***), ή ***(β)*** άπειρο, δηλαδή η διαδικασία της αλληλεπίδρασης του μαθητή με το περιβάλλον δεν τελειώνει, με αυτό να σημαίνει πως η μάθηση δεν σταματά, $T_m = \{1, 2, 3, \ldots\}$, (***<u>άπειρος χρονικός ορίζοντας</u>***).

Υπάρχει ένα πεπερασμένο σύνολο καταστάσεων S στο οποίο μπορεί να βρίσκεται ο μαθητή, με $|S| \in \mathbb{N}$. Σε κάθε χρονική στιγμή $t \in T_m$ έχουμε $s_t = s, \quad s \in S$.

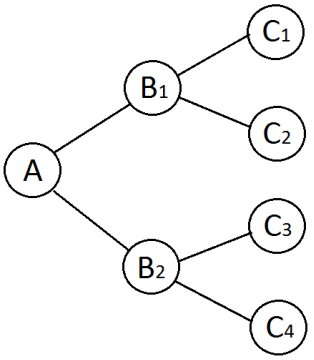
Υπάρχει ένα πεπερασμένο σύνολο ενεργειών/αποφάσεων $\mathrm{A}(s)$, που μπορεί να λάβει ο μαθητής για κάθε κατάσταση $s \in S$ στην οποία βρίσκεται, επομένως και ένα πεπερασμένο σύνολο ενεργειών/αποφάσεων A, για όλες τις δυνατές καταστάσεις με $\mathrm{A}(s) \subseteq A$. Σε κάθε χρονική στιγμή $t \in T_m$ έχουμε $\alpha_t = \alpha, \quad \alpha \in \mathrm{A}(s), \quad \gamma\iota\alpha \ s_t = s$.

Υπάρχουν άμεσες αμοιβές $r \in \mathbb{R}$ που λαμβάνει ο μαθητής από το περιβάλλον και επόμενες καταστάσεις. Μετά από κάθε απόφαση που παίρνει ο μαθητής την χρονική στιγμή $t = n \in T_m$, έχουμε την αμοιβή $r_n = r, r \in \mathbb{R}, \quad \gamma\iota\alpha\ \alpha_n = \alpha, \quad \alpha \in \mathrm{A}(s), \quad \kappa\alpha\iota\ s_n = s$, και την νέα κατάσταση $s_{n+1} \in S$ στην οποία μεταπηδά ο μαθητής και η οποία ορίζεται από το περιβάλλον. Ο τρόπος ορισμού της νέας κατάστασης $s_{n+1} = s' \in S$ από το περιβάλλον μπορεί να είναι γνωστός μέσω αντίστοιχων πιθανοτήτων μετάβασης $P_{ss'}^{a} = P\left(s_{t+1} = s' \mid s_t = s, a_t = a\right)$, ή και άγνωστος στον μαθητή. Στην τελευταία περίπτωση απλά ο μαθητής παρατηρεί την νέα κατάσταση.

Τέλος, υπάρχουν οι συναρτήσεις αξιολόγησης καταστάσεων (V : *state-value function)* και αξιολόγησης καταστάσεων-αποφάσεων *(*Q : *action-value function)*, όπως έχουν ορισθεί και βάσει αυτών γίνεται η εκτίμηση της βέλτιστης απόφασης του μαθητή.

3.3 ΛΟΓΙΚΗ ΤΗΣ ΕΝΙΣΧΥΤΙΚΗΣ ΜΑΘΗΣΗΣ

Η λογική της ενισχυτικής μάθησης είναι ότι βασίζεται στην συσσωρευμένη εμπειρία του μαθητή, δηλαδή στο πως έχει αξιολογήσει ο μαθητής τις καταστάσεις στις οποίες βρίσκεται και τις αποφάσεις τις οποίες παίρνει. Μετά από πολλές επαναλήψεις επίλυσης του προβλήματος ενισχυτικής μάθησης μπορεί ο μαθητής να έχει άποψη για το ποια κατάσταση/απόφαση μπορεί να τον οδηγήσει προς το καλύτερο ή το χειρότερο. Ένα παράδειγμα από το οποίο μπορούμε να δούμε την λογική της ενισχυτικής μάθησης είναι το παρακάτω:



Έστω το διπλανό διάγραμμα με καταστάσεις $S = \{A, B_1, B_2, C_1, C_2, C_3, C_4\}$ στο οποίο ο μαθητής ξεκινώντας από την κατάσταση A και μετακινούμενος πάντα προς τα δεξιά, αποφασίζοντας την κίνηση που θα κάνει (πάνω ή κάτω), έχει την δυνατότητα να μεταβαίνει σε επόμενες καταστάσεις όπως φαίνεται στο διάγραμμα. Σκοπό έχει ο μαθητής να βρει την διαδρομή που τον οδηγεί στον θησαυρό (κατάσταση C_3). Στο πρόβλημα αυτό η απόφαση του μαθητή ορίζει και την πιθανότητα μετάβασης στην επόμενη κατάσταση. Για κάθε μετάβαση η αμοιβή του μαθητή είναι 0 εκτός από την μετάβαση που οδηγεί στην τελική κατάσταση του θησαυρού, όπου η αμοιβή είναι 1 και σημαίνει επιτυχία για τον μαθητή. Ο μαθητής δεν γνωρίζει τίποτα για την επίλυση του προβλήματος αρχικώς, αλλά μπορεί με τυχαίες διαδρομές που θα κάνει και πολλές επαναλήψεις να βρει την σωστή διαδρομή για τον θησαυρό. Έτσι, κάθε φορά που βρίσκεται σε μία κατάσταση και επιλέγει να κινηθεί πάνω ή κάτω μπορεί να καταγράφει πόσες φορές κατέληξε σε επιτυχία, η διαδρομή του, και πόσες σε αποτυχία. Μπορεί με αυτόν τον τρόπο και ακολουθώντας μια πολιτική π, να έχει μια εκτίμηση-αποτίμηση της αξίας μιας κατάστασης $V^{\pi}(s)$ ή της αξίας μιας απόφασης σε δεδομένη κατάσταση $Q^{\pi}(s,a)$. Για παράδειγμα, κάθε φορά που βρίσκεται στην κατάσταση B_2 και επιλέγει να κινηθεί κάτω θα αποτυγχάνει, ενώ αν επιλέγει να κινηθεί

πάνω, θα φτάνει στον θησαυρό-επιτυχία. Έτσι, καταγράφει πως από την κατάσταση B_2 έχει ελπίδες να φτάσει στον θησαυρό και ¨μαθαίνει¨ όταν βρίσκεται στην B_2, να ακολουθεί την κίνηση ¨πάνω¨. Ομοίως, κάθε φορά που θα βρίσκεται στην κατάσταση B_1 από τις δοκιμές που θα κάνει, θα μάθει πως κάθε επιλογή του θα καταλήγει σε αποτυχία. Έτσι, καταγράφει πως η κατάσταση B_1 δεν είναι καθόλου καλή για αυτόν, δηλαδή θα έχει σίγουρα αποτύχει. Τέλος, από την κατάσταση A αν επιλέξει πάνω θα βρεθεί στην ¨κακή¨ κατάσταση B_1, ενώ αν επιλέξει ¨κάτω¨ θα βρεθεί στην κατάσταση B_2 από όπου έχει τον τρόπο να φτάσει στον θησαυρό. Μετά από πολλές δοκιμές και διαδρομές λοιπόν ο μαθητής ¨μαθαίνει¨ την διαδρομή για τον θησαυρό. Έχει δηλαδή την ζητούμενη πολιτική αποφάσεων π για το σύνολο των καταστάσεων του προβλήματος. Εδώ η πολιτική αποφάσεων π είναι μια αντιστοιχία μεταξύ καταστάσεων-αποφάσεων με $\pi = \{(\mathrm{A}, \text{¨}\kappa\acute{\alpha}\tau\omega\text{¨}), (\mathrm{B}_1, \text{¨}\pi\acute{\alpha}\nu\omega\text{¨ }\acute{\eta}\text{ ¨}\kappa\acute{\alpha}\tau\omega\text{¨}), (\mathrm{B}_2, \text{¨}\pi\acute{\alpha}\nu\omega\text{¨})\}$. Σημαντικό είναι πως απόφαση πρέπει να ορισθεί και για την κατάσταση B_1, έστω και αν ο μαθητής δεν βρεθεί ποτέ εκεί, καθώς η διαδρομή του για τον θησαυρό θα είναι $\mathrm{A} \xrightarrow{\kappa\acute{\alpha}\tau\omega} B_2 \xrightarrow{\pi\acute{\alpha}\nu\omega} C_3$.

Η ακριβής επίλυση του προβλήματος και ο ρόλος της συνάρτησης αξιολόγησης, θα παρουσιαστεί στα επόμενα κεφάλαια, με τον ***Δυναμικό Προγραμματισμό (DP)***, με την ***Monte Carlo*** μέθοδο και την ***μέθοδο χρονικών διαφορών (TD μέθοδο)***. Μια αρχική ιδέα για κάθε έναν από τους τρεις αυτούς τρόπους επίλυσης για πρόβλημα πεπερασμένου ορίζοντα, παρουσιάζουμε στα παρακάτω. Στον δυναμικό προγραμματισμό, έχουμε μια επίλυση του προβλήματος από το τέλος προς την αρχή, όπου είναι προφανώς πιο εύκολη η θεώρηση του προβλήματος σε ένα βήμα πριν την τελική κατάσταση. Έτσι, βρίσκεται η καλύτερη πολιτική για ένα βήμα πριν το τέλος και στην συνέχεια διαδοχικά επιλύουμε το πρόβλημα για το δεύτερο βήμα πριν το τέλος, καθώς γνωρίζουμε την καλύτερη απόφαση για το τελευταίο βήμα, και ούτω καθεξής. Για την μέθοδο Monte Carlo παίρνουμε πολλές πραγματοποιήσεις του προβλήματος υπό τυχαία πολιτική, από την αρχή έως το τέλος του χρονικού ορίζοντα και υπολογίζουμε στατιστικά στοιχεία (μέση τιμή) για την συνάρτηση αξιολόγησης κάθε κατάστασης. Έτσι, βγάζουμε συμπεράσματα ως προς το πόσο καλή είναι η παρουσία του μαθητή στην συγκεκριμένη κατάσταση για την επίτευξη του τελικού στόχου. Η βέλτιστη πολιτική καθορίζεται από αυτές τις τιμές. Ενώ η TD μέθοδος λαμβάνει βήμα προς βήμα τις διαφορές προηγουμένων εκτιμήσεων για την αξιολόγηση κάθε κατάστασης, από την παρούσα εμφανιζόμενη αμοιβή σε κάθε κατάσταση. Με τον τρόπο αυτό συνυπολογίζει παλαιότερη γνώση και τωρινή εμπειρία για να επανεκτιμήσει την αξιολόγηση κάθε κατάστασης ή ζεύγους κατάστασης/απόφασης. Κάθε μέθοδος έχει πλεονεκτήματα και μειονεκτήματα, τα οποία θα αναφέρουμε στα επόμενα κεφάλαια όπου θα δούμε αναλυτικότερα τις μεθόδους αυτές.

3.4 EXPLORATION-EXPLOITATION

Στην λογική της μάθησης, από τον μαθητή στο προηγούμενο παράδειγμα, της διαδρομής έως τον θησαυρό έγινε σιωπηρά η χρήση της διαδοχής της αξιοποίησης της υπάρχουσας γνώσης από τον μαθητή και της εξερεύνησης νέων διαδρομών-πολιτικών. Ουσιαστικά ο μαθητής ξεκίνησε χωρίς καμία γνώση για την βέλτιστη διαδρομή, και απλά έκανε μια εξερεύνηση καθώς με τυχαίο τρόπο μετέβαινε από μια κατάσταση σε μια επόμενη. Θα μπορούσε να είχε ξεκινήσει με πολιτική να επιλέγει κίνηση πάντα προς τα πάνω, και αν δεν έβρισκε τον θησαυρό να επέλεγε μια διαφορετική πολιτική. Πρώτα λοιπόν, θα εκτιμούσε τις τιμές της συνάρτησης αξιολόγησης, βάσει της εμπειρίας του, δηλαδή την υπάρχουσα έως τότε γνώση του, κάτι που στην βιβλιογραφία ονομάζεται ***exploitation***. Κατά δεύτερον, θα δοκίμαζε νέες πολιτικές με σκοπό να επιτευχθεί ο στόχος του (ή ακόμα και να βρεθεί καλύτερη λύση), κάτι που στην βιβλιογραφία ονομάζεται ***exploration.*** Είναι αναγκαίο να γίνεται χρήση της *exploitation* καθώς μέσω αυτής εκτιμούνται οι τιμές της συνάρτησης αξιολόγησης για μια πολιτική, αλλά θα πρέπει "πάντα" να διερευνούνται νέες πολιτικές, καθώς "πάντα" κινδυνεύουμε να μην βρούμε την βέλτιστη λύση εάν δεν έχουμε ελέγξει όλες τις δυνατές πολιτικές. Είναι σημαντικότατο λοιπόν, στην επίλυση ενός προβλήματος ενισχυτικής μάθησης να συμπεριλαμβάνονται και οι δύο ιδέες (*exploitation-exploration*), για να βρεθεί η βέλτιστη λύση. Το ζητούμενο είναι: κατά πόσο θα συνεχίζεται η εξερεύνηση νέων πολιτικών καθώς θα προχωράει η μάθηση;

Υπάρχουν διάφορες τεχνικές για το θέμα αυτό! Μία είναι να επιλέγονται πάντα αποφάσεις που οδηγούν σε καταστάσεις με μεγαλύτερη τιμή (*greedy-policy*), αν και αυτό προϋποθέτει αρχικώς να έχουν δοθεί αρκετά μεγάλες τιμές στην συνάρτηση αξιολόγησης, οπότε κατά την διόρθωσή/επανεκτίμησή τους να μειώνονται και έτσι να προκύπτει κάποια άλλη ως βέλτιστη απόφαση. Με αυτόν τον τρόπο, έως ότου να σταθεροποιηθούν οι τιμές της συνάρτησης αξιολόγησης γίνεται συγχρόνως και *exploration*.

Μια δεύτερη τεχνική είναι η *ε-πολιτική* (*ε-greedy*), κατά την οποία κάθε φορά επιλέγεται η βέλτιστη απόφαση με πιθανότητα *(1-ε)* ενώ με πιθανότητα ε/k επιλέγεται κάποια από τις υπόλοιπες k αποφάσεις. Με αυτόν τον τρόπο, πάντα θα γίνεται έλεγχος πάνω σε νέες πολιτικές αν και η τιμή του $\varepsilon \in (0,1)$ παίζει σημαντικό ρόλο, καθώς μπορεί να είναι σταθερή ή ακόμα και να μεταβάλλεται, συγκεκριμένα να μειώνεται.

Μια τρίτη τεχνική είναι να επιλέγεται με τυχαίο τρόπο μια από τις αποφάσεις, όπου όμως όσο προχωράει η μάθηση να τείνει η επιλογή περισσότερο προς την βέλτιστη έως εκείνη την χρονική στιγμή απόφαση (*greedy policy*). Αυτή ονομάζεται *softmax policy* και ορίζεται από μια κατανομή πιθανότητας πάνω στις δυνατές αποφάσεις, η οποία όμως δεν είναι σταθερή ως προς τον χρόνο. Τέτοιες κατανομές είναι οι ***Gibbs*** ή ***Boltzmann***, όπου η πιθανότητα να επιλεγεί η απόφαση α την στιγμή t είναι :

$$\Pr\left(a_t = a \middle| s_t = t\right) = \frac{e^{Q(s,a)/\tau}}{\sum_b e^{Q(s,a)/\tau}}$$

όπου $\tau > 0$ είναι η λεγόμενη θερμοκρασία (παράμετρος) και όσο μεγαλύτερη είναι, τόσο και οι αποφάσεις είναι περισσότερο ισοπίθανες, ενώ όσο πιο κοντά στο μηδέν είναι τόσο η επιλογή της απόφασης είναι πιο κοντά στην *greedy policy*. Η θερμοκρασία θα πρέπει να μειώνεται. Μπορεί να είναι εξαρτώμενη και από την χρονική στιγμή t με σκοπό σταδιακά να μειώνεται, οπότε και η πολιτική αρχικώς να κάνει περισσότερο αναζήτηση σε νέες πολιτικές (*exploration*) και σταδιακά να αξιοποιεί την υπάρχουσα εμπειρία *(exploitation)*.

3.5 ΔΙΑΔΟΧΙΚΗ ΑΝΑΝΕΩΣΗ ΤΙΜΩΝ

Κατά την μάθηση δύο είναι οι σημαντικοί στόχοι, η εκτίμηση των τιμών της συνάρτησης αξιολόγησης για μια πολιτική $V^{\pi}(s)$ $ή$ $Q^{\pi}(s,a)$ και η εύρεση μιας καλύτερης πολιτικής. Η διαδοχή αυτών των δύο στόχων οδηγεί στην εύρεση της βέλτιστης πολιτικής. Για την εκτίμηση των τιμών της συνάρτησης αξιολόγησης χρησιμοποιείται η τεχνική της ***ανανέωσης τιμών*** :

$$NEW = OLD + a \cdot (TARGET - OLD),$$

όπου NEW είναι η νέα "τρέχουσα" τιμή για την ζητούμενη εκτίμηση και OLD είναι η έως τώρα (παλιά) τιμή της ζητούμενης εκτίμησης, ενώ $TARGET$ είναι η νέα επί μέρους γνώση για την ζητούμενη ποσότητα προς εκτίμηση.

Θα δώσουμε ένα παράδειγμα για την κατανόηση του παραπάνω τύπου. Έστω πως διαδοχικά παρατηρούμε τιμές μιας τυχαίας μεταβλητής ($X = x_i$ και εδώ είναι $x_i = TARGET$), και θέλουμε να βρούμε την αναμενόμενη τιμή αυτής. Γενικά θα μπορούσαμε να συλλέξουμε n τιμές της τ.μ. και να πάρουμε την μέση τιμή αυτών $E[X] = \frac{x_1 + x_2 + ... + x_n}{n}$ κάτι που θα χρειαζόταν όλο και μεγαλύτερη μνήμη και υπολογισμούς. Εναλλακτικά, θα μπορούσαμε να βρίσκουμε την μέση τιμή σε κάθε νέα παρατήρηση της τ.μ. και να χρησιμοποιούμε μόνον την τελευταία μέση τιμή και την νέα παρατήρηση για να ανανεώσουμε την τιμή της μέσης τιμής, και βέβαια το πλήθος των παρατηρήσεων *n*. Έτσι:

$$E_{n+1}[X] = \frac{E_n[X] \cdot n + x_{n+1}}{n+1} = \frac{E_n[X] \cdot n + E_n[X] - E_n[X] + x_{n+1}}{n+1} =$$
$$= \frac{E_n[X] \cdot (n+1) + (x_{n+1} - E_n[X])}{n+1} = E_n[X] + \frac{1}{n+1}(x_{n+1} - E_n[X])$$

Θέτοντας $OLD = E_n[X],\ NEW = E_{n+1}[X],\ TARGET = x_{n+1},\ a = \frac{1}{n+1}$ η τελευταία σχέση για την ανανέωση της μέσης τιμής γίνεται:

$$NEW = OLD + \frac{1}{n+1} \cdot (TARGET - OLD)$$

ενώ σε κάθε ανανέωση τιμών χρειαζόμαστε μόνο τρεις θέσεις μνήμης, δύο προσθέσεις έναν πολλαπλασιασμό και μια διαίρεση.

Επίσης, στην περίπτωση που έχουμε ένα σύστημα που μεταβάλλεται (ή την μέση τιμή της παραπάνω τ.μ. να μεταβάλλεται αργά), είναι προτιμότερο να δίνουμε βαρύτητα για την μέση τιμή περισσότερο στις τελευταίες τιμές που έχουμε παρατηρήσει. Με άλλα λόγια, πρέπει να λαμβάνουμε υπόψη μας την τελευταία εμπειρία μας περισσότερο από την παλιά. Αυτό θα μπορούσε να γίνει αν για την ανανέωση δεν είχαμε $a=\frac{1}{n+1}$, αλλά σταθερό $a\in(0,1]$ κάτι που θα έκανε και τους υπολογισμούς ακόμα πιο εύκολους. Ο συντελεστής a ονομάζεται ***βήμα διόρθωσης*** και όσο πιο κοντά είναι στο 1 τόσο περισσότερο συνυπολογίζονται στην εκτίμηση της συνάρτησης αξιολόγησης οι τελευταίες τιμές και λιγότερο όσες είναι σε προηγούμενα βήματα. Ενώ, όσο το βήμα διόρθωσης είναι κοντά στο μηδέν, συνυπολογίζονται οι παλαιότερες τιμές περισσότερο για την εκτίμηση της συνάρτησης αξιολόγησης.

$$NEW = OLD + a\cdot(TARGET - OLD)$$

Ο τρόπος αυτός ανανέωσης λειτουργεί ως ένας σταθμισμένος μέσος. Θα είναι:

$$\begin{aligned}
E_{n+1}[X] &= E_n[X] + a\left(x_{n+1} - E_n[X]\right) = a\cdot x_{n+1} + E_n[X]\cdot(1-a) = \\
&= a\cdot x_{n+1} + \left(a\cdot x_n + E_{n-1}[X]\cdot(1-a)\right)\cdot(1-a) = \\
&= a\cdot x_{n+1} + a(1-a)\cdot x_n + E_{n-1}[X]\cdot(1-a)^2 = \\
&= a\cdot x_{n+1} + a(1-a)\cdot x_n + a(1-a)^2\cdot x_{n-1} + E_{n-2}[X]\cdot(1-a)^3 = \\
= \cdots &= a\cdot x_{n+1} + a(1-a)\cdot x_n + a(1-a)^2\cdot x_{n-1} + \cdots + a(1-a)^{n-1}\cdot x_2 + (1-a)^n\cdot x_1 \ ,
\end{aligned}$$

με τους συντελεστές να αθροίζονται στη μονάδα:

$$\begin{aligned}
&a + a(1-a) + a(1-a)^2 + \cdots + a(1-a)^{n-1} + (1-a)^n = \\
&= a\cdot\sum_{0}^{n-1}(1-a)^i + (1-a)^n = \cancel{a}\cdot\frac{1-(1-a)^n}{\cancel{1-(1-a)}} + (1-a)^n = 1
\end{aligned}$$

Ο κανόνας ανανέωσης $NEW = OLD + a\cdot(TARGET - OLD)$ θα χρησιμοποιηθεί πολύ στα επόμενα κεφάλαια, με σταθερό βήμα ανανέωσης *α,* και για κάθε αλγόριθμο διαφορετικό στόχο $TARGET$.

Γ΄ ΜΕΡΟΣ ΤΡΟΠΟΙ ΕΠΙΛΥΣΗΣ ΤΟΥ ΠΡΟΒΛΗΜΑΤΟΣ ΕΝΙΣΧΥΤΙΚΗΣ ΜΑΘΗΣΗΣ

Οι τρόποι επίλυσης του προβλήματος της ενισχυτικής μάθησης που θα αναπτύξουμε και που ο κάθε ένας έχει πλεονεκτήματα και μειονεκτήματα είναι :

Α΄: Ο Δυναμικός Προγραμματισμός

Β΄: Η Monte Carlo Μέθοδος

Γ΄: Η Μέθοδος Χρονικών Διαφορών, TD μέθοδος (ενός ή περισσοτέρων βημάτων).

Στην TD μέθοδο θα αναπτύξουμε και κάποιους από τους βασικούς αλγορίθμους που χρησιμοποιούν την μέθοδο αυτή, όπως ο αλγόριθμος ***Sarsa*** και ο ***Q*** αλγόριθμος.

Στα επόμενα τρία κεφάλαια θα αναφερθούμε αντιστοίχως στους τρεις τρόπους/μεθόδους επίλυσης του προβλήματος Ενισχυτικής Μάθησης, με το $6^{ο}$ κεφάλαιο να αναφέρεται σε μεθόδους Χρονικών Διαφορών ενός βήματος, ενώ το $7^{ο}$ κεφάλαιο σε μεθόδους Χρονικών Διαφορών περισσοτέρων βημάτων που συνδυάζουν τις μεθόδους Monte Carlo και Χρονικών Διαφορών ενός βήματος.

ΚΕΦΑΛΑΙΟ 4ο
ΔΥΝΑΜΙΚΟΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ

Το πρόβλημα της ενισχυτικής μάθησης για να επιλυθεί αλγοριθμικά ακριβώς μέσω Δυναμικού Προγραμματισμού θα πρέπει να έχει:

- Καταστάσεις πεπερασμένου πλήθους, δηλαδή $s \in S, \mu\varepsilon \left|S\right| \in \mathbb{N}$
- Αποφάσεις πεπερασμένου πλήθους, δηλαδή $a \in A, \mu\varepsilon \left|A\right| \in \mathbb{N}$
- Άμεσες αμοιβές γνωστές, δηλαδή $r = r(s,a) \in \mathbb{R}, \mu\varepsilon \max\{r\} \le M \in \mathbb{R}$
- Πιθανότητες μετάβασης στην επόμενη κατάσταση επίσης γνωστές $P_{ss'}^{a}$

4.1 *Η ΜΕΘΟΔΟΣ ΕΠΑΝΑΛΗΠΤΙΚΗΣ ΑΠΟΤΙΜΗΣΗΣ ΠΟΛΙΤΙΚΗΣ*

Αν υπάρχει βέλτιστη πολιτική π^*, τότε αυτή θα ικανοποιεί τις εξισώσεις βελτιστοποίησης για την συνάρτηση αξιολόγησης $V^*(s)$ (και τις $Q^*(s,a)$), ή αλλιώς τις γνωστές *εξισώσεις Bellman*:

$$V^*(s) = \max_a \left\{ r(s,a) + \gamma \sum_{s'} P_{ss'}^{a} \cdot V^*(s') \right\} \quad , \forall s \in S$$

$$Q^*(s,a) = r(s,a) + \gamma \cdot \max_{a'} \left\{ \sum_{s'} P_{ss'}^{a} \cdot Q^*(s',a') \right\} \quad , \forall s \in S$$

Εδώ πρέπει να σημειωθεί πως για την περίπτωση που έχουμε πρόβλημα πεπερασμένου χρονικού ορίζοντα θα πρέπει να έχουμε και αντίστοιχες τιμές $V^*(s)$ και $Q^*(s,a)$ για κάθε χρονική περίοδο, αλλά αυτό θα μπορούσε να γίνει με αναπροσαρμογή των καταστάσεων έτσι ώστε να εμπεριέχεται η χρονική στιγμή στην κατάσταση. Δηλαδή θα μπορούσαμε μία κατάσταση s αναλόγως της χρονική στιγμή t στην οποία συναντάται, να την διαχωρίζουμε από την ίδια κατάσταση s σε μία άλλη χρονική στιγμή t' (με κατάλληλη αναπροσαρμογή βέβαια των πιθανοτήτων μετάβασης).

Το σκεπτικό γενικότερα, για την επίλυση είναι να υπολογίζονται οι τιμές της συνάρτησης αξιολόγησης για μια πολιτική π, η οποία και ακολουθείται, και να βελτιώνεται η πολιτική βάσει των τιμών αυτών. Θα διευκρινιστεί η διαδικασία αυτή στα παρακάτω.

Έστω μια πολιτική π που θέλουμε να ακολουθήσουμε. Στην πολιτική αυτή θα αντιστοιχούν για κάθε κατάσταση οι τιμές της συνάρτησης αξιολόγησης $V^{\pi}(s), s \in S$. Δηλαδή, έχουμε ένα σύστημα $\left|S\right|$ το πλήθος αγνώστων (τις τιμές $V^{\pi}(s)$), με $\left|S\right|$ το πλήθος γραμμικές εξισώσεις (τις εξισώσεις *Bellman*), κάτι που σημαίνει πως υπάρχει λύση στο σύστημα αυτό. Βέβαια, η λύση μπορεί να είναι επίπονη ή και δύσκολο να βρεθεί, από υπολογιστικής άποψης, αλλά σε αυτό θα αναφερθούμε παρακάτω.

Για τον υπολογισμό των τιμών $V^{\pi}(s), s \in S$ χρησιμοποιούμε τον αλγόριθμο προσέγγισης ***Iterative Policy Evaluation*** (*επαναληπτική αποτίμηση πολιτικής*). Σύμφωνα με αυτόν, ξεκινώντας με τυχαίες τιμές για τις $|S|$ το πλήθος άγνωστες τιμές $V^{\pi}(s)$ του συστήματος των εξισώσεων βελτιστοποίησης, ανανεώνουμε τις τιμές αυτές σύμφωνα με την παρακάτω σχέση, και για κάθε κατάσταση

$$V^{\pi}(s) = \mathrm{E}^{\pi}\left[r_t + \gamma \cdot V^{\pi}(s_{t+1}) \,|\, s_t = s \right] = \sum_a \pi(s,a) \sum_{s'} P^a_{ss'} \left[r(s,a) + \gamma \cdot V^{\pi}(s') \right] \quad , \forall s \in S \quad (4.1)$$

Έτσι, για κάθε κατάσταση $s \in S$ και από τις συνεχείς ανανεώσεις τιμών, έχουμε τις ακολουθίες $V_0^{\pi}(s), V_1^{\pi}(s), V_2^{\pi}(s), \ldots, V_{\kappa}^{\pi}(s), V_{\kappa+1}^{\pi}(s), \ldots$ για κάθε $s \in S$, οι οποίες αποδεικνύεται ότι συγκλίνουν στις ζητούμενες τιμές $V^{\pi}(s), s \in S$ για $\gamma < 1$. Οπότε, βάσει τις (4.1), η ανανέωση γίνεται :

$$V_{\kappa+1}^{\pi}(s) = \sum_a \pi(s,a) \sum_{s'} P^a_{ss'} \left[r(s,a) + \gamma \cdot V_{\kappa}^{\pi}(s') \right] \quad , \forall s \in S \qquad (4.2)$$

Η ανανέωση των τιμών στο αριστερό μέλος των εξισώσεων (4.2), μπορεί να γίνεται χρησιμοποιώντας στο δεξί μέλος, τις παλιές τιμές $V_{\kappa}^{\pi}(s')$ ή ακόμα και τις πιο πρόσφατα ενημερωμένες τιμές $V_{\kappa+1}^{\pi}(s')$ της αντίστοιχης ακολουθίας, όταν αυτές είναι διαθέσιμες.

Για κάθε κατάσταση η ακολουθία $V_n^{\pi}(s)$ μπορεί να σταθεροποιείται σε μια τιμή ή και να συγκλίνει σε αυτήν χωρίς να "πιάνει" την τιμή αυτή. Μπορούμε και στις δύο περιπτώσεις να έχουμε μία συνθήκη τερματισμού των ανανεώσεων με τον αντίστοιχο ψευδοκώδικα:

$\max_{s \in S} \left| V_{\kappa+1}^{\pi}(s) - V_{\kappa}^{\pi}(s) \right| \le \varepsilon$ όπου $\varepsilon > 0$ μικρός πραγματικός αριθμός.

Iterative Policy Evaluation

$\pi \leftarrow \eta\ \pi o \lambda \iota \tau \iota \kappa \acute{\eta}$

$\varepsilon \leftarrow 0,01\ (\acute{\eta}\ o \pi o \iota o \sigma \delta \acute{\eta} \pi o \tau \varepsilon\ \mu \iota \kappa \rho \acute{o} \varsigma \ldots)$

$\Gamma \iota \alpha\ \kappa \acute{\alpha} \theta \varepsilon\ s \in S, \quad V^{\pi}(s) = 0\ \ (\alpha \upsilon \theta \alpha \acute{\iota}\ \rho \varepsilon \tau \alpha)$

repeat

  $\delta \leftarrow 0$

  $\Gamma \iota \alpha\ \kappa \acute{\alpha} \theta \varepsilon\ s \in S,$

    $palia \leftarrow V^{\pi}(s)$

    $V^{\pi}(s) \leftarrow \sum_a \pi(s,a) \sum_{s'} P^a_{ss'} \left[r(s,a) + \gamma \cdot V^{\pi}(s') \right]$

    $\delta \leftarrow \max\left(\delta, | palia - V^{\pi}(s) | \right)$

until $\delta < \varepsilon$

output $V^{\pi}(s)\ \gamma \iota \alpha\ \kappa \acute{\alpha} \theta \varepsilon\ s \in S.$

4.2 ΒΕΛΤΙΩΣΗ ΠΟΛΙΤΙΚΗΣ

Αφού έχει υπολογιστεί η συνάρτηση αξιολόγησης για μια πολιτική ζητούμενο είναι πλέον η βελτίωση της πολιτικής αυτής. Η βελτίωση γίνεται με την βοήθεια της συνάρτησης αξιολόγησης καθώς αυτή δείχνει κατά πόσο είναι "καλή" η πολιτική αυτή, και αυτό γιατί μέσω της πολιτικής καθορίστηκαν οι τιμές της συνάρτησης αξιολόγησης. Για την βελτίωση μιας πολιτικής χρησιμοποιείται ο λεγόμενος αλγόριθμος ***βελτίωσης πολιτικής*** (***policy improvement***).

Ο αλγόριθμος αυτός προσπαθεί για μία δεδομένη πολιτική της οποίας έχουν υπολογιστεί οι τιμής της συνάρτησης αξιολόγησης, να βελτιώσει σε ένα και μόνο βήμα, όταν βρίσκεται σε κάποια κατάσταση $s \in S$ την πολιτική και να λάβει μια καλύτερη απόφαση για αυτήν την κατάσταση. Συγκεκριμένα, για μια ντετερμενιστική πολιτική π αν υπάρχει κατάσταση s και απόφαση α, τέτοια ώστε για το σύνολο των καταστάσεων να είναι:

$$r(s,a) + \gamma \sum_{s'} P_{ss'}^{a} \cdot V^{\pi}(s') \geq r(s,\pi(s)) + \gamma \sum_{s'} P_{ss'}^{\pi(s)} \cdot V^{\pi}(s')$$

$$\Leftrightarrow \qquad Q^{\pi}(s,a) \geq V^{\pi}(s)$$

δηλαδή αν υπάρχει απόφαση *α* για την κατάσταση s, ώστε ακολουθώντας την *α* όταν βρισκόμαστε στην s και την πολιτική *π* για κάθε άλλη κατάσταση, να έχουμε καλύτερη συνάρτηση αξιολόγησης για το σύνολο των καταστάσεων, τότε μπορούμε να πάρουμε ως νέα πολιτική την π' με

$$\pi'(state) = \begin{cases} \pi(s), \, για \, state \neq s \\ a, \quad για \, state = s \end{cases}$$

και θα είναι $\pi' \geq \pi$ δηλαδή θα έχουμε μια καλύτερη ή ίση πολιτική σε σύγκριση με αυτήν που ξεκινήσαμε. Αν επί πλέον, έχουμε $V^{\pi'}(s) = V^{\pi}(s) \quad , \forall s \in S$ τότε και οι δύο πολιτικές είναι βέλτιστες πολιτικές.

Η σύγκλιση εγγυάται μέσω του POLICY IMPROVEMENT THEOREM, από την εργασία των Jaakkola, Singh και Jordan (1995).

4.3 POLICY ITERATION

Συνοψίζοντας, για να βρεθεί η βέλτιστη πολιτική π^* σε πεπερασμένου χρόνου προβλήματα Δυναμικού Προγραμματισμού, ξεκινώντας από μία τυχαία πολιτική π_0, διαδοχικά υπολογίζουμε την συνάρτηση αξιολόγησης που αντιστοιχεί σε αυτήν την πολιτική (*policy evaluation*), βελτιώνουμε την πολιτική αυτή (*policy improvement*) βρίσκοντας την επόμενη πολιτική π_1, υπολογίζουμε εκ νέου την συνάρτηση αξιολόγησης για την νέα πολιτική και ούτω καθεξής. Τελικώς, θα βρούμε την βέλτιστη πολιτική π^*. Η διαδικασία αυτή ορίζει τον παρακάτω αλγόριθμο «*Policy Iteration»:*

Policy Iteration

1. $\Gamma\iota\alpha\ \kappa\acute{\alpha}\theta\varepsilon\ s \in S,\quad o\rho\acute{\iota}\zeta\varepsilon\tau\alpha\iota\quad \pi \in A(s)\ \ (\alpha\upsilon\theta\alpha\acute{\iota}\rho\varepsilon\tau\alpha)$

   $\Gamma\iota\alpha\ \kappa\acute{\alpha}\theta\varepsilon\ s \in S,\quad o\rho\acute{\iota}\zeta\varepsilon\tau\alpha\iota\quad V^{\pi}(s) = 0\ \ (\alpha\upsilon\theta\alpha\acute{\iota}\rho\varepsilon\tau\alpha)$

   $\varepsilon \leftarrow 0{,}01\,(\acute{\eta}\ o\pi o\iota o\sigma\delta\acute{\eta}\pi o\tau\varepsilon\ \mu\iota\kappa\rho\acute{o}\varsigma\ldots)$

2. (*policy evaluation*)

   repeat

   $\delta \leftarrow 0$

   $\Gamma\iota\alpha\ \kappa\acute{\alpha}\theta\varepsilon\ s \in S,$

   $palia \leftarrow V(s)$

   $V(s) \leftarrow r(s,\pi(s)) + \gamma \cdot \sum_{s'} P_{ss'}^{\pi(s)} \cdot V(s')$

   $\delta \leftarrow \max\left(\delta, |\ palia - V(s)\ |\right)$

   until $\quad \delta < \varepsilon$

3. (*policy improvement*)

   $policy\_state \leftarrow True$

   $\Gamma\iota\alpha\ \kappa\acute{\alpha}\theta\varepsilon\ s \in S,$

   $\beta \leftarrow \pi(s)$

   $\pi(s) \leftarrow \arg\max_{a}\left\{ r(s,a) + \gamma \cdot \sum_{s'} P_{ss'}^{a} \cdot V(s') \right\}$

   If $\quad \beta \neq \pi(s) \quad$ *then* $\quad policy\_state \leftarrow False$

   If $\quad policy\_state \quad$ *then* $\quad STOP, \quad$ *else* $\quad GO\ TO\ 2.$

output $\quad \pi(s)\ \gamma\iota\alpha\ \kappa\acute{\alpha}\theta\varepsilon\ s \in S.$

Η παράμετρος ε στον αλγόριθμο παίζει σημαντικό ρόλο στην ταχύτητα σύγκλισης. Αν είναι αρκετά μικρό ($\varepsilon \simeq 0$), η ακολουθία των $V_n^{\pi_i}(s)$ θα είναι ε κοντά στην πραγματική τιμή των $V^{\pi_i}(s)$ για κάθε s, αλλά αυτό μπορεί να απαιτεί πολλά βήματα/επαναλήψεις του αλγορίθμου, και μάλιστα μόνο στο κομμάτι της προσέγγισης των τιμών $V^{\pi_i}(s)$, δηλαδή για κάθε μία από τις πολιτικές π_i. Αυτό είναι σαφές πως μπορεί να καθυστερεί πολύ τον αλγόριθμο για την τελική εύρεση της βέλτιστης πολιτικής. Αν το ε είναι "μεγάλο" μπορεί ενδεχομένως να μην βελτιώνει σε κάποιο *loop* τις τιμές $V_n^{\pi_i}(s), s \in S$. Υπάρχει όμως και αντίστοιχος αλγόριθμος, ο οποίος πραγματοποιεί λίγα βήματα ανανέωσης στο κομμάτι *policy evaluation*, χωρίς να περιμένει να επιτευχθεί ε-σύγκλιση. Ακόμα και για μία επανάληψη/ανανέωση των τιμών $V_\kappa^{\pi_i}(s)$ για μια πολιτική π_i το αποτέλεσμα είναι ικανοποιητικό. Ο αλγόριθμος αυτός ονομάζεται ***value iteration***. Διαφέρει από τον αλγόριθμο *policy iteration* στο τμήμα του αλγορίθμου που κάνει *policy evaluation* για κάποια πολιτική, καθώς γίνεται μόνο ένα βήμα ανανέωσης τιμών βάσει της σχέσης:

$$V_{\kappa+1}^{\pi_i}(s) = \max_{a}\left\{ r(s,a) + \gamma \cdot \sum_{s'} P_{ss'}^{a} \cdot V_\kappa^{\pi_i}(s') \right\}, \forall s \in S$$

και όταν έχει επέλθει η σύγκλιση (κατά ε *κοντά*) θα έχει βρεθεί η V^* (και η πολιτική π^*).

Value Iteration

1. $\Gamma\iota\alpha\ \kappa\acute{\alpha}\theta\varepsilon\ s \in S,\quad o\rho\acute{\iota}\zeta\varepsilon\tau\alpha\iota\quad V^{\pi}(s) = 0\ \ (\alpha\upsilon\theta\alpha\acute{\iota}\rho\varepsilon\tau\alpha)$

   $\varepsilon \leftarrow 0{,}01\,(\acute{\eta}\ o\pi o\iota o\sigma\delta\acute{\eta}\pi o\tau\varepsilon\ \mu\iota\kappa\rho\acute{o}\varsigma\ldots)$

2. *repeat*

   $\delta \leftarrow 0.\quad \Gamma\iota\alpha\ \kappa\acute{\alpha}\theta\varepsilon\ s \in S,$

   $palia \leftarrow V(s)$

   $$V(s) \leftarrow \max_a \left\{ r(s,a) + \gamma \cdot \sum_{s'} P^a_{ss'} \cdot V(s') \right\}$$

   $$\delta \leftarrow \max\left(\delta, | palia - V(s) |\right)$$

   until $\quad \delta < \varepsilon$

   output $\quad \pi$

$$\tau\acute{\varepsilon}\tau o\iota o\quad \acute{\omega}\sigma\tau\varepsilon : \pi(s) = arg\max_a \left\{ r(s,a) + \gamma \cdot \sum_{s'} P^a_{ss'} \cdot V(s') \right\} \gamma\iota\alpha\ \kappa\acute{\alpha}\theta\varepsilon\ s \in S.$$

4.4 ΥΠΕΡ ΚΑΙ ΚΑΤΑ ΤΟΥ ΔΥΝΑΜΙΚΟΥ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ

Ο Δυναμικός Προγραμματισμός είναι σίγουρα ένας αποδεδειγμένα σωστός τρόπος για την επίλυση τέτοιων προβλημάτων όπως η ενισχυτική μάθηση, αλλά απαιτεί πλήρη γνώση του περιβάλλοντος. Απαιτεί δηλαδή την γνώση της συνάρτησης που δίνει τις άμεσες αμοιβές όπως και τις πιθανότητες μεταπήδησης στην επόμενη κατάσταση. Αυτή η γνώση του περιβάλλοντος δεν είναι πάντα δυνατή και αυτό είναι στα μειονεκτήματα του Δ.Π. Επίσης, πρέπει να ισχύει και η ιδιότητα Markov, η οποία δεν ισχύει σε πολλά προβλήματα. Παρ' όλα αυτά ο Δ.Π. είναι ένα πολύ ισχυρό εργαλείο για θεωρητική προσέγγιση τέτοιων προβλημάτων.

Σε προβλήματα όπου υπάρχει μεγάλο πλήθος καταστάσεων ή και αποφάσεων, η επίλυση μέσω του Δ.Π. γίνεται πολύ δύσκολη έως και αδύνατη. Ακόμα και για προβλήματα με πλήθος καταστάσεων της τάξης εκατοντάδων καταστάσεων, οι υπολογιστές αδυνατούν υπολογιστικά (από άποψη απαίτησης μνήμης) να ολοκληρώσουν τους παραπάνω αλγορίθμους. Είναι το πρόβλημα που στην βιβλιογραφία ονομάζεται «Πρόβλημα/Κατάρα της Διάστασης» (***"Curse of Dimensionality"*** *Bellman 1957*).Το πλήθος των καταστάσεων, καθώς αυξάνεται, αυξάνει εκθετικά το πλήθος των αγνώστων σε τέτοια προβλήματα, και οι απαιτήσεις σε υπολογιστική ισχύ και μνήμη εκτινάσσονται, με αποτέλεσμα και ισχυροί ακόμα ηλεκτρονικοί υπολογιστές, πολύ γρήγορα σε σχέση με την αύξηση του αριθμού των καταστάσεων να αδυνατούν να ολοκληρώσουν τους αλγορίθμους. Θα δώσουμε στο επόμενο κεφάλαιο ένα παράδειγμα όπου φαίνεται το πρόβλημα της κατάρας της διάστασης. Παρ' όλα αυτά ο Δυναμικός Προγραμματισμός παραμένει ένας τρόπος επίλυσης αρκετά συντομότερος από επίλυση με γραμμικό προγραμματισμό ή και πλήρη διερεύνηση σε όλο τον χώρο των καταστάσεων/αποφάσεων για την εύρεση της βέλτιστης πολιτικής.

Τον Δυναμικό Προγραμματισμό έχουν συνδέσει με την Ενισχυτική Μάθηση οι Minsky (1961), Andreae (1969b), Werbos (1977), Watkins (1989).

ΚΕΦΑΛΑΙΟ 5ο
ΜΕΘΟΔΟΣ MONTE CARLO

Εισαγωγή

Οι Monte Carlo μέθοδοι γενικά χρησιμοποιούν την μέση τιμή από πλήθος δεδομένων για να κάνουν εκτίμηση ενός χαρακτηριστικού. Στο πρόβλημα της ενισχυτικής μάθησης, για να μπορεί να επιλυθεί με Monte Carlo, θα πρέπει να έχουμε:

- Καταστάσεις πεπερασμένου πλήθους, δηλαδή $s \in S, \mu\varepsilon \left|S\right| \in \mathbb{N}$
- Αποφάσεις πεπερασμένου πλήθους, δηλαδή $a \in A, \mu\varepsilon \left|A\right| \in \mathbb{N}$
- Άμεσες αμοιβές, άγνωστες εκ των προτέρων στον μαθητή, αλλά παρατηρούμενες από αυτόν, όπως και παρατηρούμενες επόμενες καταστάσεις μετά από κάθε βήμα/απόφαση. Το μοντέλο του περιβάλλοντος είναι άγνωστο στον μαθητή.
- Πεπερασμένο χρόνο ολοκλήρωσης της διαδοχής καταστάσεων και αποφάσεων, δηλαδή πεπερασμένο χρονικό ορίζοντα.

Τις άμεσες αμοιβές, όπως και τις μεταβάσεις σε επόμενη κατάσταση ο μαθητής απλά τις παρατηρεί και δεν γνωρίζει κατά κανόνα τον μηχανισμό παραγωγής των στοιχείων αυτών. Η παρατήρηση πραγματοποιείται διαδοχικά μετά από κάθε απόφαση το μαθητή. Αυτό μπορεί να συμβαίνει με δύο τρόπους: (α) με πραγματική αλληλεπίδραση με το περιβάλλον, οπότε, όταν πραγματοποιηθεί η αμοιβή, αλλά και η μετάβαση στην επόμενη κατάσταση, τότε και μόνο μπορεί αυτή να παρατηρηθεί, και (β) με προσομοίωση, οπότε παράγονται από κάποιον μηχανισμό (προσομοιώνονται), και οι αμοιβές και η επόμενη κατάσταση, αλλά ο τρόπος παραγωγής δεν είναι γνωστός στον μαθητή ούτε στοιχεία (μέση τιμή αμοιβής ή πιθανότητες μετάβασης) για τα δεδομένα αυτά. Αξίζει να σημειωθεί πως υπάρχουν τρόποι προσομοίωσης όπου δεν είναι πλήρως γνωστές, ούτε στον μηχανισμό παραγωγής τέτοιων δεδομένων, οι κατανομές βάσει των οποίων παράγονται αυτά,.

Ονομάζουμε ***επεισόδιο/σενάριο*** ολόκληρη την διαδοχή καταστάσεων και αποφάσεων από μια αρχική κατάσταση έως τον τερματισμό αυτής της διαδοχής, είτε αυτή συμβαίνει μετά από συγκεκριμένο πλήθος χρονικών βημάτων, είτε φτάνοντας σε μία τερματική κατάσταση.

5.1 ΕΚΤΙΜΗΣΗ MONTE CARLO

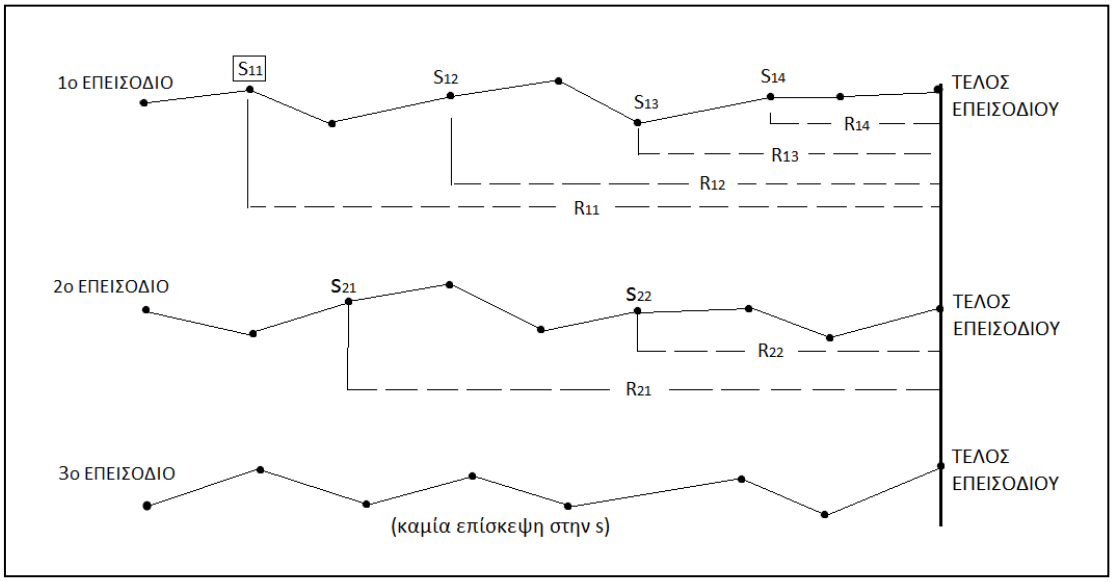
Η λογική της μεθόδου *Monte Carlo* είναι πως παρατηρεί την ολοκλήρωση κάθε επεισοδίου /σεναρίου όπου κατά την διάρκειά του συλλέγει τις αμοιβές που ακολούθησαν κάθε κατάσταση s και απόφαση a. Τις αμοιβές αυτές χρησιμοποιεί για να εκτιμήσει την μέση τιμή για κάθε αθροιστική αμοιβή που ακολούθησε ένα ζεύγος (s,a). Έτσι, χρειάζεται μεγάλο πλήθος επεισοδίων/σεναρίων (θεωρητικά άπειρο), για να μπορεί να υπολογίσει αξιόπιστες μέσες τιμές και να τις συνδέσει με τις καταστάσεις/αποφάσεις. Οι μέσες τιμές που εξάγονται με την *Monte Carlo* είναι εκτιμήσεις των τιμών των συναρτήσεων αξιολόγησης και ιδίως της συνάρτησης αξιολόγησης καταστάσεων-αποφάσεων. Κατά τα άλλα, η *Monte Carlo* ακολουθεί την λογική του Δυναμικού Προγραμματισμού, δηλαδή, ακολουθώντας δεδομένη πολιτική κάνει τις αντίστοιχες εκτιμήσεις για την συνάρτηση

αξιολόγησης της πολιτικής αυτής (*policy evaluation*), εν συνεχεία διορθώνει (βελτιώνει) την πολιτική (*policy improvement*), και τελικώς μετά από διαδοχικές βελτιώσεις της πολιτικής βρίσκει την βέλτιστη πολιτική $\pi*$.

Συγκεκριμένα, αν θέλουμε να εκτιμήσουμε την τιμή $V^{\pi}(s)$ της συνάρτησης αξιολόγησης καταστάσεων για την πολιτική π, θα πρέπει ακολουθώντας την πολιτική π, να ολοκληρώσουμε *N* το πλήθος επεισόδια/σενάρια. Κάθε φορά που διερχόμαστε από την κατάσταση s, αυτή καταγράφεται όπως καταγράφονται και οι αθροιστικές αμοιβές που ακολουθούν την κατάσταση αυτή έως το τέλος του επεισοδίου/σεναρίου. Παίρνοντας την μέση τιμή αυτών των αθροιστικών αμοιβών για τα *N* επεισόδια/σενάρια για τις φορές που περάσαμε από την κατάσταση s, έχουμε μία εκτίμηση της $V^{\pi}(s)$.

Μία σημαντική λεπτομέρεια ορίζει δύο διαφορετικές μεθόδους: η πρώτη ονομάζεται ***every-visit*** και η δεύτερη ***first-visit*** μέθοδος. Στην *every-visit* καταγράφονται σε κάθε επεισόδιο/σενάριο όλες οι επισκέψεις στην κατάσταση s, όπως βέβαια και οι αντίστοιχες αμοιβές. Η μέση τιμή υπολογίζεται από το σύνολο αυτών των καταγραφών. Στην *first-visit* μέθοδο καταγράφεται σε κάθε επεισόδιο/σενάριο μόνο η πρώτη επίσκεψη στην κατάσταση s και αντιστοίχως μόνο το σύνολο των αμοιβών μετά την πρώτη αυτή επίσκεψη, από όπου υπολογίζεται και η αθροιστική αμοιβή. Η μέση τιμή υπολογίζεται από το σύνολο αυτών των πρώτων καταγραφών. Αναλόγως του προβλήματος, ενδέχεται οι *every-visit* επισκέψεις να είναι πολύ περισσότερες από τις *first-visit* επισκέψεις, και ενώ οι *first-visit* επισκέψεις να φτάνουν σε πλήθος το πολύ το πλήθος των επεισοδίων/σεναρίων, οι *every-visit* επισκέψεις να είναι πολύ περισσότερες από το πλήθος των επεισοδίων/σεναρίων.

Στο παρακάτω σχήμα φαίνονται οι διαφορές των δύο μεθόδων, όπου για 3 επεισόδια/σενάρια έχουμε από μία πραγματοποίηση. Για τις επισκέψεις σε μία συγκεκριμένη κατάσταση s συμβολίζουμε με s_{ij}, όπου με i δηλώνεται το αντίστοιχο επεισόδιο και με j δηλώνεται η $j-οστή$ επίσκεψη στο επεισόδιο/σενάριο. Αντιστοίχως, συμβολίζουμε με R_{ij} τις αθροιστικές αμοιβές.



Από την *every-visit* θα είχαμε για την μέση τιμή των αθροιστικών αμοιβών της κατάστασης s, και επομένως μία εκτίμηση της $V^{\pi}(s)$:

$$V^{\pi}(s) = \frac{R_{11} + R_{12} + R_{13} + R_{14} + R_{21} + R_{22}}{6}$$

Ενώ από την *first-visit* θα είχαμε:

$$V^{\pi}(s) = \frac{R_{11} + R_{21}}{2}$$

Οι *first-visit* μέθοδος έχει καλύτερες ιδιότητες από την *every-visit*, καθώς σε κάποια προβλήματα μετά την επίσκεψη σε μία κατάσταση ενδέχεται να επανερχόμαστε σε αυτήν, οπότε μέρος της αθροιστικής αμοιβής από μία διέλευση θα καταγράφεται και ως αθροιστική αμοιβή για την επόμενη ή τις επόμενες. Η *every-visit*, αναφορικά με την καταγραφή των επισκέψεων σε μία κατάσταση s, θα χρησιμοποιηθεί στην *TD-Learning* (Μάθηση Χρονικών Διαφορών), στο επόμενο κεφάλαιο. Για την ακρίβεια, στην μάθηση Χρονικών Διαφορών θα δούμε πως οι ανανεώσεις των τιμών των εκτιμήσεων της συνάρτησης αξιολόγησης γίνονται σε κάθε βήμα, οπότε κάθε φορά που βρισκόμαστε στην κατάσταση s μέσα σε ένα επεισόδιο, ουσιαστικά θα καταγράψουμε την κατάσταση s και ότι επακολουθεί αυτής. Καλύτερα για την *Monte Carlo,* θα χρησιμοποιούμε την *first-visit* μέθοδο. Οι εκτιμήτριες $V_n^{\pi}(s)$ των τιμών $V^{\pi}(s)$, είναι αμερόληπτες, και συγκλίνουν στις πραγματικές τιμές, όταν το πλήθος των διελεύσεων από κάθε κατάσταση s τείνει στο άπειρο ($n \to \infty$).

5.2 BOOTSTRAPPING

Κατά κανόνα οι εκτιμήσεις των τιμών της συνάρτησης αξιολόγησης καταστάσεων είναι για διαφορετικές καταστάσεις ανεξάρτητες μεταξύ τους, σε αντίθεση όπως είδαμε με τον Δυναμικό Προγραμματισμό, όπου οι τιμές συνδέονται μεταξύ τους με τις εξισώσεις βελτιστοποίησης. Στον Δυναμικό προγραμματισμό, δηλαδή, κάθε τιμή υπολογίζεται βάσει των υπολοίπων, όπως και κάθε άλλη τιμή στηρίζεται και στην πρώτη. Η ιδιότητα αυτή στην στατιστική ορολογία ονομάζεται *Bootstrap*. Η *Monte Carlo* δεν κάνει *Bootstrap* και αυτό είναι πολύ καλό, από την άποψη πως αν θέλει ο μαθητής μπορεί να υπολογίσει τις τιμές της συνάρτησης αξιολόγησης μόνο για κάποιο υποσύνολο των καταστάσεων και όχι για ολόκληρο το σύνολο των καταστάσεων. Αυτό χρειάζεται να το κάνει γιατί κάποιες από τις καταστάσεις μπορεί να είναι σπάνιο να τις "συναντήσουμε" στο πρόβλημα, ή ακόμα και γιατί μπορεί να θέλουμε να συγκρίνουμε κάποιες καταστάσεις (ή αποφάσεις) μεταξύ τους (μερική επίλυση, π.χ. αν θέλουμε να επιλέξουμε μεταξύ δύο αποφάσεων) και όχι να επιλύσουμε εξαρχής το πρόβλημα. Αυτή, λοιπόν, η μερική επίλυση του προβλήματος είναι μία πολύ καλή ιδιότητα της *Monte Carlo* γιατί μειώνει πάρα πολύ το υπολογιστικό κόστος, σε σχέση με την εξολοκλήρου επίλυση του προβλήματος.

5.3 Η ΣΥΝΑΡΤΗΣΗ ΑΞΙΟΛΟΓΗΣΗΣ ΚΑΤΑΣΤΑΣΕΩΝ-ΑΠΟΦΑΣΕΩΝ $Q^{\pi}(s,a)$

Για την μέθοδο *Monte Carlo* είναι συνήθως πιο βολικό να εκτιμάται/υπολογίζεται όχι η συνάρτηση αξιολόγησης καταστάσεων $V^{\pi}(s)$, αλλά η συνάρτηση αξιολόγησης καταστάσεων-αποφάσεων $Q^{\pi}(s,a)$. Αυτό συμβαίνει γιατί στην μέθοδο *Monte Carlo* δεν είναι απαραίτητο το μοντέλο του περιβάλλοντος και επομένως θα υπάρχει δυσκολία στο να υπολογιστεί η βέλτιστη πολιτική $\pi*$, έστω και αν έχει βρεθεί η V^*. Σε αντίθεση, όταν έχει υπολογιστεί η $Q^*(s,a)$, η βέλτιστη απόφαση $\alpha*$ για κάθε κατάσταση s, βρίσκεται από την σχέση $\alpha^* = \arg\max_a\{Q^*(s,a)\}$. Θυμίζουμε ότι $Q^{\pi}(s,a)$ είναι η αξιολόγηση της κατάστασης s όταν λαμβάνεται η απόφαση a (a ανεξάρτητη της πολιτικής π), ενώ στις υπόλοιπες καταστάσεις ακολουθείται η πολιτική π. Αυτό συνεπάγεται πως για δύο δεδομένες αποφάσεις a_1, a_2 μίας κατάστασης s, καλύτερη είναι αυτή που έχει μεγαλύτερη τιμή της συνάρτησης αξιολόγησης καταστάσεων-αποφάσεων:

$$\Gamma\iota\alpha\ \tau\eta\nu\ \kappa\alpha\tau\alpha\sigma\tau\acute{\alpha}\sigma\eta\ s, a_1\ \kappa\alpha\lambda\acute{\upsilon}\tau\epsilon\rho\eta\ \tau\eta\varsigma\ a_2 \Leftrightarrow Q(s,a_1) > Q(s,a_2).$$

Αυτό βέβαια, διαχωρίζει την κατάσταση s με βάση όλες τις δυνατές αποφάσεις $a \in A(s)$. Δηλαδή, αν για την κατάσταση s έχουμε κ_s δυνατές αποφάσεις, τότε από την s προκύπτουν οι $(s,a_1),(s,a_2),\ldots,(s,a_{\kappa_s})$ καταστάσεις για την συνάρτηση Q. Επομένως, οι $|S|$ το πλήθος τιμές για την συνάρτηση αξιολόγησης V, γίνονται έως και $|S|\cdot|A|$ το πλήθος καταστάσεις-αποφάσεις για την συνάρτηση αξιολόγησης Q.

Η *Monte Carlo* μέθοδος, για την εύρεση της βέλτιστης λύσης, απαιτεί δύο συνθήκες:

Συνθήκη (Ι): να παρατηρηθεί πολύ μεγάλος αριθμός επεισοδίων/σεναρίων, ώστε να είναι αξιόπιστες οι εκτιμήσεις των τιμών $Q^{\pi}(s,a)$, για μία πολιτική π.

Συνθήκη (ΙΙ):όλοι οι δυνατοί συνδυασμοί (s,a) να προσπελαθούν, ώστε να έχουμε εκτιμήσεις για κάθε δυνατό συνδυασμό καταστάσεων-αποφάσεων.

ΣΥΝΘΗΚΗ (Ι)

Για την συνθήκη (Ι), όπως και στον Δυναμικό Προγραμματισμό, μπορούμε να σταματήσουμε την *policy evaluation* πριν αυτή να ολοκληρωθεί (πριν επέλθει δηλαδή η σύγκλιση προς τις τιμές $Q^{\pi}(s,a)$). Αντιστοίχως με την *value iteration* του Δυναμικού Προγραμμαισμού. και στην *Monte Carlo* μπορούμε ακολουθώντας συγκεκριμένη πολιτική να κάνουμε ένα βήμα υπολογισμού των $Q^{\pi_\kappa}(s,a)$ (ένα βήμα ανανέωσης των τιμών και αυτό βέβαια μετά το πέρας ενός επεισοδίου), και ακολούθως να κάνουμε *policy improvement* καλυτερεύοντας την υπάρχουσα πολιτική π_κ προς την επόμενη $\pi_{\kappa+1}$, επιλέγοντας την καλύτερη απόφαση βάσει των τελευταίως ενημερωμένων τιμών $Q^{\pi_\kappa}(s,a)$. Δηλαδή, για κάθε κατάσταση s, να επιλέξουμε ως βέλτιστη απόφαση $\pi_{\kappa+1}(s) = \arg\max_\alpha\{Q^{\pi_\kappa}(s,a)\} = \alpha_s^*$. Όντως, η νέα πολιτική $\pi_{\kappa+1}$ είναι καλύτερη της π_κ καθώς :

$$Q^{\pi_\kappa}\left(s,\alpha_s^*\right) = Q^{\pi_\kappa}\left(s,\arg\max_\alpha\{Q^{\pi_\kappa}(s,a)\}\right) = \max_\alpha\left\{Q^{\pi_\kappa}(s,a)\right\} \geq Q^{\pi_\kappa}\left(s,\pi_\kappa(s)\right) = V^{\pi_\kappa}(s)$$

δηλαδή, επιλέγοντας την α_s^* (βέλτιστη απόφαση έως "τώρα" βάσει των $Q^{\pi_\kappa}(s,a)$), ακολουθώντας την π_κ έχουμε καλύτερη τιμή για την συνάρτηση αξιολόγησης V^{π_κ} τουλάχιστον για την κατάσταση s.

Έτσι, ξεκινώντας από μία αυθαίρετη πολιτική π_0 και με αυθαίρετες αρχικές τιμές για την συνάρτηση αξιολόγησης $Q^{\pi_0}(s,a) \overset{\pi.\chi.}{=} 0, \ \forall(s,a)$, μετά το πρώτο επεισόδιο/σενάριο ανανεώνουμε τις εκτιμήτριες/τιμές των $Q^{\pi_0}(s,a)$, και επιλέγουμε για κάθε κατάσταση s ως καλύτερη απόφαση αυτήν που έχει καλύτερη τιμή $Q^{\pi_0}(s,\cdot)$. Αυτήν την απόφαση ορίζουμε ως απόφαση για την επόμενη πολιτική π_1 για την κατάσταση s και αυτό για κάθε κατάσταση. "Τρέχουμε" ή παρατηρούμε το επόμενο επεισόδιο/σενάριο και ανανεώνουμε τις τιμές $Q^{\pi_0}(s,a)$ σε $Q^{\pi_1}(s,a)$, οπότε και επιλέγουμε τις καλύτερες αποφάσεις $\forall s$ και τις θέτουμε ως αποφάσεις της επόμενης πολιτικής π_2, και ούτω καθεξής. Τελικώς, φτάνουμε στην βέλτιστη πολιτική. Εδώ, το «τελικώς» ελέγχεται από διαφόρους ελέγχους σύγκλισης, όπως διαγράμματα, στατιστικά στοιχεία κ.τ.λ.

ΣΥΝΘΗΚΗ (ΙΙ) - ΠΡΟΣΟΜΟΙΩΣΗ

Για την συνθήκη (ΙΙ), η οποία απαιτεί όλα τα ζεύγη καταστάσεων-αποφάσεων να προσπελαθούν, ώστε να έχουμε εκτιμήσεις για το σύνολο του προβλήματος, και ειδικά όταν η μάθηση μπορεί να πραγματοποιηθεί μέσω προσομοίωσης, μπορούμε να επιλέγουμε στην αρχή κάθε επεισοδίου, ισοπίθανα μία από όλες τις δυνατές (s,a). Έτσι, σε βάθος χρόνου θα έχουμε διατρέξει ως αρχική κατάσταση/απόφαση όλους τους δυνατούς συνδυασμούς, και θα έχουμε όλες τις ζητούμενες εκτιμήσεις. Ο τρόπος αυτός της τυχαίας προσπέλασης των αρχικών καταστάσεων-αποφάσεων, επειδή είναι σαν να εξερευνούμε όλες τελικά τις καταστάσεις, στην βιβλιογραφία αναφέρεται ως ***Exploring Starts.***

Ο αλγόριθμος *Monte Carlo* (προσομοίωση) με τυχαίες αρχικές καταστάσεις σε κάθε επεισόδιο/σενάριο είναι:

Monte Carlo E.S.– simulation *(first-visit / Exploring Starts)*

$\Gamma\iota\alpha\ \kappa\acute{\alpha}\theta\varepsilon\ s\in S, a\in A(s)\quad o\rho\acute{\iota}\zeta\varepsilon\tau\alpha\iota\quad Q(s,a)\leftarrow 0\ \ (\alpha\upsilon\theta\alpha\acute{\iota}\rho\varepsilon\tau\alpha)$

$\pi(s)\leftarrow(\alpha\upsilon\theta\alpha\acute{\iota}\rho\varepsilon\tau\eta)$

$\mathrm{Re}turns(s,a)\leftarrow\kappa\varepsilon\nu\acute{\eta}\ \lambda\acute{\iota}\sigma\tau\alpha$

Repeat 1. $\Pi\rho o\sigma o\mu o\acute{\iota}\omega\sigma\eta\ \varepsilon\nu\acute{o}\varsigma\ \varepsilon\pi\varepsilon\iota\sigma o\delta\acute{\iota}o\upsilon\ \mu\varepsilon\ \tau\upsilon\chi\alpha\acute{\iota}\alpha\ \alpha\rho\chi\iota\kappa\acute{\eta}\ \kappa\alpha\tau\acute{\alpha}\sigma\tau\alpha\sigma\eta$ $\kappa\alpha\iota\ \upsilon\pi\acute{o}\ \tau\eta\nu\ \pi o\lambda\iota\tau\iota\kappa\acute{\eta}\ \pi.$

2. $\Gamma\iota\alpha\ \kappa\acute{\alpha}\theta\varepsilon\ (s,a)\ \pi o\upsilon\ \kappa\alpha\tau\alpha\gamma\rho\acute{\alpha}\phi\theta\eta\kappa\varepsilon\ \sigma\tau o\ \varepsilon\pi\varepsilon\iota\sigma\acute{o}\delta\iota o:$

$R\leftarrow\acute{\alpha}\theta\rho o\iota\sigma\mu\alpha\ \alpha\mu o\iota\beta\acute{\omega}\nu\ \mu\varepsilon\tau\acute{\alpha}\ \tau\eta\nu\ \pi\rho\acute{\omega}\tau\eta\ (s,a)$

$\pi\rho o\sigma\acute{\alpha}\rho\tau\eta\sigma\eta\ \tau o\upsilon\ R\ \sigma\tau\eta\nu\ \lambda\acute{\iota}\sigma\tau\alpha\ R(s,a)$

$Q(s,a)\leftarrow\mu\acute{\varepsilon}\sigma\eta\ \tau\iota\mu\acute{\eta}\{R(s,a)\}$

3. $\Gamma\iota\alpha\ \kappa\acute{\alpha}\theta\varepsilon\ s\ \pi o\upsilon\ \kappa\alpha\tau\alpha\gamma\rho\acute{\alpha}\phi\theta\eta\kappa\varepsilon\ \sigma\tau o\ \varepsilon\pi\varepsilon\iota\sigma\acute{o}\delta\iota o$

$\pi(s)\leftarrow\arg\max_a\{Q(s,a)\}.$

until $(\Sigma\upsilon\nu\theta\acute{\eta}\kappa\eta\ \sigma\acute{\upsilon}\gamma\kappa\lambda\iota\sigma\eta\varsigma)$

Η συνθήκη σύγκλισης στον αλγόριθμο καθορίζεται έτσι ώστε, οι τελικές εκτιμήσεις να είναι αξιόπιστες, δηλαδή αμερόληπτες και με μικρή τυπική απόκλιση ώστε να είμαστε βέβαιοι πως έχει επέλθει σύγκλιση. Αυτό μπορεί να γίνεται εναλλακτικώς, αν αντί της εντολής repeat τρέξουμε τις αντίστοιχες εντολές για συγκεκριμένο πλήθος επαναλήψεων, και μετά να γίνεται ο έλεγχος της σύγκλισης.

ΣΥΝΘΗΚΗ (II) – ΣΤΟΧΑΣΤΙΚΗ ΠΟΛΙΤΙΚΗ

Για την συνθήκη (II), υπάρχει και δεύτερος τρόπος αντιμετώπισης, εφαρμόσιμος και στην περίπτωση όπου μπορεί να γίνει προσομοίωση, αλλά και σε περιπτώσεις όπου αυτό είναι αδύνατο. Υπάρχουν προβλήματα όπου απαιτούν την άμεση αλληλεπίδραση με το περιβάλλον και η μάθηση γίνεται μέσω πραγματικής εμπειρίας και όχι μέσω προσομοίωσης. Ο μαθητής, όπως έχουμε αναφέρει και προηγουμένως, παρατηρεί απευθείας το περιβάλλον και συγκεκριμένα τις αμοιβές και τις καταστάσεις που ακολουθούν κάθε του απόφαση. Σε τέτοιες περιπτώσεις αν ο μαθητής έχει ντετερμενιστική πολιτική, ενδεχομένως δεν θα περάσει ποτέ από κάποιες καταστάσεις και αποφάσεις. Με επιλογή, όμως, στοχαστικής πολιτικής και μάλιστα τέτοιας ώστε, να δίνει μη μηδενική πιθανότητα επιλογής σε κάθε δυνατή απόφαση, σε βάθος χρόνου θα έχει επιλέξει όλες τις δυνατές αποφάσεις, και αυτό θα έχει οδηγήσει τον μαθητή στην προσπέλαση όλων των δυνατών καταστάσεων/αποφάσεων. Δηλαδή, πάντα μέσω της στοχαστικής πολιτικής, μπορούμε να επιλέγουμε την καλύτερη απόφαση, αλλά να αφήνουμε ανοικτό και το ενδεχόμενο επιλογής οποιασδήποτε άλλης απόφασης, ανεξαρτήτως του χαρακτηρισμού της, έως τότε, ως "καλής" ή "κακής". Υπάρχουν, δύο μέθοδοι επιλογής τέτοιων στοχαστικών πολιτικών. Είναι η ***on-policy*** και η ***off-policy*** μέθοδος. Η πρώτη θέλοντας να εκτιμήσει μία πολιτική, ακολουθεί μία αντίστοιχη ελαφρώς αλλαγμένη στοχαστική πολιτική, ενώ η δεύτερη μέθοδος, για να εκτιμήσει μία πολιτική ακολουθεί μία άλλη πολιτική, πιθανόν και πολύ διαφορετική.

5.4 ΜΕΘΟΔΟΣ ON POLICY – ε-greedy ΠΟΛΙΤΙΚΗ

Οι *ε*-greedy πολιτικές, (σύνολο πολιτικών για $\varepsilon \in (0,1)$), είναι στοχαστικές πολιτικές, και ορίζονται ως "μετασχηματισμός" μίας greedy ντετερμενιστικής πολιτικής (greedy=άπληστη: αποζητά την μοναδική καλύτερη απόφαση χωρίς να υπολογίζει άλλες παραμέτρους). Θυμίζουμε πως, ακολουθώντας μία πολιτική *π* έχουμε εκτιμήσεις για τις $Q^{\pi}(s,a)$, και η *greedy* πολιτική είναι η ντετερμενιστική πολιτική που ορίζεται από την σχέση $\pi(s) = \max_{a} \left\{ Q^{\pi}(s,a) \right\} = a*$. Ως στοχαστική θα γραφόταν :

greedy πολιτική: $\pi(s) = (0,0,...,0,1,0,...,0)$ με μονάδα που αντιστοιχεί στην απόφαση $a*$.

Η αντίστοιχη *ε*-greedy στοχαστική πολιτική π', ορίζεται βάσει της greedy πολιτικής π, όπου ίσες πιθανότητες $\varepsilon / |\mathrm{A}(s)|$ επιλογής δίδονται σε κάθε απόφαση μηδενικής πιθανότητας της greedy πολιτικής.

Έτσι, έχουμε: ***ε-greedy πολιτική*** $\pi'(s,a)=\begin{cases}\dfrac{\varepsilon}{|\mathrm{A}(s)|}, & a\neq a^*\\ 1-\varepsilon+\dfrac{\varepsilon}{|\mathrm{A}(s)|}, & a=a^*\end{cases}$

Και εδώ ισχύει το *Policy Improvement Theorem*

Δηλαδή, $\pi'\geq\pi$ (και βέβαια όταν ισχύει ισότητα αυτή θα είναι και η βέλτιστη πολιτική).

Και είναι $\pi'\geq\pi$ γιατί:

$$Q^{\pi}\left(s,\pi'(s)\right)=\sum_a \pi'(s,a)\cdot Q^{\pi}\left(s,a\right)=\frac{\varepsilon}{|\mathrm{A}(s)|}\sum_a Q^{\pi}(s,a)+(1-\varepsilon)\cdot\max_a Q^{\pi}\left(s,a\right)\geq$$

$$\geq\frac{\varepsilon}{|\mathrm{A}(s)|}\sum_a Q^{\pi}(s,a)+(1-\varepsilon)\cdot\sum_a\frac{\pi(s,a)-\dfrac{\varepsilon}{|\mathrm{A}(s)|}}{(1-\varepsilon)}Q^{\pi}(s,a)=$$

$$=\cancel{\frac{\varepsilon}{|\mathrm{A}(s)|}\sum_a Q^{\pi}(s,a)}-\cancel{\frac{\varepsilon}{|\mathrm{A}(s)|}\sum_a Q^{\pi}(s,a)}+\sum_a\pi(s,a)Q^{\pi}(s,a)=$$

$$=V^{\pi}(s,a)$$

Ο αλγόριθμος *Monte Carlo On-Policy* (*ε-shoft*) είναι:

***Monte Carlo On-Policy** (ε-greedy policy)*

Για κάθε $s\in S, a\in A(s)$ ορίζεται $Q(s,a)\leftarrow 0$ (αυθαίρετα)

$\pi(s)\leftarrow$ (αυθαίρετη, $\varepsilon-greedy$)

$\mathrm{Re}turns(s,a)\leftarrow$ κενή λίστα

Repeat

1. Προσομοίωση ενός επεισοδίου υπό την πολιτική π.
2. Για κάθε (s,a) που καταγράφθηκε στο επεισόδιο:
   - $R\leftarrow$ άθροισμα αμοιβών μετά την πρώτη (s,a)
   - προσάρτηση του R στην λίστα $R(s,a)$
   - $Q(s,a)\leftarrow$ μέση τιμή$\{R(s,a)\}$
3. Για κάθε s που καταγράφθηκε στο επεισόδιο
   - $\alpha^*\leftarrow \arg\max_a\{Q(s,a)\}$
   - $\pi(s,a)\leftarrow\begin{cases}\dfrac{\varepsilon}{|\mathrm{A}(s)|}, & a\neq a^*\\ 1-\varepsilon+\dfrac{\varepsilon}{|\mathrm{A}(s)|}, & a=a^*\end{cases}$

until (Συνθήκη σύγκλισης)

Η συνθήκη σύγκλισης στον αλγόριθμο, καθορίζεται έτσι ώστε οι τελικές εκτιμήσεις να είναι αξιόπιστες, δηλαδή αμερόληπτες και με μικρή τυπική απόκλιση ώστε να είμαστε βέβαιοι πως έχει επέλθει σύγκλιση. Αυτό μπορεί να γίνεται εναλλακτικώς, αν αντί της εντολής repeat τρέξουμε τις αντίστοιχες εντολές για συγκεκριμένο πλήθος επαναλήψεων, και μετά να γίνεται ο έλεγχος της σύγκλισης.

5.5 ΜΕΘΟΔΟΣ OFF POLICY

Στην *On policy* μέθοδο αυτό που κάνει ο αλγόριθμος είναι να ακολουθεί μία συγκεκριμένη πολιτική για την λήψη των αποφάσεων και την παραγωγή ενός επεισοδίου/σεναρίου. Αλλά μετά το πέρας του επεισοδίου η πολιτική αυτή βελτιώνεται και στο επόμενο επεισόδιο οι αποφάσεις λαμβάνονται βάσει της νέας βελτιωμένης αυτής πολιτικής. Υπάρχει όμως και η δυνατότητα να ακολουθείται μία πολιτική, ως προς την λήξη των αποφάσεων και την παραγωγή του επεισοδίου, αλλά να εκτιμάται μία δεύτερη πολιτική. Σε αυτήν την περίπτωση η πολιτική που ακολουθείται για την παραγωγή του συνόλου των επεισοδίων είναι πάντα η ίδια. Σε αυτό οφείλεται και η ονομασία ***off policy*** καθώς εκτιμώνται οι τιμές της συνάρτησης αξιολόγησης για μια πολιτική $V^{\pi}(s)$ ενώ ακολουθείται μία άλλη σταθερή πολιτική *π'*.

Για να μπορεί μπορούμε να εκτιμήσουμε τις $V^{\pi}(s)$ της *π* πολιτικής, ακολουθώντας την *π'* πολιτική, θα πρέπει για κάθε απόφαση που μπορούμε να λάβουμε όταν βρισκόμαστε στην κατάσταση *s* με την πολιτική *π* να έχουμε πιθανότητα μη μηδενική να λάβουμε την ίδια απόφαση με την πολιτική *π'*, δηλαδή να ισχύει η παρακάτω συνθήκη:

Συνθήκη : αν *π(s,a)>0 τότε π'(s,a)>0*

Για την *μέθοδο Monte Carlo* αυτό είναι εύκολο αρκεί να ακολουθείται μία *ε-soft* πολιτική (όλες οι δυνατές αποφάσεις να έχουν αυστηρά θετική πιθανότητα να επιλεγούν σε κάθε κατάσταση). Έστω ότι έχουμε *N* επεισόδια στα οποία ακολουθήσαμε την *π'* πολιτική, από τα οποία έχουμε N_S επεισόδια που επισκεφτήκαμε την κατάσταση *s* και έχουμε και τις αντίστοιχες αθροιστικές αμοιβές πρώτης μετάβασης, $R_i(s), i=1,\ldots,N_S$. Έστω $P_i(s)$ και $P_i'(s)$ οι πιθανότητες πραγματοποίησης της σειράς των γεγονότων, ξεκινώντας από την *s* και ακολουθώντας αντιστοίχως τις *π* και *π'*. Δηλαδή,

$$P_i(s) = \Pr\left(s_{v_i} = s, a_{v_i}, s_{v_i+1}, a_{v_i+1}, \ldots, s_T \mid \pi\right) = \prod_{k=v_i}^{T} \pi(s_k, a_k) P_{s_k s_{k+1}}^{a_k}$$

$$P_i'(s) = \Pr\left(s_{v_i} = s, a_{v_i}, s_{v_i+1}, a_{v_i+1}, \ldots, s_T \mid \pi'\right) = \prod_{k=v_i}^{T} \pi'(s_k, a_k) P_{s_k s_{k+1}}^{a_k}$$

Ορίζουμε ως τιμές της συνάρτησης αξιολόγησης τον σταθμισμένο μέσο:

$$V^{\pi}(s) = \frac{\sum_{i=1}^{N_S} \frac{P_i(s)}{P_i'(s)} \cdot R_i(s)}{\sum_{i=1}^{N_S} \frac{P_i(s)}{P_i'(s)}} = \frac{\sum_{i=1}^{N_S} \frac{\prod_{k=v_i}^{T} \pi(s_k, a_k)}{\prod_{k=v_i}^{T} \pi'(s_k, a_k)} \cdot R_i(s)}{\sum_{i=1}^{N_S} \frac{\prod_{k=v_i}^{T} \pi(s_k, a_k)}{\prod_{k=v_i}^{T} \pi'(s_k, a_k)}},$$

όπου στην τελευταία ισότητα οι παράγοντες των πιθανοτήτων μεταπήδησης απλοποιούνται, και μάλιστα χωρίς να είναι απαραίτητο να γνωρίζουμε το μοντέλο του

περιβάλλοντος, αλλά μόνο τις πιθανότητες επιλογής των αποφάσεων, δηλαδή εξαρτώνται μόνο από τις πολιτικές. Παρατηρούμε στα παραπάνω ομοιότητα με τον αλγόριθμο ***Metropolis Hastings Monte Carlo***.

Ο αντίστοιχος αλγόριθμος για την εκτίμηση των $Q^{\pi}(s,a)$ όπου η π' είναι ντετερμινιστική:

Monte Carlo Off-Policy

$\Gamma\iota\alpha\ \kappa\acute{\alpha}\theta\varepsilon\ s\in S, a\in A(s)\quad o\rho\acute{\iota}\zeta\varepsilon\tau\alpha\iota\quad Q(s,a)\leftarrow 0\ \ (\alpha\upsilon\theta\alpha\acute{\iota}\rho\varepsilon\tau\alpha)$

$N(s,a)\leftarrow 0$

$W(s,a)\leftarrow 0$

$\pi(s)\leftarrow(\alpha\upsilon\theta\alpha\acute{\iota}\rho\varepsilon\tau\eta\ \nu\tau\varepsilon\tau\varepsilon\rho\mu\iota\nu\iota\sigma\tau\iota\kappa\acute{\eta}\ \pi\omega\lambda\iota\tau\iota\kappa\acute{\eta})$

Repeat

1. $\pi\rho\omega\sigma\omega\mu\omega\acute{\iota}\omega\sigma\eta\ \varepsilon\nu\acute{o}\varsigma\ \varepsilon\pi\varepsilon\iota\sigma\omega\delta\acute{\iota}\omega\upsilon\ \upsilon\pi\acute{o}\ \tau\eta\nu\ \pi\omega\lambda\iota\tau\iota\kappa\acute{\eta}\ \pi'.$

   $(s_0,a_0,r_0,s_1,a_1,r_1,s_2,a_2,r_2,...,s_{T-1},a_{T-1},r_{T-1},s_T)$

   $\beta\rho\varepsilon\varsigma\ \tau\eta\nu\ \tau\varepsilon\lambda\varepsilon\upsilon\tau\alpha\acute{\iota}\alpha\ \chi\rho\omega\nu\iota\kappa\acute{\eta}\ \sigma\tau\iota\gamma\mu\acute{\eta}\ \tau\ \gamma\iota\alpha\ \tau\eta\nu\ o\pi o\acute{\iota}\alpha\ \iota\sigma\chi\acute{\upsilon}\varepsilon\iota\ \alpha_\tau\neq\pi(s_\tau)$

2. $\Gamma\iota\alpha\ \kappa\acute{\alpha}\theta\varepsilon\ (s,a)\ \pi o\upsilon\ \kappa\alpha\tau\alpha\gamma\rho\acute{\alpha}\varphi\theta\eta\kappa\varepsilon\ \sigma\tau o\ \varepsilon\pi\varepsilon\iota\sigma\acute{o}\delta\iota o\ \mu\varepsilon\tau\acute{\alpha}\ \tau\eta\nu\ \tau:$

   $t\leftarrow\eta\ \pi\rho\acute{\omega}\tau\eta\ \chi\rho\omega\nu\iota\kappa\acute{\eta}\ \sigma\tau\iota\gamma\mu\acute{\eta}\ \mu\varepsilon\tau\acute{\alpha}\ \tau\eta\nu\ \tau\ \pi o\upsilon\ \sigma\upsilon\nu\alpha\nu\tau\acute{\eta}\theta\eta\kappa\varepsilon\ \tau o\ (s,a)$

   $R_t\leftarrow\tau o\ \alpha\nu\tau\acute{\iota}\sigma\tau o\iota\chi o\ \acute{\alpha}\theta\rho o\iota\sigma\mu\alpha\ \alpha\mu o\iota\beta\acute{\omega}\nu$

   $w\leftarrow\dfrac{1}{\prod_{k=t+1}^{T-1}\pi'(s_k,a_k)}$

   $N(s,a)\leftarrow N(s,a)+w\cdot R_t$

   $W(s,a)\leftarrow W(s,a)+w$

   $Q(s,a)\leftarrow\dfrac{N(s,a)}{W(s,a)}$

3. $\Gamma\iota\alpha\ \kappa\acute{\alpha}\theta\varepsilon\ s\ \pi o\upsilon\ \kappa\alpha\tau\alpha\gamma\rho\acute{\alpha}\varphi\theta\eta\kappa\varepsilon\ \sigma\tau o\ \varepsilon\pi\varepsilon\iota\sigma\acute{o}\delta\iota o$

   $\pi(s)\leftarrow arg\max_a\{Q(s,a)\}$

until $(\Sigma\upsilon\nu\theta\acute{\eta}\kappa\eta\ \sigma\acute{\upsilon}\gamma\kappa\lambda\iota\sigma\eta\varsigma)$

Η συνθήκη σύγκλισης στον αλγόριθμο καθορίζεται έτσι ώστε, οι τελικές εκτιμήσεις να είναι αξιόπιστες, δηλαδή αμερόληπτες και με μικρή τυπική απόκλιση ώστε να είμαστε βέβαιοι πως έχει επέλθει σύγκλιση. Αυτό μπορεί να γίνεται εναλλακτικώς, αν αντί της εντολής repeat τρέξουμε τις αντίστοιχες εντολές για συγκεκριμένο πλήθος επαναλήψεων, και μετά να γίνεται ο έλεγχος της σύγκλισης.

Η μέθοδος o*ff policy* έχει ένα μειονέκτημα όταν η πολιτική π είναι ντετερμινιστική. Επειδή, αποκόπτει το επεισόδιο και συλλέγει στοιχεία μόνο από την "ουρά" του μπορεί να παράγονται επεισόδια τα οποία δεν χρησιμοποιούνται στην εκτίμηση, ή να χρησιμοποιούνται πολύ λίγα δεδομένα από το τέλος κάθε επεισοδίου. Ενδεχομένως μπορεί αντί της ντετερμινιστικής πολιτικής π να εκτιμάται μία στοχαστική πολιτική, όπου το φαινόμενο αυτό δεν θα υπάρχει. Η εφαρμογή της *off policy* στην μέθοδο χρονικών διαφορών (αλγόριθμος Q), δεν παρουσιάζει τέτοια προβλήματα.

5.6 ΠΛΕΟΝΕΚΤΗΜΑΤΑ-ΜΕΙΟΝΕΚΤΗΜΑΤΑ ΤΗΣ MONTE CARLO

Η μέθοδος *Monte Carlo* είναι ο δεύτερος τρόπος επίλυσης του προβλήματος της Ενισχυτικής Μάθησης μετά τον Δυναμικό Προγραμματισμό. Η σημαντική διαφοροποίηση στο πρόβλημα έναντι του Δυναμικού Προγραμματισμού είναι πως δεν είναι απαραίτητη η γνώση του μοντέλου του περιβάλλοντος. Αυτό δίνει, κατά πρώτον, την δυνατότητα προσομοίωσης στους αλγορίθμους (όποτε δεν απαιτείται η αλληλεπίδραση με το περιβάλλον), ενώ κατά δεύτερον, εφαρμόζεται και σε προβλήματα όπου πραγματικά δεν γνωρίζουμε τίποτα για το μοντέλο του περιβάλλοντος και έχουμε μόνο την παρατήρηση του μαθητή για να λαμβάνουμε τις αμοιβές και τις επόμενες καταστάσεις.

Ένα τρίτο πλεονέκτημα, μετά την προσομοίωση και την, ενδεχομένως, όχι πλήρη γνώση του περιβάλλοντος, είναι η Μαρκοβιανή ιδιότητα. Έχει παρατηρηθεί, πως σε προβλήματα όπου δεν ισχύει η Μαρκοβιανή ιδιότητα, οι αλγόριθμοι δουλεύουν ικανοποιητικά. Αυτό συμβαίνει γιατί η *Monte Carlo* δεν κάνει *Bootstrapping*, δηλαδή δεν στηρίζεται σε εκτιμήσεις για να βγάλει άλλες εκτιμήσεις, απλά ανανεώνει τις εκτιμήσεις της καθώς συλλέγει περισσότερα δεδομένα.

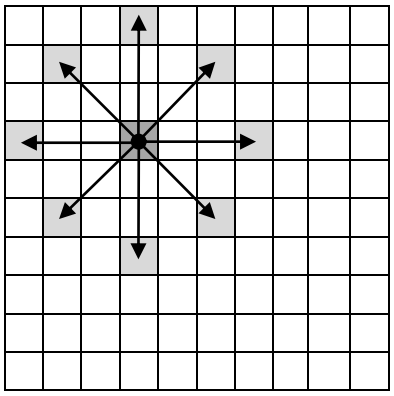
Τέταρτο πλεονέκτημα της *Monte Carlo* έναντι του Δυναμικού Προγραμματισμού, είναι πως μπορεί να ασχοληθεί μόνο με ένα υποσύνολο των καταστάσεων και όχι με ολόκληρο το πρόβλημα. Κάτι τέτοιο μειώνει πολύ τον χρόνο επίλυσης (αν και πάντα ελλοχεύει ο κίνδυνος λανθασμένων εκτιμήσεων). Είναι ζητούμενη, όμως, αυτή η ιδιότητα ειδικά σε περιπτώσεις όπου κάποιο σύνολο καταστάσεων είναι απίθανο να συμβούν κατά την ροή της πραγματοποίησης του προβλήματος, ή είναι σαφώς χειρότερες ως καταστάσεις από άλλες και επομένως δεν θέλουμε να καθυστερούν τους υπολογισμούς κατά την επίλυση (ειδικά σε προσομοίωση, όπου επιλέγεται να μην διατρέχονται οι καταστάσεις αυτές).

5.7 Η ΚΑΤΑΡΑ ΤΗΣ ΔΙΑΣΤΑΣΗΣ

Η κατάρα της διάστασης εμφανίζεται και στην *Monte Carlo*, αλλά λόγω του πλεονεκτήματός της, επιλεκτικά να περιορίζεται η επίλυση σε κάποιο υποσύνολο των καταστάσεων, και κυρίως λόγω του ότι στον Δυναμικό Προγραμματισμό στην επίλυση εμπλέκονται όλες οι δυνατές επόμενες καταστάσεις, και επομένως και οι αντίστοιχες τιμές της συνάρτησης αξιολόγησης, μπορεί το φαινόμενο να είναι μικρότερο στην *Monte Carlo* από ότι στον Δυναμικό Προγραμματισμό.

Ένα παράδειγμα για να πάρουμε μία ιδέα του φαινομένου της κατάρας της διάστασης, είναι το παρακάτω, όπου θα δώσουμε εμμέσως την λύση του και όχι απευθείας με Δυναμικό Προγραμματισμό ή *Monte Carlo*.

Θεωρούμε 100 κελιά σε σχηματισμό τετραγώνου 10x10. Σε αυτά θέλουμε να τοποθετήσουμε με την σειρά τους αριθμούς 1,2,3,…,99,100 ξεκινώντας με την τοποθέτηση του αριθμού 1, από το κελί στην πάνω αριστερή θέση. Σε κάθε κελί μπορεί να τοποθετηθεί μόνο ένας αριθμός και το επόμενο κελί επιλέγεται από τα δυνατά “ελεύθερα” κελιά μέσα στον σχηματισμό του 10x10 τετραγώνου, με τρόπο μετάβασης όπως φαίνεται στο διπλανό σχήμα.



Επιλογή κίνησης προς το επόμενο κελί. Για μετακίνηση οριζοντίως ή καθέτως αφήνουμε 2 κενά κελιά, ενώ για μετακίνηση διαγωνίως αφήνουμε 1 κενό κελί.

Το μέγιστο πλήθος κελιών, τα οποία μπορούν να επιλεχθούν για την τοποθέτηση του επόμενου αριθμού είναι οκτώ, και βέβαια μπορεί να φτάνει και σε μηδέν το πλήθος αυτών των κελιών. Το πλήθος των κελιών μετάβασης εξαρτάται από την θέση (κοντά σε ακριανές στήλες ή γραμμές του σχηματισμού) αλλά και την πιθανή κατάληψη, από προηγούμενους αριθμούς, στις θέσεις κελιών όπου θα μπορούσε να γίνει μετάβαση όπως δόθηκε στο σχήμα. Η τοποθέτηση των αριθμών θα τερματιστεί όταν τοποθετηθεί και το 100-οστό κελί (με τον αριθμό 100), ή όταν δεν υπάρχει άλλο ελεύθερο κελί για μετάβαση - τοποθέτηση του επόμενου αριθμού. Οι πιθανές καταστάσεις του 10x10 σχηματισμού είναι πάρα πολλές και δύσκολο να υπολογιστούν, αλλά είναι πιθανόν της τάξης του 10^35, όπως μπορούμε ενδεικτικά να υπολογίσουμε παρακάτω και μάλιστα χωρίς ακρίβεια χρησιμοποιώντας μία μόνο συμπλήρωση του τετραγώνου. Ακόμα και αν θεωρήσουμε ως κατάσταση την θέση του τελευταίου αριθμού *n* που τοποθετήθηκε και τις θέσεις των ελεύθερων κελιών (ανεξαρτήτως της ακριβής θέσης των *n-1* προηγουμένων αριθμών) το πλήθος για τις δυνατές καταστάσεις είναι επίσης πολύ μεγάλο και βέβαια τέτοια μεγέθη καταστάσεων είναι **απαγορευτικά για τον Δυναμικό Προγραμματισμό**. Μπορούμε, όμως, να δοκιμάσουμε την *Monte Carlo* μέθοδο, και μάλιστα με πολιτική ή οποία ορίζει τυχαία επιλογή (με ίσες πιθανότητες) μεταξύ των δυνατών επιλογών για την τοποθέτηση του επόμενου αριθμού.

Η τοποθέτηση και των 100 αριθμών είναι εφικτή, και κάθε τέτοια τοποθέτηση θα την ονομάζουμε λύση του προβλήματος. Μία τέτοια λύση είναι η παρακάτω:

1	43	72	2	44	71	3	45	70	4
88	98	21	89	99	22	58	***100***	23	59
31	78	49	32	81	50	33	82	51	34
12	42	73	13	57	90	14	46	69	5
87	97	20	79	96	17	80	95	24	60
30	77	48	91	74	47	26	83	52	35
11	41	86	16	56	85	15	37	68	6
64	92	19	63	93	18	62	94	25	61
29	76	55	28	75	54	27	84	53	36
10	40	65	9	39	66	8	38	67	7

Για τον υπολογισμό του πλήθους των εναλλακτικών διαδρομών-τοποθέτησης των 100 αριθμών, μπορούμε να πάρουμε για κάθε ένα κελί από τα 100, τον αριθμό των επιλογών που είχαμε ως επόμενη τοποθέτηση, και αυτό για την συγκεκριμένη λύση! Αυτός δεν είναι εντελώς σωστός υπολογισμός, καθώς για διαφορετική επιλογή θα μπορούσε ο τερματισμός της τοποθέτησης να είχε πραγματοποιηθεί πριν το 100-οστό κελί. Για κάθε τοποθέτηση, λοιπόν, ενός αριθμού, σημειώσαμε το πλήθος των δυνατών επιλογών που είχαμε την δεδομένη στιγμή και φαίνονται στον διπλανό πίνακα, όπου συμπληρώθηκε κατά στήλες αντιστοίχως με την σειρά τοποθέτησης των αριθμών στον πίνακα της δοσμένης λύσης.

3	4	2	2	3	2	2	3	1	1
4	3	2	5	4	2	4	2	1	2
4	6	1	4	1	1	4	5	2	2
2	5	2	3	3	1	2	3	1	1
4	5	1	3	2	1	3	3	3	2
4	4	7	1	3	2	3	1	2	2
2	7	4	5	6	1	1	1	1	1
4	5	4	2	5	4	3	2	1	1
4	4	2	3	3	1	2	4	1	1
2	4	3	2	3	3	1	1	2	1

Το γινόμενο των αριθμών αυτών μας δίνει μία ένδειξη του πλήθους των καταστάσεων, και για την λύση που δώσαμε το πλήθος αυτό είναι $7.3867 \cdot 10^{35}$. Μπορούμε να παρατηρήσουμε εδώ πως για τα τελευταία 25 κελιά το πλήθος πολλαπλασιάζεται κατά $1^{15} \cdot 2^8 \cdot 3 \cdot 4 = 3072$, πολύ μικρός αριθμός σε σχέση με το συνολικό! Αυτό οφείλεται στο ότι καθώς συμπληρώνονται κελιά με αριθμούς μειώνονται και οι δυνατές επιλογές για την τοποθέτηση των επόμενων αριθμών. Έτσι, μπορούμε να ισχυριστούμε πως το πλήθος των καταστάσεων δεν αλλάζει δραματικά είτε τερματιστεί η διαδικασία της τοποθέτησης των αριθμών στο 100 είτε νωρίτερα, άλλωστε όπως προαναφέραμε ο παραπάνω υπολογισμός

είναι απλά μία ένδειξη του πλήθους των καταστάσεων του προβλήματος. Παρατηρούμε πως στα αρχικά κελιά οι εναλλακτικές είναι 3, 4 ,5 έως και 7 ενώ στις τελευταίες 25 τοποθετήσεις έχουμε κυρίως 1 και 2 ενώ έχουμε μόνο για ένα κελί 3 και για άλλο ένα 4 εναλλακτικές τοποθετήσεις. Το μέσο πλήθος επιλογών για την επόμενη τοποθέτηση είναι 2,68 για την συγκεκριμένη λύση.

Υπάρχουν λύσεις για το πρόβλημα αυτό, και θεωρητικά είμαστε "σίγουροι", με πιθανότητα 1, πως ακόμα και με τυχαία τοποθέτηση των αριθμών, μία τέτοια λύση θα βρεθεί "κάποτε" και βέβαια εννοούμε μετά από πεπερασμένο πλήθος δοκιμών. Όπως προαναφέραμε το πλήθος των καταστάσεων είναι απαγορευτικό για την χρήση του Δυναμικού Προγραμματισμού, αλλά για την χρήση της *Monte Carlo* δεν έχουμε κανένα πρόβλημα, τουλάχιστον στο να δοκιμάσουμε την εύρεση της λύσης μέσω αυτής!

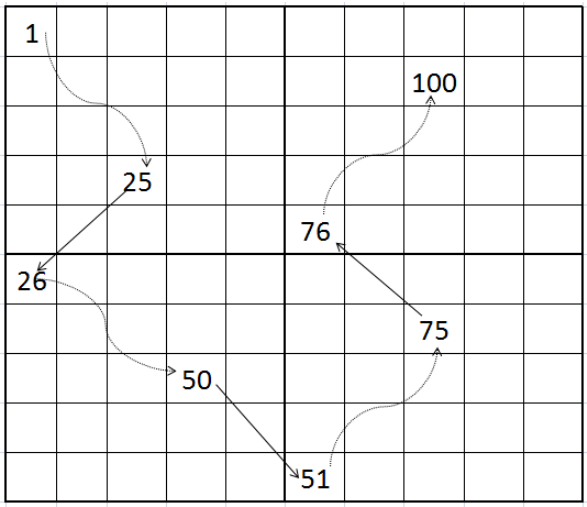
Με αλγόριθμο γραμμένο στην *MATLAB (παράρτημα 2)* και όπως αναφέραμε τυχαία επιλογή του επόμενου κελιού για την τοποθέτηση του επομένου αριθμού, κάναμε κάποιες δοκιμές. Ενδεικτικά αναφέρουμε πως, ο αλγόριθμος έπρεπε να τρέξει 7183 φορές για να καταφέρει να τοποθετήσει έως και κάποιον αριθμό μεγαλύτερο του 92 για μία και μόνο φορά! Εάν χρησιμοποιούσαμε την *Monte Carlo* μέθοδο για να εκτιμήσουμε έστω και μία από τις καταστάσεις δεν θα μπορούσαμε σε εύλογο χρονικό διάστημα. Ακόμα αναφέρουμε πως για να καταφέρει ο αλγόριθμος να τοποθετήσει έως και το αριθμό 98 έκανε 134.346 προσπάθειες (οι 134.345 ανεπιτυχείς), για την τοποθέτηση έως και του αριθμού 99 για να δώσει αποτέλεσμα και μάλιστα σε δύο προσπάθειες (2,5 και 4,25 ώρες) χρειάστηκε αντιστοίχως πάνω από 5.360.000 και πάνω από 21.930.000 προσπάθειες. Όταν ο αλγόριθμος χρησιμοποιήθηκε για να βρεθεί μία λύση (τοποθέτηση και του 100), μετά από 2 ημέρες και 10 ώρες έχοντας κάνει πάνω από 113.000.000 προσπάθειες δεν είχε καταφέρει να βρει ακόμα λύση, οπότε και τερματίστηκε ηθελημένα το πρόγραμμα!

Βέβαια, λύση ακόμα και μέσω της *Monte Carlo* μπορεί να δοθεί εμμέσως!

<u>Μικρότερη διάσταση!</u>

Το παραπάνω πρόβλημα μπορεί να γίνει πιο απλό αν τον αρχικό σχηματισμό 10x10 τετραγώνων τον χωρίσουμε σε 4 μικρότερα τετράγωνα 5x5. Οι καταστάσεις έτσι μειώνονται δραματικά και η επίλυση γίνεται εξαιρετικά πιο εύκολη. Παρά την μείωση του πλήθους των τετραγώνων κάθε πλευράς στο μισό, ή του πλήθους όλων των τετραγώνων στο ένα τέταρτο, ο χρόνος επίλυσης είναι μερικά εκατοστά του δευτερολέπτου!

Συγκεκριμένα, για να μπορεί να συμπληρωθεί το 10x10 τετράγωνο χωρισμένο στα τέσσερα, δηλαδή χωρισμένο σε τεταρτημόρια, θα πρέπει από εκεί που σταμάτησε το $1^{ο}$ τεταρτημόριο να μπορεί να συνεχιστεί η τοποθέτηση των επόμενων 25 αριθμών στο επόμενο τεταρτημόριο και ομοίως στα επόμενα 2 τεταρτημόρια. Μία τέτοια υλοποίηση μπορεί να γίνει σύμφωνα με το παρακάτω σχήμα, όπου με την καμπύλη γραμμή σημειώνεται το επιμέρους πρόβλημα του 5x5 τετραγώνου:



Εύκολα πραγματοποιείται και η επιλεκτική επίλυση: να τερματίσουμε την τοποθέτηση των 25 αριθμών σε συγκεκριμένο κελί. Για τον τερματισμό στο κελί (4,3) του 5x5 πίνακα, ο αλγόριθμος δίνει διάφερες λύσεις. Ενδεικτικά παραθέτουμε τρεις:

1	18	15	4	19
8	23	12	9	24
16	5	20	17	14
2	10	25	3	11
7	22	13	6	21

1	9	6	19	12
15	21	3	16	24
7	18	13	8	5
2	10	25	20	11
14	22	4	17	23

1	16	5	2	19
24	11	8	23	12
7	3	20	15	4
9	17	25	10	18
21	14	7	22	13

Η λύση τελικά, βρίσκεται αν για κάθε τεταρτημόριο χρησιμοποιήσουμε μία από τις παραπάνω λύσεις (5x5) και μεταπηδούμε στο επόμενο τεταρτημόριο, όπου θα συνεχίσουμε την τοποθέτηση με τον ίδιο τρόπο, αλλά βέβαια με την αρίθμηση να αφορά την επόμενη 25-άδα. Ομοίως και στα επόμενα 2 τεταρτημόρια. Βέβαια, υπάρχει και άλλος τρόπος επίλυσης, λαμβάνοντας υπόψη για τα κενά τετράγωνα το πλήθος των υπολοίπων κενών τετραγώνων με τα οποία μπορούν να έχουν “επικοινωνία”, αλλά δεν είναι θέμα της διπλωματικής αυτής.

Γενικά για την *τοποθέτηση των 25 αριθμών στο 5x5* τετράγωνο απαιτούνται από τον αλγόριθμο μόνο **21 προσπάθειες κατά μέσο όρο. Εδώ διακρίνεται το φαινόμενο της κατάρας της διάστασης καθώς για το αρχικό πρόβλημα των 100 τετραγώνων δεν βρέθηκε λύση ούτε σε 113.000.000 προσπάθειες από τον αλγόριθμο**.

Για τους σχηματισμούς 5x5, 6x6 και 7x7 μέσω του αλγορίθμου, καταμετρήσαμε το πλήθος των προσπαθειών έως ότου βρεθεί η πρώτη λύση, και από 1000 τέτοιες καταμετρήσεις, για κάθε σχηματισμό, έχουμε τα παρακάτω στοιχεία:

Σχηματισμός	min	max	mean	std
5x5	1	140	21,54	21,03
6x6	2	5.482	690,38	684,44
7x7	15	399.952	38.550,00	39.688,00

ΚΕΦΑΛΑΙΟ 6ο.
ΜΑΘΗΣΗ ΧΡΟΝΙΚΩΝ ΔΙΑΦΟΡΩΝ ΕΝΟΣ ΒΗΜΑΤΟΣ , TD(0)-Learning

6.1 *ΑΛΓΟΡΙΘΜΟΣ TD(0)-Learning*

Η Μάθηση Χρονικών Διαφορών (*Temporal Difference Learning*) χρησιμοποιεί αλγορίθμους που συνδυάζουν ιδέες και τεχνικές που χρησιμοποιούν οι αλγόριθμοι της *Monte Carlo* και του Δυναμικού Προγραμματισμού, και συγχρόνως την τεχνική της διαδοχικής ανανέωσης τιμών. Έτσι, συνδυάζει πλεονεκτήματα και του Δυναμικού Προγραμματισμού και της *Monte Carlo*. Θυμίζουμε πως για την *Monte Carlo* θέλουμε την πραγματοποίηση πλήθους επεισοδίων, παρατηρώντας αμοιβές και καταστάσεις, χωρίς απαραιτήτως την γνώση του μοντέλου του περιβάλλοντος, ώστε μετά το πέρας ενός αριθμού επεισοδίων να έχουμε ένα δείγμα των αθροιστικών αμοιβών για κάθε ζεύγος (s,α), και έτσι να μπορούμε να πάρουμε μία εκτίμηση της $Q(s,a)$ από τον αντίστοιχο δειγματικό μέσο. Αυτό προϋποθέτει προβλήματα πεπερασμένου χρόνου. Από την άλλη, ο Δυναμικός Προγραμματισμός προϋποθέτει πλήρη γνώση του μοντέλου του περιβάλλοντος, αλλά μπορεί να εφαρμοστεί σε προβλήματα και πεπερασμένου και απείρου χρονικού ορίζοντα. Ο Δυν. Προγρ. εμπλέκει την τιμή της συνάρτησης αξιολόγησης μίας κατάστασης με κάθε άλλη τιμή της συνάρτησης αξιολόγησης της αμέσως επόμενης κατάστασης (*Bootstrap*), και αυτό περιπλέκει τους υπολογισμούς του Δυναμικού Προγραμματισμού.

Η μέθοδος Χρονικών Διαφορών είναι συνδυασμός κατά έναν τρόπο του Δυναμικού Προγραμματισμού και της *Monte Carlo*, καθώς λαμβάνει ως παρατήρηση-δείγμα από ένα και μόνο βήμα t την άμεση αμοιβή r_t και βάσει αυτής ανανεώνει την τιμή της συνάρτησης αξιολόγησης για την κατάσταση που βρισκόταν (σταθμισμένος μέσος/*Monte Carlo*), στηριζόμενη και στις τιμές της συνάρτησης αξιολόγησης των καταστάσεων που μεταπηδά (*Bootstrap/Δυναμικός Προγραμματισμός*). Συγκεκριμένα παίρνει την διαφορά της τιμής της συνάρτησης αξιολόγησης στην κατάσταση που βρίσκεται *OLD* $V(s)$, έως εκείνη την στιγμή, από την τιμή της συνάρτησης αξιολόγησης της επόμενης κατάστασης s' ανηγμένης κατά κάποιον τρόπο στην κατάσταση στην οποία βρισκόταν $\left(r_t + \gamma \cdot V(s') \leftrightarrow TARGET\right)$. Χρησιμοποιεί έτσι και την παρατήρηση-δείγμα r_t, και τις αμέσως απόμενες εκτιμήσεις των καταστάσεων ως στόχο $TARGET$ για την ανανέωση των τιμών της συνάρτησης αξιολόγησης.

Συγκριτικά για την *Monte Carlo* θα είχαμε για *every visit* και σταθερή διαδοχική ανανέωση τιμών (με βήμα *α*), μετά από την ολοκλήρωση ενός επεισοδίου όπου παρατηρήθηκε αθροιστική αμοιβή R_t μετά την κατάσταση *s* στο βήμα *t* έχουμε:

$$\underset{NEW}{V(s)} \leftarrow \underset{OLD}{V(s)} + \alpha \cdot \underbrace{\left[\quad R_t \quad - V(s) \right]}_{TARGET - \ OLD}$$

Για την $\mathrm{T}D(0)$ έχουμε επίσης για *every visit* και σταθερή διαδοχική ανανέωση τιμών (με βήμα *α*), μετά από μόνο ένα βήμα και όχι μετά την ολοκλήρωση ενός επεισοδίου:

$$\underset{NEW}{V(s)} \leftarrow \underset{OLD}{V(s)} + \alpha \cdot \underbrace{\left[r_t + \gamma \cdot V(s') - V(s) \right]}_{TARGET - \ OLD}$$

Θυμίζουμε πως $\alpha \in [0,1]$ είναι το βήμα ανανέωσης τιμών, και γ ο εκπτωτικός παράγοντας.
Ο τελευταίος τύπος ανανέωσης τιμών μπορεί να γραφτεί διαφορετικά:

$$V(s) \underset{NEW}{\leftarrow} V(s) \underset{OLD}{+} \alpha \cdot \underbrace{\left[r_t + \gamma \cdot V(s') - V(s) \right]}_{TARGET - \ OLD}$$

$$V(s) \underset{NEW}{\leftarrow} V(s) \underset{OLD}{+} \alpha \cdot \underbrace{\left[r_t + \gamma \cdot V(s') \right]}_{TARGET} - \alpha \cdot V(s) \underset{OLD}{}$$

$$V(s) \underset{NEW}{\leftarrow} (1-a) \cdot V(s) \underset{OLD}{} + \alpha \cdot \underbrace{\left[r_t + \gamma \cdot V(s') \right]}_{TARGET}$$

Ο τύπος αυτός δείχνει ότι η νέα τιμή προκύπτει ως σταθμισμένος μέσος της παλιάς τιμής και του στόχου - $TARGET$ (που προκύπτει μέσω της νέας παρατήρησης) με βάρη (1-α) και α αντιστοίχως, που ορίζονται από το βήμα ανανέωσης. Όσο μικρότερο είναι το βήμα *α* τόσο λιγότερο επηρεάζεται η τιμή $V(s)$ από την νέα παρατήρηση, αλλά σε βάθος χρόνου συνυπολογίζονται όλες οι παρατηρήσεις.
Ο αλγόριθμος που υλοποιεί την παραπάνω μέθοδο ονομάζεται ***TD(0)*** και είναι :

TD(0) – *(every-visit / σταθερού βήματος ανανέωση)*

$\Gamma\iota\alpha\ \kappa\acute{\alpha}\theta\varepsilon\ s \in S, a \in A(s) \quad o\rho\acute{\iota}\zeta\varepsilon\tau\alpha\iota \quad V(s)\ (\alpha\upsilon\theta\alpha\acute{\iota}\rho\varepsilon\tau\alpha), \gamma\iota\alpha\ \pi o\lambda\iota\tau\iota\kappa\acute{\eta}\ \pi$

$\Gamma\iota\alpha\ \kappa\acute{\alpha}\theta\varepsilon\ \acute{\varepsilon}\nu\alpha\ \alpha\pi\acute{o}\ \ \mathrm{N}\ \pi\lambda\acute{\eta}\theta o\varsigma\ \varepsilon\pi\varepsilon\iota\sigma o\delta\acute{\iota}\omega\nu: \quad \alpha\rho\chi\iota\kappa\acute{\eta}\ \kappa\alpha\tau\acute{\alpha}\sigma\tau\alpha\sigma\eta\ s$

Repeat

$\alpha \leftarrow \pi(s) \quad (\alpha\pi\acute{o}\varphi\alpha\sigma\eta\ \ \alpha\pi\acute{o}\ \tau\eta\nu\ \pi o\lambda\iota\tau\iota\kappa\acute{\eta}\ \pi).$

$\pi\alpha\rho\alpha\tau\acute{\eta}\rho\eta\sigma\varepsilon\ \ r\ (\acute{\alpha}\mu\varepsilon\sigma\eta\ \ \alpha\mu o\iota\beta\acute{\eta})\ \kappa\alpha\iota\ \ s'\ (\varepsilon\pi\acute{o}\mu\varepsilon\nu\eta\ \kappa\alpha\tau\acute{\alpha}\sigma\tau\alpha\sigma\eta)$

$V(s) \leftarrow V(s) + \alpha \cdot \left[r + \gamma \cdot V(s') - V(s) \right]$

$s \leftarrow s'$

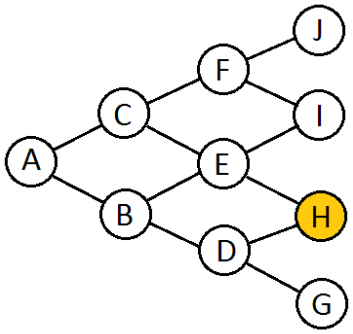
until $\ s = s_T\ (\tau\varepsilon\lambda\iota\kappa\acute{\eta}\ \kappa\alpha\tau\acute{\alpha}\sigma\tau\alpha\sigma\eta)$

Έχουμε έτσι δηλαδή, πως την ανανέωση των τιμών βασίζεται τόσο στις τιμές της επόμενης κατάστασης (όπως περίπου και στον Δυναμικό Προγραμματισμό), όσο και στην άμεση αμοιβή που παρατηρεί ο μαθητής (r_t) στο βήμα *t,* (όπως περίπου και στην *Monte Carlo*). Δηλαδή, **η** $\mathrm{T}D(0)$ **συνδυάζει τον Δυναμικό Προγραμματισμό και την *Monte Carlo* με τρόπο που είναι απαλλαγμένος από κάποια μειονεκτήματα αυτών.**

- Δεν χρειάζεται η $\mathrm{T}D(0)$ να έχει πεπερασμένο χρονικό ορίζοντα, ώστε να ανανεώνει τις τιμές στο τέλος κάθε επεισοδίου, αυτό το κάνει σε κάθε βήμα. Μπορεί να εφαρμοστεί και σε προβλήματα πεπερασμένου και σε άπειρου χρονικού ορίζοντα.
- Δεν χρειάζεται η $\mathrm{T}D(0)$ να γνωρίζει το μοντέλο του περιβάλλοντος, απλά παρατηρεί τις αμοιβές και την επόμενη κατάσταση.
- Δεν έχει πεπλεγμένους υπολογισμούς όπως ο Δυναμικός Προγραμματισμός. Απλά ανανεώνει σε κάθε βήμα τις τιμές με ελάχιστες πράξεις, παρ'όλα αυτά έχει την ιδιότητα *Bootstrap.*
- Μπορεί να χρησιμοποιεί εκπτωτικό παράγοντα όπως ο Δυναμικός Προγραμματισμός.
- Μπορεί να παράγει τα στοιχεία του προβλήματος με προσομοίωση, όπως η *Monte Carlo.*

Η σύγκλιση της ***TD(0)*** έχει αποδειχθεί από τον ***Sutton (1988)*** για τον μέσο, και από τον ***Dayan (1992)*** για σύγκλιση με πιθανότητα 1, βασισμένος στην δουλειά των ***Watkins & Dayan (1992)***.

Οι αλγόριθμοι *Monte Carlo* και *TD(0)* συγκλίνουν στην ίδια τελικά βέλτιστη λύση, αλλά έως την τελική ταύτισή τους, ως προς τα αποτελέσματα, υπάρχει μια μικρή διαφορά. Η *Monte Carlo* βρίσκει την εκτιμήτρια η οποία ελαχιστοποιεί το μέσο τετραγωνικό σφάλμα ***MSE***, ενώ η *TD(0)* βρίσκει την εκτιμήτρια που είναι σωστή για το μοντέλο Μέγιστης πιθανοφάνιας για την Μαρκοβιανή διαδικασία.



Εφαρμόσαμε και τους δύο αλγορίθμους στο πρόβλημα προς το χρυσάφι, όπως στο κεφάλαιο 3 (μόνο που εδώ χρησιμοποιήσαμε περισσότερες καταστάσεις όπως φαίνεται στο διπλανό σχήμα). Ο θησαυρός βρίσκεται στην θέση *H* και κάθε φορά μπορούμε να κινηθούμε προς τα δεξιά, επιλέγοντας κίνηση είτε προς τα πάνω είτε προς τα κάτω. Σκοπός είναι να βρεθεί η διαδρομή προς το θησαυρό. Με τυχαία μετακίνηση έχουμε πως ξεκινώντας από την θέση *A* έχουμε 1/8 πιθανότητα να καταλήξουμε στην *H*, από την *B* έχουμε 2/4 πιθανότητα να καταλήξουμε στην *H*, από την *C* έχουμε 1/4 πιθανότητα να καταλήξουμε στην *H*, από την *D* έχουμε 1/2 πιθανότητα να καταλήξουμε στην *H*, από την *E* έχουμε 1/2 πιθανότητα να καταλήξουμε στην *H*, από την *F* έχουμε 0/2 πιθανότητα να καταλήξουμε στην *H*, ενώ για τις *G,I* και *J* η πιθανότητα είναι 0. Ουσιαστικά μας ενδιαφέρει η εκτίμηση των πιθανοτήτων εύρεσης του θησαυρού αν ξεκινήσουμε από μία από τις 6 πρώτες καταστάσεις.

Στο ***παράρτημα 3*** έχουμε τους αντίστοιχους αλγόριθμους και από 20.000 προσπάθειες με πολιτική ισοπίθανη επιλογή μετακίνησης πάνω ή κάτω βρήκαμε, για τις εκτιμήσεις των πιθανοτήτων εύρεσης του θησαυρού ξεκινώντας από την κάθε κατάσταση, όπως και για την ρίζα του μέσου τετραγωνικού σφάλματος όπου φαίνεται πως είναι μικρότερο για την *Monte Carlo*:

Μέθοδος	A	B	C	D	E	F	MSE^0.5
Monte Carlo	0.3706	0.5006	0.2473	0.4996	0.5009	0	0.0022
TD(0)	0.3747	0.4967	0.2563	0.5195	0.5172	0	0.0110
ΘΕΩΡΗΤΙΚΕΣ ΠΙΘΑΝΟΤΗΤΕΣ	0.375	0.5	0.25	0.5	0.5	0	

Σημαντικοί αλγόριθμοι της μάθησης ***Χρονικών Διαφορών*** (***TD Learning***) οι οποίοι βασίζονται στην επόμενη και μόνο παρατήρηση - (στο επόμενο κεφάλαιο έχουμε και αλγορίθμους που βασίζονται σε περισσότερες παρατηρήσεις) - είναι :

- Ο αλγόριθμος ***Sarsa*** , ο οποίος είναι αλγόριθμος *on policy*,
- Ο αλγόριθμος ***Q*** , ο οποίος είναι αλγόριθμος *off policy*, και
- Ο αλγόριθμος ***R*** , ο οποίος είναι αλγόριθμος για προβλήματα απείρου ορίζοντα χωρίς εκπτωτικό παράγοντα.

6.2 *ΑΛΓΟΡΙΘΜΟΣ SARSA (TD ενός βήματος / on policy)*

Ο αλγόριθμος ***Sarsa*** είναι ένας τρόπος επίλυσης του προβλήματος της Ενισχυτικής Μάθησης. Προσπαθεί να βρει την βέλτιστη πολιτική μέσω της εκτίμησης των τιμών της συνάρτησης αξιολόγησης καταστάσεων αποφάσεων $Q(s,a)$. Είναι ένας αλγόριθμος *on policy*, δηλαδή βελτιώνει την πολιτική την οποία και ακολουθεί κατά την διαδικασία αλληλεπίδρασης με το περιβάλλον και μέσω της οποίας παράγονται οι άμεσες αμοιβές και οι επόμενες παρατηρήσεις.

Ο αλγόριθμος ***Sarsa*** παίρνει το όνομά του από την αλληλουχία καταστάσεων αμοιβής και αποφάσεων κατά την διαδικασία αλληλεπίδρασης με το περιβάλλον (ή την προσομοίωση που παράγει τα αντίστοιχα στοιχεία). Συγκεκριμένα, από την κατάσταση στην οποία βρίσκεται κάποια χρονική στιγμή, ***S**tate=s*, παίρνει απόφαση ***a**ction=α* παρατηρεί την άμεση αμοιβή ***r**eward=r* και την επόμενη κατάσταση ***s**tate=s'* και τελικά παίρνει και την απόφαση για την επόμενη κατάσταση ***a**ction=α'*. Έτσι, από τα αρχικά των ***S**tate **a**ction **r**eward **s**tate **a**ction*, παίρνει το όνομά του ***Sarsa***.

Ο ***Sarsa*** ανανεώνει τις τιμές $Q(s,a)$ μετά από κάθε βήμα, με στόχο $TARGET$ το άθροισμα της άμεσης αμοιβής r_t και της τιμής $Q(s',a')$ της επόμενης κατάστασης s' και απόφασης a', ανηγμένης ένα βήμα πριν με την βοήθεια του εκπτωτικού παράγοντα γ. Ο κανόνας ανανέωσης είναι λοιπόν:

$$\underbrace{Q(s,a)}_{NEW} \leftarrow \underbrace{Q(s,a)}_{OLD} + \alpha \cdot \underbrace{\left[r_t + \gamma \cdot Q(s',a') - Q(s,a) \right]}_{TARGET \;-\; OLD}$$

με τον στόχο να είναι $TARGET = r_t + \gamma \cdot Q(s',a')$

ενώ αν η s' είναι τελική κατάσταση λαμβάνουμε το $Q(s',a')$ ως μηδέν.

Τελικά, η βέλτιστη πολιτική βρίσκεται με τον παρακάτω αλγόριθμο, όπου ακολουθώντας μία πολιτική (Greedy-πολιτική ως προς την $Q(s,a)$) διορθώνει την πολιτική αυτή καθώς ανανεώνονται οι τιμές της $Q(s,a)$.

SARSA / TD ενός βήματος On-Policy *(ε-greedy policy)*

$\Gamma\iota\alpha\ \kappa\acute{\alpha}\theta\varepsilon\ s \in S, a \in A(s) \quad o\rho\acute{\iota}\zeta\varepsilon\tau\alpha\iota \quad Q(s,a) \leftarrow 0 \ \ (\alpha\upsilon\theta\alpha\acute{\iota}\rho\varepsilon\tau\alpha)$

$\pi(s) \leftarrow (\alpha\upsilon\theta\alpha\acute{\iota}\rho\varepsilon\tau\eta, \quad \varepsilon - greedy\ \pi o\lambda\iota\tau\iota\kappa\acute{\eta}\ \alpha\pi\acute{o}\ \tau\eta\nu\ Q)$

$Repeat\ \gamma\iota\alpha\ \kappa\acute{\alpha}\theta\varepsilon\ \varepsilon\pi\varepsilon\iota\sigma\acute{o}\delta\iota o:$

$\quad A\rho\chi\iota\kappa\acute{\eta}\ \kappa\alpha\tau\acute{\alpha}\sigma\tau\alpha\sigma\eta\ s\ (\pi\rho o\sigma o\mu o\acute{\iota}\omega\sigma\eta\ \acute{\eta}\ \pi\alpha\rho\alpha\tau\acute{\eta}\rho\eta\sigma\eta)$

$\quad \varepsilon\pi\iota\lambda o\gamma\acute{\eta}\ \alpha\pi\acute{o}\varphi\alpha\sigma\eta\varsigma\ a\ \gamma\iota\alpha\ \tau\eta\nu\ s\ (\mu\varepsilon\ \pi o\lambda\iota\tau\iota\kappa\acute{\eta}\ \varepsilon - greedy\ \alpha\pi\acute{o}\ \tau\eta\nu\ Q)$

$\quad Repeat\ (\gamma\iota\alpha\ \kappa\acute{\alpha}\theta\varepsilon\ \beta\acute{\eta}\mu\alpha\ \tau o\upsilon\ \varepsilon\pi\varepsilon\iota\sigma o\delta\acute{\iota}o\upsilon):$

$\quad\quad \lambda\acute{\eta}\psi\eta\ \alpha\pi\acute{o}\varphi\sigma\eta\varsigma\ a\ \kappa\alpha\iota\ \pi\alpha\rho\alpha\tau\acute{\eta}\rho\eta\sigma\eta\varsigma\ r, s'$

$\quad\quad \varepsilon\pi\iota\lambda o\gamma\acute{\eta}\ \alpha\pi\acute{o}\varphi\alpha\sigma\eta\varsigma\ a'\ \gamma\iota\alpha\ \tau\eta\nu\ s'\ (\mu\varepsilon\ \pi o\lambda\iota\tau\iota\kappa\acute{\eta}\ \varepsilon - greedy\ \alpha\pi\acute{o}\ \tau\eta\nu\ Q)$

$\quad\quad Q(s,a) \leftarrow Q(s,a) + \alpha \cdot \left[r + \gamma \cdot Q(s',a') - Q(s,a) \right]$

$\quad\quad s \leftarrow s'$

$\quad\quad a \leftarrow a'$

$\quad until\ (\eta\ \kappa\alpha\tau\acute{\alpha}\sigma\tau\alpha\sigma\eta\ s\ \nu\alpha\ \varepsilon\acute{\iota}\nu\alpha\iota\ \tau\varepsilon\lambda\iota\kappa\acute{\eta})$

$until \quad (\Sigma\upsilon\nu\theta\acute{\eta}\kappa\eta\ \sigma\acute{\upsilon}\gamma\kappa\lambda\iota\sigma\eta\varsigma)$

Για τον αλγόριθμο ***Sarsa*** ισχύει ότι συγκλίνει με πιθανότητα 1 σε βέλτιστη πολιτική και αντίστοιχη συνάρτηση αξιολόγησης καταστάσεων-αποφάσεων, αρκεί να επισκέπτονται τα ζεύγη καταστάσεων-αποφάσεων άπειρες φορές, και η πολιτική συγκλίνει οριακά στην *greedy πολιτική* (με την παράμετρο *ε* της *ε-greedy* πολιτικής να τείνει στο μηδέν)*, από την εργασία των Singh, Jaakkola, Littman και Szepesvari (2000) «Convergence Results for Single-Step On policy Reinforcement-Learning Algorithms»***.**

6.3 *ΑΛΓΟΡΙΘΜΟΣ Q (TD ενός βήματος / off policy)*

Ένας άλλος αλγόριθμος της ***TD*** μάθησης για την επίλυση του προβλήματος της Ενισχυτικής Μάθησης, είναι ο αλγόριθμος ***Q-Learning (Watkins 1989)***, ή πιο απλά αλγόριθμος ***Q***, ο οποίος είναι ένας αλγόριθμος ***off-policy***, δηλαδή εύρεση της βέλτιστης πολιτικής ***π**** χωρίς να ακολουθείται η πολιτική αυτή κατά την παραγωγή των επεισοδίων, αλλά μία άλλη πολιτική.

Η μορφή της ανανέωσης των τιμών της συνάρτησης Q για τον αλγόριθμο αυτόν για ένα βήμα (αντιστοίχως με τον αλγόριθμο ***TD(0)*** ενός βήματος), είναι :

$$Q(s_t,a_t) \leftarrow Q(s_t,a_t) + \alpha \cdot \left[r_t + \gamma \cdot \max_a Q(s_{t+1},a) - Q(s_t,a_t) \right]$$

Σε κάθε βήμα η πολιτική που ακολουθείται είναι υπεύθυνη για την αναγκαία επίσκεψη όλων των δυνατών ζευγαριών καταστάσεων-αποφάσεων και βέβαια δίνει την απόφαση a_t για την τωρινή κατάσταση s_t, ενώ ο αλγόριθμος υπολογίζει την ανανέωση των τιμών της συνάρτησης Q βάσει των καλύτερων αποφάσεων για την κατάσταση του επόμενου βήματος a_{t+1}. Η συνάρτηση Q τελικώς συγκλίνει με πιθανότητα 1 στην βέλτιστη συνάρτηση αξιολόγησης $Q*$ ανεξαρτήτως της πολιτικής που ακολουθείται (αρκεί να είναι τέτοια ώστε να “εγγυάται” την επίσκεψη των καταστάσεων-αποφάσεων όπως έχουμε προαναφέρει), από τις εργασίες των ***Watkins (1989)*** *και* ***Watkins & Dayan (1992)***.

Q-Learning / TD ενός βήματος Off-Policy *αλγόριθμος*

Για κάθε $s \in S, a \in A(s)$ *ορίζεται* $Q(s,a) \leftarrow 0$ *(αυθαίρετα)*

$\pi(s) \leftarrow$ *(αυθαίρετη ,* ε *– greedy πολιτική από την* Q*)*

Repeat για κάθε επεισόδιο:

  Αρχική κατάσταση s *(προσομοίωση ή παρατήρηση)*

  επιλογή απόφασης a *για την* s *(με πολιτική* ε *– greedy από την* Q*)*

  Repeat (για κάθε βήμα του επεισοδίου):

    λήψη απόφασης a *για την* s *(με πολιτική π.χ.* ε *– greedy από την* Q*)*

    παρατήρηση r, s'

    $Q(s,a) \leftarrow Q(s,a) + \alpha \cdot \left[r + \gamma \cdot \max_{a'} Q(s',a') - Q(s,a) \right]$

    $s \leftarrow s'$

  until (η κατάσταση s *να είναι τελική)*

until (Συνθήκη σύγκλισης)

Η συνθήκη σύγκλισης στον αλγόριθμο καθορίζεται έτσι ώστε, οι τελικές εκτιμήσεις να είναι αξιόπιστες, δηλαδή αμερόληπτες και με μικρή τυπική απόκλιση ώστε να είμαστε βέβαιοι πως έχει επέλθει σύγκλιση.

Όπως προαναφέραμε, διαφέρει από τον αλγόριθμο ***On policy TD(0)*** στο ότι για την επόμενη κατάσταση η επιλογή της απόφασης γίνεται ουσιαστικά με greedy πολιτική και όχι με μία *ε-greedy* πολιτική (ή *soft* πολιτική γενικότερα). Αυτό απλοποιεί πολύ τους υπολογισμούς και φυσικά η βέλτιστη πολιτική ορίζεται από την συνάρτηση Q που έχει προκύψει από τον αλγόριθμο, ως η πολιτική που λαμβάνει ως βέλτιστη απόφαση εκείνη την απόφαση που μεγιστοποιεί την $Q(s,\cdot)$:

$$\pi(s) = \arg\max_a \{Q(s,a)\}.$$

6.4 *ΑΛΓΟΡΙΘΜΟΣ R (TD, απείρου χρονικού ορίζοντα χωρίς εκπτωτικό παράγοντα)*

Στο πρόβλημα της ενισχυτικής μάθησης όπου δεν έχουμε τερματικές καταστάσεις ή πεπερασμένο γενικά χρονικό ορίζοντα, αλλά συγχρόνως δεν έχουμε και εκπτωτικό παράγοντα (*Undiscounted Continuing Task*), το ζητούμενο είναι να υπολογιστεί η μέγιστη μέση αμοιβή ανά χρονική μονάδα ρ^π. Η πολιτική που για την οποία θα προκύπτει η μέγιστη τιμή της ρ^π, θα ονομάζεται βέλτιστη πολιτική για αυτό το πρόβλημα.

$$\text{Ορίζεται} \quad \rho^\pi = \lim_{n\to\infty} \frac{1}{n} \sum_{t=1}^{n} E_\pi \left[r_t \right]$$

Ορίζεται η συνάρτηση αξιολόγησης σε συνάρτηση με την ρ^π και έστω πως η διαδικασία είναι εργοδική, ώστε να μην εξαρτάται η μέση αμοιβή ρ^π από την αρχική κατάσταση s_0 (κάθε κατάσταση μπορεί να είναι επισκέψιμη από κάθε άλλη υπό οποιαδήποτε πολιτική). Η συνάρτηση αξιολόγησης καταστάσεων όταν ακολουθείται συγκεκριμένη πολιτική *π* ορίζεται ως:

$$\tilde{V}^\pi(s) = \sum_{k=1}^{\infty} E_\pi \left[r_{t+k} - \rho^\pi \mid s_t = s \right].$$

Αντιστοίχως η συνάρτηση αξιολόγησης καταστάσεων-αποφάσεων όταν ακολουθείται συγκεκριμένη πολιτική *π* ορίζεται ως:

$$\tilde{Q}^\pi(s,a) = \sum_{k=1}^{\infty} E_\pi \left[r_{t+k} - \rho^\pi \mid s_t = s, a_t = a \right].$$

Αντιστοίχως με τον αλγόριθμο Q, που είναι *off policy* αλγόριθμος, έτσι και ο αλγόριθμος ***R*** χρησιμοποιεί μία πολιτική $\boldsymbol{\pi_g}$, την οποία ακολουθεί κατά την διεξαγωγή του προβλήματος μάθησης (*Behavior policy*) και η οποία θα μπορεί να είναι μία *ε-soft* πολιτική, ώστε να είναι δυνατός να επιλέξουμε σε βάθος χρόνου όλες τις δυνατές αποφάσεις, αλλά και μία πολιτική $\boldsymbol{\pi}$ (*Estimation policy*) για την οποία εκτιμά την συνάρτηση αξιολόγησης. Η πολιτική αυτή συγκλίνει τελικώς στην βέλτιστη πολιτική.

Για τον αλγόριθμο ***R*** έχουμε δύο παραμέτρους βηματικής ανανέωσης, έναν για την ανανέωση των τιμών της συνάρτησης αξιολόγησης Q, και έναν για την ανανέωση των τιμών της μέσης αμοιβής ανά χρονική μονάδα $\rho^{\pi} = \rho$ για τον αλγόριθμο.

R-Learning / TD Off-Policy *αλγόριθμος*

$\Gamma\iota\alpha\ \kappa\acute{\alpha}\theta\varepsilon\ s \in S, a \in A(s) \quad o\rho\acute{\iota}\zeta\varepsilon\tau\alpha\iota \quad \rho \quad Q(s,a) \quad (\alpha\upsilon\theta\alpha\acute{\iota}\rho\varepsilon\tau\alpha)$

$\kappa\alpha\iota \quad \tau\upsilon\chi\alpha\acute{\iota}\alpha \quad s \quad \kappa\alpha\iota \quad \alpha$

Repeat

$s\ (\pi\rho o\sigma o\mu o\acute{\iota}\omega\sigma\eta\ \acute{\eta}\ \pi\alpha\rho\alpha\tau\acute{\eta}\rho\eta\sigma\eta)$

$\varepsilon\pi\iota\lambda o\gamma\acute{\eta}\ \alpha\pi\acute{o}\varphi\alpha\sigma\eta\varsigma\ a\ \gamma\iota\alpha\ \tau\eta\nu\ s\ (\mu\varepsilon\ \pi o\lambda\iota\tau\iota\kappa\acute{\eta}\ \varepsilon - greedy\ \alpha\pi\acute{o}\ \tau\eta\nu\ Q)$

$\lambda\acute{\eta}\psi\eta\ \alpha\pi\acute{o}\varphi\alpha\sigma\eta\varsigma\ a\ ,\ \pi\alpha\rho\alpha\tau\acute{\eta}\rho\eta\sigma\eta \quad r, s'$

$$Q(s,a) \leftarrow Q(s,a) + \alpha \cdot \left[r - \rho + \max_{a'} Q(s',a') - Q(s,a) \right]$$

$$\text{A}\nu \quad Q(s,a) = \max_{a} Q(s,a)$$

$$\tau\acute{o}\tau\varepsilon \quad \rho \leftarrow \rho + \beta \left[r - \rho + \max_{a'} Q(s',a') - \max_{a} Q(s,a) \right]$$

$$s \leftarrow s'$$

until $(\Sigma\upsilon\nu\theta\acute{\eta}\kappa\eta\ \sigma\acute{\upsilon}\gamma\kappa\lambda\iota\sigma\eta\varsigma)$

Για τον αλγόριθμο ***R*** δεν υπάρχει απόδειξη σύγκλισης, οπότε απλά αναφέρεται στην διπλωματική αυτή, και τίθεται προς έλεγχο και διασταύρωση αποτελεσμάτων.

ΚΕΦΑΛΑΙΟ 7ο.
ΜΑΘΗΣΗ ΧΡΟΝΙΚΩΝ ΔΙΑΦΟΡΩΝ, TD(λ)-Learning - ELIGIBILITY TRACES

7.1 *ΧΡΟΝΙΚΕΣ ΔΙΑΦΟΡΕΣ n ΒΗΜΑΤΩΝ, n-step TD*

Οι μέθοδοι Χρονικών Διαφορών με την παράμετρο ***λ***, ***TD(λ)*** είναι συνδυασμός της ***TD(0)*** και της ***Monte Carlo***. Κάθε αλγόριθμος ***TD(λ)*** χρησιμοποιεί με διαφορετικά βάρη τις άμεσες αμοιβές που ακολουθούν μία απόφαση για να εκτιμήσει τις τιμές της συνάρτησης αξιολόγησης. Η διαφορά των δύο μεθόδων είναι πως η ***Monte Carlo*** περιμένει έως το τέλος ενός ολόκληρου επεισοδίου, όπου πλέον γνωρίζει όλες τις άμεσες αμοιβές που έχουν ακολουθήσει τις καταστάσεις που προσπελάστηκαν στο επεισόδιο, και βάσει αυτών ως παρατηρήσεις, μπορεί να εκτιμήσει (ή και να ανανεώνει) τις τιμές της συνάρτησης αξιολόγησης. Ενώ, η ***TD(0)*** παίρνει, ως παρατήρηση, μόνο την άμεση αμοιβή που ακολουθεί μία κατάσταση και βάσει αυτής (αθροίζοντας και την τιμή της συνάρτησης αξιολόγησης της επόμενης κατάστασης) διορθώνει την τιμή της συνάρτησης αξιολόγησης της κατάστασης που προηγήθηκε.

Συγκεκριμένα, για την *Monte Carlo*, ο κανόνας ανανέωσης τιμών με σταθερή διαδοχική ανανέωση τιμών (με βήμα *α*), με εκπτωτικό παράγοντα $\gamma \in (0,1]$, μετά από την ολοκλήρωση σε T χρονικά βήματα ενός επεισοδίου, και αν για πρώτη φορά στο επεισόδιο που προσπελάσθηκε η κατάσταση *s* ήταν την χρονική στιγμή *t*, θα είναι :

$$\underset{NEW}{V(s)} \leftarrow \underset{OLD}{V(s)} + \alpha \cdot \underbrace{\left[\quad R_t \quad - V(s) \right]}_{TARGET - \quad OLD}$$

Όπου $TARGET = R_t = r_t + \gamma \cdot r_{t+1} + \gamma^2 \cdot r_{t+2} + ... + \gamma^{T-t-2} \cdot r_{T-1} + \gamma^{T-t-1} \cdot r_T$

Αντιστοίχως, η ***TD(0)*** λαμβάνει ως παρατήρηση-δείγμα την άμεση αμοιβή r_t του βήματος t και μόνο αυτού, και βάσει αυτής (αθροίζοντας και την τιμή της συνάρτησης αξιολόγησης της επόμενης κατάστασης), ανανεώνει την τιμή της συνάρτησης αξιολόγησης για την κατάσταση που βρισκόταν:

$$\underset{NEW}{V(s)} \leftarrow \underset{OLD}{V(s)} + \alpha \cdot \underbrace{\left[r_t + \gamma \cdot V(s') - V(s) \right]}_{TARGET - \quad OLD},$$

όπου ο στόχος για ένα βήμα συμβολίζεται $TARGET = R_t^{(1)} = r_t + \gamma \cdot V(s_{t+1} = s')$.

Ένας συνδυασμός των παραπάνω μεθόδων γίνεται αν πάρουμε περισσότερες από μία παρατηρήσεις (*n* το πλήθος), τις *n* κατά σειρά επόμενες άμεσες αμοιβές μετά από μία κατάσταση, αλλά και λιγότερες από όσες υπολείπονται έως το τέλος του επεισοδίου, ώστε να χρησιμοποιηθούν ως στόχος (*TARGET*). Εάν παρατηρήσουμε, λοιπόν, $n \le T - t$ άμεσες αμοιβές μετά από μία κατάσταση $s_t = s$, μπορούμε να ανανεώσουμε τις τιμές της συνάρτησης αξιολόγησης συμβολίζοντας ως στόχο το άθροισμα για n βήματα συν την έως τότε εκτίμηση της συνάρτησης αξιολόγησης της κατάστασης *n* βήματα μετά (s_{t+n}):

$$TARGET = R_t^{(n)} = r_t + \gamma \cdot r_{t+1} + \gamma^2 \cdot r_{t+2} + ... + \gamma^{n-1} \cdot r_{t+n-1} + \gamma^n \cdot V\left(s_{t+n}\right), \quad n \le T - t$$

όπου βέβαια για κάθε $n > T-t$ ο στόχος ταυτίζεται με τον στόχο της *Monte Carlo* :

$$TARGET = R_t^{(n)} = R_t = r_t + \gamma \cdot r_{t+1} + \gamma^2 \cdot r_{t+2} + ... + \gamma^{n-1} \cdot r_{t+n-1} + \gamma^n \cdot r_T, \quad n > T-t$$

Για $n = 1,2,...,T-t$ ο αντίστοιχος κανόνας ανανέωσης για τον αλγόριθμο χρονικών διαφορών ***n-step*** TD θα είναι :

$$\underset{NEW}{V(s)} \leftarrow \underset{OLD}{V(s)} + \alpha \cdot \underbrace{\left[\quad R_t^{(n)} - V(s) \right]}_{TARGET - \ OLD}.$$

Ο κανόνας αυτός μπορεί να ανανεώνει τις τιμές της συνάρτησης αξιολόγησης ***on-line*** (όποτε δηλαδή κατά την διάρκεια ενός επεισοδίου έχουν καταγραφεί οι *n* επόμενες αμοιβές για κάποια κατάσταση), ή ***off-line***, δηλαδή οι ανανεώσεις να γίνονται για όλες τις καταστάσεις μαζί, μετά το πέρας του επεισοδίου.

7.2 ΑΛΓΟΡΙΘΜΟΣ TD(λ) "Forward View"

Ένας ακόμα τρόπος συνδυασμού των παραπάνω ***n-step TD*** αλγορίθμων, ώστε να γίνει κάποιος αλγόριθμος μεταξύ του ***TD(0)*** και της ***Monte Carlo***, είναι για κάθε ***n-step TD*** αλγόριθμο να δοθεί κάποιο βάρος και να βρεθεί ένας σταθμισμένος μέσος αυτών, ως στόχος για την ανανέωση των τιμών της συνάρτησης αξιολόγησης. Ο στόχος αυτός ουσιαστικά δίνει διαφορετικό βάρος σε κάθε αθροιστική αμοιβή για τα επόμενα *n* βήματα, με την αιτιολογία πως η πρώτη άμεση αμοιβή οφείλεται (με το αντίστοιχο βάρος) στην κατάσταση και απόφαση που είχαμε στο βήμα *t*, το άθροισμα της πρώτης και δεύτερης επόμενης άμεσης αμοιβής, οφείλεται (με διαφορετικό βάρος μικρότερο του πρώτου, ώστε να φθίνει κατά ***(1-λ)·100%*** , $\lambda \in [0,1]$) στην κατάσταση και απόφαση που είχαμε στο βήμα *t*, και ούτω καθεξής.

Αναλυτικότερα, έστω πως μετά την χρονική στιγμή t έχουμε τις παρατηρήσεις έως το τέλος το επεισοδίου: $s_t, r_t, s_{t+1}, r_{t+1}, s_{t+2}, r_{t+2}, ..., s_{T-1}, r_{T-1}, s_T, r_T$. Τότε μπορούμε να ορίσουμε για όλους τους δυνατούς στόχους *n* βημάτων $R_t^{(1)}, R_t^{(2)}, R_t^{(3)}, R_t^{(4)}, ..., R_t^{(T-t-1)}, R_t^{(T-t)}$, αντιστοίχως τα βάρη $(1-\lambda), (1-\lambda)\lambda, (1-\lambda)\lambda^2, (1-\lambda)\lambda^3, ..., (1-\lambda)\lambda^{T-t-2}, \lambda^{T-t-1}$ όπου $\lambda \in [0,1]$ τα οποία αθροίζονται στην μονάδα. Έτσι, σημειώνοντας πως στα παρακάτω, ως συμβολισμό θα έχουμε το $\lambda^0 = 1$ για $\lambda = 0$, μπορούμε να έχουμε ως στόχο τον αντίστοιχο μέσο R_t^λ :

$$R_t^\lambda = \sum_{n=1}^{\infty} (1-\lambda)\lambda^{n-1} \cdot R_t^{(n)} = (1-\lambda) \cdot \sum_{n=1}^{\infty} \lambda^{n-1} \cdot R_t^{(n)}, \quad \lambda \in [0,1],$$

Η οποία μπορεί να γραφτεί και ως

$$\begin{aligned} R_t^\lambda &= (1-\lambda) \cdot \sum_{n=1}^{T-t-1} \lambda^{n-1} \cdot R_t^{(n)} + (1-\lambda) \cdot \sum_{n=T-t}^{\infty} \lambda^{n-1} \cdot R_t^{(n)} = \\ &= (1-\lambda) \cdot \sum_{n=1}^{T-t-1} \lambda^{n-1} \cdot R_t^{(n)} + (1-\lambda) \cdot R_t \sum_{n=T-t}^{\infty} \lambda^{n-1} = \\ &= (1-\lambda) \cdot \sum_{n=1}^{T-t-1} \lambda^{n-1} \cdot R_t^{(n)} + \lambda^{T-t-1} \cdot R_t \quad , \end{aligned}$$

δηλαδή $R_t^\lambda = (1-\lambda) \cdot \sum_{n=1}^{T-t-1} \lambda^{n-1} \cdot R_t^{(n)} + \lambda^{T-t-1} \cdot R_t \ , \quad \lambda \in [0,1].$

Ομοίως με τον ***n-step TD*** αλγόριθμο, και εδώ ο αλγόριθμος δεν είναι εύκολα υλοποιήσιμος καθώς απαιτεί πολύ περισσότερη υπολογιστική ισχύ και μνήμη, συν το γεγονός πως θα πρέπει να τελειώσει το κάθε επεισόδιο ώστε να γίνουν στο τέλος οι ανανεώσεις των τιμών (***off-line ανανέωση)***.

Ο κανόνας ανανέωσης θα είναι και εδώ παρόμοιος με τα προηγούμενα:

$$\underbrace{V_{t+1}(s)}_{NEW} \leftarrow \underbrace{V_t(s)}_{OLD} + \alpha \cdot \underbrace{\left[\; R_t^{\lambda} - V_t(s) \right]}_{TARGET - \; OLD}$$

για κάθε κατάσταση $s_t = s$.

Ο αλγόριθμος λέγεται και "*Forward view*" γιατί για κάθε ανανέωση πρέπει να γνωρίζουμε όλες τις επόμενες αμοιβές έως το τέλος του επεισοδίου.

7.3 ΑΛΓΟΡΙΘΜΟΣ TD(λ) "Backward View"

Μία διαφορετική ανάλυση του κανόνα ανανέωσης τιμών για τον αλγόριθμο TD(λ) "*Forward View*" μπορεί να δώσει έναν πιο εύκολα υλοποιήσιμο αλγόριθμο. Όπως έχουμε πει για κάθε κατάσταση $s_t = s$ χρησιμοποιείται ως στόχος ο σταθμισμένος μέσος των αθροιστικών αμοιβών *n* βημάτων για *n*=1,2,… έως το τέλος του επεισοδίου:

$$TARGET = R_t^{\lambda} = (1-\lambda) \cdot \sum_{n=1}^{T-t-1} \lambda^{n-1} \cdot R_t^{(n)} + \lambda^{T-t-1} \cdot R_t \;, \quad \lambda \in [0,1]$$

Όμως για τα διάφορα *n*, στους όρους $R_t^{(n)}$, υπάρχουν οι άμεσες αμοιβές r_{t+n-1} και συγκεκριμένα η r_t υπάρχει σε όλους τους όρους του R_t^{λ}, ο r_{t+1} σε όλους εκτός του $R_t^{(1)}$ πρώτου κ.τ.λ. Αναλυτικά έχουμε:

$$\begin{aligned}
&R_t^{\lambda} - V(s_t) = (1-\lambda) \cdot \sum_{n=1}^{T-t-1} \lambda^{n-1} \cdot R_t^{(n)} + \lambda^{T-t-1} \cdot R_t - V(s_t), \quad \lambda \in [0,1] \\
&= (1-\lambda) \cdot \left[R_t^{(1)} \right] + (1-\lambda) \cdot \lambda \cdot \left[R_t^{(2)} \right] + \ldots + (1-\lambda) \cdot \lambda^{T-t-2} \cdot \left[R_t^{(T-t-1)} \right] + \lambda^{T-t-1} \cdot R_t - V(s_t) = \\
&= \quad (1-\lambda) \cdot \left[r_t + \gamma \cdot V(s_{t+1}) \right] \quad - V(s_t) + \\
&+ \quad (1-\lambda) \cdot \lambda \cdot \left[r_t + \gamma \cdot r_{t+1} + \gamma^2 \cdot V(s_{t+2}) \right] + \\
&+ \quad (1-\lambda) \cdot \lambda^2 \cdot \left[r_t + \gamma \cdot r_{t+1} + \gamma^2 \cdot r_{t+2} + \gamma^3 \cdot V(s_{t+3}) \right] + \\
&+ \quad (1-\lambda) \cdot \lambda^3 \cdot \left[r_t + \gamma \cdot r_{t+1} + \gamma^2 \cdot r_{t+2} + \gamma^3 \cdot r_{t+3} + \gamma^4 \cdot V(s_{t+4}) \right] + \\
&\quad \vdots \qquad \vdots \qquad \vdots \qquad \vdots \qquad \vdots \\
&+ (1-\lambda) \cdot \lambda^{T-t-2} \cdot \left[r_t + \gamma \cdot r_{t+1} + \gamma^2 \cdot r_{t+2} + \gamma^3 \cdot r_{t+3} + \cdots + \gamma^{T-t-2} \cdot r_{T-2} + \gamma^{T-t-1} \cdot V(s_{T-1}) \right] + \\
&\quad + \lambda^{T-t-1} \cdot \left[r_t + \gamma \cdot r_{t+1} + \gamma^2 \cdot r_{t+2} + \gamma^3 \cdot r_{t+3} + \cdots + \gamma^{T-t-2} \cdot r_{T-2} + \gamma^{T-t-1} \cdot r_{T-1} + \gamma^{T-t} \cdot V(s_T) \right] \;,
\end{aligned}$$

όπου η $V(s_T) = r_T$ είναι η τελική αμοιβή του επεισοδίου.

Η τελευταία παράσταση κάνοντας αναγωγή στους όρους των άμεσων αμοιβών γράφεται:

$$
\begin{aligned}
&R_t^\lambda - V(s_t) = \\
&= \Big[(1-\lambda) + \cancel{(1-\lambda)\lambda} + \cancel{(1-\lambda)\lambda^2} + \ldots + \cancel{(1-\lambda)\lambda^{T-t-2}} + \cancel{\lambda^{T-t-1}} \Big] \cdot r_t + (1-\lambda)\gamma V(s_{t+1}) - V(s_t) + \\
&\quad + \gamma \cdot \Big[(1-\lambda)\lambda + (1-\lambda)\lambda^2 + (1-\lambda)\lambda^3 + \ldots + (1-\lambda)\lambda^{T-t-2} + \lambda^{T-t-1} \Big] \cdot r_{t+1} + (1-\lambda)\lambda\gamma^2 V(s_{t+2}) + \\
&\quad + \gamma^2 \cdot \Big[\qquad (1-\lambda)\lambda^2 + (1-\lambda)\lambda^3 + \ldots + (1-\lambda)\lambda^{T-t-2} + \lambda^{T-t-1} \Big] \cdot r_{t+2} + (1-\lambda)\lambda^2\gamma^3 V(s_{t+3}) + \\
&\quad + \gamma^3 \cdot \Big[\qquad\qquad + (1-\lambda)\lambda^3 + \ldots + (1-\lambda)\lambda^{T-t-2} + \lambda^{T-t-1} \Big] \cdot r_{t+3} + (1-\lambda)\lambda^3\gamma^4 V(s_{t+4}) + \\
&\quad \vdots \qquad\quad \vdots \qquad \vdots \qquad \vdots \qquad \vdots \qquad \vdots \qquad \vdots \qquad \vdots \qquad \vdots \\
&+\gamma^{T-t-2} \cdot \Big[\qquad\qquad\qquad\qquad (1-\lambda)\lambda^{T-t-2} + \lambda^{T-t-1} \Big] \cdot r_{T-2} + (1-\lambda)\lambda^{T-t-2}\gamma^{T-t-1} V(s_{T-1}) + \\
&+\gamma^{T-t-1} \cdot \Big[\qquad\qquad\qquad\qquad\qquad\qquad \lambda^{T-t-1} \Big] \cdot r_{T-1} \qquad + \qquad \lambda^{T-t-1}\gamma^{T-t} V(s_T)
\end{aligned}
$$

όπου μετά από τις απλοποιήσεις έχουμε :

$$
\begin{aligned}
R_t^\lambda - V(s_t) &= 1 \cdot r_t + (1-\lambda)\gamma V(s_{t+1}) - V(s_t) + \\
&\quad + \gamma\lambda \cdot r_{t+1} + (1-\lambda)\lambda\gamma^2 V(s_{t+2}) + \\
&\quad + \gamma^2\lambda^2 \cdot r_{t+2} + (1-\lambda)\lambda^2\gamma^3 V(s_{t+3}) + \\
&\quad + \gamma^3\lambda^3 \cdot r_{t+3} + (1-\lambda)\lambda^3\gamma^4 V(s_{t+4}) + \\
&\quad \vdots \qquad\quad \vdots \qquad\quad \vdots \qquad\quad \vdots \\
&+\gamma^{T-t-2}\lambda^{T-t-2} \cdot r_{T-2} + (1-\lambda)\lambda^{T-t-2}\gamma^{T-t-1} V(s_{T-1}) + \\
&+\gamma^{T-t-1}\lambda^{T-t-1} \cdot r_{T-1} \qquad + \qquad \lambda^{T-t-1}\gamma^{T-t} V(s_T)
\end{aligned}
$$

Από αυτήν την μορφή φαίνεται πώς αν $\lambda = 0$ έχουμε τον αλγόριθμο ***TD(0)***, ενώ αν $\lambda = 1$ προκύπτει ο αλγόριθμος ***TD(1)*** που είναι η ***Monte Carlo*** για πεπερασμένο πρόβλημα. Αναδιατάσσοντας κατάλληλα τις τιμές τις συνάρτησης αξιολόγησης $V(s_{\ldots})$ έχουμε:

$$
\begin{aligned}
R_t^\lambda - V(s_t) &= 1 \cdot r_t + \gamma V(s_{t+1}) - V(s_t) + \\
&\quad + \gamma\lambda \cdot r_{t+1} + \lambda\gamma^2 V(s_{t+2}) - \lambda\gamma V(s_{t+1}) + \\
&\quad + \gamma^2\lambda^2 \cdot r_{t+2} + \lambda^2\gamma^3 V(s_{t+3}) - \lambda^2\gamma^2 V(s_{t+2}) + \\
&\quad + \gamma^3\lambda^3 \cdot r_{t+3} + \lambda^3\gamma^4 V(s_{t+4}) - \lambda^3\gamma^3 V(s_{t+3}) + \\
&\quad \vdots \qquad\quad \vdots \qquad\quad \vdots \qquad\quad \vdots \\
&+\gamma^{T-t-2}\lambda^{T-t-2} r_{T-2} + \lambda^{T-t-2}\gamma^{T-t-1} V(s_{T-1}) - \lambda^{T-t-2}\gamma^{T-t-2} V(s_{T-2}) + \\
&+\gamma^{T-t-1}\lambda^{T-t-1} \cdot r_{T-1} + \lambda^{T-t-1}\gamma^{T-t} V(s_T) - \lambda^{T-t-1}\gamma^{T-t-1} V(s_{T-1}) \Leftrightarrow \\
R_t^\lambda - V(s_t) &= \qquad \Big[r_t + \gamma V(s_{t+1}) - V(s_t) \Big] + \\
&\quad + \gamma\lambda \cdot \Big[r_{t+1} + \gamma V(s_{t+2}) - V(s_{t+1}) \Big] + \\
&\quad + (\gamma\lambda)^2 \cdot \Big[r_{t+2} + \gamma V(s_{t+3}) - V(s_{t+2}) \Big] + \\
&\quad + (\gamma\lambda)^3 \cdot \Big[r_{t+3} + \gamma V(s_{t+4}) - V(s_{t+3}) \Big] + \\
&\quad \vdots \qquad\quad \vdots \qquad\quad \vdots \qquad\quad \vdots \\
&\quad + (\gamma\lambda)^{T-t-2} \cdot \Big[r_{T-2} + \gamma V(s_{T-1}) - V(s_{T-2}) \Big] + \\
&\quad + (\gamma\lambda)^{T-t-1} \cdot \Big[r_{T-1} + \gamma V(s_T) - V(s_{T-1}) \Big]
\end{aligned}
$$

όπου γίνεται σαφές πως η διαφορά $R_t^{\lambda} - V(s_t)$ στον κανόνα ανανέωσης του αλγορίθμου TD(λ) "*Forward View*", χρησιμοποιεί τις διαφορές $\left[r_{t+i} + \gamma V(s_{t+i+1}) - V(s_{t+i})\right]$ που θα χρησιμοποιούσε ο αλγόριθμος ***TD(0)*** για την κατάσταση s_{t+i}, i βήματα μπροστά, ανατιμημένες με τον αποπληθωριστικό παράγοντα γ και τον παράγοντα λ. Ο παράγοντας λ θεωρείται παράγοντας μνήμης, με την έννοια πως όσο μεταγενέστερη είναι μία αμοιβή από μία κατάσταση, τόσο λιγότερο θα πρέπει να οφείλεται αυτή η αμοιβή στην συγκεκριμένη κατάσταση. Με ένα άλλο σκεπτικό, επειδή η διαδοχή των καταστάσεων είναι αυτή που οδηγεί στην αντίστοιχη διαδοχή των αμοιβών, η διόρθωση θα πρέπει να γίνεται σε όλες τις καταστάσεις που έχουν προηγηθεί, και μάλιστα σε μικρότερο βαθμό για κάθε κατάσταση αναλόγως της χρονικής διαφοράς μεταξύ αμοιβής και κατάστασης. Δηλαδή, μία αμοιβή θα παίξει ρόλο στην διόρθωσης της τιμής μίας κατάστασης i βήματα πίσω, ανατιμημένη (ως αξία λόγω πληθωρισμού) με τον παράγοντα γ^i και "ανατιμημένη" (από άποψη αιτίου-αποτελέσματος) με τον παράγοντα μνήμης λ^i.

Επομένως, κάθε διαφορά $\left[r_t + \gamma V(s_{t+1}) - V(s_t)\right]$ που θα χρησιμοποιούσε ο αλγόριθμος ***TD(0)*** για την κατάσταση s_t, θα πρέπει να πολλαπλασιάζεται με τον παράγοντα ανατίμησης $(\gamma\lambda)^i$ (και φυσικά με τον βηματικό παράγοντα μάθησης *α*), ώστε να εφαρμοστεί ως διόρθωση σε κάθε κατάσταση i βήματα πίσω. Αυτή η θεώρηση δίνει το χαρακτηριστικό όνομα ***"Backward View"*** στον αλγόριθμο ***TD(λ)***, καθώς **αναδρομικά** μπορούμε σε κάθε βήμα να εφαρμόζουμε τις διορθώσεις που αντιστοιχούν σε προηγούμενες καταστάσεις και οι οποίες βασίζονται στην αμοιβή του συγκεκριμένου βήματος, προσαρμοσμένες με τον παράγοντα $(\gamma\lambda)^v$. Αν σε κάθε βήμα αντιστοιχίσουμε για όλες τις προηγούμενες καταστάσεις του επεισοδίου, έναν συντελεστή "ανατίμησης" $(\gamma\lambda)^v$, τότε στο επόμενο βήμα για τις ίδιες προηγούμενες καταστάσεις θα πρέπει αυτός ο συντελεστής να γίνει $(\gamma\lambda)\cdot(\gamma\lambda)^v = (\gamma\lambda)^{v+1}$, καθώς στην διαδοχή των καταστάσεων του επεισοδίου, αυτές οι καταστάσεις θα είναι κατά ένα βήμα "παλαιότερες", ενώ για την κατάσταση του νέου βήματος ο αντίστοιχος συντελεστής θα πρέπει να είναι $(\gamma\lambda)^0 = 1$. Τέλος, αν η κατάσταση του νέου βήματος έχει συναντηθεί σε προηγούμενο βήμα του επεισοδίου, θα πρέπει να έχουμε αντίστοιχο συντελεστή και για τις δύο επισκέψεις ή έναν συντελεστή ως άθροισμα αυτών, καθώς θα αντιστοιχούν δύο διορθώσεις με δύο αντίστοιχους συντελεστές "ανατίμησης", δηλαδή $(\gamma\lambda)\cdot(\gamma\lambda)^v + 1 = (\gamma\lambda)^{v+1} + 1$. Οι αναδρομικές αυτές διορθώσεις είναι εύκολα υλοποιήσιμες αν σε κάθε βήμα καταγράφουμε και ανανεώνουμε τους συντελεστές "ανατίμησης" που αναφέραμε, τους οποίους μπορούμε να ονομάζουμε ***<u>ίχνη προσπέλασης</u>*** (βιβλιογραφία ***<u>eligibility traces</u>***). Οπότε συμβολίζουμε αντιστοίχως $e_t(s)$ τα ίχνη προσπέλασης για την κατάσταση s την χρονική στιγμή t και ο κανόνας ανανέωσης αυτών θα είναι:

$$e_t(s) = (\gamma\lambda)\cdot e_{t-1}(s) + I_{ss_t}$$
$$\text{όπου} \quad I_{ss_t} \ \text{δείκτρια με} \quad I_{ss_t} = 1, \ \text{αν } s_t = s$$

Έχουμε, λοιπόν, τον κανόνα ανανέωσης για τον αλγόριθμο ***TD(λ) "Backward View"*** με την βοήθεια των ιχνών προσπέλασης $e_t(s)$ να είναι:

$$V(s) \leftarrow V(s) + \alpha \cdot \left[r_t + \gamma V(s_{t+1}) - V(s_t) \right] \cdot e_t(s),$$

όπου η ανανέωση γίνεται για κάθε κατάσταση που έχει προσπελαστεί από την αρχή του επεισοδίου και έως το βήμα t (κατάσταση s_t) του επεισοδίου. Μέσω των ανανεώσεων αυτών, αθροίζονται οι αντίστοιχες διορθώσεις σε κάθε κατάσταση και στο τέλος του επεισοδίου όλες οι ανανεώσεις θα έχουν πραγματοποιηθεί για όλες τις καταστάσεις.

Θα πρέπει να σημειώσουμε εδώ πως ο αλγόριθμος ***TD(1)*** είναι ο αλγόριθμος *Monte Carlo* για πρόβλημα πεπερασμένου χρονικού ορίζοντα Αλλά ο ***TD(1)*** αλγόριθμος εφαρμόζεται εξίσου εύκολα και σε ***προβλήματα απείρου χρονικού*** ορίζοντα καθώς δεν χρειάζεται τις επόμενες αμοιβές για να εκτιμήσει τις τιμές της συνάρτησης αξιολόγησης, όπως απαιτεί η *Monte Carlo*

Ο αλγόριθμος ***TD(λ) "Backward View"*** είναι :

TD(λ) "Backward View" / *TD On-line αλγόριθμος με eligibility traces*

Για κάθε $s \in S$ ορίζεται $V(s)$ (αυθαίρετα)
Repeat για κάθε επεισόδιο:
  Για κάθε $s \in S$ ορίζεται $e(s) = 0$
  αρχικό s (προσομοίωση ή παρατήρηση)
  Repeat (για κάθε βήμα του επεισοδίου):
    λήψη απόφασης a για την s (με πολιτική π)
    παρατήρηση των r και s'
    $D\_TARGET \leftarrow r + \gamma V(s') - V(s)$
    $e(s) \leftarrow e(s) + 1$
    για κάθε $s \in \{\text{επεισόδιο}\}$
      $V(s) \leftarrow V(s) + \alpha \cdot D\_TARGET \cdot e(s)$
      $e(s) \leftarrow (\gamma\lambda) \cdot e(s)$
    $s \leftarrow s'$
  until s να είναι τελική
until (Συνθήκη σύγκλισης)

Η συνθήκη σύγκλισης στον αλγόριθμο καθορίζεται έτσι ώστε, οι τελικές εκτιμήσεις να είναι αξιόπιστες, δηλαδή αμερόληπτες και με μικρή τυπική απόκλιση ώστε να είμαστε βέβαιοι πως έχει επέλθει σύγκλιση. Αυτό μπορεί να γίνεται εναλλακτικώς, αν αντί της εντολής repeat τρέξουμε τις αντίστοιχες εντολές για συγκεκριμένο πλήθος επαναλήψεων, και μετά να γίνεται ο έλεγχος της σύγκλισης.

7.4 *SARSA (λ) ΑΛΓΟΡΙΘΜΟΣ*

Όπως και ο αλγόριθμος ***Sarsa*** που αναφέραμε στο προηγούμενο κεφάλαιο, ο ***Sarsa(λ)*** αλγόριθμος είναι ένας *on policy* αλγόριθμος για την επίλυση του προβλήματος της ενισχυτικής μάθησης, χρησιμοποιώντας όμως και τα ίχνη προσπέλασης του TD(λ). Ουσιαστικά ο αλγόριθμος ακολουθώντας μία πολιτική εκτιμά την συνάρτηση αξιολόγησης της πολιτικής αυτής και βελτιώνει εν συνεχεία την πολιτική προς μία καλύτερη, έως ότου προκύψει η βέλτιστη πολιτική. Καθώς ο ***Sarsa(λ)*** χρησιμοποιεί την συνάρτηση αξιολόγησης Q δηλαδή εκτιμά τις τιμές της Q, $Q(s,a) \quad \gamma\iota\alpha\ \forall s, \kappa\alpha\iota\ \forall \alpha \in A(s)$, χρειάζεται και τα αντίστοιχα ίχνη προσπέλασης για τις χρονικές στιγμές t : $e_t(s,a) \quad \gamma\iota\alpha\ \forall s, \kappa\alpha\iota\ \forall \alpha \in A(s)$.

Επομένως, για κάθε ανανέωση στο βήμα t :

$$Q_{t+1}(s,a) = Q_t(s,a) + a \cdot D\_TARGET_t \cdot e_t(s,a) \quad \forall s,a \ ,$$

όπου η διαφορά της υπάρχουσας τιμής από τον στόχο την στιγμή *t* είναι:

$$D\_TARGET_t = r_t + \gamma \cdot Q_t\left(s_{t+1}, a_{t+1}\right) - Q_t\left(s_t, a_t\right).$$

Τα ίχνη προσπέλασης: $e_t(s,a) = (\gamma\lambda) \cdot e_{t-1}(s,a) + I_{ss_t} \cdot I_{aa_t}$

$$\text{όπου} \quad I_{ss_t} \quad \text{δείκτρια με} \quad I_{ss_t} = 1, \ \text{αν } s_t = s$$

$$\text{όπου} \quad I_{aa_t} \quad \text{δείκτρια με} \quad I_{aa_t} = 1, \ \text{αν } a_t = a$$

Τα ίχνη προσπέλασης δίνουν έναν βαθμό επιρροής της αντίστοιχης κατάστασης και απόφασης στην αμοιβή που προκύπτει *n* βήματα μετά, ή αλλιώς κατά πόσο πρέπει να συνυπολογιστεί μία αμοιβή στην ανανέωση της τιμής της συνάρτησης αξιολόγησης *n* βήματα πίσω (και πάντα για μία ακολουθία καταστάσεων αμοιβών σε ένα επεισόδιο).

Ο αντίστοιχος αλγόριθμος επίλυσης του προβλήματος της ενισχυτικής μάθησης είναι :

Sarsa(λ) / TD(λ) On-Policy *αλγόριθμος με eligibility traces*

$\Gamma\iota\alpha\ \kappa\acute{\alpha}\theta\varepsilon\ s \in S, a \in A(s) \quad o\rho\acute{\iota}\zeta\varepsilon\tau\alpha\iota \quad Q(s,a) \quad (\alpha\upsilon\theta\alpha\acute{\iota}\rho\varepsilon\tau\alpha)$

Repeat $\gamma\iota\alpha\ \kappa\acute{\alpha}\theta\varepsilon\ \varepsilon\pi\varepsilon\iota\sigma\acute{o}\delta\iota o$:

$\Gamma\iota\alpha\ \kappa\acute{\alpha}\theta\varepsilon\ s \in S, a \in A(s) \quad o\rho\acute{\iota}\zeta\varepsilon\tau\alpha\iota \quad e(s,a) = 0$

$\alpha\rho\chi\iota\kappa\acute{\alpha} \quad s \quad \kappa\alpha\iota \quad \alpha \quad (\pi\rho o\sigma o\mu o\acute{\iota}\omega\sigma\eta\ \acute{\eta}\ \pi\alpha\rho\alpha\tau\acute{\eta}\rho\eta\sigma\eta)$

Repeat $(\gamma\iota\alpha\ \kappa\acute{\alpha}\theta\varepsilon\ \beta\acute{\eta}\mu\alpha\ \tau o\upsilon\ \varepsilon\pi\varepsilon\iota\sigma o\delta\acute{\iota}o\upsilon)$:

$\lambda\acute{\eta}\psi\eta\ \alpha\pi\acute{o}\phi\alpha\sigma\eta\varsigma\ a\ \gamma\iota\alpha\ \tau\eta\nu\ s, \pi\alpha\rho\alpha\tau\acute{\eta}\rho\eta\sigma\eta \quad r \quad \kappa\alpha\iota \quad s'$

$\varepsilon\pi\iota\lambda o\gamma\acute{\eta}\ \alpha\pi\acute{o}\phi\alpha\sigma\eta\varsigma\ a'\ \gamma\iota\alpha\ \tau\eta\nu\ s'\ (\mu\varepsilon\ \pi o\lambda\iota\tau\iota\kappa\acute{\eta}\ \varepsilon - greedy\ \alpha\pi\acute{o}\ \tau\eta\nu\ Q)$

$D\_TARGET \leftarrow r + \gamma Q(s',a') - Q(s,a)$

$e(s,a) \leftarrow e(s,a) + 1$

$\gamma\iota\alpha\ \kappa\acute{\alpha}\theta\varepsilon\ (s,a) \in \{\varepsilon\pi\varepsilon\iota\sigma\acute{o}\delta\iota o\}$:

$Q(s,a) \leftarrow Q(s,a) + \alpha \cdot D\_TARGET \cdot e(s,a)$

$e(s,a) \leftarrow (\gamma\lambda) \cdot e(s,a)$

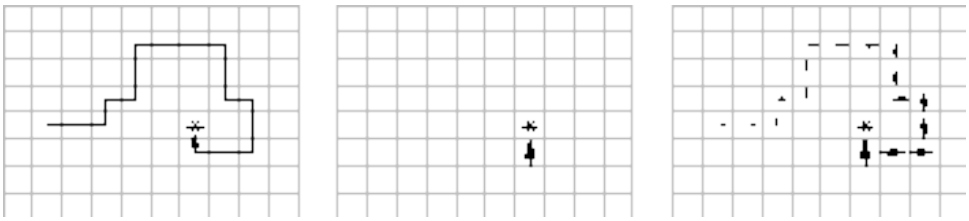
$s \leftarrow s' \quad \kappa\alpha\iota \quad a \leftarrow a'$

until s $\nu\alpha\ \varepsilon\acute{\iota}\nu\alpha\iota\ \tau\varepsilon\lambda\iota\kappa\acute{\eta}$

until $(\Sigma\upsilon\nu\theta\acute{\eta}\kappa\eta\ \sigma\acute{\upsilon}\gamma\kappa\lambda\iota\sigma\eta\varsigma)$

Η συνθήκη σύγκλισης στον αλγόριθμο καθορίζεται έτσι ώστε, οι τελικές εκτιμήσεις να είναι αξιόπιστες, δηλαδή αμερόληπτες και με μικρή τυπική απόκλιση ώστε να είμαστε βέβαιοι πως έχει επέλθει σύγκλιση. Αυτό μπορεί να γίνεται εναλλακτικώς, αν αντί της

εντολής repeat τρέξουμε τις αντίστοιχες εντολές για συγκεκριμένο πλήθος επαναλήψεων, και μετά να γίνεται ο έλεγχος της σύγκλισης.



Στο σχήμα φαίνεται η διαφορά του αλγορίθμου *Sarsa από τον Sarsa(λ)*. Μία διαδρομή όπως καταγράφτηκε κατά την διάρκεια ενός επεισοδίου, έως το τελικό επιθυμητό σημείο (το τετράγωνο με το *), φαίνεται στο αριστερό τετράγωνο. Η τελική αμοιβή του τετραγώνου αυτού (που είναι και η επιθυμητή κατάληξη της διαδρομής), επηρεάζει σύμφωνα με τον αλγόριθμο *Sarsa ενός βήματος,* την τιμή της συνάρτησης αξιολόγησης του προτελευταίου τετραγώνου (από το οποίο μεταπήδησε στο τελικό). Αυτό φαίνεται στο μεσαίο τετράγωνο. Αντιστοίχως, η τελική αμοιβή του τετραγώνου με το *, επηρεάζει σύμφωνα με τον αλγόριθμο *Sarsa(λ) (με λ=0,9)*, την τιμή της συνάρτησης αξιολόγησης όλων των τετραγώνων της διαδρομής που ακολουθήθηκε στο επεισόδιο, όπως φαίνεται στο δεξιό τετράγωνο με τα αντίστοιχα βάρη όπου φαίνεται και η επιρροή του παράγοντα μνήμης ***λ***.

7.5 *Q(λ) ΑΛΓΟΡΙΘΜΟΣ ή Watkins's Q(λ)*

Όπως και ο αλγόριθμος ***Q*** ενός βήματος του προηγουμένου κεφαλαίου, ο ***Q(λ)*** αλγόριθμος είναι ένας *off policy* αλγόριθμος για την επίλυση του προβλήματος της ενισχυτικής μάθησης, χρησιμοποιώντας και τα ίχνη προσπέλασης του TD(λ). Ουσιαστικά ο αλγόριθμος ακολουθώντας μία συγκεκριμένη πολιτική (*soft* ή *ε-greedy* πολιτική, για να κάνει *exploration* κατά την πραγματοποίηση του επεισοδίου), εκτιμά την συνάρτηση αξιολόγησης που τελικά συγκλίνει στη βέλτιστη συνάρτηση αξιολόγησης ***Q**** και από όπου προκύπτει η βέλτιστη πολιτική (*greedy* πολιτική).

Ο αλγόριθμος ***Q*** χρησιμοποιεί μία *soft* ή *ε-greedy* πολιτική για να εξασφαλίσει πως θα προσπελαστούν όλες οι καταστάσεις-αποφάσεις σε βάθος χρόνου. Θεωρούμε στον αλγόριθμο αυτόν, πως οι αμοιβές που προκύπτουν στα επόμενα βήματα μίας κατάστασης θα πρέπει να έχουν αντίκτυπο στην κατάσταση, μόνο αν οι αποφάσεις που λήφθηκαν από την κατάσταση έως και την αμοιβή αυτή, ακολουθούν την *greedy* πολιτική (τέτοια θα είναι και η τελική βέλτιστη πολιτική). Δηλαδή, αν και κατά την διάρκεια του επεισοδίου επιτρέπεται βάσει της *soft* ή *ε-greedy* πολιτικής να λαμβάνουμε αποφάσεις που δεν είναι φαινομενικά οι καλύτερες ***–exploratory απόφαση***- (ως προς τις εκτιμήσεις έως τότε της συνάρτησης *Q*), εν τούτοις, για την ανανέωση των τιμών (με τα αντίστοιχα ίχνη προσπέλασης) θα λαμβάνονται μόνο οι αμοιβές που ακολουθούσαν μία σειρά από *greedy* αποφάσεις ***–exploitation απόφαση***- και μόνο.

Αυτό είναι εύκολα υλοποιήσιμο με την βοήθεια των ιχνών προσπέλασης, καθώς μπορούμε κάθε φορά που λαμβάνεται μία ***exploratory απόφαση*** τα ίχνη προσπέλασης να μηδενίζονται, οπότε και στον αντίστοιχο όρο διόρθωσης του *TARGET* δεν υπάρχουν οι αντίστοιχες ανανεώσεις.

Επομένως, έχουμε για κάθε ανανέωση στο βήμα t :

$$Q_{t+1}(s,a) = Q_t(s,a) + a \cdot D\_TARGET_t \cdot e_t(s,a) \quad \forall s,a \text{ του επεισοδίου,}$$

με τη διαφορά του στόχου: $D\_TARGET_t = r_{t+1} + \gamma \cdot \max_{a'} Q_t\left(s_{t+1}, a'_t\right) - Q_t\left(s_t, a_t\right)$

και ίχνη προσπέλασης: $e_t(s,a) = I_{ss_t} \cdot I_{aa_t} + (\gamma\lambda) \cdot e_{t-1}(s,a) \cdot I_{a^* a_t}$

$$\mu\varepsilon \quad I_{ss_t} \ \delta\varepsilon\acute{\iota}\kappa\tau\rho\iota\alpha \ \mu\varepsilon \quad I_{ss_t} = 1, \ \alpha\nu \ s_t = s$$
$$\mu\varepsilon \quad I_{aa_t} \ \delta\varepsilon\acute{\iota}\kappa\tau\rho\iota\alpha \ \mu\varepsilon \quad I_{aa_t} = 1, \ \alpha\nu \ a_t = a$$
$$\mu\varepsilon \quad I_{a^* a_t} \ \delta\varepsilon\acute{\iota}\kappa\tau\rho\iota\alpha \ \mu\varepsilon \quad I_{a^* a_t} = 1, \ \alpha\nu \ a_t = a^* \quad (\beta\acute{\varepsilon}\lambda\tau\iota\sigma\tau\eta \ \pi o\lambda\iota\tau\iota\kappa\acute{\eta} \ \beta\acute{\alpha}\sigma\varepsilon\iota \ Q)$$

Δηλαδή: $I_{a^* a_t} = 1, \quad \alpha\nu \ Q_{t-1}(s_t, a_t) = \max_a \left\{ Q_{t-1}(s_t, a) \right\}$

Ο αντίστοιχος αλγόριθμος είναι :

Q(λ) / TD(λ) Off-Policy *αλγόριθμος με eligibility traces*

$\Gamma\iota\alpha \ \kappa\acute{\alpha}\theta\varepsilon \ s \in S, a \in A(s) \quad o\rho\acute{\iota}\zeta\varepsilon\tau\alpha\iota \quad Q(s,a) \quad (\alpha\upsilon\theta\alpha\acute{\iota}\rho\varepsilon\tau\alpha)$

Repeat $\gamma\iota\alpha \ \kappa\acute{\alpha}\theta\varepsilon \ \varepsilon\pi\varepsilon\iota\sigma\acute{o}\delta\iota o$:

$\Gamma\iota\alpha \ \kappa\acute{\alpha}\theta\varepsilon \ s \in S, a \in A(s) \quad o\rho\acute{\iota}\zeta\varepsilon\tau\alpha\iota \quad e(s,a) = 0$

$\alpha\rho\chi\iota\kappa\acute{\alpha} \quad s \quad \kappa\alpha\iota \quad \alpha \quad (\pi\rho o\sigma o\mu o\acute{\iota}\omega\sigma\eta \ \acute{\eta} \ \pi\alpha\rho\alpha\tau\acute{\eta}\rho\eta\sigma\eta)$

Repeat ($\gamma\iota\alpha \ \kappa\acute{\alpha}\theta\varepsilon \ \beta\acute{\eta}\mu\alpha \ \tau o\upsilon \ \varepsilon\pi\varepsilon\iota\sigma o\delta\acute{\iota}o\upsilon$) :

$\lambda\acute{\eta}\psi\eta \ \alpha\pi\acute{o}\phi\alpha\sigma\eta\varsigma \ a \ \gamma\iota\alpha \ \tau\eta\nu \ s, \pi\alpha\rho\alpha\tau\acute{\eta}\rho\eta\sigma\eta \quad r \quad \kappa\alpha\iota \quad s'$

$\varepsilon\pi\iota\lambda o\gamma\acute{\eta} \ \alpha\pi\acute{o}\phi\alpha\sigma\eta\varsigma \ a' \ \gamma\iota\alpha \ \tau\eta\nu \ s' \ (\mu\varepsilon \ \pi o\lambda\iota\tau\iota\kappa\acute{\eta} \ \varepsilon - greedy \ \alpha\pi\acute{o} \ \tau\eta\nu \ Q)$

$a^* \leftarrow \arg\max_b \left\{ Q(s', b) \right\}$

$D\_TARGET \leftarrow r + \gamma Q(s', a^*) - Q(s,a)$

$e(s,a) \leftarrow e(s,a) + 1$

$\gamma\iota\alpha \ \kappa\acute{\alpha}\theta\varepsilon \ (s,a) \in \{\varepsilon\pi\varepsilon\iota\sigma\acute{o}\delta\iota o\}$:

$Q(s,a) \leftarrow Q(s,a) + \alpha \cdot D\_TARGET \cdot e(s,a)$

if $a' = a^*$ *then* $e(s,a) \leftarrow (\gamma\lambda) \cdot e(s,a)$

else $e(s,a) \leftarrow 0$

$s \leftarrow s'$

$a \leftarrow a'$

until s $\nu\alpha \ \varepsilon\acute{\iota}\nu\alpha\iota \ \tau\varepsilon\lambda\iota\kappa\acute{\eta}$

until $(\Sigma\upsilon\nu\theta\acute{\eta}\kappa\eta \ \sigma\acute{\upsilon}\gamma\kappa\lambda\iota\sigma\eta\varsigma)$

Η συνθήκη σύγκλισης στον αλγόριθμο καθορίζεται έτσι ώστε, οι τελικές εκτιμήσεις να είναι αξιόπιστες, δηλαδή αμερόληπτες και με μικρή τυπική απόκλιση ώστε να είμαστε βέβαιοι πως έχει επέλθει σύγκλιση. Αυτό μπορεί να γίνεται εναλλακτικώς, αν αντί της εντολής repeat τρέξουμε τις αντίστοιχες εντολές για συγκεκριμένο πλήθος επαναλήψεων, και μετά να γίνεται ο έλεγχος της σύγκλισης.

ΚΕΦΑΛΑΙΟ 8ο.

ΕΦΑΡΜΟΓΕΣ

Στο τέταρτο μέρος της παρούσας διπλωματικής θα εφαρμόσουμε κάποιους από τους αλγόριθμους που αναλύσαμε στα τελευταία κεφάλαια. Θα πάρουμε αποτελέσματα από την εφαρμογή των αλγορίθμων σε δύο διαφορετικά προβλήματα, και θα εξετάσουμε την απόδοση των αλγορίθμων. Συγκεκριμένα θα εφαρμόσουμε τους αλγόριθμους ***SARSA, SARSA(λ), Q*** και ***Q(λ)***, στο πρόβλημα ελέγχου αποθεμάτων, όπως θα ορίσουμε λεπτομερώς παρακάτω.

Γενικότερα, όπως έχουμε αναφέρει, οι αλγόριθμοι ***SARSA(λ)*** και ***Q(λ)***, μέσω της παραμέτρου μνήμης ***λ*** χρησιμοποιούνται για προβλήματα όπου είτε υπάρχει το φαινόμενο της καθυστερημένης αμοιβής, δηλαδή όταν κάποιο ζεύγος κατάστασης-απόφασης επιφέρει αμοιβή σε χρονική περίοδο μεταγενέστερης του ζεύγους, είτε όταν οι καταστάσεις και αποφάσεις είναι αρκετά συνδεδεμένες μεταξύ τους, ώστε να θεωρούμε πως κάποια αλλαγή στην αξιολόγηση μίας κατάστασης θα πρέπει να επιφέρει αλλαγές και στις τιμές αξιολόγησης καταστάσεων και αποφάσεων που προηγήθηκαν αυτής. Η τελευταία θεώρηση μπορεί να μην στηρίζεται στο φαινόμενο της καθυστερημένης αμοιβής, αλλά στο ότι λόγω των εξισώσεων *Bellman*, ή από την διαφορά παρατήρησης και επόμενης τιμής αξιολόγησης από την τωρινή τιμή, όποια διόρθωση γίνεται θα επηρεάσει και διορθώσεις σε επόμενες χρονικές περιόδους. Αυτό θα γίνεται γιατί σε κάποια επόμενη διαδοχή καταστάσεων-αποφάσεων όπου θα προσπελαθεί η τωρινή κατάσταση-απόφαση, και επειδή θα έχει αλλάξει η τιμή αξιολόγησής της η αντίστοιχη διαφορά $[TARGET - OLD]$ η οποία και θα εμφανισθεί, θα επιφέρει αλλαγές στην κατάσταση-απόφαση που προηγήθηκε και ούτω καθεξής. Επομένως, είναι κατά έναν τρόπο λογικό να εφαρμόσουμε τους αλγόριθμους αυτούς, με την παράμετρο μνήμης, γιατί θα γίνονται περισσότερες από μία αλλαγές στις τιμές αξιολόγησης σε κάθε βήμα του αλγορίθμου και επομένως ενδέχεται να έχουμε γρηγορότερη σύγκλιση στον αλγόριθμο. Προσοχή βέβαια απαιτεί η επιλογή της τιμής της παραμέτρου *λ* καθώς μία τιμή κοντά στο 1 θα επιφέρει αλλαγές στις τιμές αξιολόγησης αρκετών καταστάσεων-αποφάσεων που προηγήθηκαν και αυτό μπορεί να είναι λανθασμένο. Η επιλογή της τιμής θα πρέπει να είναι τέτοια ώστε να είναι λογικές οι αλλαγές σε τέτοιο πλήθος καταστάσεων-αποφάσεων που να ταιριάζει στο κάθε πρόβλημα. Θα αναφερθούμε στην επιλογή της τιμής του *λ* και στις εφαρμογές παρακάτω.

Επίσης, οι αλγόριθμοι αυτοί μπορούν να εφαρμοστούν και σε προβλήματα πεπερασμένου χρονικού ορίζοντα, αλλά και σε προβλήματα απείρου χρονικού ορίζοντα. Πάντα η αντίστοιχη εφαρμογή και ο ορισμός του προβλήματος καθορίζουν ποιόν από τους δύο χρονικούς ορίζοντες θα εφαρμόσουμε.

Το πρόβλημα για το οποίο θα εφαρμόσουμε τους αλγόριθμους είναι το πρόβλημα ελέγχου αποθεμάτων. Δοσμένου ενός αποθηκευτικού χώρου (αποθήκη), θέλουμε να βρούμε τον βέλτιστο αριθμό προϊόντων που πρέπει να υπάρχουν κάθε χρονική περίοδο στην αποθήκη, (π.χ. επειδή αποφασίζουμε να παράγουμε ή να παραγγείλουμε κάποιον

αριθμό τεμαχίων του προϊόντος, ενώ συγχρόνως υπάρχει κάποια ζήτηση για το προϊόν αυτό και καλύπτεται από την αποθήκη αυτή). Η αποθήκη τροφοδοτείται με δική μας απόφαση και αδειάζει λόγω της ζήτησης, οπότε ο βέλτιστος αριθμός τεμαχίων με τα οποία θα αποφασίσουμε να συμπληρώσουμε (κατά ένα μέρος) την αποθήκη είναι συνάρτηση της ζήτησης που υπάρχει για το προϊόν αυτό, του κέρδους ανά μονάδα προϊόντος, αλλά και του αποθέματος του προϊόντος. Θα ασχοληθούμε με δύο βασικές παραλλαγές του προβλήματος. Στην παράγραφο ***8.1*** θα ασχοληθούμε με το πρόβλημα ελέγχου αποθεμάτων για ένα προϊόν, ενώ στην παράγραφο ***8.2*** θα ασχοληθούμε με το πρόβλημα ελέγχου αποθεμάτων για δύο προϊόντα σε κοινή αποθήκη.

8.1 ΈΛΕΓΧΟΣ ΑΠΟΘΕΜΑΤΩΝ (ΈΝΑ ΠΡΟΪΟΝ)

Για το πρόβλημα ελέγχου αποθεμάτων για ένα προϊόν θεωρούμε πως διαθέτουμε μία αποθήκη στην οποία μπορούμε να αποθηκεύουμε το προϊόν. Το προϊόν είναι συγκεκριμένου όγκου ανά τεμάχιο και έτσι στην αποθήκη μπορούν να αποθηκευτούν από μηδέν έως και έναν μέγιστο αριθμό τεμαχίων/μονάδων. Θα συμβολίζουμε το μέγιστο δυνατό πλήθος μονάδων της αποθήκης με MAX. Θεωρούμε ακόμα διακριτές χρονικές στιγμές στις οποίες παρατηρούμε το απόθεμα της αποθήκης. Επομένως κάθε τέτοια χρονική στιγμή έχουμε ως κατάσταση της αποθήκης το πλήθος των τεμαχίων του προϊόντος στην αποθήκη και αυτό θα είναι το σύνολο καταστάσεων S={0,1,2,…,MAX}. Η απόφαση που πρέπει κάθε φορά (σε κάθε χρονική περίοδο) να πάρουμε, είναι η ποσότητα παραγγελίας ή παραγωγής του προϊόντος, ώστε να έχουμε το μεγαλύτερο κέρδος. Αυτή η απόφαση λαμβάνεται αμέσως μετά την παρατήρηση της κατάστασης s (απόθεμα της αποθήκης) στην αρχή της χρονικής περιόδου. Θεωρούμε πως η παραγωγή γίνεται άμεσα και συγχρόνως και η τοποθέτηση στην αποθήκη, οπότε η παραγωγή περιορίζεται πάντα από την πεπερασμένη χωρητικότητα της αποθήκης και θα είναι από το σύνολο {0,1,2,…,(MAX-s)}. Κατά την διάρκεια της υπόλοιπης χρονικής περιόδου γίνονται πωλήσεις, οι οποίες για την προσομοίωση που θα κάνουμε εμείς θα ακολουθούν κάποια κατανομή, αλλά γενικότερα μπορεί να εφαρμοστεί και σε πραγματικό πρόβλημα όπου η ζήτηση θα είναι άγνωστη μεν αλλά η συγκεκριμένη τιμή της ζήτησης στην δεδομένη χρονική στιγμή θα παρατηρείται στην αρχή κάθε χρονικής περιόδου.

Κάθε τεμάχιο προϊόντος έχει συγκεκριμένη τιμή πώλησης καθ' όλη την διάρκεια του «πειράματος» και συγκεκριμένο κόστος αποθήκευσης για κάθε χρονική περίοδο. Ακόμα, για το κόστος παραγωγής υπάρχει συγκεκριμένο κόστος ανά τεμάχιο προϊόντος c, αλλά ενδέχεται, όταν υπάρχει παραγωγή έστω και ενός προϊόντος, να υπάρχει και ένα σταθερό κόστος K που αντιστοιχεί στο κόστος θέσεως σε λειτουργία την γραμμή παραγωγής. Επομένως για την παραγωγή A τεμαχίων έχουμε κόστος παραγωγής:

$$\kappa\acute{o}\sigma\tau o\varsigma = K + c \cdot A, \quad \gamma\iota\alpha\ A > 0 \ .$$

Κάθε χρονική στιγμή, γνωρίζουμε το απόθεμα της αποθήκης και αυτή είναι η κατάστασή μας s_t από το σύνολο S={0,1,2,…,MAX}, αποφασίζουμε παραγωγή a_t και κατά την διάρκεια

της παρούσας χρονικής περιόδου «έρχεται» η ζήτηση για το προϊόν οπότε και στην αρχή της επομένης περιόδου παρατηρούμε την νέα κατάσταση s_{t+1}. Η αμοιβή που δεχόμαστε σε κάθε χρονική στιγμή είναι το κέρδος από τις πωλήσεις, και στόχος μας είναι η μέγιστη αθροιστική αμοιβή. Η βέλτιστη απόφαση παραγωγής ώστε να μεγιστοποιηθεί η αθροιστική αυτή αμοιβή υπολογίζεται για δύο παραλλαγές του προβλήματος. Πρώτον για πεπερασμένο ορίζοντα Τ και δεύτερον για άπειρο ορίζοντα.

Πριν παρουσιάζουμε τις δύο περιπτώσεις θα αναφερθούμε στις παραμέτρους του προβλήματος. Εκτός από τις δεδομένες τιμές κόστους, πώλησης, αποθήκευσης κ.α. έχουμε και τον συντελεστή αποπληθωρισμού g για την αναγωγή των αμοιβών στην χρονική στιγμή όπου υπολογίζουμε τις τιμές αυτές. Ειδικά για το πρόβλημα άπειρου χρονικού ορίζοντα αυτός θα πρέπει να είναι αυστηρά μικρότερος του ένα. Επίσης για τους αλγόριθμους έχουμε την παράμετρο βηματικής μάθησης $a \in (0,1)$, η οποία μπορεί να είναι σταθερή και αρκετά κοντά στο μηδέν, ώστε να συνυπολογίζει περισσότερο και παλαιές τιμές όπως έχουμε αναφέρει στα προηγούμενα κεφάλαια. Αλλά αν η παράμετρος a είναι αρχικά κοντά στο ένα και επειδή αρχικά οι τιμές της συνάρτησης αξιολόγησης είναι τυχαίες (εμείς τις έχουμε θέσει αρχικά μηδέν), από την παρατήρηση των πραγματικών (μέσω τις προσομοίωσης) άμεσων αμοιβών κάθε χρονική στιγμή, θα έχουμε καλύτερες διορθώσεις/ανανεώσεις των τιμών της συνάρτησης αξιολόγησης. Θυμίζουμε τον κανόνα ανανέωσης:

$$NEW = OLD + a \cdot (TARGET - OLD)$$

Επομένως, θα πρέπει να έχουμε αρχικά “μεγάλες” τιμές στην παράμετρο a (π.χ. 0,6 ή 0,8) και όσο έχουμε περισσότερα στοιχεία (περισσότερα σενάρια του αλγορίθμου), η τιμή της παραμέτρου a θα πρέπει να μειώνεται ώστε να έχουμε έναν σταθμισμένο μέσο ο οποίος δεν θα στηρίζεται πολύ στις τελευταίες τιμές που “συνάντησε” ο αλγόριθμος.

Αντιστοίχως, πρέπει να σκεφτούμε για την παράμετρο e η οποία υπάρχει στην $e-greedy$ πολιτική. Θυμίζουμε πως υπάρχει το πρόβλημα διαδοχής *exploration exploitation* όπου θα πρέπει αρχικώς να εξερευνούμε/ψάχνουμε νέες καταστάσεις-αποφάσεις με σκοπό να βρούμε και νέες πιθανόν καλύτερες αποφάσεις, αλλά σταδιακά θα πρέπει να λαμβάνουμε και τις βέλτιστες αποφάσεις ώστε να υπάρχει τελικά σύγκλιση για τον αλγόριθμο. Έτσι, και για την παράμετρο e αρχικά έχουμε μεγάλες τιμές (0,8 ή και 0,9) και όσο πληθαίνουν τα επεισόδια και η αποκτώμενη εμπειρία του μαθητή (εδώ ο αλγόριθμος) η παράμετρος e θα πρέπει να μειώνεται.

Τέλος για το πρόβλημα ελέγχου αποθεμάτων ενός προϊόντος και άπειρη χωρητικότητα της αποθήκης υπάρχει γνωστή λύση και η οποία εξαρτάται από το αν υπάρχει σταθερό κόστος K θέσεως της γραμμής παραγωγής σε λειτουργία ή όχι (οπότε και το κόστος είναι ανάλογο του πλήθους παραγωγής A).

Η γνωστή πολιτική παραγωγής είναι της παρακάτω μορφής για τις δύο περιπτώσεις:

Για $K=0$ και άπειρο χρονικό ορίζοντα η παραγωγή είναι τέτοια ώστε να συμπληρώνεται το απόθεμα της αποθήκης έως έναν συγκεκριμένο αριθμό S'. Δηλαδή, για απόθεμα s μικρότερο του S' η βέλτιστη απόφαση είναι $(S'-s)$, ενώ για απόθεμα μεγαλύτερο του S' η βέλτιστη απόφαση παραγωγής είναι μηδέν.

Για $K>0$ και άπειρο χρονικό ορίζοντα η παραγωγή είναι τέτοια ώστε να συμπληρώνεται το απόθεμα της αποθήκης έως έναν συγκεκριμένο αριθμό S' μόνο για τις περιπτώσεις όπου έχουμε απόθεμα μικρότερο από ένα πλήθος s', $με\ s'<S'$. Δηλαδή, για απόθεμα s μικρότερο ή ίσο του s' η βέλτιστη απόφαση είναι $(S'-s)$, ενώ για απόθεμα μεγαλύτερο του s' η βέλτιστη απόφαση παραγωγής είναι μηδέν.

<u>ΠΡΟΒΛΗΜΑ ΠΕΠΕΡΑΣΜΕΝΟΥ ΧΡΟΝΙΚΟΥ ΟΡΙΖΟΝΤΑ ΜΕ ΣΤΑΘΕΡΟ ΚΟΣΤΟΣ ΜΗΔΕΝ</u>

Για το πρόβλημα πεπερασμένου χρονικού ορίζοντα και μάλιστα για βάθος 10 χρονικών μονάδων, τρέξαμε τον αντίστοιχο αλγόριθμο ***Q*** για τις παρακάτω τιμές, ενώ για την τελευταία χρονική στιγμή (*t=10*) θεωρήσαμε πως το υπολειπόμενο απόθεμα της αποθήκης πουλήθηκε στο ένα δέκατο της κανονικής του τιμής (άδειασμα της αποθήκης).

ΒΑΘΟΣ ΧΡΟΝΟΥ	ΜΕΓΕΘΟΣ ΑΠΟΘΗΚΗΣ	ΤΙΜΗ ΠΩΛΗΣΗΣ ΑΝΑ ΤΕΜΑΧΙΟ	ΣΤΑΘΕΡΟ ΚΟΣΤΟΣ ΠΑΡΑΓΩΓΗΣ K	ΚΟΣΤΟΣ ΠΑΡΑΓΩΓΗΣ ΑΝΑ ΤΕΜΑΧΙΟ	ΚΟΣΤΟΣ ΑΠΟΘΗΚΕΥΣΗΣ
10	10	15	0	5	2

Το κέρδος από την πώληση ενός τεμαχίου του προϊόντος είναι 10 χρηματικές μονάδες εάν πουληθεί άμεσα ενώ αν παραμείνει στην αποθήκη και πουληθεί την επόμενη χρονική περίοδο η είσπραξη θα είναι 8, την παρεπόμενη 6 και ούτω καθεξής (χωρίς υπολογισμό του αποπληθωριστικού παράγοντα).

Η ζήτηση που θέσαμε για το προϊόν ήταν 5 ή 6 τεμάχια με πιθανότητες ½, σταθερή για κάθε χρονική περίοδο.

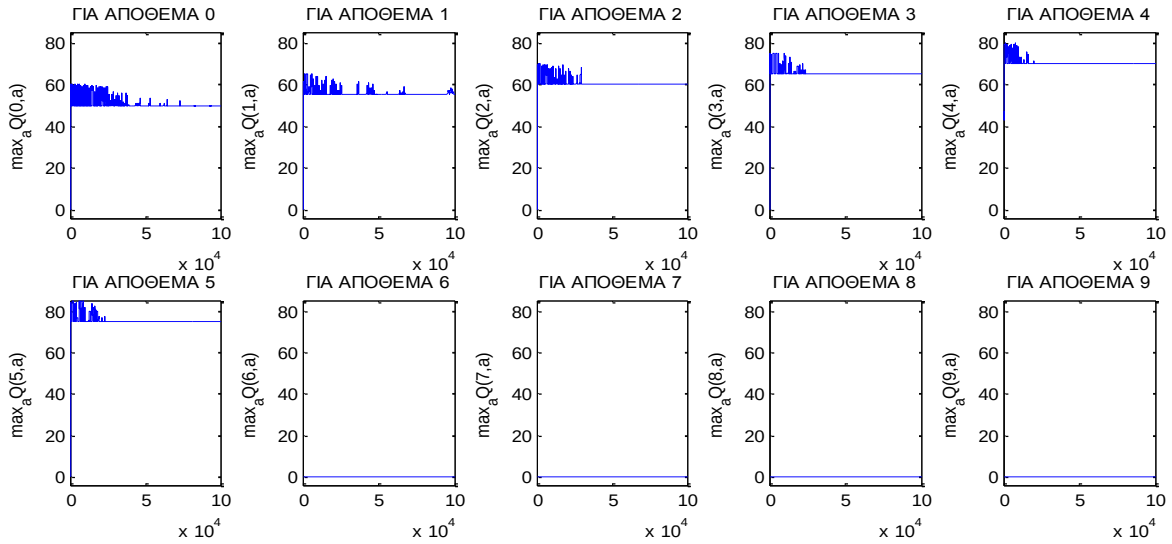
Οι τιμές των παραμέτρων για τον αλγόριθμο που δόθηκαν αρχικά ήταν:

Παράμετρος e (e-greedy πολιτικής) $^{(*)}$	Παράμετρος βηματικής μάθησης α $^{(*)}$	Παράμετρος αποπληθωρισμού g	Πλήθος επαναλήψεων N
0,8	0,8	0,9	100.000

$^{(*)}$ Όπως αναφέραμε η παράμετρος αυτή μειώνεται κατά την διάρκεια του αλγορίθμου.

Για κάθε χρονική στιγμή των 10 χρονικών βημάτων υπάρχουν οι αντίστοιχες τιμές της συνάρτησης αξιολόγησης, για όλες τις καταστάσεις, και οι αντίστοιχες βέλτιστες αποφάσεις.

Για την 9^{η} χρονική περίοδο (μία περίοδο πριν την τερματική και το άδειασμα της αποθήκης), ο αλγόριθμος συνέκλινε στις παρακάτω τιμές ως προς την μέγιστη τιμή (συναρτήσει της απόφασης) για κάθε κατάσταση (ΑΠΟΘΕΜΑ της αποθήκης):

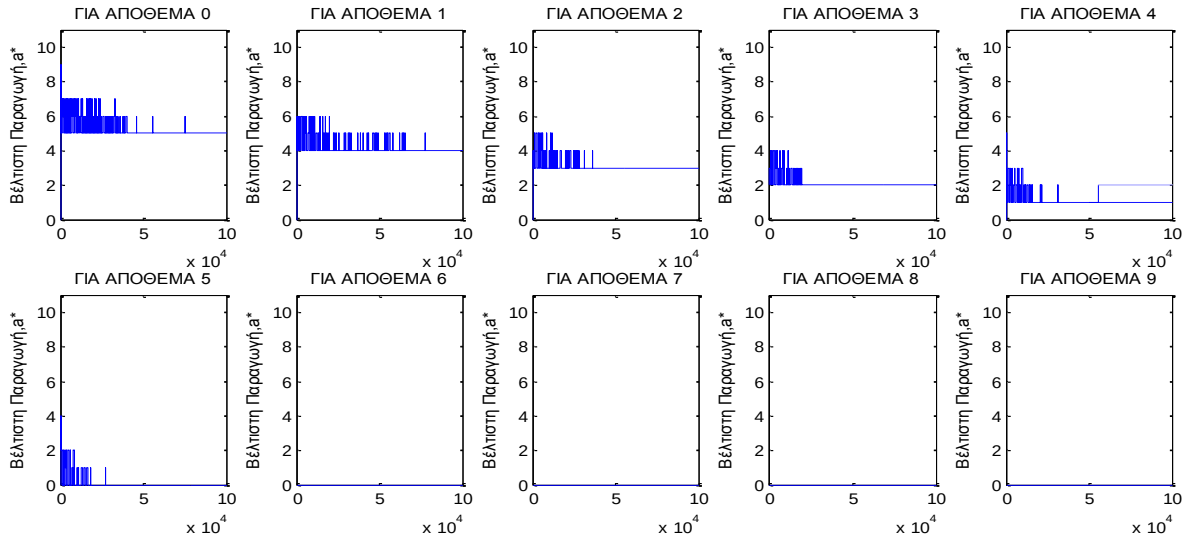


Μέγιστη τιμή της συνάρτησης Q για την 9η περίοδο (από τις 10) για κάθε μία από τις καταστάσεις του προβλήματος.

Για παράδειγμα, το τρίτο διάγραμμα δίνει την μέγιστη τιμή, όπως αυτή διαμορφώθηκε στο πέρασμα 100.000 επαναλήψεων του αλγορίθμου, για την περίπτωση που έχουμε απόθεμα 2 τεμάχια στην αποθήκη. Συγκεκριμένα εδώ είναι 60 χρηματικές μονάδες, και αντιστοιχεί στην πώληση των 2 τεμαχίων αποθέματος συν την πώληση των τεμαχίων που θα παραχθούν (ως βέλτιστη απόφαση α*=3) ενώ στην αποθήκη θα έχουμε νέο απόθεμα 0 τεμάχια, καθώς η ζήτηση είναι 5 ή 6 τεμάχια και εμείς έχουμε ως απόφαση τέτοια παραγωγή, ώστε το απόθεμα μετά την παραγωγή και πριν να διατεθούν τα προϊόντα προς πώληση να φτάνει τα 5 τεμάχια.

Παρατηρούμε πως οι περιπτώσεις να έχουμε απόθεμα 6,7,8,9 ή και 10 τεμάχια εμφανίζεται με μηδέν τιμή. Αυτό συμβαίνει γιατί όσο η αποθήκη έχει μέγιστη χωρητικότητα 10 και η ζήτηση είναι τουλάχιστον 5, δεν μπορούν να μείνουν αδιάθετα πάνω από 5 τεμάχια προϊόντος.

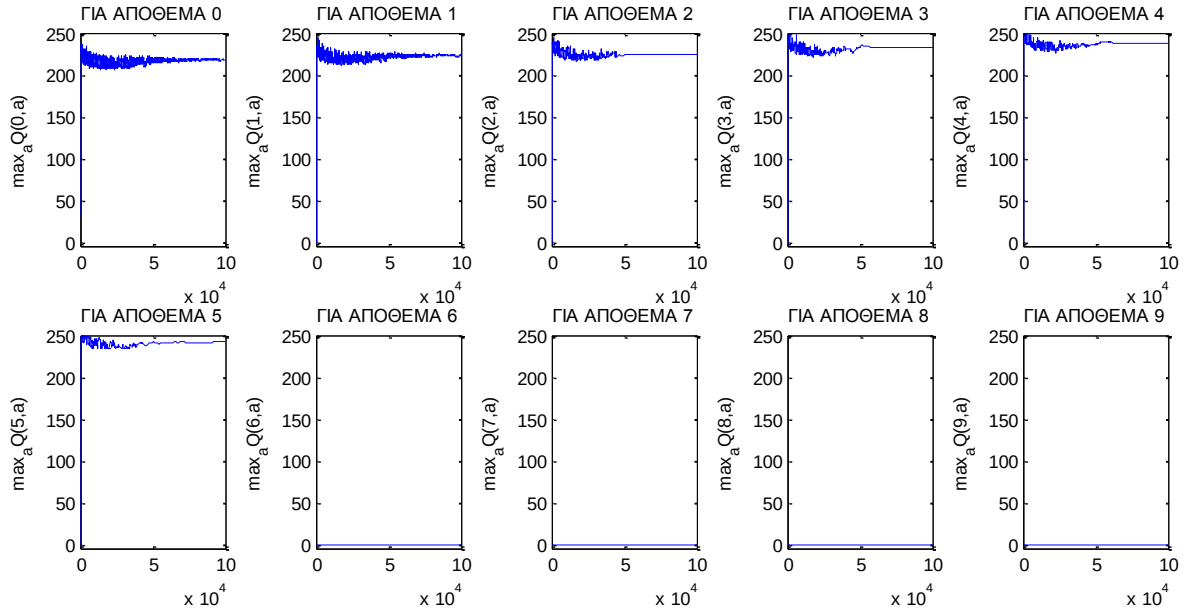
Οι αντίστοιχες βέλτιστες αποφάσεις παραγωγής α* όπως αυτές διαμορφώθηκαν κατά την διάρκεια των 100.000 επαναλήψεων φαίνονται στα παρακάτω διαγράμματα:



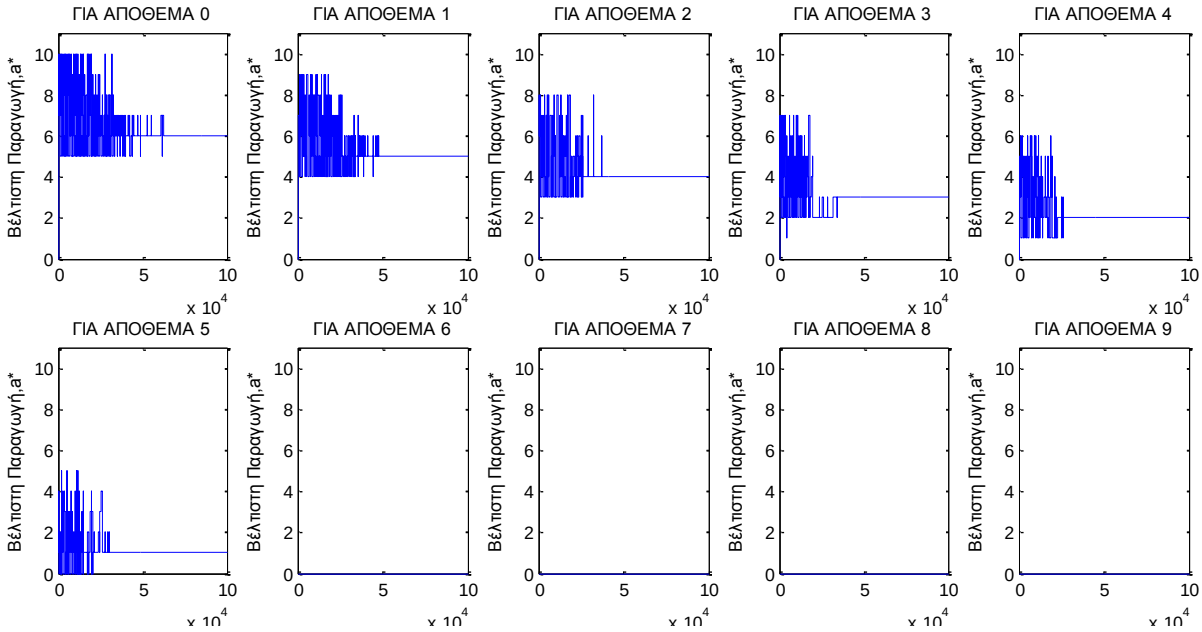
Βέλτιστη απόφαση για κάθε μία από τις καταστάσεις της 9ης περιόδου, από τις 10 του προβλήματος.

Φαίνεται από τα διαγράμματα αυτά πως η πολιτική είναι ίδια με αυτήν του προβλήματος άπειρου χρονικού ορίζοντα και άπειρης χωρητικότητας της αποθήκης, όπως προαναφέραμε. Δηλαδή, έχουμε μία πολιτική (για την $9^{η}$ περίοδο) που ορίζει παραγωγή τέτοια ώστε να συμπληρώνεται το απόθεμα της αποθήκης έως έναν συγκεκριμένο αριθμό S'. Εδώ $S'=5$.

Αντιστοίχως για την $5^{η}$ χρονική περίοδο (πέντε περιόδους πριν την τερματική και το άδειασμα της αποθήκης), ο αλγόριθμος συνέκλινε στις παρακάτω τιμές ως προς την μέγιστη τιμή (συναρτήσει της απόφασης) για κάθε κατάσταση (ΑΠΟΘΕΜΑ της αποθήκης):



Παρατηρούμε εδώ τις μεγαλύτερες τιμές από ότι στην περίπτωση της $9^{ης}$ περιόδου, καθώς υπολογίζεται η αθροιστική αμοιβή έως το τέλος (και της $10^{ης}$ περιόδου). Οι τιμές αυτές είναι περίπου 220-240 χρηματικές μονάδες καθώς υπάρχει και ο παράγοντας αποπληθωρισμού g για της επόμενες 5 περιόδους. Είναι φανερό και εδώ πως έχουμε σύγκλιση των τιμών, και οι αντίστοιχες βέλτιστες αποφάσεις φαίνονται από τα παρακάτω διαγράμματα:

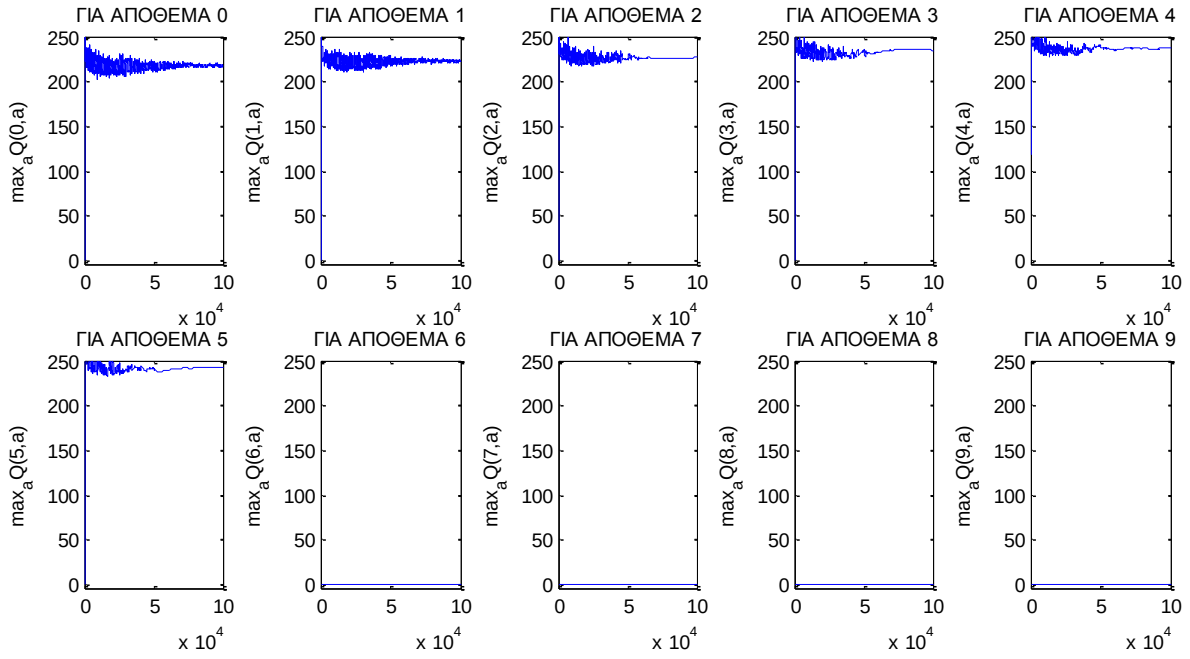


Εδώ διακρίνεται μία παρόμοια πολιτική όπως αυτή που είδαμε στην 9η περίοδο. Και εδώ έχουμε μία πολιτική όπου ορίζει παραγωγή τόσων τεμαχίων όσων χρειάζονται για την συμπλήρωση συγκεκριμένου αποθέματος S', αλλά στην 5η περίοδο αυτή η τιμή ορίζεται στα 6 τεμάχια ($S'=6$). Αυτό βέβαια είναι λογικό γιατί όπως είπαμε η ζήτηση είναι 5 ή 6 τεμάχια και καθώς έχουμε την δυνατότητα να αποθηκεύσουμε το ένα τεμάχια, που ενδεχομένως θα μείνει αδιάθετο στην αγορά, θα πουληθεί στην κανονική του τιμή την επομένη περίοδο (ενώ στην 9η περίοδο θα έμενε στην αποθήκη και θα κοστολογείτο στο 1/10 της τιμής του). Επομένως συμφέρει να έχουμε 6 διαθέσιμα τεμάχια προς πώληση στην αρχή της περιόδου, και εάν πουληθούν και τα έξι έχουμε το μέγιστο κέρδος, ενώ αν πουληθούν τα πέντε θα έχω κατά 3 χρηματικές μονάδες μικρότερο κέρδος (κόστος αποθήκευσης 2, είσπραξη 10 με αποπληθωριστικό παράγοντα 0,9 είναι: 10·0,9-2=7).

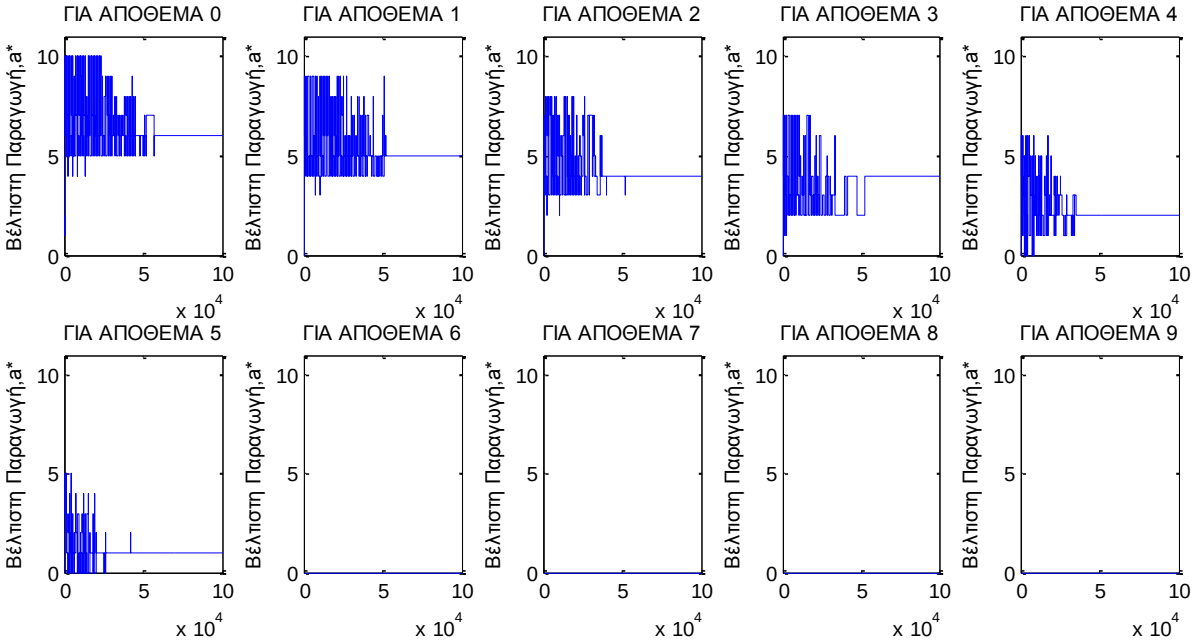
_____ Αλγόριθμος Q(λ) _____

Από τον αντίστοιχο αλγόριθμο ***Q(λ)*** έχουμε ίδια αποτελέσματα ειδικά για τιμές της παραμέτρου ***λ*** μικρότερες του 0,5, (ειδικά για λ=0 οι δύο αλγόριθμοι ξέρουμε πως ταυτίζονται). Για τιμές της παραμέτρου λ>0,5 και ειδικά για τιμές πολύ κοντά στο 1 (λ>0,98) ο αλγόριθμος ενώ δίνει τα ίδια αποτελέσματα φαίνεται σε πάρα πολύ μικρό βαθμό να καθυστερεί στην σύγκλιση. Ενώ στην περίπτωση όπου πήραμε λ=0,5 ή 0,6 είχαμε ελάχιστα πιο γρήγορη σύγκλιση. Όπως προαναφέραμε η παράμετρος λ είναι ο βαθμός που αποδίδουμε κάποια αλλαγή των τιμών και σε προηγούμενες καταστάσεις. Πρώτον λοιπόν, όσο πλησιέστερο το λ στο 1 τόσο οι διορθώσεις (σωστές οι λανθασμένες) αποδίδονται και σε περισσότερες παλαιότερες καταστάσεις και έτσι διορθώνουν σειριακά τις αντίστοιχες τιμές και αυτών. Μία τιμή κοντά στο 0,5 επιφέρει αλλαγή στην κατάσταση που προηγήθηκε της τωρινής κατά 50% ενώ σε 2 καταστάσεις πριν αλλαγή κατά 25%, σε 3 κατά 12,5% και από εκεί και πέρα πολύ μικρή αλλαγή. Αυτή η τιμή της λ θα μπορούσε κάλλιστα να εφαρμοστεί σε προβλήματα όπου οι καταστάσεις είναι σε μεγάλο βαθμό συνδεδεμένες μεταξύ τους, με συνέπεια κάθε κατάσταση να επηρεάζει 2, 3 ή και 4 καταστάσεις πριν και μετά από αυτήν. Σε μία τέτοια περίπτωση ο αλγόριθμος Q(λ) θα ήταν πιο γρήγορος από τον Q, αλλά στην περίπτωση του δικού μας προβλήματος και μάλιστα με μέγιστη χωρητικότητα της αποθήκης 10 και ζήτηση 5 ή 6, το τι θα γίνει στην μία περίοδο δεν μπορεί εκ των πραγμάτων να επηρεάσει καταστάσεις με χρονική διαφορά πάνω από 2 βήματα. Για παράδειγμα εάν έχουμε γεμάτη την αποθήκη (10 τεμάχια) τότε με την συγκεκριμένη ζήτηση σε δύο χρονικά βήματα (στην τωρινή στιγμή και την επόμενη) το απόθεμα αυτό θα έχει τελειώσει είτε παράξουμε είτε όχι το προϊόν.

Τα αντίστοιχα διαγράμματα για την περίπτωση πεπερασμένου χρονικού ορίζοντα (10 βημάτων) και για την 5η χρονική περίοδο είναι με λ=0,98 από τον αλγόριθμο Q(λ):



Για την σύγκλιση των μέγιστων τιμών, ως προς την απόφαση, της συνάρτησης αξιολόγησης Q(s,a) της 5ης χρονικής περιόδου:



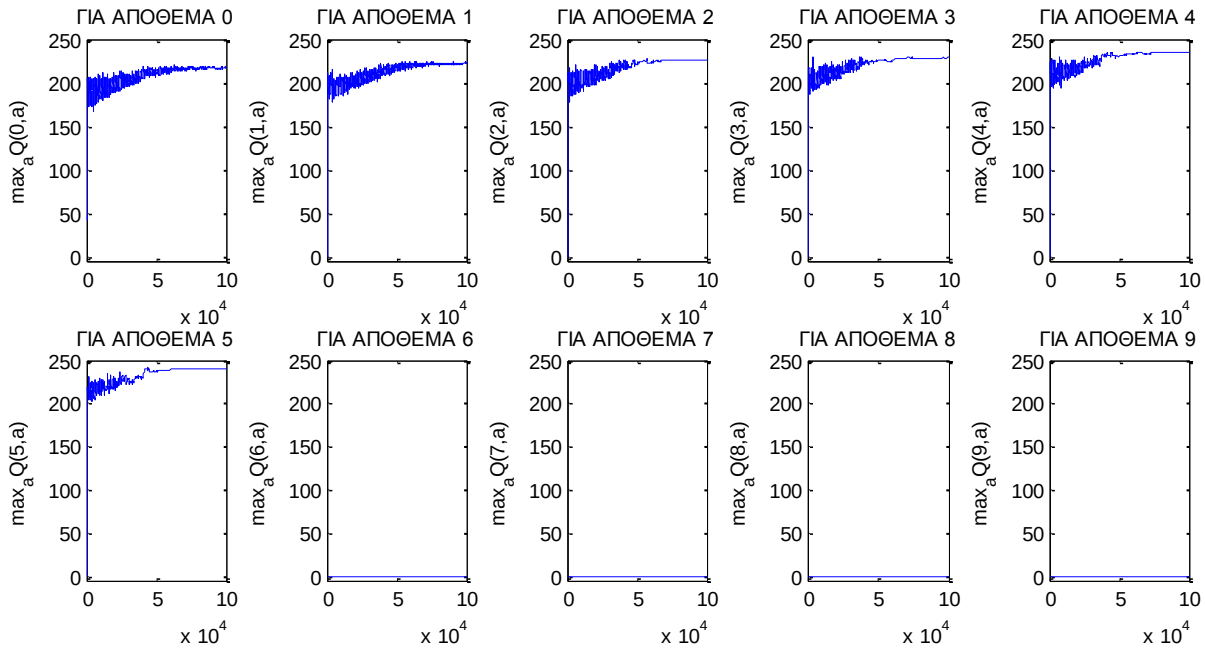
Παρατηρούμε γενικά πως στον αλγόριθμο Q η σύγκλιση στις τιμές της συνάρτησης αξιολόγησης γίνεται πολύ γρήγορα, και μπορεί να παρατηρηθεί ότι στα αρχικά βήματα του αλγορίθμου υπερεκτιμούνται οι τιμές αυτές φτάνοντας και στις 2.500-3.000 χρηματικές μονάδες(στα παρακάτω διαγράμματα δεν φαίνεται γιατί έχει τυπωθεί συγκεκριμένο μόνο εύρος του κατακόρυφου άξονα), αλλά γρήγορα επανέρχονται στα σωστά επίπεδα και σταθεροποιούνται όπως φαίνεται στο σχήμα.

Ενώ οι βέλτιστες αποφάσεις δίνουν το ίδιο αποτέλεσμα. Έχουμε πολιτική τέτοιας παραγωγής ώστε το απόθεμα μαζί με το πλήθος των παραγόμενων τεμαχίων να γίνεται 6.

Αλγόριθμος SARSA

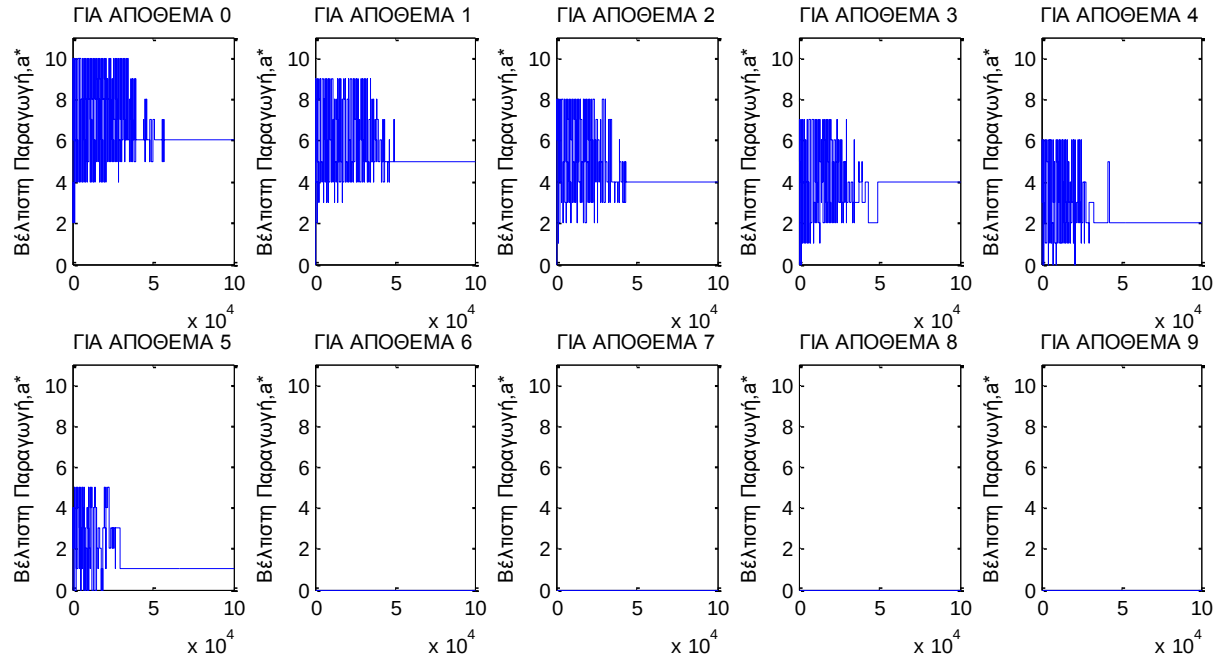
Για τον αλγόριθμο SARSA, για τα ίδια δεδομένα του προβλήματος, και ίδιες τιμές στις παραμέτρους, παρατηρείται εξίσου καλή σύγκλιση στις τιμές της συνάρτησης αξιολόγησης και μάλιστα χωρίς να υπερεκτιμάται στην αρχή του αλγορίθμου όπως στον αλγόριθμο Q. Αυτό οφείλεται στην λογική των αλγορίθμων, καθώς ο αλγόριθμος SARSA πάντα βελτιώνει και ανανεώνει τις τιμές βάσει της άμεσης εμπειρίας του (εκεί όπως έχουμε πει οφείλει και την ονομασίας του: State Action Reward State Action), οπότε και μέχρι να σταθεροποιηθούν οι τιμές όλων των καταστάσεων-αποφάσεων η προσέγγιση στις σωστές τιμές γίνεται σταδιακά, ενώ ο αλγόριθμος Q πάντα ανανεώνει τις τιμές βάση της άμεσης αμοιβής και της βέλτιστης επόμενης απόφασης (όπου κι θα αποφέρει την μεγαλύτερη αθροιστική αμοιβή), και για τον λόγο αυτό υπερεκτιμά αρχικά τις τιμές της συνάρτησης αξιολόγησης.

Για το πρόβλημα που έχουμε θέσει λοιπόν, ο αλγόριθμος SARSA συγκλίνει στις αντίστοιχες τιμές της συνάρτησης αξιολόγησης για την 5η χρονική περίοδο από τις 10, όπως φαίνεται στα διαγράμματα:



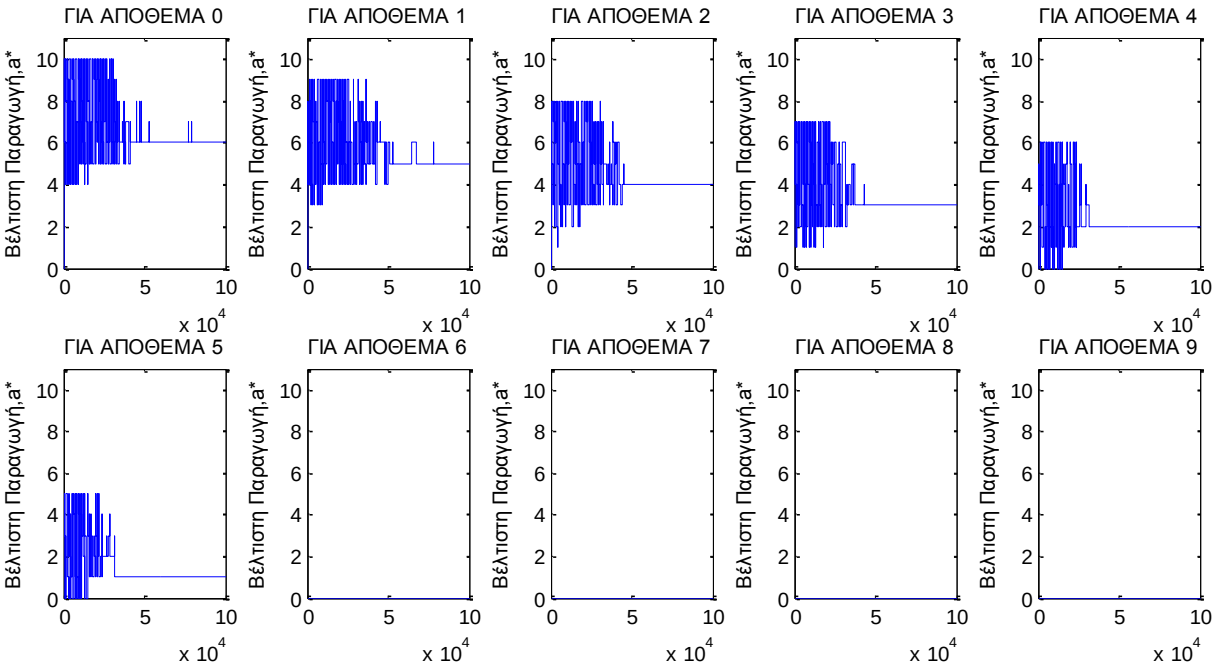
Όσον αφορά τις βέλτιστες αποφάσεις που προκύπτουν από τον αλγόριθμο SARSA για την 5η περίοδο **δεν έχουμε πάντα σωστά αποτελέσματα.** Στις περισσότερες φορές που τρέξαμε τον αλγόριθμο (ακόμα και με διάφορες τιμές για τις παραμέτρους e και α), οι βέλτιστες αποφάσεις ήταν κατά κανόνα σωστές εκτός από μία ή και δύο σε κάποιες περιπτώσεις. Ενώ συγκεκριμένα περιμέναμε μία πολιτική όπου θα απαιτεί το απόθεμα συν την παραγόμενη ποσότητα να είναι σταθερή ($S'=6$), σε κάποια από τις καταστάσεις (σε μία από αυτές ή και δύο σπανιότερα), είχαμε μία πολιτική όπου απαιτούσε το αντίστοιχο άθροισμα να είναι $S'=6\pm1$ ή $S'=6\pm2$. Καλύτερη απόδοση του αλγορίθμου είχαμε μόνο όταν αλλάξαμε την τιμή της παραμέτρου g και την θέσαμε 0,6 από 0,9 που είχαμε στις υπόλοιπες εφαρμογές.

Ο αλγόριθμος SARSA λοιπόν, δεν ήταν σταθερός στην εκτίμηση των βέλτιστων αποφάσεων όπως διακρίνεται από το παρακάτω διάγραμμα, όπου φαίνονται οι βέλτιστες αποφάσεις για την $5^{η}$ χρονική περίοδο, όπως διαμορφώθηκαν αυτές κατά την διάρκεια των 100.000 επαναλήψεων:

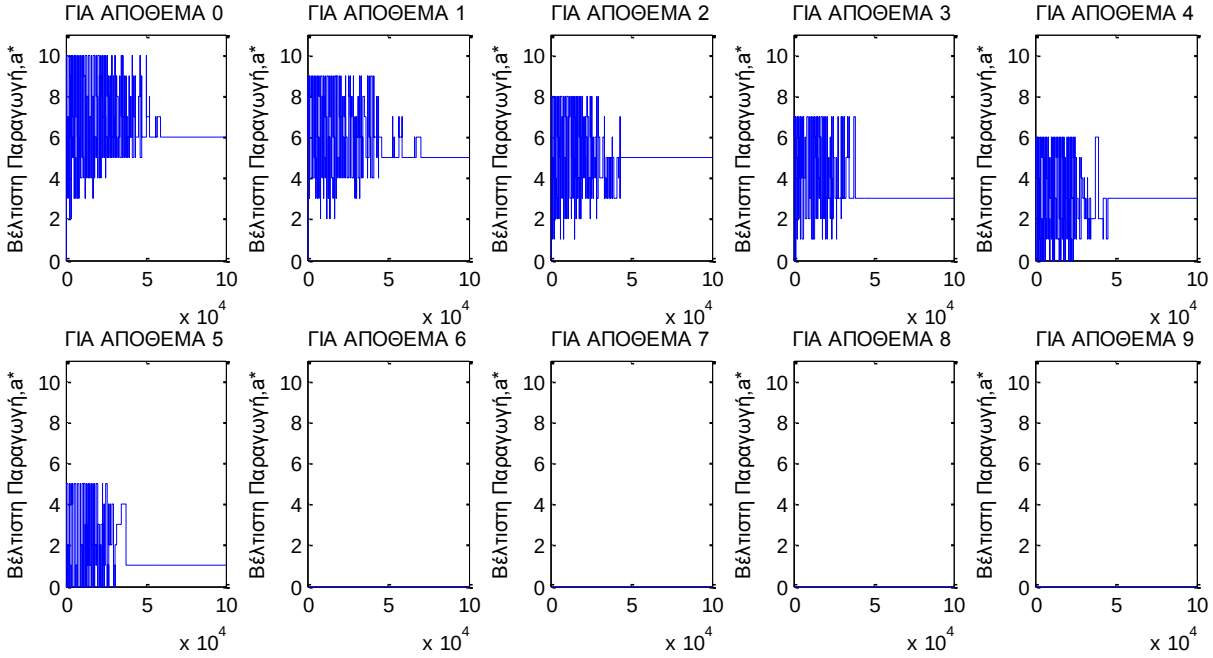


Διακρίνεται στην περίπτωση όπου το απόθεμα είναι 3 πως η βέλτιστη παραγωγή από τον αλγόριθμο είναι 4 τεμάχια αντί 3 που θα έπρεπε, όπως βγάζει και ο αλγόριθμος Q. Πρέπει να σημειωθεί βέβαια πως για τις περιόδους 8,9 και 10 δεν παρατηρήθηκε κανένα πρόβλημα ως προς την σωστή απόδοση των βέλτιστων αποφάσεων από τον αλγόριθμο.

Το παραπάνω λάθος δεν εμφανιζόταν πάντα με κάθε εφαρμογή του αλγορίθμου SARSA για το πρόβλημα, όπως φαίνεται και στα παρακάτω διαγράμματα:

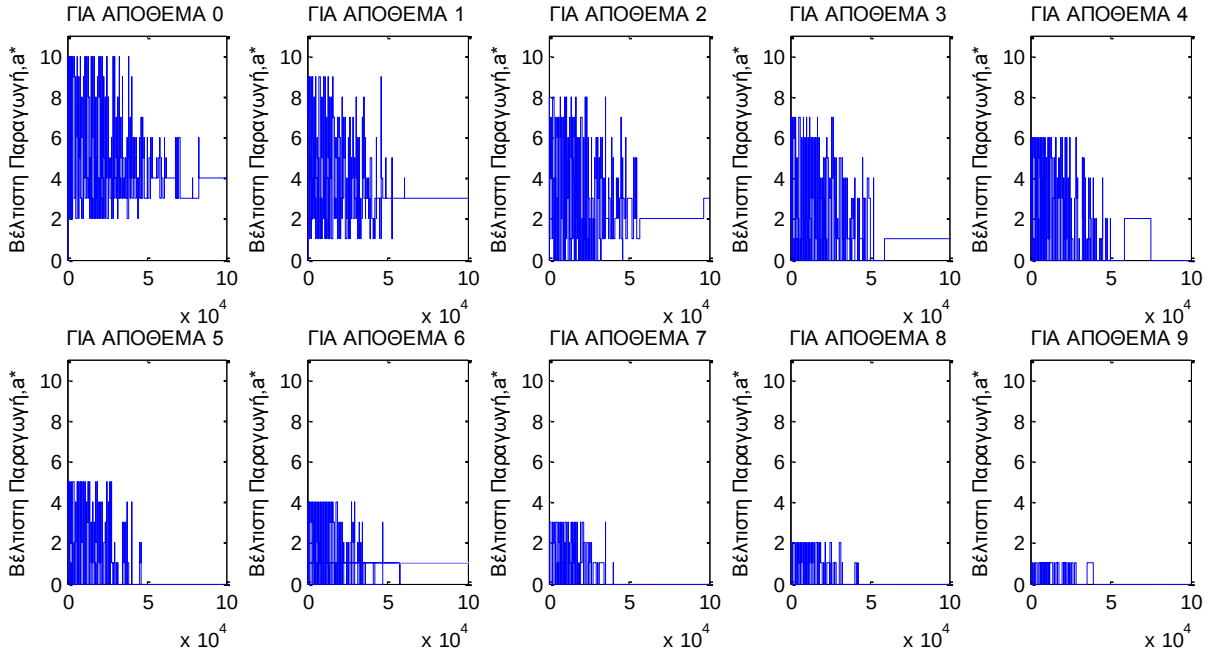


Ομοίως από την εφαρμογή του αλγορίθμου SARSA(λ), για το ίδιο πρόβλημα, είχαμε παρόμοια αποτελέσματα τα οποία περιείχαν αντίστοιχα λάθη ως προς τις βέλτιστες αποφάσεις. Αυτά τα λάθη έγιναν ελαφρώς πιο έντονα όταν η παράμετρος λ πλησίαζε το 1. Ενδεικτικά παραθέτουμε τα αντίστοιχα διαγράμματα της 5^{ης} χρονικής περιόδου, με λ=0,95:



Παρατηρούμε πως στις περιπτώσεις για απόθεμα 2 και 4 έχουμε λανθασμένη απόφαση καθώς το άθροισμα αποθέματος και παραγόμενης ποσότητας είναι 7 και στις δύο περιπτώσεις αντί 6 όπως θα έπρεπε.

Ακόμα τρέξαμε τον αλγόριθμο Q για ζήτηση 1,2,3,4 ή 5 τεμαχίων. Η πολιτική που προκύπτει είναι παρόμοια, με την μόνη διαφορά πως καθυστερεί στην σύγκλιση ο αλγόριθμος (απαιτεί περισσότερες επαναλήψεις). Για την 5^{η} περίοδο τα αποτελέσματα ήταν για τον Q:



<u>Παρατηρήσεις κριτική για την εφαρμογή</u>:

Τα λάθη που επισημάναμε στα αποτελέσματα των αλγορίθμων δεν σημαίνουν πως οι αντίστοιχοι αλγόριθμοι, SARSA και SARSA(λ), είναι λανθασμένοι ή χειρότεροι από τους 2 αλγορίθμους, Q και Q(λ). Ενδέχεται τα λάθη αυτά να προκύπτουν από την φύση του προβλήματος και την αντίστοιχη λογική των αλγορίθμων. Οι αλγόριθμοι SARSA εμπλέκουν άμεσα για την ανανέωση των τιμών δύο διαδοχικές καταστάσεις-αποφάσεις οι οποίες προσπελάθηκαν κατά την διάρκεια του αλγορίθμου, ενώ αυτό δεν το κάνουν οι αλγόριθμοι Q. Οι Q αλγόριθμοι δεν εμπλέκουν μία κατάσταση-απόφαση με την αμέσως επόμενη κατάσταση-απόφαση, αλλά με την επόμενη κατάσταση σε συνδυασμό με την καλύτερη απόφαση της νέας κατάστασης, και αυτό φαίνεται να ταιριάζει καλύτερα στην φύση του συγκεκριμένου προβλήματος, αλλά συγχρόνως αυτή η αλγοριθμική προσέγγιση είναι η αιτία που αποφεύγονται λανθασμένες διορθώσεις/ανανεώσεις στις τιμές της συνάρτησης αξιολόγησης. Μία διόρθωση σε ένα ζεύγος κατάστασης-αποφασης δεν στηρίζεται στην επόμενη κατάσταση-απόφαση. Επειδή κάθε απόφαση που λαμβάνεται στον αλγόριθμο είναι e-greedy απόφαση, ενδέχεται στην επόμενη κατάσταση να ληφθεί απόφαση όχι βέλτιστη (αυτό επιθυμούμε για exploration), και αυτό να “χρεωθεί” στην κατάσταση-απόφαση που βρισκόμαστε. Το τελευταίο είναι κάτι που ενδεχομένως να προκαλέσει προβλήματα στην απόδοση των αλγορίθμων SARSA και ειδικά αν δεν επιλεχθούν κατάλληλες τιμές για τις παραμέτρους e και *α*.

Επομένως, στο συγκεκριμένο πρόβλημα οι αλγόριθμοι Q και Q(λ) συμπεριφέρονται καλύτερα από τους αλγόριθμους SARSA, και βέβαια δίνουν σωστές βέλτιστες αποφάσεις στο πρόβλημα ελέγχου αποθεμάτων. Ακόμα στην περίπτωση όπου το λ είναι πάρα πολύ κοντά στο ένα (0,98 ή 0,99) ακόμα και ο αλγόριθμος Q(λ) φαίνεται να κάνει κάποια λάθη, αλλά σε μικρότερο βαθμό και συχνότητα από ότι ο αλγόριθμος SARSA(λ) με αντίστοιχες τιμές στις παραμέτρους. Θεωρούμε πως ο λόγος που γίνονται αυτά τα λάθη, είναι ο ίδιος λόγος στον οποίο οφείλονται τα λάθη των αλγορίθμων SARSA. Ενδεχόμενες μη βέλτιστες αποφάσεις (λόγω e-greedy πολιτικής και ζητούμενης exploration) ορίζουν διορθώσεις σε προηγούμενες καταστάσεις-αποφάσεις.

ΠΡΟΒΛΗΜΑ ΠΕΠΕΡΑΣΜΕΝΟΥ ΧΡΟΝΙΚΟΥ ΟΡΙΖΟΝΤΑ ΜΕ ΣΤΑΘΕΡΟ ΚΟΣΤΟΣ

Εφαρμόσαμε τους 4 αλγόριθμους για το ίδιο πρόβλημα αλλά με σταθερό κόστος έναρξης παραγωγής K=30 και κόστος ανά μονάδα προϊόντος ίδια. Παραθέτουμε τις τιμές του προβλήματος:

ΒΑΘΟΣ ΧΡΟΝΟΥ	ΜΕΓΕΘΟΣ ΑΠΟΘΗΚΗΣ	ΤΙΜΗ ΠΩΛΗΣΗΣ ΑΝΑ ΤΕΜΑΧΙΟ	ΣΤΑΘΕΡΟ ΚΟΣΤΟΣ ΠΑΡΑΓΩΓΗΣ K	ΚΟΣΤΟΣ ΠΑΡΑΓΩΓΗΣ ΑΝΑ ΤΕΜΑΧΙΟ	ΚΟΣΤΟΣ ΑΠΟΘΗΚΕΥΣΗΣ
10	10	15	30	5	2

Το κέρδος από την πώληση ενός τεμαχίου του προϊόντος είναι 10 χρηματικές μονάδες εάν πουληθεί άμεσα ενώ υπήρχε στην αποθήκη, αλλά -20 χρηματικές μονάδες εάν ήταν το μόνο που παράχθηκε στην τωρινή περίοδο, ενώ αν παραχθούν 2 τεμάχια το κέρδος για το κάθε ένα θα είναι -5. Το σταθερό κόστος θέσεως της γραμμής παραγωγής σε λειτουργία είναι αυτό που πρέπει να καταμεριστεί σε, μεγαλύτερο των 2, πλήθος τεμαχίων ώστε με την πώλησή τους να έχουμε θετικό κέρδος.

Είναι γνωστή η λύση του προβλήματος για $K > 0$ και άπειρο χρονικό ορίζοντα. Η παραγωγή είναι τέτοια ώστε να συμπληρώνεται το απόθεμα της αποθήκης έως έναν συγκεκριμένο αριθμό S' μόνο για τις περιπτώσεις όπου έχουμε απόθεμα μικρότερο από ένα πλήθος s', $με\ s' < S'$. Δηλαδή, για απόθεμα s μικρότερο ή ίσο του s' η βέλτιστη απόφαση είναι $(S' - s)$, ενώ για απόθεμα μεγαλύτερο του s' η βέλτιστη απόφαση παραγωγής είναι μηδέν.

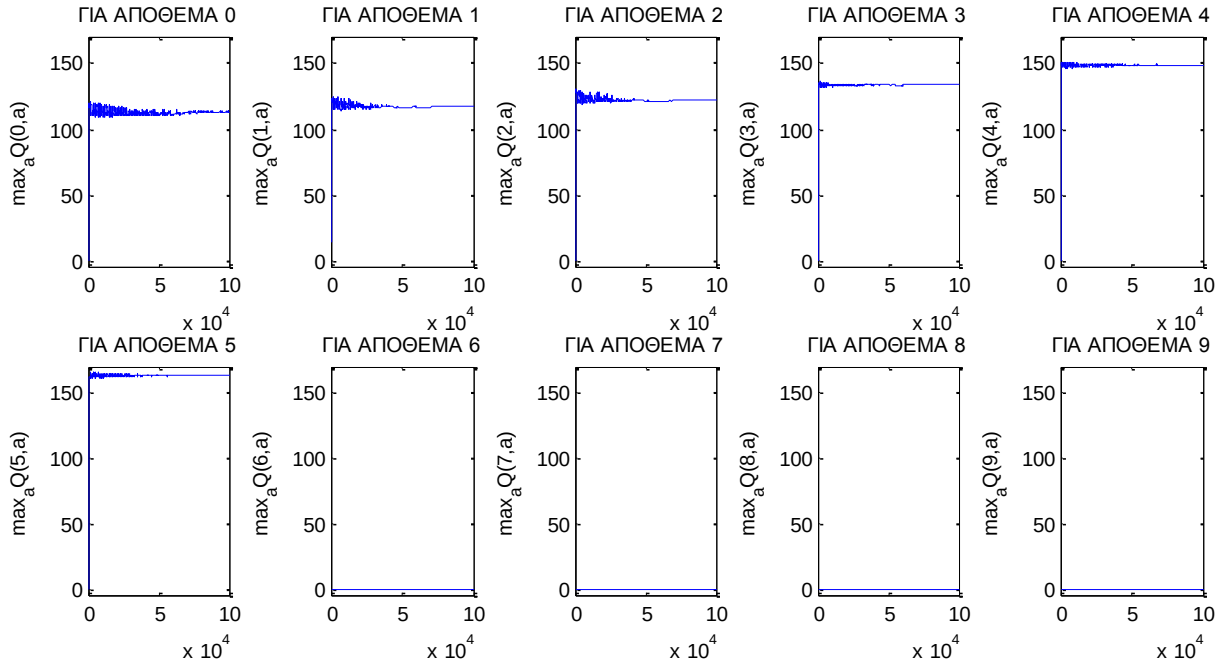
Η ζήτηση που θέσαμε για το προϊόν ήταν πάλι 5 ή 6 τεμάχια με πιθανότητες ½, σταθερή για κάθε χρονική περίοδο.

Οι τιμές των παραμέτρων για τον αλγόριθμο που δόθηκαν αρχικά ήταν ίδιες με την περίπτωση όπου δοκιμάσαμε με το *K=0*:

Παράμετρος e (e-greedy πολιτικής) (*)	Παράμετρος βηματικής μάθησης α (*)	Παράμετρος αποπληθωρισμού g	Πλήθος επαναλήψεων N
0,8	0,8	0,9	100.000

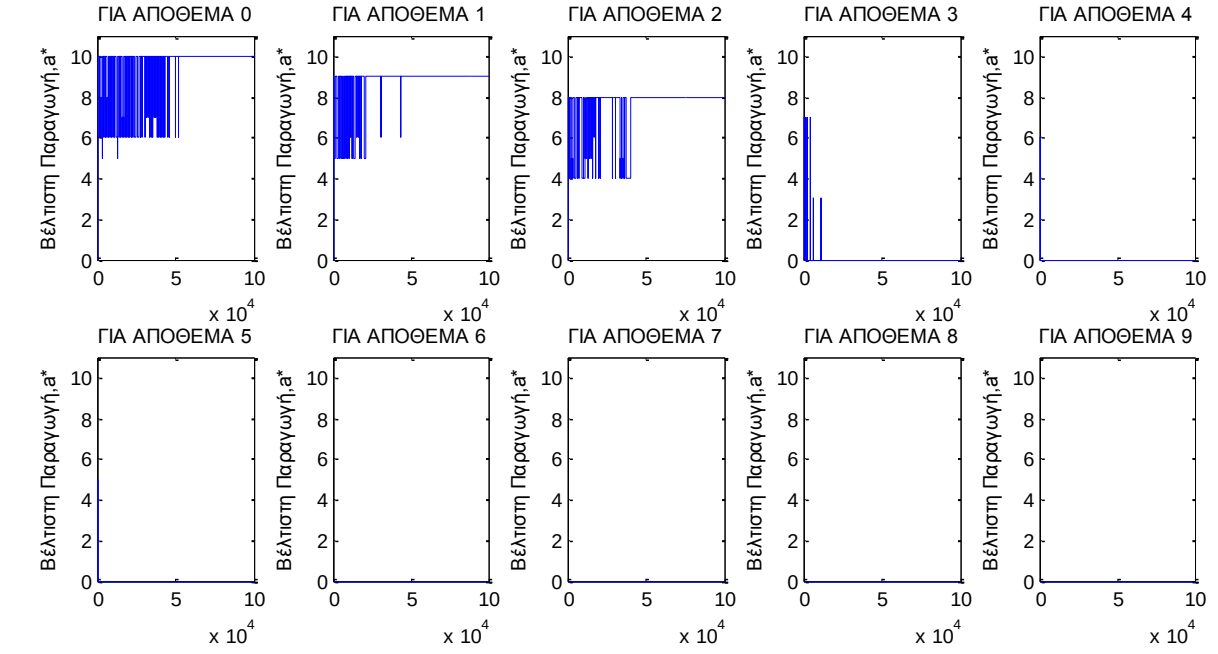
(*) Όπως αναφέραμε η παράμετρος αυτή μειώνεται κατά την διάρκεια του αλγορίθμου.

Για την 5η χρονική περίοδο από τις 10 οι αντίστοιχες τιμές της συνάρτησης αξιολόγησης, για όλες τις καταστάσεις, και οι αντίστοιχες βέλτιστες αποφάσεις φαίνονται στα παρακάτω διαγράμματα όπως προέκυψαν από τον αλγόριθμο Q:



Παρατηρούμε την σύγκλιση των τιμών περίπου με τον ίδιο τρόπο όπως και στην περίπτωση όπου είχαμε *K=0*, αρχικώς πολύ γρήγορα ξεφεύγουν οι τιμές προς τα πάνω (υπερεκτιμούνται οι καταστάσεις) αλλά γρήγορα σταθεροποιούνται γύρο από την σωστή τιμή και όσο αυξάνεται η εμπειρία (πλήθος επαναλήψεων του αλγορίθμου) η αυξομείωση γύρω από την τιμή γίνεται όλο και μικρότερη. Ακόμα παρατηρούμε και εδώ πως για τις καταστάσεις με απόθεμα 5 έως και 10 έχουμε τιμή της συνάρτησης αξιολόγησης ίση με μηδέν καθώς ποτέ δεν θα μείνει τέτοιο απόθεμα στην αποθήκη με την δεδομένη ζήτηση.

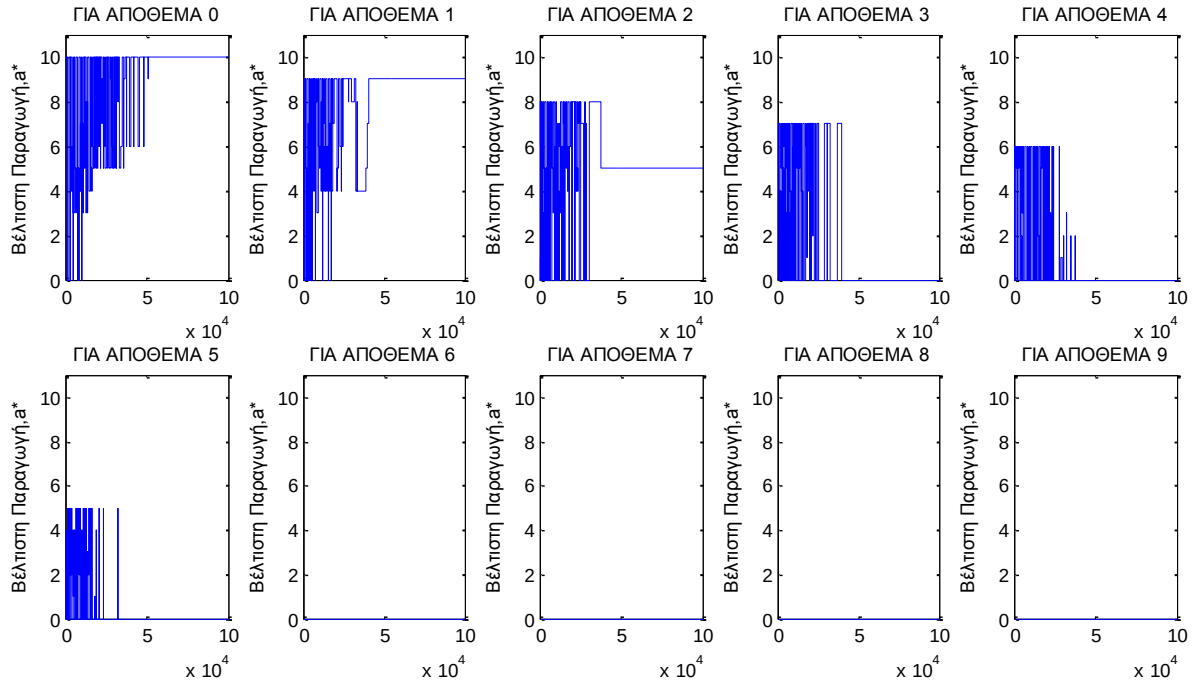
Για την βέλτιστη απόφαση παραγωγής της 5^{ης} περιόδου από τις 10, και όπως διαμορφώθηκε αυτή μέσω των N επαναλήψεων του αλγόριθμου, έχουμε τα παρακάτω διαγράμματα:



Η πολιτική που προκύπτει από τα διαγράμματα και για την 5η περίοδο είναι: «Όταν το απόθεμα είναι κάτω από 3 τεμάχια πρέπει η παραγωγή να είναι τέτοια, ώστε το νέο απόθεμα πριν την διάθεση του προϊόντος στην αγορά, να είναι ίσο με 10 τεμάχια». Δηλαδή έχουμε μία πολιτική $(s', S') = (3,10)$ όπως συμβολίζεται.

Για τους αλγόριθμους Q(λ), SARSA και SARSA(λ) τα αποτελέσματα ήταν παρόμοια και παρατηρήθηκαν τα ίδια προβλήματα όπως και στην περίπτωση με *K=0*, οπότε έχουμε και τις ίδιες παρατηρήσεις-κριτική για την απόδοση των αλγορίθμων.

Ενδεικτικά παραθέτουμε τα διαγράμματα για την 5η πάλι περίοδο από τον αλγόριθμο SARSA(λ) με λ=0,98:



Διακρίνεται στο 3ο διάγραμμα, για την περίπτωση με απόθεμα 2, πως αντί για βέλτιστη απόφαση παραγωγής 8 τεμάχια, ο αλγόριθμος μας δίνει παραγωγή 5 τεμαχίων. Τέτοια λάθη εμφανίζονταν αρκετά συχνά στις δοκιμές των αλγορίθμων για τους SARSA και SARSA(λ).

8.2 ΈΛΕΓΧΟΣ ΑΠΟΘΕΜΑΤΩΝ (ΔΥΟ ΠΡΟΪΟΝΤΑ)

Εφαρμόσαμε τους αλγορίθμους SARSA, SARSA(λ), Q και Q(λ) για την περίπτωση του πιο σύνθετου προβλήματος ελέγχου αποθεμάτων για δύο προϊόντα για το οποίο δεν είναι γενικά γνωστή η δομή της βέλτιστης πολιτικής. Έχουμε μία αποθήκη πεπερασμένης χωρητικότητας MAX, όπου αποθηκεύουμε δύο προϊόντα με σκοπό να διατεθούν στην αγορά. Στην αρχή κάθε χρονικής περιόδου γνωρίζουμε το απόθεμα της αποθήκης, σε πλήθος τεμαχίων των δύο προϊόντων (φυσικός αριθμός). Πρέπει να πάρουμε απόφαση στην παρούσα χρονική περίοδο για το πλήθος τεμαχίων από κάθε προϊόν που θα παράγουμε. Αμέσως μετά την παραλαβή των νέων τεμαχίων των προϊόντων, και έως το τέλος της χρονικής περιόδου διατίθεται το προϊόν στην αγορά οπότε και μειώνεται το απόθεμα της αποθήκης. Όσα τεμάχια μένουν αδιάθετα στην αγορά παραμένουν στην αποθήκη (απόθεμα την αποθήκης για την επομένη χρονική περίοδο) και για κάθε τεμάχιο υπάρχει κόστος αποθήκευσης. Ζητούμενο είναι να βρεθεί η βέλτιστη παραγωγή για τα δύο προϊόντα, ώστε να μεγιστοποιηθεί το συνολικό κέρδος.

Αξίζει να σημειωθεί πως οι διαστάσεις του προβλήματος σε σχέση με το αντίστοιχο πρόβλημα ενός προϊόντος, αυξάνεται κατά πολύ, καθώς για μία αποθήκη με χωρητικότητα 10 τεμαχίων στο πρόβλημα ενός προϊόντος έχουμε αντιστοιχούν 11^2 (καταστάσεις, αποφάσεις) ενώ στο πρόβλημα με δύο προϊόντα οι (καταστάσεις, αποφάσεις) γίνονται 11^4. Έχουμε δηλαδή για τα ζεύγη καταστάσεων - αποφάσεων, την τετράδα (απόθεμα $1^{ου}$,απόθεμα $2^{ου}$,παραγωγή $1^{ου}$, παραγωγή $2^{ου}$). Οι αντίστοιχοι αλγόριθμοι καλούνται να εκτιμήσουν τις τιμές της συνάρτησης αξιολόγησης $Q(s_1, s_2, a_1, a_2)$. Όπου s_1 είναι το απόθεμα του πρώτου προϊόντος και s_2 του δευτέρου, ενώ a_1 και a_2 οι αποφάσεις για την παραγωγή του $1^{ου}$ και $2^{ου}$ προϊόντος.

Για τους τέσσερις αλγόριθμους χρησιμοποιήθηκε για το πρόβλημα *exploration* η πολιτική *e-greedy*, όπου η παράμετρος *e* μειωνόταν καθώς αυξάνονταν οι επαναλήψεις-ανανεώσεις των αλγορίθμων. Ομοίως μειωνόταν και η παράμετρος βηματικής μάθησης *α*. Όσο για τα ίχνη προσπέλασης (*eligibility traces*) επειδή το πλήθος των καταστάσεων ακόμα και για μικρό μέγεθος αποθήκης είναι τέτοιο που είτε προκαλεί πρόβλημα μνήμης στον υπολογιστή είτε καθυστερεί πάρα πολύ την διαδικασία ανανέωσης, και λόγω της φύσης των ιχνών προσπέλασης (σε κάθε βήμα πολλαπλασιάζονται με *λ·γ<1*), λαμβάνεται κάθε φορά τέτοιο πλήθος επαναλήψεων ώστε τα ίχνη να είναι μεγαλύτερα από μία τιμή (π.χ. 0,2), και έτσι για μικρότερες τιμές όπου και οι διορθώσεις θα είναι αμελητέες να μην γίνονται υπολογισμοί. Τέλος για το πρόβλημα αναπτύχθηκαν οι αντίστοιχοι αλγόριθμοι για άπειρο χρονικό ορίζοντα.

Για το κόστος παραγωγής των 2 προϊόντων εκτός του κόστους παραγωγής ανά τεμάχιο έχουμε και για κάθε προϊόν ένα σταθερό κόστος θέσεως της γραμμής παραγωγής σε λειτουργία K_1 και K_2 αντιστοίχως. Έτσι το κόστος παραγωγής για a_1 και a_2 μονάδες προϊόντος είναι αντίστοιχα:

$$\text{κόστος}_1 = K_1 + c_1 \cdot a_1$$
$$\text{κόστος}_2 = K_2 + c_2 \cdot a_2$$

Παραθέτουμε κάποιες από τις εφαρμογές που τρέξαμε για το συγκεκριμένο πρόβλημα.

Γενικά φαίνεται πως οι βέλτιστες αποφάσεις είναι συνδυασμός των βέλτιστων πολιτικών, όπως αυτές είναι γνωστές από το πρόβλημα ενός προϊόντος, αλλά με όποιους περιορισμούς μπορεί να υπάρχουν ως προς την χωρητικότητα της αποθήκης και συγχρόνως ως προς την προτεραιότητα που θα πρέπει να δίνουμε σε ένα από τα δύο προϊόντα (λόγω μεγαλύτερου κέρδους ανά τεμάχιο).

<u>ΕΦΑΡΜΟΓΗ 1η</u>

ΜΕΓΕΘΟΣ ΑΠΟΘΗΚΗΣ	ΤΙΜΗ ΠΩΛΗΣΗΣ 1ΟΥ ΚΑΙ 2ΟΥ ΠΡΟΪΟΝΤΟΣ	ΣΤΑΘΕΡΟ ΚΟΣΤΟΣ ΠΑΡΑΓΩΓΗΣ K_1 ΚΑΙ K_2	ΚΟΣΤΟΣ ΠΑΡΑΓΩΓΗΣ c_1 ΚΑΙ c_2	ΚΟΣΤΟΣ ΑΠΟΘΗΚΕΥΣΗΣ	ΚΕΡΔΟΣ ΑΝΑ ΤΕΜΑΧΙΟ	ΖΗΤΗΣΗ
8	20	0	5	6	$15\cdot\alpha_1$	5
	10	0	5	6	$5\cdot\alpha_2$	5

Το κέρδος από την πώληση ενός τεμαχίου του 1ου προϊόντος είναι μεγαλύτερο από το κέρδος ανά τεμάχιο του 2ου προϊόντος, και όπως φαίνεται από τον παραπάνω πίνακα δεν υπάρχει σταθερό κόστος παραγωγής. Επίσης, φαίνεται και η ζήτηση που έχουμε βάλει για την συγκεκριμένη εφαρμογή που είναι 5 τεμάχια για κάθε προϊόν.

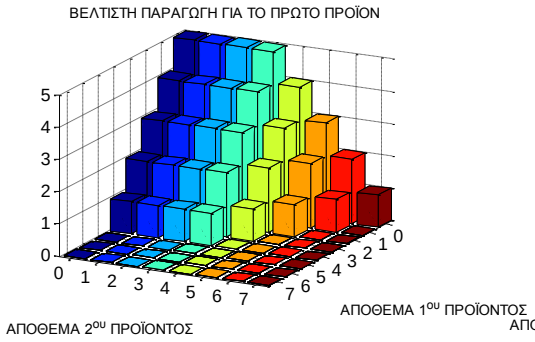
Από τα αποτελέσματα του αλγόριθμου *Q*, που εκτίμησε τις τιμές της συνάρτησης αξιολόγησης Q, έχουμε την εκτιμώμενη βέλτιστη πολιτική από τις μέγιστες τιμές ως προς την απόφαση για την παραγωγή του 1ου προϊόντος και για κάθε συνδυασμό αποθεμάτων 1ου και 2ου προϊόντος.

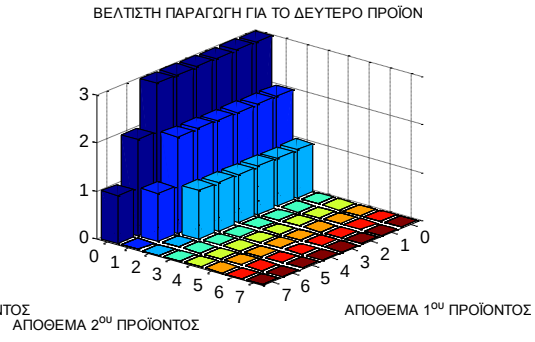
Δηλαδή ο αλγόριθμος έχει υπολογίσει τις τιμές $Q(s_1, s_2, a_1, a_2)$ για κάθε $s_1, s_2, a_1, a_2 \in \{0,1,2,\ldots,MAX\}$ και για κάθε περίπτωση όπου το απόθεμα (κατάσταση) στην αποθήκη είναι (s_1, s_2),

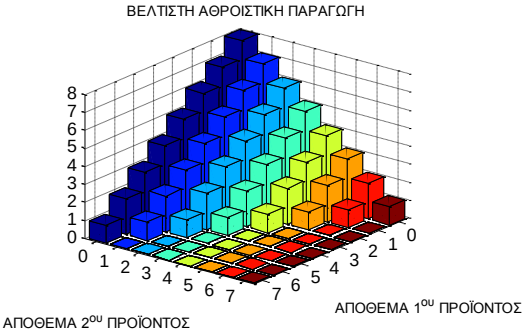
η βέλτιστη απόφαση για το 1ο προϊόν είναι $a_1^* = \max_{a_1}\{Q(s_1, s_2, a_1, a_2)\}$ και ομοίως

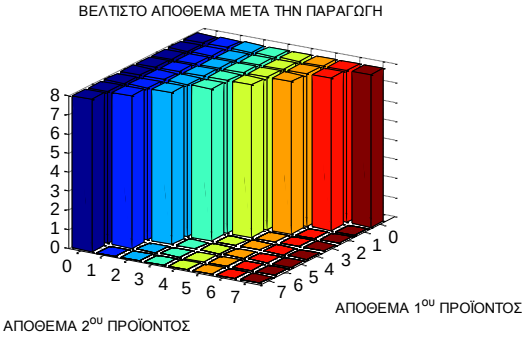
η βέλτιστη απόφαση για το 2ο προϊόν είναι $a_2^* = \max_{a_2}\{Q(s_1, s_2, a_1, a_2)\}$.

Οι τιμές για τα παραπάνω δεδομένα από τον **αλγόριθμο Q** φαίνονται στα παρακάτω διαγράμματα, όπου το ύψος κάθε μπάρας δείχνει την βέλτιστη ποσότητα παραγωγής του αντίστοιχου προϊόντος. Συγκεκριμένα, στα 2 πρώτα φαίνεται η βέλτιστη παραγωγή για το 1ο και 2ο προϊόν και για κάθε συνδυασμό αποθεμάτων, ενώ στο 3ο φαίνεται η αθροιστική παραγωγή των 2 προϊόντων για κάθε συνδυασμό αποθεμάτων και στο 4ο διάγραμμα φαίνεται η πληρότητα της αποθήκης πριν την διάθεση των προϊόντων στην αγορά.









Προκύπτει λοιπόν, ως πολιτική για το συγκεκριμένο παράδειγμα ότι θα πρέπει για το $1^{ο}$ προϊόν να παράγουμε τόση ποσότητα όση χρειάζεται για να συμπληρωθούν 5 τεμάχια στην αποθήκη (όση και η ζήτηση), ενώ για το $2^{ο}$ προϊόν πρέπει να παραχθεί τόση ποσότητα όση χρειάζεται για να συμπληρωθούν 3 τεμάχια. Βέβαια, με τον περιορισμό πως υπάρχει ο απαιτούμενος χώρος στην αποθήκη, γιατί όπως παρατηρείται “στις άκρες” του διαγράμματος για το $1^{ο}$ και $2^{ο}$ προϊόν, η παραγωγή πέφτει σταδιακά έως το μηδέν. Ακόμα, για το $2^{ο}$ προϊόν η παραγωγή δεν είναι όση απαιτεί η ζήτηση γιατί το $1^{ο}$ προϊόν δίνει μεγαλύτερο κέρδος και έτσι αφού ικανοποιηθεί η ζήτησα του $1^{ου}$ προϊόντος παράγεται τόση ποσότητα του $2^{ου}$ προϊόντος ώστε να γεμίσει η αποθήκη.

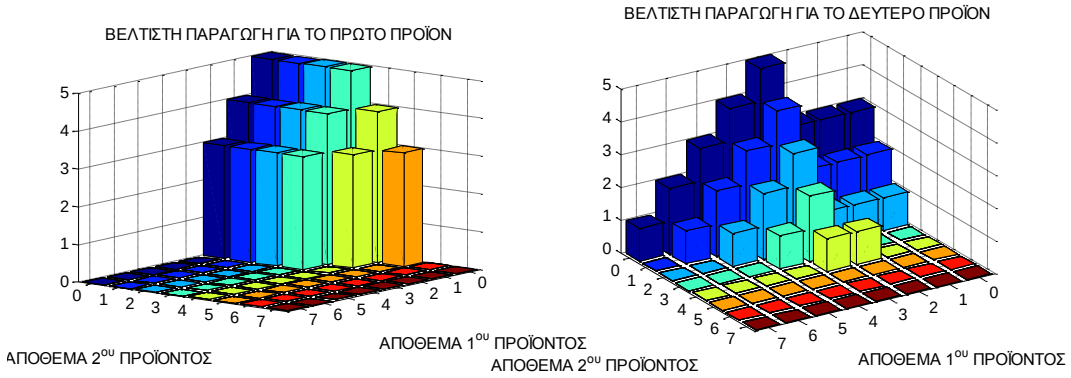
ΕΦΑΡΜΟΓΗ $2^{η}$ και παραλλαγές

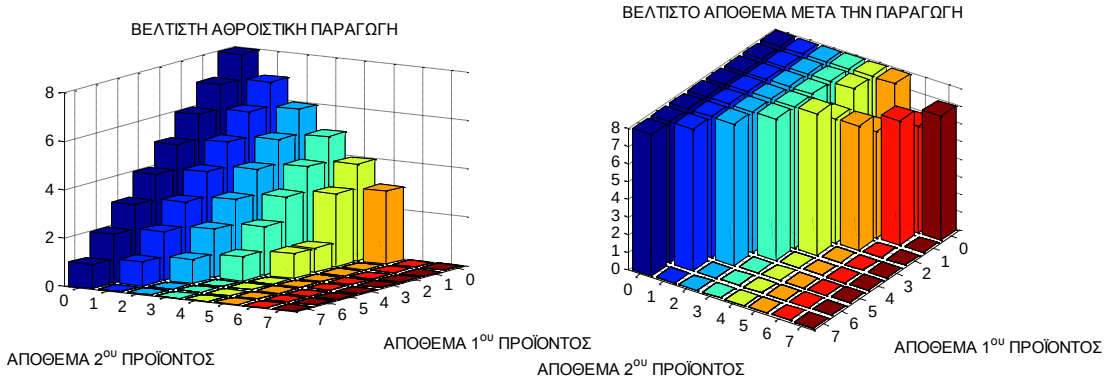
Σε μια δεύτερη εφαρμογή ορίσαμε τα ίδια δεδομένα όπως στην πρώτη εφαρμογή με μόνη διαφορά το σταθερό κόστος παραγωγής για το πρώτο προϊόν, όπου το θέσαμε 30.

ΜΕΓΕΘΟΣ ΑΠΟΘΗΚΗΣ	ΤΙΜΗ ΠΩΛΗΣΗΣ $1^{ΟΥ}$ ΚΑΙ $2^{ΟΥ}$ ΠΡΟΪΟΝΤΟΣ	ΣΤΑΘΕΡΟ ΚΟΣΤΟΣ ΠΑΡΑΓΩΓΗΣ K_1 ΚΑΙ K_2	ΚΟΣΤΟΣ ΠΑΡΑΓΩΓΗΣ c_1 ΚΑΙ c_2	ΚΟΣΤΟΣ ΑΠΟΘΗΚΕΥΣΗΣ	ΚΕΡΔΟΣ ΑΝΑ ΤΕΜΑΧΙΟ	ΖΗΤΗΣΗ
8	20	30	5	6	$15\cdot\alpha_1-30$	5
	10	0	5	6	$5\cdot\alpha_2$	5

Το κέρδος από την πώληση ενός τεμαχίου του $1^{ου}$ προϊόντος είναι μεγαλύτερο από το κέρδος ανά τεμάχιο του $2^{ου}$ προϊόντος, αλλά για παραγωγή μεγαλύτερη των 2 τεμαχίων.

Από τα αποτελέσματα του **αλγόριθμου *Q***, ομοίως με την πρώτη εφαρμογή φαίνονται στα παρακάτω διαγράμματα.





Η πολιτική που προκύπτει για το πρώτο προϊόν είναι της μορφής (3,5), δηλαδή τέτοια ώστε όταν έχω μικρότερο απόθεμα από 3 τεμάχια να παράγω τόση ποσότητα ώστε να συμπληρωθούν 5 τεμάχια στην αποθήκη (και βέβαια περιοριζόμενη συγχρόνως και από την κενή διαθέσιμη ποσότητα της αποθήκης). Ενώ για το δεύτερο προϊόν, η πολιτική είναι να συμπληρώνω πάντα το απόθεμα έως τα 5 τεμάχια αλλά περιοριζόμενη πάντα από δύο παράγοντες. Πρώτον από την κενή διαθέσιμη ποσότητα της αποθήκης, και δεύτερον από την προτεραιότητα της παραγωγής του πρώτου προϊόντος.

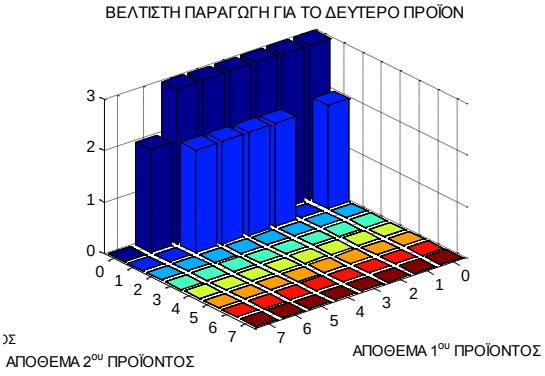
Η πολιτική για την παραγωγή των δύο προϊόντων είναι και εδώ συνδυασμός των πολιτικών που υπάρχουν για το ένα προϊόν, με τους αντίστοιχους περιορισμούς βέβαια, είτε φυσικούς (χωρητικότητα) είτε προτεραιότητας κάποιου εκ των δύο προϊόντων.

Από την εφαρμογή του **αλγόριθμου Q(λ)** προέκυψαν παρόμοια αποτελέσματα με ελάχιστα λάθη, ειδικά όταν η παράμετρος λ ήταν μεγαλύτερη του 0,7. Αλλά από το διάγραμμα γενικότερα διαφαίνεται η πολιτική αν παραβλέψουμε τα λάθη αυτά. Θα αναφερθούμε και παρακάτω σε αυτό. Επίσης, **σημαντικό συμπέρασμα ήταν από την εφαρμογή των αλγορίθμων SARSA και SARSA(λ), ότι δεν απέδωσαν καθόλου καλά**, καθώς δεν συνέκλιναν σε κάποια λύση ακόμα και μετά από πολλές επαναλήψεις και δοκιμές των τιμών των παραμέτρων e,α και λ αλλά και εναλλακτικούς συνδυασμούς αυτών.

Τα προαναφερθέντα λάθη που παρουσίαζε ο αλγόριθμος Q(λ) ήταν ελάχιστα και μάλιστα με κατάλληλη επιλογή των παραμέτρων εξαλείφονταν. Ένα παράδειγμα φαίνεται παρακάτω, όπου έχουμε θέσει πάλι σταθερή ζήτηση (5,5) για τα 2 προϊόντα:

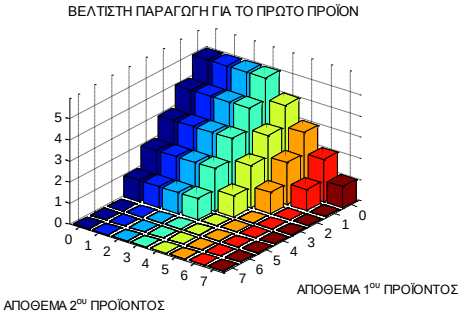
ΜΕΓΕΘΟΣ ΑΠΟΘΗΚΗΣ	ΤΙΜΗ ΠΩΛΗΣΗΣ $1^{ΟΥ}$ ΚΑΙ $2^{ΟΥ}$ ΠΡΟΪΟΝΤΟΣ	ΣΤΑΘΕΡΟ ΚΟΣΤΟΣ ΠΑΡΑΓΩΓΗΣ K_1 ΚΑΙ K_2	ΚΟΣΤΟΣ ΠΑΡΑΓΩΓΗΣ c_1 ΚΑΙ c_2	ΚΟΣΤΟΣ ΑΠΟΘΗΚΕΥΣΗΣ	ΚΕΡΔΟΣ ΑΝΑ ΤΕΜΑΧΙΟ	ΖΗΤΗΣΗ
8	20	0	5	6	$15\cdot\alpha_1$	5
	10	8	5	6	$5\cdot\alpha_2$-8	5

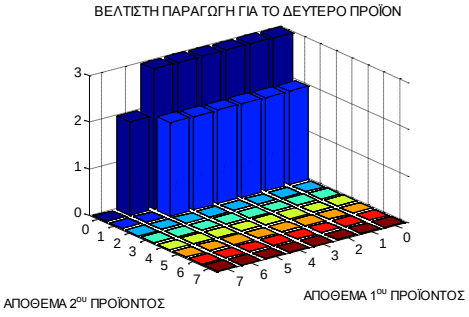
Όπου για τον **Q(λ)** αλγόριθμο με τιμή 0,9 για την παράμετρο λ έχουμε για τις βέλτιστες αποφάσεις του $2^{ου}$ προϊόντος το διάγραμμα :

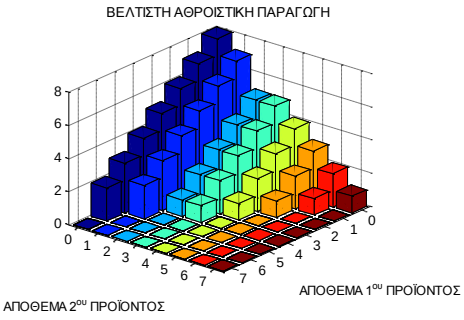


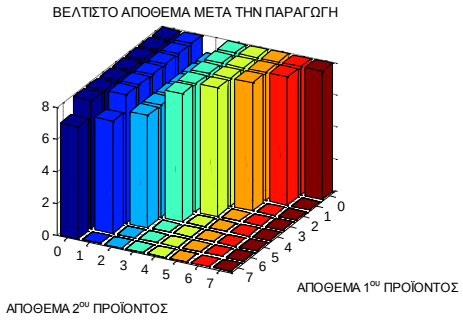
όπου φαίνεται πως για έναν συνδυασμό αποθεμάτων (την κατάσταση (1,1), δηλαδή ένα τεμάχιο από κάθε προϊόν), η βέλτιστη απόφαση είναι 0, ενώ οι τιμές για συνδυασμούς παρόμοιους (διπλανές “μπάρες” στο διάγραμμα) είναι 2 τεμάχια. Φαίνεται λοιπόν, πως η σωστή τιμή θα πρέπει να είναι 2. Τέτοια λάθη μπορούν να παραλειφθούν αν διαβλέπουμε ένα συγκεκριμένο μοτίβο στα διαγράμματα.

Η τιμή 2 διορθώνεται από τον αλγόριθμο Q(λ) όταν θέσουμε το λ=0,7 (ή και μικρότερο). Αυτό φαίνεται στα διαγράμματα:





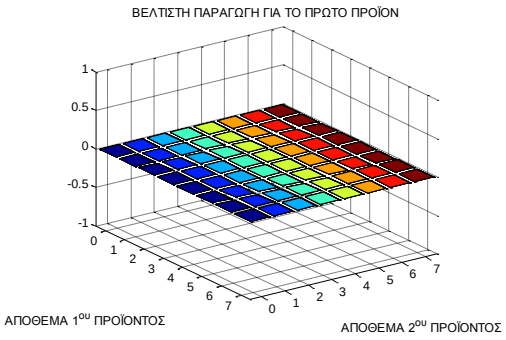


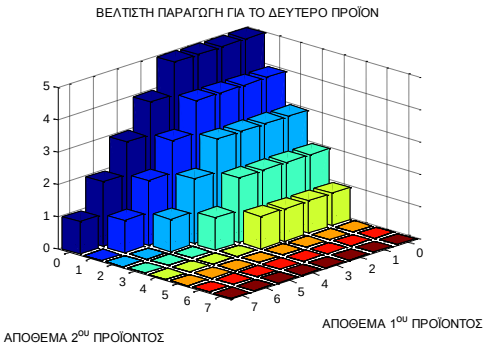


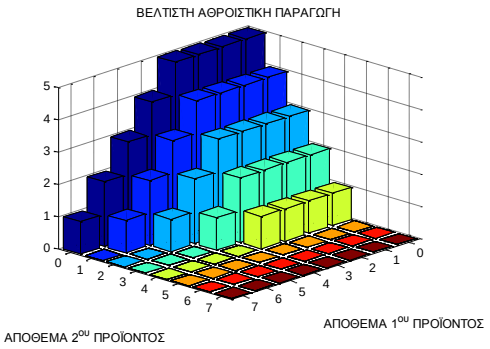
Τα διαγράμματα από τον αλγόριθμο Q είναι ακριβώς τα ίδια, με τα παραπάνω, κάτι που δείχνει πως μάλλον ο αλγόριθμος Q δουλεύει καλύτερα από τον Q(λ) και σίγουρα καλύτερα από τους SARSA αλγορίθμους.

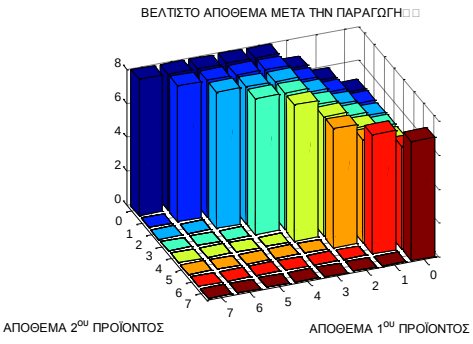
Ακόμα θα αναφέρουμε και ένα όπου ο αλγόριθμος έδειξε πως το ένα από τα δύο προϊόντα δεν συμφέρει να το παράγουμε. Τα δεδομένα του συγκεκριμένου παραδείγματος ήταν τα παρακάτω με ζήτηση (5,5) σταθερή:

ΜΕΓΕΘΟΣ ΑΠΟΘΗΚΗΣ	ΤΙΜΗ ΠΩΛΗΣΗΣ $1^{ΟΥ}$ ΚΑΙ $2^{ΟΥ}$ ΠΡΟΪΟΝΤΟΣ	ΣΤΑΘΕΡΟ ΚΟΣΤΟΣ ΠΑΡΑΓΩΓΗΣ K_1 ΚΑΙ K_2	ΚΟΣΤΟΣ ΠΑΡΑΓΩΓΗΣ c_1 ΚΑΙ c_2	ΚΟΣΤΟΣ ΑΠΟΘΗΚΕΥΣΗΣ	ΚΕΡΔΟΣ ΑΝΑ ΤΕΜΑΧΙΟ	ΖΗΤΗΣΗ
8	20	0	5	6	$15 \cdot \alpha_1$	5
	10	8	5	6	$5 \cdot \alpha_2 - 8$	5









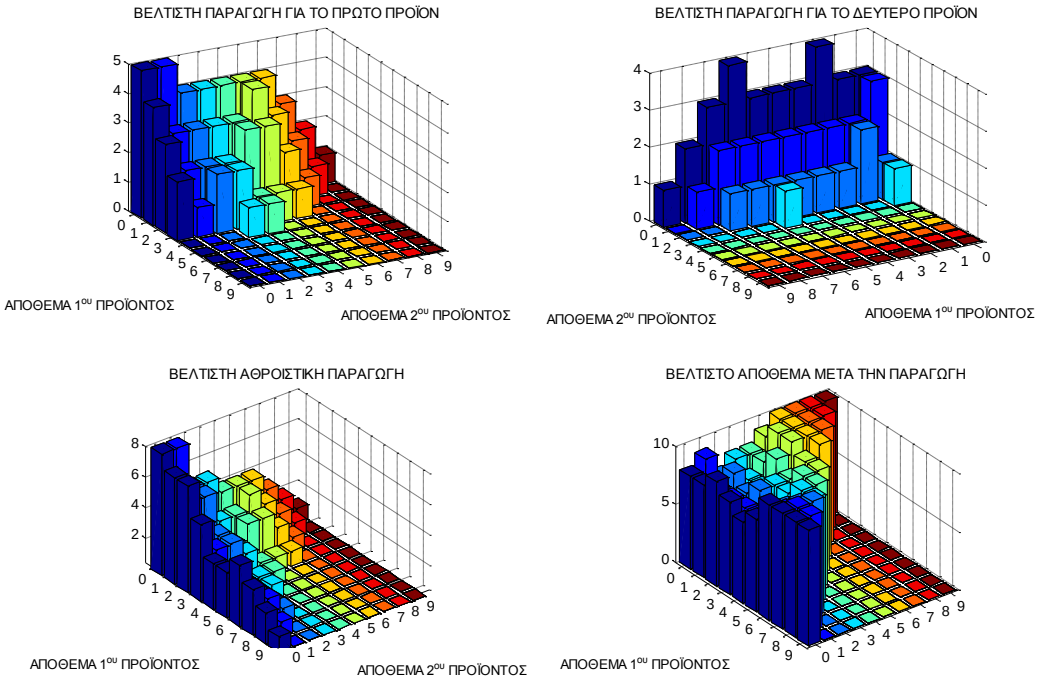
Η πολιτική εδώ είναι να μην παραχθεί καθόλου το $1^{ο}$ προϊόν, ενώ το δεύτερο να παράγεται έτσι ώστε να συμπληρώνεται έως το αριθμό των 5 τεμαχίων και βέβαια υπό όποιον φυσικό περιορισμό για την χωρητικότητα της αποθήκης.

ΕΦΑΡΜΟΓΗ 3^{η}

Ως μια 3^{η} εφαρμογή πήραμε την ζήτηση να είναι τυχαία μεταβλητή. Θέσαμε την ζήτηση, μέσω προσομοίωσης να είναι 3,4 ή και 5 τεμάχια για το πρώτο προϊόν και 3 ή 4 τεμάχια για το δεύτερο. Για τις υπόλοιπες παραμέτρους θέσαμε για την εφαρμογή:

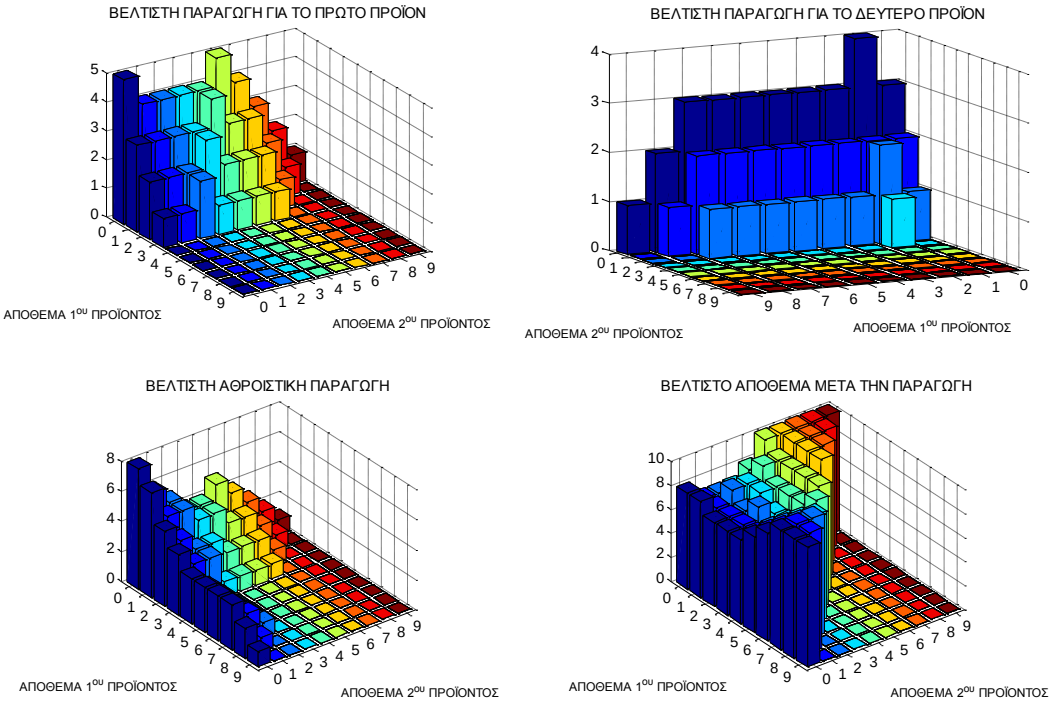
ΜΕΓΕΘΟΣ ΑΠΟΘΗΚΗΣ	ΤΙΜΗ ΠΩΛΗΣΗΣ 1^{ΟΥ} ΚΑΙ 2^{ΟΥ} ΠΡΟΪΟΝΤΟΣ	ΣΤΑΘΕΡΟ ΚΟΣΤΟΣ ΠΑΡΑΓΩΓΗΣ K_1 ΚΑΙ K_2	ΚΟΣΤΟΣ ΠΑΡΑΓΩΓΗΣ c_1 ΚΑΙ c_2	ΚΟΣΤΟΣ ΑΠΟΘΗΚΕΥΣΗΣ	ΚΕΡΔΟΣ ΑΝΑ ΤΕΜΑΧΙΟ	ΖΗΤΗΣΗ Ισοπίθανα από τα σύνολα
10	20	0	5	6	$15\cdot\alpha_1$	{3,4,5}
	10	0	5	6	$5\cdot\alpha_2$	{3,4}

Ο αλγόριθμος Q(λ) έδωσε για λ=0,4 τα παρακάτω διαγράμματα. Φαίνεται σε αυτά (όχι ξεκάθαρα, πιθανόν λόγω τις τυχαίας ζήτησης, ή και του παράγοντα λ), ότι μια διαφαινόμενη πολιτική είναι τέτοια ώστε να παράγει τόσα τεμάχια ώστε πριν την διάθεση του προϊόντος στην αγορά να υπάρχουν στην αποθήκη 4 ή και 5 τεμάχια από το πρώτο προϊόν (μέση ζήτηση 4 τεμάχια), ενώ από το δεύτερο τέτοια παραγωγή ώστε να υπάρχουν στην αποθήκη 3 ή και τέσσερα (σε λιγότερες περιπτώσεις) τεμάχια του δεύτερου προϊόντος.



Ο λόγος που για το πρώτο προϊόν έχουμε αυτήν την εναλλακτική (4 ή και 5 τεμάχια), έναντι του δευτέρου (και εδώ εμφανίζεται η εναλλακτική αλλά σαφώς σε λιγότερες περιπτώσεις, σε 6 μόνο από όλες τις δυνατές περιπτώσεις όπως φαίνεται από το 2^{ο} διάγραμμα, έναντι 12 όπως φαίνονται από το 1^{ο} διάγραμμα), είναι λογικά γιατί το κέρδος ανά μονάδα προϊόντος είναι αρκετά μεγαλύτερο για το 1^{ο} αντί για το 2^{ο} προϊόν.

Ο αλγόριθμος Q έδωσε για την ίδια εφαρμογή τα παρακάτω διαγράμματα:



Όμοια πολιτική διαφαίνεται από τα διαγράμματα αυτά του αλγόριθμου Q, όπως και από τον αλγόριθμο Q(λ) προηγουμένως.

<u>Κριτική των αλγορίθμων</u>

Από τις εφαρμογές των τεσσάρων αλγορίθμων στα διάφορα προβλήματα που δοκιμάσαμε, γενικότερο συμπέρασμα είναι πως οι αλγόριθμοι SARSA και SARSA(λ) δεν είναι τόσο καλοί όσο είναι οι Q και Q(λ). Όπως έχουμε αναφέρει οι αλγόριθμοι SARSA και SARSA(λ) λαμβάνουν, είτε από την άμεση αλληλεπίδραση με το περιβάλλον, είτε μέσω προσομοίωσης «τυχαίες» αμοιβές και λαμβάνουν e-greedy πολιτικές (για τις εφαρμογές που κάναμε), για να εκτιμήσουν («μάθουν»), μέσω των διαδοχικών ανανεώσεων, τις τιμές της συνάρτησης αξιολόγησης. Έτσι λοιπόν, σε κάθε κατάσταση που βρίσκεται ο «μαθητής» λαμβάνει μια e-τυχαία απόφαση και μεταπηδά σε μια νέα κατάσταση όπου θα πάρει μια νέα e-τυχαία απόφαση, όπου βάσει αυτών θα κάνει και την ανανέωση των αντίστοιχων τιμών της πρώτης κατάστασης. Αυτή ή ελευθερία σε δύο βήματα και κυρίως στο $2^{ο}$ βήμα (καθώς αποδίδει τυχαία επόμενη κατάσταση-απόφαση στην προηγούμενη), είναι που καθυστερεί την σύγκλιση του αλγόριθμου, και πιθανόν να απαιτεί ειδικές τρόπο για την μείωση των παραμέτρων e και α , αναλόγως του προβλήματος, ώστε να συγκλίνει ο αλγόριθμος. Το πρόβλημα αυτό γίνεται εντονότερο στο πρόβλημα ελέγχου αποθεμάτων δύο προϊόντων, λόγω του πολύ μεγαλύτερου πλήθους καταστάσεων που υπάρχουν στο πρόβλημα αυτό σε σχέση με το αντίστοιχο πρόβλημα ενός προϊόντος. Όπως φαίνεται και από τα διαγράμματα της προηγουμένης ενότητας οι αλγόριθμοι SARSA μπορεί να

συγκλίνουν στο πρόβλημα αποθεμάτων 1 προϊόντος, αλλά σε λάθος απόφαση για κάποιες περιπτώσεις (ενδέχεται ακόμα και αν φαίνεται πως έχει συγκλίνει ο αλγόριθμος, πολύ αργά –μετά από πάρα πολλές επαναλήψεις- να αλλάζει η βέλτιστη απόφαση, αλλά αυτό δεν είναι καθόλου εύκολο να φανεί).

Αντιθέτως, οι αλγόριθμοι Q και Q(λ), βασίζουν την ανανέωση των τιμών που κάνουν για την τωρινή κατάσταση-απόφαση, στην βέλτιστη απόφαση της επομένης κατάστασης και αυτό είναι που κάνει τον αλγόριθμο να είναι πολύ πιο αποδοτικός από τους SARSA αλγορίθμους. Ακόμα και όταν αλλάζει κάποια τιμή όπου με την σειρά της αλλάζει την βέλτιστη απόφαση, αυτό θα επηρεάζει όλες τις καταστάσεις-αποφάσεις που θα οδηγούν σε αυτήν την νέα βέλτιστη απόφαση, οπότε οι αντίστοιχες διορθώσεις θα είναι «μαζικές» και όχι λόγω τυχαίας προσπέλασης αυτής της κατάστασης-απόφασης.

Τέλος, για την απόδοση των αλγορίθμων Q ή SARSA σε σχέση με τους αντίστοιχους Q(λ) και SARSA(λ), όπως έχουμε αναφέρει και στην προηγουμένη ενότητα, φαίνεται να αποδίδουν καλύτερα οι SARSA και Q, αλλά αυτό μπορεί να αποδίδεται στην φύση των προβλημάτων που εφαρμόσαμε! Η ουσία της παραμέτρου λ είναι ότι μέσω αυτής μπορεί να αποδίδεται κάποια αλλαγή μίας κατάστασης, όχι στην προηγουμένη, αλλά σε όλες τις προηγούμενες καταστάσεις-αποφάσεις που προηγήθηκαν αυτής (σε μικρότερο βαθμό βέβαια όσο παλαιότερα συναντάμε τις άλλες καταστάσεις-αποφάσεις). Έχουμε , δηλαδή, το πρόβλημα της καθυστερημένης αμοιβής το οποίο προσπαθεί να αποδοθεί μέσω της παραμέτρου λ.

Μια δεύτερη θεώρηση της χρήσης της παραμέτρου λ, είναι απλά πως οι διορθώσεις που γίνονται σε κάθε βήμα, θα πρέπει αναδρομικώς να αφορούν και καταστάσεις-αποφάσεις που προηγήθηκαν της τωρινής και της επομένης. Με αυτόν τον τρόπο, και μέσω των πολλών «διαδρομών» που κάνουμε μέσω της προσομοίωσης ή της επανάληψης του «πειράματος μάθησης», θα έχουμε τελικώς μια πιο γρήγορη ανανέωση των τιμών της συνάρτησης αξιολόγησης και συνεπώς γρηγορότερη σύγκλιση του αλγορίθμου.

Ένα μειονέκτημα της δεύτερης αυτής θεώρησης, πιθανόν να είναι η πολύ μικρή διαφορά στις τιμές σύγκλισης ή η στενή σχέση πολλών από τις καταστάσεις μεταξύ τους. Ειδικά το τελευταίο είναι πολύ πιθανότερο αίτιο της απόδοσης λανθασμένων ανανεώσεων στις τιμές της συνάρτησης αξιολόγησης. Ακόμα η επανάληψη στην διαδοχή των ίδιων καταστάσεων (ακόμα και περιοδικώς), ενδέχεται να προκαλεί ανανεώσεις που βασίζονται ουσιαστικά στην ίδια την κατάσταση ή κατάσταση-απόφαση. Γενικότερα, η παράμετρος λ δεν πρέπει να είναι κοντά στο 1 γιατί προκαλεί λανθασμένες ανανεώσεις, σε καταστάσεις που δεν σχετίζονται με την «πηγή» της ανανέωσης.

Τέλος, για την κριτική της απόδοσης των αλγορίθμων πρέπει να αναφέρουμε πως οι τιμές των παραμέτρων e και α, είναι σημαντικότατες. Η παράμετρος μάθησης α, θα μπορούσε να είναι από την αρχή του αλγορίθμου μικρή, χωρίς ιδιαίτερο πρόβλημα (θα καθυστερούσε ελαφρώς χρονικά την σύγκλιση). Αλλά, η παράμετρος e (θυμίζουμε είναι η πιθανότητα με την οποία δεν θα λάβουμε σε κάθε βήμα την βέλτιστη έως τότε απόφαση),

είναι αυτή που καθορίζει πόσο καλά θα γίνει *exploration*, δηλαδή κατά πόσο θα «ψάξουμε» κατά την μάθηση για νέες, ενδεχομένως καλύτερες καταστάσεις-αποφάσεις. Αυτό είναι αναγκαίο να γίνει αλλά επίσης αναγκαίο για την σύγκλιση είναι, πως θα πρέπει να σταματήσει να γίνεται (έστω και να μειωθεί σημαντικά) *exploration*, αφού έχουν προσπελαθεί όλες οι καταστάσεις-αποφάσεις μεγάλο αριθμό , ώστε να είναι οι τελικές τιμές της συνάρτησης αξιολόγησης αξιόπιστες. Αυτό θα πρέπει να υπολογιστεί για κάθε πρόβλημα προσεκτικά. Για τα δύο προβλήματα που θεωρήσαμε, έλεγχος αποθεμάτων για 1 και 2 προϊόντα, η μείωση της τιμής της παραμέτρου e ήταν πολύ πιο αργή στο $2^{ο}$ πρόβλημα καθώς οι καταστάσεις εκεί ήταν σημαντικά περισσότερες.

<u>ΠΑΡΑΤΗΡΗΣΕΙΣ</u>

Για περισσότερο πολύπλοκα προβλήματα θα μπορούσε να γίνει συνδυασμός των μεθόδων *Monte Carlo* και των αλγορίθμων Q ή/και SARSA , όπου με προσεκτική επιλογή βασικών καταστάσεων θα μπορούσε η συνάρτηση αξιολόγησης για αυτές να βρεθεί μέσω της μεθόδου *Monte Carlo* (όπου για μεμονωμένες καταστάσεις είναι αρκετά συντομότερη), και για τις υπόλοιπες καταστάσεις να χρησιμοποιηθούν οι Q / SARSA ή και οι αντίστοιχοι με την παράμετρο λ.

Σε κάθε περίπτωση οι αλγόριθμοι που χρησιμοποιήσαμε είναι αρκετά γρήγοροι καθώς απλά ανανεώνουν τις τιμές κάποιου/κάποιων πίνακα/πινάκων. Δυσκολία εμφανίζεται μόνο στην περίπτωση του πεπερασμένου ορίζοντα (υπάρχει πρόβλημα μνήμης) και βέβαια στην περίπτωση όπου έχουμε πολύ μεγάλο πλήθος καταστάσεων, όπου το πρόβλημα είναι και μνήμης αλλά και προσπέλασης όλων των καταστάσεων αρκετές φορές.

ΠΑΡΑΡΤΗΜΑ 1 - Αλγόριθμοι MATLAB

ΓΙΑ ΤΟ ΑΠΛΟ ΝΕΥΡΟΝΙΟ

```
% GIA THN EYTHEIA y=ax+b.'H W1*X+W2*Y=W3-> -W3+W1*X+W2*Y=0 ->
% [W3,W1,W2]*[-1,X,Y]'=0
hold
x1=[-1;-1;-3.2];          % GIA TO 1o SHMEIO (-1,-3.2)
x2=[-1;3;8.5];            % GIA TO 2o SHMEIO (3,8.5)
x3=[-1;1.2;4.5];          % GIA TO 3o SHMEIO (1.2,4.5)
x=[x1 x2 x3];             % TO SYNOLO TWN EPITHYMITWN SHMEIWN
d=[1 1 -1];               % EPITHYMITH EXODOS GIA TA 3 SHMEIA
                          % XARAKTHRISMOS TWN SHMEIWN WS PROS TO HMIEPIPEDO
                          % POY THELOYME NA ANHKOYN : 1 GIA TO ENA HMIEPIPEDO
                          % -1 GIA TO 2o HMIEPIPEDO, WS PROS THN EYTHEIA
g=0.01;                   % SYNTELESTHS MATHHSHS
w=randn(3,1)*0.1;         % TYXAIA ARXIKA BARH, PERIPOY MHDEN
y=sign(w'*x);             % EXODOS NEYRONIOY +1 'H -1 GIA TA 3 SHMEIA
xlim([min(x(2,:))-1,max(x(2,:))+1])
ylim([min(x(3,:))-1,max(x(3,:))+1])
plot(x(2,:),x(3,:),'r.','MarkerSize',16)% TA 3 DOSMENA SHMEIA ME KOKKINO
t=1-(min(d==y));          % DEIKTRIA "1=OXI, AN EINAI SWSTH H EYTHEIA"
                          % SWSTH SHMAINEI NA DIAXWRIZEI TO EPIPEDO, WSTE TA
                          % SHMEIA NA ANHKOYN STO EPITHYMITO HMIEPIPEDO
xmin=min(x(2,:))-1;   xmax=max(x(2,:))+1;
number=0;
N=900;  ab=zeros(N,2);  % GIA TIS TELEYTAIES N EYTHEIES
while t
i=mod(number,N)+1;
w=w+g*(d(1)-y(1))*x(:,1);  % DIORTHWSH GIA TA BARH LOGW TOY 1ou SHMEIOY
w=w+g*(d(2)-y(2))*x(:,2);  % DIORTHWSH GIA TA BARH LOGW TOY 2ou SHMEIOY
w=w+g*(d(3)-y(3))*x(:,3);  % DIORTHWSH GIA TA BARH LOGW TOY 3ou SHMEIOY
y=sign(w'*x);              % NEA EXODOS NEYRONIOY
ab(i,:)=[-w(2)/w(3),w(1)/w(3)];          % [a,b] THS EUTHEIAS y=ax+b
t=1-(min(d==y));                         % LATHOS
number=number+1;                         % KATAMETRHSH PROSPATHEIWN
end
for i=1:N
    plot([xmin,xmax],[ab(i,1)*xmin+ab(i,2),ab(i,1)*xmax+ab(i,2)])
                                         % PLOT THS EYTHEIAS TOY NEYRONIOY
end
plot(x(2,:),x(3,:),'r.','MarkerSize',16)% TA 3 DOSMENA SHMEIA ME KOKKINO
AB=[-w(2)/w(3),w(1)/w(3)];                % [a,b] THS EUTHEIAS y=ax+b
plot([xmin,xmax],[AB(1)*xmin+AB(2),AB(1)*xmax+AB(2)],'r-','LineWidth',2)
                                         % TELIKA SWSTH EYTHEIA ME KOKKINO
AB
```

ΠΑΡΑΡΤΗΜΑ 2 - Αλγόριθμοι MATLAB

ΓΙΑ ΤΟ ΠΡΟΒΛΗΜΑ ΤΩΝ 10Χ10 ΤΕΤΡΑΓΩΝΩΝ

```
function [ k ] = Ekato(i,j,n)
%ΕΠΙΣΤΡΕΦΕΙ ΤΟ ΣΤΟΙΧΕΙΟ ΤΟΥ ΠΙΝΑΚΑ nxn ΠΟΥ ΑΝΤΙΣΤΟΙΧΕΙ ΣΤΗΝ i,j ΘΕΣΗ
e=1-(i>0)*(i<(n+1))*(j>0)*(j<(n+1));
if e
    k=0; %'ΛΑΘΟΣ ΣΤΟΙΧΕΙΟ'
 else
    k=(i-1)*n+j;
end
```

```
function [ G ] = NxN( n )
%ΓΙΑ ΠΙΝΑΚ nxn ΣΤΟΙΧΕΙΩΝ ΕΠΙΣΤΡΕΦΕΙ ΣΤΟΝ G ΤΑ ΤΕΤΡΑΓΩΝΑ ΜΕ ΤΑ ΟΠΟΙΑ
%ΓΕΙΤΝΙΑΖΕΙ (ΜΠΟΡΕΙ ΝΑ ΕΠΙΚΟΙΝΩΝΕΙ)
A=zeros(n);  G=zeros(n*n,8);
for i=1:n
for j=1:n
   G(Ekato(i,j,n),:)=[Ekato(i,j-3,n)*(j-
3>0),Ekato(i,j+3,n)*(j+3<(n+1)),Ekato(i-3,j,n)*(i-
3>0),Ekato(i+3,j,n)*(i+3<(n+1)),Ekato(i-2,j-2,n)*(i-2>0)*(j-2>0),Ekato(i-
2,j+2,n)*(i-2>0)*(j+2<(n+1)),Ekato(i+2,j-2,n)*(i+2<(n+1))*(j-
2>0),Ekato(i+2,j+2,n)*(i+2<(n+1))*(j+2<(n+1))];
end
end
end
```

```
function [ tora,A ] = peripatos_10x10()
%ΠΕΡΙΠΑΤΟΣ 10x10 ΤΕΡΤΑΓΩΝΟΥ, ΞΕΚΙΝΩΝΤΑΣ ΑΠΟ ΤΗΝ ΠΑΝΩ ΑΡΙΣΤΕΡΗ ΓΩΝΙΑ A(1,1)
A=zeros(1,100);
tora=1;             %tora ΕΙΝΑΙ Ο ΑΡΙΘΜΟΣ ΠΟΥ ΘΑ ΤΟΠΟΘΕΤΗΘΕΙ
Geitones=NxN(10);
thesi=1;            %Ο ΑΡΙΘΜΟΣ "tora" ΘΑ ΤΟΠΟΘΕΤΗΘΕΙ...
A(thesi)=tora;      %... ΣΤΗΝ "thesi" (i,j) ΤΟΥ ΠΠΙΝΑΚΑ A
Geitones(Geitones==thesi)=0;
error=1;
while error
    %tora=tora+1;
    k=sum(Geitones(thesi,:)>0); %k ΔΥΝΑΤΕΣ ΕΠΙΛΟΓΕΣ ΜΕΤΑΒΑΣΗΣ
    if k>0
        epilogi=dunirand1(k);   %ΣΕ ΠΟΙΑ ΑΠΟ ΤΙΣ k ΕΠΙΛΟΓΕΣ ΜΕΤΑΠΗΔΩ
        vec=Geitones(thesi,:);  %ΤΟ ΔΙΑΝΥΣΜΑ ΤΩΝ ΓΕΙΤΩΝΩΝ
        vec(vec==0)=[];         %  "   "  ΧΩΡΙΣ ΤΑ ΜΗΔΕΝΙΚΑ
        thesi=vec(epilogi);     %ΝΕΑ ΘΕΣΗ ΣΤΟΝ ΠΙΝΑΚΑ
        Geitones(Geitones==thesi)=0;
        tora=tora+1;            %Ο ΕΠΟΜΕΝΟΣ ΑΡΙΘΜΟΣ ΠΡΟΣ ΤΟΠΟΘΕΤΗΣΗ
        A(thesi)=tora;          %ΤΟΠΟΘΕΤΗΣΗ ΤΟΥ ΑΡΙΘΜΟΥ tora
    else
        error=0;
    end
end
end
```

```
% ΟΙ ΕΝΤΟΛΕΣ ΜΕ ΤΙΣ ΟΠΟΙΕΣ ΤΡΕΧΟΥΜΕ ΤΟΝ «ΠΕΡΙΠΑΤΟ» ΕΩΣ ΟΤΟΥ ΤΟΠΟΘΕΤΗΘΕΙ
% ΚΑΠΟΙΟΣ ΑΡΙΘΜΟΣ ΜΕΓΑΛΥΤΕΡΟΣ ΤΟΥ …99 (ΨΑΧΝΩ ΛΥΣΗ… ΤΟΠΟΘΕΤΗΣΗ ΚΑΙ ΤΟΥ 100)
T=1; i=0; f=1; while T<=99
[T,A]=peripatos_10x10;i=i+1;
if (i/10000>=f)   i      % ΤΥΠΩΝΕΙ ΤΟ ΠΛΗΘΟΣ ΤΩΝ ΠΡΟΣΠΑΘΕΙΩΝ ΑΝΑ 10.000
f=f+1; end
end
```

```
function [ tora,a ] = peripatos_nxn(n)
%ΠΕΡΙΠΑΤΟΣ nxn ΤΕΡΤΑΓΩΝΟΥ, ΞΕΚΙΝΩΝΤΑΣ ΑΠΟ ΤΗΝ ΠΑΝΩ ΑΡΙΣΤΕΡΗ ΓΩΝΙΑ A(1,1)
A=zeros(1,n*n);
tora=1;             %tora ΕΙΝΑΙ Ο ΑΡΙΘΜΟΣ ΠΟΥ ΘΑ ΤΟΠΟΘΕΤΗΘΕΙ
Geitones=NxN(n);
thesi=1;            % Ο ΑΡΙΘΜΟΣ "tora" ΘΑ ΤΟΠΟΘΕΤΗΘΕΙ...
A(thesi)=tora;      %... ΣΤΗΝ "thesi" (i,j) ΤΟΥ ΠΠΙΝΑΚΑ A
Geitones(Geitones==thesi)=0;
error=1;
while error
    %tora=tora+1;
    k=sum(Geitones(thesi,:)>0); % k ΔΥΝΑΤΕΣ ΕΠΙΛΟΓΕΣ ΜΕΤΑΒΑΣΗΣ
    if k>0
        epilogi=dunirand1(k);    % ΣΕ ΠΟΙΑ ΑΠΟ ΤΙΣ k ΕΠΙΛΟΓΕΣ ΜΕΤΑΠΗΔΩ
        vec=Geitones(thesi,:);   % ΤΟ ΔΙΑΝΥΣΜΑ ΤΩΝ ΓΕΙΤΩΝΩΝ
        vec(vec==0)=[];          %  "   "  ΧΩΡΙΣ ΤΑ ΜΗΔΕΝΙΚΑ
        thesi=vec(epilogi);      % ΝΕΑ ΘΕΣΗ ΣΤΟΝ ΠΙΝΑΚΑ
        Geitones(Geitones==thesi)=0;
        tora=tora+1;             % Ο ΕΠΟΜΕΝΟΣ ΑΡΙΘΜΟΣ ΠΡΟΣ ΤΟΠΟΘΕΤΗΣΗ
        A(thesi)=tora;           % ΤΟΠΟΘΕΤΗΣΗ ΤΟΥ ΑΡΙΘΜΟΥ tora
    else  error=0;
    end
end
k=0;  a=zeros(n);               % ΣΤΟΝ ΠΙΝΑΚΑ a ΑΠΟΘΗΚΕΥΟΝΤΑΙ ΟΙ
for i=1:n                       % ΤΟΠΟΠΕΤΗΜΕΝΟΙ ΑΡΙΘΜΟΙ
    a(i,:)=A((k*n+1):(k*n+n));
    k=k+1;
end
end
```

```
% GIA THN PERIPTWSH nXn (n=5)
% ΟΙ ΕΝΤΟΛΕΣ ΜΕ ΤΙΣ ΟΠΟΙΕΣ ΤΡΕΧΟΥΜΕ ΤΟΝ «ΠΕΡΙΠΑΤΟ» ΕΩΣ ΟΤΟΥ ΤΟΠΟΘΕΤΗΘΕΙ
% ΚΑΠΟΙΟΣ ΑΡΙΘΜΟΣ ΜΕΓΑΛΥΤΕΡΟΣ ΤΟΥ …24 (ΨΑΧΝΩ ΛΥΣΗ… ΤΟΠΟΘΕΤΗΣΗ ΚΑΙ ΤΟΥ 25)
T=1; i=0; while T<=24
[T,A]=peripatos_nxn(5);i=i+1;
end
T
A
% GIA THN PERIPTWSH nXn (n=5)
% ΟΙ ΕΝΤΟΛΕΣ ΜΕ ΤΙΣ ΟΠΟΙΕΣ ΤΡΕΧΟΥΜΕ ΤΟΝ «ΠΕΡΙΠΑΤΟ» 1000 ΦΟΡΕΣ ΕΩΣ ΟΤΟΥ
% ΤΟΠΟΘΕΤΗΘΕΙ ΚΑΙ ΤΟ 25 ΓΙΑ ΝΑ ΒΡΕΘΕΙ Ο ΜΕΣΟΣ ΑΡΙΘΜΟΣ ΠΡΟΣΠΑΘΙΩΝ ΓΙΑ ΤΙΣ
% ΟΠΟΙΕΣ ΤΟΠΟΘΕΤΗΘΗΚΕ ΚΑΙ ΤΟ 25
fores=1000;
Ii=zeros(1,fores);
for k=1:fores
    T=1; i=0; while T<=24
    [T,A]=peripatos_nxn(5);i=i+1;
    end
    Ii(k)=i;
end
mean(Ii)    % ΑΠΟΤΕΛΕΣΜΑ ΠΕΡΙΠΟΥ 21
```

```
% GIA THN PERIPTWSH nXn (n=5): ΕΠΙΛΕΚΤΙΚΗ ΤΟΠΟΘΕΤΗΣΗ ΤΟΥ ΑΡΙΘΜΟΥ 25
    A=zeros(5); T=1; i=0; while A(4,3)~=25  % ΘΕΣΗ ΤΟΥ 25 ΣΤΟ ΚΕΛΙ (4,3)
    [T,A]=peripatos_nxn(5);i=i+1;           % i ΠΛΗΘΟΣ ΕΠΑΝΑΛΗΨΕΩΝ
    end
    A
```

ΠΑΡΑΡΤΗΜΑ 3 - Αλγόριθμοι MATLAB

ΓΙΑ ΤΟ ΠΡΟΒΛΗΜΑ ΤΟΥ ΤΑΞΙΔΙΟΥ ΠΡΟΣ ΤΟ ΧΡΥΣΑΦΙ

```
function [ epomeno ] = EPOMENO( twra )
% EPOMENH THESI STO TAXIDI PROS TO XRYSAFI
%KATASTASEIS A=1, B=2, C=3, D=4, E=5, F=6, G=I=J=0, H=1
pano=(rand>0.5);          % GIA THN PITHANOTHTA PANW 'H KATW ISOPITHANA
switch (twra)
    case 1
        epomeno=3*pano+2*(1-pano);
    case 2
        epomeno=5*pano+4*(1-pano);
    case 3
        epomeno=6*pano+5*(1-pano);
    case 4
        epomeno=pano;     % y=1 TELIKH AMOIBH/EPITYXIA TOY H
    case 5
        epomeno=(1-pano); % y=1 TELIKH AMOIBH/EPITYXIA TOY H
    otherwise
        epomeno=0;        % y=0 TELIKH AMOIBH/APOTYXIA TOY H
end
end
```

```
function [w] = TD_xrusafi(n)
% TD METHODOS GIA THN EKTIMHSH w TWN PITHANOTHTWN EYRESHS
% KATASTASEIS A=1, B=2, C=3, D=4, E=5, F=6, G=I=J=0, H=1

w=rand(1,6);                % ARXIKH TYXAIA EKTIMHSH PITHANOTHTWN EYRESHS
                            % XEKINONTAS APO KATHE MIA APO TIW 6 PRWTES
KATASTASEIS
a=0.003;                    % BHMA ANANEWSHS
s=eye(6);
for i=1:n                   % EPANALHPSH GIA n EPEISODIA
  twra=1;                   % ARXH EPEISODIOY ME THN KATASTASH A=1
  for j=1:2                 % META APO DYO BHMATA (EWS TIS KATASTASEIS
D,E,F)
    epomeno=EPOMENO(twra);  % function EPOMENO ORIZEI THN EPOMENH KATASTASH
    w=w+a*(w*s(epomeno,:)'-w*s(twra,:)')*s(twra,:); % ANANEWSH TIMWN
    twra=epomeno;           % METABASH STHN EPOMENH KATASTASH
  end
  y=EPOMENO(twra);          % TRITO BHMA EPEISODIOY(EWS TIS G,H,I,J), ME
y=TELIKH AMOIBH 0,1
  w=w+a*(y-w*s(twra,:)')*s(twra,:);                % ANANEWSH TIMWN
end
end
```

```
function [P ] = MC_taxidi( n )
% Monte Carlo METHODOS GIA THN EKTIMHSH TWN PITHANOTHTWN EYRESHS
% TOY THYSAYROY XEKINONTAS KATHE FORA KAI APO DIAFORETIKH KATASTASH
P=zeros(1,6);               % PITHANOTHTES EYRESHS GIA TIS 6 KATASTASEIS
Y=zeros(1,n);               % GIA THN KATAGRAFH TWN n EKBASEWN TOY TAJIDIOY
for j=1:n
    twra=1;                 % APO THN KATASTASH A=1
    for i=1:3
        twra=EPOMENO(twra); %OLO TO TAXIDI EWS TO TELOS / 3 BHMATA
```

```
    end
    Y(j)=twra;                % TELIKH EKBASH... 0 'H  1
end
P(1)=sum(Y)/n;                % MESH TIMH = PIUANOTHTA EYRESHS TOY THUSAYROY

Y=zeros(1,n);
for j=1:n
    twra=2;                   % APO THN KATASTASH B=2
    for i=1:2
        twra=EPOMENO(twra); %OLO TO TAXIDI EWS TO TELOS / 2 BHMATA
    end
    Y(j)=twra;                % TELIKH EKBASH... 0 'H  1
end
P(2)=sum(Y)/n;                % MESH TIMH = PIUANOTHTA EYRESHS TOY THUSAYROY

Y=zeros(1,n);
for j=1:n
    twra=3;                   % APO THN KATASTASH C=3
    for i=1:2
        twra=EPOMENO(twra); %OLO TO TAXIDI EWS TO TELOS / 2 BHMATA
    end
    Y(j)=twra;                % TELIKH EKBASH... 0 'H  1
end
P(3)=sum(Y)/n;                % MESH TIMH = PIUANOTHTA EYRESHS TOY THUSAYROY

Y=zeros(1,n);
for j=1:n
    twra=4;                   % APO THN KATASTASH D=4
    Y(j)=EPOMENO(twra);      %TO TELOS/1 BHMA KAI TELIKH EKBASH... 0 'H  1
end
P(4)=sum(Y)/n;                % MESH TIMH = PIUANOTHTA EYRESHS TOY THUSAYROY

Y=zeros(1,n);
for j=1:n
    twra=5;                   % APO THN KATASTASH G=5
    Y(j)=EPOMENO(twra);      %TO TELOS/1 BHMA KAI TELIKH EKBASH... 0 'H  1
end
P(5)=sum(Y)/n;                % MESH TIMH = PIUANOTHTA EYRESHS TOY THUSAYROY

Y=zeros(1,n);
for j=1:n
    twra=6;                   % APO THN KATASTASH F=6
    Y(j)=EPOMENO(twra);      %TO TELOS/1 BHMA KAI TELIKH EKBASH... 0 'H  1
end
P(6)=sum(Y)/n;                % MESH TIMH = PIUANOTHTA EYRESHS TOY THUSAYROY
end
```

```
A=[0.375 0.5 0.25 0.5 0.5 0];    % OI SWSTES PITHANOTHTES EYRESHS TOY THYSAYROY
MC=MC_taxidi( 20000 )      % EKTIMHSH TWN PITHANOTHTEN APO DEIGMA 10000 DIADROMWN
RMSE_MC=(sum([MC-A].*[MC-A])/6)^0.5%RIZA MESOY TETRAGWNIKOY SF/TOS MONTE CARLO

A=[0.375 0.5 0.25 0.5 0.5 0]; % OI SWSTES PITHANOTHTES EYRESHS TOY THYSAYROY
TD=TD_xrusafi(20000)       % EKTIMHSH TWN PITHANOTHTEN APO DEIGMA 10000 DIADROMWN
RMSE_TD=(sum([TD-A].*[TD-A])/6)^0.5%RIZA MESOY TETRAGWNIKOY SFALMATOS GIATHN TD
```

ΠΑΡΑΡΤΗΜΑ 4 - Αλγόριθμοι MATLAB

ΓΙΑ ΤΟ ΠΡΟΒΛΗΜΑ ΕΛΕΓΧΟΥ ΑΠΟΘΕΜΑΤΩΝ

Ενδεικτικά παραθέτουμε 2 από τους αλγορίθμους
Για το πρόβλημα με δυο προϊόντα, τους Q και Q(λ).

```
function [ Q ] =
inf_APOTHEMATA_2_Q(MAX,TIMH_PWLHSHS_1,TIMH_PWLHSHS_2,K1,K2,COSTOS_PARAGWGHS
_1,COSTOS_PARAGWGHS_2,COSTOS_APOTHHKEYSHS,N,e,a,g)
%** Q **PROBLHMA APOTHEMATWN: GIA 2 PROIONTA KAI APEIRO XRONIKO ORIZONTA
% STATE H KATASTASH (APO8EMA TWN 2 PROIONTWN)
k=1;'[e,a,g]='
[e,a,g]
TIMH_PWLHSHS=[TIMH_PWLHSHS_1,TIMH_PWLHSHS_2];K=[K1,K2];
COSTOS_PARAGWGHS=[COSTOS_PARAGWGHS_1,COSTOS_PARAGWGHS_2];
Q=zeros(MAX+1,MAX+1,MAX+1,MAX+1);
                % Q(APOFASH1,APOFASH1,KATASTASH1,KATASTASH2)
                % DEIKTHS i=APOFASH PARAGWGHS+1 KAI
                % DEIKTHS j=APOFASH PARAGWGHS+1,OMOIWS TWN
                % KATASTASEWN, 3os KAI 4os DEIKTHS
%%%%%%%%%%%%%%%%%% EPILOGH TYXAIAS ARXIKHS KATASTASHS GIA KATHE EPEISODIO
U=(MAX+2)*(MAX+1)/2;
for epeisodio=1:N
   epilogi=randi(U);  % ISOPITHANH EPILOGH APO TOYS SYNDIASMOYS KATASTASEWN
   S1=0;z=MAX+1;  [epeisodio/N,e,a]
    while ((epilogi-z)>0)
       S1=S1+1;
       epilogi=epilogi-z;
       z=z-1;
    end
    S2=epilogi-1;
    STATE=[S1,S2];  % ARXIKH KATASTASH: TO APOTHEMA THS APOTHHKHS
    APOFASH=e_2_greedy_Q_inf(Q,STATE,e);  % 2 TIMES
    ei=floor(epeisodio/(k*2000));
    if ei
        e=max(0.01,e*(100-ei)/100);k=k+ei; % MEIWNETAI O PARAGONTAS
EXPLORATION
        a=max(0.01,a*(100-ei)/100);        % MEIWNETAI O PARAGONTAS
MATHHSHS
    end
    for w=1:10
         ZHTHSH=[randi(3)+2,randi(2)+2];         %
         PWLHSH=min(STATE+APOFASH,ZHTHSH);
         STATE_1=STATE+APOFASH-PWLHSH;
         AMOIBH=sum(TIMH_PWLHSHS.*PWLHSH-K.*(APOFASH~=[0,0])-
COSTOS_PARAGWGHS.*APOFASH-COSTOS_APOTHHKEYSHS*STATE_1);
         APOFASH_1=e_2_greedy_Q_inf(Q,STATE_1,0);  %H PRWTH APOFASH APO TIS
a* THS EPOMENHS KATASTASHS
TARGET=AMOIBH+g*Q(APOFASH_1(1)+1,APOFASH_1(2)+1,STATE_1(1)+1,STATE_1(2)+1);
     OLD=Q(APOFASH(1)+1,APOFASH(2)+1,STATE(1)+1,STATE(2)+1);
     Q(APOFASH(1)+1,APOFASH(2)+1,STATE(1)+1,STATE(2)+1)=OLD+a*(TARGET-OLD);
     STATE=STATE_1;
     APOFASH=APOFASH_1;
   end
end
beep
end
```

```
function [ a ] =e_2_greedy_Q_inf(Q,State,e)
%APOFASH e-greedy APO THN Q  GIA THN KATASTASH State
SIZE=size(Q);
MAX=SIZE(1)-1;
ST_1_2=State(1)+State(2);
a_Max=MAX-ST_1_2;n=a_Max+1;
if n>0
APOFASEIS=Q(1:(n),1:(n),State(1)+1,State(2)+1); %LOGW MAX XWRITIKOTHTAS
DYNATES_APOFASEIS=APOFASEIS(1,1:n);
for i=2:n
    DYNATES_APOFASEIS=[DYNATES_APOFASEIS,APOFASEIS(i,1:(n+1-i))];
end  % OI DYNATES_APOFASEIS EINAI TO TRIGWNIKO PLEGMA... WS DIANYSMA
DYNATES_APOFASEIS;
mmAx=max(DYNATES_APOFASEIS);
THESEIS=find(DYNATES_APOFASEIS==(mmAx));
m=length(DYNATES_APOFASEIS);  %PLHTHOS APOFASEWN
m_max=length(THESEIS); [m,m_max]  ;     %PLHTHOS MEGISTWN THS Q
if m~=m_max
     PITHANOTHTES=ones(1,m)*e/(m-m_max);
     PITHANOTHTES(THESEIS)=(1-e)/m_max;
else PITHANOTHTES=ones(1,m)/m;
end                              % ORISMOS PITHANOTHTAS GIA KATHE APOFASH
%%%%%%%%%%%%%%%%%%% EPILOGH TYXAIAS PARAGWGHS APO TO PLEGMA...%%%%%%%%%%%%
PITH_SUM=cumsum(PITHANOTHTES); % KAI ATHRISTIKHS PITHANOTHTAS...
PITH_SUM(m);
u=rand;                         % GIA THN APILOGH THS APOFASHS
i=1;
while PITH_SUM(i)<u
    i=i+1;             % TO i-OSTO STOIXEIO TOY DIANYSMATOS TWN APOFASEWN..
end                    % ..KAI POY ANTISTOIXEI STO "DIAKRITO TRIGWNO..."
a1=0;z=n;
while ((i-z)>0)        % ANAKTISH TWN APOFASEWN a1 a2 APO TO DIANYSMA
    a1=a1+1;
    i=i-z;
    z=z-1;
end
a2=i-1;
a=[a1,a2];
else
    a=[0,0];
end
end
```

```
function [ Q ] =
inf_APOTHEMATA_3_Q_L(MAX,TIMH_PWLHSHS_1,TIMH_PWLHSHS_2,K1,K2,COSTOS_PARAGWG
HS_1,COSTOS_PARAGWGHS_2,COSTOS_APOTHHKEYSHS,N,e,a,g,L)
%**Q(L)**PROBLHMA APOTHEMATWN: GIA 2 PROIONTA KAI APEIRO XRONIKO ORIZONTA
% STATE H KATASTASH (APO8EMA TWN 2 PROIONTWN)
k=1;'[e,a,g,L]='
[e,a,g,L]
TIMH_PWLHSHS=[TIMH_PWLHSHS_1,TIMH_PWLHSHS_2];K=[K1,K2];
COSTOS_PARAGWGHS=[COSTOS_PARAGWGHS_1,COSTOS_PARAGWGHS_2];
Q=zeros(MAX+1,MAX+1,MAX+1,MAX+1);
                % Q(APOFASH1,APOFASH1,KATASTASH1,KATASTASH2)
                % DEIKTHS i=APOFASH PARAGWGHS+1 KAI
                % DEIKTHS j=APOFASH PARAGWGHS+1,OMOIWS TWN
                % KATASTASEWN, 3os KAI 4os DEIKTHS
U=(MAX+2)*(MAX+1)/2;
W=ceil(log(0.2)/log(L*g))+1; % PLH8OS APO IXNH < 0.2
for epeisodio=1:N            % N LOOPS GIA EXPLORATION
   %%%%%%%%%%%%%%%% EPILOGH TYXAIAS ARXIKHS KATASTASHS GIA KATHE EPEISODIO
   epilogi=randi(U);  % ISOPITHANH EPILOGH APO TOYS SYNDIASMOYS KATASTASEWN
   S1=0;z=MAX+1; [epeisodio/N,e,a]
    while ((epilogi-z)>0)
       S1=S1+1;    epilogi=epilogi-z;    z=z-1;
    end
    S2=epilogi-1;
    STATE=[S1,S2];  % ARXIKH KATASTASH: TO APOTHEMA THS APOTHHKHS
    APOFASH=e_2_greedy_Q_inf(Q,STATE,e);  % 2 TIMES
    ELIG=zeros(W,5);
    ei=floor(epeisodio/(k*20000));
    if ei
        e=max(0.01,e*0.95);k=k+ei;
        a=max(0.01,a*0.95);
    end
    for w=1:W
         ZHTHSH=[randi(3)+2,randi(2)+2];
        PWLHSH=min(STATE+APOFASH,ZHTHSH);
        STATE_1=STATE+APOFASH-PWLHSH;
        AMOIBH=sum(TIMH_PWLHSHS.*PWLHSH-K.*(APOFASH~=[0,0])-
COSTOS_PARAGWGHS.*APOFASH-COSTOS_APOTHHKEYSHS*STATE_1);
        APOFASH_1_e=e_2_greedy_Q_inf(Q,STATE_1,e);  % EPOMENH APOFASH, e-
GREEDY APOFASH THS EPOMENHS KATASTASHS
[i,j]=find(Q(:,:,STATE_1(1)+1,STATE_1(2)+1)==max(max(Q(:,:,STATE_1(1)+1,STA
TE_1(2)+1))),1);
        APOFASH_1=[i-1,j-1];          % H a* APOFASH THS EPOMENHS KATASTASHS
DEIKTRIA_elig=(Q(APOFASH_1_e(1)+1,APOFASH_1_e(2)+1,STATE(1)+1,STATE(2)+1)==
Q(APOFASH_1(1)+1,APOFASH_1(2)+1,STATE(1)+1,STATE(2)+1));
TARGET=AMOIBH+g*Q(APOFASH_1(1)+1,APOFASH_1(2)+1,STATE_1(1)+1,STATE_1(2)+1);
        OLD=Q(APOFASH(1)+1,APOFASH(2)+1,STATE(1)+1,STATE(2)+1);
        D_TARGET=TARGET-OLD;
        ELIG(w,:)=[APOFASH(1),APOFASH(2),STATE(1),STATE(2),1]; %8ETOYME TO
IXNOS ISO ME 1
        for c=1:w
Q(ELIG(c,1)+1,ELIG(c,2)+1,ELIG(c,3)+1,ELIG(c,4)+1)=Q(ELIG(c,1)+1,ELIG(c,2)+
1,ELIG(c,3)+1,ELIG(c,4)+1)+a*D_TARGET*ELIG(c,5);
        end  % ANANEWNOYME TIS TIMES GIA TIS "TELEYTAIES" W KATASTASEIS-
APOFASEIS (L*g<0.2)
        ELIG(1:(w),5)=ELIG(1:(w),5)*L*g*DEIKTRIA_elig; % ALLAZOYME TA IXNH
GIA TO EPOMENO BHMA
        STATE=STATE_1;      APOFASH=APOFASH_1_e;
    end
end
end
```

ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] Andreae, J.H.(1969b), "Learning machines - a unified view", In Meetham, A.R. and Hudson, R.A. editors, *Encyclopedia of information, Linguistics, and Control,* pages 261-270, Pergamon, Oxford.

[2] Bellman, R. E. (1957), "Dynamic programming", *NJ: Princeton University Press*, Princeton.

[3] Dayan,P. (1992),"The convergence of TD(λ) for general λ",*Machine Learning*,**8**:341-362.

[4] Jaakkola T. Singh S. and Jordan M. (1995) "Reinforcement Learning Algorithm for Partially Observable Markov Decision Problems", *The MIT Press*, Massachusetts.

[5] Minsky, M.L. (1961), "Steps toward artificial intelligence", *Proceedings of the Institute of Radio Engineers*, 49: 8-30, New York.

[6] Puterman, M. (1994), "Markovian Decision Process*", J. Wiley and Sons,* New York.

[7] Ross, S. (1981), "Introduction to Stochastic Dynamic Programming", *Academic Pres,* New York.

[8] Singh, S., Jaakkola, T., Littman, M. and Szepesvari, C. (2000), "Convergence Results for Single-Step On-Policy Reinforcement-Learning Algorithms", *Machine Learning, **39**,287-308, Kluwer Academic Publishers*, Netherlands.

[9] Sutton, R. S. (1988), "Learning to predict by the method of temporal differences", *Machine Learning*, **3**:9-44.

[10] Sutton,R. and Barto,A. (1998), "Reinforcement Learning: An Introduction", *The MIT Press*, Massachusetts.

[11] Watkins, C.J.C.H., (1989), "Learning from Delayed Rewards", *PhD Thesis Cambridge University*, Cabridge, England.

[12] Watkins, C.J.C.H. and Dayan, P. (1992), "Q-learning", *Machine Learning*, **8**: 279-292.

[13] Werbos, P. (1977), "Advanced forecasting methods for global crisis warning and models of intelligence", *General Systems Yearbook, **22**:25-38.*

[14] Τζαφέστας, Σπ. (2002), "Υπολογιστική Νοημοσύνη", *ΕΜΠ*, Αθήνα.

ΔΙΑΔΙΚΤΥΟ

Ρεφανίδης Ιωάννης , Μάθημα Νευρωνικών Δικτύων, Πανεπιστήμιο Μακεδονίας:
http://ai.uom.gr/Courses/AdvancedNeuralNetworks/Material/NeuralNetworksAll.pdf

Wikipedia: Artificial neural network
http://en.wikipedia.org/wiki/Artificial_neural_network

Using neural networks:
http://neuralnetworksanddeeplearning.com/chap1.html