ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ

Ανάπτυξη μιας Προσαρμοστικής Πολιτικής Αντικατάστασης Αρχείων, με χρήση Ενισχυτικής Μάθησης, σε Ιεραρχικά Συστήματα Αποθήκευσης

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

του

ΘΕΟΔΩΡΟΥ Ι. ΡΕΚΑΤΣΙΝΑ

Επιβλέπων : Τιμολέων Σελλής
Καθηγητής Ε.Μ.Π.

Αθήνα, Ιούλιος 2010

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ
ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ

Ανάπτυξη μιας Προσαρμοστικής Πολιτικής Αντικατάστασης Αρχείων, με χρήση Ενισχυτικής Μάθησης, σε Ιεραρχικά Συστήματα Αποθήκευσης

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

του

ΘΕΟΔΩΡΟΥ Ι. ΡΕΚΑΤΣΙΝΑ

Επιβλέπων : Τιμολέων Σελλής
Καθηγητής Ε.Μ.Π.

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την Πέμπτη, 8 Ιουλίου 2010.

.....	.....	.....
Τιμολέων Σελλής	Ιωάννης Βασιλείου	Νεκτάριος Κοζύρης
Καθηγητής Ε.Μ.Π.	Καθηγητής Ε.Μ.Π.	Αναπλ. Καθηγητής Ε.Μ.Π.

Αθήνα, Ιούλιος 2010

.....

ΘΕΟΔΩΡΟΣ Ι. ΡΕΚΑΤΣΙΝΑΣ

Διπλωματούχος Ηλεκτρολόγος Μηχανικός και Μηχανικός Υπολογιστών Ε.Μ.Π.

Ευχαριστίες

Με την ολοκλήρωση της εκπόνησης της διπλωματικής μου εργασίας θα ήθελα να απευθύνω τις ευχαριστίες μου σε όλους τους ανθρώπους που με βοήθησαν και με συνόδεψαν στα χρόνια των σπουδών μου στο Πολυτεχνείο.

Αρχικά θα ήθελα να ευχαριστήσω βαθύτατα τον καθηγητή μου Τίμο Σελλή, ο οποίος υπήρξε δίπλα μου καθόλη τη διάρκεια της διπλωματικής με κατανόηση και συμπαράσταση. Θέλω να τον ευχαριστήσω βαθύτατα για την εμπιστοσύνη που μου έδειξε στηρίζοντας με σε κάθε νέο βήμα κατά τη διάρκεια των προπτυχιακών σπουδών μου και για την ώθηση που μου έδωσε να ασχοληθώ με το αντικείμενο της Μηχανικής Μάθησης.

Ακόμα, ευχαριστώ από καρδιάς τους φίλους μου και τους συμφοιτητές μου για τις όμορφες στιγμές που ζήσαμε αυτά τα χρόνια, τη συμπαράσταση και την κατανόησή τους.

Τέλος θα ήθελα να πω ένα μεγάλο ευχαριστώ στους γονείς μου για τη στήριξή τους και τις πολύτιμες συμβουλές που μου προσέφεραν μέχρι σήμερα. Η αστείρευτη αγάπη τους και συμπαράστασή τους αποτελεί το μεγαλύτερο στήριγμα στην πορεία μου.

Θοδωρής Ρεκατσίνας

Περίληψη

Η διαχείριση αρχείων σε ιεραρχικά συστήματα αποθήκευσης είναι ένα ενεργό πεδίο έρευνας. Λόγω της συνεχούς αύξησης του όγκου των δεδομένων η χρήση ιεραρχικών συστημάτων αποθήκευσης για τη διαχείριση δεδομένων γίνεται όλο και περισσότερο απαραίτητη. Τα συστήματα αυτά έχουν περισσότερες παραμέτρους και πολιτικές που πρέπει να ρυθμιστούν σε σχέση με παραδοσιακά συστήματα αποθήκευσης. Ένας από τους σημαντικότερους παράγοντες της απόδοσης των συστημάτων αυτών είναι ο χρόνος εξυπηρέτησης των εισερχόμενων αιτήσεων προσπέλασης αρχείων. Ο χρόνος εξυπηρέτησης μπορεί να μειωθεί εάν αυξηθεί το πλήθος των αιτήσεων που εξυπηρετούνται από το υψηλότερο και παράλληλα ταχύτερο επίπεδο της ιεραρχίας μνήμης. Δυστυχώς είναι αδύνατο για τους διαχειριστές τέτοιων συστημάτων να ανιχνεύσουν πιθανά μοτίβα προσπέλασης των αρχείων. Επίσης κλασσικές στατικές πολιτικές αντικατάστασης αποτυγχάνουν να μεγιστοποιήσουν την απόδοση του συστήματος καθώς δε λαμβάνουν υπόψη τους σημαντικές παραμέτρους των αρχείων όπως το μέγεθος τους. Έναυσμα για την έρευνα μας αποτέλεσε η βελτίωση των πολιτικών αντικατάστασης που χρησιμοποιεί το ιεραρχικό σύστημα αποθήκευσης CASTOR, το οποίο σχεδιάστηκε και υλοποιήθηκε στο CERN.

Η παρούσα εργασία παρουσιάζει δύο νέες προσαρμοστικές πολιτικές αντικατάστασης αρχείων, οι οποίες χρησιμοποιούν μεθόδους Ενισχυτικής Μάθησης για τη ρύθμιση των παραμέτρων τους. Αρχικά προτείνουμε μια γενικευμένη μοντελοποίηση του προβλήματος αντικατάστασης των αρχείων ως πρόβλημα ενισχυτικής μάθησης. Η μοντελοποίηση των δύο προσεγγίσεων που προτείνονται, βασίζεται στην αναγωγή του προβλήματος αντικατάστασης αρχείων στο πρόβλημα stochastic Knapsack και min-Knapsack αντίστοιχα. Αφού μοντελοποιήσουμε το πρόβλημα caching ως πρόβλημα ενισχυτικής μάθησης ενσωματώνουμε τις νέες πολιτικές σε ένα προσομοιωτή του συστήματος CASTOR, ο οποίος αναπτύχθηκε ειδικά για την αξιολόγηση των πολιτικών αντικατάστασης. Σύμφωνα με τα πρώτα αποτελέσματα οι νέες πολιτικές επιτυγχάνουν καλύτερη απόδοση από τις ήδη υπάρχουσες στατικές πολιτικές του συστήματος.

Λέξεις Κλειδιά: Προσαρμοστική Πολιτική Αντικατάστασης Αρχείων, Γενικευμένο Πρόβλημα Caching, Ιεραρχικά Συστήματα Αποθήκευσης, CERN, CASTOR, Ενισχυτική Μάθηση, Στοχαστικό Πρόβλημα Σακιδίου, Πρόβλημα Ελαχιστοποίησης Σακιδίου

Abstract

As more data needs to be storeδ, hierarchical storage systems are becoming more and more widespread in the industry. These multi-tier systems have more tunable parameters and policies than traditional storage systems. A very important performance factor for such systems is the processing time of incoming file access requests. The processing time can be drastically reduced if most of the incoming requests are served directly from the highest and fastest storage level of the hierarchy. Unfortunately it is impossible for system administrators to monitor user request access patterns. Moreover, typical file replacement policies such as LRU and LFU often fail to achieve near-optimal performance as they do no not take into account important file parameters, such as file size. The starting point of our research was the enhancement of caching policies employed in CASTOR, a hierarchical storage system developed at CERN.

In this thesis we first propose a generic formulation of the caching problem as a Reinforcement Learning problem and then develop two novel adaptive policies, based on the stochastic Knapsack and min-Knapsack problems respectively. Afterwards we embed those new policies into a CASTOR simulation, specifically developed in order to test file caching policies. According our experiment results show that the new adaptive policies perform better than any non-adaptive caching policy already employed.

Keywords: Adaptive File Replacement Policy, Generalized Caching Problem, Hierarchical Storage Systems, CERN, CASTOR, Reinforcement Learning, Stochastic Knapsack Problem, Min-Knapsack Problem

Πίνακας περιεχομένων

1

Εισαγωγή

1.1 Ιεραρχικά Συστήματα Αποθήκευσης - Διαχείριση Αρχείων

Με τη συνεχή αύξηση του όγκου των δεδομένων που παράγεται σε ψηφιακή μορφή, οι αποθήκες δεδομένων αποτελούν ένα πεδίο με έντονο ερευνητικό ενδιαφέρον. Καθημερινά παράγονται petabytes δεδομένων είτε από επιστημονικά πειράματα είτε από εταιρίες πληροφορικής. Η ανάγκη της διατήρησης και διαχείρισης αυτού του όγκου των δεδομένων οδήγησε στη διαδεδομένη χρήση των ιεραρχικών συστημάτων αποθήκευσης (hierarchical storage management systems). Τα ιεραρχικά συστήματα αποθήκευσης εισάγουν ένα νέο επίπεδο στην ιεραρχία της αποθήκευσης δεδομένων. Πέραν της κύριας μνήμης και των δευτερευόντων μέσων αποθήκευσης (μαγνητικοί σκληροί δίσκοι ή δίσκοι στερεάς κατάστασης) χρησιμοποιούνται και τριτογενή μέσα αποθήκευσης, όπως μαγνητικές κασέτες αποθήκευσης ή αφαιρούμενοι σκληροί δίσκοι. Λόγω των πολλαπλών επιπέδων ο αριθμός των παραμέτρων που καθορίζουν τη λειτουργία των συστημάτων αυτών είναι αυξημένος σε σχέση με τα παραδοσιακά συστήματα αποθήκευσης δεδομένων. Σε τόσο περίπλοκα συστήματα είναι αναγκαία η αυτόματη προσαρμογή των παραμέτρων και των πολιτικών λειτουργίας τους, ώστε να μη διαταράσσεται η λειτουργία τους από μεταβολές του εξωτερικού περιβάλλοντός τους. Ένα από τα σημαντικότερα σημεία για την αποδοτική λειτουργία των ιεραρχικών συστημάτων αποθήκευσης είναι η μεταφορά και διαχείριση αρχείων ανάμεσα στα διάφορα επίπεδα. Όπως θα δούμε στη συνέχεια της παρούσας εργασίας

γίνονται πολλές προσπάθειες για την ανάπτυξη προσαρμοστικών αλγορίθμων και πολιτικών με σκοπό τη βελτίωση της διαχείρισης των αρχείων σε τέτοια συστήματα.

1.2 Αντικείμενο διπλωματικής

Στην παρούσα διπλωματική εργασία επικεντρώνουμε στις πολιτικές αντικατάστασης αρχείων στα υψηλότερα επίπεδα της ιεραρχίας και ερευνούμε τον τρόπο με τον οποίο μια προσαρμοστική πολιτική αντικατάστασης αρχείων μπορεί να ενσωματωθεί σε ένα ιεραρχικό σύστημα. Ισχυριζόμαστε ότι το πεδίο της Ενισχυτικής Μάθησης (Reinforcement Learning) παρέχει τους κατάλληλους αλγορίθμους βελτιστοποίησης για τη μάθηση μιας βέλτιστης πολιτικής αντικατάστασης αρχείων και δείχνουμε πώς το γενικευμένο πρόβλημα της προσωρινής αποθήκευσης αρχείων (file caching problem) μπορεί να μοντελοποιηθεί ως πρόβλημα ενισχυτικής μάθησης. Αφού προτείνουμε δύο διαφορετικές προσεγγίσεις για την υλοποίηση μιας προσαρμοστικής πολιτικής αντικατάστασης αρχείων, ενσωματώνουμε τους νέους αλγορίθμους στο σύστημα CASTOR (Cern Advanced STORage manager), στο οποίο θα εκτελέσουμε τα πειράματά μας ώστε να αξιολογήσουμε την απόδοση των νέων πολιτικών.

Πριν τη σχετική μελέτη του προβλήματος αναπτύχθηκε ένας προσομοιωτής του συστήματος CASTOR, αφού η πρόσβαση στο πραγματικό σύστημα ήταν αδύνατη. Το σύστημα CASTOR είναι ένα τυπικό ιεραρχικό σύστημα αποθήκευσης που αναπτύχθηκε στο CERN και χρησιμοποιείται για την αποθήκευση δεδομένων που παράγονται από τα πειράματα του CERN αλλά και προσωπικών αρχείων των επιστημόνων. Στο χαμηλότερο επίπεδο της ιεραρχίας τα δεδομένα αποθηκεύονται σε κασέτες ενώ στο υψηλότερο σε σκληρούς δίσκους.

Η σημαντικότερη πρόκληση στην έρευνά μας εισάγεται λόγω του πλήθους των αρχείων που διαχειρίζεται το σύστημα CASTOR και του πλήθους των αιτήσεων που δέχεται από χρήστες. Λόγω της κλίμακας του συστήματος κάθε λειτουργία του πρέπει να τηρεί ορισμένους περιορισμούς πολυπλοκότητας και να χαρακτηρίζεται από επεκτασιμότητα (scalability) ως προς το πλήθος των αρχείων του συστήματος. Τέλος επισημαίνουμε ότι οι αιτήσεις δεν προκύπτουν από κάποια εφαρμογή αλλά από τους ίδιους τους χρήστες, συνήθως χωρίς να ακολουθούν κάποιο δεδομένο μοτίβο. Το γεγονός αυτό καθιστά ιδιαίτερα ενδιαφέρουσα την εφαρμογή προσαρμοστικών μεθόδων διαχείρισης αρχείων στο σύστημα.

1.2.1 Συνεισφορά

Στην παρούσα διπλωματική εργασία στόχος μας είναι η ανάπτυξη μιας εξ' ολοκλήρου προσαρμοστικής πολιτικής για την αντικατάσταση και διαγραφή αρχείων από το υψηλότερο επίπεδο της ιεραρχίας.

Ισχυριζόμαστε ότι το πεδίο της Ενισχυτικής Μάθησης παρέχει τους κατάλληλους αλγορίθμους και τις κατάλληλες μεθόδους για την ανάπτυξη μιας τέτοιας πολιτικής και ερευνούμε την εφαρμογή του στο πρόβλημα της αντικατάστασης αρχείων. Τελικά προτείνουμε μια γενικευμένη μοντελοποίηση του προβλήματος αντικατάστασης αρχείων ως πρόβλημα ενισχυτικής μάθησης και δύο νέους τρόπους επίλυσης του προβλήματος με χρήση μεθόδων ενισχυτικής μάθησης.

Η μόνη δημοσίευση σχετική δημοσίευση την παρούσα στιγμή είναι η [24], η οποία όμως δεν προτείνει μια νέα προσαρμοστική πολιτική αλλά ένα ενδιάμεσο μηχανισμό για επιλογή της καταλληλότερης στατικής πολιτικής από ένα σύνολο προκαθορισμένων στατικών πολιτικών. Για την υλοποίηση του μηχανισμού αυτού χρησιμοποιείται ενισχυτική μάθηση.

Τέλος ενσωματώνουμε την προτεινόμενη μοντελοποίηση σε μια προσομοίωση του συστήματος CASTOR, την οποία αναπτύξαμε για την εκτέλεση και αξιολόγηση πολιτικών αντικατάστασης αρχείων. Η ενσωμάτωση των νέων μεθόδων σε ένα σύστημα που διαχειρίζεται Petabytes δεδομένων και εκατομμύρια αρχείων αποτελεί μια ιδιαίτερη πρόκληση και μας επιτρέπει να εξετάσουμε κατά πόσο μπορούν να εφαρμοστούν κλασσικές και μη τεχνικές βελτιστοποίησης σε τέτοια συστήματα. Εδώ πρέπει να τονίσουμε ότι παράλληλα με την ανάπτυξη των νέων προσαρμοστικών μεθόδων αναπτύχθηκε και μια νέα στατική πολιτική αντικατάστασης, η οποία λειτουργεί καλύτερα από τις ήδη υπάρχουσες πολιτικές στο σύστημα CASTOR.

1.3 Οργάνωση κειμένου

Η παρούσα εργασία χωρίζεται σε 7 κεφάλαια. Στο κεφάλαιο 2 γίνεται μια εισαγωγή στα Ιεραρχικά Συστήματα Αποθήκευσης και περιγράφεται το σύστημα CASTOR πάνω στο οποίο εξετάζουμε την εφαρμογή των νέων προσαρμοστικών πολιτικών αντικατάστασης που προτείνουμε. Εκτός του παραπάνω γίνεται μια βιβλιογραφική αναφορά σε στατικές πολιτικές αντικατάστασης αρχείων αλλά και σχετικές εργασίες που προτείνουν προσαρμοστικές πολιτικές. Στο κεφάλαιο 3 γίνεται μια εισαγωγή στο πεδίο της Ενισχυτικής Μάθησης, αναφέρονται οι σημαντικότερες κατηγορίες αλγορίθμων για διακριτούς και συνεχείς χώρους καταστάσεων. Το κεφάλαιο 4 προσφέρει μια ολοκληρωμένη ανάλυση για το πώς το πρόβλημα αντικατάστασης αρχείων μπορεί να μοντελοποιηθεί ως πρόβλημα ενισχυτικής μάθησης. Παρουσιάζουμε τις διαφορετικές προσεγγίσεις που ακολουθήσαμε και τελικά

προτείνουμε δύο προσαρμοστικές πολιτικές αντικατάστασης που βασίζονται στη μέθοδο της ενισχυτικής μάθησης. Στο κεφάλαιο 5 παρουσιάζονται και αναλύονται τα αποτελέσματα των πειραμάτων που εκτελέσαμε, ενώ στο κεφάλαιο 6 επιχειρείται μια σύνοψη της εργασίας και παρουσιάζονται δυνατότητες μελλοντικής έρευνας. Τέλος, το κεφάλαιο 7 παρουσιάζει τη σχετική βιβλιογραφία.

2

Ανάλυση αλγορίθμων caching σε Ιεραρχικά Συστήματα Διαχείρισης

Η χρήση βοηθητικής μνήμης (caching) είναι ένας ιδιαίτερα αποδοτικός τρόπος για τη μείωση του χρόνου προσπέλασης δεδομένων αφού περιορίζει τη μεταφορά τους από δευτερογενή μέσα αποθήκευσης. Ένα ιεραρχικό σύστημα αποθήκευσης αρχείων μπορεί να μοντελοποιηθεί κατά αναλογία με την ιεραρχία μνήμης σε ένα υπολογιστή, αφού και εκεί τα υψηλότερα επίπεδα της ιεραρχίας προσφέρουν μεγαλύτερη ταχύτητα προσπέλασης αλλά περιορισμένο χώρο αποθήκευσης. Επομένως κλασσικές πολιτικές αντικατάστασης, όπως η Least Recently Used (LRU) ή Least Frequently Used (LFU) μπορούν να χρησιμοποιηθούν και για τη διαχείριση αρχείων σε ιεραρχικά συστήματα αποθήκευσης.

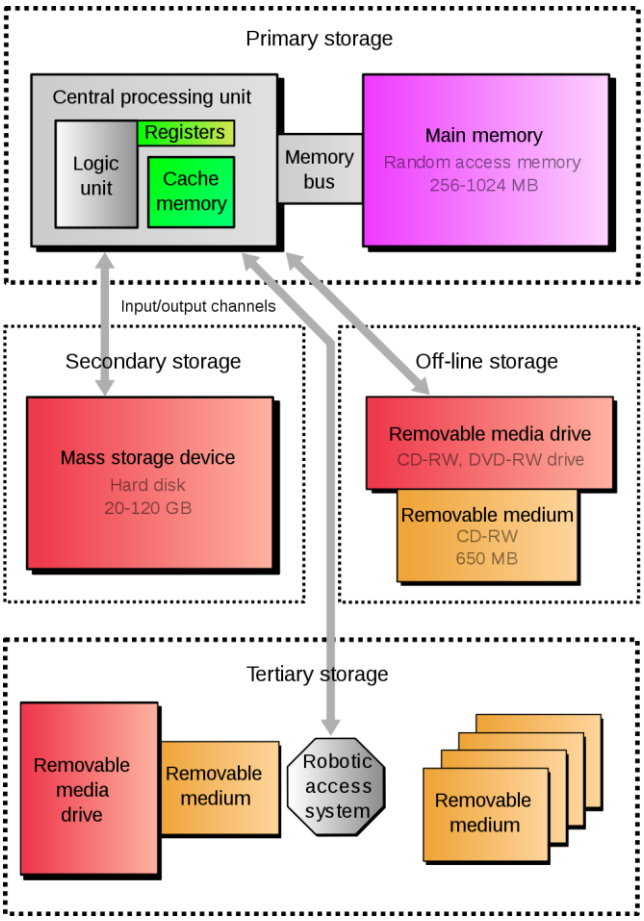
Το βασικό στοιχείο που λαμβάνουν υπόψη τους οι περισσότερες από τις κλασσικές πολιτικές αντικατάστασης είναι ο χρόνος. Η κυρία διαφορά των ιεραρχικών συστημάτων αποθήκευσης δεδομένων με τις τυπικές ιεραρχίες μνήμης είναι ότι στα πρώτα γίνεται αντικατάσταση ολόκληρων αρχείων διαφορετικού μεγέθους και όχι κάποιου μπλοκ δεδομένων προκαθορισμένου μεγέθους. Είναι προφανές ότι μια βέλτιστη πολιτική αντικατάστασης εκτός από το χρόνο προσπέλασης των αρχείων θα πρέπει να λαμβάνει υπόψη το μέγεθος των αρχείων που πρόκειται να αντικατασταθούν καθώς και άλλες παραμέτρους του συστήματος.

Στο συγκεκριμένο κεφάλαιο αφού παρουσιάσουμε συνοπτικά τη δομή των ιεραρχικών συστημάτων αποθήκευσης δεδομένων, θα αναφέρουμε και θα σχολιάσουμε διαφορετικές

γνωστές πολιτικές αντικατάστασης αρχείων καθώς και σχετικές εργασίες που έχουν γίνει στο χώρο των προσαρμοστικών πολιτικών διαχείρισης δεδομένων για ιεραρχικά συστήματα αποθήκευσης.

2.1 Εισαγωγή στα Ιεραρχικά Συστήματα Αποθήκευσης

Τα ιεραρχικά συστήματα επεκτείνουν μια τυπική ιεραρχία αποθήκευσης εισάγοντας ένα νέο επίπεδο τριτογενούς αποθήκευσης. Όπως φαίνεται και στο σχήμα 2.1, στο τελευταίο επίπεδο συναντάμε συστοιχίες είτε από αφαιρούμενους μαγνητικούς δίσκους είτε από μαγνητικές ταινίες ιδιαίτερα μεγάλης χωρητικότητας. Εν γένει τα μέσα αυτά είναι πιο φτηνά αλλά και πιο αργά σε σχέση με αποθηκευτικά μέσα των παραπάνω επιπέδων. Τα ιεραρχικά συστήματα αποθήκευσης χρησιμοποιήθηκαν αρχικά για να μειώσουν το κόστος αποθήκευσης μεγάλου όγκου δεδομένων. Παλιά ή πολύ μεγάλα αρχεία μεταφέρονται σε μαγνητικές ταινίες ή αφαιρούμενους δίσκους με σκοπό την εξοικονόμηση αποθηκευτικού χώρου στα υψηλότερα επίπεδα της ιεραρχίας μνήμης.



Σχήμα 2.1: Δομή ιεραρχίας αποθήκευσης

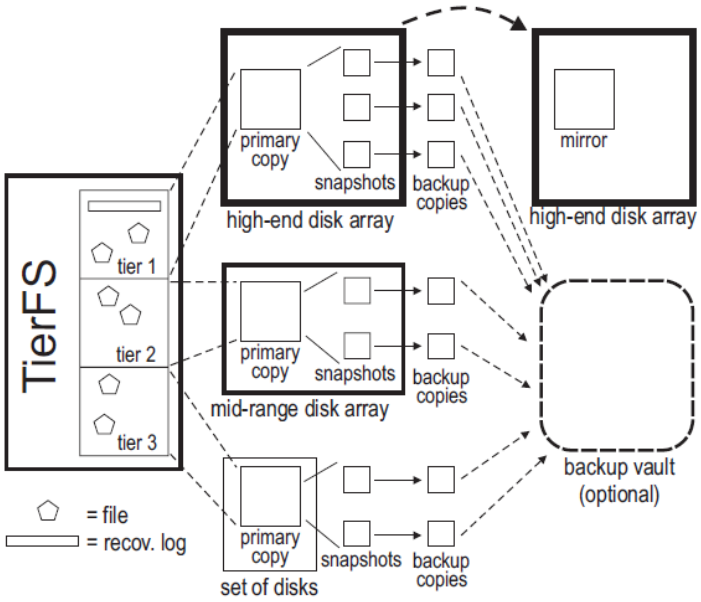
Παρά τη σημαντική μείωση του κόστους και την αύξηση της χωρητικότητα των σκληρών δίσκων τα τελευταία χρόνια, η χρήση των ιεραρχικών συστημάτων αποθήκευσης εξακολουθεί να είναι αναγκαία. Πλέον η ποσότητα των δεδομένων που παράγεται είναι της τάξης των Petabytes, καθιστώντας και πάλι απαγορευτικό το κόστος της αποκλειστικής χρήσης γρήγορων μέσων αποθήκευσης.

Η ενσωμάτωση τριτογενούς αποθηκευτικού χώρου σε ένα σύστημα υλοποιείται με την επέκταση του ίδιου του συστήματος αρχείων (file system). Σε πολλά συστήματα η ανάγνωση αρχείων είναι δυνατή μόνο στο υψηλότερο επίπεδο της ιεραρχίας. Συγκεκριμένα στο χαμηλότερο επίπεδο αποθηκεύονται όλα τα αρχεία του συστήματος, αντίγραφα των οποίων μεταφέρονται στο υψηλότερο επίπεδο προς ανάγνωση και επεξεργασία. Συχνά χρησιμοποιούμενα και μικρά αρχεία διατηρούνται στο υψηλότερο επίπεδο της ιεραρχίας ενώ μεγαλύτερα και ανενεργά αρχεία μεταφέρονται στο χαμηλότερο επίπεδο, όπου και φυλάσσονται σε μαγνητικές ταινίες.

Όταν κάποια εφαρμογή ζητήσει ένα αρχείο, το σύστημα ελέγχει εάν αυτό είναι διαθέσιμο στο πρώτο επίπεδο της ιεραρχίας. Σε περίπτωση που το αρχείο είναι διαθέσιμο, συνεχίζει κανονικά η εκτέλεση της εφαρμογής. Διαφορετικά το αρχείο αντιγράφεται από το τριτογενές μέσο αποθήκευσης στο δευτερογενές μέσο και η εφαρμογή συνεχίζει με την προσπέλαση του νέου τοπικού αντίγραφου που δημιουργήθηκε. Η παραπάνω διαδικασία συμβαίνει με διαφανή, προς την εφαρμογή, τρόπο.

Τα τελευταία χρόνια ο συνεχώς αυξανόμενος όγκος δεδομένων και η ανάπτυξη νέων τεχνολογιών αποθήκευσης, όπως οι δίσκοι σταθερής κατάστασης, έχουν οδηγήσει στην εισαγωγή πολλαπλών επιπέδων στα ιεραρχικά συστήματα αποθήκευσης. Σύμφωνα με την αρχιτεκτονική πολλαπλών επιπέδων (multi-tier architecture) κάθε επίπεδο προσφέρει διαφορετικές υπηρεσίες όσον αφορά την απόδοση, την αξιοπιστία και τη δυνατότητα ανάκτησης αρχείων.

Όπως φαίνεται στο σχήμα 2.2 ένα σύστημα αποθήκευσης πολλαπλών επιπέδων μπορεί να έχει τρείς βαθμίδες. Στην πρώτη και ακριβότερη βαθμίδα συναντώνται συστάδες δίσκων υψηλής απόδοσης, όπου λαμβάνονται συχνά στιγμιότυπα του περιεχομένου των δίσκων με σκοπό την εγγυημένη ανάκτηση των δεδομένων σε περίπτωση σφάλματος. Στο δεύτερο επίπεδο είναι εγκατεστημένοι σκληροί δίσκοι χαμηλότερης απόδοσης και η λήψη των στιγμιότυπων γίνεται με μικρότερη συχνότητα. Στο τελευταίο επίπεδο του συστήματος συναντάμε συστάδες αφαιρούμενων δίσκων πολύ μεγάλης χωρητικότητας, οι οποίες χρησιμοποιούνται κυρίως για την ιστορική αποθήκευση αρχείων. Σε αντίθεση με τα τυπικά ιεραρχικά συστήματα αποθήκευσης τα συστήματα πολλαπλών επιπέδων επιτρέπουν την ανάγνωση αρχείων από όλα τα επίπεδα. Φυσικά η ταχύτητα προσπέλασης μειώνεται όσο μεταβαίνουμε χαμηλότερα στην ιεραρχία.



Σχήμα 2.2: Το TierFs αποτελεί παράδειγμα ενός ιεραρχικού συστήματος αποθήκευσης, το οποίο παρέχει αυξημένες δυνατότητες ανάκτησης δεδομένων (Πηγή [7]).

2.2 CERN Advanced Storage Manager (CASTOR)

Το σύστημα CASTOR είναι ένα τυπικό ιεραρχικό σύστημα αποθήκευσης που αναπτύχθηκε στο CERN και χρησιμοποιείται για την αποθήκευση δεδομένων που παράγονται από τα πειράματα του CERN αλλά και προσωπικών αρχείων των επιστημόνων. Τα αρχεία είναι αποθηκευμένα είτε σε σκληρούς δίσκους είτε σε μαγνητικές ταινίες και η πρόσβαση σε αυτά γίνεται μέσω του πρωτοκόλλου RFIO (Remote File Input / Output). Σήμερα το σύστημα CASTOR διαχειρίζεται 161,5 εκατομμύρια αρχεία μεγέθους 23,8 Petabytes, από τα οποία τα 23,6 Petabytes προέρχονται από πειραματικά δεδομένα.

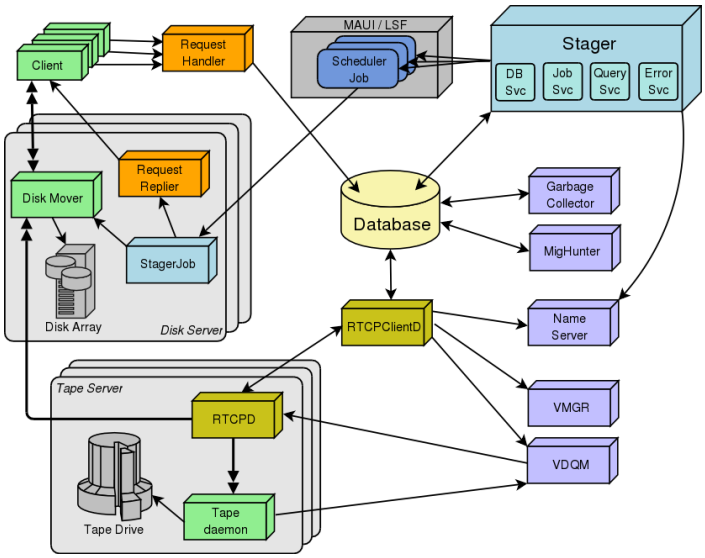
2.2.1 Ιστορία και περιορισμοί

Τον Ιανουάριο του 1999 το CERN άρχισε να κατασκευάζει την πρώτη έκδοση του σημερινού του Datacenter, το οποίο αποτελείται από χιλιάδες κόμβους συνδεδεμένους μεταξύ τους σε ένα τοπικό δίκτυο. Το σύστημα αυτό ενσωματώνει και ένα επίπεδο τριτογενών μέσων αποθήκευσης όπως μαγνητικές ταινίες και αφαιρούμενοι σκληροί δίσκοι, για την αποθήκευση αλλά και ιστορική συλλογή δεδομένων από τα πειράματα του LHC. Η πρώτη έκδοση του συστήματος CASTOR ήταν έτοιμη το 2001. Με τη ραγδαία αύξηση του όγκου των δεδομένων εμφανίστηκαν και τα πρώτα σημαντικά προβλήματα του συστήματος. Τα σημαντικότερο από αυτά ήταν η μη επαρκής κλιμακωσιμότητα του συστήματος με την αύξηση του αριθμού των αρχείων. Στην αρχική αρχιτεκτονική του συστήματος μια κεντρική

ψηφίδα του συστήματος ήταν υπεύθυνη να διατηρεί ένα κατάλογο με πληροφορίες για όλα τα αρχεία. Η συνεχής αύξηση του αριθμού των αρχείων οδήγησε την εκτόξευση του μεγέθους του καταλόγου, πράγμα το οποίο οδήγησε σε σημαντική μείωση της απόδοσης του συστήματος. Αυτή η αρχιτεκτονική άλλαξε ριζικά και σήμερα οι πληροφορίες για τα αρχεία κρατούνται σε μια κεντρική βάση δεδομένων.

2.2.2 Αρχιτεκτονική του συστήματος

Το σύστημα CASTOR έχει σπονδυλωτή (modular) αρχιτεκτονική με μια κεντρική βάση δεδομένων για τη διαχείριση πληροφοριών σχετικά με το σύστημα. Τα κυριότερα τμήματα του συστήματος μπορούν να χωριστούν σε πέντε βασικές ψηφίδες όπως φαίνεται και στο σχήμα 2.3.



Σχήμα 2.3: Αρχιτεκτονική του ιεραρχικού συστήματος αποθήκευσης CASTOR

2.2.2.1 Stager

Η μονάδα Stager είναι υπεύθυνη για τη διαχείριση των ομάδων των σκληρών δίσκων στο υψηλότερο επίπεδο της ιεραρχίας του συστήματος αποθήκευσης. Οι λειτουργίες της είναι η δέσμευση χώρου για την αντιγραφή αρχείων από τις μαγνητικές ταινίες στους σκληρούς δίσκους, η διατήρηση ενός καταλόγου με τα αρχεία που βρίσκονται αποθηκευμένα στους σκληρούς δίσκους του συστήματος και η διαγραφή αρχείων από τους δίσκους σε περίπτωση μειωμένου αποθηκευτικού χώρου.

2.2.2.2 Name Server

Ο βασικός ρόλος αυτής της ψηφίδας είναι η δημιουργία ενός ενιαίου καταλόγου με αρχεία (file directory) πίσω από το οποίο είναι κρυμμένα τα τριτογενή μέσα αποθήκευσης. Επίσης η

ψηφίδα αυτή διατηρεί ένα χάρτη που αντιστοιχεί τα διάφορα αρχεία του συστήματος σε μια τοποθεσία στα τριτογενή μέσα αποθήκευσης.

2.2.2.3 Τριτογενή μέσα αποθήκευσης – Μαγνητικές ταινίες

Στο χαμηλότερο επίπεδο της ιεραρχίας του συστήματος συναντάμε βιβλιοθήκες (jukeboxes) με μαγνητικές ταινίες, στις οποίες αποθηκεύονται τα δεδομένα. Το χρηματικό κόστος αποθήκευσης ανά Gigabyte στις ταινίες είναι ακόμα και σήμερα πολύ χαμηλότερο από αυτό στους σκληρούς δίσκους. Παράλληλα τα μαγνητικά μέσα αποθήκευσης θεωρούνται πιο αξιόπιστα σε σχέση με τα οπτικά. Παρόλα αυτά το κόστος ανάγνωσης αρχείων από τις μαγνητικές ταινίες ανέρχεται στην τάξη των λεπτών και όχι δευτερολέπτων σε σχέση με τους οπτικούς δίσκους.

2.2.2.4 Βάση δεδομένων

Η βάση δεδομένων είναι η κεντρική ψηφίδα του συστήματος CASTOR. Χρησιμοποιείται την αποθήκευση της κατάστασης του ίδιου του συστήματος αλλά και των διεργασιών που τρέχουν σε αυτό.

2.2.2.5 Client

Η τελευταία συνιστώσα του συστήματος είναι ο Client. Χρησιμοποιείται από εφαρμογές ή χρήστες για την επικοινωνία με το σύστημα. Μέσω αυτού μια εφαρμογή ζητάει ένα υπάρχον ή δημιουργεί ένα νέο αρχείο στο σύστημα.

2.2.3 Δομή του υψηλότερου επιπέδους της ιεραρχίας

Το υψηλότερο επίπεδο μνήμης του συστήματος CASTOR χωρίζεται σε διαφορετικές κλάσεις υπηρεσιών (Service Classes). Η διαφοροποίηση των κλάσεων έγκειται στις υπηρεσίες τις οποίες παρέχουν προς τους χρήστες αλλά και τη γενικότερη λειτουργία τους. Για παράδειγμα τα προνόμια και ο διαθέσιμος αποθηκευτικός χώρος κάθε χρήστη (user quota) διαφέρουν από κλάση σε κλάση. Επίσης υπάρχουν ειδικές κλάσεις, οι οποίες προορίζονται για την άμεση αποθήκευση πειραματικών δεδομένων (raw data) και τη μετέπειτα αντιγραφή και κατανομή τους στις κασέτες του χαμηλότερου επιπέδου. Τέλος υπάρχουν κλάσεις που περιέχουν αρχεία των χρηστών (user scratch files), τα οποία δεν αποθηκεύονται στις κασέτες του χαμηλότερου επίπεδου. Κάθε κλάση αποτελείται από διαφορετικές ομάδες δίσκων και χωρίζεται σε συστήματα αρχείων (filesystems). Η συγκεκριμένη αρχιτεκτονική της μνήμης επιτρέπει καλύτερο διαχωρισμό του αποθηκευτικού χώρου και παρέχει μεγαλύτερη ευελιξία στη διαχείριση αυτού.

2.3 Στατικές πολιτικές αντικατάστασης αρχείων

Όπως αναφέρθηκε και στην προηγούμενη υποενότητα, το επίπεδο δευτερογενούς αποθηκευτικού χώρου χρησιμοποιείται όπως η κρυφή μνήμη στα τυπικά συστήματα ηλεκτρονικών υπολογιστών. Όσο καλύτερη διαχείριση των αρχείων γίνεται στο επίπεδο αυτό τόσο βελτιώνεται η απόδοση του συστήματός μας καθώς μειώνεται ο χρόνος προσπέλασης δεδομένων σε αργά αποθηκευτικά μέσα. Πολλές από τις στατικές πολιτικές που χρησιμοποιούνται σε ένα τυπικό σύστημα υπολογιστή χρησιμοποιούνται και στα ιεραρχικά συστήματα αποθήκευσης. Για το λόγο αυτό θα παρουσιάσουμε σύντομα τις πιο γνωστές από αυτές.

2.3.1 OPT ή Αλγόριθμος του Belady

Η βέλτιστη πολιτική αντικατάστασης αρχείων επιλέγει τα αρχεία που θα διαγράψει λαμβάνοντας υπόψη πότε αυτά θα προσπελαστούν στο μέλλον. Αυτό που θα χρησιμοποιηθεί αργότερα στο μέλλον είναι και αυτό που επιλέγεται για αντικατάσταση. Όπως έχει αποδειχθεί στο [3] η συγκεκριμένη πολιτική αντικατάστασης έχει τη βέλτιστη δυνατή απόδοση σε μνήμες που έχουμε αντικατάσταση μπλοκ μνήμης ενιαίου μεγέθους. Αντίθετα στο [17] αποδεικνύεται ότι ο συγκεκριμένος αλγόριθμος δεν είναι βέλτιστος σε συστήματα με μη ομοιόμορφο μέγεθος αρχείων. Όπως είναι προφανές σε ένα πραγματικό σύστημα δεν είναι δυνατό να γνωρίζουμε τις μελλοντικές αιτήσεις για αρχεία, επομένως η συγκεκριμένη πολιτική χρησιμοποιείται σε προσομοιωτές για την αξιολόγηση των υπολοίπων.

2.3.2 RAND

Η συγκεκριμένη πολιτική αντικατάστασης επιλέγει τυχαία τα αρχεία που θα διαγράψει, όπως άλλωστε υποδηλώνει και το όνομά της. Καθώς η απόφαση είναι εντελώς τυχαία, κάθε αποδοτική πολιτική αντικατάστασης πρέπει να λειτουργεί μακράν καλύτερα από τη RAND. Επομένως η RAND αποτελεί το κάτω φράγμα για όλες τις πολιτικές.

2.3.3 LRU

Η στρατηγική Least Recently Used (LRU) αυτή είναι ίσως η πιο δημοφιλής ανάμεσα στις στατικές πολιτικές αντικατάστασης. Το μοναδικό κριτήριο που λαμβάνει υπόψη είναι ο χρόνος. Σύμφωνα με την LRU, αντικαθίστανται τα αρχεία στα οποία δεν έγινε αναφορά για το μεγαλύτερο χρονικό διάστημα. Όπως είναι προφανές η συγκεκριμένη πολιτική εκμεταλλεύεται τη χρονική τοπικότητα των δεδομένων. Παρά την απλότητα της, η LRU παρουσιάζει τα επόμενα παθολογικά σενάρια. Κατά το πρώτο το μέγεθος του συνόλου των αρχείων μιας ομάδας εργασιών είναι μεγαλύτερο από την κρυφή μνήμη και η εφαρμογή

παρουσιάζει ένα επαναλαμβανόμενο μοτίβο αιτήσεων. Το σενάριο αυτό μας οδηγεί σε συνεχόμενες αστοχίες της μνήμης. Το δεύτερο παθολογικό σενάριο προκύπτει από το γεγονός ότι η LRU λαμβάνει υπόψη μόνο το χρόνο και όχι το μέγεθος των αρχείων που διαγράφει. Ας υποθέσουμε ότι έχουμε μια μνήμη, όπως φαίνεται στον πίνακα 2.1, οργανωμένη σύμφωνα με την LRU.

…		
f	3	10
e	1	11
d	6	16
c	2	17
b	2	24
a	1	27
objected	**size**	**last reference**

Πίνακας 2.1: Μνήμη οργανωμένη σύμφωνα με την πολιτική LRU

Εάν η μνήμη μας είναι πλήρης και θέλουμε να αντιγράψουμε ένα αρχείο μεγέθους 6 τότε η πολιτική LRU θα διαγράψει από τη μνήμη τα αρχεία a, b, c και d ώστε να δημιουργήσει τον απαραίτητο ελεύθερο χώρο για την αντιγραφή του νέου αρχείου. Βλέπουμε όμως ότι θα αρκούσε μόνο η διαγραφή του αρχείου d και η αντικατάστασή του με το νέο αρχείο. Επομένως η πολιτική LRU διαγράφει περισσότερα αρχεία από όσα πραγματικά χρειάζεται αυξάνοντας έτσι την πιθανότητα να έχουμε αστοχίες στο μέλλον. Από το αντίστοιχο πρόβλημα πάσχουν όλες οι πολιτικές που λαμβάνουν υπόψη τους μόνο το χρόνο ως κριτήριο.

2.3.4 FIFO

Η στρατηγική First in first out (FIFO) βασίζεται στο χρόνο παραμονής ενός αρχείου στο σύστημα. Σύμφωνα με τη στρατηγική αυτή, τα αρχεία που διαγράφονται είναι τα παλαιότερα που είναι αποθηκευμένα στο σύστημα. Η στρατηγική FIFO βασίζεται στη λογική ότι τα αρχεία που βρίσκονται για μεγαλύτερο χρονικό διάστημα στη μνήμη έχουν και τις λιγότερες πιθανότητες για αναφορά στο μέλλον.

2.3.5 LFU

Η στρατηγική Least Frequently Used (LFU) προσπαθεί να διατηρεί αποθηκευμένα στα υψηλότερα επίπεδα της ιεραρχίας τα πιο δημοφιλή αρχεία, αυτά δηλαδή με το μεγαλύτερο αριθμό προσπελάσεων. Σύμφωνα με τη στρατηγική αυτή τα αρχεία που επιλέγονται να διαγραφούν είναι αυτά που χρησιμοποιούνται λιγότερο. Η πολιτική LFU πάσχει από το παθολογικό σενάριο, όπου αρχικά δημοφιλής αρχεία με ιδιαίτερα υψηλό αριθμό αναφορών δεν προσπελαύνονται ξανά στο μέλλον ενώ συνεχίζουν να παραμένουν στα υψηλά επίπεδα της ιεραρχίας μνήμης.

2.3.6 Ομοιότητες και διαφορές με την εργασία μας

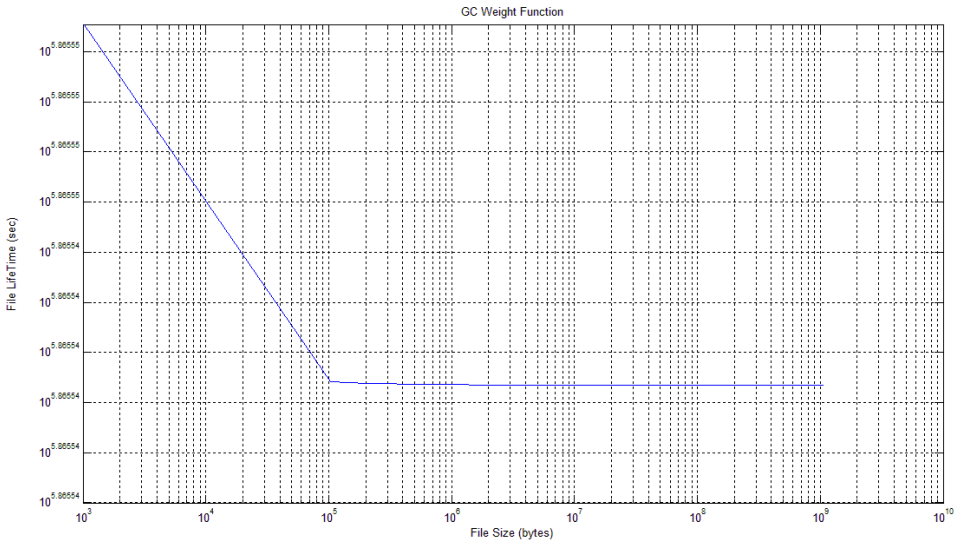
Οι περισσότερες από τις στατικές πολιτικές αντικατάστασης αρχείων που παρουσιάστηκαν χρησιμοποιούν μόνο μια παράμετρο για να επιλέξουν τα αρχεία που θα διαγράψουν από την κρυφή μνήμη. Πιστεύουμε ότι μια βέλτιστη πολιτική αντικατάστασης πρέπει να ενσωματώνει όσο το δυνατό περισσότερες παραμέτρους σχετικά με την κατάσταση του συστήματος και την κατάσταση των αρχείων σε αυτό. Ο αλγόριθμος ενισχυτικής μάθησης που αναπτύσσουμε λαμβάνει υπόψη του πολλαπλές παραμέτρους όπως το μέγεθος των αρχείων, ο χρόνος προσπέλασης και διαθεσιμότητας των αρχείων και ο αριθμός των αναφορών ενός αρχείου.

2.4 Πολιτικές αντικατάστασης αρχείων στο σύστημα CASTOR

Στο σύστημα CASTOR η αντικατάσταση και διαγραφή αρχείων ενεργοποιείται αυτόματα ανά τακτά χρονικά διαστήματα και μόνο εάν ικανοποιούνται ορισμένες συνθήκες. Μόλις ένας server – filesystem φτάσει το 90% της συνολικής χωρητικότητάς του ενεργοποιείται η διαγραφή αρχείων με σκοπό η τρέχουσα χωρητικότητα να πέσει στο 85% της συνολικής. Κατά τη διάρκεια της διαγραφής αρχείων το σύστημα συνεχίζει να εξυπηρετεί τις εισερχόμενες αιτήσεις και να αντιγράφει εάν χρειαστεί αρχεία στο υψηλότερο επίπεδο της μνήμης. Στην περίπτωση που η τρέχουσα χωρητικότητα ενός server ξεπεράσει το 95% της συνολικής, ενώ εκτελείται η διαγραφή αρχείων, τότε απενεργοποιείται η αντιγραφή αρχείων στο συγκεκριμένο server μέχρι να επανέλθει η τρέχουσα χωρητικότητά του στα επιθυμητά επίπεδα.

Στο σύστημα CASTOR οι διαχειριστές έχουν τη δυνατότητα να καθορίσουν χειροκίνητα την πολιτική αντικατάστασης των αρχείων. Συγκεκριμένα μπορούν να επιλέξουν ανάμεσα στις πολιτικές FIFO, LRU και μιας custom πολιτικής (“Default” Policy) που έχει αναπτυχθεί στο CERN. Η default πολιτική βασίζεται σε μια ad-hoc συνάρτηση για τον υπολογισμό του

κόστους διαγραφής κάθε αρχείου. Στο σχήμα 2.4 βλέπουμε το κόστος διαγραφής συναρτήσει του μεγέθους του αρχείου. Είναι προφανής η τάση της συγκεκριμένης πολιτικής να διατηρεί μικρά αρχεία για μεγάλο χρονικό διάστημα στη μνήμη.



Σχήμα 2.4: Φαίνεται η Ad-hoc συνάρτηση της default πολιτικής αντικατάστασης αρχείων

Στο σημείο αυτό πρέπει να αναφέρουμε ότι είναι δυνατή η χρήση διαφορετικών πολιτικών αντικατάστασης ανά κλάση υπηρεσιών (service class). Επίσης η διαγραφή αρχείων εκτελείται ανά filesystem βάσει της πολιτικής αντικατάστασης αρχείων που αντιστοιχεί στην κλάση υπηρεσιών, στην οποία ανήκει το συγκεκριμένο filesystem.

2.5 Σχετικές Εργασίες

Στη βιβλιογραφία βρέθηκαν αρκετές εργασίες που προσεγγίζουν το θέμα της διαχείρισης αρχείων σε ιεραρχικά συστήματα από διαφορετικές σκοπιές. Στην ενότητα αυτή θα περιγράψουμε σύντομα τις διαφορετικές λύσεις που έχουν προταθεί καθώς και τις ομοιότητες και διαφορές με την προσέγγιση που ακολουθείται στη συγκεκριμένη εργασία.

2.5.1 Η προσαρμοστική πολιτική ObjectLRU (OLRU)

Στο [2] οι Hahn et al. παρουσιάζουν ένα νέο προσαρμοστικό αλγόριθμο αντικατάστασης αρχείων για ιεραρχικά συστήματα αποθήκευσης. Σε αντίθεση με τους συμβατικούς και στατικούς αλγορίθμους αντικατάστασης αρχείων, ο OLRU λαμβάνει υπόψη του και άλλες παραμέτρους σχετικές με τα υποψήφια προς αντικατάσταση αρχεία. Στη συνέχεια παρουσιάζουμε τα κριτήρια που εισάγουν οι Hahn et al στο πρόβλημα βελτιστοποίησης και συγκεκριμένα ελαχιστοποίησης που καλείται να επιλύσει ο OLRU.

Έστω ότι η πιθανότητα μελλοντικής αναφοράς είναι ίδια για όλα τα αρχεία που είναι αποθηκευμένα στο υψηλότερο επίπεδο της ιεραρχίας. Όπως είναι αναμενόμενο η πιθανότητα μιας ευστοχίας της μνήμης είναι υψηλότερη αν διατηρήσουμε όσο το δυνατόν περισσότερα αρχεία στη μνήμη. Ένα από τα κριτήρια βελτιστοποίησης δίνεται από τον επόμενο τύπο

$$K_n = n - 1 \qquad (2.1)$$

Σύμφωνα με το επόμενο κριτήριο, αν s_{rep} είναι το συνολικό μέγεθος της μνήμης που πρέπει να απελευθερώσουμε τότε το πρέπει το άθροισμα του μεγέθους των αρχείων που θα διαγράψουμε να είναι κοντά ή ίσο με το s_{rep}, δηλαδή

$$K_s = \sum_{i=1}^{n} s_i - s_{rep} \quad (2.2)$$

Εκτός των παραπάνω κριτηρίων εισάγεται και ένα νέο κριτήριο, το οποίο ενσωματώνει την παράμετρο του χρόνου όπως αυτή χρησιμοποιείται από την πολιτική LRU. Εάν p_i είναι η θέση ενός αρχείου στη στοίβα που διατηρεί η πολιτική LRU για τα αρχεία που βρίσκονται στο υψηλότερο επίπεδο της μνήμης, τότε έχουμε το επόμενο κριτήριο

$$K_p = \frac{1}{n(n+1)} \sum_{i=1}^{n} (p_{max} - p_i + 1) - \frac{1}{2} \quad (2.3)$$

Ο παραπάνω τύπος μας δίνει ένα ασθενές κριτήριο αξιολόγησης για τις μελλοντικές αναφορές σε ένα αρχείο του συστήματος, αφού η θέση του στη στοίβα της πολιτικής LRU σχετίζεται με τις αναφορές σε αυτό. Η χρήση του κριτηρίου αυτού εμποδίζει τη διατήρηση ιδιαίτερα παλιών αρχείων στο υψηλότερο επίπεδο της ιεραρχίας μνήμης.

Το τελευταίο κριτήριο που λαμβάνει υπόψη του ο αλγόριθμος OLRU είναι αυτό του κόστους προσπέλασης των αρχείων. Στα ιεραρχικά συστήματα αποθήκευσης ο όρος κόστος αναφέρεται στο χρονικό διάστημα ανάμεσα στην αίτηση για ένα αρχείο και στην ολοκλήρωση της αντιγραφής του στο υψηλότερο επίπεδο της ιεραρχίας μνήμης. Είναι προφανές ότι το κόστος εξαρτάται άμεσα από τη συνολική κατάσταση του συστήματος. Ο τύπος για το τελευταίο κριτήριο είναι

$$K_k = \sum_{i=1}^{n} k_i \quad (2.4)$$

Συνδυάζοντας τα παραπάνω κριτήρια έχουμε την επόμενη συνάρτηση βάρους

$$W = c_n K_n + c_s K_s + c_r K_r + c_p K_p + c_k K_k \quad (2.5)$$

την οποία καλείται να ελαχιστοποιήσει σε κάθε απόφαση διαγραφής ο αλγόριθμος OLRU. Χρησιμοποιώντας τις σταθερές c_i στην παραπάνω εξίσωση, ο αλγόριθμος OLRU μπορεί να προσαρμόζει την πολιτική αντικατάστασης ανάλογα με την κατάσταση του συστήματος. Στην εργασία τους οι Hahn et al. επιλέγουν τη χρήση γενετικών αλγορίθμων για την online ρύθμιση των παραμέτρων αυτών κατά το χρόνο λειτουργίας του συστήματος.

Μια απλοϊκή προσέγγιση για την επίλυση του παραπάνω προβλήματος ελαχιστοποίησης θα ήταν να υπολογίσουμε την τιμή W για όλους τους δυνατούς συνδυασμούς και να επιλέξουμε αυτόν με το ελάχιστο W. Όμως αν υπάρχουν n αρχεία στη μνήμη οι δυνατοί συνδυασμοί είναι 2^n καθιστώντας αδύνατο τον παραπάνω τρόπο απόφασης. Ένας τρόπος για να μειώσουμε δραστικά το πλήθος των συνδυασμών που εξετάζονται είναι να λαμβάνουμε υπόψη για διαγραφή μόνο τα αρχεία στο τέλος της LRU-στοίβας. Είναι προφανές ότι η παραπάνω προσέγγιση οδηγεί σε μια υποβέλτιστη λύση του προβήματος. Παρόλα αυτά η πολιτική OLRU παρουσιάζει αυξημένη απόδοση, όπως φαίνεται από πειραματικά δεδομένα, σε σχέση με την κλασσική πολιτική LRU, ειδικά όταν το μέγεθος των αρχείων που πρέπει να διαγραφούν από τη μνήμη είναι μεγάλο.

2.5.1.1 Ομοιότητες και διαφορές

Η νέα προσαρμοστική πολιτική που παρουσιάζουμε στην παρούσα εργασία λαμβάνει υπόψη αντίστοιχα μεγέθη με την εργασία στο [2]. Παράμετροι όπως το μέγεθος και το κόστος προσπέλασης ενσωματώνονται και στην πολιτική αντικατάστασης που παρουσιάζεται στην παρούσα εργασία. Σε αντίθεση με την παραπάνω εργασία, η παρούσα αναφορά ακολουθεί διαφορετική προσέγγιση για την επίλυση του προβλήματος βελτιστοποίησης που αντιμετωπίζουμε. Συγκεκριμένα χρησιμοποιείται η μέθοδος της ενισχυτικής μάθησης, η θεωρία της οποίας πηγάζει από το δυναμικό προγραμματισμό.

2.5.2 Μια πολιτική αντικατάστασης για ιεραρχικά συστήματα αποθήκευσης συνεχών αρχείων πολυμέσων

Οι περισσότερες διαδικτυακές εφαρμογές πολυμέσων απαιτούν πολύ μεγάλη χωρητικότητα για την αποθήκευση και διαχείριση δεκάδων Terabytes ή και Petabytes δεδομένων καθιστώντας ιδανική τη χρήση ιεραρχικών συστημάτων αποθήκευσης. Επειδή ο χρόνος απόκρισης είναι ιδιαίτερα σημαντικός σε εφαρμογές τέτοιου τύπου, είναι απαραίτητη η χρήση μιας βέλτιστης πολιτικής αντικατάστασης αρχείων στο υψηλότερο επίπεδο της ιεραρχίας. Στο [6] οι Xu et al. προτείνουν τον αλγόριθμο Least Waiting Probability (LWP), ο

οποίος χρησιμοποιεί το μέτρο του εκτιμώμενου χρόνου αναμονής για να αποφασίσει ποια αρχεία θα διαγράψει από το υψηλότερο επίπεδο της ιεραρχίας μνήμης. Πριν παρουσιάσουμε τον αλγόριθμο θα ορίσουμε τις μετρικές που χρησιμοποιεί.

2.5.2.1 Βαθμός αναμονής του χρήστη και άλλα μέτρα

Ο βαθμός αναμονής του χρήστη είναι μια μετρική που εισάγουν οι Xu et al. για την αξιολόγηση της συνολικής απόδοσης του συστήματος. Έστω ότι το O_i υποδηλώνει το i-οστό αντικείμενο στο σύστημα, το F υποδηλώνει ένα αρχείο του συστήματος ενώ το D το σύνολο των αρχείων που βρίσκονται στους οπτικούς δίσκους στο υψηλότερο επίπεδο της ιεραρχίας και το $F_{i,j} \in O_i$ υποδηλώνει το j-οστό αρχείο του αντικειμένου O_i. Κάθε αρχείο χαρακτηρίζεται από το μέγεθός του s, τον αναμενόμενο χρόνο προσπέλασης EAT (estimated access time) και την πιθανότητα προσπέλασης Pa.

Ορισμός 1. File Ready Time (FRT): Ο χρόνος FRT αντιστοιχεί με το χρονικό διάστημα από την αίτηση για ένα αρχείο μέχρι τη μεταφορά του και πλήρη αντιγραφή του στο υψηλότερο επίπεδο της ιεραρχίας αποθήκευσης.

Ορισμός 2. Expected Access Time (EAT): Ο χρόνος $EAT(F_{i,j})$ ενός αρχείου $F_{i,j}$ είναι το χρονικό διάστημα από την αρχική αίτηση για ένα αντικείμενο O_i μέχρι την αίτηση για το αρχείο $F_{i,j}$ του αντικειμένου αυτού και δίνεται από τον επόμενο τύπο

$$EAT(F_{i,j}) = \begin{cases} 0, j = 0 \\ EAT(F_{i,j-1}) + L(F_{i,j-1}), j > 0 \end{cases} \quad (2.6)$$

όπου L(F) είναι ο χρόνος αναπαραγωγής του αρχείου πολυμέσων F.

Ορισμός 3. User Waiting Rate (UWR): Το μέτρο UWR μας παρέχει μια εκτίμηση για την αναμονή του χρήστη και σχετίζεται με το χρόνο αναμονής του χρήστη από την αρχική αίτηση για ένα αρχείο F μέχρι την έναρξη αναπαραγωγής αυτού. Το νέο αυτό μέτρο χρησιμοποιείται για την αξιολόγηση της απόδοσης του συστήματος. Είναι προφανές ότι ο βαθμός UWR εξαρτάται άμεσα από τον αριθμό των αστοχιών στη μνήμη του συστήματος.

Ορισμός 4. File Waiting Probability (FWP): Για ένα αρχείο F η πιθανότητα αναμονής είναι η πιθανότητα να ισχύει FRT(F) > EAT, δηλαδή η πιθανότητα ο χρήστης που ζήτησε το αρχείο F να περιμένει για την αναπαραγωγή του αρχείου αφού αυτό δεν ήταν στο υψηλό επίπεδο της ιεραρχίας και πρέπει να μεταφερθεί από κάποιο τριτογενές μέσο αποθήκευσης. Θεωρούμε την κατανομή της χρονικής απόκρισης του συστήματος γνωστή και συμβολίσουμε με R(t) την πιθανότητα να ισχύει FRT(F) > t τότε η πιθανότητα αναμονής για το αρχείο F είναι:

$$FWP(F) = Pa * R(EAT) \qquad (2.7)$$

Καθώς η κατανομή της χρονικής απόκρισης του συστήματος είναι ιδιαίτερα περίπλοκη οι Xu et al. χρησιμοποιούν μια προσέγγισης αυτής. Το μέτρο που μόλις παρουσιάστηκε χρησιμοποιείται από τον αλγόριθμο LWP για την επιλογή των αρχείων προς αντικατάσταση.

2.5.2.2 Ο αλγόριθμος Least Waiting Probability (LWP)

Ο αλγόριθμος LWP χρησιμοποιεί το μέτρο της πιθανότητας αναμονής για ένα αρχείο για να αποφασίσει ποια αρχεία θα διαγράψει από το υψηλό επίπεδο της ιεραρχίας μνήμης. Θεωρούμε ότι η πιθανότητα προσπέλασης ενός αρχείου Pa αντικαθίσταται από το πλήθος των προσπελάσεων του αρχείου αυτού Ca.

Αλγόριθμος 2.1 Least Waiting Probability

1. Το σύστημα λαμβάνει μια αίτηση για ένα αντικείμενο Ο. Ανανεώνει τον μετρητή Ca των αρχείων F που ανήκουν στο Ο. Για κάθε αρχείο του αντικειμένου Ο ανανεώνει την πιθανότητα FWP.
2. Εάν όλα τα αρχεία του αντικειμένου υπάρχουν στο δίσκο αρχίζει η αναπαραγωγή αυτού.
3. Διαφορετικά για κάθε αρχείο F_i που ανήκει στο Ο και δεν βρίσκεται στο δίσκο ζητείται η μεταφορά του από τα τριτογενή μέσα αποθήκευση. Για κάθε αρχείο υπολογίζεται ο χρόνος FRT. Εάν ισχύει $FRT_i < EAT_i$ για όλα τα αρχεία ξεκινά η αναπαραγωγή του αντικειμένου Ο, διαφορετικά αυξάνει ο μετρητής του χρόνου αναμονής. Εάν χρειαστεί διαγράφονται τα αρχεία με τη μικρότερη πιθανότητα FWP.
4. Κάθε 24 ώρες μείωσε τους μετρητές προσπέλασης όλων των αρχείων κατά 1 και ανανέωσε την πιθανότητα FWP αυτών.

Όπως φαίνεται από τα πειραματικά δεδομένα στο [6] ο αλγόριθμος LWP παρουσιάζει αυξημένη απόδοση σε σχέση με τη στατική πολιτική αντικατάστασης LFU κατά 13%.

2.5.2.3 Ομοιότητες και διαφορές

Ο αλγόριθμος που παρουσιάζεται στο [6] εισάγει ένα νέο μέτρο διαφορετικό του χρόνου, το οποίο χρησιμοποιείται για την αντικατάσταση αρχείων. Αντίθετα με την παρούσα εργασία ο LWP ακολουθεί διαφορετική προσέγγιση καθώς δεν είναι προσαρμοστικός. Φυσικά και εμείς αξιολογούμε τα μεγέθη που χρησιμοποιεί ο LWP, όπως ο αριθμός των προσπελάσεων κάθε αρχείου και ο χρόνος αναμονής εξυπηρέτησης του χρήστη.

2.5.3 Μια προσαρμοστική πολιτική online μεταφοράς αρχείων σε ιεραρχικά συστήματα αποθήκευσης πολλαπλών επιπέδων

Όπως αναφέρθηκε προηγουμένως, ο συνεχώς αυξανόμενος όγκος των δεδομένων οδήγησε στη χρήση ιεραρχικών συστημάτων αποθήκευσης πολλαπλών επιπέδων. Ένας από τους

σημαντικότερους δείκτες της απόδοσης ενός τέτοιου συστήματος είναι ο χρόνος απόκρισης των αιτήσεων I/O. Ο χρόνος απόκρισης ελαχιστοποιείται εάν το σύστημα παρουσιάζει αυξημένο ποσοστό επιτυχιών στα υψηλότερα επίπεδα της μνήμης. Στο [8] ο David Vengerov παρουσιάζει μια νέα προσαρμοστική πολιτική μεταφοράς των αρχείων ανάμεσα στα διαφορετικά επίπεδα ενός συστήματος πολλαπλών επιπέδων μνήμης.

Τα ιεραρχικά συστήματα πολλαπλών επιπέδων διαθέτουν μεγάλο αριθμό παραμέτρων, οι οποίες επηρεάζουν άμεσα την απόδοσή τους. Η χειροκίνητη ρύθμιση των παραμέτρων αυτών από τους διαχειριστές οδηγεί συνήθως σε υπο-βέλτιστες καταστάσεις του συστήματος και κατά συνέπεια σε αυξημένους χρόνους απόκρισης. Αντί αυτού οι παράμετροι κάθε επίπεδου του συστήματος θα μπορούσαν να ελέγχονται από ένα πράκτορα-ελεγκτή, ο οποίος με χρήση μηχανικής μάθησης θα επέλεγε κάθε φορά τις δράσεις εκείνες που βελτιστοποιούν την απόδοση του συστήματος.

Θεωρούμε ότι για κάθε επίπεδο i του συστήματος έχουμε ένα διάνυσμα s_i που περιγράφει την κατάσταση του επιπέδου. Επίσης υπάρχει ένας πράκτορας A_i, ο οποίος ενεργεί στο επίπεδο i. Συγκεκριμένα κάθε πράκτορας A_i καλείται να αποφασίσει εάν τα αρχεία που βρίσκονται στο επίπεδο i πρέπει να μεταφερθούν σε ένα υψηλότερο επίπεδο της ιεραρχίας λόγω αυξημένης ζήτησης. Για τη λήψη αυτών των αποφάσεων κάθε πράκτορας A_i μαθαίνει μια συνάρτηση κόστους $C_i(s_i)$, όπου s είναι το διάνυσμα κατάστασης του επιπέδου i. Η συνάρτηση κόστους προβλέπει το μελλοντικό μέσο κόστος των αιτήσεων προσπέλασης αρχείων στο επίπεδο I (average weighted response time - AWRT).

Το διάνυσμα κατάστασης ενός επιπέδου αποτελείται από μεταβλητές, οι οποίες εξαρτώνται από το μέγεθος των αρχείων που είναι αποθηκευμένα στο επίπεδο, τον αναμενόμενο χρόνο προσπέλασης των αρχείων του επιπέδου και τον εκτιμώμενο χρόνο αναμονής των αιτήσεων προσπέλασης αρχείων του επιπέδου. Μόλις το σύστημα λάβει μια αίτηση για ανάγνωση ή επεξεργασία κάποιου αρχείου που δε βρίσκεται στο υψηλότερο επίπεδο της ιεραρχίας, αποφασίζει βάσει των προσεγγιστικών συναρτήσεων κόστους C εάν το αρχείο αυτό πρέπει να παραμείνει στο επίπεδο που βρίσκεται ή πρέπει να μεταφερθεί στο αμέσως υψηλότερο και ταχύτερο επίπεδο της μνήμης.

Όσο πιο ακριβής είναι οι προσεγγίσεις των συναρτήσεων κόστους τόσο καλύτερες είναι οι αποφάσεις που λαμβάνονται για τη μεταφορά των αρχείων. Στο [8] η βέλτιστη συνάρτηση κόστους προσεγγίζεται από ένα γραμμικό συνδυασμό κατάλληλων γραμμικώς ανεξάρτητων συναρτήσεων βάσεων. Οι συναρτήσεις βάσεις προκύπτουν ως κανονικοποιημένα βάρη ενός συνόλου ασαφών κανόνων (fuzzy rule-base). Για την ανανέωση της τιμής κάθε κανόνα του συνόλου σαφών κανόνων και επομένως τη βελτίωση των προσεγγιστικών συναρτήσεων κόστους χρησιμοποιείται ενισχυτική μάθηση. Όπως φαίνεται από τα πειραματικά δεδομένα που παρουσιάζονται στο [8], η παραπάνω προσαρμοστική πολιτική ανακατανομής αρχείων

σε ιεραρχικά συστήματα οδηγεί σε μείωση του χρόνου απόκρισης του συστήματος έως και 35% σε σχέση με τις στατικές πολιτικές διαχείρισης αρχείων.

Στο σημείο αυτό θεωρήσαμε σκόπιμο να μην αναπτύξουμε το θεωρητικό μέρος του [8] καθώς αυτό επικαλύπτεται σε μεγάλο βαθμό αυτό της παρούσας εργασίας, το οποίο αναπτύσσεται στο επόμενο κεφάλαιο.

2.5.3.1 Ομοιότητες και διαφορές

Στο [8] προτείνεται μια προσαρμοστική πολιτική ανακατανομής αρχείων σε ιεραρχικά συστήματα. Όπως και στην παρούσα εργασία έτσι και στο [8] η πολιτική που προτείνεται βασίζεται στη μεθοδολογία της ενισχυτικής μάθησης. Το [8] χρησιμοποιήθηκε ως βάση για το ξεκίνημα της έρευνάς μας λόγω της μεθοδολογίας που χρησιμοποιεί και της αρχιτεκτονικής ασαφών κανόνων που προτείνει (fuzzy rule-base architecture). Παρά το κοινό θεωρητικό υπόβαθρο η προσαρμοστική πολιτική αντικατάστασης που αναπτύσσουμε χρησιμοποιεί διαφορετικό αλγόριθμο και ενσωματώνει ένα εντελώς διαφορετικό μηχανισμό λήψης αποφάσεων. Περισσότερα για αυτή θα παρουσιαστούν στο κεφάλαιο 4.

2.5.4 Χρήση ενισχυτικής μάθησης για επιλογή στατικών πολιτικών

Μια διαφορετική προσέγγιση για την επίλυση του προβλήματος της αντικατάστασης αρχείων παρουσιάζεται στο [24] από τους Gramacy et al. όπου χρησιμοποιείται η ενισχυτική μάθηση για την επιλογή της κατάλληλης κάθε φορά στατικής πολιτικής αντικατάστασης αρχείων. Κάθε φορά ο πράκτορας μάθησης που ελέγχεται από την *master policy*, επιλέγει μια από τις στατικές πολιτικές που "γνωρίζει". Οι στατικές πολιτικές που γνωρίζει ο πράκτορας λαμβάνουν υπόψη διαφορετικά κριτήρια όπως οι χρονικές αναφορές στα αντικείμενα αλλά και το μέγεθος αυτών.

Η πολιτική που θα χρησιμοποιηθεί καθορίζεται από τη νέα προσαρμοστική κύρια πολιτική. Για την αξιολόγηση των επιμέρους πολιτικών το σύστημα διατηρεί μια ξεχωριστή εικονική κρυφή μνήμη, προσομοιώνοντας έτσι τη λειτουργία κάθε πολιτικής. Ο πράκτορας αξιολογεί τις επιμέρους πολιτικές αντικατάστασης των αρχείων εξετάζοντας το miss rate των εικονικών κρυφών μνημών και επιλέγει αυτή με το μικρότερο κόστος προσπέλασης των αρχείων.

2.5.4.1 Ομοιότητες και διαφορές

Σε αντίθεση με το [24] η παρούσα εργασία έχει ως στόχο την δημιουργία μιας νέας πολιτικής αντικατάστασης αρχείων, η οποία θα λαμβάνει υπόψη παράγοντες όπως το μέγεθος και τη συχνότητα των αναφορών στα αρχεία. Επίσης και εμείς θα χρησιμοποιήσουμε τη μέθοδο της ενισχυτικής μάθησης για την εκμάθηση της συνάρτησης χρησιμότητας των ζευγών κατάσταση – δράση του συστήματος.

3

Ενισχυτική Μάθηση

Η ιδέα της μάθησης μέσα από την αλληλεπίδραση μας με το περιβάλλον είναι ίσως η πρώτη που μας έρχεται στο μυαλό όταν αναφερόμαστε στη φύση της μάθησης. Κατά τη διάρκεια της μάθησης ο άνθρωπος πάντα παρατηρεί την αντίδραση του περιβάλλοντος στις δράσεις που αυτός εκτελεί και προσπαθεί να κατανοήσει και να επηρεάσει αυτό που συμβαίνει μέσω της συμπεριφοράς του. Στο [9] οι R. Sutton και A. Barto αναπτύσσουν τη θεωρία της ενισχυτικής μάθησης, μιας μορφής μηχανικής μάθησης, η οποία βασίζεται στην παραπάνω ιδέα της αλληλεπίδρασης του περιβάλλοντος με διάφορους πράκτορες μέσω της συμπεριφοράς και των δράσεων των τελευταίων. Στο κεφάλαιο αυτό θα παρουσιάσουμε τη θεωρία, στην οποία βασίζονται οι αλγόριθμοι της ενισχυτικής μάθησης και θα αναλύσουμε ορισμένους από αυτούς.

3.1 Εισαγωγή

Στο κέντρο της θεωρίας της ενισχυτικής μάθησης βρίσκεται η εκμάθηση μιας συμπεριφοράς και πολιτικής επιλογής δράσεων, η οποία αποσκοπεί στη μεγιστοποίηση μιας συνάρτησης κέρδους. Αντίθετα με τις περισσότερες μορφές της μηχανικής μάθησης ο πράκτορας δεν γνωρίζει βάσει κάποιου εξωτερικού ελεγκτή τη βέλτιστη δράση που πρέπει να εκτελέσει αλλά πρέπει να εξάγει τη δική του βέλτιστη πολιτική δράσεων παρατηρώντας παλαιότερες αλληλεπιδράσεις του με το περιβάλλον.

Κάθε πράκτορας πρέπει να έχει μερική ή πλήρη γνώση της κατάστασής του μέσα στο περιβάλλον και ένα συγκεκριμένο στόχο, ο οποίος σχετίζεται με την κατάστασή του. Οι δράσεις που λαμβάνει επηρεάζουν άμεσα την κατάσταση του. Στο σημείο αυτό πρέπει να επισημάνουμε ότι ο πράκτορας μαθαίνει στον ίδιο χώρο με τον οποίο δρα. Επομένως πρέπει να επιλέγει δράσεις, οι οποίες σύμφωνα με την υπάρχουσα γνώση του επιφέρουν μεγάλο κέρδος. Βέβαια εφαρμόζοντας πάντα μια άπληστη πολιτική ο πράκτορας μπορεί να οδηγηθεί σε υποβέλτιστες λύσεις. Για το λόγο αυτό είναι αναγκαία η εξερεύνηση του χώρου των δράσεων με σκοπό την εύρεση νέων, οι οποίες μπορεί μακροπρόθεσμα να οδηγήσουν σε αύξηση του συνολικού κέρδους. Τα παραπάνω περιγράφουν μια από τις μεγαλύτερες προκλήσεις στο πεδίο της ενισχυτικής μάθησης, το πρόβλημα της ισορροπίας ανάμεσα στην αναζήτηση νέας γνώσης και την εκμετάλλευση της υπάρχουσας (exploration vs. exploitation).

Το πρόβλημα της μηχανικής μάθησης σχετίζεται άμεσα με το πρόβλημα του βέλτιστου ελέγχου σε στοχαστικά συστήματα που ικανοποιούν την ιδιότητα Markov. Μάλιστα μπορούμε να θεωρήσουμε ότι οι αλγόριθμοι ενισχυτικής μάθησης συγκαταλέγονται στην ευρύτερη οικογένεια των αλγορίθμων δυναμικού προγραμματισμού. Στο [10] οι Bertsekas και Tsitsiklis παρουσιάζουν μια πιο αυστηρή μαθηματική θεμελίωση της ενισχυτικής μάθησης, η οποία βασίζεται στη θεωρία του δυναμικού προγραμματισμού. Όπως παρατηρούν οι ίδιοι οι μέθοδοι ενισχυτικής μάθησης φαίνονται να είναι αρκετά αποδοτικές σε προβλήματα όπου κλασσικές μέθοδοι δυναμικού προγραμματισμού απέτυχαν να εφαρμοστούν. Το παραπάνω καθιστά τους αλγορίθμους ενισχυτικής μάθησης ιδιαίτερα ενδιαφέροντες και για το λόγο αυτό υπάρχει αρκετή έρευνα γύρω από αυτό το πεδίο.

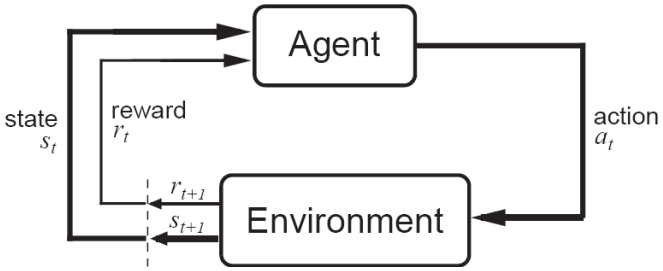
3.2 Το πρόβλημα της ενισχυτικής μάθησης

Η βασική ιδέα του προβλήματος της ενισχυτικής μάθησης είναι η απόκτηση γνώσης μέσω της αλληλεπίδρασης με το περιβάλλον για την επίτευξη ενός συγκεκριμένου στόχου. Θα ονομάζουμε πράκτορα την οντότητα που μαθαίνει και λαμβάνει τις αποφάσεις. Θεωρούμε ότι ο πράκτορας αλληλεπιδρά συνεχώς με το περιβάλλον. Συγκεκριμένα ο πράκτορας επιλέγει πράξεις και τις εκτελεί ενώ το περιβάλλον αντιδρά σε αυτές επιβραβεύοντας κάθε φορά την τελευταία δράση και ενημερώνοντας τον πράκτορα για τη νέα κατάστασή του (trial – error method). Στο σχήμα 3.1 βλέπουμε την αναπαράσταση του προβλήματος σύμφωνα με τους Sutton και Barto.

Η παραπάνω περιγραφή μας θυμίζει το πρόβλημα του δυναμικού προγραμματισμού όπως αυτό διατυπώνεται από τον Bertsekas στο [11]. Σύμφωνα με αυτό, έχουμε το διάνυσμα κατάστασης του συστήματος, το οποίο εξελίσσεται με την πάροδο του χρόνου. Κάθε χρονική στιγμή εισέρχεται στο σύστημα ένα σήμα ελέγχου, το οποίο οδηγεί το σύστημα σε μια νέα κατάσταση. Στο σύστημα μπορεί να εισέρχεται και θόρυβος. Έχουμε ότι

$$x_{k+1} = f_k(x_k, u_k, w_k), \quad k = 0, 1, \ldots, N-1 \text{ (3.1)}$$

όπου x_k είναι το διάνυσμα κατάστασης του συστήματος, u_k το σήμα ελέγχου που επιλέγεται τη χρονική στιγμή k και w_k είναι ο θόρυβος που εισέρχεται στο σύστημα. Η συνάρτηση f_k περιγράφει το σύστημα και συγκεκριμένα το μηχανισμό βάσει του οποίου ανανεώνεται στο διάνυσμα κατάστασης του συστήματος.



Σχήμα 3.1: Σχηματική αναπαράσταση του προβλήματος της ενισχυτικής μάθησης (πηγή [9]).

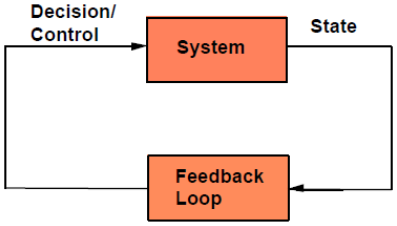
Κάθε αλλαγή του διανύσματος κατάστασης συνοδεύεται και από κάποιο κέρδος. Η συνάρτηση κέρδους είναι αθροιστική και μάλιστα το συνολικό κέρδος είναι

$$g_N(x_N) + \sum_{k=0}^{N-1} g_k(x_k, u_k, w_k), \quad (3.2)$$

όπου $g_N(x_N)$ είναι το τελικό κέρδος που έλαβε το σύστημα στο τέλος της διαδικασίας. Όμως λόγω της ύπαρξης του στοχαστικού θορύβου στο σύστημα το κέρδος είναι μια τυχαία μεταβλητή. Ζητάμε επομένως τη μεγιστοποίηση της εκτιμώμενης τιμής

$$E\{g_N(x_N) + \sum_{k=0}^{N-1} g_k(x_k, u_k, w_k)\} \quad (3.3)$$

του παραπάνω αθροιστικού κέρδους. Στο σχήμα 3.2 βλέπουμε μια απλοϊκή αναπαράσταση του προβλήματος δυναμικού προγραμματισμού.



Σχήμα 3.2: Σχηματική αναπαράσταση του προβλήματος δυναμικού προγραμματισμού.

Όπως βλέπουμε στα προβλήματα δυναμικού προγραμματισμού γνωρίζουμε επακριβώς τη συνάρτηση κέρδους και μπορούμε επιλύοντας το πρόβλημα να βρούμε τα βέλτιστα σήματα ελέγχου που οδηγούν σε μεγιστοποίηση του αθροιστικού κέρδους. Αντίθετα στα προβλήματα ενισχυτικής μάθησης δε γνωρίζουμε επακριβώς τη συνάρτηση κέρδους. Αυτό που προσπαθεί να επιτύχει ο πράκτορας σε ένα τέτοιο πρόβλημα είναι να αναπτύξει μια όσο το δυνατόν καλύτερη προσέγγιση της συνάρτησης κέρδους χρησιμοποιώντας τη μέθοδο της δοκιμής και λάθους που αναφέραμε στην αρχή της ενότητας.

Η προσέγγιση της βέλτιστης συνάρτησης κέρδους συνοδεύεται από τη μάθηση μιας βέλτιστης πολιτικής, δηλαδή μιας απεικόνισης από το σύνολο των καταστάσεων στο σύνολο των δράσεων, η οποία μεγιστοποιεί το αθροιστικό βραβείο που λαμβάνει ο πράκτορας από το περιβάλλον. Ο πράκτορας εκμεταλλεύεται τη γνώση του για να βελτιώνει συνεχώς την προσέγγιση της συνάρτησης κόστους. Στην αρχή της διαδικασίας μάθησης, όπου η γνώση για το περιβάλλον είναι μειωμένη, ο πράκτορας πρέπει να προωθεί την εξερεύνηση με σκοπό την απόκτηση νέας γνώσης. Όσο περνάει ο χρόνος και βελτιώνεται η γνώση του πράκτορα για το περιβάλλον και την εργασία που εκτελεί, ο πράκτορας επιλέγει δράσεις που αποσκοπούν στη μεγιστοποίηση του κέρδους.

Εάν εξετάσουμε μια επανάληψη της διαδικασίας μάθησης παρατηρούμε τα επόμενα. Αρχικά ο πράκτορας γνωρίζει την τρέχουσα κατάστασή του έστω s_t. Βάσει της πολιτικής δράσεων π που έχει σχηματίσει από τις προηγούμενες αλληλεπιδράσεις με το περιβάλλον επιλέγει μια δράση a από το χώρο των πιθανών δράσεων. Ουσιαστικά ο πράκτορας εκτιμά ότι η δράση a θα τον οδηγήσει σε μια νέα κατάσταση s_{t+1} με μια πιθανότητα p. Η μετάβαση αυτή συνοδεύεται με μια επιβράβευση έστω $r(s_t, s_{t+1})$. Παρατηρούμε ότι η νέα κατάσταση s_{t+1} εξαρτάται μόνο από την κατάσταση s_t και τη δράση a που επιλέχθηκε.

Η παραπάνω ιδιότητα ονομάζεται ιδιότητα Markov και το πρόβλημα που καλείται να επιλύσει ο πράκτορας μπορεί να μοντελοποιηθεί ως μια διαδικασία αποφάσεων Markov (Markov Decision Process). Η ιδιότητα αυτή είναι ιδιαίτερα σημαντική για προβλήματα μάθησης καθώς μας επιτρέπει να αγνοούμε τις παρελθοντικές καταστάσεις και δράσεις του συστήματος.

Τέλος πρέπει να αναφέρουμε ότι το παραπάνω πλαίσιο μάθησης είναι ιδιαίτερα ευέλικτο και μπορεί να εφαρμοστεί σε πολλά διαφορετικά προβλήματα. Για παράδειγμα οι δράσεις που λαμβάνει ένας πράκτορας μπορούν να εφαρμοστούν είτε ως σήμα ελέγχου σε ένα σύστημα χαμηλού επιπέδου, όπως η τάση στους κινητήρες ενός ρομποτικού βραχίονα είτε ως αποφάσεις σε συστήματα υψηλότερου επιπέδου, όπως η αγορά μετοχών σε μια χρηματιστηριακή αγορά.

3.3 Στόχοι και συνάρτηση επιβράβευσης

Η συνάρτηση επιβράβευσης αποτελεί ένα από τα σημαντικότερα σημεία της μεθοδολογίας της μηχανικής μάθησης. Στην ουσία η εργασία του πράκτορα και ο στόχος αυτού μοντελοποιούνται ως μια συνάρτηση επιβράβευσης. Ο στόχος του πράκτορα είναι να μεγιστοποιήσει το συνολικό άθροισμα των επιβραβεύσεων που λαμβάνει και όχι μόνο το άμεσο κέρδος του.

Τα προβλήματα ενισχυτικής μάθησης χωρίζονται όπως και τα προβλήματα δυναμικού προγραμματισμού σε προβλήματα πεπερασμένου και μη πεπερασμένου ορίζοντα. Ανάλογα με την κατηγορία του προβλήματος, ο πράκτορας καλείται να λύσει ένα διαφορετικό πρόβλημα βελτιστοποίησης. Στα προβλήματα πεπερασμένου ορίζοντα έχουμε άθροιση των επιβραβεύσεων για πεπερασμένο αριθμό βημάτων ενώ στα προβλήματα μη πεπερασμένου ορίζοντα έχουμε την επ' αόριστον άθροιση των επιβραβεύσεων.

Πριν παρουσιάσουμε τις δύο κατηγορίες προβλημάτων θα αναφέρουμε ορισμένους μαθηματικούς συμβολισμούς που θα χρησιμοποιήσουμε στη συνέχεια. Με π θα συμβολίζουμε την πολιτική που ακολουθεί ο πράκτορας, δηλαδή την λογική βάσει της οποίας επιλέγει τις δράσεις του. Επίσης με $p_{ij}(\mu_k(i))$ θα συμβολίζουμε την πιθανότητα μετάβασης από την κατάσταση i στην κατάσταση j εάν τη χρονική στιγμή k επιλέξουμε μια δράση βάσει της πολιτικής μ. Τέλος συμβολίζουμε με $g(i_t,\mu_t,i_{t+1})$ την επιβράβευση για μετάβαση από την κατάσταση i_t στην κατάσταση i_{t+1} εάν τη χρονική στιγμή t επιλέξουμε μια δράση βάσει της πολιτικής μ_t.

3.3.1 Προβλήματα πεπερασμένου ορίζοντα (Finite Horizon problems)

Στα προβλήματα πεπερασμένου ορίζοντα η διαδικασία της μάθησης έχει ένα πεπερασμένο αριθμό βημάτων έστω N. Όπως αναφέρθηκε προηγουμένως ο στόχος του πράκτορα είναι η μεγιστοποίηση του αθροίσματος των επιμέρους επιβραβεύσεων που λαμβάνει σε κάθε μια από τις N μεταβάσεις κατάστασης. Η συνάρτηση κέρδους μοντελοποιείται ως το άθροισμα της επιμέρους επιβράβευσης για κάθε ένα από τα N βήματα, δηλαδή

$$R = \sum_{i=1}^{N-1} r_i \quad (3.4)$$

Πολλά προβλήματα που μοντελοποιούνται ως προβλήματα πεπερασμένου ορίζοντα επιδέχονται αναλυτικές λύσεις με εφαρμογή δυναμικού προγραμματισμού. Παρόλα αυτά σε αρκετές περιπτώσεις με μεγάλο αριθμό καταστάσεων ή βημάτων, η εφαρμογή δυναμικού προγραμματισμού δεν είναι πρακτική λόγω του αυξημένου χρόνου εκτέλεσης του αλγορίθμου, αφού η πολυπλοκότητα του είναι ανάλογη του αριθμού των βημάτων N και του αριθμού των πιθανών καταστάσεων s.

3.3.2 Προβλήματα μη πεπερασμένου ορίζοντα (Infinite Horizon problems)

Πολλά προβλήματα με ιδιαίτερα μεγάλο αριθμό πιθανών καταστάσεων μπορούν να μοντελοποιηθούν ως προβλήματα μη πεπερασμένου ορίζοντα. Τα προβλήματα αυτής της κατηγορίας παρουσιάζουν ιδιαίτερο ενδιαφέρον γιατί η βέλτιστη πολιτική επιλογής δράσεων είναι συνήθως στατική και δεν αλλάζει ανάλογα με την κατάσταση στην οποία βρίσκεται το σύστημα. Τα προβλήματα μη πεπερασμένου χρονικού ορίζοντα μπορούν να χωριστούν σε τρείς επιμέρους κατηγορίες, όπου και στις τρείς το συνολικό κέρδος ύστερα από άπειρο αριθμό βημάτων δίνεται από τη σχέση

$$J^{\pi}(i) = \lim_{N\to\infty} E\left[\sum_{k=0}^{N-1} \gamma^k g(i_k, \mu_k(i_k), i_{k+1}) \mid i_0 = i\right] \quad (3.5)$$

a) *Stochastic shortest path problems*

Στα προβλήματα αυτά ισχύει γ = 1 και εκτός αυτού υποθέτουμε ότι υπάρχει μια επιθυμητή κατάσταση τερματισμού στην οποία το σύστημα παραμένει επ' αόριστον χωρίς νέες αλλαγές. Επίσης υποθέτουμε ότι το σύστημα θα μεταβεί πάντα στην κατάσταση αυτή ανεξάρτητα από τις ενδιάμεσες καταστάσεις και τα ενδιάμεσα βήματα. Επομένως στόχος του συστήματος είναι να μεταβεί στην κατάσταση αυτή με όσον το δυνατόν λιγότερες ενδιάμεσες μεταβάσεις. Ουσιαστικά το πρόβλημα αυτό είναι ένα πρόβλημα πεπερασμένου ορίζοντα όπου ο αριθμός N των συνολικών μεταβάσεων είναι μια τυχαία μεταβλητή, η οποία εξαρτάται από την πολιτική επιλογής κινήσεων.

b) *Discounted problems*

Στα προβλήματα αυτά η παράμετρος γ λαμβάνει τιμές στο διάστημα a ∈ [0,1). Επιπλέον το γεγονός ότι οι δυνατές καταστάσεις του συστήματος και οι δυνατές δράσεις ανήκουν σε γνωστά και πεπερασμένα σύνολα διασφαλίζει ότι κάθε φορά το κόστος μετάβασης από μια κατάσταση σε μια άλλη είναι πάντα μικρότερο από μια σταθερά M. Επομένως το αθροιστικό κέρδος J^{π} είναι το άπειρο άθροισμα των επιβραβεύσεων των επιμέρους βημάτων του αλγορίθμου, το οποίο είναι φραγμένο κατά απόλυτη τιμή από τη φθίνουσα γεωμετρική πρόοδο γ^kM, γ < 1. Στην περίπτωση αυτή ο πράκτορας ενδιαφέρεται για όλο το μελλοντικό άθροισμα των ενισχύσεων βεβαρημένο όμως με τον παράγοντα γ. Ισχύει ότι

$$R = \sum_{i=1}^{N-1} \gamma^i r_i \mid \gamma \in [0,1) \quad (3.6)$$

c) *Average cost per stage problems*

Σε πολλά από τα προβλήματα η τιμή του αθροιστικού κέρδους γίνεται άπειρο καθιστώντας αδύνατη την επίλυση του καθολικού προβλήματος βελτιστοποίησης. Ένας τρόπος για να αντιμετωπίσουμε αυτό το πρόβλημα είναι να λάβουμε υπόψη το μέσο συνολικό κόστος, το οποίο δίνεται από τον τύπο

$$\lim_{N\to\infty} \frac{1}{N} J_N^{\pi}(i) \quad (3.7)$$

Στα προβλήματα αυτής της κατηγορίας ο πράκτορας μάθησης ενδιαφέρεται για δράσεις που μεγιστοποιούν τη μέση ενίσχυση σε κάθε βήμα και δε τον ενδιαφέρει η χρονική κατανομή της ενίσχυσης.

3.4 Συναρτήσεις Τιμών - Χρησιμότητας

Ένα από τα κυριότερα χαρακτηριστικά των μεθόδων ενισχυτικής μάθησης είναι η χρήση προσεγγιστικών συναρτήσεων τιμών για την αξιολόγηση των καταστάσεων του συστήματος. Οι συναρτήσεις αυτές παρέχουν στον πράκτορα μια εκτίμηση για την αξία της τρέχουσας κατάστασής του και την αξία των δράσεων που μπορεί να επιλέξει σε αυτή τη συγκεκριμένη κατάσταση. Οι προσεγγιστικές αυτές συναρτήσεις παρέχουν μια εκτίμηση για μελλοντικό κέρδος που θα λάβει ο πράκτορας ανάλογα με την νέα κατάσταση στην οποία θα βρεθεί. Ο στόχος είναι ο πράκτορας να αποθηκεύει τη γνώση του για το πόσο καλή είναι μια κατάσταση ή πόσο καλό είναι να εκτελέσει μια δράση όταν βρίσκεται σε μια κατάσταση. Όπως είναι προφανές οι συναρτήσεις τιμών εξαρτώνται άμεσα από την πολιτική επιλογής δράσεων που ακολουθεί ο πράκτορας.

Ορίζουμε ως $V^{\pi}(s)$ την τιμή της κατάστασης s όταν ακολουθείται η πολιτική π. Το $V^{\pi}(s)$ υποδηλώνει το προσδοκώμενο κέρδος αν θεωρήσουμε ως αρχική κατάσταση του συστήματος την s και ότι η επιλογή των μελλοντικών δράσεων θα γίνεται βάσει της πολιτικής π. Για προβλήματα ενισχυτικής μάθησης αλλά και γενικότερα για Markov διαδικασίες αποφάσεων η τιμή $V^{\pi}(s)$ μπορεί να οριστεί ως

$$V^{\pi}(s) = E_{\pi}\{R_t | s_t = s\} = E_{\pi}\left\{\sum_{k=0}^{\infty} \gamma^k \, r_{t+k+1} | s_t = s\right\}, \quad (3.8)$$

όπου ως $E_{\pi}\{R\}$ ορίζεται η εκτιμώμενη τιμή του κέρδους δεδομένου ότι ο πράκτορας ακολουθεί την πολιτική π.

Όταν ο πράκτορας δεν ενδιαφέρεται μόνο για τις καταστάσεις που βρέθηκε αλλά και για τις δράσεις που επέλεξε, ορίζουμε κατά αντιστοιχία με τα προηγούμενα, ως $Q^{\pi}(s,a)$ το εκτιμώμενο κέρδος εάν το σύστημα βρίσκεται στην κατάσταση s και ο πράκτορας επιλέξει τη δράση a ακολουθώντας πάντα την πολιτική π. Έχουμε λοιπόν ότι

$$Q^{\pi}(s,a) = E_{\pi}\{R_t | s_t = s, a_t = a\} = E_{\pi}\left\{\sum_{k=0}^{\infty} \gamma^k \, r_{t+k+1} | s_t = s, a_t = a\right\}, \quad (3.9)$$

Όση περισσότερη γνώση συγκεντρώνει ο πράκτορας τόσο πιο ακριβείς είναι οι προσεγγίσεις των συναρτήσεων $V^{\pi}(s)$ και $Q^{\pi}(s,a)$. Ανάλογα με το πλήθος των πιθανών καταστάσεων του συστήματος και το πλήθος των πιθανών δράσεων που μπορεί να επιλέξει ο πράκτορας, ακολουθούνται διαφορετικές μέθοδοι για την προσέγγιση των συναρτήσεων τιμών. Για μικρό αριθμό διαφορετικών ζευγών καταστάσεων-δράσεων ο πράκτορας διατηρεί ένα πίνακα στον οποίο αποθηκεύει τη μέση τιμή των ενισχύσεων που έλαβε στο παρελθόν κάθε ζεύγος. Όσες περισσότερες φορές επισκεφθεί ο πράκτορας ένα ζεύγος κατάστασης-δράσης τόσο πιο ακριβής είναι η εκτίμηση για το μελλοντικό κέρδος. Όταν ο αριθμός των καταστάσεων είναι ιδιαίτερα μεγάλος τότε είναι αδύνατο να διατηρήσει το σύστημα ένα πίνακα όπως τον προηγούμενο. Στην περίπτωση αυτή επιλέγεται κάποια παραμετρική προσεγγιστική μέθοδος, όπου η προσέγγιση επιτυγχάνεται με κατάλληλη ρύθμιση των παραμέτρων.

Όπως αναφέρθηκε προηγουμένως τα προβλήματα ενισχυτικής μάθησης μπορούν να εκφραστούν βάσει ενός μοντέλου Markov. Αποδεικνύεται ότι οι συναρτήσεις τιμών $V^{\pi}(s)$ και $Q^{\pi}(s,a)$ ικανοποιούν την αναδρομική εξίσωση Bellman. Συγκεκριμένα αποδεικνύονται οι επόμενες σχέσεις

$$V^{\pi}(s) = \sum_{a} \pi(s,a) \sum_{a'} P^{a}_{ss'} \left[R^{a}_{ss'} + \gamma V^{\pi}(s')\right] \quad (3.10)$$

$$Q^{\pi}(s,a) = \sum_{s'} P^{a}_{ss'} \left[R^{a}_{ss'} + \gamma \sum_{s'} \pi(s',a') Q^{\pi}(s',a')\right] \quad (3.11)$$

Οι παραπάνω εξισώσεις εκφράζουν τη σχέση ανάμεσα στην τιμή μιας κατάστασης του συστήματος και των μελλοντικών καταστάσεων που την ακολουθούν. Η εξίσωση Bellman προσθέτει τα επιμέρους κέρδη όλων των πιθανών μεταβάσεων από μια κατάσταση s σε μια άλλη s΄ πολλαπλασιασμένα με την αντίστοιχη πιθανότητα μετάβασης που μας δίνει το μοντέλο Markov. Κάθε συνάρτηση τιμών ικανοποιεί τη συνάρτηση Bellman που προκύπτει για την πολιτική π που ακολουθεί ο πράκτορας.

3.5 Βέλτιστες συναρτήσεις τιμών και βέλτιστη πολιτική

Όπως αναφέρθηκε προηγουμένως η επίλυση ενός προβλήματος ενισχυτικής μάθησης έχει ως στόχο την εύρεση μιας πολιτικής επιλογής δράσεων τέτοιας ώστε να μεγιστοποιείται το αθροιστικό κέρδος που λαμβάνει ο πράκτορας. Σε πεπερασμένες διαδικασίες αποφάσεων Markov ορίζεται μια σχέση μερικής διάταξης επί των πολιτικών επιλογής δράσεων. Μια πολιτική π λέγεται ότι έχει μεγαλύτερη ή ίση αξία από μια άλλη πολιτική π' εάν το

αναμενόμενο κέρδος ακολουθώντας την π είναι μεγαλύτερο ή ίσο από αυτό που θα λάβει ο πράκτορας εάν ακολουθήσει την π΄. Ισχύει λοιπόν ότι

$$\pi \geq \pi' \Leftrightarrow V^{\pi}(s) \geq V^{\pi'}(s) \quad (3.12)$$

Υπάρχει πάντα τουλάχιστον μια πολιτική π^* μεγαλύτερης ή ίσης αξίας με όλες τις υπόλοιπες. Η πολιτική αυτή ονομάζεται βέλτιστη καθώς ακολουθώντας την ο πράκτορας λαμβάνει το μέγιστο δυνατό κέρδος. Παρόλο που μπορεί να υπάρχουν πολλές διαφορετικές βέλτιστες πολιτικές όλες έχουν την ίδια συνάρτηση $V^*(s)$ αξιολόγησης καταστάσεων – δράσεων, η οποία ονομάζεται βέλτιστη συνάρτηση τιμών και ισχύει

$$V^*(s) = \max_{\pi} V^{\pi}(s), \forall s \in S \quad (3.13)$$

Η συνάρτηση τιμών $V^*(s)$ ικανοποιεί τη συνάρτηση Bellman για τη βέλτιστη πολιτική π^* και έχουμε ότι ισχύει

$$V^*(s) = \max_{a \in A(s)} E\{r_{t+1} + \gamma V^*(s_{t+1}) | s_t = s, a_t = a\}$$

$$= \max_{a \in A(s)} \left\{ \sum_{s'} P_{ss'}^a [R_{ss'}^a + \gamma V^*(s')] \right\} \quad (3.14)$$

$$Q^*(s,a) = E\left\{ r_{t+1} + \gamma \max_{a' \in A(s_{t+1})} Q^*(s_{t+1}, a') \,|\, s_t = s, a_t = a \right\}$$

$$= \sum_{s'} P_{ss'}^a \left[R_{ss'}^a + \gamma \max_{a' \in A(s_{t+1})} Q^*(s', a') \right] \quad (3.15)$$

Για πεπερασμένες αλυσίδες αποφάσεων Markov οι παραπάνω εξισώσεις έχουν μοναδική λύση ανεξαρτήτως πολιτικής. Εάν το σύστημα έχει N καταστάσεις η εξίσωση βέλτιστου Bellman αποτελεί ένα σύστημα N εξισώσεων με N αγνώστους. Εάν οι πιθανότητες μετάβασης και τα αντίστοιχα κέρδη μεταξύ των διαφορετικών καταστάσεων είναι γνωστά, δηλαδή εάν έχουμε πλήρη γνώση του μοντέλου Markov, τότε η εξίσωση Bellman μπορεί να λυθεί με αναλυτικό τρόπο.

Εάν βρεθεί η βέλτιστη συνάρτηση τιμής τότε είναι αρκετά εύκολο να βρεθεί μια βέλτιστη πολιτική. Εάν το σύστημα βρίσκεται σε μια κατάσταση s θα υπάρχει τουλάχιστον μια δράση, η οποία προκύπτει από την επίλυση του προβλήματος βελτιστοποίησης που εισάγουν οι (3.14) ή (3.15). Το παραπάνω αντιστοιχεί με έρευνα ενός βήματος (one-step search) και χρήση μιας άπληστης πολιτικής βάσει της βέλτιστης συνάρτησης τιμών.

Βέβαια στα περισσότερα προβλήματα είναι άγνωστο το μοντέλο Markov και για την επίλυση της εξίσωσης Bellman χρησιμοποιείται η μέθοδος της ενισχυτικής μάθησης. Στα περισσότερα προβλήματα βέλτιστου ελέγχου και σε όλα τα προβλήματα ενισχυτικής μάθησης είναι άγνωστα τα δυναμικά χαρακτηριστικά του συστήματος. Οι αλγόριθμοι ενισχυτικής μάθησης προσπαθούν να δημιουργήσουν το μοντέλο Markov του συστήματος βάσει των παρατηρήσεων τους για τις καταστάσεις που βρέθηκαν και τις δράσεις που επέλεξαν. Είναι

προφανές ότι ο πράκτορας πρέπει να εξερευνήσει το άγνωστο περιβάλλον για να είναι σε θέση να εκμεταλλευτεί αργότερα τη γνώση αυτή και να ακολουθήσει μια βέλτιστη πολιτική.

3.6 Επιλογή Δράσεων - Εξερεύνηση έναντι Εκμετάλλευσης

Όταν ένα σύστημα μάθησης μαθαίνει να ελέγχει ένα άγνωστο σύστημα σε ένα άγνωστο περιβάλλον πρέπει να λάβει υπόψη δύο αντικρουόμενους στόχους. Από τη μια πρέπει να εξερευνήσει το περιβάλλον ώστε να αποκτήσει τη γνώση (υπό-)βέλτιστων πολιτικών. Από την άλλη πρέπει να εκμεταλλευτεί τη γνώση που έχει αποκτήσει για να επιλέξει τις δράσεις εκείνες που μεγιστοποιούν το κέρδος του. Επομένως είναι αναγκαίο να επιλέγονται δράσεις τέτοιες ώστε να υπάρχει μια ισορροπία ανάμεσα στους δύο παραπάνω στόχους.

Στη συνέχεια θα παρουσιάσουμε τις δημοφιλέστερες στρατηγικές επιλογής δράσεων, οι οποίες εξασφαλίζουν ισορροπία ανάμεσα στην εξερεύνηση και την εκμετάλλευση. Οι στρατηγικές μπορούν να χωριστούν σε δύο ομάδες: *στρατηγικές κατευθυνόμενης εξερεύνησης* και *στρατηγικές μη-κατευθυνόμενης εξερεύνησης*. Οι στρατηγικές της πρώτης κατηγορίας δε λαμβάνουν υπόψη την ήδη υπάρχουσα γνώση του πράκτορα και εμπεριέχουν τυχαιότητα ως προς την επιλογή των δράσεων, ενώ αυτές της δεύτερης κατηγορίας χρησιμοποιούν την ήδη υπάρχουσα γνώση του πράκτορα κατευθύνοντας με αυτό τον τρόπο την επιλογή δράσεων.

Πριν συνεχίσουμε με την παρουσίαση των στρατηγικών πρέπει να ορίσουμε σύντομα τις δύο ακραίες στρατηγικές επιλογής δράσεων. Η στρατηγική, η οποία εκμεταλλεύεται συνεχώς τη γνώση του πράκτορα για το σύστημα, ονομάζεται άπληστη (greedy) στρατηγική. Σύμφωνα με αυτή κάθε φορά ο πράκτορας επιλέγει μια δράση a τέτοια ώστε

$$a = greedy\big(Q_t(s,a)\big) = arg\max_{a' \in A(s)} Q_t(s,a') \quad (3.16)$$

Στο άλλο άκρο συναντάμε την τυχαία στρατηγική κατά την οποία ο πράκτορας επιλέγει τυχαία μια δράση ακολουθώντας μια ομοιόμορφη κατανομή.

3.6.1 Μη - κατευθυνόμενη εξερεύνηση

Οι στρατηγικές μη κατευθυνόμενης εξερεύνησης επιλέγουν τυχαίες δράσεις και δεν χρησιμοποιούν την ήδη υπάρχουσα γνώση του πράκτορα μάθησης. Συνήθως χρησιμοποιούν τις εκτιμήσεις των συναρτήσεων τιμών που διαθέτει ο πράκτορας για να εξισορροπήσουν τους δύο διαφορετικούς στόχους της εξερεύνησης και της εκμετάλλευσης. Στη συνέχεια θα παρουσιάσουμε τις βασικότερες μη κατευθυνόμενες στρατηγικές επιλογής δράσεων.

3.6.1.1 Τυχαία (Ομοιόμορφη) Εξερεύνηση

Ο πιο συνηθισμένος τρόπος εξερεύνησης σε ένα άγνωστο περιβάλλον είναι η επιλογή τυχαίων δράσεων από μια ομοιόμορφη κατανομή. Η συγκεκριμένη στρατηγική επιλογής

δράσεων προτιμάτε όταν δε λαμβάνουμε υπόψη το επιπλέον κόστος της εξερεύνησης κατά τη διαδικασία μάθησης. Ακολουθώντας τη στρατηγική αυτή το σύστημα δεν εκμεταλλεύεται καθόλου τη γνώση που έχει αποκτήσει, πράγμα που την καθιστά ιδιαίτερα χρήσιμη για προβλήματα που δεν είναι διαθέσιμη μια προσέγγιση των συναρτήσεων τιμών. Ισχύει

$$a = random\big(Q_t(s,a)\big) \Leftrightarrow \Pr(a_t = a) = \frac{1}{|A(s)|} \quad (3.17)$$

Για να υπάρχει ισορροπία ανάμεσα στην εκμετάλλευση και την εξερεύνηση, όταν χρησιμοποιείται η τυχαία εξερεύνηση, έχουν προταθεί δύο ευριστικά μοντέλα. Το πρώτο ονομάζεται ε-Greedy και το δεύτερο ε-Decreasing.

a) ε-Greedy

Σύμφωνα με το μοντέλο αυτό η ισορροπία ανάμεσα στην εξερεύνηση και την εκμετάλλευση ελέγχεται από τον παράγοντα ε, όπου ε ∈ [0,1]. Κάθε φορά που ο πράκτορας καλείται να επιλέξει μια δράση, επιλέγει με πιθανότητα 1-ε μια δράση που προκύπτει από την άπληστη στρατηγική και με πιθανότητα ε μια δράση που προκύπτει από την τυχαία στρατηγική. Η συγκεκριμένη προσέγγιση δεν επηρεάζεται από το χρονικό ορίζοντα της διαδικασίας μάθησης αφού επιλέγει πάντα τυχαίες δράσεις με την ίδια πιθανότητα. Το παραπάνω ίσως να μην είναι επιθυμητό μετά από μεγάλο αριθμό επαναλήψεων της διαδικασίας μάθησης, όπου ο πράκτορας διαθέτει ένα αρκετά καλό προσεγγιστικό μοντέλο Markov του συστήματος.

b) ε-Decreasing

Η μέθοδος ε-Decreasing αποτελεί μια παραλλαγή της μεθόδου ε-Greedy. Εδώ η πιθανότητα να επιλέξει ο πράκτορας μια τυχαία δράση μειώνεται όσο αυξάνεται ο αριθμός των επαναλήψεων της διαδικασίας μάθησης. Επομένως η συγκεκριμένη μέθοδος προάγει την εξερεύνηση στα αρχικά στάδια της διαδικασίας μάθησης και εκμεταλλεύεται την αποκτηθείσα γνώση στα μετέπειτα στάδια. Είναι προφανές ότι η πολιτική ε-Decreazing αντιμετωπίζει το πρόβλημα του κόστους των τυχαίων δράσεων όταν έχουμε καλή γνώση του μοντέλου Markov του συστήματος.

3.6.1.2 Επιλογή δράσεων βάσει των συναρτήσεων τιμών

Στην προηγούμενη υποενότητα είδαμε ότι η εξερεύνηση γίνεται επιλέγοντας με πιθανότητα ε τυχαίες δράσεις από μια ομοιόμορφη κατανομή, η οποία είναι ανεξάρτητη των εκτιμήσεων των συναρτήσεων τιμών. Ένας διαφορετικός τρόπος να εξισορροπήσουμε την εξερεύνηση και την εκμετάλλευση είναι να χρησιμοποιήσουμε την εκτιμώμενη αξία κάθε δράσης.

Συγκεκριμένα όσο μεγαλύτερη είναι η αξία της δράσης τόσο μεγαλύτερη πιθανότητα έχει η συγκεκριμένη δράση να επιλεγεί. Πλέον η κατανομή σταματάει να είναι ομοιόμορφη. Το παραπάνω μας διασφαλίζει ότι το σύστημα μπορεί ταυτόχρονα να εξερευνά και να εκμεταλλεύεται την αποκτηθείσα γνώση. Ισχύει ότι

$$a = uniform\big(Q_t(s,a)\big) \Leftrightarrow \Pr(a_t = a) = \frac{Q_t(s,a)}{\sum_{a' \in A(s)} Q_t(s,a')} \quad (3.18)$$

3.6.1.3 Boltzmann distribution

Σύμφωνα με αυτή τη μέθοδο επιλογής δράσεων η πιθανότητα επιλογής μιας δράσης α δίνεται από την επόμενη μη γραμμική συνάρτηση

$$P(a) = \frac{e^{Q_t(s,a)/T}}{\sum_{a' \in A(s)} e^{Q_t(s,a')/T}}, \qquad (3.19)$$

όπου T > 0 είναι ένας παράγοντας κέρδους που συχνά ονομάζεται θερμοκρασία. Η παράμετρος αυτή καθορίζει κατά πόσο ο πράκτορας επιλέγει εντελώς τυχαία τις διαφορετικές δράσεις ή λαμβάνει υπόψη του το εκτιμώμενο κέρδος κάθε δράση κατά την επιλογή. Όταν το $T \to 0$ έχουμε μόνο εκμετάλλευση της αποκτηθείσας γνώσης ενώ όταν το $T \to \infty$ έχουμε μόνο εξερεύνηση και μάλιστα οι δράσεις επιλέγονται από μια ομοιόμορφη κατανομή.

3.6.2 Κατευθυνόμενη εξερεύνηση

Αντίθετα με τις μεθόδους εξερεύνησης που παρουσιάστηκαν μέχρι τώρα, η κατευθυνόμενη εξερεύνηση διατηρεί γνώση για τη διαδικασία της μάθησης και μάλιστα για τις καταστάσεις του συστήματος και την επιλογή των δράσεων κατά αυτή. Στη συνέχεια θα παρουσιάσουμε τρείς μεθόδους κατευθυνόμενης εξερεύνησης που βασίζονται σε α) μετρητές, β) εκτιμήσεις σφαλμάτων και γ) πρόσφατες αναφορές.

3.6.2.1 Μέθοδοι που βασίζονται σε μετρητές

Για κάθε κατάσταση του συστήματος διατηρούμε ένα μετρητή c(s), στον οποίο διατηρούμε των αριθμό των επισκέψεων στη συγκεκριμένη κατάσταση. Οι δράσεις επιλέγονται βάσει ενός γραμμικού συνδυασμού ενός παράγοντα εξερεύνησης και ενός παράγοντα εκμετάλλευσης, όπως φαίνεται στη συνέχεια

$$eval_c(a) = \gamma * Q(s,a) + \frac{c(s)}{E[c|s,a]}, \quad (3.20)$$

όπου ο παράγοντας $E[c|s,a] = \sum_{s'} P_{ss'}(a) * c(s')$ εκφράζει μια εκτίμηση κατά πόσο το σύστημα θα βρεθεί στην κατάσταση s΄ αν ακολουθήσει τη δράση a στην κατάσταση s. Το $\gamma \geq 0$ είναι ένας παράγοντας που καθορίζει το βάρος της εκμετάλλευσης.

Γενικότερα η συγκεκριμένη μέθοδος αποφασίζει να επισκεφθεί καταστάσεις βασιζόμενη στον αριθμό των επισκέψεων αυτών στο παρελθόν. Στη συνέχεια θα παρουσιάσουμε δύο επεκτάσεις της παρούσας μεθόδου, οι οποίες διασφαλίζουν ταχύτερη μάθηση.

3.6.2.2 Μέθοδοι μετρητών με συντελεστή απόσβεσης

Οι μέθοδοι με απλούς μετρητές καταστάσεων δε λαμβάνουν υπόψη τη χρονική στιγμή που εμφανίστηκε κάποια κατάσταση κατά τη διαδικασία μάθησης. Θεωρούμε δύο καταστάσεις s και s΄ με ίσους μετρητές, όπου η s προέκυψε στην αρχή της διαδικασίας μάθησης ενώ η s΄ αργότερα. Στην περίπτωση αυτή η εξερεύνηση πρέπει να προτιμήσει την κατάσταση s σε σχέση με την s΄. Για το λόγο αυτό σε κάθε επανάληψη της διαδικασίας μάθησης ανανεώνουμε κάθε μετρητή σύμφωνα με την επόμενη σχέση

$$c(s) \leftarrow \lambda * c(s)\ \forall s\ \in S, \quad (3.21)$$

3.6.2.3 Μέθοδοι μετρητών με συντελεστή λάθους

Ένας άλλος τρόπος για τη βελτίωση της διαδικασίας μάθησης είναι να λαμβάνουμε υπόψη της αλλαγές των εκτιμώμενων συναρτήσεων τιμών. Σε κάθε βήμα της μάθησης αποθηκεύουμε τη μεταβολή των συναρτήσεων τιμών. Όσο μεγαλύτερη είναι η μεταβολή στη συνάρτηση τιμών που παρατηρήθηκε ενώ το σύστημα βρισκόταν στην κατάσταση s τόσο μεγαλύτερη είναι η πιθανότητα ότι και η αξία των γειτονικών καταστάσεων της s έχει μεταβληθεί. Είναι προφανές ότι η εξερεύνηση πρέπει να προάγει την επίσκεψη καταστάσεων, των οποίων η αξία άλλαξε πρόσφατα. Επεκτείνοντας τον τύπο (3.20) έχουμε

$$eval_{c/error}(a) = \gamma * \hat{Q}(s,a) + \frac{c(s)}{E[c|s,a]} + \beta * E\left[\Delta\hat{Q}|s,a\right], \quad (3.22)$$

όπου β > 0 είναι μια σταθερά που καθορίζει κατά πόσο λαμβάνεται υπόψη η διαφορά των συναρτήσεων τιμών.

3.6.2.4 Εξερεύνηση βάσει χρονικής αναφοράς

Σύμφωνα με τη μέθοδο εξερεύνησης βάσει χρονικής αναφοράς ο πράκτορας προτιμά να λάβει δράσεις, οι οποίες θα μεταφέρουν το σύστημα σε γειτονικές με την τρέχουσα καταστάσεις και επισκέφτηκαν πιο πρόσφατα. Συγκεκριμένα σε κάθε πιθανή κατάσταση s του συστήματος αντιστοιχεί μια τιμή χρονικής αναφοράς *p(s)*, η οποία ισούται με το πλήθος των δράσεων που επιλέχτηκαν από την τελευταία φορά που το σύστημα βρισκόταν στην κατάσταση s. Πλέον οι πιθανές δράσεις αξιολογούνται βάσει της σχέσης

$$eval_r(a) = \gamma * Q(s,a) + \sqrt{E[p|s,a]} \quad (3.23)$$

Ο πράκτορας επιλέγει να εκτελέσει τη δράση με τη μεγαλύτερη αξία όπως αυτή προκύπτει από την (3.22).

3.7 Μέθοδοι Επίλυσης με Δυναμικό Προγραμματισμό

Ο όρος δυναμικός προγραμματισμός αναφέρεται σε μια συλλογή αλγορίθμων που μπορούν να χρησιμοποιηθούν για την εύρεση βέλτιστων πολιτικών σε προβλήματα για τα οποία είναι γνωστό το πλήρες μοντέλο Markov. Τέτοιοι αλγόριθμοι δεν χρησιμοποιούνται για την επίλυση των προβλημάτων ενισχυτικής μάθησης όχι μόνο λόγω της έλλειψης ενός μοντέλου του συστήματος αλλά και εξαιτίας της υπολογιστικής πολυπλοκότητας των αλγορίθμων αυτών. Παρόλα αυτά επιλέξαμε να παρουσιάσουμε σύντομα τη λογική των αλγορίθμων αυτών καθώς αποτελεί τη μαθηματική βάση των αλγορίθμων ενισχυτικής μάθησης που θα παρουσιάσουμε στη συνέχεια. Πριν παρουσιάσουμε τις μεθόδους δυναμικού προγραμματισμού πρέπει να αναφέρουμε σύντομα τα παρακάτω.

Θεωρούμε ότι το περιβάλλον μπορεί να περιγραφεί ως μια πεπερασμένη Markov αλυσίδα αποφάσεων. Θεωρούμε τα πεπερασμένα σύνολα των πιθανών καταστάσεων S και των πιθανών δράσεων του πράκτορα A(s), όπου s ∈ S. Εκτός αυτού θεωρούμε ότι τα δυναμικά χαρακτηριστικά του συστήματος, δηλαδή ο πίνακας των πιθανοτήτων μετάβασης, είναι γνωστά. Έχουμε ότι $Pr^{a}_{ss'} = \Pr\{s_{t+1} = s' | s_t = s, a_t = a\}$. Επίσης το εκτιμώμενο κέρδος για μια μετάβαση από την κατάσταση s σε μια κατάσταση s΄ μέσω μιας δράσης a δίνεται από τον τύπο $R^{a}_{ss'} = E\{r_{t+1} | s_{t+1} = s', s_t = s, a_t = a\}$. Τέλος γνωρίζουμε ότι η βέλτιστη συνάρτηση τιμών – χρησιμότητας για ένα δεδομένο πρόβλημα ικανοποιεί τη συνάρτηση Bellman, δηλαδή

$$V^*(s) = \max_{a \in A(s)} E\{r_{t+1} + \gamma V^*(s_{t+1}) | s_t = s, a_t = a\}$$

$$= \max_{a \in A(s)} \left\{ \sum_{s'} P^{a}_{ss'} [R^{a}_{ss'} + \gamma V^*(s')] \right\} \qquad (3.24)$$

Από την παραπάνω εξίσωση ορίζουμε τον τελεστή Bellman ως

$$TV = \max_{a \in A(s)} \left\{ \sum_{s' \in S} P^{a}_{ss'} [R^{a}_{ss'} + V(s')] \right\} \quad (3.25)$$

Η αναδρομική εξίσωση Bellman μπορεί να γραφτεί πλέον ως $V = TV$. Όπως αποδεικνύεται στο [10] η βέλτιστη συνάρτηση τιμής ικανοποιεί την

$$V^* = TV^* \quad (3.26)$$

και μάλιστα η εξίσωση αυτή έχει μοναδική λύση. Εκτός αυτού αποδεικνύεται και το επόμενο λήμμα.

Λήμμα. (Monotonicity Lemma) Για κάθε n-διάστατο διάνυσμα V και V΄ τέτοιο ώστε V(i) ≤ V΄(i) για i ∈ [1,n] και μια στατική πολιτική π ισχύει:

$$T^t V(i) \leq T^t V'(i)$$
$$T_d^t V(i) \leq T_d^t V'(i) \qquad (3.27)$$

για όλα τα i και t.

3.7.1 Ο αλγόριθμος του δυναμικού προγραμματισμού

Ο αλγόριθμος του δυναμικού προγραμματισμού προκύπτει άμεσα από την παραπάνω αναδρομική σχέση αλλά και το λήμμα μονοτονίας. Αρχίζοντας από μια τυχαία συνάρτηση τιμών *V(s)* και δεδομένης μιας πολιτικής *π*, ο αλγόριθμος δημιουργεί μια ακολουθία νέων πολιτικών π_1, π_2 … και *αξιολογεί* κάθε μια από αυτές. Κατά την αξιολόγηση υπολογίζει για κάθε νέα πολιτική *π΄* το προσδοκώμενο συνολικό κέρδος $V^{\pi'}(s)$, για κάθε s ∈ S. Από την (3.26) και (3.27) βρίσκουμε τη νέα καλύτερη πολιτική βάσει του τύπου

$$T_{\pi'} V^{\pi} = TV^{\pi} \quad (3.28)$$

Η διαδικασία αυτή επαναλαμβάνεται με τη νέα πολιτική *π΄* στη θέση της π εκτός και αν ισχύει $V^{\pi'}(s) = V^{\pi}(s)$ για κάθε s ∈ S, οπότε και τερματίζει επιστρέφοντας την πολιτική *π*.

Αποδεικνύεται ότι ο παραπάνω αλγόριθμος της αξιολόγησης πολιτικών τερματίζει επιστρέφοντας τη βέλτιστη πολιτική επιλογής δράσεων. Έχοντας παρουσιάσει τα βασικά της μεθοδολογίας του δυναμικού προγραμματισμού προχωρούμε στην παρουσίαση των μεθόδων επίλυσης προβλημάτων, στα οποία δεν είναι διαθέσιμο το μοντέλο του συστήματος.

Αλγόριθμος 3.1 Αξιολόγηση πολιτικής με Δυναμικό Προγραμματισμό

Θεωρούμε μια πολιτική *π*

$V(s) \leftarrow 0$, *για κάθε* $s \in S$

Επανέλαβε

$\Delta \leftarrow 0$

Για κάθε $s \in S$:

$u \leftarrow V(s)$

$V(s) \leftarrow \sum_a \pi(s,a) \sum_{s'} P_{ss'}^a \left[R_{ss'}^a + \gamma V(s')\right]$

$\Delta \leftarrow \max(\Delta, |u - V(s)|)$

μέχρι $\Delta < \theta$ (ένα μικρό θετικό αριθμό)

Επέστρεψε $V \approx V^*$

Αλγόριθμος 3.2 Επιλογή νέας πολιτικής με Δυναμικό Προγραμματισμό

Θεωρούμε μια πολιτική π

Αρχικοποιούμε τα V(s) με τυχαίες τιμές για κάθε $s \in S$

Επανέλαβε

$\Delta \leftarrow 0$

Για κάθε $s \in S$*:*

$u \leftarrow V(s)$

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

$\Delta \leftarrow \max(\Delta, |u - V(s)|)$

μέχρι $\Delta < \theta$ (ένα μικρό θετικό αριθμό)

policy-stable ← true

Για κάθε $s \in S$*:*

b ← π(s)

$\pi(s) \leftarrow arg\max_a \sum_{s'} P^a_{ss'} \left[R^a_{ss'} + \gamma V(s')\right]$

εάν b≠π(s) τότε policy-stable ← false

εάν policy-stable, τότε έξοδος αλλιώς πήγαινε στο βήμα 2

3.8 Επίλυση με αναπαράσταση πινάκων (Lookup Table)

Όπως αναφέραμε προηγουμένως η μέθοδος του δυναμικού προγραμματισμού μπορεί να εφαρμοστεί μόνο σε προβλήματα που είναι γνωστό το πλήρες μοντέλο Markov. Στα προβλήματα ενισχυτικής μάθησης το μοντέλο του συστήματος είναι άγνωστο. Για το λόγο αυτό χρησιμοποιούνται διαφορετικές μέθοδοι για την επίλυση των προβλημάτων αυτών.

Στην παρούσα ενότητα θα παρουσιάσουμε τις μεθόδους που χρησιμοποιούν ένα πίνακα για την αναπαράσταση της συνάρτησης τιμών – χρησιμότητας του συστήματος. Αυτό σημαίνει ότι διατηρούμε στη μνήμη μια ξεχωριστή μεταβλητή για κάθε κατάσταση του συστήματος. Είναι προφανές ότι οι μέθοδοι που χρησιμοποιούν πίνακες δεν μπορούν να χρησιμοποιηθούν για την επίλυση προβλημάτων με μεγάλο πλήθος πιθανών καταστάσεων. Παρόλα αυτά οι αλγόριθμοι αυτοί είναι ιδιαίτερα χρήσιμοι γιατί:

1) Εφαρμόζονται σχετικά εύκολα σε προβλήματα με μικρό και μεσαίο πλήθος δυνατών καταστάσεων, για τα οποία δε γνωρίζουμε το πλήρες μοντέλο του συστήματος.
2) Αποτελούν τη βάσει γενικευμένων αλγορίθμων που μπορούν να εφαρμοστούν σε προβλήματα με πολύ μεγάλο πλήθος δυνατών καταστάσεων.

3.8.1 Μάθηση με χρήση προσομοίωση Monte Carlo

Η βασική ιδέα στην προσομοίωση Monte Carlo είναι ο υπολογισμός του μέσου όρου των τιμών που λαμβάνει μια τυχαία μεταβλητή. Χρησιμοποιώντας αυτή την ιδέα ως βάση, η μέθοδος Monte Carlo προσπαθεί να εκτιμήσει τις τιμές των συναρτήσεων V(s) και Q(s,a) δεδομένης μιας πολιτικής π κρατώντας τους μέσους όρους των ενισχύσεων που λαμβάνει ο πράκτορας σε διαφορετικά επεισόδια της διαδικασίας μάθησης. Οι μέθοδοι Monte Carlo εφαρμόζονται κυρίως σε επεισοδιακά προβλήματα μάθησης.

Θεωρούμε ότι έχουμε ένα επεισοδιακό πρόβλημα στο οποίο η κατάσταση 0 είναι τερματική. Μπορούμε αν αρχίσουμε από μια κατάσταση s να παράγουμε ένα μεγάλο πλήθος πιθανών αλληλουχιών καταστάσεων – δράσεων που τελικά οδηγεί το σύστημα στην κατάσταση 0. Για κάθε κατάσταση που επισκέπτεται ο αλγόριθμος διατηρεί τον μέσο όρο των ενισχύσεων που έλαβε. Όταν οι μέσοι όροι για όλες τις πιθανές καταστάσεις του συστήματος αρχίσουν να συγκλίνουν τότε ο πράκτορας έχει βρει τη βέλτιστη συνάρτηση τιμών του συστήματος. Πλέον είναι εύκολο να επιλέξουμε τη βέλτιστη πολιτική λύνοντας το πρόβλημα βελτιστοποίησης $\alpha = arg\max_{\alpha \in A(s)} V^*(s)$.

3.8.2 Μέθοδοι χρονικών διαφορών (Temporal Difference Methods)

Η μέθοδοι χρονικών διαφορών (TD) αποτελούν μια στοχαστική προσεγγιστική μέθοδο για την επίλυση κατάλληλα ανασχηματισμένων μορφών της εξίσωσης Bellman. Η προσομοίωση με Monte Carlo που παρουσιάστηκε προηγουμένως μπορεί να θεωρηθεί σαν μια στοχαστική προσεγγιστική μέθοδο Robbins-Monro για την επίλυση της εξίσωσης

$$V^{\pi}(s_t) = E\left[\sum_{m=0}^{\infty} r_{t+m}\right] \quad (3.29)$$

Επεκτείνοντας τον παραπάνω συλλογισμό μπορούμε να δημιουργήσουμε νέες προσεγγιστικές μεθόδους για την επίλυση της εξίσωσης Bellman $V^{\pi}(s) = E[r + V^{\pi}(s')]$. Μια τέτοια στοχαστική προσεγγιστική μέθοδος δίνεται από την επόμενη εξίσωση

$$V(s_t) = V(s_t) + \alpha\big(r_{t+1} + V(s_{t+1}) - V(s_t)\big) \quad (3.30)$$

Ο παραπάνω κανόνας ανανέωσης εκτελείται κάθε φορά που το σύστημα επισκέπτεται την κατάσταση s_t. Όπως βλέπουμε στις παραπάνω εξισώσεις η τιμή $V^{\pi}(s)$ στην (3.29) εξαρτάται από το άθροισμα των επιμέρους σημάτων ενίσχυσης για όλα τα βήματα της διαδικασίας μάθησης, ενώ στην (3.30) εξαρτάται μόνο από την προηγούμενη τιμή $V^{\pi}(s')$ και το τελευταίο σήμα επιβράβευσης. Επομένως οι μέθοδοι χρονικών διαφορών καθιστούν δυνατή την

ανανέωση των εκτιμήσεων των συναρτήσεων τιμών – χρησιμότητας μετά από κάθε επανάληψη "κατάσταση – δράση – κατάσταση" της διαδικασίας μάθησης. Αυτό επιτρέπει στον πράκτορα να μαθαίνει καθώς ενεργεί (online μάθηση).

Πολλές φορές είναι δυνατό τα μονοπάτια καταστάσεων που ακολουθεί η διαδικασία μάθησης να έχουν διαφορετικό μήκος. Καθώς δεν έχουμε ιδιαίτερη γνώση για το μήκος των μονοπατιών που πρέπει να προτιμήσουμε, μπορούμε να εισάγουμε ένα παράγοντα βάρους λ στη διαδικασία μάθησης. Για το λ έχουμε ότι λ ∈ [0,1]. Επίσης ορίζουμε το d_m ως $d_m = r_{m+1} + V(s_{m+1}) - V(s_t)$. Ενσωματώνοντας το βάρος λ στην εξίσωση (3.30) παίρνουμε τη νέα προσεγγιστική μέθοδο

$$V(s_t) = V(s_t) + \alpha \sum_{m=k}^{\infty} \lambda^{m-k} d_m \quad (3.31)$$

Διαισθητικά μπορούμε να πούμε ότι τα πρώτα μονοπάτια τις διαδικασίας μάθησης έχουν πλέον μεγαλύτερη βαρύτητα σε σχέση με τα τελευταία. Η παραπάνω εξίσωση μας δίνει την οικογένεια των TD(λ) αλγορίθμων.

Όσο ο όρος λ τείνει στο 1 τόσο περισσότερα βήματα της διαδικασίας μάθησης λαμβάνουμε υπόψη για την ανανέωση της εκτίμησης ενώ όσο το λ τείνει στο 0 ο πράκτορας λαμβάνει υπόψη μόνο τα πιο πρόσφατα βήματα της μάθησης. Στις δύο ακραίες περιπτώσεις ισχύουν τα επόμενα. Εάν λ = 1 τότε έχουμε επίλυση με τη μέθοδο Monte Carlo αφού χρησιμοποιούμε το μέσο όρο όλων των σημάτων επιβράβευσης. Αντίθετα εάν λ = 0 λαμβάνουμε υπόψη μόνο το αμέσως προηγούμενο σήμα επιβράβευσης καταλήγοντας έτσι στον αλγόριθμο TD(0), όπου χρησιμοποιείται ο επόμενος κανόνας ανανέωσης

$$V(s_t) = V(s_t) + \alpha\big(r_{t+1} + V(s_{t+1}) - V(s_t)\big) \quad (3.32)$$

Στο σημείο αυτό πρέπει να επισημάνουμε ότι οι εξισώσεις που παρουσιάστηκαν παραπάνω αναφέρονται σε shortest path προβλήματα μη πεπερασμένου ορίζοντα. Η επέκταση της μεθόδου επίλυσης με χρονικές διαφορές και σε discounted problems είναι άμεση και δίνεται από τον επόμενο τύπο στον οποίο έχουμε εισάγει τον παράγοντα ελάττωσης γ.

$$V(s_t) = V(s_t) + \alpha(r_{t+1} + \gamma V(s_{t+1}) - V(s_t) \quad (3.33)$$

Ο αλγόριθμος TD(0) για discounted προβλήματα φαίνεται στη συνέχεια. Περισσότερα για την επίδραση της παραμέτρου λ θα δούμε σε επόμενη ενότητα.

Αλγόριθμος 3.3 TD(0) για προσέγγιση της συνάρτησης V^{π}

Θεωρούμε μια πολιτική π

Αρχικοποιούμε τα V(s) με τυχαίες τιμές για κάθε $s \in S$

Αρχικοποιούμε το σύστημα στην κατάσταση s_0.

Επανέλαβε

Για κάθε συνολικό επεισόδιο της μάθησης

Για κάθε βήμα της διαδικασίας μάθησης

α ← επέλεξε μια δράση βάσει της πολιτικής π

Εκτέλεσε τη δράση

Παρατήρησε το σήμα ενίσχυσης r και τη νέα κατάσταση s′

$V(s) \leftarrow V(s) + \alpha(r + \gamma V(s') - V(s))$

s ← s′

μέχρι να βρεθεί το σύστημα σε τερματική κατάσταση

Στην πιο απλή εφαρμογή του αλγορίθμου TD οι ανανεώσεις των εκτιμήσεων της αξίας κάθε κατάστασης του συστήματος εκτελούνται ταυτόχρονα μόλις τελειώσει όλη η διαδικασία μάθησης για τη συγκεκριμένη πολιτική π. Έπειτα αξιολογούμε την πολιτική π και εφαρμόζουμε ξανά τον αλγόριθμο TD. Αποδεικνύεται ότι η διαδικασία αυτή συγκλίνει στην βέλτιστη πολιτική. Η διαδικασία αυτή ονομάζεται on-policy διαδικασία μάθησης καθώς σε κάθε επανάληψή της, αξιολογούμε την πολιτική επιλογής δράσεων του πράκτορα.

Μια διαφορετική υλοποίηση προκύπτει εάν ανανεώνουμε τις εκτιμήσεις της αξίας κάθε κατάστασης του συστήματος έπειτα από κάθε βήμα της διαδικασίας μάθησης λαμβάνοντας υπόψη το άμεσο σήμα ενίσχυσης και την προηγούμενη εκτιμώμενη αξία της κατάστασης. Η υλοποίηση αυτή εξασφαλίζει την απευθείας σύγκλιση της συνάρτησης τιμών στη βέλτιστη συνάρτηση ανεξάρτητα από την πολιτική που ακολουθεί ο πράκτορας (off-policy method).

3.8.2.1 On-policy μέθοδοι χρονικών διαφορών

Όταν ο πράκτορας καλείται να λύσει το πρόβλημα βελτιστοποίησης προσπαθώντας να βρει μια βέλτιστη πολιτική, τότε χρησιμοποιούνται συναρτήσεις τιμών για τα ζεύγη δράσεων – καταστάσεων $Q^{\pi}(s,a)$. Επεκτείνοντας το μηχανισμό ανανέωσης που φαίνεται στην εξίσωση (3.33) έχουμε των επόμενο μηχανισμό τον τιμών $Q^{\pi}(s,a)$ για μια συγκεκριμένη πολιτική π.

$$Q(s_t, a_t) = Q(s_t, a_t) + \alpha(r_{t+1} + Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)) \quad (3.34)$$

Από τον παραπάνω νόμο ανανέωσης προκύπτει ο αλγόριθμος SARSA (State-Action-Reward-State-Action) που φαίνεται στη συνέχεια

Αλγόριθμος 3.3 SARSA algorithm

Αρχικοποιούμε τα Q(s,a) με τυχαίες τιμές για κάθε $s \in S$

Αρχικοποιούμε το σύστημα στην κατάσταση s_0.

Επανέλαβε

Για κάθε συνολικό επεισόδιο της μάθησης

α ← επέλεξε μια δράση βάσει της πολιτικής π όπως αυτή προκύπτει

από τις τιμές Q(s,a) (π.χ. ε-greedy)

Για κάθε βήμα της διαδικασίας μάθησης

Εκτέλεσε τη δράση α

Παρατήρησε το σήμα ενίσχυσης r και τη νέα κατάσταση s′

α′ ← επέλεξε μια δράση βάσει της πολιτικής π όπως αυτή προκύπτει

από τις τιμές Q(s,a) (π.χ. ε-greedy)

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

s ← s′

α ← α′

μέχρι να βρεθεί το σύστημα σε τερματική κατάσταση

Όπως αποδεικνύεται ο αλγόριθμος SARSA συγκλίνει αφού βασίζεται σε ανανέωση των εκτιμήσεων βάσει χρονικών διαφορών.

3.8.2.2 Off-policy μέθοδοι χρονικών διαφορών

Μια εναλλακτική υπολογιστική μέθοδος που μπορεί να χρησιμοποιηθεί όταν είναι άγνωστο το πλήρες μοντέλο Markov του συστήματος, είναι η μέθοδος Q-learning. Εδώ ο πράκτορας ανανεώνει τις τιμές των παραγόντων Q, οι οποίοι συνδέονται άμεσα με την αξία των ζευγών κατάσταση – δράση, κατά τη διάρκεια της διαδικασίας μάθησης. Η μέθοδος Q-learning αποτελεί μια off-policy μέθοδο καθώς οι παράγοντες Q συγκλίνουν στους βέλτιστους παράγοντες Q^*.

Αρχικά θεωρούμε ένα πρόβλημα ελάχιστου μονοπατιού (γ = 1). Ορίζουμε ως βέλτιστο παράγοντα $Q^*(s,a)$ για ένα ζεύγος κατάστασης – δράσης την τιμή

$$Q^*(s,a) = \sum_{s\prime=0}^{n} p_{ss'}(a)(r_{ss'} + V^*(s')) \quad (3.35)$$

Επίσης η εξίσωση Bellman μας δίνει ότι $V^*(s) = \max_{a \in A(s)} Q^*(s,a)$ επομένως η εξίσωση (3.35) παίρνει την επόμενη μορφή.

$$Q^*(s,a) = \sum_{s\prime=0}^{n} p_{ss\prime}(a)(r_{ss\prime} + \max_{a\in A(s\prime)} Q^*(s',a)) \quad (3.36)$$

Η επίλυση του παραπάνω συστήματος εξισώσεων μας δίνει τη βέλτιστη τιμή των παραγόντων Q. Εάν επεκταθεί η επαναληπτική μέθοδος αξιολόγησης πολιτικών, όπως αυτή παρουσιάστηκε στους προηγούμενους αλγορίθμους, τότε για την εξίσωση (3.36) παίρνουμε τον επόμενο νόμο ανανέωσης, ο οποίος αποτελεί και τη βάση του αλγορίθμου Q learning.

$$Q(s,a) = \ Q(s,a) + a\left[r_{ss\prime} + \gamma \max_{a\in A(s\prime)} Q^*(s',a) \ - Q(s,a)\right] (3.37)$$

Όπως αποδεικνύεται αναλυτικά στο [10] η μέθοδος Q –learning συγκλίνει στη συνάρτηση τιμών Q^* και στη βέλτιστη πολιτική. Ακολουθεί ο αλγόριθμος Q learning.

Αλγόριθμος 3.4 Q-learning algorithm

Αρχικοποιούμε τα Q(s,a) με τυχαίες τιμές για κάθε $s \in S$

Αρχικοποιούμε το σύστημα στην κατάσταση s_0.

Επανέλαβε για κάθε συνολικό επεισόδιο της μάθησης

Αρχικοποίησε την κατάσταση s

Για κάθε βήμα της διαδικασίας μάθησης

α ← επέλεξε μια δράση βάσει της πολιτικής π όπως αυτή προκύπτει

από τις τιμές Q(s,a) (π.χ. ε-greedy)

Εκτέλεσε την α παρατήρησε το σήμα ενίσχυσης r και τη νέα κατάσταση s′

α′ ← επέλεξε μια δράση βάσει της πολιτικής π όπως αυτή προκύπτει

από τις τιμές Q(s,a) (π.χ. ε-greedy)

$Q(s,a) \ \leftarrow \ Q(s,a) + \alpha(r + \gamma \max_{a\prime\in A(s\prime)} Q(s',a') - Q(s,a))$

s ← s′

μέχρι να βρεθεί το σύστημα σε τερματική κατάσταση

3.9 Επίλυση με μηχανισμούς προσέγγισης συναρτήσεων

Οι μέθοδοι επίλυσης προβλημάτων ενισχυτικής μάθησης που αναφέραμε μέχρι τώρα χρησιμοποιούν ένα πίνακα, κάθε εγγραφή του οποίου αναπαριστά την εκτιμώμενη αξία ενός ζεύγους κατάστασης – δράσης. Είναι προφανές ότι αυτή η μέθοδος αναπαράστασης μπορεί να χρησιμοποιηθεί μόνο σε προβλήματα όπου το σύστημα έχει σχετικά μικρό αριθμό διακριτών καταστάσεων και διακριτών δράσεων. Σε περίπτωση που ο αριθμός των καταστάσεων - δράσεων είναι μεγάλος ή ο χώρος καταστάσεων - δράσεων είναι συνεχής είναι αδύνατη η αναπαράσταση με πίνακα λόγω του μεγέθους της απαιτούμενης μνήμης. Η μνήμη που απαιτείται δεν είναι το μόνο πρόβλημα. Ο όγκος και ο χρόνος συλλογής των

δεδομένων που απαιτούνται για να έχουμε ακριβείς εκτιμήσεις για το μοντέλο Markov του συστήματος είναι ιδιαίτερα μεγάλος.

Εν γένει όταν το πλήθος των δυνατών καταστάσεων του συστήματος είναι ιδιαίτερα μεγάλο ο πράκτορας μάθησης πρέπει να είναι σε θέση να γενικεύει την αποκτηθείσα μάθηση και να την αξιοποιεί για την επιλογή δράσεων ακόμα και σε άγνωστες καταστάσεις που επισκέπτεται πρώτη φορά. Για τη γενίκευση της γνώσης χρησιμοποιείται η μέθοδος της *προσέγγισης συναρτήσεων* (function approximation). Στη συνέχεια της παρούσας ενότητας θα παρουσιάσουμε διαφορετικές αρχιτεκτονικές προσέγγισης συναρτήσεων και τις αντίστοιχες επεκτάσεις των αλγορίθμων ενισχυτικής μάθησης που παρουσιάστηκαν προηγουμένως.

3.9.1 Αρχιτεκτονικές μηχανισμών προσέγγισης συναρτήσεων

Οι μηχανισμοί προσέγγισης συναρτήσεων μπορούν να χωριστούν σε δύο μεγάλες κατηγορίες τους παραμετρικούς και μη-παραμετρικούς μηχανισμούς. Στην παρούσα αναφορά θα αναφερθούμε μόνο σε παραμετρικούς προσεγγιστικούς μηχανισμούς. Για περισσότερες πληροφορίες σχετικά με τους μη παραμετρικούς μηχανισμούς αλλά και παραδείγματα ο αναγνώστης παραπέμπεται στο [27].

3.9.1.1 Παραμετρικοί μηχανισμοί

Οι παραμετρικοί μηχανισμοί προσέγγισης συναρτήσεων αποτελούν στην ουσία μια αναπαράσταση από το χώρο παραμέτρων στο χώρο των συναρτήσεων που προσπαθούν να προσεγγίσουν. Ο ίδιος ο μηχανισμός καθώς και το πλήθος των παραμέτρων που χρησιμοποιούνται, καθορίζονται εξαρχής και είναι ανεξάρτητοι από τα δεδομένα που συλλέγονται κατά τη διάρκεια της διαδικασίας μάθησης. Οι τιμές των παραμέτρων του μηχανισμού προσέγγισης καθορίζονται χρησιμοποιώντας τα δεδομένα που συλλέγει ο πράκτορας κατά της διάρκεια της διαδικασίας μάθησης και τα σήματα ενίσχυσης που λαμβάνει.

Στο σημείο αυτό θεωρούμε την αντικειμενική συνάρτηση τιμών ενός προβλήματος Q και την προσέγγισή της $\hat{Q} = F(\theta)$ ή $\hat{Q}(s,a) = [F(\theta)](s,a)$ για κάθε ζεύγος κατάστασης – δράσης s,a. Εάν χρησιμοποιήσουμε πίνακα για την αναπαράσταση του προβλήματος το μέγεθος αυτού θα είναι ανάλογο με το |S|*|A|. Αντίθετα εάν χρησιμοποιήσουμε την παραπάνω προσέγγιση τότε το πλήθος των τιμών που πρέπει να διατηρούμε στη μνήμη ισούται με το πλήθος των παραμέτρων θ_i, το οποίο είναι αρκετά μικρότερο από το |S|*|A|. Επιτυγχάνουμε έτσι μια αρκετά συμπαγή αναπαράσταση του χώρου καταστάσεων – δράσεων. Παρόλα αυτά δεν πρέπει να αγνοούμαι το σφάλμα που εισάγει η παραπάνω προσεγγιστική τιμή της συνάρτησης τιμών.

Αναλυτικότερα έχουμε ότι μια συνάρτηση τιμών μπορεί να αναπαρασταθεί ως γραμμικός συνδυασμός n συναρτήσεων βάσεων (basis functions - BFs) $\varphi_1, \dots, \varphi_n : S \times A \to \mathbb{R}$ και ενός n-διάστατου διανύσματος θ.

$$[F(\theta)](s,a) = \sum_{i=1}^{n} \varphi_i(s,a)\theta_i = \boldsymbol{\varphi}^T(s,a)\theta \quad (3.38)$$

Οι συναρτήσεις βάσεις αναφέρονται στη βιβλιογραφία και ως χαρακτηριστικά (features) του χώρου καταστάσεων - δράσεων.

Για την αναπαράσταση των συναρτήσεων βάσεων μπορούν να χρησιμοποιηθούν διαφορετικές μέθοδοι. Μια ιδιαίτερα διαδεδομένη μέθοδος είναι η χρήση κανονικοποιημένων (ελλειπτικών) Γκαουσιανών ακτινωτών συναρτήσεων βάσεων (Gaussian Radial Basis Functions - RBFs). Ο τύπος μιας RBF δίνεται από την επόμενη σχέση

$$\bar{\varphi}_i(x) = \frac{\varphi_i'(x)}{\sum_{j=1}^{N} \varphi_j'(x)}, \quad \varphi_i'(x) = exp\left(-\frac{1}{2}[x - c_i]^T B_i^{-1}[x - c_i]\right) \quad (3.39)$$

Στην εξίσωση οι $\varphi_i'(x)$ αποτελούν τις μη – κανονικοποιημένες RBFs, το διάνυσμα $\boldsymbol{c}$ περιέχει τα κέντρα των $\varphi_i'(x)$ και ο συμμετρικός θετικά ορισμένος πίνακας $B_i \in \mathbb{R}^{DxD}$ περιέχει τα πλάτη των RBFs.

Ένας διαφορετικός τρόπος αναπαράστασης των συναρτήσεων βάσεων προτείνεται στο [10] από τους Bertsekas και Tsitsiklis χρησιμοποιώντας της σύμπτυξη καταστάσεων (state aggregation). Σύμφωνα με τη μέθοδο της σύμπτυξης καταστάσεων ο χώρος καταστάσεων S χωρίζεται σε N ξένα μεταξύ τους υποσύνολα. Έστω X^i το σύνολο i του παραπάνω διαχωρισμού. Για κάθε πιθανή δράση $\alpha \in A(s)$ αντιστοιχούμε την ίδια τιμή Q για όλα τα s που ανήκουν στο υποσύνολο X^i. Στην ουσία οι συναρτήσεις βάσεις φ_i παίρνουν τιμές στο σύνολο {0,1}. Έχουμε ότι:

$$\bar{\varphi}_i(s) = \begin{cases} 1, \varepsilon\acute{\alpha}\nu\ s \in X_i \\ 0, \delta\iota\alpha\varphi o\rho\varepsilon\tau\iota\kappa\acute{\alpha} \end{cases} \quad (3.40)$$

Επειδή τα σύνολα X^i είναι ξένα μεταξύ τους, κάθε φορά μόνο μια συνάρτηση βάση παίρνει την τιμή 1. Ο παραπάνω μηχανισμός μπορεί να χρησιμοποιηθεί και για τη διακριτοποίηση του επαυξημένου χώρου καταστάσεων – δράσεων.

Έχουμε ότι

$$\bar{\varphi}_{i,j}(s,\alpha) = \begin{cases} 1, \varepsilon\acute{\alpha}\nu\ s \in X_i\ \kappa\alpha\iota\ a \in A_j \\ 0, \delta\iota\alpha\varphi o\rho\varepsilon\tau\iota\kappa\acute{\alpha} \end{cases} \quad (3.41)$$

Άλλοι τύποι παραμετρικών μηχανισμών αναφέρονται αναλυτικά στη σχετική βιβλιογραφία και περιλαμβάνουν μεθόδους όπως το tile coding [9], multilinear interpolation και τη μέθοδο Kuhn triangulation.

3.9.2 Αλγόριθμοι ενισχυτικής μάθησης με χρήση παραμετρικών προσεγγιστικών μηχανισμών

Στη συνέχεια της παρούσας εργασίας θα παρουσιάσουμε τις επεκτάσεις των αλγορίθμων ενισχυτικής μάθησης για αναπαράσταση των συναρτήσεων τιμών με παραμετρικούς μηχανισμούς προσεγγίσεων. Όπως και προηγουμένως θεωρούμε ότι το μοντέλο που περιγράφει το περιβάλλον αλλά και την αξία των καταστάσεων – δράσεων είναι άγνωστο. Από τους αλγορίθμους για αναπαράσταση με πίνακες έχουμε ότι ο γενικευμένος κανόνας ανανέωσης των τιμών Q(s,a) δίνεται από τον τύπο

$$Q(s,a) = \ Q(s,a) + a\left[r_{ss'} + \gamma \max_{a \in A(s')} Q^*(s',a) \ - Q(s,a)\right]$$

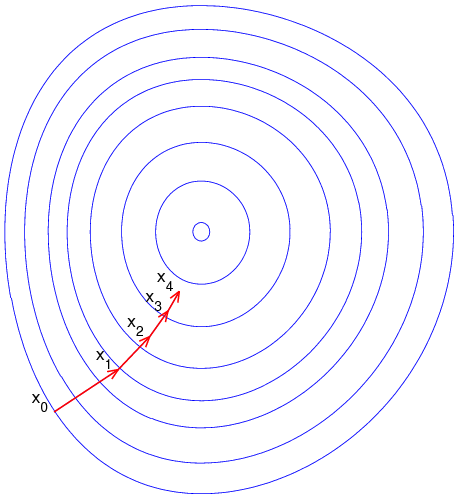
Επίσης θεωρούμε μια γραμμική προσέγγιση της συνάρτησης τιμών

$$\hat{Q}(s,a) = \boldsymbol{\varphi}^T(s,a)\boldsymbol{\theta} \quad (3.42)$$

Πλέον η τιμή Q(s,a) στον παραπάνω κανόνα ανανέωσης περιέχει τις παραμέτρους θ. Ένας άμεσος τρόπος για την ανανέωση των παραμέτρων θ βάσει του παραπάνω κανόνα ανανέωσης είναι η χρήση της μεθόδου καθόδου βαθμίδων (gradient descent). Φυσικά απαιτούμε η έκφραση (3.42) να είναι παραγωγίσιμη. Σύμφωνα με τη μέθοδο gradient descent η νέα τιμή που λαμβάνουν οι παράμετροι θ είναι ανάλογη με την πρώτη παράμετρο μιας αντικειμενικής συνάρτησης χρησιμότητας – ενέργειας την οποία επιθυμούμε να ελαχιστοποιήσουμε, δηλαδή

$$\theta_{k+1} = \theta_k - \gamma \nabla F(\theta_k) \quad (3.43)$$

Ανάλογα με τη συνάρτηση χρησιμότητας που χρησιμοποιούμε προκύπτουν και διαφορετικοί κανόνες ανανέωσης των παραμέτρων θ. Πριν προχωρήσουμε στην παρουσίαση των νέων αλγορίθμων θα παρουσιάσουμε δύο διαφορετικές αντικειμενικές συναρτήσεις καθώς και τους κανόνες ανανέωσης που προκύπτουν από αυτές.



Σχήμα 3.3: Σχηματική αναπαράσταση της μεθόδου gradient descent.

3.9.2.1 Αντικειμενικές συναρτήσεις αξιολόγησης

Αντικειμενική ονομάζεται μια συνάρτηση ενέργειας που εξαρτάται από τις παραμέτρους θ, την οποία επιθυμούμε να ελαχιστοποιήσουμε ανανεώνοντας αυτές. Η πιο συνηθισμένη αντικειμενική συνάρτηση είναι το μέσο τετραγωνικό σφάλμα μεταξύ της προσεγγιστικής τιμής ενός μεγέθους και της πραγματικής του τιμής. Ως πρώτη αντικειμενική συνάρτηση μπορούμε λοιπόν να χρησιμοποιήσουμε το μέσο τετραγωνικό σφάλμα (**M**ean **S**quare **E**rror) της προσεγγιστικής τιμής της συνάρτησης τιμών $\hat{Q}(s,a)$ και της πραγματικής βέλτιστης τιμής Q(s,a) της συνάρτησης. Η αντικειμενική συνάρτηση γίνεται

$$MSE(\theta) = \sum_{s,a \in S \times A} d_{s,a} \left(\hat{Q}(s,a) - Q(s,a)\right)^2 \quad (3.44),$$

όπου $d_{s,a}$ είναι μια παράμετρος βάρους. Όπως αποδεικνύουν οι Tsitsiklis et al. στο [16] ο αλγόριθμος χρονικών διαφορών (Temporal Difference learning) για on-policy μάθηση με χρήση προσεγγιστικών συναρτήσεων συγκλίνει με πιθανότητα 1 στη βέλτιστη συνάρτηση τιμών εάν ενσωματωθεί σε αυτόν η παραπάνω αντικειμενική συνάρτηση. Κάτι τέτοιο δεν ισχύει και για την περίπτωση των off policy αλγορίθμων όπως ο Q learning. Παρόλα αυτά σε πολλές εφαρμογές που γίνεται χρήση του Q learning με προσεγγιστικές συναρτήσεις το σύστημα παρουσιάζει σύγκλιση.

Στο [14] οι Sutton et al. προτείνουν μια νέα αντικειμενική συνάρτηση, η οποία αντιπροσωπεύει το πόσο καλά η προσεγγιστική συνάρτηση V_θ ικανοποιεί την εξίσωση Bellman του συστήματος. Η πραγματική συνάρτηση τιμών V ικανοποιεί ακριβώς την εξίσωση Bellman

$$V = R + \gamma PV \stackrel{\text{def}}{=} TV\,, \quad (3.45)$$

όπου R είναι ένα διάνυσμα που περιέχει τις τιμές $E\{r_{t+1} \mid s_t = s\}$ και T είναι ο τελεστής Bellman. Βάσει των παραπάνω ορίζουν το μέσο τετραγωνικό σφάλμα Bellman (**M**ean – **S**quare **B**ellman **E**rror) ως

$$MSBE(\theta) = \|V_\theta - TV_\theta\|_D^2 \ (3.46)$$

όπου D είναι μια μήτρα βάρους με τιμές d_s, στη διαγώνιό της. Βέβαια ο όρος TV_θ δεν μπορεί να αναπαρασταθεί συναρτήσει του V_θ για κάθε θ. Εισάγεται λοιπόν ένας όρος Π, ο οποίος αποτελεί την προβολή μιας τιμής *v* στην κοντινότερη αναπαράσταση της από την προσεγγιστική συνάρτηση V_θ. Έχουμε

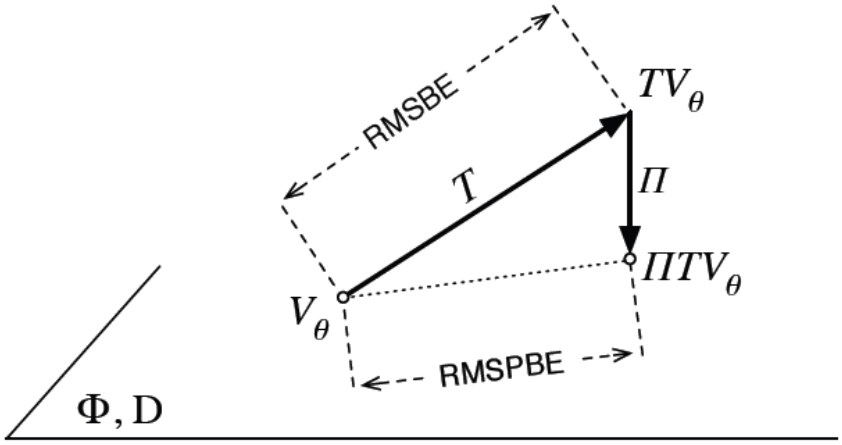
$$\Pi v = V_\theta, \text{όπου } \theta = arg\min_\theta \|V_\theta - v\|_D^2$$

Για μια γραμμική αναπαράσταση του $V_\theta = \Phi\theta$ ο τελεστής προβολής είναι γραμμικός και ανεξάρτητος από το θ και ισχύει

$$\Pi = \Phi(\Phi^T D\Phi)^{-1}\Phi^T D \quad (3.47)$$

Χρησιμοποιώντας τον τελεστή της προβολής που μόλις αναφέρθηκε οι Sutton et al. εισάγουν μια νέα αντικειμενική συνάρτηση, το μέσο τετραγωνικό προβαλλόμενο σφάλμα Bellman

$$MSPBE(\theta) = \|V_\theta - \Pi TV_\theta\|_D^2 \quad (3.48)$$



Σχήμα 3.4: Σχηματική αναπαράσταση των τετραγωνικών ριζών των δύο αντικειμενικών συναρτήσεων σφάλματος – Bellman (Πηγή [27])

Στο σχήμα 3.4 φαίνεται η γεωμετρική σχέση των δύο τετραγωνικών σφαλμάτων Bellman που αναφέρθηκαν παραπάνω.

3.9.2.2 Κανόνες ανανέωσης παραμέτρων θ

Στην παρούσα υποενότητα θα παρουσιάσουμε τους κανόνες ανανέωσης για τις παραμέτρους θ, όπως αυτοί προκύπτουν με χρήση της μεθόδου gradient descent και των αντικειμενικών συναρτήσεων που παρουσιάστηκαν παραπάνω.

Αρχικά θεωρούμε την αντικειμενική συνάρτηση του απλού μέσου τετραγωνικού σφάλματος που φαίνεται στη σχέση (3.43). Εφαρμόζοντας τη μέθοδο gradient descent έχουμε ότι

$$\theta_{k+1} = \theta_k - \frac{1}{2}a_k \frac{\partial}{\partial \theta_k}\left[Q^*(s_k, a_k) - \hat{Q}_k(s_k, a_k)\right]^2$$

$$= \theta_k + a_k\left[Q^*(s_k, a_k) - \hat{Q}_k(s_k, a_k)\right]\frac{\partial}{\partial \theta_k}\hat{Q}_k(s_k, a_k)$$

Φυσικά η πραγματική τιμή της συνάρτησης τιμών Q είναι άγνωστη. Μπορεί όμως να προσεγγιστεί από τον όρο

$$r_{k+1} + \gamma \max_{a'} \hat{Q}_k(s_{k+1}, a')$$

Τελικά ο κανόνας ανανέωσης των παραμέτρων θ γίνεται

$$\theta_{k+1} = \theta_k + a_k\left[r_{k+1} + \gamma \max_{a'} \hat{Q}_k(s_{k+1}, a') - \hat{Q}_k(s_k, a_k)\right]\frac{\partial}{\partial \theta_k}\hat{Q}_k(s_k, a_k) \quad (3.49)$$

Εάν θεωρήσουμε ότι χρησιμοποιούμε μια μέθοδο προσέγγισης όπως αυτή της έκφρασης (3.42) τότε ο παραπάνω κανόνας γίνεται

$$\theta_{k+1} = \theta_k + a_k \left[r_{k+1} + \gamma \max_{a'} (\varphi^T(s_{k+1}, a')\theta_k) - \varphi^T(s_k, a_k)\theta_k \right] \varphi(s_k, a_k) \quad (3.50)$$

Στην περίπτωση που χρησιμοποιήσουμε την αντικειμενική συνάρτηση που προτείνεται από τους Sutton et al. στο [14] έχουμε ένα νέο κανόνα ανανέωσης για τις παραμέτρους θ. Πριν παρουσιάσουμε το νέο κανόνα ανανέωσης των παραμέτρων θ πρέπει να ορίσουμε τις επόμενες εκτιμώμενες τιμές. Ισχύει:

$$\mathrm{E}[\varphi\varphi^T] = \sum_s d_s \varphi_s \varphi_s^T = \Phi^T D \Phi \quad (3.51)$$

$$\mathrm{E}[\delta\varphi] = \sum_s d_s \varphi_s \left(R_s + \gamma \sum_{s'} P_{ss'} V_\theta(s') - V_\theta(s) \right) = \Phi^T D (T V_\theta - V_\theta) \quad (3.52)$$

Επίσης μετά από πράξεις έχουμε ότι

$$\Pi^T D \Pi = D^T \Phi (\Phi^T D \Phi)^{-1} \Phi^T D \quad (3.53)$$

Χρησιμοποιώντας τις παραπάνω σχέσεις έχουμε για την αντικειμενική εξίσωση της σχέσης (3.47)

$$\begin{aligned} MSPBE(\theta) &= \|V_\theta - \Pi T V_\theta\|_D^2 = \|\Pi(V_\theta - \Pi T V_\theta)\|_D^2 \\ &= \big(\Pi(V_\theta - T V_\theta)\big)^T D \big(\Pi(V_\theta - T V_\theta)\big) \\ &= (V_\theta - T V_\theta)^T \Pi^T D \Pi (V_\theta - T V_\theta) \\ &= (V_\theta - T V_\theta)^T D^T \Phi (\Phi^T D \Phi)^{-1} \Phi^T D \ (V_\theta - T V_\theta) \\ &= \big(\Phi^T D (T V_\theta - V_\theta)\big)^T (\Phi^T D \Phi)^{-1} \Phi^T D (T V_\theta - V_\theta) \\ &= \mathrm{E}[\delta\varphi]^T \mathrm{E}[\varphi\varphi^T]^{-1} \mathrm{E}[\delta\varphi] \end{aligned}$$

Εάν χρησιμοποιήσουμε μια γραμμική μέθοδο πρόβλεψης έχουμε ότι

$$w \approx \mathrm{E}[\varphi\varphi^T]^{-1} \mathrm{E}[\delta\varphi] \quad (3.54)$$

Χρησιμοποιώντας την εξίσωση (3.54) η παράγωγος της αντικειμενικής συνάρτησης MSPBE γίνεται

$$-\frac{1}{2} \nabla MSPBE(\theta) = \mathrm{E}[(\varphi - \gamma\varphi')\varphi^T] \mathrm{E}[\varphi\varphi^T]^{-1} \mathrm{E}[\delta\varphi] \approx \mathrm{E}[(\varphi - \gamma\varphi')\varphi^T] w$$

Προκύπτει λοιπόν ο αλγόριθμος GTD2 [14] με τον κανόνα ανανέωσης των παραμέτρων θ σύμφωνα με τη μέθοδο gradient descent να είναι

$$\theta_{k+1} = \theta_k - \frac{1}{2} a_k \nabla MSPBE(\theta) = \theta_k + a_k \big(\varphi_k - \gamma \varphi'_k\big)\big(\varphi_k^T w_k\big), \quad (3.55)$$

όπου οι παράμετροι w_k ανανεώνονται σύμφωνα με τον κανόνα

$$w_{k+1} = w_k + \beta_k \big(\delta_k - \varphi_k^T w_k\big) \varphi_k$$

Η παράγωγος της αντικειμενικής συνάρτησης MSPBE μπορεί να πάρει και μια διαφορετική μορφή εκτός της παραπάνω. Έχουμε ότι

$$
\begin{aligned}
-\frac{1}{2}\nabla MSPBE(\theta) &= \mathrm{E}[(\varphi - \gamma\varphi')\varphi^T]\mathrm{E}[\varphi\varphi^T]^{-1}\mathrm{E}[\delta\varphi] \\
&= (\mathrm{E}[\varphi\varphi^T] - \gamma\mathrm{E}[\varphi'\varphi^T])\mathrm{E}[\varphi\varphi^T]^{-1}\mathrm{E}[\delta\varphi] \\
&= \mathrm{E}[\delta\varphi] - \gamma\mathrm{E}[\varphi'\varphi^T]\mathrm{E}[\varphi\varphi^T]^{-1}\mathrm{E}[\delta\varphi] \\
&\approx \mathrm{E}[\delta\varphi] - \gamma\mathrm{E}[\varphi'\varphi^T]w
\end{aligned}
$$

Ο νέος κανόνας ανανέωσης που προκύπτει και ενσωματώνεται στον αλγόριθμο TDC [14] είναι

$$\theta_{k+1} = \theta_k + a_k\delta_k\varphi_k - \alpha\gamma\varphi'_k\left(\varphi_k^T w_k\right), \quad (3.56)$$

όπου οι παράμετροι w_k ανανεώνονται όπως παραπάνω. Στο [14] οι Sutton et al. αποδεικνύουν τη σύγκλιση των δύο παραπάνω αλγορίθμων. Στη συνέχεια παρουσιάζουμε τους αλγορίθμους μάθησης που προκύπτουν με χρήση των προσεγγιστικών συναρτήσεων τιμών και των παραπάνω αντικειμενικών συναρτήσεων. Η βασική διαφορά σε σχέση με τους αλγορίθμους για αναπαράσταση με πίνακα είναι ότι αντί για την απευθείας ανανέωση των τιμών των συναρτήσεων V ή Q έχουμε ανανέωση των παραμέτρων θ βάσει των κανόνων που παρουσιάστηκαν. Κατά τα άλλα η γενική φιλοσοφία των αλγορίθμων παραμένει η ίδια.

3.9.2.3 Temporal Difference (TD) learning with MSE Objective Function

Αλγόριθμος 3.5 TD(0) – linear function approximator

Θεωρούμε μια πολιτική *π*

Αρχικοποιούμε το διάνυσμα παραμέτρων $\boldsymbol{\theta}$ *με τυχαίες τιμές (e.g.* $\boldsymbol{\theta} \leftarrow 0$)

Αρχικοποιούμε το σύστημα στην κατάσταση s_0.

Επανέλαβε για κάθε βήμα $k = 0, 1, 2, \ldots$ *της διαδικασίας μάθησης*

α ← επέλεξε μια δράση βάσει της πολιτικής π (π.χ. ε-greedy policy)

Εκτέλεσε τη δράση

Παρατήρησε το σήμα ενίσχυσης r και τη νέα κατάσταση s′

$\theta_{k+1} = \theta_k + a_k[r + \gamma\varphi^T(s')\theta_k - \varphi^T(s)\theta_k]\varphi(s)$

s ← s′

3.9.2.4 Q – learning with MSE Objective Function

Αλγόριθμος 3.6 Q learning – linear function approximator

Αρχικοποιούμε το διάνυσμα παραμέτρων $\boldsymbol{\theta}$ *με τυχαίες τιμές (e.g.* $\boldsymbol{\theta} \leftarrow 0$)

Αρχικοποιούμε το σύστημα στην κατάσταση s_0.

Επανέλαβε για κάθε βήμα k = 0, 1, 2, … της διαδικασίας μάθησης

α ← επέλεξε μια δράση βάσει της πολιτικής π όπως αυτή προκύπτει από τις τιμές $\hat{Q}(s,a) = \varphi^T(s,a)\theta$ (π.χ. ε-greedy policy)

Εκτέλεσε τη δράση a

Παρατήρησε το σήμα ενίσχυσης r και τη νέα κατάσταση s΄

$\theta_{k+1} = \theta_k + a_k[r + \gamma \max_{a'}(\varphi^T(s',a')\theta_k) - \varphi^T(s,a)\theta_k]\varphi(s,a)$

s ← s΄

3.9.2.5 SARSA learning with MSE Objective Function

Αλγόριθμος 3.6 SARSA – linear function approximator

Αρχικοποιούμε το διάνυσμα παραμέτρων $\boldsymbol{\theta}$ *με τυχαίες τιμές (e.g.* $\boldsymbol{\theta} \leftarrow 0$)

Αρχικοποιούμε το σύστημα στην κατάσταση s_0.

α ← επέλεξε μια δράση βάσει της πολιτικής π όπως αυτή προκύπτει από τις τιμές $\hat{Q}(s,a) = \varphi^T(s,a)\theta$ (π.χ. ε-greedy policy)

Επανέλαβε για κάθε βήμα k = 0, 1, 2, … της διαδικασίας μάθησης

Εκτέλεσε τη δράση a

Παρατήρησε το σήμα ενίσχυσης r και τη νέα κατάσταση s΄

α' ← επέλεξε μια δράση βάσει της πολιτικής π όπως αυτή προκύπτει από τις τιμές $\hat{Q}(s,a) = \varphi^T(s,a)\theta$ (π.χ. ε-greedy policy)

$\theta_{k+1} = \theta_k + a_k[r + \gamma\varphi^T(s',a')\theta_k - \varphi^T(s,a)\theta_k]\varphi(s,a)$

s ← s΄ , α' ← α

3.9.2.6 TD with gradient correction (TDC) with MSPBE

Αλγόριθμος 3.7 TDC – linear function approximator

Θεωρούμε μια πολιτική π

Αρχικοποιούμε το διάνυσμα παραμέτρων $\boldsymbol{\theta}$ *με τυχαίες τιμές (e.g.* $\boldsymbol{\theta} \leftarrow 0$)

Αρχικοποιούμε το διάνυσμα παραμέτρων $\boldsymbol{w}$ *με τυχαίες τιμές (e.g.* $\boldsymbol{w} \leftarrow 0$)

Αρχικοποιούμε το σύστημα στην κατάσταση s_0.

Επανέλαβε για κάθε βήμα $k = 0, 1, 2, \ldots$ *της διαδικασίας μάθησης*

α ← επέλεξε μια δράση βάσει της πολιτικής π (π.χ. ε-greedy policy)

Εκτέλεσε τη δράση

Παρατήρησε το σήμα ενίσχυσης r και τη νέα κατάσταση s′

$\theta_{k+1} = \theta_k + a_k \delta_k \varphi_k - \alpha\gamma\varphi'_k\left(\varphi_k^T w_k\right)$

$w_{k+1} = w_k + \beta_k\left(\delta_k - \varphi_k^T w_k\right)\varphi_k$

s ← s′

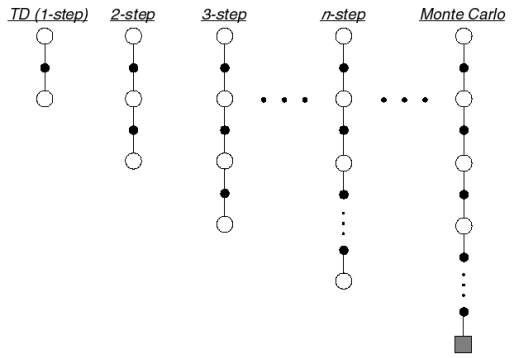
3.10 Eligibility Traces – Παράγοντας λ

Τα ίχνη επιλογής (eligibility traces) είναι ένας από τους ισχυρότερους μηχανισμούς που έχουν εισαχθεί στο πεδίο της ενισχυτικής μάθησης και εκφράζονται από τον παράγοντα λ που παρουσιάστηκε στην υποενότητα 3.8.2. Υπάρχουν δύο ερμηνείες για το ρόλο των eligibility traces στην επίλυση προβλημάτων ενισχυτικής μάθησης. Σύμφωνα με την πρώτη, πιο θεωρητική άποψη, τα ίχνη αυτά αποτελούν μια σύνδεση της μεθόδου TD με τη μέθοδο Monte Carlo. Σύμφωνα με μια πιο πρακτική θεώρηση τα eligibility traces αποτελούν ένα είδος προσωρινής μνήμης για τις καταστάσεις του συστήματος στο άμεσο παρελθόν και τις δράσεις που επιλέχθηκαν σε αυτές. Οι μέθοδοι επίλυσης προβλημάτων ενισχυτικής μάθησης που παρουσιάστηκαν προηγουμένως μπορούν να συνδυαστούν με eligibility traces. Θα παρουσιάσουμε τους νέους αυτούς αλγορίθμους στο τέλος της παρούσας υποενότητας. Πριν από αυτό θα παρουσιάσουμε πιο αναλυτικά τις δύο διαφορετικές ερμηνείες των eligibility traces.

3.10.1 Η TD πρόβλεψη n-βημάτων και η θεωρητική ερμηνεία των eligibility traces

Σύμφωνα με την πρώτη ερμηνεία τα eligibility traces αποτελούν μια γέφυρα μεταξύ των μεθόδων χρονικών διαφορών και των μεθόδων Monte Carlo. Θεωρούμε μια συνάρτηση τιμών V^{π}, την οποία προσπαθούμε να προσεγγίσουμε. Όπως είδαμε σε προηγούμενη ενότητα οι μέθοδοι Monte Carlo κρατάνε μια backup τιμή για κάθε κατάσταση του συστήματος που

παρατηρείται σε μια ολοκληρωμένη ακολουθία καταστάσεων, η οποία περιέχει αρχική και τελική κατάσταση. Αντίθετα οι προσεγγιστικές τιμές που διατηρούν οι μέθοδοι χρονικών διαφορών εξαρτώνται μόνο από την τιμή της αμέσως προηγούμενης κατάστασης και του σήματος επιβράβευσης που έλαβε ο πράκτορας (TD(0)). Όπως φαίνεται στο σχήμα 3.3 μια διαφορετική προσέγγιση είναι να διατηρούμε τις τιμές για περισσότερες από μια παλαιότερες καταστάσεις αλλά σε καμία περίπτωση για ολόκληρη την αλυσίδα καταστάσεων. Οι τιμές εξαρτώνται από τα προηγούμενα σήματα ενίσχυσης καθώς και τις παλαιότερες εκτιμήσεις.



Σχήμα 3.5 : Το φάσμα όλων των μεθόδων μάθησης από την TD(0) με one-step backup μέχρι τη μέθοδο Monte Carlo με full backup.

Στο σημείο αυτό πρέπει να ορίσουμε ξανά την αντικειμενική συνάρτηση που προσπαθεί να μεγιστοποιήσει κάθε ένας από τους παραπάνω μηχανισμούς μάθησης. Ονομάζουμε στόχο n βημάτων την έκφραση

$$R_t^{(n)} = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \cdots + \gamma^{n-1} r_{t+n} + \gamma^n V(s_{t+n}),\ \ n \in \{1, \dots, T-t\} \quad (3.57)$$

Ερμηνεύουμε τις μεθόδους αξιολόγησης πολιτικής βάσει της εξίσωσης (3.57). Η μέθοδος Monte Carlo χρησιμοποιεί ανανεώσεις που δίνονται από τον τύπο

$$V(s) \leftarrow V(s) + a\big(R - V(s)\big) \quad (3.58)$$

έχοντας ως στόχο τη μεγιστοποίηση του $R_t^{(T-t)}$, δηλαδή λαμβάνει υπόψη την αξία και των T βημάτων της διαδικασίας μάθησης. Στο άλλο άκρο του φάσματος που φαίνεται στο σχήμα 3.5 η μέθοδος TD(0) αξιολογεί την εκτίμηση της συνάρτησης τιμών που διατηρεί βάσει του νόμου

$$V(s) \leftarrow V(s) + a\big(r + \gamma V(s') - V(s)\big) \quad (3.59)$$

έχοντας ως στόχο τη μεγιστοποίηση του $R_t^{(1)}$, δηλαδή λαμβάνει υπόψη μόνο την τιμή ενός βήματος. Στον αλγόριθμο TD(λ) η τιμή του λ σχετίζεται με τον αριθμό των βημάτων για τα οποία κρατάει backup ο πράκτορας. Γενικεύοντας τα παραπάνω έχουμε ότι ο αλγόριθμος

TD(λ) χρησιμοποιεί ένα σταθμισμένο μέσο όρο των backup τιμών για τα n-βήματα της διαδικασίας μάθησης, όπου η τιμή κάθε βήματος i πολλαπλασιάζεται με ένα παράγοντα λ^{i-1}, $0 \leq \lambda \leq 1$. Η συνολική επιβράβευση δίνεται από τον τύπο

$$R_t^{\lambda} = (1-\lambda)\sum_{n=1}^{\infty} \lambda^{n-1} R_t^{(n)} = (1-\lambda) \sum_{n=1}^{T-t-1} \lambda^{n-1} R_t^{(n)} + \lambda^{T-t-1} R_t \quad (3.60)$$

Παρατηρώντας την εξίσωση (3.60) είναι πλέον ξεκάθαρη η λειτουργία του παράγοντα λ. Μάλιστα βλέπουμε ότι για λ = 1 οι μέθοδος TD(λ = 1) ταυτίζεται με τη μέθοδο Monte Carlo. Η προσέγγιση που παρουσιάστηκε στην παρούσα ενότητα αποτελεί τη θεωρητική ερμηνεία των eligibility traces. Σε κάθε βήμα της διαδικασίας μάθησης ο πράκτορας λαμβάνει υπόψη τα σήματα ενίσχυσης επόμενων καταστάσεων της αλυσίδας καταστάσεων και αποφασίζει ποια δράση θα εκτελέσει. Είναι προφανές ότι η υλοποίηση του παραπάνω μηχανισμού και η ενσωμάτωση του στους αλγορίθμους μηχανικής μάθησης που παρουσιάστηκαν σε προηγούμενες ενότητες δεν είναι άμεση. Στην επόμενη ενότητα θα παρουσιάσουμε μια πρακτική ερμηνεία των eligibility traces ισοδύναμη με τη θεωρητική.

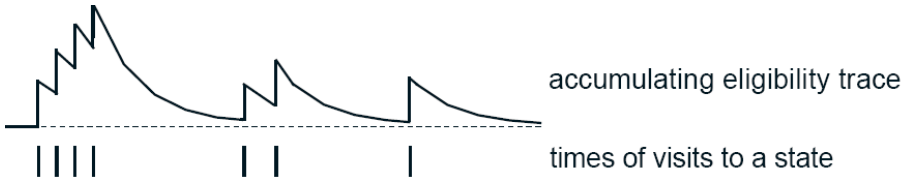
3.10.2 Η μηχανιστική ερμηνεία των eligibility traces

Η μηχανιστική ερμηνεία των eligibility traces που θα παρουσιάσουμε στην παρούσα ενότητα είναι ιδιαίτερα χρήσιμη λόγω της απλότητας αλλά και της ευκολίας υλοποίησής της. Στη νέα αυτή προσέγγιση υπάρχει μια νέα μεταβλητή μνήμης για κάθε κατάσταση του συστήματος. Αναφερόμαστε στις μεταβλητές αυτές ως ίχνη επιλογής (eligibility traces).

Συμβολίζουμε το eligibility trace μιας κατάσταση s τη χρονική στιγμή t ως $e_t(s) \in R$. Σε κάθε βήμα της διαδικασίας μάθησης τα eligibility traces των καταστάσεων του συστήματος μειώνονται σύμφωνα με τον επόμενο κανόνα

$$e_t(s) = \begin{cases} \gamma\lambda e_{t-1}(s), & s \neq s_t \\ \gamma\lambda e_{t-1}(s) + 1, & s = s_t \end{cases}, \lambda \in [0,1], s \in S \quad (3.61)$$

Το eligibility trace μιας κατάστασης μειώνεται με το χρόνο εκτός και αν το σύστημα βρεθεί στην κατάσταση αυτή, οπότε έχουμε αύξηση κατά 1. Επομένως τα eligibility traces αποτελούν ένα δείκτη για τις καταστάσεις στις οποίες έχει βρεθεί το σύστημα στο πρόσφατο παρελθόν. Ο παραπάνω κανόνας ανανέωσης αναφέρεται στα συσσωρευτικά (accumulative) eligibility traces, των οποίων η τιμή φθίνει όπως φαίνεται στο σχήμα 3.6.



Σχήμα 3.6: Ανανέωση τιμών των συσσωρευτικών eligibility traces

Εναλλακτικά μπορούν να χρησιμοποιηθούν eligibility traces αντικατάστασης, όπου

$$e_t(s) = \begin{cases} \gamma\lambda e_{t-1}(s), & s \neq s_t \\ 1, & s = s_t \end{cases}, \lambda \in [0,1], s \in S \quad (3.62)$$

Ενσωματώνοντας τα eligibility traces στη μέθοδο TD παίρνουμε τον επόμενο κανόνα ανανέωσης, ο οποίος αποτελεί τη βάση του αλγορίθμου TD(λ) που θα παρουσιάσουμε στη συνέχεια.

$$V_{t+1}(s_t) = V_t(s_t) + a[r_{t+1} + \gamma V_t(s_{t+1}) - V_t(s_t)]e_t(s_t)\,, \quad (3.63)$$

Όπως αποδεικνύεται στο [9] η μηχανιστική προσέγγιση που μόλις παρουσιάσαμε είναι ισοδύναμη με τη θεωρητική προσέγγιση.

3.10.3 Αλγόριθμοι χρονικών διαφορών με χρήση eligibility traces – Αναπαράσταση με πίνακα

Στην υποενότητα αυτή θα παρουσιάσουμε τις επεκτάσεις των αλγορίθμων TD, SARSA και Q, οι οποίες χρησιμοποιούν eligibility traces. Σε όλους τους αλγορίθμους χρησιμοποιείται ένας πίνακας για τη διατήρηση των εκτιμήσεων της αξίας των καταστάσεων του συστήματος.

3.10.3.1 Αλγόριθμος TD(λ)

Αλγόριθμος 3.5 TD(λ) αλγόριθμος

Θεωρούμε μια πολιτική π
Αρχικοποιούμε τα V(s) με τυχαίες τιμές για κάθε $s \in S$
Αρχικοποιούμε τα $e(s) = 0$ *για κάθε* $s \in S$
Επανέλαβε για κάθε συνολικό επεισόδιο της μάθησης
  Επανέλαβε για κάθε βήμα της διαδικασίας μάθησης
    α ← επέλεξε μια δράση βάσει της πολιτικής π
    Εκτέλεσε τη δράση
    Παρατήρησε το σήμα ενίσχυσης r και τη νέα κατάσταση s′
    $\delta \leftarrow r + \gamma V(s') - V(s)$
    $e(s) \leftarrow e(s) + 1$
    Για κάθε $s \in S$
      $V(s) \leftarrow V(s) + \alpha\delta e(s)$
      $s \leftarrow s'$
  μέχρι να βρεθεί το σύστημα σε τερματική κατάσταση

3.10.3.2 Αλγόριθμος SARSA(λ) και Watkin's Q(λ)

Όπως είδαμε στον παραπάνω αλγόριθμο κρατάμε στον πίνακα την εκτιμώμενη αξία κάθε κατάστασης s. Στις μεθόδους SARSA και Q το σύστημα διατηρεί εκτιμήσεις για την τιμή των ζευγών κατάσταση – δράση. Επομένως πρέπει να αναδιατυπώσουμε τους κανόνες ανανέωσης των eligibility traces ώστε αυτοί να λαμβάνουν υπόψη τα ζεύγη κατάσταση – δράση.

Για συσσωρευτικά eligibility traces ο κανόνας γίνεται

$$e_t(s,a) = \begin{cases} \gamma\lambda e_{t-1}(s) + 1, & s = s_t, a = a_t \\ \gamma\lambda e_{t-1}(s), & \alpha\lambda\lambda\iota\acute{\omega}\varsigma \end{cases}, \lambda \in [0,1], s \in S, a \in A \quad (3.64)$$

και για eligibility traces αντικατάστασης γίνεται

$$e_t(s,a) = \begin{cases} \gamma\lambda e_{t-1}(s,a), s \neq s_t \\ 0, s = s_t \; \kappa\alpha\iota \; a \neq a_t \\ 1, s = s_t \; \kappa\alpha\iota \; a = a_t \end{cases}, \lambda \in [0,1], s \in S, a \in A \quad (3.65)$$

Ακολουθούν οι επεκτάσεις των μεθόδων SARSA και Q.

Αλγόριθμος 3.6 SARSA(λ) algorithm

Αρχικοποιούμε τα Q(s,a) με τυχαίες τιμές για κάθε $s \in S, a \in A$

Αρχικοποιούμε τα $e(s,a) = 0$ *για κάθε* $s \in S, a \in A$

Επανέλαβε

Για κάθε συνολικό επεισόδιο της μάθησης

α ← επέλεξε μια δράση βάσει της πολιτικής π όπως αυτή προκύπτει

από τις τιμές Q(s,a) (π.χ. ε-greedy)

Για κάθε βήμα της διαδικασίας μάθησης

Εκτέλεσε τη δράση α

Παρατήρησε το σήμα ενίσχυσης r και τη νέα κατάσταση s′

α′ ← επέλεξε μια δράση βάσει της πολιτικής π όπως αυτή προκύπτει

από τις τιμές Q(s,a) (π.χ. ε-greedy)

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

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

Για κάθε $s \in S, a \in A$

$Q(s,a) \leftarrow Q(s,a) + \alpha\delta e(s,a)$

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

s ← s′

α ← α′

μέχρι να βρεθεί το σύστημα σε τερματική κατάσταση

Αλγόριθμος 3.4 Watkin's Q(λ)

Αρχικοποιούμε τα Q(s,a) με τυχαίες τιμές για κάθε $s \in S$, $a \in A$

Αρχικοποιούμε τα e(s,a) = 0 για κάθε $s \in S$, $a \in A$

Επανέλαβε

Για κάθε συνολικό επεισόδιο της μάθησης

Αρχικοποίησε την κατάσταση s

Για κάθε βήμα της διαδικασίας μάθησης

α ← επέλεξε μια δράση βάσει της πολιτικής π όπως αυτή προκύπτει από τις τιμές Q(s,a) (π.χ. ε-greedy)

Εκτέλεσε τη δράση α

Παρατήρησε το σήμα ενίσχυσης r και τη νέα κατάσταση s′

α′ ← επέλεξε μια δράση βάσει της πολιτικής π όπως αυτή προκύπτει από τις τιμές Q(s,a) (π.χ. ε-greedy)

$a^* \leftarrow arg\max_{a' \in A(s')} Q(s', a')$

$\delta \leftarrow r + \gamma \max_{a' \in A(s')} Q(s', a') - Q(s, a)$

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

Για κάθε $s \in S$, $a \in A$

$Q(s, a) \leftarrow Q(s, a) + \alpha\delta e(s, a)$

If ($\alpha^* = \alpha'$) then $e(s, a) \leftarrow \gamma\lambda e(s, a)$ else

$e(s, a) \leftarrow 0$

s ← s′

α ← α′

μέχρι να βρεθεί το σύστημα σε τερματική κατάσταση

3.10.4 Αλγόριθμοι χρονικών διαφορών με χρήση eligibility traces – Αναπαράσταση με function approximators

Ο μηχανισμός των eligibility traces μπορεί να επεκταθεί και για τους αλγορίθμους που χρησιμοποιούν αναπαράσταση των συναρτήσεων τιμών με κάποια προσεγγιστική συνάρτηση. Ας θυμηθούμε εδώ ότι στους αλγορίθμους με αναπαράσταση πινάκων διατηρείται ένα eligibility trace για κάθε κελί του πίνακα τιμών V(s) ή Q(s,a). Επεκτείνοντας το πλαίσιο αυτό, έχουμε ότι στους αλγορίθμους με προσεγγιστικές συναρτήσεις διατηρούμε ένα eligibility trace για κάθε παράμετρο θ.

Όπως αναφέρεται στο [30] τα συσσωρευτικά eligibility traces πλέον ανανεώνονται σύμφωνα με τον κανόνα

$$\bar{e}_{k+1} = \gamma\lambda\bar{e}_k + \nabla_{\theta_k}\widehat{\mathrm{Q}}_\theta(s_k, a_k), \qquad (3.66)$$

όπου $\bar{e}_0 = 0$. Αντίστοιχα για τα eligibility traces αντικατάστασης προτείνεται ο επόμενος κανόνας

$$\bar{e}_{k+1} = \max\left(\gamma\lambda\bar{e}_k, \nabla_{\theta_k}\widehat{\mathrm{Q}}_\theta(s_k, a_k)\right), \qquad (3.67)$$

Η ενσωμάτωση των eligibility traces που μόλις παρουσιάστηκαν στους αλγορίθμους της ενότητας 3.9 είναι άμεση και προκύπτει κατά αναλογία με την εφαρμογή των eligibility traces σε αλγορίθμους για αναπαράσταση με πίνακα.

4

Η αντικατάσταση αρχείων ως πρόβλημα Ενισχυτικής Μάθησης

Η διατήρηση κρυφών μνημών (caching) είναι μια ιδιαίτερα αποτελεσματική τεχνική για τη βελτίωση της απόδοσης ενός συστήματος. Οι χρήση τους μάλιστα επεκτάθηκε από τα παραδοσιακά συστήματα επεξεργαστών σε συστήματα μεγαλύτερης κλίμακας, όπως ιεραρχίες αποθήκευσης και web servers. Η σημαντικότερη διαφορά στις δύο παραπάνω εφαρμογές των κρυφών μνημών είναι η επόμενη. Στις παραδοσιακές μνήμες έχουμε διαχείριση μπλοκ δεδομένων, όπου κάθε μπλοκ έχει σταθερό μέγεθος. Αντίθετα στη δεύτερη κατηγορία μνημών γίνεται αποθήκευση ολόκληρων αρχείων διαφορετικών μεγεθών. Το κόστος ανάκτησης ενός αρχείου εάν αυτό δε βρίσκεται στη μνήμη cache εξαρτάται από το μέγεθός του και άλλους παράγοντες, όπως ο φόρτος εργασίας στο δίκτυο. Βλέπουμε ότι πλέον δεν έχουμε ομοιόμορφο μέγεθος αρχείων ούτε και ομοιόμορφο κόστος ανάκτησης των αρχείων. Επομένως η απόφαση ποια αρχεία θα διαγραφούν από τη μνήμη όταν αυτή γεμίσει, αποτελεί ένα ιδιαίτερα ενδιαφέρον πρόβλημα βελτιστοποίησης. Στο κεφάλαιο αυτό θα παρουσιάσουμε πώς η αντικατάσταση αρχείων μπορεί να μοντελοποιηθεί ως ένα πρόβλημα αποφάσεων Markov. Ακολουθώντας την προτεινόμενη μοντελοποίηση θα εξετάσουμε τη δυνατότητα της εφαρμογής ενισχυτικής μάθησης για την επίλυση του προβλήματος.

4.1 Διατύπωση του προβλήματος

Στην ενότητα αυτή διατυπώνουμε το γενικότερο πρόβλημα βελτιστοποίησης που προκύπτει και συζητάμε για τον αν είναι εφικτή ή όχι μια γενικευμένη επίλυση. Προηγουμένως πρέπει να εισάγουμε ορισμένους συμβολισμούς και υποθέσεις.

4.1.1 Συμβολισμοί και υποθέσεις

Θεωρούμε ότι έχουμε ένα πεπερασμένο σύνολο αρχείων U που αναπαρίσταται ως {1, 2 ,… , N}. Σε κάθε αρχείο i ∈ U αντιστοιχεί ένα μέγεθος $a_i \in R$ και ένα κόστος ανάκτησης $c_i \in R$. Κρυφή μνήμη ονομάζουμε ένα σύνολο B ⊂ U τέτοιο ώστε

$$\sum_{i \in B} a_i \leq B, \quad (4.1)$$

όπου B είναι η χωρητικότητα της μνήμης.

Επίσης συμβολίζουμε με ρ : N → U μια ακολουθία αιτήσεων για ανάγνωση αρχείων και με $\sigma_k = i$ την αίτηση ανάγνωσης του i αρχείου τη χρονική στιγμή k. Το μέγιστο κόστος εξυπηρέτησης της ρ είναι

$$W_{max}(\rho) \triangleq \sum_{k=1}^{m} c_{\sigma_\kappa} \quad (4.2)$$

Το σύνολο των αρχείων που βρίσκονται μέσα στην cache αποτελούν την κατάστασή της. Είναι προφανές ότι με την εξυπηρέτηση νέων αιτήσεων η κατάσταση της μνήμης αλλάζει. Εάν B_k είναι η κατάσταση της μνήμης τη χρονική στιγμή k τότε ο χώρος των πιθανών καταστάσεων της μνήμης είναι

$$B = \left\{ S \subset U | \sum_{i \in S} a_i \leq B \right\} \quad (4.3)$$

Το σύνολο B_j συμβολίζει τη συλλογή όλων των καταστάσεων που περιέχουν το αντικείμενο j. Από τα παραπάνω μπορεί να οριστεί μια ακολουθία καταστάσεων B_0, B_1, …, B_m, όπου B_0 είναι η αρχική κατάσταση της μνήμης και B_m η τελική.

4.1.2 Το γενικευμένο πρόβλημα caching

Το γενικευμένο πρόβλημα διατυπώνεται ως εξής. Δοσμένου ενός συνόλου αντικειμένων U με μεγέθη $a_i \in R$ και ένα κόστος ανάκτησης $c_i \in R$ όπου $i \in R$, μιας κρυφής μνήμης μεγέθους B με αρχική κατάσταση B_0 και μιας ακολουθίας αιτήσεων $\rho = \{\sigma_1, \sigma_2, \ldots, \sigma_m\}$ βρες μια ακολουθία καταστάσεων $\{B_i\}_{i=1}^{m}$ τέτοια ώστε:

i. Για όλα τα i =1, 2, … , m ισχύει

$$\sum_{j \in B} a_j \leq B \ ,$$

ii. Για όλα τα i =1, 2, … , m έχουμε

$$B_i = \begin{cases} (B_{i-1} - \mathcal{E}_i) \cup \{\sigma_i\} & \varepsilon\acute{\alpha}\nu\ \sigma_i \notin B_{i-1} \\ B_{i-1} & \varepsilon\acute{\alpha}\nu\ \sigma_i \in B_{i-1} \end{cases},$$

όπου $\mathcal{E}_i \subset B_{i-1}$.

iii. και το κόστος $W(\{B_i\}, \rho, B) = \sum_{i=1}^{m} \delta_i \, c_{\sigma_i}$ με

$$\delta_i \triangleq \begin{cases} 0 & \varepsilon\acute{\alpha}\nu\ \sigma_i \notin B_{i-1} \\ 1 & \varepsilon\acute{\alpha}\nu\ \sigma_i \in B_{i-1} \end{cases}$$

ελαχιστοποιείται

4.2 Θεωρητική επίλυση

Θεωρούμε ότι έχουμε μια ακολουθία από εισερχόμενες αιτήσεις για προσπέλαση αρχείων. Επιπλέον το σύστημα εξυπηρετεί τις αιτήσεις σειριακά. Στη χειρότερη περίπτωση, όπου κανένα από τα αρχεία δε βρίσκεται στο υψηλότερο επίπεδο της ιεραρχίας μνήμης, το κόστος ανάκτησης των αρχείων είναι $W_{max}(\rho)$. Θεωρούμε ότι οι αιτήσεις των χρηστών ακολουθούν ένα μοντέλο Markov με μήτρα πιθανοτήτων $[p_{ij}]$.

Βάσει των παραπάνω το πρόβλημα αντικατάστασης αρχείων μοντελοποιείται ως μια διαδικασία Markov διακριτού χρόνου. Συγκεκριμένα θεωρούμε ότι ο χώρος καταστάσεων της μνήμης cache είναι S = B x U. Κάθε χρονική στιγμή η γενικευμένη κατάσταση του συστήματος ταυτίζεται με ένα διατεταγμένο ζεύγος της κατάστασης της μνήμης cache του συστήματος και της εισερχόμενης αίτησης προσπέλασης ενός αρχείου i.

Σε κάθε δυνατή κατάσταση του συστήματος αντιστοιχεί ένα σύνολο πιθανών δράσεων A(s). Μια δράση $\alpha \in A(s)$ περιγράφεται από ένα σύνολο αντικειμένων, τα οποία θα διαγραφούν από τη μνήμη cache. Συμβολίζουμε με c(s,a) το κόστος διαγραφής των αρχείων που περιέχονται στην α. Σύμφωνα με το μοντέλο Markov έχουμε ότι όταν το σύστημα βρίσκεται

σε μια κατάσταση s = (S,i) και επιλεχθεί μια δράση α τότε ανεξαρτήτως της ιστορίας καταστάσεων – δράσεων του συστήματος :

i) Το σύστημα λαμβάνει άμεσα ένα κόστος c(s,a).

ii) Το σύστημα μεταβαίνει σε μια νέα κατάσταση s΄ με πιθανότητα $p_{ss'}$ για την οποία ισχύει $S' = \big(S - A(s)\big) \cup \{i\}$.

Σύμφωνα με τον αλγόριθμο του δυναμικού προγραμματισμού έχουμε για την αντικειμενική συνάρτηση, ότι

$$J_k(s) = \min_{a \in A(s)} \left\{ c(s,a) + \sum_{s' \in S} p_{ss'} J_{k-1}(s') \right\} \quad (4.4)$$

όπου $J_0(s) = 0$. Η παραπάνω συνάρτηση μας δίνει το ελάχιστο εκτιμώμενο κόστος που αποφέρουν οι αποφάσεις διαγραφής αρχείων. Με χρήση δυναμικού προγραμματισμού μπορούμε να επιλύσουμε την παραπάνω εξίσωση βρίσκοντας έτσι τη βέλτιστη πολιτική αντικατάστασης των αρχείων, η οποία ελαχιστοποιεί το εκτιμώμενο συνολικό κόστος προσπέλασης των αρχείων για ένα πεπερασμένο ορίζοντα k βημάτων.

Είναι προφανές ότι ο χώρος καταστάσεων και δράσεων αυξάνει εκθετικά σε σχέση με το πλήθος των αρχείων που βρίσκονται στο σύστημα. Για παράδειγμα εάν έχουμε n αρχεία αποθηκευμένα στη μνήμη το πλήθος των πιθανών δράσεων είναι 2^n. Όπως αποδεικνύεται στο [17] το γενικευμένο πρόβλημα αντικατάστασης αρχείων είναι NP-complete. Η απόδειξη γίνεται με αναγωγή του προβλήματος στο πρόβλημα 0-1 Knapsack. Λόγω της "κατάρας των διαστάσεων" (course of dimensionality) του προβλήματος προτιμάτε η χρήση ευριστικών στατικών πολιτικών αντικατάστασης αρχείων, όπως η LRU ή LFU, σε συστήματα αποθήκευσης μεγάλης κλίμακας.

4.3 Μοντελοποίηση του συστήματος αποθήκευσης αρχείων CASTOR

Σε προηγούμενη ενότητα παρουσιάσαμε την εσωτερική δομή του υψηλότερου επίπεδου ιεραρχίας μνήμης του συστήματος CASTOR. Όπως αναφέρθηκε η διαγραφή αρχείων εκτελείται ξεχωριστά σε κάθε server, ο οποίος ανήκει σε κάποια κλάση υπηρεσιών (service class). Μάλιστα είναι δυνατό σε κάθε κλάση υπηρεσιών να αντιστοιχεί και διαφορετική πολιτική αντικατάστασης αρχείων. Τα μοτίβα προσπέλασης των αρχείων διαφέρουν σε κάθε κλάση με αποτέλεσμα να είναι άλλη η βέλτιστη πολιτική σε κάθε μια από αυτές.

Καθώς σε ένα ιεραρχικό σύστημα αποθήκευσης αρχείων όπως το σύστημα CASTOR είναι δυνατή η προσθήκη ή αφαίρεση νέων κλάσεων υπηρεσιών και επιπλέον κάθε κλάση διαθέτει ξεχωριστά χαρακτηριστικά από τις άλλες, είναι αναγκαία η χρήση μιας συνάρτησης τιμών – χρησιμότητας Q(s,a) ανά κλάση. Η χρήση μιας μοναδικής καθολικής συνάρτησης τιμών θα

ήταν μη πρακτική, αφού η αντιστοίχηση ενός εκτιμώμενου μελλοντικού κόστους για κάθε διαφορετική κατανομή αρχείων κάθε κλάσης με μια και μόνο συνάρτηση είναι αδύνατη. Λόγω των παραπάνω στην παρούσα εργασία αντιμετωπίζουμε κάθε κλάση υπηρεσιών ως μια ξεχωριστή οντότητα λήψης αποφάσεων. Σε κάθε κλάση αντιστοιχεί ένα πράκτορας μάθησης, ο οποίος καθορίζει την πολιτική αντικατάστασης των αρχείων για την κλάση αυτή σύμφωνα με τα δεδομένα που έχει συλλέξει κατά τη διαδικασία μάθησης.

Κάθε πράκτορας μαθαίνει τη συνάρτηση τιμών V(s) της κλάσης υπηρεσιών στην οποία ενεργεί. Η συνάρτηση τιμών V(s) παρέχει στο σύστημα μια εκτίμηση για το μελλοντικό μέσο κόστος της προσπέλασης των αρχείων. Η μοντελοποίηση της κατάστασης του συστήματος είναι ιδιαίτερα σημαντική για τη διαδικασία της μάθησης. Όπως είδαμε στην υποενότητα 4.1.2 η κατάσταση του συστήματος μια χρονική στιγμή t περιγράφεται πλήρως από την ένωση του συνόλου των αρχείων που βρίσκονται αποθηκευμένα στο υψηλότερο επίπεδο της ιεραρχίας μνήμης και του συνόλου των αρχείων που ζητά να προσπελάσει ο χρήστης την ίδια χρονική στιγμή. Καθώς όμως η πολυπλοκότητα της διαδικασίας μάθησης είναι εκθετική ως προς το πλήθος των μεταβλητών που χρησιμοποιούνται για την περιγραφή του συστήματος, είναι απαραίτητη η χρήση περιορισμένων μεταβλητών, οι οποίες αναπαριστούν το σύστημα με ακρίβεια. Στη συνέχεια θα παρουσιάσουμε τις μεταβλητές που εισαγάγαμε για την περιγραφή του συστήματος και των παραπάνω συνόλων.

4.3.1 Μεταβλητές του συστήματος

Κάθε πράκτορας μάθησης διατηρεί ορισμένες μεταβλητές που περιγράφουν την τρέχουσα κατάσταση της κλάσης υπηρεσιών στην οποία επενεργεί. Οι μεταβλητές που χρησιμοποιούμε στην παρούσα εργασία βασίζονται στην έννοια της θερμοκρασίας κάθε αρχείου του συστήματος. Στην ουσία η θερμοκρασία του αρχείου αποτελεί μια εκτίμηση για τη συχνότητα των προσπελάσεων του αρχείου. Για τον υπολογισμό της θερμοκρασίας το σύστημα κρατά ένα ιστορικό των προσπελάσεων κάθε αρχείου.

Αν θεωρήσουμε την προσπέλαση ενός αρχείου k τη χρονική στιγμή t, η εκτίμηση για το χρονικό διάστημα ανάμεσα σε δύο προσπελάσεις του αρχείου ανανεώνεται βάσει του κανόνα

$$\tau_k = \alpha\tau_k + (1-\alpha)(t - L_k), \qquad (4.5)$$

όπου L_k είναι η χρονική στιγμή της τελευταίας προσπέλασης του αρχείου k και $\alpha < 1$ είναι μια σταθερά, η οποία καθορίζει τη βαρύτητα των παλαιότερων εκτιμήσεων. Από τα παραπάνω έχουμε ότι η θερμοκρασία T_k του αρχείου k είναι ίση με $T_k = \frac{1}{\tau_k}$.

Στη συνέχεια θα ορίσουμε τις μεταβλητές, οι οποίες περιγράφουν το σύστημα. Συμβολίζουμε με S_s το διάνυσμα των μεταβλητών που περιγράφει το σύνολο των αρχείων που είναι

αποθηκευμένα στο υψηλότερο επίπεδο της ιεραρχίας και S_r το διάνυσμα των μεταβλητών που περιγράφουν το σύνολο των αρχείων που ζητούν να προσπελάσουν οι χρήστες.

Για το S_s έχουμε τις επόμενες μεταβλητές:

- s_s^1 = μέση θερμοκρασία των αρχείων που είναι αποθηκευμένα στο υψηλότερο επίπεδο της ιεραρχίας μνήμης. Η συγκεκριμένη μεταβλητή παρέχει μια εκτίμηση για το πλήθος των μελλοντικών εισερχόμενων αιτήσεων προσπέλασης αρχείων ανά μια χρονική μονάδα. Επίσης παρέχει στον πράκτορα μια εκτίμηση για το αν οι χρήστες ζητάνε την προσπέλαση των ίδιων ή διαφορετικών αρχείων.
- s_s^2 = μέση σταθμισμένη θερμοκρασία όλων των αρχείων που είναι αποθηκευμένα στο υψηλότερο επίπεδο της μνήμης (μέσος όρος του γινομένου της θερμοκρασίας και του μεγέθους του αρχείου). Η μεταβλητή αυτή αποτελεί μια εκτίμηση για το μέγεθος των δεδομένων που θα ζητήσουν να προσπελάσουν οι χρήστες στο άμεσο μέλλον. Για μια συγκεκριμένη τιμή της μεταβλητής s_s^1 όσο μεγαλύτερη η τιμή της s_s^2 τόσο μεγαλύτερο το μέγεθος των αρχείων που ζητάνε οι χρήστες.

Για το S_r έχουμε τις επόμενες μεταβλητές:

- s_r^1 = μέση θερμοκρασία των αρχείων που περιέχονται στις αιτήσεις των χρηστών για ένα συγκεκριμένο προκαθορισμένο χρονικό παράθυρο.
- s_r^2 = μέσο μέγεθος των αρχείων που περιέχονται στις αιτήσεις των χρηστών για ένα συγκεκριμένο προκαθορισμένο χρονικό παράθυρο.

Οι παραπάνω μεταβλητές περιγράφουν επαρκώς τις δυνατές καταστάσεις του συστήματος. Τελικά έχουμε το επαυξημένο διάνυσμα κατάστασης του συστήματος είναι το $\mathbf{S} = \mathbf{S_s} \times \mathbf{S_r}$.
Καθώς όλες οι παραπάνω μεταβλητές είναι συνεχείς επιλέξαμε να χρησιμοποιήσουμε ένα μηχανισμό προσέγγισης συναρτήσεων για την αναπαράσταση της συνάρτησης τιμών $V(\mathbf{s})$. Ο προσεγγιστικός μηχανισμός που χρησιμοποιείται βασίζεται στη διακριτοποίηση του χώρου καταστάσεων με χρήση ασαφών κανόνων (fuzzy rulebase representation). Σε επόμενη υποενότητα παρουσιάζουμε λεπτομερώς την αρχιτεκτονική του συγκεκριμένου μηχανισμού.

4.3.2 Μεταβλητές των δράσεων του συστήματος

Σύμφωνα με το γενικευμένο πρόβλημα caching η δράση που καλείται να λάβει ο πράκτορας είναι να επιλέξει ένα σύνολο αρχείων η διαγραφή του οποίου ελαχιστοποιεί το εκτιμώμενο αθροιστικό κόστος προσπέλασης των αρχείων. Είναι προφανές ότι το μέγεθος του χώρου καταστάσεων είναι εκθετικό σε σχέση με το πλήθος των αρχείων. Εκτός αυτού οι δράσεις που μπορεί να λάβει ο πράκτορας είναι διαφορετικές σε κάθε επανάληψη της διαδικασίας μάθησης.

Λόγω των παραπάνω είναι απαραίτητη η χρήση γενικευμένων, περιεκτικών μεταβλητών για τη μοντελοποίηση των χαρακτηριστικών κάθε δράσης του πράκτορα. Οι δύο σημαντικότερες μεταβλητές που χαρακτηρίζουν το σύνολο των αρχείων που επέλεξε να διαγράψει ο πράκτορας είναι το μέγεθος και η συχνότητα προσπέλασης των αρχείων που περιέχονται σ το σύνολο αυτό.

Ορίζουμε λοιπόν τις επόμενες μεταβλητές

- s_α^1 = μέση θερμοκρασία των αρχείων προς διαγραφή.
- s_α^2 = μέσο μέγεθος των αρχείων προς διαγραφή.

Ενσωματώνοντας το διάνυσμα δράσης στο διάνυσμα κατάστασης έχουμε ότι ο χώρος καταστάσεων – δράσεων είναι πλέον $\mathbf{S_s}$ x $\mathbf{S_r}$ x $\mathbf{S_\alpha}$. Καθώς οι παραπάνω μεταβλητές είναι συνεχείς χρησιμοποιείται και εδώ ο προσεγγιστικός μηχανισμός των ασαφών συνόλων για τη διακριτοποίηση του χώρου δράσεων.

4.3.3 Αναπαράσταση της συνάρτησης τιμών με ασαφείς κανόνες

Στην παρούσα εργασία χρησιμοποιούμε ένα σύνολο ασαφών κανόνων *f(x)* για την προσέγγιση της συνάρτησης τιμών *V(s)* κάθε κλάσης υπηρεσιών. Για την κατασκευή των συγκεκριμένων κανόνων ο χώρος καταστάσεων *S* διακριτοποιείται σε *N* ασαφή υποσύνολα A_j^i, τα οποία μπορεί να επικαλύπτονται μεταξύ τους. Κάθε ασαφές υποσύνολο περιγράφεται από μια συνάρτηση (membership function) $\mu_{A_j^i}$ τέτοια ώστε

$$\mu_{A_j^i}: X \to [0,1],$$

όπου i = 1, …, N. Μια κατάσταση s_j του συστήματος ανήκει στο υποσύνολο A_j^i με ένα βαθμό βεβαιότητας $\mu_{A_j^i}$.

Γενικότερα ένα σύνολο κανόνων δέχεται ένα διάνυσμα μεταβλητών $x \in \mathbb{R}^n$ ως είσοδο και το αντιστοιχεί σε μια τιμή εξόδου y. Οι κανόνες του συνόλου έχουν τη μορφή

Κανόνας i: IF (x_1 is A^i_1) and (x_2 is A^i_2) and … and (x_n is A^i_n) then (output = p^i),

όπου p_i είναι η τιμή εξόδου του κανόνα i. Η έξοδος του συνόλου των ασαφών κανόνων *f(x)* ισούται με το σταθμισμένο μέσο όρο των τιμών εξόδου p_i των επιμέρους κανόνων. Ισχύει ότι

$$y = f(x) = \frac{\sum_{i=1}^{M} p^i w^i(x)}{\sum_{i=1}^{M} w^i(x)}, \quad (4.6)$$

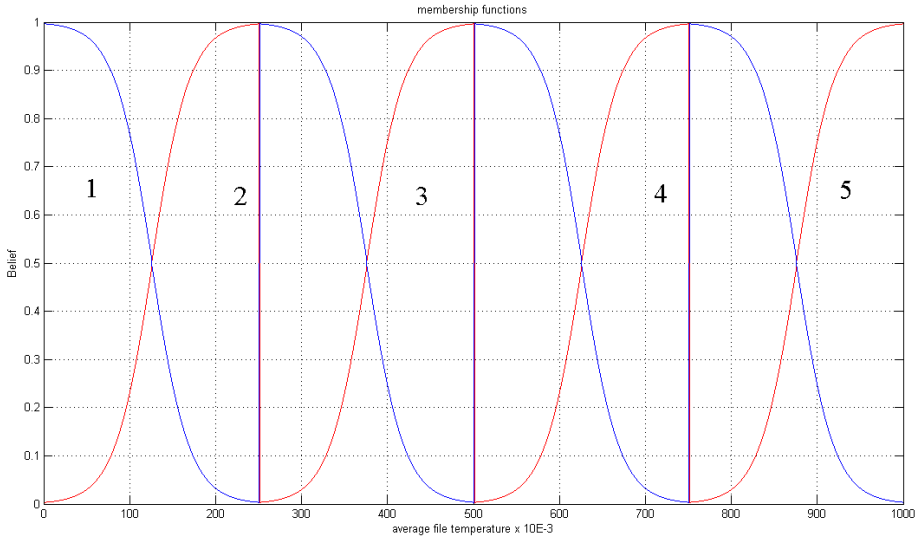
όπου M είναι ο αριθμός των ασαφών κανόνων και $w^i(x)$ είναι το βάρος του κανόνα i. Το $w^i(x)$ δίνεται από τον τύπο

$$w^i(x) = \prod_{j=1}^{K} \mu_{A_j^i}(x_j), \quad (4.7)$$

όπου $\mu_{A_j^i}(x_j)$ είναι η membership function που παρέχει το βαθμό βεβαιότητας ότι $x_j \in A_j^i$. Το σύνολο *f(x)* περιέχει ένα κανόνα για κάθε δυνατό συνδυασμό των ασαφών υποσυνόλων A_j^i, η

ένωση των οποίων καλύπτει ολόκληρο το χώρο καταστάσεων και κατ' επέκταση όλες τις δυνατές τιμές του διανύσματος εισόδου *x*. Καθώς τα υποσύνολα A_j^i επικαλύπτονται η τιμή εξόδου της f(x) εξαρτάται από τις διάφορες παραμέτρους p^i. Τέλος επισημαίνουμε ότι κάθε πράκτορας μάθησης διατηρεί τις δικές του παραμέτρους p^i τις οποίες μαθαίνει κατά τη διαδικασία μάθησης.

Στο επόμενο σχήμα φαίνεται ένα παράδειγμα διακριτοποίησης των μεταβλητών του διανύσματος κατάστασης. Θεωρούμε τη μέση θερμοκρασία s_s^1 των αρχείων που περιέχονται σε μια κλάση υπηρεσιών. Όπως φαίνεται στο σχήμα 4.1 η μεταβλητή χωρίζεται σε πέντε ασαφή υποσύνολα με χρήση σιγμοειδών συναρτήσεων.



Σχήμα 4.1: Membership functions της μεταβλητής s_s^1

Οι σιγμοειδείς συναρτήσεις κόκκινου χρώματος δίνονται από τον τύπο $\mu_L(s_s^1) = \frac{1}{\left(1+a_i e^{-b_i s_s^1}\right)}$,όπου a_i είναι η τιμή της μεταβλητής s_s^1 για την οποία κάθε μια από τις συναρτήσεις μ παίρνει την τιμή 0.5 και b_i είναι μια σταθερά που καθορίζει τη μορφή-κλίση της συνάρτησης. Οι συναρτήσεις μπλε χρώματος δίνονται από τον τύπο $\mu_S(s_s^1) = 1 - \mu_L(s_s^1)$. Αντίστοιχα διακριτοποιούνται και οι υπόλοιπες μεταβλητές που περιγράφουν το χώρο καταστάσεων.

Έχοντας παρουσιάσει το μηχανισμό του συνόλου ασαφών κανόνων παρουσιάζουμε πώς μπορεί να χρησιμοποιηθεί ως προσέγγιση της συνάρτησης τιμών V_θ του συστήματος. Θεωρούμε ότι η V_θ δίνεται από τον τύπο $V_\theta(s) = \sum_{i=1}^{L} p^i \varphi^i(s)$, όπου φ^i είναι κατάλληλες συναρτήσεις βάσεις. Σύμφωνα με τους αλγόριθμους ενισχυτικής μάθησης που παρουσιάστηκαν στην ενότητα 3.9 κάθε πράκτορας μαθαίνει τις δικές του παραμέτρους p^i χρησιμοποιώντας ως συναρτήσεις βάσεις τα κανονικοποιημένα βάρη $\frac{w^i(x)}{\sum_{i=1}^{M} w^i(x)}$, όπως αυτά προκύπτουν από το σύνολο ασαφών κανόνων *f(x).*

4.4 Βασικός βρόχος μάθησης

Στα τυπικά συστήματα κλειστού βρόχου τα αποτελέσματα μιας δράσης παρατηρούνται την επόμενη χρονική στιγμή αφού το σύστημα μεταβεί σε μια νέα κατάσταση. Χρησιμοποιώντας το παραπάνω προσομοιώσαμε το σύστημα CASTOR με τον επόμενο τρόπο.

Αρχικά το σύστημα εξυπηρετεί τις αιτήσεις των χρηστών παρατηρώντας την απόδοση της μνήμης. Μόλις είναι απαραίτητη η διαγραφή αρχείων, το σύστημα βρίσκεται σε μια κατάσταση s. Σύμφωνα με την προσομοίωση το σύστημα σταματά να εξυπηρετεί τις αιτήσεις των χρηστών και αποφασίζει ποια αρχεία θα διαγράψει λαμβάνοντας μια δράση a. Στη συνέχεια συνεχίζει να εξυπηρετεί τις αιτήσεις των χρηστών λαμβάνοντας ένα σήμα ενίσχυσης r, το οποίο είναι άμεσα συνδεδεμένο με το κόστος προσπέλασης των αρχείων και την απόδοση της μνήμης. Μόλις είναι και πάλι αναγκαία η διαγραφή αρχείων θεωρούμε ότι το σύστημα βρίσκεται σε μια κατάσταση s'. Πριν τη νέα διαγραφή αρχείων ο πράκτορας ανανεώνει την εκτίμησή του για τη συνάρτηση τιμών Q. Η παραπάνω διαδικασία περιγράφει το βασικό βρόχο μάθησης στο σύστημά μας.

4.5 Συνάρτηση επιβράβευσης των δράσεων

Η επιλογή της συνάρτησης επιβράβευσης πρέπει να είναι τέτοια ώστε να κατευθύνει τις επιλογές διαγραφής αρχείων προς την κατεύθυνση της βέλτιστης επιλογής αρχείων. Στη συγκεκριμένη μοντελοποίηση υιοθετούμε τις αρνητικές ενισχύσεις. Μια επιλογή διαγραφής αρχείων θεωρείται καλή όταν το μελλοντικό κόστος προσπέλασης αρχείων παραμένει μικρό και ο ρυθμός αστοχιών στη μνήμη είναι χαμηλός. Συνδυάζοντας αυτά τα δύο παίρνουμε την επόμενη συνάρτηση επιβράβευσης.

$$r = -\frac{total\ cache\ misses - compulsory\ misses}{total\ requests} * (episode\ average\ file\ recall\ cost) \quad (4.8)$$

Στην παραπάνω συνάρτηση ο όρος *episode_average_file_recall_cost* αποτελεί στην ουσία μια εκτίμηση του πέναλτι για ανάκτηση ενός αρχείου από τις κασέτες που βρίσκονται στο χαμηλότερο επίπεδο της ιεραρχίας μνήμης. Τέλος ως *compulsory misses* χαρακτηρίζουμε τις αστοχίες που προκύπτουν από αναφορές σε αρχεία, τα οποία αντιγράφονται για πρώτη φορά από τις κασέτες στο υψηλότερο επίπεδο της ιεραρχίας αποθήκευσης.

4.6 Αρχικές προσεγγίσεις επίλυσης

Κατά τη διάρκεια εκπόνησης της παρούσας εργασίας δοκιμάσαμε να μοντελοποιήσουμε το πρόβλημα της αντικατάστασης αρχείων ως πρόβλημα ενισχυτικής μάθησης με διαφορετικούς

τρόπους. Στην ενότητα αυτή θα παρουσιάσουμε δύο διαφορετικές προσεγγίσεις που ακολουθήσαμε, οι οποίες απορρίφθηκαν στην πορεία. Εκτός από τις προσεγγίσεις, παρουσιάζουμε και τους λόγους για τους οποίους απορρίφθηκαν. Τέλος θα παρουσιάσουμε στην επόμενη ενότητα δυο διαφορετικές προσεγγίσεις, οι οποίες βασίζονται στην αντιστοιχία του προβλήματος caching με το πρόβλημα knapsack. Θεωρούμε ότι οι δύο τελευταίες προσεγγίσεις μοντελοποιούν με σαφή τρόπο το πρόβλημα caching ως πρόβλημα ενισχυτικής μάθησης.

4.6.1 ObjectLRU inspired policy

Εμπνευσμένοι από την πολιτική ObjectLRU που αναφέραμε στην ενότητα 2.5.1 δοκιμάσαμε να αναπτύξουμε μια παρόμοια προσαρμοστική πολιτική αντικατάστασης αρχείων. Η νέα προσαρμοστική πολιτική που δοκιμάσαμε βασίζεται στη διάταξη των αρχείων που προκύπτει από την πολιτική LRU και ενσωματώνει σε αυτή νέες παραμέτρους αξιολόγησης όπως το μέγεθος του αρχείου και ο αριθμός των προσπελάσεων.

Θυμίζουμε ότι στο σύστημα CASTOR η διαγραφή αρχείων γίνεται σε τακτά διαστήματα και μόνον όταν ο τρέχων ελεύθερος χώρος της μνήμης πέσει κάτω από κάποιο όριο. Για να ενσωματώσουμε τη νέα προσαρμοστική πολιτική θεωρούμε ότι διαγράφουμε αρχεία από τη μνήμη μόνο εάν ένα νέο αρχείο πρέπει να αντιγραφεί στη μνήμη και η μνήμη είναι γεμάτη. Ο έλεγχος αυτός γίνεται κάθε φορά που ένας χρήστης ζητάει να προσπελάσει κάποιο αρχείο.

Έστω Q(s,a) η συνάρτηση τιμών του συστήματος, η οποία αποτελεί μια εκτίμηση του μελλοντικού κόστους προσπέλασης. Θεωρούμε ότι το σύστημα βρίσκεται σε μια κατάσταση s. Για να αποφασίσει ο πράκτορας ποια αρχεία θα διαγράψει από το σύστημα, εξετάζει κάθε φορά τα πρώτα N υποψήφια αρχεία προς διαγραφή, όπως αυτά προκύπτουν από τη διάταξη της πολιτικής LRU. Για τα N αυτά αρχεία εξετάζει τις 2^N πιθανές δράσεις που προκύπτουν. Αρχικά υπολογίζει τις τιμές του διανύσματος $\boldsymbol{a}$ που χαρακτηρίζει τη συγκεκριμένη δράση και έπειτα υπολογίζει την τιμή Q($\boldsymbol{s,a}$). Ο πράκτορας επιλέγει τη δράση με το μικρότερο εκτιμώμενο αθροιστικό κόστος προσπέλασης Q.

Στην υλοποίηση που δοκιμάσαμε θεωρήσαμε N = 10 και επομένως υπήρχαν 1024 πιθανές δράσεις. Χρησιμοποιήσαμε ως βάση τον αλγόριθμο SARSA(λ) με function approximator και συσσωρευτικά eligibility traces. Εκτός αυτού υποθέτουμε ότι η κρυφή μνήμη είναι συνεχώς γεμάτη οπότε εξετάζουμε εάν απαιτείται διαγραφή αρχείων πριν την εξυπηρέτηση κάθε αίτησης.

Τέλος πρέπει να αναφέρουμε ότι ανανεώνουμε τις παραμέτρους p_i ανά k ή περισσότερες αιτήσεις προσπέλασης αρχείων. Χρησιμοποιείται αυτός ο τρόπος δειγματοληψίας των μεταβάσεων κατάστασης ώστε ο πράκτορας να έχει αρκετό χρόνο για να συλλέξει στατιστικά στοιχεία σχετικά με το κόστος προσπέλασης των αιτήσεων στο σύστημα. Για

μικρές τιμές του k η μάθηση είναι ιδιαίτερα ασταθής ενώ για πολύ μεγάλες τιμές αυτού είναι πολύ αργή.

Η παραπάνω προσέγγιση αποδείχθηκε ακατάλληλη για την επίλυση του προβλήματος caching στο σύστημα CASTOR. Κατ' επέκταση θεωρείται ακατάλληλη και για άλλα ιεραρχικά συστήματα αποθήκευσης αρχείων μεγάλης κλίμακας με δομή παρόμοια με αυτή του συστήματος CASTOR.

Το κυριότερο πρόβλημα της συγκεκριμένης προσέγγισης είναι ότι η διαγραφή αρχείων γίνεται ανά αίτηση του χρήστη και όχι ανά τακτά διαστήματα. Λόγω της δομής του συστήματος CASTOR τα στοιχεία των αρχείων, συμπεριλαμβανομένου και του βάρος της πολιτικής LRU, διατηρούνται σε ένα πίνακα, το πλήθος εγγραφών του οποίου ανέρχεται σε εκατομμύρια. Επομένως η συνεχόμενη ανάκτηση των N πρώτων αρχείων βάσει της ταξινόμησης της πολιτικής LRU είναι ιδιαίτερα χρονοβόρα εάν γίνεται για κάθε εισερχόμενη αίτηση. Λόγω του παραπάνω οι χρόνοι εκτέλεσης των αλγορίθμων μάθησης δεν είναι αποδεκτοί.

Εκτός των παραπάνω η έκρηξη του χώρου δράσεων παραμένει ως πρόβλημα. Ο αριθμός N πρέπει να είναι μικρός για να έχουμε αποδεκτούς χρόνους εκτέλεσης. Βέβαια όσο μικρότερο είναι το N τόσο μειώνεται το κέρδος έναντι της LRU.

4.6.2 Διαχωρισμός των αρχείων σε κλάσεις

Μια διαφορετική προσέγγιση είναι να χωρίσουμε τα αρχεία σε κλάσεις ανάλογα με τη συχνότητα προσπέλασής τους και το μέγεθός τους. Όπως αναφέραμε σε προηγούμενη υποενότητα τα παραπάνω μεγέθη είναι δύο παράμετροι που χαρακτηρίζουν το σύνολο των αρχείων προς διαγραφή στο γενικευμένο πρόβλημα caching.

Μπορούμε να διακριτοποιήσουμε το χώρο των δράσεων σε $|s_a^1| = M$ ξεχωριστά υποσύνολα - κατηγορίες ως προς τη συχνότητα προσπέλασης των αρχείων και σε $|s_a^2| = N$ ξεχωριστά υποσύνολα - κατηγορίες ως προς το μέγεθος των αρχείων. Η διακριτοποίηση των αρχείων γίνεται δυναμικά βάσει της κατανομής των αρχείων που βρίσκονται στη μνήμη. Ο πράκτορας πλέον επιλέγει να διαγράψει αρχεία που ανήκουν σε μια κλάση i ως προς τη συχνότητα προσπέλασης και σε μια κλάση j ως προς το μέγεθος, επιλέγοντας έτσι τα χαρακτηριστικά των αρχείων που θα διαγραφούν από τη μνήμη. Πλέον ο χώρος των δράσεων έχει μέγεθος $|s_a^1|*|s_a^2| = M*N$.

Βλέπουμε ότι η συγκεκριμένη προσέγγιση δεν παρουσιάζει το πρόβλημα της έκρηξης του μεγέθους του χώρου δράσεων καθώς δεν εξαρτάται από το πλήθος των αρχείων στη μνήμη. Επιπλέον μπορεί να ενσωματωθεί στο σύστημα CASTOR χωρίς να τροποποιηθεί ο μηχανισμός διαγραφής των αρχείων. Παρά το σημαντικό πλεονέκτημα ως προς το μέγεθος του χώρου των δράσεων η συγκεκριμένη προσέγγιση παρουσιάζει την επόμενη αδυναμία.

Όπως αναφέραμε στο σύστημα CASTOR η πολιτική της αντικατάστασης των αρχείων είναι ίδια για όλα τα συστήματα αρχείων που ανήκουν σε μια συγκεκριμένη κλάση υπηρεσιών. Βάσει της αρχικής μοντελοποίησης το σύστημα διαθέτει ένα πράκτορα μάθησης ανά κλάση υπηρεσιών. Επομένως οι αποφάσεις για τα χαρακτηριστικά των αρχείων που θα διαγραφούν λαμβάνονται σε επίπεδο κλάσης υπηρεσιών, σύμφωνα με την επιλογή δράσεων που παρουσιάστηκαν στην προηγούμενη παράγραφο.

Το πρόβλημα που παρουσιάζει η συγκεκριμένη προσέγγιση προκύπτει επειδή η διαγραφή γίνεται σε επίπεδο filesystem και όχι κλάσης υπηρεσιών. Θεωρούμε ότι ο πράκτορας επιλέγει να διαγράψει αρχεία που ανήκουν σε μια κλάση i ως προς τη συχνότητα προσπέλασης και σε μια κλάση j ως προς το μέγεθος. Υπάρχει το ενδεχόμενο κάποιο filesystem της κλάσης υπηρεσιών να μην περιέχει αρχεία με τα συγκεκριμένα χαρακτηριστικά.

Μια λύση στο πρόβλημα αυτό θα ήταν η δυναμική αλλαγή ανά filesystem των ορίων των συνόλων που διακριτοποιούν το χώρο δράσεων. Όμως η συγκεκριμένη επίλυση θα δημιουργούσε ασυνέχειες στη διαδικασία της μάθησης με αποτέλεσμα ο πράκτορας να είναι ιδιαίτερα ασταθής.

4.7 Επίλυση με αναγωγή στο πρόβλημα stochastic Knapsack

Από την θεωρητική ανάλυση στην αρχή του κεφαλαίου είναι προφανής η σχέση του προβλήματος caching με το πρόβλημα knapsack. Όπως και στο πρόβλημα knapsack έτσι και στο πρόβλημα caching οι δύο μεγαλύτερες προκλήσεις είναι το μέγεθος του χώρου καταστάσεων και το μέγεθος του χώρου δράσεων του συστήματος.

Ο χώρος καταστάσεων μπορεί να αναπαρασταθεί αποδοτικά χρησιμοποιώντας τις μεταβλητές και το μηχανισμό του συνόλου ασαφών κανόνων που παρουσιάστηκαν παραπάνω. Η προσέγγιση αυτή επιτρέπει στον πράκτορα μάθησης να γενικεύει την αποκτηθείσα γνώση και να λαμβάνει καλές αποφάσεις ακόμα και σε καταστάσεις που δεν έχει επισκεφτεί ποτέ.

Παραμένει όμως το πρόβλημα του μεγέθους του χώρου δράσεων. Για να αντιμετωπίσουμε το συγκεκριμένο πρόβλημα παραλληλίζουμε το πρόβλημα caching με μια συγκεκριμένη μορφή του προβλήματος knapsack, το stochastic-Knapsack. Στην εκδοχή αυτή του προβλήματος Knapsack τα αντικείμενα φτάνουν σειριακά στο σύστημα και κάθε φορά καλούμαστε να επιλέξουμε αν θα βάλουμε το αντικείμενο στο σακίδιο ή θα το απορρίψουμε οριστικά. Βλέπουμε ότι στη συγκεκριμένη μορφή του προβλήματος ο χώρος των πιθανών δράσεων μειώνεται σε μόλις 2 δράσεις τις {accept item, reject item}.

Στη συνέχεια θα ορίσουμε το πρόβλημα stochastic-knapsack και θα παρουσιάσουμε την αντιστοιχία του προβλήματος caching με αυτό. Επίσης θα παρουσιάσουμε αναλυτικά την επίλυση του προβλήματος με χρήση ενισχυτικής μάθησης.

4.7.1 Το πρόβλημα stochastic-Knapsack

Στο stochastic Knapsack το σύστημα λαμβάνει αντικείμενα βάσει μιας στοχαστικής διαδικασίας. Κάθε αντικείμενο συνοδεύεται από μια αξία, η οποία γίνεται γνωστή στο σύστημα μόνο κατά την άφιξη του αντικειμένου σε αυτό. Η κατανομή των αξιών των αντικειμένων θεωρείται γνωστή και είναι ανεξάρτητη από το χρόνο άφιξης των αντικειμένων. Επίσης η χωρητικότητα του σακιδίου θεωρείται γνωστή. Για κάθε αντικείμενο που λαμβάνει το σύστημα πρέπει να λάβει μια απόφαση εάν θα το κρατήσει και θα το τοποθετήσει στο σακίδιο ή θα το απορρίψει οριστικά. Εάν κρατήσει το αντικείμενο τότε λαμβάνει ένα κέρδος που σχετίζεται άμεσα με την αξία του αντικειμένου, διαφορετικά εάν το αντικείμενο απορριφθεί το σύστημα λαμβάνει κάποιο πέναλτι. Όπως αναφέραμε εάν το αντικείμενο απορριφθεί δε μπορεί να χρησιμοποιηθεί στο μέλλον.

Στόχος του συστήματος είναι η εύρεση μιας βέλτιστης πολιτικής αποδοχής των εισερχόμενων αντικειμένων με σκοπό τη μεγιστοποίηση του συνολικού εκτιμώμενου κέρδους. Η πολιτική που ακολουθεί το σύστημα μπορεί να είναι είτε προσαρμοστική είτε στατική.

Ορισμός 4.1 (Προσαρμοστική πολιτική) Μια προσαρμοστική πολιτική είναι μια συνάρτηση $\wp: 2^{[n]} \times [0,1] \rightarrow [n]$. Αναλυτικότερα έχουμε ότι δεδομένου ενός συνόλου αντικειμένων *J* και μιας χωρητικότητας *c*, η πολιτική επιλέγει να βάλει το αντικείμενο $\wp(J, c)$ στο σακίδιο. Η διαδικασία αυτή επαναλαμβάνεται μέχρι να γεμίσει το σακίδιο.

Ορισμός 4.2 (Στατική πολιτική) Μια στατική πολιτική είναι στην ουσία μια διάταξη αντικειμένων $\mathcal{O} = \{i_1, \ldots, i_n\}$ βάσει ενός στατικού κριτηρίου. Τα αντικείμενα εισάγονται στο σακίδιο με τη σειρά την οποία ορίζει η διάταξη μέχρι αυτό να γεμίσει.

Έχοντας ορίσει το πρόβλημα stochastic – knapsack μπορούμε να περιγράψουμε τη σχέση του προβλήματος caching με αυτό. Όπως αναφέραμε σε προηγούμενη ενότητα στο σύστημα CASTOR καλούμαστε να επιλέξουμε ποια αρχεία θα διαγράψουμε από ένα σύνολο διαθέσιμων αρχείων *J*. Ο χώρος που πρέπει να αδειάσουμε στο δίσκο είναι γνωστός και ίσος με *c* (αρχική χωρητικότητα του σακιδίου). Θεωρούμε μια τυχαία διάταξη $\mathcal{O} = \{i_1, \ldots, i_n\}$ των υποψήφιων προς διαγραφή αρχείων. Σαρώνουμε σειριακά τη διάταξη των αρχείων και για κάθε αρχείο αποφασίζουμε εάν θα διαγραφεί (τοποθέτηση στο σακίδιο) ή θα παραμείνει στη μνήμη (απόρριψη). Βλέπουμε ότι έχουμε δύο πιθανές δράσεις {keep file, delete file}.

Η παραπάνω διαδικασία επαναλαμβάνεται μέχρι το συνολικό μέγεθος των αρχείων που θα διαγράψουμε να είναι ίσο ή μεγαλύτερο με το συνολικό χώρο που θέλουμε να αποδεσμεύσουμε. Μπορούμε επίσης να περιορίσουμε τη διαδικασία απόφασης σε ένα μόνο πέρασμα των αρχείων *J*. Με τον τρόπο αυτό η πολυπλοκότητα γίνεται ανάλογη του πλήθους των υποψήφιων προς διαγραφή αρχείων. Τα αρχεία συνοδεύονται και εδώ από ένα κέρδος, το

οποίο εξαρτάται από το μέγεθός τους και την πιθανότητα μελλοντικής τους προσπέλασης. Στόχος του συστήματος είναι η μεγιστοποίηση του εκτιμώμενου συνολικού κέρδους.

Στο [32] προτείνεται μια προσαρμοστική πολιτική επίλυσης του προβλήματος stochastic Knapsack, η οποία βασίζεται στο δυναμικό προγραμματισμό. Θεωρούμε ότι έχουμε μια διάταξη αντικειμένων j, .., n και η εναπομένουσα χωρητικότητα του σακιδίου είναι k. Επίσης ορίζουμε ως V(j,k) τη βέλτιστη εκτιμώμενη συνολική αξία του προβλήματος knapsack για την παραπάνω διάταξη και χωρητικότητα. Η δυναμική πολιτική επιλογής των αρχείων βασίζεται στον επόμενο κανόνα

$$V(j,k) = max\{V(j+1,k), v_j + V(j+1,k-s_j)\}$$

4.7.2 Επίλυση του προβλήματος με χρήση ενισχυτικής μάθησης

Σε αντιστοιχία με το πρόβλημα stochastic Knapsack μπορεί να εφαρμοστεί μια προσαρμοστική πολιτική αντικατάστασης και στο πρόβλημα αντικατάστασης αρχείων. Σύμφωνα με την προηγούμενη ανάλυσή μας ο χώρος καταστάσεων ισούται με $S = \mathbf{S_s} \times \mathbf{S_r}$, όπου S_s είναι το διάνυσμα των μεταβλητών που περιγράφουν το σύστημα και S_r είναι το διάνυσμα των μεταβλητών που περιγράφουν τις αιτήσεις των χρηστών.

Εάν αντιμετωπίσουμε το πρόβλημα αντικατάστασης των αρχείων αντίστοιχα με το πρόβλημα stochastic knapsack πρέπει να εισάγουμε το διάνυσμα S_α, το οποίο περιγράφει το αρχείο προς εξέταση. Επίσης πρέπει να επεκτείνουμε το διάνυσμα κατάστασης προσθέτοντας μια νέα μεταβλητή, τον εναπομείναντα χώρο μνήμης *c* που πρέπει να ελευθερώσουμε. Η νέα μεταβλητή αντιστοιχεί με την χωρητικότητα του σακιδίου. Όπως και οι υπόλοιπες έτσι και η νέα μεταβλητή είναι συνεχής και για την αναπαράστασή της χρησιμοποιούμε ασαφή σύνολα. Τέλος επισημαίνουμε ότι για την αναπαράσταση της συνάρτηση τιμών χρησιμοποιείται ο μηχανισμός του συνόλου ασαφών κανόνων που παρουσιάστηκε στην ενότητα 4.3.3.

Σύμφωνα με την προηγούμενη ενότητα έχουμε μόνο δύο δυνατές κινήσεις. Κάθε φορά το υπό εξέταση αρχείο μπορεί είτε να διαγραφεί από τη μνήμη είτε να διατηρηθεί σε αυτή. Επεκτείνοντας τη μέθοδο Q – learning για αναπαράσταση με function approximators και διακριτές δράσεις προκύπτει ότι πρέπει να διατηρείται μια προσεγγιστική συνάρτηση Q ανά δράση. Επομένως ο πράκτορας μάθησης διατηρεί τις συναρτήσεις τιμών $V_{del}(s) = \sum_{i=1}^{M} \theta_{del}^{i}\, \varphi_i(s)$ και $V_{keep}(s) = \sum_{i=1}^{M} \theta_{keep}^{i}\, \varphi_i(s)$.

Θεωρούμε *c* τον χώρο που πρέπει να απελευθερωθεί στη μνήμη. Έστω j το υπό εξέταση αρχείο. Το διάνυσμα S_a είναι $S_a = (temp_j, filesize_j)$. Επίσης το διάνυσμα S_s είναι γνωστό. Από τα παραπάνω έχουμε ότι το επαυξημένο διάνυσμα κατάστασης

είναι S = S_s x S_r x S_a x c. Για να αποφασίσει ποια δράση θα λάβει ο πράκτορας εκτελεί τα επόμενα.

Υπολογίζει την τιμή $V_{del}(s)$ και την τιμή $V_{keep}(s)$. Επιλέγει να διαγράψει το αρχείο με πιθανότητα $\frac{exp\left(-\frac{V_{del}(s)}{T}\right)}{exp\left(-\frac{V_{del}(s)}{T}\right)+exp\left(-\frac{V_{keep}(s)}{T}\right)}$ ή να κρατήσει το αρχείο με πιθανότητα $\frac{exp\left(-\frac{V_{keep}(s)}{T}\right)}{exp\left(-\frac{V_{del}(s)}{T}\right)+exp\left(-\frac{V_{keep}(s)}{T}\right)}$. Η παράμετρος T στις παραπάνω σχέσεις αποτελεί τον παράγοντας θερμοκρασίας που καθορίζει τη σχέση μεταξύ εξερεύνησης και εκμετάλλευσης. Πριν προχωρήσουμε με την ανάλυσή μας παρουσιάζουμε το γενικευμένο αλγόριθμο μάθησης για την παραπάνω μοντελοποίηση. Υπενθυμίζουμε ότι χρησιμοποιούμε το μηχανισμό των function approximators για την αναπαράσταση των δύο συναρτήσεων τιμών.

Στη συνέχεια θα παρουσιάσουμε πως μπορεί να ενσωματωθεί ο παρακάτω αλγόριθμος στο ιεραρχικό σύστημα CASTOR. Υπενθυμίζουμε ότι η διαγραφή αρχείων γίνεται μόλις ο ελεύθερος χώρος της μνήμης πέσει κάτω από ένα προκαθορισμένο επίπεδο. Διαγράφονται τόσα αρχεία ώστε ο ελεύθερος χώρος στη μνήμη να επανέλθει στα επιθυμητά επίπεδα.

Αλγόριθμος 4.1 stochastic Caching

Αρχικοποιούμε όλες τις παραμέτρους p_k με 0

Αρχικοποιούμε τα $e_k = 0$

Επανέλαβε

Για κάθε συνολικό επεισόδιο της μάθησης

Αρχικοποίησε την κατάσταση s_0

$i = 0$

Επανέλαβε

Διάλεξε μια δράση a_j χρησιμοποιώντας την πολιτική π, όπως αυτή προκύπτει από τις τρέχουσες παραμέτρους p

Εκτέλεση τη δράση a_i και παρατήρησε το σήμα ενίσχυσης r_i και την επόμενη κατάσταση s_{i+1}

$i = i + 1$

Μέχρι ο ελεύθερος χώρος στη μνήμη να επανέλθει στα επιθυμητά επίπεδα

Για $j = i – 1$

Ανανέωσε τις παραμέτρους p για όλες τις δράσεις που έλαβες

Ο παραπάνω μηχανισμός δε μας επιτρέπει να βλέπουμε άμεσα το αποτέλεσμα των δράσεων που λαμβάνει ο πράκτορας, καθώς σε κάθε διαδικασία διαγραφής ο πράκτορας λαμβάνει πολλαπλές αποφάσεις για διαγραφή ή διατήρηση αρχείων. Για να αντιμετωπίσουμε το πρόβλημα αυτό προτείνουμε τον επόμενο τρόπο.

Κατά τη διάρκεια της διαδικασίας διαγραφής κρατάμε προσωρινά σε δύο πίνακες το ιστορικό των δράσεων που έλαβε ο πράκτορας. Οι δύο πίνακες αντιστοιχούν στις δύο δράσεις {delete file, keep file}. Όταν ο πράκτορας αποφασίσει ότι πρέπει να διαγράψει κάποιο αρχείο εισάγει μια εγγραφή στον πρώτο πίνακα, η οποία περιέχει την τρέχουσα τιμή της παραμέτρου *c*, τα χαρακτηριστικά του αρχείου {temperature, filesize}και την τιμή $V_{del}(s)$. Αντίστοιχα όταν αποφασίσει να κρατήσει κάποιο αρχείο στη μνήμη εισάγει μια εγγραφή στον δεύτερο πίνακα.

Έχοντας κρατήσει το ιστορικό των δράσεων του πράκτορα συνεχίζουμε να εξυπηρετούμε τις αιτήσεις των χρηστών μέχρι να χρειαστεί η ενεργοποίηση του μηχανισμού διαγραφής αρχείων. Πλέον έχουμε λάβει από το σύστημα ένα γενικευμένο αθροιστικό σήμα ενίσχυσης R, το οποίο αποτελείται από δύο ποσότητες τις R_{hits} και $R_{misses.}$ Η πρώτη ποσότητα είναι θετική και ισούται με το εκτιμώμενο κέρδος του συστήματος για τις αιτήσεις που εξυπηρέτησε από την cache. Το κέρδος ισούται με υποθετικό κόστος εξυπηρέτησης από το χαμηλότερο επίπεδο της μνήμης. Αντίστοιχα η δεύτερη ποσότητα είναι αρνητική (penalty) και ισούται με το κόστος ανάκτησης των αρχείων επί μείον ένα. Επίσης γνωρίζουμε την επόμενη κατάσταση του συστήματος. Πριν προχωρήσουμε στην εκτέλεση της διαγραφής των αρχείων ανανεώνουμε τις παραμέτρους p των συναρτήσεων V_{keep}, V_{del} χρησιμοποιώντας το προηγούμενο ιστορικό διαγραφής και το σήμα ενίσχυσης R.

4.7.3 Αλγόριθμοι stochastic Caching

Στην ενότητα αυτή παρουσιάζουμε αναλυτικά τους αλγορίθμους που μοντελοποιούν το πρόβλημα της αντικατάστασης αρχείων ως ένα πρόβλημα ενισχυτικής μάθησης κατά αντιστοιχία με το πρόβλημα stochastic knapsack.

Αλγόριθμος 4.2 stochastic Caching – Action Selection using Boltzmann distribution

Input: file_temperature, file_size, diskpace_to_be_freed

Output: action a ∈{delete, keep}

Set s = current_system_state x (*file_temperature, file_size)* x *diskpace_to_be_freed*

Compute V_{del}

Compute V_{keep}

$$Pr(del) = \frac{exp\left(-\frac{V_{del}(s)}{T}\right)}{exp\left(-\frac{V_{del}(s)}{T}\right)+exp\left(-\frac{V_{keep}(s)}{T}\right)}$$

$$Pr(keep) = \frac{exp\left(-\frac{V_{keep}(s)}{T}\right)}{exp\left(-\frac{V_{del}(s)}{T}\right)+exp\left(-\frac{V_{keep}(s)}{T}\right)}$$

P = rand

If (P ≤ Pr(del)) then

a ← delete

Insert into delete_decision (file_temperature, file_size, diskpace_to_be_freed,V_{del}*)*

else

a ← keep

Insert into keep_decision (file_temperature, file_size, diskpace_to_be_freed,V_{keep}*)*

Αλγόριθμος 4.3 stochastic Caching – File deletion

```
For every file system i
do
  Compute diskspace_to_be_freed
  diskspace_freed = 0
  For every file j in file system i
  do
    if (diskspace_freed < diskspace_to_be_freed) then
        c ← diskspace_to_be_freed – diskspace_freed
        a ← select_action(temp_j, filesize_j, c)
        if (a is delete) then diskspace_freed += filesize_j
    else
        break
    end for
 end for
```

Αλγόριθμος 4.4 stochastic Caching – update theta values

Input: system state s, system state s′, total reward r

files_deleted ← *number of delete_decision Table entries*

files_kept ← *number of keep_decision Table entries*

//Scale down reward

$r_{del} = R_{misses} / files\_deleted$

$r_{keep} = R_{hit} / files\_kept$

$\alpha \leftarrow learning\ rate$

For every entry i in delete_decision Table

do

$delta \leftarrow r_{del} - i.V_{del}$

$\bar{\theta}_{del} \leftarrow \bar{\theta}_{del} + a * delta * \bar{\varphi}(s, i.temperature, i.filesize, i.c)$

end for

For every entry i in keep_decision Table

do

$delta \leftarrow r_{keep} - i.V_{keep}$

$\bar{\theta}_{keep} \leftarrow \bar{\theta}_{keep} + a * delta * \bar{\varphi}(s, i.temperature, i.filesize, i.c)$

end for

Αλγόριθμος 4.5 stochastic Caching – Q learning with function approximation

Initialize $\theta_{del} = 0$ *and* $\theta_{keep} = 0$ *epoch = 0*

While (epoch < A)

initialize system – run simulation until File Deletion is needed

$s \leftarrow$ *grab system state*

While (file request stream is not over)

file_deletion()

run simulation until File Deletion is needed

$r \leftarrow$ *grab reward*

$s' \leftarrow$ *grab system state*

update_theta_values (s,s′,r)

$s \leftarrow s'$

end loop

$epoch \leftarrow epoch + 1$

decrease learning rate and action selection Temperature

end loop

4.8 Επίλυση με αναγωγή στο πρόβλημα min-Knapsack

Στην ενότητα αυτή θα παρουσιάσουμε μια διαφορετική προσέγγιση μοντελοποίησης του προβλήματος caching ως πρόβλημα ενισχυτικής μάθησης. Η δομή του γενικευμένου προβλήματος caching παρουσιάζει αρκετές ομοιότητες με τη μορφή μιας διαφορετικής έκδοσης του προβλήματος Knapsack, το πρόβλημα min-Knapsack. Στην αρχή της συγκεκριμένης ενότητας θα παρουσιάσουμε το πρόβλημα min-Knapsack ενώ στη συνέχεια θα αναλύσουμε τη μοντελοποίηση που προτείνουμε.

Σε αντίθεση με την προηγούμενη προσέγγιση του stochastic Caching, εδώ ο χώρος πιθανών δράσεων παραμένει ίδιος με το γενικευμένο πρόβλημα caching. Αντί να τροποποιήσουμε τη μοντελοποίηση του προβλήματος με στόχο τη μείωση των πιθανών δράσεων, επιλέξαμε να αντιμετωπίσουμε την κατάρα των διαστάσεων περιορίζοντας την εξερεύνηση του πράκτορα σε καλές περιοχές του χώρου των δράσεων. Είναι αναγκαίο η εξερεύνηση του πράκτορα να ξεκινάει από μια δράση, η οποία δίνει μια σχετικά καλή λύση του προβλήματος και να συνεχίζει σε καλές γειτονικές περιοχές του χώρου δράσεων.

Για την εύρεση μιας καλής αρχικής λύσης προτείνουμε μια στατική πολιτική αντικατάστασης αρχείων ενώ για την εξερεύνηση του χώρου δράσεων προτείνουμε ένα συνδυασμό της μεθόδου προσομοιωμένης ανόπτησης με τη μέθοδο της ενισχυτικής μάθησης. Περισσότερες λεπτομέρειες ακολουθούν στη συνέχεια της ενότητας.

4.8.1 Το πρόβλημα min-Knapsack

Πριν προχωρήσουμε με την ανάλυση μας πρέπει να ορίσουμε το πρόβλημα min-Knapsack. Σύμφωνα με το [33] το πρόβλημα ορίζεται ως εξής:

Ορισμός 4.3 (Min – Knapsack Problem) Έστω n αντικείμενα όπου σε κάθε αντικείμενο i αντιστοιχούν δύο θετικοί ακέραιοι (c_i, w_i), όπου c_i είναι η αξία του αντικειμένου και w_i το βάρος του. Δεδομένου ενός θετικού ακεραίου M βρες κάποια x_1, x_2, …, x_n τέτοια ώστε

$$minimize \quad \sum_{j=1}^{n} c_j x_j$$

$$s.t. \quad \sum_{j=1}^{n} w_j x_j \geq M$$

$$x_j \in \{0,1\}, j = 1, \dots, n$$

Καθώς το πρόβλημα είναι NP-hard όπως και η τυπική μορφή του προβλήματος 0-1 knapsack, οι Csirik et al. [33] προτείνουν ένα ευριστικό αλγόριθμο 2-OPT για την επίλυση του

προβλήματος. Ο αλγόριθμος αυτός βασίζεται στην ταξινόμηση των αρχείων σε αύξουσα σειρά ως προς το σχετικό κόστος τους $\frac{c_i}{w_i}$.

4.8.2 Προσεγγιστική στατική πολιτική

Όπως αναφέρθηκε στην αρχή της ενότητας είναι αναγκαία η εύρεση μιας σχετικά καλής δράσης, η οποία χρησιμοποιείται ως αρχή στην εξερεύνηση του χώρου δράσεων. Κατά αντιστοιχία με την ευριστική λύση του προβλήματος min-knapsack που προτείνεται στο [33] εισάγουμε μια προσεγγιστική στατική πολιτική για την αντικατάσταση των αρχείων.

Τα αρχεία που βρίσκονται αποθηκευμένα στη μνήμη χαρακτηρίζονται από το μέγεθός τους *size* και τη θερμοκρασία τους *temp*. Το μέγεθος ενός αρχείου αντιστοιχεί στο βάρος ενός αντικειμένου για το πρόβλημα knapsack. Εκτός του βάρους είναι απαραίτητος και ο ορισμός της αξίας των αρχείων.

Ο στόχος στο πρόβλημα min-knapsack είναι εύρεση ενός συνόλου αντικειμένων, τα οποία καταλαμβάνουν χώρο μεγαλύτερο από τη χωρητικότητα του σακιδίου και η συνολική τους αξία είναι ελάχιστη. Αντίστοιχα, στο πρόβλημα αντικατάστασης αρχείων στόχος είναι η ελαχιστοποίηση του συνολικού κόστους προσπέλασης των αρχείων που θα διαγραφούν από τη μνήμη. Επομένως η αξία κάθε αρχείου σχετίζεται με το εκτιμώμενο κόστος προσπέλασης του αρχείου. Ορίζουμε λοιπόν ως αξία του αρχείου *i* το μέγεθος

$$value(i) = \Pr_{\text{request}}(i) * recall\_cost(i), \qquad (4.9)$$

όπου η πιθανότητα προσπέλασης του αρχείου *i* είναι ίση με

$$\Pr_{\text{request}}(i) = temp_i = \frac{1}{request\ interval_i} \quad (4.10)$$

Το κόστος ανάκλησης του αρχείου από το χαμηλότερο επίπεδο της ιεραρχίας δίνεται για το σύστημα CASTOR από τον τύπο

$$recall\ cost = t_{queuing} + \frac{file\ size\ in\ MB}{90\frac{MB}{sec}}, (4.11)$$

όπου $t_{queuing}$ είναι ο εκτιμώμενος χρόνος αναμονής για την εξυπηρέτησης της αίτησης του χρήστη και τα 90 MB/sec είναι ο ρυθμός ανάγνωσης από τις κασέτες του χαμηλότερου επιπέδου της μνήμης. Για μια καλή εκτίμηση του $t_{queuing}$ είναι απαραίτητη η διατήρηση στατιστικών κατά τη διάρκεια λειτουργίας του συστήματος.

Έχοντας αναφέρει τα παραπάνω ορίζουμε την επόμενη στατική πολιτική αντικατάστασης. Έστω M ο συνολικός χώρος που πρέπει να απελευθερωθεί σε ένα σύστημα αρχείων. Ταξινομούμε τα αρχεία που περιέχονται στο σύστημα αυτό σε αύξουσα σειρά ως προς τη

σχετική τους αξία $\frac{value(i)}{size(i)}$ και επιλέγουμε να διαγράψουμε τα πρώτα *n* αρχεία το συνολικό μέγεθος των οποίων ισούται ή υπερβαίνει οριακά την τιμή της παραμέτρου M.

Όπως φαίνεται η νέα αυτή πολιτική εκτός της συχνότητα προσπέλασης του κάθε αρχείου λαμβάνει υπόψη και άλλες παραμέτρους όπως το μέγεθος του αρχείου αλλά ο χρόνος απόκρισης του συστήματος. Χρησιμοποιώντας την πολιτική αυτή παίρνουμε μια σχετικά καλή λύση για το πρόβλημα αντικατάστασης αρχείων, την οποία μπορούμε να βελτιώσουμε χρησιμοποιώντας τον τρόπο που προτείνεται στη συνέχεια.

4.8.3 Εξερευνώντας το χώρο των δράσεων με Προσομοιωμένη Ανόπτηση

Στη βιβλιογραφία συναντήσαμε περιπτώσεις [20][21] όπου προτείνεται η προσεγγιστική επίλυση (Simulated Annealing) του προβλήματος 0-1 Knapsack με χρήση της μεθόδου προσομοιωμένης ανόπτησης. Η προσομοιωμένη ανόπτηση είναι ένας στοχαστικός ευριστικός αλγόριθμος, ο οποίος χρησιμοποιείται για την εύρεση προσεγγιστικών λύσεων σε προβλήματα μεγάλης κλίμακας.

Ο αλγόριθμος SA ξεκινάει από μια αποδεκτή κατάσταση s με αξία f(s) και εκτελώντας μια τυχαία περιπλάνηση στο χώρο καταστάσεων προσπαθεί να βρει νέες καταστάσεις s΄ ∈ S με μεγαλύτερη αξία από την τρέχουσα επιλεγμένη κατάσταση s. Η αξιολόγηση των νέων καταστάσεων γίνεται με τον επόμενο τρόπο:

Ο αλγόριθμος δέχεται μια νέα κατάσταση s΄ ως αντικατάσταση της s με πιθανότητα

$$p(< s, s' >) = \begin{cases} 1, & εάν\ f(s') > f(s) \\ e^{\frac{f(s') - f(s)}{T}}, & διαφορετικά \end{cases}, \quad (4.12)$$

όπου T είναι η παράμετρος θερμοκρασίας του αλγορίθμου.

Χρησιμοποιώντας κατάλληλα τον αλγόριθμο SA μπορούμε να εξερευνήσουμε το χώρο δράσεων και να βρούμε μια καλή προσεγγιστική λύση για το πρόβλημα αντικατάστασης αρχείων. Πριν παρουσιάσουμε τον αλγόριθμο SA για την επίλυση του προβλήματος caching πρέπει να αναφέρουμε ορισμένες παραδοχές.

Αρχικά πρέπει να επισημάνουμε ότι ταυτίζουμε το χώρο δράσεων του πράκτορα μάθησης του συστήματος με το χώρο καταστάσεων που εξετάζει ο αλγόριθμος SA. Έτσι μια κατάσταση s αντιστοιχεί με ένα σύνολο αρχείων, υποψήφιων προς διαγραφή. Συμβολίζουμε το σύνολο των αρχείων με *A* και με θεωρούμε ότι κάθε σύνολο *A* έχει αξία *V(A)*. Για να ενισχύσουμε την απόδοση του αλγορίθμου SA κατευθύνουμε την έρευνα του χώρου καταστάσεων σε μια περιοχή καλών δράσεων, ορίζοντας ως αρχική δράση (αρχική κατάσταση του αλγορίθμου) το σύνολο *A* των αρχείων που προκύπτει από τη στατική πολιτική που παρουσιάστηκε στην προηγούμενη υποενότητα.

Οι νέες δράσεις που ερευνά ο αλγόριθμος SA είναι γειτονικές με την τρέχουσα επιλεγμένη δράση του *A*. Για την κατασκευή γειτονικών δράσεων ο αλγόριθμος επιλέγει τυχαία κάποιο αρχείο που ανήκει στο σύνολο *A* και το αντικαθιστά με κάποιο τυχαίο αρχείο που δεν ανήκε στο *A*. Ο τρόπος κατασκευής νέων δράσεων παρουσιάζεται λεπτομερώς στον αλγόριθμο που φαίνεται στη συνέχεια.

Δεδομένης μιας καλής αρχικής λύσης μπορούμε να ενισχύσουμε περαιτέρω την απόδοση του αλγορίθμου SA περιορίζοντας την έρευνα του σε καλές περιοχές του χώρου δράσεων. Αποφεύγουμε έτσι ένα από τα σημαντικότερα μειονεκτήματα του αλγορίθμου SA, το οποίο είναι ότι σπαταλά αρκετό χρόνο για να εξερευνήσει κακές κινήσεις, αφού πολλές από τις τυχαίες καταστάσεις που εξετάζει έχουν ιδιαίτερα χαμηλή αξία.

Για να πετύχουμε το παραπάνω ταξινομούμε τα αρχεία κατά αύξουσα σειρά αξίας, όπως αυτή ορίζεται στον τύπο (4.9). Τα αρχεία που επιλέγονται να προστεθούν στο σύνολο *A* των αρχείων προς διαγραφή ανήκουν στα top-k αρχεία της παραπάνω ταξινόμησης. Στη συνέχεια παρατίθεται ο αλγόριθμος SA για προσεγγιστική επίλυση του προβλήματος caching.

Αλγόριθμος 4.6 Simulated Annealing Deletion

Input: system state s, request state r,
Output: Q, action state a
Total_space ← *diskspace_to_be_freed*
A ← ∅
A ← *compute initial solution according to Greedy Policy* ($\frac{\Pr_{\text{request}}(i) * recall\_cost(i)}{file\ size\ i}$)
//compute the value of A
$Q_{selected\ action}$ ← *Q(s,r,A)*
//Initialize current capacity
current_space ← $\sum_{j \in A} w_j$
//Initialize temperature to T_0
$T \leftarrow T_0$
loop_flag ← *0, flag* ← *0*
*while (loop_flag = 0) and (T > 0.1*T₀) then*
    flag ← *0*
    for 1 to L
        Temp_files ← *order files j where j* ∉ *A in non-decreasing order according to their value*
        Temp_files ← *select top – k files from Temp_files*
        file_to_add ← *select a random file from Temp_files*
        if (current_space < total_space) then

```
                    A ← A ∪ file_to_add
            else
                    file_to_remove ← select a random file from A
                    delta_space ← file_to_add .filesize – file_to_remove.filesize
                    Q_temp_action ← Q(s,r, (A – {file_to_remove})∪ file_to_add)
                    delta_Q ← Q_temp_action – Q_selected_action
                    if (current_space + delta_space ≥ total_space) then
                            if (delta_Q > 0) or (e^(delta_Q/T) > rand) then
                                    current_space ←current_space + delta_space
                                    Q_selected_action = Q_temp_action
                                    A ← A – {file_to_remove})∪ file_to_add
                                    flag ← 1
                            end if
                    end if
            end if

        end for loop
        T ← T_0*0.95
        if (flag = 0) then
                loop_flag ← 1
        else
                loop_flag ← 0
        end if
end while loop
a ← compute action parameters of set A
Q← Q_selected_action
```

4.8.4 Ενισχυτική μάθηση για την εκμάθηση της συνάρτησης χρησιμότητας

Όπως βλέπουμε στην προηγούμενη παράγραφο ο αλγόριθμος προσομοιωμένης ανόπτησης χρησιμοποιεί μια αντικειμενική συνάρτηση χρησιμότητας για την αξιολόγηση των δράσεων που εξετάζει. Μια πρώτη προσέγγιση θα ήταν να χρησιμοποιήσουμε το άθροισμα της αξίας των αρχείων προς διαγραφή, όπως αυτή ορίζεται στον τύπο (4.9). Η συγκεκριμένη αντικειμενική συνάρτηση μας παρέχει μια εκτίμηση για το κόστος προσπέλασης των αρχείων που θα διαγραφούν από το σύστημα.

Θα ήταν προτιμότερο εάν ο παραπάνω αλγόριθμος χρησιμοποιούσε μια συνάρτηση χρησιμότητας, η οποία ανάλογα με την κατάσταση του συστήματος και το μοτίβο προσπέλασης των αρχείων από τους χρήστες παρείχε για κάθε πιθανή δράση μια εκτίμηση για το μελλοντικό μέσο κόστος προσπέλασης των αρχείων. Για την εκμάθηση μιας τέτοιας συνάρτησης χρησιμότητας θα χρησιμοποιήσουμε τη μέθοδο της ενισχυτικής μάθησης.

Ο νέος αλγόριθμος που προτείνουμε βασίζεται σε ένα συνδυασμό της ενισχυτικής μάθησης με τη μέθοδο της προσομοιωμένης ανόπτησης. Η προσομοιωμένη ανόπτηση χρησιμοποιείται για την εξερεύνηση του χώρου δράσεων και την επιλογή μιας καλής δράσης, ενώ η ενισχυτική μάθηση χρησιμοποιείται για την εκμάθηση μια καλής συνάρτησης χρησιμότητας. Ακολουθεί ο αλγόριθμος που συνδυάζει τις δύο παραπάνω μεθόδους με στόχο την υλοποίηση μιας προσαρμοστικής πολιτικής για την επίλυση του προβλήματος αντικατάστασης αρχείων.

Αλγόριθμος 4.7 Simulated Annealing combined with Reinforcement Learning – SARSA(λ)

Initialize $\theta = 0$
epoch = 0
While (epoch < A)
    initialize system – run simulation until File Deletion is needed
    s ← *grab system state*
    s_r ← *grab request state*
    ($Q_{previous}$, *A)* ←*simmulated annealing deletion (s, s_r)*
    While (file request stream is not over)
        run simulation until File Deletion is needed
        r ← *grab reward*
        $\bar{e} \leftarrow \gamma\lambda\bar{e} + \nabla_\theta Q(s, s\_r, A)$
        s´ ← *grab system state*
        s_r´ ← *grab request state*
        ($Q_{current}$, *A´)* ←*simmulated annealing deletion (s´, s_r´)*
        delta ← $r + \gamma\, Q_{current}$ - $Q_{previous}$
        $\bar{\theta} \leftarrow \theta + a * delta * \bar{e}$
        $s \leftarrow s'$
        $s\_r \leftarrow s\_r'$
        $Q_{previous} = Q_{current}$
    end loop
    epoch ← *epoch + 1*
    decrease learning rate
end loop

5

Υλοποίηση και Αξιολόγηση

Στο κεφάλαιο αυτό θα παρουσιάσουμε την πλατφόρμα προσομοίωσης που υλοποιήσαμε για την αξιολόγηση των αλγορίθμων που προτείναμε στο προηγούμενο κεφάλαιο. Επίσης θα παρουσιάσουμε τις παραμέτρους αξιολόγησης που μπορούμε να χρησιμοποιήσουμε. Τέλος παραθέτουμε τα πρώτα αποτελέσματα που έχουν προκύψει από την εκτέλεση των παραπάνω αλγορίθμων μέχρι την τρέχουσα χρονική στιγμή.

5.1 Σύστημα αξιολόγησης

Καθώς η παρούσα εργασία ξεκίνησε ως μια προσπάθεια βελτίωσης των πολιτικών αντικατάστασης του ιεραρχικού συστήματος διαχείρισης CASTOR, δημιουργήσαμε ένα προσομοιωτή του συστήματος αυτού για την εκτέλεση των πειραμάτων μας.

Το σύστημα CASTOR βασίζεται σε ένα database schema στο οποίο διατηρούνται οι απαραίτητες πληροφορίες για τα αρχεία, τα οποία διαχειρίζεται, όπως το μέγεθός τους, η κατάστασή τους και διάφορα στατιστικά στοιχεία. Επειδή ο αριθμός των αιτήσεων αλλά και των αρχείων είναι ιδιαίτερα μεγάλος είναι απαραίτητη η χρήση μιας βάσης δεδομένων ώστε να μπορεί το σύστημα να εξυπηρετήσει κάθε αίτηση ανεξάρτητα από τις υπόλοιπες. Παράλληλα με τη βάση υπάρχουν πολλοί daemons, οι οποίοι εκτελούν τις απαραίτητες διεργασίες σε χαμηλότερο επίπεδο, όπως η αντιγραφή ενός αρχείου από τις κασέτες στους δίσκους.

Για την υλοποίηση του προσομοιωτή θεωρήσαμε σκόπιμο να χρησιμοποιήσουμε και εμείς μια βάση δεδομένων παρόμοια με αυτή του πραγματικού συστήματος. Οι διάφορες εργασίες, όπως η διαγραφή αρχείων από την cache ή αντιγραφή αρχείων από τις κασέτες στους δίσκους, προσομοιώνονται με ορισμένες διεργασίες στη βάση. Οι διεργασίες αυτές υλοποιούνται ως εσωτερικές διεργασίες της βάσης και είναι γραμμένες σε PL/SQL.

Στη συνέχεια θα παρουσιάσουμε σύντομα το σχήμα της βάσης που χρησιμοποιούμε καθώς και τις διάφορες διεργασίες. Εδώ πρέπει να τονίσουμε ότι ο προσομοιωτής μας παρέχει τη δυνατότητα της ταυτόχρονης εκτέλεσης διαφορετικών πολιτικών αντικατάστασης για την ίδια ροή αιτήσεων των χρηστών. Έχουμε λοιπόν τη δυνατότητα να συγκρίνουμε αμέσως τις διαφορετικές πολιτικές αντικατάστασης.

5.1.1 Database Schema

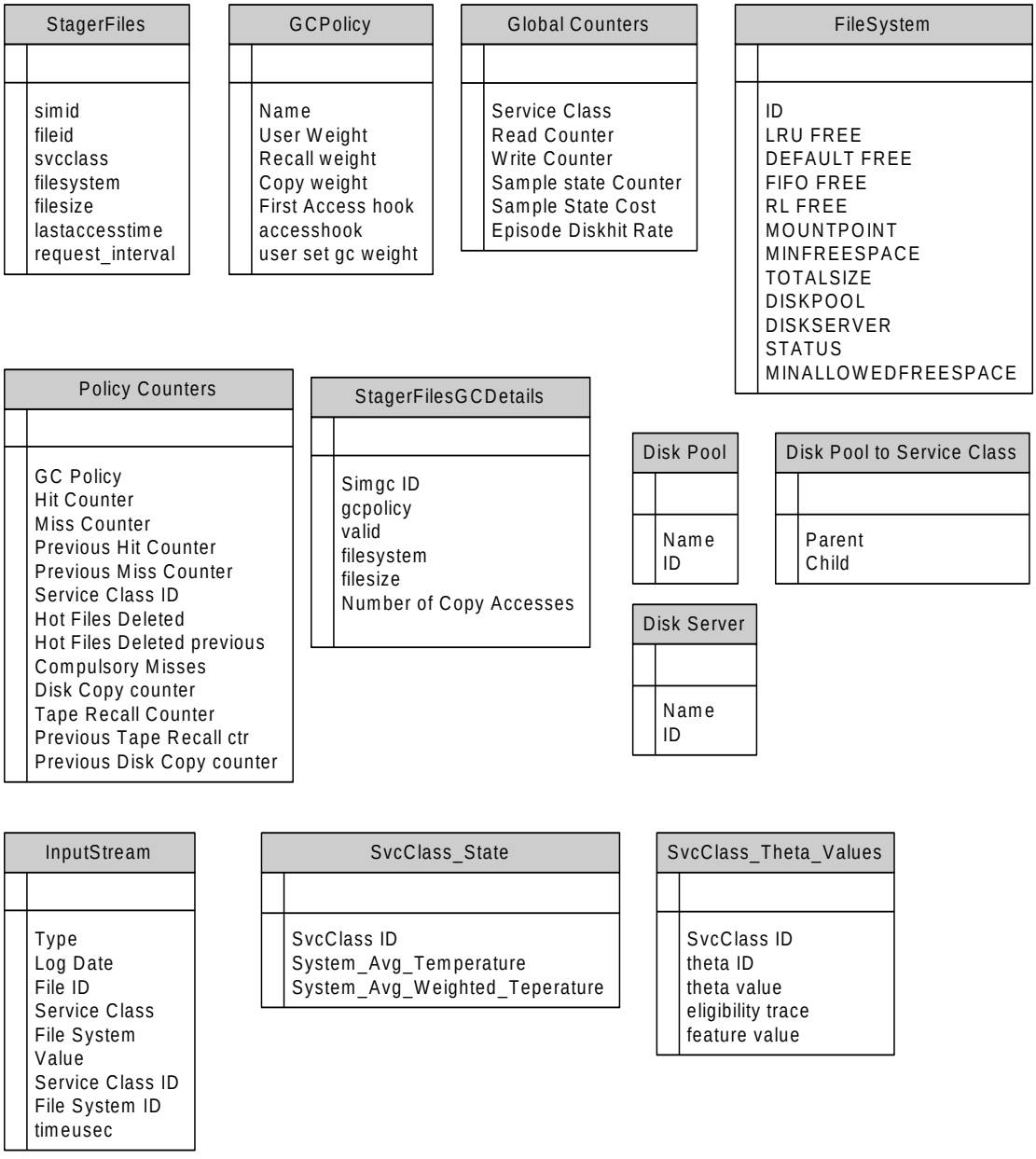
Στο σχήμα 5.1 φαίνονται μερικοί από τους βασικούς πίνακες του προσομοιωτή. Παρόμοιοι πίνακες χρησιμοποιούνται και στο πραγματικό σύστημα για τη διαχείριση των αιτήσεων αλλά και τη διαχείριση στοιχείων σχετικά με τα αρχεία του συστήματος. Οι τρεις βασικοί πίνακες του προσομοιωτή μας είναι οι StagerFiles, StagerFilesGCDetails και InputStream.

Στον πρώτο πίνακα διατηρούμε γενικές πληροφορίες για τα αρχεία του συστήματος. Επειδή ο αριθμός των αρχείων για τα οποία διατηρούμε πληροφορίες ανέρχεται περίπου στα 7.000.000 και εφαρμόζεται διαφορετική πολιτική αντικατάστασης ανά κλάση υπηρεσιών αποφασίσαμε να χωρίσουμε τον συγκεκριμένο πίνακα σε partitions ανά κλάση υπηρεσιών. Ο παραπάνω διαχωρισμός συνδυασμένος με τα κατάλληλα ευρετήρια μειώνει σημαντικά το χρόνο εκτέλεσης των διεργασιών στη βάση.

Στον δεύτερο πίνακα διατηρούμε πιο εξειδικευμένες πληροφορίες που αφορούν τις διαφορετικές πολιτικές αντικατάστασης για κάθε αρχείο που περιέχεται στον πρώτο. Η συγκεκριμένη υλοποίηση μας επιτρέπει να εξετάζουμε παράλληλα την απόδοση όλων των πολιτικών που θέλουμε να αξιολογήσουμε. Τα δεδομένα που διατηρούνται στον συγκεκριμένο πίνακα χρησιμοποιούνται επί το πλείστον από τον αλγόριθμο διαγραφής. Υπενθυμίζουμε ότι η διαγραφή αρχείων εκτελείται ανά file system. Το παραπάνω σε συνδυασμό με το μέγεθος του πίνακα μας οδήγησε στη διαμέριση του πίνακα σε partitions ανά filesystem. Πάλι ο παραπάνω διαχωρισμός συνδυασμένος με τα κατάλληλα ευρετήρια μειώνει σημαντικά το χρόνο εκτέλεσης των διεργασιών στη βάση.

Τέλος στον τρίτο πίνακα έχουμε αποθηκεύσει τις αιτήσεις των χρηστών για ένα συγκεκριμένο χρονικό διάστημα αφού πρώτα τις έχουμε επεξεργαστεί ώστε κάθε εγγραφή να περιέχει όλες τις απαραίτητες πληροφορίες σχετικά με κάθε αίτηση. Μερικές από τις πληροφορίες αυτές είναι ο τύπος της αίτησης, το μέγεθος του αρχείου και το file system στο οποίο θα αντιγραφεί. Εκτός των παραπάνω πινάκων υπάρχουν και άλλοι πίνακες, όπως

πίνακες στους οποίους διατηρούμε στατιστικά στοιχεία και οι πίνακες που διατηρούμε τις μεταβλητές της διαδικασίας ενισχυτικής μάθησης.



Σχήμα 5.1: Το Database schema στο οποίο βασίζεται ο προσομοιωτής του συστήματος

5.1.2 Διεργασίες PL/SQL

Εσωτερικά στη βάση δεδομένων λειτουργούν διάφορες διεργασίες, οι οποίες προσομοιώνουν τη λειτουργία του πραγματικού συστήματος. Στην παρούσα εργασία θεωρήθηκε σκόπιμο να μην παρουσιάσουμε αναλυτικά τις διεργασίες αυτές καθώς η υλοποίησή τους δεν παρουσιάζει κάποιο ιδιαίτερο ενδιαφέρον. Γενικά ο τρόπος λειτουργίας του προσομοιωτή είναι ο επόμενος.

Η ανάγνωση των εισερχόμενων αιτήσεων χωρίζεται σε επεισόδια της μισής ώρας. Οι αιτήσεις που περιέχονται στο τρέχον χρονικό παράθυρο αποκωδικοποιούνται και εκτελούνται οι ανάλογες εσωτερικές διεργασίες, όπως η αντιγραφή ενός αρχείου από κάποια κασέτα στην cache ή η αντιγραφή του από κάποιο άλλο filesystem που ανήκει σε διαφορετική κλάση υπηρεσιών.

Εκτός των παραπάνω σε κάθε προσπέλαση αρχείου αντιστοιχεί ένας κόστος. Θεωρούμε ότι για αρχεία, τα οποία βρίσκονται ήδη αποθηκευμένα στη μνήμη ο χρόνος αναμονής του χρήστη και το κόστος προσπέλασης του αρχείου είναι μηδενικά. Για αρχεία τα οποία πρέπει να αντιγραφούν είτε από κάποια κασέτα είτε από κάποιο άλλο σκληρό δίσκο το κόστος προσπέλασης υπολογίζεται σύμφωνα με τον τρόπο που αναφέρεται στη συνέχεια.

Το κόστος αντιγραφής ενός αρχείου στο υψηλότερο επίπεδο της ιεραρχίας μνήμης είναι ίσο με το χρόνο που απαιτείται για την αντιγραφή του αρχείου από τις κασέτες συν μια χρονική καθυστέρηση για την εξυπηρέτηση της αίτησης. Για αντιγραφή αρχείων από άλλους δίσκους η αναμονή για εξυπηρέτηση είναι μηδενική. Αντίθετα για αντιγραφή αρχείων από τις κασέτες η αναμονή εξυπηρέτησης μπορεί να ανέρχεται ακόμα και σε ώρες.

Ο χρόνος αντιγραφής εξαρτάται από το μέγεθος του αρχείου και την ταχύτητα ανάγνωσης των μέσων αποθήκευσης. Στο σημείο αυτό θεωρούμε ότι η ταχύτητα μεταφοράς δεδομένων στο δίκτυο, το οποίο χρησιμοποιείται για τη μεταφορά του αρχείου, είναι μεγαλύτερη από την ταχύτητα των μέσων αποθήκευσης. Επομένως η ταχύτητα αντιγραφής περιορίζεται από την ταχύτητα ανάγνωσης των μέσων αποθήκευσης. Έτσι για αντιγραφή αρχείου από μια κασέτα ο χρόνος ανάγνωσης είναι 90 MB/sec ενώ για αντιγραφή από δίσκο είναι 70 MB/sec.

Η χρονική καθυστέρηση εξυπηρέτησης της αίτησης εξαρτάται από το φόρτο του δικτύου και το πλήθος των αιτήσεων που δέχεται το σύστημα. Για το χρόνο αναμονής έχουμε μαζέψει στατιστικά στοιχεία από το πραγματικό σύστημα, καθώς δεν έχουμε άμεση πρόσβαση σε αυτό. Αυτό που κάνουμε είναι να διατηρούμε το μέσο όρο και την τυπική απόκλιση της χρονικής καθυστέρησης των αιτήσεων για κάθε μέρα που εμφανίζεται στο αποθηκευμένο ιστορικό των αιτήσεων των χρηστών. Ο προσομοιωτής λαμβάνει κάποιο δείγμα από μια γκαουσιανή κατανομή, η οποία χρησιμοποιεί το μέσο όρο και την τυπική απόκλιση που χαρακτηρίζουν τη μέρα που αντιστοιχεί στην αίτηση.

5.2 Παράμετροι αξιολόγησης

Στην παρούσα ενότητα θα παρουσιάσουμε ορισμένες μετρικές, οι οποίες μπορούν να χρησιμοποιηθούν για την αξιολόγηση της απόδοσης των διαφορετικών πολιτικών αντικατάστασης. Επιγραμματικά οι μετρικές αυτές είναι:

- Hit rate
- Maximum Improvement
- Byte hit rate

Όπως είναι γνωστό το hit rate ισούται με το ποσοστό των αιτήσεων προσπέλασης αρχείων που βρίσκονταν αποθηκευμένα στη μνήμη cache επί του συνολικού αριθμού των αιτήσεων. Η συγκεκριμένη μετρική αποτελεί το γενικότερο κριτήριο αξιολόγησης της απόδοσης μιας πολιτικής αντικατάστασης αρχείων.

Η μετρική maximum improvement ορίζεται ως εξής

$$Maximum\ Improvement = 1 - \%Compulsory\ Misses - \%Hit\ Ratio \quad (5.1)$$

Η συγκεκριμένη μετρική μας προσφέρει ένα κριτήριο αξιολόγησης, το οποίο αγνοεί τον αριθμό των υποχρεωτικών αστοχιών. Η μετρική αυτή βρίσκει ενδιαφέρουσα εφαρμογή σε ένα σύστημα, όπως ο CASTOR, στο οποίο εισέρχονται συνεχώς νέα πειραματικά δεδομένα, τα οποία δεν έχουν προσπελαστεί στο παρελθόν. Επεκτείνοντας την παραπάνω μετρική μπορούμε να πάρουμε και ένα μέτρο σύγκρισης με την πολιτική LRU. Έχουμε τον επόμενο τύπο:

$$MI_{LRU} = \frac{1 - \%Compulsory\ Misses - \%Hit\ Ratio}{\%LRU\ Hit\ Ratio} \quad (5.2)$$

Όμως το ποσοστό επιτυχίας μιας μνήμης από μόνο του δε περιγράφει ικανοποιητικά την απόδοση μιας μνήμης που τα αντικείμενα έχουν διαφορετικό μέγεθος. Είναι λοιπόν αναγκαία η χρήση μιας νέας μετρικής που ονομάζεται byte hit rate. Η μετρική αυτή ορίζεται από τον επόμενο τύπο

$$Byte\ Hit\ Rate = \frac{Συνολικό\ μέγεθος\ των\ αιτήσεων\ που\ εξυπηρετήθηκαν\ από\ την\ Cache}{Συνολικό\ μέγεθος\ όλων\ των\ αιτήσεων\ που\ εξυπηρετήθηκαν} \quad (5.3)$$

Η παραπάνω μετρική μας δίνει μια εικόνα για το κόστος εξυπηρέτησης των αιτήσεων που δέχεται το σύστημα.

5.3 Εκτέλεση πειραμάτων και Αποτελέσματα

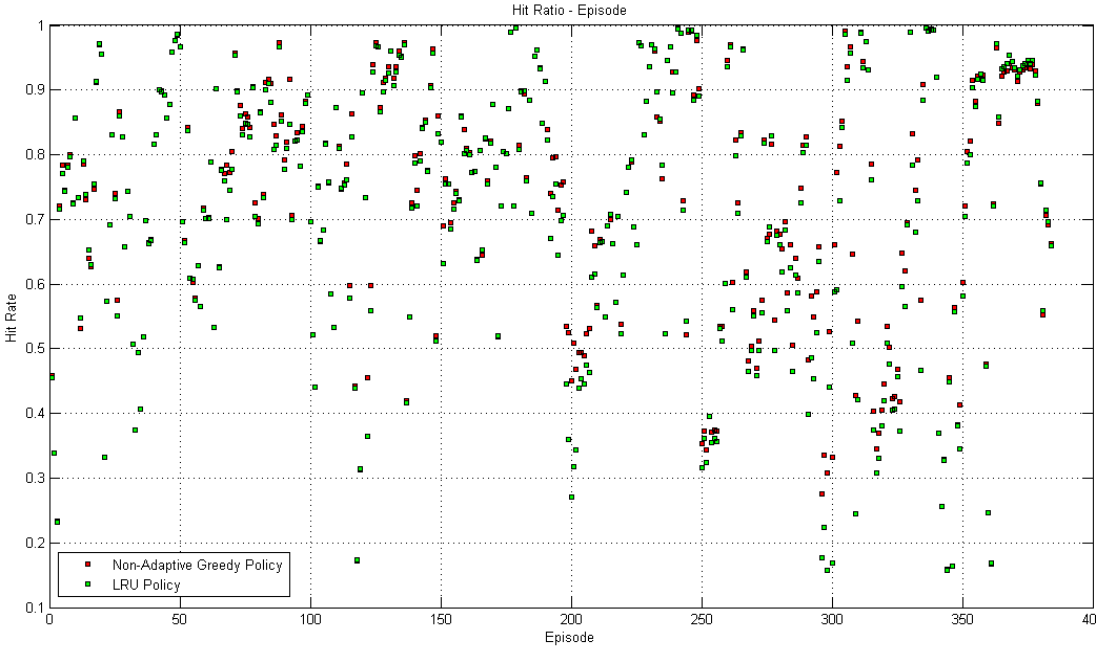
Πριν παρουσιάσουμε τα αποτελέσματα των πειραμάτων που εκτελέσαμε πρέπει να επισημάνουμε ότι τα πειράματα που έχουμε εκτελέσει μέχρι στιγμής αποτελούν μια ένδειξη για την εφαρμογή των μεθόδων αντικατάστασης αρχείων που προτείνουμε. Ως την τρέχουσα χρονική στιγμή έχουμε εκτελέσει ορισμένα πειράματα για να αξιολογήσουμε τη δεύτερη από τις δύο προσαρμοστικές πολιτικές που προτείνουμε. Συγκεκριμένα εκτελέσαμε ορισμένα πειράματα χρησιμοποιώντας τη νέα στατική πολιτική που προτείνεται στην ενότητα 4.8.2, η οποία αποτελεί τη βάση της προσαρμοστικής πολιτικής που συνδυάζει τις μεθόδους της ενισχυτικής μάθησης και της προσομοιωμένης ανόπτησης. Αφού αξιολογήσαμε τη νέα στατική πολιτική εκτελέσαμε ορισμένα πειράματα χρησιμοποιώντας την ολοκληρωμένη προσαρμοστική μέθοδο RL – SA.

Στο σημείο αυτό πρέπει να τονίσουμε ότι για μια πλήρη αξιολόγηση των νέων μεθόδων είναι αναγκαία η εκτέλεση πειραμάτων για την αξιολόγηση της μεθόδου stochastic Caching και επιπλέον η εκτέλεση πειραμάτων για περισσότερες επαναλήψεις της βασικής διαδικασίας μάθησης. Κατόπιν αφού πάρουμε ασφαλέστερα συμπεράσματα για την απόδοση των νέων μεθόδων κρίνεται απαραίτητη η εφαρμογή τους στο πραγματικό σύστημα CASTOR, για εξαγωγή ακριβέστερων μετρήσεων. Συνεχίζουμε με την παρουσίαση των αποτελεσμάτων.

5.3.1 Νέα Άπληστη Στατική Πολιτική Αντικατάστασης Αρχείων

Αρχικά εκτελέσαμε ορισμένα πειράματα για να αξιολογήσουμε την νέα στατική πολιτική που προτείνεται στην ενότητα 4.8.2. Όπως αναφέρθηκε σε προηγούμενο κεφάλαιο η νέα αυτή πολιτική χρησιμοποιείται ως βάση για την προσαρμοστική πολιτική που χρησιμοποιούμε σε συνδυασμό με τη μέθοδο της προσομοιωμένης ανόπτησης. Στα σχήματα που ακολουθούν συγκρίνουμε την απόδοση της Greedy πολιτικής με την πολιτική LRU, η οποία δίνει τα καλύτερα αποτελέσματα από όλες τις πολιτικές που είναι ήδη ενσωματωμένες στο σύστημα CASTOR.

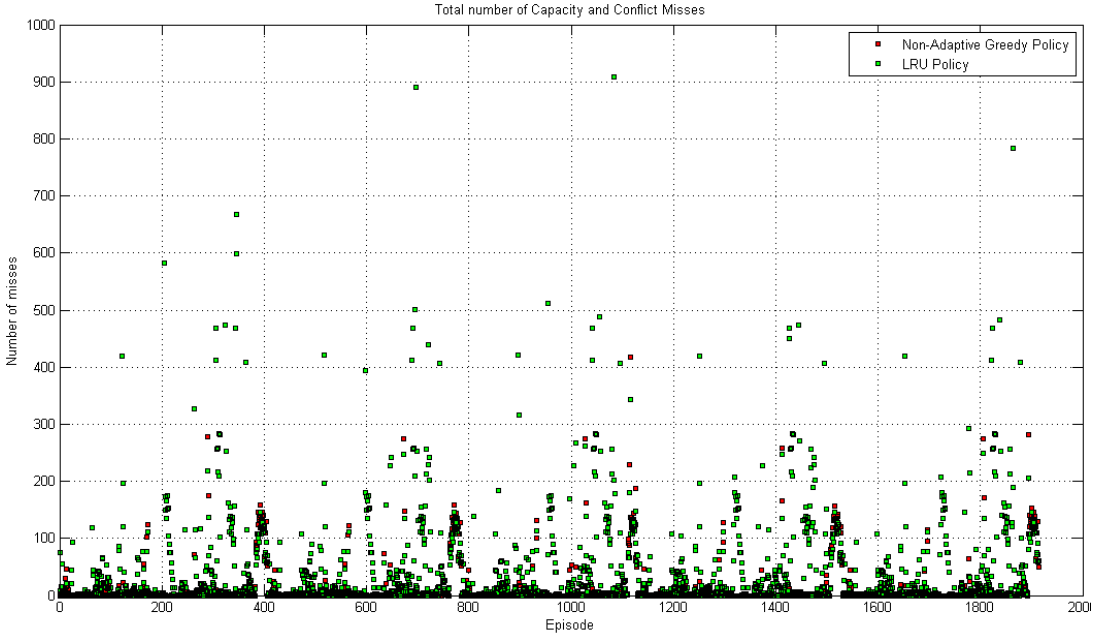
Στο σχήμα 5.1 βλέπουμε με πράσινο το ρυθμό των ευστοχιών της άπληστης στατικής πολιτικής που προτείνουμε και με κόκκινο της πολιτικής LRU για περίπου 400 επεισόδια. Κάθε επεισόδιο περιέχει τις αιτήσεις των χρηστών για ένα χρονικό παράθυρο διάρκειας μισής ώρας. Στα επεισόδια περιέχονται από 3000 μέχρι 20000 αιτήσεις. Παρατηρούμε ότι τα κόκκινα σημεία στο σχήμα βρίσκονται στις περισσότερες περιπτώσεις πάνω από τα πράσινα πράγμα που δείχνει ότι η νέα άπληστη πολιτική που προτείνεται παρουσιάζει μεγαλύτερο ρυθμό ευστοχιών.



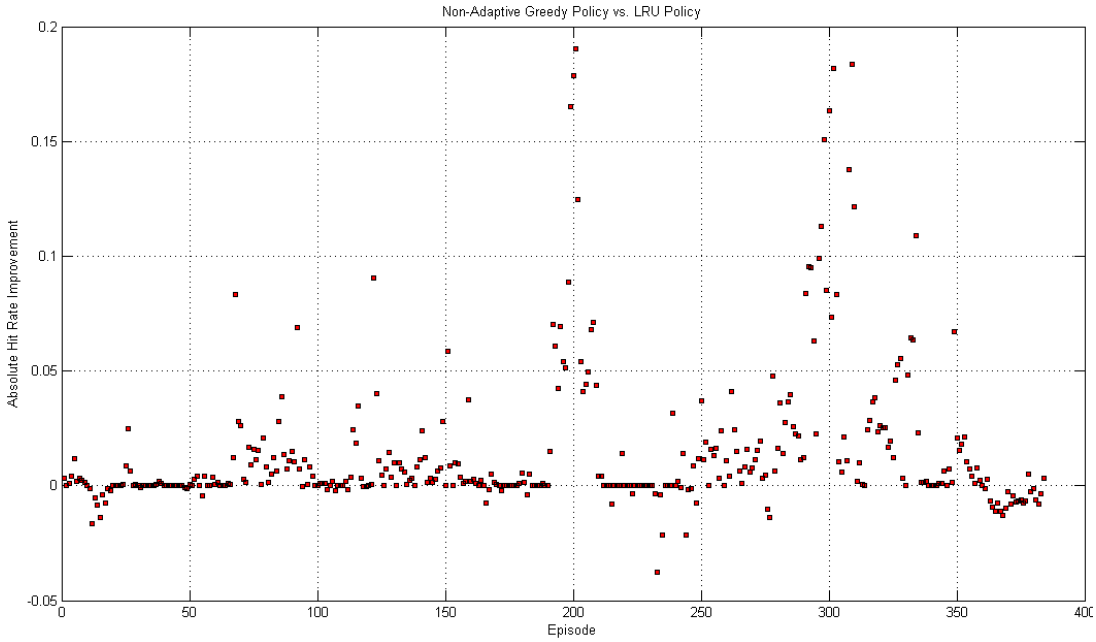
Σχήμα 5.1: Στο σχήμα αυτό παρουσιάζεται το ποσοστό ευστοχίας για τις πολιτικές LRU και Greedy. Με πράσινο απεικονίζεται η LRU ενώ με κόκκινο η Greedy. Παρουσιάζονται τα αποτελέσματα ~400 επεισοδίων προσομοίωσης διάρκειας μισής ώρας το καθένα.

Στο σχήμα 5.2 βλέπουμε το πλήθος των μη υποχρεωτικών αστοχιών της μνήμης για τη νέα στατική πολιτική που προτείνεται (κόκκινο χρώμα) και για την πολιτική LRU (πράσινο χρώμα). Βλέπουμε ότι η νέα στατική πολιτική παρουσιάζει σημαντικά μικρότερο αριθμό μη υποχρεωτικών αστοχιών. Υπενθυμίζουμε ότι μια υποχρεωτική αστοχία προκύπτει από ένα αρχείο που εισέρχεται πρώτη φορά στο σύστημα, π.χ. για ένα νέο σύνολο πειραματικών δεδομένων. Το παραπάνω επιβεβαιώνεται και από τα σχήματα 5.3 και 5.4, όπου φαίνεται η απόλυτη και ποσοστιαία αύξηση αντίστοιχα του hit rate για τη νέα στατική πολιτική αντικατάστασης έναντι του hit rate της LRU.

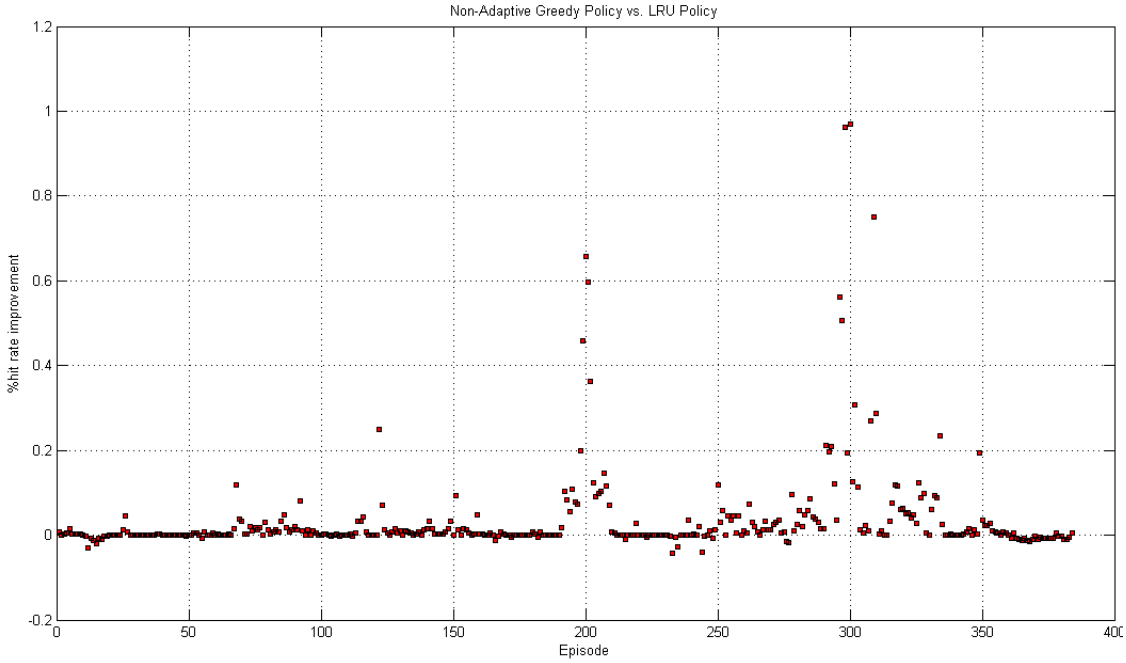
.



Σχήμα 5.2: Στο σχήμα αυτό παρουσιάζεται ο αριθμός των non-Compulsory misses για τις πολιτικές LRU και Greedy.Παρουσιάζονται τα αποτελέσματα ~2000 επεισοδίων προσομοίωσης διάρκειας μισής ώρας το καθένα.



Σχήμα 5.3: Στο σχήμα αυτό παρουσιάζεται η αύξηση του hit rate της Greedy πολιτικής έναντι της πολιτικής LRU. Παρουσιάζονται τα αποτελέσματα ~400 επεισοδίων προσομοίωσης διάρκειας μισής ώρας το καθένα.



Σχήμα 5.4: Στο σχήμα αυτό παρουσιάζεται η ποσοστιαία αύξηση του hit rate της Greedy πολιτικής έναντι της πολιτικής LRU. Παρουσιάζονται τα αποτελέσματα ~400 επεισοδίων προσομοίωσης διάρκειας μισής ώρας το καθένα.

5.3.1.1 Συμπεράσματα

Από τα παραπάνω διαγράμματα προκύπτει το συμπέρασμα ότι η νέα άπληστη πολιτική που προτείνεται δίνει από μόνη της καλύτερα συμπεράσματα από την πολιτική LRU και κατά συνέπεια από όλες τις υπόλοιπες πολιτικές που είναι ενσωματωμένες στο σύστημα CASTOR. Η βελτιωμένη απόδοση ήταν αναμενόμενη καθώς η νέα πολιτική λαμβάνει υπόψη της εκτός από τη συχνότητα προσπέλασης των αρχείων, το μέγεθος των αρχείων αλλά και το φόρτο εργασίας του δικτύου και του συστήματος. Αξίζει να υπενθυμίσουμε ότι βασίζεται στην 2-OPT άπληστη προσεγγιστική επίλυση του προβλήματος Knapsack, μια εκδοχή του οποίου αποτελεί και το πρόβλημα Caching με αρχεία διαφορετικού μεγέθους.

Όπως φαίνεται στο σχήμα 5.2 αρκετές φορές των πλήθος των μη – υποχρεωτικών αστοχιών της μνήμης όταν ακολουθείται η πολιτική LRU είναι αυξημένο και υπερβαίνει τις 400 πράγμα το οποίο δεν ισχύει για τη νέα προσαρμοστική πολιτική όπου ο αριθμός των μη – υποχρεωτικών αστοχιών είναι ιδιαίτερα μειωμένος. Σε αυτή τη διαφορά οφείλονται τα αυξημένα ποσοστά βελτίωσης του ρυθμού ευστοχιών (βλ. σχήμα 5.4), τα οποία αρκετές φορές υπερβαίνουν το 20%. Μάλιστα η μέγιστη αύξηση ξεπερνά το 90% για δύο επεισόδια αιτήσεων μισής ώρας.

Μπορούμε λοιπόν με ασφάλεια να πούμε ότι η νέα στατική πολιτική παρουσιάζει πολύ καλύτερη απόδοση από τις ήδη υπάρχουσες πολιτικές. Λόγω της απλότητάς της είναι ιδιαίτερα πιθανή η χρήση της στο πραγματικό σύστημα CASTOR. Συνδυάζοντας τη στατική πολιτική με ενισχυτική μάθηση περιμένουμε περαιτέρω βελτίωση των αποτελεσμάτων

5.3.2 Προσαρμοστική Πολιτική Αντικατάστασης Αρχείων – Συνδυάζοντας την Ενισχυτική Μάθηση με την Προσομοιωμένη Ανόπτηση

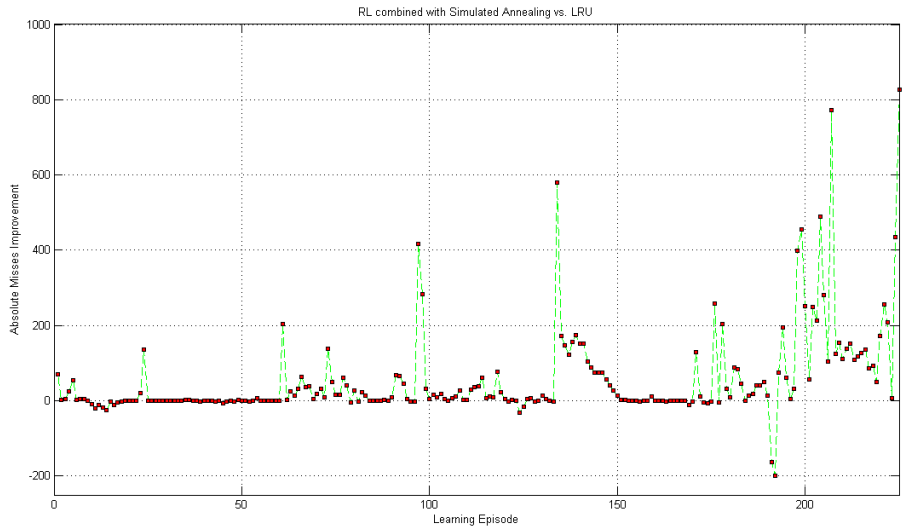
Έχοντας ως βάση μια στατική πολιτική, η οποία δίνει ήδη καλύτερα αποτελέσματα από την πολιτική LRU, προσπαθούμε να βελτιώσουμε την απόδοση του συστήματος ακόμα περισσότερο συνδυάζοντας τη νέα πολιτική με τις μεθόδους της ενισχυτικής μάθησης και της προσομοιωμένης ανόπτησης. Συνδυάζοντας τα παραπάνω έχουμε τη μέθοδο της παραγράφου 4.8.4. Στη συνέχεια παραθέτουμε ορισμένα αποτελέσματα από τα πειράματα που έχουμε εκτελέσει.

Στον πίνακα (πίνακας 5.1) που ακολουθεί φαίνεται ο αριθμός των ευστοχιών και αστοχιών της μνήμης για τις 4 πολιτικές αντικατάστασης αρχείων που είναι υλοποιημένες στο σύστημά μας. Υπενθυμίζουμε ότι έχουμε αποθηκεύσει τη ροή των εισερχόμενων αιτήσεων για ένα πραγματικό μήνα και αναπαράγουμε αυτή κατά τη διάρκεια μιας πλήρους εποχής προσομοίωσης μάθησης. Επίσης η χρονική διάρκεια προσομοίωσης μιας εποχής ανέρχεται στις 2 μέρες. Πρέπει να επισημάνουμε ότι για να εισάγουμε στοχαστικό θόρυβο στο σύστημα, οι αιτήσεις που ανήκουν στο ίδιο λεπτό ταξινομούνται με τυχαίο τρόπο πριν εισέλθουν στον προσομοιωτή.

Περισσότερος θόρυβος εισέρχεται στο σύστημά μας θεωρώντας ότι χρόνος αναμονής εξυπηρέτησης ενός αρχείου είναι στοχαστική μεταβλητή και δειγματοληπτείται από μια γνωστή γκαουσιανή κατανομή. Για τις στατικές πολιτικές αντικατάστασης η μικροδιαφορές στους μετρητές προκύπτουν από το γεγονός ότι οι αιτήσεις των χρηστών εισέρχονται στο σύστημά μας με στοχαστικό τρόπο. Παρά το θόρυβο παρατηρούμε ότι για την πολιτική αντικατάστασης που χρησιμοποιεί ενισχυτική μάθηση ο αριθμός των ευστοχιών της μνήμης συνεχώς αυξάνεται.

Epoch	*GC Policy*	*Hit Counter*	*Miss Counter*	*Compulsory Misses*
1	LRU	626788	262322	245079
1	FIFO	563924	325186	245079
1	RL_SA	638692	250418	245079
1	Default	621031	268079	245079
2	LRU	626848	262262	245079
2	FIFO	563999	325111	245079
2	RL_SA	639843	249267	245079
2	Default	621066	268044	245079
3	LRU	626826	262284	245079
3	FIFO	563928	325182	245079
3	RL_SA	639856	249254	245079
3	Default	620941	268169	245079
4	LRU	626857	262253	245079
4	FIFO	563779	325331	245079
4	RL_SA	639910	249200	245079
4	Default	621046	268064	245079

Πίνακας 5.1: Στον Πίνακα αυτό φαίνονται οι επιτυχίες και οι αστοχίες της μνήμης για 4 πλήρεις εποχές μάθησης και όλες τις διαθέσιμες πολιτικές αντικατάστασης του συστήματος CASTOR. Παρατηρούμε τη σταθερότητα και τη συνεχόμενη αύξηση της νέας προσαρμοστικής πολιτικής.



Σχήμα 5.5: Στο σχήμα αυτό παρουσιάζεται η διαφορά στον αριθμό των αστοχιών ανάμεσα στην προσαρμοστική πολιτική που προτείνουμε (RL combined with SA) και της πολιτικής LRU. Παρουσιάζονται τα αποτελέσματα ~250 επεισοδίων μάθηση

5.3.2.1 Συμπεράσματα

Από τα πειραματικά δεδομένα που φαίνονται παραπάνω βλέπουμε ότι η νέα προσαρμοστική πολιτική που προτείνεται παρουσιάζει καλύτερα ποσοστά ευστοχίας από την πολιτική LRU και τις υπόλοιπες πολιτικές αντικατάστασης αρχείων. Μάλιστα στον πίνακα 1 βλέπουμε ότι ο αριθμός των ευστοχιών της μνήμης αυξάνεται με την πάροδο των εποχών μάθησης. Παρατηρώντας την 4η εποχή βλέπουμε ότι σε ένα σύνολο 889110 αιτήσεων έχουμε μόνο 4121 μη υποχρεωτικές αστοχίες. Αντίστοιχα η πολιτική LRU παρουσιάζει 17174 μη υποχρεωτικές αστοχίες. Τα παραπάνω μας δίνουν μια μείωση των μη υποχρεωτικών αστοχιών έναντι της LRU περίπου 76%. Ενδεικτικά αναφέρεται ότι η αντίστοιχη μείωση την πρώτη εποχή μάθησης ήταν περίπου 69%.

Η συνεχής βελτίωση του ρυθμού ευστοχίας επιβεβαιώνεται και από το σχήμα 5.5. Στο σχήμα αυτό βλέπουμε ένα παράθυρο μιας εποχής της διαδικασίας μάθησης, το οποίο αποτελείται από ~250 επεισόδια μάθησης. Βλέπουμε ότι με το χρόνο το πλήθος των επεισοδίων στα οποία παρατηρούμε θετικό κέρδος αυξάνεται. Επίσης αυξάνονται και οι τιμές του κέρδους. Εδώ ως κέρδος ορίζουμε το πλήθος των non-compulsory αστοχιών της πολιτικής LRU μείον το πλήθος των non-compulsory αστοχιών της νέας πολιτικής.

Βέβαια πρέπει να επισημάνουμε ότι τα παραπάνω δεδομένα μας δείχνουν μια χαμηλή ταχύτητα βελτίωσης της απόδοσης του αλγορίθμου, πράγμα το οποίο μπορεί να βελτιωθεί με κατάλληλη ρύθμιση των παραμέτρων μάθησης. Είναι λοιπόν αναγκαία η περαιτέρω μελέτη των προτεινόμενων αλγορίθμων.

6

Επίλογος

6.1 Σύνοψη και συμπεράσματα

Η εφαρμογή προσαρμοστικών πολιτικών αντικατάστασης αρχείων σε ιεραρχικά συστήματα αποθήκευσης είναι ένα πολύ δύσκολο πρόβλημα και η υπάρχουσα έρευνα έχει να επιδείξει διαφορετικές προσεγγίσεις και αλγορίθμους για τη λύση του. Κατά τη διάρκεια της έρευνάς μας παρατηρήσαμε ότι οι αλγόριθμοι που προτείνονται είναι άμεσα συνδεδεμένοι με τα συστήματα στα οποία εφαρμόζονται και βασίζονται κυρίως σε ευριστικές και όχι αναλυτικές μεθόδους επίλυσης.

Για παράδειγμα έχει μελετηθεί σε μεγάλο βαθμό το γενικευμένο πρόβλημα caching σε Web Servers και έχουν προταθεί διάφορες βέλτιστες ή υποβέλτιστες λύσεις [5][19][22] για αυτό. Οι Web Servers μπορούν να μοντελοποιηθούν ως ιεραρχικά συστήματα αποθήκευσης. Μια σημαντική διαφορά με τα ιεραρχικά συστήματα αποθήκευσης, τα οποία μελετάμε στην παρούσα εργασία είναι ότι εδώ η προσπέλαση των αρχείων είναι δυνατή μόνο από το υψηλότερο επίπεδο της μνήμης σε αντίθεση με τις προηγούμενες εργασίες, όπου η προσπέλαση των αρχείων είναι δυνατή είτε από την cache του Web Server είτε από το σκληρό δίσκο. Το παραπάνω παρέχει την ελευθερία της μη – υποχρεωτικής διαγραφής αρχείων από την cache [19].

Το κύριο αποτέλεσμα της εργασίας αυτής είναι ότι έδειξε πως το πρόβλημα αντικατάστασης αρχείων μπορεί να μοντελοποιηθεί με σαφή τρόπο ως πρόβλημα ενισχυτικής μάθησης και

προσαρμοστικές πολιτικές αντικατάστασης αρχείων μπορούν να ενσωματωθούν σε συστήματα κλίμακας Petabytes. Επίσης κατά τη διάρκεια της έρευνάς μας προτείναμε μια νέα στατική πολιτική αντικατάστασης, η οποία δε λαμβάνει υπόψη της μόνο τη συχνότητας προσπέλασης των αρχείων στη μνήμη αλλά και το μέγεθός τους καθώς και το φόρτο του δικτύου. Η συγκεκριμένη πολιτική παρουσιάζει ιδιαίτερα ενδιαφέροντα αποτελέσματα σε σχέση με στατικές πολιτικές, όπως η LRU και χρησιμοποιείται ως βάση για μια από τις προσαρμοστικές πολιτικές που προτείνουμε.

Αρχικά παρουσιάσαμε μια βιβλιογραφική μελέτη σχετική με τα Ιεραρχικά Συστήματα Αποθήκευσης αρχείων και το πεδίο της Ενισχυτικής Μάθησης. Στη συνέχεια δείξαμε πως το γενικευμένο πρόβλημα caching μπορεί να μοντελοποιηθεί ως πρόβλημα αποφάσεων Markov και πώς με κατάλληλη τροποποίηση των χώρων καταστάσεων και δράσεων μπορούν χρησιμοποιηθούν αποδοτικά μέθοδοι ενισχυτικής μάθησης για την επίλυσή του. Τέλος παρουσιάσαμε δύο τρόπους εφαρμογής αλγορίθμων ενισχυτικής μάθησης για την επίλυσή του, τους οποίους ενσωματώσαμε και εξετάσαμε σε μια προσομοίωση ενός ιεραρχικού συστήματος μεγάλης κλίμακας.

Από την παραπάνω μελέτη διαπιστώνουμε ότι είναι δυνατή η εφαρμογή προσαρμοστικών αλγορίθμων σε ιεραρχικά συστήματα αποθήκευσης μεγάλης κλίμακας υπό ορισμένες προϋποθέσεις. Πριν την εφαρμογή κάποιας προσαρμοστικής πολιτικής αρχείων είναι απαραίτητη η τροποποίηση των γενικευμένων μεθόδων που παρουσιάζονται και η εξειδίκευσή τους για το σύστημα στο οποίο θα εφαρμοστούν. Είναι λοιπόν αναγκαία η τροποποίηση παραμέτρων, όπως ο ρυθμός μάθησης των αλγορίθμων, αλλά και βασικών συναρτήσεων που χρησιμοποιούν οι μέθοδοι, όπως η συνάρτηση επιβράβευσης.

Επίσης καταλήγουμε στο γεγονός ότι είναι προτιμότερη η off-line εκπαίδευση των πρακτόρων μάθησης και η μετέπειτα μεταφορά τους (αφού έχουν μάθει τις κατάλληλες συναρτήσεις τιμών) στο πραγματικό σύστημα. Η άμεση ενσωμάτωσή τους στο πραγματικό σύστημα και η on-line εκπαίδευσή τους μπορεί να οδηγήσουν σε ανεπιθύμητες συμπεριφορές του συστήματος με ιδιαίτερα μεγάλο κόστος. Ας σκεφτούμε για παράδειγμα τη διαγραφή από το δίσκο ενός συνόλου αρχείων με πειραματικά δεδομένα, το οποίο δεν έχει αντιγραφεί – αποθηκευτεί σε τριτογενή μέσα.

Επίσης ιδιαίτερη προσοχή χρειάζεται στο γεγονός ότι η πολυπλοκότητα των συγκεκριμένων αλγορίθμων μπορεί να οδηγήσει σε μειωμένη απόδοση του συστήματος λόγω πρόχειρης σχεδίασης κατά τη διάρκεια ενσωμάτωσης τους. Για όλους τους παραπάνω λόγους, φαίνεται να προτιμάτε σε εμπορικά συστήματα η χρήση στατικών πολιτικών αντικατάστασης αρχείων, όπως η LRU, και ευριστικών μεθόδων που εκτός από χρονικές παραμέτρους λαμβάνουν υπόψη τους και άλλες παραμέτρους όπως το μέγεθος του αρχείου.

Βέβαια από τα πειράματα που εκτελέσαμε μπορούμε να πούμε με ασφάλεια ότι κακές αποφάσεις διαγραφής μπορούν να αποφευχθούν χρησιμοποιώντας τη δεύτερη προσέγγιση μηχανικής μάθησης που προτείνεται στην παρούσα εργασία. Η χρήση μιας ιδιαίτερα καλής λύσης ως βάση για τον αλγόριθμο μάθησης επιτρέπει στον πράκτορα να αποφεύγει ιδιαίτερα άσχημες δράσεις και να πετυχαίνει ιδιαίτερα μειωμένο αριθμό μη υποχρεωτικών αστοχιών έναντι των στατικών πολιτικών. Φυσικά το κόστος υπολογισμού ενός καλού συνόλου αρχείων προς διαγραφή παραμένει αυξημένο σε σχέση με μια στατική πολιτική. Παρόλα αυτά το ποσοστό βελτίωσης είναι τέτοιο που αξίζει να ερευνηθεί μια βελτιστοποιημένη ενσωμάτωση των παραπάνω αλγορίθμων σε πραγματικά συστήματα.

6.2 Μελλοντικές επεκτάσεις

Η προφανής μελλοντική επέκταση της παρούσας εργασίας είναι η υποβολή των παραπάνω μεθόδων σε διεξοδικές δοκιμές με χρήση του προσομοιωτή που αναπτύχθηκε, για μεγαλύτερο όγκο αιτήσεων των χρηστών. Το παραπάνω δεν αποσκοπεί μόνο στην παραγωγή ακριβέστερων αποτελεσμάτων για την απόδοση των προσαρμοστικών πολιτικών αλλά και στην εκπαίδευση των πρακτόρων μάθησης. Ιδιαίτερα ενδιαφέρουσα θα ήταν η ενσωμάτωση και υλοποίηση των παραπάνω μεθόδων στο πραγματικό σύστημα CASTOR.

Ιδιαίτερο ενδιαφέρον παρουσιάζει η αξιολόγηση της προτεινόμενης μοντελοποίησης του χώρου καταστάσεων και δράσεων σε άλλα πραγματικά ιεραρχικά συστήματα αποθήκευσης αρχείων, η αξιολόγηση της απόδοση των μεθόδων σε αυτά καθώς και η αξιολόγηση των χαρακτηριστικών σύγκλισης των προτεινόμενων μεθόδων, όπως η ταχύτητα σύγκλισης και η απόκλιση από μια βέλτιστη λύση.

7

Βιβλιογραφία

[1] A. Silberschatz, P. B. Galvin, G. Gagne, *Operating System Concepts*, John Wiley & Sons, 2008

[2] U. Hahn, W. Dilling, D. Kaletta, *Improved Adaptive Replacement Algorithm for Disk Caches in HSM Systems*, 16th IEEE Symposium on Mass Storage Systems, 1999

[3] I.L. traiger, J. gesei, D. R. Slutz. *Evaluation Techniques for Storage Hierrarchies*, IBM Systems Journal archive. Volume 9 , Issue 2 Pages: 78-117, June 1970

[4] *CASTOR official web site* http://castor.web.cern.ch/castor/

[5] B. Sonah, M.B. Ito, *New Asaptive Object Replacement Policy for Video-On-Demand Systems*, 6th IEEE International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunications Systems, 1998

[6] Y. Xu, C. Xing, L. Zhou, *A cache replacement algorithm in hierarchical storage of continuous media object*, 5th International Conference WAIAM, 2004

[7] M. Aguilera, K. Keeton, A. Merchant, K. K. M. Reddy, M. Uysal, *Improving recoverability in Multi-tier Storage Systems*, 37th Annual IEEE/IFIP International Conference on Dependable Systems and Networks, 2007

[8] D. Vengerov, *A reinforcement learning framework for online data migration in hierarchical storage systems*, The Journal of Supercomputing, Volume 43,Issue 1, Pages 1-19, 2008

[9] R. Sutton, A. Barto, *Reinforcement Learning: An Introduction*, MIT Press Cambridge, 1998

[10] D. Bertsekas, J. Tsitsiklis, *Neuro-Dynamic Programming*, Athena Scientific, 1996

[11] D. Bertsekas, *Dynamic Programming and Optimal Control Vol.1*, Athena Scientific, 2005

[12] M.D. Lee, S. Zhang, M. Munro, M. Steyvers, *Using Heuristic Models to Understand Human and Optimal Decision-Making on Bandit Problems*, In Proceedings of the Ninth International Conference on Cognitive Modeling, 2009

[13] S. Thrun, *Efficient Exploration In Reinforcement Learning*, Technical Report CMU-CS-92-102, CS Department Carnegie Mellon University, 1992

[14] R. Sutton, H. R. Maei, D. Precup, S. Bhatnagar, D. Silver, C. Szepesvari, E. Wiewiora, *Fast Gradient – Descent methods for temporal-difference learning with linear function approximation*, In Proceedings of the 26th Annual International Conference on Machine Learning, 2009

[15] D. Precup, R. Sutton, *Eligibility Traces for Off-policy Policy Evaluation*, In Proceedings of the 17th International Conference on Machine Learning, 2000

[16] J. N. Tsitsiklis, B. Van Roy, *An analysis of temporal-difference learning with function approximation*, IEEE Transactions on Automatic Control, vol. 42, pp. 674-690, 1997

[17] S. H. Khayat, *On Optimal Replacement of Nonuniform Cache Objects*, IEEE Transactions on Computers, vol. 49, pp. 769-778, 2000

[18] S. H. Khayat, *Optimal Solution of Off-line and On-line Generalized Caching*, Technical report WUCS-96-20, 1996

[19] O. Bahat, A. M. Makowski, *Optimal Replacement Policies for Non-Uniform Cache Objects With Optional Eviction,* IEEE Infocom, 2003

[20] A. Liu, J. Wang, G. Han, S. Wang, J. Wen, *Improved Simulated Annealing Algorithm Solving for 0/1 Knapsack Problem*, 6th International Conference on Intelligent Systems Design and Application, 2006

[21] C. Rickert, *Benchmarking a Simulated Annealing Approximation Algorithm for the 0-1 Knapsack*

[22] N. Niclausse, Z. Liu, P. Nain, *A new Efficient Caching Policy for the World Wide Web*, Workshop on Internet Server Preformance, 1998

[23] L. Cherkasove, *Improving WWW Proxies Performance with Greedy-Dual-Size-Frequency Caching Policy,* HP Laboratories Technical Report HPL, 1998

[24] R. Gramacy, M. Warmuth, S. A. Brandt, I. Ari, *Adaptive Caching by Refetching,* Advances in Neural Information Processing Systems 15, MIT Press, 2005

[25] T. Stroesslin, W. Gernster, *Reinforcement Learning in Continuous State and Action Space,* Artificial Neural Network and Neural Information Processing, Joint International Conference ICANN/ICONIP, 2003

[26] H. van Hasselt, M.A. Wiering, *Reinforcement Learning in Continuous Action Spaces*, IEEE International Symposium on Approximate Dynamic Programming and Reinforcement Learning, 2007

[27] L. Busoniu, R. Babuska, B. de Schutter, D. Ernst, *Reinforcement Learning and Dynamic Programming Using Function Approximators,* CRC Press, 2010

[28] M. Lagoudakis, R. Parr, *Least-Squares Policy Iteration,* Journal of Machine Learning Research, 2003

[29] H. R. Maei, R. Sutton, *GQ(λ): A general gradient algorithm for temporal-difference prediction learning with eligibility traces,* The Third Conference on Artificial General Intelligence, 2010

[30] K. Amling, *Replacing Eligibility Trace for action-value Learning with Function Approximation, In Proceedings of the 15th European Symposium on Artificial Neural Networks,* 2007

[31] A. Kleywegt, J. Papastavrou, *The Dynamic and Stochastic Knapsack Problem*, Operations Research, volume 46, issue 1, Pages 17-35, 1998

[32] B. Dean, M. Goemans, J. Vondrak, Approximating the Stochastic Knapsack Problem: The Benefit of Adaptivity, 45th Annual IEEE Symposium on Foundations of Computer Science, 2004

[33] J. Csirik, J.B. G. Frenk, M. Labbe, S. Zhang, Heuristics for the 0-1 min-Knapsack problem, Acta Cybernetica Volume 10, Issue 1-2, Pages 15-20, 1991

[34] X. Han, K. Makino, Online Minimization Knapsack Problem, Approximation and Online Algorithms, Springerlink, 2010

[35] Κ. Τζούμας, Βελτιστοποίηση Ερωτημάτων με Eddies και Ενισχυτική Μάθηση, Εθνικό Μετσόβιο Πολυτεχνείο, 2007

[36] T. Mitchell, Machine Learning, Mc Graw Hill, 1997

[37] B. Reed, D. Long, *Analysis of Caching Algorithms for Distributed File Systems*, ACM SIGOPS Operating Systems Review, Vol. 30, pp. 12-21, 1996

[38] D. Bertsekas, *Dynamic Programming and Optimal Control Vol.2*, Athena Scientific, 2005

[40] M. Grzes, D. Kudenko, *Robustness Analysis of SARSA(λ): Different Models of Reward and Initialization*, AIMSA 2008

[41] R. Sutton, D. McAllester, S. Singh, Y. Mansour, *Policy Gradient Methods for Reinforcement Learning with Function Approximation*, Advances in Neural Information Processing Systems 12, pp. 1057-1063, MIT Press, 2000

[42] P. Dayan, T. Sejnowski, *TD(λ) Converges with Probability 1*, Machine Learning, 14, pp. 295-301, 1994

[43] R. Jacobs, *Increased Rates of Convergence Through Learning Rate Adaptation*, Neural Networks, Vol. 1, pp. 295-307, 1988