Εθνικό Μετσόβιο Πολυτεχνείο
Σχολή Ηλεκτρολόγων Μηχανικών
και Μηχανικών Υπολογιστών

Τομέας Τεχνολογίας Πληροφορικής
και Υπολογίστων

Αξιοποίηση τεχνικών ενισχυτικής μάθησης για την παροχή υπηρεσίας εγγυημένης ποιότητας και την αποδοτικότερη χρήση των πόρων πολυπύρηνων επεξεργαστών

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

ΙΩΑΝΝΑ Κ. ΒΑΒΕΛΙΔΟΥ

Επιβλέπων : Νεκτάριος Κοζύρης
Καθηγητής Ε.Μ.Π.

Αθήνα, Ιούλιος 2021

Εθνικό Μετσόβιο Πολυτεχνείο
Σχολή Ηλεκτρολόγων Μηχανικών
και Μηχανικών Υπολογιστών

Τομέας Τεχνολογίας Πληροφορικής
και Υπολογίστων

Αξιοποίηση τεχνικών ενισχυτικής μάθησης για την παροχή υπηρεσίας εγγυημένης ποιότητας και την αποδοτικότερη χρήση των πόρων πολυπύρηνων επεξεργαστών

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

ΙΩΑΝΝΑ Κ. ΒΑΒΕΛΙΔΟΥ

Επιβλέπων : Νεκτάριος Κοζύρης
Καθηγητής Ε.Μ.Π.

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 26η Ιουλίου 2021.

Νεκτάριος Κοζύρης Καθηγητής Ε.Μ.Π.	Διονύσιος Πνευματικάτος Καθηγητής Ε.Μ.Π.	Γεώργιος Γκούμας Αν. Καθηγητής Ε.Μ.Π.

Αθήνα, Ιούλιος 2021

Ιωάννα Κ. Βαβελίδου

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

Περίληψη

Όλο και περισσότερες από τις υπηρεσίες που φιλοξενούνται στα σύγχρονα κέντρα δεδομένων αποτελούν υπηρεσίες κρίσιμης απόκρισης. Η ανάγκη να διασφαλιστούν οι αυστηροί στόχοι ποιότητας υπηρεσίας των εφαρμογών αυτών οδηγεί συχνά τα κέντρα δεδομένων στην απαγόρευση της συνεκτέλεσής τους με άλλες εφαρμογές, ώστε να αποφευχθεί ο ανταγωνισμός στους κοινόχρηστους πόρους, όπως είναι η κρυφή μνήμη τελευταίου επιπέδου. Η πρακτική αυτή συνεπάγεται τη χαμηλή χρησιμοποίηση των εξυπηρετητών και, κατ' επέκταση, την αύξηση του συνολικού κόστους ιδιοκτησίας των κέντρων δεδομένων. Για την αντιμετώπιση του προβλήματος, έχουν πλέον ενσωματωθεί στους σύγχρονους επεξεργαστές τεχνολογίες που υποστηρίζουν τόσο την παρακολούθηση της χρήσης όσο και τον διαμοιρασμό των κοινόχρηστων πόρων. Με τον τρόπο αυτό, δίνεται η δυνατότητα ασφαλούς συνεκτέλεσης των εφαρμογών κρίσιμης απόκρισης με εφαρμογές βέλτιστης προσπάθειας. Στην παρούσα διπλωματική εργασία, αξιοποιούμε τις εν λόγω τεχνολογίες καθώς και τεχνικές βαθιάς ενισχυτικής μάθησης προκειμένου να διαμοιράσουμε την κρυφή μνήμη τελευταίου επιπέδου ενός πολυπύρηνου συστήματος με τρόπο που θα εξασφαλίζει αφενός τη διατήρηση της ποιότητας της κρίσιμης υπηρεσίας και αφετέρου τη μεγαλύτερη, κατά το δυνατόν, αξιοποίηση του εξυπηρετητή από τις εφαρμογές βέλτιστης προσπάθειας. Για την αξιολόγηση του μηχανισμού, εκτελούμε συνεκτελέσεις των υπηρεσιών κρίσιμης απόκρισης Xapian και Img-dnn της σουίτας Tailbench με ένα πλήθος από εφαρμογές βέλτιστης προσπάθειας. Αποδεικνύουμε πως η επίδοση της κρίσιμης υπηρεσίας διαφυλάσσεται σε ικανοποιητικό βαθμό, ενώ ταυτόχρονα αξιοποιούνται ευκαιρίες αύξησης της επίδοσης των εφαρμογών χαμηλής προτεραιότητας.

Λέξεις κλειδιά

Διαμοιρασμός Κοινόχρηστης Κρυφής Μνήμης, LLC, Ποιότητα Υπηρεσίας, Χρησιμοποίηση Πόρων Επεξεργαστή, Πολυεπεξεργαστικά Συστήματα, Συνεκτέλεση Εφαρμογών, Intel Cache-Allocation, Intel Cache-Monitoring, Tailbench, Βαθιά Ενισχυτική Μάθηση, Νευρωνικά Δίκτυα

Abstract

An increasingly large number of the services hosted in contemporary data centers are latency critical services. In order to ensure that the stringent Quality of Service targets of these applications are met, data centers often disallow their simultaneous execution with other applications, so that contention in the shared resources, such as the last level cache, is prevented. The disadvantage of this policy is low server utilization and, subsequently, increased total cost of ownership for the data centers. To tackle this problem, modern processors have introduced technologies that support the monitoring as well as the partitioning of common resources, thus allowing the safe colocation of latency critical and best effort applications. In this thesis, we utilize these technologies along with deep reinforcement learning techniques in order to dynamically partition the last level cache of a multi-core system in a way that maintains the performance of the latency critical application and, at the same time, achieves high server utilization. To evaluate the mechanism, we colocate the latency critical applications Xapian and Img-dnn of the Tailbench suite with a variety of best effort applications. We prove that the Quality of Service of the latency critical application is adequately protected, while opportunities to increase the performance of the best effort applications are exploited.

Key words

Shared Cache Partitioning, LLC, Quality of Service, Server Resource Utilization, Multiprocessors, Colocation, Intel Cache-Allocation, Intel Cache-Monitoring, Tailbench, Deep Reinforcement Learning, Neural Networks

Ευχαριστίες

Θα ήθελα να ευχαριστήσω θερμά τον επιβλέποντα καθηγητή αυτής της διπλωματικής, κ. Νεκτάριο Κοζύρη, για την ευκαιρία που μου έδωσε να ασχοληθώ με το συγκεκριμένο θέμα, το ερέθισμα που μου προσέφερε να γνωρίσω τον τομέα των υπολογιστικών συστημάτων, καθώς και για το ενδιαφέρον που μου καλλιέργησε κατά τη διάρκεια των σπουδών μου.

Ιδιαίτερες ευχαριστίες θα ήθελα να δώσω στον ερευνητή κ. Κωνσταντίνο Νίκα για την καθοδήγησή του και τη διαρκή και άμεση στήριξή του. Τόσο οι τεχνικές όσο και οι θεωρητικές συζητήσεις μας πάνω στο θέμα της διπλωματικής αυτής αλλά και στον ευρύτερο τομέα της αρχιτεκτονικής υπολογιστών μού δίδαξαν πολλά και ενίσχυσαν το ενδιαφέρον μου να συνεχίσω τις σπουδές μου σε διδακτορικό επίπεδο.

Ευχαριστώ, ακόμα, τα μέλη της εξεταστικής επιτροπής, τον κ. Διονύσιο Πνευματικάτο και τον κ. Γεώργιο Γκούμα, όχι μόνο ως εξεταστές, αλλά και για όσα μου προσέφεραν ως καθηγητές μου.

Έπειτα, θέλω να ευχαριστήσω τους φίλους και κοντινούς μου ανθρώπους οι οποίοι ομόρφυναν τα φοιτητικά μου χρόνια. Ευχαριστώ ιδιαίτερα τον Αλέξανδρο, που υπήρξε ο καλύτερος συνεργάτης και μεγάλο στήριγμά μου.

Τέλος, το μεγαλύτερο ευχαριστώ οφείλω στην οικογένεια μου, και κυρίως τους γονείς και την αδελφή μου, για την ανιδιοτελή στήριξη και αγάπη τους όλα αυτά τα χρόνια.

Η παρούσα διπλωματική, όμως, αφιερώνεται στη θεία μου, Βούλα, και τη γιαγιά μου, Πίτσα, οι οποίες δεν πρόλαβαν να με δουν να ολοκληρώνω τις σπουδές μου, όμως είμαι σίγουρη ότι θα ήταν απερίγραπτα περήφανες για εμένα και τα μελλοντικά βήματά μου.

Ιωάννα Κ. Βαβελίδου,

Αθήνα, 26η Ιουλίου 2021

Περιεχόμενα

Κατάλογος πινάκων

Κατάλογος σχημάτων

Κεφάλαιο 1

Εισαγωγή

1.1 Κίνητρο

Ιδιωτικά και δημόσια πλαίσια νέφους (cloud frameworks) επιτρέπουν τη φιλοξενία ενός αυξημένου αριθμού από φορτία εργασίας (workloads) σε κέντρα δεδομένων μεγάλης κλίμακας με δεκάδες χιλιάδες εξυπηρετητές (servers). Από το συνολικό κόστος ιδιοκτησίας (Total Cost of Ownership, TCO) των σύγχρονων κέντρων δεδομένων, το μεγαλύτερο ποσοστό, τυπικά 50-70%, καταλαμβάνουν οι εξυπηρετητές. Συνεπώς, η μεγιστοποίηση της χρησιμοποίησής τους είναι καθοριστικής σημασίας για τη δυνατότητα συνεχούς κλιμάκωσης των κέντρων δεδομένων. Στην πραγματικότητα, όμως, η μέση χρησιμοποίηση των εξυπηρετητών στα περισσότερα κέντρα δεδομένων είναι πολύ χαμηλή, μεταξύ 10% και 50%. Το κόστος της υποχρησιμοποίησης αυτής μπορεί να είναι ιδιαίτερα σημαντικό. Για παράδειγμα, οι εξυπηρετητές διαδικτυακής αναζήτησης (websearch) της Google βρίσκονται, κατά μέσο όρο, σε κατάσταση απραξίας (idleness) το 30% της ημέρας. Υποθετικά, για μια συστάδα (cluster) 10,000 εξυπηρετητών, η απραξία αυτή ισοδυναμεί με 3,000 αναξιοποίητους εξυπηρετητές [15].

Σταδιακά, όλο και περισσότερες εφαρμογές στα κέντρα δεδομένων ανήκουν στην κατηγορία υπηρεσιών κρίσιμης απόκρισης (latency-critical, LC, services), δηλαδή υπηρεσιών που θέτουν αυστηρές εγγυήσεις ποιότητας υπηρεσίας (quality of service, QoS) ως προς τη λανθάνουσα καθυστέρησή (tail latency) τους. Παραδείγματα τέτοιων εφαρμογών είναι τα κοινωνικά δίκτυα, οι μηχανές αναζήτησης, το λογισμικό ως υπηρεσία (software-as-a-service), οι διαδικτυακοί χάρτες, το ηλεκτρονικό ταχυδρομείο κ.α. Το φορτίο των LC εφαρμογών μεταβάλλεται σημαντικά στον χρόνο λόγω των ημερήσιων μοτίβων και των απρόβλεπτων αιχμών στις προσβάσεις των χρηστών. Προκειμένου να διατηρηθούν οι QoS στόχοι των υπηρεσιών αυτών, συνηθίζεται να προδιαγράφονται για αυτές πόροι που επαρκούν για την περίπτωση του χειρότερου σεναρίου. Ως αποτέλεσμα, σε διαστήματα χαμηλής κίνησης, μεγάλο μέρος των πόρων αυτών μένει αναξιοποίητο.

Μια υποσχόμενη τακτική για την αξιοποίηση των τυχόν αναξιοποίητων πόρων είναι η εκτέλεση εφαρμογών βέλτιστης προσπάθειας (best-effort, BE, ή batch tasks) στους ίδιους εξυπηρετητές με τις υπηρεσίες κρίσιμης απόκρισης. Η κυρίαρχη πρόκληση αυτής της προσέγγισης είναι οι παρεμβολές (interference) των συνεκτελούμενων φορτίων εργασίας στους μοιραζόμενους πόρους, όπως είναι οι κρυφές και μη μνήμες, τα κανάλια E/E και οι σύνδεσμοι δικτύου (network links). Ακόμα και μικρές παρεμβολές είναι ικανές να προκαλέσουν σημαντικές παραβιάσεις του QoS των LC υπηρεσιών, οι οποίες μπορεί να προκαλέσουν στα κέντρα δεδομένων απώλειες εκατομμυρίων

δολαρίων [8].

Ως λύση για την αποτροπή των παρεμβολών και τη διατήρηση της ποιότητας των LC υπηρεσιών, η πιο συντηρητική προσέγγιση είναι απλώς η απαγόρευση κοινής χρήσης πόρων μεταξύ LC και άλλων εφαρμογών [14]. Με αυτόν τον τρόπο διατηρείται το QoS των LC εφαρμογών, όμως η χρησιμοποίηση των πόρων του συστήματος παραμένει χαμηλή. Η δεύτερη προσέγγιση στοχεύει στην αποφυγή της συνεκτέλεσης εφαρμογών που είναι πιθανόν να εμφανίσουν παρεμβολές κατά τη συνεκτέλεσή τους [3]. Ενώ η τεχνική αυτή βελτιώνει, σε σχέση με την προηγούμενη, την αξιοποίηση των πόρων, είναι περιοριστική ως προς τις επιλογές των εφαρμογών που μπορούν να συνεκτελεστούν. Επιπλέον, απαιτεί την εξαγωγή πληροφοριών που μπορεί να μην είναι διαθέσιμες κατά την υποβολή των εργασιών στα δημόσια νέφη από τους χρήστες. Η τρίτη προσέγγιση επικεντρώνεται στην εξάλειψη των παρεμβολών μέσω του διαμερισμού πόρων (resource partitioning) μεταξύ των συνεκτελούμενων εφαρμογών, κάνοντας χρήση τεχνικών απομόνωσης σε επίπεδο λειτουργικού ή υλικού. Έτσι, προστατεύεται το QoS της LC υπηρεσίας, ενώ ταυτόχρονα οι υπηρεσίες βέλτιστης προσπάθειας επωφελούνται από την αξιοποίηση των αχρησιμοποίητων πόρων.

1.2 Συνεισφορά Εργασίας

Στα πλαίσια της παρούσας διπλωματικής εργασίας, μελετάται η συνεκτέλεση υπηρεσιών κρίσιμης απόκρισης με διαφορετικές διεργασίες βέλτιστης προσπάθειας. Στόχος μας είναι ο κατάλληλος δυναμικός διαμοιρασμός της Last-Level-Cache (LLC), ώστε να διασφαλίζεται η ποιότητα της εφαρμογής κρίσιμης απόκρισης και, ταυτόχρονα, να αξιοποιείται κατά το μέγιστο δυνατόν ο επεξεργαστής, μέσω της παράλληλης εκτέλεσης των εφαρμογών βέλτιστης προσπάθειας.

Αξιοποιώντας και επεκτείνοντας προηγούμενη εργασία του εργαστηρίου, κάνουμε χρήση μεθόδων βαθιάς ενισχυτικής μάθησης για την αυτοματοποιημένη λήψη αποφάσεων ως προς τον διαμοιρασμό της LLC και, ως εκ τούτου, τον έλεγχο του ανταγωνισμού μεταξύ των διαφορετικών κατηγοριών εφαρμογών. Ο πράκτορας ενισχυτικής μάθησης συλλέγει εμπειρία και προσπαθεί να την αξιοποιήσει για τον εντοπισμό της βέλτιστης ανάθεσης των ways της LLC για κάθε διαφορετικό επίπεδο ανταγωνισμού που συναντά.

Για τη λήψη των αποφάσεών του, ο πράκτορας στηρίζεται σε μετρικές που συλλέγονται με χρήση της τεχνολογίας Cache Monitoring Technology (CMT) που παρέχεται από την Intel, ενώ η εκάστοτε απόφαση εφαρμόζεται στο σύστημα με χρήση της τεχνολογίας Cache Allocation Technology (CAT) της Intel.

Αξιολογούμε την επίδοση διαφορετικών εκδοχών του πράκτορα στις εφαρμογές κρίσιμης απόκρισης Xapian και Img-dnn της σουίτας Tailbench, συνεκτελούμενες με ένα πλήθος από διαφορετικές εφαρμογές βέλτιστης προσπάθειας.

Κεφάλαιο 2

Υπόβαθρο

2.1 Εισαγωγή

Όπως έχουμε επισημάνει, η συνεκτέλεση εφαρμογών στοχεύει στην καλύτερη αξιοποίηση των πόρων του συστήματος, όμως δημιουργεί ανταγωνισμό για τους κοινόχρηστους πόρους και κατ' επέκταση βλάπτει την ποιότητα υπηρεσίας των LC εφαρμογών. Προκύπτει, λοιπόν, η ανάγκη ελέγχου του ανταγωνισμού αυτού μέσω του κατάλληλου διαμοιρασμού των κοινόχρηστων πόρων.

Είναι γεγονός πως υπάρχουν μηχανισμοί λογισμικού που μπορούν να απομονώσουν την εκτέλεση των διεργασιών όσον αφορά συγκεκριμένους πόρους, όπως είναι οι πυρήνες του επεξεργαστή, το μέγεθος της κύριας μνήμης και η χρήση του δικτύου. Παρ' όλα αυτά, υπάρχουν και πόροι, όπως είναι η κρυφή μνήμη τελευταίου επιπέδου, που δεν μπορούν να διαμοιραστούν με τέτοιους μηχανισμούς.

Η επίδραση της κρυφής μνήμης τελευταίου επιπέδου είναι καθοριστικής σημασίας για την ποιότητα και την επίδοση των εκτελούμενων εφαρμογών. Μάλιστα, ο ανταγωνισμός για την LLC μπορεί να οδηγήσει ακόμα και σε κορεσμό του bandwidth της κύριας μνήμης, πόρος που επίσης δεν μπορεί να διαμοιραστεί με κάποιον μηχανισμό λογισμικού. Συνεπώς, η δυνατότητα παρακολούθησης και διαμοιρασμού της LLC είναι απαραίτητη. Σε αυτή την κατεύθυνση έχει συμβάλει η ανάπτυξη της τεχνολογίας RDT από την Intel, η οποία έχει υποστήριξη από το υλικό και ενσωματώνεται στους νέους επεξεργαστές της Intel.

2.2 Intel RDT

Οι νέοι επεξεργαστές της οικογένειας Intel Xeon E5 v4 υποστηρίζουν την τεχνολογία Intel RDT (Resource Director Technology), η οποία επιτρέπει την επίβλεψη και διαχείριση κοινόχρηστων πόρων του επεξεργαστή. Σε αυτήν περιλαμβάνονται οι επιμέρους τεχνολογίες Cache Monitoring Technology (CMT), Memory Bandwidth Monitoring (MBM), Cache Allocation Technology (CAT) και Memory Bandwidth Allocation (MBA).

Η τεχνολογία CMT επιτρέπει στο λειτουργικό σύστημα ή σε οποιονδήποτε διαχειριστή του συστήματος μνήμης να επιβλέπει τη χρήση της LLC από τις εφαρμογές που τρέχουν στο σύστημα. Αντίστοιχα, η τεχνολογία MBM επιτρέπει την επίβλεψη του DRAM bandwidth ανά εφαρμογή,

πυρήνα ή και ομάδα αυτών. Η τεχνολογία CAT δίνει τη δυνατότητα σε οποιονδήποτε διαχειριστή του συστήματος να καθορίσει το τμήμα της LLC που μπορεί να χρησιμοποιήσει κάθε εφαρμογή, πυρήνας ή/και ομάδα αυτών. Τέλος, η MBA επιτρέπει τη ρύθμιση του ρυθμού αιτήσεων για δεδομένα από τη μνήμη ανά φυσικό πυρήνα.

Οι τεχνολογίες CMT-CAT που χρησιμοποιούμε στην παρούσα διπλωματική υποστηρίζονται από το υλικό με την προσθήκη καταχωρητών και με τη χρήση επιπέδων αφαίρεσης (abstraction layers) που συσχετίζουν τους λογικούς πυρήνες (hardware threads) με τις εκτελούμενες εφαρμογές. Για την τεχνολογία CAT, υπάρχει ένα επίπεδο αφαίρεσης μεταξύ διεργασιών (software threads) και λογικών πυρήνων (hardware threads), το Class of Service (COS). Ο επεξεργαστής διαθέτει ένα πλήθος διαφορετικών COS, το καθένα από τα οποία αποτελεί ένα είδος bitmask σε επίπεδο ways και προσδιορίζει το τμήμα της cache στο οποίο οι πυρήνες ή οι διεργασίες έχουν δυνατότητα εγγραφής νέων blocks. Αντίθετα, η ανάγνωση νέων blocks επιτρέπεται από το σύνολο της cache. Συνεπώς, για την ανάθεση ενός τμήματος LLC, προσδιορίζεται αρχικά το επιθυμητό bitmask για κάποιο COS και στη συνέχεια αυτό αντιστοιχίζεται με την ομάδα πυρήνων ή διεργασιών που επιθυμούμε.

2.3 Σχετικές Εργασίες

Στην ενότητα αυτή περιγράφουμε κάποιους από τους μηχανισμούς διαχείρισης του ανταγωνισμού στους κοινόχρηστους πόρους του επεξεργαστή που συναντήσαμε στη βιβλιογραφία.

2.3.1 Διαχείριση ανταγωνισμού με χρήση ευριστικών κανόνων

Στη μελέτη των K. Nikas *et al.* (2019) [20] αναπτύσσεται ο Dicer, ένας μηχανισμός δυναμικού διαμοιρασμού της κρυφής μνήμης που έχει στόχο την ικανοποίηση των αναγκών μιας εφαρμογής υψηλής προτεραιότητας και την ταυτόχρονη υψηλή χρησιμοποίηση του server, μέσω της παράλληλης συνεκτέλεσης, σε διαφορετικούς πυρήνες, άλλων εφαρμογών χαμηλής προτεραιότητας. Ο μηχανισμός αυτός παρακολουθεί hardware μετρικές των εφαρμογών και, βασιζόμενος σε ένα σύνολο ευριστικών κανόνων, δημιουργεί ένα μοντέλο απόδοσης (performance model) για την εκτίμηση της απόδοσης των συνεκτελούμενων εφαρμογών και τη λήψη περιοδικών αποφάσεων για τα ways που θα αναθέσει στην κάθε ομάδα. Η συλλογή των hardware μετρικών και η εφαρμογή των αποφάσεων γίνεται με χρήση της τεχνολογίας RDT. Καθώς ο μηχανισμός αυτός στηρίζεται μόνο σε hardware μετρικές και όχι σε software μετρικές, όπως είναι το tail latency, που αποτυπώνουν την τελική εμπειρία του χρήστη, δεν εγγυάται τη διατήρηση του QoS της εφαρμογής υψηλής προτεραιότητας.

Στην εργασία των Lo *et al.* (2015) [15] επιχειρείται ο ταυτόχρονος διαμοιρασμός όλων των κοινόχρηστων πόρων κατά τη συνεκτέλεση των εφαρμογών. Συγκεκριμένα, διαπιστώνεται ότι η αιτία που οδηγεί σε κακή ποιότητα υπηρεσίας είναι ο κορεσμός οποιουδήποτε από τους μοιραζόμενους πόρους και, συνεπώς, ο μηχανισμός Heracles που υλοποιείται στοχεύει στην αποτροπή του κορεσμού αυτού. Η αρχιτεκτονική του περιλαμβάνει έναν ελεγκτή που επιβλέπει κάθε διαμοιραζόμενο πόρο, καθώς επίσης και έναν κύριο ελεγκτή που λαμβάνει αποφάσεις για τη συνεκτέλεση

με βάση, και πάλι, κάποιους ευριστικούς κανόνες. Πιο αναλυτικά, ο κύριος ελεγκτής επιβλέπει την καθυστέρηση και το φορτίο της LC εφαρμογής και ανάλογα απαγορεύει ή επιτρέπει την συνεκτέλεση των BE εφαρμογών. Επιπλέον, είναι υπεύθυνος να ενημερώσει τους υπο-ελεγκτές για το αν θα πρέπει να παραχωρήσουν περισσότερους πόρους στις BE εργασίες ή να προστατεύσουν την LC εφαρμογή. Στις περιπτώσεις που παρατηρείται αύξηση του φορτίου ή παραβίαση του QoS, ο ελεγκτής επιβάλλει τη διακοπή των υπόλοιπων συνεκτελούμενων εφαρμογών και ενημερώνει τους υπο-ελεγκτές να παραχωρήσουν όλους τους πόρους στη LC εφαρμογή. Στη συνέχεια, όταν ομαλοποιηθεί η εκτέλεσή της, ενεργοποιεί ξανά τις BE εργασίες. Σε αντίθεση με την προηγούμενη μελέτη που βασίζεται στη χρήση του IPC (instructions per cycle), εδώ η λήψη των αποφάσεων στηρίζεται και στη software μετρική tail latency της υπηρεσίας κρίσιμης απόκρισης. Η προσέγγιση αυτή έχει το πλεονέκτημα ότι το tail latency είναι το πλέον αντιπροσωπευτικό μέγεθος για να ποσοτικοποιήσει την ποιότητα της εφαρμογής, ενώ αντίθετα δεν υπάρχει πάντα σαφής συσχέτιση μεταξύ του IPC και του tail latency για τις υπηρεσίες κρίσιμης απόκρισης. Από την άλλη, ένα πιθανό μειονέκτημα είναι η απώλεια γενικότητας, αφού για κάποιες υπηρεσίες κρίσιμης απόκρισης πιθανόν να απαιτείται ειδική μέριμνα για την εξαγωγή του μεγέθους αυτού.

Η πρόσφατη μελέτη PARTIES των Chen *et al.* (2019) [1] διαφοροποιείται ως προς το ότι στοχεύει στη διατήρηση των latency στόχων πολλαπλών συνεκτελούμενων LC υπηρεσιών. Ομοίως με τον Heracles, η δουλειά αυτή δε βασίζεται στην κατασκευή μοντέλου επίδοσης, αλλά, αντίθετα, προσαρμόζει τον διαμοιρασμό των πόρων με άπληστο (greedy) τρόπο, με βάση την κατάσταση του συστήματος σε πραγματικό χρόνο. Πιο συγκεκριμένα, αυξάνει ή μειώνει σταδιακά έναν πόρο, π.χ. πλήθος πυρήνων, χωρητικότητα μνήμης κ.λπ., για μία LC εργασία τη φορά και έπειτα αξιολογεί την παρατηρούμενη απόδοση. Αυτή η διαδικασία συνεχίζεται έως ότου να ικανοποιηθεί το QoS όλων των εργασιών LC, αν αυτό είναι εφικτό, και στη συνέχεια οι εναπομείναντες πόροι ανατίθενται στις BE εργασίες.

2.3.2 Χρήση τεχνικών Μηχανικής Μάθησης στον τομέα των Συστημάτων Υπολογιστών

Οι παραπάνω μελέτες χρησιμοποιούν κλασικές ευριστικές μεθόδους για τη διαχείριση του ανταγωνισμού των πόρων μεταξύ των συνεκτελούμενων εφαρμογών. Σε άλλες πρόσφατες εργασίες, όμως, αξιοποιούνται τεχνικές μηχανικής μάθησης, οι οποίες γνώρισαν αλματώδη ανάπτυξη τα τελευταία χρόνια.

Η αναβίωση της μηχανικής μάθησης από τα τέλη της δεκαετίας του 1990 έχει καταστεί δυνατή χάρη σε σημαντικές προόδους στην υπολογιστική ικανότητα των επεξεργαστών και την ανάπτυξη των big data [19]. Η ικανότητα των αλγορίθμων μηχανικής μάθησης να εντοπίζουν πολύπλοκα μοτίβα σε δεδομένα που είναι εξαιρετικά δύσκολο να αναλυθούν με το χέρι βοηθάει στην παραγωγή αποτελεσματικών μοντέλων πρόβλεψης. Κάποιοι τομείς που επωφελήθηκαν περισσότερο από αυτές τις νέες τεχνικές είναι η όραση υπολογιστών και η επεξεργασία φυσικής γλώσσας [11], το gaming [17] και τα συστήματα συστάσεων [13]. Ενώ οι αρχιτέκτονες υπολογιστών συνέβαλαν στη ραγδαία βελτίωση της απόδοσης των τεχνικών μηχανικής μάθησης μέσω της υλοποίησης αποδοτικών GPU και της χρήσης προσαρμοσμένου υλικού (custom hardware), η αντίστροφη συνεισφορά, δηλαδή η αξιοποίηση των αλγορίθμων μηχανικής μάθησης για τη

βελτίωση της απόδοσης των συστημάτων υπολογιστών, υπήρξε πολύ πιο περιορισμένη. Παρ' όλα αυτά, οι σχετικά πρόσφατες εργασίες που επιστρατεύουν τις τεχνικές αυτές στον τομέα των συστημάτων έχουν παραγάγει ιδιαίτερα ελπιδοφόρα αποτελέσματα. Μεταξύ άλλων, έχουν χρησιμοποιηθεί τεχνικές μηχανικής μάθησης στους τομείς του branch prediction [5], του cache replacement [26], του prefetching [22], του performance predictability [4] κ.α.

2.3.3 Διαχείριση ανταγωνισμού με χρήση μεθόδων μηχανικής μάθησης

Ως προς τον τομέα της διαχείρισης του ανταγωνισμού στους πόρους του επεξεργαστή, κάποιες χαρακτηριστικές μελέτες που αντικαθιστούν τους ευριστικούς κανόνες με τεχνικές μηχανικής μάθησης είναι αυτές που ακολουθούν.

Στην εργασία των Wang *et al.* [32] γίνεται χρήση νευρωνικών δικτύων για τον προσδιορισμό του πόρου που αποτελεί, κάθε στιγμή, τη βασική αιτία παρεμβολών, με σκοπό να διασφαλιστεί το QoS της LC υπηρεσίας. Επιπλέον, γίνεται προσωρινή χρήση εμπειρικών κανόνων κατά την online εκπαίδευση του νευρωνικού, ώστε να αποφευχθεί η ανάγκη για συλλογή δεδομένων και offline εκπαίδευση.

Οι Y. Li *et al.* [12] χρησιμοποιούν έναν πράκτορα ενισχυτικής μάθησης ο οποίος λαμβάνει ως είσοδο hardware και software μετρικές για να επιλέξει την τιμή της τάσης και της συχνότητας των πυρήνων που αντιστοιχούν στις εφαρμογές βέλτιστης προσπάθειας. Προσπαθούν, λοιπόν, να αντιμετωπίσουν το πρόβλημα του ανταγωνισμού στη μοιραζόμενη κρυφή μνήμη τελευταίου επιπέδου όχι με άμεσο τρόπο, δηλαδή διαμοιράζοντας τα ways της LLC, αλλά μέσω του Dynamic Voltage and Frequency Scaling (DVFS). Ως προς την υλοποίηση του πράκτορα, χρησιμοποιείται ο αλγόριθμος Q-learning με χρήση Q-table.

Οι R. Nishtala *et al.* [21] από την άλλη, χρησιμοποιούν βαθιά ενισχυτική μάθηση, δηλαδή αντικαθιστούν το Q-table με νευρωνικά δίκτυα. Διερευνούν, ακόμα, το πώς μπορούν να προσεγγίσουν το tail latency αποκλειστικά με hardware μετρικές. Μία επιπλέον διαφορά της μελέτης αυτής από τις προηγούμενες είναι ότι στοχεύουν στον διαμοιρασμό των πόρων μεταξύ πολλών υπηρεσιών κρίσιμης απόκρισης.

Κεφάλαιο 3

Θεωρία Ενισχυτικής Μάθησης

3.1 Κατηγορίες Μηχανικής Μάθησης

Ο κλάδος της μηχανικής μάθησης εμφανίζει τα τελευταία χρόνια ραγδαία ανάπτυξη. Το γεγονός αυτό οφείλεται κυρίως στις προσπάθειες των ερευνητών να δημιουργήσουν αλγορίθμους που βελτιώνουν από μόνοι τους την απόδοσή τους με βάση την εμπειρία που αποκτούν.

Σύμφωνα με τον ορισμό του Tom M. Mitchell [16], ένα υπολογιστικό πρόγραμμα λέγεται ότι μαθαίνει από μια εμπειρία E, ως προς ένα σύνολο εργασιών T και ένα μέτρο απόδοσης P, εάν η απόδοσή του σε εργασίες του T, όπως αυτή μετριέται από το P, βελτιώνεται με την E. Επομένως, προκειμένου να περιγραφεί πλήρως ένα πρόβλημα μηχανικής μάθησης, θα πρέπει αρχικά να οριστεί η επιθυμητή εργασία που καλείται να εκτελέσει ο αλγόριθμος μάθησης. Επιπλέον, θα πρέπει να καθοριστεί ένα μέτρο απόδοσης, βάσει του οποίου αξιολογείται η ικανότητα του αλγορίθμου να εκτελεί ορθά τη συγκεκριμένη εργασία. Τέλος, πρέπει να προσδιοριστεί ο τρόπος με τον οποίον αποκτάται η εμπειρία E, χάρη στην οποία η απόδοση του αλγορίθμου βελτιώνεται με την πάροδο του χρόνου.

Ο τρόπος με τον οποίον ο αλγόριθμος μηχανικής μάθησης αποκτά την απαραίτητη εμπειρία καθορίζει το είδος της μάθησης. Τα τρία βασικότερα είδη μηχανικής μάθησης είναι:

- επιβλεπόμενη μάθηση (supervised learning)
- μη επιβλεπόμενη μάθηση (unsupervised learning)
- ενισχυτική μάθηση (reinforcement learning)

3.1.1 Επιβλεπόμενη και μη επιβλεπόμενη μάθηση

Οι αλγόριθμοι επιβλεπόμενης μάθησης (supervised learning) δέχονται ως είσοδο ένα σύνολο δεδομένων που περιλαμβάνει ορισμένα παραδείγματα μαζί με τις αντίστοιχες τιμές-στόχους. Στην περίπτωση αυτή, το υπό εκπαίδευση μοντέλο επιχειρεί να μάθει τη σχέση που συνδέει κάθε παράδειγμα με την αντίστοιχη τιμή-στόχο. Με άλλα λόγια, το μοντέλο προσπαθεί να βρει μια συνάρτηση η οποία, με είσοδο οποιοδήποτε διάνυσμα εισόδου, θα παράγει μια έξοδο όπως αυτή των δεδομένων εκπαίδευσης.

Αντίθετα, οι αλγόριθμοι μη επιβλεπόμενης μάθησης (unsupervised learning) δέχονται ως είσοδο ένα σύνολο δεδομένων που περιλαμβάνει ορισμένα παραδείγματα, χωρίς όμως αυτά να αντιστοιχίζονται σε κάποιες τιμές-στόχους. Στην περίπτωση αυτή, το υπό εκπαίδευση μοντέλο επιχειρεί να εξαγάγει τις απαραίτητες πληροφορίες αποκλειστικά από τα πρότυπα (patterns) των παραδειγμάτων εισόδου. Συνεπώς, όπως και στην επιβλεπόμενη μάθηση, η εμπειρία που αποκτούν οι αλγόριθμοι μη επιβλεπόμενης μάθησης βασίζεται σε ένα συγκεκριμένο σύνολο δεδομένων. Ωστόσο, στην επιβλεπόμενη μάθηση το σύνολο αυτό είναι επισημειωμένο, ενώ στη μη επιβλεπόμενη μάθηση κάτι τέτοιο δεν ισχύει.

Σε αντίθεση με τα δύο προαναφερθέντα είδη αλγορίθμων μηχανικής μάθησης, ορισμένοι αλγόριθμοι δε βασίζονται σε κάποιο συγκεκριμένο σύνολο δεδομένων προκειμένου να αποκτήσουν εμπειρία και κατ' επέκταση να βελτιώσουν την απόδοσή τους. Ένα τέτοιο παράδειγμα αποτελούν οι αλγόριθμοι ενισχυτικής μάθησης, οι οποίοι αναλύονται στη συνέχεια.

3.1.2 Ενισχυτική Μάθηση

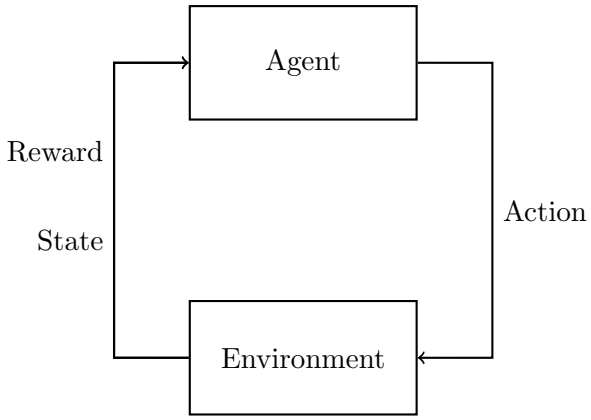
Οι αλγόριθμοι ενισχυτικής μάθησης αλληλεπιδρούν με ένα περιβάλλον και μαθαίνουν μέσω της ανάδρασης. Σε αντίθεση με την επιβλεπόμενη και τη μη επιβλεπόμενη μάθηση, στις οποίες δίνεται ως είσοδος ένα συγκεκριμένο σύνολο δεδομένων εκπαίδευσης, στην περίπτωση της ενισχυτικής μάθησης, ένας πράκτορας που λειτουργεί σε ένα περιβάλλον προσπαθεί να επιτύχει έναν συγκεκριμένο στόχο εκτελώντας ορισμένες δράσεις. Μέσω της ανάδρασης με το περιβάλλον, ο πράκτορας προσπαθεί σταδιακά να βελτιώσει την απόδοσή του ως προς τον στόχο. Στην περίπτωση, λοιπόν, της ενισχυτικής μάθησης, το υπό εκπαίδευση μοντέλο επιχειρεί να μάθει τη βέλτιστη αντιστοίχιση καταστάσεων σε δράσεις, με στόχο τη μεγιστοποίηση ενός αριθμητικού μεγέθους ανταμοιβής. Με άλλα λόγια, το μοντέλο προσπαθεί να εντοπίσει τις δράσεις εκείνες που αποδίδουν κάθε φορά τη μεγαλύτερη ανταμοιβή. Επομένως, η εμπειρία που αποκτούν οι αλγόριθμοι ενισχυτικής μάθησης βασίζεται στη διαδικασία αλληλεπίδρασης του πράκτορα με το περιβάλλον του.

3.2 Βασικές Έννοιες Ενισχυτικής Μάθησης

Στην περίπτωση της ενισχυτικής μάθησης, ένας πράκτορας (agent) αλληλεπιδρά με ένα περιβάλλον (environment) πραγματοποιώντας συγκεκριμένες δράσεις (actions). Κάθε τέτοια δράση έχει ως αποτέλεσμα τη μεταβολή της κατάστασης (state) του περιβάλλοντος και την παροχή μιας ανταμοιβής (reward) στον πράκτορα. Μέσω της παραπάνω διαδικασίας, ο πράκτορας επιδιώκει να αποκτήσει εμπειρία, προκειμένου να επιλέγει κάθε φορά τις κατάλληλες δράσεις που θα τον οδηγήσουν στην επίτευξη των στόχων του. Η παραπάνω διαδικασία παρουσιάζεται συνοπτικά στο Σχήμα 3.1.

Προκειμένου να γίνει σαφής η διαδικασία, είναι χρήσιμο να οριστούν οι ακόλουθες έννοιες:

- Ως **σύνολο καταστάσεων** (S) ορίζουμε όλες τις καταστάσεις στις οποίες μπορεί να βρεθεί το περιβάλλον με το οποίο αλληλεπιδρά ο πράκτορας. Κάθε κατάσταση του S απο-



Σχήμα 3.1: Αλληλεπίδραση πράκτορα-περιβάλλοντος ενισχυτικής μάθησης

τελείται από το σύνολο των δεδομένων που χρησιμοποιούνται προκειμένου να περιγραφεί το περιβάλλον.

- Ως **σύνολο διαθέσιμων δράσεων** (A) ορίζουμε όλες τις δράσεις που είναι σε θέση να εκτελέσει ο πράκτορας κατά την αλληλεπίδρασή του με το περιβάλλον. Η δράση που εκτελείται από τον πράκτορα σε μια συγκεκριμένη χρονική στιγμή επηρεάζει την κατάσταση του περιβάλλοντος την αμέσως επόμενη χρονική στιγμή.

- Ως **συνάρτηση ανταμοιβής** (R) ορίζουμε τη συνάρτηση εκείνη που αποδίδει μια συγκεκριμένη ανταμοιβή σε κάθε χρονική στιγμή. Η συνάρτηση αυτή θα πρέπει να επιλέγεται προσεκτικά, διότι η μεγιστοποίησή της είναι αυτή που καθορίζει τη σωστή πορεία της μάθησης.

Έχοντας περιγράψει τις βασικότερες έννοιες της ενισχυτικής μάθησης, είναι χρήσιμο να αναφερθούμε και στο δίλημμα μεταξύ εξερεύνησης και εκμετάλλευσης (exploration-exploitation dilemma). Στο ξεκίνημα της αλληλεπίδρασής του με το περιβάλλον, ο πράκτορας δεν έχει απολύτως καμία γνώση για αυτό. Επομένως, σε πρώτη φάση και μέχρι να αποκτήσει την απαραίτητη γνώση, ο πράκτορας δρα τυχαία. Καθώς σταδιακά συσσωρεύει χρήσιμη γνώση σχετικά με το περιβάλλον, επιλέγει μεταξύ εξερεύνησης και εκμετάλλευσης. Θα πρέπει δηλαδή να αποφασίσει αν θα συνεχίσει να δρα τυχαία εξερευνώντας περαιτέρω το περιβάλλον ή αν θα εκμεταλλευτεί τη χρήσιμη γνώση που έχει ήδη συσσωρεύσει προκειμένου να λάβει ορθότερες αποφάσεις. Σε ορισμένες περιπτώσεις, η απόφαση μεταξύ εξερεύνησης και εκμετάλλευσης είναι ιδιαίτερα δύσκολη. Από τη μία πλευρά, η συνεχής εκμετάλλευση συχνά αποκρύπτει πιθανές καταστάσεις του περιβάλλοντος οι οποίες δεν έχουν ακόμα εξερευνηθεί. Από την άλλη, η συνεχής εξερεύνηση περιορίζει την ανταμοιβή που λαμβάνει ο πράκτορας κατά την αλληλεπίδρασή του με το περιβάλλον. Θα πρέπει επομένως να διατηρείται η κατάλληλη ισορροπία μεταξύ εξερεύνησης και εκμετάλλευσης.

Σημειώνεται ότι οι αλγόριθμοι ενισχυτικής μάθησης εκτελούνται σε επεισόδια (episodes). Ένα επεισόδιο είναι μια αλληλουχία από καταστάσεις, δράσεις και επιβραβεύσεις, το οποίο ολοκληρώνεται με μία τερματική κατάσταση, από την οποία ο αλγόριθμος δεν μπορεί να προχωρήσει σε άλλη κατάσταση.

3.3 Μοντελοποίηση και Εξισώσεις Ενισχυτικής Μάθησης

Κάθε πρόβλημα ενισχυτικής μάθησης μπορεί να μοντελοποιηθεί ως μια Μαρκοβιανή Διαδικασία Αποφάσεων (Markov Decision Process). Η Μαρκοβιανή Διαδικασία Αποφάσεων αποτελεί ένα είδος επέκτασης της Μαρκοβιανής Αλυσίδας και ορίζεται ως ένα σύνολο
$< S, A, P, R, \gamma >$, όπου:

- S: ένα πεπερασμένο σύνολο καταστάσεων
- A: ένα πεπερασμένο σύνολο δράσεων
- P: ένας πίνακας πιθανοτήτων μεταβάσεων με βάση τον οποίο έχουμε ότι

$$P^a_{ss'} = P[S_{t+1} = s^{'}|S_t = s, A_t = a]$$

- R: μια συνάρτηση ανταμοιβής για την οποία ισχύει ότι

$$R^a_s = E[R_{t+1}|S_t = s, A_t = a]$$

- γ: ένας παράγοντας έκπτωσης όπου $\gamma \in [0, 1]$

Οι καταστάσεις μιας Μαρκοβιανής Διαδικασίας Αποφάσεων πληρούν τη μαρκοβιανή ιδιότητα. Αυτό σημαίνει πως η επόμενη κατάσταση εξαρτάται κάθε φορά αποκλειστικά και μόνο από την παρούσα κατάσταση και όχι από άλλες προηγούμενες καταστάσεις, επομένως ούτε από τον τρόπο με τον οποίο δημιουργήθηκε η παρούσα κατάσταση. Η μαρκοβιανή ιδιότητα συνοψίζεται στον ακόλουθο μαθηματικό τύπο:

$$P[S_{t+1}|S_t] = P[S_{t+1}|S_1, ..., S_t]$$

Η συνάρτηση ανταμοιβής που ορίστηκε παραπάνω επιστρέφει κάθε φορά την ανταμοιβή που παρέχεται στον πράκτορα μετά την εκτέλεση μιας συγκεκριμένης δράσης. Επειδή, ωστόσο, κατά τον προσδιορισμό μελλοντικών ανταμοιβών υπεισέρχεται ορισμένες φορές αβεβαιότητα, χρησιμοποιείται ο παράγοντας έκπτωσης. Ο παράγοντας αυτός σταθμίζει τις επιβραβεύσεις, δίνοντας μεγαλύτερη αξία στις πιο άμεσες εξ αυτών. Τελικά, μέσω αυτής της στάθμισης προκύπτει η συνάρτηση επιστροφής G_t, η οποία ορίζεται ως το άθροισμα όλων των μελλοντικών ανταμοιβών πολλαπλασιαζόμενων με τον αντίστοιχο συντελεστή αβεβαιότητας.

$$G_t = \sum_{n=0}^{\infty} \gamma^n R_{t+n+1}$$

Προκειμένου να επιλεγούν οι σωστές δράσεις, είναι απαραίτητο να οριστεί μια πολιτική, δηλαδή μια κατανομή πάνω στις δράσεις δεδομένων των καταστάσεων. Η πολιτική καθορίζει την πιθανότητα εκτέλεσης μιας δράσης με βάση την παρούσα κατάσταση, περιγράφοντας έτσι πλήρως τη συμπεριφορά του πράκτορα.

$$\pi(a|s) = P[A_t = a|S_t = s]$$

Τέλος, ιδιαίτερη σημασία έχουν η συνάρτηση αξίας κατάστασης (state value function) και η συνάρτηση αξίας δράσης (action value function). Ως συνάρτηση αξίας κατάστασης ορίζεται η αναμενόμενη τιμή των επιστροφών που θα έχουμε αν ξεκινήσουμε από την κατάσταση s και ακολουθήσουμε την πολιτική π. Κατ' αναλογία, ως συνάρτηση αξίας δράσης ορίζεται η αναμενόμενη τιμή των επιστροφών που θα έχουμε αν ξεκινήσουμε από την κατάσταση s, εκτελέσουμε τη δράση a και ακολουθήσουμε στη συνέχεια την πολιτική π. Οι δύο παραπάνω συναρτήσεις ορίζονται μαθηματικά ως εξής:

$$V_\pi(s) = E_\pi[G_t|S_t = s]$$
$$Q_\pi(s, a) = E_\pi[G_t|S_t = s, A_t = a]$$

Βασικό στόχο κατά την επίλυση ενός προβλήματος ενισχυτικής μάθησης αποτελεί ο προσδιορισμός των βέλτιστων συναρτήσεων αξίας κατάστασης και αξίας δράσης. Αυτό συμβαίνει, διότι, εάν προσδιοριστεί η βέλτιστη συνάρτηση αξίας δράσης, είναι πλέον εύκολο να οριστεί μια βέλτιστη πολιτική, με βάση την οποία επιλέγεται η δράση με τη μεγαλύτερη τιμή για την παρούσα κατάσταση. Για τον προσδιορισμό των βέλτιστων συναρτήσεων αξίας κατάστασης και αξίας δράσης έχουν προταθεί διάφορες τεχνικές. Αυτές περιλαμβάνουν, μεταξύ άλλων, μεθόδους δυναμικού προγραμματισμού, σε περιπτώσεις που είναι διαθέσιμο ένα πλήρες μοντέλο του προβλήματος, ή τη μέθοδο Monte Carlo, σε περιπτώσεις που ένα τέτοιο μοντέλο δεν είναι διαθέσιμο. Αξίζει, τέλος, να σημειωθεί ότι στην περίπτωση των Μαρκοβιανών Διαδικασιών Αποφάσεων ισχύει το ακόλουθο θεώρημα:

Θεώρημα 3.3.1. *Για κάθε Μαρκοβιανή Διαδικασία Αποφάσεων:*

- *Υπάρχει βέλτιστη πολιτική τέτοια ώστε:*

$$\pi_{βέλτιστη} \geq \pi, \forall \pi$$

- *Κάθε βέλτιστη πολιτική επιτυγχάνει τη βέλτιστη συνάρτηση αξίας κατάστασης*

$$V_{\pi_{βέλτιστη}}(s) = V_{βέλτιστη}(s)$$

- *Κάθε βέλτιστη πολιτική επιτυγχάνει τη βέλτιστη συνάρτηση αξίας δράσης*

$$Q_{\pi_{βέλτιστη}}(s, a) = Q_{βέλτιστη}(s, a)$$

3.4 Τεχνικές Ενισχυτικής Μάθησης

Όπως αναφέρθηκε και προηγουμένως, σκοπός του πράκτορα σε ένα πρόβλημα ενισχυτικής μάθησης είναι η επιλογή των δράσεων εκείνων που μεγιστοποιούν ένα κριτήριο ανταμοιβής. Σε

κάθε χρονική στιγμή t, ο πράκτορας επιλέγει με βάση την κατάσταση s_t του περιβάλλοντος μια δράση a_t. Χάρη σε αυτή τη δράση λαμβάνει μια ανταμοιβή r_t, ενώ το περιβάλλον βρίσκεται πλέον στην κατάσταση s_{t+1}. Η διαδικασία αυτή επαναλαμβάνεται εκ νέου με στόχο την επιλογή μιας σειράς δράσεων βάσει της οποίας μεγιστοποιείται η συνολική ανταμοιβή. Οι τεχνικές που χρησιμοποιούνται κατά κόρον στο πλαίσιο της ενισχυτικής μάθησης χωρίζονται σε δύο βασικές κατηγορίες:

- **Τεχνικές βασισμένες στην αξία** (Value Based): Οι τεχνικές αυτές αντιστοιχίζουν κάθε ζεύγος κατάστασης-δράσης σε μια τιμή που αποτελεί ένα είδος αξίας (Q-value). Όσο μεγαλύτερη είναι η αξία ενός τέτοιου ζεύγους τόσο καλύτερη είναι η επιλογή της συγκεκριμένης δράσης δοθείσης της συγκεκριμένης κατάστασης. Σε αυτήν την κατηγορία τεχνικών ανήκει ο αλγόριθμος Q-learning και οι παραλλαγές του όπως τα Deep Q-Networks (DQN), τα Double Deep Q-Networks και τα Dueling Deep Q-Networks.
- **Τεχνικές βασισμένες στην πολιτική** (Policy Based): Οι τεχνικές αυτές επιχειρούν να προσδιορίσουν τη βέλτιστη πολιτική απευθείας, χωρίς τον υπολογισμό των Q-values. Σε αυτήν την κατηγορία τεχνικών ανήκουν οι policy gradient αλγόριθμοι και ο αλγόριθμος REINFORCE.

Μία προσέγγιση που συνδυάζει τις δύο παραπάνω τεχνικές αποτελούν τα Actor-Critic μοντέλα τα οποία αναλύονται στην ενότητα 3.6.6.

3.5 Q-learning

Στην ενότητα αυτή θα αναλύσουμε τον γενικό αλγόριθμο Q-learning και κάποιες ειδικές περιπτώσεις του. Όταν δε μας είναι γνωστή η Μαρκοβιανή Διαδικασία Αποφάσεων αλλά είναι δυνατή η αλληλεπίδραση με το περιβάλλον, ενδείκνυνται μέθοδοι ενισχυτικής μάθησης. Ο Q-learning είναι ένας αλγόριθμος ενισχυτικής μάθησης ο οποίος αναπτύχθηκε από τον Watkins [34] και προσπαθεί να μάθει την αξία μίας δράσης σε μια συγκεκριμένη κατάσταση. Πρόκειται για έναν αλγόριθμο που δεν απαιτεί μοντέλο του περιβάλλοντος, έχοντας έτσι τη δυνατότητα να διαχειριστεί προβλήματα με στοχαστικές μεταβάσεις και επιβραβεύσεις.

Ο Q-learning είναι αλγόριθμος δύο παραμέτρων, της κατάστασης S και ενός συνόλου δράσεων A ανά κατάσταση. Αρχικά, ορίζουμε ως γ τον παράγοντα έκπτωσης, τον οποίο μπορούμε να σκεφτούμε και ως πιθανότητα επιτυχίας («καλής συνέχειας») σε ένα συγκεκριμένο βήμα. Ο αλγόριθμος λειτουργεί με βήματα. Μετά από Δt βήματα, η βαρύτητα του εκάστοτε βήματος θα είναι $\gamma^{\Delta t}$. Αυτό σημαίνει ότι στον αλγόριθμο δίνεται μεγαλύτερη βαρύτητα στα πρώτα βήματα.

Τα δεδομένα του αλγορίθμου θα αποθηκευτούν σε έναν πίνακα δύο διαστάσεων, μία για κάθε κατάσταση του S και μία για κάθε δράση του A. Ονομάζουμε αυτόν τον πίνακα Q-table. Αρχικά, πριν ξεκινήσει δηλαδή η μάθηση, βάζουμε στον Q-table αυθαίρετες τιμές. Κατά τη μάθηση, για κάθε χρονική στιγμή t που ο δράστης επιλέγει μια δράση a_t, παρατηρεί μια επιβράβευση r_t, εισέρχεται σε μια νέα κατάσταση s_{t+1} και η τιμή $Q(s_t, a_t)$ ανανεώνεται. Όπως και στα περισσότερα μοντέλα ενισχυτικής μάθησης, η μεταβολή που θα υπάρξει στην τιμή $Q(s_t, a_t)$ θα είναι

συνάρτηση της παλαιάς τιμής και της πληροφορίας που θα αποκτηθεί από το νέο βήμα. Στο εξής, θα συμβολίζουμε την τιμή του Q-table για την κατάσταση s_t και τη δράση a_t πριν από την ανανέωση των δεδομένων ως $Q^{old}(s_t, a_t)$ και μετά την ανανέωση των δεδομένων ως $Q^{new}(s_t, a_t)$.

Το ποσοστό μάθησης (learning rate) του αλγορίθμου, που συμβολίζεται με α και λαμβάνει τιμές στο $[0, 1]$, έχει ιδιαίτερη σημασία για την προσαρμοστικότητα των Q-values σε αλλαγές. Για την απόκτηση της $Q^{new}(s_t, a_t)$, αθροίζονται τρεις όροι. Ο ένας είναι:

$$(1 - \alpha) \cdot Q^{old}(s_t, a_t) \tag{3.1}$$

ο οποίος δηλώνει την εξάρτηση της νέας τιμής από την παλαιά τιμή. Όσο πιο κοντά στο 1 είναι το α, τόσο μικρότερη είναι η εξάρτηση της νέας τιμής από την παλαιά τιμή.

Στο σημείο αυτό παρατηρούμε ότι οι Q-values πρέπει να είναι τόσο μεγαλύτερες, όσο μεγαλύτερη είναι και η επιβράβευση στο συγκεκριμένο βήμα. Επομένως, προστίθεται και ο όρος:

$$\alpha \cdot r_t \tag{3.2}$$

ο οποίος δηλώνει ότι όσο μεγαλύτερη είναι η επιβράβευση, τόσο μεγαλύτερη θα είναι και η τιμή της συνάρτησης αξίας δράσης, κάτι που είναι φυσιολογικό, καθώς η αξία αυξάνεται όταν η επιβράβευση αυξάνεται. Τέλος, ο τρίτος όρος που προστίθεται είναι ο:

$$\alpha \cdot \gamma \cdot \max_{a' \in |\mathcal{A}|} Q(s_{t+1}, a') \tag{3.3}$$

μέσω του οποίου συνυπολογίζεται η μέγιστη επιβράβευση που μπορεί να ληφθεί από την κατάσταση s_{t+1}. Στον συγκεκριμένο όρο, γίνεται αντιληπτή η ύπαρξη του γ, δηλαδή του παράγοντα έκπτωσης, ο οποίος παρατηρούμε ότι δίνει μεγαλύτερη βαρύτητα στις αρχικές μεταβάσεις. Ακόμα, βλέπουμε ότι υπάρχει το ποσοστό μάθησης τόσο στον συγκεκριμένο όσο και στον προηγούμενο όρο. Αυτό συμβαίνει καθώς, αν θέλουμε ανελαστικές μεταβολές στις Q-values, τότε με ένα χαμηλό ποσοστό μάθησης μπορούμε να περιορίσουμε την επιρροή και των δύο αυτών όρων.

Συγκεντρωτικά, από τις σχέσεις (3.1), (3.2) και (3.3) προκύπτει

$$Q^{new}(s_t, a_t) = Q^{old}(s_t, a_t) + \alpha \cdot \left(r_t + \gamma \cdot \max_{a' \in |\mathcal{A}|} Q(s_{t+1}, a') - Q^{old}(s_t, a_t) \right) \tag{3.4}$$

που είναι η νέα τιμή του Q-table για την κατάσταση s_t και τη δράση a_t.

Ο όρος

$$r_t + \gamma \cdot \max_{a' \in |\mathcal{A}|} Q(s_{t+1}, a') \tag{3.5}$$

είναι η επιβράβευση από την κατάσταση s_t αθροιζόμενη με τη μέγιστη επιβράβευση από την κατάσταση s_{t+1}. Ο συγκεκριμένος όρος συμβολίζεται με y_t και ορίζεται ως ο στόχος (target) του αλγορίθμου, καθώς δείχνει το βέλτιστο αποτέλεσμα που μπορεί να επιτύχει. Όπως και σε άλλους αλγορίθμους ενισχυτικής μάθησης, έτσι και στον Q-learning, σκοπός είναι να επιλέγονται οι δράσεις που επιτυγχάνουν το στόχο μέσω της εκμετάλλευσης, αλλά και η συνεχής προσαρμογή του στόχου μέσω της εξερεύνησης.

Εκτελούμε τον αλγόριθμο για M επεισόδια και για T συνολικά χρονικές στιγμές. Ένα επεισόδιο ολοκληρώνεται όταν η κατάσταση s_{t+1} είναι τερματική κατάσταση. Τονίζουμε εδώ ότι δεν είναι απαραίτητο ο αλγόριθμος να εκτελείται σε επεισόδια, αλλά μπορεί να εκτελείται αενάως, σε ένα θεωρητικά ατέρμονο επεισόδιο, σε περίπτωση που δε βρίσκει τερματική κατάσταση ή δεν υπάρχει τέτοια.

Όσον αφορά το δίλημμα της εκμετάλλευσης-εξερεύνησης, η μέθοδος Q-learning χρησιμοποιεί συνήθως την ϵ-άπληστη πολιτική (ϵ-greedy). Σύμφωνα με αυτή, με πιθανότητα $\epsilon \in [0, 1]$ ο αλγόριθμος λαμβάνει μια τυχαία απόφαση, ενώ με πιθανότητα $1 - \epsilon$ δρα άπληστα, επιλέγοντας τη δράση εκείνη που εκτιμά ότι θα του προσφέρει τη μεγαλύτερη ανταμοιβή.

Algorithm 1 Αλγόριθμος Q-learning

for $i = 1$ to $|S|$ **do**
  for $j = 1$ to $|A|$ **do**
    $Q(i, j) \leftarrow 0$
  end for
end for
for *episode* $= 1$ to M **do**
  Βρες την τωρινή κατάσταση του περιβάλλοντος, s_1
  for $t = 1$ to T **do**
    $x \leftarrow$ random number $\in [0, 1]$
    if $x \leq \epsilon$ **then**
      $a_t \leftarrow$ τυχαία δράση (εξερεύνηση)
    else
      $a_t \leftarrow \max_{a' \in |\mathcal{A}|} Q(s_t, a')$ (εκμετάλλευση)
    end if
    Εκτέλεσε τη δράση a_t, παρατήρησε την επιβράβευση r_t
    και την επόμενη κατάσταση s_{t+1}
    $Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \cdot [r_t + \gamma \cdot \max_{a' \in |\mathcal{A}|} Q(s_{t+1}, a') - Q(s_t, a_t)]$
    $s_t \leftarrow s_{t+1}$
  end for
end for

Όπως είδαμε, ο Q-learning χρησιμοποιεί έναν πίνακα δύο διαστάσεων για την αναπαράσταση όλων των συνδυασμών καταστάσεων και δράσεων. Το γεγονός αυτό μπορεί να δημιουργήσει πρόβλημα μνήμης για πολύ μεγάλους χώρους καταστάσεων, ακόμα κι αν ο πίνακας είναι αραιός (sparse). Το συγκεκριμένο πρόβλημα, δηλαδή η ανεπάρκεια της μνήμης να καλύψει όλη την απαιτούμενη πληροφορία, αντιμετωπίζεται με μία τεχνική που συναντάμε στις επόμενες ενότητες.

3.6 Βαθιά ενισχυτική μάθηση

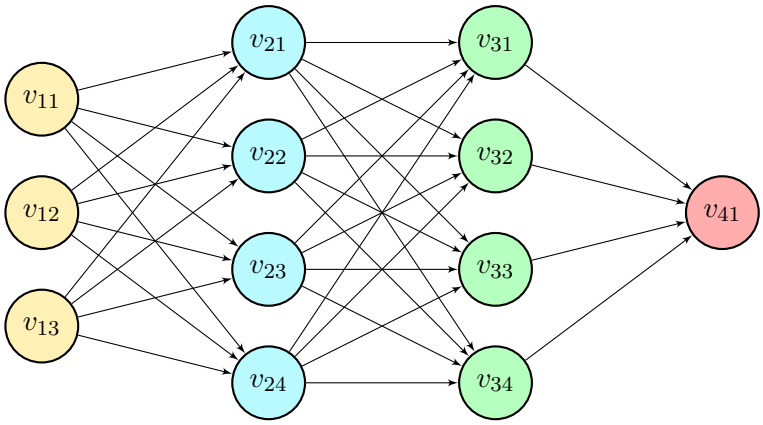
3.6.1 Παραδείγματα και χαρακτηριστικά

Ο όρος Βαθιά Ενισχυτική Μάθηση αναφέρεται στο πεδίο της Ενισχυτικής Μάθησης εμπλουτισμένο με τη χρήση νευρωνικών δικτύων. Η Βαθιά Μάθηση και γενικότερα τα νευρωνικά δίκτυα χρησιμοποιούνται έναντι πινάκων μνήμης, καθώς εκπαιδεύονται με τρόπο ώστε να μπορούν έξυπνα να παίρνουν αποφάσεις, χωρίς να χρειάζεται να αποθηκεύουν μεγάλη πληροφορία, όπως γίνεται με τους πίνακες.

Χαρακτηριστικό της Βαθιάς Ενισχυτικής Μάθησης είναι ότι, έχοντας πλέον το στοιχείο της Βαθιάς Μάθησης, οι είσοδοι που δέχονται οι πράκτορες μπορούν να είναι αρκετά μεγάλες σε μέγεθος, ενώ μπορεί κάλλιστα να είναι και μη δομημένες. Οι συγκεκριμένες δυνατότητες ήταν αυτές που πυροδότησαν τη γέννηση της Βαθιάς Ενισχυτικής Μάθησης, καθώς το 2013 η εταιρεία DeepMind δημιούργησε ένα μοντέλο βαθιάς μηχανικής μάθησης για να εκπαιδεύσει έναν πράκτορα ενισχυτικής μάθησης, ο οποίος παίζει βιντεοπαιχνίδια Atari με στόχο τη μεγιστοποίηση του σκορ του σε αυτά. Το αποτέλεσμα ήταν εντυπωσιακό, με το μοντέλο να πετυχαίνει αποτελέσματα καλύτερα από αυτά που πετύχαιναν οι άνθρωποι στα αντίστοιχα βιντεοπαιχνίδια.

Η συνέχεια ήταν ακόμα πιο εντυπωσιακή, καθώς δημιουργήθηκαν μοντέλα Βαθιάς Ενισχυτικής Μάθησης ακόμα και για τα πιο απαιτητικά παιχνίδια. Συγκεκριμένα, το 2016, η DeepMind ανέπτυξε τη μηχανή AlphaGo για το παιχνίδι Go, η οποία ξεκινούσε εκτελώντας τυχαίες κινήσεις και στη συνέχεια ανέπτυσσε γνώση μέσω βαθιάς ενισχυτικής μάθησης. Το συγκεκριμένο επίτευγμα ήταν αξιοσημείωτο, καθώς η επίδοση της μηχανής ήταν εξαιρετική, δεδομένου ότι το Go είναι ένα παιχνίδι με τεράστιο χώρο καταστάσεων, ο οποίος είναι αδύνατο να αποθηκευτεί σε μνήμη υπολογιστή [27].

Η πραγματική δυναμική όμως της βαθιάς ενισχυτικής μάθησης γίνεται αντιληπτή με ακόμα πιο έκδηλο τρόπο παρατηρώντας την ιστορία των μηχανών που έχουν αναπτυχθεί για το σκάκι. Η πρώτη μηχανή που αναπτύχθηκε για το σκάκι ήταν η Deep Blue, η οποία ξεκίνησε να κατασκευάζεται το 1985 και ολοκληρώθηκε το 1996, κερδίζοντας τον τότε παγκόσμιο πρωταθλητή Garry Kasparov. Η συγκεκριμένη μηχανή είχε μεγάλη υπολογιστική ισχύ, αλλά ο τρόπος λειτουργίας της βασιζόταν στην αποθήκευση καταστάσεων στη μνήμη του υπολογιστή. Ενώ η νίκη της Deep Blue τότε ήταν οριακή, σταδιακά, με την ανάπτυξη όλο και ισχυρότερου υλικού, δημιουργήθηκαν αρκετά ισχυρότερες σκακιστικές μηχανές. Το 2018, η κορυφαία σκακιστική μηχανή αποθήκευσης καταστάσεων ήταν η Stockfish, την οποία δεν μπορούσαν να ανταγωνιστούν ακόμα και οι καλύτεροι πρωταθλητές. Το 2018, η DeepMind ανέπτυξε τη μηχανή AlphaZero, η οποία έπαιζε σκάκι με χρήση βαθιάς ενισχυτικής μάθησης. Αποφασίστηκε έτσι μια σύγκρουση της μηχανής AlphaZero με τη μηχανή Stockfish ή, με άλλα λόγια, μία σύγκρουση μίας «παραδοσιακής» μηχανής (Stockfish) με μία μηχανή βαθιάς ενισχυτικής μάθησης (AlphaZero). Η σύγκρουση έγινε τόσο με περιορισμένο, όσο και με απεριόριστο υλικό (hardware). Στη σύγκρουση με περιορισμένο υλικό, η AlphaZero μπορούσε να εκπαιδευτεί μόλις για 9 ώρες. Το αποτέλεσμα ήταν εντυπωσιακό. Από τα 100 παιχνίδια, η AlphaZero κέρδισε τα 25 και έχασε μόλις τα 3, ενώ 72 έληξαν ισόπαλα. Στη σύγκρουση με απεριόριστο υλικό, η AlphaZero, από το σύνολο των 1200 παιχνιδιών, κέρδισε τα 290, έχασε μόλις τα 24, ενώ 886 έληξαν ισόπαλα [28].



Σχήμα 3.2: Τεχνητό Νευρωνικό Δίκτυο

Κατέστη, επομένως, φανερό ότι οι μηχανές βαθιάς ενισχυτικής μάθησης όχι μόνο μπορούν να αποδώσουν καλύτερα από ανθρώπους, αλλά μπορούν με άνεση να αποδώσουν καλύτερα και από αντίστοιχες μηχανές που έχουν πολύ μεγάλη αποθηκευτική ισχύ όμως δε χρησιμοποιούν βαθιά ενισχυτική μάθηση. Με τον τρόπο αυτό, η βαθιά ενισχυτική μάθηση απέκτησε μεγάλη δημοτικότητα, με αρκετούς ερευνητές να επιδιώκουν να την ενσωματώσουν στην αντιμετώπιση πρακτικών προβλημάτων.

3.6.2 Τεχνητά Νευρωνικά Δίκτυα

Τα τεχνητά νευρωνικά δίκτυα είναι υπολογιστικά συστήματα εμπνευσμένα από τα βιολογικά νευρωνικά δίκτυα των οργανισμών. Στην παρούσα εργασία, όταν αναφερόμαστε σε νευρωνικά δίκτυα θα εννοούμε τα τεχνητά νευρωνικά δίκτυα.

Γενικότερα, ένα νευρωνικό δίκτυο χρησιμοποιείται για να προσεγγίσει συναρτήσεις οι οποίες είτε είναι σχεδόν αδύνατο να προσεγγιστούν είτε η αποθήκευση των αποτελεσμάτων τους απαιτεί έναν τόσο μεγάλο χώρο καταστάσεων που είναι αδύνατο να αποθηκευτούν όλες οι επιθυμητές καταστάσεις με ακρίβεια.

Η δομή των νευρωνικών δικτύων μοιάζει αρκετά με τη δομή των δικτύων ενός βιολογικού εγκεφάλου [24]. Ένα νευρωνικό δίκτυο αποτελείται από κόμβους που ονομάζονται νευρώνες και συνδέονται μεταξύ τους μέσω ακμών. Οι ακμές αυτές ονομάζονται συνάψεις. Στο σχήμα 3.2, για παράδειγμα, οι νευρώνες αναπαρίστανται με κύκλους, ενώ οι συνάψεις με βέλη που συνδέουν νευρώνες. Οι συνάψεις χρησιμοποιούνται για να μεταφέρουν σήματα μεταξύ των νευρώνων. Ενώ η παραπάνω περιγραφή (με κόμβους και ακμές) παραπέμπει σε ένα γράφο, τα νευρωνικά δίκτυα δεν αντιμετωπίζονται ως γράφοι, αλλά ως στρώσεις επιπέδων νευρώνων. Τα σήματα, επομένως, μεταφέρονται από έναν νευρώνα σε έναν άλλο νευρώνα επόμενου επιπέδου. Στο σχήμα 3.2, κάθε επίπεδο αναπαρίσταται και με διαφορετικό χρώμα.

Σε κάθε νευρωνικό δίκτυο υπάρχει ένα επίπεδο εισόδου και ένα επίπεδο εξόδου. Στο σχήμα 3.2, με κίτρινο χρώμα είναι το επίπεδο εισόδου και με κόκκινο χρώμα το επίπεδο εξόδου. Διευκρινίζουμε εδώ ότι δεν είναι απαραίτητο το επίπεδο εξόδου να αποτελείται από έναν μόνο νευρώνα· είναι πιθανές και αρχιτεκτονικές με περισσότερους από έναν νευρώνες εξόδου. Τα επίπεδα εισόδου

και εξόδου ονομάζονται φανερά επίπεδα, καθώς ο χρήστης μπορεί να δει την τιμή εισόδου και την τιμή εξόδου, χωρίς να γνωρίζει την αρχιτεκτονική του νευρωνικού δικτύου. Τα ενδιάμεσα επίπεδα ονομάζονται κρυφά επίπεδα.

Τα σήματα που μεταδίδονται μεταξύ των νευρώνων είναι πραγματικοί αριθμοί. Κάθε νευρώνας μπορεί να έχει παραπάνω από μία εισόδους. Ο νευρώνας εκτελεί έναν υπολογισμό μιας συνάρτησης (πιθανώς μη γραμμικής) των εισόδων του και δίνει το αποτέλεσμα αυτό ως σήμα σε έναν ή περισσότερους νευρώνες του επόμενου επιπέδου. Είναι πιθανό οι νευρώνες να έχουν ένα κατώφλι για την έξοδο, κάτι που σημαίνει ότι, αν η τιμή εξόδου δεν υπερβαίνει αυτό το κατώφλι, τότε το σήμα εξόδου του νευρώνα είναι μηδενικό. Οι νευρώνες και οι συνάψεις έχουν βάρη, τα οποία διαφοροποιούνται όσο το νευρωνικό δίκτυο προσαρμόζεται στη λειτουργία που θέλει να επιτελέσει. Το βάρος μπορεί να ενισχύσει ή να μειώσει την ισχύ ενός σήματος, δρώντας ως πολλαπλασιαστικός παράγοντας στο σήμα. Νευρωνικά δίκτυα, όπως του σχήματος 3.2 είναι γνωστά και ως Multilayer Perceptrons (MLPs), επειδή ακριβώς έχουν πολλές στρώσεις (επίπεδα).

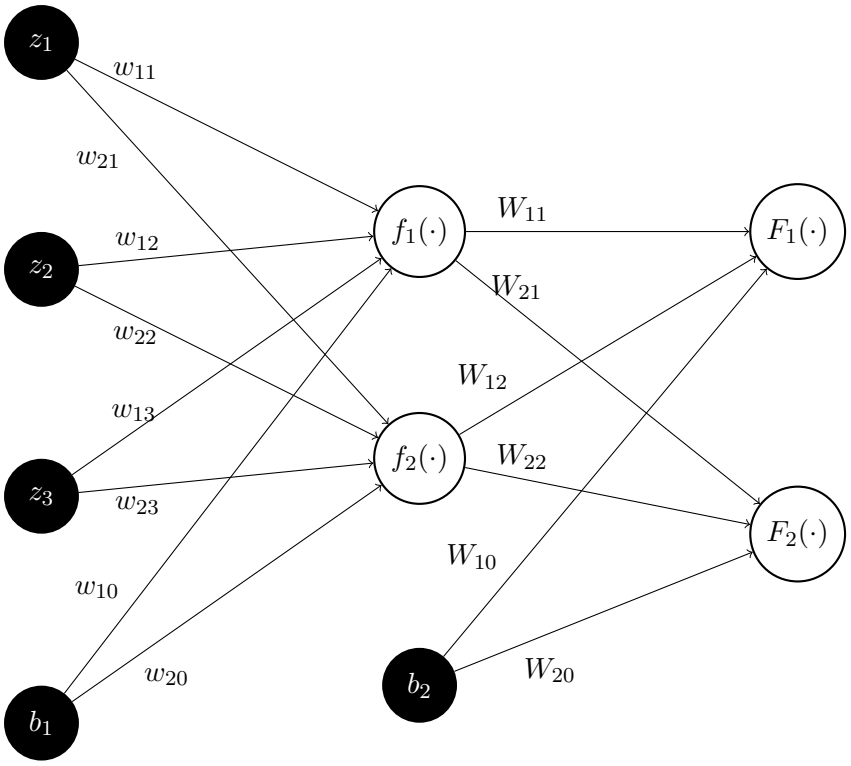
Μαθηματικά, η διαδικασία που περιγράφηκε παραπάνω και την οποία εκτελούν οι νευρώνες των νευρωνικών δικτύων ονομάζεται συνάρτηση ενεργοποίησης (activation function). Ένα παράδειγμα νευρωνικού δικτύου είναι αυτό στο οποίο κάθε νευρώνας έχει γραμμική συνάρτηση ενεργοποίησης, δηλαδή η έξοδος κάθε νευρώνα είναι γραμμικός συνδυασμός των εισόδων του, μαζί με την προσθήκη κάποιας σταθεράς. Έτσι, αν θεωρήσουμε x_i, με $i \in [1, k]$, τις k εισόδους του νευρώνα, w_i, με $i \in [0, k]$, τα βάρη των συνάψεων μεταξύ της i-οστής εισόδου και του νευρώνα (το i είναι 0 για τη σύναψη που συνδέει τον νευρώνα με τον σταθερό όρο) και b το bias (ο σταθερός όρος), προκύπτει ότι η έξοδος y του νευρώνα δίνεται από τη σχέση

$$y = \sum_{i=1}^{k} x_i w_i + w_0 b \tag{3.6}$$

Για παράδειγμα, στο σχήμα 3.3, αν υποθέσουμε ότι όλοι οι νευρώνες έχουν γραμμική συνάρτηση ενεργοποίησης, τότε ορίζοντας ως o_1, o_2, O_1, O_2 τις εξόδους από τους νευρώνες f_1, f_2, F_1, F_2 αντίστοιχα, έχουμε ότι λόγω της (3.6) θα ισχύουν οι σχέσεις:

$$o_1 = z_1 w_{11} + z_2 w_{12} + z_3 w_{13} + w_{10}$$
$$o_2 = z_1 w_{21} + z_2 w_{22} + z_3 w_{23} + w_{20}$$
$$O_1 = o_1 W_{11} + o_2 W_{12} + W_{10}$$
$$O_2 = o_1 W_{21} + o_2 W_{22} + W_{20}$$

Τα νευρωνικά δίκτυα όμως, ειδικά όταν πρέπει να προσεγγίσουν μια περίπλοκη συνάρτηση, δεν είναι πάντα ακριβή αν είναι γραμμικά. Μάλιστα, αποδεικνύεται ότι τα γραμμικά νευρωνικά δίκτυα μπορούν να απλοποιηθούν ώστε να έχουν μόνο δύο στρώματα, το στρώμα εισόδου και το στρώμα εξόδου. Συνήθως, λοιπόν, γίνεται χρήση μη γραμμικών συναρτήσεων ενεργοποίησης στα περισσότερα νευρωνικά δίκτυα. Συχνά χρησιμοποιούμενες συναρτήσεις είναι οι σιγμοειδείς (3.7) και (3.8), ενώ τα τελευταία χρόνια συναντάται συχνά και η ReLU (Rectified Rectangular Unit, (3.9)).



Σχήμα 3.3: Τεχνητό Νευρωνικό Δίκτυο με bias

$$y(x_i) = (1 + e^{-x_i})^{-1} \tag{3.7}$$

$$y(x_i) = \tanh(x_i) \tag{3.8}$$

$$y(x_i) = x_i^+ = \max(0, x_i) \tag{3.9}$$

3.6.3 Εκπαίδευση Νευρωνικών Δικτύων

Από τα προηγούμενα προκύπτει ότι, από τη μορφολογία του νευρωνικού δικτύου, ο κάθε νευρώνας θα χρησιμοποιεί πάντοτε κάθε φορά την ίδια συνάρτηση ενεργοποίησης, ενώ δεν είναι δυνατό να αφαιρεθούν συνάψεις ή να δημιουργηθούν νέες. Επομένως, για να αλλάξει η συμπεριφορά ενός νευρωνικού δικτύου και να προσαρμοστεί στα ερεθίσματα του περιβάλλοντος, ο μόνος τρόπος είναι να μεταβληθούν τα βάρη του.

Χωρίς βλάβη της γενικότητας, μπορούμε να ορίσουμε ότι οι έξοδοι $\hat{y}$ ενός νευρωνικού δικτύου είναι συνάρτηση των εισόδων x του δικτύου και των βαρών w του δικτύου, δηλαδή $\hat{y} = f(x; w)$. Προφανώς, επειδή το νευρωνικό δίκτυο εκφράζει προσεγγιστική συνάρτηση, οι έξοδοι $\hat{y}$ δεν είναι πάντα σωστές, αλλά αποτελούν προσεγγίσεις των σωστών τιμών. Ορίζουμε επομένως τη συνάρτηση απώλειας (loss function) $L(\hat{y}, y) = L(f(x; w), y)$ μεταξύ των πραγματικών τιμών y και των τιμών εξόδου του νευρωνικού δικτύου $\hat{y}$.

Το πρόβλημα εκπαίδευσης του νευρωνικού δικτύου ανάγεται, λοιπόν, στο πρόβλημα βελτιστοποίησης των βαρών του νευρωνικού δικτύου. Αναζητούμε, με λίγα λόγια, τα βάρη w_* εκείνα που θα ελαχιστοποιήσουν τη συνάρτηση απώλειας L, δηλαδή τα w_* για τα οποία ισχύει:

$$w_* = \arg\min_w L(w) \tag{3.10}$$

Η πιο γνωστή μέθοδος προσαρμογής των βαρών του μοντέλου στα δεδομένα είναι η μέθοδος της οπίσθιας διάδοσης, γνωστή και ως backpropagation [25]. Η μέθοδος αυτή υπολογίζει τη μερική παράγωγο της συνάρτησης απώλειας ως προς τα βάρη του δικτύου. Ξεκινώντας από το τελευταίο επίπεδο και χρησιμοποιώντας τον κανόνα της αλυσίδας, υπολογίζει αποδοτικά τις παραγώγους. Τονίζεται ότι η backpropagation μέθοδος χρησιμοποιείται μόνο για τον υπολογισμό των μερικών παραγώγων και όχι για την εκπαίδευση του δικτύου.

Ενώ οι παράγωγοι υπολογίζονται με backpropagation, η βελτιστοποίηση γίνεται συνήθως μέσω του αλγορίθμου καθόδου κλίσης (gradient descent) [6]. Ο αλγόριθμος καθόδου κλίσης χρησιμοποιείται για τον υπολογισμό ενός τοπικού ελαχίστου μιας διαφορίσιμης συνάρτησης. Στον αλγόριθμο καθόδου κλίσης, πραγματοποιούνται επαναλαμβανόμενα βήματα στην αντίθετη κατεύθυνση της μερικής παραγώγου της συνάρτησης, αντίθετα δηλαδή από την κατεύθυνση που η μερική παράγωγος έχει τη μεγαλύτερη τιμή. Διαισθητικά, μπορούμε να σκεφτούμε τον αλγόριθμο gradient descent σαν να θέλουμε να κατεβούμε ένα βουνό στο οποίο δεν έχουμε ορατότητα, έχουμε όμως πληροφορία για την κλίση του προς κάθε κατεύθυνση, στο σημείο που βρισκόμαστε. Επειδή θέλουμε να κατεβούμε περισσότερο, θα βρούμε το σημείο που έχει τη μεγαλύτερη κλίση προς τα κάτω και θα ακολουθήσουμε εκείνη την κατεύθυνση. Επαναληπτικά, όσο βρίσκουμε καθοδική κατεύθυνση, θα κατεβαίνουμε όλο και περισσότερο, επομένως τελικά θα οδηγηθούμε σε ένα τοπικό ελάχιστο. Με τον τρόπο αυτό, προσπαθούμε να εκπαιδεύσουμε το δίκτυο. Έχοντας τη μερική παράγωγο ως προς τα βάρη, προσπαθούμε να ελαχιστοποιήσουμε όσο το δυνατόν περισσότερο την τιμή της συνάρτησης απώλειας, επομένως ανανεώνουμε κάθε φορά τα βάρη στην αντίθετη κατεύθυνση της παραγώγου.

3.6.4 DQN (Deep Q-Networks)

Όπως είδαμε, μεγάλο πρόβλημα του κλασικού Q-learning είναι ότι όσο αυξάνεται ο χώρος καταστάσεων, γίνεται όλο και δυσκολότερο να χωρέσουν τα δεδομένα (καταστάσεις και δράσεις) στον πίνακα (Q-table), επομένως το Q-learning με χρήση Q-table μπορεί να αποδειχθεί εξαιρετικά μη αποδοτικό για μεγάλους χώρους καταστάσεων. Προκύπτει, λοιπόν, η ανάγκη ενσωμάτωσης κάποιου νευρωνικού δικτύου. Η προσαρμογή όμως των νευρωνικών δικτύων στις ανάγκες της ενισχυτικής μάθησης δεν είναι εύκολη υπόθεση, καθώς αυτή εμφανίζει μια σειρά από ιδιαιτερότητες που την καθιστούν επιρρεπή σε αστάθειες. Κάποιες από τις ιδιαιτερότητες αυτές είναι οι συσχετισμοί που υπάρχουν στην αλληλουχία παρατηρήσεων, τη στιγμή που η βαθιά μάθηση υποθέτει ανεξάρτητα και ομοιόμορφα κατανεμημένα δεδομένα, καθώς και το γεγονός ότι μικρές ανανεώσεις στις Q-values μπορεί να αλλάξουν δραματικά την πολιτική του δράστη [17].

Επομένως, στόχος είναι να βρεθεί μια παραλλαγή της μεθόδου Q-learning, η οποία θα είναι σε θέση να εκπαιδεύει αποτελεσματικά ένα νευρωνικό δίκτυο, ώστε να είναι δυνατή η επίτευξη βαθιάς μάθησης. Ουσιαστικά, ψάχνουμε ένα νευρωνικό δίκτυο του οποίου τα βάρη θα προσαρμόσουμε με τέτοιο τρόπο ώστε να προσεγγίζουν τη συμπεριφορά της συνάρτησης Q. Αναζητούμε, δηλαδή, ένα δίκτυο που μέσω της συνάρτησής του, $Q(s, a; w)$, θα εκτελεί την προσέγγιση

$$Q(s, a; w) \approx Q^*(s, a) \tag{3.11}$$

όπου Q^* οι πραγματικές τιμές της συνάρτησης Q.

Το συγκεκριμένο νευρωνικό ονομάζεται Q-network. Στόχος για το συγκεκριμένο δίκτυο είναι η προσαρμογή των βαρών για την ελαχιστοποίηση μίας συνάρτησης απωλειών. Η συνάρτηση απωλειών για τα DQN είναι η εξής:

$$L_i(w_i) = \mathbb{E}[(y_i - Q(s, a; w))^2] \tag{3.12}$$

όπου $\mathbb{E}$ η αναμενόμενη τιμή και y_i ο στόχος μετά την i-οστή επανάληψη, όπως αυτός ορίστηκε στην εξίσωση 3.5.

Στις περιπτώσεις που έχουμε μεγάλο πλήθος δειγμάτων και δεν είναι δυνατό να χρησιμοποιηθούν όλα για την εκπαίδευση του δικτύου, μπορούμε να χρησιμοποιήσουμε μία τεχνική που ονομάζεται αναπαραγωγή εμπειρίας (experience replay). Σύμφωνα με την τεχνική αυτή, επιλέγουμε για τη μάθηση ένα υποσύνολο των δειγμάτων με τυχαίο τρόπο, εξασφαλίζοντας, έτσι, μικρότερη συσχέτιση μεταξύ των δειγμάτων και μεγαλύτερη αποδοτικότητα στη δειγματοληψία [18].

Στον αλγόριθμο 2 παρουσιάζουμε τη συμπεριφορά του Deep Q-network. Η συμπεριφορά του αλγορίθμου μοιάζει αρκετά με το Q-learning, όμως περιέχει τη νέα συνάρτηση απώλειας (αφού πλέον πρόκειται για νευρωνικό δίκτυο) καθώς και την αναπαραγωγή εμπειρίας που αναφέραμε παραπάνω. Ο αλγόριθμος λειτουργεί ξανά με επεισόδια, ενώ αυτή τη φορά έχει και μία μνήμη επανάληψης (replay memory) D, ώστε να αποθηκεύονται τα τελευταία δείγματα για τυχαία δειγματοληψία (experience replay). Για τη βελτιστοποίηση του νευρωνικού δικτύου χρησιμοποιείται η τεχνική gradient descent. Για την εξερεύνηση του περιβάλλοντος χρησιμοποιείται η ϵ-greedy πολιτική, όπως εξηγήθηκε στην ενότητα 3.5.

Από τη στιγμή που δημιουργήθηκε ο αλγόριθμος DQN, έχουν προταθεί διάφορες βελτιστοποιήσεις. Μία από αυτές είναι η Double DQN, η οποία προκύπτει από την ανάγκη αντιστάθμισης της υπερεκτίμησης που ενδέχεται να κάνει το νευρωνικό δίκτυο για την τιμή της συνάρτησης Q [7]. Συγκεκριμένα, επειδή στον κλασικό αλγόριθμο DQN χρησιμοποιείται η ίδια η συνάρτηση Q για την εκτίμηση του μεγίστου αλλά και την επιλογή της επόμενης δράσης, είναι συχνό φαινόμενο η προβλεπόμενη τιμή της Q να είναι μεγαλύτερη από την κανονική. Η DeepMind προτείνει ένα μοντέλο στο οποίο η επιλογή της δράσης και η εκτίμηση της Q πραγματοποιούνται από δύο διαφορετικά, αλλά ίδιας αρχιτεκτονικής, νευρωνικά δίκτυα, αντιμετωπίζοντας έτσι τη σύγχυση που προκαλείται από τη χρήση του ίδιου νευρωνικού δικτύου για δύο διαφορετικούς σκοπούς.

Μία διαφορετική βελτιστοποίηση είναι το Dueling DQN, το οποίο αναλύεται στην επόμενη ενότητα.

Algorithm 2 Deep Q-Network

Αρχικοποίησε το Q-Network με τυχαία βάρη w
Αρχικοποίησε τη μνήμη επανάληψης D στη χωρητικότητά της, N
for *episode* $= 1$ to M **do**
  Βρες την τωρινή κατάσταση του περιβάλλοντος, s_1
  for $t = 1$ to T **do**
    $x \leftarrow$ random number $\in [0, 1]$
    if $x \leq \epsilon$ **then**
      $a_t \leftarrow$ τυχαία δράση (εξερεύνηση)
    else
      $a_t \leftarrow \max_{a' \in |\mathcal{A}|} Q(s_t, a'; w)$ (εκμετάλλευση)
    end if
    Εκτέλεσε τη δράση a_t, παρατήρησε την επιβράβευση r_t
    και την επόμενη κατάσταση s_{t+1}
    $s_t \leftarrow s_{t+1}$
    Δειγματοληψία k τυχαίων μεταβάσεων από την D
    if s_t is terminal **then**
      $y_t \leftarrow r_t$
    else
      $y_t \leftarrow r_t + \gamma \cdot \max_{a' \in |\mathcal{A}|} Q(s_{t+1}, a'; w)$
    end if
    Κάνε βήμα gradient descent στο $(y_t - Q(s_t, a_t; w))^2$
  end for
end for

3.6.5 Dueling DQN

Όπως είδαμε στην ενότητα 3.3, η συνάρτηση Q είναι η συνάρτηση αξίας δράσης, ενώ η συνάρτηση V είναι η συνάρτηση αξίας κατάστασης. Επομένως, η Q αναπαριστά την αξία της επιλογής μιας συγκεκριμένης δράσης δοθείσης μίας κατάστασης, ενώ η V αναπαριστά την αξία της συγκεκριμένης κατάστασης ανεξάρτητως της δράσης που θα ληφθεί. Ορίζουμε το πλεονέκτημα (advantage) ως την παρακάτω ποσότητα:

$$A(s, a) = Q(s, a) - V(s) \tag{3.13}$$

Διαισθητικά, το πλεονέκτημα δείχνει πόσο επικερδής είναι η επιλογή μιας δράσης σε σχέση με τις υπόλοιπες για τη δοθείσα κατάσταση.

Η αρχιτεκτονική μονομαχίας (dueling architecture) προτάθηκε από τους Wang *et al.* [33], ώστε να διαχωριστούν οι αξίες της κατάστασης και τα εξαρτώμενα από την κατάσταση πλεονεκτήματα σε δύο διαφορετικές ροές (streams). Η συγκεκριμένη αρχιτεκτονική έχει το πλεονέκτημα ότι, για μερικές εφαρμογές, δεν είναι απαραίτητο να γνωρίζουμε την αξία κάθε δράσης σε κάθε χρονικό βήμα. Οι συγγραφείς του paper έδωσαν ως παράδειγμα το παιχνίδι της Atari με όνομα Enduro, στο οποίο δεν είναι απαραίτητο ο δράστης να λάβει κάποια δράση μέχρι να φτάσει η στιγμή της σύγκρουσης.

Διαχωρίζοντας επομένως τους δύο εκτιμητές, η αρχιτεκτονική dueling έχει το πλεονέκτημα ότι μπορεί να μάθει ποιες καταστάσεις είναι επικερδείς και ποιες όχι, χωρίς να χρειάζεται να μαθαίνει την επίδραση που θα έχουν οι συγκεκριμένες δράσεις για κάθε κατάσταση. Η συγκεκριμένη αρχιτεκτονική έχει εφαρμογές σε περιπτώσεις όπου οι δράσεις δεν επηρεάζουν πάντα το περιβάλλον με τρόπο που έχει ενδιαφέρον για το μοντέλο.

Αναφορικά με τη συνένωση των δύο ροών, ο φαινομενικά κατάλληλος τρόπος σύνδεσης θα ήταν το άθροισμά τους:

$$Q(s, a; w, \alpha, \gamma) = V(s; w, \gamma) + A(s, a; w, \alpha) \tag{3.14}$$

Παρ' όλα αυτά, το άθροισμα συχνά δεν πετυχαίνει επιθυμητά αποτελέσματα για τους παρακάτω λόγους:

- Δεν είναι ορθό να υποθέσουμε ότι η $V(s; w, \gamma)$ και η $A(s, a; w, \alpha)$ δίνουν σωστές εκτιμήσεις της συνάρτησης αξίας κατάστασης και του πλεονεκτήματος κατάστασης, αντίστοιχα. Επομένως προκύπτουν ισχυρά σφάλματα από τη naive πρόσθεση αυτών των δύο τιμών.
- Δεδομένου του naive αθροίσματος (δηλαδή του $V + A$), δεν μπορούμε να ανακτήσουμε τις τιμές της V και της A. Εμπειρικά, στο paper των Wang *et al.* παρατηρείται ότι αυτή η έλλειψη δυνατότητας ανάκτησης δημιουργεί σημαντικά προβλήματα απόδοσης.

Για την αντιστάθμιση των παραπάνω σφαλμάτων, γίνεται χρήση εμπρόσθιας αντιστοίχισης (forward mapping) ως εξής:

$$Q(s,a;w,\alpha,\gamma) = V(s;w,\gamma) + A(s,a;w,\alpha) - \max_{a' \in |\mathcal{A}|} A(s,a';w,\alpha) \tag{3.15}$$

Με τον τρόπο αυτό, για την πιο επικερδή δράση η τιμή της Q γίνεται ίση με την τιμή της V, λύνοντας έτσι το ζήτημα της ανακτησιμότητας.

Εναλλακτικά, στα πειράματα των Wang *et al.*, χρησιμοποιείται το μοντέλο:

$$Q(s,a;w,\alpha,\gamma) = V(s;w,\gamma) + A(s,a;w,\alpha) - \frac{1}{|A|} \sum_{a' \in |\mathcal{A}|} A(s,a';w,\alpha) \tag{3.16}$$

Παρατηρούμε ότι ο τελευταίος όρος εκφράζει τον μέσο όρο των πλεονεκτημάτων για την κατάσταση s, για όλες τις πιθανές δράσεις στην κατάσταση αυτή. Με βάση τον παραπάνω τύπο, επιλέγουμε τη βέλτιστη δράση a^* με χρήση του τύπου:

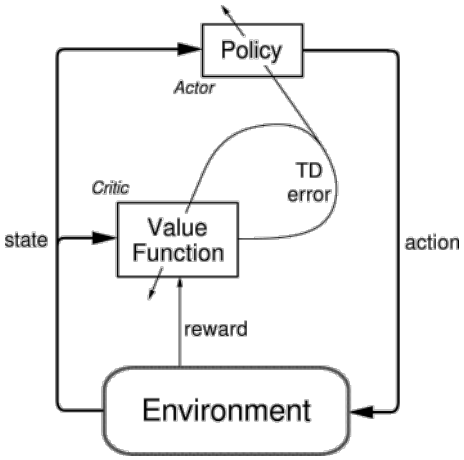
$$a^* = \arg\max_{a' \in |\mathcal{A}|} Q(s,a';w,\alpha,\gamma) \tag{3.17}$$

3.6.6 Actor-Critic

Τα Actor-Critic μοντέλα συνδυάζουν τα πλεονεκτήματα των value based και policy based τεχνικών, με αποτέλεσμα να χρησιμοποιούνται κατά κόρον σε εφαρμογές ενισχυτικής μάθησης. Πιο συγκεκριμένα, τα μοντέλα αυτά αποτελούνται από δύο επιμέρους τμήματα: έναν policy based δράστη (actor) και έναν value based κριτή (critic).

Ο policy based actor επιλέγει κάθε φορά την κατάλληλη δράση με βάση την παρούσα κατάσταση του περιβάλλοντος, ενώ ο value based critic υπολογίζει τα Q-values κάθε τέτοιου ζεύγους κατάστασης-δράσης. Πιο αναλυτικά, ο actor λαμβάνει ως είσοδο κάθε φορά την παρούσα κατάσταση και επιστρέφει τη βέλτιστη δυνατή δράση. Πρόκειται για μια policy based προσέγγιση, αφού επιχειρεί να προσδιορίσει τη βέλτιστη πολιτική απευθείας. Αντιθέτως, ο critic αξιολογεί τη δράση υπολογίζοντας την αξία της. Πρόκειται επομένως για μια value based προσέγγιση. Καθώς η διαδικασία επαναλαμβάνεται, τα δύο αυτά μοντέλα βελτιώνονται διαρκώς στο task που έχουν αναλάβει. Τα actor και critic μοντέλα προσεγγίζουν συγκεκριμένες συναρτήσεις, συνεπώς συχνά αποτελούν νευρωνικά δίκτυα. Το actor νευρωνικό δίκτυο έχει ως στόχο την επιλογή της βέλτιστης δράσης δοθείσης μιας κατάστασης του περιβάλλοντος, ενώ το critic νευρωνικό δίκτυο υπολογίζει το Q-value για ένα συγκεκριμένο ζεύγος κατάστασης-δράσης. Δέχεται δηλαδή ως είσοδο την κατάσταση του περιβάλλοντος, τη δράση που επέλεξε το actor νευρωνικό δίκτυο και την ανταμοιβή για την επιλεγμένη δράση, ενώ επιστρέφει ως έξοδο την αξία του συγκεκριμένου ζεύγους κατάστασης-δράσης [29].

Τα δύο νευρωνικά δίκτυα εκπαιδεύονται ξεχωριστά και, καθώς περνάει ο χρόνος, ο μεν actor επιλέγει όλο και καλύτερες δράσεις, ο δε critic γίνεται όλο και καλύτερος στην αξιολόγησή τους. Η αρχιτεκτονική ενός Actor-Critic μοντέλου παρουσιάζεται στο Σχήμα 3.4.



Σχήμα 3.4: Αρχιτεκτονική Actor-Critic μοντέλου [29]

Κεφάλαιο 4

Πειραματική πλατφόρμα και εφαρμογές

4.1 Χαρακτηριστικά μηχανήματος

Στον πίνακα 4.1 φαίνονται τα χαρακτηριστικά του Intel® Xeon® Processor E5-2630 v4, όπου διεξήχθησαν τα πειράματά μας.

Hardware	
Processor	Intel Xeon E5-2630 v4 (Broadwell), 10 cores, 2.2GHz, SMT disabled
LLC	25MB, 20-way set associative
Memory Bandwidth	68.3 Gbps per channel
Memory	128GB DDR4

Πίνακας 4.1: Χαρακτηριστικά του επεξεργαστή μας

4.2 Tailbench

Η σουίτα μετροπρογραμμάτων Tailbench [9] περιλαμβάνει εφαρμογές κρίσιμης απόκρισης ποικίλων χαρακτηριστικών καθώς και μια αξιόπιστη πειραματική μεθοδολογία που καθιστά εύκολη την εκτέλεση αυτών των εφαρμογών είτε σε πραγματικά συστήματα είτε σε προσομοιωτές. Η μεθοδολογία αυτή αποφεύγει πολλές παγίδες που πλήττουν άλλους συμβατικούς ελεγκτές φορτίου (load testers) [36]. Για παράδειγμα, κάποιοι load testers της σουίτας Cloudsuite μοντελοποιούν ένα closed-loop σύστημα, όπου ένα συγκεκριμένο πλήθος από νήματα είναι επιφορτισμένο με την αποστολή των αιτήσεων και τη λήψη των απαντήσεων. Καθένα από αυτά τα νήματα περιμένει (block wait) την επιστροφή του προηγούμενου προκειμένου να στείλει ένα νέο request. Συνεπώς, όταν το latency της υπηρεσίας τείνει να αυξηθεί, τα νήματα περιμένουν περισσότερο χρόνο τις απαντήσεις, άρα καθυστερούν να στείλουν νέες αιτήσεις. Αυτό έχει σαν αποτέλεσμα να μειώνεται ο ρυθμός αποστολής των αιτήσεων από τον loader και άρα η τιμή του latency να υποτιμάται σημαντικά. Ο load tester της Tailbench, αντιθέτως, ακολουθεί μια open-loop αρχιτεκτονική που είναι συμβατή με τη συμπεριφορά των latency critical εφαρμογών στην πράξη: δέχονται αιτήσεις από ένα μεγάλο πλήθος χρηστών με ρυθμό ανεξάρτητο από το throughput

τους. Ακόμα, πριν από κάθε πείραμα μετρήσεων, προηγείται μια warmup περίοδος επαρκούς αριθμού requests, ώστε να εξασφαλιστεί ότι οι μετρήσεις μας αφορούν μόνο την εκτέλεση σταθερής κατάστασης (steady-state execution).

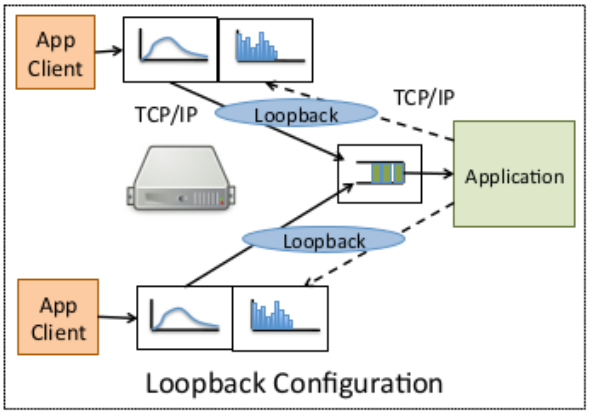
Η Tailbench παρέχει τη δυνατότητα εκτέλεσης του κάθε μετροπρογράμματος σε τρεις διαφορετικές διατάξεις (configurations):

1. **Networked configuration**: ο server και ο client τοποθετούνται σε διαφορετικά μηχανήματα και επικοινωνούν μεταξύ τους μέσω δικτύου. Αυτή η διάταξη επιτρέπει να ληφθούν υπ' όψιν όλες οι διαφορετικές αιτίες του tail latency, συμπεριλαμβανομένων των network link και switch καθυστερήσεων, καθώς και των network stack καθυστερήσεων.

2. **Loopback configuration**: ο server και ο client τοποθετούνται στο ίδιο μηχάνημα και επικοινωνούν μεταξύ τους με TCP/IP. Η διάταξη αυτή εστιάζει στην επεξεργασία των αιτήσεων, ενώ αγνοεί τις καθυστερήσεις δικτύου. Αντίθετα, αποτυπώνει τις περισσότερες καθυστερήσεις που προκαλούνται από το network stack.

3. **Integrated configuration**: ο server και ο client υλοποιούνται ως μία διεργασία και επικοινωνούν μεταξύ τους μέσω κοινής μνήμης. Η διάταξη αυτή, που αναπτύχθηκε για να διευκολύνει και να επιταχύνει την εκτέλεση σε περιβάλλον προσομοίωσης, δεν περιλαμβάνει τα network stack overheads.

Στην παρούσα εργασία χρησιμοποιούμε τα μετροπρογράμματα της Tailbench στο loopback configuration. Σημειώνεται ότι στο configuration αυτό, όπως και στο networked configuration, έχει δοθεί προσοχή στο να υπάρχουν επαρκείς clients, ώστε να μην είναι έντονη η επίδραση του client-side queuing.



Σχήμα 4.1: Tailbench: Loopback configuration [9]

Στη συνέχεια παρουσιάζουμε τα βασικά χαρακτηριστικά των δύο εφαρμογών κρίσιμης απόκρισης που μελετούμε στην παρούσα διπλωματική.

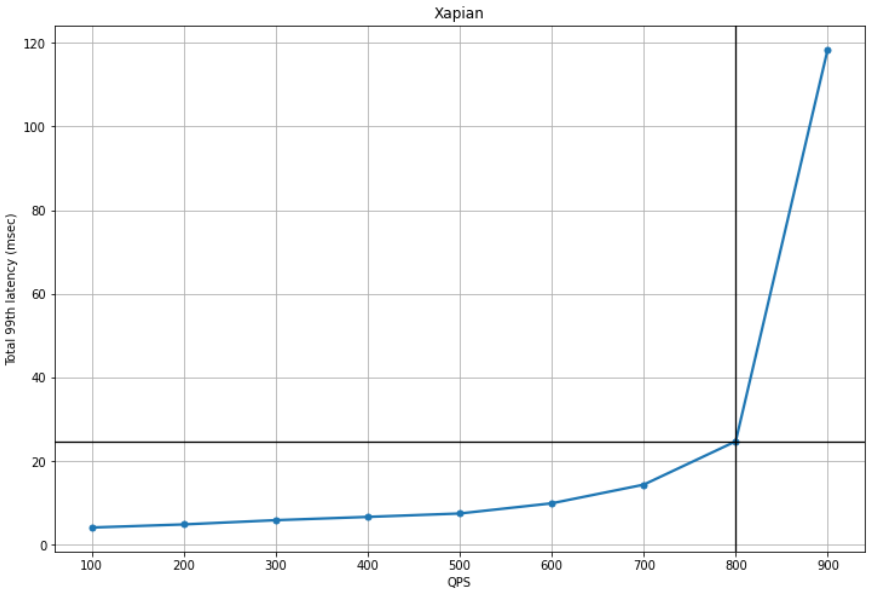
4.3 Εφαρμογές Κρίσιμης Απόκρισης

4.3.1 Xapian

Η Xapian αποτελεί μια μηχανή αναζήτησης ανοικτού κώδικα γραμμένη σε C++, η οποία χρησιμοποιείται συχνά τόσο σε δημοφιλείς ιστοσελίδες (π.χ. Debian wiki) όσο και σε πλαίσια λογισμικού (π.χ. Catalyst). Οι διαδικτυακές μηχανές αναζήτησης διαχειρίζονται δεδομένα της τάξης των Petabytes, τα οποία μοιράζονται σε χιλιάδες κόμβους-φύλλα (leaf nodes), όπου και πραγματοποιείται ο κύριος όγκος της επεξεργασίας. Η Xapian έχει υλοποιηθεί με τρόπο ώστε να αναπαριστά έναν κόμβο-φύλλο. Τα δεδομένα αναζήτησης που χρησιμοποιεί προέρχονται από την αγγλική εκδοχή της Wikipedia τον Ιούλιο του 2013. Τα αιτήματα επιλέγονται τυχαία και ακολουθούν κατανομή Zipfian, η οποία έχει αποδειχθεί ότι μοντελοποιεί ικανοποιητικά τις κατανομές αιτήσεων στις διαδικτυακές αναζητήσεις.

Για να βρούμε το μέγιστο φορτίο εισόδου που μπορεί να εξυπηρετήσει ο server και τη σχέση μεταξύ του latency και του φορτίου που παράγει ο loader, ξεκινάμε με χαμηλό QPS (queries-per-second) και σταδιακά το αυξάνουμε, μέχρι το σημείο-γόνατο όπου η εφαρμογή αρχίζει να απορρίπτει requests στην πλευρά του server. Στο σχήμα 4.2 φαίνεται η σχέση μεταξύ του 99^{th} percentile tail latency για τον server της Xapian και του QPS του loader του. Ορίζουμε την τιμή του QPS για το οποίο εμφανίζεται το γόνατο ως μέγιστο φορτίο (max load), δηλαδή το μέγιστο throughput για το οποίο η εφαρμογή μπορεί να διατηρήσει το QoS της. Ορίζουμε, επομένως, ως QoS στόχο της εφαρμογής την τιμή του 99^{th} percentile latency για QPS ίσο με το max load. Παρατηρούμε ότι για την περίπτωση της Xapian το μέγιστο φορτίο είναι τα 800 QPS και το QoS target τα 25 msec. Στη συνέχεια της παρούσας εργασίας, λοιπόν, θα χρησιμοποιούμε τη Xapian με φορτίο 800 QPS και θα θεωρούμε ότι είμαστε εντός του QoS στόχου μας, για το σύνολο της εκτέλεσης, αν το συνολικό 99^{th} percentile latency είναι μικρότερο ή ίσο από 25 msec. Σημειώνουμε ότι αυτά τα νούμερα εξαρτώνται, φυσικά, από τα χαρακτηριστικά του μηχανήματος στο οποίο εκτελούμε τα πειράματά μας.

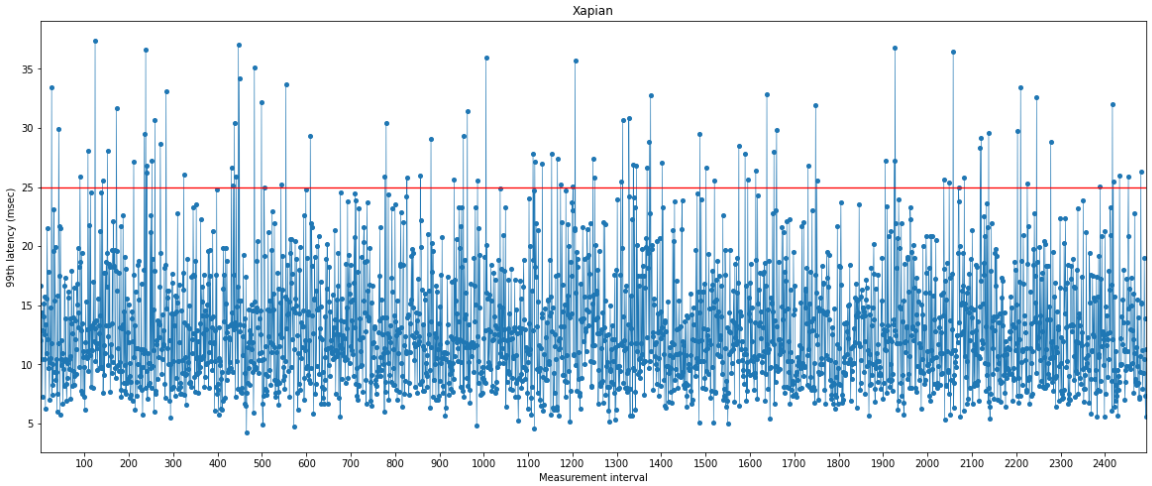
Ωστόσο, ο πράκτορας ενισχυτικής μάθησης που θα χρησιμοποιήσουμε λαμβάνει αποφάσεις ανά τακτά χρονικά διαστήματα. Για την αξιολόγηση του πόσο ευεργετική ήταν η εκάστοτε απόφαση και άρα για τη σταδιακή βελτίωση, μέσω ανάδρασης, της συμπεριφοράς του πράκτορα, χρειαζόμαστε να ποσοτικοποιήσουμε την επίδραση της κάθε απόφασης στο latency της LC εφαρμογής. Συνεπώς, δε μας αρκεί να διαθέτουμε το 99^{th} percentile latency για το σύνολο της εκτέλεσης της εφαρμογής, αλλά πρέπει να το γνωρίζουμε για κάθε χρονικό παράθυρο ξεχωριστά. Το χρονικό παράθυρο για το οποίο θα πρέπει να υπολογίσουμε το latency ξεκινάει από τη στιγμή εφαρμογής μιας νέας απόφασης του πράκτορα στο σύστημα και έχει σταθερή διάρκεια ίση με ένα πλήρες βήμα του πράκτορα, δηλαδή με τον χρόνο που μεσολαβεί μεταξύ δύο διαδοχικών αποφάσεων του πράκτορα. Επισημαίνουμε ότι η σουίτα Tailbench, σε αντίθεση με άλλες σουίτες μετροπρογραμμάτων, δεν παρέχει έτοιμο το latency που αντιστοιχεί σε παράθυρο της επιλογής μας, παρά μόνο το latency για το σύνολο της εκτέλεσης μιας εφαρμογής. Για τον λόγο αυτόν, κρατάμε τα sojourn times για όλα τα αιτήματα εντός του εκάστοτε παραθύρου και υπολογίζουμε για αυτά το 99^{th} percentile latency. Περισσότερες πληροφορίες για την υλοποίηση αυτής της λειτουργίας αναφέρουμε στην ενότητα 4.5.3.



Σχήμα 4.2: Xapian: QPS - 99$^{\text{th}}$ percentile latency

Διαισθητικά, το μέγεθος του παραθύρου που θα επιλέξουμε πρέπει να είναι επαρκώς μικρό ώστε να μπορεί να αποτυπώσει τις διαφορετικές φάσεις τόσο της LC όσο και των BE εφαρμογών, οι οποίες καθορίζουν το επίπεδο του ανταγωνισμού κάθε στιγμή. Με άλλα λόγια, αν το παράθυρο είναι πολύ μεγάλο, τότε και οι αποφάσεις του πράκτορα θα λαμβάνονται σε πολύ αραιά διαστήματα και άρα θα χαθούν ευκαιρίες για fine-grained προσαρμογή στις μεταβολές του ανταγωνισμού. Για παράδειγμα, αν για κάποιο χρονικό διάστημα διάρκειας μικρότερης από αυτής του παραθύρου ο ανταγωνισμός για πόρους είναι χαμηλός, θα έχει χαθεί μια ευκαιρία του πράκτορα να βελτιώσει τη χρησιμοποίηση των πόρων με το να αναθέσει περισσότερα ways στις BEs. Επιπροσθέτως, η διάρκεια του παραθύρου καθορίζει και το πλήθος των βημάτων βελτιστοποίησης που πραγματοποιεί ο αλγόριθμος ενισχυτικής μάθησης εντός ενός συγκεκριμένου χρόνου. Σε κάθε βήμα αλληλεπίδρασης του πράκτορα με το περιβάλλον, συλλέγουμε ένα νέο δείγμα πληροφοριών και ταυτόχρονα εκτελούμε ένα βήμα βελτιστοποίησης. Η επιλογή ενός μεγαλύτερου διαστήματος συνεπάγεται μείωση των συλλεγόμενων δειγμάτων και των βημάτων βελτιστοποίησης, κάτι που πιθανώς να επηρεάζει αρνητικά τη διαδικασία της εκπαίδευσης. Ταυτόχρονα, όμως, ένα πολύ μικρό παράθυρο είναι επίσης μη θεμιτό. Υπολογίζουμε πως από τη στιγμή που ο πράκτορας θα λάβει μια απόφαση, χρειάζεται ένα διάστημα μερικών milliseconds για να εφαρμοστεί η απόφαση αυτή στο σύστημα, μέσω της τεχνολογίας CAT. Συνεπώς, στη μέτρηση του latency ενός παραθύρου πιθανώς να προσμετρώνται τιμές που αφορούν την προηγούμενη απόφαση. Όσο μικρότερο είναι το διάστημα που χρησιμοποιούμε, τόσο μεγαλύτερη θα είναι η συνεισφορά αυτών των τιμών. Ακόμα, σε ένα μικρό διάστημα αντιστοιχεί και μικρότερο πλήθος αιτήσεων, άρα στην εκτίμηση της τιμής του latency μπορεί να συμμετέχει ένας ανεπαρκής αριθμός απαντήσεων. Στην περίπτωση της Xapian αλλά και, όπως θα δούμε στη συνέχεια, της Img-dnn, αυτός ο κίνδυνος είναι υπαρκτός, αφού το QPS τους δεν είναι πολύ υψηλό.

Λαμβάνοντας υπ' όψιν όλα τα παραπάνω, καταλήξαμε στην τιμή 200 msec για τη διάρκεια του παραθύρου. Επιλέξαμε μάλιστα την ίδια τιμή τόσο για τη Xapian όσο και για την Img-dnn, αφενός επειδή τα QPS για τα οποία τις χρησιμοποιούμε δεν είναι πολύ διαφορετικά μεταξύ τους και αφετέρου για να απλοποιήσουμε την υλοποίησή μας και να μπορούμε να μεταβαίνουμε από τη

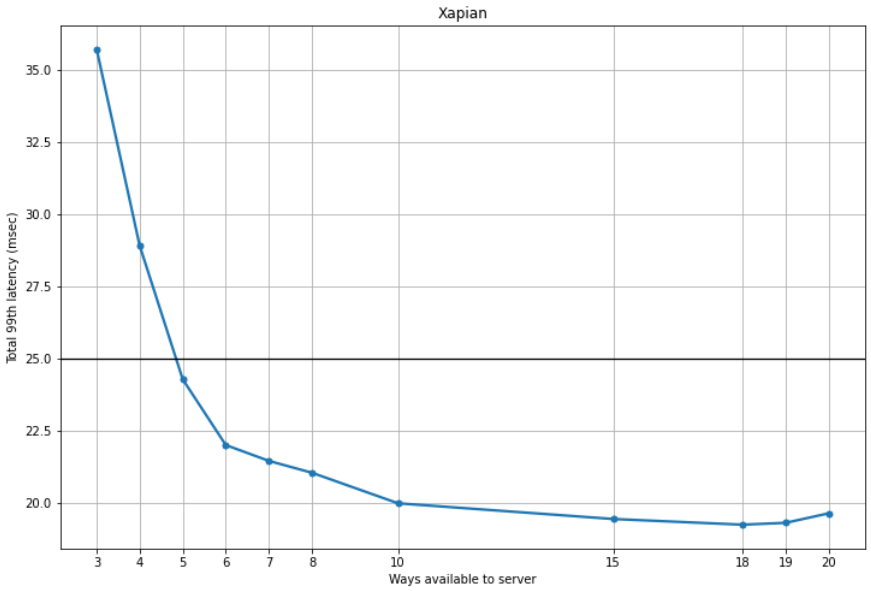


Σχήμα 4.3: Xapian: 99$^{\text{th}}$ percentile latency ανά παράθυρο

μία εφαρμογή στην άλλη χωρίς ο πράκτορας να αλλάζει τον ρυθμό με τον οποίο λειτουργεί. Παρ' όλα αυτά, η επιλογή αυτή δεν είναι δεσμευτική, ενώ στην περίπτωση που είχαμε LC εφαρμογές με πολύ διαφορετικά QPS μεταξύ τους, τότε σίγουρα θα επιλέγαμε διαφορετική διάρκεια παραθύρου για την καθεμία.

Στην εικόνα 4.3 φαίνεται η τιμή του latency ανά παράθυρο για την Xapian. Όπως ήταν αναμενόμενο, το latency κάποιων παραθύρων ξεπερνάει το QoS target των 25 msec που θέσαμε. Αυτό είναι διαισθητικά κατανοητό, αφού το αποτέλεσμα του υπολογιζόμενου latency εξαρτάται πλέον από πολύ λιγότερα αιτήματα και συνεπώς είναι περισσότερο ευαίσθητο στα outliers, τα οποία και καθορίζουν την τιμή του 99$^{\text{th}}$ percentile latency. Για την περίπτωση της Xapian, βρίσκουμε ότι σε απομονωμένη εκτέλεσή της στο σύστημα, χωρίς BE εφαρμογές στο socket της, το ποσοστό των παραθύρων που είναι εκτός του QoS κυμαίνεται κοντά στην τιμή 3.73%. Η τιμή αυτή θα αποτελέσει το μέτρο σύγκρισης για τα πειράματά μας στη συνέχεια. Με άλλα λόγια, θα συγκρίνουμε (κανονικοποιούμε) το ποσοστό παραθύρων που ξεπερνούν τα 25 msec υπό συνθήκες συνεκτέλεσης με το ποσοστό 3.43%, προκειμένου να έχουμε ένα μέτρο του πόσο επιβαρύνθηκε η LC εφαρμογή σε σχέση με την περίπτωση της απομονωμένης εκτέλεσής της στο σύστημα.

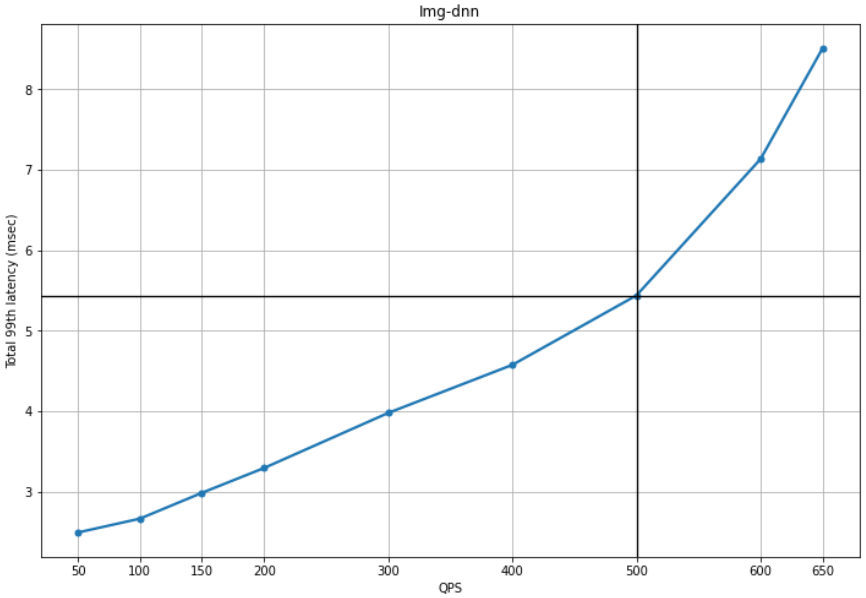
Στο σχήμα 4.4 παρουσιάζεται το πώς μεταβάλλεται το συνολικό latency της εφαρμογής ανάλογα με το πλήθος των ways που ανατίθενται στον server της, σε συνθήκες απομονωμένης εκτέλεσης. Παρατηρούμε ότι, όταν η Xapian εκτελείται μόνη της στο σύστημα, ικανοποιεί το QoS target της με ένα υποσύνολο των διαθέσιμων ways της LLC. Συγκεκριμένα, της αρκούν 5 από τα 20 ways της LLC ή ισοδύναμα 6.25 MB από τα 25 MB. Από την ανάλυση αυτή συμπεραίνουμε ότι υπάρχει περιθώριο συνεκτέλεσης της Xapian με άλλες εφαρμογές. Στην περίπτωση που απαιτούσε το σύνολο των ways για να διατηρήσει το QoS της, η συνεκτέλεσή της με BE εφαρμογές δε θα ήταν εφικτή.



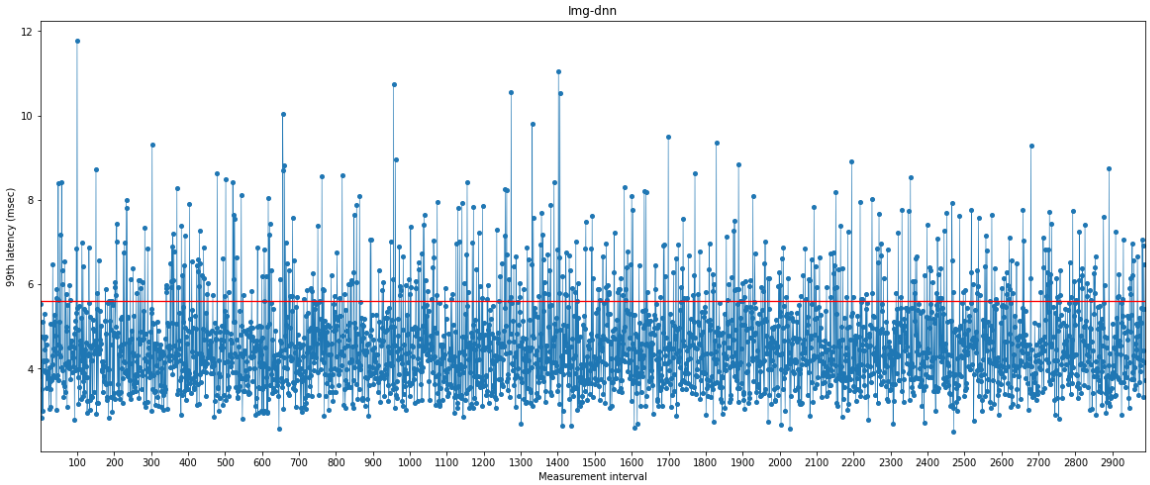
Σχήμα 4.4: Xapian: ways available to server - 99$^{\text{th}}$ percentile latency

4.3.2 Img-dnn

Η Img-dnn αποτελεί μια εφαρμογή αναγνώρισης χειρογράφων (handwriting recognition) που στηρίζεται στη βιβλιοθήκη OpenCV. Η αναγνώριση χειρογράφων ανήκει στη γενικότερη κατηγορία εφαρμογών αναγνώρισης εικόνας και βρίσκει πολλές εφαρμογές στην οπτική αναγνώριση χαρακτήρων, την αναζήτηση από εικόνα (imaged-based search), την αυτόματη τοποθέτηση ετικέτας σε εικόνα (image tagging) και πολλές άλλες διαδικτυακές εφαρμογές. Η Img-dnn χρησιμοποιεί έναν autoencoder υλοποιημένο με βαθιά νευρωνικά δίκτυα, ο οποίος αναγνωρίζει χειρόγραφους χαρακτήρες. Η εφαρμογή χρησιμοποιεί τυχαία επιλεγμένα δείγματα από την MNIST βάση δεδομένων.



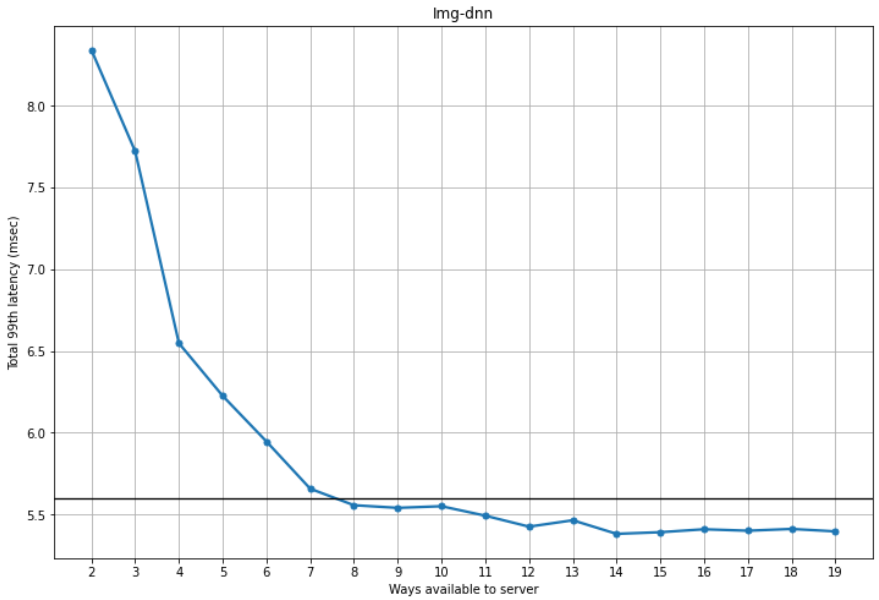
Σχήμα 4.5: Img-dnn: QPS - 99$^{\text{th}}$ percentile latency



Σχήμα 4.6: Img-dnn: 99$^{\text{th}}$ percentile latency ανά παράθυρο

Πραγματοποιώντας την ίδια ανάλυση με τη Xapian για τη σχέση latency-QPS, βρίσκουμε ότι το QoS target που θα θέσουμε για την Img-dnn είναι 5.6 msec, για QPS 500 (Σχήμα 4.5). Το ποσοστό παραθύρων που παραβιάζουν το QoS σε συνθήκες απομονωμένης εκτέλεσης βρέθηκε 16.12% (Σχήμα 4.6).

Από το σχήμα 4.7 συμπεραίνουμε ότι η Img-dnn πετυχαίνει το QoS target της με τουλάχιστον 8 ways στον server. Συνεπώς και εδώ έχουμε περιθώριο για συνεκτέλεση. Η ανάλυση αυτή μας δίνει, επιπλέον, την πληροφορία ότι η Img-dnn είναι πιο απαιτητική ως προς το μέγεθος της cache που χρειάζεται σε σχέση με την Xapian. Η αξιολόγηση, λοιπόν, των αλγορίθμων ενισχυτικής μάθησης που θα χρησιμοποιήσουμε στην πορεία θα πραγματοποιηθεί σε LC εφαρμογές διαφορετικών απαιτήσεων, κάτι το οποίο είναι πολύ χρήσιμο για τον έλεγχο της γενικότητας και της προσαρμοστικότητας των μεθόδων μας.



Σχήμα 4.7: Img-dnn: ways available to server - 99$^{\text{th}}$ percentile latency

Συγκεντρωτικά, τα παραπάνω νούμερα παρατίθενται στον πίνακα 4.2. Τα μεγέθη αυτά είναι τα μόνα τα οποία χρειαζόμαστε να γνωρίζουμε εκ των προτέρων για την LC εφαρμογή μας ώστε να πραγματοποιήσουμε τα πειράματά μας και να αξιολογήσουμε την αποδοτικότητα των αλγορίθμων ενισχυτικής μάθησης που επιστρατεύουμε.

	Max load - QPS	**QoS target latency (msec)**	**Minimum % of violations**
Xapian	800	25	3.73
Img-dnn	500	5.6	16.12

Πίνακας 4.2: Χαρακτηριστικά των LC εφαρμογών

4.4 Εφαρμογές βέλτιστης προσπάθειας

Η συνεκτέλεση των παραπάνω LC εφαρμογών πραγματοποιείται με μία ποικιλία από Best-Effort εφαρμογές από διάφορες πηγές. Συγκεκριμένα, χρησιμοποιούμε:

1. Δύο εφαρμογές Apache Spark της σουίτας Cloudsuite, μία για in-memory και μία για graph analytics (Spark Job In-Memory Analytics, Spark Job Graph Analytics).
2. Δύο εφαρμογές Μηχανικής Μάθησης που διαθέτει το Apache Spark. Για την αξιοποίησή τους έγινε χρήση ενός πραγματικού συνόλου δεδομένων από το αποθετήριο UCI [30] (Linear-SVC, RandomForestClassifier).
3. Πέντε εφαρμογές από τη σουίτα SPEC2006 (mcf, hmmer, astar, lbm, gems). Εκτός των πέντε αυτών SPEC μετροπρογραμμάτων, δοκιμάστηκε και το libquantum. Λόγω του ότι οι SPEC εφαρμογές έχουν επιλεγεί στοχευμένα για να δημιουργούν φόρτο σε διάφορους πόρους του μηχανήματος, κατά τις συνεκτελέσεις με 9 ίδια στιγμιότυπα του libquantum, παρατηρήσαμε ότι ο ανταγωνισμός είναι τέτοιος, που οι servers των LC εφαρμογών μας δεν μπορούν να ανταποκριθούν στην εισερχόμενη κίνηση. Μάλιστα, δοκιμάσαμε να αναθέσουμε όλα τα ways της LLC πλην ενός στον server της LC εφαρμογής και παρατηρήσαμε ότι το QoS target παραβιάζεται κατά πολύ. Συνεπώς, όποια απόφαση και να πάρει ο πράκτοράς μας για τον διαμοιρασμό των ways, το QoS της LC εφαρμογής δε θα εξασφαλίζεται. Για τον λόγο αυτόν απορρίψαμε το libquantum από το σύνολο των BE εφαρμογών μας.

Σημειώνεται ότι εκτελούμε όλες τις BE εφαρμογές μέσα σε Docker containers, για την εύκολη και άμεση χρήση τους.

4.5 Περιγραφή συστήματος

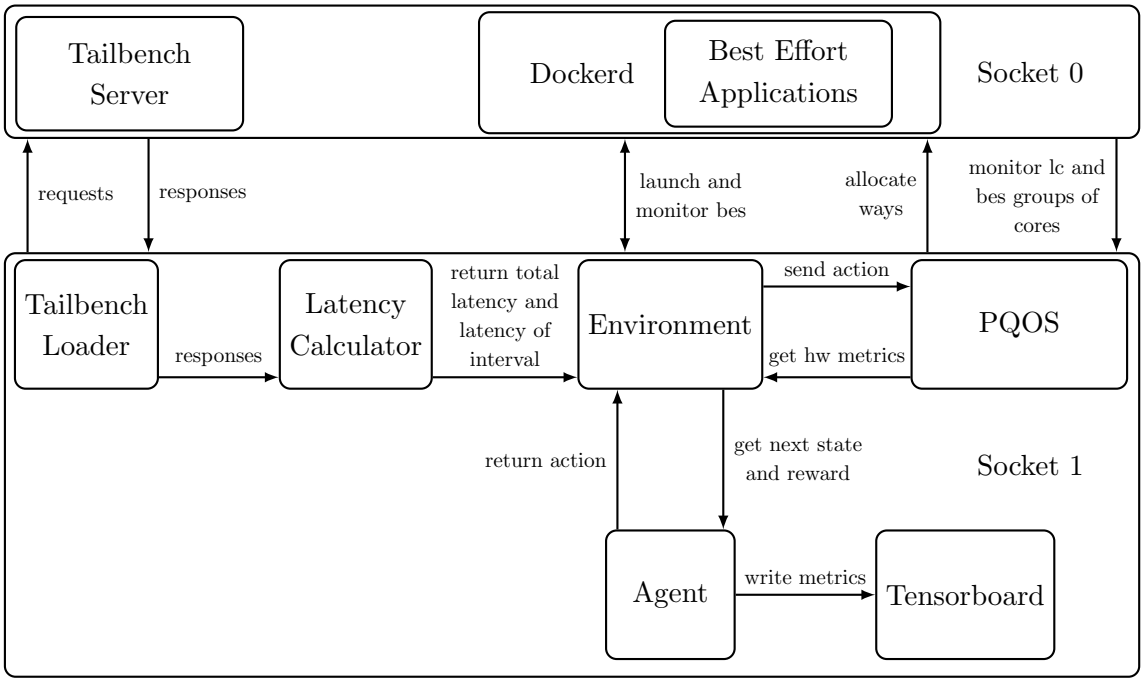
Η οργάνωση του συστήματος πάνω στο οποίο διεξήχθησαν τα πειράματά μας στηρίχθηκε σε αυτή προηγούμενης εργασίας του εργαστηρίου [23], κατάλληλα τροποποιημένης ώστε να μπορεί να χρησιμοποιηθεί για τη σουίτα Tailbench. Αναφέρουμε ότι, ενώ οι μετρήσεις μας αφορούν

μόνο δύο από τις οκτώ εφαρμογές του Tailbench, το σύστημά μας είναι οργανωμένο με τέτοιο τρόπο ώστε να είναι πολύ εύκολη η χρήση οποιασδήποτε άλλης LC εφαρμογής της Tailbench με μια απλή αλλαγή μεταβλητών περιβάλλοντος. Στο σχήμα 4.8 φαίνονται τα διάφορα μέρη που απαρτίζουν το σύστημα καθώς και το πώς αλληλεπιδρά το ένα με το άλλο. Ακολούθως θα περιγραφούν ξεχωριστά τα τμήματά του που δεν έχουν αναλυθεί έως τώρα.

4.5.1 PQOS

Όπως έχουμε ήδη αναφέρει, χρησιμοποιούμε την τεχνολογία Intel RDT για την παρακολούθηση των συνεκτελούμενων εφαρμογών μας και την ανάθεση σε αυτές τμήματος της Last Level Cache για αποκλειστική χρήση. Η προγραμματιστική διεπαφή που μας δίνει αυτήν τη δυνατότητα ονομάζεται PQOS [31]. Για αυτή χρησιμοποιούμε τον αντίστοιχο Python wrapper που παρέχεται από την Intel, αφού και το υπόλοιπο σύστημα έχει υλοποιηθεί σε Python.

Η τεχνολογία RDT μας δίνει τη δυνατότητα να παρακολουθήσουμε είτε ομάδες από pid είτε ομάδες πυρήνων. Εμείς χρησιμοποιούμε τη δεύτερη δυνατότητα. Δημιουργούμε δύο ομάδες πυρήνων, μία με τους πυρήνες που χρησιμοποιεί η υπηρεσία κρίσιμης απόκρισης και μία με τους πυρήνες που αναθέτουμε στις εφαρμογές βέλτιστης προσπάθειας. Συγκεκριμένα, στα πειράματά μας, ένας πυρήνας του socket 0 ανατέθηκε στον Tailbench server και οι υπόλοιποι 9 ανατέθηκαν στις BEs. Το PQOS μάς δίνει τη δυνατότητα παρακολούθησης, για κάθε ομάδα, μετρικών όπως το μέγεθος της LLC που καταλαμβάνει, το bandwidth που χρησιμοποιεί, αλλά και το IPC και τα misses μέσω του εργαλείου perf [35]. Επιπλέον, τροποποιούμε κατάλληλα την Python διεπαφή του PQOS ώστε, εκτός από τις hardware μετρικές που αφορούν ολόκληρη την ομάδα των BEs, να λαμβάνουμε και το IPC του κάθε BE που ολοκληρώνεται ξεχωριστά. Τα μεγέθη αυτά θα μας χρειαστούν, όπως θα δούμε στην πορεία, κατά την αξιολόγηση των αλγορίθμων μας.



Σχήμα 4.8: Αλληλεπιδράσεις μεταξύ των τμημάτων του συστήματος

Όσον αφορά το allocation των ways της LLC, στο ένα group αντιστοιχούμε ways ξεκινώντας από τα αριστερά της cache, ενώ στο άλλο από τα δεξιά αυτής. Σε κάθε group θα αντιστοιχεί τουλάχιστον ένα way, ενώ, λόγω περιορισμών του PQOS, στο group που λαμβάνει ways από αριστερά αντιστοιχίζονται τουλάχιστον δύο ways. Επιλέχθηκε αυτό το group να είναι της κρίσιμης υπηρεσίας. Κάθε φορά, λοιπόν, μετατρέπουμε την απόφαση που παίρνουμε από τον πράκτορα σε bitmasks, ξένα μεταξύ τους, και τα εφαρμόζουμε στα δύο επιλεγμένα Class of Services.

4.5.2 Tensorboard

Προκειμένου να παρακολουθήσουμε την εξέλιξη της εκπαίδευσης του νευρωνικού μας δικτύου αλλά και τις hardware και software μετρικές των συνεκτελούμενων εφαρμογών, χρησιμοποιήσαμε το Tensorboard, μια εργαλειοθήκη απεικόνισης που ανέπτυξε η Google για τις ανάγκες της Tensorflow βιβλιοθήκης της. Χάρη σε αυτό, είχαμε άμεσα διαθέσιμη όλη την απαραίτητη πληροφορία που χρειαζόμασταν για την αξιολόγηση των πειραμάτων μας αλλά και τον έγκαιρο εντοπισμό προβλημάτων και λαθών στην υλοποίησή μας.

4.5.3 Latency Calculator

Όπως αναφέρθηκε εν συντομία στην ενότητα 4.3.1, χρειαζόμαστε έναν τρόπο υπολογισμού του 99^{th} percentile latency για κάθε παράθυρο. Για τη λειτουργία αυτή υλοποιήσαμε ένα C++ script το οποίο διαβάζει συνεχώς από τον client, μέσω pipe, τα sojourn times των requests. Όπως εξηγήσαμε ήδη, ενώ η διάρκεια κάθε παραθύρου μέτρησης είναι σταθερή, η στιγμή εκκίνησής του δεν είναι προδιαγεγραμμένη. Αντίθετα, ταυτίζεται με τη στιγμή που ο agent λαμβάνει μια νέα απόφαση. Συνεπώς, κάθε φορά που εκκινεί ένα νέο παράθυρο, θέλουμε να στέλνουμε οδηγία στο script να μηδενίσει τα στατιστικά που έχει καταγράψει μέχρι τώρα και να αρχίσει την καταγραφή νέων στατιστικών (sojourn times) για χρόνο συγκεκριμένης διάρκειας. Υλοποιούμε την ασύγχρονη αυτή επικοινωνία μέσω σήματος (signal) που αποστέλλει ο loader στο script. Στο τέλος του παραθύρου μέτρησης, το script υπολογίζει το latency για τα requests που ανήκουν στο συγκεκριμένο παράθυρο και το επιστρέφει στο περιβάλλον. Θα δούμε αργότερα ότι για την αξιολόγηση των αλγορίθμων μας θα χρειαστούμε, εκτός από το latency για κάθε παράθυρο, και το latency για την πλήρη εκτέλεση της εφαρμογής. Συνεπώς, το script αποθηκεύει, επιπλέον, όλα τα sojourn times που διαβάζει σε ένα διάνυσμα και από αυτά υπολογίζει, στο τέλος της εκτέλεσης, το συνολικό latency. Σημειώνουμε ότι η Tailbench σουίτα, με τον τρόπο που είναι υλοποιημένη, αποθηκεύει και αυτή όλα τα sojourn times, τα οποία και γράφει μαζικά σε ένα binary αρχείο στο τέλος της εκτέλεσης της εφαρμογής. Στη συνέχεια, ένα Python script διαβάζει το αρχείο αυτό και εξάγει το συνολικό latency. Η υλοποίηση αυτή, που περιλαμβάνει γράψιμο σε αρχείο, δεν προκαλεί επιπλέον καθυστερήσεις, αφού γίνεται μόνο μία φορά, στο τέλος της εκτέλεσης. Εμείς, όμως, δε θα μπορούσαμε να την ακολουθήσουμε, καθώς θα χρειαζόταν να γράψουμε σε αρχείο τα στατιστικά του κάθε request ξεχωριστά, κάτι το οποίο θα επιβάρυνε πολύ τον client.

4.5.4 Περιβάλλον (Environment)

Το περιβάλλον συνδέει όλα τα υπόλοιπα τμήματα μεταξύ τους. Αρχικά, εκκινεί τον Tailbench client, ο οποίος ξεκινά να αποστέλλει αιτήσεις στον Tailbench server που τρέχει ήδη στο μηχάνημα. Στη συνέχεια, εκκινεί τις εφαρμογές βέλτιστης προσπάθειας που θα συνεκτελεστούν με την κύρια υπηρεσία. Ανά τακτά χρονικά διαστήματα, το περιβάλλον ελέγχει αν έχει ολοκληρωθεί κάποια εφαρμογή βέλτιστης προσπάθειας και αν ναι, την επανεκτελεί ή εκκινεί μια διαφορετική. Επιπλέον, εκκινεί τον πράκτορα, τον οποίο τροφοδοτεί με την κατάσταση του περιβάλλοντος και λαμβάνει πίσω την επόμενη δράση του. Εφαρμόζει, με χρήση του PQOS, την απόφαση αυτή στο φυσικό μηχάνημα και συλλέγει τις νέες τιμές μετρικών που επηρεάζονται. Έπειτα, σχηματίζει την επόμενη κατάσταση, υπολογίζει την ανταμοιβή του πράκτορα και τα αποστέλλει πίσω σε αυτόν. Όπως ήδη εξηγήσαμε, το περιβάλλον λαμβάνει τις software μετρικές, δηλαδή το latency, από τον loader (client) της LC εφαρμογής μέσω του script Latency Calculator.

4.5.5 Πράκτορας (Agent)

Ο πράκτορας λαμβάνει ως είσοδό του την κατάσταση του περιβάλλοντος και στη συνέχεια εφαρμόζει σε αυτό τις αποφάσεις του. Η προηγούμενη εργασία του εργαστηρίου στην οποία στηριχθήκαμε συμπεριλαμβάνει στην κατάσταση του περιβάλλοντος το πλήθος των misses των BE εφαρμογών κανονικοποιημένο ως προς τον χρόνο (kilo cycles), καθώς και τα ways που είναι διαθέσιμα στις BE εφαρμογές.

Το τι ορίζουμε ως κατάσταση είναι καθοριστικής σημασίας για την απόδοση του πράκτορα. Οι μετρικές (features) που την αποτελούν θα πρέπει να αποτυπώνουν το επίπεδο του ανταγωνισμού που δέχεται η κρίσιμη υπηρεσία από τις εφαρμογές βέλτιστης προσπάθειας, ώστε βάσει αυτού ο πράκτορας να αποφαίνεται για τα ways που πρέπει να διαθέσει στις BEs. Επιπλέον, θα πρέπει να είναι μετρικές που μπορούμε να συλλέξουμε μέσα σε λίγο χρόνο, αφού θέλουμε οι αποφάσεις διαμοιρασμού πόρων να λαμβάνονται ανά τακτά χρονικά διαστήματα.

Οι διακυμάνσεις των misses των best effort εφαρμογών είναι ενδεικτικές των μεταβολών του ανταγωνισμού, για αυτό και η προηγούμενη μελέτη τις επιλέγει ως μέρος της κατάστασης. Τα ways που ανατίθενται στις BEs σε κάθε βήμα είναι επίσης αντιπροσωπευτικά του επιπέδου του ανταγωνισμού για την LLC: μία μεγάλη τιμή misses, όταν έχουν ανατεθεί πολύ λίγα ways στις BEs, μπορεί να συνιστά μικρότερο ανταγωνισμό σε σχέση με μία μικρότερη τιμή misses με πολύ περισσότερα ways.

Στην παρούσα διπλωματική αξιολογούμε επιπλέον αν η επίδοση του συστήματός μας βελτιώνεται αν συμπεριλάβουμε στην κατάσταση του περιβάλλοντος μια μετρική που αφορά την LC εφαρμογή. Κάτι τέτοιο είναι μια διαισθητικά κατανοητή επιλογή, αφού για να αποτυπώσουμε πλήρως το επίπεδο ανταγωνισμού κάθε στιγμή, δεν επαρκεί να λάβουμε υπ' όψιν μόνο μετρικές των BE εφαρμογών, αλλά και της LC, δεδομένου ότι και οι δύο κατηγορίες εφαρμογών περνάνε από διαφορετικές φάσεις, δηλαδή μεταβάλλεται η συμπεριφορά τους στον χρόνο. Στην περίπτωση που η LC εφαρμογή είχε την ίδια ακριβώς συμπεριφορά και εξέλιξη στον χρόνο, τότε πιθανόν να επαρκούσε μια μετρική που αφορά μόνο τις BEs. Κάτι τέτοιο, όμως, δε συμβαίνει στην περίπτωσή μας, αφού όπως είδαμε και στα σχήματα 4.3 και 4.6, διαφορετικά παράθυρα της

LC εμφανίζουν διαφορετικό latency.

Για την υλοποίηση του πράκτορα, η προηγούμενη εργασία χρησιμοποιεί τον αλγόριθμο Dueling DQN, που αποτελεί βελτιστοποίηση του DQN. Το νευρωνικό δίκτυο του αλγορίθμου αποτελείται από δύο κρυφά επίπεδα και ως συνάρτηση ενεργοποίησης των νευρώνων χρησιμοποιείται η ELU [2]. Ως συνάρτηση απωλειών χρησιμοποιείται το μέσο τετραγωνικό σφάλμα και, τέλος, ως optimizer η μέθοδος Adam [10]. Εκτός από αυτόν τον αλγόριθμο, με τις δύο διαφορετικές εισόδους που αναφέρουμε, υλοποιούμε κι έναν πράκτορα Actor-Critic. Η συγκριτική αξιολόγηση των τριών αυτών μοντέλων παρουσιάζεται στην ενότητα 5.4.

Όσον αφορά τη συνάρτηση ανταμοιβής του πράκτορα, αυτή πρέπει να αποτυπώνει το trade-off που καλούμαστε να πετύχουμε: από τη μία πλευρά προσπαθούμε να προστατεύσουμε την επίδοση της υπηρεσίας και από την άλλη να αυξήσουμε τη χρησιμοποίηση του μηχανήματος εκτελώντας παράλληλα, όσο καλύτερα γίνεται, εφαρμογές χαμηλής προτεραιότητας. Μια συνάρτηση ανταμοιβής που εκφράζει τον παραπάνω διττό στόχο είναι η:

$$R = \begin{cases} \text{num of ways allocated to best effort apps} & \text{if latency} < \text{threshold} \\ -\text{penalty coefficient} \cdot \max \text{ ways} & \text{otherwise} \end{cases} \tag{4.1}$$

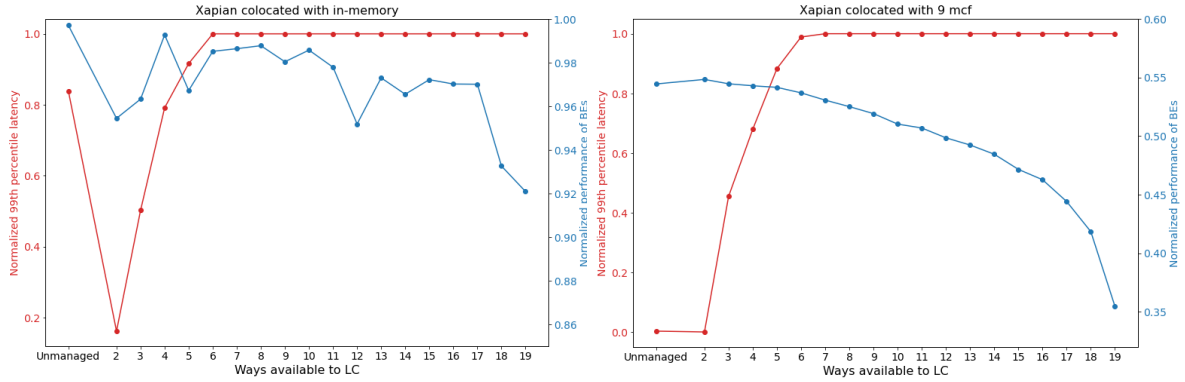
Με βάση αυτή, αν ο πράκτορας διατηρεί τον QoS στόχο του, λαμβάνει μια θετική ανταμοιβή ανάλογη του πλήθους των ways που διέθεσε στις εφαρμογές βέλτιστης προσπάθειας. Αντίθετα, αν παραβιάσει τον QoS στόχο του, τιμωρείται με μια αρνητική ανταμοιβή ίση με το μέγιστο πλήθος ways προς διάθεση επί κάποιον συντελεστή (penalty coefficient). Ο συντελεστής αυτός καθορίζει την ανοχή του πράκτορα σε παραβιάσεις του ορίου του latency. Μια ποιοτική μελέτη της επίδρασης αυτής της παραμέτρου παρουσιάζεται στην ενότητα 5.2.1.

Κεφάλαιο 5

Πειραματική μελέτη

5.1 Στατικός διαμοιρασμός της LLC

Στο σχήμα 5.1 παρουσιάζουμε, ενδεικτικά, κάποιες στατικές αναθέσεις των ways της LLC μεταξύ της Xapian και δύο διαφορετικών BE εφαρμογών. Με κόκκινο χρώμα παρουσιάζεται το κανονικοποιημένο latency, δηλαδή το target latency της εφαρμογής προς το latency που μετράμε κατά τη συνεκτέλεση. Με μπλε χρώμα απεικονίζεται η κανονικοποιημένη επίδοση του συνόλου των BEs, η οποία έχει υπολογιστεί ως ο αρμονικός μέσος των IPC των BEs κατά τη συνεκτέλεση, προς τον αρμονικό μέσο των IPC τους όταν το καθένα από αυτά εκτελείται μόνο του στο σύστημα. Και τα δύο μεγέθη έχουν βέλτιστη τιμή το 1 και χείριστη το 0.



(a) Xapian, in-memory

(b) Xapian, 9 mcf

Σχήμα 5.1: Xapian: στατικές αναθέσεις ways κατά τη συνεκτέλεση

Παρατηρούμε ότι στην κατάσταση Unmanaged, δηλαδή τη συνεκτέλεση χωρίς κανένα διαμοιρασμό της LLC, η συνεκτέλεση της Xapian με την in-memory δίνει πολύ καλύτερη διατήρηση του QoS στόχου σε σχέση με τη συνεκτέλεσή της με τις 9 mcf. Από αυτή την ανάλυση επαληθεύουμε ότι κάποιες BE εφαρμογές είναι περισσότερο απαιτητικές από άλλες ως προς τις ανάγκες τους για την LLC. Η συνεκτέλεση μιας LC εφαρμογής με BEs διαφορετικών χαρακτηριστικών είναι, στο πλαίσιο των πειραμάτων μας, ιδιαίτερα βοηθητική, αφού ενισχύει τη γενικότητα των μεθόδων μας.

Επιπλέον, από τις γραφικές παραστάσεις μπορούμε να επιβεβαιώσουμε ότι, ακόμα και κατά τη

συνεκτέλεση, η LC εφαρμογή απαιτεί ένα υποσύνολο μόνο των ways της LLC προκειμένου να διατηρήσει το latency της πολύ κοντά στο QoS target. Στην αντίθετη περίπτωση, η BE εφαρμογή θα αποδεικνυόταν τόσο απαιτητική που θα αναγκαζόμασταν να την απορρίψουμε, όπως ήδη αναφέραμε ότι κάναμε με το libquantum.

Χοντρικά, για τη συνεκτέλεση της Xapian με την in-memory, φαίνεται ότι μια πολύ καλή στατική ανάθεση για τη διατήρηση της ποιότητας της υπηρεσίας και την ταυτόχρονη επίτευξη υψηλού utilization είναι τα 6 ways στον server και τα υπόλοιπα 14 στην BE. Φυσικά, η εξαγωγή μιας τέτοιας πληροφορίας για κάθε συνδυασμό απο LC και BEs ως λύση στο πρόβλημα της συνεκτέλεσης θα ήταν ανέφικτη. Επιπροσθέτως, οι μετρικές που φαίνονται στα συγκεκριμένα διαγράμματα αφορούν το σύνολο του πειράματός μας και δεν αποτυπώνουν τις πιθανές μεταβολές του latency και του performance κατά τη διάρκεια της συνεκτέλεσης. Κατά τη διάρκεια ενός πειράματός μας, είναι πιθανό να υπάρξουν απρόβλεπτες αυξομοιώσεις του ανταγωνισμού, στις οποίες πρέπει να είμαστε σε θέση να ανταποκριθούμε στιγμιαία. Για να επιτευχθεί ο συγκεκριμένος στόχος απαιτείται ένας μηχανισμός δυναμικού διαμοιρασμού της LLC.

5.2 Επιλογή Παραμέτρων

Η κατάλληλη επιλογή παραμέτρων του περιβάλλοντος και του πράκτορα αποτελεί μια ιδιαίτερα χρονοβόρα διαδικασία. Πειραματιστήκαμε με πολλές διαφορετικές τιμές παραμέτρων, κάποιες από τις οποίες είχαν καθοριστική επίδραση στην ποιότητα των αποτελεσμάτων μας, ενώ άλλες όχι τόσο σημαντική. Παραθέτουμε πίνακες με τις τιμές παραμέτρων περιβάλλοντος (Πίνακας 5.1) και πράκτορα (Πίνακας 5.2) στις οποίες καταλήξαμε.

Παράμετροι Περιβάλλοντος	**Τιμές**
cores allocated to LC	0
cores allocated to BEs	1-9
percentile	99^{th}
penalty coefficient	4
time interval	200 msec
latency target [Xapian / Img-dnn]	25 / 5.6 msec
requests per second [Xapian / Img-dnn]	800 / 500

Πίνακας 5.1: Παράμετροι Περιβάλλοντος

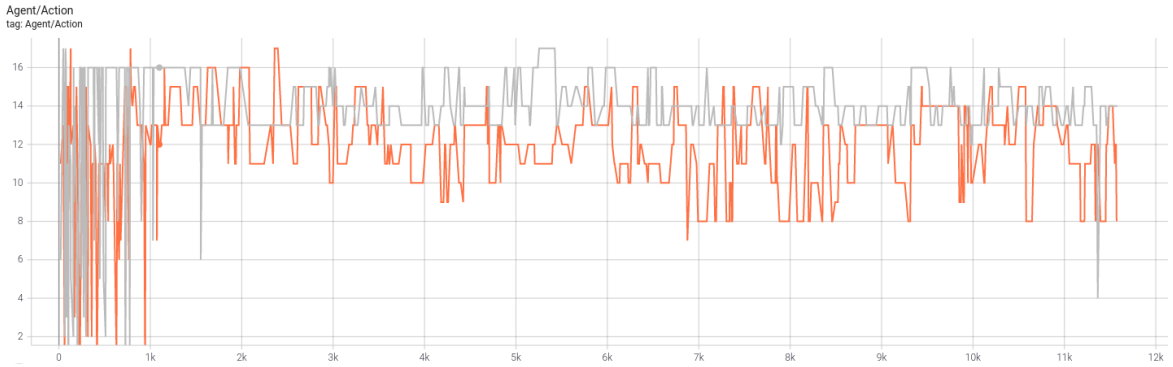
Παράμετροι Πράκτορα	**Τιμές**
hidden layers	2
layers dimensions	[24, 48]
input	[Bandwidth$_{be}$, Bandwidth$_{lc}$, ways to BEs]
output (ways to BEs)	1-18
network architecture	Dueling
algorithm	Double DQN
learning rate	0.001
gamma	0.99
epsilon start	1
epsilon decay	0.002
epsilon end	0

Πίνακας 5.2: Παράμετροι Πράκτορα

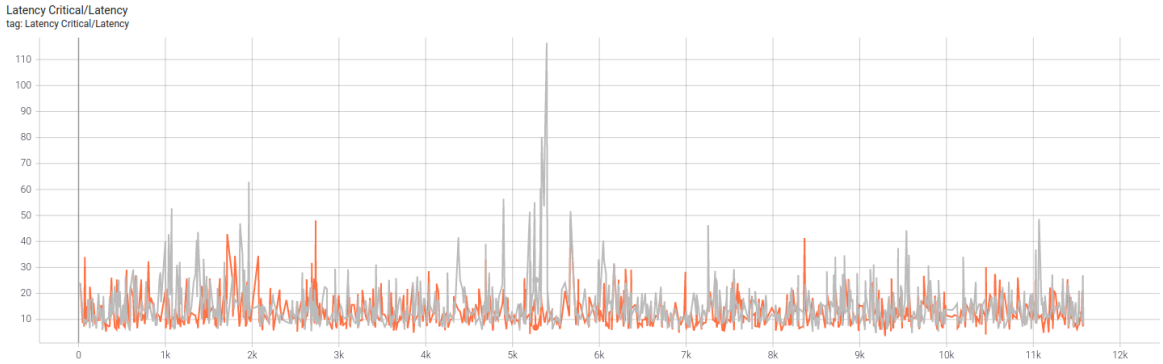
Στη συνέχεια, κρίνουμε σκόπιμο να εξηγήσουμε την επίδραση δύο παραμέτρων που είχαν ιδιαίτερα σημαντική επίδραση στην απόδοση των μοντέλων μας.

5.2.1 Ποιοτική μελέτη της παραμέτρου penalty coefficient

Η παράμετρος penalty coefficient της συνάρτησης ανταμοιβής του πράκτορα επηρεάζει την ανοχή του στις παραβιάσεις του QoS και σταθμίζει τις αποφάσεις του μεταξύ βραχυπρόθεσμων και μακροπροθέσμων ανταμοιβών. Μεγαλύτερη τιμή της παραμέτρου υποδηλώνει μικρότερη ανοχή στις παραβιάσεις του QoS. Καθώς στην παρούσα εργασία ο πρωταρχικός μας στόχος είναι η διατήρηση του QoS target και δεδομένου ότι ο έλεγχος του 99$^{\text{th}}$ percentile latency είναι ένας πολύ απαιτητικός στόχος για την ποιότητα της υπηρεσίας, επιλέγουμε μια αρκετά υψηλή τιμή για τη συγκεκριμένη παράμετρο. Στα σχήματα 5.2 φαίνονται οι αποφάσεις του πράκτορα και το latency της Xapian κατά τη συνεκτέλεσή της με την in-memory, για δύο διαφορετικές τιμές του penalty coefficient, όπως αποτυπώνονται από το Tensorboard. Με γκρι χρώμα φαίνονται οι τιμές για penalty coefficient ίσο με 2, ενώ με πορτοκαλί χρώμα για penalty coefficient ίσο με 4.



(a) Actions of agent



(b) Latency of LC

Σχήμα 5.2: Επίδραση της μετρικής penalty coefficient

Παρατηρούμε ότι για penalty coefficient ίσο με 4, ο πράκτορας παίρνει πιο «συντηρητικές» αποφάσεις, αναθέτει δηλαδή λιγότερα ways στην BE εφαρμογή, κάτι το οποίο οδηγεί σε πιο αποτελεσματικό έλεγχο του latency. Μάλιστα, η υπεροχή της μεγαλύτερης τιμής penalty coefficient γίνεται εμφανής περί τα 5.2K βήματα του πράκτορα, αφού η απόφαση του agent να παραχωρήσει μόλις 11 ways στην BE, αντί για 17 στην περίπτωση του penalty coefficient ίσο με 2, πετυχαίνει την αποτροπή ενός πολύ έντονου spike στο latency, το οποίο δεν αντιμετωπίζεται για penalty coefficient ίσο με 2.

5.2.2 Ποιοτική μελέτη της παραμέτρου ϵ_{decay}

Όσον αφορά το δίλημμα εξερεύνησης-εκμετάλλευσης, ο αλγόριθμος DDQN χρησιμοποιεί την ϵ-greedy πολιτική που αναφέραμε στην ενότητα 3.5, με τη διαφορά ότι σε κάθε βήμα ελαττώνουμε εκθετικά την πιθανότητα με την οποία εξερευνούμε (Decaying ϵ-greedy policy):

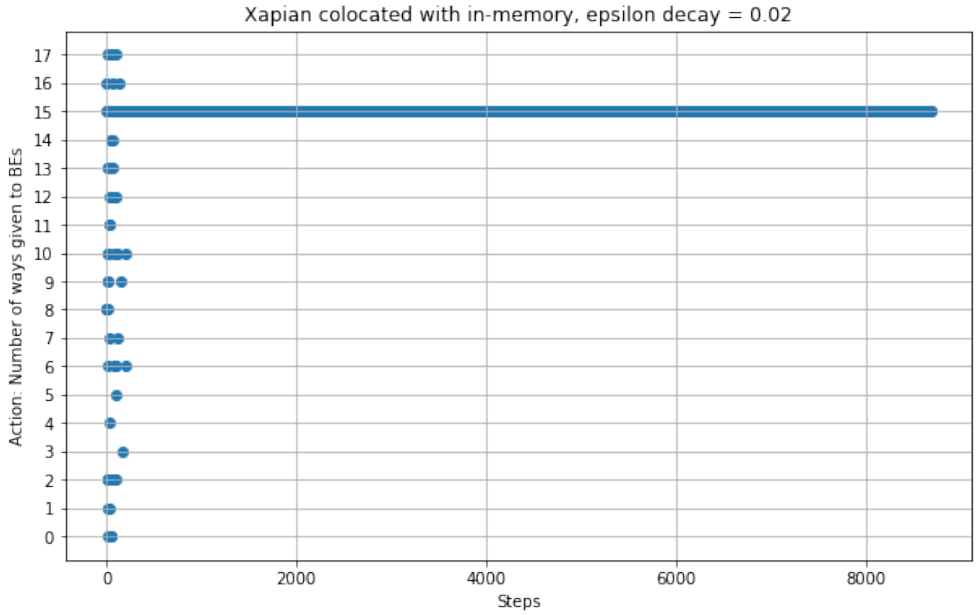
$$\epsilon = \epsilon_{end} + (\epsilon_{start} - \epsilon_{end}) \cdot \epsilon^{-steps \cdot \epsilon_{decay}}$$

Στην αρχή, λοιπόν, η επιλογή των δράσεων γίνεται εντελώς τυχαία ($\epsilon_{start} = 1$), ενώ προοδευτικά βασιζόμαστε όλο και περισσότερο στη γνώση που έχουμε συσσωρεύσει, έως ότου, τελικά, να επιλέγουμε αποκλειστικά με βάση αυτή ($\epsilon_{end} = 0$). Ο ρυθμός με τον οποίο μειώνουμε το ποσοστό εξερεύνησης ρυθμίζεται από τη μετρική ϵ_{decay}. Η τιμή της παραμέτρου αυτής επιλέχθηκε κατάλληλα ώστε, στον περιορισμένο χρόνο που διαρκούν τα πειράματά μας, ο πράκτορας από τη μία να εξερευνεί επαρκώς τους διαφορετικούς συνδυασμούς εισόδων-αποφάσεων κι από την άλλη να προλαβαίνει να εκμεταλλευτεί τη γνώση που αποκτά, ώστε να φαίνεται η επίδραση της λειτουργίας του στο τελικό αποτέλεσμα.

Στις παρακάτω γραφικές (Σχήμα 5.3) φαίνεται μια ποιοτική ερμηνεία της επίδρασης που έχουν δύο σημαντικά διαφορετικές τιμές του ϵ_{decay}. Στον οριζόντιο άξονα αποτυπώνεται το πλήθος των βημάτων του πράκτορα, ενώ στον κατακόρυφο άξονα η εκάστοτε απόφαση του πράκτορα για το πλήθος των ways που θα ανατεθούν στις BEs.

Στην πρώτη περίπτωση, όπου το ϵ_{decay} ισούται με 0.02, η εξερεύνηση ολοκληρώνεται ήδη από το step 266. Παρατηρούμε ότι έκτοτε ο πράκτορας διατηρεί συνεχώς την ίδια απόφαση. Το φαινόμενο αυτό μπορεί να ερμηνευτεί ως αποτέλεσμα της μη επαρκούς εξερεύνησης: ο πράκτορας έχει αξιολογήσει έναν περιορισμένο αριθμό συνδυασμών εισόδων-αποφάσεων, με αποτέλεσμα να επιλέγει ως βέλτιστη, για κάθε είσοδο, μία από τις λίγες αποφάσεις που έχει εξερευνήσει. Πιθανότατα, η απόφαση αυτή δεν είναι η βέλτιστη για κάθε χρονική στιγμή και, συνεπώς, ο πράκτορας αδυνατεί να προσαρμόσει δυναμικά τις αποφάσεις του στις διαφορετικές εισόδους που συναντάει.

Αντίθετα, στη δεύτερη εικόνα, όπου έχουμε ϵ_{decay} ίσο με 0.002 και η εξερεύνηση ολοκληρώνεται στο βήμα 2651, ο πράκτορας έχει εξερευνήσει περισσότερο τον χώρο των εισόδων-αποφάσεων, πράγμα που του επιτρέπει να επιλέγει διαφορετικές αποφάσεις στον χρόνο. Σημειώνουμε πως, από την άλλη, η τιμή του ϵ_{decay} δεν πρέπει να είναι υπερβολικά μικρή, γιατί έτσι ο πράκτορας θα παίρνει αποφάσεις με βάση την τύχη, και όχι τη γνώση που αποκτά, για μεγάλο ποσοστό της διάρκειας του πειράματός μας, με αποτέλεσμα να μη γίνεται εμφανής η επίδρασή του στο τελικό αποτέλεσμα.



(a) $\epsilon_{decay} = 0.02$

(b) $\epsilon_{decay} = 0.002$

Σχήμα 5.3: Επίδραση της μετρικής epsilon decay (ϵ_{decay})

5.3 Μετρικές αξιολόγησης

Το κανονικοποιημένο latency και η κανονικοποιημένη επίδοση των BEs που είδαμε στην ενότητα 5.1 αξιολογούν την απόδοση του πράκτορα μόνο ως προς έναν από τους δύο στόχους του. Για να ποσοτικοποιήσουμε, συνεπώς, τον βαθμό στον οποίο ο πράκτορας καταφέρνει να ανταποκριθεί και στους δύο στόχους του ταυτόχρονα, χρησιμοποιούμε το συνδυαστικό μέτρο SUCI [20], αφού το προσαρμόσουμε κατάλληλα στο πρόβλημά μας. Χρησιμοποιούμε, λοιπόν, την εξής μετρική:

$$SUCI = c_{SLO} \cdot EFU^{\lambda} \tag{5.1}$$

Όπου:

$$c_{SLO} = \frac{\%\text{ violations of LC alone}}{\%\text{ violations of LC in colocation}} \tag{5.2}$$

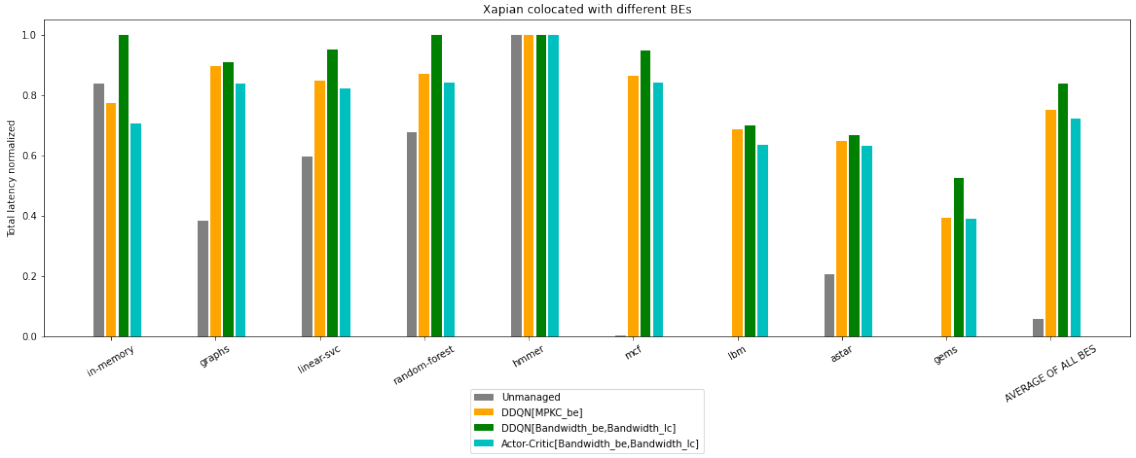
$$EFU = \frac{n}{\frac{latency^{LC}_{colocation}}{latency^{LC}_{alone}} + \sum_{i=0}^{n-1} \frac{IPC^{BE_i}_{alone}}{IPC^{BE_i}_{colocation}}} \tag{5.3}$$

Η μετρική αυτή δίνει έμφαση στη διατήρηση του QoS, άρα στο ποσοστό violations των παραθύρων, ενώ ταυτόχρονα επιβραβεύει την υψηλή χρησιμοποίηση του server, άρα τα υψηλά IPC των BEs και το χαμηλό συνολικό latency της LC. Όσο μεγαλύτερη η τιμή του λ, τόσο περισσότερη έμφαση δίνεται στο utilization του server.

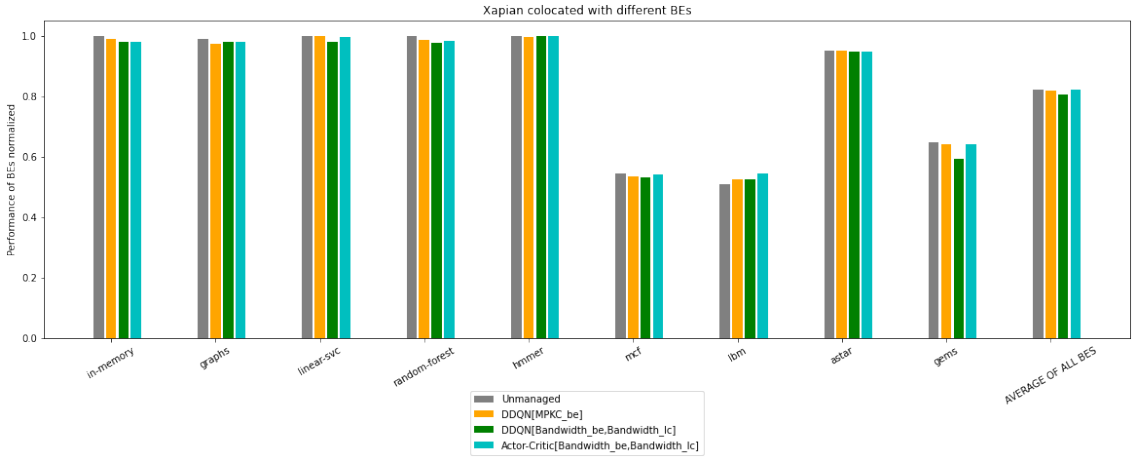
5.4 Πειραματικά αποτελέσματα

Στην υποενότητα αυτή συγκρίνουμε την απόδοση των τριών διαφορετικών μοντέλων που υλοποιήσαμε καθώς και της μη ελεγχόμενης συνεκτέλεσης (Unmanaged), για τις συνεκτελέσεις των δύο LC εφαρμογών με όλες τις BEs. Τα αποτελέσματα παρουσιάζονται ως προς όλες τις προαναφερθείσες μετρικές: το κανονικοποιημένο συνολικό latency της LC εφαρμογής, την κανονικοποιημένη επίδοση των BE εφαρμογών και τη συνδυαστική μετρική SUCI για λ ίσο με 0.5, 1 και 2. Σημειώνεται ότι η Img-dnn δεν μπορεί να ανταποκριθεί στην πίεση που ασκούν τα BEs lbm και gems, οπότε δεν τα συμπεριλαμβάνουμε στην ανάλυσή της. Στη δεξιότερη θέση του κάθε γραφήματος φαίνονται τα συγκεντρωτικά αποτελέσματα που προκύπτουν ως ο γεωμετικός μέσος των αποτελεσμάτων για τις επιμέρους εφαρμογές βέλτιστης προσπάθειας.

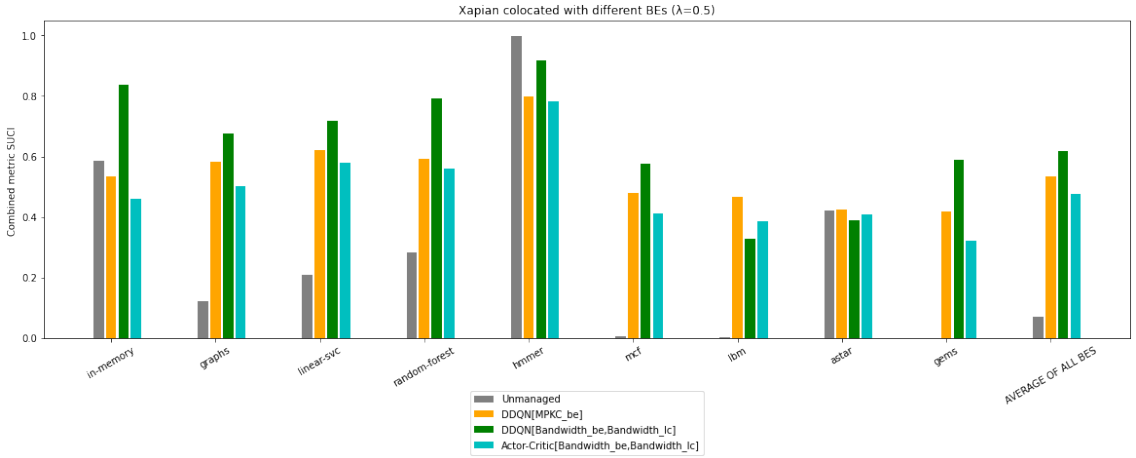
5.4.1 Αποτελέσματα για τις διαφορετικές συνεκτελέσεις της Xapian



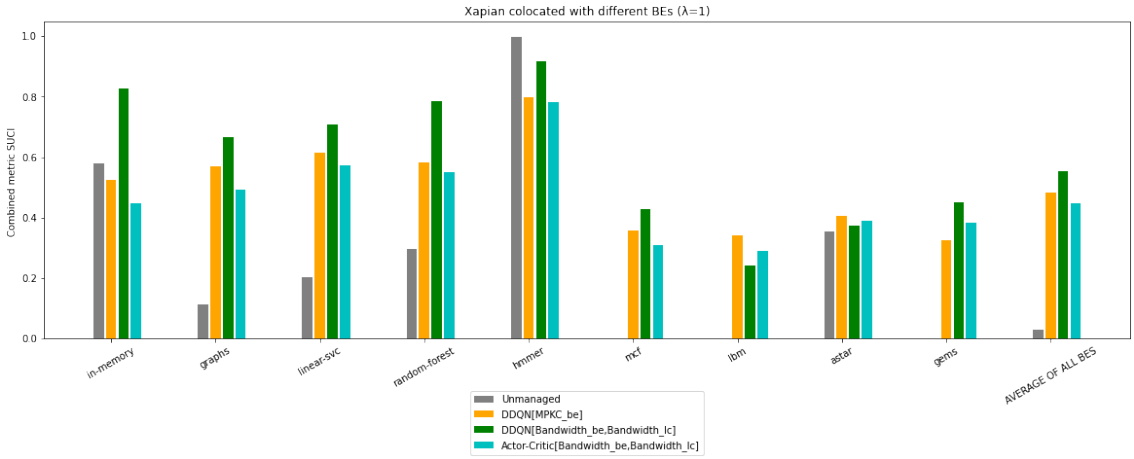
(a) Xapian: Total latency normalized



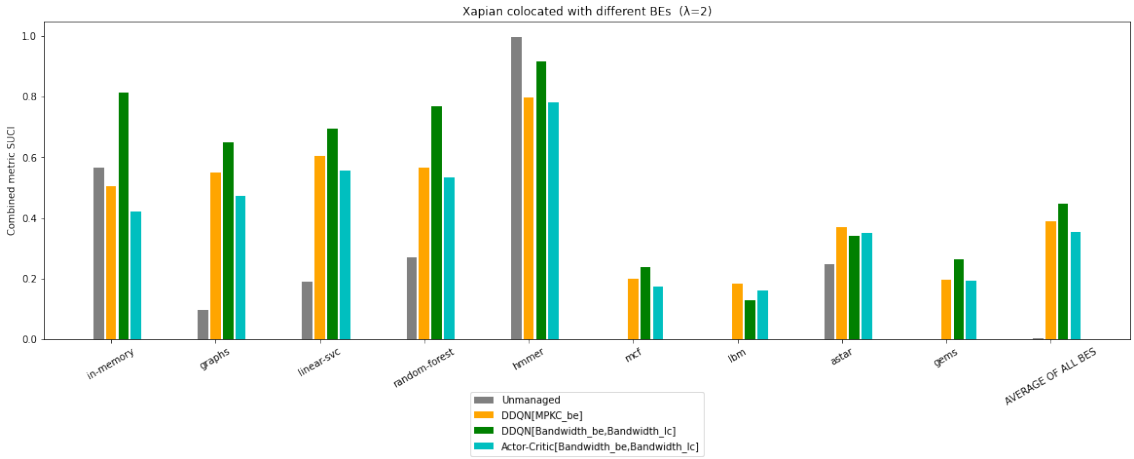
(b) Xapian: Performance of BEs normalized



(c) Xapian: SUCI for $\lambda = 0.5$



(d) Xapian: SUCI for $\lambda = 1$

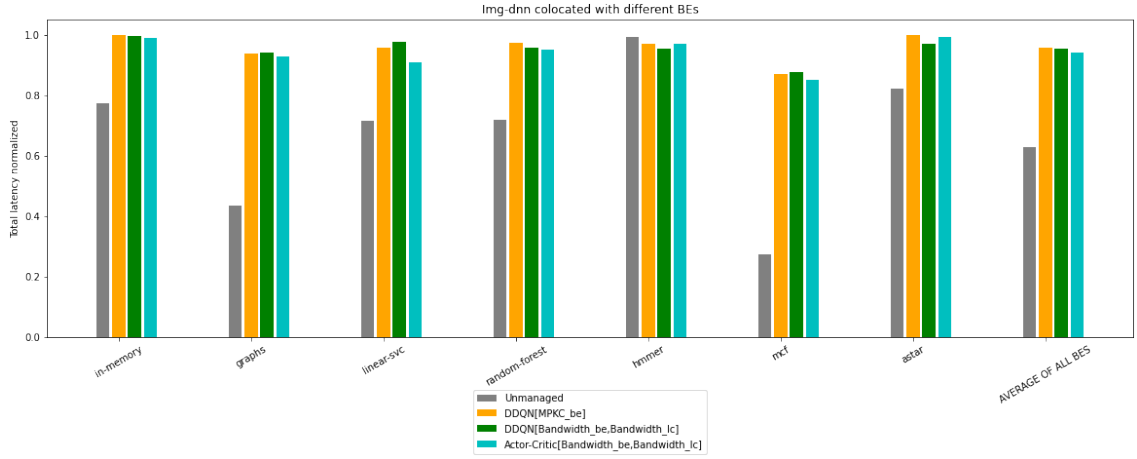


(e) Xapian: SUCI for $\lambda = 2$

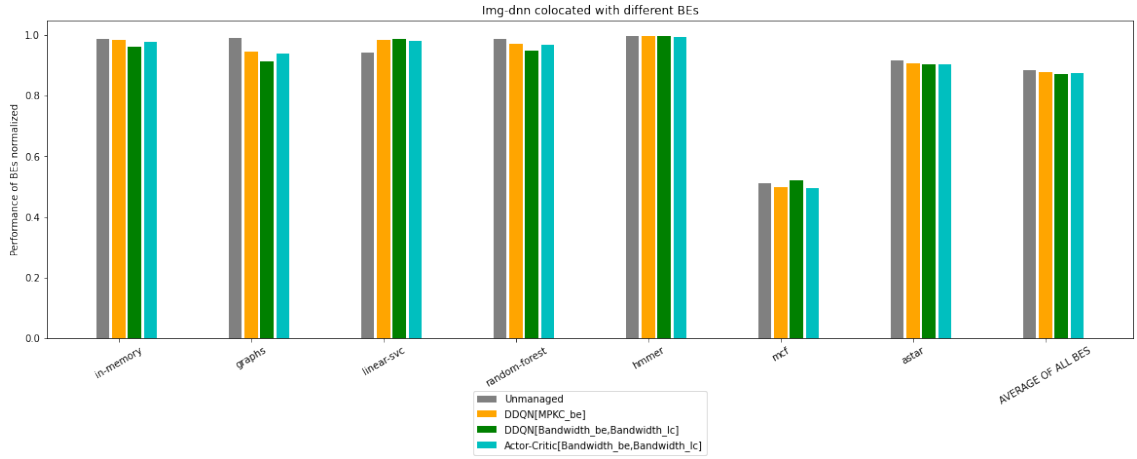
Σχήμα 5.4: Xapian: Αξιολόγηση των πρακτόρων για κάθε συνεκτέλεση

Παρατηρούμε ότι το μοντέλο DDQN με εισόδους το Bandwidth των BEs, το Bandwidth της LC και το πλήθος των ways που ανατίθενται στις BEs εμφανίζει, κατά μέσο όρο, την καλύτερη απόδοση ως προς όλες τις μετρικές, εκτός από το performance των BEs, ως προς το οποίο παρουσιάζει μικρή υστέρηση.

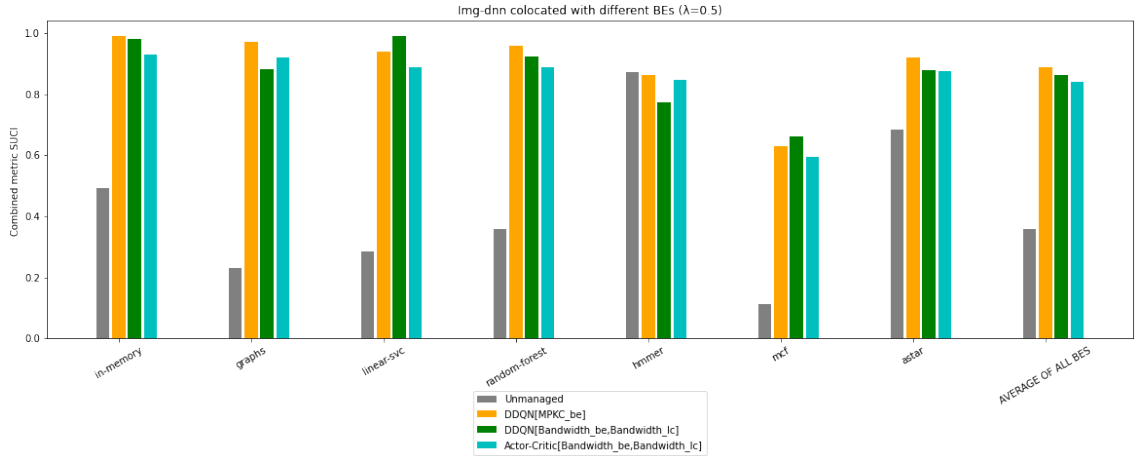
5.4.2 Αποτελέσματα για τις διαφορετικές συνεκτελέσεις της Img-dnn



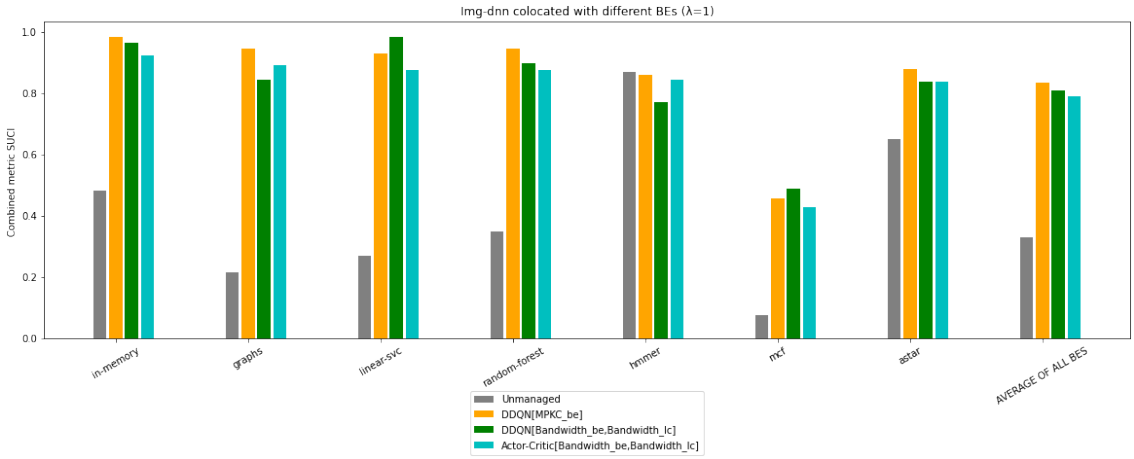
(a) Img-dnn: Total latency normalized



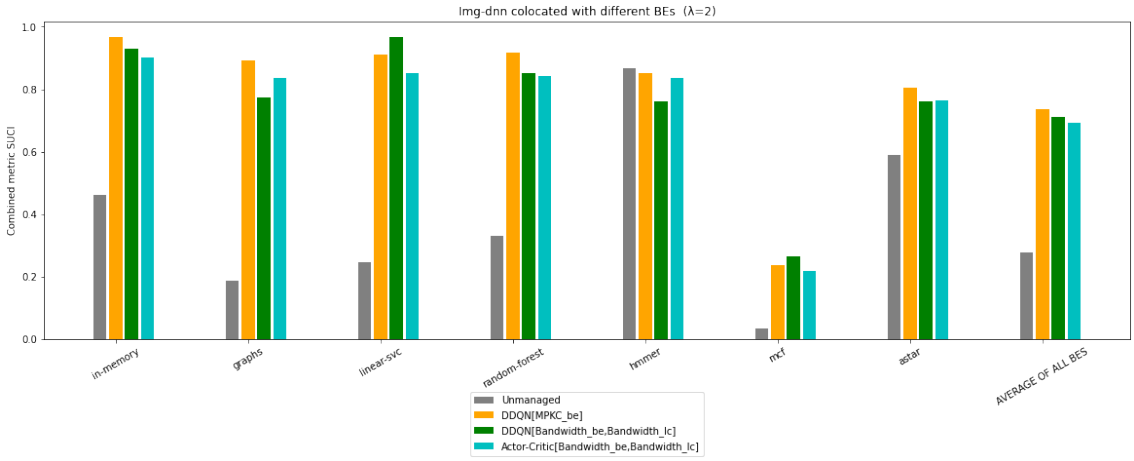
(b) Img-dnn: Performance of BEs normalized



(c) Img-dnn: SUCI for $\lambda = 0.5$



(d) Img-dnn: SUCI for $\lambda = 1$



(e) Img-dnn: SUCI for $\lambda = 2$

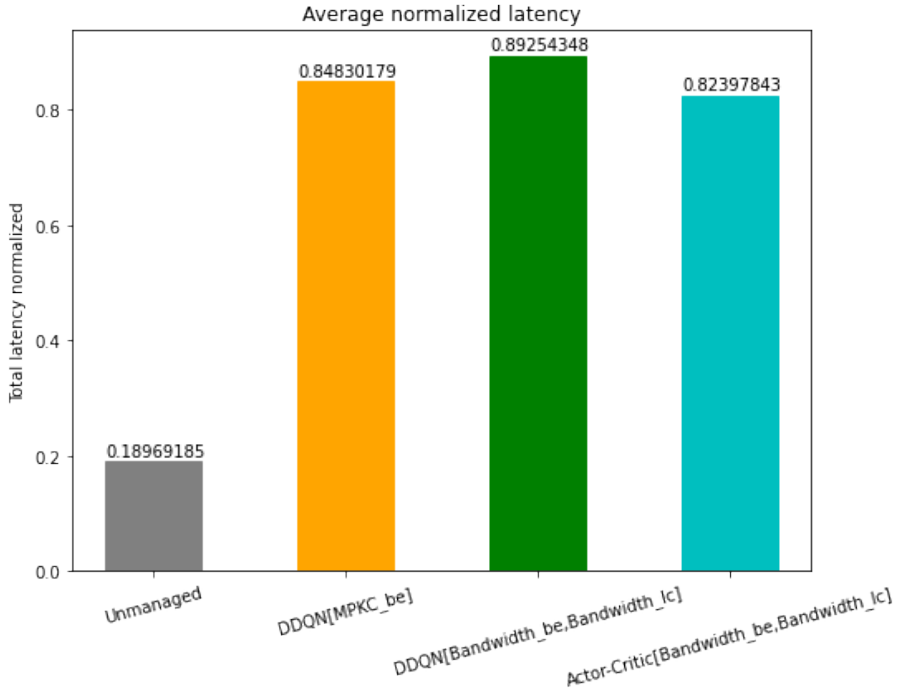
Σχήμα 5.5: Img-dnn: Αξιολόγηση των πρακτόρων για κάθε συνεκτέλεση

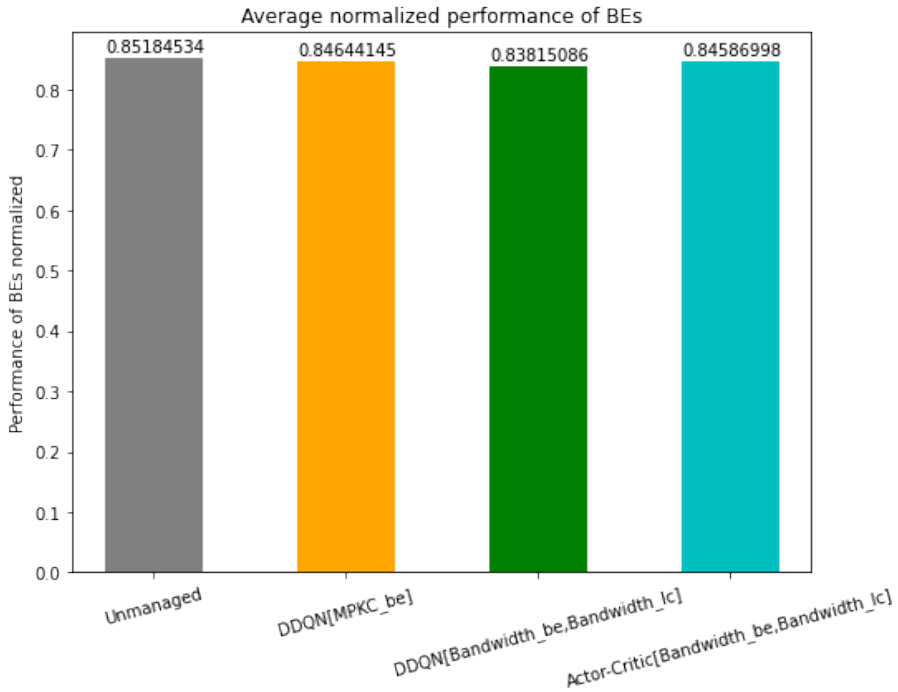
Για τις συνεκτελέσεις της Img-dnn, τα καλύτερα αποτελέσματα δίνει το μοντέλο DDQN με είσοδο τα Misses Per Kilo Cycles των BEs και το πλήθος των ways που αναθέτουμε στις BEs. Με μικρή διαφορά ακολουθεί το δεύτερο DDQN μοντέλο. Σημειώνουμε ότι οι αποκλίσεις μεταξύ των διαφορετικών LC εφαρμογών ως προς την απόδοση των μοντέλων και τις τιμές των βέλτιστων υπερπαραμέτρων για το καθένα είναι αναμενόμενες, δεδομένων των διαφορετικών χαρακτηριστικών και προτύπων που διέπουν την κάθε εφαρμογή. Ενθαρρυντικό, παρ' όλα αυτά, είναι ότι οι αποκλίσεις αυτές είναι σχετικά μικρές, πράγμα που μας επιτρέπει να επιλέξουμε ένα μοντέλο με σταθερές τιμές παραμέτρων και υπερπαραμέτρων τέτοιο ώστε να έχει ικανοποιητική επίδοση για το σύνολο των συνεκτελέσεών μας.

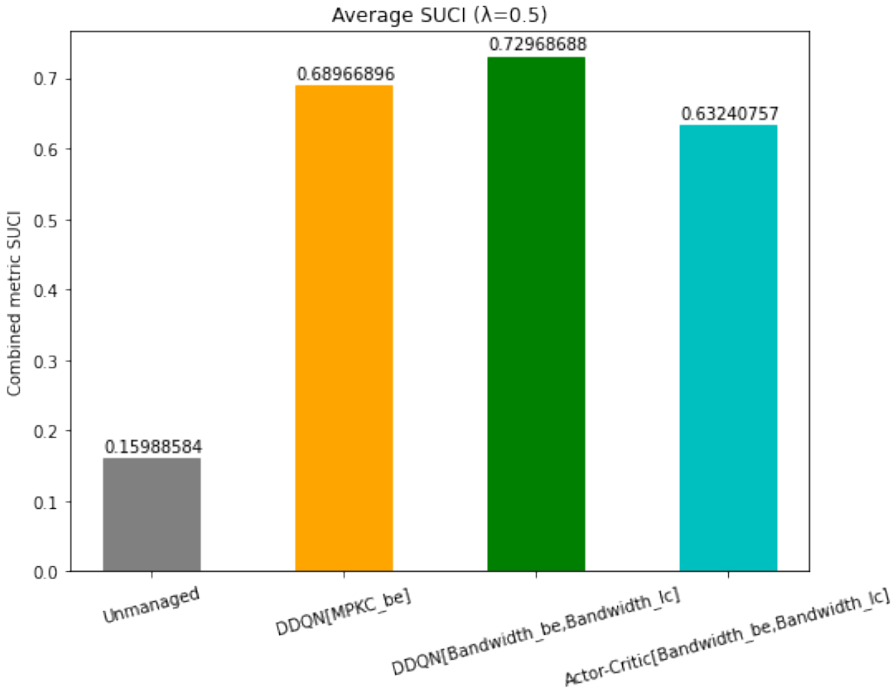
Στη συνέχεια παραθέτουμε την απόδοση των μοντέλων μας και για τις δύο LC εφαρμογές συν-

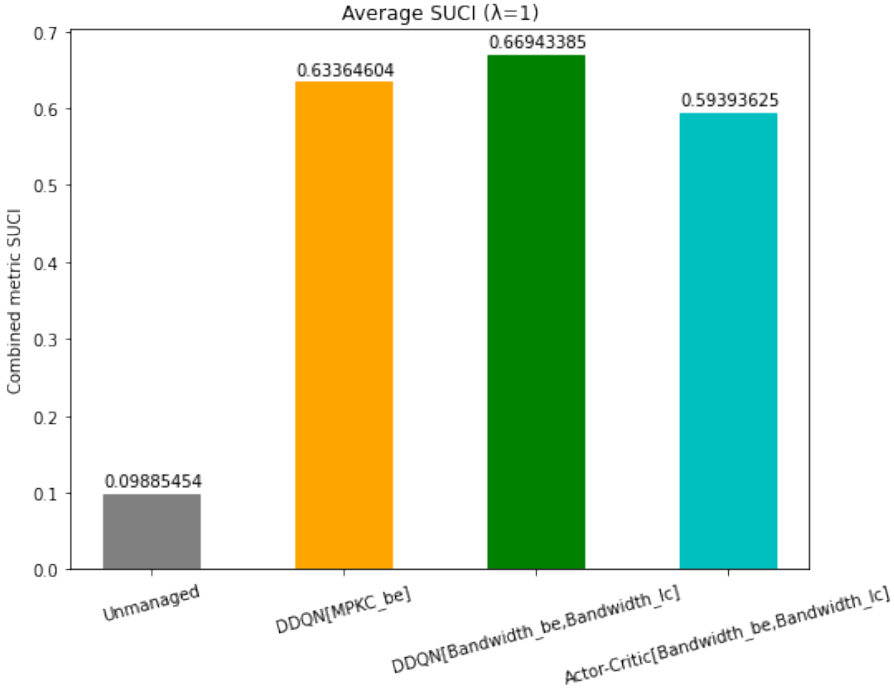
δυαστικά, η οποία υπολογίζεται ως ο γεωμετρικός μέσος των μέσων αποδόσεων της καθεμίας.

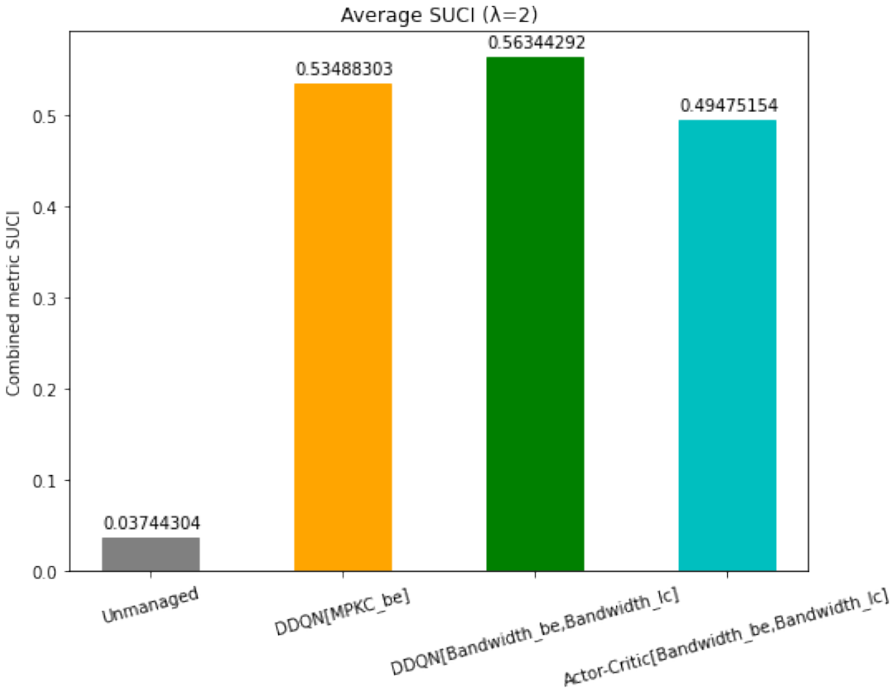
5.4.3 Αποτελέσματα για το σύνολο των διαφορετικών συνεκτελέσεων






Average SUCI (λ=0.5)
Combined metric SUCI
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.15988584
0.68966896
0.72968688
0.63240757
Unmanaged
DDQN[MPKC_be]
DDQN[Bandwidth_be,Bandwidth_lc]
Actor-Critic[Bandwidth_be,Bandwidth_lc]


Average SUCI (λ=1)
Combined metric SUCI
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.09885454
0.63364604
0.66943385
0.59393625
Unmanaged
DDQN[MPKC_be]
DDQN[Bandwidth_be,Bandwidth_lc]
Actor-Critic[Bandwidth_be,Bandwidth_lc]



Σχήμα 5.6: Αποτελέσματα για το σύνολο των συνεκτελέσεων

Συνολικά, το δεύτερο DDQN μοντέλο (DDQN[Bandwidth$_{be}$,Bandwidth$_{lc}$]) δίνει καλύτερα αποτελέσματα ως προς το latency της LC και τις συνδυαστικές μετρικές, ενώ είναι ελαφρώς χειρότερο ως προς το performance των BEs. Για τους σκοπούς της παρούσας εργασίας, λοιπόν, όπου η έμφαση δίνεται στη διατήρηση του QoS της LC εφαρμογής, το συγκεκριμένο μοντέλο επιλέγεται ως το καλύτερο. Επισημαίνουμε, για ακόμα μια φορά, ότι όσο μεγαλύτερο είναι το λ της συνδυαστικής μετρικής, τόσο πιο πολύ τιμωρούμε τον πράκτορα για την υστέρηση στο performance των BEs, για αυτό και η τιμή του SUCI φθίνει με την αύξηση του λ.

Τα συγκεντρωτικά αποτελέσματα για το βέλτιστο αυτό μοντέλο και η σύγκρισή του με τη μη ελεγχόμενη συνεκτέλεση φαίνονται και στον πίνακα 5.3.

	Latency	BE Perf.	SUCI $\lambda = 0.5$	SUCI $\lambda = 1$	SUCI $\lambda = 2$
DDQN[Bandwidth$_{be}$, Bandwidth$_{lc}$]	0.89254	0.83815	0.72969	0.66943	0.56344
Unmanaged	0.18969	0.85185	0.15989	0.09886	0.03744
% change	+370.523%	-1.608%	+356.380%	+577.191%	+1404.800%

Πίνακας 5.3: Σύγκριση βέλτιστου μοντέλου και μη ελεγχόμενης συνεκτέλεσης

Συνεπώς, με χρήση του βέλτιστου μοντέλου πετυχαίνουμε να διατηρήσουμε το QoS των δύο LC εφαρμογών κατά 89.254% στο σύνολο των συνεκτελέσεών τους. Το ποσοστό αυτό συνιστά βελτίωση +370%, περίπου, σε σχέση με τη μη ελεγχόμενη συνεκτέλεση. Πολύ μεγάλη είναι, επίσης, η αντίστοιχη βελτίωση για τις συνδυαστικές μετρικές, ενώ η επιβάρυνση του BE Performance είναι μόλις 1.608%.

Κεφάλαιο 6

Επίλογος

6.1 Σύνοψη και Συμπεράσματα

Στην παρούσα διπλωματική εργασία αξιοποιούμε τεχνικές βαθιάς ενισχυτικής μάθησης με σκοπό να προστατεύσουμε την ποιότητα της υπηρεσίας κρίσιμης απόκρισης και, ταυτόχρονα, να αυξήσουμε τη χρησιμοποίηση του επεξεργαστή, διαθέτοντας πόρους σε εφαρμογές βέλτιστης προσπάθειας.

Ο μηχανισμός που χρησιμοποιούμε βασίζεται στις Intel τεχνολογίες επίβλεψης και καταμερισμού της κρυφής μνήμης τελευταίου επιπέδου και, λαμβάνοντας υπ' όψιν μετρικές υλικού και λογισμικού, αποφασίζει δυναμικά για τον διαμοιρασμό της LLC.

Συμπερασματικά, η βαθιά ενισχυτική μάθηση αποτελεί ένα υποσχόμενο εργαλείο για την αντιμετώπιση του εν λόγω προβλήματος. Σημαντικό είναι πως για την υλοποίηση του μηχανισμού δεν απαιτείται καθόλου profiling των εφαρμογών που θα συνεκτελεστούν, παρά μόνο ο προσδιορισμός των QoS απαιτήσεων των LC εφαρμογών. Δείξαμε ότι ο πράκτορας ενισχυτικής μάθησης μαθαίνει αυτόματα τα πρότυπα που διέπουν τις διαφορετικές συνεκτελέσεις και δρα βάσει αυτών. Το μοντέλο μας ελέγχθηκε σε συνεκτελέσεις δύο εφαρμογών κρίσιμης απόκρισης της σουίτας Tailbench με ένα πλήθος από εφαρμογές βέλτιστης προσπάθειας. Παρόλο που οι υπηρεσίες κρίσιμης απόκρισης αλλά και οι εφαρμογές βέλτιστης προσπάθειας που χρησιμοποιήσαμε έχουν διαφορετικά χαρακτηριστικά και απαιτήσεις, το μοντέλο μας ανταποκρίνεται ικανοποιητικά στα διαφορετικά σενάρια ανταγωνισμού, χωρίς να απαιτείται προσαρμογή των παραμέτρων του.

Αποδείξαμε ότι είναι εφικτή η διατήρηση του επιπέδου λειτουργίας των υπηρεσιών κρίσιμης απόκρισης και η ταυτόχρονη εκτέλεση ενός πλήθους εφαρμογών χαμηλής προτεραιότητας. Επιπλέον, μελετήσαμε την απόδοση διαφορετικών πρακτόρων στις διάφορες συνεκτελέσεις και υπολογίσαμε ότι το αποδοτικότερο, κατά μέσο όρο, από αυτά τα μοντέλα διατηρεί το QoS target κατά 89.254%. Το ποσοστό αυτό συνιστά βελτίωση περίπου 370% σε σχέση με την κατάσταση μη ελεγχόμενης συνεκτέλεσης.

6.2 Συζήτηση και Επεκτάσεις

Η συγκεκριμένη εργασία θα μπορούσε να επεκταθεί σε πολλές κατευθύνσεις. Μια πρώτη επέκταση αφορά την ολιστική διαχείριση του ανταγωνισμού σε όλους τους κοινόχρηστους πόρους. Για παράδειγμα, ο μηχανισμός θα μπορούσε να λαμβάνει αποφάσεις και για το εύρος ζώνης προς την κύρια μνήμη που διατίθεται σε κάθε ομάδα πυρήνων, αξιοποιώντας την τεχνολογία MBA (Memory Bandwidth Allocation) της Intel. Κάτι τέτοιο ίσως αποδεικνυόταν χρήσιμο, ιδιαίτερα στις περιπτώσεις εκείνες όπου η υπηρεσία έχει υψηλές απαιτήσεις για bandwidth. Άλλοι πόροι που θα μπορούσαν να ρυθμίζονται δυναμικά είναι, μεταξύ άλλων, το πλήθος των πυρήνων που ανατίθενται σε κάθε ομάδα εφαρμογών και η συχνότητα λειτουργίας τους. Σε καθεμία από τις παραπάνω περιπτώσεις, θα ήταν απαραίτητο να γίνουν τροποποιήσεις στην αρχιτεκτονική του νευρωνικού δικτύου ώστε να έχει πολλαπλές εξόδους, μία για κάθε πόρο που ρυθμίζεται.

Μια άλλη βελτίωση του μηχανισμού μας αφορά την πιο συστηματική ρύθμιση των υπερπαραμέτρων του μοντέλου. Στην παρούσα μελέτη, λόγω των χρονικών περιορισμών, δε διερευνήσαμε όλους τους πιθανούς συνδυασμούς τιμών, αλλά αντίθετα, η μελέτη της επίδρασής τους στην απόδοση των μοντέλων μας έγινε πιο μεμονωμένα. Παρόλο που δεν παρατηρήσαμε δραματικές αλλαγές στα αποτελέσματά μας πραγματοποιώντας μικρές αλλαγές των περισσότερων υπερπαραμέτρων, μια πιο ολοκληρωμένη και συστηματική μελέτη της επίδρασής τους θα μπορούσε, εν δυνάμει, να μας δώσει ακόμα καλύτερα αποτελέσματα.

Επιπλέον, η επιλογή των εισόδων του νευρωνικού δικτύου πραγματοποιήθηκε μέσω δοκιμών μετρικών υλικού που ήταν διαισθητικά ταιριαστές και που οι αυξομοιώσεις τους φαίνονταν να έχουν άμεση συσχέτιση με το latency της LC εφαρμογής. Ενδιαφέρον θα παρουσίαζε, λοιπόν, μια πιο εμπεριστατωμένη επιλογή των εισόδων (features) του μοντέλου από το μεγάλο σύνολο μετρικών που μπορούμε να συλλέξουμε, ενδεχομένως επιστρατεύοντας εργαλεία στατιστικής ή ακόμα και τεχνικές μηχανικής μάθησης.

Βιβλιογραφία

[1] S. Chen, C. Delimitrou, and J. F. Martinez. Parties: Qos-aware resource partitioning for multiple interactive services. In *Proceedings of the 24th ACM International Conference on Architectural Support for Programming Languages and Operating Systems*, 2019.

[2] D.-A. Clevert, T. Unterthiner, and S. Hochreiter. Fast and accurate deep network learning by exponential linear units (elus). In *Proceedings of the 4th International Conference on Learning Representations*, 2015.

[3] C. Delimitrou, D. Sanchez, and C. Kozyrakis. Tarcil: Reconciling scheduling speed and quality in large shared clusters. In *Proceedings of the 6th ACM Symposium on Cloud Computing*, 2015.

[4] Y. Gan, Y. Zhang, K. Hu, D. Cheng, Y. He, M. Pancholi, and C. Delimitrou. Leveraging deep learning to improve the performance predictability of cloud microservices. 2019.

[5] E. Garza, S. Mirbagher-Ajorpaz, and D. A. Jiménez. Bit-level perceptron prediction for indirect branches. In *Proceedings of the 46th International Symposium on Computer Architecture*, pages 27–38, 2019.

[6] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, 2016.

[7] H. Hasselt. Double q-learning. In *Proceedings of the 23rd Annual Conference on Neural Information Processing Systems*, 2010.

[8] L. He, S. Yao, and W. Zhou. An extended fine-grained conflict detection method for shared-state scheduling in large scale cluster. *ACM Conference on Intelligent Information Processing*, page 29, 2016.

[9] H. Kasture and D. Sanchez. Tailbench: a benchmark suite and evaluation methodology for latency-critical applications. In *2016 IEEE International Symposium on Workload Characterization*, 2016.

[10] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *Proceedings of the 3rd International Conference on Learning Representations*, 2014.

[11] Y. LeCun, Y. Bengio, and G. Hinton. *Deep learning*. Nature, 2015.

[12] Y. Li, D. Sun, and B. C. Lee. Dynamic colocation policies with reinforcement learning. *ACM Transactions on Architecture and Code Optimization*, 17(1), 2020.

[13] G. Linden, B. Smith, and J. York. Amazon.com recommendations: Item-to-item collaborative filtering. *IEEE Internet Computing*, 7(1):76–80, 2003.

[14] D. Lo, L. Cheng, R. Govindaraju, L. A. Barroso, and C. Kozyrakis. Towards energy proportionality for large-scale latency-critical workloads. In *Proceedings of the 41st Annual International Symposium on Computer Architecture*, 2014.

[15] D. Lo, L. Cheng, R. Govindaraju, P. Ranganathan, and C. Kozyrakis. Heracles: Improving resource efficiency at scale. *ACM Computer Architecture News*, 43:450–462, 2015.

[16] T. M. Mitchell. *Machine Learning*. McGraw-Hill Education, 1997.

[17] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. A. Riedmiller. Playing atari with deep reinforcement learning. 2018.

[18] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518:529–533, 2015.

[19] D. Nemirovsky, T. Arkose, N. Markovic, M. Nemirovsky, O. Unsal, A. Cristal, and M. Valero. A general guide to applying machine learning to computer architecture. *Supercomputing Frontiers and Innovations*, 5(1), 2018.

[20] K. Nikas, N. Papadopoulou, D. Giantsidi, V. Karakostas, G. Goumas, and N. Koziris. Dicer: Diligent cache partitioning for efficient workload consolidation. In *Proceedings of the 48th International Conference on Parallel Processing*. Association for Computing Machinery, 2019.

[21] R. Nishtala, V. Petrucci, P. Carpenter, and M. Själander. Twig: Multi-agent task management for colocated latency-critical cloud services. In *2020 IEEE International Symposium on High Performance Computer Architecture*, 2020.

[22] L. Peled, S. Mannor, U. Weiser, and Y. Etsion. Semantic locality and context-based prefetching using reinforcement learning. In *Proceedings of the 42nd International Symposium of Computer Architecture*, 2015.

[23] Artemis Repository. http://artemis.cslab.ece.ntua.gr:8080/jspui/handle/123456789/17662.

[24] F. Rosenblatt. *Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms*. Spartan Books, Washington DC, 1961.

[25] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning representations by back-propagating errors. *Nature*, 323:533–536, 1986.

[26] Z. Shi, X. Huang, A. Jain, and C. Lin. Applying deep learning to the cache replacement problem. In *Proceedings of the 52nd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO-52)*, 2019.

[27] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. van den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, and D. Hassabis. Mastering the game of go with deep neural networks and tree search. *Nature*, pages 484–489, 2016.

[28] D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel, T. Lillicrap, K. Simonyan, and D. Hassabis. A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. *Science*, 362:1140–1144, 2018.

[29] R. Sutton and G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, 2018.

[30] UCI. https://archive.ics.uci.edu/ml/datasets.

[31] Pqos/Intel RDT Utility. https://github.com/intel/intel-cmt-cat/tree/master/pqos.

[32] S. Wang, Y.-H. Zhu, S.-P. Chen, T.-Z. Wu, W.-J. Li, X.-S. Zhan, H.-Y. Ding, W.-S. Shi, and Y.-G. Bao. A case for adaptive resource management in alibaba datacenter using neural networks. *Journal of Computer Science and Technology*, 35(1):209–220, 2020.

[33] Z. Wang, T. Schaul, M. Hessel, H. Van Hasselt, and M. Lanctot. Dueling network architectures for deep reinforcement learning. In *Proceedings of the 33rd International Conference on Machine Learning*, volume 48, 2016.

[34] C. Watkins. *Learning from Delayed Rewards*. King's College, 1989.

[35] Perf. Linux Profiling with Perf. Counters. https://perf.wiki.kernel.org.

[36] Y. Zhang, D. Meisner, J. Mars, and L. Tang. Treadmill: Attributing the source of tail latency through precise load testing and statistical inference. In *Proceedings of the 43rd International Symposium on Computer Architecture*, 2016.