ΕΛΛΗΝΙΚΗ ΔΗΜΟΚΡΑΤΙΑ
ΠΑΝΕΠΙΣΤΗΜΙΟ ΔΥΤΙΚΗΣ ΜΑΚΕΔΟΝΙΑΣ

ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ
ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ &
ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

Βελτιστοποίηση τεχνικών υπολογιστικής εκφόρτωσης με χρήση αλγορίθμων μηχανικής μάθησης

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

του

ΓΑΛΑΤΑ ΓΕΩΡΓΙΟΥ

Επιβλέπων Καθηγητής: Σαρηγιαννίδης Παναγιώτης

ΚΟΖΑΝΗ/ΟΚΤΩΒΡΙΟΣ/2024

HELLENIC DEMOCRACY
UNIVERSITY OF WESTERN MACEDONIA

FACULTY OF ENGINEERING
DEPARTMENT OF ELECTRICAL &
COMPUTER ENGINEERING

Computation offloading optimization using machine learning algorithms

THESIS

GALATAS GEORGIOS

SUPERVISOR: Professor Sarigiannidis Panagiotis

KOZANI/OCTOBER/2024

ΕΛΛΗΝΙΚΗ ΔΗΜΟΚΡΑΤΙΑ
ΠΑΝΕΠΙΣΤΗΜΙΟ ΔΥΤΙΚΗΣ ΜΑΚΕΔΟΝΙΑΣ
ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ
ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
& ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΔΗΛΩΣΗ ΜΗ ΛΟΓΟΚΛΟΠΗΣ ΚΑΙ ΑΝΑΛΗΨΗΣ ΠΡΟΣΩΠΙΚΗΣ ΕΥΘΥΝΗΣ

Δηλώνω ρητά ότι, σύμφωνα με το άρθρο 8 του Ν. 1599/1986 και τα άρθρα 2,4,6 παρ. 3 του Ν. 1256/1982, η παρούσα Διπλωματική Εργασία με τίτλο " Βελτιστοποίηση τεχνικών υπολογιστικής εκφόρτωσης με χρήση αλγορίθμων μηχανικής μάθησης" καθώς και τα ηλεκτρονικά αρχεία και πηγαίοι κώδικες που αναπτύχθηκαν ή τροποποιήθηκαν στα πλαίσια αυτής της εργασίας και αναφέρονται ρητώς μέσα στο κείμενο που συνοδεύουν, και η οποία έχει εκπονηθεί στο Τμήμα Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών του Πανεπιστημίου Δυτικής Μακεδονίας, υπό την επίβλεψη του μέλους του Τμήματος κ. Παναγιώτη Σαρηγιαννίδη αποτελεί αποκλειστικά προϊόν προσωπικής εργασίας και δεν προσβάλλει κάθε μορφής πνευματικά δικαιώματα τρίτων και δεν είναι προϊόν μερικής ή ολικής αντιγραφής, οι πηγές δε που χρησιμοποιήθηκαν περιορίζονται στις βιβλιογραφικές αναφορές και μόνον. Τα σημεία όπου έχω χρησιμοποιήσει ιδέες, κείμενο, αρχεία ή / και πηγές άλλων συγγραφέων, αναφέρονται ευδιάκριτα στο κείμενο με την κατάλληλη παραπομπή και η σχετική αναφορά περιλαμβάνεται στο τμήμα των βιβλιογραφικών αναφορών με πλήρη περιγραφή. Απαγορεύεται η αντιγραφή, αποθήκευση και διανομή της παρούσας εργασίας, εξ ολοκλήρου ή τμήματος αυτής, για εμπορικό σκοπό. Επιτρέπεται η ανατύπωση, αποθήκευση και διανομή για σκοπό μη κερδοσκοπικό, εκπαιδευτικής ή ερευνητικής φύσης, υπό την προϋπόθεση να αναφέρεται η πηγή προέλευσης και να διατηρείται το παρόν μήνυμα. Ερωτήματα που αφορούν τη χρήση της εργασίας για κερδοσκοπικό σκοπό πρέπει να απευθύνονται προς τον συγγραφέα. Οι απόψεις και τα συμπεράσματα που περιέχονται σε αυτό το έγγραφο εκφράζουν τον συγγραφέα και μόνο.

Υπογραφή Φοιτητή: _____

Περίληψη

Η παρούσα διπλωματική εργασία εξετάζει την εφαρμογή τεχνικών Ενισχυτικής Μάθησης (Reinforcement Learning) στο πεδίο της υπολογιστικής στις παρυφές του δικτύου (Mobile Edge Computing - MEC) και, κυρίως, στη βελτιστοποίηση της διαδικασίας υπολογιστικής εκφόρτωσης (computation offloading). Το MEC αποτελεί μια καινοτόμο προσέγγιση που στοχεύει στη μείωση της καθυστέρησης και στη βελτίωση της ποιότητας υπηρεσίας και απόδοσης των συσκευών χρηστών, μεταφέροντας τον υπολογιστικό φόρτο από αυτές σε κοντινούς διακομιστές στις παρυφές του δικτύου.

Η εργασία επικεντρώνεται στην αντιμετώπιση των προκλήσεων της υπολογιστικής εκφόρτωσης, όπως η δυναμική κατανομή πόρων και η βελτιστοποίηση της ενεργειακής κατανάλωσης. Αρχικά, μέσω βιβλιογραφικής ανασκόπησης, αναλύθηκαν διάφορα σενάρια υπολογιστικής εκφόρτωσης και οι υπάρχουσες υλοποιήσεις. Στη συνέχεια, διατυπώθηκε το πρόβλημα της υπολογιστικής εκφόρτωσης, το οποίο εστιάζει στη βέλτιστη κατανομή εργασιών μεταξύ της τοπικής συσκευής και του απομακρυσμένου διακομιστή, λαμβάνοντας υπόψη περιορισμούς, όπως η κατανάλωση ενέργειας και ο χρόνος καθυστέρησης. Για την αντιμετώπιση του προβλήματος, αναπτύχθηκε ένα περιβάλλον προσομοίωσης που αναπαριστά την δυναμική ενός συστήματος MEC. Στην συνέχεια, στο περιβάλλον αυτό, εφαρμόστηκαν τρεις αλγόριθμοι ενισχυτικής μάθησης: Proximal Policy Optimization (PPO), Advantage Actor Critic (A2C) και Deep Q-Network (DQN) και συγκρίθηκαν τα αποτελέσματά τους. Ιδιαίτερη έμφαση δόθηκε στη βελτιστοποίηση των υπερπαραμέτρων για την επίτευξη βέλτιστης απόδοσης.

Τα πειραματικά αποτελέσματα αποδεικνύουν ότι το πρόβλημα της υπολογιστικής εκφόρτωσης μπορεί να αντιμετωπιστεί επιτυχώς με την χρήση των παραπάνω αλγορίθμων. Ωστόσο, αναγνωρίστηκαν προκλήσεις στη διαμόρφωση ενός ρεαλιστικού περιβάλλοντος προσομοίωσης και στον σχεδιασμό κατάλληλων συναρτήσεων ανταμοιβής. Τέλος, αναλύεται η αποτελεσματικότητα της προτεινόμενης μεθόδου και παρουσιάζονται μελλοντικές ερευνητικές κατευθύνσεις, όπως η αύξηση της πολυπλοκότητας και της ακρίβειας του συστήματος MEC και η ενσωμάτωση πιο σύνθετων μοντέλων ενεργειακής κατανάλωσης.

Λέξεις Κλειδιά:

Υπολογιστική στις Παρυφές του Δικτύου

Υπολογιστική Εκφόρτωση

Ενισχυτική Μάθηση

Βαθιά Μάθηση

Abstract

This thesis examines the application of Reinforcement Learning techniques in the field of Mobile Edge Computing (MEC), specifically focusing on optimizing the computation offloading process. MEC is an innovative approach aimed at reducing latency and improving the quality of service and performance of user devices by offloading computational workloads to nearby servers at the network edge.

The study addresses the challenges of computation offloading, such as dynamic resource allocation and energy consumption optimization. Initially, through a literature review, various computation offloading scenarios and existing implementations are analyzed. Then, the computation offloading problem is formulated, focusing on the optimal task allocation between the local device and the remote server, considering constraints such as energy consumption and latency. To tackle this problem, a simulation environment was developed to represent the dynamics of an MEC system. In this environment, three reinforcement learning algorithms were applied: Proximal Policy Optimization (PPO), Advantage Actor Critic (A2C), and Deep Q-Network (DQN), and their results were compared. Special emphasis was placed on hyperparameter optimization to achieve optimal performance.

The experimental results demonstrate that the computation offloading problem can be successfully addressed using the aforementioned algorithms. However, challenges were identified in configuring a realistic simulation environment and designing appropriate reward functions. Finally, the effectiveness of the proposed method is analyzed, and future research directions are presented, such as increasing the complexity and accuracy of the MEC system and incorporating more complex energy consumption models.

Keywords:

Mobile Edge Computing

Computation Offloading

Reinforcement Learning

Deep Learning

Ευχαριστίες

Αρχικά, θα ήθελα να ευχαριστήσω τον επιβλέποντα καθηγητή κ. Παναγιώτη Σαρηγιαννίδη, ο οποίος μου έδωσε την ευκαιρία να εμβαθύνω σε ένα τόσο ενδιαφέρον αντικείμενο έρευνας. Η διαδικασία αυτή αποδείχθηκε ιδιαίτερα εποικοδομητική, διευρύνοντας τους ορίζοντές μου και επηρεάζοντας σημαντικά τον τρόπο σκέψης μου.

Θα ήθελα λοιπόν, εκτός από τον επιβλέποντα καθηγητή, να ευχαριστήσω και τον μεταδιδακτορικό ερευνητή κ. Δημήτριο Πλιάτσιο για την πολύτιμη καθοδήγηση του, καθώς και να εκφράσω την εκτίμησή μου για όλες τις συμβουλές που μου έδωσε στην διάρκεια εκπόνησης της εργασίας.

Τέλος, οφείλω ένα μεγάλο ευχαριστώ στην οικογένειά μου για την αμέριστη συμπαράσταση και κατανόηση που επέδειξαν καθ' όλη τη διάρκεια των σπουδών μου. Ιδιαίτερα, θα ήθελα να αφιερώσω αυτή την εργασία στη μνήμη του πατέρα μου, η αγάπη και καθοδήγηση του οποίου συνεχίζουν να αποτελούν πηγή έμπνευσης για μένα.

Περιεχόμενα

Κατάλογος Εικόνων

Κατάλογος Πινάκων

Κεφάλαιο 1: Εισαγωγή

1.1 Η εξέλιξη της Τεχνητής Νοημοσύνης και η ανάγκη για Υπολογιστική Παρυφής

Τα τελευταία χρόνια, ο τομέας της Τεχνητής Νοημοσύνης (ΤΝ) έχει γνωρίσει πρωτοφανή ανάπτυξη, οδηγούμενος από τις εξελίξεις στις τεχνικές μηχανικής μάθησης και τη διαθεσιμότητα τεράστιου όγκου δεδομένων. Αυτή η εκθετική αύξηση στη δημιουργία και κατανάλωση δεδομένων έχει οδηγήσει σε μια έκρηξη της ζήτησης για υπολογιστικούς πόρους, καθιστώντας αναγκαίες καινοτόμες λύσεις για αποτελεσματική επεξεργασία και αποθήκευση δεδομένων.

Οι εφαρμογές ΤΝ, ιδιαίτερα σε τομείς όπως η υπολογιστική όραση και η παραγωγική ΤΝ (generative Artificial Intelligence - AI), βρίσκονται στην πρώτη γραμμή αυτής της επανάστασης δεδομένων. Αυτές οι τεχνολογίες απαιτούν τεράστια σύνολα δεδομένων για εκπαίδευση, ωθώντας στα όρια τις παραδοσιακές υπολογιστικές υποδομές. Για παράδειγμα, εργασίες υπολογιστικής όρασης όπως η αναγνώριση εικόνων και η ανίχνευση αντικειμένων συχνά περιλαμβάνουν την επεξεργασία εκατομμυρίων εικόνων υψηλής ανάλυσης, απαιτώντας σημαντική υπολογιστική ισχύ και χωρητικότητα αποθήκευσης.

Η απότομη ανάπτυξη των εφαρμογών generative AI έχει οδηγήσει σε αυξημένη εξάρτηση από την υποδομή υπολογιστικού νέφους. Οι πλατφόρμες νέφους προσφέρουν κλιμακούμενους (scalable) και ευέλικτους (agile) πόρους που μπορούν να προσαρμοστούν στις μεταβαλλόμενες υπολογιστικές απαιτήσεις του φόρτου των εργασιών ΤΝ. Ωστόσο, η κεντρική φύση των παραδοσιακών μοντέλων υπολογιστικού νέφους εισάγει προκλήσεις, ιδιαίτερα σε σενάρια όπου η χαμηλή καθυστέρηση και η επεξεργασία σε πραγματικό χρόνο είναι κρίσιμες. Αυτό έχει οδηγήσει στην ανάπτυξη της υπολογιστικής παρυφής, το οποίο φέρνει τον υπολογισμό και την αποθήκευση δεδομένων πιο κοντά στο σημείο δημιουργίας και κατανάλωσης δεδομένων.

Η υπολογιστική παρυφής έχει αναδειχθεί ως ένα κρίσιμο παράδειγμα για την αντιμετώπιση των περιορισμών καθυστέρησης και εύρους ζώνης που σχετίζονται με τις λύσεις βασισμένες στο νέφος, ιδιαίτερα σε κρίσιμα περιβάλλοντα όπου η λήψη αποφάσεων σε πραγματικό χρόνο είναι υψίστης σημασίας. Με την επεξεργασία των δεδομένων πιο κοντά στην πηγή τους, η υπολογιστική παρυφής (edge computing) μπορεί να μειώσει σημαντικά την καθυστέρηση και να βελτιώσει τη συνολική ανταπόκριση του συστήματος.

Ένα παράδειγμα της σημαντικότητας της υπολογιστικής παρυφής δικτύου είναι στο πλαίσιο των αυτόνομων οχημάτων. Τα αυτό-οδηγούμενα οχήματα παράγουν τεράστιες ποσότητες δεδομένων αισθητήρων που απαιτούν επεξεργασία σε πραγματικό χρόνο για ασφαλή πλοήγηση και λήψη αποφάσεων. Η καθυστέρηση που εισάγεται από τη μετάδοση αυτών των δεδομένων σε έναν απομακρυσμένο διακομιστή νέφους για επεξεργασία θα μπορούσε να οδηγήσει σε δυνητικά επικίνδυνες καθυστερήσεις σε κρίσιμες καταστάσεις, όπως η αποφυγή ατυχήματος. Το edge computing πραγματοποιεί άμεση επεξεργασία των δεδομένων των αισθητήρων, επιτρέποντας ταχύτερους χρόνους απόκρισης και ενισχύοντας την ασφάλεια και την αξιοπιστία των συστημάτων αυτόνομης οδήγησης.

1.2 Σύγχρονες προκλήσεις για Υπολογιστική Παρυφής

Η σύγκλιση των κλάδων της ΤΝ, του υπολογιστικού νέφους και της υπολογιστικής παρυφών, έχει δημιουργήσει ένα πολύπλοκο οικοσύστημα που απαιτεί εξελιγμένες στρατηγικές διαχείρισης πόρων. Καθώς ο όγκος και η ποικιλία των δεδομένων συνεχίζουν να αυξάνονται, μαζί με την ποικιλομορφία των υπολογιστικών περιβαλλόντων, υπάρχει αυξανόμενη ανάγκη για έξυπνες διαδικασίες λήψης αποφάσεων για τη βελτιστοποίηση της κατανομής, αξιοποίησης και χρήσης των πόρων. Αυτή η ανάγκη για έξυπνη διαχείριση πόρων θέτει το σκηνικό για την εξερεύνηση προηγμένων τεχνικών, όπως η Ενισχυτική Μάθηση, για την αντιμετώπιση των προκλήσεων της υπολογιστικής εκφόρτωσης και της κατανομής φόρτου εργασίας σε ετερογενή υπολογιστικά περιβάλλοντα. Αξιοποιώντας αυτές τις προσεγγίσεις που βασίζονται στην ΤΝ, μπορούμε να αναπτύξουμε πιο αποτελεσματικά και ευέλικτα συστήματα που μπορούν να εξισορροπήσουν δυναμικά τις αντισταθμίσεις μεταξύ απόδοσης, κατανάλωσης ενέργειας και χρονικής καθυστέρησης σε πολύπλοκα σενάρια κατανεμημένης υπολογιστικής.

Επιπρόσθετα, η τεράστια κατανάλωση ενέργειας των κέντρων δεδομένων και, γενικότερα, των υπολογιστικών υποδομών έχει οδηγήσει στην ανάπτυξη πρωτοβουλιών πράσινου υπολογιστικού νέφους. Αυτές οι προσπάθειες στοχεύουν στη βελτιστοποίηση της χρήσης πόρων, τη βελτίωση της ενεργειακής απόδοσης και τη μείωση του ανθρακικού αποτυπώματος της λειτουργίας υπολογιστικών νεφών. Οι στρατηγικές πράσινου υπολογιστικού νέφους περιλαμβάνουν τη χρήση ανανεώσιμων πηγών ενέργειας, προηγμένων τεχνολογιών ψύξης και έξυπνων συστημάτων διαχείρισης φόρτου εργασιών. Εφαρμόζοντας αυτές τις προσεγγίσεις, οι πάροχοι νέφους μπορούν να προσφέρουν πιο βιώσιμες λύσεις υπολογιστικής, διατηρώντας παράλληλα την απόδοση και την αξιοπιστία που απαιτούνται από τις απαιτητικές σε δεδομένα εφαρμογές ΤΝ.

Η επίλυση του προβλήματος της βέλτιστης υπολογιστικής εκφόρτωσης σε περιβάλλον MEC έχει σημαντικό αντίκτυπο στην κατανάλωση ενέργειας. Η υπολογιστική εκφόρτωση επιτρέπει σε συσκευές με περιορισμένους πόρους να μεταφέρουν υπολογιστικά απαιτητικές εργασίες σε κοντινούς διακομιστές άκρου (edge server-nodes). Ωστόσο, αυτή η διαδικασία αντιμετωπίζει ορισμένες προκλήσεις:

1. **Δυναμικό Περιβάλλον**: Το περιβάλλον MEC είναι εξαιρετικά δυναμικό, με συνεχώς μεταβαλλόμενες απαιτήσεις υπολογισμού, συνθήκες δικτύου και μεταβλητούς υπολογιστικούς πόρους.

2. **Ετερογενείς Εργασίες**: Οι εφαρμογές συσκευών χρηστών δημιουργούν διαφορετικές εργασίες με ποικίλες υπολογιστικές απαιτήσεις και ανοχές σε θέματα καθυστέρησης.

3. **Ποιότητα Υπηρεσίας (Quality of Service)**: Η διασφάλιση της ποιότητας υπηρεσίας, ιδιαίτερα για εφαρμογές ευαίσθητες στην καθυστέρηση, είναι μια πρόκληση στο περιβάλλον MEC.

4. **Ενεργειακή Αποδοτικότητα**: Είναι απαραίτητο να ληφθεί υπόψη η συνολική κατανάλωση ενέργειας του συστήματος MEC.

5. **Κατανομή Πόρων**: Η αποτελεσματική κατανομή περιορισμένων διαθέσιμων πόρων μεταξύ πολλαπλών χρηστών και εργασιών αποτελεί ένα πολύπλοκο πρόβλημα βελτιστοποίησης.

1.3 Επίκεντρο και Συνεισφορά της Εργασίας

Η παρούσα διπλωματική εργασία επικεντρώνεται στην ανάπτυξη μιας έξυπνης στρατηγικής υπολογιστικής εκφόρτωσης χρησιμοποιώντας τεχνικές Ενισχυτικής Μάθησης (ΕΜ). Οι κύριες συνεισφορές περιλαμβάνουν:

1. **Προσαρμοσμένο Περιβάλλον OpenAI Gym**: Δημιουργία ενός προσαρμοσμένου περιβάλλοντος προσομοίωσης με χρήση της πλατφόρμας OpenAI Gym, που μοντελοποιεί τις πολυπλοκότητες ενός συστήματος MEC.

2. **Νέα Συνάρτηση Ανταμοιβής**: Μια συνάρτηση που συνδυάζει την εξισορρόπηση πολλαπλών στόχων, όπως η κατανάλωση ενέργειας και η χρονική καθυστέρηση για την ολοκλήρωση της διεργασίας.

3. **Υλοποίηση Αλγορίθμων ΕΜ**: Υλοποίηση διαφόρων σύγχρονων αλγορίθμων ΕΜ και συγκεκριμένα των PPO, A2C και DQN, καθώς και βελτιστοποίηση τους με υπερπαραμέτρους.

4. **Αξιολόγηση και σύγκριση απόδοσης**: Πειράματα προσομοίωσης για την αξιολόγηση των προτεινόμενων στρατηγικών και σύγκριση των αποτελεσμάτων τους.

1.4 Δομή Διπλωματικής Εργασίας

Η διπλωματική εργασία οργανώνεται ως εξής:

- **Κεφάλαιο 2: Θεωρητικό Υπόβαθρο**. Σε αυτό το κεφάλαιο παρουσιάζεται το θεωρητικό υπόβαθρο που σχετίζεται με την διπλωματική εργασία και αναλύονται έννοιες σχετικά με την Βαθιά Ενισχυτική Μάθηση, καθώς και με την Υπολογιστική Εκφόρτωση.

- **Κεφάλαιο 3**: **Διατύπωση και Επίλυση του Προβλήματος Υπολογιστικής Εκφόρτωσης.** Στο κεφάλαιο αυτό παρουσιάζεται αναλυτικά το πρόβλημα της

υπολογιστικής εκφόρτωσης και η προτεινόμενη προσέγγιση επίλυσής του. Αρχικά, πραγματοποιείται επεξήγηση των βασικών εννοιών και του σχεδιασμού της υλοποίησης. Στη συνέχεια, αναλύονται τα εργαλεία και οι βιβλιοθήκες που αξιοποιήθηκαν, καθώς και η μεθοδολογία που ακολουθήθηκε για την επίλυση του προβλήματος σε περιβάλλον MEC.

- **Κεφάλαιο 4**: **Πειραματικά Αποτελέσματα και Ανάλυση.** Σε αυτό το κεφάλαιο παρουσιάζονται τα αποτελέσματα των πειραμάτων που πραγματοποιήθηκαν μέσω προσομοίωσης και συγκρίνεται η απόδοση των αλγορίθμων σε διάφορα σενάρια.

- **Κεφάλαιο 5: Συμπεράσματα και Μελλοντική Εργασία.** Στο κεφάλαιο αυτό συνοψίζονται τα συμπεράσματα τόσο των πειραμάτων, όσο και γενικότερων ζητημάτων σχετικά με το πρόβλημα της υπολογιστικής εκφόρτωσης και προτείνονται μελλοντικές επεκτάσεις.

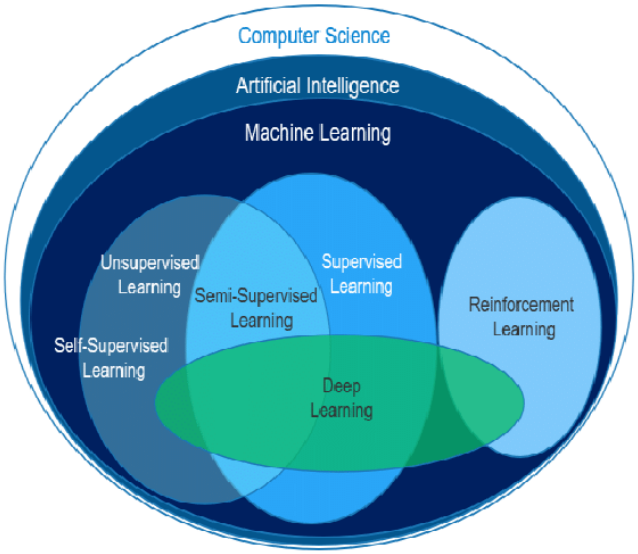
Κάθε κεφάλαιο παρέχει μια σαφή και λογική εξέλιξη των ιδεών, από τις θεμελιώδεις έννοιες έως τις νέες συνεισφορές αυτής της έρευνας. Η διπλωματική εργασία στοχεύει να αναπτύξει βέλτιστες υλοποιήσεις στο πρόβλημα της υπολογιστικής εκφόρτωσης και να προτείνει νέες μεθόδους στον τομέα της υπολογιστικής στις παρυφές του δικτύου, αποδεικνύοντας την αποτελεσματικότητα των τεχνικών ΕΜ στην αντιμετώπιση των πολύπλοκων προκλήσεων της υπολογιστικής εκφόρτωσης.

Κεφάλαιο 2: Θεωρητικό Υπόβαθρο

Στο κεφάλαιο αυτό θα γίνει παρουσίαση και ανάλυση του απαιτούμενου θεωρητικού υπόβαθρου προκειμένου να γίνει κατανοητό το περιεχόμενο της εργασίας.

2.1 Μηχανική Μάθηση και Τεχνητή Νοημοσύνη

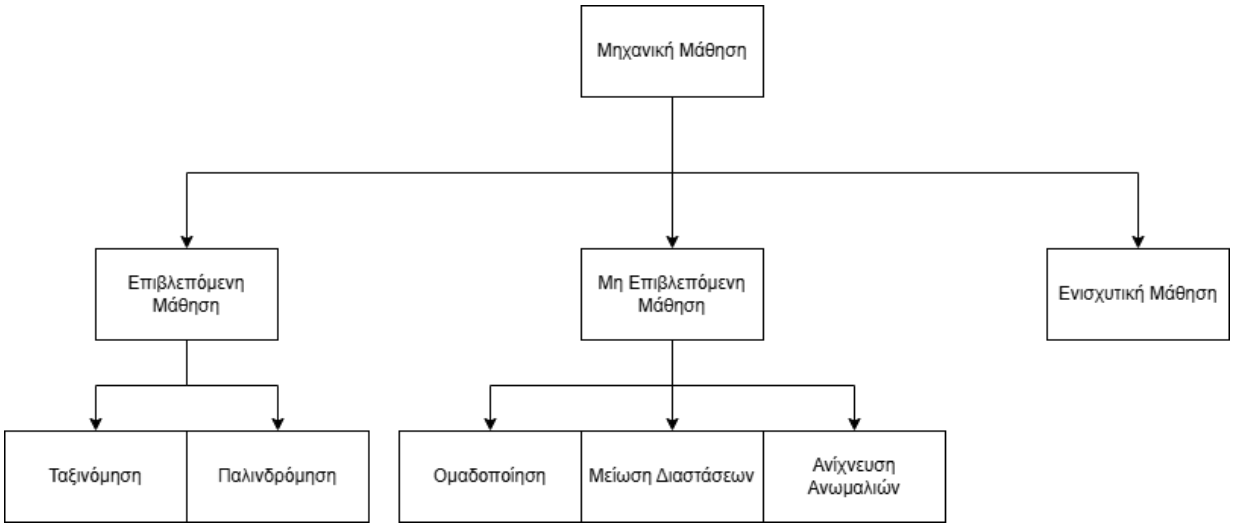
Η Μηχανική Μάθηση (ΜΜ) αποτελεί έναν κλάδο της ΤΝ, όπως φαίνεται και στην Εικόνα 1, που επικεντρώνεται στην ανάπτυξη αλγορίθμων και στατιστικών μοντέλων, επιτρέποντας στα υπολογιστικά συστήματα να βελτιώνουν την απόδοσή τους σε συγκεκριμένες εργασίες μέσω της εμπειρίας [1]. Σε αντίθεση με τον παραδοσιακό προγραμματισμό, όπου παρέχονται ρητές οδηγίες για την επίλυση ενός προβλήματος, οι αλγόριθμοι μηχανικής μάθησης χρησιμοποιούν δεδομένα για να μάθουν μοτίβα και να λαμβάνουν αποφάσεις με ελάχιστη ανθρώπινη παρέμβαση [2].



Εικόνα 1 - Σχέσεις μεταξύ Τεχνητής Νοημοσύνης, Μηχανικής Μάθησης και Βαθιάς Μάθησης [3]

Η ΤΝ, με την ευρύτερη έννοια, αναφέρεται στην προσομοίωση της ανθρώπινης νοημοσύνης σε μηχανές προγραμματισμένες να σκέφτονται και να ενεργούν όπως οι άνθρωποι [4]. Ενώ η ΤΝ περιλαμβάνει ένα ευρύ φάσμα τεχνολογιών και προσεγγίσεων, η ΜΜ έχει αναδειχθεί ως ένας από τους πιο ισχυρούς και ευρέως χρησιμοποιούμενους υποτομείς ΤΝ [5].

Οι αλγόριθμοι ΜΜ μπορούν να κατηγοριοποιηθούν ευρέως, όπως φαίνεται και στην Εικόνα 2, σε τρεις κύριους τύπους: Επιβλεπόμενη Μάθηση (Supervised Learning), Μη Επιβλεπόμενη Μάθηση (Unsupervised Learning) και Ενισχυτική Μάθηση (Reinforcement Learning) [6]. Κάθε μία από αυτές τις κατηγορίες έχει τους δικούς της αλγορίθμους, εφαρμογές και προκλήσεις. Στις επόμενες ενότητες, θα παρουσιαστεί καθεμία από αυτές τις κατηγορίες λεπτομερώς, με ιδιαίτερη έμφαση στην Ενισχυτική Μάθηση, η οποία αποτελεί το επίκεντρο της παρούσας διπλωματικής εργασίας.



Εικόνα 2 - Γενικές Κατηγορίες Μηχανικής Μάθησης

2.2 Επιβλεπόμενη Μάθηση

Η Επιβλεπόμενη Μάθηση είναι ένας τύπος ΜΜ όπου ο αλγόριθμος μαθαίνει από επισημασμένα δεδομένα (labeled data) εκπαίδευσης [7]. Σε αυτήν την προσέγγιση, ο αλγόριθμος τροφοδοτείται με ζεύγη εισόδου-εξόδου, όπου κάθε είσοδος συσχετίζεται με τη σωστή έξοδο. Ο στόχος είναι ο αλγόριθμος να μάθει μια συνάρτηση που μπορεί να αντιστοιχίσει νέες, άγνωστες εισόδους στις σωστές εξόδους τους [8].

Τα βασικά στοιχεία της επιβλεπόμενης μάθησης περιλαμβάνουν:

1. **Σύνολο δεδομένων εκπαίδευσης (training dataset):** Αποτελούνται από χαρακτηριστικά εισόδου και τις αντίστοιχες ετικέτες ή τιμές-στόχους.

2. **Μοντέλο:** Μια μαθηματική αναπαράσταση της σχέσης μεταξύ εισόδων και εξόδων.

3. **Συνάρτηση απώλειας (loss function):** Μετρά την απόκλιση των προβλέψεων του μοντέλου από τις πραγματικές τιμές-στόχους.

4. **Διαδικασία βελτιστοποίησης:** Η διαδικασία προσαρμογής των παραμέτρων του μοντέλου για την ελαχιστοποίηση της συνάρτησης απώλειας [9].

Τα προβλήματα Επιβλεπόμενης Μάθησης μπορούν να χωριστούν σε δύο κύριες κατηγορίες:

1. **Ταξινόμηση (classification):** Η εργασία κατηγοριοποίησης εισόδων σε προκαθορισμένες κλάσεις ή κατηγορίες, όπως η ανίχνευση ανεπιθύμητης αλληλογραφίας ή η ταξινόμηση εικόνων [10].

2. **Παλινδρόμηση (regression):** Η εργασία πρόβλεψης μιας συνεχούς αριθμητικής τιμής, όπως η πρόβλεψη τιμών ακινήτων ή η πρόβλεψη τιμών μετοχών [11].

Μεταξύ των συνηθισμένων αλγορίθμων Επιβλεπόμενης Μάθησης περιλαμβάνονται η Γραμμική Παλινδρόμηση (Linear Regression), η Λογιστική Παλινδρόμηση (Logistic Regression), ο αλγόριθμος Τυχαία Δάση (Random Forests), οι Μηχανές Διανυσμάτων Υποστήριξης (Support Vector Machines), ο αλγόριθμος Κ-Πλησιέστερων Γειτόνων (K-nearest neighbor) καθώς και ορισμένες κατηγορίες Νευρωνικών Δίκτυων [12].

Η Επιβλεπόμενη Μάθηση έχει εφαρμοστεί με επιτυχία σε διάφορους τομείς, συμπεριλαμβανομένης της Επεξεργασίας Φυσικής Γλώσσας (Natural Language Processing), της Υπολογιστικής Όρασης (Computer Vision) και της Προγνωστικής Ανάλυσης (Forecasting) [13]. Ωστόσο, βασίζεται σε μεγάλο βαθμό στη διαθεσιμότητα επισημασμένων δεδομένων, τα οποία μπορεί να είναι ακριβά και χρονοβόρα στην απόκτησή τους σε πολλές περιπτώσεις [14].

2.3 Μη Επιβλεπόμενη Μάθηση

Η Μη Επιβλεπόμενη Μάθηση είναι ένας τύπος ΜΜ όπου ο αλγόριθμος μαθαίνει μοτίβα και δομές από μη επισημασμένα δεδομένα [15]. Σε αντίθεση με την Επιβλεπόμενη Μάθηση, δεν υπάρχουν προκαθορισμένες ετικέτες εξόδου και ο αλγόριθμος πρέπει να ανακαλύψει την πιθανή δομή των δεδομένων μόνος του [16].

Τα βασικά στοιχεία της Μη Επιβλεπόμενης Μάθησης περιλαμβάνουν:

- **Μη επισημασμένα δεδομένα:** Αποτελούνται από χαρακτηριστικά εισόδου χωρίς αντίστοιχες ετικέτες ή τιμές-στόχους.

- **Εξαγωγή χαρακτηριστικών:** Η διαδικασία εντοπισμού σημαντικών χαρακτηριστικών ή μοτίβων στα δεδομένα.

- **Μείωση διαστάσεων:** Τεχνικές για τη μείωση του αριθμού των μεταβλητών εισόδου διατηρώντας παράλληλα σημαντικές πληροφορίες.

- **Ομαδοποίηση:** Η εργασία ομαδοποίησης παρόμοιων σημείων δεδομένων με βάση τα χαρακτηριστικά τους [17].

Τα προβλήματα Μη Επιβλεπόμενης Μάθησης μπορούν να χωριστούν σε τρεις κύριες κατηγορίες [18]:

1. Ομαδοποίηση (clustering): Περιλαμβάνει την ομαδοποίηση παρόμοιων σημείων δεδομένων σε συστάδες, όπως η τμηματοποίηση εικόνων [19].

2. Μείωση διαστάσεων: Στοχεύει στη μείωση του αριθμού των χαρακτηριστικών διατηρώντας παράλληλα σημαντικές πληροφορίες. Ενδεικτικές μέθοδοι

αποτελούν η Ανάλυση Κύριων Συνιστωσών (Principal Component Analysis-PCA) και ο αλγόριθμος t-Distributed Stochastic Neighbor Embedding (t-SNE) [20].

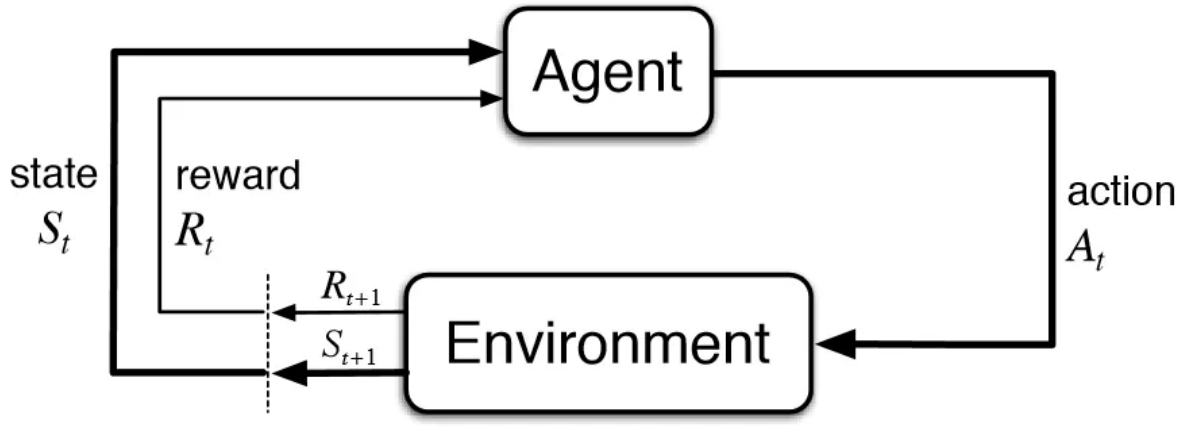
3. Ανίχνευση Ανωμαλιών (Anomaly Detection): Επικεντρώνεται στον εντοπισμό ασυνήθιστων μοτίβων ή ακραίων τιμών στα δεδομένα, όπως η ανίχνευση απάτης ή η ανίχνευση εισβολών στο δίκτυο [21].

Μεταξύ των συνηθισμένων αλγορίθμων μη επιβλεπόμενης μάθησης περιλαμβάνονται η Ομαδοποίηση K-Means (k-means clustering), η Ιεραρχική Ομαδοποίηση (Hierarchical Clustering), ο αλγόριθμος DBSCAN (Χωρική Ομαδοποίηση Εφαρμογών με Θόρυβο Βασισμένη στην Πυκνότητα), ο αλγόριθμος PCA, ο αλγόριθμος t-SNE, οι Αυτοκωδικοποιητές και τα Παραγωγικά Δίκτυα Αντιπάλων (Generative Adversarial Networks - GANs) [22].

Η Μη Επιβλεπόμενη Μάθηση είναι ιδιαίτερα χρήσιμη στην Διερευνητική Ανάλυση Δεδομένων (Exploratory Data Analysis), την εκμάθηση χαρακτηριστικών και τη γενετική μοντελοποίηση. Μπορεί να αποκαλύψει κρυμμένα μοτίβα και δομές στα δεδομένα που μπορεί να μην είναι εμφανή μέσω χειροκίνητης επιθεώρησης ή επιβλεπόμενων προσεγγίσεων [23].

2.4 Ενισχυτική Μάθηση

Η Ενισχυτική Μάθηση (ΕΜ) είναι ένας τύπος μηχανικής μάθησης που επικεντρώνεται στο πώς ένας πράκτορας θα πρέπει να λαμβάνει αποφάσεις σε ένα περιβάλλον ώστε να μεγιστοποιήσει μια Αθροιστική Ανταμοιβή (Cumulative Reward) [24]. Σε αντίθεση με την Επιβλεπόμενη και τη Μη Επιβλεπόμενη Μάθηση, η ΕΜ περιλαμβάνει έναν πράκτορα που μαθαίνει μέσω της αλληλεπίδρασης με το περιβάλλον του, λαμβάνοντας ανατροφοδότηση με τη μορφή ανταμοιβών ή ποινών [25].



Εικόνα 3 –Βασικά στοιχεία Ενισχυτικής Μάθησης [5]

Τα βασικά στοιχεία της ενισχυτικής μάθησης, όπως παρουσιάζονται και στην Εικόνα 3, περιλαμβάνουν:

- **Πράκτορας (agent):** Ο μαθητής ή ο λήπτης αποφάσεων που αλληλοεπιδρά με το περιβάλλον.

- **Περιβάλλον (environment):** Ο κόσμος στον οποίο λειτουργεί και μαθαίνει ο πράκτορας.

- **Κατάσταση (state):** Μια αναπαράσταση της τρέχουσας κατάστασης στο περιβάλλον.

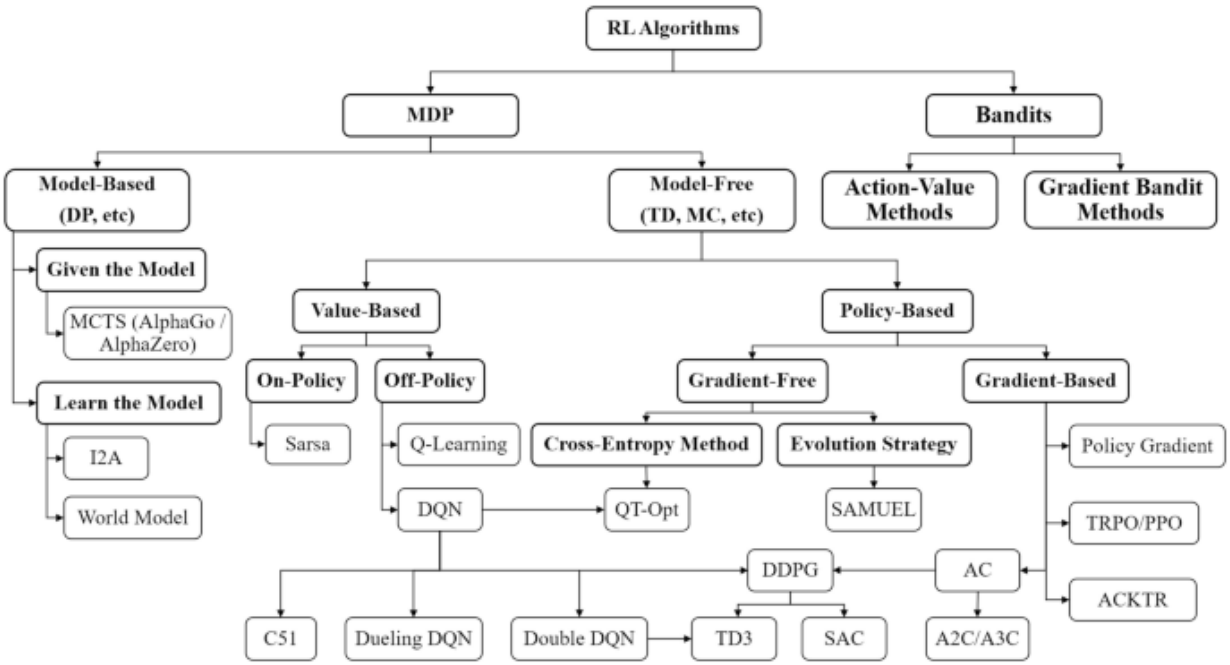
- **Δράση (action):** Μια απόφαση που λαμβάνεται από τον πράκτορα και επηρεάζει το περιβάλλον.

- **Ανταμοιβή (reward):** Ένα αριθμητικό σήμα ανατροφοδότησης που λαμβάνει ο πράκτορας από το περιβάλλον και υποδεικνύει την αξία ή την καταλληλότητα της μετάβασης από μια κατάσταση σε μια άλλη, ως αποτέλεσμα μιας συγκεκριμένης δράσης

- **Πολιτική (policy):** Η στρατηγική που χρησιμοποιεί ο πράκτορας για να καθορίσει την επόμενη δράση με βάση την τρέχουσα κατάσταση.

- **Συνάρτηση ανταμοιβής (reward function):** Αποδίδει αριθμητικές τιμές σε καταστάσεις ή ενέργειες του πράκτορα και χρησιμοποιείται για να αξιολογηθεί πόσο επιθυμητή είναι μια κατάσταση ή δράση. Μέσω αυτής της συνάρτησης, επιτυγχάνεται η μάθηση προς την επίτευξη του στόχου [26].

Γενικότερα, οι αλγόριθμοι ΕΜ χωρίζονται σε δύο κύριες κατηγορίες βάσει δύο διαφορετικών κριτηρίων. Η πρώτη διάκριση γίνεται μεταξύ αλγορίθμων βασισμένων σε πολιτική ή αξία (policy-based ή value-based), ενώ η δεύτερη διάκριση είναι μεταξύ αλγορίθμων βασισμένων σε μοντέλο και χωρίς μοντέλο (model-based και model-free). Μια κατηγοριοποίηση των αλγορίθμων ΕΜ παρουσιάζεται στην Εικόνα 4.



Εικόνα 4 - Κατηγοριοποίηση Αλγορίθμων Ενισχυτικής Μάθησης [27]

2.4.1 Προβλήματα Πολλαπλών Βραχιόνων (Multi-armed Bandits)

Το πρόβλημα των πολλαπλών βραχιόνων είναι ένα κλασικό παράδειγμα που χρησιμοποιείται για την εισαγωγή βασικών εννοιών στην ενισχυτική μάθηση [28]. Περιλαμβάνει έναν πράκτορα που πρέπει να επιλέξει μεταξύ πολλαπλών δράσεων (βραχίονες ενός μηχανήματος τυχερών παιχνιδιών), καθεμία με μια άγνωστη κατανομή ανταμοιβής. Ο στόχος είναι η μεγιστοποίηση της συνολικής ανταμοιβής με την πάροδο του χρόνου, εξισορροπώντας την εξερεύνηση (δοκιμή νέων βραχιόνων) και την εκμετάλλευση (επιλογή του βραχίονα με την υψηλότερη αναμενόμενη ανταμοιβή) [29].

Βασικές έννοιες στα προβλήματα πολλαπλών βραχιόνων περιλαμβάνουν το δίλημμα εξερεύνησης-εκμετάλλευσης, τη μετάνοια (regret), τη στρατηγική ε-greedy και τον αλγόριθμο Upper Confidence Bound (UCB) [30]. Το πρόβλημα των πολλαπλών βραχιόνων χρησιμεύει ως εισαγωγή σε πιο σύνθετα σενάρια ενισχυτικής μάθησης και εισάγει θεμελιώδεις έννοιες που εφαρμόζονται σε διάφορους αλγορίθμους ΕΜ.

2.4.2 Μαρκοβιανές Διαδικασίες Αποφάσεων (Markov Decision Processes)

Οι Μαρκοβιανές Διαδικασίες Αποφάσεων (Markov Decision Processes - MDP) παρέχουν ένα μαθηματικό πλαίσιο για τη μοντελοποίηση της λήψης αποφάσεων σε καταστάσεις όπου τα αποτελέσματα είναι εν μέρει στοχαστικά και εν μέρει υπό τον έλεγχο ενός λήπτη αποφάσεων. Χαρακτηριστικό τους αποτελεί ότι η μελλοντική κατάσταση δεν εξαρτάται από το ιστορικό των προηγούμενων καταστάσεων, αλλά μόνο από την εκάστοτε κατάσταση που βρίσκεται το σύστημα την τρέχουσα χρονική στιγμή [31]. Οι MDP είναι θεμελιώδεις για την ενισχυτική μάθηση και αποτελούν τη βάση για πολλούς αλγορίθμους της.

Τα βασικά στοιχεία των MDPs αποτελούν οι:

- **Καταστάσεις:** Ένα σύνολο πιθανών καταστάσεων ή διαμορφώσεων στο περιβάλλον.
- **Δράσεις:** Ένα σύνολο πιθανών αποφάσεων που μπορεί να λάβει ο πράκτορας.
- **Πιθανότητες μετάβασης:** Καθορίζουν την πιθανότητα μετάβασης από μια κατάσταση σε μια άλλη όταν εκτελείται μια συγκεκριμένη δράση.
- **Ανταμοιβές:** Αντιπροσωπεύουν την άμεση ανταμοιβή που λαμβάνεται μετά τη μετάβαση από μια κατάσταση σε μια άλλη λόγω μιας δράσης.
- **Παράγοντας έκπτωσης:** Μια τιμή μεταξύ 0 και 1 που καθορίζει τη σημασία των μελλοντικών ανταμοιβών [32].

Ο στόχος σε μια MDP είναι η εύρεση μιας βέλτιστης πολιτικής που μεγιστοποιεί την αναμενόμενη αθροιστική ανταμοιβή [33].

2.4.3 Δυναμικός Προγραμματισμός

Ο Δυναμικός Προγραμματισμός (Dynamic Programming - DP) είναι μια συλλογή αλγορίθμων που χρησιμοποιούνται για τον υπολογισμό βέλτιστων πολιτικών σε MDP, δεδομένου ενός πλήρους και ακριβούς μοντέλου του περιβάλλοντος [34]. Αν και οι μέθοδοι DP δεν είναι πρακτικές για προβλήματα ΕΜ μεγάλης κλίμακας λόγω των υπολογιστικών τους απαιτήσεων, παρέχουν τη βάση για πολλούς σύγχρονους αλγορίθμους ΕΜ.

Βασικοί αλγόριθμοι DP περιλαμβάνουν:

- **Αξιολόγηση πολιτικής:** Υπολογίζει τη συνάρτηση αξίας για μια δεδομένη πολιτική.
- **Βελτίωση πολιτικής:** Βελτιώνει μια πολιτική χρησιμοποιώντας την έννοια της «άπληστης πολιτικής» ως προς τη συνάρτηση αξίας της.
- **Επανάληψη πολιτικής:** Εναλλάσσεται μεταξύ αξιολόγησης πολιτικής και βελτίωσης πολιτικής για την εύρεση της βέλτιστης πολιτικής.
- **Επανάληψη αξίας:** Υπολογίζει απευθείας τη βέλτιστη συνάρτηση αξίας, από την οποία μπορεί να προκύψει μια βέλτιστη πολιτική [35].

2.4.4 Μέθοδοι Monte Carlo

Οι μέθοδοι Monte Carlo (MC) αποτελούν μια σημαντική κατηγορία αλγορίθμων στον τομέα της ΕΜ. Η βασική τους αρχή είναι η εκμάθηση από ολοκληρωμένα επεισόδια εμπειρίας, χωρίς να απαιτείται προηγούμενη γνώση της δυναμικής του περιβάλλοντος. Αυτό το χαρακτηριστικό τις καθιστά ιδιαίτερα χρήσιμες σε περιπτώσεις όπου το μοντέλο του περιβάλλοντος είναι άγνωστο ή υπερβολικά πολύπλοκο για να προσδιοριστεί με ακρίβεια [36].

Ένα από τα βασικά χαρακτηριστικά των μεθόδων MC είναι η μάθηση βασισμένη σε επεισόδια. Αυτό σημαίνει ότι οι αλγόριθμοι αυτοί ενημερώνουν τις συναρτήσεις αξίας και τις πολιτικές τους με βάση πλήρη επεισόδια αλληλεπίδρασης με το περιβάλλον. Κάθε επεισόδιο αποτελεί μια ολοκληρωμένη ακολουθία καταστάσεων, ενεργειών και ανταμοιβών, από την αρχική κατάσταση μέχρι την τελική.

Ένα άλλο σημαντικό χαρακτηριστικό των μεθόδων MC είναι η απουσία μοντέλου. Σε αντίθεση με άλλες προσεγγίσεις, οι μέθοδοι MC δεν απαιτούν ένα προκαθορισμένο μοντέλο της δυναμικής του περιβάλλοντος. Αυτό τις καθιστά ιδιαίτερα ευέλικτες και εφαρμόσιμες σε ένα ευρύ φάσμα προβλημάτων όπου η ακριβής μοντελοποίηση του περιβάλλοντος μπορεί να είναι δύσκολη ή αδύνατη.

Η δειγματοληψία αποτελεί επίσης ένα βασικό στοιχείο των μεθόδων MC. Οι αλγόριθμοι αυτοί, χρησιμοποιούν τεχνικές δειγματοληψίας για να εκτιμήσουν την αξία των καταστάσεων ή των ζευγών κατάστασης-δράσης. Μέσω της επαναλαμβανόμενης δειγματοληψίας και της συσσώρευσης εμπειρίας, οι μέθοδοι MC μπορούν να σχηματίσουν ακριβείς εκτιμήσεις των αξιών, ακόμη και σε πολύπλοκα περιβάλλοντα [37].

Τέλος, αξίζει να σημειωθεί ότι οι μέθοδοι MC έχουν ευρεία εφαρμογή τόσο στην πρόβλεψη όσο και στον έλεγχο. Στην πρόβλεψη, χρησιμοποιούνται για την εκτίμηση της συνάρτησης αξίας για μια δεδομένη πολιτική. Στον έλεγχο, αξιοποιούνται για την εύρεση της βέλτιστης πολιτικής. Αυτή η ευελιξία τις καθιστά πολύτιμα εργαλεία σε διάφορους τομείς της ΕΜ και της ΤΝ γενικότερα [38].

2.4.5 Μάθηση Χρονικής Διαφοράς

Η Μάθηση Χρονικής Διαφοράς (Temporal Difference - TD) συνδυάζει ιδέες από τον Δυναμικό Προγραμματισμό και τις μεθόδους Monte Carlo [39]. Οι μέθοδοι TD μαθαίνουν από ατελή επεισόδια με την τεχνική bootstraping (ενημέρωση εκτιμήσεων με βάση άλλες εκτιμήσεις) και μπορούν να ενημερώνουν τις εκτιμήσεις τους σε κάθε βήμα. Η ενημέρωση εκτιμήσεων πραγματοποιείται με τον υπολογισμό του σφάλματος χρονικής διαφοράς. Πρόκειται για μια γενική ιδέα, πάνω στην οποία στηρίζονται πολλοί αλγόριθμοι ΕΜ.

Μερικοί βασικοί αλγόριθμοι TD είναι:

1) **TD(0):** Ο απλούστερος αλγόριθμος TD που ενημερώνει την εκτίμησή του με βάση την αμέσως επόμενη κατάσταση.

$$V(s_t) \leftarrow V(s_t) + \alpha[r_{t+1} + \gamma V(s_{t+1}) - V(s_t)]$$

Όπου:

$V(s_t)$: Η εκτιμώμενη αξία της κατάστασης s_t

α: Ο ρυθμός μάθησης

r_{t+1}: Η άμεση ανταμοιβή

γ: Ο συντελεστής έκπτωσης για μελλοντικές ανταμοιβές

s_{t+1}: Η επόμενη κατάσταση

2) **SARSA (State-Action-Reward-State-Action):** Ένας on-policy αλγόριθμος ελέγχου TD.

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[r_{t+1} + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)]$$

Όπου:

$Q(s_t, a_t)$: Η εκτιμώμενη αξία του ζεύγους κατάστασης-ενέργειας

a_t: Η ενέργεια που εκτελείται στην κατάσταση s(t)

3) **Q-learning:** Ένας off-policy αλγόριθμος που χρησιμοποιεί temporal-difference learning για να εκτιμήσει απευθείας τη βέλτιστη συνάρτηση αξίας δράσης [40].

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

Όπου:

$\max_a Q(s_{t+1}, a)$*:* Η μέγιστη εκτιμώμενη αξία για όλες τις πιθανές ενέργειες a στην κατάσταση s(t+1)

Η μάθηση Χρονικής Διαφοράς αποτελεί βάση πολλών σύγχρονων αλγορίθμων ΕΜ λόγω της ικανότητάς της να μαθαίνει τόσο από on-policy όσο και από off-policy δεδομένα και της αποτελεσματικότητάς της στη χρήση μερικής εμπειρίας [41].

2.4.6 Σχεδιασμός και μάθηση με Μεθόδους Πινάκων

Οι Μέθοδοι Πινάκων στην ΕΜ αναφέρονται σε αλγορίθμους που αναπαριστούν συναρτήσεις αξίας ή πολιτικές χρησιμοποιώντας πίνακες ή συστοιχίες. Αυτές οι μέθοδοι είναι κατάλληλες για προβλήματα με μικρούς, διακριτούς χώρους καταστάσεων και δράσεων [42].

Βασικές έννοιες στο σχεδιασμό και τη μάθηση με Μεθόδους Πινάκων περιλαμβάνουν:

1. **Μάθηση βασισμένη σε μοντέλο έναντι μάθησης χωρίς μοντέλο:** Η μάθηση βασισμένη σε μοντέλο χρησιμοποιεί ένα μοντέλο του περιβάλλοντος για τη δημιουργία προσομοιωμένων εμπειριών, ενώ η μάθηση χωρίς μοντέλο μαθαίνει απευθείας από την εμπειρία χωρίς να μοντελοποιεί ρητά το περιβάλλον [43].

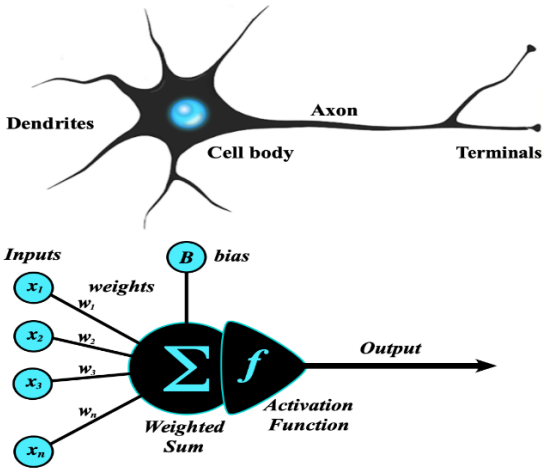
2. **Αλγόριθμος Dyna-Q:** Ένας ολοκληρωμένος αλγόριθμος σχεδιασμού, δράσης και μάθησης που χρησιμοποιεί ένα μοντέλο, το οποίο έχει μάθει, για τη δημιουργία προσομοιωμένων εμπειριών [44].

3. **Σάρωση Προτεραιότητας (Prioritized Sweeping):** Ένας αλγόριθμος σχεδιασμού που επικεντρώνει τους υπολογιστικούς πόρους στις πιο υποσχόμενες ή αβέβαιες περιοχές του χώρου καταστάσεων [45].

Αν και οι Μέθοδοι Πινάκων είναι περιορισμένες στην εφαρμογή τους σε προβλήματα μεγάλης κλίμακας, παρέχουν σημαντικές γνώσεις και αποτελούν τη βάση για πιο προηγμένες τεχνικές ΕΜ [46].

2.5 Τεχνητά Νευρωνικά Δίκτυα και Βαθιά Μάθηση

Τα Τεχνητά Νευρωνικά Δίκτυα (ΤΝΔ) είναι υπολογιστικά μοντέλα εμπνευσμένα από τη δομή και τη λειτουργία των βιολογικών νευρωνικών δικτύων [47]. Αποτελούνται από διασυνδεδεμένους κόμβους (νευρώνες) οργανωμένους σε επίπεδα, ικανούς να μάθουν πολύπλοκα μοτίβα και αναπαραστάσεις από δεδομένα.



Εικόνα 5 - Νευρώνες βιολογικού και τεχνητού νευρωνικού δικτύου[1]

Οι θεμελιώδεις έννοιες των ΤΝΔ περιλαμβάνουν:

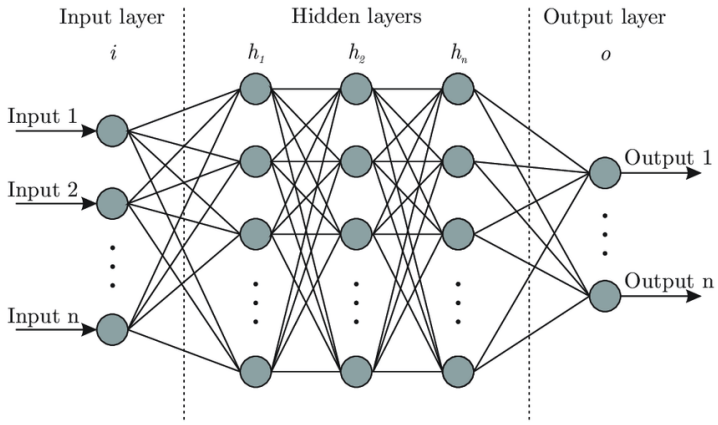
- **Νευρώνες:** Βασικές υπολογιστικές μονάδες που λαμβάνουν εισόδους, εφαρμόζουν μια συνάρτηση ενεργοποίησης και παράγουν μια έξοδο.

- **Επίπεδα:** Ομάδες νευρώνων, με ένα τυπικό νευρωνικό δίκτυο να αποτελείται από ένα επίπεδο εισόδου, ένα ή περισσότερα κρυφά επίπεδα και ένα επίπεδο εξόδου.

- **Βάρη (Weights):** Παράμετροι που καθορίζουν την ισχύ των συνδέσεων μεταξύ των νευρώνων και προσαρμόζονται κατά τη διάρκεια της εκπαίδευσης.

- **Μεροληψίες (Biases):** Αντιστοιχούν σε κάθε νευρώνα ξεχωριστά και είναι σταθερές τιμές. Συμβάλλουν στην αποτελεσματική μάθηση του δικτύου, επιτρέποντάς του να προσαρμόζεται σε μη γραμμικά δεδομένα και να μαθαίνει πιο σύνθετα μοτίβα.

- **Συναρτήσεις ενεργοποίησης:** Μη γραμμικές συναρτήσεις που εφαρμόζονται στο σταθμισμένο άθροισμα των εισόδων για την εισαγωγή μη γραμμικότητας στο δίκτυο [48].



Εικόνα 6 - Αρχιτεκτονική Τεχνητού Νευρωνικού Δικτύου [49]

[1] https://s.mriquestions.com/what-is-a-neural-network.html

Η εκπαίδευση των νευρωνικών δικτύων περιλαμβάνει τη χρήση αλγορίθμων, όπως η οπισθοδιάδοση (backpropagation) για τον αποτελεσματικό υπολογισμό των κλίσεων στα νευρωνικά δίκτυα, επιτρέποντας την προσαρμογή των βαρών και των μεροληψιών για την ελαχιστοποίηση της συνάρτησης απώλειας. Η κατάβαση κλίσης (gradient descent) είναι ένας αλγόριθμος βελτιστοποίησης που χρησιμοποιείται για την ελαχιστοποίηση της συνάρτησης απώλειας προσαρμόζοντας επαναληπτικά τις παραμέτρους του δικτύου. Οι συναρτήσεις απώλειας είναι ποσοτικά μέτρα και αξιολογούν τις αποκλίσεις των προβλέψεων του δικτύου με τις πραγματικές τιμές. Οι πιο συνηθισμένες συναρτήσεις απώλειας περιλαμβάνουν το Μέσο Τετραγωνικό Σφάλμα (Mean Squared Error) για παλινδρόμηση και την Εντροπία Διασταύρωσης (Cross-Entropy) για ταξινόμηση [50].

Η Βαθιά Μάθηση αναφέρεται στη χρήση νευρωνικών δικτύων με πολλαπλά κρυφά επίπεδα (βαθιά νευρωνικά δίκτυα) για την εκμάθηση ιεραρχικών αναπαραστάσεων των δεδομένων [51]. Η Βαθιά Μάθηση έχει επιτύχει κορυφαίες επιδόσεις σε διάφορους τομείς, συμπεριλαμβανομένης της όρασης υπολογιστών, της επεξεργασίας φυσικής γλώσσας και της αναγνώρισης ομιλίας [52].

Βασικές αρχιτεκτονικές βαθιάς μάθησης περιλαμβάνουν:

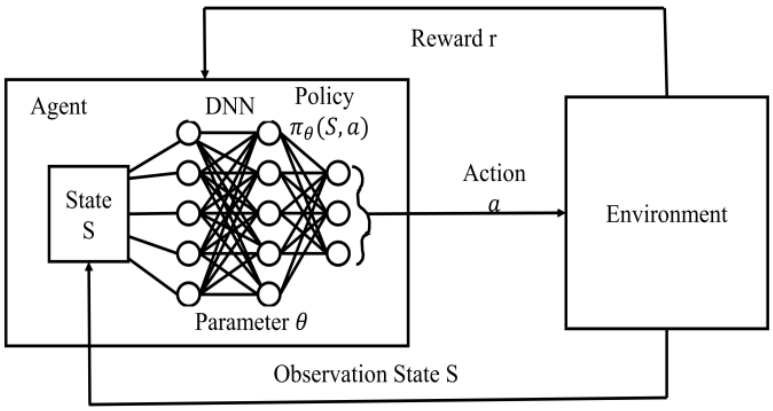
- **Συνελικτικά Νευρωνικά Δίκτυα (Convolutional Neural Networks):** Εξειδικευμένα για την επεξεργασία δεδομένων πλέγματος, όπως εικόνες [53].

- **Επαναλαμβανόμενα Νευρωνικά Δίκτυα (Recurrent Neural Networks):** Σχεδιασμένα για ακολουθιακά δεδομένα, με παραλλαγές όπως τα Long Short-Term Memory και Gated Recurrent Unit ικανά να συλλάβουν μακροπρόθεσμες συσχετίσεις στα δεδομένα [54], [55].

- **Μετασχηματιστές (Transformers):** Μια πιο πρόσφατη αρχιτεκτονική που έχει δείξει εξαιρετική απόδοση σε εργασίες επεξεργασίας φυσικής γλώσσας [56].

2.6 Βαθιά Ενισχυτική Μάθηση

Η Βαθιά Ενισχυτική Μάθηση (BEM) συνδυάζει τις δυνατότητες αναπαράστασης μάθησης των βαθιών νευρωνικών δικτύων με το πλαίσιο λήψης αποφάσεων της ενισχυτικής μάθησης [57]. Αυτός ο ισχυρός συνδυασμός έχει οδηγήσει σε σημαντικές εξελίξεις σε διάφορους τομείς, συμπεριλαμβανομένου της θεωρίας παιγνίων, της ρομποτικής και της διαχείρισης πόρων [58].



Εικόνα 7 - Πράκτορας Βαθιάς Ενισχυτικής Μάθησης [59]

Βασικές έννοιες που συναντάμε σε αλγορίθμους ΒΕΜ:

1. **Προσεγγιστική συνάρτηση:** Χρήση νευρωνικών δικτύων για την προσέγγιση συναρτήσεων αξίας ή πολιτικών, επιτρέποντας τη γενίκευση σε μη εμφανείς καταστάσεις [60].

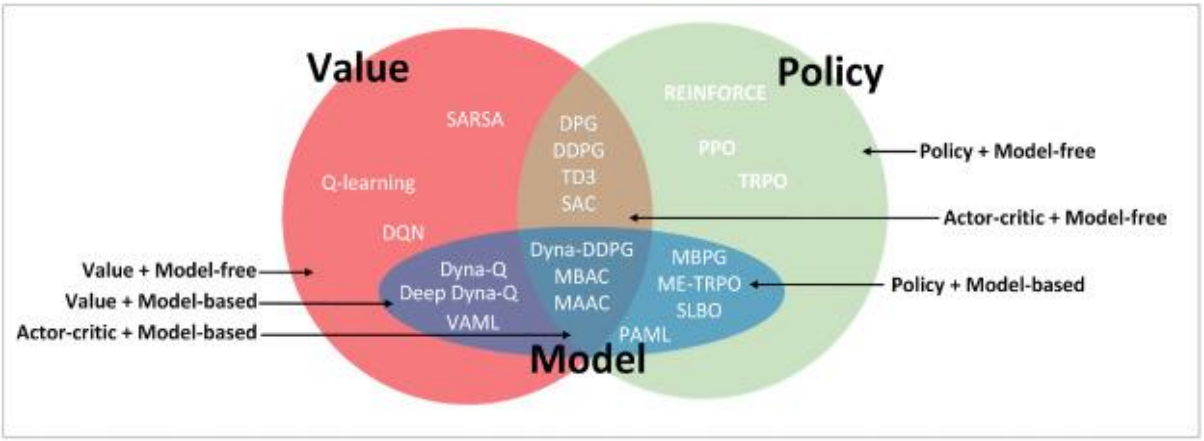
2. **Επανάληψη εμπειρίας:** Τεχνική που αποθηκεύει και επαναχρησιμοποιεί παρελθούσες εμπειρίες για τη βελτίωση της αποδοτικότητας των δειγμάτων και της σταθερότητας στη μάθηση [61].

3. **Δίκτυα-στόχοι:** Ξεχωριστά δίκτυα που χρησιμοποιούνται για τον υπολογισμό τιμών-στόχων, βελτιώνοντας τη σταθερότητα της μάθησης [62].

2.6.1 Εισαγωγή σε αλγορίθμους Βαθιάς Ενισχυτικής Μάθησης

Τα Βαθιά Δίκτυα-Q (Deep Q-Networks -DQN) είναι ένας θεμελιώδης αλγόριθμος στη βαθιά ενισχυτική μάθηση που συνδυάζει τη μάθηση Q με βαθιά νευρωνικά δίκτυα. Βασικές καινοτομίες στο DQN περιλαμβάνουν την επανάληψη εμπειρίας, η οποία αποθηκεύει και δειγματοληπτεί τυχαία παρελθούσες εμπειρίες για τη συγκέντρωση συσχετίσεων στα δεδομένα και τη βελτίωση της σταθερότητας μάθησης, καθώς και το δίκτυο-στόχο, ένα ξεχωριστό δίκτυο που χρησιμοποιείται για τον υπολογισμό τιμών Q-στόχων, ενημερωμένο περιοδικά για τη μείωση της αποσταθεροποίησης κατά την εκπαίδευση [63].

Οι μέθοδοι κλίσης πολιτικής βελτιστοποιούν άμεσα την πολιτική χωρίς να μαθαίνουν μια συνάρτηση αξίας [64]. Αυτές οι μέθοδοι είναι ιδιαίτερα χρήσιμες για συνεχείς χώρους δράσεων και μπορούν να μάθουν στοχαστικές πολιτικές. Βασικοί αλγόριθμοι κλίσης πολιτικής περιλαμβάνουν τον REINFORCE, μια απλή μέθοδο κλίσης πολιτικής που χρησιμοποιεί εκτιμήσεις Monte Carlo [65], τις μεθόδους Actor-Critic, οι οποίες συνδυάζουν ενημερώσεις κλίσης πολιτικής με μαθημένες συναρτήσεις αξίας για τη μείωση της διακύμανσης στις εκτιμήσεις κλίσης [66], και τη Βελτιστοποίηση Εγγύς Πολιτικής (Proximal Policy Optimization - PPO), μια μέθοδο βελτιστοποίησης πολιτικής που χρησιμοποιεί έναν περικομμένο υποκατάστατο στόχο (clip range) για τη διασφάλιση σταθερών ενημερώσεων [67].



Εικόνα 8 - Κατηγοριοποίηση αλγορίθμων Βαθιάς Ενισχυτικής Μάθησης [68]

Προηγμένοι αλγόριθμοι ΒΕΜ περιλαμβάνουν τη Βαθιά Ντετερμινιστική Κλίση Πολιτικής (Deep Deterministic Policy Gradient - DDPG), έναν αλγόριθμο off-policy για συνεχείς χώρους δράσεων που συνδυάζει ιδέες από το DQN και τις ντετερμινιστικές κλίσεις πολιτικής, τον Twin Delayed DDPG (TD3), μια βελτίωση του DDPG που αντιμετωπίζει την υπερεκτίμηση στην προσέγγιση της συνάρτησης Q , και τον Soft Actor-Critic (SAC), έναν αλγόριθμο off-policy που μεγιστοποιεί τόσο την αναμενόμενη απόδοση όσο και την εντροπία, ενισχύοντας έτσι την ικανότητα εξερεύνησης και την ανθεκτικότητα του συστήματος [69], [70].

2.6.2 Ζητήματα στις τεχνικές Βαθιάς Ενισχυτικής Μάθησης

Παρόλο που η βαθιά ενισχυτική μάθηση έχει επιτύχει αξιοσημείωτες επιτυχίες, παραμένουν αρκετές προκλήσεις. Η αποδοτικότητα των δειγμάτων αποτελεί σημαντικό ζήτημα, καθώς πολλοί αλγόριθμοι ΒΕΜ απαιτούν μεγάλο αριθμό αλληλεπιδράσεων με το περιβάλλον για να μάθουν αποτελεσματικά. Η σταθερότητα και η αναπαραγωγιμότητα (reproducibility) παραμένουν προκλήσεις, καθώς οι αλγόριθμοι ΒΕΜ μπορεί να είναι ευαίσθητοι σε υπερπαραμέτρους και αρχικές συνθήκες, καθιστώντας δύσκολη την αναπαραγωγή των αποτελεσμάτων. Η εξερεύνηση σε χώρους υψηλών διαστάσεων παραμένει μια σημαντική πρόκληση, ιδιαίτερα σε περιβάλλοντα όπου οι ανταμοιβές συλλέγονται σε αραιά διαστήματα, όπως στο τέλος κάθε επεισοδίου. Επιπλέον, η βελτίωση της ικανότητας των πρακτόρων ΒΕΜ να μεταφέρουν γνώση μεταξύ διαφορετικών εργασιών και να «γενικεύουν» σε νέα περιβάλλοντα αποτελεί ένα σημαντικό πεδίο έρευνας [71].

Οι μελλοντικές κατευθύνσεις έρευνας στη ΒΕΜ περιλαμβάνουν την ενσωμάτωση μαθημένων μοντέλων του περιβάλλοντος για τη βελτίωση της αποδοτικότητας των δειγμάτων και του σχεδιασμού. Η μετα-μάθηση, που στοχεύει στην ανάπτυξη αλγορίθμων που μπορούν να προσαρμόζονται γρήγορα σε νέες εργασίες ή περιβάλλοντα, αποτελεί επίσης ένα ενεργό πεδίο έρευνας. Τέλος, η επέκταση της ΒΕΜ σε σενάρια που περιλαμβάνουν πολλαπλούς αλληλοεπιδρώντες πράκτορες, γνωστή ως πολυπρακτορική Ενισχυτική Μάθηση, είναι ένας άλλος σημαντικός τομέας εξέλιξης [72].

2.6.3 Εφαρμογή Βαθιάς Ενισχυτικής Μάθησης σε προβλήματα Υπολογιστικής Εκφόρτωσης

Η ενσωμάτωση τεχνικών ΒΕΜ στο πρόβλημα της υπολογιστικής εκφόρτωσης παρουσιάζει ορισμένες ευκαιρίες και προκλήσεις. Η ικανότητα των αλγορίθμων ΒΕΜ να μαθαίνουν πολύπλοκες στρατηγικές λήψης αποφάσεων τους καθιστά ιδιαίτερα κατάλληλους για τη βελτιστοποίηση της κατανομής πόρων σε δυναμικά περιβάλλοντα ΜΕC. Ωστόσο, η εφαρμογή αυτών των τεχνικών σε πραγματικά συστήματα απαιτεί προσεκτική εξέταση ορισμένων ζητημάτων.

Στα επόμενα κεφάλαια, θα διερευνήσουμε πώς οι αρχές και οι τεχνικές που παρουσιάστηκαν σε αυτή τη βιβλιογραφική επισκόπηση μπορούν να εφαρμοστούν στο συγκεκριμένο πρόβλημα της αποφόρτισης υπολογισμών. Επιπλέον, θα εξετάσουμε τον τρόπο με τον οποίο τα Βαθιά Νευρωνικά Δίκτυα μπορούν να χρησιμοποιηθούν για την μοντελοποίηση σύνθετων περιβαλλόντων ΜΕC, και πώς οι αλγόριθμοι ενισχυτικής

μάθησης μπορούν να προσαρμοστούν για να λαμβάνουν αποφάσεις αποφόρτισης σε πραγματικό χρόνο. Τέλος, θα διερευνήσουμε τρόπους αντιμετώπισης των προκλήσεων που σχετίζονται με την εφαρμογή αυτών των τεχνικών σε πραγματικά συστήματα, όπως η διαχείριση της αβεβαιότητας, η προσαρμογή σε μεταβαλλόμενες συνθήκες δικτύου και η εξισορρόπηση πολλαπλών, συχνά αντικρουόμενων, στόχων [73].

Συμπερασματικά, ο τομέας της ΕΜ, και ιδιαίτερα της ΒΕΜ, προσφέρει ισχυρά εργαλεία για την επίλυση σύνθετων προβλημάτων λήψης αποφάσεων. Η ενσωμάτωση τεχνικών ΒΕΜ στο πρόβλημα της υπολογιστικής εκφόρτωσης παρουσιάζει ορισμένες ευκαιρίες και προκλήσεις .

2.7 Ανάγκη για Υπολογιστική Εκφόρτωση

Με την έλευση νέων τεχνολογιών και τον πολλαπλασιασμό των συσκευών υπολογισμού και αποθήκευσης, έχουν αναδυθεί νέες εφαρμογές και σενάρια που απαιτούν υψηλή υπολογιστική ισχύ και χαμηλή καθυστέρηση. Αυτές οι απαιτήσεις έχουν οδηγήσει στην ανάπτυξη νέων παραδειγμάτων υπολογισμού, όπως το Υπολογιστικό Νέφος (Cloud Computing) και το MEC [74].

2.7.1 Υπολογιστικό Νέφος

Το Υπολογιστικό Νέφος είναι ένα μοντέλο που επιτρέπει την πρόσβαση σε κοινόχρηστους υπολογιστικούς πόρους (π.χ., δίκτυα, διακομιστές, αποθηκευτικά μέσα, εφαρμογές και υπηρεσίες) μέσω δικτύου, με ελάχιστη προσπάθεια διαχείρισης ή αλληλεπίδραση με τον πάροχο υπηρεσιών [75].

Τα βασικά μοντέλα υπηρεσιών του Υπολογιστικού Νέφους περιλαμβάνουν:

- **Υποδομή ως Υπηρεσία (Infrastructure as a Service):** Παρέχει εικονικοποιημένους υπολογιστικούς πόρους μέσω του διαδικτύου.
- **Πλατφόρμα ως Υπηρεσία (Platform as a Service):** Προσφέρει μια πλατφόρμα και περιβάλλον για την ανάπτυξη εφαρμογών.
- **Λογισμικό ως Υπηρεσία (Software as a Service):** Παρέχει πρόσβαση σε λογισμικό εφαρμογών μέσω διαδικτύου.

Παρά τα πλεονεκτήματά του, το υπολογιστικό νέφος αντιμετωπίζει προκλήσεις, όπως η καθυστέρηση δικτύου και η ασφάλεια δεδομένων, οι οποίες οδήγησαν στην ανάπτυξη της κινητής υπολογιστικής στα άκρα του δικτύου.

2.7.2 Κινητή υπολογιστική παρυφής

Το MEC εισήχθη ως ένα νέο παράδειγμα που συμπληρώνει και επεκτείνει το υπολογιστικό νέφος στα άκρα του δικτύου. Η υπολογιστική εκφόρτωση αποτελεί μια κρίσιμη τεχνική στο πλαίσιο του MEC, η οποία επιτρέπει τη μεταφορά υπολογιστικά απαιτητικών εργασιών από συσκευές με περιορισμένους πόρους σε ισχυρότερους υπολογιστικούς

κόμβους στα άκρα του δικτύου. Ο κύριος στόχος αυτής της προσέγγισης είναι η βελτιστοποίηση της χρήσης των διαθέσιμων πόρων, η μείωση της κατανάλωσης ενέργειας στις τερματικές συσκευές και η βελτίωση της συνολικής απόδοσης των εφαρμογών [76].

Το MEC ορίστηκε αρχικά ως "mobile-edge computing" το 2014 από το Ευρωπαϊκό Ινστιτούτο Τηλεπικοινωνιακών Προτύπων (European Telecommunications Standards Institute). Αργότερα, το 2017, αντικατέστησε τον όρο "mobile" με τον όρο "multi-access" για να αντικατοπτρίσει τις πρόσθετες περιπτώσεις χρήσης που θα αντιμετώπιζε το MEC, συμπεριλαμβανομένης της υποστήριξης υπολογισμού και αποθήκευσης για συσκευές τελικών χρηστών μέσω άλλων τεχνολογιών, όπως οπτικές ίνες, μικροκομματικές επικοινωνίες, και δορυφορική σύνδεση [77].

Τα βασικά χαρακτηριστικά του MEC περιλαμβάνουν:

- **Εγγύτητα:** Οι διακομιστές MEC βρίσκονται κοντά στους τελικούς χρήστες, μειώνοντας την καθυστέρηση.
- **Υψηλό εύρος ζώνης:** Επιτρέπει τη μεταφορά μεγάλου όγκου δεδομένων με υψηλές ταχύτητες.
- **Επίγνωση τοποθεσίας:** Οι υπηρεσίες μπορούν να προσαρμοστούν με βάση την τοποθεσία του χρήστη.
- **Επίγνωση δικτύου:** Επιτρέπει τη βελτιστοποίηση των υπηρεσιών με βάση τις συνθήκες του δικτύου [78].

Το MEC έχει υιοθετήσει τεχνολογίες όπως η Δικτύωση Καθοριζόμενη από Λογισμικό (Software Defined Networking - SDN) και την εικονικοποίηση [79]. Το SDN και η Εικονικοποίηση Δικτυακών Λειτουργιών (Network Function Virtualization - NFV) είναι μια λογική προσθήκη στους κόμβους MEC καθώς παρέχουν δυνατότητες έξυπνης ενσωμάτωσης δικτύου και βελτιστοποίησης πόρων [80].

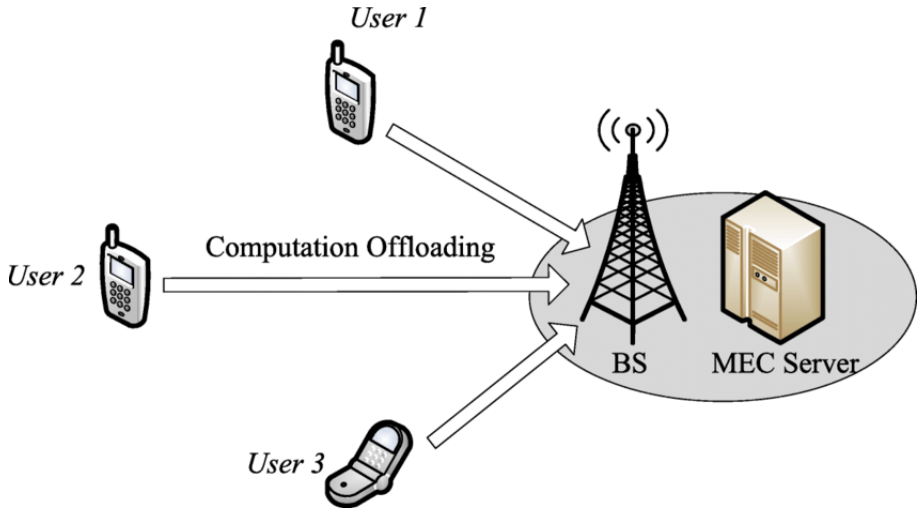
2.8 Ανάλυση Σεναρίων Υπολογιστικής Εκφόρτωσης

Η υπολογιστική εκφόρτωση, γνωστή και ως κατανομή εργασιών (task allocation), αποτελεί μια κρίσιμη τεχνική στο πλαίσιο του MEC, επιτρέποντας τη μεταφορά υπολογιστικά απαιτητικών εργασιών από συσκευές με περιορισμένους πόρους σε ισχυρότερους υπολογιστικούς κόμβους στα άκρα του δικτύου. Ο κύριος στόχος αυτής της προσέγγισης είναι η βελτιστοποίηση της χρήσης των διαθέσιμων πόρων, η μείωση της κατανάλωσης ενέργειας στις τερματικές συσκευές και η βελτίωση της συνολικής απόδοσης των εφαρμογών.

2.8.1 Βασικές έννοιες και προκλήσεις

Η υπολογιστική εκφόρτωση περιλαμβάνει τη λήψη αποφάσεων σχετικά με το ποιες εργασίες θα εκτελεστούν τοπικά στη συσκευή και ποιες θα μεταφερθούν στους διακομιστές ακμής. Αυτή η διαδικασία λήψης αποφάσεων πρέπει να λαμβάνει υπόψη διάφορους παράγοντες, όπως:

- Υπολογιστικές απαιτήσεις της εργασίας
- Διαθέσιμοι πόροι στη συσκευή και στους διακομιστές ακμής
- Συνθήκες δικτύου (bandwidth, latency)
- Κατανάλωση ενέργειας
- Χρονικοί περιορισμοί της εφαρμογής



Εικόνα 9 - Σύστημα Υπολογιστικής στα Άκρα του δικτύου [81]

Η αποτελεσματική υλοποίηση της υπολογιστικής εκφόρτωσης σε περιβάλλοντα MEC παρουσιάζει σημαντικές προκλήσεις, συμπεριλαμβανομένης της δυναμικής φύσης των δικτύων κινητής τηλεφωνίας, της ετερογένειας των συσκευών, των απαιτήσεων των εφαρμογών, καθώς και των περιορισμών σε εύρος ζώνης και ενέργεια. Ως εκ τούτου, η ανάπτυξη αποτελεσματικών στρατηγικών εκφόρτωσης και βελτιστοποίησης κατανομής εργασιών αποτελεί ένα ενεργό πεδίο έρευνας με σημαντικές επιπτώσεις στην απόδοση και την αποτελεσματικότητα των σύγχρονων κινητών δικτύων [82].

Αναζητώντας σχετική βιβλιογραφία σε υπολογιστική στα άκρα του δικτύου, παρατηρούμε τον πρώτο διαχωρισμό στις κατηγορίες υλοποίησης των προβλημάτων υπολογιστικής εκφόρτωσης. Πρόκειται για την παραγωγή εργασιών και χωρίζεται σε δυναμική και στατική. Επίσης, άλλο ένα κριτήριο διαφοροποίησης αποτελεί η στρατηγική εκφόρτωσης και συγκεκριμένα αν πρόκειται για πλήρης ή μερική, καθώς και η στρατηγική συνεργασίας αν είναι συνεργατική ή όχι. Τέλος, και ίσως πιο σημαντικό χαρακτηριστικό κατηγοριοποίησης αποτελεί ο στόχος βελτιστοποίησης, ο οποίος συνήθως εστιάζει σε βελτιστοποίηση ενέργειας ή χρονικής καθυστέρησης. Παρακάτω θα αναλυθούν κάποια από αυτά τα χαρακτηριστικά [83].

2.8.2 Σενάρια πλήρους και μερικής Υπολογιστικής Εκφόρτωσης

Η παρούσα διπλωματική εργασία, επικεντρώνεται στα σενάρια πλήρους εκφόρτωσης, όπου η απόφαση εκφόρτωσης μιας υπολογιστικής εργασίας μοντελοποιείται ως δυαδική επιλογή: είτε η εργασία εκτελείται τοπικά στην συσκευή χρήστη, είτε πραγματοποιείται πλήρης εκφόρτωση στον διακομιστή ακμής. Αυτή η προσέγγιση προσφέρει απλότητα και πρακτικότητα, μειώνοντας την πολυπλοκότητα της διαδικασίας λήψης αποφάσεων και

ευθυγραμμίζεται με τις δυνατότητες πολλών σύγχρονων εφαρμογών κινητής τηλεφωνίας [84].

Η εργασία [85] προτείνει έναν κατανεμημένο αλγόριθμο υπολογιστικής εκφόρτωσης που λαμβάνει υπόψη αυτή τη δυαδική απόφαση. Η εργασία τους δείχνει ότι ακόμη και με αυτό το απλοποιημένο μοντέλο, μπορούν να επιτευχθούν σημαντικές βελτιώσεις στην ενεργειακή απόδοση και τον χρόνο επεξεργασίας.

Στη βιβλιογραφία υπάρχουν προσεγγίσεις μερικής εκφόρτωσης, όπως στην εργασία [86], όπου η υπολογιστική εργασία κατανέμεται μεταξύ της τοπικής συσκευής και ενός απομακρυσμένου διακομιστή. Σε αυτές τις περιπτώσεις, μέρος της εργασίας εκτελείται τοπικά, ενώ το υπόλοιπο εκφορτώνεται για επεξεργασία σε εξωτερική υποδομή, επιτυγχάνοντας έτσι μια ισορροπημένη χρήση των διαθέσιμων πόρων. Ωστόσο λήφθηκε η απόφαση στην παρούσα εργασία να γίνει εστίαση στα σενάρια πλήρους εκφόρτωσης, καθώς προσφέρουν ένα καλό σημείο εκκίνησης για την ανάπτυξη και αξιολόγηση αλγορίθμων βελτιστοποίησης.

2.8.3 Στρατηγικές συνεργασίας στην Υπολογιστική Εκφόρτωση

Η Υπολογιστική Εκφόρτωση σε περιβάλλοντα MEC μπορεί να επωφεληθεί σημαντικά από διάφορες στρατηγικές συνεργασίας. Σε αυτήν την υποενότητα θα εξετάσουμε τρεις βασικές στρατηγικές συνεργασίας: της επικοινωνιακής, της υπολογιστικής και της μη συνεργατικής.

Η συνεργασία υπολογισμού μπορεί να διαχωριστεί σε δύο κατηγορίες:

1. **Συνεργασία Υπολογισμού με τον υπολογιστικό κόμβο:** Αναφέρεται στη συνεργατική επεξεργασία εργασιών που εκφορτώνονται από τις συσκευές χρήστη (User Equipment – UE) σε διαφορετικούς απομακρυσμένους κόμβους υπολογιστικής παρυφής ή νέφους. Αυτή η προσέγγιση είναι ιδιαίτερα χρήσιμη σε πολύπλοκα συστήματα πολλαπλών χρηστών και πολλαπλών εργασιών, όπου οι απαιτήσεις των εργασιών μπορεί να ποικίλλουν σημαντικά. Για παράδειγμα, σε σενάριο με εργασίες που έχουν υψηλές υπολογιστικές απαιτήσεις αλλά χωρίς αυστηρούς περιορισμούς καθυστέρησης μπορούν να εκφορτωθούν σε διακομιστές νέφους, ενώ εργασίες που απαιτούν χαμηλή καθυστέρηση μπορούν να γίνουν εκφόρτωση σε διακομιστές MEC.

2. **Συνεργασία Υπολογισμού μεταξύ συσκευών χρηστών:** Αναφέρεται στη συνεργατική επεξεργασία εργασιών μεταξύ διαφορετικών συσκευών. Αυτή η συνεργασία συχνά εμφανίζεται σε σενάρια οχημάτων. Η έρευνα σε αυτόν τον τομέα επικεντρώνεται στην κατανομή των αδρανών υπολογιστικών πόρων μεταξύ των συσκευών χρηστών.

Η συνεργασία επικοινωνίας μπορεί επίσης να διαχωριστεί σε δύο κατηγορίες:

1. **Συνεργασία επικοινωνίας μεταξύ συσκευών χρηστών:** Αναφέρεται στη συνεργασία μεταξύ διαφορετικών UE για τη συνεργατική μετάδοση δεδομένων, γνωστή και ως συνεργασία πολλαπλών αλμάτων. Αυτή η προσέγγιση είναι ιδιαίτερα χρήσιμη όταν ορισμένες UE δεν μπορούν να συνδεθούν απευθείας με τον κόμβο υπολογιστικής λόγω περιορισμένης συνδεσιμότητας ή κάλυψης δικτύου. Σε τέτοιες περιπτώσεις, ορισμένες UE λειτουργούν ως κόμβοι αναμετάδοσης για να βοηθήσουν άλλες UE στη μετάδοση των δεδομένων τους στο κόμβο. Αυτή η

στρατηγική χρησιμοποιείται συχνά σε σενάρια που περιλαμβάνουν Μη Επανδρωμένα Αεροχήματα και άλλα οχήματα.

2. **Συνεργασία επικοινωνίας μεταξύ υπολογιστικών κόμβων:** Αναφέρεται στη συνεργασία μεταξύ διαφορετικών κόμβων για τη συνεργατική μετάδοση δεδομένων. Σε αυτό το πλαίσιο, οι ερευνητές επικεντρώνονται στο πρόβλημα επιλογής κόμβων για τη μετάδοση δεδομένων .

Οι στρατηγικές συνεργασίας στην υπολογιστική εκφόρτωση προσφέρουν σημαντικές δυνατότητες για τη βελτίωση της απόδοσης και της αποτελεσματικότητας των συστημάτων MEC. Ωστόσο, η εφαρμογή τους εγείρει νέες προκλήσεις, όπως η επιλογή των κατάλληλων κόμβων συνεργασίας, η διαχείριση της πολυπλοκότητας του συστήματος και η εξασφάλιση της ασφάλειας και της ιδιωτικότητας των δεδομένων [87].

Τέλος στην 3η κατηγορία, δηλαδή στη μη συνεργατική στρατηγική, εντάσσονται οι περιπτώσεις όπου δεν πραγματοποιείται κάποια συνεργασία μεταξύ οντοτήτων. Σε αυτή την κατηγορία ανήκει και η υλοποίηση αυτής της διπλωματικής, καθώς έχει στόχο την προσέγγιση του θέματος με βάση την επιλογή του τρόπου επίλυσης του προβλήματος βελτιστοποίησης και όχι της αύξησης πολυπλοκότητας του περιβάλλοντος.

2.8.4 Τεχνικές Βελτιστοποίησης Υπολογιστικής Εκφόρτωσης

Η βελτιστοποίηση της Υπολογιστικής Εκφόρτωσης σε περιβάλλοντα MEC αποτελεί ένα πολύπλοκο πρόβλημα που απαιτεί προηγμένες τεχνικές επίλυσης. Ενώ η παρούσα εργασία επικεντρώνεται στη χρήση μεθόδων ΕΜ, είναι σημαντικό να αναγνωρίσουμε ότι υπάρχει ένα ευρύ φάσμα προσεγγίσεων που έχουν εφαρμοστεί με επιτυχία σε αυτόν τον τομέα. Η κατανόηση αυτών των εναλλακτικών μεθόδων μπορεί να προσφέρει πολύτιμες γνώσεις και να διευρύνει την οπτική μας για την αντιμετώπιση τέτοιων προβλημάτων βελτιστοποίησης.

Οι Εξελικτικοί Αλγόριθμοι (Evolutionary Algorithms - EA) αποτελούν μια σημαντική κατηγορία μεθόδων βελτιστοποίησης που εμπνέονται από τις αρχές της βιολογικής εξέλιξης. Αυτοί οι αλγόριθμοι λειτουργούν μέσω της επαναληπτικής βελτίωσης ενός πληθυσμού πιθανών λύσεων, εφαρμόζοντας μηχανισμούς όπως η επιλογή, ο ανασυνδυασμός και η μετάλλαξη. Ένας χαρακτηριστικός αλγόριθμος αυτής της κατηγορίας είναι ο Γενετικός Αλγόριθμος (Genetic Algorithm - GA). Οι GAs έχουν εφαρμοστεί με επιτυχία σε προβλήματα υπολογιστικής εκφόρτωσης, όπως για παράδειγμα στην εργασία [88], όπου χρησιμοποιήθηκαν για τη βελτιστοποίηση των αποφάσεων εκφόρτωσης και την κατανομή πόρων σε περιβάλλοντα MEC. Η ικανότητα των EA να χειρίζονται πολύπλοκους χώρους αναζήτησης και να βρίσκουν καλές λύσεις σε εύλογο χρονικό διάστημα τους καθιστά ιδιαίτερα χρήσιμους για την αντιμετώπιση των πολυδιάστατων προκλήσεων της υπολογιστικής εκφόρτωσης.

Μια άλλη σημαντική κατηγορία προσεγγίσεων είναι οι Αλγόριθμοι Νοημοσύνης Σμήνους (Swarm Intelligence Algorithms - SIA), οι οποίοι βασίζονται στη συλλογική συμπεριφορά αποκεντρωμένων, αυτο-οργανωμένων συστημάτων. Δύο αντιπροσωπευτικοί αλγόριθμοι αυτής της κατηγορίας είναι η Βελτιστοποίηση Σμήνους Σωματιδίων (Particle Swarm Optimization - PSO) και η Βελτιστοποίηση Αποικίας Μυρμηγκιών (Ant Colony Optimization - ACO). Η PSO, που μιμείται τη συμπεριφορά σμηνών στη φύση, έχει εφαρμοστεί σε προβλήματα υπολογιστικής εκφόρτωσης για τη βελτιστοποίηση της κατανομής πόρων και τη μείωση της καθυστέρησης. Επιπλέον, η ACO έχει χρησιμοποιηθεί

για τη βελτιστοποίηση των διαδρομών μετάδοσης δεδομένων και την κατανομή εργασιών σε δίκτυα MEC, όπως παρουσιάζεται στην έρευνα [89]. Η ικανότητα των SIA να προσαρμόζονται σε δυναμικά μεταβαλλόμενα περιβάλλοντα τους καθιστά ιδιαίτερα κατάλληλους για τα συνεχώς εξελισσόμενα σενάρια που συναντώνται στα σύγχρονα δίκτυα MEC .

Οι ανωτέρω εναλλακτικές προσεγγίσεις προσφέρουν διαφορετικούς τρόπους αντιμετώπισης του προβλήματος της υπολογιστικής εκφόρτωσης. Κάθε μέθοδος έχει τα δικά της πλεονεκτήματα και μπορεί να είναι ιδιαίτερα χρήσιμη σε συγκεκριμένες καταστάσεις. Για παράδειγμα, οι Εξελικτικοί Αλγόριθμοι μπορεί να είναι καλοί στο να βρίσκουν λύσεις σε πολύ περίπλοκα προβλήματα, ενώ οι Αλγόριθμοι Νοημοσύνης Σμήνους μπορεί να είναι αποτελεσματικοί σε καταστάσεις όπου το περιβάλλον αλλάζει συνεχώς. Η επιλογή της καταλληλότερης μεθόδου εξαρτάται από τις συγκεκριμένες ανάγκες και περιορισμούς του εκάστοτε προβλήματος υπολογιστικής εκφόρτωσης [89].

Κεφάλαιο 3: Διατύπωση και Επίλυση του Προβλήματος Υπολογιστικής Εκφόρτωσης

Σε αυτό το κεφάλαιο πραγματοποιείται η διατύπωση του προβλήματος της υπολογιστικής εκφόρτωσης μέσω της επεξήγησης βασικών εννοιών και του σχεδιασμού της προτεινόμενης υλοποίησης. Στη συνέχεια, παρουσιάζονται τα βασικά εργαλεία και οι βιβλιοθήκες που αξιοποιήθηκαν για την ανάπτυξη του συστήματος βελτιστοποίησης σε περιβάλλον MEC. Τέλος, παρατίθεται η μεθοδολογία επίλυσης του προβλήματος της υπολογιστικής εκφόρτωσης.

3.1 Διατύπωση του προβλήματος

Σε αυτή την ενότητα, θα παρουσιάσουμε την προσέγγιση στην προτεινόμενη υλοποίηση ενός προσαρμοσμένου περιβάλλοντος MEC που θα χρησιμοποιηθεί για την αξιολόγηση και βελτιστοποίηση των αλγορίθμων αποφόρτισης υπολογισμών. Το περιβάλλον αυτό θα βασίζεται στο πλαίσιο OpenAI Gym[2], το οποίο παρέχει ένα ευέλικτο και επεκτάσιμο περιβάλλον για την ανάπτυξη και δοκιμή αλγορίθμων ενισχυτικής μάθησης.

3.1.1 Στόχοι της Υλοποίησης

Οι κύριοι στόχοι της υλοποίησης του προσαρμοσμένου περιβάλλοντος MEC είναι:

1. **Ρεαλιστική προσομοίωση:** Να δημιουργηθεί ένα περιβάλλον που προσομοιώνει ένα δυναμικό σύστημα MEC, συμπεριλαμβανομένων των συσκευών χρηστών, του διακομιστή ακμής και του δικτύου επικοινωνίας.

2. **Ευελιξία:** Να παρέχει τη δυνατότητα εύκολης προσαρμογής των παραμέτρων του συστήματος, όπως ο αριθμός των συσκευών, τα χαρακτηριστικά των διεργασιών και οι συνθήκες του δικτύου.

3. **Υποστήριξη διαφόρων αλγορίθμων:** Να επιτρέπει την εύκολη ενσωμάτωση αλγορίθμων Ενισχυτικής Μάθησης με σκοπό την αξιολόγηση και σύγκριση διαφορετικών τεχνικών αποφόρτισης υπολογισμών.

4. **Αξιολόγηση ποσοτικών δεδομένων:** Να παρέχει ένα σύνολο ποσοτικών αποτελεσμάτων για την αξιολόγηση της απόδοσης των αλγορίθμων αποφόρτισης, όπως η κατανάλωση ενέργειας και ο χρόνος εκτέλεσης.

[2] https://github.com/openai/gym

3.1.2 Βασικά Στοιχεία του Περιβάλλοντος

Το προσαρμοσμένο περιβάλλον MEC θα περιλαμβάνει τα ακόλουθα βασικά στοιχεία:

1. **Μοντέλο συσκευών χρηστών:** Προσομοίωση των χαρακτηριστικών των συσκευών χρηστών, συμπεριλαμβανομένης της υπολογιστικής ικανότητας, καθώς και της αντιστοίχισης με κάποια από τις παραγόμενες διεργασίες.

2. **Μοντέλο διακομιστή ακμής:** Αναπαράσταση του διακομιστή MEC με τους πόρους επεξεργασίας που κατανέμει για κάθε συσκευή.

3. **Μοντέλο δικτύου:** Προσομοίωση των χαρακτηριστικών του δικτύου επικοινωνίας, συμπεριλαμβανομένου του εύρους ζώνης.

4. **Γεννήτρια εργασιών:** Δημιουργία ρεαλιστικών υπολογιστικών εργασιών με διαφορετικές απαιτήσεις και χαρακτηριστικά.

5. **Μηχανισμός αποφόρτισης:** Υλοποίηση της διαδικασίας λήψης αποφάσεων για την αποφόρτιση υπολογισμών, συμπεριλαμβανομένης της επιλογής μεταξύ τοπικής εκτέλεσης και αποφόρτισης στον διακομιστή ακμής.

6. **Συλλέκτης ποσοτικών δεδομένων:** Καταγραφή και ανάλυση των δεδομένων για την αξιολόγηση των αλγορίθμων αποφόρτισης.

Στο σύνολο των δοκιμών για την κατανόηση της συμπεριφοράς του συστήματος, πραγματοποιήθηκαν παραμετροποιήσεις και συνεπώς σχηματίστηκαν διαφορετικά περιβάλλοντα με διαφορετικά χαρακτηριστικά και πολυπλοκότητα.

3.1.3 Σχεδιασμός μοντέλων συσκευών, διακομιστή και δικτύου

Ο σχεδιασμός των μοντέλων συσκευών, διακομιστή και δικτύου αποτελεί θεμελιώδες βήμα στην ανάπτυξη ενός αποτελεσματικού συστήματος MEC. Τα μοντέλα αυτά αναπαριστούν τις βασικές οντότητες του συστήματος και τις μεταξύ τους αλληλεπιδράσεις.

Το μοντέλο συσκευών περιγράφει τα χαρακτηριστικά των συσκευών χρηστών, δηλαδή την υπολογιστική ισχύ των συσκευών, η οποία δίνεται σε GHz με εύρος από 1 έως 5. Το μοντέλο διακομιστή αναπαριστά τους πόρους που δεσμεύει ο διακομιστής ακμής για κάθε μία συσκευή και λαμβάνει τιμές από ένα εύρος τιμών μεταξύ 4 έως 9 GHz. Τέλος, το μοντέλο δικτύου περιγράφει τα χαρακτηριστικά των συνδέσεων μεταξύ των συσκευών και του διακομιστή, παρέχοντας το εύρος ζώνης για κάθε σύνδεση.

Η ακριβής μοντελοποίηση αυτών των οντοτήτων είναι κρίσιμη για την αποτελεσματική λήψη αποφάσεων σχετικά με την εκφόρτωση υπολογισμών, καθώς επιτρέπει στον αλγόριθμο ΕΜ να λαμβάνει υπόψη όλους τους σχετικούς παράγοντες κατά τη διαδικασία βελτιστοποίησης.

3.1.4 Σχεδιασμός γεννήτριας εργασιών

Η γεννήτρια εργασιών είναι υπεύθυνη για την παραγωγή και προσομοίωση των υπολογιστικών εργασιών που πρέπει να εκτελεστούν στο σύστημα. Ο σχεδιασμός της είναι καίριας σημασίας για την αξιολόγηση της απόδοσης του αλγορίθμου ΕΜ σε διάφορα σενάρια φόρτου εργασίας.

Η γεννήτρια εργασιών παράγει ένα ευρύ φάσμα εργασιών με διαφορετικά χαρακτηριστικά, όπως το μέγεθος, τις απαιτήσεις σε υπολογιστικούς πόρους και τη προθεσμία ολοκλήρωσης. Ένας καλά σχεδιασμένος μηχανισμός παραγωγής εργασιών επιτρέπει τη δοκιμή του συστήματος υπό διάφορες συνθήκες, εξασφαλίζοντας έτσι την ευελιξία και την προσαρμοστικότητα του αλγορίθμου ΕΜ σε πραγματικές συνθήκες λειτουργίας.

3.1.5 Σχεδιασμός μηχανισμού ελέγχου και εκφόρτωσης

Ο μηχανισμός ελέγχου και εκφόρτωσης αποτελεί τον πυρήνα του συστήματος λήψης αποφάσεων για την υπολογιστική εκφόρτωση. Είναι υπεύθυνος για την αξιολόγηση της τρέχουσας κατάστασης του συστήματος και τη λήψη αποφάσεων σχετικά με την επιλογή εκτέλεσης των εργασιών τοπικά ή στον διακομιστή.

Στο πλαίσιο της ΕΜ, ο μηχανισμός αυτός υλοποιείται από τον πράκτορα που αλληλοεπιδρά με το περιβάλλον. Ο σχεδιασμός του περιλαμβάνει τον καθορισμό του χώρου καταστάσεων, που αντιπροσωπεύει την τρέχουσα κατάσταση του συστήματος, και του χώρου ενεργειών, που περιλαμβάνει όλες τις πιθανές αποφάσεις εκφόρτωσης.

Η πολιτική του πράκτορα, η οποία καθορίζει τη στρατηγική λήψης αποφάσεων, βελτιστοποιείται σταδιακά μέσω της αλληλεπίδρασης με το περιβάλλον και της μάθησης από την ανατροφοδότηση που λαμβάνει.

3.1.6 Υπολογισμοί κατανάλωσης χρόνου και ενέργειας

Οι υπολογισμοί κατανάλωσης χρόνου και ενέργειας αποτελούν κρίσιμο στοιχείο του μοντέλου μας. Αυτοί οι υπολογισμοί εντάσσονται τόσο στα μοντέλα συσκευών και διακομιστών όσο και στη συνάρτηση κέρδους.

Στο πλαίσιο των μοντέλων συσκευών και διακομιστών, οι υπολογισμοί αυτοί χρησιμοποιούνται για να προσδιορίσουν [90]:

- Τον χρόνο εκτέλεσης μιας διεργασίας,
    - Τοπικά:

$$time\_cost = time^{local}_{computation} = \frac{task.required\_cycles}{device.cpu\_freq}$$

    - Απομακρυσμένα:

$$time\_cost = time^{offload}_{transmission} + time^{offload}_{computation}$$

$$time_{transmission}^{offload} = \frac{8 * (task.data\_size)}{device.bandwidth}$$

$$time_{computation}^{offload} = \frac{task.required\_cycles}{mec.allocated\_freq}$$

- Την ενεργειακή κατανάλωση για την εκτέλεση μιας εργασίας
  - Στην συσκευή χρήστη:

$$energy\_cost = energy_{computation}^{local}$$

$$energy_{computation}^{local} = \frac{(task.required\_cycles) * (device.cpu\_freq)^2}{10^{27}}$$

  - Στον διακομιστή:

$$energy\_cost = energy_{transmission}^{offload} + energy_{computation}^{offload}$$

$$energy_{transmission}^{offload} = upload\_time * k\,, \quad \text{k=0.002 kW [91]}$$

$$energy_{computation}^{offload} = \frac{(task.required\_cycles) * (mec.allocated\_freq)^2}{10^{27}}$$

Στον Πίνακα 1 γίνεται επεξήγηση και δίνεται η μονάδα μέτρησης όλων των μεταβλητών που χρησιμοποιούνται στις παραπάνω εξισώσεις.

Πίνακας 1 - Επεξήγηση μεταβλητών

Παράμετρος	**Περιγραφή**	**Εύρος τιμών**
device.bandwidth	Ο ρυθμός μετάδοσης δεδομένων της συσκευής. (Mbps)	1-5
device.cpu_freq	Η ταχύτητα εκτέλεσης διεργασιών από τη συσκευή. (GHz)	1-5
mec.allocated_freq	Η ταχύτητα εκτέλεσης διεργασιών που δεσμεύει ο διακομιστής για μια διεργασία. (GHz)	4-9
task.data_size	Το μέγεθος της διεργασίας. (MB)	1-50
task.required_cycles	Οι απαιτούμενοι κύκλοι υπολογισμού για την ολοκλήρωση εκτέλεσης της διεργασίας. (Gigacycles)	100-5000

Στο πλαίσιο της συνάρτησης κέρδους, αυτοί οι υπολογισμοί χρησιμοποιούνται για να αξιολογήσουν την αποτελεσματικότητα των αποφάσεων εκφόρτωσης και να καθοδηγήσουν τη διαδικασία μάθησης του πράκτορα.

3.1.7 Συνάρτηση κέρδους

Η συνάρτηση κέρδους (ή ανταμοιβής) αποτελεί έναν βασικό κομμάτι στη προσέγγιση επίλυσης μέσω τεχνικών ΕΜ. Η σημασία της έγκειται στο ότι καθοδηγεί τη διαδικασία μάθησης του πράκτορα, παρέχοντας ανατροφοδότηση για την ποιότητα των αποφάσεων που λαμβάνει.

Ο σχεδιασμός μιας κατάλληλης συνάρτησης κέρδους είναι κρίσιμος για την επιτυχία του αλγορίθμου ΕΜ. Πρέπει να ενσωματώνει όλους τους σχετικούς στόχους βελτιστοποίησης, όπως η ελαχιστοποίηση της χρονικής καθυστέρησης και η εξοικονόμηση ενέργειας. Η συνάρτηση κέρδους καθοδηγεί τον πράκτορα προς τη βέλτιστη συμπεριφορά.

$$total\_cost = (w_t * time\_cost) + (w_e * energy\_cost)$$

$$reward = -total\_cost$$

Η συνάρτηση συνολικού κόστους αποτελεί το κεντρικό μέτρο αξιολόγησης του συστήματος, καθώς συγκεντρώνει τα διαφορετικά κόστη που προκύπτουν από τη διαδικασία ολοκλήρωσης μιας διεργασίας. Οι συναρτήσεις time_cost και energy_cost διαφοροποιούνται με βάση την επιλογή εκτέλεσης της διεργασίας (τοπική ή απομακρυσμένη), εφαρμόζοντας τον αντίστοιχο τύπο για κάθε περίπτωση. Όπως φαίνεται και στις παραπάνω σχέσεις, το συνολικό κόστος διαμορφώνεται από την προσθήκη των επιμέρους κοστών, όπου το κάθε ένα είναι πολλαπλασιασμένο με ένα βάρος. Λόγω των πολλαπλών συνδυασμών των παραμέτρων των διεργασιών, χρειάζεται σε κάθε περίπτωση ο πράκτορας να αντιλαμβάνεται τις τιμές κατανάλωσης ενέργειας και χρόνου σχετικά με την κάθε περίπτωση, προκειμένου να αξιολογεί σωστά το πόσο επιθυμητή είναι μια κατάσταση. Για τον λόγο, αυτό στις τιμές time_cost και energy_cost πραγματοποιείται διαίρεση με την μέγιστη τιμή για κάθε περίπτωση. Έτσι, οι τιμές time_cost και energy_cost λαμβάνουν μια μορφή μεταξύ 0 και 1, ώστε να είναι συγκρίσιμες. Έπειτα, με την εφαρμογή των συντελεστών βαρύτητας μας δίνεται η δυνατότητα να καθορίζουμε την σημασία του χρόνου και της ενέργειας σε κάθε περίπτωση. Με αυτή τη συνάρτηση, έχουμε τη δυνατότητα να προσαρμόζουμε το μοντέλο μας ώστε να επιτυγχάνεται η επιθυμητή ισορροπία μεταξύ των διαφορετικών ειδών κόστους. Η συνάρτηση αμοιβής ορίζεται ως το αντίθετο του συνολικού κόστους και χρησιμοποιείται από τον πράκτορα για να αντιληφθεί πόσο καλή είναι μια επιλογή πράξης σε μια κατάσταση. Αυτή η τιμή αποτελεί και τον στόχο του πράκτορα, την οποία και προσπαθεί να αυξήσει, μέσω των κατάλληλων επιλογών που κάνει.

Η αποτελεσματικότητα της εκπαίδευσης και η τελική απόδοση του πράκτορα εξαρτώνται σε μεγάλο βαθμό από τον ορθό σχεδιασμό αυτής της συνάρτησης. Μια καλά σχεδιασμένη συνάρτηση κέρδους επιτρέπει στον πράκτορα να μάθει μια πολιτική που ισορροπεί αποτελεσματικά μεταξύ των διαφόρων στόχων του συστήματος, οδηγώντας σε βέλτιστες αποφάσεις εκφόρτωσης.

3.2 Εργαλεία και Βιβλιοθήκες

Σε αυτή την ενότητα παρουσιάζονται τα βασικά εργαλεία και οι βιβλιοθήκες που χρησιμοποιήθηκαν για την υλοποίηση του συστήματος βελτιστοποίησης υπολογιστικής εκφόρτωσης σε περιβάλλον MEC.

3.2.1 Python

Η γλώσσα προγραμματισμού Python επιλέχθηκε ως η κύρια γλώσσα για την υλοποίηση αυτής της εργασίας. Η Python έχει καθιερωθεί ως μια από τις πιο δημοφιλείς γλώσσες στους τομείς της επιστήμης δεδομένων και της μηχανικής μάθησης, χάρη στην απλότητά της, την ευελιξία της και το εκτενές οικοσύστημα βιβλιοθηκών που διαθέτει. Η ευρεία υποστήριξη της κοινότητας και η πληθώρα διαθέσιμων πόρων καθιστούν την Python ιδανική επιλογή για την ανάπτυξη και πειραματισμό με αλγορίθμους ενισχυτικής μάθησης.

3.2.2 NumPy

Η βιβλιοθήκη NumPy[3] χρησιμοποιήθηκε στην υλοποίηση για αποδοτικούς υπολογισμούς με πίνακες και μαθηματικές πράξεις. Στην κλάση MEC Environment, η NumPy χρησιμοποιείται για τη δημιουργία τυχαίων τιμών, την κανονικοποίηση των παρατηρήσεων και την εκτέλεση διαφόρων υπολογισμών.

3.2.3 Stable Baselines 3

Η Stable Baselines 3[4] είναι μια βιβλιοθήκη που παρέχει αξιόπιστες υλοποιήσεις δημοφιλών αλγορίθμων ενισχυτικής μάθησης. Σε αυτή την εργασία, χρησιμοποιήθηκε για την υλοποίηση και σύγκριση των αλγορίθμων PPO, A2C και DQN. Η Stable Baselines 3 προσφέρει ένα απλό και ενοποιημένο Application Programming Interface (API) για την εκπαίδευση και αξιολόγηση μοντέλων ενισχυτικής μάθησης.

3.2.4 PyTorch

Το PyTorch[5] αποτελεί το βασικό πλαίσιο βαθιάς μάθησης πάνω στο οποίο είναι χτισμένο το Stable Baselines 3. Πρόκειται για μια ισχυρή βιβλιοθήκη που προσφέρει δυναμική κατασκευή γραφημάτων υπολογισμού και αυτόματο υπολογισμό κλίσεων, καθώς και λειτουργίες σχετικά με όλο τον κύκλο ζωής μιας υλοποίησης νευρωνικού δικτύου, από την προεπεξεργασία και χειρισμό δεδομένων μέχρι και την αξιολόγηση του μοντέλου, χαρακτηριστικά που είναι απαραίτητα για την αποτελεσματική εκπαίδευση νευρωνικών δικτύων σε προβλήματα BEM.

3.2.5 OpenAI Gym

Το OpenAI Gym[6] παρέχει ένα τυποποιημένο πλαίσιο για την ανάπτυξη και αξιολόγηση αλγορίθμων ενισχυτικής μάθησης. Στην παρούσα εργασία, χρησιμοποιήθηκε για τη δημιουργία του προσαρμοσμένου περιβάλλοντος MEC, επιτρέποντας την εύκολη ενσωμάτωση με τους αλγορίθμους του Stable Baselines 3.

3.2.6 TensorBoard

Το TensorBoard[7], ένα εργαλείο οπτικοποίησης που αρχικά αναπτύχθηκε για το TensorFlow, χρησιμοποιήθηκε για την παρακολούθηση και την οπτικοποίηση της

[3] https://numpy.org
[4] https://github.com/DLR-RM/stable-baselines3
[5] https://pytorch.org
[6] https://github.com/openai/gym
[7] https://www.tensorflow.org/tensorboard

διαδικασίας εκπαίδευσης. Επιτρέπει την, σε πραγματικό χρόνο παρακολούθηση δεδομένων όπως η μέση ανταμοιβή και άλλα ποσοτικά μεγέθη ανά αλγόριθμο, χρήσιμα για την κατανόηση της εξέλιξης της εκπαίδευσης.

3.2.7 Optuna

Το Optuna[8] είναι ένα πλαίσιο βελτιστοποίησης υπερπαραμέτρων που χρησιμοποιήθηκε για τη βελτίωση της απόδοσης των αλγορίθμων ενισχυτικής μάθησης. Προσφέρει προηγμένες τεχνικές αναζήτησης και δυνατότητες πρώιμου τερματισμού για αποδοτική εύρεση βέλτιστων υπερπαραμέτρων.

3.3 Υλοποίηση του περιβάλλοντος σε OpenAI Gym

Σε αυτή την ενότητα, γίνεται ανάλυση της υλοποίησης του προσαρμοσμένου περιβάλλοντος MEC χρησιμοποιώντας το πλαίσιο OpenAI Gym. Η κλάση MECEnvironment αποτελεί τον πυρήνα αυτής της υλοποίησης, επεκτείνοντας την κλάση gym.Env. Στο παράρτημα Α περιέχεται ο κώδικας όλου του περιβάλλοντος.

3.3.1 Μέθοδος init

Έγινε σχεδιασμός για την επιλογή χώρων δράσεων και παρατηρήσεων, ώστε να ανταποκρίνονται στην ενιαία χρήση και από τους 3 διαφορετικούς αλγορίθμους, καθώς παρουσιάζουν διαφορετικές απαιτήσεις.

Αυτή η μέθοδος δημιουργεί:

- Ένα σύνολο συσκευών (devices) και εργασιών (tasks).
- Έναν διακομιστή MEC (mec_server).
- Τον χώρο δράσεων (action_space) ως δυαδικό χώρο 2^num_devices καταστάσεων, αντιπροσωπεύοντας όλους τους πιθανούς συνδυασμούς αποφάσεων αποφόρτισης. Ο πράκτορας μπορεί να επιλέξει οποιοδήποτε ακέραιο αριθμό σε αυτό το σύνολο.
- Τον χώρο παρατηρήσεων (observation_space) που ορίζεται ως ένα spaces.Box. Αυτός ο χώρος αντιπροσωπεύει έναν συνεχές διανυσματικό χώρο, όπου περιλαμβάνει στοιχεία για κάθε ένα χαρακτηριστικό του περιβάλλοντος. Οι τιμές σε αυτόν τον χώρο μπορούν να λάβουν οποιονδήποτε πραγματικό αριθμό εντός των προκαθορισμένων κατώτερων και ανώτερων ορίων για κάθε διάσταση, για όλες τις συσκευές, εργασίες και τον διακομιστή MEC.

[8] https://optuna.org

3.3.2 Μέθοδος reset

Η μέθοδος reset επαναφέρει το περιβάλλον στην αρχική του κατάσταση ως εξής:

- Επαναφέρει τον μετρητή βημάτων.
- Επαναρχικοποιεί όλες τις συσκευές και εργασίες.
- Επιστρέφει την αρχική παρατήρηση του περιβάλλοντος.

3.3.3 Μέθοδος step

Η μέθοδος step εκτελεί ένα βήμα στο περιβάλλον με βάση την επιλεγμένη δράση, ως εξής:

- Μετατρέπει τη διακριτή δράση σε δυαδικό διάνυσμα αποφάσεων αποφόρτισης.
- Εκτελεί την απόφαση αποφόρτισης για κάθε συσκευή και υπολογίζει το συνολικό κόστος.
- Ενημερώνει την κατάσταση του περιβάλλοντος.
- Υπολογίζει την ανταμοιβή ως το αρνητικό του συνολικού κόστους.
- Ελέγχει αν το επεισόδιο έχει ολοκληρωθεί.

3.3.4 Άλλες μέθοδοι

Οι μέθοδοι _execute_locally και _offload_to_mec υπολογίζουν το κόστος εκτέλεσης τοπικά ή στον διακομιστή MEC αντίστοιχα, λαμβάνοντας υπόψη τον χρόνο εκτέλεσης και την κατανάλωση ενέργειας.

Η μέθοδος calculate_max_division υπολογίζει τις μέγιστες τιμές χρόνου και ενέργειας που απαιτούνται για να ολοκληρωθεί η κάθε διεργασία στις περιπτώσεις τοπικής και απομακρυσμένης εκτέλεσης.

Η μέθοδος _calculate_cost, υπολογίζει την συνάρτηση ανταμοιβής, ανάλογα με την απόφαση του πράκτορα και περιλαμβάνει του συντελεστές βαρύτητας της συνάρτησης συνολικού κόστους.

Η μέθοδος _get_observation δημιουργεί το διάνυσμα παρατήρησης, συνδυάζοντας πληροφορίες από όλες τις συσκευές, εργασίες και τον διακομιστή MEC.

Η συνάρτηση compare_rl_algorithms καλεί την εκκίνηση εκπαίδευσης για κάθε έναν αλγόριθμο. Πραγματοποιεί αρχικά το στάδιο της εκπαίδευσης και στην συνέχεια το στάδιο της αξιολόγησης, όπου και γίνεται η σύγκριση των τριών σεναρίων. Τα σενάρια αυτά είναι:

1. Η επιλογή απόφασης από τον αλγόριθμο, σχετικά με το ποιες διεργασίες θα εκτελεστούν τοπικά και ποιες στο MEC server ,
2. Το σενάριο όλες οι διεργασίες να εκτελεστούν τοπικά στις συσκευές και
3. Το σενάριο όλες οι διεργασίες να εκτελεστούν στο MEC server.

Επίσης η ίδια συνάρτηση ξεκινάει την διαδικασία αναζήτησης υπερπραμέτρων που βελτιστοποιούν την απολαβή ανταμοιβών. Αντίστοιχα για τις τιμές αυτές τρέχουν οι ίδιες διαδικασίες που περιγράφηκαν πιο πάνω.

3.4 Υλοποίηση αλγορίθμων μέσω του Stable Baselines 3

Το Stable Baselines 3 αποτελεί μια υψηλού επιπέδου βιβλιοθήκη για την εκπαίδευση και αξιολόγηση αλγορίθμων ενισχυτικής μάθησης. Σε αυτή την ενότητα, αναλύονται κάποια στοιχεία και η λειτουργία των αλγορίθμων που χρησιμοποιήθηκαν στην παρούσα εργασία.

3.4.1 Proximal Policy Optimization (PPO)

Ο αλγόριθμος PPO στηρίζεται στο ότι ανανεώνει την πολιτική συντηρητικά, αφαιρώντας το κίνητρο του να αλλάξει σε μεγάλο βαθμό η τρέχουσα πολιτική από την προηγούμενη. Ο αλγόριθμος PPO υλοποιείται στην κλάση PPO του Stable Baselines 3. Η λογική του αλγορίθμου εμφανίζεται στην Εικόνα 10, ενώ η βασική ροή λειτουργίας περιλαμβάνει:

1. **Συλλογή εμπειριών:** Η μέθοδος collect_rollouts συλλέγει τροχιές αλληλεπιδράσεων με το περιβάλλον.
2. **Υπολογισμός πλεονεκτήματος:** Η μέθοδος _compute_advantage υπολογίζει τις εκτιμήσεις πλεονεκτήματος χρησιμοποιώντας Generalized Advantage Estimation (GAE).

$$L(s,a,\theta_k,\theta) = \min\left(\frac{\pi_\theta(a|s)}{\pi_{\theta_k}(a|s)}A^{\pi_{\theta_k}}(s,a),\quad g(\epsilon, A^{\pi_{\theta_k}}(s,a))\right),$$

Όπου:

L: η συνάρτηση απώλειας που στοχεύει στη βελτιστοποίηση της πολιτικής.

A: η συνάρτηση πλεονεκτήματος, που δείχνει πόσο καλύτερη είναι μια ενέργεια από τον μέσο όρο.

θ: αντιπροσωπεύει τις παραμέτρους του νευρωνικού δικτύου, με $\pi_{\theta k}(a|s)$ να είναι η πολιτική με τις παλιές παραμέτρους θk, ενώ οι παράμετροι θ είναι οι τρέχουσες παράμετροι που ενημερώνονται κατά τη διάρκεια της εκπαίδευσης για τη βελτιστοποίηση της πολιτικής του πράκτορα.

$$g(\epsilon, A) = \begin{cases} (1+\epsilon)A & A \geq 0 \\ (1-\epsilon)A & A < 0. \end{cases}$$

Όπου ε, είναι η σταθερά clip parameter και g(ε,A) είναι η συνάρτηση περιορισμού.

3. **Ενημέρωση πολιτικής:** Η μέθοδος train εκτελεί πολλαπλά epochs εκπαίδευσης, ενημερώνοντας την πολιτική και τη συνάρτηση αξίας.
4. **PPO clip updates:** Η συνάρτηση απώλειας του PPO περιορίζει τις μεγάλες αλλαγές στην πολιτική, κάνοντας ειδικευμένες αποκοπές στην συνάρτηση βελτιστοποίησης, ώστε να μειώσει το κίνητρο η νέα πολιτική να είναι μακριά από την προηγούμενη.

Algorithm 1 PPO-Clip

Input: initial policy parameters θ_0, initial value function parameters ϕ_0
for $k = 0, 1, 2, \ldots$ **do**
  Collect set of trajectories $\mathcal{D}_k = \{\tau_i\}$ by running policy $\pi_k = \pi(\theta_k)$ in the environment.
  Compute rewards-to-go $\hat{R}_t$.
  Compute advantage estimates, $\hat{A}_t$ (using any method of advantage estimation) based on the current value function V_{ϕ_k}.
  Update the policy by maximizing the PPO-Clip objective:

$$\theta_{k+1} = \arg\max_{\theta} \frac{1}{|\mathcal{D}_k|T} \sum_{\tau \in \mathcal{D}_k} \sum_{t=0}^{T} \min\left(\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_k}(a_t|s_t)} A^{\pi_{\theta_k}}(s_t, a_t), \;\; g(\epsilon, A^{\pi_{\theta_k}}(s_t, a_t)) \right),$$

  typically via stochastic gradient ascent with Adam.
  Fit value function by regression on mean-squared error:

$$\phi_{k+1} = \arg\min_{\phi} \frac{1}{|\mathcal{D}_k|T} \sum_{\tau \in \mathcal{D}_k} \sum_{t=0}^{T} \left(V_\phi(s_t) - \hat{R}_t \right)^2,$$

  typically via some gradient descent algorithm.
end for

Εικόνα 10 - Αλγόριθμος PPO [67]

3.4.2 Advantage Actor-Critic (A2C)

Ο αλγόριθμος A2C συνδυάζει στοιχεία από μεθόδους πολιτικής και αξίας (actor-critic) στοχεύοντας στη βελτίωση της σταθερότητας και αποδοτικότητας της μάθησης, με την χρήση της συνάρτησης πλεονεκτήματος (advantage function) για τη μείωση της διακύμανσης στις εκτιμήσεις των κλίσεων πολιτικής, όπως φαίνεται και στην Εικόνα 11. Ο αλγόριθμος A2C υλοποιείται στην κλάση A2C του Stable Baselines 3. Η λειτουργία του περιλαμβάνει:

1. **Συλλογή εμπειριών:** Παρόμοια με τον PPO, αλλά συνήθως με μικρότερο αριθμό βημάτων.
2. **Υπολογισμός πλεονεκτήματος:** Χρησιμοποιεί απλούστερη εκτίμηση πλεονεκτήματος σε σύγκριση με τον PPO.

$$\Delta\theta = \alpha \nabla_\theta (log \pi_\theta(s, a)) \hat{q}_w(s, a)$$

3. Ενημέρωση δικτύων: Η μέθοδος train ενημερώνει ταυτόχρονα τα δίκτυα πολιτικής (actor) και αξίας (critic).

Algorithm 1 Advantage actor-critic - pseudocode

// Assume parameter vectors θ and θ_v
Initialize step counter $t \leftarrow 1$
Initialize episode counter $E \leftarrow 1$
repeat
  Reset gradients: $d\theta \leftarrow 0$ and $d\theta_v \leftarrow 0$.
  $t_{start} = t$
  Get state s_t
  repeat
    Perform a_t according to policy $\pi(a_t|s_t;\theta)$
    Receive reward r_t and new state s_{t+1}
    $t \leftarrow t+1$
  until terminal s_t **or** $t - t_{start} == t_{max}$
  $R = \begin{cases} 0 & \text{for terminal } s_t \\ V(s_t,\theta_v) & \text{for non-terminal } s_t \text{ //Bootstrap from last state} \end{cases}$
  for $i \in \{t-1,\ldots,t_{start}\}$ **do**
    $R \leftarrow r_i + \gamma R$
    Accumulate gradients wrt θ: $d\theta \leftarrow d\theta + \nabla_\theta \log \pi(a_i|s_i;\theta)(R - V(s_i;\theta_v)) + \beta_e \partial H(\pi(a_i|s_i;\theta))/\partial\theta$
    Accumulate gradients wrt θ_v: $d\theta_v \leftarrow d\theta_v + \beta_v(R - V(s_i;\theta_v))(\partial V(s_i;\theta_v)/\partial\theta_v)$
  end for
  Perform update of θ using $d\theta$ and of θ_v using $d\theta_v$.
  $E \leftarrow E+1$
until $E > E_{max}$

Εικόνα 11 - Αλγόριθμος A2C[9]

3.4.3 Deep Q-Network (DQN)

Ο αλγόριθμος DQN χρησιμοποιεί ένα νευρωνικό δίκτυο, το οποίο δέχεται μία κατάσταση και βγάζει ως έξοδο μια προσεγγιστική τιμή αξίας-Q για κάθε δράση στην συγκεκριμένη κατάσταση. Στην Εικόνα 12 εμφανίζεται η λογική του αλγορίθμου. Ο αλγόριθμος DQN υλοποιείται στην κλάση DQN του Stable Baselines 3 και τα βασικά στοιχεία της λειτουργίας του είναι:

1. **Συλλογή εμπειριών:** Χρησιμοποιεί ε-greedy στρατηγική για εξερεύνηση.
2. **Αποθήκευση εμπειριών:** Οι εμπειρίες αποθηκεύονται σε μια στοίβα επαναλήψεων.
3. **Ενημέρωση Q-δικτύου:** Η μέθοδος train εκτελεί mini-batch ενημερώσεις του Q-δικτύου.
4. **Ενημέρωση target δικτύου:** Περιοδικά ενημερώνει το target Q-δίκτυο για σταθερότητα.

[9] https://openai.com/index/openai-baselines-acktr-a2c

Algorithm 1 Deep Q-learning with Experience Replay

Initialize replay memory $\mathcal{D}$ to capacity N
Initialize action-value function Q with random weights
for episode $= 1, M$ **do**
  Initialise sequence $s_1 = \{x_1\}$ and preprocessed sequenced $\phi_1 = \phi(s_1)$
  for $t = 1, T$ **do**
    With probability ϵ select a random action a_t
    otherwise select $a_t = \max_a Q^*(\phi(s_t), a; \theta)$
    Execute action a_t in emulator and observe reward r_t and image x_{t+1}
    Set $s_{t+1} = s_t, a_t, x_{t+1}$ and preprocess $\phi_{t+1} = \phi(s_{t+1})$
    Store transition $(\phi_t, a_t, r_t, \phi_{t+1})$ in $\mathcal{D}$
    Sample random minibatch of transitions $(\phi_j, a_j, r_j, \phi_{j+1})$ from $\mathcal{D}$
    Set $y_j = \begin{cases} r_j & \text{for terminal } \phi_{j+1} \\ r_j + \gamma \max_{a'} Q(\phi_{j+1}, a'; \theta) & \text{for non-terminal } \phi_{j+1} \end{cases}$
    Perform a gradient descent step on $(y_j - Q(\phi_j, a_j; \theta))^2$ according to equation 3
  end for
end for

Εικόνα 12 - Αλγόριθμος DQN [62]

3.5 Τεχνικές Υποστήριξης Εκπαίδευσης

Σε αυτή την ενότητα, αναλύονται οι τεχνικές που χρησιμοποιήθηκαν για την παρακολούθηση και τον έλεγχο της διαδικασίας εκπαίδευσης.

3.5.1 Χρήση Wrappers στο OpenAI Gym

Τα wrappers του OpenAI Gym επιτρέπουν την τροποποίηση της συμπεριφοράς του περιβάλλοντος, χωρίς ωστόσο οι αλλαγές να επηρεάζουν το αρχικό περιβάλλον. Στην παρούσα εργασία, χρησιμοποιήθηκε ο Monitor wrapper κατά την κύρια εκτέλεση κώδικα και ο Vectorized Environment κατά την εκτέλεση των μεθόδων στην περίπτωση εύρεσης βέλτιστων υπερπαραμέτρων.

Ο Monitor wrapper καταγράφει στατιστικά για κάθε επεισόδιο, όπως τη συνολική ανταμοιβή και τη διάρκεια του επεισοδίου.

Κατά την χρήση διανυσματοποιημένων περιβαλλόντων, στοιβάζονται πολλαπλά ανεξάρτητα περιβάλλοντα σε ένα ενιαίο περιβάλλον. Έτσι ο πράκτορας εκπαιδεύεται σε n περιβάλλοντα ταυτόχρονα. Αντίστοιχα οι ενέργειες, οι παρατηρήσεις και οι ανταμοιβές αποτελούν πλέον ένα διάνυσμα διάστασης n.

3.5.2 Χρήση καλέσματος ανάδρασης (Callback) για το TensorBoard

Υλοποιήθηκε ένα προσαρμοσμένο callback, το TensorboardCallback, για την καταγραφή των ποσοτικών δεδομένων στο TensorBoard.

Αυτό το callback καταγράφει τη μέση ανταμοιβή των επεισοδίων κάθε 10 επεισόδια. Τα δεδομένα αποθηκεύονται σε αρχεία, τα οποία μπορούμε να ανατρέξουμε τόσο κατά την διάρκεια της εκπαίδευσης, αλλά και αργότερα.

3.6 Βελτιστοποίηση υπερπαραμέτρων

Σε αυτό το σημείο της διπλωματικής εργασίας γίνεται ανάλυση της διαδικασίας εύρεσης βέλτιστων τιμών για τις υπερπαραμέτρους των αλγορίθμων.

3.6.1 Ανασκόπηση διαδικασίας

Η βελτιστοποίηση υπερπαραμέτρων με το Optuna περιλαμβάνει τα εξής βήματα:

1. **Ορισμός του χώρου αναζήτησης:** Καθορίζονται τα εύρη τιμών για κάθε υπερπαράμετρο.
2. **Υλοποίηση της αντικειμενικής συνάρτησης:** Αυτή η συνάρτηση εκπαιδεύει το μοντέλο με συγκεκριμένες υπερπαραμέτρους και επιστρέφει μια τιμή απόδοσης.
3. **Δημιουργία μελέτης Optuna:** Ορίζεται η κατεύθυνση βελτιστοποίησης (μεγιστοποίηση ή ελαχιστοποίηση) και ο αριθμός των δοκιμών.
4. **Εκτέλεση βελτιστοποίησης:** Το Optuna εκτελεί επαναληπτικά την αντικειμενική συνάρτηση με διαφορετικούς συνδυασμούς υπερπαραμέτρων.
5. **Ανάλυση αποτελεσμάτων:** Εξετάζονται οι βέλτιστες υπερπαράμετροι και η επίδρασή τους στην απόδοση του μοντέλου.

Το Optuna χρησιμοποιεί προηγμένες τεχνικές όπως το Tree-structured Parzen Estimator (TPE) για αποδοτική δειγματοληψία υπερπαραμέτρων και μηχανισμούς πρώιμου τερματισμού για την επιτάχυνση της διαδικασίας βελτιστοποίησης.

3.6.2 Εύρεση υπερπαραμέτρων για αλγόριθμο PPO

Αρχικά πραγματοποιήθηκε αναζήτηση υπερπαραμέτρων για τον αλγόριθμο PPO σε αρκετά μεγάλο διάστημα αναζήτησης και για τις υπερπαραμέτρους που συνοψίζονται στον Πίνακα 2.

Πίνακας 2 - Διάστημα αναζήτησης υπερπαραμέτρων για τον αλγόριθμο PPO

Υπερπαράμετρος	Εύρος
Learning_rate	(1e-5) – (1e-3)
N_steps	16 – 2048
Batch_size	8 - 256
N_epochs	3 - 30
Gamma	0.9 – 0.9999
Gae_lambda	0.9 – 1.0
Clip_range	0.1 – 0.4
Ent_coef	(1e-8) – (1e-1)

3.6.3 Εύρεση υπερπαραμέτρων για αλγόριθμο A2C

Στην συνέχεια, πραγματοποιήθηκε αναζήτηση υπερπαραμέτρων για τον αλγόριθμο A2C για τις υπερπαραμέτρους που συνοψίζονται στον Πίνακα 3.:

Πίνακας 3 - Διάστημα αναζήτησης υπερπαραμέτρων για τον αλγόριθμο A2C

Υπερπαράμετρος	**Εύρος**
learning_rate	(1e-5) – (1e-3)
n_steps	1 – 100
gamma	0.9 – 0.9999
gae_lambda	0.9 – 1.0
vf_coef	0 – 1
ent_coef	(1e-8) – (1e-1)

3.6.4 Εύρεση υπερπαραμέτρων για αλγόριθμο DQN

Τέλος, πραγματοποιήθηκε αναζήτηση υπερπαραμέτρων και για τον αλγόριθμο DQN για τις υπερπαραμέτρους του Πίνακα 4:

Πίνακας 4 - Διάστημα αναζήτησης υπερπαραμέτρων για τον αλγόριθμο DQN

Υπερπαράμετρος	**Εύρος**
Learning_rate	(1e-5) – (1e-3)
Buffer_size	1e4 – 1e6
Batch_size	8 - 256
Learning_starts	1000 - 50000
Gamma	0.9 – 0.9999
tau	0.001 – 0.1
Train_freq	1 – 10
Gradient_steps	(-1) – 10

3.7 Πειραματική Αξιολόγηση

Ακολουθεί η πειραματική αξιολόγηση των αλγορίθμων ενισχυτικής μάθησης, η οποία πραγματοποιήθηκε χρησιμοποιώντας τη συνάρτηση `compare_rl_algorithms`:

3.7.1 Απόδοση κατά την διάρκεια της εκπαίδευσης

Χρησιμοποιήθηκαν διάφοροι αλγόριθμοι ΕΜ για την εκπαίδευση πρακτόρων στο νέο περιβάλλον. Είναι σημαντικό να σημειωθεί ότι κατά τη διάρκεια της εκπαίδευσης, δεν είναι πάντα δυνατό να βγάλουμε ασφαλή συμπεράσματα για την απόδοση των αλγορίθμων και αυτό οφείλεται σε διάφορους παράγοντες όπως:

- **On-policy και Off-policy αλγόριθμοι:** Οι off-policy αλγόριθμοι, όπως ο DQN (Deep Q-Network), μαθαίνουν από εμπειρίες που έχουν συλλεχθεί χρησιμοποιώντας μια διαφορετική πολιτική από αυτή που βελτιστοποιούν. Αυτό σημαίνει ότι κατά τη διάρκεια της εκπαίδευσης, οι ενέργειες που εκτελούνται μπορεί να μην αντικατοπτρίζουν την τρέχουσα βέλτιστη πολιτική.

- **Εξερεύνηση και εκμετάλλευση:** Κατά τη διάρκεια της εκπαίδευσης, οι αλγόριθμοι συχνά χρησιμοποιούν στρατηγικές εξερεύνησης (όπως ε-greedy) για να ανακαλύψουν νέες, πιθανώς καλύτερες ενέργειες. Αυτό μπορεί να οδηγήσει σε φαινομενικά χαμηλότερη απόδοση κατά τη διάρκεια της εκπαίδευσης.

- **Διαφορετικοί ρυθμοί σύγκλισης:** Διαφορετικοί αλγόριθμοι μπορεί να έχουν διαφορετικούς ρυθμούς σύγκλισης, καθιστώντας δύσκολη τη σύγκριση της απόδοσής τους σε ενδιάμεσα στάδια της εκπαίδευσης. Οπότε επιλέγεται εκπαίδευση με αρκετά βήματα ώστε να ολοκληρωθεί η εκπαίδευση για κάθε αλγόριθμο.

3.7.2 Αξιολόγηση μετά το πέρας της εκπαίδευσης

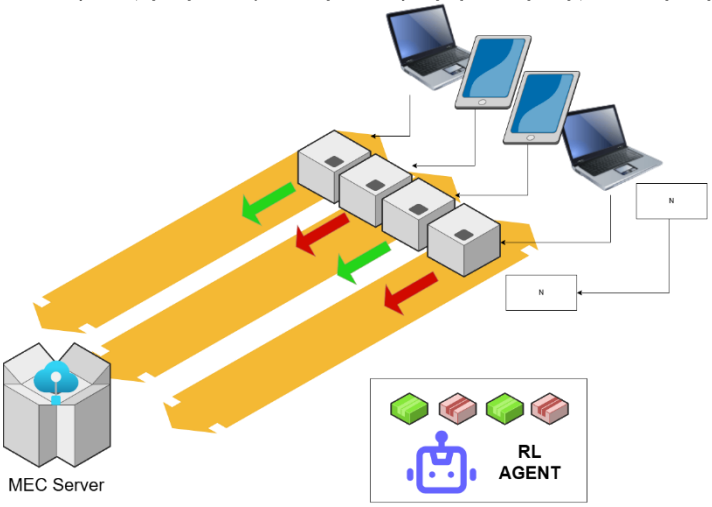
Για να ξεπεράσουμε αυτούς τους περιορισμούς και να αποκτήσουμε μια ακριβή εικόνα της απόδοσης των εκπαιδευμένων πρακτόρων, χρησιμοποιήσαμε τη μέθοδο evaluate του Stable Baselines 3. Αυτή η μέθοδος μας επιτρέπει να αξιολογήσουμε την απόδοση των εκπαιδευμένων πρακτόρων σε ένα σύνολο επεισοδίων δοκιμής, χωρίς εξερεύνηση και χρησιμοποιώντας την τρέχουσα βέλτιστη πολιτική. Με τον τρόπο αυτό μας δίνεται η δυνατότητα να συγκρίνουμε δίκαια την απόδοση των διαφόρων αλγορίθμων και να εξάγουμε αξιόπιστα συμπεράσματα σχετικά με την καταλληλότητά τους για το πρόβλημα της υπολογιστικής εκφόρτωσης.

Κεφάλαιο 4: Πειραματικά Αποτελέσματα και Ανάλυση

Στο παρόν κεφάλαιο, θα γίνει παρουσίαση των αποτελεσμάτων της προσομοίωσης του περιβάλλοντος MEC και ανάλυση της απόδοσης των διαφόρων αλγορίθμων ενισχυτικής μάθησης που εφαρμόστηκαν. Επιπλέον, θα αναλυθεί η διαδικασία εκπαίδευσης στην ΕΜ και η σημασία της ρύθμισης υπερπαραμέτρων.

4.1 Αποτελέσματα Προσομοίωσης

Η προσομοίωση διεξήχθη χρησιμοποιώντας το προσαρμοσμένο περιβάλλον MEC που υλοποιήθηκε με το OpenAI Gym, όπως περιγράφηκε στο προηγούμενο κεφάλαιο. Εφαρμόστηκαν τρεις διαφορετικοί αλγόριθμοι ενισχυτικής μάθησης: Proximal Policy Optimization (PPO), Advantage Actor-Critic (A2C), και Deep Q-Network (DQN). Σε κάθε χρονική στιγμή παράγονται ένας αριθμός εργασιών που κάθε μία αντιστοιχεί σε μία συσκευή. Ο αριθμός αυτός καθορίζεται από μία παράμετρο του περιβάλλοντος. Όπως περιγράφεται και στην Εικόνα 13, ο πράκτορας παίρνει κάθε χρονική στιγμή μια παρατήρηση του περιβάλλοντος με χαρακτηριστικά όλου του συστήματος και επιλέγει ποιες διεργασίες θα εκτελεστούν τοπικά στις συσκευές που αντιστοιχούν ή αν θα εκφορτωθούν στον διακομιστή MEC, επιτυγχάνοντας την μεγαλύτερη δυνατή συλλογή ανταμοιβής. Όπως θα δούμε παρακάτω, έγιναν δοκιμές και συγκρίνονται σενάρια με διαφορετικούς συνδυασμούς βαρών για την διαμόρφωση της συνάρτησης ανταμοιβής.



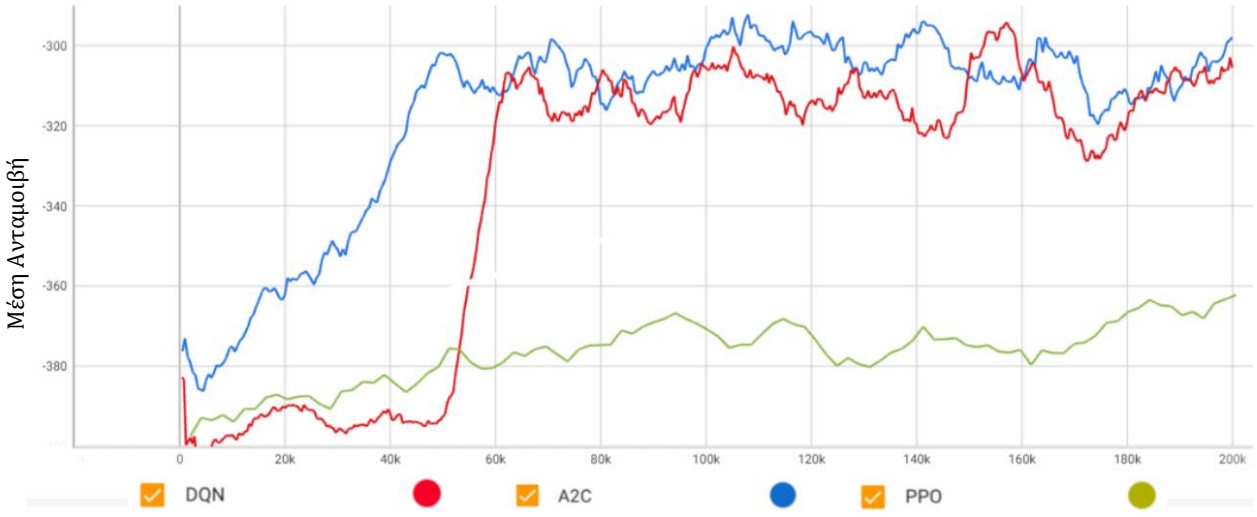
Εικόνα 13 - Εικονική αναπαράσταση συστήματος MEC

4.1.1 Περιορισμοί

Κατά την διάρκεια εκπόνησης της διπλωματικής εργασίας πραγματοποιήθηκαν δοκιμές για την δημιουργία ενός πλήρους λειτουργικού συστήματος υπολογιστικής εκφόρτωσης με δυνατότητες χειρισμού πολλαπλών διεργασιών, συσκευών και διακομιστών. Ακόμη σε ένα τέτοιο σύστημα συναντώνται πολύπλοκοι μηχανισμοί ελέγχου ροής και προγραμματισμού των διεργασιών, σενάρια κίνησης των συσκευών χρηστών, ακόμη και σενάρια προσομοίωσης αλλαγών μεταβλητών του περιβάλλοντος, συνεπώς και αλλαγής συμπεριφοράς που οφείλονται σε μη προβλεπόμενες καταστάσεις. Για αυτό το λόγο προτιμήθηκε να απλοποιηθεί το σενάριο υπολογιστικής εκφόρτωσης, προκειμένου να γίνει εστίαση του προβλήματος στην επίλυση αυτού με τεχνικές ΕΜ.

4.1.2 Πρώτες Παρατηρήσεις

Στην Εικόνα 14 παρουσιάζονται οι ανταμοιβές που συλλέγει ο κάθε αλγόριθμος κατά την διάρκεια της εκπαίδευσης. Αρχικά, με κάθε γραμμή εμφανίζεται η επίδοση του κάθε αλγορίθμου. Ο κάθετος άξονας χαρακτηρίζει την μέση ανταμοιβή που λαμβάνει ο πράκτορας ανά επεισόδιο, ενώ οι τιμές στον οριζόντιο άξονα αντιστοιχούν στον αριθμό βημάτων. Κατά την διάρκεια της εκπαίδευσης, οι αλγόριθμοι εμφανίζουν βελτίωση στην ανταμοιβή με την πάροδο του χρόνου, υποδεικνύοντας ότι μαθαίνουν και βελτιώνουν την απόδοσή τους καθώς προχωρά η εκπαίδευση. Ωστόσο, η απόδοση του PPO, δεν φαίνεται ικανοποιητική. Επίσης, παρατηρούμε ότι οι ανταμοιβές έχουν απότομες πτώσεις και ανόδους, γεγονός που μπορεί να υποδεικνύει, εκτός από ένα προκλητικό περιβάλλον, ότι οι υπερπαράμετροι δεν ευνοούν την εκπαίδευση του μοντέλου.



Εικόνα 14 - Εκπαίδευση αλγορίθμων

Για τον λόγο αυτό και για να υπάρχει μια πιο αντιπροσωπευτική αξιολόγηση των αλγορίθμων, κρίνεται χρήσιμο να πραγματοποιηθεί αναζήτηση βέλτιστων υπερπαραμέτρων. Στην επόμενη δοκιμή και αφού έτρεξε πείραμα με την χρήση του εργαλείου Optuna, βρέθηκαν οι βέλτιστες τιμές, λαμβάνοντας υπόψιν τους διαθέσιμους πόρους που είχαμε για να τρέξει η δοκιμή.

4.2 Εφαρμογή βέλτιστων υπερπαραμέτρων

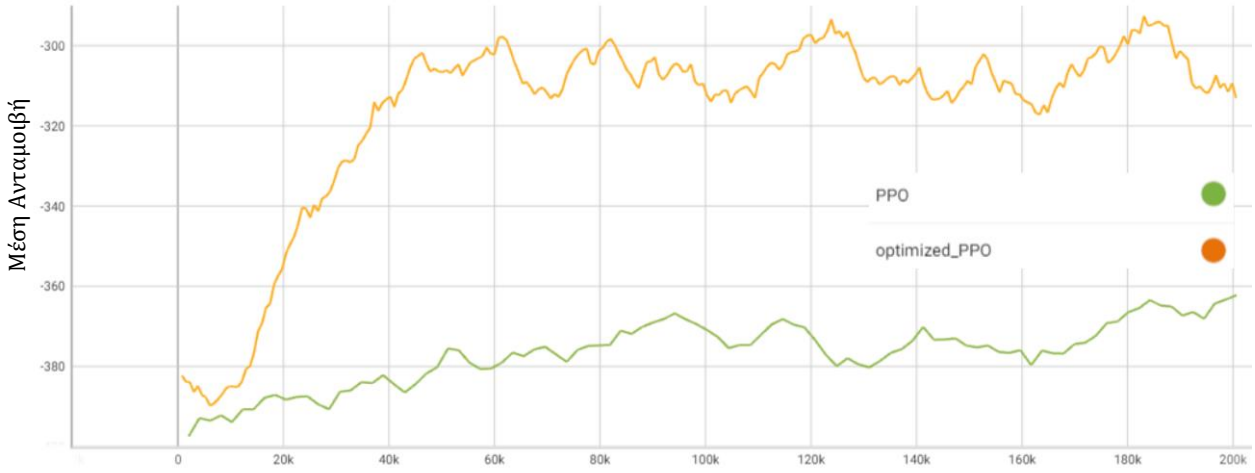
Σε αυτό το σημείο έγινε προσπάθεια να βελτιωθεί η επιστροφή ανταμοιβής στην υπάρχων υλοποίηση του περιβάλλοντος, οπότε έτρεξαν οι δοκιμές που περιγράφονται στο προηγούμενο Κεφάλαιο. Έτσι επετεύχθη στην συγκεκριμένη περίπτωση καλύτερη εκπαίδευση, συγκριτικά με τις προκαθορισμένες υπερπαραμέτρους του Stable Baselines 3.

4.2.1 Βελτιστοποίηση αλγορίθμου PPO

Με την ολοκλήρωση της εκτέλεσης του πειράματος εύρεσης βέλτιστων υπερπαραμέτρων, έχουμε τις τιμές με τις οποίες θα τρέξει το δεύτερο σενάριο με την χρήση του ίδιου αλγορίθμου για να πραγματοποιηθεί εκπαίδευση και να προκύψει το νέο μοντέλο. Οι τιμές αυτές εμφανίζονται στον Πίνακα 5 παρακάτω, ενώ ακολούθως εμφανίζεται στην Εικόνα 15 η σύγκριση των δύο μοντέλων, αυτού με hyperparameter tuning και αυτού χωρίς.

Πίνακας 5 - Αποτελέσματα εύρεσης υπερπαραμέτρων για τον αλγόριθμο PPO

Υπερπαράμετρος	**Επιλεγμένη τιμή**
Learning_rate	0.00021776512127397638
N_steps	760
Batch_size	256
N_epochs	8
Gamma	0.9144199897934076
Gae_lambda	0.9183075916025104
Clip_range	0.37498501251387906
Ent_coef	1.0331644131803022e-05



Εικόνα 15 - Σύγκριση αρχικού και βελτιστοποιημένου μοντέλου PPO

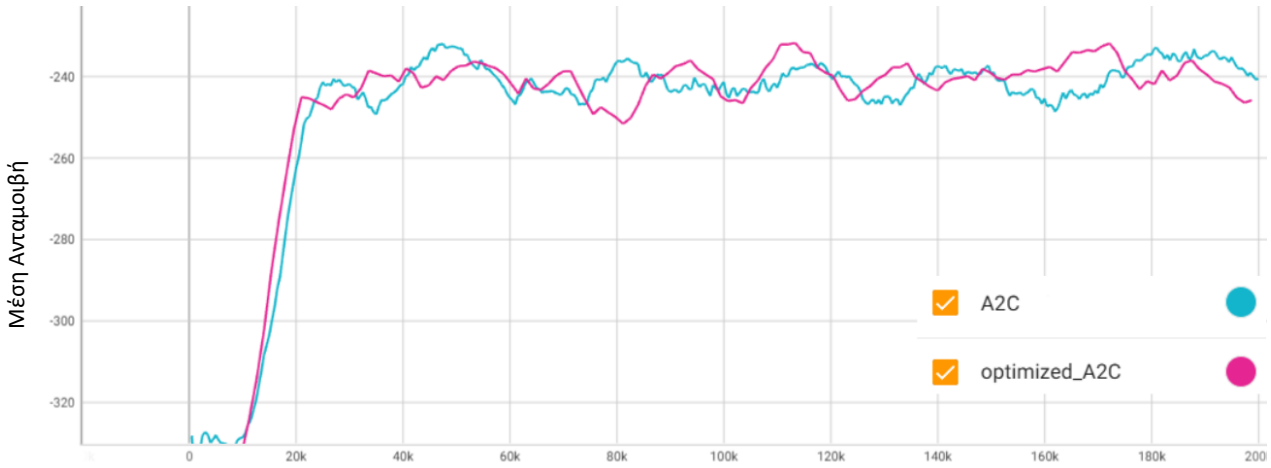
Η Εικόνα 15 παρουσιάζει, την μέση συλλογή ανταμοιβής ανά επεισόδιο από τον πράκτορα. Σε αυτή την περίπτωση, παρατηρούμε ότι το μοντέλο με βελτιστοποιημένες παραμέτρους (πορτοκαλί γραμμή) παρουσιάζει καλύτερη απόδοση σε σχέση με το μοντέλο χωρίς βελτιστοποίηση στις υπερπαραμέτρους.

4.2.2 Βελτιστοποίηση αλγορίθμου A2C

Μετά την σύγκριση της βελτιστοποίησης που πραγματοποιήθηκε στον αλγόριθμο PPO, θα παρουσιαστούν και οι τιμές με τις οποίες θα εκπαιδευτεί και ο αλγόριθμος A2C. Οι τιμές αυτές συνοψίζονται στον Πίνακα 6..

Πίνακας 6 - Αποτελέσματα εύρεσης υπερπαραμέτρων για τον αλγόριθμο A2C

Υπερπαράμετρος	Επιλεγμένη τιμή
learning_rate	0.0004977724684525267
n_steps	14
gamma	0.9167298667573246
gae_lambda	0.9456823593500105
vf_coef	0.18899376621510114
ent_coef	2.233502032574826e-08



Εικόνα 16 - Σύγκριση αρχικού και βελτιστοποιημένου μοντέλου A2C

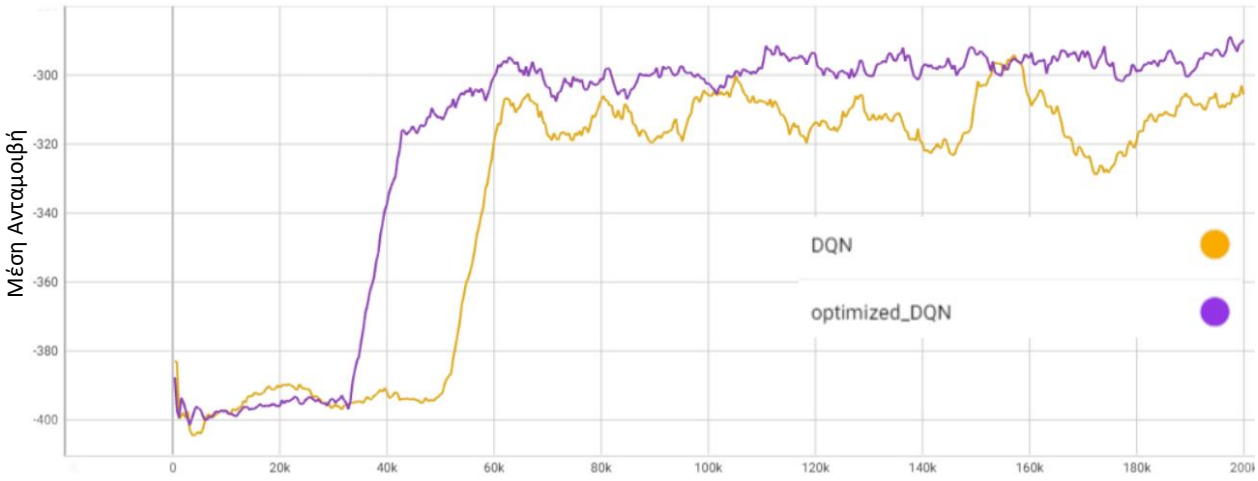
Η Εικόνα 16 παρουσιάζει, όπως και οι δύο προηγούμενες, την μέση συλλογή ανταμοιβής ανά επεισόδιο από τον πράκτορα. Σε αυτή την περίπτωση παρατηρούμε διαφορετική πορεία σε σχέση με τον προηγούμενο αλγόριθμο, τον PPO. Φαίνεται ότι και στις δύο περιπτώσεις οι πράκτορες εκπαιδεύονται γρήγορα και με τον ίδιο ρυθμό. Έτσι, αναμένεται ότι ο A2C θα έχει παρόμοια απόδοση συγκριτικά με την τροποποιημένη έκδοσή του.

4.2.3 Βελτιστοποίηση αλγορίθμου DQN

Τέλος, βελτιστοποιήθηκε ο αλγόριθμος DQN Οι τιμές με τις οποίες εκπαιδεύτηκε ο αλγόριθμος συνοψίζονται στον Πίνακα 7.

Πίνακας 7 - Αποτελέσματα εύρεσης υπερπαραμέτρων για τον αλγόριθμο DQN

Υπερπαράμετρος	Επιλεγμένη τιμή
Learning_rate	2.2350795980167074e-05
Buffer_size	610948
Batch_size	174
Learning_starts	31091
Gamma	0.9648660982239635
tau	0.048445057211731056
Train_freq	10
Gradient_steps	8



Εικόνα 17 - Σύγκριση αρχικού και βελτιστοποιημένου μοντέλου DQN

Η Εικόνα 17 παρουσιάζει την μέση συλλογή ανταμοιβής ανά επεισόδιο από τον πράκτορα. Σε αυτή την περίπτωση, του αλγορίθμου DQN, σημειώνεται βελτιωμένη επίδοση, καθώς σε κάθε βήμα παρατηρούμε να συλλέγει μεγαλύτερη ανταμοιβή σε σχέση με τον αλγόριθμο που δεν έχει υποστεί κάποια αλλαγή. Ωστόσο, τα τελικά αποτελέσματα εκπαίδευσης βρίσκονται αρκετά κοντά, άρα αναμένεται οι δύο αλγόριθμοι να εμφανίζουν παρόμοια συμπεριφορά.

4.3 Σύγκριση υπολογιστικής εκφόρτωσης

Στην πειραματική αξιολόγηση, συγκρίνεται η απόδοση του ευφυούς πράκτορα με δύο σενάρια αναφοράς: την αποκλειστικά τοπική εκτέλεση των εργασιών και την πλήρη ανάθεσή τους στον απομακρυσμένο διακομιστή MEC. Δεδομένου ότι τα χαρακτηριστικά των εργασιών παράγονται τυχαία εντός καθορισμένων ευρών, κάθε πείραμα διεξάγεται υπό διαφορετικές συνθήκες φόρτου εργασίας. Για την εξασφάλιση δίκαιης σύγκρισης, τα δύο σενάρια αναφοράς εκτελούνται παράλληλα με κάθε δοκιμή του αλγορίθμου, χρησιμοποιώντας τις ίδιες ακριβώς εργασίες.

Το περιβάλλον προσομοίωσης περιλαμβάνει 5 συσκευές χρηστών που παράγουν εργασίες σε κάθε χρονικό βήμα, ενώ κάθε επεισόδιο αποτελείται από 100 χρονικά βήματα. Συνεπώς, κάθε επεισόδιο αντιστοιχεί στη διαχείριση 500 συνολικά εργασιών (5 συσκευές × 100 βήματα). Τα αποτελέσματα που παρουσιάζονται αποτελούν τις μέσες τιμές από 100 διαφορετικά επεισόδια.

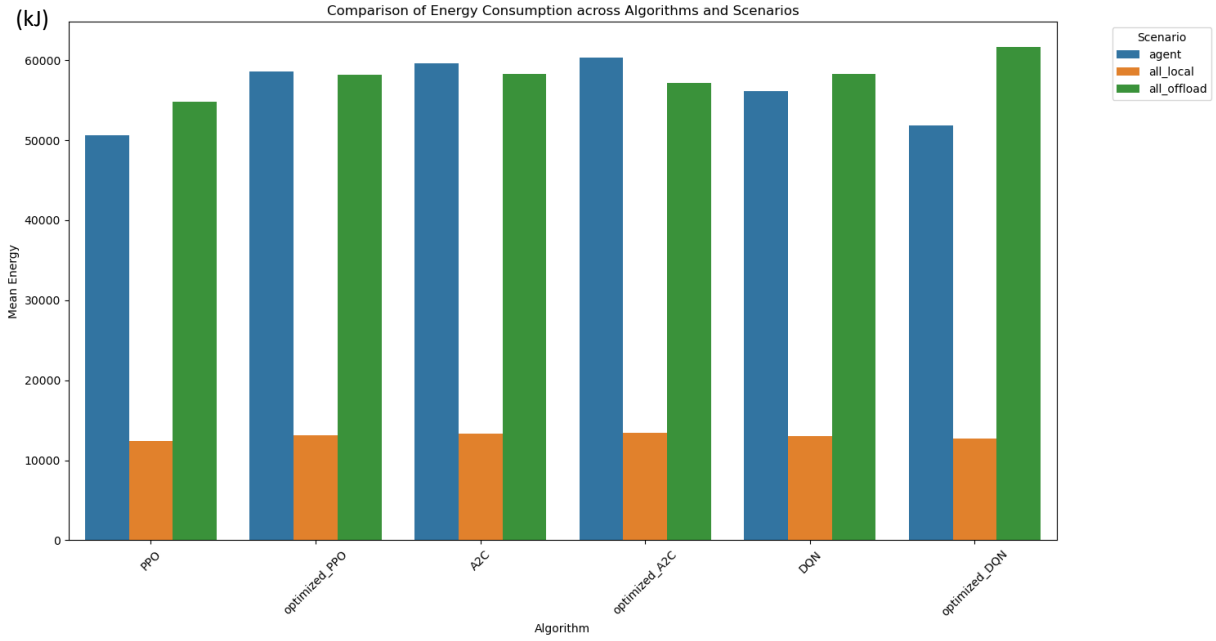
Διερευνήθηκε επίσης η επίδραση διαφορετικών συντελεστών βαρύτητας στη συνάρτηση ανταμοιβής, η οποία συνδυάζει το κόστος ενέργειας και το χρόνο εκτέλεσης. Οι παραλλαγές αυτές επέτρεψαν τη βελτιστοποίηση της συμπεριφοράς του πράκτορα ως προς διαφορετικούς στόχους απόδοσης.

4.3.1 Σύγκριση ενεργειακής κατανάλωσης

Στην υποενότητα 4.3.1 εξετάζονται οι δοκιμές που πραγματοποιήθηκαν, εστιάζοντας στη συμπεριφορά της κατανάλωσης ενέργειας. Στο σύνολο των διαγραμμάτων που περιλαμβάνονται στην υποενότητα, στον οριζόντιο άξονα παρουσιάζονται τα έξι

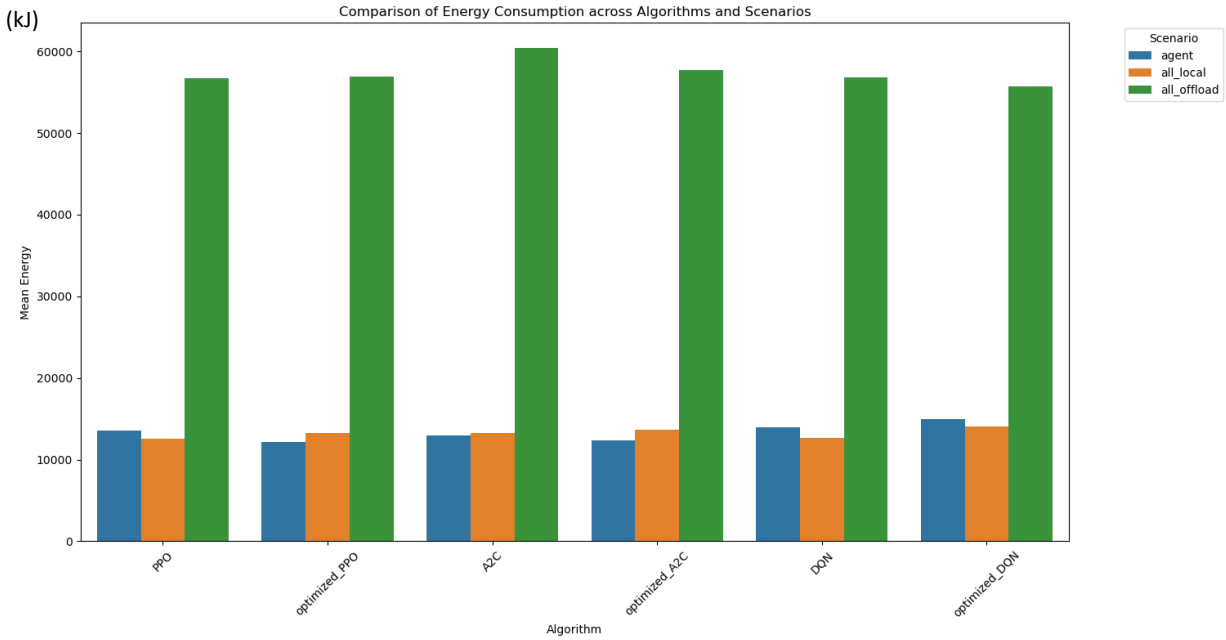
διαφορετικά μοντέλα, δύο από κάθε αλγόριθμο, εκ των οποίων, ένα το βελτιστοποιημένο και ένα χωρίς αλλαγές. Επιπλέον, στον κάθετο άξονα, παρουσιάζεται η μέση τιμή κατανάλωσης ενέργειας ανά επεισόδιο, για την ολοκλήρωση όλων των εργασιών. Η τιμή αυτή αφορά συνολική κατανάλωση όλων των συσκευών, σε ένα επεισόδιο εκατό βημάτων και μετριέται σε kJ.

Παρατηρούμε στα ακόλουθα διαγράμματα (Εικόνα 18 έως Εικόνα 21) ότι ανάλογα με τον συντελεστή βαρύτητας, οι αλγόριθμοι κάνουν διαφορετικές επιλογές, αφού η βελτιστοποίηση του προβλήματος διαμορφώνεται ανάλογα. Εφόσον η ενεργειακή κατανάλωση αποτελεί μέγεθος κόστους, οι χαμηλότερες τιμές υποδεικνύουν καλύτερη επίδοση του αλγορίθμου, καθώς αντιπροσωπεύουν μικρότερη κατανάλωση ενέργειας για την ολοκλήρωση των εργασιών.



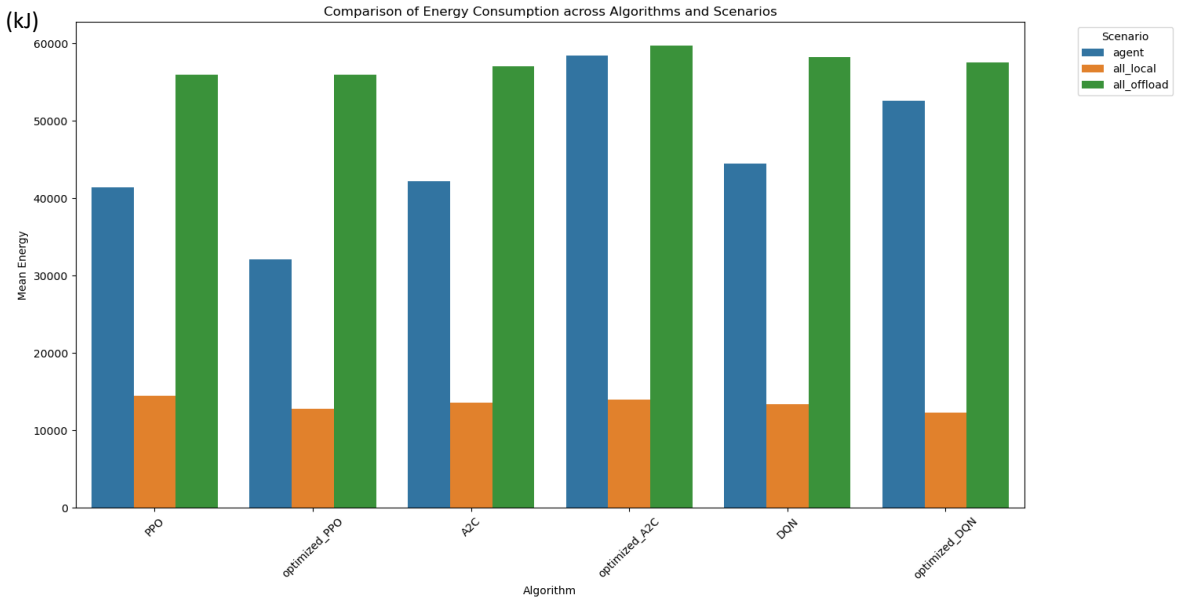
Εικόνα 18 - Σύγκριση ενεργειακής κατανάλωσης μεταξύ αλγορίθμων (w_time=1 & w_energy=0)

Στην Εικόνα 18 βλέπουμε την περίπτωση που έχουμε επιλέξει ο αλγόριθμος να βελτιστοποιήσει την χρονική καθυστέρηση (θέτοντας w_time=1 και w_energy=0). Για τον λόγο αυτό παρατηρούμε οι αποφάσεις του να μην δίνουν έμφαση στην μείωση ενεργειακής κατανάλωσης και να αποδίδουν κοντά στην κατανάλωση του διακομιστή MEC.



Εικόνα 19 - Σύγκριση ενεργειακής κατανάλωσης μεταξύ αλγορίθμων (w_time=0 & w_energy=1)

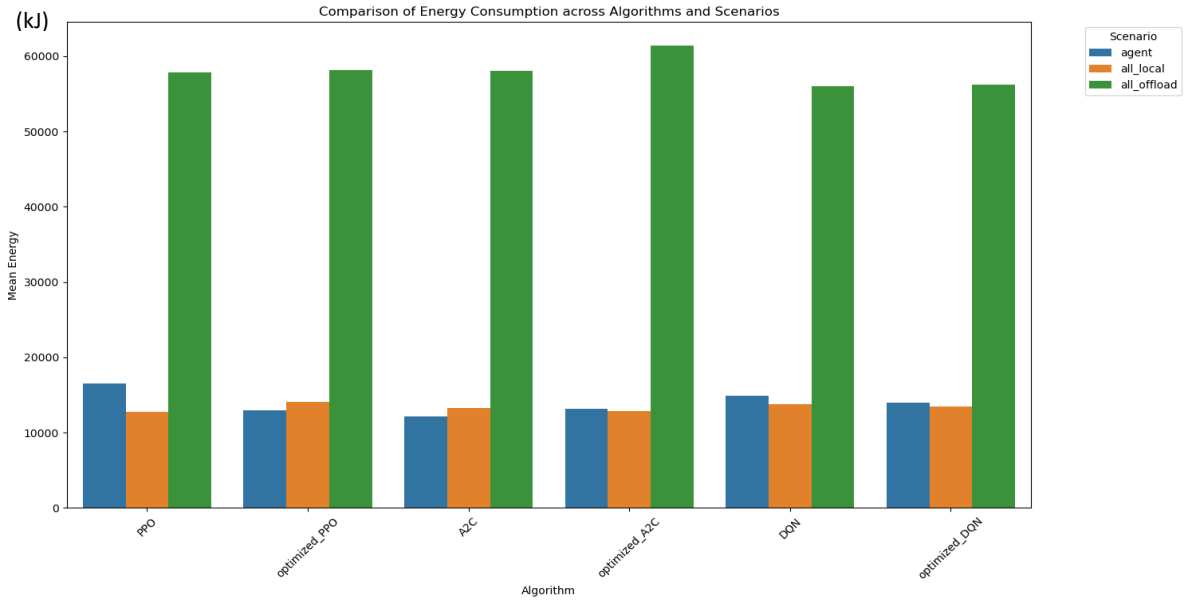
Σε αυτή τη περίπτωση θέτουμε τους συντελεστές βαρύτητας ως w_time=0 και w_energy=1. Παρατηρούμε πως οι πράκτορες, δίνουν προτεραιότητα στην μείωση της ενεργειακής κατανάλωσης. Μάλιστα στις περιπτώσεις των "PPO", "A2C" και "optimized_A2C" οι πράκτορες πετυχαίνουν χαμηλότερη ενεργειακή κατανάλωση συγκριτικά με την τοπική εκτέλεση.



Εικόνα 20 - Σύγκριση ενεργειακής κατανάλωσης μεταξύ αλγορίθμων (w_time=0.7 & w_energy=0.3)

Αφού εξετάστηκε η συμπεριφορά, όσον αφορά την ενεργειακή κατανάλωση, σε 2 ακραίες καταστάσεις, θα εξετάσουμε μία πιο ισορροπημένη προσέγγιση, με συνάρτηση ανταμοιβής να καθορίζεται τόσο από κατανάλωση ενέργειας, όσο και χρονική καθυστέρηση (Εικόνα 20). Θέτοντας w_time=0.7 και w_energy=0.3, παρατηρούμε γενικά υψηλές τιμές κατανάλωσης ενέργειας με τις τροποποιημένες μεθόδους να παρουσιάζουν αύξηση, εκτός του "optimized_PPO". Ωστόσο, δεν καθορίζεται η σωστή απόφαση του

πράκτορα μόνο από αυτό το διάγραμμα, αλλά σε συνδυασμό με το διάγραμμα χρονικής καθυστέρησης και ανταμοιβής.



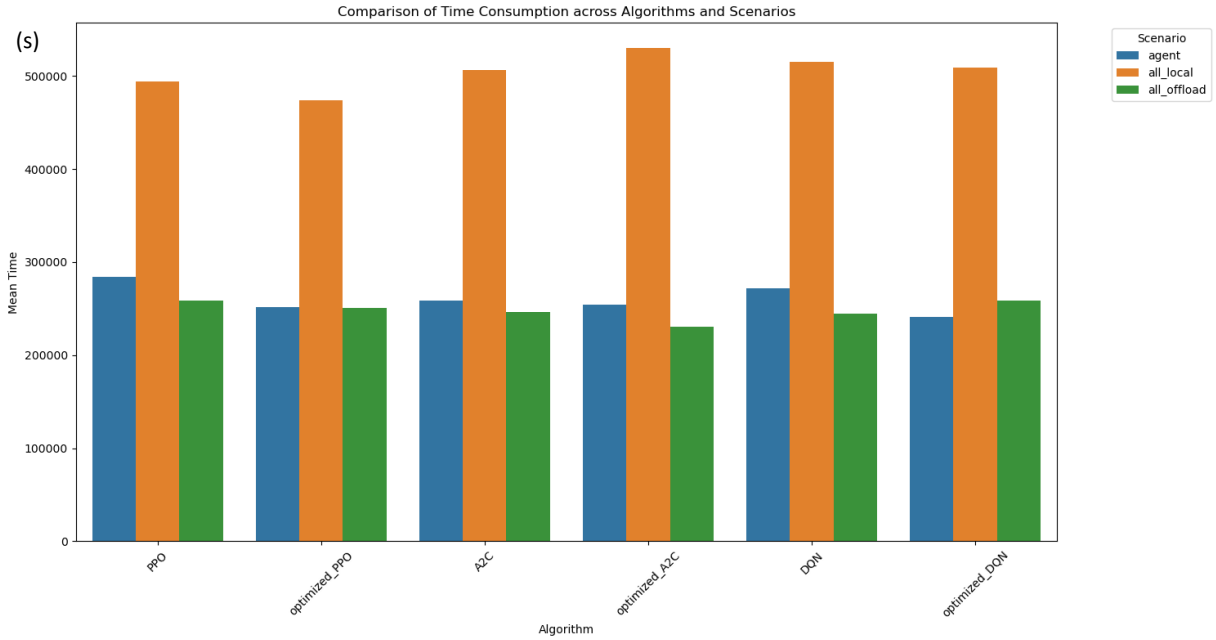
Εικόνα 21 - Σύγκριση ενεργειακής κατανάλωσης μεταξύ αλγορίθμων (w_time=0.3 & w_energy=0.7)

Στην Εικόνα 21 παρουσιάζεται η συνολική κατανάλωση ενέργειας όταν w_time=0.3 και w_energy=0.7. Όπως ήταν αναμενόμενο, η κατανάλωση ενέργειας έχει χαμηλές τιμές καθώς η συνάρτηση κόστους έχει μεγαλύτερο συντελεστή βαρύτητας για την ενέργεια. Υψηλότερες τιμές εμφανίζουν οι “PPO”, “DQN” ενώ η τροποποιημένη εκδοχή του “A2C” παρουσιάζει χειρότερη επίδοση από την μη τροποποιημένη. Επειδή, πρόκειται για ένα σενάριο που δεν έχει μοναδικό στόχο βελτιστοποίησης, θα χρειαστεί να μελετηθεί και η χρονική καθυστέρηση.

4.3.2 Σύγκριση χρονικής καθυστέρησης

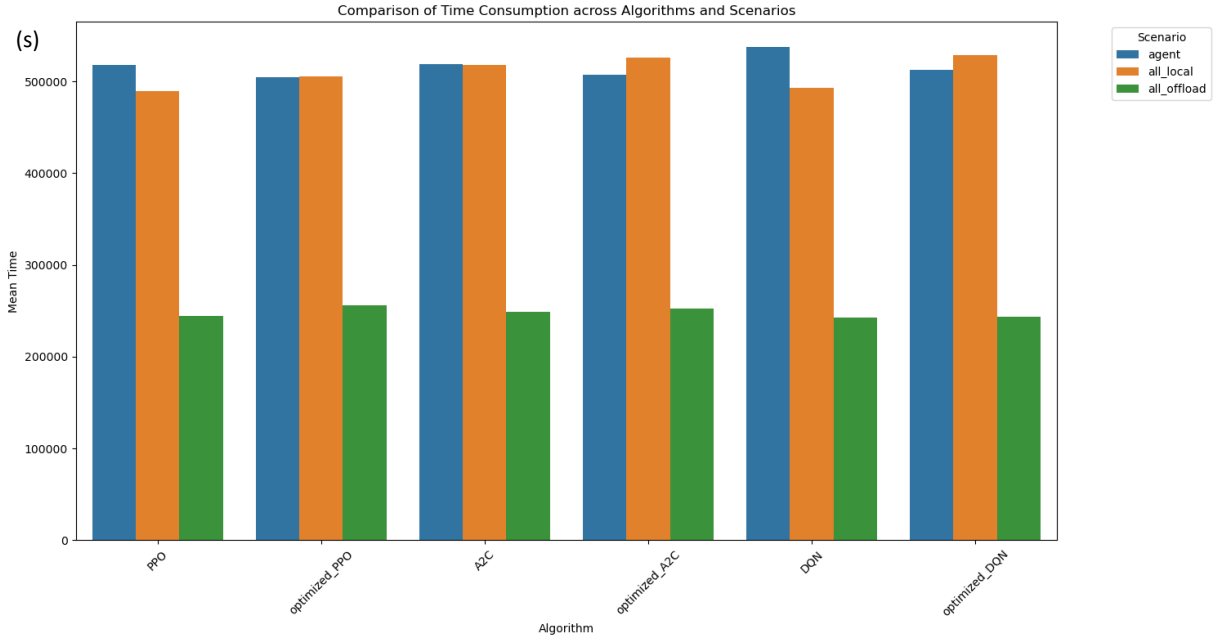
Στην υποενότητα αυτή θα εξεταστούν οι δοκιμές που πραγματοποιήθηκαν, εστιάζοντας στη συμπεριφορά της χρονικής καθυστέρησης. Στο σύνολο διαγραμμάτων της υποενότητας, ο οριζόντιος άξονας περιλαμβάνει τα έξι διαφορετικά μοντέλα, δύο από κάθε αλγόριθμο, εκ των οποίων, ένα το βελτιστοποιημένο και ένα χωρίς αλλαγές. Ενώ στον κάθετο άξονα, για κάθε σενάριο, βρίσκεται η μέση τιμή χρονικής καθυστέρησης ανά επεισόδιο, για την ολοκλήρωση όλων των εργασιών.

Αντιστοίχως, στα ακόλουθα διαγράμματα (Εικόνα 22 έως Εικόνα 25) παρατηρούμε την επιρροή του συντελεστή βαρύτητας στην διαμόρφωση του στόχου για τον πράκτορα. Ανάλογα με τις τιμές των συντελεστών, οι πράκτορες επιλέγουν και διαφορετική δράση και διαμορφώνουν το παρακάτω αποτέλεσμα σχετικά με την κατανάλωση χρόνου. Καθώς η χρονική καθυστέρηση αποτελεί μέγεθος κόστους, οι χαμηλότερες τιμές υποδεικνύουν καλύτερη επίδοση του αλγορίθμου, καθώς αντιπροσωπεύουν μικρότερη καθυστέρηση στην εκτέλεση των εργασιών.

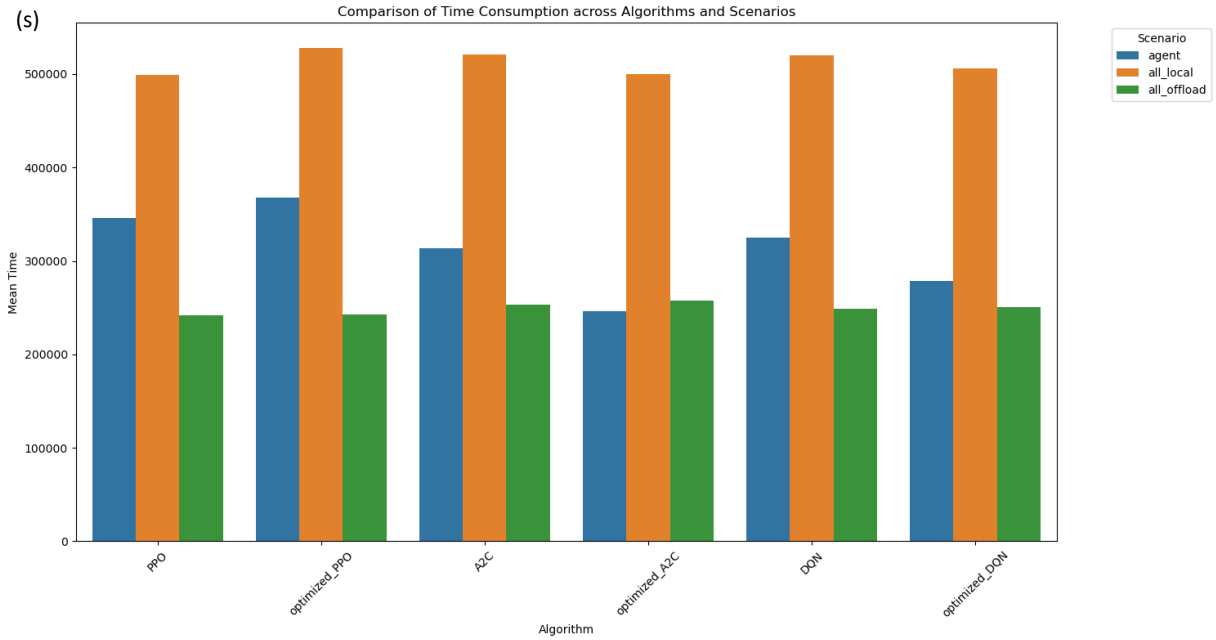


Εικόνα 22 - Σύγκριση χρονικής καθυστέρησης μεταξύ αλγορίθμων (w_time=1 & w_energy=0)

Στην Εικόνα 22 παρουσιάζεται η μέση χρονική καθυστέρηση κάθε περίπτωσης για w_time=1 και w_energy=0. Η επίδοση των πρακτόρων είναι στα επίπεδα που θα περιμέναμε, καθώς οι πράκτορες προσπαθούν να επιτύχουν την μικρότερη δυνατή χρονική καθυστέρηση. Ωστόσο, μόνο ο “optimized_DQN” πετυχαίνει να μειώσει την χρονική καθυστέρηση περισσότερο από την περίπτωση της εκτέλεσης στο διακομιστή. Αντίστοιχα και στην Εικόνα 23, οι πράκτορες έχουν τη συμπεριφορά που θα περιμέναμε, καθώς λόγω των συντελεστών βαρύτητας, δεν έχουν στόχο την μείωση της χρονικής καθυστέρησης.

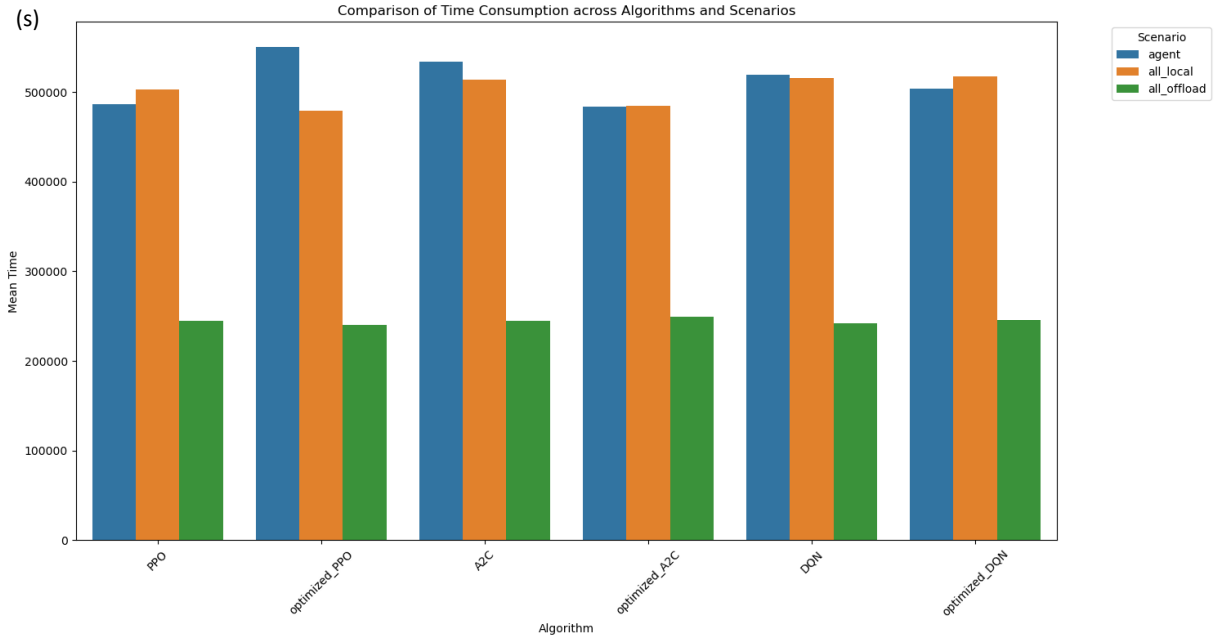


Εικόνα 23 - Σύγκριση χρονικής καθυστέρησης μεταξύ αλγορίθμων (w_time=0 & w_energy=1)



Εικόνα 24 - Σύγκριση χρονικής καθυστέρησης μεταξύ αλγορίθμων (w_time=0.7 & w_energy=0.3)

Η Εικόνα 24 παρουσιάζει τη μέση χρονική καθυστέρηση όταν w_time=0.7 και w_energy=0.3, ενώ η Εικόνα 25 παρουσιάζει τη μέση χρονική καθυστέρηση για w_time=0.7 και w_energy=0.3. Στην περίπτωση της Εικόνας 24, μειώθηκε η χρονική καθυστέρηση που πέτυχαν οι βελτιστοποιημένοι αλγόριθμοι συγκριτικά με τις μη βελτιστοποιημένες εκδοχές τους., με εξαίρεση τον “optimized_A2C”. Η συμπεριφορά αυτή είναι αναμενόμενη, καθώς η συνάρτηση κόστους έχει μεγαλύτερο συντελεστή βαρύτητας για τον χρόνο.

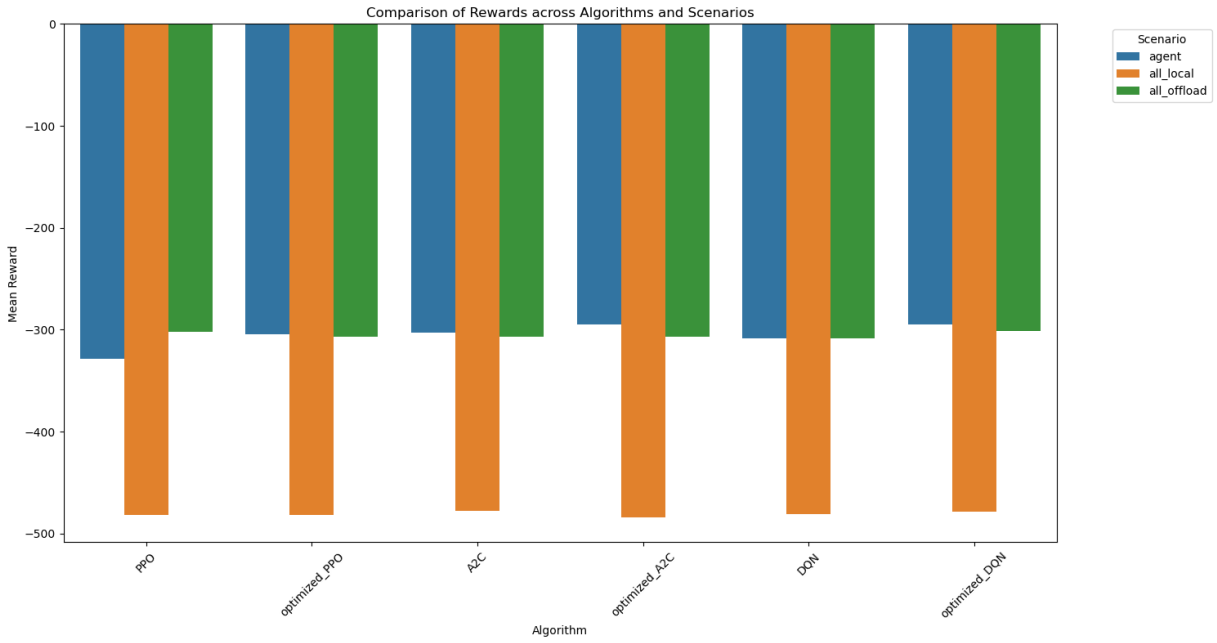


Εικόνα 25 - Σύγκριση χρονικής καθυστέρησης μεταξύ αλγορίθμων (w_time=0.3 & w_energy=0.7)

Στην Εικόνα 25 παρουσιάζεται η χρονική καθυστέρηση όταν w_time=0.3 και w_energy=0.7. Σε αυτή την περίπτωση η καθυστέρηση βρίσκεται σε υψηλά επίπεδα, καθώς δίνεται έμφαση στην βελτιστοποίηση της κατανάλωσης ενέργειας.

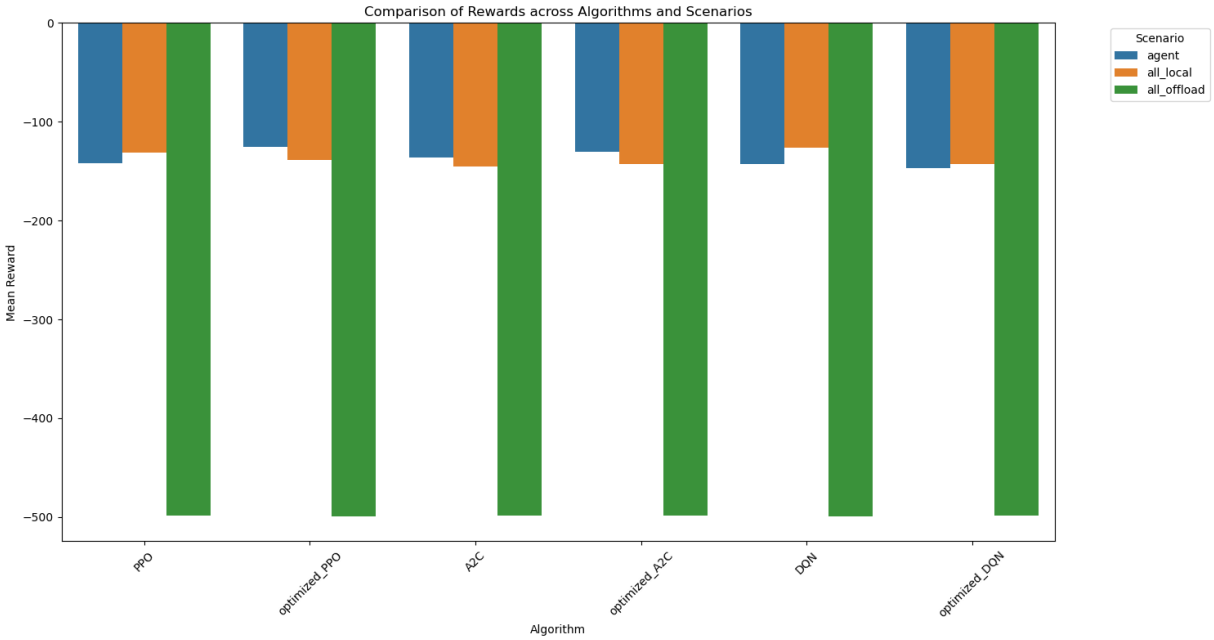
4.3.3 Σύγκριση συναρτήσεων ανταμοιβής

Τέλος, στην υποενότητα 4.3.3, παρουσιάζονται τα αποτελέσματα σχετικά με την επιστρεφόμενη ανταμοιβή. Η ανταμοιβή καθορίζεται από τα επιμέρους κόστη ενέργειας και χρόνου, και είναι αυτή που έχει ο αλγόριθμος σαν στόχο να βελτιστοποιήσει. Στα παρακάτω διαγράμματα (Εικόνα 26 έως Εικόνα 29), ο οριζόντιος άξονας περιλαμβάνει τα έξι διαφορετικά μοντέλα, δύο από κάθε αλγόριθμο, εκ των οποίων, ένα το βελτιστοποιημένο και ένα χωρίς αλλαγές. Επιπλέον, στον κάθετο άξονα παρουσιάζεται η επιστρεφόμενη ανταμοιβή ανά επεισόδιο, για την ολοκλήρωση όλων των εργασιών. Δεδομένου ότι η ανταμοιβή αντιπροσωπεύει το συνολικό κόστος, όσο πιο κοντά στο μηδέν βρίσκεται η τιμή της, τόσο καλύτερη είναι η επίδοση του αλγορίθμου.



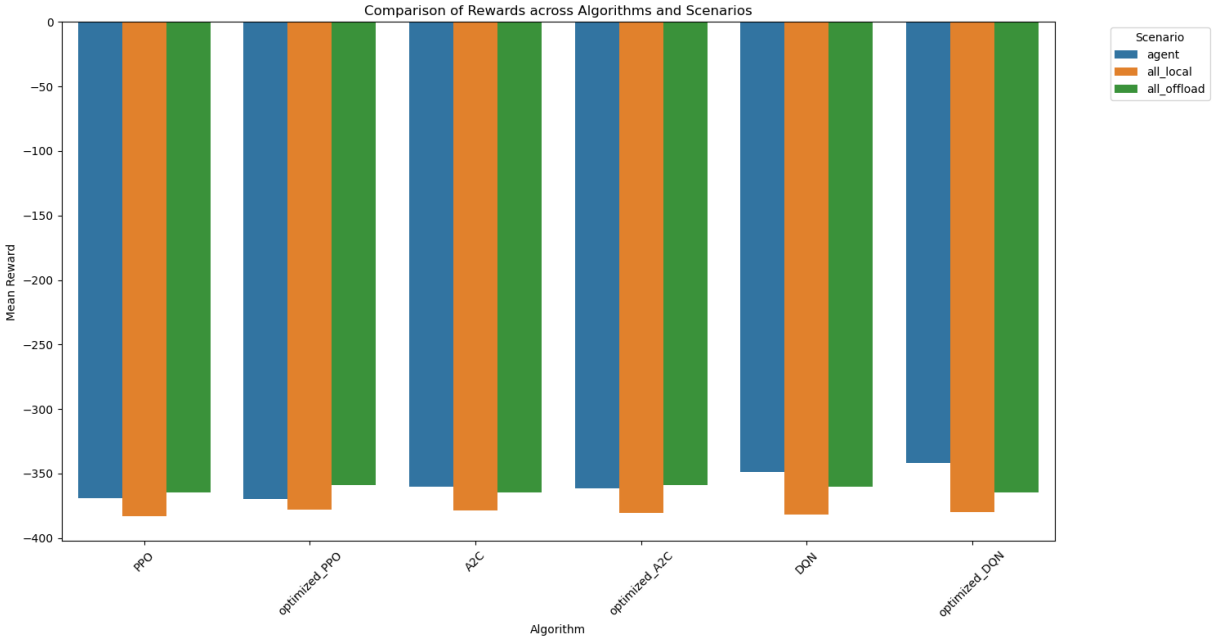
Εικόνα 26 - Σύγκριση συλλογής ανταμοιβών μεταξύ αλγορίθμων (w_time=1 & w_energy=0)

Η Εικόνα 26 παρουσιάζει τη μέση ανταμοιβή για w_time=1 και w_energy=0. Στην περίπτωση που καθορίσουμε η συνάρτηση ανταμοιβής, να διαμορφώνεται μόνο με βάση την χρονική καθυστέρηση, διαπιστώνουμε ότι ο αλγόριθμος “optimized_DQN” εμφανίζει την μεγαλύτερη ανταμοιβή. Σύμφωνα με την συλλογή ανταμοιβής, οι 5 από τους 6 αλγορίθμους εμφανίζουν ικανοποιητική συμπεριφορά.



Εικόνα 27 - Σύγκριση συλλογής ανταμοιβών μεταξύ αλγορίθμων (w_time=0 & w_energy=1)

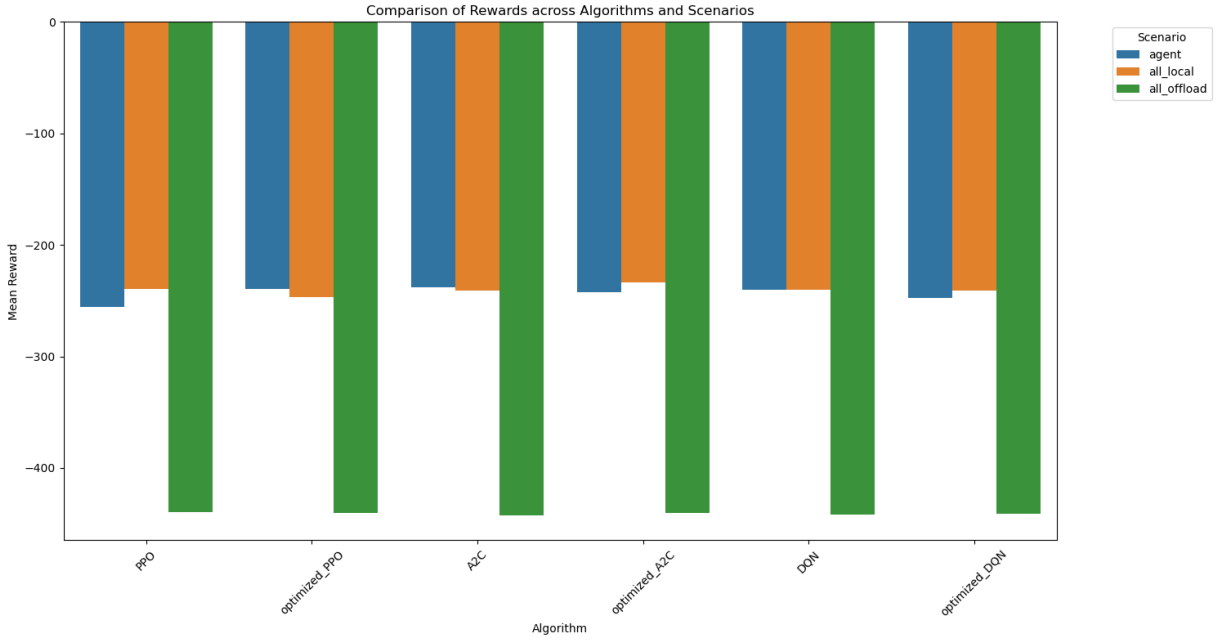
Στην Εικόνα 27 παρουσιάζεται η μέση ανταμοιβή για w_time=0 και w_energy=1, όταν δηλαδή ο στόχος βελτιστοποίησης είναι η μείωση της ενεργειακής κατανάλωσης. Σε αυτή την περίπτωση, παρατηρούμε ότι οι "optimized_PPO", "A2C" και "optimized_PPO", πρόσφεραν βελτιστοποίηση του προβλήματος.



Εικόνα 28 - Σύγκριση συλλογής ανταμοιβών μεταξύ αλγορίθμων (w_time=0.7 & w_energy=0.3)

Η Εικόνα 28 παρουσιάζει τη μέση ανταμοιβή όταν w_time=0.7 και w_energy=0.3. Το συγκεκριμένο σενάριο αφορά έναν ενδιάμεσο στόχο ο οποίος δίνει μεγαλύτερη βαρύτητα στη βελτίωση της χρονικής καθυστέρησης. Σε αυτή την περίπτωση, όλοι οι αλγόριθμοι είχαν παρόμοια συλλογή ανταμοιβής με το σενάριο όπου όλες οι διεργασίες εκτελούνται

στο διακομιστή, εκτός από τους “DQN” και “optimized_DQN” που έχουν λίγο καλύτερη επίδοση.



Εικόνα 29 - Σύγκριση συλλογής ανταμοιβών μεταξύ αλγορίθμων (w_time=0.3 & w_energy=0.7)

Η Εικόνα 29 παρουσιάζει τη μέση ανταμοιβή για w_time=0.3 και w_energy=0.7, δίνοντας μεγαλύτερη βαρύτητα στη μείωση ενεργειακής κατανάλωσης σε σχέση με την χρονική καθυστέρηση, Συγκεκριμένα, παρατηρούμε ότι μόνο οι “optimized_PPO” και “A2C” είχαν βελτιστοποιημένη απόδοση, ενώ οι υπόλοιποι εκτός του “PPO”, είχαν αρκετά καλή απόδοση.

4.4 Σχολιασμός συγκριτικών αποτελεσμάτων

Αφού πραγματοποιήθηκαν οι απαραίτητες μετρήσεις και η σύγκριση των αλγορίθμων σε κάθε διάγραμμα, σχολιάζονται τα συγκριτικά αποτελέσματα που αφορούν στη συνολική συμπεριφορά των πρακτόρων.

Κατά την σύγκριση των Εικόνων 18, 22 και 26, όταν w_time=1 και w_energy=0, παρατηρούμε ότι επιτυγχάνονται χαμηλές τιμές για την χρονική καθυστέρηση και αυξημένες για την κατανάλωση ενέργειας. Η συνάρτηση ανταμοιβής έχει τιμές κοντά στο -300 και, αντίστοιχα, χαμηλότερες τιμές στους αλγορίθμους που εμφάνιζαν χειρότερη συμπεριφορά.

Από την σύγκριση των Εικόνων 19, 23 και 27, όταν w_time=0 και w_energy=1, παρατηρούνται χαμηλές τιμές για την ενεργειακή κατανάλωση και υψηλές τιμές για την χρονική καθυστέρηση. Στην συνάρτηση ανταμοιβής οι τιμές βρίσκονται κοντά στο -140, με τον “optimized_PPO” να επιτυγχάνει την καλύτερη τιμή στο -125. Επίσης, στις περιπτώσεις των “optimized_PPO” και “A2C” συναντάμε ταυτόχρονη βελτίωση και στη χρονική καθυστέρηση, αλλά και στην ενεργειακή κατανάλωση.

Από τις Εικόνες 20, 24 και 28 που περιγράφουν ένα πιο ισορροπημένο σενάριο με έμφαση στον χρόνο (για w_time=0.7 και w_energy=0.3), παρατηρούμε διαφορετικές επιδόσεις από κάθε αλγόριθμο, ενώ όλοι έχουν χαμηλές τιμές για την συνάρτηση ανταμοιβής, με υψηλότερη αυτή του “optimized_DQN” περίπου στο -340. Οι αλγόριθμοι “optimized_PPO”, “DQN” και “optimized_DQN” επέλεξαν δράσεις που είχαν ως αποτέλεσμα υψηλή ενεργειακή κατανάλωση και χαμηλή χρονική καθυστέρηση. Παρά το γεγονός ότι οι υπόλοιποι αλγόριθμοι είχαν υψηλότερη χρονική καθυστέρηση, οι τιμές στη συνάρτηση ανταμοιβής ήταν παρόμοιες. Αυτό δείχνει ότι η συνάρτηση ανταμοιβής δεν είναι αξιόπιστη για να κριθεί ότι επιτυγχάνεται χαμηλότερη χρονική καθυστέρηση.

Για περιπτώσεις συντελεστών βαρύτητας όπου w_time=0.3 και w_energy=0.7, οι αλγόριθμοι στις Εικόνας 21, 24 και 29, εμφανίζουν βελτίωση της ενεργειακής κατανάλωσης σε συνδυασμό με αύξηση της χρονικής καθυστέρησης προκειμένου να συλλέξουν την μεγαλύτερη δυνατή επιστροφή ανταμοιβής. Την καλύτερη απόδοση παρουσιάζει ο “A2C” με τιμή -238 και ακολουθούν οι “optimized_PPO” και “DQN” με -240. Αξίζει να σημειωθεί ότι, οι “optimized_A2C” και “optimized_DQN”, έχουν επίσης υψηλή τιμή αλλά όχι καλύτερη συγκριτικά με την ανταμοιβή από το σενάριο της εκτέλεσης των διεργασιών τοπικά.

Κεφάλαιο 5: Συμπεράσματα και Μελλοντική Εργασία

Σε αυτή τη διπλωματική εργασία αξιοποιήθηκαν τεχνικές ΕΜ για τη βελτιστοποίηση της διαδικασίας υπολογιστικής εκφόρτωσης σε περιβάλλον MEC. Η βελτιστοποίηση υπολογιστικής εκφόρτωσης αποτελεί σημαντική πρόκληση στον τομέα των σύγχρονων δικτύων επικοινωνιών και υπολογιστικών συστημάτων, όπου η αποτελεσματική διαχείριση των πόρων και η βελτιστοποίηση της ενεργειακής απόδοσης είναι κρίσιμα ζητήματα. Η έρευνα επιχείρησε να συνεισφέρει στην κατανόηση και αντιμετώπιση αυτών των ζητημάτων, προτείνοντας και αξιολογώντας μια προσέγγιση βασισμένη στην ΕΜ.

Η παρούσα διπλωματική εργασία συνεισφέρει στον τομέα της βελτιστοποίησης υπολογιστικής εκφόρτωσης σε περιβάλλοντα MEC με τους εξής τρόπους:

1. Παρέχει ένα πλαίσιο για τη μοντελοποίηση και προσομοίωση περιβαλλόντων MEC με χρήση του OpenAI Gym.
2. Προσφέρει μια συγκριτική ανάλυση της απόδοσης διαφόρων αλγορίθμων ΕΜ στο πρόβλημα υπολογιστικής εκφόρτωσης.
3. Αναδεικνύει τις προκλήσεις και τις ευκαιρίες στην εφαρμογή τεχνικών ΕΜ σε πολύπλοκα, δυναμικά περιβάλλοντα.

Στην παρούσα μελέτη, το μοντέλο προσομοίωσης παράγει εργασίες με ευρεία ποικιλία χαρακτηριστικών, αντικατοπτρίζοντας την πολυπλοκότητα των πραγματικών σεναρίων σε περιβάλλοντα MEC. Η προσέγγιση επικεντρώθηκε στην υλοποίηση και σύγκριση τριών διαφορετικών αλγορίθμων ενισχυτικής μάθησης: PPO, A2C και DQN, καθώς και των βελτιστοποιημένων εκδοχών τους με την εύρεση κατάλληλων υπερπαραμέτρων για το συγκεκριμένο περιβάλλον. Η βελτιστοποίηση στόχευσε σε δύο στόχους, και συγκεκριμένα στη μείωση της χρονικής καθυστέρησης και της κατανάλωσης ενέργειας.

Η συνάρτηση ανταμοιβής που χρησιμοποιήθηκε συνδυάζει αυτούς τους δύο παράγοντες, με τη δυνατότητα προσαρμογής των βαρών τους. Στις οριακές περιπτώσεις όπου τα βάρη είναι 1 και 0 (ή το αντίστροφο), η αξιολόγηση της απόδοσης έγινε άμεσα με βάση τον κυρίαρχο στόχο βελτιστοποίησης. Ωστόσο, σε πιο ισορροπημένα σενάρια (π.χ., βάρη 0.7 και 0.3), η αξιολόγηση βασίστηκε στη συνολική βαθμολογία της συνάρτησης ανταμοιβής. Ένα σημαντικό ζήτημα που προέκυψε ήταν η δυσκολία σύγκρισης μεταξύ διαφορετικών εκτελέσεων του αλγορίθμου, λόγω της μεγάλης διακύμανσης στα χαρακτηριστικά των εργασιών. Για την αντιμετώπιση αυτού του προβλήματος, εισήχθησαν δύο ακραία σενάρια αναφοράς: η εκτέλεση όλων των εργασιών αποκλειστικά στον διακομιστή ή αποκλειστικά στη συσκευή του χρήστη. Αυτή η προσέγγιση επέτρεψε τη συγκριτική

αξιολόγηση της απόδοσης του αλγορίθμου σε σχέση με αυτά τα σταθερά σημεία αναφοράς.

Τα βασικά ευρήματα της μελέτης δείχνουν ότι οι αλγόριθμοι ΕΜ μπορούν να εφαρμοστούν αποτελεσματικά στο πρόβλημα υπολογιστικής εκφόρτωσης σε περιβάλλοντα MEC, χρειάζεται όμως προσεκτική μελέτη και σχεδιασμός της συνολικής υλοποίησης. Στα αποτελέσματα λοιπόν, των δοκιμών, παρατηρήθηκε πως η χρήση των αλγορίθμων για επιλογή δράσεων βελτίωσε την απόδοση του συστήματος, τόσο σε περιπτώσεις όπου στόχος ήταν η μείωση της ενεργειακής κατανάλωσης, όσο και η μείωση χρονικής καθυστέρησης. Στην περίπτωση μείωσης ενεργειακής κατανάλωσης καλύτερη απόδοση παρουσιάζουν οι αλγόριθμοι "optimized_PPO" και "DQN", ενώ στην περίπτωση μείωσης χρονικής καθυστέρησης ο "optimized_DQN". Σημαντικό παράγοντα αποτέλεσε η κρισιμότητα επιλογής υπερπαραμέτρων, καθώς ανάλογα με τον σχηματισμό των καταστάσεων δράσεων και παρατηρήσεων παρατηρήθηκε διαφορετική συμπεριφορά από τους αλγορίθμους.

Η μελέτη αυτή έχει ορισμένους περιορισμούς που θα μπορούσαν να αποτελέσουν τη βάση για μελλοντική έρευνα:

- Το μοντέλο προσομοίωσης απλοποιεί ορισμένες πτυχές των πραγματικών περιβαλλόντων MEC. Μελλοντική έρευνα θα μπορούσε να ενσωματώσει πιο πολύπλοκα μοντέλα δικτύου και κινητικότητας χρηστών.
- Η μελέτη περιορίστηκε σε τρεις αλγορίθμους ΕΜ. Η διερεύνηση πιο προηγμένων τεχνικών, όπως μάθηση με πολλαπλούς πράκτορες, θα μπορούσε να οδηγήσει σε ακόμη καλύτερα αποτελέσματα.
- Η βελτιστοποίηση των υπερπαραμέτρων των αλγορίθμων δεν ήταν εκτενής. Μια πιο συστηματική προσέγγιση στη ρύθμιση των υπερπαραμέτρων θα μπορούσε να βελτιώσει περαιτέρω την απόδοση. Συγκεκριμένα, στην περίπτωση του A2C, όπου η αναζήτηση βέλτιστων υπερπαραμέτρων δεν ολοκληρώθηκε επιτυχώς.

Η εφαρμογή τεχνικών ΕΜ στη βελτιστοποίηση της υπολογιστικής εκφόρτωσης σε περιβάλλοντα MEC αποτελεί ένα καίριο πεδίο έρευνας με σημαντικές προεκτάσεις στην εξέλιξη των σύγχρονων υπολογιστικών και επικοινωνιακών συστημάτων. Καθώς η ψηφιακή τεχνολογία διεισδύει όλο και περισσότερο σε κάθε πτυχή της καθημερινότητας, από τις έξυπνες πόλεις και τα αυτόνομα οχήματα μέχρι τα φορητά συστήματα υγείας, η ανάγκη για αποτελεσματική διαχείριση πόρων και βελτιστοποίηση της ενεργειακής απόδοσης καθίσταται ολοένα και πιο επιτακτική.

Τα ευρήματα και οι μεθοδολογίες που αναπτύχθηκαν σε αυτή την έρευνα μπορούν να αποτελέσουν τη βάση για περαιτέρω εξέλιξη των τεχνολογιών Edge Computing και TN, συμβάλλοντας στη δημιουργία πιο έξυπνων, αποδοτικών και βιώσιμων υπολογιστικών οικοσυστημάτων. Καθώς η τεχνολογική πρόοδος συνεχίζεται με ραγδαίους ρυθμούς, η ενσωμάτωση τέτοιων προηγμένων τεχνικών μάθησης στη διαχείριση και βελτιστοποίηση των υπολογιστικών πόρων θα είναι καθοριστική για την επίτευξη των στόχων απόδοσης, αποδοτικότητας και βιωσιμότητας των μελλοντικών ψηφιακών υποδομών και υπηρεσιών.

Παράρτημα Α – Πηγαίος Κώδικας

```
import numpy as np
import gymnasium as gym
from gymnasium import spaces
from stable_baselines3 import PPO, A2C, DQN
from stable_baselines3.common.evaluation import evaluate_policy
from stable_baselines3.common.callbacks import BaseCallback
from stable_baselines3.common.torch_layers import BaseFeaturesExtractor
from stable_baselines3.common.policies import ActorCriticPolicy
from stable_baselines3.common.monitor import Monitor
from stable_baselines3.common.logger import configure
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
import torch
import torch.nn as nn
import os
import csv
import optuna
from typing import Dict, Any
import json

class Device:
    def __init__(self):
        self.reset()

    def reset(self):
        self.cpu_freq = np.random.uniform(1, 5)  # CPU frequency in GHz
        self.bandwidth = np.random.uniform(1, 5)  # Bandwidth in Mbps
        self.allocated_mec_freq = np.random.uniform(4, 9)  # MEC server frequency in GHz

class Task:
    def __init__(self):
        self.reset()

    def reset(self):
        self.data_size = np.random.uniform(1, 50)  # Data size in MB
```

```
        self.required_cycles = np.random.uniform(100, 5000)  # Required CPU cycles in
GCycles

class MECEnvironment(gym.Env):
    def __init__(self, num_devices=5):
        super(MECEnvironment, self).__init__()

        self.num_devices = num_devices
        self.devices = [Device() for _ in range(num_devices)]
        self.tasks = [Task() for _ in range(num_devices)]

        self.action_space = spaces.Discrete(2**num_devices)
        obs_dim = num_devices * 5
        self.observation_space = spaces.Box(low=0, high=1, shape=(obs_dim,),
dtype=np.float32)

        self.current_step = 0
        self.max_steps = 100
        self.step_data = []

    def reset(self, seed=None):
        super().reset(seed=seed)
        self.current_step = 0
        for device in self.devices:
            device.reset()
        for task in self.tasks:
            task.reset()
        self.step_data = []
        return self._get_observation(), {}

    def step(self, action):
        binary_action = [int(x) for x in format(action, f'0{self.num_devices}b')]

        total_reward = 0
        total_energy = 0
        total_time = 0
        for i, a in enumerate(binary_action):
            if a == 0:
                reward, energy, time = self._execute_locally(self.devices[i], self.tasks[i])
            else:
                reward, energy, time = self._offload_to_mec(self.devices[i], self.tasks[i])
            total_reward += reward
            total_energy += energy
            total_time += time

        obs = self._get_observation()
        reward = total_reward

        self.step_data.append({
            'step': self.current_step,
```

```
            'action': binary_action,
            'reward': reward,
            'energy': total_energy,
            'time': total_time
        })

        self.current_step += 1
        done = self.current_step >= self.max_steps
        info = {'energy': total_energy, 'time': total_time}
        return obs, reward, done, False, info

    def _execute_locally(self, device, task):
        execution_time = task.required_cycles / device.cpu_freq
        energy_consumption = self._calculate_energy_consumption(device, task,
is_local=True)
        max_division_time, max_division_energy = self.calculate_max_division(device, task)
        reward = self._calculate_cost(execution_time, energy_consumption,
max_division_time, max_division_energy, local=True)
        return reward, energy_consumption, execution_time

    def _offload_to_mec(self, device, task):
        transmission_time = (task.data_size * 8) / device.bandwidth
        mec_execution_time = task.required_cycles / device.allocated_mec_freq
        total_time = transmission_time + mec_execution_time
        energy_consumption = self._calculate_energy_consumption(device, task,
is_local=False)
        max_division_time, max_division_energy = self.calculate_max_division(device, task)
        reward = self._calculate_cost(total_time, energy_consumption, max_division_time,
max_division_energy, local=False)
        return reward, energy_consumption, total_time

    def _calculate_energy_consumption(self, device, task, is_local):
        if is_local:
            return task.required_cycles * ((device.cpu_freq) ** 2) * 1e-3
        else:
            transmission_energy = ((task.data_size * 8) / device.bandwidth) * 0.02
            server_energy = task.required_cycles * ((device.allocated_mec_freq)**2) * 1e-3
            return transmission_energy + server_energy

    def calculate_max_division(self, device, task):
        self.max_div_time = max((task.required_cycles / device.cpu_freq),
((task.required_cycles / device.allocated_mec_freq)+((task.data_size * 8) /
device.bandwidth)))
        self.max_div_energy = max((task.required_cycles * ((device.cpu_freq) ** 2) * 1e-3),
((task.required_cycles * ((device.allocated_mec_freq)**2) * 1e-3)+((task.data_size * 8) /
device.bandwidth) * 0.02))
        return self.max_div_time, self.max_div_energy

    def _calculate_cost(self, time, energy, max_division_time, max_division_energy, local):
        w_t, w_e = 1, 0
```

```
        time_cost = time / max_division_time
        energy_cost = energy / max_division_energy
        costs = w_t * time_cost + w_e * energy_cost
        reward = -costs
        return reward

    def _get_observation(self):
        obs = []
        for device, task in zip(self.devices, self.tasks):
            obs.extend([
                (device.cpu_freq-1) / (5-1),
                (device.bandwidth-1) / (5-1),
                (device.allocated_mec_freq-4) / (9-4),
                (task.data_size-1) / (50-1),
                (task.required_cycles-100) / (5000-100),
            ])
        return np.array(obs, dtype=np.float32)

    def save_step_data(self, filename):
        with open(filename, 'w', newline='') as f:
            writer = csv.DictWriter(f, fieldnames=['step', 'action', 'reward', 'energy', 'time'])
            writer.writeheader()
            for step in self.step_data:
                writer.writerow(step)

class TensorboardCallback(BaseCallback):
    def __init__(self, verbose=0):
        super(TensorboardCallback, self).__init__(verbose)
        self.episode_rewards = []
        self.episode_lengths = []
        self.episode_count = 0

    def _on_step(self) -> bool:
        if self.locals["dones"]:
            self.episode_rewards.append(self.locals["rewards"][0])
            self.episode_lengths.append(self.num_timesteps - sum(self.episode_lengths))
            self.episode_count += 1
            if self.episode_count % 10 == 0:
                mean_reward = np.mean(self.episode_rewards[-10:])
                mean_length = np.mean(self.episode_lengths[-10:])
                self.logger.record("rollout/ep_rew_mean", mean_reward)
                self.logger.record("rollout/ep_len_mean", mean_length)
        return True

def make_env():
    env = MECEnvironment(num_devices=5, mec_capacity=50)
    env = Monitor(env)
    return env

def optimize_ppo(trial: optuna.Trial) -> Dict[str, Any]:
```

```
    return {
        "learning_rate": trial.suggest_loguniform("learning_rate", 1e-5, 1e-3),
        "n_steps": trial.suggest_int("n_steps", 16, 2048),
        "batch_size": trial.suggest_int("batch_size", 8, 256),
        "n_epochs": trial.suggest_int("n_epochs", 3, 30),
        "gamma": trial.suggest_uniform("gamma", 0.9, 0.9999),
        "gae_lambda": trial.suggest_uniform("gae_lambda", 0.9, 1.0),
        "clip_range": trial.suggest_uniform("clip_range", 0.1, 0.4),
        "ent_coef": trial.suggest_loguniform("ent_coef", 1e-8, 1e-1),
    }

def optimize_a2c(trial: optuna.Trial) -> Dict[str, Any]:
    return {
        "learning_rate": trial.suggest_loguniform("learning_rate", 1e-5, 1e-3),
        "n_steps": trial.suggest_int("n_steps", 1, 100),
        "gamma": trial.suggest_uniform("gamma", 0.9, 0.9999),
        "gae_lambda": trial.suggest_uniform("gae_lambda", 0.9, 1.0),
        "ent_coef": trial.suggest_loguniform("ent_coef", 1e-8, 1e-1),
        "vf_coef": trial.suggest_uniform("vf_coef", 0, 1),
    }

def optimize_dqn(trial: optuna.Trial) -> Dict[str, Any]:
    return {
        "learning_rate": trial.suggest_loguniform("learning_rate", 1e-5, 1e-3),
        "buffer_size": trial.suggest_int("buffer_size", 10000, 1000000),
        "learning_starts": trial.suggest_int("learning_starts", 1000, 50000),
        "batch_size": trial.suggest_int("batch_size", 8, 256),
        "gamma": trial.suggest_uniform("gamma", 0.9, 0.9999),
        "tau": trial.suggest_uniform("tau", 0.001, 0.1),
        "train_freq": trial.suggest_int("train_freq", 1, 10),
        "gradient_steps": trial.suggest_int("gradient_steps", -1, 10),
    }

def objective(trial: optuna.Trial, algo: str, env) -> float:
    if algo == "PPO":
        model = PPO("MlpPolicy", env, verbose=0, **optimize_ppo(trial))
    elif algo == "A2C":
        model = A2C("MlpPolicy", env, verbose=0, **optimize_a2c(trial))
    elif algo == "DQN":
        model = DQN("MlpPolicy", env, verbose=0, **optimize_dqn(trial))
    else:
        raise ValueError(f"Unknown algorithm: {algo}")

    model.learn(total_timesteps=100000)
    mean_reward, _ = evaluate_policy(model, env, n_eval_episodes=50)
    return mean_reward

def optimize_hyperparameters(algo: str, env, n_trials: int = 50) -> Dict[str, Any]:
    study = optuna.create_study(direction="maximize")
    study.optimize(lambda trial: objective(trial, algo, env), n_trials=n_trials)
```

```
    return study.best_params

def train_and_evaluate(env, model_class, model_name, total_timesteps=200000,
hyperparams=None):
    log_dir = f"./tensorboard_logs/{model_name}"
    os.makedirs(log_dir, exist_ok=True)
    new_logger = configure(log_dir, ["stdout", "tensorboard"])

    if hyperparams:
        model = model_class("MlpPolicy", env, verbose=1, tensorboard_log=log_dir,
**hyperparams)
    else:
        model = model_class("MlpPolicy", env, verbose=1, tensorboard_log=log_dir)

    model.set_logger(new_logger)
    callback = TensorboardCallback()

    try:
        model.learn(total_timesteps=total_timesteps, callback=callback)
        mean_reward, std_reward = evaluate_policy(model, env, n_eval_episodes=200)
    except Exception as e:
        print(f"Error during training {model_name}: {str(e)}")
        return None, float('-inf'), 0

    return model, mean_reward, std_reward

def run_scenarios(env, model):
    num_episodes = 100
    results = {
        'agent': {'rewards': [], 'energy': [], 'time': []},
        'all_local': {'rewards': [], 'energy': [], 'time': []},
        'all_offload': {'rewards': [], 'energy': [], 'time': []}
    }

    for _ in range(num_episodes):
        episode_data = {scenario: {'reward': 0, 'energy': 0, 'time': 0} for scenario in
results.keys()}

        # Agent scenario
        obs, _ = env.reset()
        done = False
        while not done:
            action, _ = model.predict(obs)
            obs, reward, done, _, info = env.step(action)
            episode_data['agent']['reward'] += reward
            episode_data['agent']['energy'] += info.get('energy', 0)
            episode_data['agent']['time'] += info.get('time', 0)

        # All local scenario
        obs, _ = env.reset()
```

```
        done = False
        while not done:
            all_local_action = 0  # All zeros for local execution
            obs, reward, done, _, info = env.step(all_local_action)
            episode_data['all_local']['reward'] += reward
            episode_data['all_local']['energy'] += info.get('energy', 0)
            episode_data['all_local']['time'] += info.get('time', 0)

        # All offload scenario
        obs, _ = env.reset()
        done = False
        while not done:
            all_offload_action = 2**env.num_devices - 1  # All ones for offloading
            obs, reward, done, _, info = env.step(all_offload_action)
            episode_data['all_offload']['reward'] += reward
            episode_data['all_offload']['energy'] += info.get('energy', 0)
            episode_data['all_offload']['time'] += info.get('time', 0)

        for scenario in results.keys():
            results[scenario]['rewards'].append(episode_data[scenario]['reward'])
            results[scenario]['energy'].append(episode_data[scenario]['energy'])
            results[scenario]['time'].append(episode_data[scenario]['time'])

    return results

def compare_rl_algorithms(env, total_timesteps=200000):
    algorithms = [
        ("PPO", PPO),
        ("optimized_PPO", PPO),
        ("A2C", A2C),
        ("optimized_A2C", A2C),
        ("DQN", DQN),
        ("optimized_DQN", DQN)
    ]

    all_results = []
    best_params = {}

    # Predefined hyperparameters for the already found best results
    best_params["PPO"] = {
        'learning_rate': 0.00021776512127397638,   #7.746429460012185e-05,
        'n_steps': 760,    #734,
        'batch_size': 256, #101,
        'n_epochs': 8, #15,
        'gamma': 0.9144199897934076,   #0.9452064632034392,
        'gae_lambda': 0.9183075916025104,  #0.9695003841959792,
        'clip_range': 0.37498501251387906, #0.3382938299765875,
        'ent_coef': 1.0331644131803022e-05,    #2.0584042532590406e-06
    }
    best_params["A2C"] = {
```

```
    'learning_rate': 0.0004977724684525267,
#0.0004977724684525267,3.49820668101203e-05
    'n_steps': 14, #11,
    'gamma': 0.9167298667573246,   #0.9167298667573246,0.9407971996014384
    'gae_lambda': 0.9456823593500105,  #0.9456823593500105,0.9393872057187274
    'ent_coef': 2.233502032574826e-08, #2.233502032574826e-
08,4.251946784156469e-07
    'vf_coef': 0.18899376621510114, #0.18899376621510114,0.7807320513267174
  }
  best_params["DQN"] = {
    'learning_rate': 2.2350795980167074e-05,   #0.00027311863228128834,
    'buffer_size': 610948, #359674,
    'learning_starts': 31091,  #39700,
    'batch_size': 174, #181,
    'gamma': 0.9648660982239635,   #0.9736630706117637,
    'tau': 0.048445057211731056,   #0.07062676666370174,
    'train_freq': 10,
    'gradient_steps': 8   #2
  }

  for name, algo_class in algorithms:
    print(f"Training and evaluating {name}...")

    if name == "optimized_PPO":
      if "PPO" not in best_params:
        print(f"Optimizing hyperparameters for PPO...")
        best_params["PPO"] = optimize_hyperparameters("PPO", env)
      hyperparams = best_params["PPO"]
    elif name == "optimized_A2C":
      if "A2C" not in best_params:
        print(f"Optimizing hyperparameters for A2C...")
        best_params["A2C"] = optimize_hyperparameters("A2C", env)
      hyperparams = best_params["A2C"]
    elif name == "optimized_DQN":
      if "DQN" not in best_params:
        print(f"Optimizing hyperparameters for DQN...")
        best_params["DQN"] = optimize_hyperparameters("DQN", env)
      hyperparams = best_params["DQN"]
    else:
      hyperparams = None

    model, mean_reward, std_reward = train_and_evaluate(env, algo_class, name,
total_timesteps, hyperparams)

    if model is not None:
      scenario_results = run_scenarios(env, model)

      for scenario, data in scenario_results.items():
        all_results.append({
          "Algorithm": name,
```

```
                "Scenario": scenario,
                "Mean Reward": np.mean(data['rewards']),
                "Std Reward": np.std(data['rewards']),
                "Mean Energy": np.mean(data['energy']),
                "Std Energy": np.std(data['energy']),
                "Mean Time": np.mean(data['time']),
                "Std Time": np.std(data['time'])
            })

    df = pd.DataFrame(all_results)

    # Save results to CSV
    df.to_csv("algorithm_scenario_comparison.csv", index=False)

    # Save best hyperparameters to JSON
    with open("best_hyperparameters.json", "w") as f:
        json.dump(best_params, f, indent=4)

    # Plot comparisons
    plot_comparison(df, "Reward", "Comparison of Rewards across Algorithms and
Scenarios")
    plot_comparison(df, "Energy", "Comparison of Energy Consumption across Algorithms
and Scenarios")
    plot_comparison(df, "Time", "Comparison of Time Consumption across Algorithms and
Scenarios")

    return df

def plot_comparison(df, metric, title):
    plt.figure(figsize=(15, 8))
    sns.barplot(x="Algorithm", y=f"Mean {metric}", hue="Scenario", data=df)
    plt.title(title)
    plt.ylabel(f"Mean {metric}")
    plt.xticks(rotation=45)
    plt.legend(title="Scenario", bbox_to_anchor=(1.05, 1), loc='upper left')
    plt.tight_layout()
    plt.savefig(f"{metric.lower()}_comparison.png")
    plt.close()

if __name__ == "__main__":
    print("Starting MEC Offloading Experiment with Specific Runs and DQN Optimization")
    env = make_env()
    print("\nTraining and Comparing RL Algorithms...")
    algorithm_comparison = compare_rl_algorithms(env)
    print("RL Algorithm and Scenario Comparison Results:")
    print(algorithm_comparison)
    print("\nSimulation completed.")
```

Βιβλιογραφία

[1] T. M. Mitchell, Machine Learning. New York, NY, USA: McGraw-Hill, 1997.

[2] I. Goodfellow, Y. Bengio, and A. Courville, Deep Learning. Cambridge, MA, USA: MIT Press, 2016.

[3] G. Tangirala, S. Garikipati, D. M. K. Chaitanya, and V. Sharma, "A Review on Optimization Techniques of Antennas Using AI and ML / DL Algorithms," Int. J. Adv. Microw. Technol., vol. 7, pp. 288-295, 2022.

[4] S. J. Russell and P. Norvig, Artificial Intelligence: A Modern Approach, 4th ed. Hoboken, NJ, USA: Pearson, 2020.

[5] M. I. Jordan and T. M. Mitchell, "Machine learning: Trends, perspectives, and prospects," Science, vol. 349, no. 6245, pp. 255-260, Jul. 2015.

[6] R. S. Sutton and A. G. Barto, Reinforcement Learning: An Introduction, 2nd ed. Cambridge, MA, USA: MIT Press, 2018.

[7] T. Hastie, R. Tibshirani, and J. Friedman, The Elements of Statistical Learning: Data Mining, Inference, and Prediction, 2nd ed. New York, NY, USA: Springer, 2009.

[8] C. M. Bishop, Pattern Recognition and Machine Learning. New York, NY, USA: Springer, 2006.

[9] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," Nature, vol. 521, no. 7553, pp. 436-444, May 2015.

[10] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in Advances in Neural Information Processing Systems 25, 2012, pp. 1097-1105.

[11] D. Silver et al., "Mastering the game of Go with deep neural networks and tree search," Nature, vol. 529, no. 7587, pp. 484-489, Jan. 2016.

[12] V. N. Vapnik, Statistical Learning Theory. New York, NY, USA: Wiley-Interscience, 1998.

[13] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in Proc. of the 2019 Conf. of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, 2019, pp. 4171-4186.

[14] X. Zhu, "Semi-supervised learning literature survey," Dept. Comput. Sci., Univ. Wisconsin-Madison, Madison, WI, USA, Tech. Rep. 1530, 2005.

[15] G. E. Hinton and T. J. Sejnowski, Eds., Unsupervised Learning: Foundations of Neural Computation. Cambridge, MA, USA: MIT Press, 1999..

[16] Y. Bengio, A. Courville, and P. Vincent, "Representation learning: A review and new perspectives," IEEE Trans. Pattern Anal. Mach. Intell., vol. 35, no. 8, pp. 1798-1828, Aug. 2013.

[17] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise," in Proc. 2nd Int. Conf. Knowledge Discovery and Data Mining, 1996, pp. 226-231.

[18] A. Boulesnane, "Evolutionary Dynamic Optimization and Machine Learning," in Advanced Machine Learning with Evolutionary and Metaheuristic Techniques, arXiv, In press, 2023.

[19] L. Breiman, "Random forests," Mach. Learn., vol. 45, no. 1, pp. 5-32, Oct. 2001.

[20] L. van der Maaten and G. Hinton, "Visualizing data using t-SNE," J. Mach. Learn. Res., vol. 9, pp. 2579-2605, Nov. 2008.

[21] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," ACM Comput. Surv., vol. 41, no. 3, pp. 1-58, Jul. 2009.

[22] I. Goodfellow et al., "Generative adversarial nets," in Advances in Neural Information Processing Systems 27, 2014, pp. 2672-2680.

[23] G. E. Hinton and T. J. Sejnowski, Eds., Unsupervised Learning: Foundations of Neural Computation. Cambridge, MA, USA: MIT Press, 1999.

[24] L. P. Kaelbling, M. L. Littman, and A. W. Moore, "Reinforcement learning: A survey," J. Artif. Intell. Res., vol. 4, pp. 237-285, May 1996.

[25] V. Mnih et al., "Human-level control through deep reinforcement learning," Nature, vol. 518, no. 7540, pp. 529-533, Feb. 2015.

[26] C. J. Watkins and P. Dayan, "Q-learning," Mach. Learn., vol. 8, no. 3-4, pp. 279-292, May 1992.

[27] H. Zhang and T. Yu, "Taxonomy of Reinforcement Learning Algorithms," in Deep Reinforcement Learning, H. Dong, Z. Ding, and S. Zhang, Eds. Singapore: Springer, 2020.

[28] P. Auer, N. Cesa-Bianchi, and P. Fischer, "Finite-time analysis of the multiarmed bandit problem," Mach. Learn., vol. 47, no. 2-3, pp. 235-256, May 2002.

[29] T. Lattimore and C. Szepesvári, Bandit Algorithms. Cambridge, UK: Cambridge University Press, 2020.

[30] S. Bubeck and N. Cesa-Bianchi, "Regret analysis of stochastic and nonstochastic multi-armed bandit problems," Found. Trends Mach. Learn., vol. 5, no. 1, pp. 1-122, 2012.

[31] M. L. Puterman, Markov Decision Processes: Discrete Stochastic Dynamic Programming. Hoboken, NJ, USA: John Wiley & Sons, 2014.

[32] R. Bellman, Dynamic Programming. Princeton, NJ, USA: Princeton University Press, 1957.

[33] R. A. Howard, Dynamic Programming and Markov Processes. Cambridge, MA, USA: MIT Press, 1960.

[34] D. P. Bertsekas, Dynamic Programming and Optimal Control, 4th ed. Belmont, MA, USA: Athena Scientific, 2012.

[35] W. B. Powell, Approximate Dynamic Programming: Solving the Curses of Dimensionality, 2nd ed. Hoboken, NJ, USA: John Wiley & Sons, 2011.

[36] R. Y. Rubinstein and D. P. Kroese, Simulation and the Monte Carlo Method, 3rd ed. Hoboken, NJ, USA: John Wiley & Sons, 2016.

[37] P. Dayan, "The convergence of TD(λ) for general λ," Mach. Learn., vol. 8, no. 3-4, pp. 341-362, May 1992.

[38] R. S. Sutton, "Learning to predict by the methods of temporal differences," Mach. Learn., vol. 3, no. 1, pp. 9-44, Aug. 1988.

[39] G. A. Rummery and M. Niranjan, "On-line Q-learning using connectionist systems," Dept. Eng., Univ. Cambridge, Cambridge, UK, Tech. Rep. CUED/F-INFENG/TR 166, 1994.

[40] G. Tesauro, "Temporal difference learning and TD-Gammon," *Commun. ACM*, vol. 38, no. 3, pp. 58-68, Mar. 1995.

[41] S. Chen and Y. Li, "An Overview of Robust Reinforcement Learning," *2020 IEEE International Conference on Networking, Sensing and Control (ICNSC)*, Nanjing, China, 2020, pp. 1-6.

[42] C. G. Atkeson and J. C. Santamaria, "A comparison of direct and model-based reinforcement learning," in Proc. Int. Conf. Robot. Autom., 1997, pp. 3557-3564.

[43] R. S. Sutton, "Integrated architectures for learning, planning, and reacting based on approximating dynamic programming," in Proc. 7th Int. Conf. Mach. Learn., 1990, pp. 216-224.

[44] A. W. Moore and C. G. Atkeson, "Prioritized sweeping: Reinforcement learning with less data and less time," Mach. Learn., vol. 13, no. 1, pp. 103-130, Oct. 1993.

[45] M. G. Lagoudakis and R. Parr, "Least-squares policy iteration," J. Mach. Learn. Res., vol. 4, pp. 1107-1149, Dec. 2003.

[46] M. Rothmann and M. Porrmann, "A Survey of Domain-Specific Architectures for Reinforcement Learning," in *IEEE Access*, vol. 10, pp. 13753-13767, 2022

[47] K. Mathia and J. Clark, "On neural network hardware and programming paradigms," *Proceedings of the 2002 International Joint Conference on Neural Networks. IJCNN'02 (Cat. No.02CH37290)*, Honolulu, HI, USA, 2002, pp. 2692-2697 vol.3

[48] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," Nature, vol. 323, no. 6088, pp. 533-536, Oct. 1986.

[49] F. Bre, J. M. Gimenez, and V. D. Fachinotti, "Prediction of wind pressure coefficients on building surfaces using Artificial Neural Networks," Energy and Buildings, vol. 158, pp. 1490-1504, 2017.

[50] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in Proc. 3rd Int. Conf. Learn. Represent., 2015, pp. 1-15.

[51] Y. Bengio, "Learning deep architectures for AI," Found. Trends Mach. Learn., vol. 2, no. 1, pp. 1-127, 2009.

[52] J. Schmidhuber, "Deep learning in neural networks: An overview," Neural Netw., vol. 61, pp. 85-117, Jan. 2015.

[53] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," Proc. IEEE, vol. 86, no. 11, pp. 2278-2324, Nov. 1998.

[54] S. Hochreiter and J. Schmidhuber, "Long short-term memory," Neural Comput., vol. 9, no. 8, pp. 1735-1780, Nov. 1997.

[55] A. Vaswani et al., "Attention is all you need," in Advances in Neural Information Processing Systems 30, 2017, pp. 5998-6008.

[56] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath, "Deep reinforcement learning: A brief survey," IEEE Signal Process. Mag., vol. 34, no. 6, pp. 26-38, Nov. 2017.

[57] D. Silver et al., "Mastering the game of go without human knowledge," Nature, vol. 550, no. 7676, pp. 354-359, Oct. 2017.

[58] V. Mnih et al., "Asynchronous methods for deep reinforcement learning," in Proc. Int. Conf. Mach. Learn. (ICML), 2016, pp. 1928-1937.

[59] I. H. Sarker, "Deep Cybersecurity: A Comprehensive Overview from Neural Network and Deep Learning Perspective," SN Comput. Sci., vol. 2, no. 3, pp. 1-20, May 2021.

[60] L.-J. Lin, "Self-improving reactive agents based on reinforcement learning, planning and teaching," Mach. Learn., vol. 8, no. 3-4, pp. 293-321, May 1992.

[61] H. van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double q-learning," in Proc. 30th AAAI Conf. Artif. Intell., 2016, pp. 2094-2100.

[62] V. Mnih et al., "Playing atari with deep reinforcement learning," arXiv:1312.5602, 2013.

[63] Z. Wang et al., "Dueling network architectures for deep reinforcement learning," in Proc. 33rd Int. Conf. Mach. Learn., 2016, pp. 1995-2003.

[64] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," Mach. Learn., vol. 8, no. 3-4, pp. 229-256, May 1992.

[65] R. S. Sutton, D. A. McAllester, S. P. Singh, and Y. Mansour, "Policy gradient methods for reinforcement learning with function approximation," in Advances in Neural Information Processing Systems 12, 2000, pp. 1057-1063.

[66] V. R. Konda and J. N. Tsitsiklis, "Actor-critic algorithms," in Advances in Neural Information Processing Systems 12, 2000, pp. 1008-1014.

[67] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," arXiv:1707.06347, 2017.

[68] A. T. D. Perera and P. Kamalaruban, "Applications of reinforcement learning in energy systems," Renewable and Sustainable Energy Reviews, vol. 137, p. 110618, 2021.

[69] T. P. Lillicrap et al., "Continuous control with deep reinforcement learning," arXiv:1509.02971, 2015.

[70] S. Fujimoto, H. van Hoof, and D. Meger, "Addressing function approximation error in actor-critic methods," in Proc. 35th Int. Conf. Mach. Learn., 2018, pp. 1587-1596.

[71] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in Proc. 35th Int. Conf. Mach. Learn., 2018, pp. 1861-1870.

[72] P. Henderson et al., "Deep reinforcement learning that matters," in Proc. 32nd AAAI Conf. Artif. Intell., 2018, pp. 3207-3214.

[73] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath, "Deep reinforcement learning: A brief survey," IEEE Signal Process. Mag., vol. 34, no. 6, pp. 26-38, Nov. 2017.

[74] P. Mach and Z. Becvar, "Mobile edge computing: A survey on architecture and computation offloading," IEEE Commun. Surveys Tuts., vol. 19, no. 3, pp. 1628-1656, 3rd Quart. 2017.

[75] N. Abbas, Y. Zhang, A. Taherkordi, and T. Skeie, "Mobile edge computing: A survey," IEEE Internet Things J., vol. 5, no. 1, pp. 450-465, Feb. 2018.

[76] P. Mell and T. Grance, "The NIST definition of cloud computing," Nat. Inst. Stand. Technol., Gaithersburg, MD, USA, Tech. Rep. 800-145, 2011.

[77] ETSI, "Mobile-Edge Computing – Introductory Technical White Paper," ETSI, Sophia Antipolis, France, White Paper, Sep. 2014.

[78] Y. C. Hu, M. Patel, D. Sabella, N. Sprecher, and V. Young, "Mobile edge computing—A key technology towards 5G," ETSI, Sophia Antipolis, France, White Paper 11, Sep. 2015.

[79] K. Kumar, J. Liu, Y. H. Lu, and B. Bhargava, "A survey of computation offloading for mobile systems," Mobile Netw. Appl., vol. 18, no. 1, pp. 129-140, Feb. 2013.

[80] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, "A survey on mobile edge computing: The communication perspective," IEEE Commun. Surveys Tuts., vol. 19, no. 4, pp. 2322-2358, 4th Quart. 2017.

[81] Z. Chen and X. Wang, "Decentralized computation offloading for multi-user mobile edge computing: a deep reinforcement learning approach," EURASIP J. Wireless Commun. Netw., vol. 2020, no. 1, pp. 1-188, Sep. 2020.

[82] W. Zhang, Y. Wen, and D. O. Wu, "Collaborative task execution in mobile cloud computing under a stochastic wireless channel," IEEE Trans. Wireless Commun., vol. 14, no. 1, pp. 81-93, Jan. 2015.

[83] H. Flores, S. N. Srirama, and R. Buyya, "Computational offloading or data binding? Bridging the cloud infrastructure to the proximity of the mobile user," in Proc. IEEE 2nd Int. Conf. Mobile Cloud Comput., Services, Eng. (MobileCloud), 2017, pp. 160-163.

[84] S. Zhang, N. Yi, and Y. Ma, "A Survey of Computation Offloading With Task Types," IEEE Trans. Intell. Transp. Syst., vol. 25, no. 8, pp. 8313-8333, Aug. 2024.

[85] X. Chen, L. Jiao, W. Li, and X. Fu, "Efficient Multi-User Computation Offloading for Mobile-Edge Cloud Computing," IEEE/ACM Trans. Netw., vol. 24, no. 5, pp. 2795-2808, Oct. 2016.

[86] X. Chen and G. Liu, "Energy-Efficient Task Offloading and Resource Allocation via Deep Reinforcement Learning for Augmented Reality in Mobile Edge Networks," IEEE Internet Things J., vol. 8, no. 13, pp. 10843-10856, Jul. 2021.

[87] H. Guo et al., "Mobile-edge computation offloading for ultra-dense IoT networks," IEEE Internet Things J., vol. 5, no. 6, pp. 4977-4988, Dec. 2018.

[88] J. He, "Optimization of Edge Delay Sensitive Task Scheduling Based on Genetic Algorithm," *2022 International Conference on Algorithms, Data Mining, and Information Technology (ADMIT)*, Xi'an, China, 2022, pp. 155-159

[89] H. Jin, M. A. Gregory, and S. Li, "A Review of Intelligent Computation Offloading in Multiaccess Edge Computing," IEEE Access, vol. 10, pp. 71481-71495, 2022.

[90] J. Li, H. Gao, T. Lv and Y. Lu, "Deep reinforcement learning based computation offloading and resource allocation for MEC," *2018 IEEE Wireless Communications and Networking Conference (WCNC)*, Barcelona, Spain, 2018, pp. 1-6

[91] A. Mahesri and V. Vardhan, "Power Consumption Breakdown on a Modern Laptop," in Power-Aware Computer Systems, B. Falsafi and T.N. VijayKumar, Eds. Berlin, Heidelberg: Springer, 2005, pp. 165-180.

Συντομογραφίες - Αρκτικόλεξα - Ακρωνύμια

Ακρωνύμια στην Ελληνική γλώσσα

ΕΜ	Ενισχυτική Μάθηση
κ.λπ.	και λοιπά
ΜΜ	Μηχανική Μάθηση
ΤΝ	Τεχνητή Νοημοσύνη

Ακρωνύμια στην Αγγλική γλώσσα

ACO	Ant Colony Optimization
DP	Dynamic Programming
EA	Evolutionary Algorithms
GA	Genetic Algorithms
GAN	Generative Adversarial Network
IaaS	Infrastructure as a Service
MC	Monte Carlo
MDP	Markov Decision Processes
MEC	Multi-Access/Mobile Edge Computing
PaaS	Platform as a Service
PCA	Principal Component Analysis
PSO	Particle Swarm Optimization
QoS	Quality of Service
SaaS	Software as a Service
SIA	Swarm Intelligence Algorithms
UCB	Upper Confidence Bound

Απόδοση Ξενόγλωσσων Όρων

Ανταμοιβή	Reward
Βαθιά Μάθηση	Deep Learning
Γενετικοί Αλγόριθμοι	Genetic Algorithms
Γραμμική Παλινδρόμηση	Linear Regression
Διακομιστής Άκρου	Edge Server
Δράση	Action
Δυναμικός Προγραμματισμός	Dynamic Programming
Ενισχυτική Μάθηση	Reinforcement Learning
Εξελικτικοί Αλγόριθμοι	Evolutionary Algorithms
Επισημασμένα Δεδομένα	Labeled Data
Ευέλικτο	Agile
Κατανομή Εργασιών	Task Allocation
Κατάσταση	State
Κλιμακώσιμο	Scalable
Λογισμικό ως Υπηρεσία	Software as a Service
Λογιστική Παλινδρόμηση	Logistic Regression
Μαρκοβιανές Διαδικασίες Αποφάσεων	Markov Decision Processes
Μεταφορά Υπολογιστικού Φόρτου	Computation Offloading
Μηχανές Διανυσμάτων Υποστήριξης	Support Vector Machines
Μηχανική Μάθηση	Machine Learning
Παραγωγική Τεχνητή Νοημοσύνη	Generative Artificial Intelligence
Περιβάλλον	Environment
Πλατφόρμα ως Υπηρεσία	Platform as a Service
Ποιότητα Υπηρεσίας	Quality of Service
Πολιτική	Policy
Πράκτορας	Agent
Συνάρτηση Ανταμοιβής	Reward Function
Συνάρτηση Αξίας	Value Function
Συνάρτηση Απώλειας	Loss Function
Σύνολο Δεδομένων Εκπαίδευσης	Training Dataset
Τεχνητή Νοημοσύνη	Artificial Intelligence
Υποδομή ως Υπηρεσία	Infrastructure as a Service
Υπολογιστική Εκφόρτωση	Computation Offloading
Υπολογιστική Νέφους	Cloud Computing
Υπολογιστική στα Άκρα του Δικτύου	Edge Computing
Χώρος Δράσεων	Action Space
Χώρος Παρατηρήσεων	Observation Space