ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ
ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΤΟΜΕΑΣ ΕΠΙΚΟΙΝΩΝΙΩΝ, ΗΛΕΚΤΡΟΝΙΚΗΣ ΚΑΙ
ΣΥΣΤΗΜΑΤΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ

Ανάθεση Cloud Native Εφαρμογών σε Κατανεμημένες και Συνεργατικές Υποδομές IoT-Edge-Cloud με Χρήση Τεχνικών Μηχανικής Μάθησης

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

του

Γεώργιος-Μιχαήλ Παπαγεωργίου

Επιβλέπων: Εμμανουήλ Βαρβαρίγος
Καθηγητής Ε.Μ.Π

Αθήνα, Ιούλιος 2025

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ
ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΤΟΜΕΑΣ ΕΠΙΚΟΙΝΩΝΙΩΝ, ΗΛΕΚΤΡΟΝΙΚΗΣ ΚΑΙ
ΣΥΣΤΗΜΑΤΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ

Ανάθεση Cloud Native Εφαρμογών σε Κατανεμημένες και Συνεργατικές Υποδομές IoT-Edge-Cloud με Χρήση Τεχνικών Μηχανικής Μάθησης

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

του

Γεώργιος-Μιχαήλ Παπαγεωργίου

Επιβλέπων: Εμμανουήλ Βαρβαρίγος
Καθηγητής Ε.Μ.Π

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 2η Ιουλίου 2025.

.....	.....	.....
Εμμανουήλ Βαρβαρίγος	Θεοδώρα Βαρβαρίγου	Κωνσταντίνος Τσέρπες
Καθηγητής Ε.Μ.Π	Καθηγήτρια Ε.Μ.Π	Επίκουρος Καθηγητής Ε.Μ.Π

Αθήνα, Ιούλιος 2025

.....

Γεώργιος-Μιχαήλ Παπαγεωργίου

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

Copyright © Γεώργιος-Μιχαήλ Παπαγεωργίου, 2025

Με επιφύλαξη παντός δικαιώματος. All rights reserved.

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 δεν πρέπει να ερμηνευθεί ότι αντιπροσωπεύουν τις επίσημες θέσεις του Εθνικού Μετσόβιου Πολυτεχνείου.

Περίληψη

Η εκρηκτική αύξηση των συσκευών IoT και η υιοθέτηση κατανεμημένων υποδομών edge–fog–cloud καθιστούν επιτακτική την ανάγκη για έξυπνη ενορχήστρωση φορτίων. Η παρούσα εργασία εξετάζει την βέλτιστη ανάθεση cloud-native μικροϋπηρεσιών σε ένα τριεπίπεδο συνεχές IoT-edge–fog–cloud, με στόχο τη μείωση του κόστους και της κατανάλωσης ενέργειας, εξασφαλίζοντας παράλληλα εγγυήσεις QoS στους τελικούς χρήστες. Αρχικά, το πρόβλημα διατυπώνεται ως Mixed-Integer Linear Programming (MILP) που ενσωματώνει περιορισμούς υπολογιστικής ισχύος, μνήμης, εύρους ζώνης, καθυστέρησης, αναπαραγωγής και μετανάστευσης. Παρά τη βέλτιστη λύση που παρέχει, ο MILP γίνεται ανεφάρμοστος σε πραγματικό χρόνο λόγω εκθετικής πολυπλοκότητας. Για την υπέρβαση του εμποδίου αυτού, αναπτύσσεται ένας ενορχηστρωτής Βαθιάς Ενισχυτικής Μάθησης (DRL) βασισμένος στον αλγόριθμο Maskable Proximal Policy Optimisation (PPO). Η μάσκα ενεργειών απορρίπτει μη επιτρεπτές ενέργειες, επιταχύνοντας την εκπαίδευση και διασφαλίζοντας την τήρηση όλων των περιορισμών. Ο πράκτορας εκπαιδεύεται με γνώμονα τις λύσεις MILP και αξιολογείται σε δύο τοπολογίες (15 και 45 κόμβων) υπό ένα στατικό και τρία δυναμικά σενάρια κίνησης. Τα πειραματικά αποτελέσματα δείχνουν ότι η πολιτική DRL επιτυγχάνει κόστος πολύ κοντά στο βέλτιστο, κατά κύριο λόγο 100 % αποδοχή αιτημάτων και έως πολύ μικρότερο χρόνο λήψης απόφασης έναντι του επιλυτή MILP, τηρώντας όλους τους περιορισμούς πόρων και καθυστέρησης. Συνεπώς, η προτεινόμενη προσέγγιση επιτρέπει ευέλικτη και ενεργειακά αποδοτική ενορχήστρωση cloud-native εφαρμογών IoT σε μεγάλες, συνεργατικές υποδομές edge.

Λέξεις Κλειδιά: Κατανομή πόρων, Cloud-native, Microservices, Τοπολογία Edge–Fog–Cloud, Βαθιά Ενισχυτική Μάθηση, Proximal Policy Optimisation, MILP, Ενορχήστρωση IoT, Ενεργειακή αποδοτικότητα

Abstract

The explosive growth of IoT devices and the adoption of distributed edge–fog–cloud infrastructures make intelligent workload orchestration imperative. This thesis investigates the optimal placement of cloud-native microservices across a three-tier IoT–edge–fog–cloud continuum, aiming to reduce both deployment cost and energy consumption while simultaneously guaranteeing Quality-of-Service (QoS) for end users. The problem is first expressed as a Mixed-Integer Linear Programming (MILP) model that embeds constraints on compute, memory, bandwidth, latency, replication, and migration. Although MILP yields optimal solutions, its exponential complexity renders it unsuitable for real-time operation. To overcome this barrier, we design a Deep Reinforcement Learning (DRL) orchestrator based on the Maskable Proximal Policy Optimisation (PPO) algorithm. Action masking eliminates invalid moves, accelerating training and ensuring strict constraint satisfaction. The agent is trained with MILP solutions as guidance and evaluated on two topologies (15 and 45 nodes) under one static and three dynamic traffic scenarios. Experimental results show that the DRL policy achieves near-optimal cost, essentially 100 % request acceptance, and markedly lower decision latency compared with the MILP solver, while respecting all resource and latency constraints. Consequently, the proposed approach enables flexible and energy-efficient orchestration of cloud-native IoT applications in large-scale, collaborative edge environments.

Key Words: Resource allocation, Cloud-native, Microservices, Edge–fog–cloud Topology, Deep Reinforcement Learning, Proximal Policy Optimisation, MILP, IoT orchestration, Energy efficiency

Ευχαριστίες

Θα ήθελα να ευχαριστήσω τον καθηγητή κ. Εμμανουήλ Βαρβαρίγο για την υποστήριξη, την εμπιστοσύνη καθώς και την επίβλεψη αυτής της διπλωματικής εργασίας, δίνοντάς μου τη δυνατότητα να εμβαθύνω σε ένα θεματικό πεδίο με αυξανόμενο πρακτικό και ερευνητικό ενδιαφέρον.

Επίσης, ευχαριστώ θερμά τον Διδάκτορα Πολυζώη Σούμπλη για τη συνεχή καθοδήγησή του και την εξαιρετική συνεργασία σε όλα τα στάδια εκπόνησης της εργασίας.

Τέλος, θα ήθελα να ευχαριστήσω τους γονείς μου, την αδερφή μου και τους φίλους μου για τη στήριξή και την έμπνευση που μου έδωσαν καθ' όλη τη διάρκεια της ακαδημαϊκής μου πορείας.

Αθήνα, Ιούλιος 2025

Γεώργιος-Μιχαήλ Παπαγεωργίου

Πίνακας Περιεχομένων

Κεφάλαιο 1: Εισαγωγή

1.1 Γενικά

Η ταχύτατη εξέλιξη της τεχνολογίας και η αυξανόμενη ανάγκη για αποτελεσματικά δίκτυα έχουν δημιουργήσει ένα δυναμικό περιβάλλον που απαιτεί νέες προσεγγίσεις και καινοτομίες. Στην εποχή του Internet-of-Things (IoT) και του Cloud Computing οι εφαρμογές δομούνται με την αρχιτεκτονική Microservices, αναδεικνύοντας την σημασία ευέλικτων και αποτελεσματικών τεχνικών διαχείρισης των δικτύων που τις εξυπηρετούν.

Στην παραδοσιακή μονολιθική αρχιτεκτονική, όλη η λειτουργικότητα μιας εφαρμογής βρίσκεται συγκεντρωμένη σε μία ενιαία δομή. Τα διάφορα τμήματα του λογισμικού είναι έντονα αλληλεξαρτώμενα, με αποτέλεσμα να απαιτείται η ταυτόχρονη παρουσία όλων για τη σωστή λειτουργία ή ενημέρωση της εφαρμογής. Η επεξεργασία και η αποθήκευση δεδομένων πραγματοποιούνται αντίστοιχα ως μοναδικές διαδικασίες σε έναν μόνο εξυπηρετητή ή μία μόνο βάση δεδομένων.

Στη σύγχρονη εποχή, η σταδιακή διάδοση των δικτύων κινητής τηλεφωνίας πέμπτης γενιάς (5G) δημιουργεί το κατάλληλο υπόβαθρο για την ανάπτυξη νέων τεχνολογιών και καινοτόμων εφαρμογών που αξιοποιούν τα νέα χαρακτηριστικά του δικτύου. Προκειμένου οι εφαρμογές να παρέχουν αναβαθμισμένες υπηρεσίες, γίνονται ολοένα και πιο σύνθετες, απαιτώντας συχνές αλλαγές και συνεχή βελτίωση. Επιπλέον, μια διακοπή στη λειτουργία τους μπορεί να προκαλέσει σημαντικές οικονομικές απώλειες.

Μέσα σε αυτό το δυναμικό και ανταγωνιστικό πλαίσιο, το παραδοσιακό μονολιθικό μοντέλο θεωρείται ανεπαρκές, οδηγώντας στην ανάγκη για μια νέα προσέγγιση: το cloud-native μοντέλο. Το cloud-native αξιοποιεί τις εγγενείς δυνατότητες του cloud, προσφέροντας μεταξύ άλλων αυξημένη ανθεκτικότητα και ευκολότερη διαχείριση. Κεντρικό ρόλο στη σχεδίαση cloud-native εφαρμογών διαδραματίζουν οι μικροϋπηρεσίες (microservices), δηλαδή ανεξάρτητες μονάδες λογισμικού, καθεμία με τον δικό της κώδικα και λειτουργικότητα. Η εκτέλεση των μικροϋπηρεσιών πραγματοποιείται σε απομονωμένα υπολογιστικά περιβάλλοντα, τα containers, τα οποία εξασφαλίζουν τους απαιτούμενους πόρους από το σύστημα.

Το ζήτημα της κατανομής πόρων για την υποστήριξη cloud-native εφαρμογών συνδέεται άρρηκτα με την έννοια του ενορχηστρωτή (orchestrator). Ο ενορχηστρωτής είναι ένα λογισμικό το οποίο, μεταξύ άλλων, μέσω ειδικού ενσωματωμένου μηχανισμού, αναλαμβάνει την αυτόματη δέσμευση των απαιτούμενων πόρων για την

εξυπηρέτηση του εκάστοτε φόρτου εργασίας, που στη συγκεκριμένη περίπτωση αφορά μικροϋπηρεσίες (microservices) ενθυλακωμένες σε containers.

Για μικρές τοπολογίες και φόρτους εργασιών, είναι δυνατό να υπολογιστεί η βέλτιστη τοποθέτηση σε κάθε δυνατή κατάσταση με χρήση τεχνικών Integer Linear Programming (ILP). Ωστόσο, καθώς κλιμακώνονται οι δομές που εξετάζουμε, η χρήση ILP με αυτόν τον τρόπο αποδεικνύεται υπερβολικά χρονοβόρα, και συνεπώς μη κατάλληλη για πραγματικές συνθήκες.

Για μια πληρέστερη εικόνα, παρακινείται ο αναγνώστης να εμβαθύνει περισσότερο στις αναφερθείσες έννοιες των κεφαλαίων της θεωρίας, μέσω των παραπομπών.

1.2 Αντικείμενο Διπλωματικής Εργασίας

Η παρούσα διπλωματική εργασία εστιάζει στην αξιοποίηση της τεχνολογίας της βαθιάς ενισχυτικής μάθησης, προκειμένου να ανατεθούν βέλτιστα μικροϋπηρεσίες cloud-native εφαρμογών στα συστήματα των διάφορων στρωμάτων του δικτύου edge-fog-cloud.

Στην προσέγγιση που προτείνεται, γίνεται χρήση βαθιάς ενισχυτικής μάθησης έτσι ώστε να εκπαιδεύσουμε έναν agent-ενορχηστρωτή ο οποίος μαθαίνει μια βέλτιστη πολιτική ανάθεσης, προκειμένου να λαμβάνει τις κρίσιμες αποφάσεις κατανομής στην τοπολογία near edge, far edge. Με αυτόν τον τρόπο μπορεί να γίνει διαχείριση του φόρτου του δικτύου χωρίς της χρήση Integer Linear Programming.

Εστιάζουμε κυρίως στην ελαχιστοποίηση του κόστους της τοποθέτησης των microservices, και συνεπώς της ενεργειακής του κατανάλωσης ενώ ταυτόχρονα εξυπηρετούνται όλοι οι χρήστες του δικτύου.

1.3 Οργάνωση του τόμου

Η παρούσα διπλωματική εργασία ακολουθεί την παρακάτω δομή:

Το πρώτο Κεφάλαιο αποτελεί μια εισαγωγή η οποία στοχεύει στην συνοπτική παρουσίαση του αντικειμένου στο οποίο εστιάζει η παρούσα διπλωματική.

Στο Κεφάλαιο 2 παρουσιάζονται οι τεχνολογίες που είναι άρρηκτα συνδεδεμένες με την κατανόηση και την επίλυση του υπό συζήτηση προβλήματος. Συγκεκριμένα παρουσιάζονται οι τεχνολογίες του Cloud Computing, των Cloud-Native εφαρμογών καθώς και των Machine Learning & Deep Reinforcement Learning.

Στο Κεφάλαιο 3 εξετάζεται η σχετική εργασία πάνω στο αντικείμενο που έχει γίνει στο παρελθόν.

Το κεφάλαιο 4 αποτελεί μια εισαγωγή στο πρακτικό μέρος της εργασίας καθώς εκεί γίνεται η διατύπωση του προβλήματος και παρουσιάζεται η τοπολογία edge-fog-cloud.

Το κεφάλαιο 5 περιέχει την διατύπωση της προτεινόμενης λύσης βαθιάς ενισχυτικής μάθησης του προβλήματος.

Στο Κεφάλαιο 6, αναλύεται η τεχνική υλοποίησης της λύσης, συμπεριλαμβανομένων των βιβλιοθηκών και εργαλείων που χρησιμοποιήθηκαν.

Ύστερα, στο κεφάλαιο 7, παρουσιάζονται τα αποτελέσματα των πειραματικών σεναρίων με τη βοήθεια γραφικών αναπαραστάσεων διαφόρων metrics.

Τέλος, στο Κεφάλαιο 8, εμπεριέχεται μια σύνοψη, συζητούνται τα συμπεράσματα των πειραματικών δοκιμών και προτείνονται μελλοντικές επεκτάσεις.

Μέρος Ι: ΘΕΩΡΗΤΙΚΟ ΜΕΡΟΣ

Κεφάλαιο 2: Θεωρητικό Υπόβαθρο

2.1 Cloud Computing

2.1.1 Ορισμός και βασικές αρχές

Το Cloud Computing (Υπολογιστικό Νέφος) αναφέρεται στην παροχή φιλοξενούμενων υπηρεσιών μέσω του Διαδικτύου [1]. Ο όρος "Cloud" προέρχεται από την τηλεφωνία. Μέχρι τη δεκαετία του 1990, τα κυκλώματα δεδομένων, συμπεριλαμβανομένων αυτών που υποστήριζαν την κίνηση του Διαδικτύου, ήταν φυσικά συνδεδεμένα με καλώδια. Αργότερα, οι πάροχοι τηλεπικοινωνιών μεγάλων αποστάσεων εισήγαγαν τις Υπηρεσίες Εικονικών Ιδιωτικών Δικτύων (VPN) για επικοινωνία δεδομένων, προσφέροντας εγγυημένο εύρος ζώνης με χαμηλότερο κόστος μέσω της δυναμικής μεταγωγής κίνησης για τη βελτιστοποίηση της χρήσης του δικτύου. Αυτή η προσέγγιση καθιστούσε δύσκολο τον ακριβή προσδιορισμό των διαδρομών που θα ακολουθούσε η κίνηση δεδομένων. Ο όρος "telecom cloud" χρησιμοποιήθηκε για να περιγράψει αυτό το μοντέλο δικτύωσης, και το Cloud Computing βασίζεται σε παρόμοια φιλοσοφία [2].

Υπάρχουν αρκετοί ορισμοί για το Cloud Computing, ο πιο ευρέως αποδεκτός είναι αυτός που δίνεται από το NIST (National Institute of Standards and Technology):

"Το υπολογιστικό νέφος είναι ένα μοντέλο που επιτρέπει την πανταχού παρούσα, βολική, κατά παραγγελία δικτυακή πρόσβαση σε μια κοινόχρηστη δεξαμενή διαμορφώσιμων υπολογιστικών πόρων (π.χ. δίκτυα, διακομιστές, αποθήκευση, εφαρμογές και υπηρεσίες), οι οποίοι μπορούν να παρέχονται και να απελευθερώνονται γρήγορα με ελάχιστη προσπάθεια διαχείρισης ή αλληλεπίδραση με τον πάροχο υπηρεσιών. Αυτό το μοντέλο νέφους αποτελείται από πέντε βασικά χαρακτηριστικά, τρία μοντέλα υπηρεσιών και τέσσερα μοντέλα ανάπτυξης" [3].

Οι βασικές αρχές του Cloud Computing είναι οι εξής:

1. **Αυτοεξυπηρέτηση κατά Απαίτηση (On-Demand Self-Service):** Οι χρήστες μπορούν αυτόνομα να διαμορφώνουν υπολογιστικούς πόρους (π.χ., χρόνο διακομιστή, αποθήκευση δεδομένων, εφαρμογές) ανάλογα με τις ανάγκες τους, χωρίς να απαιτείται άμεση αλληλεπίδραση με τον πάροχο υπηρεσιών.
2. **Ευρεία Πρόσβαση στο Δίκτυο (Broad Network Access):** Οι υπηρεσίες cloud είναι προσβάσιμες μέσω του δικτύου, με χρήση τυποποιημένων μηχανισμών,

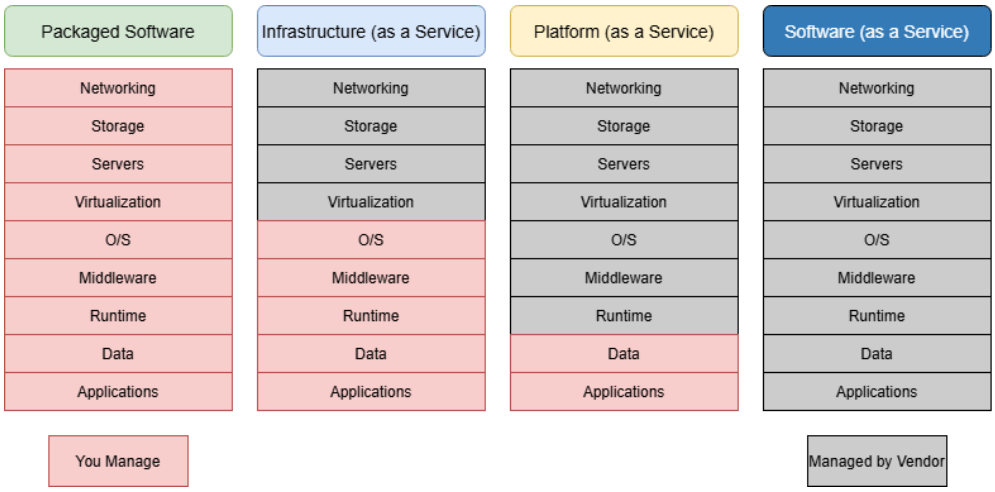
από μια ποικιλία συσκευών (π.χ., κινητά τηλέφωνα, φορητούς υπολογιστές, tablets).

3. **Συνένωση Πόρων (Resource Pooling):** Οι πάροχοι συγκεντρώνουν τους πόρους τους για να εξυπηρετήσουν πολλαπλούς χρήστες μέσω μοντέλου πολλαπλών ενοικιαστών. Οι πόροι κατανέμονται και επανακατανέμονται δυναμικά με βάση τη ζήτηση, εξασφαλίζοντας ευελιξία και ανεξαρτησία από την τοποθεσία.
4. **Γρήγορη Ελαστικότητα και Κλιμάκωση (Rapid Elasticity and Scalability):** Οι πόροι μπορούν να αυξηθούν ή να μειωθούν γρήγορα, αυτόματα ή χειροκίνητα, για να καλύψουν τις μεταβαλλόμενες ανάγκες. Στους χρήστες, οι διαθέσιμοι πόροι συχνά φαίνονται πρακτικά απεριόριστοι.
5. **Μετρήσιμη Υπηρεσία (Measured Service):** Η χρήση των πόρων παρακολουθείται, μετριέται και τιμολογείται αυτόματα βάσει της πραγματικής χρήσης, προσφέροντας διαφάνεια και έλεγχο των εξόδων [1][3][4].

2.1.2 Τα τρία κύρια Μοντέλα Υπηρεσιών (Infrastructure-as-a-Service, Platform-as-a-Service, Software-as-a-Service).

Το Cloud περιβάλλον συνήθως αποτελείται από τρία κύρια στοιχεία, που αντιπροσωπεύουν διαφορετικούς τύπους υπηρεσιών: Υπηρεσίες Λογισμικού, Υπηρεσίες Πλατφόρμας και Υπηρεσίες Υποδομής. Όταν το περιβάλλον αυτό απεικονίζεται ως πυραμίδα, οι Υπηρεσίες Λογισμικού βρίσκονται στην κορυφή, ενώ οι Υπηρεσίες Υποδομής αποτελούν τη βάση. Καθώς ανεβαίνουμε προς τις Υπηρεσίες Λογισμικού, αυξάνεται το επίπεδο αφαίρεσης, ενώ καθώς κατεβαίνουμε προς τις Υπηρεσίες Υποδομής, αυξάνεται το επίπεδο ελέγχου. Με βάση αυτή τη δομή, οι υπηρεσίες Cloud κατηγοριοποιούνται ως εξής [1]:

- Λογισμικό ως Υπηρεσία (Software as a Service - SaaS)
- Πλατφόρμα ως Υπηρεσία (Platform as a Service - PaaS)
- Υποδομή ως Υπηρεσία (Infrastructure as a Service - IaaS)



Εικόνα 1: Cloud Computing Stack: Architectural difference in three cloud computing architectures: IaaS PaaS SaaS [5]

Στη συνέχεια θα παρουσιαστούν εν συντομία οι κατηγορίες αυτές:

Το **Λογισμικό ως Υπηρεσία (Software as a Service - SaaS)** είναι ένα μοντέλο παροχής λογισμικού, στο οποίο ο καταναλωτής έχει τη δυνατότητα να αξιοποιεί εφαρμογές που λειτουργούν πάνω σε υποδομές υπολογιστικού νέφους [3]. Η προσέγγιση αυτή καθιστά μη αναγκαία την τοπική εγκατάσταση του λογισμικού, καθώς και την διαχείρισή του στη συσκευή, που είναι σημαντικά μειονεκτήματα του παραδοσιακού μοντέλου διαμοίρασης λογισμικού. Μειώνοντας τις απαιτήσεις hardware και λογισμικού, που επιφέρουν υψηλό αρχικό κόστος, το SaaS λειτουργεί με διάφορα μοντέλα πληρωμής όπως συνδρομή, τιμολόγηση βάσει συναλλαγών, κατανομή κερδών, κατανομή ιδιοκτησίας και έσοδα βάσει διαφημίσεων (π.χ. pay per click), ο πελάτης έχει την επιλογή να πληρώσει για όσο χρειάζεται και όταν το χρειάζεται ανάλογα με τις ανάγκες του [6][7].

Συνοψίζοντας, το SaaS προσφέρει σημαντικά πλεονεκτήματα όπως μείωση κόστους, ευκολία χρήσης και ταχεία υλοποίηση, καθιστώντας το ιδιαίτερα ελκυστικό για επιχειρήσεις που επιδιώκουν ευελιξία και αποδοτικότητα. Παραδείγματα SaaS αποτελούν οι εφαρμογές του Google Workspace (Gmail, Google Docs, Google Drive), το Dropbox καθώς και το Netflix [8].

Η **Πλατφόρμα ως Υπηρεσία (Platform as a Service - PaaS)** αντί για έτοιμο λογισμικό παρέχει στον πελάτη ένα πλήρες περιβάλλον ανάπτυξης και εκτέλεσης, χρησιμοποιώντας εργαλεία όπως APIs και προγραμματιστικές γλώσσες, με απώτερο σκοπό την δημιουργία λογισμικού. Ο χρήστης του PaaS δεν έχει την ευθύνη διαχείρισης ή εποπτείας της υποκείμενης υποδομής του Cloud (όπως servers, storage, Operating systems), αλλά διαθέτει τη δυνατότητα να διαχειρίζεται τις αναπτυγμένες εφαρμογές

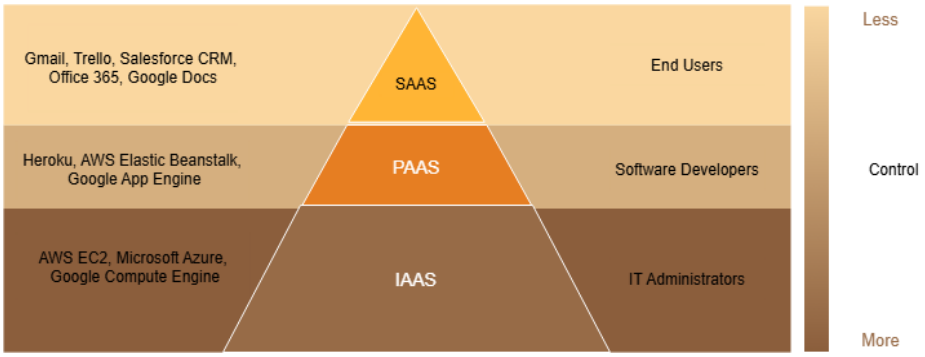
και, σε ορισμένες περιπτώσεις, να ρυθμίζει το περιβάλλον φιλοξενίας αυτών των εφαρμογών. Έτσι δεν είναι αναγκαία η εγκατάσταση και διαχείριση των διαφόρων περιβαλλόντων που είναι αναγκαία για τα λοιπά στάδια ανάπτυξης εφαρμογών από τον χρήστη, ο οποίος μπορεί πλέον να αφοσιωθεί σε πιο παραγωγικές ενέργειες [9]. Όταν το λογισμικό είναι έτοιμο για να χρησιμοποιηθεί, μπορεί πλέον να διαμοιραστεί στους λοιπούς χρήστες μέσω του internet [10]. Επιπροσθέτως, καθώς ο προγραμματισμός, debugging και testing του λογισμικού γίνεται στο ίδιο περιβάλλον, εξαλείφει προβλήματα που δημιουργούνται όταν τα προγράμματα αναπτύσσονται σε ένα περιβάλλον αλλά εκτελούνται σε διαφορετικό [11].

Συνοψίζοντας, η πλατφόρμα ως υπηρεσία επιταχύνει και απλοποιεί την ανάπτυξη λογισμικού ενώ ταυτόχρονα μειώνει το ρίσκο. Σύγχρονα παραδείγματα είναι το Microsoft Azure αλλά και το Google App Engine [12].

Η **Υποδομή ως Υπηρεσία (Infrastructure as a Service - IaaS)** αναφέρεται στην παροχή διακομιστών, αποθήκευσης και τεχνολογιών virtualization για την προσφορά υπηρεσιών με τρόπο παρόμοιο με κοινές υπηρεσίες κοινής ωφέλειας. Η υποδομή περιλαμβάνει εγκαταστάσεις, δίκτυα επικοινωνίας, φυσικούς κόμβους υπολογιστών και μια δεξαμενή virtualized πόρων που διαχειρίζεται ο πάροχος υπηρεσιών. Ως υπηρεσία περιλαμβάνει στοιχεία που βρίσκονται υπό τον έλεγχο του χρήστη, όπως εικονικές μηχανές, τα λειτουργικά τους συστήματα, την αποθήκευση και τη διαχείριση αυτών των πόρων [7]. Μπορεί να θεωρηθεί ως το πιο βασικό επίπεδο του Cloud Computing, προσφέροντας τα βασικά στοιχεία όπως επεξεργαστική ισχύ, αποθηκευτικό χώρο και άλλους πόρους χαμηλού επιπέδου υλικού και δικτύου που παρέχονται εικονικά και κατ' απαίτηση μέσω του διαδικτύου [9]. Επίσης, παρακολουθεί τη χρήση των πόρων με την πάροδο του χρόνου, επιτρέποντας τη χρέωση βάσει συμφωνημένων τιμολογίων. Αυτό το μοντέλο απαλλάσσει τους χρήστες από τη διαχείριση της φυσικής και εικονικής υποδομής, διατηρώντας ταυτόχρονα τον έλεγχο πάνω στο λειτουργικό σύστημα, τις ρυθμίσεις και το λογισμικό που εκτελείται στις εικονικές μηχανές [7].

Όπως είναι προφανές, η Υποδομή ως Υπηρεσία έχει το πλεονέκτημα ότι μειώνει το κόστος εκτέλεσης εφαρμογών ή αποθήκευσης δεδομένων, αφού δε χρειάζεται ο πελάτης να επενδύσει σε φυσικές εγκαταστάσεις και ειδικούς IT για να τις διαχειρίζονται. Ταυτόχρονα, λόγω του μοντέλου πληρωμών της ελαστικότητας και της κλιμάκωσης, δεν υπάρχει το ρίσκο μεγαλύτερης ή μικρότερης επένδυσης από ότι είναι αναγκαίο [13]. Παραδείγματα IaaS αποτελούν το Google Cloud, Amazon Web Services (AWS) και Microsoft Azure [14].

Οι τρείς κύριες αυτές κατηγορίες Cloud Computing είναι ξεκάθαρα σχεδιασμένες για διαφορετικούς σκοπούς, εκτελώντας διαφορετικούς ρόλους, και θα προτιμηθούν από διαφορετικούς χρήστες ανάλογα με τις ανάγκες τους.



Εικόνα 2 : Πυραμίδα Νέφους (Cloud Pyramid): IaaS, PaaS and SaaS [15]

2.1.3 Τα τέσσερα μοντέλα ανάπτυξης του Νέφους

Τα μοντέλα ανάπτυξης καθορίζουν τον τρόπο με τον οποίο το μοντέλο υπηρεσίας υλοποιείται και ξεχωρίζουν το κοινό στο οποίο θα είναι διαθέσιμες οι υπηρεσίες νέφους. Ακολουθεί σύντομη περιγραφή των τεσσάρων αυτών μοντέλων:

- **Private Cloud**

  Οι υπηρεσίες ιδιωτικού υπολογιστικού νέφους προορίζονται αποκλειστικά για μία οργανωτική μονάδα. Αυτές οι υπηρεσίες μπορούν να υποστηρίξουν διαφορετικές επιχειρηματικές μονάδες εντός της ίδιας οργάνωσης, με λογικό διαχωρισμό που διασφαλίζει ότι οι λειτουργίες της μίας μονάδας δεν επηρεάζουν την άλλη. Η διαχείριση και η ιδιοκτησία της υποδομής του cloud μπορεί να πραγματοποιείται είτε από την ίδια την οργάνωση είτε από τρίτο πάροχο. Το ιδιωτικό cloud υποστηρίζει όλα τα μοντέλα υπηρεσιών, όπως SaaS, PaaS και IaaS, και είναι προσβάσιμο μόνο από έμπιστους χρήστες. Χαρακτηρίζεται από αυξημένη ασφάλεια των δεδομένων των πελατών και μεγαλύτερο έλεγχο της υποδομής του νέφους.

- **Community Cloud**

  Η υποδομή νέφους σχεδιάζεται και παρέχεται για συγκεκριμένες οργανώσεις που μοιράζονται κοινά ενδιαφέροντα ή ανάγκες, όπως πρότυπα ασφαλείας δεδομένων, τεχνολογία, επιχειρησιακές λειτουργίες ή απαιτήσεις

συμμόρφωσης. Η διαχείριση και η ιδιοκτησία της υποδομής μπορεί να ανήκει είτε στις οργανώσεις της κοινότητας είτε σε τρίτους φορείς, ενώ η φυσική τοποθεσία της μπορεί να βρίσκεται είτε στις εγκαταστάσεις των εμπλεκομένων οργανισμών είτε σε εκείνες του τρίτου φορέα.

- **Public cloud**

  Στο μοντέλο αυτό, η υποδομή νέφους είναι διαθέσιμη στο ευρύ κοινό και λειτουργεί με ένα μοντέλο πολλαπλών ενοικιαστών (multi-tenancy), όπου κάθε χρήστης λαμβάνει το δικό του διακριτό εικονικό περιβάλλον. Οι χρήστες του δημόσιου cloud μοιράζονται την ίδια υποδομή, η οποία ανήκει, λειτουργεί και συντηρείται από έναν εξωτερικό πάροχο υπηρεσιών. Η φυσική τοποθεσία των διακομιστών και των κέντρων δεδομένων παραμένει εντελώς αόρατη για τους χρήστες. Η χρέωση βασίζεται στις υπηρεσίες που καταναλώνονται και στη χρονική διάρκεια χρήσης. Διάφορα μοντέλα τιμολόγησης εφαρμόζονται για τον υπολογισμό του κόστους για τους πελάτες. Αυτό το μοντέλο υλοποίησης παρέχει σημαντική εξοικονόμηση κόστους, καθώς τα έξοδα κατανέμονται μεταξύ πολλών χρηστών.

- **Hybrid Cloud**

  Αυτή η μορφή υλοποίησης συνδυάζει δύο ή περισσότερες διαφορετικές υποδομές υπολογιστικού νέφους (δημόσιες, ιδιωτικές και κοινοτικές). Παρά τον συνδυασμό τους, οι υποδομές αυτές διατηρούν την ανεξάρτητη λειτουργία τους, ενώ συνεργάζονται για να διασφαλίσουν τη φορητότητα εφαρμογών και δεδομένων [16].

2.1.4 Πλεονεκτήματα και προκλήσεις

Υπάρχει πληθώρα λόγων για τους οποίους το Cloud Computing υιοθετείται από συνεχώς περισσότερες επιχειρήσεις και χρήστες, μεταξύ άλλων:

1. Μείωση των δαπανών που σχετίζονται με την παροχή IT υπηρεσιών, επιτρέποντας την αξιοποίηση πόρων για άλλες δραστηριότητες, όπως η ενσωμάτωση υπηρεσιών.

2. Ελάφρυνση των ευθυνών διαχείρισης, δίνοντας τη δυνατότητα στο βασικό προσωπικό της επιχείρησης να επικεντρωθεί στην παραγωγή και την καινοτομία.
3. Απλούστευση και επιτάχυνση των λειτουργιών, προσφέροντας ευκολία χρήσης και ευελιξία, επιτρέποντας στις επιχειρήσεις να διαθέτουν επιπλέον πόρους γρήγορα και χωρίς ιδιαίτερη προσπάθεια. Απλούστευση και επιτάχυνση των λειτουργιών, προσφέροντας ευκολία χρήσης και ευελιξία, επιτρέποντας στις επιχειρήσεις να διαθέτουν επιπλέον πόρους γρήγορα και χωρίς ιδιαίτερη προσπάθεια.
4. Παροχή πρόσβασης σε μικρές επιχειρήσεις σε υπηρεσίες και πόρους IT που διαφορετικά θα ήταν ανέφικτοι, εξισορροπώντας τον ανταγωνισμό μεταξύ μεγάλων και μικρότερων επιχειρήσεων.
5. Παροχή καινοτόμων και προηγμένων αρχιτεκτονικών υπολογιστών, ενισχύοντας τις δυνατότητες ανάπτυξης και νέων λύσεων.
6. Διασφάλιση συνέχειας λειτουργίας και αποκατάστασης μετά από καταστροφές μέσω μιας ολοκληρωμένης γκάμας εξωτερικά παρεχόμενων υπηρεσιών και πόρων ICT.
7. Ενίσχυση της επιχειρηματικής ευελιξίας και επεκτασιμότητας, επιτρέποντας στις επιχειρήσεις να ανταποκρίνονται αποτελεσματικά στις ταχέως μεταβαλλόμενες συνθήκες [1].

Η τεχνολογία Cloud Computing δεν είναι όμως δίχως προκλήσεις και ζητήματα σε κατηγορίες όπως: αξιοπιστία και διαθεσιμότητα της υποδομής και του συστήματος, διαχείριση και διακυβέρνηση δεδομένων, διαχείριση υπηρεσιών, έλεγχος και παρακολούθηση προϊόντων και διεργασιών [1].

Συγκεκριμένα, λίγα από τα ζητήματα που μπορεί να παρουσιαστούν:

- Vendor Lock-In - Αδυναμία μεταφοράς δεδομένων και/ή προγράμματος από ένα πάροχο νέφους σε κάποιον άλλο, λόγω ασυμβίβαστων τεχνολογιών μεταξύ τους.
- Ελλιπής Κρυπτογράφηση και denial of service (DOS) επιθέσεις – θεωρούνται σημαντικοί κίνδυνοι ασφάλειας σε περιβάλλοντα νέφους και μπορεί να οδηγήσουν σε data loss ή data leakage.

- Φυσικές καταστροφές μπορεί να επηρεάσουν την υποδομή ενός πάροχου, με αποτέλεσμα οι πελάτες να επηρεαστούν από φυσικές καταστροφές που λαμβάνουν χώρα μακριά από τους ίδιους.
- Το cloud computing υπάγεται σε διάφορους νομικούς περιορισμούς, που γίνονται πιο περίπλοκοι αν λάβουμε υπόψη ότι πολλοί πάροχοι υπολογιστικού νέφους είναι πολυεθνικοί [17].

2.2 Cloud-Native Applications:

Ο όρος "Cloud" εμπεριέχεται τόσο στο "Cloud-Computing" αλλά και στο "Cloud-Native". Το "Cloud" είναι το *πού* γίνεται ο υπολογισμός ενώ το "Cloud-Native" είναι τεχνική για την αξιοποίηση των δυνατοτήτων του cloud.

2.2.1 Ορισμός και Χαρακτηριστικά

Η Cloud-Native αρχιτεκτονική είναι μια προσέγγιση διαφορετική από την κλασσική μονολιθική αρχιτεκτονική, στην οποία οι συχνές αλλαγές δυσχεραίνουν το χρήστη, οι νέες τεχνολογίες είναι περίπλοκο ή και αδύνατο να υλοποιηθούν [18].

Στη σύγχρονη εποχή, οι άνθρωποι χρησιμοποιούν εφαρμογές στην καθημερινότητά τους για βασικές τους ενέργειες, πράγμα που δημιουργεί νέες προσδοκίες για τα αναγκαία χαρακτηριστικά μιας εφαρμογής. Ο χρήστης περιμένει πλέον οι εφαρμογές που χρησιμοποιεί να είναι συνεχώς διαθέσιμες, να προσφέρουν ομαλή εμπειρία σε πολλαπλές συσκευές και πλατφόρμες, και ταυτόχρονα να ενημερώνονται συχνά με καινούργια χαρακτηριστικά που ανταποκρίνονται στις εξελισσόμενες ανάγκες του. Η κινητικότητα και η συνεχής σύνδεση των χρηστών δημιουργούν την απαίτηση για λογισμικό ικανό να ανταποκρίνεται σε αυξημένα και μεταβαλλόμενα φορτία αιτήσεων. Επιπλέον, οι διασυνδεδεμένες συσκευές ("things") σχηματίζουν ένα κατανεμημένο δίκτυο δεδομένων πρωτοφανούς μεγέθους, το οποίο απαιτεί νέες προσεγγίσεις αποθήκευσης και επεξεργασίας. Σε συνδυασμό με τη διαθεσιμότητα σύγχρονων πλατφόρμων για την εκτέλεση του λογισμικού, οι ανάγκες αυτές οδήγησαν άμεσα στην ανάδυση του cloud-native λογισμικού [19].

Συνοψίζοντας, το cloud-native λογισμικό πρέπει να είναι κατανεμημένο, να λειτουργεί συνεχώς σε ένα διαρκώς μεταβαλλόμενο περιβάλλον και να είναι το ίδιο διαρκώς μεταβαλλόμενο. Για να ανταπεξέλθει στις ανάγκες αυτές, είναι αναγκαίο το λογισμικό να είναι: Containerized, Dynamically orchestrated και microservices oriented [20].

- Containerized: Κάθε στοιχείο της εφαρμογής, όπως οι εφαρμογές, οι διεργασίες και οι βιβλιοθήκες, ενσωματώνεται σε ξεχωριστό container. Αυτή η πρακτική ενισχύει την αναπαραγωσιμότητα, διασφαλίζει τη διαφάνεια και επιτυγχάνει αποτελεσματική απομόνωση πόρων.
- Δυναμικά ενορχηστρωμένο (Dynamically orchestrated): Τα Containers προγραμματίζονται και ελέγχονται συνεχώς, εξασφαλίζοντας τη μέγιστη αποδοτικότητα στη χρήση των πόρων.
- Με Προσανατολισμό στα Microservices (Microservices Oriented): Οι εφαρμογές κατακερματίζονται σε μικροϋπηρεσίες, γεγονός που βελτιώνει σημαντικά την ευκολία διαχείρισης και την προσαρμοστικότητά τους [20].

2.2.2 Βασικές Τεχνολογίες και Εργαλεία

Παρακάτω παρουσιάζονται βασικές τεχνολογίες και εργαλεία για την ανάπτυξη και τη χρήση των Cloud Native εφαρμογών.

2.2.2.1 Εικονικοποίηση (Virtualization)

Για να κατανοήσουμε καλύτερα τις cloud-native εφαρμογές, καθώς και τις έννοιες του Containerization και της δυναμικής ενορχήστρωσης, είναι απαραίτητο να αναλύσουμε την έννοια της εικονικοποίησης.

Το NIST ορίζει την εικονικοποίηση ως εξής:

"Εικονικοποίηση είναι η προσομοίωση του λογισμικού ή/και του υλικού πάνω στο οποίο εκτελείται άλλο λογισμικό. Αυτό το προσομοιωμένο περιβάλλον ονομάζεται εικονική μηχανή" [21].

Η εικονικοποίηση ορίζεται αλλού ως μια τεχνολογία που εισάγει ένα λογισμικό επίπεδο αφαίρεσης μεταξύ του υλικού (Hardware) και του λειτουργικού συστήματος(Operating System - OS), καθώς και των εφαρμογών που εκτελούνται πάνω σε αυτό. Το επίπεδο αυτό, γνωστό ως διαχειριστής εικονικών μηχανών (virtual machine monitor - VMM ή hypervisor), αποκρύπτει τους φυσικούς πόρους του συστήματος από το λειτουργικό σύστημα. Εφόσον ο VMM ελέγχει απευθείας το υλικό αντί του λειτουργικού συστήματος, επιτρέπεται η ταυτόχρονη εκτέλεση πολλαπλών, ενδεχομένως διαφορετικών, λειτουργικών συστημάτων στην ίδια φυσική υποδομή. Ως αποτέλεσμα, η φυσική πλατφόρμα διαχωρίζεται σε διακριτές λογικές μονάδες, που ονομάζονται εικονικές μηχανές (VMs).

Ο όρος "εικονικότητα" διαφοροποιείται από την "πραγματικότητα" μόνο σε θεωρητικό επίπεδο, διατηρώντας ωστόσο την ίδια ουσία και λειτουργικότητα. Στον τομέα της πληροφορικής, ένα εικονικό περιβάλλον γίνεται αντιληπτό από τις εφαρμογές και τα υπόλοιπα συστήματα με τον ίδιο τρόπο όπως ένα φυσικό, παρά τις διαφορές στη θεμελιώδη υλοποίησή του [22].

Το cloud computing βασίζεται σε μεγάλο βαθμό στην τεχνολογία της εικονικοποίησης. Ωστόσο, είναι σημαντικό να την αντιλαμβανόμαστε ως μια τεχνολογία που επιτρέπει τη λειτουργία του cloud computing και όχι ως συνώνυμη έννοια, ιδιαίτερα στο μοντέλο Infrastructure as a Service (IaaS). Η εικονικοποίηση συμβάλλει καθοριστικά στην κλιμάκωση, την ελαστικότητα, τη διαχείριση πόρων και την ανάπτυξη λύσεων. Ακόμα κι αν η υποδομή ήταν πλήρως εικονικοποιημένη, θα υπήρχαν επιπλέον ζητήματα προς επίλυση, όπως η έλλειψη εργαλείων αυτοματοποίησης και διαχείρισης εφαρμογών σε εικονικές μηχανές. Συνεπώς, η εικονικοποίηση πρέπει να θεωρείται βασική τεχνολογία που επιτρέπει τη λειτουργία του cloud computing, αλλά όχι ταυτόσημη με αυτό.

Με τη χρήση του virtualization, επιτρέπεται σε πολλούς χρήστες να μοιράζονται φυσικές τεχνολογικές υποδομές. Οι ειδικοί πληροφορικής αποδίδουν λογικές ονομασίες στους φυσικούς πόρους και δημιουργούν δείκτες που επιτρέπουν την πρόσβαση σε αυτούς όποτε απαιτείται. Κατά συνέπεια, η εικονικοποίηση δημιουργεί μια εικονική ή λογική εκδοχή ενός πόρου, όπως μια μονάδα αποθήκευσης, ένας διακομιστής, ένας επιτραπέζιος υπολογιστής, ένα λειτουργικό σύστημα ή ένα δίκτυο, επιτρέποντας στους χρήστες να τα αξιοποιούν χωρίς πρόσθετη προσπάθεια [23].

Αξίζει να σημειωθεί η σημασία της εικονικοποίησης σε τρείς βασικούς τομείς: απομόνωση εργασιών, συγκέντρωση φορτίου εργασίας και μετανάστευση εργασιών.

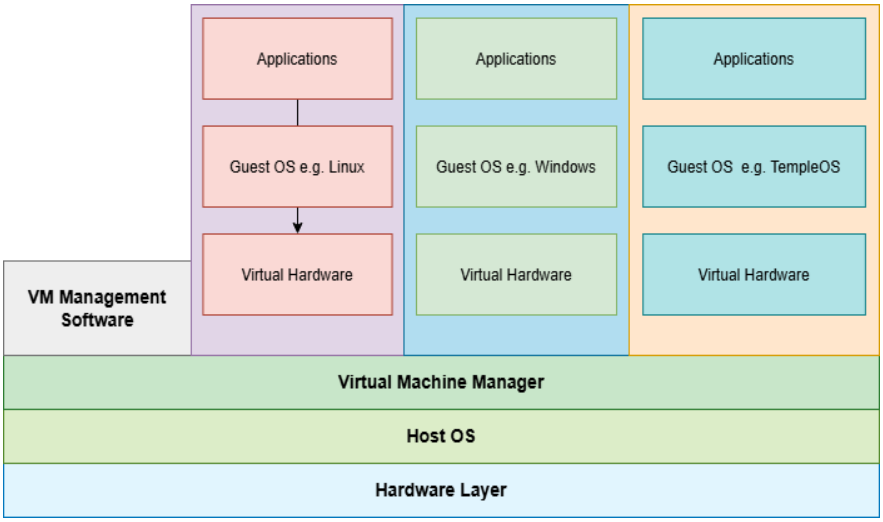
Η απομόνωση εργασιών (workload isolation) εξασφαλίζει ότι κάθε εικονική μηχανή (VM) λειτουργεί ανεξάρτητα από τις άλλες, περιορίζοντας τις επιπτώσεις αποτυχιών ή επιθέσεων σε ένα απομονωμένο περιβάλλον, ενισχύοντας την ασφάλεια και την αξιοπιστία των συστημάτων.

Η συγκέντρωση φορτίου εργασίας (workload consolidation) επιτρέπει την εκτέλεση πολλαπλών εφαρμογών σε λιγότερους φυσικούς διακομιστές, ελαττώνοντας τη χρήση πόρων και μειώνοντας το κόστος συντήρησης και λειτουργίας των υποδομών.

Η μετανάστευση φορτίου εργασίας (workload migration) διευκολύνει τη δυναμική μεταφορά εικονικών μηχανών μεταξύ διακομιστών, εξασφαλίζοντας αδιάλειπτη λειτουργία, ευελιξία στη διαχείριση των υποδομών και βελτιωμένη κατανομή πόρων [24].

Η εικονικοποίηση επιτρέπει την αφαίρεση και απομόνωση των χαμηλότερων επιπέδων λειτουργικότητας καθώς και του υποκείμενου υλικού. Αυτό παρέχει τη δυνατότητα φορητότητα των ανώτερων λειτουργιών, ενώ ταυτόχρονα διευκολύνει τον διαμοιρασμό και τη συγκέντρωση των φυσικών πόρων. Οι διαφορετικές προσεγγίσεις της εικονικοποίησης ταξινομούνται στις εξής κατηγορίες:

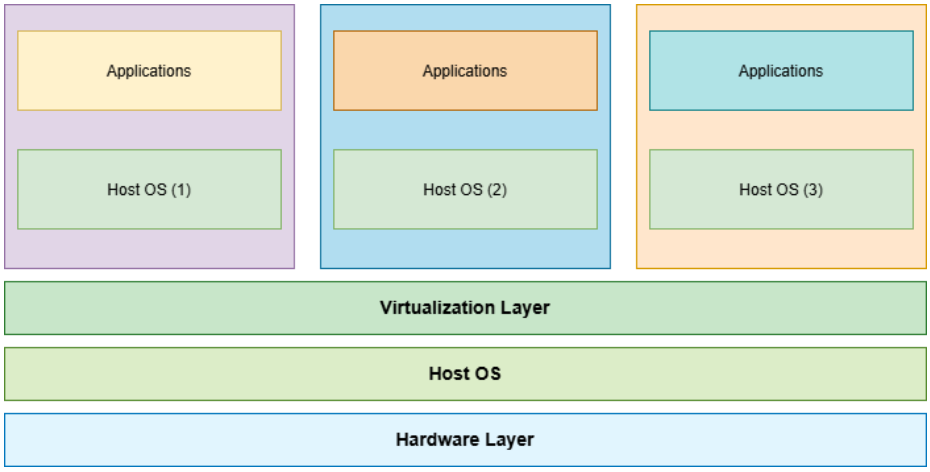
- **Πλήρης εικονικοποίηση (Full Virtualization)**: Σε αυτή την προσέγγιση, το λειτουργικό σύστημα-πελάτης (guest OS) εκτελείται χωρίς τροποποίηση, είτε με τον hypervisor να εκτελείται ως εφαρμογή στο λειτουργικό σύστημα-οικοδεσπότη (host OS), είτε απευθείας στο φυσικό υλικό (bare-metal). Το βασικό πλεονέκτημα είναι η ευκολία υλοποίησης και συμβατότητας, ενώ κύριο μειονέκτημα είναι η μειωμένη απόδοση (έως και 30%) συγκριτικά με ένα φυσικό περιβάλλον.



Εικόνα 3: Full Virtualization Architecture

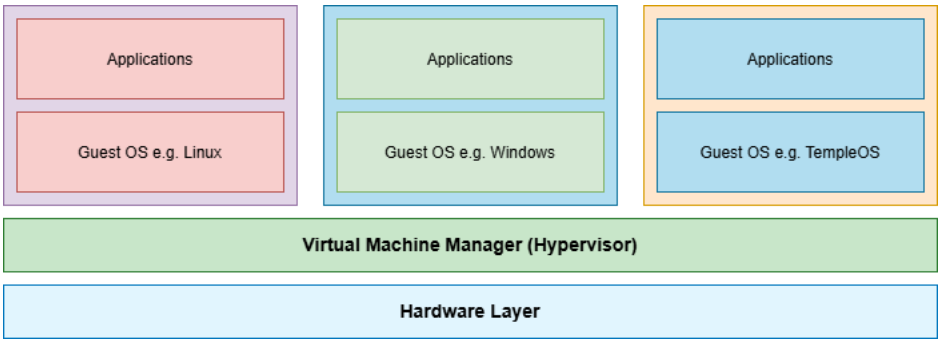
- **Εικονικοποίηση στο επίπεδο λειτουργικού συστήματος (OS-Layer ή Container-based Virtualization)**: Η εικονικοποίηση αυτή τρέχει περισσότερα instances του ίδιου λειτουργικού παράλληλα, με κάθε VM να χρησιμοποιεί την ίδια εικόνα OS. Ο hypervisor αποτελεί ένα επίπεδο λογισμικού που εκτελείται πάνω από το OS και χρησιμοποιεί τον ίδιο πυρήνα (kernel) για όλες τις VM. Οι διαχειριστές του συστήματος μπορούν να κατανέμουν τους πόρους κατά τη δημιουργία ή την εκτέλεση των Virtual machines. Είναι πιο αποδοτική σε σχέση με την πλήρη εικονικοποίηση, αλλά περιορίζεται στο να εκτελεί containers που βασίζονται

στο ίδιο OS με το υποκείμενο σύστημα (Δε μπορεί να τρέξει Windows, για παράδειγμα, πάνω σε Linux).



Εικόνα 4: OS Layer Virtualization Architecture

- **Εικονικοποίηση σε επίπεδο υλικού (Hardware-Layer Virtualization)**: Σε αυτήν την προσέγγιση, ο hypervisor τρέχει απευθείας πάνω στο φυσικό υλικό, ελέγχοντας και συγχρονίζοντας την πρόσβαση του guest OS σε αυτό. Προσφέροντας υψηλό επίπεδο απομόνωσης και απόδοσης, είναι μία προσέγγιση που συχνά χρησιμοποιείται στην αγορά.



Εικόνα 5 : Hardware-Layer Virtualization Architecture

- **Παραεικονικοποίηση (Paravirtualization)**: Σε αυτή τη μέθοδο, Ο hypervisor λειτουργεί επίσης πάνω στο hardware, αλλά απαιτεί το guest OS να είναι τροποποιημένο για να επικοινωνεί άμεσα μαζί του. Ένα από τα κύρια χαρακτηριστικά της παραεικονικοποίησης είναι ότι ο VMM είναι σχετικά απλός, γεγονός που επιτρέπει στην παραεικονικοποίηση να πετυχαίνει επιδόσεις πολύ κοντά σε αυτές του μη-εικονικοποιημένου υλικού. Οι εικονικές μηχανές γνωρίζουν ότι εκτελούνται σε ένα εικονικό περιβάλλον ενώ η

παραεικονικοποίηση επιτυγχάνει απόδοση που πλησιάζει την απόδοση μη-εικονικοποιημένου υλικού.

Ουσιαστικά, η παραεικονικοποίηση παρέχει υψηλή απόδοση, θυσιάζοντας εν μέρει τη συμβατότητα, καθώς απαιτεί συγκεκριμένες τροποποιήσεις στα guest OS, κάτι που δεν συμβαίνει στην πλήρη εικονικοποίηση.

- **Εικονικοποίηση εφαρμογών (Application Virtualization)**: Οι virtualized εφαρμογές αυτές έχουν σχεδιαστεί για να εκτελούνται σε ένα μικρό απομονωμένο περιβάλλον που περιέχει μόνο τους πόρους που απαιτούνται για την εκτέλεσή τους, χωρίς να απαιτείται πλήρης εγκατάσταση στο λειτουργικό σύστημα του χρήστη. Δεν υφίσταται ένας παραδοσιακός hypervisor, τα μικρά αυτά απομονωμένα εικονικά περιβάλλοντα δρουν ως ένα στρώμα μεταξύ των εφαρμογών και του host OS.
- **Εικονικοποίηση πόρων (Resource Virtualization)**: Περιλαμβάνει την ομαδοποίηση ή τον διαμοιρασμό φυσικών πόρων όπως δίσκοι, χώροι αποθήκευσης και πόροι δικτύου οι οποίοι προσφέρονται στη συνέχεια σε virtualized μορφή.
- **Εικονικοποίηση αποθηκευτικού χώρου (Storage Virtualization)**: Είναι μία εξειδικευμένη μορφή εικονικοποίησης πόρων, όπου αποθηκευτικοί πόροι σε ένα δίκτυο συνδυάζονται σε μία λογική μονάδα που φαίνεται σαν μία ενιαία αποθηκευτική συσκευή στον χρήστη [22].

Η εικονικοποίηση, πέρα από τη βελτίωση της απόδοσης των συστημάτων, αποτελεί κρίσιμο εργαλείο για την αποτελεσματική διαχείριση υπολογιστικών υποδομών, τόσο σε cloud όσο και σε παραδοσιακά περιβάλλοντα [24].

2.2.2.2 Containers και Container Orchestration

2.2.2.2.1 Containers

Οι εικονικές μηχανές (VMs) έχουν αποτελέσει τη βασική τεχνολογία στο επίπεδο της υποδομής, επιτρέποντας την εικονικοποίηση λειτουργικών συστημάτων. Τα κοντέινερ (containers) βασίζονται σε μια παρόμοια αρχή, αλλά αποτελούν μια πιο ελαφριά μορφή virtualization, καθώς απαιτούν λιγότερους υπολογιστικούς πόρους και μειώνουν το χρόνο εκτέλεσης. Εξαιτίας αυτών των πλεονεκτημάτων, προτείνονται ως μια πιο ευέλικτη λύση για τη διαχείριση εφαρμογών στο cloud.

Παρόλο που τόσο οι εικονικές μηχανές όσο και τα containers αξιοποιούν τεχνικές εικονικοποίησης, καλύπτουν διαφορετικές ανάγκες. Τα containers λειτουργούν ως εργαλεία για τη διάθεση και εκτέλεση λογισμικού, ακολουθώντας τη φιλοσοφία του Platform-as-a-Service (PaaS). Δίνουν έμφαση στη φορητότητα και στη διαλειτουργικότητα, ενώ αξιοποιούν την εικονικοποίηση σε επίπεδο λειτουργικού συστήματος. Αντίθετα, οι εικονικές μηχανές εστιάζουν στην κατανομή και διαχείριση του υλικού, ενσωματώνοντας τη λογική του Infrastructure-as-a-Service (IaaS).

Σε ορισμένες περιπτώσεις, τα containers μπορούν να αντικαταστήσουν τις εικονικές μηχανές, προσφέροντας δυναμική εκχώρηση των φυσικών πόρων μέσω του κατακερματισμού των φορτίων εργασίας (workloads) μεταξύ διαφορετικών cloud υποδομών [25].

Οι τεχνολογίες των containers διακρίνονται στις εξής βασικές κατηγορίες [26]:

Containers Συστήματος (System Containers):

Παρέχουν ένα περιβάλλον παρόμοιο με μια εικονική μηχανή, καθώς περιλαμβάνουν ένα πλήρες λειτουργικό σύστημα, επιτρέποντας την εκτέλεση πολλαπλών διεργασιών μέσα σε έναν μόνο container.

Τα container συστήματος επιτρέπουν την εκτέλεση ενός πλήρους λειτουργικού συστήματος σε ένα container. Η έννοια της διαστρωμάτωσης ισχύει επίσης και για το σύστημα container. Ξεκινώντας από μια βασική εικόνα λειτουργικού συστήματος, μπορεί να προσαρμοστεί προσθέτοντας βιβλιοθήκες, εργαλεία και δεδομένα σε νέα στρώματα.

Τα Linux Container (LXC) και OpenVZ επιτρέπουν την κοινή χρήση ενός πυρήνα Linux μεταξύ των container που έχουν κατασκευαστεί από την ίδια εικόνα. Το LXC χρησιμοποιεί τυπικό πυρήνα, ενώ το OpenVZ χρησιμοποιεί έναν εξειδικευμένο πυρήνα. Το Windows Hyper-V Container (WHC) είναι η έκδοση της Microsoft για το container συστήματος, για να εγγυηθεί μια ισχυρή απομόνωση του περιβάλλοντος τα WHC δεν μοιράζονται τον πυρήνα μεταξύ των container [26].

Containers Εφαρμογών (Application Containers):

Ένα container εφαρμογής έχει σχεδιαστεί ειδικά για την εκτέλεση εφαρμογών ή μεμονωμένων στοιχείων εφαρμογών σε ένα περιβάλλον container. Η δομή του βασίζεται σε μια σειρά από επίπεδα εικόνων (image layers), όπου κάθε επίπεδο συμβάλλει στη δημιουργία του τελικού container.

Συνήθως, οι προγραμματιστές ξεκινούν με μια βασική εικόνα, η οποία περιλαμβάνει τον πυρήνα του λειτουργικού συστήματος και τις προεπιλεγμένες βιβλιοθήκες. Στη συνέχεια, η εφαρμογή προστίθεται σε ένα νέο επίπεδο, το οποίο μπορεί να χωριστεί σε δύο μέρη: ένα χαμηλότερο επίπεδο που περιέχει τον πηγαίο κώδικα και τις βιβλιοθήκες και ένα ανώτερο επίπεδο με το εκτελέσιμο αρχείο. Επιπλέον, τα δεδομένα της εφαρμογής και οι εντολές εκτέλεσης αποθηκεύονται σε ανώτερα επίπεδα.

Αφού κατασκευαστεί η πολυεπίπεδη εικόνα του container, όλα τα επίπεδα εκτός από το τελευταίο γίνονται διαθέσιμα μόνο για ανάγνωση (read-only), ενώ το ανώτερο επίπεδο παραμένει αναγνώσιμο και εγγράψιμο (read-write), αλλά μη διατηρήσιμο, που σημαίνει ότι το περιεχόμενό του χάνεται όταν το container διαγραφεί.

Η πιο δημοφιλής τεχνολογία container εφαρμογών είναι το Docker, μια πολυπλατφορμική λύση που υποστηρίζει Linux, macOS και Windows. Το Docker επεκτείνει το LXC, ενσωματώνοντας API σε επίπεδο πυρήνα και εφαρμογής, ώστε να διευκολύνει τη διαχείριση των containers και των εφαρμογών. Επιπλέον, το LXC μπορεί να χρησιμοποιηθεί ως container εφαρμογής. Το rkt είναι μια εναλλακτική τεχνολογία που έχει αναπτυχθεί ειδικά για την εκτέλεση containers σε cloud-native περιβάλλοντα, όπως το CoreOS (Container Linux). Αντίστοιχα, η Windows Server Container (WSC) αποτελεί την έκδοση της Microsoft για τη διαχείριση containers εφαρμογών [26].

Διαχειριστές Container (Container Managers):

Πρόκειται για εργαλεία που αυτοματοποιούν την ανάπτυξη, την κλιμάκωση και τη διαχείριση των containers σε πολλαπλά συστήματα, εξασφαλίζοντας βέλτιστη χρήση των πόρων και υψηλή διαθεσιμότητα.

Ένας διαχειριστής containers (container manager) είναι ένα πλαίσιο (framework) που παρέχει ένα σύνολο API για την εύκολη διαχείριση ολόκληρου του κύκλου ζωής των containers. Οι λύσεις διαχείρισης containers διακρίνονται σε δύο κατηγορίες:

- On-premise λύσεις, που απαιτούν εγκατάσταση, ρύθμιση και διαχείριση σε ιδιωτικά data centers ή στο cloud.
- Διαχειριζόμενες λύσεις, που παρέχονται ως υπηρεσίες από παρόχους cloud και δεν απαιτούν άμεση διαχείριση από τον χρήστη.

Το Docker αναπτύχθηκε ως σύστημα διαχείρισης containers, ενώ το συνολικό οικοσύστημα διαχείρισης containers συνεχώς εξελίσσεται.

Ορισμένες cloud πλατφόρμες που παρέχουν υπηρεσίες διαχείρισης containers είναι:

- Google Kubernetes Engine (GKE) [27]
- Microsoft Azure Container Services [28]
- Amazon Elastic Container Service (ECS) [29]

Αυτές οι πλατφόρμες υποστηρίζουν Docker και LXC. Για containers συστήματος, η διαχείριση του LXC γίνεται μέσω LXD, ενώ το OpenVZ διαθέτει επίσης API διαχείρισης containers [26].

Τόσα τα system όσο και τα application containers λειτουργούν παρόμοια με μια εφαρμογή, καθώς εκτελούνται ως διαδικασίες πάνω από το λειτουργικό σύστημα (OS), με τη διαφορά ότι κάθε container είναι πλήρως απομονωμένο μέσα στον δικό του χώρο διευθύνσεων. Ωστόσο, σε αντίθεση με μια κανονική διεργασία, ένα (application) container δεν περιλαμβάνει μόνο το κύριο εκτελέσιμο αρχείο της εφαρμογής, αλλά πακετάρει και όλες τις βιβλιοθήκες και λοιπές εξαρτήσεις που απαιτούνται για την εκτέλεσή της.

Δύο βασικά χαρακτηριστικά του πυρήνα του Linux που καθιστούν δυνατή την κοινή χρήση της ίδιας φυσικής μηχανής από πολλά containers είναι οι χώροι ονομάτων (namespaces) και οι ομάδες ελέγχου (Cgroups).

Namespaces (Χώροι Ονομάτων): Παρέχουν αφαίρεση των πόρων του πυρήνα, όπως τα process IDs, τα δικτυακά interfaces και τα hostnames, επιτρέποντας σε κάθε container να θεωρεί ότι διαθέτει ένα πλήρως απομονωμένο σύστημα πόρων.

Cgroups (Ομάδες Ελέγχου): Ένας μηχανισμός διαχείρισης πόρων στον πυρήνα του Linux που επιτρέπει τον καθορισμό πόρων, όπως CPU, μνήμη και δικτύου, για κάθε container από τους διαχειριστές. Οι διατιθέμενοι πόροι μπορούν να προσαρμοστούν δυναμικά ώστε να μην επιτρέπεται σε ένα container να χρησιμοποιεί περισσότερα από τα καθορισμένα όρια [30].

2.2.2.2.2 Container Orchestration

Η ενορχήστρωση containers επιτρέπει στους παρόχους cloud και εφαρμογών να προσδιορίζουν τη διαδικασία επιλογής, ανάπτυξης, παρακολούθησης και δυναμικής διαμόρφωσης εφαρμογών που αποτελούνται από πολλά containers (multi-container), μέσα στο cloud περιβάλλον.

Η ενορχήστρωση επικεντρώνεται στη διαχείριση κατά την εκτέλεση, υποστηρίζοντας την ανάπτυξη, λειτουργία και συντήρηση των containers.

Ένας container orchestrator προσφέρει συνήθως τις εξής δυνατότητες:

Διαχείριση ορίων πόρων (Resource Limit Control), Προγραμματισμός (scheduling), Εξισορρόπηση φορτίου (load balancing), Έλεγχος υγείας (health monitoring), Ανοχή σε σφάλματα (fault tolerance), Αυτόματη κλιμάκωση (autoscaling).

- **Έλεγχος Πόρων (Resource Limit Control)**

  Ο έλεγχος πόρων διασφαλίζει ότι κάθε container έχει σαφώς καθορισμένα όρια CPU και μνήμης, προκειμένου να αποφεύγεται η αλόγιστη κατανάλωση πόρων και η παρεμβολή μεταξύ containers. Οι διαχειριστές containers παρέχουν API που επιτρέπουν την ακριβή ρύθμιση αυτών των ορίων, εξασφαλίζοντας μια δίκαιη κατανομή των υπολογιστικών πόρων.

- **Προγραμματισμός (Scheduling)**

  Ο προγραμματισμός αφορά την τοποθέτηση των containers σε διακριτούς κόμβους του συστήματος, με βάση διαθεσιμότητα πόρων, προτεραιότητες ή συνδυασμό αυτών. Σε πιο πολύπλοκα περιβάλλοντα, μπορούν να ενσωματωθούν εξωτερικοί προγραμματιστές (custom schedulers) για μεγαλύτερη ευελιξία.

- **Εξισορρόπηση Φορτίου (Load Balancing)**

  Ο load balancer κατανέμει δυναμικά τα αιτήματα μεταξύ των διαθέσιμων containers. Το round-robin είναι η προεπιλεγμένη μέθοδος κατανομής, αλλά μπορούν να χρησιμοποιηθούν εξωτερικοί εξισορροπητές φορτίου για βελτιωμένη απόδοση.

- **Έλεγχος Υγείας (Health Check)**

  Οι έλεγχοι υγείας διασφαλίζουν ότι κάθε container ανταποκρίνεται σωστά, μέσω:

  - Ελέγχου ανοιχτών θυρών TCP/UDP/SSH
  - Επαλήθευσης HTTP αιτημάτων και αποκρίσεων

  Εάν ένας container αποτύχει στους ελέγχους, αντικαθίσταται αυτόματα.

- **Ανοχή σε Σφάλματα (Fault Tolerance)**

  Η ανοχή σε σφάλματα διασφαλίζεται μέσω αντιγράφων containers και μηχανισμών υψηλής διαθεσιμότητας, ώστε το σύστημα να διατηρεί τη σταθερότητά του ακόμη και σε περιπτώσεις αποτυχίας.

- **Αυτόματη Κλιμάκωση (Autoscaling)**

  Το autoscaling προσθέτει ή αφαιρεί containers αυτόματα, βασιζόμενο στη χρήση CPU και μνήμης, ενώ υποστηρίζει προσαρμοσμένες πολιτικές για πιο ευέλικτη διαχείριση φορτίου [26].

2.2.3 Microservices and Microservice Architecture

2.2.3.1. Ορισμός και σύγκριση με μονολιθική αρχιτεκτονική

Η αρχιτεκτονική microservices έχει αποκτήσει ιδιαίτερη σημασία στην ανάπτυξη cloud-native εφαρμογών, οι οποίες αξιοποιούν πλήρως τις δυνατότητες του cloud computing για να γίνουν ανθεκτικές, επεκτάσιμες και εύκολα διαχειρίσιμες σε κατανεμημένα περιβάλλοντα.

Η μετάβαση από τις μονολιθικές στις microservices αρχιτεκτονικές αποτελεί σημαντική αλλαγή στον τρόπο ανάπτυξης και διαχείρισης των λογισμικών. Οι μονολιθικές αρχιτεκτονικές, που αποτέλεσαν το κυρίαρχο πρότυπο για δεκαετίες, βασίζονται στη δημιουργία μιας εφαρμογής ως ενιαίου κώδικα [31].

Τα microservices αντιθέτως, αποτελούν ένα σύγχρονο αρχιτεκτονικό πρότυπο, το οποίο στοχεύει στη δημιουργία συστημάτων λογισμικού ως συνόλων μικρών και αυτόνομων υπηρεσιών, τα οποία μπορούν να αναπτυχθούν, δοκιμασθούν και κλιμακωθούν σε διαφορετικές πλατφόρμες [32][31].

2.2.3.2 Τεχνικά χαρακτηριστικά και Επικοινωνία Microservices

Ένα **microservice** είναι μία συνεκτική και αυτόνομη διεργασία που επικοινωνεί με άλλα συστήματα μέσω απλών και ελαφριών πρωτοκόλλων. Η επικοινωνία αυτή μπορεί να είναι είτε σύγχρονη μέσω RESTful APIs και HTTP είτε ασύγχρονη, μέσω ουρών μηνυμάτων (message queues) όπως το RabbitMQ ή το Kafka [33][31].

Για παράδειγμα, σε κάποια cloud-native streaming εφαρμογή, μια υπηρεσία που προτείνει μουσική στο χρήστη θεωρείται microservice μόνο εφόσον αποκλειστικά παρέχει νέες προτάσεις. Δεν θα πρέπει να περιλαμβάνει επιπλέον λειτουργίες όπως δημιουργία λιστών αναπαραγωγής ή διαχείρισης της εισόδου του χρήστη στην εφαρμογή.

Σε τεχνικό επίπεδο, οι μικρο-υπηρεσίες είναι ανεξάρτητες μονάδες που αναπτύσσονται μεμονωμένα και διαθέτουν αποκλειστικά συστήματα αποθήκευσης δεδομένων, όπως ξεχωριστές βάσεις δεδομένων, εφαρμόζοντας μια αποκεντρωμένη προσέγγιση στη διαχείριση δεδομένων [31][33].

Μια αρχιτεκτονική μικρο-υπηρεσιών είναι ένα είδος κατανεμημένης εφαρμογής, στην οποία όλες οι λειτουργικές της μονάδες υλοποιούνται ως διακριτά microservices. Συνεχίζοντας το προηγούμενο παράδειγμα της music-streaming εφαρμογής, η υπηρεσία που παράγει προτάσεις μουσικής θα μπορούσε να αλληλοεπιδρά με τις εξής υπηρεσίες:

- Υπηρεσία Πιστοποίησης (Authentication Service): Επαληθεύει την ταυτότητα χρήστη μέσω REST API.
- Υπηρεσία Λιστών Αναπαραγωγής (Playlist Service): Αποθηκεύει και ανακτά δεδομένα από ξεχωριστή βάση NoSQL.
- Υπηρεσία Καταγραφής (Logging Service): Αποστέλλει ασύγχρονα μηνύματα μέσω Kafka για παρακολούθηση σφαλμάτων.

2.2.3.3 Service Function Chaining (SFC)

Η αλυσιδωτή σύνδεση λειτουργιών υπηρεσίας (SFC) είναι μια τεχνολογία που επιτρέπει την ευέλικτη διαχείριση της κίνησης συγκεκριμένων υπηρεσιών/εφαρμογών, παρέχοντας λύσεις για την ταξινόμηση των ροών και την επιβολή κατάλληλων πολιτικών κατά μήκος της ροής σύμφωνα με τις απαιτήσεις της υπηρεσίας και τις λαμβάνοντας υπόψη την κατάσταση διαθεσιμότητας του δικτύου. Η SFC ορίζεται ως ένα αλυσιδωτό σύνολο υπηρεσιών λειτουργιών (SFs) που διαχειρίζεται την κυκλοφορία της παράδοσης (επίπεδο δεδομένων), του ελέγχου και της παρακολούθησης (control) μιας συγκεκριμένης υπηρεσίας/εφαρμογής [34].

Σχετικοί Ορισμοί:

Λειτουργία υπηρεσίας: Λειτουργία που είναι υπεύθυνη για τη συγκεκριμένη επεξεργασία των λαμβανόμενων πακέτων. Μια λειτουργία υπηρεσίας μπορεί να ενεργεί σε διάφορα επίπεδα μιας στοίβας πρωτοκόλλων (π.χ. στο επίπεδο δικτύου ή σε άλλα επίπεδα OSI).

Πολλαπλές λειτουργίες υπηρεσίας μπορούν να ενσωματωθούν στο ίδιο στοιχείο δικτύου. Εάν θεωρήσουμε για παράδειγμα ένα σύστημα "προστασίας" ως υπηρεσία,

τότε τα στοιχεία του, δηλαδή το Firewall, το Deep Packet Inspection και το ο σαρωτής ιών θα ήταν τα συστατικά του και θα ονομάζονταν υπηρεσίες.

Αλυσίδα λειτουργιών υπηρεσίας: είναι ένα διατεταγμένο ή μερικώς διατεταγμένο σύνολο αφηρημένων λειτουργιών υπηρεσίας (SF) και των περιορισμών διάταξης που πρέπει να εφαρμόζονται σε πακέτα, πλαίσια ή/και ροές που επιλέγονται ως αποτέλεσμα της ταξινόμησης. Η υπονοούμενη σειρά μπορεί να μην να είναι γραμμική εξέλιξη, καθώς η αρχιτεκτονική επιτρέπει SFCs που αντιγράφονται σε περισσότερους από έναν κλάδους, και επιτρέπει επίσης περιπτώσεις όπου υπάρχει ευελιξία στη σειρά με την οποία πρέπει να εφαρμοστούν οι λειτουργίες εξυπηρέτησης [35].

2.2.3.3.1 Continuous Delivery και DevOps σε Microservices Αρχιτεκτονικές

Η ανάπτυξη μιας αρχιτεκτονικής microservices απαιτεί τη διαχείριση πολλαπλών ανεξάρτητων εφαρμογών, όπως microservices και gateways. Κάθε microservice ή gateway απαιτεί εξειδικευμένες ρυθμίσεις, τόσο στο επίπεδο της εφαρμογής όσο και στις υπηρεσίες του παρόχου (π.χ., AWS, Azure). Ένα από τα μεγαλύτερα προβλήματα σε αυτήν την αρχιτεκτονική είναι η έκδοση υπηρεσιών (service versioning), καθώς η διάθεση μιας νέας έκδοσης μπορεί να προκαλέσει ασυμβατότητες με άλλες υπηρεσίες. Επομένως, είναι κρίσιμο οι ομάδες ανάπτυξης και οι χρήστες των υπηρεσιών να συνεργάζονται στενά για να διαχειριστούν ομαλά τις αναβαθμίσεις και τις αποσύρσεις υπηρεσιών [36].

Η Συνεχής Παράδοση (Continuous Delivery) είναι μια πρακτική ανάπτυξης λογισμικού που επιτρέπει την άμεση διάθεση εφαρμογών σε οποιοδήποτε περιβάλλον. Στόχος της είναι η αυτοματοποίηση του κύκλου ζωής της ανάπτυξης λογισμικού στο μέγιστο δυνατό βαθμό, αξιοποιώντας τεχνικές όπως η Συνεχής Ενσωμάτωση (Continuous Integration - CI) και η Συνεχής Ανάπτυξη (Continuous Deployment - CD), καθώς και τις αρχές του DevOps.

Το DevOps είναι μια κουλτούρα συνεργασίας που ενισχύει τη συνεργασία μεταξύ ομάδων ανάπτυξης και λειτουργιών, από τα πρώτα στάδια κάθε έργου. Ο στόχος του είναι να επιταχύνει τη διάθεση λογισμικού και να προσφέρει ευελιξία σε όλα τα στάδια ανάπτυξης των εφαρμογών. Δεδομένου ότι οι αρχιτεκτονικές microservices περιλαμβάνουν πολλαπλές ανεξάρτητες υπηρεσίες, η αυτοματοποίηση της διαδικασίας παράδοσης καθίσταται απαραίτητη [32].

2.2.3.4 Τελική σύγκριση μονολιθικών εφαρμογών και εφαρμογών με microservices αρχιτεκτονική

Η συγκεντρωτική σύγκριση Μονολιθικής και Microservices Αρχιτεκτονικής εφαρμογών παρουσιάζεται στον παρακάτω πίνακα:

Χαρακτηριστικό	**Μονολιθική Αρχιτεκτονική**	**Αρχιτεκτονική Microservices**
Δομή	Ενιαία εφαρμογή, όλα τα modules είναι μέρος του ίδιου κώδικα.	Διάσπαση της εφαρμογής σε μικρές, αυτόνομες υπηρεσίες.
Κλιμάκωση	Κλιμάκωση ολόκληρης της εφαρμογής, ανεξάρτητα από το ποια μέρη χρειάζονται πόρους.	Επιλεκτική κλιμάκωση μόνο των απαιτούμενων υπηρεσιών.
Ανάπτυξη	Οι ομάδες ανάπτυξης εργάζονται σε έναν κοινό κώδικα.	Οι ομάδες ανάπτυξης εργάζονται ανεξάρτητα σε διαφορετικές υπηρεσίες.
Τεχνολογίες	Μία κύρια τεχνολογία (single tech stack).	Δυνατότητα χρήσης διαφορετικών τεχνολογιών ανά υπηρεσία.
Σφάλματα	Ένα σφάλμα μπορεί να επηρεάσει ολόκληρη την εφαρμογή.	Τα σφάλματα επηρεάζουν μόνο το σχετικό microservice.
Επικοινωνία	Εσωτερική επικοινωνία.	Επικοινωνία μέσω REST/gRPC APIs ή message queues (Kafka, RabbitMQ).
Δεδομένα	Ενιαία βάση δεδομένων για όλη την εφαρμογή.	Κάθε microservice μπορεί να έχει δική του βάση δεδομένων.
DevOps & CI/CD	Δύσκολη αυτοματοποίηση λόγω του μονολιθικού χαρακτήρα.	Ιδανικό για DevOps, καθώς επιτρέπει ανεξάρτητες αναπτύξεις και releases.

Κατάλληλο για:	Μικρές εφαρμογές ή legacy συστήματα.	Μεγάλες, κατανεμημένες, cloud-native εφαρμογές.

Πίνακας 2.1: Σύγκριση Μονολιθικής και Microservices Αρχιτεκτονικής

2.2.4 Διαχείριση cloud-native εφαρμογών σε κατανεμημένα περιβάλλοντα

Έχοντας πλέον αναλύσει το cloud computing, καθώς και τις cloud-native εφαρμογές που χρησιμοποιούν microservice αρχιτεκτονική, μπορούμε να εξετάσουμε την πολύ-επίπεδη αρχιτεκτονική Edge/Fog/Cloud.

2.2.4.1 Συνεργασία Edge, Fog και Cloud

Παρότι οι όροι Edge και Fog Computing έχουν χρησιμοποιηθεί με ελαφρώς διαφορετικές έννοιες από διάφορους ερευνητές, εδώ θεωρούμε ως Edge Node κάθε συσκευή με ενσωματωμένο επεξεργαστή ή πολυεπεξεργαστικό σύστημα, όπως ρομπότ, οχήματα με μπαταρία, smartphones, Raspberry Pi ή Arduino. Τα Fog Nodes, από την άλλη, είναι υποδομές παρόμοιες με αυτές του Cloud, που βρίσκονται γεωγραφικά κοντά στους Edge κόμβους και παρέχουν πρόσθετη υπολογιστική ισχύ [37].

Σε αντίθεση με τους πόρους του Cloud, οι πόροι στο Edge παρουσιάζουν τρεις βασικές προκλήσεις:

- Περιορισμένους πόρους – Οι Edge συσκευές διαθέτουν ασθενέστερους επεξεργαστές και περιορισμένο ενεργειακό προϋπολογισμό, γεγονός που μειώνει τις δυνατότητες υπολογιστικής επεξεργασίας.
- Ετερογένεια – Τα συστήματα αυτά βασίζονται σε διαφορετικές αρχιτεκτονικές επεξεργαστών, καθιστώντας δύσκολη την ενοποίηση και τη βελτιστοποίηση.
- Δυναμικότητα – Οι φόρτοι εργασίας μεταβάλλονται δυναμικά, ενώ οι εφαρμογές ανταγωνίζονται για τους ίδιους περιορισμένους πόρους.

Η διαχείριση πόρων, επομένως, αποτελεί κεντρικό πρόβλημα στα περιβάλλοντα Fog και Edge computing [38].

Η αποτελεσματική σύζευξη Edge-Fog-Cloud μπορεί να επιτύχει ταχύτερη εκτέλεση εφαρμογών και βελτιστοποιημένη διαχείριση δεδομένων:

- Το Edge Computing εκτελεί επεξεργασία απευθείας στις συσκευές, μειώνοντας τον χρόνο απόκρισης και την κυκλοφοριακή επιβάρυνση του δικτύου.

- Το Fog Computing προσφέρει ενδιάμεση υπολογιστική ισχύ κοντά στις πηγές δεδομένων, υποστηρίζοντας τις Edge συσκευές.
- Το Cloud Computing αναλαμβάνει όταν απαιτούνται μεγαλύτεροι πόροι, ιδιαίτερα σε περιπτώσεις υπερφόρτωσης [39].

2.2.4.2 Resource Allocation σε περιβάλλοντα Νέφους

Το μοντέλο Edge / Fog Computing αποτελεί ένα υπολογιστικό πρότυπο το οποίο στοχεύει στην εκτέλεση των λειτουργιών των έξυπνων εφαρμογών κοντά στις IoT (Internet of Things) συσκευές, αποφεύγοντας την ανάγκη αποστολής δεδομένων σε απομακρυσμένα data center.

Η αρχιτεκτονική microservices σε συνδυασμό με containers ενισχύει την ευελιξία και την αυτοματοποίηση στην ανάπτυξη, αναδιάταξη και επεκτασιμότητα των εφαρμογών IoT. Τεχνολογίες όπως Docker, Kubernetes, OpenShift και Swarm επιτρέπουν την ενορχήστρωση και διαχείριση των container σε περιβάλλοντα Edge και Fog [37].

Υπάρχει πληθώρα τεχνικών και αρχιτεκτονικών για να γίνει διαχείριση των πόρων στο fog/edge computing, κάποιες από τις οποίες παρουσιάζονται συνοπτικά παρακάτω.

2.2.4.2.1 Αρχιτεκτονικές Data Flow

Οι αρχιτεκτονικές ροής δεδομένων βασίζονται στην κατεύθυνση μεταφοράς των υπολογιστικών φορτίων και δεδομένων στο σύνολο του υπολογιστικού οικοσυστήματος. Για παράδειγμα, τα φορτία μπορεί να μετακινηθούν από τις συσκευές χρήστη προς τους κόμβους Edge, ή αντίστροφα, από το Cloud προς την περιφέρεια του δικτύου. Παραδείγματα αποτελούν η συσσώρευση και η εκφόρτωση.

Συσσώρευση (aggregation): Στο μοντέλο αυτό, ο κόμβος Edge συλλέγει δεδομένα από πολλές τελικές συσκευές και εκτελεί μερική επεξεργασία (π.χ. φιλτράρισμα ή περικοπή). Στόχος είναι η μείωση του φόρτου επικοινωνίας, αποφεύγοντας τη μετάδοση περιττής ή πλεονάζουσας πληροφορίας εκτός του επιπέδου Edge.

Εκφόρτωση (Offloading): Η εκφόρτωση αναφέρεται στη μεταφορά εξυπηρετητών, εφαρμογών ή δεδομένων κοντά στην περιφέρεια του δικτύου. Η προσέγγιση αυτή είτε ενισχύει τις υπολογιστικές δυνατότητες μεμονωμένων ή ομαδικών χρηστών, είτε φέρει υπηρεσίες Cloud πλησιέστερα στην πηγή των δεδομένων, βελτιώνοντας την απόκριση και την αποδοτικότητα [38].

Μια αξιοσημείωτη στρατηγική είναι η αναπαραγωγή υπηρεσιών (service replication) σε όλα τα επίπεδα (Edge, Fog, Cloud), για την αύξηση της αξιοπιστίας και ανθεκτικότητας. Όταν οι Edge και Fog κόμβοι υπερφορτώνονται, τα αιτήματα μπορούν να ανακατευθυνθούν στο Cloud, το οποίο λειτουργεί ως εφεδρική επιλογή. Ωστόσο, η αναπαραγωγή υπηρεσιών δημιουργεί τεχνικές προκλήσεις, όπως ασυνέπειες μεταξύ των αντιγράφων δεδομένων και υπολογισμών, οι οποίες πρέπει να αντιμετωπιστούν προσεκτικά [37].

2.2.4.2.2 Αρχιτεκτονικές Control

Μια δεύτερη προσέγγιση για την κατηγοριοποίηση των αρχιτεκτονικών διαχείρισης πόρων σε περιβάλλοντα Fog και Edge βασίζεται στον τρόπο άσκησης ελέγχου επί των πόρων. Σύμφωνα με το παρόν άρθρο, διακρίνονται δύο βασικοί τύποι αρχιτεκτονικής ελέγχου: η κεντρική (centralized) και η κατανεμημένη (distributed).

Η κεντρική αρχιτεκτονική ελέγχου χρησιμοποιεί έναν ενιαίο ελεγκτή, ο οποίος λαμβάνει αποφάσεις για την υπολογιστική επεξεργασία, τη δικτύωση και την επικοινωνία μεταξύ των πόρων στο Edge.

Αντιθέτως, στην κατανεμημένη αρχιτεκτονική, η λήψη αποφάσεων κατανέμεται σε πολλούς κόμβους edge, επιτρέποντας έναν πιο αποκεντρωμένο και αυτόνομο έλεγχο των πόρων [38].

2.2.4.2.3 Αρχιτεκτονικές Tenancy

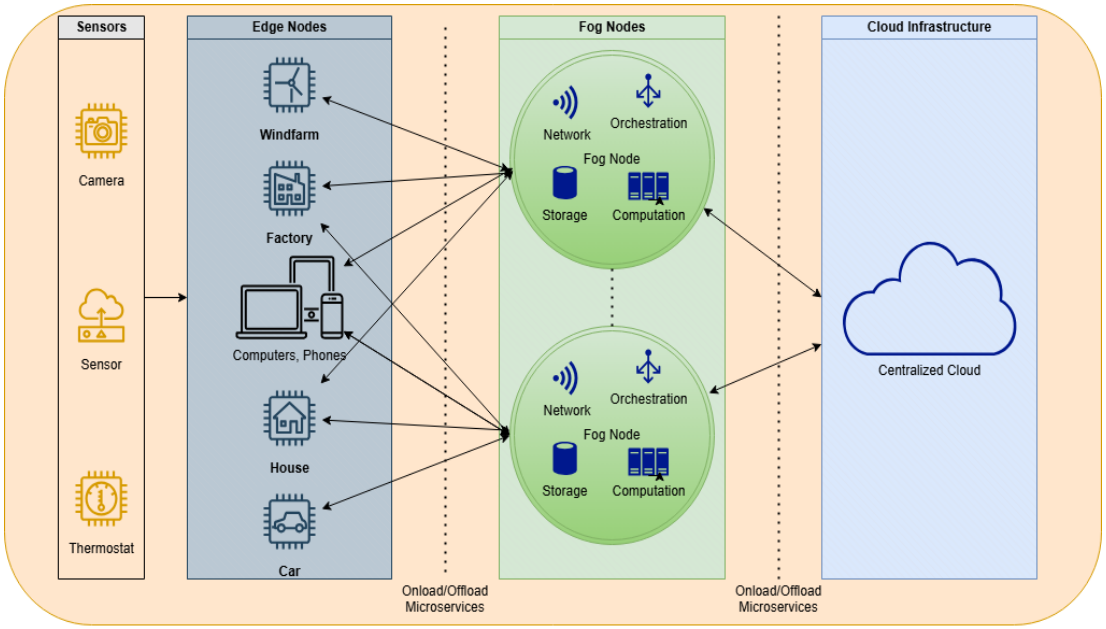
Μια τρίτη μέθοδος για την κατηγοριοποίηση των αρχιτεκτονικών διαχείρισης πόρων σε περιβάλλοντα Fog/Edge είναι η έννοια του καθεστώτος χρήσης (tenancy). Στα κατανεμημένα συστήματα, ο όρος αυτός αναφέρεται στο κατά πόσο οι υποκείμενοι υλικοί πόροι χρησιμοποιούνται αποκλειστικά από έναν χρήστη ή μοιράζονται μεταξύ πολλών, με σκοπό τη βελτίωση της αποδοτικότητας και της ενεργειακής χρήσης.

Ένα σύστημα μονοκατοχής (single-tenant) σημαίνει ότι οι πόροι χρησιμοποιούνται αποκλειστικά από έναν μόνο φορέα ή οργανισμό.

Αντίθετα, ένα σύστημα πολυκατοχής (multi-tenant) επιτρέπει σε πολλαπλούς φορείς να μοιράζονται τους ίδιους πόρους, βελτιστοποιώντας έτσι την εκμετάλλευση των υποδομών και τη μείωση της κατανάλωσης ενέργειας.

Για συστήματα κατανεμημένα και προσβάσιμα από το ευρύ κοινό, η πολυκατοχή είναι κρίσιμη για την επίτευξη επεκτασιμότητας και οικονομικής βιωσιμότητας [38].

Στο παρακάτω σχήμα γίνεται αναπαράσταση του μοντέλου fog/edge/cloud, όπου microservices γίνονται offload μεταξύ των επιπέδων όταν είναι αναγκαίο, ενώ στο fog επίπεδο υπάρχει οριζόντιο offloading μεταξύ των nodes.



Εικόνα 6: Edge/Fog/Cloud Computing Continuum

2.3 Machine Learning

Ορισμός βάση Mitchell T.:

Σύμφωνα με τον ορισμό του Mitchell T., ένα πρόγραμμα υπολογιστή θεωρείται ότι μαθαίνει από μια εμπειρία E, αναφορικά με μία κατηγορία εργασιών T και ένα μέτρο απόδοσης P, όταν η απόδοσή του στις εργασίες της κατηγορίας T — όπως αξιολογείται από το μέτρο P — βελτιώνεται μέσω της εμπειρίας E [40].

Δηλαδή, η διαδικασία μάθησης υφίσταται όταν η απόδοση του προγράμματος βελτιώνεται με την πάροδο του χρόνου, ως αποτέλεσμα της εμπειρίας του, και αυτή η πρόοδος μπορεί να μετρηθεί ποσοτικά μέσω του δείκτη P.

Το σύστημα μαθαίνει από το παρελθόν και προσπαθεί να αποκτήσει τη βελτιστοποιημένη γνώση για λήψη αποφάσεων, όπως συμβαίνει σε περιβάλλοντα Μαρκοβιανών διαδικασιών απόφασης (Markov Decision Process - MDPs). Εκεί, η μηχανή επιλέγει ενέργειες που μεγιστοποιούν την απόδοση, προσαρμόζοντας διαρκώς τη στρατηγική της ώστε να επιτυγχάνει ακριβέστερα αποτελέσματα [41].

Οι αλγόριθμοι μηχανικής μάθησης κατηγοριοποιούνται συνήθως σε τέσσερις βασικές κατηγορίες, ανάλογα με τον τρόπο μάθησης:

Επιτηρούμενη μάθηση (Επιβλεπόμενη Μάθηση, Supervised learning) - Η επιβλεπόμενη μάθηση αποσκοπεί στη δημιουργία μιας συνάρτησης που αντιστοιχίζει τις εισόδους σε επιθυμητές εξόδους. Χρησιμοποιείται ευρέως σε προβλήματα ταξινόμησης, όπου ο στόχος είναι να μάθει το σύστημα ένα προκαθορισμένο σύστημα κατηγοριοποίησης. Ο αλγόριθμος λαμβάνει παραδείγματα εισόδου-εξόδου και εκπαιδεύεται ώστε να προβλέπει τη σωστή κατηγορία για νέα δεδομένα.

Μη Επιτηρούμενη μάθηση (Μη επιβλεπόμενη, Unsupervised learning) - Η μη επιβλεπόμενη μάθηση χρησιμοποιείται όταν τα δεδομένα δεν φέρουν ετικέτες εξόδου, με σκοπό την ανακάλυψη κρυμμένων δομών ή προτύπων. Η τεχνική αυτή αναγνωρίζει κανονικότητες ή ανωμαλίες μέσα σε σύνολα παρατηρήσεων, χωρίς να απαιτείται αξιολόγηση ακρίβειας.

Ημι-επιτηρούμενη μάθηση (Ημι-επιβλεπόμενη, Semi-supervised learning) - Η ημι-επιβλεπόμενη μάθηση εφαρμόζεται όταν μόνο ένα μέρος των δεδομένων είναι χαρακτηρισμένο. Συνδυάζει την επιβλεπόμενη και τη μη επιβλεπόμενη μάθηση, αξιοποιώντας ένα μικρό αριθμό δεδομένων με ετικέτες και μεγαλύτερο όγκο μη επισημασμένων δεδομένων.

Ενισχυτική Μάθηση (Reinforcement learning) - Η ενισχυτική μάθηση επιτρέπει σε πράκτορες ή λογισμικά να αναπτύσσουν βέλτιστες στρατηγικές συμπεριφοράς, μέσω

της αλληλεπίδρασής τους με το περιβάλλον. Η εκμάθηση βασίζεται σε ανατροφοδότηση (αμοιβή ή ποινή), με στόχο τη μέγιστη απόδοση εντός ενός συγκεκριμένου πλαισίου.

2.3.1 Χρήση Μηχανικής Μάθησης για Resource Allocation στο Νέφος

Στα προβλήματα πολυδιάστατης ανάθεσης πόρων, λαμβάνονται υπόψη διαφορετικοί τύποι πόρων, όπως CPU, μνήμη και αποθηκευτικός χώρος. Αυτή η πολυπλοκότητα καθιστά το πρόβλημα ισχυρά NP-δύσκολο, δηλαδή υπολογιστικά δύσκολο να επιλυθεί με βέλτιστο τρόπο [42].

Η κατανομή πόρων σε περιβάλλοντα Fog είναι ένα NP-δυσχερές πρόβλημα, με στόχο να διανεμηθούν οι διαθέσιμοι πόροι σε πολλαπλές εφαρμογές και υπηρεσίες, επιδιώκοντας είτε τη μείωση είτε τη μεγιστοποίηση διαφόρων και συχνά αντικρουόμενων στόχων, ενώ ταυτόχρονα τηρούνται συγκεκριμένοι περιορισμοί. Μια εκτενής ανασκόπηση της βιβλιογραφίας δείχνει ότι οι προτεινόμενες λύσεις βασίζονται σε τέσσερις κύριες κατηγορίες αλγορίθμων:

2.3.1.1 Αλγόριθμοι βασισμένοι σε heuristics

Οι αλγόριθμοι αυτοί επικεντρώνονται στην αντιμετώπιση του προβλήματος κατανομής πόρων χωρίς εγγύηση παγκόσμιας βέλτιστης λύσης. Συνηθέστερα αξιοποιούν παραδοσιακές μεθόδους όπως για παράδειγμα greedy search, ώστε να βρουν γρήγορα ικανοποιητικές κατανομές [43].

Οι ευρετικές μέθοδοι έχουν σχεδιαστεί για να βρίσκουν καλές, προσεγγιστικές λύσεις σε δύσκολα συνδυαστικά προβλήματα που διαφορετικά δεν μπορούν να επιλυθούν με τους διαθέσιμους αλγορίθμους βελτιστοποίησης. Μια ευρετική είναι μια τεχνική άμεσης αναζήτησης που χρησιμοποιεί ευνοϊκούς κανόνες για να εντοπίσει βελτιωμένες λύσεις. Το πλεονέκτημα των ευρετικών μεθόδων είναι ότι συνήθως βρίσκουν γρήγορα λύσεις. Το μειονέκτημα είναι ότι η ποιότητα της λύσης (σε σχέση με το βέλτιστο) είναι γενικά άγνωστη [44].

Πιο συγκεκριμένα, ο greedy search αλγόριθμος επιβάλλει βελτίωση της τιμής της αντικειμενικής συνάρτησης με κάθε κίνηση αναζήτησης. Η αναζήτηση τερματίζεται σε ένα τοπικό βέλτιστο όπου δεν είναι δυνατή περαιτέρω βελτίωση [44].

2.3.1.2 Ακριβείς αλγόριθμοι βελτιστοποίησης (exact optimization):

Σχεδιάζονται για να εξασφαλίζουν την εύρεση της βέλτιστης λύσης. Τυπικά χρησιμοποιούν μεικτούς ακεραίους προγραμματισμούς (MIP) ή δυαδικούς ακεραίους προγραμματισμούς (BIP), αλλά λόγω του υψηλού υπολογιστικού κόστους εφαρμόζονται κυρίως σε μικρότερης κλίμακας προβλήματα [43].

Αξίζει να αναλυθεί περαιτέρω ο γραμμικός προγραμματισμός (linear programming) καθώς και ο mixed integer linear programming που χρησιμοποιήθηκε αργότερα στην επίλυση του προβλήματος.

Ένα (γραμμικό) μικτό ακέραιο πρόγραμμα (MIP) είναι ένα πρόγραμμα βελτιστοποίησης σε ένα σύνολο ακέραιων μεταβλητών (αγνώστων) και ένα σύνολο μεταβλητών πραγματικών τιμών (συνεχών), οι περιορισμοί είναι όλες οι γραμμικές εξισώσεις ή ανισώσεις, και ο στόχος είναι ένα γραμμική συνάρτηση που πρέπει να ελαχιστοποιηθεί (ή να μεγιστοποιηθεί). Αυτό μπορεί να είναι γραφεί μαθηματικά ως εξής

$$min\ cx\ +\ hy$$

$$Ax\ +\ Gy\ \geq b$$

$$x \in R_+^p,\ y \in Z_+^n$$

όπου R_+^p, ο χώρος των p-διάστατων μη αρνητικών πραγματικών διανυσμάτων: Z_+^n συμβολίζει το χώρο των nd-διάστατων μη αρνητικών ακέραιων διανυσμάτων: το x υποδηλώνει τις p συνεχείς μεταβλητές: y, δηλώνει τις n ακέραιες μεταβλητές- A και G είναι $m\ \times\ p$ και $m \times n$ πίνακες, αντίστοιχα. Το b είναι ένα m-vector (το διάνυσμα απαιτήσεων ή το διάνυσμα της δεξιάς πλευράς) και c και h είναι p- και n-διάστατα διανύσματα γραμμής.

Τα MIPs στα οποία p = 0 ονομάζονται (γραμμικά) ακέραια προγράμματα (Integer Programs - IP): εκείνα στα οποία τα όρια των ακέραιων μεταβλητών είναι όλα 0 και 1, ονομάζονται δυαδικά ή 0-1 MIPs, και εκείνα στα οποία το n = 0 ονομάζονται γραμμικά προγράμματα (Linear Programs - LP). Το πρόβλημα στο οποίο η αντικειμενική συνάρτηση ή/και οι περιορισμοί περιλαμβάνουν μη γραμμικές συναρτήσεις της μορφής

$$\min\{\, c(x,y) \colon g_i(x,y) \geq b_i i = 1, \dots, m,\ x \in R_+^p,\ y \in Z_+^n\}$$

είναι ένα μικτό ακέραιο μη γραμμικό πρόγραμμα (MINLP)[45].

Τα MIP επιλύονται σε τακτική βάση σε πολλούς τομείς των επιχειρήσεων, της διοίκησης, της επιστήμης και της μηχανικής. Η μοντελοποίηση προβλημάτων ως MIPs δεν είναι τετριμμένη. Πρέπει πρώτα να οριστούν οι μεταβλητές, στη συνέχεια ένα

σύνολο γραμμικών περιορισμών, ώστε να χαρακτηρίζεται ακριβώς το σύνολο των εφικτών λύσεων, και τέλος, μια γραμμική αντικειμενική συνάρτηση που πρέπει να ελαχιστοποιηθεί ή να μεγιστοποιηθεί.

2.3.1.3 Αλγόριθμοι βασισμένοι σε meta-heuristics

Στοχεύουν στην ευρετική προσέγγιση της βέλτιστης λύσης, αναζητώντας το χώρο των δυνατών λύσεων ευρύτερα [43]. Τα παραδείγματα συμπεριλαμβάνουν τοπική αναζήτηση (Local Search - LS), προσομοίωση ανόπτησης (simulated annealing - SA), αναζήτηση tabu (Tabu Search), γενετικούς αλγόριθμους (Genetic Algorithms- GA), αλγόριθμοι μυρμηγκιών (ant algorithms - AA), και επαναλαμβανόμενη τοπική αναζήτηση (ILS) [46].

Οι μεταευρετικές μέθοδοι έχουν σχεδιαστεί κυρίως για να αποφεύγουν τον εγκλωβισμό σε τοπικά βέλτιστα με επιτρέποντας κατώτερες κινήσεις, εάν είναι απαραίτητο. Η ελπίδα είναι ότι η πρόσθετη ευελιξία αναζήτησης θα οδηγήσει σε καλύτερη λύση.

Σε αντίθεση με την άπληστη ευρετική, η οποία τερματίζει πάντα όταν ένα τοπικό βέλτιστο είναι επιτυγχάνεται, ο τερματισμός μιας meta-heuristic αναζήτησης μπορεί να βασίζεται σε ένα από τα ακόλουθα σημεία αναφοράς:

1. Ο αριθμός των επαναλήψεων αναζήτησης υπερβαίνει έναν καθορισμένο αριθμό.
2. Ο αριθμός των επαναλήψεων από την τελευταία καλύτερη λύση υπερβαίνει έναν καθορισμένο αριθμό.
3. Η γειτονιά που σχετίζεται με το τρέχον σημείο αναζήτησης είναι είτε κενή είτε δεν μπορεί να οδηγήσει σε μια νέα βιώσιμη κίνηση αναζήτησης.
4. Η ποιότητα της τρέχουσας καλύτερης λύσης είναι αποδεκτή

Υπάρχουν δύο τύποι καθόδου τοπικής αναζήτησης [46]: α) μια "άπληστη κάθοδος" (GD), και β) μια "απότομη κάθοδος" (SD). Και στις δύο περιπτώσεις, η απόφαση για την αντικατάσταση της τρέχουσας λύσης από τη νέα (δηλ. να μετακινηθεί στη γειτονική λύση) ή όχι είναι θετική εάν μόνο η νέα λύση είναι σίγουρα καλύτερη από την τρέχουσα (δηλαδή, η διαφορά των τιμών της αντικειμενικής συνάρτησης είναι αρνητική [$\Delta f=f(s')-f(s)<0$, όπου $s' \in N(s)$] - εδώ, υποθέτουμε ότι ο στόχος είναι η ελαχιστοποίηση της αντικειμενικής συνάρτησης). Η διαδικασία αναζήτησης συνεχίζεται μέχρι η τρέχουσα

λύση, s, να είναι τοπικά βέλτιστη, δηλαδή δεν υπάρχει καλύτερη λύση στην γειτονιά (∀s'∈N(s): f(s')≥f(s)).

Η αρχή του αλγορίθμου **SA** είναι απλή [44]: έναρξη από μια τυχαία λύση. Δεδομένης μιας λύσης s, επιλέξτε μια γειτονική λύση s' και υπολογίστε τη διαφορά των τιμών της αντικειμενικής συνάρτησης, Δf=f(s')-f(s). Εάν η τιμή της αντικειμενικής συνάρτησης βελτιώνεται (Δf<0), τότε αντικαταστήστε την τρέχουσα λύση με τη νέα. Εάν Δf≥0, τότε αποδέχεστε μια κίνηση με πιθανότητα $\mathrm{P}(\Delta f) = \mathrm{e}^{-\Delta \mathrm{f}/\mathrm{t}}$, όπου t είναι η τρέχουσα θερμοκρασία (Η σταθερά Boltzmann δεν απαιτείται κατά την εφαρμογή του αλγορίθμου σε συνδυαστικά προβλήματα).

Οι GA (genetic algorithms) λειτουργούν με κάποια ομάδα (P) (που ονομάζεται πληθυσμός) λύσεων (s1, s2, ..., si, ...) (που ονομάζονται άτομα) από το S. Αυτό είναι αρκετά διαφορετικό από τον παραπάνω δύο προσεγγίσεις, οι οποίες μπορούν να θεωρηθούν ως ευρετικές μέθοδοι μεμονωμένων λύσεων. Έτσι, κάθε άτομο (si) συνδέεται με κάποια καταλληλόλητα που αντιστοιχεί στον αντικειμενικό στόχο τιμή της συνάρτησης (f(si)). Στην περίπτωση της ελαχιστοποίησης προβλήματος, όσο μικρότερη είναι η τιμή της αντικειμενικής συνάρτησης, τόσο πιο κατάλληλο το άτομο και τόσο μεγαλύτερη είναι η πιθανότητα το άτομο να επιβιώσει στο διαδικασία εξέλιξης. Με την πάροδο πολλών γενεών, η καλύτερη προσαρμογή άτομα τείνουν να επικρατούν, ενώ τα λιγότερο κατάλληλα άτομα τείνουν να πεθαίνουν [46].

Οι αλγόριθμοι μυρμηγκιών προτάθηκαν για πρώτη φορά ως κάποιο σύστημα συνεργαζόμενων πρακτόρων (ένα σύστημα πολλαπλών πρακτόρων) για διάφορα προβλήματα βελτιστοποίησης. Οι αλγόριθμοι μυρμηγκιών εμπνεύστηκαν από την παρατήρηση πραγματικών αποικιών μυρμηγκιών. Ενώ περπατούν από τη φωλιά προς τις πηγές τροφής και αντίστροφα, τα μυρμήγκια εναποθέτουν στο έδαφος μια ουσία που ονομάζεται φερομόνη, σχηματίζοντας με αυτόν τον τρόπο μια φερομόνη ίχνος. Ο ρόλος αυτού του μονοπατιού είναι να καθοδηγεί τα μυρμήγκια προς την πηγή της τροφής (ή στη φωλιά). Έχει αποδειχθεί ότι η ποσότητα φερομόνης που αφήνει ένα μυρμήγκι εξαρτάται από την ποσότητα της τροφής που βρέθηκε. Είναι επίσης προφανές ότι όσο περισσότερα τα μυρμήγκια που φτάνουν στην πηγή της τροφής, τόσο ισχυρότερη η φερομόνη που αφήνει. Δεδομένου ότι οι συντομότερες διαδρομές από τη φωλιά προς τις πηγές τροφής θα ταξιδεύουν με μια υψηλότερη συχνότητα από ό,τι οι πιο μακρινές, η ποσότητα των φερομόνης θα αυξάνεται ταχύτερα στις μικρότερες διαδρομές (δηλ. η ποσότητα φερομόνης θα είναι σχετική με το μήκος της διαδρομής). Έτσι, όταν υπάρχουν πολλά μονοπάτια (από τη φωλιά προς τις πηγές τροφής), μια αποικία μυρμηγκιών μπορεί να εκμεταλλευτεί τα ίχνη φερομόνης (που αφήνουν τα μεμονωμένα μυρμήγκια) για να ανακαλύψει το συντομότερο μονοπάτι, δηλαδή τα

μυρμήγκια είναι σε θέση να βελτιστοποιήσουν τη διαδρομή τους μέσω της διαδικασίας που περιγράφεται παραπάνω.

Μια παρόμοια διαδικασία μπορεί να μεταφερθεί στη συνδυαστική βελτιστοποίηση: οι λύσεις ενός προβλήματος μπορούν να κατασκευαστεί χρησιμοποιώντας μια στατιστική (πληροφορίες) των λύσεων που έχουν κατασκευαστεί προηγουμένως. Αυτή η στατιστική παίζει το ρόλο του ίχνους φερομόνης και δίνει μεγαλύτερη βαρύτητα στις καλύτερες λύσεις. Ένα τέτοιο μοντέλο είναι σε θέση να κατασκευάσει λύσεις καλύτερης ποιότητας από ό,τι μια διαδικασία καθοδηγούμενη μόνο από αξιολογήσεις αντικειμενικών συναρτήσεων [46].

Οι αναλογίες των πραγματικών μυρμηγκιών και των συνδυαστικών βελτιστοποίησης είναι οι εξής: (α) η αναζήτηση των πραγματικών μυρμηγκιών περιοχή αναζήτησης αντιστοιχεί στο σύνολο των λύσεων του συνδυαστικής βελτιστοποίησης- (β) η ποσότητα των τροφής σε μια πηγή αντιστοιχεί στην αντικειμενική συνάρτηση (γ) τα ίχνη φερομόνης αντιστοιχούν σε κάποια (προσαρμοστική) μνήμη [46].

Η βασική ιδέα του **ILS** είναι να επιτύχουμε καλύτερα αποτελέσματα βελτιστοποίησης μέσω της ανακατασκευής (καταστροφής) μιας υφιστάμενης λύσης και μια ακόλουθη διαδικασία βελτίωσης (ανοικοδόμησης). Με την εφαρμογή αυτού του τύπου διαδικασίας με επαναληπτικό τρόπο επιδιώκεται η επίτευξη υψηλής ποιότητας λύσεις. Έτσι, στην πρώτη φάση της διαδικασίας, κάποιος ανακατασκευάζει (μεταλλάσσει) την υπάρχουσα λύση. Το σκεπτικό είναι ότι η συνέχιση της αναζήτησης από την ανακατασκευασμένη λύση μπορεί να επιτρέψει τη διαφυγή από ένα τοπικό βέλτιστο και να προσπαθήσει να βρει καλύτερες λύσεις. Στη δεύτερη φάση, προσπαθεί κανείς να βελτιώσει τη λύση που μόλις "καταστράφηκε" με τον καλύτερο δυνατό τρόπο. Ελπίζουμε ότι η νέα λύση είναι καλύτερη από τη λύση ή τις λύσεις που προέκυψαν στην προηγούμενη φάση ή στις προηγούμενες φάσεις της βελτίωσης (συνήθως, η βελτίωση είναι ένα πιο εξελιγμένο μέρος της μεθόδου) [46].

2.3.1.4 Άλλες μέθοδοι

Προηγμένες τεχνικές όπως βαθιά μάθηση (Deep Learning), θεωρία σύνθετων δικτύων (Complex Network Theory), θεωρία παιγνίων (Game Theory) συγκαταλέγονται στις άλλες μεθόδους επίλυσης μεταξύ άλλων. Σε αυτές τις προσεγγίσεις, αρχικά διατυπώνεται μοντέλο του συστήματος, στη συνέχεια ορίζονται οι συναρτήσεις σκοπού και τέλος επιβάλλονται περιορισμοί στις μεταβλητές που πρέπει να βελτιστοποιηθούν [43].

Στην παρούσα διπλωματική εργασία χρησιμοποιείται Βαθιά Ενισχυτική Μάθηση, η οποία αναλύεται περαιτέρω αργότερα.

Δεδομένου ότι η κατανομή πόρων είναι ένα NP-δύσκολο πρόβλημα, οι βέλτιστοι αλγόριθμοι μπορούν να εφαρμοστούν μόνο σε μικρής κλίμακας σύνολα δεδομένων. Αντίθετα, οι προσεγγιστικοί ή ευρετικοί αλγόριθμοι παρουσιάζουν χαμηλή υπολογιστική αποδοτικότητα και μειωμένη ακρίβεια στα αποτελέσματα σε σύγκριση με τη βέλτιστη λύση [42].

Τα τελευταία χρόνια, οι πάροχοι υπηρεσιών νέφους και οι χρήστες εργάζονται από κοινού για έναν αποδοτικό τρόπο διαχείρισης των υπολογιστικών πόρων. Από τη μία πλευρά, οι χρήστες θέλουν να λαμβάνουν την καλύτερη ποιότητα υπηρεσίας (QoS) με το ελάχιστο κόστος, ενώ οι πάροχοι υπολογιστικού νέφους θέλουν να αυξήσουν τα έσοδά τους.

Οι αποδοτικές στρατηγικές κατανομής είναι ζωτικής σημασίας τόσο για τους παρόχους νέφους όσο και για τους χρήστες. Διαφορετικές πολιτικές παροχής μπορούν να εφαρμοστούν ανάλογα με την κατάσταση του δικτύου υποδομής ή της τρέχουσας ζήτησης των χρηστών [47].

Γίνεται ξεκάθαρη πλέον πως η χρήση της μηχανικής μάθησης για την κατανομή πόρων σε δίκτυα Edge/Fog/Cloud είναι μια κατεύθυνση που αξίζει να ερευνηθεί.

2.3.2 Markov Decision Processes & Reinforcement Learning

Μέχρι τώρα εξετάσαμε σε γενικές γραμμές τις τεχνικές επιβλεπόμενης, μη-επιβλεπόμενης, ημι-επιβλεπόμενης και ενισχυτικής μάθησης. Ωστόσο, σε περιβάλλοντα όπου οι αποφάσεις πρέπει να ληφθούν διαδοχικά και σε πραγματικό χρόνο, όπως στην αυτόματη κλιμάκωση μικροϋπηρεσιών ή στην κατανομή φόρτου μεταξύ edge, fog και cloud το reinforcement learning παρέχει ένα ισχυρό πλαίσιο μοντελοποίησης.

Η ενισχυτική μάθηση (reinforcement learning) αποτελεί τεχνική που ανάγεται στις απαρχές της κυβερνητικής, της στατιστικής, της ψυχολογίας, των νευροεπιστημών και της επιστήμης των υπολογιστών. Προσφέρει μια ελκυστική προσέγγιση για τον προγραμματισμό πρακτόρων (agents) με χρήση αμοιβών και ποινών (rewards, penalties), χωρίς την ανάγκη να προσδιορίζεται ρητά η μέθοδος εκτέλεσης της εργασίας [48].

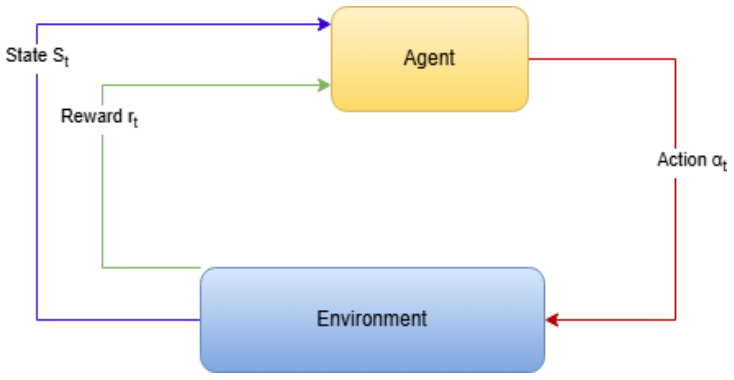
Σε αντίθεση με την επιβλεπόμενη μάθηση, όπου κάθε παράδειγμα εκπαίδευσης συνοδεύεται από την ακριβή σωστή έξοδο, η ενισχυτική μάθηση λαμβάνει πολύ

λιγότερη ανατροφοδότηση. Στην επιβλεπόμενη μάθηση, η απόδοση μετριέται εύκολα με βάση τον αριθμό των σωστών προβλέψεων, κάτι που αποδίδει ένα σαφές μέτρο ακρίβειας. Το κύριο ζήτημα είναι η εκμάθηση ενός μοντέλου που να γενικεύει καλά σε νέα, αθέατα δεδομένα. Στη μη επιβλεπόμενη μάθηση, η δυσκολία έγκειται στο να δημιουργηθεί μια χρήσιμη ομαδοποίηση ή διάσταση των δεδομένων έτσι ώστε να ανακύψουν χρήσιμες "κλάσεις" από μόνες τους.

Αντιθέτως, στην ενισχυτική μάθηση, η μόνη ανατροφοδότηση που υπάρχει είναι ένας μόνο αριθμητικός δείκτης ανταμοιβής, αντί για σαφείς "σωστές απαντήσεις". Ο δείκτης αυτός είναι αξιολογητικός, όχι καθοδηγητικός, δεν δείχνει ποια ακριβώς ενέργεια ήταν σωστή. Ως εκ τούτου, οι αλγόριθμοι RL χρειάζεται να αφιερώσουν επιπλέον προσπάθεια στην ερμηνεία αυτού του περιορισμένου σήματος, ώστε να αξιολογούν και να βελτιώνουν τη συμπεριφορά τους με την πάροδο του χρόνου [49].

Η μάθηση μέσω ενισχύσεων εστιάζει στην εκμάθηση συμπεριφοράς από έναν agent, ο οποίος αλληλοεπιδράει με ένα δυναμικό περιβάλλον μέσω δοκιμών και σφαλμάτων. Στο κλασικό μοντέλο reinforcement learning, ο agent συνδέεται με το περιβάλλον του μέσω αντίληψης και δράσης. Σε κάθε βήμα:

Ο agent λαμβάνει μια είσοδο (input) που παρέχει πληροφορίες για την τρέχουσα κατάσταση (state, s) του περιβάλλοντος. Στη συνέχεια επιλέγει μια ενέργεια (action, a). Η ενέργεια αυτή μεταβάλλει την κατάσταση του περιβάλλοντος και επιστρέφεται στον agent ένα ενισχυτικό σήμα (reward, r) που εκφράζει την αξία αυτής της μετάβασης. Η συμπεριφορά (B) του agent επιδιώκει τη μέγιστη σωρευτική ανταμοιβή με την πάροδο του χρόνου. Μέσω συστηματικής επανάληψης και δοκιμών, και με χρήση εξειδικευμένων αλγορίθμων, ο agent μαθαίνει να επιλέγει καλύτερες ενέργειες [48].



Εικόνα 7:Reinforcement Learning Agent-Environment Interaction

Για τη μαθηματική διατύπωση του προβλήματος λήψης αποφάσεων στο RL, θα χρησιμοποιήσουμε το πλαίσιο Μαρκοβιανών Διαδικασιών Λήψεως Αποφάσεων (Markov Decision Process - MDP). Από τον αυστηρό ορισμό των MDPs μπορούν να αναπαρασταθούν οι καταστάσεις, οι ενέργειες, οι συναρτήσεις μετάβασης και οι συναρτήσεις ανταμοιβής. Ύστερα, θα αναλύσουμε πως το reinforcement learning προσπαθεί να μεγιστοποιήσει την ανταμοιβή χρησιμοποιώντας την κατάλληλη πολιτική.

2.3.2.1 Markov Decision Processes

Συχνά είναι δυνατόν να αναπαραστήσουμε τη συμπεριφορά ενός συστήματος, φυσικού ή μαθηματικού, περιγράφοντας όλες τις διαφορετικές καταστάσεις που μπορεί να καταλάβει (ενδεχομένως άπειρες) και υποδεικνύοντας τον τρόπο με τον οποίο κινείται μεταξύ αυτών των καταστάσεων. Το σύστημα που μοντελοποιείται θεωρείται ότι καταλαμβάνει μία και μόνο μία κατάσταση σε κάθε στιγμή και η εξέλιξή του αναπαρίσταται με μεταβάσεις από κατάσταση σε κατάσταση. Αυτές οι μεταβάσεις υποτίθεται ότι συμβαίνουν στιγμιαία, η πραγματική μετακίνηση από μια κατάσταση σε μια άλλη στην επόμενη κατάσταση καταναλώνει μηδενικό χρόνο. Εάν η μελλοντική εξέλιξη του συστήματος εξαρτάται μόνο από την τρέχουσα κατάσταση και όχι από την προηγούμενη ιστορία του (memory-less), τότε το σύστημα μπορεί να αναπαρασταθεί από μια διαδικασία Markov. Ακόμα και όταν το σύστημα δεν διαθέτει αυτή την ιδιότητα Markov ρητά, είναι συχνά δυνατό να κατασκευαστεί μια αντίστοιχη σιωπηρή αναπαράσταση. Παραδείγματα χρήσης των διαδικασιών Markov μπορούν να βρεθούν στις βιολογικές, φυσικές και κοινωνικές επιστήμες, καθώς και φυσικά σε προβλήματα μηχανικής [50].

Ένα MDP ορίζεται λοιπόν από ένα σύνολο καταστάσεων, ένα σύνολο ενεργειών, τους κανόνες μετάβασης μεταξύ τους και μια συνάρτηση ανταμοιβής. Σε συνέχεια, θα αναλύσουμε κάθε στοιχείο ξεχωριστά.

Κατάσταση

Ο χώρος καταστάσεων του περιβάλλοντος S ορίζεται ως το πεπερασμένο σύνολο:

$$S = \{s_1, s_2, \dots, s_N\}$$

μεγέθους N, δηλαδή |S| = N. Μια κατάσταση S_i αποτελεί μοναδικό χαρακτηρισμό όλων των κρίσιμων στοιχείων που καθορίζουν την τρέχουσα κατάσταση του προβλήματος που μοντελοποιείται.

Αν το πρόβλημα είναι το σκάκι για παράδειγμα, η περιγραφή της θέσης όλων των πιονιών είναι μια κατάσταση [49].

Ενέργειες

Ο χώρος ενεργειών A ορίζεται ως ένα πεπερασμένο σύνολο:

$$A = \{\alpha_1, \alpha_2, \dots, \alpha_\kappa\}$$

Μεγέθους |A| = K. Οι ενέργειες χρησιμοποιούνται για τον έλεγχο της κατάστασης του συστήματος. Στην εκάστοτε κατάσταση $s \in S$, το σύνολο των διαθέσιμων ενεργειών για την κατάσταση αυτή συμβολίζεται ως $\mathrm{A(s)}$ όπου $\mathrm{A(s)} \subseteq \mathrm{A}$. Σε ορισμένα μοντέλα και καταστάσεις, δεν είναι διαθέσιμο πάντα όλο το σύνολο κάποιων ενεργειών, ορισμένες ενέργειες θεωρούνται μη επιτρεπτές [49].

Επιστρέφοντας στο προηγούμενο παράδειγμα, ένα λευκό πιόνι δε μπορεί να εκτελέσει κάποια ενέργεια που θα το έκανε να τελειώσει την κίνησή του σε σημείο της σκακιέρας που υπάρχει ήδη άλλο λευκό πιόνι.

Συνάρτηση Μετάβασης

Εφαρμόζοντας την ενέργεια α∈A σε μια κατάσταση $s \in S$, τότε το σύστημα πραγματοποιεί μια μετάβαση από την s σε μια νέα κατάσταση $s' \in S$ με βάση μια κατανομή πιθανοτήτων στο σύνολο των πιθανών μεταβάσεων. Η συνάρτηση μετάβασης ορίζεται ως $T : S \times A \times S \rightarrow [0,1]$

Το σύστημα θα μεταβεί δηλαδή στην νέα κατάσταση s’ αν κάνει ενέργεια α στην κατάσταση s με πιθανότητα $T(s,a,s')$

Οι προδιαγραφές της συνάρτησης είναι:

1. $0 \leq T(s,a,s') \leq 1$ για κάθε $s, s' \in S$ και $a \in A$
2. Για κάθε s και a, το άθροισμα των πιθανοτήτων μετάβασης είναι 1.

Εάν μια ενέργεια a δεν επιτρέπεται σε κατάσταση s, μπορούμε να ορίσουμε T(s,a,s’)=0 για όλα τα s’.

Χρονίζουμε τα βήματα με έναν διακριτό δείκτη t = [1,2,…]. Έτσι το s_t είναι η κατάσταση στο χρόνο t και το s_{t+1} η κατάσταση στο χρόνο t+1 [49].

Ένα σύστημα είναι Markovian εάν η πιθανότητα της επόμενης κατάστασης εξαρτάται μόνο από την τρέχουσα κατάσταση και ενέργεια, όχι από ολόκληρο το ιστορικό, ισχύει δηλαδή:

$$P(s_{t+1} \mid s_t, \alpha_t, s_{t-1}, \alpha_{t-1}, \dots) = P(s_{t+1} \mid s_t, \alpha_t) = T(s_t, \alpha_t s_{t+1}).$$

Αυτό σημαίνει ότι μόνο το s_t είναι απαραίτητο για να προβλέψουμε την κατανομή του s_{t+1}. Όταν για πρόβλεψη χρειάζονται οι τελευταίες k καταστάσεις μιλάμε για k-Markov μοντέλα, ενώ το συνηθισμένο MDP είναι 1-Markov. Κάθε k-Markov μοντέλο μπορεί να μετατραπεί σε ισοδύναμο 1-Markov MDP. Η ιδιότητα Markov αποτελεί διαχωριστική γραμμή μεταξύ των MDP και πιο γενικών σχημάτων όπως τα Partially observable Markov Decision Process (POMDP) [49].

Συνάρτηση Ανταμοιβής

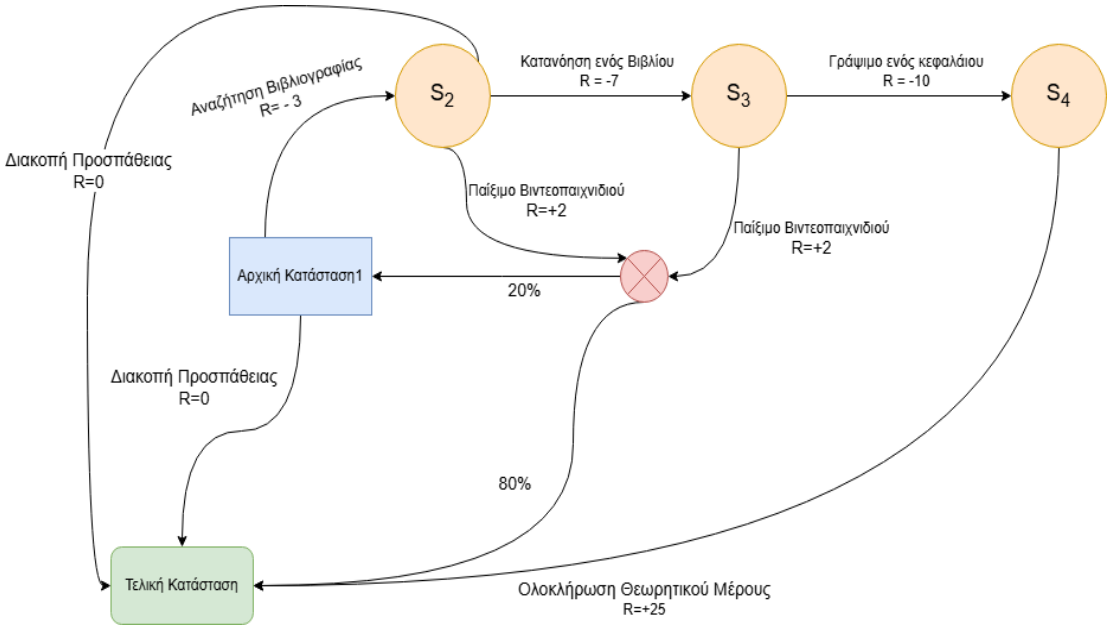
Η συνάρτηση ανταμοιβής είναι ένα σημαντικό μέρος του MDP που καθορίζει σιωπηρά την στόχο της μάθησης. Η συνάρτηση ανταμοιβής καθορίζει αμοιβές για να είναι το μοντέλο στην κατάσταση αυτή η να κάνει κάποια ενέργεια σε κάποια κατάσταση. Ορίζεται είτε ως $R : S \times A \times S \rightarrow \mathbb{R}$ είτε ως $R : S \times A \rightarrow \mathbb{R}$, ανάλογα εάν οι ανταμοιβές δίνονται για συγκεκριμένες μεταβάσεις από κατάσταση σε κατάσταση ή αν δίνονται απλώς για να εκτελείται η ενέργεια α στην κατάσταση s [49].

Παραδείγματος χάρη, στο σκάκι μπορούμε να ανταμείψουμε θετικά τις νικητήριες καταστάσεις στη σκακιέρα (αυτές για τις οποίες ο αντίπαλος βασιλιάς βρίσκεται σε ματ) και να ανταμείψουμε αρνητικά αυτές για τις οποίες χάνει (βρίσκεται σε ματ). Οι υπόλοιπες θέσεις μπορούν να έχουν μηδενική ανταμοιβή ή ελαφρώς θετική/αρνητική. Ο στόχος του πράκτορα είναι λοιπόν να πραγματοποιήσει ενέργειες που οδηγούν τη σκακιέρα σε καταστάσεις που είναι καλύτερες για αυτόν με απώτερο σκοπό να κερδίσει το παιχνίδι.

Η Μαρκοβιανή Διαδικασία λήψης αποφάσεων

Ορισμός: Μια διαδικασία απόφασης Markov είναι μια πλειάδα (tuple) $\langle S, A, T, R \rangle$ στην οποία S είναι ένα πεπερασμένο σύνολο καταστάσεων, A ένα πεπερασμένο σύνολο ενεργειών, T μια συνάρτηση μετάβασης που ορίζεται ως T : S×A×S → [0,1] και R μια συνάρτηση ανταμοιβής που ορίζεται ως R : S×A×S → $\mathbb{R}$.

Συχνά τα MDP απεικονίζονται ως ένα γράφημα μετάβασης καταστάσεων όπου οι κόμβοι αντιστοιχούν σε καταστάσεις και οι κατευθυνόμενες ακμές υποδηλώνουν τις μεταβάσεις [49].



Εικόνα 8: Παράδειγμα MDP

Πολιτικές MDP

Δεδομένου ενός MDP $\langle S, A, T, R \rangle$ μια πολιτική είναι μια υπολογίσιμη συνάρτηση που εξάγει για κάθε κατάσταση s ∈ S μια ενέργεια a ∈ A(s). Η πολιτική μπορεί να είναι ντετερμινιστική ή στοχαστική.

Μια ντετερμινιστική πολιτική π είναι μια συνάρτηση που ορίζεται ως π : S → A. Μια στοχαστική πολιτική ως π : S × A → [0,1] έτσι ώστε για κάθε κατάσταση s ∈ S, να ισχύει ότι π(s,a) ≥ 0 και $\sum_{a \in A} \pi(s, a) = 1$.

Η εφαρμογή μιας πολιτικής σε ένα MDP γίνεται με τον ακόλουθο τρόπο: Πρώτον, δημιουργείται μια αρχική κατάσταση s_0 από την αρχική κατανομή καταστάσεων I. Στη συνέχεια, η πολιτική π προτείνει την ενέργεια α_0 = π (s_0) και η ενέργεια αυτή εκτελείται. Με βάση τη συνάρτηση μετάβασης T και τη συνάρτηση ανταμοιβής R, γίνεται μετάβαση στην κατάσταση s_1, με πιθανότητα T(s_0, α_0, s_1) και λαμβάνεται ανταμοιβή r_0 = R(s_0, α_0, s_1) . Αυτή η διαδικασία συνεχίζεται, παράγοντας s_0, α_0, r_0 s_1, α_1, r_1 και τα λοιπά.

Εάν η εργασία είναι επεισοδιακή, η διαδικασία ολοκληρώνεται σε κατάσταση $s_{στόχος}$ και επανεκκινείται σε μια νέα κατάσταση που αντλείται από το I. Εάν η εργασία είναι συνεχής, η ακολουθία καταστάσεων μπορεί να επεκταθεί επ' αόριστον.

Η πολιτική είναι μέρος του πράκτορα και στόχος της είναι να ελέγχει το περιβάλλον που μοντελοποιείται ως MDP [49].

Ιδανικές Πολιτικές – Ιδανικές συναρτήσεις αξίας

Καθώς ο σκοπός του agent είναι να λάβει θετικές αμοιβές, ωστόσο, υπάρχουν διάφοροι τρόποι να ληφθεί υπόψη το μέλλον στον τρόπο με τον οποίο να συμπεριφερθούμε τώρα. Υπάρχουν βασικά τρία μοντέλα βελτιστοποίησης στο MDP, τα οποία αρκούν για να καλύψουν τις περισσότερες προσεγγίσεις της βιβλιογραφίας. Τα τρία αυτά μοντέλα προσεγγίζουν διαφορετικά το πώς πρέπει να λαμβάνουμε το μέλλον υπόψιν και τί πρέπει να μεγιστοποιηθεί: i) έχουμε πεπερασμένο ορίζοντα μήκους h και μεγιστοποιούμε την αναμενόμενη αμοιβή σε h βήματα, ii) έχουμε άπειρο ορίζοντα και έκπτωση με παράγοντα έκπτωσης $0 \leq \gamma \leq 1$ για τις μελλοντικές αμοιβές (έτσι ώστε το άθροισμα παραμένει πεπερασμένο) και τελικά iii) εξετάζουμε την αναμενόμενη μέση ανταμοιβή και προσπαθούμε να επιτύχουμε μεγιστοποίηση της μακροπρόθεσμης μέσης ανταμοιβής.

$$i) E\left[\sum_{t=0}^{h} r_t\right] \quad \text{ii)} E\left[\sum_{t=0}^{\infty} \gamma^t r_t\right] \quad \text{iii)} \lim_{h\to\infty} E\left[\frac{1}{h}\sum_{t=0}^{h} r_t\right]$$

[49]

Στο πλαίσιο της εργασίας αυτής, θα ασχοληθούμε κυρίως με το 2ο μοντέλο.

Οι συναρτήσεις αξίας ορίζουν μια μερική σειρά πάνω στις πολιτικές, και αντιπροσωπεύουν μια προσέγγιση του πόσο καλό είναι για τον πράκτορα να είναι σε μια συγκεκριμένη κατάσταση ή να κάνει μια ενέργεια στην κατάσταση αυτή. Η αξία της κατάστασης s υπό πολιτική π συμβολίζεται $V^{\pi}(s)$ και είναι η αναμενόμενη αμοιβή όταν αρχίζουμε στην κατάσταση s και ακολουθούμε πολιτική π. Χρησιμοποιώντας το μοντέλο άπειρου ορίζοντα με έκπτωση, μπορούμε να γράψουμε

$$\mathrm{V}^{\pi}(\mathrm{s}) \;=\; \mathrm{E}_{\pi}\{\; \textstyle\sum_{k=0}^{\infty} \gamma^{k}\, r_{t+k} \;\; |\, \mathrm{s}_{\mathrm{t}} \;=\; \mathrm{s}\,\}$$

Μια θεμελιώδης ιδιότητα των συναρτήσεων αξίας είναι ότι ικανοποιούν ορισμένες αναδρομικές ιδιότητες. Για οποιαδήποτε πολιτική π και οποιαδήποτε κατάσταση s η έκφραση της παραπάνω εξίσωσης μπορεί αναδρομικά να οριστεί ως προς τη λεγόμενη εξίσωση Bellman (1957):

$$V^{\pi}(s) = E_{\pi}\{r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \cdots | s_t = t\}$$
$$= E_{\pi}\{r_t + \gamma V^{\pi}(s_{t+1}) | s_t = s\}\backslash$$

$$= \sum_{s'} T(s, \pi(s), s') \big(R(s, a, s') + \gamma V^{\pi}(s') \big)$$

Μια πολιτική π ορίζεται ότι είναι καλύτερη ή ίση με μια πολιτική π′ εάν η αναμενόμενη απόδοση της είναι μεγαλύτερη ή ίση με εκείνη της π' για όλες τις καταστάσεις. Με άλλα λόγια, π ≥ π' αν και μόνο εάν $v_{\pi(s)} \geq v_{\pi'(s)}$ για κάθε $s \in S$ [49] [51].

Υπάρχει πάντα τουλάχιστον μία πολιτική που είναι καλύτερη ή ίση με όλες τις άλλες πολιτικές και ονομάζεται βέλτιστη πολιτική, η οποία λαμβάνει τη μεγαλύτερη αμοιβή. Ισχύει για την πολιτική αυτή $V_{(S)}^{\pi^*} \geq V_{(s)}^{\pi}$ για κάθε $s \in S$ και όλες τις πολιτικές π. Αν και μπορεί να υπάρχουν περισσότερες από μία, συμβολίζουμε όλες τις βέλτιστες πολιτικές με π_*. Μοιράζονται την ίδια συνάρτηση κατάστασης-αξίας, που ονομάζεται βέλτιστη συνάρτηση κατάστασης-αξίας, συμβολίζεται με V* και ορίζεται ως εξής :

$V^*(s) = \max_{\pi} v_{\pi}(s) = \max_{\pi} E(\sum_{t=0}^{\infty} \gamma^t r_t) = \max_{a \in A} \sum_{s' \in S} T(s, a, s') \big(R(s, a, s') + \gamma V^*(s') \big)$

[48][49][51]

Συγκεκριμένα, παραπάνω αναφερόμαστε στη βέλτιστη αξία μιας κατάστασης ως το αναμενόμενο discounted άπειρο άθροισμα της ανταμοιβής που θα κερδίσει ο πράκτορας αν ξεκινήσει σε αυτή την κατάσταση και εκτελέσει τη βέλτιστη πολιτική [48].

Για να επιλεχθεί η βέλτιστη ενέργεια, δεδομένης της βέλτιστης συνάρτησης αξίας V*, μπορεί να εφαρμοστεί ο ακόλουθος κανόνας:

$$\pi^*(s) = \arg\max_{a} \sum_{s' \in S} T(s, a, s') \big(R(s, a, s') + \gamma V^*(s') \big)$$

Αυτή η πολιτική ονομάζεται greedy (άπληστη) καθώς επιλέγει άπληστα την καλύτερη ενέργεια χρησιμοποιώντας την συνάρτηση αξίας V. Χρήσιμες είναι επίσης οι συναρτήσεις Q, ιδιαίτερα σε model-free λύσεις που θα εξετάσουμε αργότερα, καθώς αυτές εκπαιδεύονται αντί για συναρτήσεις V.

Η ανάλογή βέλτιστη τιμή κατάστασης-ενέργειας είναι:

$$Q^*(s, a) = \sum_{s'} T(s, a, s') \left(R(s, a, s') + \gamma \max_{a'} Q^*(s', a') \right)$$

Η σχέση συναρτήσεων Q και V είναι η εξής :

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

$$V^*(s) = \max_{a} Q^*(s, a) [49]$$

Λύνοντας ένα Μοντέλο MDP

Το να λύσουμε ένα MDP σημαίνει να βρούμε την βέλτιστη πολιτική π*. Υπάρχουν αρκετοί αλγόριθμοι και τεχνικές που χρησιμοποιούνται για να το επιτύχουμε αυτό, όπως δυναμικός προγραμματισμός (Dynamic Programming – DP) και ενισχυτική μάθηση (Reinforcement Learning – RL).

Ο δυναμικός προγραμματισμός χρησιμοποιείται όταν έχουμε ένα ορθό μοντέλο και προσπαθούμε να προσδιορίσουμε την βέλτιστη πολιτική, ενώ όταν δε είναι γνωστό κάποιο τέτοιο μοντέλο εκ των προτέρων χρησιμοποιείται ενισχυτική μάθηση. Οι τεχνικές επίλυσης κατηγοριοποιούνται λοιπόν σε Model-Based και Model-Free [48][49].

2.3.2.2 Model-Based Τεχνικές Επίλυσης: Value Iteration, Policy Iteration

Ο όρος δυναμικός προγραμματισμός (ΔΠ) αναφέρεται σε μια συλλογή αλγορίθμων που μπορεί να χρησιμοποιηθεί για τον υπολογισμό βέλτιστων πολιτικών δεδομένου ενός τέλειου μοντέλου του περιβάλλοντος ως διαδικασία απόφασης Markov (MDP) [51].

Ο δυναμικός προγραμματισμός θεωρείται ευρέως ο μόνος εφικτός τρόπος επίλυσης γενικών προβλημάτων στοχαστικού βέλτιστου ελέγχου. Πάσχει από αυτό που ο Bellman ονόμασε "κατάρα της διαστατικότητας", που σημαίνει ότι οι υπολογιστικές του απαιτήσεις αυξάνονται εκθετικά με τον αριθμό των μεταβλητών κατάστασης, αλλά είναι εξακολουθεί να είναι πολύ πιο αποτελεσματικός και πιο ευρέως εφαρμόσιμος από οποιαδήποτε άλλη γενική μέθοδο [51].

Όταν αυτοί οι μέθοδοι συζητούνται στη βιβλιογραφία όλες υποθέτουν ένα τυπικό MDP $\langle S, A, T, R \rangle$ όπου τα σύνολα καταστάσεων και ενεργειών είναι πεπερασμένα και διακριτά, ώστε να μπορούν να αποθηκευτούν σε πίνακες.

Αν και οι ιδέες του DP μπορούν να εφαρμόζονται σε προβλήματα με συνεχείς χώρους καταστάσεων και δράσεων, ακριβείς λύσεις είναι δυνατές μόνο σε ειδικές περιπτώσεις. Ένας συνήθης τρόπος για την απόκτηση προσεγγιστικών λύσεις για εργασίες με συνεχείς καταστάσεις και δράσεις είναι η κβαντοποίηση των χώρων καταστάσεως και δράσης και στη συνέχεια η εφαρμογή μεθόδων DP πεπερασμένων καταστάσεων [51].

Δύο βασικές μέθοδοι ΔΠ είναι η επανάληψη αξίας (Value Iteration) και η επανάληψη πολιτικής (Policy Iteration).

2.3.2.3 Model-Free Τεχνικές Επίλυσης

Η ενισχυτική μάθηση ασχολείται κυρίως με το πώς θα επιτευχθεί η βέλτιστη πολιτική όταν ένα τέτοιο μοντέλο δεν είναι γνωστό εκ των προτέρων. Ο πράκτορας πρέπει να αλληλοεπιδράσει άμεσα με το περιβάλλον του για να αποκτήσει πληροφορίες οι οποίες, με τη βοήθεια ενός κατάλληλου αλγορίθμου, μπορούν να υποστούν επεξεργασία για να παραχθεί μια βέλτιστη πολιτική [48].

Η ενισχυτική μάθηση προσθέτει στα MDPs έμφαση στην προσέγγιση και την ελλιπή πληροφόρηση, καθώς και την ανάγκη για δειγματοληψία και εξερεύνηση. Η έλλειψη ενός μοντέλου δημιουργεί την ανάγκη δειγματοληψίας του MDP για τη συλλογή στατιστικής γνώσης σχετικά με αυτό το άγνωστο μοντέλο. Υπάρχουν πολλές τεχνικές RL χωρίς μοντέλο, οι οποίες διερευνούν το περιβάλλον κάνοντας ενέργειες, εκτιμώντας έτσι το ίδιο είδος τιμών κατάστασης και συναρτήσεων αξίας κατάστασης-δράσης με τις τεχνικές που βασίζονται σε μοντέλο [49].

Στα περιβάλλοντα χωρίς μοντέλο (model-free), υπάρχουν δύο βασικές επιλογές:

Η πρώτη είναι να μάθουμε πρώτα το μοντέλο μετάβασης και ανταμοιβής από την αλληλεπίδραση με το περιβάλλον. Στη συνέχεια, όταν το μοντέλο είναι επαρκώς σωστό, εφαρμόζονται όλες οι μέθοδοι DP, όπως value iteration ή policy iteration. Αυτός ο τύπος μάθησης ονομάζεται έμμεση ή model-based RL.

Η δεύτερη επιλογή, που ονομάζεται άμεση ενισχυτική μάθηση, είναι να προχωρήσουμε κατευθείαν στην εκτίμηση των τιμών για τις ενέργειες, χωρίς καν να εκτιμήσουμε το μοντέλο του MDP.

Επιπλέον, υπάρχουν και μικτές μορφές μεταξύ αυτών των δύο [49].

On Policy vs Off-Policy

Οι μέθοδοι onpolicy επιχειρούν να αξιολογήσουν ή να βελτιώσουν την πολιτική που χρησιμοποιείται για την αποφάσεις, ενώ οι μέθοδοι εκτός πολιτικής αξιολογούν ή βελτιώνουν μια πολιτική διαφορετική από εκείνη που χρησιμοποιείται για τη δημιουργία των δεδομένων [51].

Στις μεθόδους εκτός πολιτικής, η μάθηση είναι απλή όταν χρησιμοποιούνται τροχιές που δεν λαμβάνονται απαραίτητα υπό την τρέχουσα πολιτική, αλλά από μια διαφορετική πολιτική συμπεριφοράς β (s, a). Σε αυτές τις περιπτώσεις, η επανάληψη εμπειριών επιτρέπει την επαναχρησιμοποίηση δειγμάτων από μια διαφορετική

πολιτική συμπεριφοράς. Αντίθετα, οι μέθοδοι που βασίζονται στην πολιτική συνήθως εισάγουν μια προκατάληψη (bias) όταν χρησιμοποιούνται με έναν replay buffer, καθώς οι τροχιές συνήθως δεν λαμβάνονται αποκλειστικά βάσει της τρέχουσας πολιτικής π [52].

Εξερεύνηση - Exploration

Μια σημαντική πτυχή των αλγορίθμων χωρίς μοντέλα είναι ότι υπάρχει ανάγκη για εξερεύνηση. Επειδή το μοντέλο είναι άγνωστο, ο εκπαιδευόμενος πρέπει να δοκιμάσει διάφορες ενέργειες για να δει τα αποτελέσματά τους. Ένας αλγόριθμος μάθησης πρέπει να επιτύχει μια ισορροπία μεταξύ εξερεύνησης και εκμετάλλευσης, δηλαδή για να κερδίσει μεγάλη ανταμοιβή πρέπει να εκμεταλλευτεί τις τρέχουσες γνώσεις του σχετικά με τις καλές ενέργειες, αν και μερικές φορές πρέπει να δοκιμάσει διαφορετικές ενέργειες για να εξερευνήσει το περιβάλλον για την εύρεση πιθανών καλύτερων ενεργειών. Η πιο βασική στρατηγική εξερεύνησης είναι η πολιτική ε-greedy, όπου ο μαθητευόμενος παίρνει την τωρινή καλύτερη του ενέργεια με πιθανότητα (1-ε) και μια τυχαία άλλη ενέργεια με πιθανότητα ε [49].

Μέθοδοι Monte Carlo & Temporal Difference Learning

Οι μέθοδοι Monte Carlo βασίζονται αποκλειστικά στην εμπειρία, σε ακολουθίες δειγμάτων καταστάσεων, ενεργειών και ανταμοιβών που προέρχονται από πραγματική ή προσομοιωμένη αλληλεπίδραση με το περιβάλλον. Η μάθηση από πραγματική εμπειρία είναι εντυπωσιακή, καθώς δεν απαιτεί προηγούμενη γνώση των δυναμικών του περιβάλλοντος, αλλά παρόλα αυτά μπορεί να επιτύχει βέλτιστη συμπεριφορά.

Η μάθηση από προσομοιωμένη εμπειρία είναι επίσης πολύ ισχυρή. Σε αυτή την περίπτωση απαιτείται μοντέλο, αλλά αρκεί να μπορεί να παράγει δειγματοληπτικές μεταβάσεις και όχι να παρέχει τις πλήρεις πιθανότητες για όλες τις δυνατές μεταβάσεις όπως απαιτεί ο δυναμικός προγραμματισμός. Σε αρκετές περιπτώσεις, είναι εύκολο να δημιουργηθούν δείγματα με βάση τις επιθυμητές πιθανότητες, αλλά πρακτικά αδύνατο να προσδιοριστούν ρητά όλες οι πιθανότητες [51].

Αντί συναρτήσεων αξίας με τη χρήση δυναμικών μεθόδων προγραμματισμού, οι μέθοδοι Monte Carlo εκτιμούν τις αναμενόμενες αμοιβές από μια κατάσταση με τη μέση τιμή της απόδοσης από πολλαπλές εκκινήσεις μιας πολιτικής [53]. Οι μέθοδοι Monte Carlo εκπαιδεύουν τις συναρτήσεις αξίας και τις βέλτιστες πολιτικές μέσα από

επεισόδια-δείγματα (sample episodes), παρέχοντας τουλάχιστον αρκετά πλεονεκτήματα έναντι των μεθόδων δυναμικού προγραμματισμού (DP):

Άμεση εκμάθηση από εμπειρία → Επιτρέπουν την εκμάθηση της βέλτιστης συμπεριφοράς απευθείας από την αλληλεπίδραση με το περιβάλλον, χωρίς να απαιτείται κανένα μοντέλο για τις δυναμικές του περιβάλλοντος.

Ευελιξία στη χρήση προσομοιώσεων → Μπορούν να χρησιμοποιηθούν σε προσομοιωμένα ή δειγματοληπτικά μοντέλα, καθώς στις περισσότερες εφαρμογές είναι εύκολο να δημιουργηθούν δείγματα επεισοδίων, ακόμη κι αν η κατασκευή ενός ρητού μοντέλου μεταβάσεων είναι δύσκολη ή ανέφικτη.

Εστιασμένη αξιολόγηση → Επιτρέπουν τη στοχευμένη και αποδοτική αξιολόγηση ενός περιορισμένου υποσυνόλου καταστάσεων, δηλαδή μπορεί να εκτιμηθεί με ακρίβεια μια περιοχή ενδιαφέροντος χωρίς να χρειάζεται εκτενής υπολογισμός για ολόκληρο το χώρο καταστάσεων.

Ανθεκτικότητα σε παραβιάσεις της ιδιότητας Markov → Τα Monte Carlo συστήματα είναι λιγότερο ευάλωτα σε περιπτώσεις όπου δεν ισχύει αυστηρά η Markov ιδιότητα, επειδή δεν ενημερώνουν τις τιμές τους βάσει εκτιμήσεων για διαδοχικές καταστάσεις [51].

Temporal-Difference Learning

Αν έπρεπε να προσδιορίσει κανείς μια ιδέα ως κεντρική και νέα για την ενισχυτική μάθηση, αυτή θα ήταν αναμφίβολα η μάθηση της χρονικής διαφοράς (TD). Η μάθηση TD είναι ένας συνδυασμός ιδεών Monte Carlo και ιδεών δυναμικού προγραμματισμού (DP). Όπως οι μέθοδοι Monte Carlo, έτσι και οι μέθοδοι TD μπορούν να μάθουν απευθείας από την ακατέργαστη εμπειρία χωρίς μοντέλο της δυναμικής του περιβάλλοντος. Όπως και οι μέθοδοι DP, οι μέθοδοι TD ενημερώνουν τις εκτιμήσεις που βασίζονται εν μέρει σε άλλες εκτιμήσεις που έχουν μάθει, χωρίς να περιμένουν ένα τελικό αποτέλεσμα [51]. Δεν απαιτούνται πλήρεις σαρώσεις σε ολόκληρο το χώρο καταστάσεων - μόνο κατά μήκος έμπειρων διαδρομών ενημερώνονται οι τιμές και οι ενημερώσεις πραγματοποιούνται μετά από κάθε βήμα [49].

Ένα πρόβλημα που εμφανίζεται είναι πως είναι δύσκολο να εκτιμηθεί η χρησιμότητα κάποιας δράσης, εάν τα πραγματικά αποτελέσματα της συγκεκριμένης δράσης μπορούν να γίνουν αντιληπτά μόνο πολύ αργότερα.

Μια δυνατότητα είναι να περιμένουμε μέχρι το τέλος (ενός επεισοδίου π.χ.) και να τιμωρήσουμε ή επιβραβεύσουμε συγκεκριμένες ενέργειες που λήφθηκαν. Αυτή η

μέθοδος εμφανίζει προβλήματα εάν δε γνωρίζουμε από πριν εάν θα υπάρξει κάποιο "τέλος" και επίσης καταναλώνει πολλή μνήμη.

Αντ' αυτού μπορούμε να χρησιμοποιήσουμε μηχανισμούς παρόμοιους με αυτούς του value iteration προκειμένου να προσαρμόσουμε την εκτιμώμενη αξία μιας κατάστασης με βάση την άμεση ανταμοιβή και την εκτιμώμενη (προ-εξοφλημένη) αξία της επόμενης κατάστασης. Αυτό ονομάζεται γενικά μάθηση χρονικής διαφοράς, η οποία είναι ένας γενικός μηχανισμός που διέπει τις μεθόδους χωρίς μοντέλα. Η κύρια διαφορά με τους κανόνες ενημέρωσης για τις προσεγγίσεις DP είναι ότι η συνάρτηση μετάβασης T και η συνάρτηση ανταμοιβής R δεν μπορούν πλέον να εμφανίζονται στους κανόνες ενημέρωσης. Η γενική κατηγορία αλγορίθμων που αλληλοεπιδρούν με το περιβάλλον και ενημερώνουν τις εκτιμήσεις τους μετά από κάθε εμπειρία ονομάζεται online RL [48][49].

Μάθηση TD (0)

Τόσο οι μέθοδοι TD όσο και οι μέθοδοι Monte Carlo χρησιμοποιούν την εμπειρία για την επίλυση του προβλήματος πρόβλεψης. Δεδομένης κάποιας εμπειρίας ακολουθώντας μια πολιτική π, και οι δύο μέθοδοι ενημερώνουν την εκτίμησή τους v για v_π για τις μη τερματικές καταστάσεις S_t που εμφανίζονται σε αυτή την εμπειρία.

Χονδρικά μιλώντας, οι μέθοδοι Monte Carlo περιμένουν μέχρι να γίνει γνωστή η απόδοση μετά την επίσκεψη και στη συνέχεια χρησιμοποιούν αυτή την απόδοση ως στόχο για το V_{S_t}. Μια απλή μέθοδος Monte Carlo για κάθε επίσκεψη, κατάλληλη για μη σταθερά περιβάλλοντα, είναι η εξής:

$$V(S_t) \leftarrow V(S_t) + \alpha[G_t - V(S_t)]$$

όπου G_t είναι η πραγματική απόδοση μετά τη χρονική στιγμή t, και $\alpha \in [0,1]$ είναι μια σταθερή παράμετρος βήματος γνωστή και ως ρυθμός εκμάθησης, που καθορίζει κατά πόσο οι τιμές ενημερώνονται.

Ενώ οι μέθοδοι Monte Carlo πρέπει να περιμένουν μέχρι το τέλος του επεισοδίου για να προσδιορίσουν την αύξηση του V_{S_t} (μόνο τότε είναι γνωστό το G_t), οι μέθοδοι TD χρειάζεται να περιμένουν μόνο μέχρι το επόμενο χρονικό βήμα. Τη χρονική στιγμή $t+1$ σχηματίζουν αμέσως έναν στόχο και κάνουν μια χρήσιμη ενημέρωση χρησιμοποιώντας την παρατηρούμενη ανταμοιβή R_{t+1} και την εκτίμηση V_{t+1} . Η απλούστερη μέθοδος TD, γνωστή ως TD(0), επιλύει το πρόβλημα πρόβλεψης, δηλαδή εκτιμά το Vπ για κάποια πολιτική π, με έναν online, σταδιακό τρόπο. Το T D(0) μπορεί να χρησιμοποιηθεί για την αξιολόγηση μιας πολιτικής και λειτουργεί μέσω της χρήσης

του ακόλουθου κανόνα ενημέρωσης:

$$V(S_t) \leftarrow V(S_t) + \alpha[R_{t+1} + \gamma V(S_{t+1}) - V(S_t)]$$

Ή αλλιώς:

$$V_{k+1}(s) \coloneqq V_k(s) + \alpha\big(r + \gamma V_k(s') - V_k(s)\big)$$

Στην πραγματικότητα, ο στόχος για την ενημέρωση Monte Carlo είναι G_t, ενώ ο στόχος για την ενημέρωση TD είναι $R_{t+1} + \gamma V(S_{t+1})$. Η TD(0) είναι μέθοδος bootstrapping, καθώς ο στόχος της είναι αυτός καθ' αυτός.

Ο TD(0) μπορεί να γραφτεί ως ψευδοκώδικας λοιπόν ως εξής:

Input: η πολιτική π που θα αξιολογηθεί

Αρχικοποίηση V_S αυθέραιτα, π.χ. V_S = 0, για κάθε κατάσταση s

Επανάλαβε (για κάθε επεισόδιο) :

Αρχικοποίηση S

Επανάλαβε (για κάθε βήμα του επεισοδίου):

A ← ενέργεια που δίνει η π για την S

Κάνε ενέργεια A; παρατήρησε αμοιβή R και επόμενη κατάσταση, S'

$V(S) \leftarrow V(S) + \alpha[R + \gamma V(S') - V(S)]$

S ← S'

Μέχρι το S να είναι τερματικό

[51]

Q-Learning

Μία από τις βασικές μεθόδους για την εκτίμηση των συναρτήσεων Q-value χωρίς την ανάγκη μοντέλου είναι ο αλγόριθμος Q-learning. Η βασική ιδέα στο Q-learning είναι η σταδιακή εκτίμηση των τιμών Q για τις ενέργειες, με βάση την ανατροφοδότηση (δηλαδή τις ανταμοιβές) και τη συνάρτηση Q-value του πράκτορα. Ο κανόνας ενημέρωσης είναι μια παραλλαγή της ιδέας της μάθησης TD, χρησιμοποιώντας τις τιμές Q και έναν ενσωματωμένο τελεστή μεγίστου πάνω στις τιμές Q της επόμενης κατάστασης, προκειμένου να ενημερωθεί η Q_t σε Q_{t+1}:

$$Q_{k+1}(s_t, a_t) \leftarrow Q_k(s_t, a_t) + \alpha\left(r_t + \gamma \max_a Q_k\,(s_{t+1}, a) - Q_k(s_t, a_t)\right)$$

Σε αυτή την περίπτωση, η μαθημένη συνάρτηση δράσης-αξίας, Q, προσεγγίζει άμεσα την q_*, τη βέλτιστη συνάρτηση αξίας δράσης, ανεξάρτητα από την ακολουθούμενη πολιτική. Αυτό απλοποιεί δραματικά την ανάλυση του αλγορίθμου και επιτρέπει την έγκαιρη αποδείξεις σύγκλισης. Η πολιτική εξακολουθεί να έχει επίδραση στο ότι καθορίζει ποια ζευγάρια κατάστασης-δράσης επισκέπτονται και ενημερώνονται. Ωστόσο, το μόνο που απαιτείται για σωστή σύγκλιση είναι ότι όλα τα ζεύγη συνεχίζουν να ενημερώνονται [49][51].

Αυτό σημαίνει ότι, αν και το ζήτημα της εξερεύνησης-εκμετάλλευσης πρέπει να αντιμετωπιστεί, οι λεπτομέρειες της στρατηγικής εξερεύνησης δεν θα επηρεάσουν τη σύγκλιση της αλγορίθμου μάθησης. Για τους λόγους αυτούς, το Q-learning είναι ο πιο δημοφιλής και φαίνεται να είναι ο πιο αποτελεσματικός αλγόριθμος χωρίς μοντέλο. Ωστόσο, δεν αντιμετωπίζει κανένα από τα ζητήματα που σχετίζονται με τη γενίκευση σε μεγάλες καταστάσεις και/ή δράσεις χώρων. Επιπλέον, μπορεί να συγκλίνει αρκετά αργά σε μια καλή πολιτική [48].

Στην πράξη, μια γενική απόδειξη της σύγκλισης στη βέλτιστη συνάρτηση αξίας είναι διαθέσιμη (Watkins και Dayan, 1992) με την προϋπόθεση ότι ισχύουν οι εξής συνθήκες:

1. τα ζεύγη κατάστασης - δράσης αναπαρίστανται διακριτά, και
2. γίνεται δειγματοληψία όλων των ενεργειών επανειλημμένα σε όλες τις καταστάσεις (γεγονός που εξασφαλίζει επαρκή εξερεύνηση, επομένως δεν απαιτείται πρόσβαση στο μοντέλο μετάβασης).

Αυτή η απλή ρύθμιση είναι συχνά ανεφάρμοστη λόγω των υψηλών διαστάσεων (ενδεχομένως συνεχούς) χώρου καταστάσεων-δράσεων. Συνεπώς, απαιτείται μια παραμετροποιημένη συνάρτηση αξίας Q(s, a; θ), όπου το θ αναφέρεται σε κάποιες παραμέτρους που καθορίζουν τις τιμές Q [54].

Ο ψευδοκώδικας του q-learning φαίνεται παρακάτω.

Αρχικοποίηση Q(s, a), ∀s ∈ S, a ∈ A(s), αυθαίρετα, και Q(terminal-state, ·) = 0

Επανάλαβε (για κάθε επεισόδιο) :

Αρχικοποίηση κατάστασης S

Επανάληψη (για κάθε βήμα του επεισοδίου):

Επιλογή A από το S χρησιμοποιώντας πολιτική που προκύπτει από το Q (π.χ. ε-greedy)

Κάνε ενέργεια A, παρατήρησε αμοιβή R και επόμενη κατάσταση, S'

$$Q(S,A) \leftarrow Q(S,A) + \alpha\left[R + \gamma \max_a Q\,(S', a) - Q(S,A)\right]$$

$S \leftarrow S'$;

έως ότου το S είναι τερματικό

[51]

2.3.3 Deep Reinforcement Learning

Οι προηγούμενες προσεγγίσεις, ενώ είναι επιτυχημένοι αλγόριθμοι μάθησης μέσω εμπειρίας, δεν είχαν επεκτασιμότητα και ήταν εγγενώς περιορισμένες σε αρκετά προβλήματα χαμηλών διαστάσεων. Αυτοί οι περιορισμοί υφίστανται επειδή οι RL αλγόριθμοι μοιράζονται τα ίδια προβλήματα πολυπλοκότητας με άλλους αλγορίθμους: πολυπλοκότητα μνήμης, υπολογιστική πολυπλοκότητα και, στην περίπτωση των αλγορίθμων μηχανικής μάθησης, πολυπλοκότητα δειγμάτων. Αυτό που έχουμε παρακολουθήσει τα τελευταία χρόνια - την άνοδο της βαθιάς μάθησης ή αλλιώς Deep Learning - μας έδωσε νέα εργαλεία για την αντιμετώπιση αυτών των προβλημάτων [53].

Η βαθιά μάθηση είναι ένας νέος τομέας στη μελέτη της μηχανικής μάθησης. Το κίνητρό της έγκειται στη δημιουργία και προσομοίωση του ανθρώπινου εγκεφάλου από τεχνητά νευρωνικά δίκτυα, τα οποία μιμούνται τον μηχανισμό του ανθρώπινου εγκεφάλου για την ερμηνεία των δεδομένων, σε τομείς της όρασης, της επεξεργασίας εικόνας, της ομιλίας και του ήχου των υπολογιστών [55].

Η έννοια της βαθιάς μάθησης προέρχεται από την έρευνα των τεχνητών νευρωνικών δικτύων (artificial neural networks - ANN). Το μαθηματικό μοντέλο των ANN αποτελείται από στρώματα νευρώνων. Τα ANNs είναι κατανεμημένοι παράλληλοι αλγόριθμοι επεξεργασίας πληροφοριών, οι οποίοι χρησιμοποιούνται για την προσομοίωση της συμπεριφοράς των ζωικών νευρώνων. Ανάλογα με την πολυπλοκότητα του συστήματος, τα ANN υλοποιούν το σκοπό της επεξεργασίας πληροφοριών προσαρμόζοντας τη διασύνδεση μεταξύ μεγάλου αριθμού εσωτερικών κόμβων.

Η λεγόμενη βαθιά μάθηση είναι το νευρωνικό δίκτυο που αποτελείται από νευρώνες πολλαπλών στρωμάτων για την προσέγγιση της λειτουργίας της μηχανικής μάθησης. Η δομή της βαθιάς μάθησης είναι το πολυστρωματικό perceptron με πολλαπλά κρυφά στρώματα. Συνδυάζοντας χαρακτηριστικά χαμηλού επιπέδου για να σχηματίσουν πιο αφηρημένες αναπαραστάσεις υψηλού επιπέδου κατηγοριών ή χαρακτηριστικών, η βαθιά μάθηση μπορεί να ανακαλύψει τα κατανεμημένα χαρακτηριστικά των δεδομένων [55].

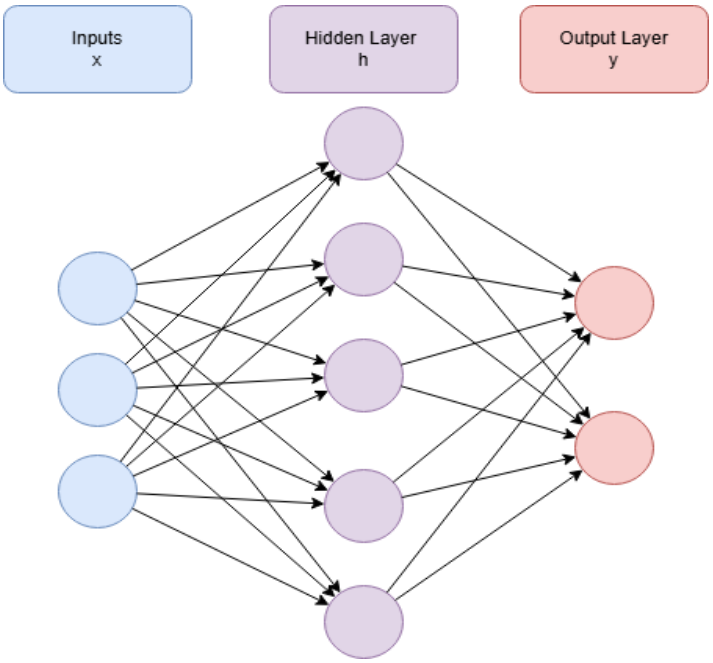
Κατ' αρχάς, ας περιγράψουμε ένα πολύ απλό νευρωνικό δίκτυο με ένα πλήρως συνδεδεμένο κρυφό στρώμα (όπως στην εικόνα 9). Στο πρώτο στρώμα δίνεται η είσοδος τιμές (δηλαδή τα χαρακτηριστικά εισόδου) x με τη μορφή διανύσματος στήλης μεγέθους n_x $(n_x \in N)$. Οι τιμές του επόμενου κρυφού στρώματος είναι ένας μετασχηματισμός αυτών των τιμών με μια μη γραμμική παραμετρική συνάρτηση, η οποία είναι ένας πίνακας πολλαπλασιασμός με W1 μεγέθους $n_h \times n_x$ $(n_h \in N)$, και έναν όρο προκατάληψης (bias) b_1 μεγέθους n_h ακολουθούμενη από έναν μη γραμμικό μετασχηματισμό:

$$h = A(W_1 \cdot x + b_1),$$

όπου A είναι η συνάρτηση ενεργοποίησης. Αυτή η μη γραμμική συνάρτηση ενεργοποίησης είναι αυτό που κάνει το μετασχηματισμό σε κάθε επίπεδο μη γραμμικό, το οποίο τελικά παρέχει την εκφραστικότητα του νευρωνικού δικτύου. Το κρυφό στρώμα h μεγέθους n_h μπορεί εν συνεχεία να μετασχηματιστεί σε άλλα σετ τιμών μέχρι τον τελευταίο μετασχηματισμό που μας δίνει τις τιμές εξόδου y. Στη συγκεκριμένη περίπτωση:

$$y = (W_2 \cdot h + b_2),$$

where W_2 is of size $n_y \times n_h$ and b_2 is of size $n_y(n_y \in N)$. [54]



Εικόνα 9: Παράδειγμα απλού νευρωνικού δικτύου με ένα πλήρως συνδεδεμένο κρυφό στρώμα

Η ενισχυτική μάθηση και η βαθιά μάθηση είναι δύο πολύ διαφορετικοί αλγόριθμοι της μηχανικής μάθησης, αλλά ο αλγόριθμος της βαθιάς μάθησης μπορεί να

χρησιμοποιηθεί για την υλοποίηση της ενισχυτικής μάθησης, η οποία αποτελεί τη βαθιά ενισχυτική μάθηση (Deep Reinforcement Learning - DRL).

Η ενισχυτική μάθηση ορίζει την στόχο της βελτιστοποίησης, ενώ η βαθιά μάθηση δίνει τον μηχανισμό λειτουργίας, ο οποίος είναι η προσέγγιση για τον χαρακτηρισμό του προβλήματος και ο τρόπος επίλυσης του προβλήματος. Ως εκ τούτου, η βαθιά ενισχυτική μάθηση έχει ένα είδος ικανότητας να επιλύει πολλά πολύπλοκα προβλήματα. Έτσι, η βαθιά ενισχυτική μάθηση μπορεί να γίνει κατανοητή ως μέρος της ενισχυτικής μάθησης που επιτυγχάνεται με τη βαθιά μάθηση. Στην ουσία, η DRL μπορεί να θεωρηθεί ως μάθηση και σκεπτόμενος αλγόριθμος.

Στις μεθόδους βαθιάς ενισχυτικής μάθησης χρησιμοποιούμε βαθιά νευρωνικά δίκτυα για να προσεγγίσουμε οποιαδήποτε από τις ακόλουθες συνιστώσες της ενίσχυσης μάθησης: συνάρτηση αξίας, $V_{(s,\theta)}$ ή $Q_{(s,a;\theta)}$, πολιτική π (a| s; θ), και μοντέλο (κατάσταση μετάβαση και ανταμοιβή). Όπου οι παράμετροι θ είναι τα βάρη στα βαθιά νευρωνικά δίκτυα [55].

Παραδείγματος χάρη, το deep learning μπορεί να εφαρμοστεί στις value-based μεθόδους (πχ. Q-learning) που στοχεύουν στην κατασκευή μιας συνάρτησης αξίας, η οποία στη συνέχεια μας επιτρέπει να ορίσουμε μια πολιτική. Ο κύριος ρόλος των νευρωνικών δικτύων είναι η εξαγωγή χαρακτηριστικών από εξαιρετικά δομημένα δεδομένα. Ως εκ τούτου, η συνάρτηση Q μπορεί να αναπαρασταθεί με νευρωνικό δίκτυο, με την κατάσταση και τη δράση ως είσοδο και την αντίστοιχη τιμή Q (Q-value) ως έξοδο. Σε γενικές γραμμές, πρόκειται στην πραγματικότητα για ένα κλασικό νευρωνικό δίκτυο με συνελικτικό τρόπο, με τρία συνελικτικά στρώματα και ακολουθούμενα από δύο πλήρως συνδεδεμένα στρώματα [54][55].

2.3.3.1 Policy Gradient Μέθοδοι για βαθιά ενισχυτική μάθηση

Θα επικεντρωθούμε τώρα σε μια συγκεκριμένη οικογένεια αλγορίθμων ενισχυτικής μάθησης που χρησιμοποιούν μεθόδους κλίσης πολιτικής (Policy Gradient Methods). Αυτές οι μέθοδοι βελτιστοποιούν έναν στόχο απόδοσης (συνήθως την αναμενόμενη αθροιστική ανταμοιβή) βρίσκοντας μια καλή πολιτική (π.χ. μια παραμετροποιημένη πολιτική νευρωνικού δικτύου όταν μιλάμε για βαθιά ενισχυτική μάθηση) χάρη σε παραλλαγές της στοχαστικής κλίσης ανόδου σε σχέση με τις παραμέτρους της πολιτικής. Οι μέθοδοι κλίσης πολιτικής ανήκουν σε μια ευρύτερη κατηγορία μεθόδων βασισμένων στην πολιτική που περιλαμβάνει, μεταξύ άλλων, στρατηγικές εξέλιξης. Αυτές οι μέθοδοι χρησιμοποιούν ένα σήμα μάθησης που προέρχεται από τη

δειγματοληψία των παραμέτρων πολιτικής και το σύνολο των πολιτικών αναπτύσσεται προς πολιτικές που επιτυγχάνουν καλύτερες αποδόσεις [54].

Οι μέθοδοι κλίσης πολιτικής λειτουργούν με τον υπολογισμό ενός εκτιμητή της πολιτικής κλίσης και τη σύνδεσή του σε έναν αλγόριθμο στοχαστικής κλίσης ανόδου. Ο πιο συχνά χρησιμοποιούμενος εκτιμητής κλίσης έχει την μορφή

$$\hat{g} = \widehat{E_t}\left[\nabla_\theta \log \pi_\theta\,(\,a_t \mid s_t\,)\widehat{A_t}\right]$$

όπου πθ είναι μια στοχαστική πολιτική και $\widehat{A_t}$ είναι ένας εκτιμητής της συνάρτησης πλεονεκτήματος στο χρονικό βήμα t.

Εδώ, η προσδοκία $E_t[\ldots]$ υποδηλώνει τον εμπειρικό μέσο όρο για μια πεπερασμένη δέσμη δειγμάτων, σε έναν αλγόριθμο που εναλλάσσεται μεταξύ δειγματοληψίας και βελτιστοποίησης [56].

Υπάρχει πληθώρα μεθόδων Policy Gradient, όπως στοχαστικές, ντετερμινιστικές, actor-critic και natural.

Οι φυσικές (natural) κλίσεις πολιτικής είναι εμπνευσμένες από την ιδέα των φυσικών κλίσεων για τις ενημερώσεις της πολιτικής. Οι μέθοδοι κλίσης φυσικής πολιτικής χρησιμοποιούν την πιο απότομη κατεύθυνση που δίνεται από τη μετρική της πληροφορίας Fisher, η οποία χρησιμοποιεί την πολλαπλότητα της συνάρτησης στόχου. Στην απλούστερη μορφή της πιο απότομης ανόδου για μια συνάρτηση στόχου $J(w)$, η ενημέρωση είναι της μορφής $\Delta w \propto \nabla_w J(w)$. Αυτό σημαίνει ότι η ενημέρωση ακολουθεί την κατεύθυνση που μεγιστοποιεί το $\big(J(w) - J(w + \Delta w)\big)$ υπό περιορισμό $|\Delta w|_2$.

Η φυσική κλίση ανόδου για τη βελτίωση της πολιτικής π_ω δίνεται από τη σχέση

$$\Delta w \propto F_w^{-1} \nabla_w V^{\pi^w}(\cdot),$$

όπου $F_{\ w}$ είναι ο πίνακας πληροφοριών Fisher που δίνεται από τη σχέση

$$F_w = E_{\pi_w}\left[\nabla_w \log \pi_w\,(s,\cdot)\big(\nabla_w \log \pi_w\,(s,\cdot)\big)^T\right].$$

Το μειονέκτημα των φυσικών κλίσεων είναι ότι, στην περίπτωση των νευρωνικών δικτύων και του μεγάλου αριθμού παραμέτρων τους, είναι συνήθως ανέφικτο να υπολογιστεί, να αντιστραφεί και να αποθηκευτεί ο πίνακας πληροφοριών Fisher. Αυτός είναι ο λόγος για τον οποίο οι φυσικές κλίσεις πολιτικής συνήθως δεν χρησιμοποιούνται στην πράξη για βαθιά RL - ωστόσο, έχουν βρεθεί εναλλακτικές λύσεις εμπνευσμένες από αυτή την ιδέα [54].

2.3.3.2 Trust Region Optimization

Ως τροποποίηση της μεθόδου φυσικής κλίσης, οι μέθοδοι βελτιστοποίησης πολιτικής που βασίζονται σε μια περιοχή εμπιστοσύνης, στοχεύουν στη βελτίωση της πολιτικής ενώ την μεταβάλλουν με ελεγχόμενο τρόπο. Αυτές οι μέθοδοι βελτιστοποίησης πολιτικής που βασίζονται σε περιορισμούς επικεντρώνονται στον περιορισμό των αλλαγών σε μια πολιτική, χρησιμοποιώντας την KL απόκλιση μεταξύ των κατανομών δράσης. Περιορίζοντας το μέγεθος της ενημέρωσης της πολιτικής, οι μέθοδοι περιοχής εμπιστοσύνης περιορίζουν επίσης τις αλλαγές στις κατανομές κατάστασης, εξασφαλίζοντας βελτιώσεις στην πολιτική [54].

Έστω $r_t(\theta)$ που υποδηλώνει τον λόγο πιθανοτήτων με:

$r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}$, έτσι ώστε $r(\theta_{\text{old}}) = 1$.

Ο αλγόριθμος TRPO μεγιστοποιεί έναν "υποκατάστατο" στόχο.

$$L^{CPI}(\theta) = \widehat{E_t}\left[\frac{\pi_\theta(a_t \mid s_t)}{\pi_{\theta_{\text{old}}}(a_t \mid s_t)}\widehat{A_t}\right] = \widehat{E_t}\left[r_t(\theta)\widehat{A_t}\right].$$

[56]

Ο αλγόριθμος Proximal Policy Optimization (PPO) είναι μια παραλλαγή του αλγορίθμου TRPO, η οποία διατυπώνει τον περιορισμό ως μια ποινή ή έναν στόχο αποκοπής, αντί να χρησιμοποιεί τον περιορισμό KL. Σε αντίθεση με τον TRPO, ο PPO εξετάζει την τροποποίηση της συνάρτησης στόχου για την επιβολή ποινής σε αλλαγές στην πολιτική που σπρώχνουν το $r_t(w) = \frac{\pi_w + \Delta w(s,a)}{\pi_w(s,a)}$ μακριά από το 1.

Ο στόχος αποκοπής που μεγιστοποιεί ο PPO δίνεται από τη σχέση

$$L^{CLIP}(\theta) = \widehat{E_t}\left[\min\left(r_t(\theta)\widehat{A_t},\ \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\widehat{A_t}\right)\right]$$

Όπου $\varepsilon \in R$ μια υπερπαράμετρος, π.χ. ε = 0.2. Αυτή η συνάρτηση στόχου περικλείει τον λόγο πιθανοτήτων για να περιορίσει τις μεταβολές του r_t στο διάστημα $[1-\varepsilon, 1+\varepsilon]$ [54][56].

Το κίνητρο του αλγορίθμου αυτού για τον στόχο είναι το εξής: Ο πρώτος όρος μέσα στο min είναι L^{CPI} . Ο δεύτερος όρος, $\text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\widehat{A_t}$, τροποποιεί το υποκατάστατο στόχο με το ψαλίδισμα του λόγου πιθανοτήτων, το οποίο αφαιρεί το κίνητρο για μετακίνηση του r_t εκτός του διαστήματος [1 - ε, 1 + ε]. Τέλος, λαμβάνουμε

το ελάχιστο του περικομμένου και του μη περικομμένου στόχου, έτσι ώστε ο τελικός στόχος είναι ένα κατώτερο όριο (δηλ. ένα απαισιόδοξο όριο) του μη περικομμένου στόχου. Με αυτό σχήμα, αγνοούμε την αλλαγή του λόγου πιθανοτήτων μόνο όταν θα έκανε τον στόχο να βελτιωθεί, και τη συμπεριλαμβάνουμε όταν θα χειροτέρευε τον στόχο.

Κεφάλαιο 3: Σχετική Εργασία

Η αναζήτηση αλγορίθμων και μεθόδων για την ανάθεση πόρων σε δίκτυα ομίχλης-νέφους είναι ένα ανοιχτό ζήτημα που απασχολεί την επιστημονική κοινότητα. Τα συστήματα που δημιουργούνται σε αυτόν τον τομέα της έρευνας των δικτύων μπορούν να μειώσουν καθυστέρηση και κόστος και να ανοίξουν νέους ορίζοντες στο cloud computing.

Η τοπολογία του δικτύου, οι τεχνολογίες που χρησιμοποιούνται καθώς και η προσέγγιση για τη λύση του προβλήματος είναι ποικίλες, χρησιμοποιώντας πληθώρα τεχνικών από τα μαθηματικά και την επιστήμη των υπολογιστών.

Στην παρούσα ενότητα συνοψίζονται σχετικές εργασίες ανάθεσης microservices και διαχείρισης πόρων σε edge-fog-cloud υποδομές ως πηγή έμπνευσης, πλαισιώνοντας ταυτόχρονα τη δικιά μας εργασία συγκρίνοντας τις μεθοδολογίες, τα περιβάλλοντα λειτουργίας, τα metrics απόδοσης και τα κύρια αποτελέσματα των ερευνών.

Στο [57] εξετάζεται η χρήση Q-Learning και βαθιάς ενισχυτικής μάθησης με αλγόριθμο DQN για κατανομή πόρων και offloading σε MEC (mobile edge computing). Προκειμένου να ελαχιστοποιηθεί το συνολικό κόστος του εξεταζόμενου συστήματος MEC, βελτιστοποιείται από κοινού η απόφαση εκφόρτωσης και η κατανομή των υπολογιστικών πόρων.

Οι Wang et al. [58] προτείνουν έναν αλγόριθμο κατανεμημένης ενισχυτικής μάθησης (distributed reinforcement learning), για να βοηθήσουν την ενσωμάτωση του SFC στο δίκτυο edge computing. Προσομοιώνουν προγραμματιστικά ένα μεσαίου μεγέθους edge δίκτυο υπολογιστών που αποτελείται από 100 κόμβους και περίπου 600 συνδέσεις. Εξετάζουν μακροχρόνια τον μέσο όρο κέρδους, ρυθμό αποδοχής SFC, και λόγο κέρδους/κόστους του αλγορίθμου τους, ο οποίος παρουσιάζει σημαντικά πλεονεκτήματα.

Το άρθρο των Wang et al. [59] προτείνει έναν αλγόριθμο βαθιάς ενισχυτικής μάθησης (Deep Reinforcement Learning - DRL), βασισμένο στο Proximal Policy Optimization (PPO), για την αποτελεσματική δρομολόγηση IoT εφαρμογών σε ετερογενή περιβάλλοντα edge και fog computing. Η μεθοδολογία DRLIS που προτείνουν αξιοποιεί ένα μοντέλο κόστους με βάση το Directed Acyclic Graph (DAG) για την βελτιστοποίηση χρόνου απόκρισης και εξισορρόπησης φορτίου. Τα πειράματα που περιγράφονται έδειξαν πως το DRLIS επιτυγχάνει βελτιώσεις επιδόσεων έως και 49%, 60% και 55% όσον αφορά την εξισορρόπηση φορτίου, το χρόνο απόκρισης και την σταθμισμένο

κόστος, αντίστοιχα όταν συγκρίνεται με παραδοσιακούς metaheuristic και άλλους αλγορίθμους ενίσχυσης (π.χ. NSGA2, NSGA3, Q-learning).

Οι Zhang et al. [60] προτείνουν για την εκφόρτωση υπολογισμών και το σύστημα κατανομής πόρων μια χρήση του αλγορίθμου PPO βασισμένη σε Device-to-Device (D2D). Κατασκευάζουν ένα ρεαλιστικό περιβάλλον δικτύου Multi-Access Edge computing και αναπτύσσουν ένα μοντέλο διαδικασίας απόφασης Markov που ελαχιστοποιεί την απώλεια χρόνου και την κατανάλωση ενέργειας. Η ενσωμάτωση ενός πλαισίου εκφόρτωσης με βάση την επικοινωνία D2D επιτρέπει τη συνεργατική εκφόρτωση εργασιών μεταξύ των τελικών συσκευών και των διακομιστών MEC, βελτιώνοντας τόσο τη χρήση των πόρων όσο και την υπολογιστική αποδοτικότητα. Το μοντέλο MDP επιλύεται χρησιμοποιώντας τον αλγόριθμο PPO σε βαθιά ενισχυτική μάθηση, για να προκύψει μια βέλτιστη πολιτική για την εκφόρτωση και την κατανομή των πόρων.

Οι Mansoureh Zare et al. [61] επικεντρώνονται στην τοποθέτηση subordinate υπηρεσιών εφαρμογών IoT σε διακομιστές ομίχλης (Service Placement Problem – SPP). Ο στόχος τους είναι η επίτευξη αποτελεσματικού προγραμματισμού όσον αφορά τη μείωση του κόστους και των παραγόντων καθυστέρησης υπό περιορισμούς προθεσμίας και πόρων. Για την επίτευξη αυτών των στόχων, προτείνουν ένα σύστημα βασισμένο σε DRL, το οποίο ονομάζεται A3C-SPP. Το A3C-SPP είναι εξοπλισμένο με τον αλγόριθμο A3C ως μια νέα προσέγγιση βασισμένη σε DRL για την αποτελεσματική επίλυση του SPP σε ετερογενή περιβάλλοντα ομίχλης. Σύμφωνα με τα αποτελέσματα των πειραμάτων τους, το προτεινόμενο σχήμα μπορεί να κάνει deploy υπηρεσίες IoT με κατανεμημένο και αυτόνομο τρόπο σε κατάλληλες υπηρεσίες ομίχλης. Επιπλέον, στο A3C-SPP χρησιμοποιείται με την πάροδο του χρόνου μια τεχνική δυναμικής εξαγωγής κατανομής πόρων, με την οποία εξοικονομούνται περισσότεροι πόροι για τη διαχείριση μελλοντικών αιτημάτων.

Τέλος οι Jose Santos et al. [47] μοντελοποιούν ένα δίκτυο fog και χρησιμοποιούν πράκτορες RL για να βρουν τις κατάλληλες αποφάσεις κατανομής, με στόχο τη μείωση του συνολικού κόστους του συστήματος. Ανέπτυξαν ένα περιβάλλον που ονομάζεται gym-fog για να γεφυρωθεί το χάσμα μεταξύ των λύσεων που βασίζονται στην ILP και των αλγορίθμων RL. Κατάφεραν να διδάξουν τους πράκτορες RL πώς να κατανέμουν υπηρεσίες στο Fog Computing, επιδεικνύοντας έως και 60% ποσοστό αποδοχής των αιτημάτων των χρηστών ενώ μείωσαν τα κόστη.

Ο σκοπός της παρούσας εργασίας είναι να αναδείξουμε την αποτελεσματικότητα του PPO για την ανάθεση microservices σε edge-fog-cloud δίκτυα, ελαχιστοποιώντας το

κόστος και την κατανάλωση ενέργειας σε σχέση με τη βέλτιστη λύση που υπολογίζουμε μέσω ILP.

Άρθρο	**Προσέγγιση λύσης**	**Τοπολογία & Τεχνολογίες**	**Κύρια Metrics**
[57]	Q-Learning, DQN	Mobile Edge Computing	Συνολικό κόστος
[58]	Distributed Reinforcement Learning	Edge (100 κόμβοι)	Μέσος ορος κέρδους, Ρυθμός αποδοχής SFC, λόγος κέρδος/κόστος
[59]	PPO-based DRL, DRLIS	Edge-Fog (ετερογενές)	Εξισορρόπηση φορτίου, χρόνος απόκρισης, σταθμισμένο κόστος
[60]	PPO + D2D	Multi-Access Edge	Καθυστέρηση, κατανάλωση ενέργειας
[61]	A3C	IoT-Fog	Κόστος, Καθυστέρηση
[47]	Q-Learning, MILP oracle	Edge-Fog-Cloud + LPWAN	Κόστος, Αποδοχή αιτημάτων

Πίνακας 3.1: Σύνοψη της Σχετικής Εργασίας

Μέρος ΙΙ: ΠΡΑΚΤΙΚΟ ΜΕΡΟΣ

Σε αυτό το μέρος, πραγματευόμαστε την χρήση βαθιάς ενισχυτικής μάθησης και συγκεκριμένα του αλγορίθμου Proximal Policy Optimization για να αναθέσουμε αλυσίδες μικροϋπηρεσιών σε ένα edge-fog-cloud δίκτυο.

Περιγράφονται αναλυτικά όλες οι εσωτερικές λειτουργίες και παραδοχές του συστήματος που προσομοιώθηκε. Παρουσιάζεται το δίκτυο edge-fog-cloud καθώς και η λύση γραμμικού προγραμματισμού μικτών ακεραίων για το πρόβλημα της ανάθεσης microservices σε αυτό και στη συνέχεια αναλύεται η δικιά μας λύση βαθιάς ενισχυτικής μάθησης.

Αρχικά περιγράφουμε το πρόβλημα και το περιβάλλον του δικτύου, καθώς και την λύση MILP.

4.1 Γενικά

Στη σημερινή εποχή της ραγδαίας τεχνολογικής εξέλιξης, η καθημερινότητά μας μετασχηματίζεται συνεχώς και οι νέες τεχνολογίες ενσωματώνονται σε όλο και περισσότερες πτυχές της ζωής μας. Από την επικοινωνία και την εργασία έως τις αγορές και την ψυχαγωγία, οι ψηφιακές λύσεις παίζουν πλέον καθοριστικό ρόλο. Μέσα σε αυτό το πλαίσιο, η ταχεία και οικονομική διαχείριση των υπηρεσιών αναδεικνύεται ως μια από τις πλέον σημαντικές προκλήσεις της σύγχρονης εποχής.

Η τεχνολογία βαθιάς ενισχυτικής μάθησης προσφέρει μια καινοτόμο προσέγγιση για την αντιμετώπιση αυτού του ζητήματος, παρέχοντας μηχανισμούς που μπορούν να ανταπεξέλθουν στο δυναμικό φορτίο σε δίκτυα όπου είναι αδύνατος ο ακριβής υπολογισμός της βέλτιστης κατάστασης.

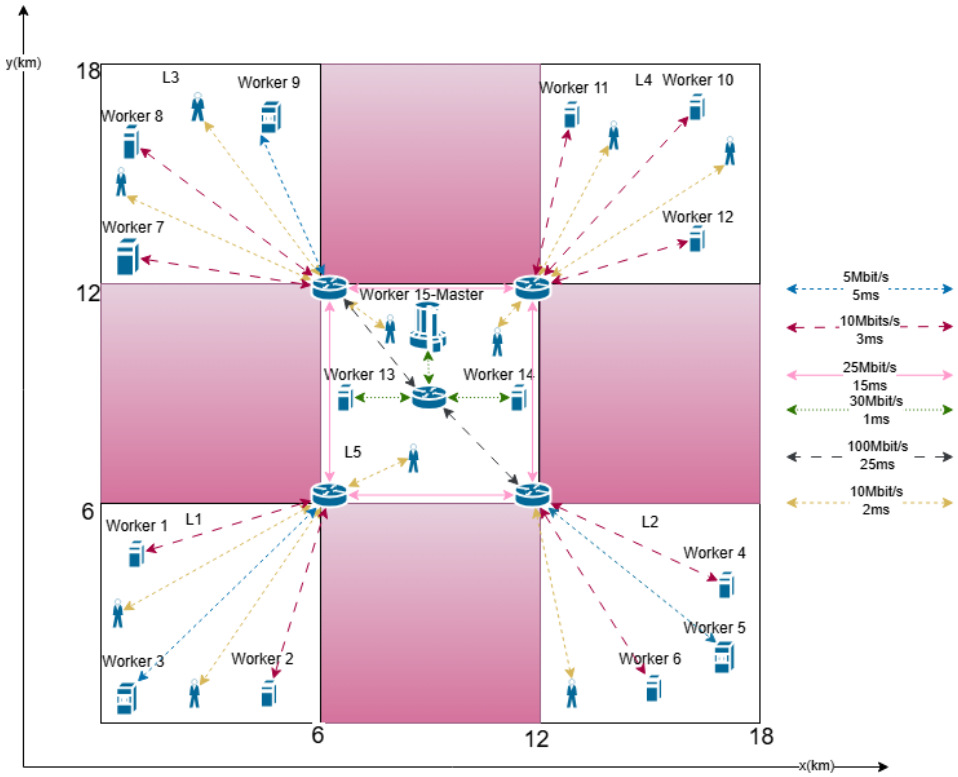
4.2 Διατύπωση προβλήματος – Problem Formulation

Το πρόβλημα κατανομής IoT έχει μοντελοποιηθεί ως μια διατύπωση MILP (mixed-integer linear programming) που παρουσιάστηκε αρχικά στο [62]. Το μοντέλο εξετάζει μια τοπολογία edge-fog-cloud στην οποία μπορούν να διατεθούν αλυσίδες υπηρεσιών σε containers. Μια εφαρμογή IoT αποσυντίθεται σε ένα σύνολο micro-services (μικρο-υπηρεσιών), οι οποίες έχουν ένα συγκεκριμένο παράγοντα αντιγραφής. Πολλοί χρήστες αναμένεται να έχουν πρόσβαση σε αυτές τις μικρο-υπηρεσίες.

Η υποδομή fog-cloud διαχειρίζεται ένα σύνολο κόμβων, στο οποίο πρέπει να κατανεμηθούν περιπτώσεις μικρο-υπηρεσιών με βάση τις απαιτήσεις της και υπόκειται σε πολλαπλούς περιορισμούς. Για παράδειγμα, όλες οι μικρο-υπηρεσίες που συνθέτουν μια συγκεκριμένη εφαρμογή πρέπει να κατανεμηθούν στο δίκτυο, ώστε η εφαρμογή να μπορεί να θεωρηθεί deployed.

Η υποδομή διαθέτει cloud κόμβους με υψηλούς πόρους, καθώς και edge/fog κόμβους οι οποίοι διαθέτουν περιορισμένη υπολογιστική δύναμη και μνήμη σε σχέση με τους cloud. Οι κόμβοι του δικτύου καθώς και οι χρήστες του υπάρχουν σε διάφορες

τοποθεσίες, με αυξημένη καθυστέρηση καθώς η απόσταση μεταξύ τους αυξάνεται. Ακολουθεί ένα ποιοτικό σχήμα της υποδομής [47].



Εικόνα 10: Υποδομή Fog Cloud που Χρησιμοποιήθηκε

4.2.1 Μεταβλητές

Όσον αφορά την διαμόρφωσή του cloud, ένα σετ εφαρμογών A που αποτελούνται από microservices S πρέπει να κατανεμηθούν σε κόμβους $\mathrm{n} \in N$.

Κάθε εφαρμογή α έχει ένα αναγνωριστικό SFC $\mathrm{id} \in ID$. Όλες οι μικρό-υπηρεσίες έχουν ένα μέγιστο αριθμό από replicas που δίνεται από το β. Ο παράγοντας αντιγραφής για μια συγκεκριμένη μικρό-υπηρεσία $s \in S$ για την η εφαρμογή α με το αναγνωριστικό SFC id δίνεται από το $\beta_{a,id,s}$. Έτσι, το μοντέλο καθορίζει τον ακριβή αριθμό αντιγράφων για κάθε μικρό-υπηρεσία ανάλογα με τον εξεταζόμενο στόχο (ο οποίος εδώ είναι η μείωση του κόστους).

Κάθε μικρό-υπηρεσία s έχει μια απαίτηση CPU και μια απαίτηση μνήμης που αντιπροσωπεύεται από ω_s (σε cpu) και γ_s (σε GB) αντίστοιχα. Για παράδειγμα, απαίτηση CPU ίση με 0,5 cpu (δηλ. 500 millicpu) σημαίνει ότι η μικρο-υπηρεσία χρειάζεται το 50% ενός πυρήνα για να λειτουργήσει σωστά.

Επίσης, κάθε μικρο-υπηρεσία έχει μια ελάχιστη απαίτηση εύρους ζώνης που αναπροσαρμόζεται, που εκφράζεται με δ_s (σε Mbit/s). Ένας δυαδικός πίνακας τοποθέτησης P χρησιμοποιείται για να αναπαραστήσει σε ποιόν κόμβο n κατανέμεται το αντίγραφο β_i μιας μικρο-υπηρεσίας s. Ο προηγούμενος πίνακας τοποθέτησης $P_{s,\beta_i}^{a,id}(n)^{i-1}$(n) προστίθεται στο μοντέλο, έτσι ώστε αποφάσεις να μπορούν να ληφθούν όσον αφορά τις μετακινήσεις υπηρεσιών, δεδομένου ότι το μοντέλο έχει πληροφορίες σχετικά με το πού έχουν κατανεμηθεί τα instances μικρο-υπηρεσιών στην προηγούμενη επανάληψη.

Στη συνέχεια, ο πίνακας μεταναστεύσεων υπηρεσιών M χρησιμοποιείται για να υποδείξει εάν ένα συγκεκριμένο αντίγραφο μικρό-υπηρεσίας έχει ανακατανεμηθεί σε άλλο κόμβο.

Συγκεκριμένα, αν $M_{s,\beta_i}^{a,id} = 1$, η replica β_i της μικρο-υπηερσίας s για την εφαρμογή α με το SFC αναγνωριστικό id έχει ανακατανεμηθεί με βάση τον προηγούμενο πίνακα τοποθέτησης.

Επιπλέον, ο συντελεστής μετανάστευσης υπηρεσιών μ αντιπροσωπεύει το μέγιστο επιτρεπόμενο ποσοστό ανακατανομής υπηρεσιών μεταξύ των επαναλήψεων του μοντέλου.

Όσον αφορά την ασύρματη διατύπωση, το μοντέλο υποστηρίζει δύο LPWAN τεχνολογίες, οι οποίες αναπαρίστανται μέσω γραμμικών εξισώσεων: IEEE 802.11ah και LoRaWAN.

Οι διατυπώσεις LoRaWAN βασίζονται στην εργασία που παρουσιάζεται στο Dawaliby et al. (2018)[63]. Ο δυαδικός πίνακας I_{gw} υποδεικνύει εάν η πύλη gw είναι πύλη IEEE 802.11ah, ενώ ο δυαδικός πίνακας L_{gw} καθορίζει εάν η πύλη gw είναι πύλη LoRaWAN.

Επίσης, ο πίνακας καθυστέρησης πρόσβασης στην πύλη που δίνεται από το K_{gw} αντιπροσωπεύει την καθυστέρηση πρόσβασης από την πύλη gw στην υποδομή νέφους-ομίχλης. Χωρίς απώλεια της γενικότητας, εάν I_{gw} = 1 (δηλαδή πύλη IEEE 802.11 ah), η καθυστέρηση πρόσβασης αντιστοιχεί σε 2 ms. Διαφορετικά, εάν $L_{gw} = 1$ (δηλ. πύλη LoRaWAN), η καθυστέρηση πρόσβασης είναι ίση με 5 ms.

Ο πίνακας αποστάσεων D υποδεικνύει την απόσταση (σε μέτρα) μεταξύ μιας πύλης gw και ενός αισθητήρα sr. Ένας πρόσθετος δυαδικός πίνακας συσχέτισης A χρησιμοποιείται για να δείξει αν ένας αισθητήρας sr μπορεί να συσχετιστεί με μια πύλη gw. Η συσχέτιση αυτή βασίζεται στον πίνακα αποστάσεων D και στα AID που είναι διαθέσιμα σε κάθε πύλη. Μία IEEE 802.11ah πύλη δεν μπορεί να έχει περισσότερους από 8192 συνδεδεμένους σταθμούς όπως αναφέρεται στο τελευταίο

πρότυπο. Ωστόσο, ο περιορισμός της συσχέτισης έχει οριστεί σε 50, δεδομένου ότι έχει ληφθεί υπόψη μια αστική ανάπτυξη macro με εκτεταμένη εμβέλεια.

Για τις πύλες LoRaWAN, ο περιορισμός συσχέτισης έχει οριστεί σε 100, δεδομένου ότι ο μέγιστος ρυθμός δεδομένων που επιτυγχάνεται από κάθε αισθητήρα μειώνεται σημαντικά όταν συσχετίζεται ένας μεγαλύτερος αριθμός των αισθητήρων.

Έτσι, με τη δημιουργία ενός κατώτερο περιορισμού, υποτίθεται ότι οι καλές συνθήκες καναλιού επιτυγχάνονται πάντοτε και ότι όλοι οι αισθητήρες μπορούν να έχουν πρόσβαση στις εφαρμογές που γίνονται deploy, εάν συνδέονται με μία πύλη. Επιπλέον, για το IEEE 802.11 ah, ο περιορισμός της απόστασης ορίζεται στα χίλια μέτρα, επειδή αυτή είναι η μέγιστη εμβέλεια κάλυψης στα δίκτυα IEEE 802.11ah, ενώ για το LoRaWAN το όριο ορίζεται στα τέσσερις χιλιάδες μέτρα.

Αν η $D_{gw,sr}$ είναι μικρότερή από το επιβαλλόμενο όριο, ο αισθητήρας sr μπορεί να συνδεθεί με την πύλη gw και τότε $A_{sr,gw} = 1$, διαφορετικά, $A_{sr,gw} = 0$.

Επιπλέον, η έννοια του τεμαχισμού του δικτύου έχει συμπεριληφθεί στο μοντέλο με την προσθήκη του συνόλου των slices (φετών) SL. Ο δυαδικός πίνακας application slice $A_{\alpha,sl}$ υποδεικνύει εάν η φέτα sl είναι υπεύθυνη για την ασύρματη κυκλοφορία που ανήκει στην εφαρμογή a, ενώ ο δυαδικός πίνακας συσχέτισης slice $A_{sr,sl}$ υποδεικνύει εάν ο αισθητήρας sr πρέπει να συσχετιστεί με τη φέτα sl λόγω της εφαρμογής που του έχει ανατεθεί.

Ο πίνακας εύρους ζώνης $B_{sl,gw}$ περιέχει το διαθέσιμο εύρος ζώνης (σε Mbit/s) για τη φέτα sl στην πύλη gw. Στη συνέχεια, ο πίνακας ρυθμού δεδομένων $\eta_{sr,sl,gw}$ περιέχει το διαθέσιμο εύρος ζώνης για τον αισθητήρα sr που έχει αντιστοιχιστεί στη φέτα sl στην πύλη gw.

Για το LoRaWAN, ο ρυθμός δεδομένων κάθε gw εξαρτάται από τον παράγοντα διασποράς (SF) που υιοθετεί κάθε αισθητήρας sr. Ο SF αφορά τον λόγο μεταξύ του ρυθμού συμβόλων και του ρυθμού chip.

Το LoRaWAN διαδίδει κάθε σύμβολο με ρυθμό 2SF chips ανά σύμβολο με SF = {7, &tnqh_02025, 12}. Στο μοντέλο, ο SF έχει οριστεί σε 9. Επιπλέον, ο πίνακας χρόνου μεταφοράς δεδομένων T_{sr} αντιστοιχεί στο χρόνο διάδοσης (σε ms) για ένα φτάσει ένα μεμονωμένο μήνυμα μεταφόρτωσης από τον αισθητήρα sr στη σχετική πύλη.

Ο χρόνος μεταφοράς δεδομένων εξαρτάται από την επιλεγμένη τεχνολογία LPWAN και από τον συνολικό αριθμό των συνδεδεμένων αισθητήρων στη συγκεκριμένη πύλη, ο οποίος επηρεάζεται από τον παράγοντα εύρους ζώνης ζ_{gw} της πύλης. Ουσιαστικά, το

διαθέσιμο εύρος ζώνης ανά αισθητήρα μπορεί να μειωθεί ανάλογα με τον αριθμό των συνδεδεμένων αισθητήρων σε κάθε πύλη. Περισσότερες λεπτομέρειες σχετικά με τον τρόπο με τον οποίο επηρεάζεται ο χρόνος μεταφοράς δεδομένων από αυτόν τον παράγοντα δίνονται στην επόμενη ενότητα. Στη συνέχεια, για την πλήρη ελαχιστοποίηση της αναμενόμενης καθυστέρησης κάθε χρήστη κατά την πρόσβαση στη συγκεκριμένη εφαρμογή, προστέθηκε μια άλλη μεταβλητή απόφασης ψ_u, η οποία περιέχει το χρόνο διάδοσης (σε ms) ενός αιτήματος του χρήστη u για να φτάσει στην εκχωρημένη εφαρμογή, πιο συγκεκριμένα για να αποκτήσει πρόσβαση σε μια περίπτωση της τελευταίας μικρο-υπηρεσίας στην αλυσίδα υπηρεσιών της εφαρμογής με την οποία είναι συνδεδεμένος ο χρήστης u.

Τα Variables που χρησιμοποιούνται στο μοντέλο εμπεριέχονται στους ακόλουθους πίνακες. Οι μεταβλητές που περιγράφονται σε αυτό μέρος της διπλωματικής εργασίας ορίστηκαν αρχικά στο [62].

Συγκεκριμένα ο Πίνακας 4.1 περιέχει τις Μεταβλητές εισόδου που σχετίζονται με την υποδομή νέφους.

Σύμβολο	**Περιγραφή**
N	Το σύνολο κόμβων στους οποίους εκτελούνται οι περιπτώσεις μικρο-υπηρεσιών.
A	Το σύνολο όλων των εφαρμογών IoT. Κάθε εφαρμογή αποτελείται από ένα σύνολο διαφορετικών μικρο-υπηρεσιών.
S	Το σύνολο όλων των μικρο-υπηρεσιών.
ID	Το σύνολο αναγνωριστικών SFC.
U	Το σύνολο των χρηστών.
L	Το σύνολο των τοποθεσιών όπου οι χρήστες μπορούν να έχουν πρόσβαση σε μια δεδομένη εφαρμογή a ϵ A.
$\Phi_{u,a}$	Ο πίνακας ανάθεσης χρηστών. Αν $\Phi_{u,a}$= 1, ο χρήστης u χρησιμοποιεί την εφαρμογή α.
ρ_a	Ο μέγιστος αριθμός χρηστών που μπορούν να συνδεθούν με μια εφαρμογή.
ρ_s	Ο μέγιστος αριθμός χρηστών που μπορούν να συνδεθούν με μια περίπτωση μικρο-υπηρεσίας.

λ_u	Το κόστος σύνδεσης του χρήστη u για την ανατεθειμένη εφαρμογή.
β	Ο μέγιστος παράγοντας αντιγραφής για κάθε μικρό-υπηρεσία.
u_s	Δείκτης πρώτης θέσης στην SFC. Αν u_s = 1, η μικρο-υπηρεσία s είναι η πρώτη στη σειρά υπηρεσιών.
l_s	Δείκτης τελευταίας θέσης στην SFC. Αν l_s= 1, η μικρο-υπηρεσία s είναι η τελευταία στη σειρά υπηρεσιών.
μ	Ο παράγοντας μετανάστευσης υπηρεσίας, που αντιπροσωπεύει το μέγιστο επιτρεπόμενο ποσοστό επανατοποθετήσεων μικρό-υπηρεσιών.
$I_{\alpha,s}$	Ο πίνακας instances. Αν $I_{\alpha,s}$, η μικρο-υπηρεσία s ανήκει στην εφαρμογή a. Διαφορετικά, η μικρό-υπηρεσία s δεν αποτελεί μέρος της εφαρμογής α.
$R_{s,n}$	Ο πίνακας σχέσεων. Αν $R_{s,n}$ = 1, η μικρο-υπηρεσία s μπορεί να αναπτυχθεί στον κόμβο n. Διαφορετικά, δεν μπορεί να γίνει deployed στον κόμβο n λόγω υλικού ή περιορισμών λογισμικού.
Ω_n	Η συνολική χωρητικότητα CPU (σε cpu) του κόμβου n ϵ N.
Γ_n	Η συνολική χωρητικότητα μνήμης (σε GB) του κόμβου n ϵ N.
Δ_n	Η χωρητικότητα εύρους ζώνης (σε Mbit/s) του κόμβου n ϵ N.
ω_s	Η απαίτηση σε CPU (σε cpu) της μικρο-υπηρεσίας s ϵ S.
γ_s	Η απαίτηση σε μνήμη (σε GB) της μικρο-υπηρεσίας s ϵ S.
δ_s	Η απαίτηση σε εύρος ζώνης (σε Mbit/s) της μικρο-υπηρεσίας s ϵ S.
$B_{n1,n2}$	Ο πίνακας εύρους ζώνης μεταξύ κόμβων n1 και n2 (σε Mbit/s).
$\tau_{n1,n2}$	Ο πίνακας καθυστέρησης μεταξύ κόμβων n1 και n2 (σε ms).
$\tau_{l1,ln2}$	Ο πίνακας καθυστέρησης μεταξύ τοποθεσιών l1 και l2 (σε ms).
$\tau_{n,l}$	Ο πίνακας καθυστέρησης μεταξύ κόμβου n και τοποθεσίας l (σε ms).

$\tau_{u,l}$	Ο πίνακας καθυστέρησης μεταξύ χρήστη u και τοποθεσίας l (σε ms).
$\tau_{u,n}$	Ο πίνακας καθυστέρησης μεταξύ χρήστη u και κόμβου n (σε ms).
C_{s_i,s_j}	Ο πίνακας επικοινωνίας που δείχνει το ελάχιστο απαιτούμενο εύρος ζώνης (σε Mbit/s) μεταξύ των μικρο-υπηρεσιών S_i (πηγή) και S_j (υποδοχέας).
$E_{n,l}$	Αν $E_{n,l}$= 1, ο κόμβος n βρίσκεται στην τοποθεσία l.
$E_{u,l}$	Αν $E_{u,l}$= 1, ο χρήστης u βρίσκεται στην τοποθεσία l.
α_{s_i,s_j}	Ο πίνακας υπηρεσιών. Αν α_{s_i,s_j}= 1, η ροή μεταξύ S_i και S_j πρέπει να εγγυάται εύρος ζώνης· αλλιώς όχι.

Πίνακας 4.1: Μεταβλητές εισόδου που σχετίζονται με την υποδομή νέφους.

Ο Πίνακας 4.2 εμπεριέχει τις Μεταβλητές εισόδου που σχετίζονται με τη διαστασιολόγηση του ασύρματου δικτύου.

Σύμβολο	**Περιγραφή**
GW	Το σύνολο των ασύρματων πυλών.
SR	Το σύνολο των αισθητήρων (sensors).
SL	Το σύνολο των network slices (φετών δικτύου).
Θ_{gw}	Το σύνολο διαθέσιμων Αναγνωριστικών Ανάθεσης (AID) σε πύλη $gw \in GW$.
Θ_{sr}	Κάθε αισθητήρας $sr \in SR$ χρειάζεται ένα AID για να συνδεθεί με πύλη.
$\Phi_{sr,\alpha}$	Ο πίνακας ανάθεσης αισθητήρων. Αν $\Phi_{sr,\alpha}$= 1, ο αισθητήρας sr χρησιμοποιεί την εφαρμογή α.
$D_{gw,sr}$	Ο πίνακας αποστάσεων (σε μέτρα) μεταξύ πύλης gw και αισθητήρα sr.
$PL_{gw,sr}$	Ο πίνακας Path Loss (σε dB) μεταξύ πύλης gw και αισθητήρα sr.
$A_{gw,sr}$	Ο πίνακας συσχέτισης. Αν $A_{gw,sr}$= 1, ο αισθητήρας sr μπορεί να συσχετιστεί με την πύλη gw.
$A_{\alpha,sl}$	Ο πίνακας slice εφαρμογής. Αν $A_{\alpha,sl}$= 1, το slice sl είναι υπεύθυνο για την ασύρματη κίνηση της εφαρμογής α.
$A_{sr,sl}$	Ο πίνακας ανάθεσης slice. Αν $A_{sr,sl}$= 1, ο αισθητήρας sr πρέπει να ανατεθεί στο slice sl λόγω της συσχετισμένης εφαρμογής του.
$A_{sl,gw}$	Ο πίνακας εύρους ζώνης (σε Mbit/s) διαθέσιμος για το slice sl στην πύλη gw.
$\eta_{sr,sl,gw}$	Ο πίνακας ρυθμού δεδομένων (σε Mbit/s) για τον αισθητήρα sr ανατεθειμένο στο slice sl στην πύλη gw.
$\tau_{gw,l}$	Ο πίνακας καθυστέρησης (σε ms) μεταξύ πύλης gw και τοποθεσίας l.
$\tau_{gw,n}$	Ο πίνακας καθυστέρησης (σε ms) μεταξύ πύλης gw και κόμβου n.
I_{gw}	Ο πίνακας IEEE 802.11ah. Αν I_{gw}= 1, η πύλη gw υποστηρίζει το πρότυπο IEEE 802.11ah.
L_{gw}	Ο πίνακας LoRaWAN. Αν L_{gw} = 1, η πύλη gw είναι LoRaWAN.

K_{gw}	Ο πίνακας καθυστέρησης πρόσβασης πύλης (σε ms) από την πύλη gw προς την υποδομή Fog-cloud.
π_{α}	Ο αριθμός bits που πρέπει να μεταδοθούν σε κάθε μήνυμα upload για την εφαρμογή a.
π_{sr}	Ο αριθμός bits που πρέπει να μεταδοθούν από κάθε αισθητήρα sr βάσει της εφαρμογής του.
$E_{gw,l}$	Αν $E_{gw,l}$= 1, η πύλη gw βρίσκεται στην τοποθεσία l.
$E_{sr,l}$	Αν $E_{sr,l}$= 1, ο αισθητήρας sr βρίσκεται στην τοποθεσία l.

Πίνακας 4.2: Μεταβλητές εισόδου που σχετίζονται με τη διαστασιολόγηση του ασύρματου δικτύου.

Ο Πίνακας 4.3 περιέχει τις Μεταβλητές απόφασης του μοντέλου MILP.

Σύμβολο	**Περιγραφή**
$G_{a,id}$	Ο πίνακας αποδοχής εφαρμογών. Εάν $G_{a,id}$= 1, η εφαρμογή α με το αναγνωριστικό SFC id μπορεί να κατανεμηθεί. Εάν $G_{a,id}$= 0, η εφαρμογή α με το αναγνωριστικό SFC id δεν μπορεί να διατεθεί.
$G_{a,id,s}$	Ο πίνακας αποδοχής μικρό-υπηρεσιών. Εάν $G_{a,id,s}$= 1, η μικρουπηρεσία s για την εφαρμογή a με το αναγνωριστικό SFC id μπορεί να διατεθεί. Εάν $G_{a,id,s}$= 0, η μικρο-υπηρεσία s για την εφαρμογή a με το αναγνωριστικό SFC id δεν μπορεί να διατεθεί.
$G_{u,a,id}$	Ο πίνακας αποδοχής χρηστών. Εάν $G_{u,a,id}$= 1, ο χρήστης u σχετίζεται με την εφαρμογή a με το αναγνωριστικό SFC id.
$\beta_{a,id,s}$	Ο παράγοντας αντιγραφής της μικρο-υπηρεσίας s για την εφαρμογή a με το αναγνωριστικό SFC id.
$P_{s,\beta_i}^{a,id}(n)$	Ο πίνακας τοποθέτησης. Αν $P_{s,\beta_i}^{a,id}(n)$= 1, το αντίγραφο β_i της μικρο-υπηρεσίας s εκτελείται στον κόμβο n για την εφαρμογή α με το αναγνωριστικό SFC id.
$P_{s,\beta_i}^{a,id}(n)^{i-1}$	Ο πίνακας τοποθέτησης από την προηγούμενη επανάληψη.
$F_{s_j,\beta_j}^{a,id,s_i,\beta_i}(n_1,n_2)$	Ο πίνακας δυαδικής ροής δείχνει ότι το αντίγραφο β_i από τη μικρο-υπηρεσία s_i και το αντίγραφο β_j από τη μικρο-υπηρεσία s_j κατανέμονται στον κόμβο n1 και n2, αντίστοιχα, για την εφαρμογή a με το αναγνωριστικό SFC id.
$\Lambda_{a,id}$	Ο πίνακας καθυστέρησης SFC υποδεικνύει το χρόνο (σε ms) για τη διέλευση όλων των πιθανών διαδρομών στην αλυσίδα υπηρεσιών για την εφαρμογή a με το αναγνωριστικό SFC id.
$M_{s,\beta_i}^{a,id}$	Ο πίνακας μεταναστεύσεων υπηρεσιών. Εάν $M_{s,\beta_i}^{a,id} = 1$, το αντίγραφο β_i της μικρουπηρεσίας s για την εφαρμογή α με το αναγνωριστικό SFC id έχει ανακατανεμηθεί σε άλλον κόμβο με βάση την προηγούμενη επανάληψη.

U_{gw}	Ο πίνακας χρήσης της πύλης. U_{gw}= 1 υποδηλώνει ότι υπάρχει τουλάχιστον ένας αισθητήρας που συνδέεται με την πύλη gw.
U_n	Ο πίνακας χρήσης κόμβων. U_n = 1 υποδηλώνει ότι υπάρχει τουλάχιστον ένα αντίγραφο μικρό-υπηρεσίας κατανεμημένο στον κόμβο n.
$U_{s,n}$	Ο πίνακας εκτέλεσης μικρο-υπηρεσιών. Εάν $U_{s,n}$= 1, ένα αντίγραφο της μικρο-υπηρεσίας s διατίθεται στον κόμβο n. Εάν $U_{s,n}$ = 0, κανένα αντίγραφο της μικρο-υπηρεσίας s δεν διατίθεται στον κόμβο n.
$U_{s,\beta_i}^{u,a,id}(n)$	Ο πίνακας εξυπηρέτησης χρηστών. Εάν $U_{s,\beta_i}^{u,a,id}(n)$ = 1, ο χρήστης u συσχετίζεται με το αντίγραφο β_i της μικρο-υπηρεσίας s που διατίθεται στον κόμβο n για την εφαρμογή α με το αναγνωριστικό SFC id.
$U_{sr,gw}$	Ο πίνακας συσχέτισης αισθητήρων. Εάν $U_{sr,gw}$= 1, ο αισθητήρας sr συνδέεται με την πύλη gw.
$U_{sr,sl,gw}$	Ο πίνακας εκτέλεσης αισθητήρων. Εάν $U_{sr,sl,gw}$ = 1, ο αισθητήρας sr ανατίθεται στη φέτα sl στην πύλη gw.
ζ_{gw}	Ο παράγοντας εύρους ζώνης της πύλης.
T_{sr}	Ο πίνακας χρόνου μεταφοράς δεδομένων υποδεικνύει το χρόνο διάδοσης (σε ms) ενός μεμονωμένου μηνύματος μεταφόρτωσης από τον αισθητήρα sr.
ψ_u	Ο πίνακας καθυστέρησης χρήστη υποδεικνύει το χρόνο διάδοσης (σε ms) μιας αίτησης από τον χρήστη u στην τελευταία μικρό-υπηρεσία της ανατεθείσας αλυσίδας υπηρεσιών.

Πίνακας 4.3: Μεταβλητές απόφασης του μοντέλου MILP

4.2.2 Περιορισμοί του προβλήματος - Problem Constraints

Έχουν προστεθεί πολλαπλοί περιορισμοί για να αντικατοπτρίζουν τις επεκτάσεις σχετικά με την αναπαραγωγή υπηρεσιών, την αλυσίδα υπηρεσιών και τις ασύρματες διατυπώσεις στο μοντέλο.

Οι αισθητήρες είναι διασκορπισμένοι σε όλη την περιοχή του δικτύου για τη συλλογή δεδομένων. Κάθε αισθητήρας πρέπει να συνδεθεί με μία ασύρματη πύλη για να μπορεί να στέλνει δεδομένα στην υποδομή Fog-cloud. Το διαθέσιμο εύρος ζώνης ανά αισθητήρα επηρεάζεται από το εύρος ζώνης της πύλης παράγοντα, ο οποίος μπορεί να αυξήσει το χρόνο μεταφοράς δεδομένων του αισθητήρα. Η υποδομή Fog-cloud διαχειρίζεται ένα σύνολο κόμβων, στους οποίους οι μικρο-υπηρεσίες πρέπει να κατανέμονται με βάση τις απαιτήσεις της και να υπόκεινται σε πολλαπλούς περιορισμούς:

1. Οι κόμβοι έχουν περιορισμένες δυνατότητες (π.χ. CPU και μνήμη).
2. Τα instances μικρο-υπηρεσιών δεν μπορούν να ενεργοποιηθούν σε κάθε κόμβο, λόγω ειδικών απαιτήσεων υλικού ή λογισμικού.
3. Οι πύλες έχουν περιορισμένη χωρητικότητα, βάσει ενός μέγιστου αριθμού αναγνωριστικών συσχετίσεων (AID).
4. Οι αισθητήρες πρέπει να συσχετίζονται με το κομμάτι πύλης που έχει διατεθεί για τη συγκεκριμένη εφαρμογή στην οποία προσπαθούν να έχουν πρόσβαση.
5. Ο χρόνος μεταφοράς δεδομένων των αισθητήρων εξαρτάται από την επιλεγμένη LPWAN τεχνολογία.
6. Όλες οι μικρο-υπηρεσίες που συνθέτουν μια εφαρμογή πρέπει να κατανέμονται στο δίκτυο.
7. Οι χρήστες πρέπει να έχουν πρόσβαση στην εκχωρημένη εφαρμογή τους.
8. Ο συντελεστής αντιγραφής κάθε μικρο-υπηρεσίας εξαρτάται από τον αριθμό των αιτήσεων των χρηστών.

Αρχικά, ο στόχος έχει ενημερωθεί ώστε να λαμβάνει υπόψη την ανατεθείσα εφαρμογή του χρήστη, όπως φαίνεται στην (1):

$$(1)\ \max \sum_{u \in U} \sum_{a \in A} \sum_{id \in ID} \Phi_{u,a} \times G_{u,a,r}$$

Για να περιοριστεί η συσχέτιση των χρηστών με μια συγκεκριμένη εφαρμογή, χρησιμοποιήθηκε ένας περιορισμός που αντιπροσωπεύεται από την (2) για να εγγυηθεί ότι ο μέγιστος αριθμός χρηστών ανά εφαρμογή τηρείται:

$$(2)\ \forall a \in A, id \in ID: \sum_{u \in U} \Phi_{u,a} \times G_{u,a,id} \leq \rho_a$$

Έχει επίσης συμπεριληφθεί ένας περιορισμός για να διασφαλιστεί ότι κάθε χρήστης συνδέεται με μία μόνο εφαρμογή, όπως φαίνεται στην (3):

$$(3)\ \forall u \in U: \sum_{a \in Aid} \sum_{e \in ID} \Phi_{u,a} \times G_{u,a,id} = 1$$

Έχουν επίσης προστεθεί περιορισμοί συσχέτισης υπηρεσιών για να εξασφαλιστεί ότι οι χρήστες συνδέονται με μια συγκεκριμένη μικρο-υπηρεσία για όλες τις μικρο-υπηρεσίες της παραδεκτής εφαρμογής τους. Έτσι, οι χρήστες μπορούν να έχουν πρόσβαση στην εφαρμογή που τους έχει ανατεθεί, εάν είναι συνδεδεμένοι με όλες τις μικρο-υπηρεσίες τους, όπως φαίνεται στην (4).

$\forall u \in U, a \in A, id \in ID$:

$$(4) \sum_{s \in S} \sum_{\beta_i \in \beta} \sum_{n \in N} U_{s,\beta_i}^{u,a,id}(n) = \Phi_{u,a} \times \sum_{s \in S} I_{a,s}$$

Επίσης, κάθε χρήστης μπορεί να γίνει δεκτός μόνο σε μόνο ένα αντίγραφο της ίδιας μικρό-υπηρεσίας, όπως δίνεται από την (5).

$\forall u \in U, a \in A, id \in ID, s \in S$:

$$(5) \sum_{\beta_i \in \beta} \sum_{n \in N} U_{s,\beta_i}^{\mathrm{u},a,id}(n) = \Phi_{u,a} \times I_{a,s}$$

Στη συνέχεια, κάθε αντίγραφο μικρο-υπηρεσίας υπόκειται σε περιορισμό συσχέτισης με βάση το ρ_s και το κόστος συσχέτισης λ_u του χρήστη, όπως φαίνεται στην (6).

$\forall a \in A,\ id \in ID,\ s \in S,\ \beta_i \in \beta,\ n \in N$:

$$(6) \sum_{u \in U} \lambda_u \times \Phi_{u,a} \times U_{s,\beta_i}^{u,a,id}(n) \leq \rho_s$$

Επίσης, έχει συμπεριληφθεί ένας περιορισμός για να διασφαλιστεί ότι οι χρήστες συνδέονται μόνο με τις μικρο-υπηρεσίες που διατίθενται στο δίκτυο, ως φαίνεται στην (7).

$\forall u \in U, a \in A, id \in ID, s \in S, \beta_i \in \beta, n \in N$:

$$(7)\ U_{s,\beta_i}^{\mathrm{u},a,id}(n) \leq P_{s,\beta_i}^{a,id}(n)$$

Στη συνέχεια, υπάρχουν δύο περιορισμοί που αντικατοπτρίζουν τις σχέσεις μεταξύ των πινάκων αποδοχής G. Μια μικρό-υπηρεσία θεωρείται instantiated εάν ο αναμενόμενος αριθμός των instances μικρό-υπηρεσιών είναι ίσος με τον αριθμό των διατιθέμενων replicas στο δίκτυο. Επίσης, μια εφαρμογή γίνεται αποδεκτή μόνο εάν όλες οι μικρο-υπηρεσίες της έχουν γίνει δεκτές στο δίκτυο (δηλαδή όλα τα instances που έχουν γίνει deploy στο δίκτυο). Αυτοί οι περιορισμοί αναπαρίστανται από τις (8) και (9), αντίστοιχα.

$$\forall a \in A,\ id \in ID,\ s \in S:$$

$$\mathrm{G_{a,id,s}} = 1 \text{ αν ισχύει : } I_{a,s} \times \beta_{\mathrm{a,id,s}} = \sum_{\beta_i, n \in \beta, N} \mathrm{P_{s,\beta_i}^{a,id}(n)}$$

$$\alpha\lambda\lambda\iota\acute{\omega}\varsigma\ G_{a,id,s} = 0\ (8)$$

Και

$$\forall a \in A,\ id \in ID:$$

$$G_{a,id,s} = 1\ \alpha\nu\ \iota\sigma\chi\acute{\upsilon}\varepsilon\iota \sum_{s \in S} I_{a,s} = \sum_{s \in S} G_{a,id,s}$$

$$\alpha\lambda\lambda\iota\acute{\omega}\varsigma\ G_{a,id,s} = 0\ (9)$$

Έχει προστεθεί ένας περιορισμός για τον περιορισμό της κατανομής μιας περίπτωσης της ίδιας μικρο-υπηρεσίας ανά κόμβο. Έτσι, μόνο ένα αντίγραφο μικρο-υπηρεσίας μπορεί να αναπτυχθεί στον ίδιο κόμβο. Αυτός ο περιορισμός παρουσιάζεται στη (10).

$$(10)\ \forall a \in A,\ id \in ID,\ s \in S,\ \beta_i \in \beta: \sum_{n \in N} P_{s,\beta_i}^{a,id}(n) \leq 1.0$$

Οι περιορισμοί της CPU και της μνήμης έχουν επίσης μοντελοποιηθεί, όπως φαίνεται στις (11) και (12), αντίστοιχα. Ομοίως, στη (13), έχουν οριστεί οι περιορισμοί εύρους ζώνης. Οι απαιτήσεις εύρους ζώνης έχουν προστεθεί στο μοντέλο έτσι ώστε οι

αναμενόμενες απαιτήσεις εύρους ζώνης να μπορούν να ικανοποιηθούν, δεδομένης της περιορισμένης χωρητικότητας της σύνδεσης δικτύου.

$$\forall n \in N: \sum_{a \in A} \sum_{\mathrm{id} \in ID} \sum_{s \in S} \sum_{\beta_i \in \beta} P_{s,\beta_i}^{a,id}(n) \times \omega_s \leq \Omega_n \ (11)$$

$$\forall n \in N: \sum_{a \in A} \sum_{\mathrm{id} \in ID} \sum_{s \in S} \sum_{\beta_i \in \beta} P_{s,\beta_i}^{a,id}(n) \times \gamma_s \leq \Gamma_{\boldsymbol{n}} \ (12)$$

$$\forall n \in N: \sum_{a \in A} \sum_{\mathrm{id} \in ID} \sum_{s \in S} \sum_{\beta_i \in \beta} P_{s,\beta_i}^{a,id}(n) \times \delta_s \leq \Delta_{\boldsymbol{n}} \ (13)$$

Τέλος, προστίθενται περιορισμοί σχετικά με την καθυστέρηση του χρήστη ψ_u. Η καθυστέρηση χρήστη αντιστοιχεί στο χρόνο διάδοσης (σε ms) ενός αιτήματος από το χρήστη u για να φτάσει σε μια περίπτωση της τελευταίας μικρο-υπηρεσίας στην αλυσίδα υπηρεσιών της εφαρμογής με την οποία είναι συνδεδεμένος ο χρήστης u, όπως εκφράζεται από την (14). Όπως φαίνεται, η καθυστέρηση χρήστη εξαρτάται από τον κόμβο στον οποίο έχει κατανεμηθεί το τελευταίο αντίγραφο μικρο-υπηρεσίας. Εάν το αντίγραφο της μικρο-υπηρεσίας είναι τοποθετημένο σε κόμβο μακριά από τον χρήστη, η καθυστέρηση χρήστη θα είναι σαφώς υψηλότερη. Κατά συνέπεια, η καθυστέρηση E2E είναι διπλάσια της τιμής του ψ_u, καθώς αντιπροσωπεύει το χρόνο που χρειάζεται ένα αίτημα χρήστη για να φτάσει στην τελευταία υπηρεσία της αλυσίδας και να επιστρέψει στον χρήστη.

$$\forall u \in U,\ a \in A,\ id \in ID,\ s \in S,\ \beta_i \in \beta,\ n \in N:$$

$$\text{αν } G_{u,a,id} = 1 \quad (u \text{ συσχετισμένος με εφ. } a \text{ με SFC id } id)$$

$$\text{αν } l_s \times U_{s,\rho_i}^{\mu,a,id}(n) = 1 \quad (u \text{ συνδεδεμένος με το πιο πρόσφατο instance})$$

τότε

$$(14)\ \psi_u = \tau_{u,n}\ (\sigma\varepsilon\ ms)$$

Το μοντέλο καθώς και ο τρόπος υπολογισμού της βέλτιστης τοποθέτησης, την οποία χρησιμοποιούμε για να εκπαιδεύσουμε εν συνεχεία τον agent, αναπτύσσονται εκτενώς στο [62].

4.2.3 Στόχος βελτιστοποίησης

Το μοντέλο MILP έχει ως στόχο την ελαχιστοποίηση του κόστους, και συνεπώς την αύξηση της ενεργειακής απόδοσης.

Σύμβολο	Περιγραφή
$P_{s,\beta_i}^{a,id}(n)$	Ο πίνακας τοποθέτησης. Όταν $P_{s,\beta_i}^{a,id}(n) = 1$, το αντίγραφο β_i της μικρο-υπηρεσίας s εκτελείται στον κόμβο n για την εφαρμογή α με SFC αναγνωριστικό id
$\overline{\omega_n}$	Το σχετικό βάρος του κόμβου n
ω_s	Η απαίτηση CPU (σε cpu) της μικρο-υπηρεσίας s
γ_s	Η απαίτηση μνήμης (σε GB) της μικρο-υπηρεσίας s
δ_s	Η απαίτηση εύρους ζώνης (σε Mbit/s) της μικρο-υπηρεσίας s

Πίνακας 4.4 : Μεταβλητές που χρησιμοποιούνται για την ελαχιστοποίηση του συνολικού κόστους του συστήματος.

Χρησιμοποιώντας τον Πίνακα 4.1, ο στόχος αυτός μπορεί να εκφραστεί ως εξής [47]:

$$\sum_{a\in A}\sum_{id\in \mathrm{ID}}\sum_{s\in S}\sum_{\beta_i\in \beta}\sum_{n\in N} P_{s,\beta_i}^{a,id}(n)\times \varpi_n \times \omega_s \times \gamma_s \times \delta_s$$

τον οποίο και θέλουμε να ελαχιστοποιήσουμε.

Ο αριθμός των δυαδικών μεταβλητών τοποθέτησης $P_{s,\beta_i}^{a,id}(n)$ είναι:

$$|\boldsymbol{A}| \times |\boldsymbol{S}| \times |\boldsymbol{\beta}| \times |\boldsymbol{N}|$$

Το MILP συνδυάζει συνδυαστική βελτιστοποίηση (ακέραιες μεταβλητές) με γραμμικούς περιορισμούς/στόχους. Η επίλυση απαιτεί τη διερεύνηση εκθετικών χώρων λύσεων στη χειρότερη περίπτωση (π.χ. branch-and-bound) [44]. Η ελαχιστοποίηση του κόστους αναμένεται να γίνεται σε εκθετικά περισσότερο χρόνο καθώς εξετάζονται σενάρια με περισσότερους χρήστες, κόμβους, εφαρμογές ή μικροϋπηρεσίες, καθώς ο χώρος λύσεων κλιμακώνεται με $O\ (|\boldsymbol{A}| \times |\boldsymbol{S}| \times |\boldsymbol{\beta}| \times |\boldsymbol{N}|)$.

Παρόλο που το σύνολο των χρηστών του συστήματος U δεν αναφέρεται, αυξάνει έμμεσα τον χώρο των λύσεων (και συνεπώς την πολυπλοκότητα) καθώς, λόγω των περιορισμών, είναι αναγκαίο να γίνουν deploy περισσότερες μικροϋπηρεσίες για να εξυπηρετηθούν.

Κάθε πρόσθετος περιορισμός (π.χ. μεταναστεύσεις, capacity, slices, LPWAN) πέρα από αυτούς πολλαπλασιάζει τον χώρο των επιτρεπτών λύσεων, καθιστώντας την MILP απαίτηση μνήμης και χρόνου επίλυσης ακόμη υψηλότερη.

Για την MILP επίλυση χρησιμοποιήθηκε ο solver IBM ILOG CPLEX ILP, και παρατηρήθηκε εκθετική αύξηση του χρόνου καθώς αυξάνονται οι χρήστες για τους οποίους έπρεπε να υπολογιστεί η βέλτιστη ανάθεση, πράγμα που επιβεβαιώνει την NP-hard πολυπλοκότητα του προβλήματος.

5.1 Προτεινόμενη Λύση

Η προσέγγιση που προτείνεται στην παρούσα διπλωματική εργασία είναι η αξιοποίηση του αλγορίθμου Maskable Proximal Policy Optimization (PPO) προκειμένου να διδάξουμε σε έναν agent μια πολιτική κατανομής microservices η οποία θα πλησιάζει την βέλτιστη, όσον αφορά την μείωση του κόστους και συνεπώς την κατανάλωση ενέργειας. Προσπαθούμε δηλαδή να ελαχιστοποιηθεί το κόστος

$$\sum_{a\in A}\sum_{id\in \mathrm{ID}}\sum_{s\in S}\sum_{\beta_i\in\beta}\sum_{n\in N} P_{s,\beta_i}^{a,id}(n)\times \varpi_n \times \omega_s \times \gamma_s \times \delta_s$$

της τοποθέτησης μικρο-υπηρεσιών στην τοπολογία ενώ ταυτόχρονα, θα πρέπει να τηρούνται όλοι οι προαναφερθέντες περιορισμοί και να εξυπηρετούνται όσον το δυνατό περισσότεροι χρήστες.

Ο λόγος που στρεφόμαστε στην βαθιά ενισχυτική μάθηση είναι για την εύρεση λύσεων σε πραγματικό χρόνο μέσω του εκπαιδευμένου agent, πράγμα που δεν επιτρέπει η πολυπλοκότητα του Mixed-Integer Linear Programing.

Θα χρησιμοποιήσουμε την λύση που μας προσφέρει ο Mixed-Integer Linear προγραμματισμός, ως οδηγό για την ενισχυτική μάθηση, μειώνοντας τις αποκλίσεις της λύσης του πράκτορα από την βέλτιστη λύση.

Η μάσκα ενεργειών που εφαρμόζεται στον PPO μας δίνει τη δυνατότητα να επισπεύσουμε την εκπαίδευση, με τον πράκτορα να μαθαίνει γρηγορότερα μια αποδοτική πολιτική, χωρίς να χρειαστεί να αποδώσουμε αρνητική αμοιβή για ενέργειες που δεν είναι δυνατές βάσει του περιβάλλοντος. Παραδείγματος χάρη, τοποθέτηση παραπάνω του ενός instance του ίδιου microservice σε ένα node.

Εξετάζονται δύο περιβάλλοντα. Η κύρια διαφορά τους είναι ο αριθμός κόμβων που έχει το καθένα - δεκαπέντε (15) και σαράντα πέντε(45). Συνεπώς τους δόθηκε η ονομασία περιβάλλον-15 και περιβάλλον-45 αντίστοιχα.

Και τα δύο περιβάλλοντα εξυπηρετούν μία (1) εφαρμογή που αποτελείται από μια αλυσίδα τριών (3) μικροϋπηρεσιών. Καθώς έχουμε μία (1) εφαρμογή με ένα (1) αναγνωριστικό-id, αναφερόμαστε στην εφαρμογή 0 με id = 0.

5.2 Χώρος Κατάστασης – State Space

Ο χώρος κατάστασης περιέχει όλες τις πιθανές καταστάσεις του περιβάλλοντος ανά στιγμή. Ο agent για την κάθε κατάσταση γνωρίζει για το περιβάλλον σε κάθε βήμα:

- Τον αριθμό των χρηστών
- Το ποσοστό των αιτήσεων χρηστών που γίνονται αποδεκτές
- Τα replicas του microservice 1 που έχουν τοποθετηθεί τόσο από τον RL agent τόσο και από την βέλτιστη MILP λύση
- Τα replicas του microservice 2 που έχουν τοποθετηθεί τόσο από τον RL agent τόσο και από την βέλτιστη MILP λύση
- Τα replicas του microservice 3 που έχουν τοποθετηθεί τόσο από τον RL agent τόσο και από την βέλτιστη MILP λύση
- Το κόστος της τοποθέτησης του RL
- Το κόστος της τοποθέτησης της MILP λύσης
- Τους διαθέσιμους πόρους ανά κόμβο (CPU, Memory, Bandwidth)

5.3 Χώρος Ενεργειών – Action Space

Ο Reinforcement Learning agent μπορεί να πραγματοποιήσει τρία (3) είδη ενεργειών:

- Να μην κάνει τίποτα
- Να τοποθετήσει ένα αντίγραφο Si της εφαρμογής σε κάποιο συγκεκριμένο κόμβο Nj
- Να τερματίσει ένα αντίγραφο Si της εφαρμογής από κάποιο συγκεκριμένο κόμβο Nj

Η κάθε ενέργεια που μπορεί να εκτελέσει ο agent ορίζεται από ένα tuple (operation, service, node) που δηλώνει τον τύπο της ενέργειας, το microservice που σχετίζεται με αυτήν και το node στο οποίο εκτελείται.

Έστω C ο αριθμός των ειδών ενεργειών που μπορούν να εκτελεστούν. Συγκεκριμένα C=3.

Ο χώρος ενεργειών λοιπόν είναι ένας διακριτός χώρος μεγέθους:

$$C * S * N$$

Ο χώρος ενεργειών έχει σχεδιαστεί με αυτόν τον τρόπο για να διασφαλιστεί ότι ο πράκτορας μπορεί να βρει καλύτερες αποφάσεις κατανομής επιλέγοντας να αναθέσει ένα συγκεκριμένο instance μικροϋπηρεσίας σε κάποιον συγκεκριμένο κόμβο. Ο πράκτορας μπορεί να αναθέσει υπηρεσίες κατ' αυτόν τον τρόπο έως ότου όλα τα αιτήματα των χρηστών ικανοποιούνται.

Ταυτόχρονα, μπορεί να τερματίζει μικροϋπηρεσίες σε κόμβους, σε περίπτωση που δεν είναι πλέον αναγκαίες ή να μην εκτελεί καμία δράση, εάν δεν είναι αναγκαίο.

Ο στόχος μας είναι να διδάξουμε στον agent ότι ένας συγκεκριμένος αριθμός μικρο-υπηρεσιών πρέπει να διατεθεί για την κατάλληλη λειτουργία της αλυσίδας και για την υποστήριξη όλων των αιτημάτων των χρηστών.

5.4 Μάσκα Ενεργειών – Action Mask

Για να αφαιρέσουμε τις άκυρες ενέργειες από την εξέταση, χρησιμοποιήθηκε μια μάσκα δράσης, η οποία υποδεικνύει κάθε δράση ως έγκυρη ή άκυρη σε κάθε κατάσταση

Τα αποτελέσματα που συζητούνται στο [64] δείχνουν ότι η λαμβανόμενη ανταμοιβή είναι πολύ υψηλότερη όταν ο αριθμός των βημάτων εκπαίδευσης είναι μικρός, ή ισοδύναμα, ο χρόνος εκπαίδευσης είναι μικρότερος για το ίδιο επίπεδο λαμβανόμενης απόδοσης. Επομένως, αξίζει να αφαιρεθούν οι άκυρες ενέργειες, όταν αυτό είναι εφικτό.

Συγκεκριμένα, η μάσκα μας επιστρέφει έναν μονοδιάστατο πίνακα bool μήκους όσο ο χώρος ενεργειών, όπου True σημαίνει ότι η ενέργεια είναι αυτή τη στιγμή νόμιμη. Διάταξη: op ∈ {0-noop, 1-deploy, 2-terminate}, service ∈ range(0-2), node ∈ (0-14)

Συγκεκριμένα, η μάσκα υπολογίζεται σε κάθε βήμα και είναι μηδενική κατά την αρχικοποίησή της. Στη συνέχεια, για κάθε είδος ενέργειας ελέγχουμε ποιοι είναι οι νόμιμοι συνδυασμοί που μπορούν να χρησιμοποιηθούν για εκείνο το είδος ενέργειας σε εκείνο το βήμα, και προσαρμόζουμε αναλόγως την μάσκα.

Αφήνουμε μόνο μία (1) ενέργεια no-operation, καθώς δεν υπάρχει λόγος να λάβουμε υπόψιν μας σε ποιο node γίνεται το no-operation, και δεν σχετίζεται με κάποια μικρο-υπηρεσία.

Για τις ενέργειες deploy, ελέγχουμε αν ο κόμβος έχει τους διαθέσιμους πόρους για να εξυπηρετήσει την εκάστοτε μικρο-υπηρεσία Si, για κάθε i, και απαγορεύουμε την τοποθέτηση υπηρεσιών σε κόμβους που δεν μπορούν να τις υποστηρίξουν. Για τις

μικρο-υπηρεσίες που μπορεί να εξυπηρετήσει κάποιος κόμβος Nj, συνεχίζουμε στον δεύτερο έλεγχο, και ελέγχουμε αν έχουμε φτάσει τον μέγιστο αριθμό αντιγράφων (2 replicas επιτρέπονται για κάθε μικρο-υπηρεσία) της συγκεκριμένης μικρο-υπηρεσίας σε όλο το σύστημα. Τέλος, ελέγχουμε αν έχουμε αν έχουμε ήδη αντίγραφο της συγκεκριμένης μικρο-υπηρεσίας στο συγκεκριμένο κόμβο Nj, και αν ναι, πάλι απαγορεύουμε την ενέργεια deploy. Τελικά, επιτρέπονται μόνο οι ενέργειες τοποθέτησης microservice που τηρούν όλους τους περιορισμούς του συστήματος.

Για τις ενέργειες terminate, ελέγχουμε εάν υπάρχει αντίγραφο microservice Si στον κόμβο, καθώς σε διαφορετική περίπτωση, δε θα είχε νόημα ο τερματισμός της.

5.5 Συνάρτηση Επιβράβευσης – Reward Function

Η συνάρτηση επιβράβευσης ελέγχει πότε ο agent αμείβεται, και συνεπώς είναι καθοριστική για την ενισχυτική μάθηση.

Χωρίζεται σε όρους, ο καθένας από τους οποίους αντιπροσωπεύει μια διαφορετική πτυχή που θέλουμε να έχει η εκπαιδευμένη πολιτική :

1ος Όρος: R_{cost}

Αρχικά υπολογίζουμε το κόστος της τοποθέτησης του RL agent και το συγκρίνουμε με το κόστος της βέλτιστης τοποθέτησης που υπολογίζεται μέσω MILP. Ανάλογα με το πόσο μικρότερο είναι αυτό το χάσμα κόστους, αυξάνουμε την αμοιβή. Εάν 1(·) η συνάρτηση ένδειξης (1 αν αληθές, 0 αλλιώς), C_{RL} το κόστος της τοποθέτησης του RL agent και C_{MILP} το βέλτιστο κόστος αναφοράς, τότε η επιβράβευση λόγω του χάσματος κόστος υπολογίζεται ως εξής:

Έστω $g = \frac{C_{\text{MILP}} - C_{\text{RL}}}{C_{\text{MILP}}}$ το κανονικοποιημένο χάσμα κόστους

Τότε $G = \begin{cases} \text{clip}(g, -2, 2) & \alpha\nu C_{RL} > C_{MILP}, \\ -10 & \text{διαφορετικά}, \end{cases}$

Όπου $clip(x, a, b)$ η συνάρτηση κοπής έτσι ώστε $\text{clip}(x, a, b) = \begin{cases} a, & x < a, \\ x, & a \le x \le b, \\ b, & x > b. \end{cases}$

Τελικά ο όρος κόστους R_{cost} = 20 G – 20* 1 ($C_{RL} > 4C_{MILP}$)

2ος όρος: R_{match}

Έστω n=3 ο αριθμός των υπηρεσιών (εδώ 0,1,2), και ας ορίσουμε για κάθε υπηρεσία s, r_s και m_s ως τους αριθμούς αντιγράφων υπηρεσιών που τοποθέτησαν οι λύσεις RL και MILP αντιστοίχως για κάθε υπηρεσία s, τότε

$$R_{\text{match}} = \sum_{s=0}^{2} 5 \ * \ 1\ (r_s = m_s)$$

Όπου ξανά 1(·) η ενδεικτοφόρος συνάρτηση (ισούται με 1 αν ισχύει η συνθήκη, αλλιώς με 0).

3ος Όρος: P_{replica}

Ο συνολικός αριθμός αντιγράφων υπηρεσιών είναι:

$$R = \sum_{s=0}^{2} r_s\,, \quad M = \sum_{s=0}^{2} m_s$$

Τότε ο υπερβάλλων αριθμός αντιγράφων υπηρεσιών:

$$\Delta = \max\{\, R - M, 0\}$$

Χρησιμοποιείται για να επιβάλλουμε ποινή κατ' αυτόν τον τρόπο:

$$P_{\text{replica}} = -5 \max\left\{\sum_{s=0}^{2} r_s - \sum_{s=0}^{2} m_s\,, 0\right\}$$

που είναι και ο 3ος όρος.

Ο 4ος όρος βασίζεται πάνω στο βαθμό αποδοχής αιτημάτων των χρηστών, τον οποίο ιδανικά θα θέλαμε να διατηρήσουμε στο 100%.

Υπολογίζουμε το $\mathrm{R}_{\mathrm{acc}}$ πολλαπλασιάζοντας το acceptance rate που κυμαίνεται από 0 έως 100 με 0.05 ώστε να είναι στην ίδια κλίμακα με τις υπόλοιπες αμοιβές.

Έτσι $\mathrm{R}_{\mathrm{acc}}$ = 0.05 * acceptance_rate

Ο 5ος όρος αφορά κάποιες ελαφριές ποινές σε περίπτωση που η μάσκα ενεργειών αποτύχει σε κάποιο βήμα και ο agent αποπειραθεί κάποια παράνομη ενέργεια. Τότε εφαρμόζουμε μια ποινή -5 ανά περιορισμό που σχεδόν αθετήθηκε.

```
penalty = 0.0
if self.constraint_max_services_on_node:
    penalty -= 5.0
if self.constraint_add_service_max_replicas_reached:
    penalty -= 5.0
if self.constraint_terminate_service_without_deployed_services:
    penalty -= 5.0
if self.constraint_node_capacity:
    penalty -= 5.0
```

Στα πειράματα, δεν έχει τύχει να συνυπολογιστεί αυτή η ποινή, συνεπώς θα μπορούσε να αφαιρεθεί με ασφάλεια. Ωστόσο, δεν είναι περίπλοκη ούτε επιβραδύνει την εκτέλεση, και ο όρος αυτός έχει διατηρηθεί.

Έστω τέσσερις δυαδικοί (Boolean) δείκτες που ενεργοποιούνται όταν ο πράκτορας παραβιάζει τον αντίστοιχο περιορισμό στο τρέχον βήμα t

$\chi_t^{(1)}$	Υπέρβαση μέγιστου πλήθους υπηρεσιών σε κόμβο
$\chi_t^{(2)}$	Υπέρβαση μέγιστου επιτρεπτού πλήθους αντιγράφων μιας υπηρεσίας
$\chi_t^{(3)}$	Απόπειρα τερματισμού υπηρεσίας χωρίς να υπάρχουν αναπτυγμένα αντίγραφα
$\chi_t^{(4)}$	Υπέρβαση χωρητικότητας πόρων κόμβου

Έτσι, $\chi_t^{(k)} = 1$ εάν ο περιορισμός k παραβιάζεται στο βήμα t, διαφορετικά $\chi_t^{(k)} = 0$. Τότε

$$R_{\text{pen},t} = -5 \sum_{k=1}^{4} \chi_t^{(k)}$$

που είναι εν τέλει ο 5ος όρος της συνάρτησης.

Ο 6ος όρος χρησιμοποιεί ξανά τα r_s και m_s ως τους αριθμούς αντιγράφων υπηρεσιών που τοποθέτησαν οι λύσεις RL και MILP αντιστοίχως για κάθε υπηρεσία s, και συγκεκριμένα το λόγο τους $\rho_s = \frac{r_s}{m_s}$, που στην περίπτωση που μηδενίζεται το m_s, ισούται με 0, το οποίο δε συμβαίνει στην πράξη, καθώς η βέλτιστη τοποθέτηση microservices τοποθετεί τουλάχιστον ένα αντίγραφο από κάθε μια εκ των τριών ώστε να λειτουργεί ορθά η αλυσίδα.

Το μέρος που συνεισφέρει θετικά στον όρο coverage (κάλυψης) ορίζεται λοιπόν ως

$$R_{\text{coverage+}} = \sum_{s=0}^{2} (2 \min\{ \rho_s, 1\} - 1).$$

και κάθε ένας από τους τρεις (3) αυτούς όρους (για τις 3 μικρο-υπηρεσίες) κυμαίνεται από -1 έως +1.

Ταυτόχρονα, επιβάλλεται ποινή για υπηρεσίες οι οποίες δεν έχουν καθόλου κάλυψη ίση με

$$\mathrm{R}_{\text{coverage-}} = -5 \sum_{\mathrm{s}=0}^{2} 1(\rho_{\mathrm{s}} = 0),$$

όπου χρησιμοποιούμε την ενδεικτοφόρο συνάρτηση 1(·) για να εξετάσουμε εάν το πηλίκο ρ_s ισούται με 0, στην οποία περίπτωση επιβάλουμε ποινή -5.

Τελικά:

$$R_{\text{coverage}} = \sum_{s=0}^{2} (2 \min\{ \rho_s, 1\} - 1) - 5 \sum_{s=0}^{2} 1(\rho_s = 0)$$

Ο 7ος όρος, $\mathrm{R}_{\mathrm{shaping}}$ αποτελείται από δύο κομμάτια.

Το πρώτο κομμάτι χρησιμοποιείται για να κινητοποιήσει τον πράκτορα να τοποθετήσει μικροϋπηρεσίες. Είναι μια αμοιβή που είναι σχετικά μικρή σε σχέση με τις υπόλοιπες, αλλά βοηθάει στο να αρχίσει η εξερεύνηση στην αρχή της εκμάθησης. Έτσι, το $\mathrm{R}_{\mathrm{shapingE}} = 0.1$ εάν η ενέργεια είναι τύπου deploy, ενώ $\mathrm{R}_{\mathrm{shapingE}}$ = -0.05 εάν η ενέργεια είναι τύπου 0 – no operation.

Το μεγαλύτερο μέρος του $\mathrm{R}_{\mathrm{shaping}}$ είναι το δεύτερο στο οποίο προσθέτουμε πάντα 2 εάν όλα τα πηλίκα ρ_s είναι μεγαλύτερα ή ίσα του ένα και, επιπροσθέτως, επιβραβεύουμε τον πράκτορα με αμοιβή +30 εάν το κόστος της τοποθέτησής του είναι ίσο με το κόστος της βέλτιστης MILP τοποθέτησης.

```
if all(r >= 1.0 for r in ratios):
    shaping += 2 #always
    if cost_rl == cost_milp:
        shaping += 30.0
```

Ισχύει λοιπόν $R_{shapingM} = 2 + 30 * 1(C_{RL} = C_{MILP})$, αν $\rho_s \geq 1 \; \forall \; s$, ή 0 αν δεν ισχύει αυτή η συνθήκη.

Τελικά:

$$R_{shaping} = \mathrm{R_{shapingE}} + R_{shapingM}$$

Η συνάρτηση ανταμοιβής συνολικά δίνεται από τον ακόλουθο τύπο:

R = R_{cost} + R_{match} + P_{replica} + R_{acc} + R_{pen} + R_{coverage} + $R_{shaping}$

την τιμή της οποίας κάνουμε clip σε εύρος [-100, +100].

Κεφάλαιο 6: Τεχνική Υλοποίηση

Στο κεφάλαιο αυτό αναλύεται η τεχνική υλοποίηση της λύσης, και τα αποτελέσματα της εκπαίδευσης.

6.1 Προγραμματιστικό περιβάλλον

Για την υλοποίηση μας έγινε χρήση της γλώσσας προγραμματισμού Python, καθώς και αρκετών βιβλιοθηκών της.

Η εργασία πραγματοποιήθηκε με χρήση Windows 10 μέσα από ένα Windows subsystem for Linux (WSL) από το οποίο είχαμε πρόσβαση στο command line των Linux. Η εκπαίδευση και αξιολόγηση, καθώς και λοιπά scripts, εκτελέστηκαν μέσω του wsl command line.

Χρήσιμο εργαλείο για τη συγγραφή του κώδικα αποτέλεσε το Visual Studio Code, μέσα από το οποίο υλοποιήθηκαν τα αρχεία περιβάλλοντος, προσομοίωσης εκπαίδευσης, και αξιολόγησης.

6.2 Βιβλιοθήκες Python

Ο κώδικας εκπαίδευσης, προσομοίωσης και αξιολόγησης, καθώς και το περιβάλλον είναι γραμμένα σε γλώσσα Python. Έγινε χρήση αρκετών βιβλιοθηκών python τις οποίες αναλύουμε παρακάτω.

6.2.1 Stable-Baselines3

Βασική βιβλιοθήκη υπήρξε η Stable-Baselines3 (sb3) η οποία είναι ένα σύνολο αξιόπιστων υλοποιήσεων αλγορίθμων ενισχυτικής μάθησης σε PyTorch, μια βελτιστοποιημένη βιβλιοθήκη τανυστών για βαθιά μάθηση με χρήση GPU και CPU.

Οι υλοποιήσεις της Stable-Baselines3 είναι υψηλής ποιότητας. Οι αλγόριθμοι επαληθεύονται σε σχέση με δημοσιευμένα αποτελέσματα, συγκρίνοντας τις καμπύλες μάθησης των πρακτόρων. Επιπλέον, όλες οι συναρτήσεις είναι τυποποιημένες (παράμετροι και τύποι επιστροφής) και τεκμηριώνονται με συνεπές ύφος, ενώ οι περισσότερες συναρτήσεις καλύπτονται από δοκιμές μονάδας. Η ομάδα που επιβλέπει

τη βιβλιοθήκη ελέγχει ότι όλες οι αλλαγές περνούν τις δοκιμές μονάδας και τον έλεγχο τύπου, και επικυρώνει του στυλ του κώδικα και το documentation.

Το Stable-Baselines3 περιέχει τους ακόλουθους σύγχρονους on- και off-policy αλγορίθμους, που χρησιμοποιούνται συνήθως ως πειραματικές γραμμές βάσης: A2C (Mnih et al., 2016), PPO (Schulman et al., 2017), DDPG (Lillicrap et al., 2016), SAC (Haarnoja et al., 2018), TD3 (Fujimoto et al., 2018), HER (Andrychowicz et al., 2017) και DQN (Mnih et al., 2015). [65]

Συγκεκριμένα στην παρούσα διπλωματική εργασία έγινε χρήση της βιβλιοθήκης stable baselines 3 για την χρήση ενός maskable PPO αλγορίθμου ο οποίος τεκμηριώνεται στο [66].

6.2.2 Gym

Χρήσιμη ήταν επίσης η βιβλιοθήκη Gym, η οποία είναι μια βιβλιοθήκη ανοικτού κώδικα Python για την ανάπτυξη και τη σύγκριση αλγορίθμων ενισχυτικής μάθησης, παρέχοντας ένα πρότυπο API για την επικοινωνία μεταξύ αλγορίθμων μάθησης και περιβαλλόντων, καθώς και ένα πρότυπο σύνολο περιβαλλόντων που είναι συμβατά με αυτό το API [67].

Το OpenAI Gym επικεντρώνεται στο επεισοδιακό περιβάλλον της ενισχυτικής μάθησης, όπου η εμπειρία του πράκτορα αναλύεται σε μια σειρά από επεισόδια. Σε κάθε επεισόδιο, η αρχική κατάσταση του πράκτορα λαμβάνεται τυχαία από μια κατανομή και η αλληλεπίδραση συνεχίζεται μέχρι το περιβάλλον να φτάσει σε μια τελική κατάσταση. Ο στόχος στην επεισοδιακή ενισχυτική μάθηση είναι η μεγιστοποίηση της προσδοκίας της συνολικής ανταμοιβής ανά επεισόδιο και η επίτευξη ενός υψηλού επίπεδου απόδοσης σε όσο το δυνατόν λιγότερα επεισόδια [68].

6.2.3 Numpy

Το NumPy είναι το θεμελιώδες πακέτο για επιστημονικούς υπολογισμούς στην Python. Πρόκειται για μια βιβλιοθήκη Python που παρέχει ένα αντικείμενο πολυδιάστατων πινάκων, διάφορα παράγωγα αντικείμενα (όπως οι masked πίνακες) και μια σειρά από ρουτίνες για γρήγορες λειτουργίες σε πίνακες, συμπεριλαμβανομένων μαθηματικών, λογικών, χειρισμού σχημάτων, ταξινόμησης, επιλογής, εισόδου/εξόδου, διακριτών μετασχηματισμών Fourier, βασικής γραμμικής άλγεβρας, βασικών στατιστικών πράξεων, τυχαίας προσομοίωσης και πολλά άλλα [69].

Χρησιμοποιήθηκε για δημιουργία και πράξεις πινάκων, unraveling, clipping, λήψη τυχαίας επιλογής και όχι μόνο.

6.3 Μεταβλητές Περιβάλλοντος

Ενώ είχαμε αναφερθεί πριν σε κάποιες από αυτές, εδώ αναφέρουμε όλες τις μεταβλητές του περιβάλλοντος που προσομοιώθηκε, έτσι ώστε να υπολογιστεί η βέλτιστη λύση και να εκπαιδευτεί η πολιτική του πράκτορα.

Εξετάσθηκαν δύο τοπολογίες edge-fog, το περιβάλλον-15 και το περιβάλλον-45. Η κύρια διαφορά μεταξύ τους είναι ο αριθμός τον κόμβων - διαθέτουν 15 και 45 αντίστοιχα. Το καθένα από τα δύο, καθώς και οι τιμές που το περιγράφουν αναλύονται εν συνεχεία.

6.3.1 Περιβάλλον 15

Οι τιμές για την τοπολογία αυτή υιοθετήθηκαν από το περιβάλλον που χρησιμοποιήθηκε από τους Santos J. et al [47] έτσι ώστε να μπορούν να χρησιμοποιηθούν τα αποτελέσματά της έρευνάς τους ως βάση σύγκρισης. Η μόνη τιμή που μεταβάλλαμε είναι αυτή των μέγιστων χρηστών στο δυναμικό σενάριο, καθώς ο ILP έβρισκε λύσεις μόνο μέχρι τριάντα δύο (32) χρήστες. Αυτό μπορεί να οφείλεται σε έλλειψη υπολογιστικής δύναμης του υλικού που χρησιμοποιήθηκε.

Στην έρευνα εξετάστηκε συνολική έκταση 324 km^2 . Η υποδομή fog-cloud αναπτύσσεται σε πέντε τοποθεσίες L, όπου είναι δυνατή η κατανομή μικρο-υπηρεσιών. Κάθε τοποθεσία διαχειρίζεται ένα σύνολο τριών κόμβων, ο καθένας με δική του υπολογιστική ικανότητα καθώς και βάρος το οποίο χρησιμοποιείται για να υπολογιστεί το ολικό κόστος του συστήματος [47].

Μεταβλητή	**Τιμή**
Αριθμός Κόμβων	15
Ονομασία Κόμβου	'worker1', 'worker2', 'worker3', 'worker4', 'worker5', 'worker6', 'worker7', 'worker8', 'worker9', 'worker10', 'worker11', 'worker12', 'worker13', 'worker14', 'worker15'
Αριθμός Τοποθεσιών	5

Αριθμός Υπηρεσιών	3
Αριθμός Replicas	2
CPU Κόμβου	2.0, 2.0, 1.0, 2.0, 1.0, 2.0, 2.0, 2.0, 1.0, 2.0, 2.0, 2.0, 6.0, 6.0, 8.0
Μνήμη Κόμβου	4.0, 4.0, 2.0, 4.0, 2.0, 4.0, 4.0, 4.0, 2.0, 4.0, 4.0, 4.0, 16.0, 16.0, 24.0
Bandwidth Κόμβου	10.0, 10.0, 5.0, 10.0, 5.0, 10.0, 10.0, 10.0, 5.0, 10.0, 10.0, 10.0, 30.0, 30.0, 30.0
Βάρος Κόμβου	2.0, 2.0, 1.0, 2.0, 1.0, 2.0, 2.0, 2.0, 1.0, 2.0, 2.0, 2.0, 3.0, 3.0, 3.0
Τοποθεσία κόμβου	2, 2, 2, 0, 0, 0, 3, 3, 3, 1, 1, 1, 4, 4, 4
Ονομασία Τοποθεσίας	'l1', 'l2', 'l3', 'l4', 'l5'
Μέγιστοι Χρήστες	32*
Ελάχιστοι Χρήστες	1

Πίνακας 6.1: Πίνακας Μεταβλητών Περιβάλλοντος-15

Πιο συγκεντρωμένα, ο παρακάτω πίνακας δείχνει τα χαρακτηριστικά του κάθε κόμβου:

Κόμβος	**CPU (cpu)**	**RAM(Mi)**	**BAND(Mbit/s)**	**Βάρος Κόμβου**
Worker 1	2.0	4.0	10.0	2.0
Worker 2	2.0	4.0	10.0	2.0
Worker 3	1.0	2.0	5.0	1.0
Worker 4	2.0	4.0	10.0	2.0
Worker 5	1.0	2.0	5.0	1.0
Worker 6	2.0	4.0	10.0	2.0
Worker 7	2.0	4.0	10.0	2.0
Worker 8	2.0	4.0	10.0	2.0
Worker 9	1.0	2.0	5.0	1.0
Worker 10	2.0	4.0	10.0	2.0

Worker 11	2.0	4.0	10.0	2.0
Worker 12	2.0	4.0	10.0	2.0
Worker 13	6.0	16.0	30.0	3.0
Worker 14	6.0	16.0	30.0	3.0
Worker 15	8.0	24.0	30.0	3.0

Πίνακας 6.2: Πίνακας Κόμβων περιβάλλοντος-15

Οι μεταβλητές που ορίζουν τις τρείς μικρο-υπηρεσίες παρουσιάζονται περαιτέρω στον ακόλουθο πίνακα:

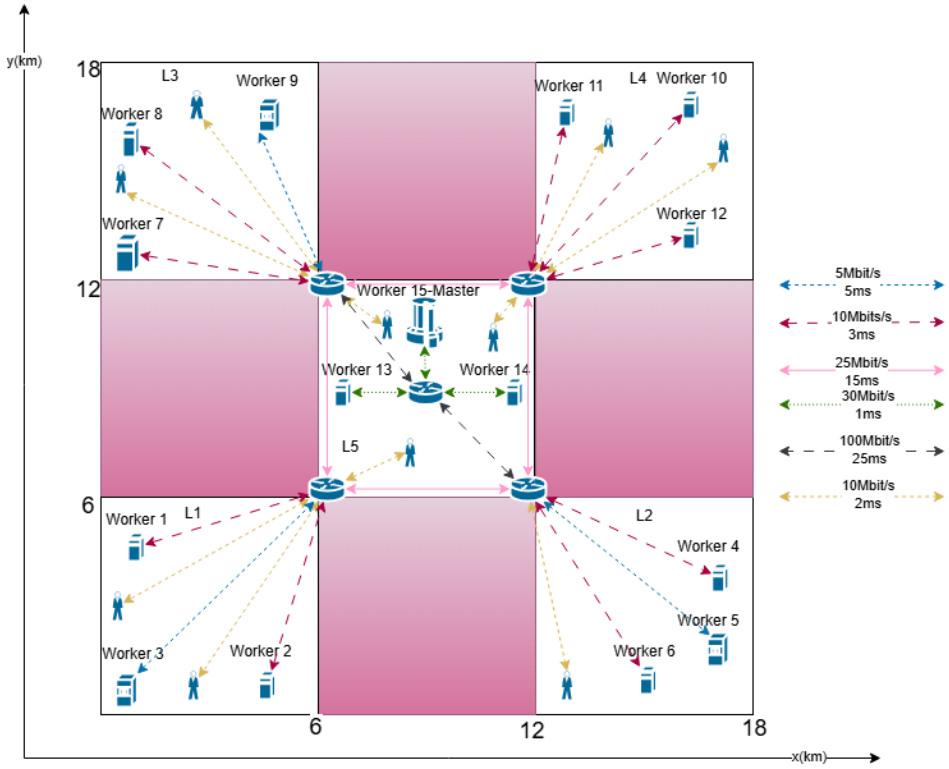
Μεταβλητή	**Τιμή 1ης υπηρεσίας**	**Τιμή 2ης υπηρεσίας**	**Τιμή 3ης υπηρεσίας**
Όνομα Υπηρεσίας	Serv 1	Serv 2	Serv 3
CPU Υπηρεσίας	0.25	0.5	0.5
Μνήμη Υπηρεσίας	0.25	1.0	1.0
Bandwidth Υπηρεσίας	4.0	8.0	5.0

Πίνακας 6.3: Πίνακας Τιμών Υπηρεσιών

Η δομή της αλυσίδας SFC για την εφαρμογή είναι $s_1 \rightarrow s_2 \rightarrow s_3$

Ο χώρος ενεργειών τελικά είναι: $C * S * N = 3 * 3 * 15 = 135$

Υπενθυμίζουμε πως η υποδομή του δικτύου ομίχλης μπορεί να αναπαρασταθεί ποιοτικά ως εξής:



Εικόνα 11: Υποδομή Fog Cloud που Χρησιμοποιήθηκε

6.3.2 Περιβάλλον 45

Όπως προαναφέρθηκε, στο περιβάλλον-45 εξετάζουμε μια τοπολογία 45 nodes, γεγονός που επιφέρει αλλαγές και σε άλλες τιμές. Οι πίνακες 6.1 και 6.2 δεν ισχύουν για το περιβάλλον-45, αλλά συνεχίζουμε να χρησιμοποιούμε μία εφαρμογή με 3 microservices και το SFC chain, όπως ορίζει ο πίνακας 6.3. Ο μέγιστος αριθμός replicas έχει πλέον αυξηθεί σε πέντε, καθώς υπάρχουν περισσότεροι κόμβοι στους οποίους μπορούν να γίνουν deploy.

Εξετάζουμε πλέον ένα δίκτυο που εκτείνεται σε εννέα τοποθεσίες L, με πέντε κόμβους ανά τοποθεσία, όπως φαίνεται παρακάτω.

Μεταβλητή	**Τιμή**
Αριθμός Κόμβων	45
Ονομασία Κόμβου	'worker1', 'worker2', 'worker3', 'worker4', 'worker5', 'worker6', 'worker7', 'worker8', 'worker9', 'worker10', 'worker11', 'worker12', 'worker13', 'worker14', 'worker15', 'worker16', 'worker17', 'worker18', 'worker19', 'worker20', 'worker21', 'worker22', 'worker23', 'worker24', 'worker25', 'worker26', 'worker27', 'worker28', 'worker29', 'worker30', 'worker31', 'worker32', 'worker33', 'worker34', 'worker35', 'worker36', 'worker37', 'worker38', 'worker39', 'worker40', 'worker41', 'worker42', 'worker43', 'worker44', 'worker45'
Αριθμός Τοποθεσιών	9
Αριθμός Υπηρεσιών	3
Αριθμός Replicas	5
CPU Κόμβου	1.0, 2.0, 2.0, 2.0, 1.0, # L1 1.0, 2.0, 2.0, 2.0, 1.0, # L2 1.0, 2.0, 2.0, 2.0, 1.0, # L3 1.0, 2.0, 2.0, 2.0, 1.0, # L4 8.0, 8.0, 8.0, 8.0, 8.0, # L5 1.0, 2.0, 2.0, 2.0, 1.0, # L6

	1.0, 2.0, 2.0, 2.0, 1.0, # L7 1.0, 2.0, 2.0, 2.0, 1.0, # L8 1.0, 2.0, 2.0, 2.0, 1.0 # L9
Μνήμη Κόμβου	2.0, 4.0, 4.0, 4.0, 2.0, # L1 2.0, 4.0, 4.0, 4.0, 2.0, # L2 2.0, 4.0, 4.0, 4.0, 2.0, # L3 2.0, 4.0, 4.0, 4.0, 2.0, # L4 16.0, 16.0, 16.0, 16.0, 16.0, # L5 2.0, 4.0, 4.0, 4.0, 2.0, # L6 2.0, 4.0, 4.0, 4.0, 2.0, # L7 2.0, 4.0, 4.0, 4.0, 2.0, # L8 2.0, 4.0, 4.0, 4.0, 2.0 # L9,
Bandwidth Κόμβου	5.0, 10.0, 10.0, 10.0, 5.0, # L1 5.0, 10.0, 10.0, 10.0, 5.0, # L2 5.0, 10.0, 10.0, 10.0, 5.0, # L3 5.0, 10.0, 10.0, 10.0, 5.0, # L4 30.0, 30.0, 30.0, 30.0, 30.0, # L5 5.0, 10.0, 10.0, 10.0, 5.0, # L6 5.0, 10.0, 10.0, 10.0, 5.0, # L7 5.0, 10.0, 10.0, 10.0, 5.0, # L8 5.0, 10.0, 10.0, 10.0, 5.0] # L9
Βάρος Κόμβου	1.0, 2.0, 2.0, 2.0, 1.0, # L1 1.0, 2.0, 2.0, 2.0, 1.0, # L2 1.0, 2.0, 2.0, 2.0, 1.0, # L3 1.0, 2.0, 2.0, 2.0, 1.0, # L4 3.0, 3.0, 3.0, 3.0, 3.0, # L5 1.0, 2.0, 2.0, 2.0, 1.0, # L6

	1.0, 2.0, 2.0, 2.0, 1.0, # L7 1.0, 2.0, 2.0, 2.0, 1.0, # L8 1.0, 2.0, 2.0, 2.0, 1.0] # L9
Τοποθεσία κόμβου	0,0, 0, 0, 0,1,1,1,1,1,2,2,2,2,2,3,3 3, 3, 3, 4, 4, 4,4,4,5,5,5,5,5,6,6,6,6,6,7,7,7,7,7,8,8,8,8,8
Ονομασία Τοποθεσίας	'l1', 'l2', 'l3', 'l4', 'l5', 'l6', 'l7', 'l8', 'l9'
Μέγιστοι Χρήστες	32*
Ελάχιστοι Χρήστες	1

Πίνακας 6.4: Πίνακας Μεταβλητών Περιβάλλοντος-45

Το περιβάλλον-45 περιπλέκει το πρόβλημα που πρέπει να μάθει να λύνει ο πράκτορας, καθώς με περισσότερους κόμβους μεγαλώνει τόσο ο χώρος κατάστασης όσο και ο χώρος ενεργειών. Η μάσκα ενεργειών αποτελεί εδώ κρίσιμο στοιχείο, επισπεύδοντας την εκπαίδευση σημαντικά.

Πιο συγκεντρωμένα, τα χαρακτηριστικά του κάθε κόμβου φαίνονται στον παρακάτω πίνακα.

Κόμβος	**Τοποθεσία**	**CPU (cpu)**	**RAM(Mi)**	**BAND(Mbit/s)**	**Βάρος Κόμβου**
worker1	l1	1.0	2.0	5.0	1.0
worker2	l1	2.0	4.0	10.0	2.0
worker3	l1	2.0	4.0	10.0	2.0
worker4	l1	2.0	4.0	10.0	2.0
worker5	l1	1.0	2.0	5.0	1.0
worker6	l2	1.0	2.0	5.0	1.0
worker7	l2	2.0	4.0	10.0	2.0
worker8	l2	2.0	4.0	10.0	2.0
worker9	l2	2.0	4.0	10.0	2.0
worker10	l2	1.0	2.0	5.0	1.0
worker11	l3	1.0	2.0	5.0	1.0
worker12	l3	2.0	4.0	10.0	2.0
worker13	l3	2.0	4.0	10.0	2.0
worker14	l3	2.0	4.0	10.0	2.0
worker15	l3	1.0	2.0	5.0	1.0
worker16	l4	1.0	2.0	5.0	1.0
worker17	l4	2.0	4.0	10.0	2.0
worker18	l4	2.0	4.0	10.0	2.0
worker19	l4	2.0	4.0	10.0	2.0

worker20	l4	1.0	2.0	5.0	1.0
worker21	l5	8.0	16.0	30.0	3.0
worker22	l5	8.0	16.0	30.0	3.0
worker23	l5	8.0	16.0	30.0	3.0
worker24	l5	8.0	16.0	30.0	3.0
worker25	l5	8.0	16.0	30.0	3.0
worker26	l6	1.0	2.0	5.0	1.0
worker27	l6	2.0	4.0	10.0	2.0
worker28	l6	2.0	4.0	10.0	2.0
worker29	l6	2.0	4.0	10.0	2.0
worker30	l6	1.0	2.0	5.0	1.0
worker31	l7	1.0	2.0	5.0	1.0
worker32	l7	2.0	4.0	10.0	2.0
worker33	l7	2.0	4.0	10.0	2.0
worker34	l7	2.0	4.0	10.0	2.0
worker35	l7	1.0	2.0	5.0	1.0
worker36	l8	1.0	2.0	5.0	1.0
worker37	l8	2.0	4.0	10.0	2.0
worker38	l8	2.0	4.0	10.0	2.0
worker39	l8	2.0	4.0	10.0	2.0
worker40	l8	1.0	2.0	5.0	1.0
worker41	l9	1.0	2.0	5.0	1.0
worker42	l9	2.0	4.0	10.0	2.0
worker43	l9	2.0	4.0	10.0	2.0
worker44	l9	2.0	4.0	10.0	2.0
worker45	l9	1.0	2.0	5.0	1.0

Πίνακας 6.5: Πίνακας Κόμβων περιβάλλοντος-45

Ο χώρος ενεργειών τελικά είναι: $C * S * N = 3 * 3 * 45 = 405$.

6.4 Υπερ-παράμετροι και λοιπές μεταβλητές Εκπαίδευσης

Η βιβλιοθήκη SB3 τεκμηριώνει και χρησιμοποιεί HyperParameters (υπερ-παραμέτρους) οι οποίοι επηρεάζουν την διαδικασία της εκπαίδευσης [70]. Ακολουθεί ένας πίνακας που αναφέρει τις τιμές των HyperParameters που χρησιμοποιήσαμε κατά την εκπαίδευση του maskable PPO agent για όλα τα περιβάλλοντα και σενάρια, καθώς και σύντομες εξηγήσεις και σχόλια.

Υπερ-παράμετρος	**Τιμή που χρησιμοποιήθηκε**	**Τι ρυθμίζει**	**Σχόλιο**
policy	MlpPolicy	Τον τύπο νευρωνικού δικτύου (εδώ feed-forward MLP). παράγει κοινή βάση + δύο κεφάλια (actor/vf)	Απλό & ταχύ για state-vector. Δυο Κεφάλια
Learning_rate - Ρυθμός Εκμάθησης	3e-4	Τον Ρυθμό Εκμάθησης	Μεγάλο βήμα σημαίνει αστάθεια, μικρό βήμα σημαίνει αργή μάθηση· το 3e-4 είναι σχετικά ασφαλές για PPO.
n_steps	2048	Μήκος rollout πριν από κάθε update	Μεγαλύτερο ⇒ λιγότερο θόρυβο στα advantages, αλλά απαιτεί RAM & πιο αραιά updates.
batch_size	64	Μέγεθος minibatch στα updates	2048/64 = 32 ενημερώσεις. ανά rollout· ισορροπεί ταχύτητα/θόρυβο.
n_epochs	10	Αριθμός εποχών κατά τη βελτιστοποίηση της υποκατάστατης απώλειας	Περισσότερες εποχές σημαίνει καλύτερη εκμετάλλευση δεδομένων αλλά, ενέχει τον κίνδυνο υπερ-βελτιστοποίησης σε ξεπερασμένες τροχιές
gamma	0.99	Συντελεστής έκπτωσης	Καθορίζει πόσο "μακριά" βλέπει ο πράκτορας (~45 βήματα εδώ)
gae_lambda	0.95	Συντελεστής για την αντιστάθμιση της μεροληψίας έναντι της διακύμανσης για τον εκτιμητή γενικευμένου πλεονεκτήματος	Bias–variance αντιστάθμιση μεταξύ TD και Monte-Carlo.

clip_range	0.20	Παράμετρος περικοπής- για τη συνάρτηση αξίας	Περιορίζει μεγάλα άλματα πολιτικής, σταθεροποιεί PPO.
ent_coef	0.01	Συντελεστής εντροπίας για τον υπολογισμό του loss.	Ενθαρρύνει δοκιμές διαφορετικών νόμιμων ενεργειών.
vf_coef	0.5	Συντελεστής συνάρτησης αξίας για τον υπολογισμό του loss.	Ισορροπεί μάθηση actor–critic.
max_grad_norm	0.5	Η μέγιστη τιμή για την περικοπή κλίσης	Προστασία από σπάνια gradient spikes.
net_arch	[64, 64]	Δομή MLP: 2 κρυφά στρώματα	Αρκετή εκφραστικότητα χωρίς πολλές παραμέτρους (~13 k).
activation_fn	$Tanh$	Μη-γραμμικότητα ανά στρώμα	Ομαλή, περιορισμένη έξοδος – σταθερή αρχικοποίηση.
ortho_init	True	Ορθογώνια αρχικοποίηση βαρών	Εξισορροπεί τις αρχικές κλίσεις.
seed	42	Σπόρος για τις ψεύδο-τυχαίες γεννήτριες αριθμών.	Χρησιμοποιήθηκε ο σπόρος για ακριβή αναπαραγωγή των πειραμάτων.
normalize_advantage	True	Αν θα ομαλοποιηθεί ή όχι το πλεονέκτημα A_t πριν το back-prop	Επιπτώσεις στη σταθερότητα της κλίσης

Πίνακας 6.6: Πίνακας Υπερπαραμέτρων

Αρχικά εξετάστηκαν δύο περιπτώσεις. Στην πρώτη περίπτωση, οι χρήστες παραμένουν σταθερά δέκα (10) καθ' όλη τη διάρκεια της εκπαίδευσης και αξιολόγησης, ενώ στη δεύτερη, ανά τακτά χρονικά διαστήματα, οι χρήστες αυξάνονται ή μειώνονται.

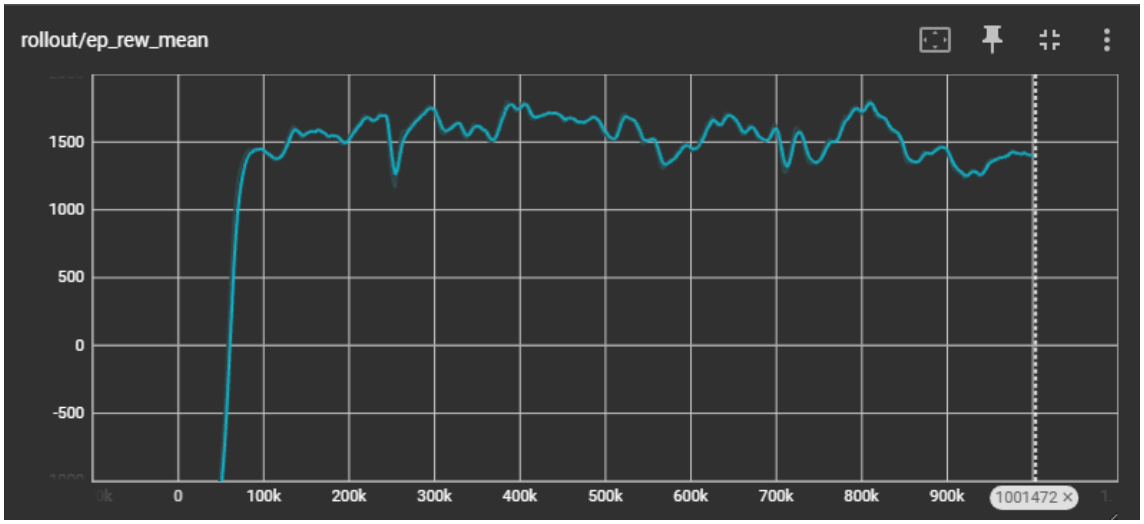
Παρακάτω αναλύονται τα αποτελέσματα του Deep Reinforcement Learning agent.

Για τη παρακολούθηση των διαφόρων metrics χρησιμοποιήθηκε TensorBoard, που παρέχει την οπτικοποίηση και τα εργαλεία που απαιτούνται για πειραματισμό μηχανικής μάθησης.

7.1 Στατικό Πειραματικό Σενάριο – Περιβάλλον 15

Στο στατικό σενάριο ο αριθμός των χρηστών παραμένει σταθερός και ίσος με δέκα (10). Ο πράκτορας αρκεί να μάθει μια πολιτική που διαχειρίζεται αυτό το φορτίο. Σκοπός μας είναι να μειώσουμε το κόστος όσο το δυνατό περισσότερο, καθώς και να κρατήσουμε το ποσοστό αποδοχής στο μέγιστο, δηλαδή 100%.

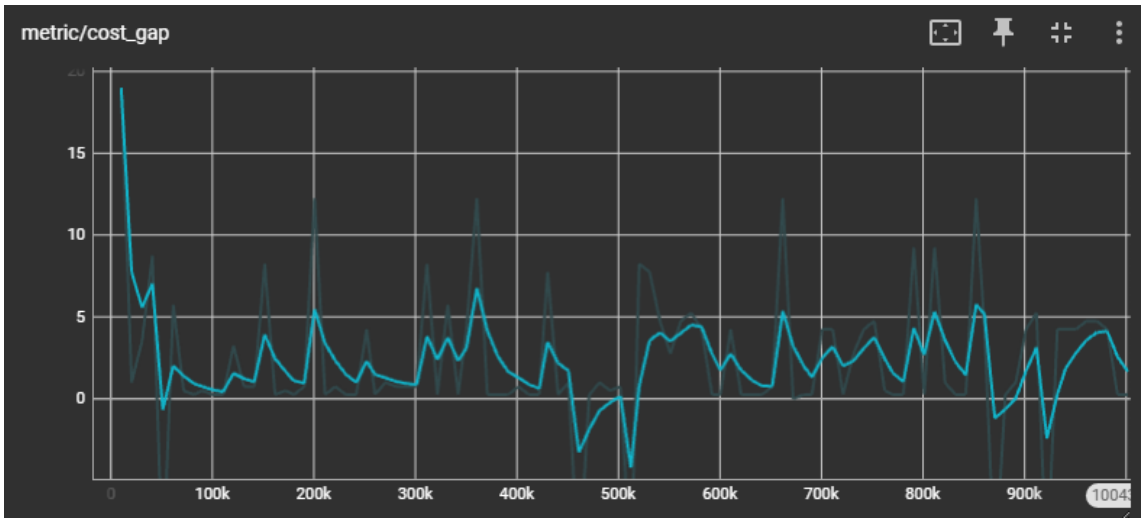
Η εκπαίδευση διήρκεσε 1.000.000 timesteps (χρονικά βήματα) αν και η πολιτική φαίνεται να καταστάλαξε λίγο μετά τα 100.000 βήματα καθώς μετά από τότε η μέση αμοιβή ανά 2048 βήματα φαίνεται να σταθεροποιείται.



Εικόνα 12: Μέση Ανταμοιβή ανά 2048 βήματα στην εκπαίδευση για το στατικό σενάριο του περιβάλλοντος-15

Το χάσμα κόστους μεταξύ MILP λύσης και της λύσης του πράκτορα μικραίνει επίσης πολύ γρήγορα, πετυχαίνοντας τιμές κόστους μόλις 0.25 μονάδες περισσότερες από την βέλτιστη κατανομή μικρο-υπηρεσιών. Αυτό είναι ένα κόστος μόλις 2.32% ακριβότερο από το καλύτερο δυνατό σενάριο το οποίο συγκεκριμένα για δέκα (10) χρήστες στο δίκτυο αυτό ήταν 10.75 μονάδες κόστους.

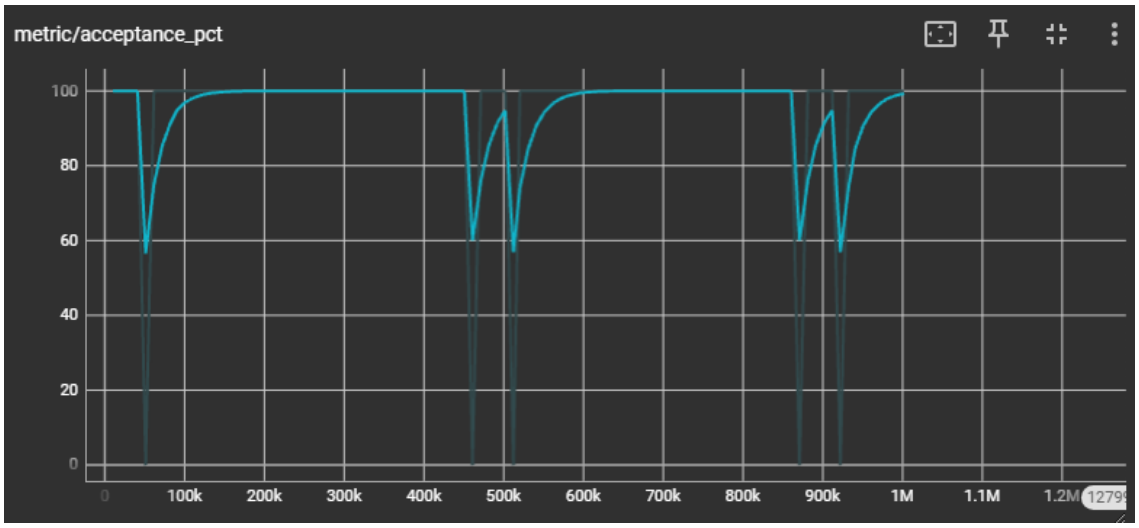
Μετρώντας τη μεταβολή του cost gap ανά 10.000 βήματα, βλέπουμε ότι κατά μέσο όρο η λύση του RL agent ήταν κατά 2.195 μονάδες ακριβότερη, που σημαίνει ότι η λύση είναι 20.42% χειρότερη από τη βέλτιστη. Σαφώς ο μέσος όρος αυτός είναι επηρεασμένος από τα πρώιμα στάδια της εκπαίδευσης του πράκτορα, κατά τα οποία εξερευνάει και έχει ένα μεγάλο χάσμα κόστους.



Εικόνα 13: Χάσμα Κόστους ανά 10.000 βήματα στην εκπαίδευση για το στατικό σενάριο του περιβάλλοντος-15

Ο πράκτορας μαθαίνει αρκετά γρήγορα μια πολιτική που διατηρεί το ποσοστό αποδοχής αιτημάτων των χρηστών στο 100%, δηλαδή χτίζει πλήρως μια λειτουργική αλυσίδα υπηρεσιών $s_1 \rightarrow s_2 \rightarrow s_3$ που μπορεί να εξυπηρετήσει και τους δέκα (10) χρήστες.

Τα σημεία που έχουν τιμή 0 μπορούν να αποδοθούν στο γεγονός ότι, όταν λάβαμε την τιμή του acceptance percentage, είχε επαναφερθεί μόλις το περιβάλλον σε μια αρχική κατάσταση, και συνεπώς δεν είχε γίνει deploy καμία υπηρεσία.



Εικόνα 14: Ποσοστό αποδοχής αιτημάτων χρηστών ανά 10.000 βήματα στην εκπαίδευση για το στατικό σενάριο του περιβάλλοντος-15

7.2 Δυναμικό Πειραματικό Σενάριο – Περιβάλλον-15

Στο δυναμικό σενάριο ο πράκτορας καλείται να ανταπεξέλθει σε ένα φόρτο χρηστών ο οποίος συνεχώς μεταβάλλεται μεταξύ 1 και 32.

Σε ένα περιβάλλον υπολογιστών ομίχλης, η ζήτηση των χρηστών μπορεί να αυξομειώνεται με πολύ διαφορετικούς τρόπους. Εμείς εξετάσαμε τρείς:

Πιθανότητα σύνδεσης/αποσύνδεσης ενός χρήστη κάθε φορά

Ξαφνικές εκρηκτικές αιχμές ή πτώσεις (π.χ. πλήθος χρηστών, διακοπές λειτουργίας)

Ομαλοί περιοδικοί κύκλοι (π.χ. ημερήσια μοτίβα, χρονοδιαγράμματα δέσμης εργασιών)

Για να δημιουργήσουμε μια στιβαρή πολιτική τοποθέτησης μικροϋπηρεσιών, πρέπει να δοκιμάσουμε τον αλγόριθμο RL σε κάθε τύπο μεταβολής. Ως εκ τούτου, υλοποιήσαμε τρεις "full dynamic" λειτουργίες στο περιβάλλον:

Προεπιλεγμένη: ±1 χρήστης ανά συμβάν με σταθερές πιθανότητες

Poisson: τυχαία μεγέθη άλματος που προέρχονται από κατανομές Poisson

Wave: συνεχείς ημιτονοειδείς διακυμάνσεις συν θόρυβο

Προς το παρόν χρησιμοποιείται ένας δυναμικός τρόπος μεταβολής χρηστών τον οποίο θα ονομάσουμε προεπιλεγμένο δυναμικό τρόπο στο 7.2 και 7.4. Αυτός ο τρόπος

χρησιμοποιήθηκε επίσης από τους Santos J. et al [47], και συνεπώς αποτελεί καλό σημείο αναφοράς για να κρίνουμε τα αποτελέσματα των πειραμάτων.

Στα υποκεφάλαια 7.5, 7.6 και 7.7 αναλύονται οι άλλοι δύο μέθοδοι καθώς και τα αποτελέσματα της εκπαίδευσης των πρακτόρων που τις χρησιμοποιούν.

Στην προεπιλεγμένη μέθοδο, ο αριθμός των χρηστών αρχικοποιείται στους δέκα (10) και κάθε πέντε (5) βήματα, ο αριθμός αυτός τυχαία:

Είτε αυξάνεται (50% των περιστάσεων)

Είτε μειώνεται (15% των περιστάσεων)

Είτε μένει σταθερός (35% των περιστάσεων)

Αν Δ_u η μεταβολή των χρηστών ανα 5 βήματα, τότε ισχύει:

$$\Delta_u \in \{-1,0,+1\} \quad \mu\varepsilon \quad P(\Delta=-1)=0{,}15, \quad P(\Delta=0)=0{,}35, \quad P(\Delta=+1)=0{,}50$$

Η μέση τιμή μεταβολής (drift) είναι +0,35 χρήστες ανά 5 βήματα, και υπολογίζεται ως εξής:

$$E[\Delta] = (-1)\cdot 0{,}15 + 0\cdot 0{,}35 + (+1)\cdot 0{,}50 = 0{,}35.$$

Το drift λοιπόν ανά βήμα, είναι $\delta = \frac{0.35}{5} = 0.07.$

Ο αναμενόμενος αριθμός χρηστών σε ένα βήμα t είναι λοιπόν

$$E[U_t] = U_0 + 0.07t, \quad U_0 = 10$$

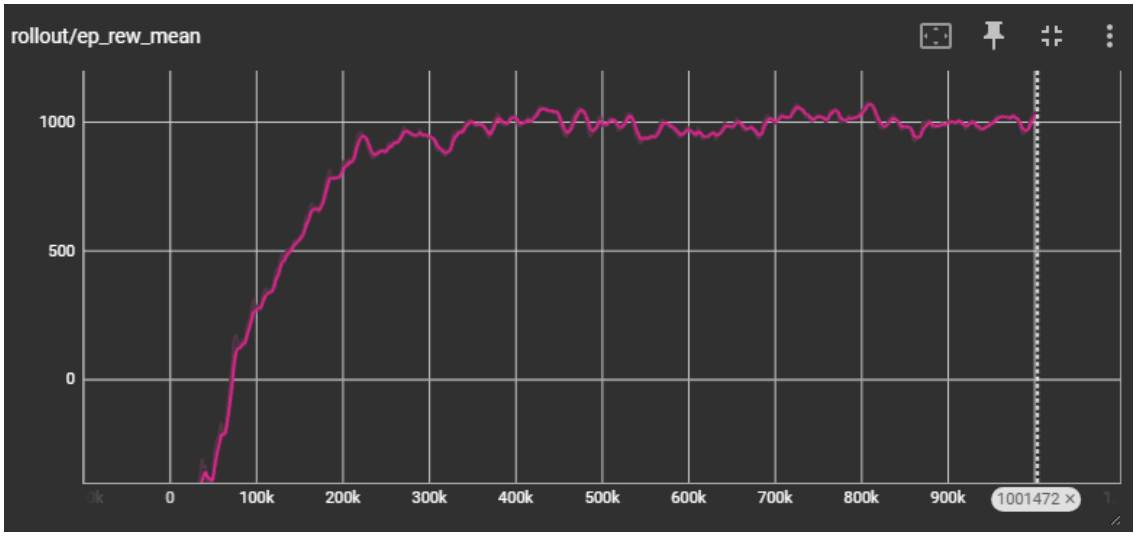
Η μέση τιμή σε κάθε επεισόδιο (100 βημάτων) αναμένεται να είναι:

$$\frac{1}{100}\sum_{t=1}^{100}(10+0.07\,t) = 10 + 0.07\cdot\frac{100\cdot\frac{101}{2}}{100} = 10 + 0.07\cdot 50.5 \approx 13.535.$$

Το δυναμικό σενάριο είναι σαφώς πιο περίπλοκο από το στατικό, και προσεγγίζει καλύτερα πραγματικές συνθήκες δικτύου ομίχλης. Σε ένα δυναμικό σενάριο ο πράκτορας πρέπει να μάθει μια πολιτική η οποία συνεχώς προσαρμόζεται στις συνθήκες του δικτύου.

Για το δυναμικό σενάριο δημιουργήθηκε ένα επιπλέον script αξιολόγησης το οποίο χρησιμοποιούσε τον εκπαιδευμένο πλέον πράκτορα για εκατό (100) επεισόδια εκατό (100) βημάτων, με πέντε διαφορετικές τιμές seed (σπόρου), έτσι ώστε να

αξιολογήσουμε την επίδοσή του. Τα αποτελέσματα αυτού του script συζητούνται στη συνέχεια παράλληλα με τα δεδομένα που παράχθηκαν κατά την εκπαίδευση.



Εικόνα 15: Μέση Ανταμοιβή ανά 2048 βήματα στην εκπαίδευση για το δυναμικό σενάριο του περιβάλλοντος-15

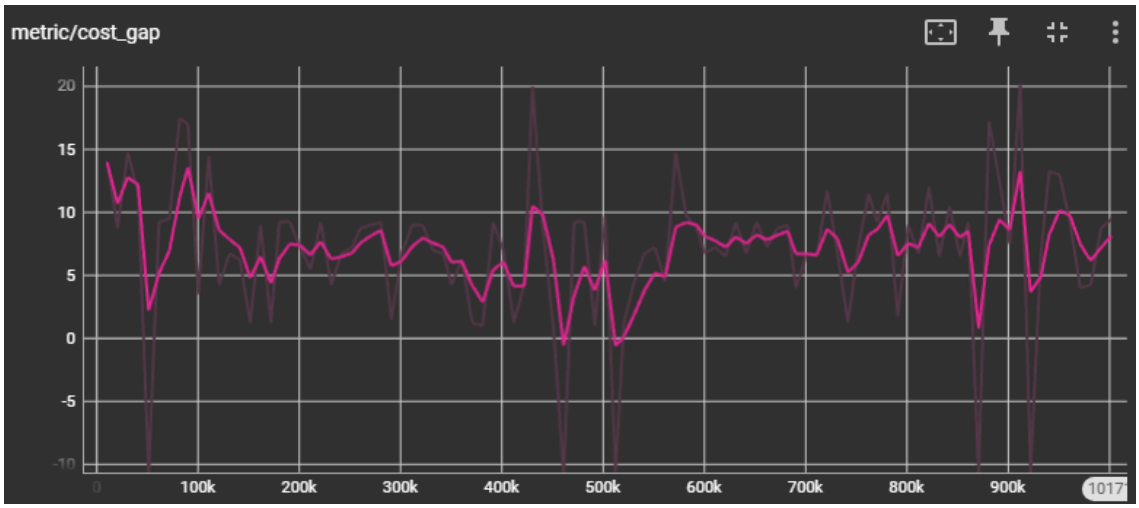
Η αμοιβή σταθεροποιείται αργότερα σε σχέση με το στατικό σενάριο, μετά από 350.000 βήματα, και κυμαίνεται σε μικρότερα επίπεδα. Αυτό είναι ένα λογικό αποτέλεσμα, λαμβάνοντας υπόψιν την αύξηση της δυσκολίας της εργασίας προς εκμάθηση. Η μικρότερη μέση αμοιβή ανά 2048 βήματα υποδηλώνει ότι ο πράκτορας δεν φέρνει εις πέρας όλους τους στόχους, όπως προηγουμένως.

Όσον αφορά το χάσμα κόστους, υπάρχει αναμενόμενη αύξηση. Με περισσότερους χρήστες να χρησιμοποιούν το δίκτυο, είναι αναγκαίο να τοποθετηθούν περισσότερες μικρο-υπηρεσίες, και συνεπώς η βέλτιστη λύση έχει μεγαλύτερο κόστος. Το καλύτερο χάσμα κόστους που πέτυχε ο πράκτορας στο δυναμικό σενάριο κατά την εκπαίδευση ήταν μία (1) μονάδα κόστους, και δεδομένου ότι για τριάντα δύο (32) χρήστες το βέλτιστο κόστος ήταν 18.20, η DRL λύση ήταν μόλις 5.5% χειρότερη από την ιδανική.

Όταν πραγματοποιήσαμε αξιολόγηση του εκπαιδευμένου πλέον πράκτορα, λάβαμε τα εξής αποτελέσματα από τα οποία φαίνεται ότι το μέσο χάσμα κόστους ήταν 7.07 μονάδες, δηλαδή 38.85% χειρότερη από την ιδανική κατά μέσο όρο:

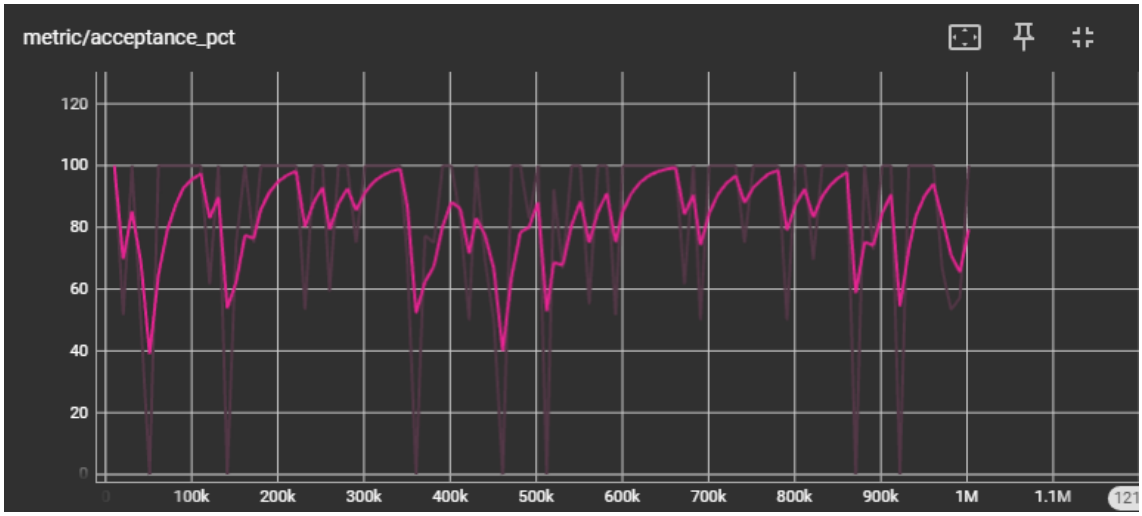
policy	**Μέσος Αριθμός Αντιγράφων**	**Μέσο Κόστος RL**	**Μέσο Κόστος MILP**	**Μέσο Χάσμα Κόστους**	**Μέσος Αριθμός Χρηστών**
PPO-Dynamic-15	4.95	25.27	18.20	7.07	24.47

Πίνακας 7.1



Εικόνα 16: Χάσμα Κόστους ανά 10.000 βήματα στην εκπαίδευση για το δυναμικό σενάριο του περιβάλλοντος-15

Το ποσοστό αποδεχομένων αιτημάτων αυξομειώνεται κατά τη διάρκεια της εκπαίδευσης, σε αντίθεση με την σχετικά σταθερή του 100% τιμή κατά το στατικό σενάριο, αλλά κατά κύριο λόγο παραμένει πάνω από 70%, όπως φαίνεται στο παρακάτω γράφημα.



Εικόνα 17: Ποσοστό αποδοχής αιτημάτων χρηστών ανά 10.000 βήματα στην εκπαίδευση για το δυναμικό σενάριο του περιβάλλοντος-15

Όταν προχωρήσαμε στην αξιολόγηση του πράκτορα, παρατηρήθηκε ότι το μέσο ποσοστό αποδοχής είναι 90.64%, το οποίο είναι 30% ανώτερο από το 60% ποσοστό αποδοχής που παρουσιάζουν οι Santos et. al [47] με χρήση του αλγορίθμου Q-Learning

σε παρόμοιες συνθήκες. Οι λόγοι των microservices s_0 $s_1 s_2$ που δηλώνουν πόσες υπηρεσίες έκανε deploy ο πράκτορας σε σχέση με την λύση MILP παραμένουν κατά μέσο όρο κοντά στο ένα (1).

Ελέγχθηκε εάν επιχειρήθηκαν παράνομες ενέργειας από τον πράκτορα και επιβεβαιώθηκε πως ο μέσος αριθμός παράνομων ενεργειών διατηρήθηκε καθ' όλη τη διάρκεια της αξιολόγησης στο μηδέν.

policy	**Μέσος Αριθμός Παράνομων Ενεργειών**	**Μέσο Ποσοστό Αποδοχής**	**Μέσος Λόγος s_0**	**Μέσος Λόγος s_1**	**Μέσος Λόγος s_2**
PPO-Dynamic-15	0.0	90.64%	1.10	0.91	1.04

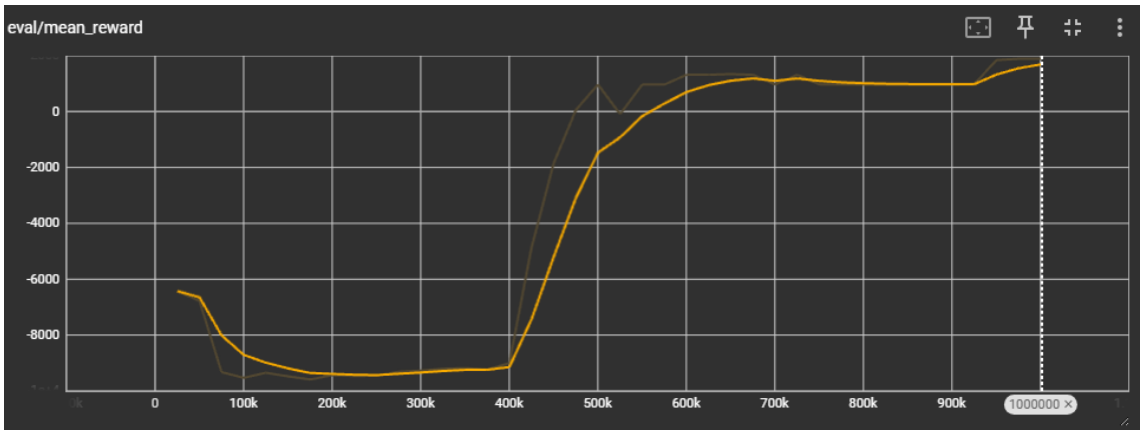
Πίνακας 7.2

Η αναγκαιότητα της βαθιάς ενισχυτικής μάθησης για την επίλυση του προβλήματος που πραγματεύεται η παρούσα διπλωματική εργασία γίνεται εμφανής με την αύξηση των χρηστών. Οι λύσεις ILP σταματούν να είναι βιώσιμες, λόγου του κόστους αλλά και του χρόνου υπολογισμού. Ακόμη και για τριάντα δύο (32) χρήστες, με μία εφαρμογή τριών (3) microservices, σε περιβάλλον δεκαπέντε (15) κόμβων, ο χρόνος υπολογισμού της MILP λύσης αυξήθηκε ραγδαία σε σχέση με το στατικό σενάριο των δέκα (10) χρηστών.

7.3 Στατικό Πειραματικό Σενάριο – Περιβάλλον-45

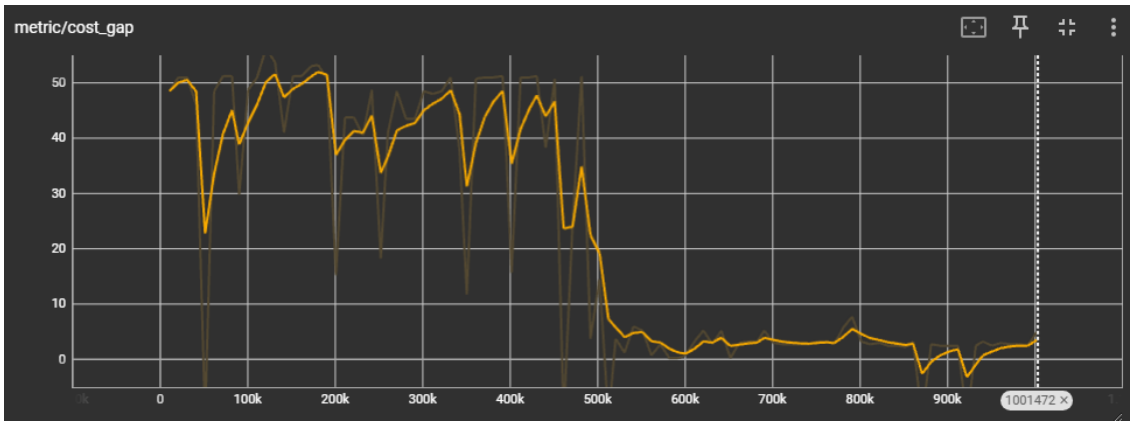
Διατηρείται ξανά ο αριθμός των χρηστών σταθερός και ίσος με 10, καθώς πραγματοποιούμε εκπαίδευση για 1.000.000 βήματα.

Αυτή τη φορά, σε σχέση με το Περιβάλλον-15, χρειάστηκαν περισσότερα επεισόδια προτού ο agent σταματήσει να κάνει σημαντικές μεταβολές στην πολιτική του. Παρατηρούμε ότι αυτό γίνεται μετά τα 600.000 βήματα, ενώ στο περιβάλλον-15 χρειάστηκαν περίπου 100.000. Η αύξηση της πολυπλοκότητας ήταν αναμενόμενο να επιφέρει καθυστέρηση στην εκμάθηση.



Εικόνα 18: Μέση Ανταμοιβή ανά 2048 βήματα στην εκπαίδευση για το στατικό σενάριο του περιβάλλοντος-45

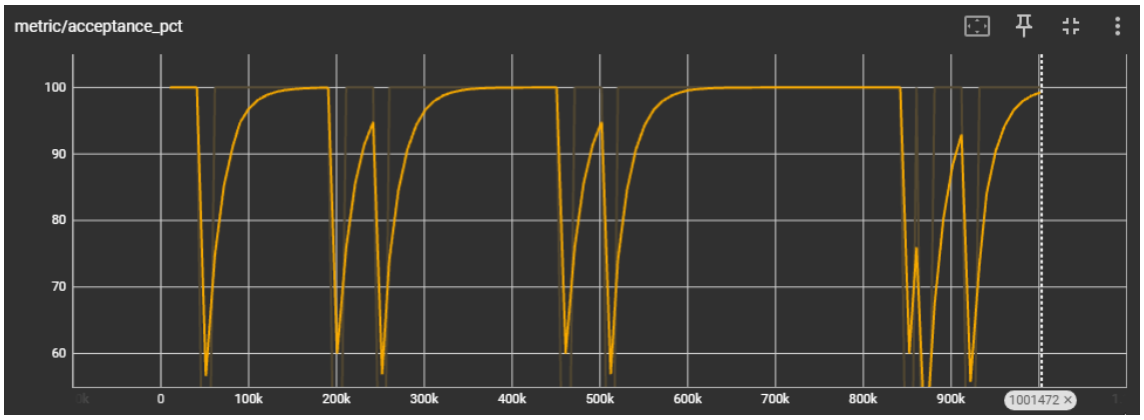
Το χάσμα κόστους μεταξύ MILP λύσης και της λύσης του πράκτορα μικραίνει επίσης ραγδαία μετά τα 600.000 βήματα, πετυχαίνοντας τιμές κόστους μόλις 2.5 μονάδες περισσότερες από την βέλτιστη κατανομή μικρο-υπηρεσιών.



Εικόνα 19: Χάσμα Κόστους ανά 10.000 βήματα στην εκπαίδευση για το στατικό σενάριο του περιβάλλοντος-45

Ο πράκτορας και σε αυτό το περιβάλλον μαθαίνει αρκετά γρήγορα μια πολιτική που διατηρεί το ποσοστό αποδοχής αιτημάτων των χρηστών στο 100%, δηλαδή χτίζει πλήρως μια λειτουργική αλυσίδα υπηρεσιών $s_1 \rightarrow s_2 \rightarrow s_3$ που μπορεί να εξυπηρετήσει και τους δέκα (10) χρήστες.

Τα σημεία που έχουν τιμή 0 μπορούν να αποδοθούν στο γεγονός ότι όταν λάβαμε την τιμή του acceptance percentage είχε επαναφερθεί μόλις το περιβάλλον σε μια αρχική κατάσταση, και συνεπώς δεν είχε γίνει deploy καμία υπηρεσία.



Εικόνα 20: Ποσοστό αποδοχής αιτημάτων χρηστών ανά 10.000 βήματα στην εκπαίδευση για το στατικό σενάριο του περιβάλλοντος-45

7.4 Δυναμικό Πειραματικό Σενάριο – Περιβάλλον-45

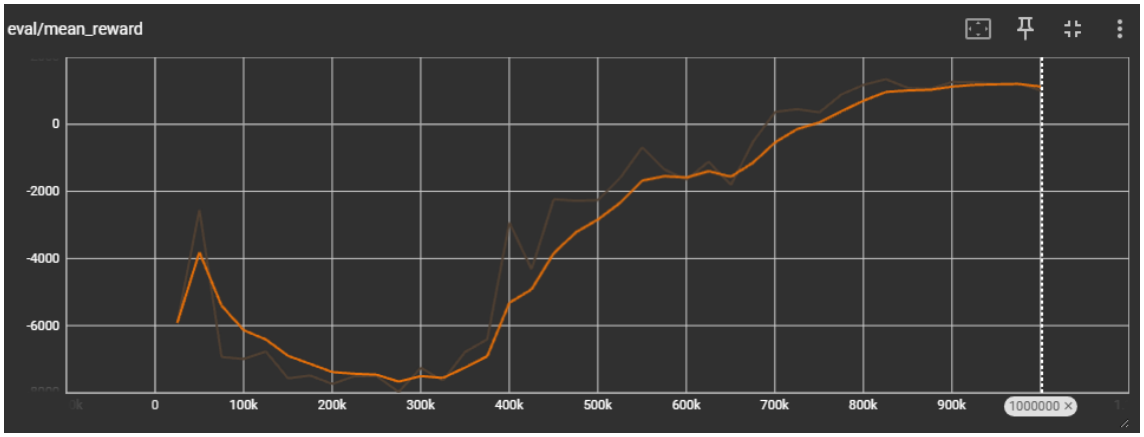
Υπενθυμίζουμε ότι στο προεπιλεγμένο δυναμικό σενάριο ο πράκτορας καλείται να ανταπεξέλθει σε ένα φόρτο χρηστών ο οποίος συνεχώς μεταβάλλεται μεταξύ 1 και 32. Ο αριθμός των χρηστών αρχικοποιείται στους δέκα (10) και κάθε πέντε (5) βήματα, ο αριθμός αυτός τυχαία:

Είτε αυξάνεται (50% των περιστάσεων)

Είτε μειώνεται (15% των περιστάσεων)

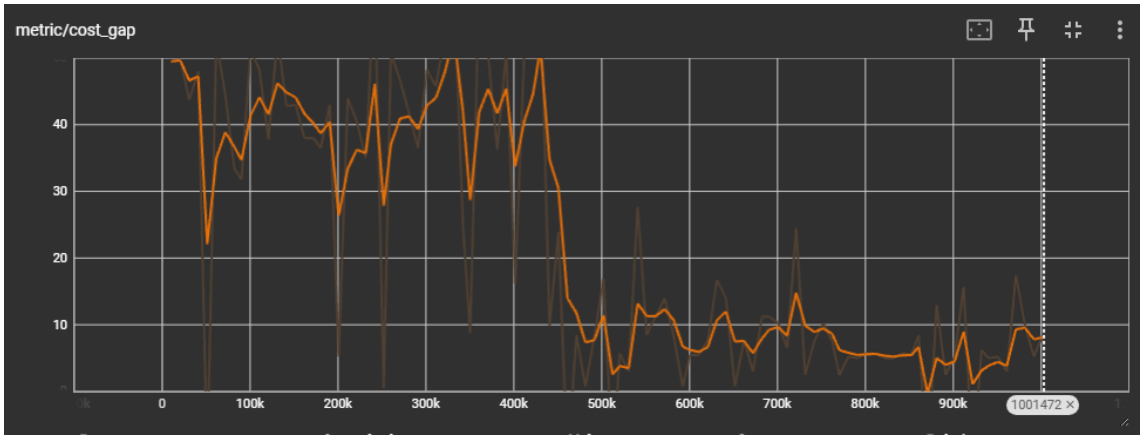
Είτε μένει σταθερός (35% των περιστάσεων)

Η μέση αμοιβή στο δυναμικό σενάριο των 45 κόμβων φαίνεται να σταθεροποιείται μετά τα 800.000 βήματα, περισσότερα δηλαδή και από το δυναμικό σενάριο του περιβάλλοντος-15 αλλά και το στατικό του περιβάλλοντος-45.



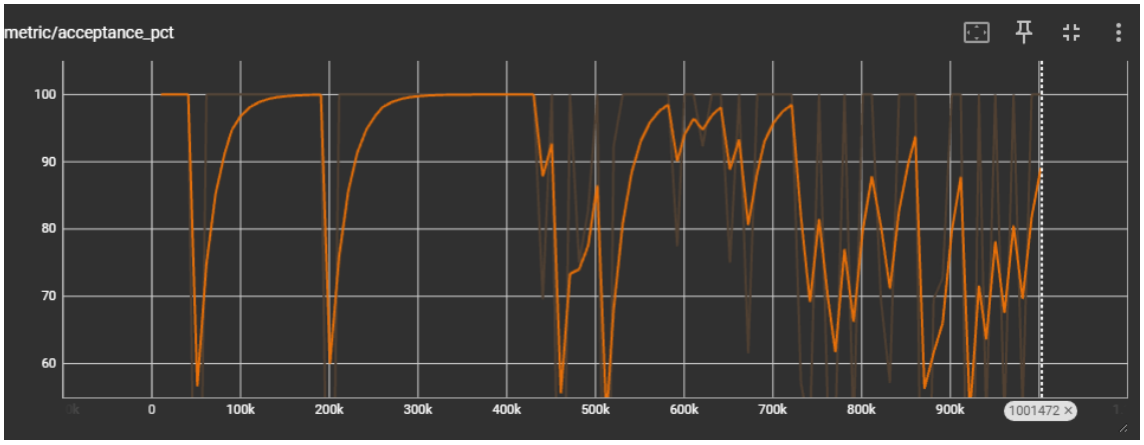
Εικόνα 21: Μέση Ανταμοιβή ανά 2048 βήματα στην εκπαίδευση για το δυναμικό σενάριο του περιβάλλοντος-45

Το χάσμα κόστους μειώνεται δραστικά περίπου στα 500.000 βήματα, που ο πράκτορας μαθαίνει να μην κάνει υπερβολικά deploy. Ύστερα από τότε, η διαφορά κόστους της τοποθέτησης του agent από τη βέλτιστη, παραμένει κατά κύριο λόγο κάτω από τις 10 μονάδες, ενώ επιτυγχάνεται χάσμα κόστους ίσο με 2.5 μονάδες σε συγκεκριμένα επεισόδια.



Εικόνα 22: Χάσμα Κόστους ανά 10.000 βήματα στην εκπαίδευση για το δυναμικό σενάριο του περιβάλλοντος-45

Όπως και στο δυναμικό σενάριο του περιβάλλοντος-15, το ποσοστό αποδοχής παραμένει πάνω από 70%. Όσον αφορά τα διαστήματα 100% αποδοχής, ο πράκτορας δεν είχε μάθει ακόμη να μην κάνει over-deploy, σπαταλώντας πόρους, αλλά διατηρώντας μια λειτουργική αλυσίδα υπηρεσιών.



Εικόνα 23: Ποσοστό αποδοχής αιτημάτων χρηστών ανά 10.000 βήματα στην εκπαίδευση για το δυναμικό σενάριο του περιβάλλοντος-45

Εξετάζοντας τα αποτελέσματα του script αξιολόγησης, λαμβάνουμε τα εξής δεδομένα των ακόλουθων δύο (2) πινάκων: Ο μέσος αριθμός replicas, παραμένει περίπου 5, ενώ

το μέσο χάσμα κόστους έχει μειωθεί κατά δύο μονάδες σε σχέση με το δυναμικό σενάριο του περιβάλλοντος-15. Ένα μέσο χάσμα κόστους 5.09 μονάδων, όταν το μέσο κόστος MILP είναι 19.09, σημαίνει ότι κατά μέσο όρο ο πράκτορας πραγματοποίησε μια ανάθεση των υπηρεσιών η οποία ήταν κατά 26.66% πιο δαπανηρή από την ιδανική.

Ο αριθμός των παράνομων ενεργειών παραμένει στο 0, πράγμα που σημαίνει ότι η μάσκα λειτουργεί ανεξαρτήτως του περιβάλλοντος, ενώ το μέσο ποσοστό αποδοχής κυμαίνεται στο 77.95%.

policy	Μέσος Αριθμός Αντιγράφων	Μέσο Κόστος RL	Μέσο Κόστος MILP	Μέσο Χάσμα Κόστους
PPO-Dynamic-45	4.90	24.18	19.09	5.09

Πίνακας 7.3

policy	Μέσος Αριθμός Παράνομων Ενεργειών	Μέσο Ποσοστό Αποδοχής	Μέσος Λόγος s_0	Μέσος Λόγος s_1	Μέσος Λόγος s_2
PPO-Dynamic-45	0.0	77.95%	0.75	1.05	1.00

Πίνακας 7.4

7.5 Δυναμικό Πειραματικό Σενάριο με αφίξεις Poisson – Περιβάλλον-45

Σε αυτό το πείραμα, δε χρησιμοποιούμε πλέον τον προεπιλεγμένο δυναμικό τρόπο μεταβολής χρηστών, αλλά γίνεται χρήση μιας κατανομής Poisson με $\lambda_{\text{inc}} = 2.0$ αφίξεων και $\lambda_{\text{dec}} = 1.5$ αναχωρήσεων.

Μοντελοποιούμε εκρηκτική κίνηση με αυτόν τον τρόπο: κατά μέσο όρο +2 αφίξεις και -1,5 αναχωρήσεις ανά διάστημα, που σημαίνει περιστασιακές μεγάλες διακυμάνσεις.

Οι τιμές των λ προσομοιώνουν μέτριες εκρήξεις, με ελαφρώς περισσότερες αφίξεις από ό,τι αναχωρήσεις κατά μέσο όρο, για να δοκιμάσουμε την κλιμάκωση υπό πίεση.

Ανά πέντε βήματα έχουμε ένα update

$$E[\Delta] = \lambda_{\text{inc}} - \lambda_{\text{dec}} = 2.0 - 1.5 = 0.5$$

Το drift ανά βήμα είναι λοιπόν

$$\delta = \frac{0.5}{5} = 0.10$$

και συνεπώς ο αναμενόμενος αριθμός χρηστών στο βήμα t είναι:

$$E[U_t] = 10 + 0.10t$$

Για T = 100 περιμένουμε τον αριθμό των χρηστών:

$$E[U_{100}] = 10 + 0.10 * 100 = 20$$

Η μέση τιμή χρηστών σε κάθε επεισόδιο (100 βημάτων) αναμένεται να είναι:

$$10 + 0.10 \cdot \frac{100 \cdot \frac{101}{2}}{100} = 10 + 0.10 \cdot 50.5 = 10 + 5.05 = 15.05$$

7.6 Δυναμικό Πειραματικό Σενάριο με κυματικές αφίξεις – Περιβάλλον-45

Σε αυτό το πείραμα, δε χρησιμοποιούμε πλέον τον προεπιλεγμένο δυναμικό τρόπο μεταβολής χρηστών, αλλά γίνεται χρήση μιας κυματοειδούς συνάρτησης έτσι ώστε

$$U_t = U_{min} + A\left(1 + \sin\left(\frac{2\pi t}{T}\right)\right) + \mathcal{N}(0, (\sigma)^2)$$

Όπου A = $\frac{U_{max} - U_{min}}{2} = \frac{32 - 1}{2} = 15.5,\ T = 50, \sigma = 0.1A$

Η μέση τιμή χρηστών αναμένεται να είναι $\frac{U_{max} + U_{min}}{2}$, δηλαδή 16.5.

Ο θόρυβος σ είναι ίσος με 0.1 * A προσθέτοντας ρεαλιστικό jitter γύρω από το τέλειο ημιτονοειδές κύμα της συνάρτησης.

7.7 Σύγκριση Αποτελεσμάτων των τριών μεθόδων διαχείρισης χρηστών

Τρεις πράκτορες εκπαιδεύτηκαν, κρατώντας τον ίδιο σπόρο (seed) για 1.000.000 βήματα ο καθένας, στα δυναμικά σενάρια, στο περιβάλλον-45. Κάθε πείραμα χρησιμοποίησε διαφορετική μέθοδο αυξομείωσης χρηστών, έτσι ώστε να αξιολογήσουμε τον maskable PPO αλγόριθμο σε διαφορετικές συνθήκες.

Ακολουθούν τα διαγράμματα που συγκρίνουν τα κύρια metrics της εκπαίδευσης βαθιάς ενισχυτικής μάθησης στα τρία αυτά σενάρια. Τα χρώματα των τριών γραφικών παραστάσεων είναι ως εξής:

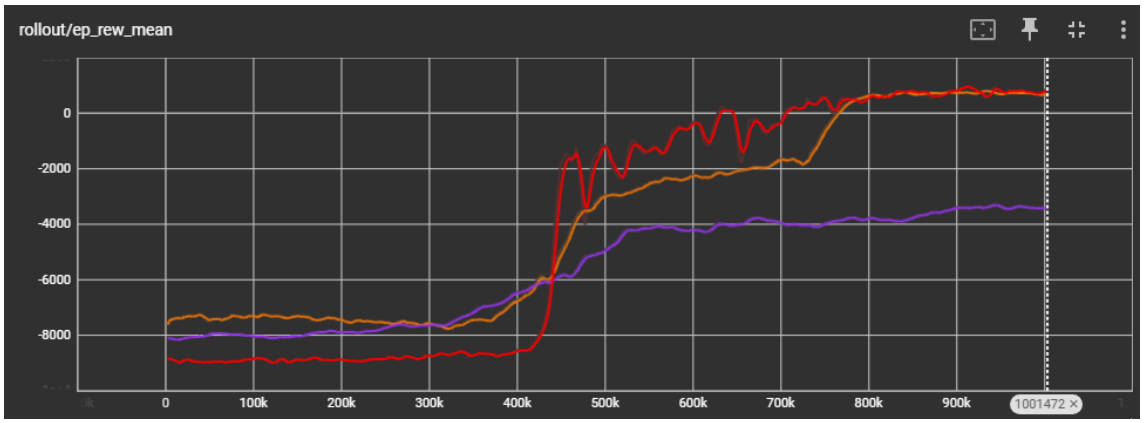
Default	
Poisson	
Wave	

Πίνακας 7.5

Αρχικά, όσον αφορά την αμοιβή, το προεπιλεγμένο σενάριο και το poisson είχαν πολύ παρόμοια αποτελέσματα μετά τα 800.000 βήματα, διατηρώντας μια συνολική αμοιβή περίπου 700 μονάδων ανά επεισόδιο. Κατά την εκπαίδευση του wave σεναρίου, ο πράκτορας δεν κατάφερε ποτέ να λάβει θετικά επίπεδα ανταμοιβής.

Αυτό ήταν αναμενόμενο καθώς στο default και στο poisson σενάριο, ο αριθμός χρηστών "δρα" με drift (0.35 ή 0.5 ανά update) προς τα πάνω, οπότε ο πράκτορας μαθαίνει σταδιακά να αυξάνει τα replicas του και σταθεροποιεί τη συμπεριφορά του.

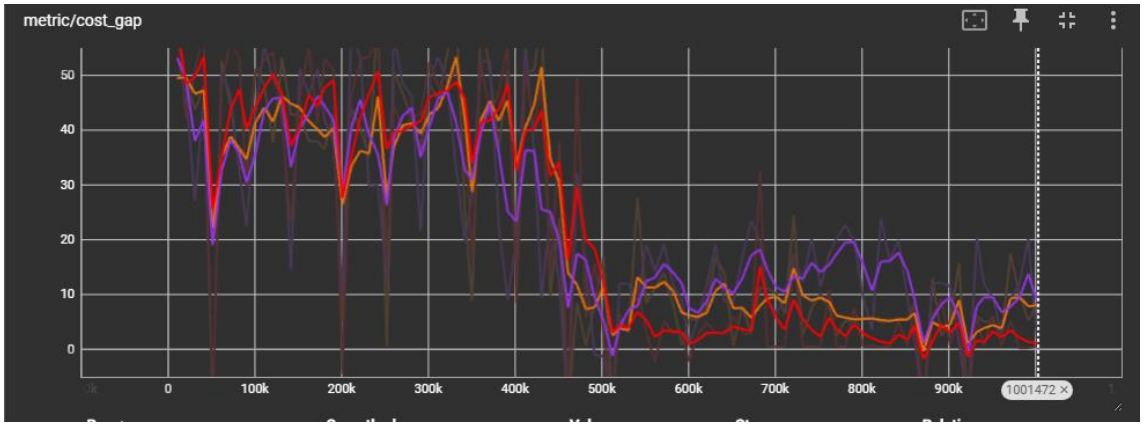
Στο wave, ο πληθυσμός ταλαντώνεται γύρω από τη γραμμή $\frac{U_{max}+U_{min}}{2}$ με περίοδο T=50 βημάτων και προσθήκη θορύβου. Αυτό δημιουργεί ένα δυσκολότερο μοτίβο σε σχέση με τα άλλα δύο σενάρια, από το οποίο είναι δύσκολο να εξαχθεί ένας σταθερός κανόνας τοποθέτησης. Η πολιτική στο σενάριο wave δε φαίνεται να έχει σημαντικές βελτιώσεις μετά τα 550.000 βήματα, σε αντίθεση με τα υπόλοιπα δύο σενάρια, στα οποία παρατηρείται σημαντική αύξηση της μέσης ανταμοιβής μετά τα 400.000 βήματα, και ξανά ύστερα από τα 700.00 βήματα, έως ότου η πολιτική κατασταλάζει μετά τα 800.00 βήματα.



Εικόνα 24: Σύγκριση Μέσης Ανταμοιβής ανά 2048 βήματα για τις τρείς μεθόδους διαχείρισης χρηστών στο περιβάλλον-45

Το χάσμα κόστους, όπως μπορούμε να καταλάβουμε και από την μέση ανταμοιβή, κυμαίνεται σε παρόμοια επίπεδα για τις πρώτες δύο μεθόδους, ενώ είναι μεγαλύτερο στο σενάριο που χρησιμοποιεί την wave μέθοδο. Στο σενάριο Poisson, ο πράκτορας φαίνεται να έμαθε την πολιτική που μειώνει το κόστος όσο το δυνατό περισσότερο, πετυχαίνοντας τιμές μόλις 0.25 πιο δαπανηρές από τη βέλτιστη λύση σε ορισμένα επεισόδια.

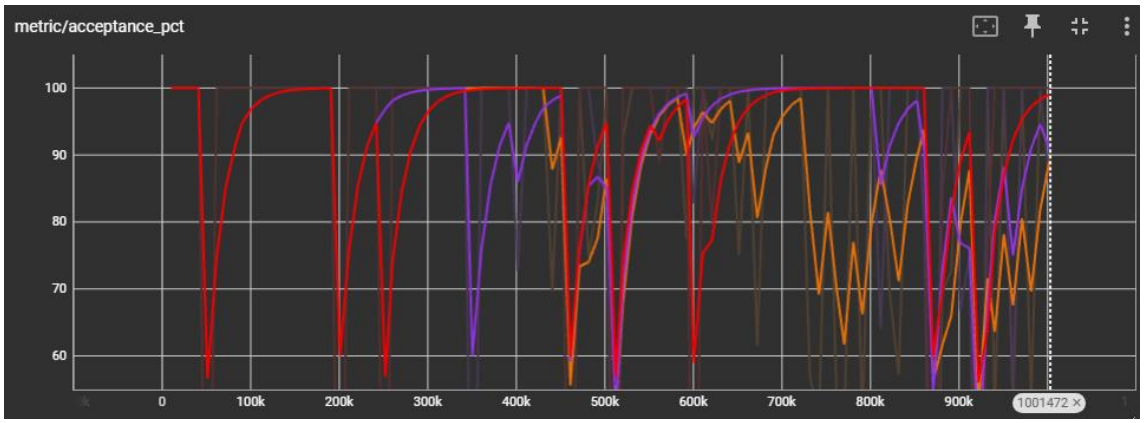
Όπως προαναφέρθηκε, στο προεπιλεγμένο σενάριο η χαμηλότερη τιμή χάσματος κόστους που επιτεύχθηκε ήταν 2.5, ενώ στο wave scenario 3.5.



Εικόνα 25: Σύγκριση Χάσματος Κόστους ανά 10.000 βήματα στην εκπαίδευση για τις τρείς μεθόδους διαχείρισης χρηστών περιβάλλοντος-45

Το ποσοστό των αιτημάτων χρηστών που γίνονται αποδεκτά ακολουθεί παρόμοια πορεία στα τρία σενάρια. Παρατηρούμε ότι μέχρι να μειωθεί το χάσμα κόστους και να αυξηθεί η αμοιβή (στα 500-600 χιλιάδες βήματα περίπου), και τα τρία σενάρια διατηρούν σχεδόν πάντα 100% το ποσοστό αυτό, πράγμα που μπορεί να σημαίνει ότι κάνουν υπερ-ανάθεση, σπαταλώντας άσκοπα πόρους. Μετά από το σημείο αυτό, στο προεπιλεγμένο σενάριο, παρατηρείται μείωση του ποσοστού, αν και παραμένει πάνω από 60%. Στα υπόλοιπα δυο σενάρια, ο πράκτορας κατά κύριο λόγο επιτυγχάνει 100% acceptance rate, και αποκλίνει από την τιμή αυτή μόνο σε συγκεκριμένα σημεία, που μπορεί να αποδοθεί σε σφάλμα δειγματοληψίας, μετρώντας δηλαδή το acceptance rate προτού ο πράκτορας να έχει προλάβει να αναθέσει μικρο-υπηρεσίες, καθώς βρίσκεται στην αρχή κάποιου επεισοδίου.

Όσον αφορά το ποσοστό αυτό, ο πράκτορας φαίνεται να διαχειρίζεται καλύτερα εκ των τριών σεναρίων το Poisson σενάριο, και χειρότερα το προεπιλεγμένο σενάριο.



Εικόνα 26: Ποσοστό αποδοχής αιτημάτων χρηστών ανά 10.000 βήματα στην για τις τρείς μεθόδους διαχείρισης χρηστών περιβάλλοντος-45

7.8 Σύγκριση χρόνου εκτέλεσης MILP και PPO

Δημιουργήθηκε ένα script αξιολόγησης για να εξετάσουμε και να καταγράψουμε τους χρόνους εκτέλεσης της Mixed Integer Linear Programming λύσης, καθώς και τον μέσο χρόνο εκτέλεσης ενός επεισοδίου με τη χρήση του εκπαιδευμένου PPO agent.

Λάβαμε τα εξής αποτελέσματα για το χρόνο εκτέλεσης ανά χρήστη, σε δευτερόλεπτα:

User Count	MILP Execution Time	PPO Episode Run Time
1	0.0850	0.5405
2	0.1490	0.5450
3	0.2130	0.5480
4	0.2660	0.5225
5	0.2710	0.5250
6	0.2960	0.6013
7	0.3550	0.5800
8	0.4110	0.6252
9	0.3810	0.5850
10	0.4750	0.5880
11	0.4770	0.5900
12	0.5400	0.5920
13	0.6020	0.5940

14	0.6260	0.5950
15	0.7060	0.5934
16	0.6920	0.5228
17	6.9850	0.6500
18	10.1250	0.6597
19	8.1020	0.6400
20	6.7830	0.6313
21	11.2520	0.6600
22	8.6840	0.5857
23	7.7060	0.6412
24	7.9970	0.6300
25	10.3010	0.6086
26	11.9510	0.6316
27	12.6730	0.6702
28	11.5040	0.6400
29	10.7450	0.6450
30	18.4320	0.6550
31	11.9800	0.5902
32	23.7300	0.6053
Average	5.7970	0.6030

Πίνακας 7.6

Παρατηρούμε πως ο χρόνος των επεισοδίων του αλγορίθμου PPO παραμένει σταθερός (περίπου 0.60 s ανά επεισόδιο).

Αυτό επιβεβαιώνει ότι το κόστος του πράκτορα δεν εξαρτάται από τον αριθμό μεταβλητών της ILP, και ο αλγόριθμος μπορεί να χρησιμοποιηθεί σε μεγαλύτερα περιβάλλοντα.

Παρατηρείται επίσης εκθετική αύξηση του χρόνου εκτέλεσης του μοντέλου MILP μετά τους 15-16 χρήστες.

- 1-15 χρήστες: κάτω από 1 s − πρακτικά κατάλληλο για near-real-time.
- 17-23 χρήστες: 7-10 s − ήδη ασύμφορο για online σύστημα.

- 24-32 χρήστες: 8-24 s − πρακτικά εκτός ορίων για real-time χρήση.

Όσον αφορά τους μέσους όρους, αποδεικνύεται περίπου 9 φορές ταχύτερος ο PPO κατά μέσο όρο – και > 40× γρηγορότερος στις υψηλές τιμές χρηστών.

Για έως και 15 ταυτόχρονους χρήστες, ο MILP παραμένει βιώσιμος, όμως ο PPO ήδη δίνει περίπου 0.55 s, άρα κερδίζει σε ταχύτητα χωρίς σημαντική απώλεια βέλτιστου κόστους.

Για περισσότερους από 15 χρήστες, μόνο ο PPO μπορεί να υποστηρίξει αποκρίσεις πραγματικού χρόνου (κάτω του δευτερολέπτου).

Καθώς το δίκτυο κλιμακώνεται (π.χ. 60+ κόμβοι, 100+ χρήστες), περιμένουμε ο λόγος ταχυτήτων να αυξηθεί περαιτέρω υπέρ του PPO.

Μέρος ΙΙΙ: ΕΠΙΛΟΓΟΣ

Κεφάλαιο 8: Συμπεράσματα και Μελλοντικές Επεκτάσεις

8.1 Σύνοψη

Η παρούσα διπλωματική εργασία αποσκοπούσε στην εύρεση μιας προσέγγισης βαθιάς ενισχυτικής μάθησης για κατανομή microservices could native εφαρμογών σε ένα περιβάλλον edge-fog-cloud, με στόχο την ελαχιστοποίηση του κόστους, ενώ ταυτόχρονα εξυπηρετούνται οι χρήστες του δικτύου.

Στο πλαίσιο αυτό, εξετάσαμε τη χρήση βαθιάς ενισχυτικής μάθησης και συγκεκριμένα του stable baselines-3 Maskable PPO αλγόριθμου για το πρόβλημα κατανομής μικρό-υπηρεσιών σε υβριδική υποδομή Fog–Cloud υπό δυναμικές συνθήκες φορτίου.

Το περιβάλλον-15 αποτελείτο από 15 κόμβους σε 5 τοποθεσίες, ενώ το περιβάλλον-45 από 45 κόμβους σε 9 τοποθεσίες. Και τα δύο κάνουν deploy 3 microservices, μία αλυσίδα SFC $s_1 \rightarrow s_2 \rightarrow s_3$ που τις συνδέει, και φορτία 10 χρηστών στο στατικό πειραματικό σενάριο και από 1 έως και 32 χρήστες στο δυναμικό.

Εκτελέστηκαν τρία δυναμικά σενάρια στο περιβάλλον-45, το καθένα με διαφορετικό τρόπο διαχείρισης του αριθμού των χρηστών.

8.2 Συμπεράσματα Πειραματικών Δοκιμών

Σε συνέχεια βιβλιογραφικής έρευνας και πρακτικών δοκιμών, διαπιστώθηκε ότι η χρήση του Maskable PPO αλγορίθμου αποδίδει αρκετά καλύτερα από χρήση του Q-learning στο ίδιο περιβάλλον, προσεγγίζοντας τα κόστη της βέλτιστης λύσης MILP σε έναν πάρα πολύ ικανοποιητικό βαθμό, ενώ εξυπηρετεί όλους τους χρήστες.

Στο πιο περίπλοκο (σε σχέση με το στατικό) δυναμικό πειραματικό σενάριο ο πράκτορας μαθαίνει μια πολιτική που πετυχαίνει αρκετά υψηλά ποσοστά κάλυψης των αιτημάτων των χρηστών, με το κόστος του deployment που πραγματοποιεί να είναι σε λογικά πλαίσια. Σε σχέση με τον αλγόριθμο Q-Learning, ο Maskable PPO και σε αυτό το σενάριο κατάφερε αύξηση του ποσοστού των αιτήσεων που αποδέχθηκαν, ενώ διατήρησε το κόστος πλησιέστερα σε αυτό του MILP.

Βάσει των πειραμάτων που έγιναν στο περιβάλλον-45, ο αλγόριθμος αποδείχθηκε αποδοτικός και σε διαφορετικές μεθόδους μεταβολής του φορτίου χρηστών.

Συγκρίνοντας τον χρόνο εκτέλεσης του μοντέλου MILP με το run-time ενός επεισοδίου, έγινε ξεκάθαρο το όφελος χρήσης βαθιάς ενισχυτικής μάθησης με την κλιμάκωση του περιβάλλοντος.

Τα αποτελέσματα αυτά δείχνουν τη χρησιμότητα της βαθιάς ενισχυτικής μάθησης για την επίλυση του προβλήματος, καθώς με την μεγέθυνση του δικτύου, του φόρτου εργασιών, του πλήθους μικρο-υπηρεσιών και των λοιπών μεταβλητών οι λύσεις ILP δεν μπορούν να εφαρμοστούν. Η βαθιά ενισχυτική μάθηση μειώνει το κόστος του υπολογισμού καθώς και το χρόνο, το οποίο την καθιστά μια υποσχόμενη λύση για ανάθεση cloud native εφαρμογών σε δίκτυα fog-cloud σε πραγματικό χρόνο.

Σενάριο	**Cost Gap**	**Cost Gap %**	**Acceptance Rate**	**Χρόνος ILP**	**Σχόλια**
Στατικό	0.25 μον. (ελάχιστο)	2.32 %	100 %	Σταθερός, αποδεκτός	Πολιτική σταθεροποιείται περίπου μετά από 100 k βήματα
Δυναμικό (εκπαίδευση)	1.00 μον. (ελάχιστο)	5.5 %	40 - 100%, συνήθως καλύτερο του 70%	Διαρκώς αυξανόμενος συναρτήσει του αριθμού των χρηστών	Πολιτική σταθεροποιείται περίπου μετά από 350 k βήματα
Δυναμικό (αξιολόγηση)	7.07 μον. κατά μέσο όρο	38.85 % κατά μέσο όρο	90.64 % Κατά μέσο όρο	Διαρκώς αυξανόμενος συναρτήσει του αριθμού των χρηστών	-

Πίνακας 8.1 Σύγκριση Αποτελεσμάτων για το περιβάλλον-15

Σενάριο	**Cost Gap**	**Acceptance Rate**	**Χρόνος ILP**	**Σχόλια**
Στατικό	2.5 μον. (ελάχιστο)	100 %	Σταθερός, αποδεκτός	Πολιτική σταθεροποιείται περίπου μετά από 600k βήματα
Δυναμικό – προεπιλεγμένος τρόπος(εκπαίδευση)	2.5 μον. (ελάχιστο)	60 - 100%, συνήθως καλύτερο του 70%	Διαρκώς αυξανόμενος συναρτήσει του αριθμού των χρηστών	Πολιτική σταθεροποιείται περίπου μετά από 800k βήματα
Δυναμικό - προεπιλεγμένος τρόπος (αξιολόγηση)	5.09 μον. κατά μέσο όρο	77.95% Κατά μέσο όρο	Διαρκώς αυξανόμενος συναρτήσει του αριθμού των χρηστών	-
Δυναμικό – Poisson	0.25 μον. (ελάχιστο)	100%	Διαρκώς αυξανόμενος συναρτήσει του αριθμού των χρηστών	Πολιτική σταθεροποιείται περίπου μετά από 800k βήματα
Δυναμικό – Wave	3.5 μον. (ελάχιστο)	100%	Διαρκώς αυξανόμενος συναρτήσει του αριθμού των χρηστών	Πολιτική σταθεροποιείται περίπου μετά από 550k βήματα

Πίνακας 8.2: Σύγκριση Αποτελεσμάτων για το περιβάλλον-45

8.3 Συμβολή και περιορισμοί

Η συμβολή της παρούσας διπλωματικής εργασίας έγκειται στην εισαγωγή maskable PPO σε περιβάλλον ομίχλης, αποδεικνύοντας την αποτελεσματικότητα του αλγορίθμου με αποτελέσματα υψηλής αποδοτικότητας. Επιπροσθέτως, η εργασία συγκρίνει τα αποτελέσματα που έχει στην εκπαίδευση τόσο η αύξηση των κόμβων του περιβάλλοντος, όσο και η ποικιλία τρόπων μεταβολής των χρηστών του δικτύου.

Όσον αφορά τους περιορισμούς, καταρχάς δεν υπάρχει βελτιστοποίηση άλλων metrics, όπως για παράδειγμα της καθυστέρησης. Επιπροσθέτως, το δυναμικό σενάριο εξετάζει φόρτο δικτύου έως και τριάντα δύο (32) ταυτόχρονων χρηστών.

Τα αποτελέσματα είναι εξαρτώμενα από υποθέσεις για τα βάρη των κόμβων και τις τιμές τόσο των πόρων που διαθέτει κάθε κόμβος, όσο και αυτές των microservices.

8.4 Μελλοντικές Επεκτάσεις

Το σύστημα που αναπτύχθηκε και χρησιμοποιήθηκε χρήζει βελτίωσης και επέκτασης. Στην συγκεκριμένη ενότητα θα συζητηθούν πιθανές μελλοντικές επεκτάσεις που θα μπορούσαν να αξιοποιηθούν για την ανάπτυξη αποτελεσματικών συστημάτων σε πραγματικές συνθήκες.

Αρχικά, μια σημαντική κατεύθυνση αποτελεί η κλιμάκωση. Η αύξηση δηλαδή των κόμβων, τοποθεσιών, μέγιστων χρηστών, εφαρμογών, μικρο-υπηρεσιών, με αφαίρεση της MILP-λύσης ως οδηγό, η οποία πλέον δε θα υπολογίζεται εύκολα. Θα ήταν απαραίτητη η επεξεργασία και ο εμπλουτισμός της συνάρτησης ανταμοιβής καθώς και της μεθόδου αξιολόγησης προκειμένου να εξερευνηθεί η κατεύθυνση αυτή.

Επόμενη σημαντική επέκταση είναι η αξιοποίηση τεχνολογιών όπως Kubernetes και εργαλεία προσομοίωσης όπως CloudSim για την δοκιμή του συστήματος είτε σε πραγματικό είτε σε προσομοιωμένο δίκτυο. Η ανάθεση πραγματικών εφαρμογών σε ένα πραγματικό δίκτυο χρησιμοποιώντας βαθιά ενισχυτική μάθηση θα ήταν ένα σημαντικό βήμα.

Επιπροσθέτως, μια αξιοσημείωτη μελλοντική επέκταση που πρέπει να διερευνηθεί είναι να έχει το σύστημα ως κύριο στόχο την μείωση της συνολικής μέσης καθυστέρησης ή την μεγιστοποίηση των χρηστών που μπορούν να εξυπηρετηθούν ανά πάσα χρονική στιγμή. Παραμένει το ζήτημα της μεταβολής της συνάρτησης ανταμοιβής, η οποία θα πρέπει να προσαρμοστεί στο νέο στόχο του συστήματος.

Θα μπορούσε επίσης να πραγματοποιηθεί διεξαγωγή σύγκρισης του Maskable PPO με αλγορίθμους off-policy όπως SAC και TD3, ώστε να αξιολογηθεί η αποτελεσματικότητα και η ταχύτητα σύγκλισης σε περιβάλλοντα με έντονα δυναμικά φορτία.

Τέλος, μπορεί να συνδυαστεί η ελαχιστοποίηση του κόστους και της καθυστέρησης, με στόχο να μάθει ο πράκτορας μια πολιτική που λαμβάνει υπόψιν και τα δύο αυτά metrics. Η βελτιστοποίηση αυτή πολλών κριτηρίων θα επέτρεπε στον πράκτορα να εφαρμοστεί σε δίκτυα που αποσκοπούν να εξυπηρετήσουν αποτελεσματικά και γρήγορα τον πελάτη ενώ ταυτόχρονα είναι οικονομικά για τους παρόχους και φιλικά προς το περιβάλλον.

Βιβλιογραφία

[1] Z. Mahmood, “Cloud Computing for Enterprise Architectures: Concepts, Principles and Approaches,” 2011, pp. 3–19. doi: 10.1007/978-1-4471-2236-4_1.

[2] R. Giordanelli and C. Mastroianni, “The Cloud Computing Paradigm: Characteristics, Opportunities and Research Issues,” 2010.

[3] P. M. Mell and T. Grance, “The NIST definition of cloud computing,” Gaithersburg, MD, 2011. doi: 10.6028/NIST.SP.800-145.

[4] A. Strømmen-Bakhtiar and A. R. Razavi, “Should the ‘CLOUD’ be Regulated? An Assessment,” 2011.

[5] *Proceedings of 2017 Second IEEE International Conference on Electrical, Computer and Communication Technologies : 22-24, February 2017, SVS College of Engineering, Coimbatore, Tamil Nadu, India*. IEEE, 2017.

[6] B. Kaur, “Software As A Service: A Brief Study,” *International Research Journal of Engineering and Technology*, 2015, [Online]. Available: www.irjet.net

[7] J. Gibson, R. Rondeau, D. Eveleigh, and Q. Tan, “Benefits and challenges of three cloud computing service models,” in *2012 Fourth International Conference on Computational Aspects of Social Networks (CASoN)*, Sao Carlos, Brazil, 2012, pp. 198–205.

[8] “21 Best Software as a Service (SaaS) Examples | SaaS Academy.” Accessed: Jan. 12, 2025. [Online]. Available: https://www.saasacademy.com/blog/saas-examples

[9] H. Yang and M. Tate, “A descriptive literature review and classification of cloud computing research,” *Communications of the Association for Information Systems*, vol. 31, no. 1, pp. 35–60, 2012, doi: 10.17705/1cais.03102.

[10] N. X. Wang, N. B. Wang, and N. J. Huang, “Cloud computing and its key techniques,” in *2011 IEEE International Conference on Computer Science and Automation Engineering 2*, Shanghai, China, Jun. 2011.

[11] G. Lawton, “Developing Software Online with Platform-as-a-Service Technology,” 2000. [Online]. Available: www.rollbase.com/

[12] “What Is Platform as a Service (PaaS)? Definition, Examples, Components, and Best Practices - Spiceworks.” Accessed: Jan. 12, 2025. [Online]. Available: https://www.spiceworks.com/tech/cloud/articles/what-is-platform-as-a-service/#_002

[13] “What is IaaS (Infrastructure as a Service)? | Google Cloud.” Accessed: Jan. 12, 2025. [Online]. Available: https://cloud.google.com/learn/what-is-iaas

[14] “Infrastructure as a Service | Atlassian.” Accessed: Jan. 12, 2025. [Online]. Available: https://www.atlassian.com/microservices/cloud-computing/infrastructure-as-a-service

[15] “Cloud Pyramid: IaaS, PaaS and SaaS | GigaCloud Education.” Accessed: Jan. 12, 2025. [Online]. Available: https://gigacloud.eu/articles/cloud-pyramid-iaas-paas-and-saas/

[16] A. Gajbhiye and K. M. P. Shrivastva, “Cloud computing: Need, enabling technology, architecture, advantages and challenges,” in *2014 5th International Conference - Confluence The Next Generation Information Technology Summit (Confluence)*, Noida, India, 2014, pp. 1–7.

[17] I. Mrhaoaurh, C. Okar, A. Namir, and N. Chafiq, “Challenges of cloud computing use: A systematic literature review,” in *MATEC Web of Conferences*, EDP Sciences, Sep. 2018. doi: 10.1051/matecconf/201820000007.

[18] “Introduction to cloud-native applications - .NET | Microsoft Learn.” Accessed: Jan. 18, 2025. [Online]. Available: https://learn.microsoft.com/en-us/dotnet/architecture/cloud-native/introduction

[19] C. Davis and G. Kim, “M A N N I N G Designing change-tolerant software.”

[20] D. Gannon, R. Barga, W. Services, and N. Sundaresan, “Cloud-Native Applications.”

[21] K. A. Scarfone, M. P. Souppaya, and P. Hoffman, “Guide to security for full virtualization technologies,” Gaithersburg, MD, 2011. doi: 10.6028/NIST.SP.800-125.

[22] J. Sahoo, S. Mohapatra, and R. Lath, “Virtualization: A survey on concepts, taxonomy and associated security issues,” in *2nd International Conference on Computer and Network Technology, ICCNT 2010*, 2010, pp. 222–226. doi: 10.1109/ICCNT.2010.49.

[23] Roger McHaney, *Cloud Technologies: An Overview of Cloud Computing Technologies for Managers*, First Edition. © 2021 John Wiley & Sons, Ltd., 2021.

[24] Richard Uhlig *et al.*, "Intel virtualization technology," *IEEE Computer*, vol. 38, no. 5, pp. 48–56, 2005, doi: 10.1109/MC.2005.

[25] C. Pahl, "Containerization and the PaaS Cloud," *IEEE Cloud Computing*, vol. 2, no. 3, pp. 24–31, 2015, doi: 10.1109/MCC.2015.51.

[26] E. Casalicchio, "Container Orchestration: A Survey," in *EAI/Springer Innovations in Communication and Computing*, Springer Science and Business Media Deutschland GmbH, 2019, pp. 221–235. doi: 10.1007/978-3-319-92378-9_14.

[27] "Google Kubernetes Engine (GKE) | Google Cloud." Accessed: Jun. 24, 2025. [Online]. Available: https://cloud.google.com/kubernetes-engine?hl=en

[28] "Microsoft Azure container services documentation | Microsoft Learn." Accessed: Jun. 24, 2025. [Online]. Available: https://learn.microsoft.com/en-us/azure/containers/

[29] "Fully Managed Container Solution – Amazon Elastic Container Service (Amazon ECS) - Amazon Web Services." Accessed: Jun. 24, 2025. [Online]. Available: https://aws.amazon.com/ecs/

[30] Q. Zhang, L. Liu, C. Pu, Q. Dou, L. Wu, and W. Zhou, "A Comparative Study of Containers and Virtual Machines in Big Data Environment," in *IEEE International Conference on Cloud Computing, CLOUD*, IEEE Computer Society, Sep. 2018, pp. 178–185. doi: 10.1109/CLOUD.2018.00030.

[31] Oyekunle Claudius Oyeniran, Adebunmi Okechukwu Adewusi, Adams Gbolahan Adeleke, Lucy Anthony Akwawa, and Chidimma Francisca Azubuko, "Microservices architecture in cloud-native applications: Design patterns and scalability," *Computer Science & IT Research Journal*, vol. 5, no. 9, pp. 2107–2124, Sep. 2024, doi: 10.51594/csitrj.v5i9.1554.

[32] A. Celesti and P. Leitner, Eds., *Advances in Service-Oriented and Cloud Computing*, vol. 567. in Communications in Computer and Information Science, vol. 567. Cham: Springer International Publishing, 2016. doi: 10.1007/978-3-319-33313-7.

[33] N. Dragoni *et al.*, "Microservices: yesterday, today, and tomorrow," Jun. 2016, [Online]. Available: http://arxiv.org/abs/1606.04036

[34] A. M. Medhat, T. Taleb, A. Elmangoush, G. A. Carella, S. Covaci, and T. Magedanz, “Service Function Chaining in Next Generation Networks: State of the Art and Research Challenges,” *IEEE Communications Magazine*, vol. 55, no. 2, pp. 216–223, Feb. 2017, doi: 10.1109/MCOM.2016.1600219RP.

[35] D. Bhamare, R. Jain, M. Samaka, and A. Erbad, “A survey on service function chaining,” *Journal of Network and Computer Applications*, vol. 75, pp. 138–155, Nov. 2016, doi: 10.1016/j.jnca.2016.09.001.

[36] M. Villamizar *et al.*, “Evaluating the monolithic and the microservice architecture pattern to deploy web applications in the cloud,” in *2015 10th Colombian Computing Conference, 10CCC 2015*, Institute of Electrical and Electronics Engineers Inc., Nov. 2015, pp. 583–590. doi: 10.1109/ColumbianCC.2015.7333476.

[37] S. Taherizadeh, V. Stankovski, and M. Grobelnik, “A capillary computing architecture for dynamic internet of things: Orchestration of microservices from edge devices to fog and cloud providers,” *Sensors (Switzerland)*, vol. 18, no. 9, Sep. 2018, doi: 10.3390/s18092938.

[38] C. H. Hong and B. Varghese, “Resource management in fog/Edge computing: A survey on architectures, infrastructure, and algorithms,” *ACM Comput Surv*, vol. 52, no. 5, Sep. 2019, doi: 10.1145/3326066.

[39] G. Merlino, R. Dautov, S. Distefano, and D. Bruneo, “Enabling Workload Engineering in Edge, Fog, and Cloud Computing through OpenStack-based Middleware,” *ACM Trans Internet Technol*, vol. 19, no. 2, Apr. 2019, doi: 10.1145/3309705.

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

[41] A. Raj, “A Review on Machine Learning Algorithms,” *Int J Res Appl Sci Eng Technol*, vol. 7, no. 6, pp. 792–796, Jun. 2019, doi: 10.22214/ijraset.2019.6138.

[42] J. Zhang, N. Xie, X. Zhang, K. Yue, W. Li, and D. Kumar, “Machine learning based resource allocation of cloud computing in auction,” *Computers, Materials and Continua*, vol. 56, no. 1, pp. 123–135, 2018, doi: 10.3970/cmc.2018.03728.

[43] S. Mehta, A. Singh, and K. K. Singh, “Role of machine learning in resource allocation of fog computing,” in *Proceedings of the Confluence 2021: 11th International Conference on Cloud Computing, Data Science and Engineering*,

Institute of Electrical and Electronics Engineers Inc., Jan. 2021, pp. 262–266. doi: 10.1109/Confluence51648.2021.9377095.

[44] Hamdy A. Taha, *Operations Research An Introduction*, 10th ed. Pearson Education Limited, 2017.

[45] Laurence A. Wolsey, *Mixed Integer Programming*. In Wiley Encyclopedia of Computer Science and Engineering, 2008. doi: https://doi.org/10.1002/9780470050118.ecse244.

[46] A. Misevičius, T. Blažauskas, J. Blonskis, and J. Smolinskas, "AN OVERVIEW OF SOME HEURISTIC ALGORITHMS FOR COMBINATORIAL OPTIMIZATION PROBLEMS," 2004.

[47] J. Santos, T. Wauters, B. Volckaert, and F. De Turck, "Reinforcement Learning for Service Function Chain Allocation in Fog Computing," 2021, ch. 7, pp. 147–173. doi: 10.1002/9781119675525.ch7.

[48] L. Pack Kaelbling, M. L. Littman, A. W. Moore, and S. Hall, "Reinforcement Learning: A Survey," 1996.

[49] M. Van Otterlo and M. Wiering, "ALO 12 - Reinforcement Learning and Markov Decision Processes," in *Wiering, M., van Otterlo, M. (eds) Reinforcement Learning. Adaptation, Learning, and Optimization*, vol. 12, 2012. doi: https://doi.org/10.1007/978-3-642-27645-3_1.

[50] William J. Stewart, *Probability, Markov Chains, Queues, and Simulation: The Mathematical Basis of Performance Modelling*, 1st ed. Princeton University Press, 2009.

[51] R. S. Sutton and A. G. Barto, "Reinforcement Learning: An Introduction Second edition, in progress."

[52] V. François-Lavet, P. Henderson, R. Islam, M. G. Bellemare, and J. Pineau, "An introduction to deep reinforcement learning," *Foundations and Trends in Machine Learning*, vol. 11, no. 3–4, pp. 219–354, Dec. 2018, doi: 10.1561/2200000071.

[53] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath, "Deep reinforcement learning: A brief survey," Nov. 01, 2017, *Institute of Electrical and Electronics Engineers Inc.* doi: 10.1109/MSP.2017.2743240.

[54] V. François-Lavet, P. Henderson, R. Islam, M. G. Bellemare, and J. Pineau, “An introduction to deep reinforcement learning,” *Foundations and Trends in Machine Learning*, vol. 11, no. 3–4, pp. 219–354, Dec. 2018, doi: 10.1561/2200000071.

[55] F. Tan, P. Yan, and X. Guan, “Deep Reinforcement Learning: From Q-Learning to Deep Q-Learning,” in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, Springer Verlag, 2017, pp. 475–483. doi: 10.1007/978-3-319-70093-9_50.

[56] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, “Proximal Policy Optimization Algorithms,” Jul. 2017, [Online]. Available: http://arxiv.org/abs/1707.06347

[57] Ji Li, Hui Gao, Tiejun Lv, and Yueming Lu, “Deep Reinforcement Learning based Computation Offloading and Resource Allocation for MEC,” IEEE, 2018.

[58] W. Wang, S. Chen, P. Zhang, and K. Liu, “Reinforcement-Learning-Assisted Service Function Chain Embedding Algorithm in Edge Computing Networks,” *Electronics (Switzerland)*, vol. 13, no. 15, Aug. 2024, doi: 10.3390/electronics13153007.

[59] Z. Wang, M. Goudarzi, M. Gong, and R. Buyya, “Deep Reinforcement Learning-based Scheduling for Optimizing System Load and Response Time in Edge and Fog Computing Environments,” Sep. 2023, [Online]. Available: http://arxiv.org/abs/2309.07407

[60] C. Zhang, C. Wu, M. Lin, Y. Lin, and W. Liu, “Proximal Policy Optimization for Efficient D2D-Assisted Computation Offloading and Resource Allocation in Multi-Access Edge Computing,” *Future Internet*, vol. 16, no. 1, Jan. 2024, doi: 10.3390/fi16010019.

[61] M. Zare, Y. Elmi Sola, and H. Hasanpour, “Towards distributed and autonomous IoT service placement in fog computing using asynchronous advantage actor-critic algorithm,” *Journal of King Saud University - Computer and Information Sciences*, vol. 35, no. 1, pp. 368–381, Jan. 2023, doi: 10.1016/j.jksuci.2022.12.006.

[62] J. Santos, T. Wauters, B. Volckaert, and F. De Turck, “Towards end-to-end resource provisioning in Fog Computing over Low Power Wide Area Networks,” *Journal of Network and Computer Applications*, vol. 175, Feb. 2021, doi: 10.1016/j.jnca.2020.102915.

[63] *2018 14th International Conference on Network and Service Management : 5-9 November 2018, Rome, Italy*. Institute of Electrical and Electronics Engineers, 2018.

[64] C. Y. Tang, C. H. Liu, W. K. Chen, and S. D. You, “Implementing action mask in proximal policy optimization (PPO) algorithm,” *ICT Express*, vol. 6, no. 3, pp. 200–203, Sep. 2020, doi: 10.1016/j.icte.2020.05.003.

[65] A. Raffin, A. Hill, A. Gleave, A. Kanervisto, M. Ernestus, and N. Dormann, “Stable-Baselines3: Reliable Reinforcement Learning Implementations,” 2021. [Online]. Available: https://github.com/DLR-RM/stable-baselines3.

[66] “Maskable PPO — Stable Baselines3 - Contrib 2.7.0a0 documentation.” Accessed: Jun. 17, 2025. [Online]. Available: https://sb3-contrib.readthedocs.io/en/master/modules/ppo_mask.html

[67] “GitHub - openai/gym: A toolkit for developing and comparing reinforcement learning algorithms.” Accessed: Jun. 17, 2025. [Online]. Available: https://github.com/openai/gym

[68] G. Brockman *et al.*, “OpenAI Gym,” Jun. 2016, Accessed: Jun. 17, 2025. [Online]. Available: https://arxiv.org/pdf/1606.01540

[69] “NumPy documentation — NumPy v2.3 Manual.” Accessed: Jun. 18, 2025. [Online]. Available: https://numpy.org/doc/stable/

[70] “Stable Baselines3 Documentation Release 2.7.0a0 Stable Baselines3 Contributors,” 2025.