Εθνικο Μετσοβιο Πολυτεχνειο

Σχολη Ηλεκτρολογων Μηχανικων και Μηχανικων Υπολογιστων

Τομεας Επικοινωνιων, Ηλεκτρονικης και Συστηματων Πληροφορικης

Μεταφόρτωση δεδομένων σε τεχνολογίες Κινητής Υπολογιστικής στα Άκρα του Δικτύου βασισμένη στη Θεωρία Παιγνίων και στην Ισορροπία Ικανοποίησης

Διπλωματικη Εργασια

της

ΣΥΛΒΙΑΣ-ΔΗΜΗΤΡΑΣ ΜΑΝΚΑ

Επιβλέπων: Συμεών Παπαβασιλείου
Καθηγητής Ε.Μ.Π.

Εργαστηριο Διαχειρισης Βελτιστου Σχεδιασμου Δικτυων Τηλεματικης

Αθήνα, Ιούλιος 2019

Εθνικό Μετσόβιο Πολυτεχνείο
Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών
Τομέας Επικοινωνιών, Ηλεκτρονικής και Συστημάτων Πληροφορικής
Εργαστήριο Διαχείρισης Βέλτιστου Σχεδιασμού Δικτύων Τηλεματικής

Μεταφόρτωση δεδομένων σε τεχνολογίες Κινητής Υπολογιστικής στα Άκρα του Δικτύου βασισμένη στη Θεωρία Παιγνίων και στην Ισορροπία Ικανοποίησης

Διπλωματικη Εργασια

της

ΣΥΛΒΙΑΣ-ΔΗΜΗΤΡΑΣ ΜΑΝΚΑ

Επιβλέπων: Συμεών Παπαβασιλείου
Καθηγητής Ε.Μ.Π.

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 5η Ιουλίου 2019.

(Υπογραφή)	*(Υπογραφή)*	*(Υπογραφή)*
.....	.....	.....
Συμεών Παπαβασιλείου	Θεοδώρα Βαρβαρίγου	Ιωάννα Ρουσσάκη
Καθηγητής Ε.Μ.Π.	Καθηγήτρια Ε.Μ.Π.	Επίκ. Καθηγήτρια Ε.Μ.Π.

Αθήνα, Ιούλιος 2019

(Υπογραφή)

.....

Συλβια Δημητρα Μανκα

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

Εθνικό Μετσόβιο Πολυτεχνείο
Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών
Τομέας Επικοινωνιών, Ηλεκτρονικής και Συστημάτων Πληροφορικής
Εργαστήριο Διαχείρισης Βέλτιστου Σχεδιασμού Δικτύων Τηλεματικής

Ευχαριστίες

Την ολοκλήρωση της παρούσας διπλωματικής την οφείλω σε μεγάλο βαθμό στον επιβλέποντα καθηγητή μου, κ. Συμεών Παπαβασιλείου, την καθηγήτρια κα. Ειρήνη-Ελένη Τσιροπούλου καθώς και τον διδακτορικό φοιτητή Παύλο Αθανασόπουλο. Τους ευχαριστώ από καρδιάς για την εμπιστοσύνη που μου έδειξαν αναθέτοντάς μου ένα τόσο ενδιαφέρον και σύγχρονο θέμα, για το αμέριστο ενδιαφέρον τους και τον χρόνο που μου αφιέρωσαν. Η εκπόνηση της εργασίας χάρη στην υποδειγματική μας συνεργασία και συνεχή καθοδήγησή τους, αποτέλεσε μία δημιουργική εμπειρία, από την οποία προσκόμισα πολύτιμες γνώσεις και εφόδια.

Ένα μεγάλο ευχαριστώ οφείλω επίσης στα παιδιά του εργαστηρίου του κ. Παπαβασιλείου, για την συμβολή τους στην επιλογή διπλωματικής μου και που με ενέταξαν σε ένα τόσο ευχάριστο περιβάλλον, δίνοντάς μου πολύ συχνά πολύτιμες συμβουλές και τα απαραίτητα ευχάριστα διαλείμματα από την διαδικασία της διπλωματικής.

Πολύ σημαντικό ρόλο για την πορεία μου όλα αυτά τα χρόνια και την ολοκλήρωση των φοιτητικών μου σπουδών κατέχει η οικογένειά μου που δεν σταμάτησε ποτέ να με στηρίζει υπομονετικά και την ευχαριστώ θερμά γι΄ αυτό. Τέλος, ευχαριστώ όλους τους φίλους μου με τους οποίους μοιράστηκα τα φοιτητικά μου χρόνια, κάνοντάς τα μία ξεχωριστή και ανεπανάληπτη περίοδο της ζωής μου.

Περίληψη

Σε ένα περιβάλλον ολοένα αυξανόμενων υπολογιστικών αναγκών και παραγωγής όγκου πληροφοριών, η τεχνολογία νέφους αποτελεί πλέον ισχυρό σύμμαχο ώστε να πληρούνται οι απαιτήσεις του χρήστη σε επίπεδο υπηρεσιών. Μία επέκταση της τεχνολογίας νέφους για τα συστήματα κινητών συσκευών, που ανταποκρίνεται και στις απαιτήσεις για εξαιρετικά γρήγορη αποκρισιμότητα, είναι η Κινητή Υπολογιστική στα Άκρα του Δικτύου (ΚΥΑΔ). Στα περιβάλλοντα ΚΥΑΔ, όπου κινητές συσκευές προσπαθούν να μεταφορτώσουν απαιτητικές εργασίες σε εξυπηρετητές στα άκρα του δικτύου, κρίνεται επιτακτική η ανάγκη αποδοτικού συντονισμού ώστε να γίνεται η καλύτερη δυνατή αξιοποίηση των πόρων και κατανομής του φορτίου. Προς αυτή την κατεύθυνση, η θεωρία παιγνίων σε συνδυασμό με την ενισχυτική μάθηση προσφέρουν ισχυρές κατανεμημένες λύσεις με χαμηλό υπολογιστικό κόστος.

Στην παρούσα διπλωματική, μελετήθηκε ένα περιβάλλον ΚΥΑΔ με πολλαπλούς εξυπηρετητές άκρης και πολλαπλούς χρήστες κινητών συσκευών, οι οποίοι εκτός από την επιλογή του εξυπηρετητή, μπορούσαν επιπλέον να προσδιορίσουν το ποσοστό της διεργασίας που θα μεταφορτώσουν ώστε να ικανοποιήσουν τους περιορισμούς τους σε χρόνο απόκρισης και ενεργειακή κατανάλωση. Για την επιλογή του εξυπηρετητή από τους χρήστες υιοθετήθηκε μία λύση βασισμένη στην θεωρία παιγνίων κατά την οποία οι χρήστες συμπεριφέρονταν σαν αυτόματα μάθησης, προκειμένου να μεγιστοποιηθεί η συνολική ευχαρίστηση του συστήματος. Σε κάθε εποχή του αλγορίθμου μάθησης, για να προσδιοριστεί το ποσοστό που θα μεταφορτώσουν οι χρήστες στον εξυπηρετητή που έχουν επιλέξει και να καθοριστεί η τιμή στην συνάρτηση ευχαρίστησης τους, αντιμετωπίζουμε το πρόβλημα πάλι σαν ένα παίγνιο και αναζητούμε την εύρεση ενός σημείου ισορροπίας ικανοποίησης, ή γενικευμένης ισορροπίας ικανοποίησης, αν δεν είναι εφικτό όλοι οι παίκτες να ικανοποιηθούν ταυτόχρονα.

Η αξιολόγηση της ορθότητας και αποτελεσματικότητας του προτεινόμενου πλαισίου πραγματοποιήθηκε μέσω μοντελοποίησης και προσομοίωσης. Συγκεκριμένα, εκτενή πειράματα εκτελέστηκαν ώστε να μελετηθούν τα θεμελιώδη χαρακτηριστικά λειτουργίας του προτεινόμενου ενιαίου αλγοριθμικού πλαισίου, η δυνατότητα κλιμακωσιμότητας του, καθώς και η σύγκριση του με εναλλακτικές στρατηγικές επίλυσης.

Λέξεις Κλειδιά

Κινητή Υπολογιστική στα Άκρα του Δικτύου, Υπολογιστική Μεταφόρτωση, Αυτόματα Μάθησης, Ενισχυτική Μάθηση, Θεωρία Παιγνίων, Ισορροπία Ικανοποίησης

Abstract

In an environment of ever-increasing computational needs and data generation, cloud computing arises as a powerful paradigm in meeting demanding Quality of Service (QoS) requirements. Mobile Edge Computing (MEC) has emerged as an extension of cloud computing, enabling the realization of computation-intensive and latency-critical applications at the resource-limited mobile devices. Due to the partial offloading of computation-intensive tasks from mobile devices to the MEC servers, there is an urgent need for efficient orchestration, in order to optimize resource utilization and load allocation. To this end, game theory combined with reinforcement learning offers powerful distributed solutions with low computational cost.

Subject of this thesis is a MEC environment containing multiple edge servers and mobile users, in which users try not only to select a MEC server to get associated with but also to determine a percentage of their data to offload in order to meet their latency and energy constraints. A game theoretic approach was adopted, in which users acting as learning automata aim to maximize their overall utility or satisfaction. In order to determine the percentage that users would offload to the server in each epoch of the learning algorithm, we modeled the problem as a satisfaction game and searched for a satisfaction equilibrium point. For cases where it was not possible for all users to simultaneously satisfy their constraints, a generalized satisfaction equilibrium was explored instead.

The evaluation of the correctness and effectiveness of the proposed framework was achieved through modeling and simulation. In particular, extensive experiments were performed, aiming to study the fundamental operating characteristics of the proposed framework and its scalability property, while being compared with alternative data offloading strategies.

Keywords

Mobile Edge Computing, Computational Offloading, Learning Automata, Reinforcement Learning, Game Theory, Satisfaction Equilibrium

Περιεχόμενα

Κατάλογος Σχημάτων

Κεφάλαιο 1

Εισαγωγή

1.1 Εισαγωγικές Έννοιες και Ορισμοί

1.1.1 Πρόλογος

Στην σύγχρονη κοινωνία, που χαρακτηρίζεται από ραγδαία εξέλιξη της τεχνολογίας, οι έξυπνες κινητές συσκευές αποκτούν όλο και περισσότερη δημοτικότητα. Όσο οι τεχνολογίες κινητών αναπτύσσονται, εφαρμογές απαιτητικές σε υπολογιστική ισχύ και ενεργειακή κατανάλωση, όπως η επεξεργασία εικόνας και φυσικής γλώσσας, αναγνώριση προσώπων, διαδραστικά παιχνίδια κλπ, καθιστούν απαραίτητη την ενίσχυση της υπολογιστικής ισχύος των περιορισμένων σε πόρους κινητών συσκευών[10].

Ένα θεμελιώδες εμπόδιο στην μελλοντική ανάπτυξη των κινητών συσκευών αποτελούν οι τεχνολογίες μπαταριών[12]. Οι επεξεργαστές γίνονται γρηγορότεροι, οι οθόνες πιο ακριβείς, οι συσκευές διαθέτουν όλο και περισσότερους αισθητήρες. Κατά συνέπεια, οι απαιτήσεις των κινητών σε ενεργειακή κατανάλωση ολοένα αυξάνονται και οι μπαταρίες των κινητών αδυνατούν να ανταποκριθούν. Δυστυχώς, οι τεχνολογίες που συνδέονται με τις μπαταρίες των κινητών συσκευών δείχνουν ότι τα εμπόδια αυτά δεν θα ξεπεραστούν εύκολα[38].

Όπως είναι αναμενόμενο, η αδυναμία των κινητών συσκευών να ανταποκριθούν στις σύγχρονες επιταγές έχει προσελκύσει μεγάλο ερευνητικό ενδιαφέρον. Μία ιδιαίτερα δημοφιλής τεχνική προς αυτή την κατεύθυνση είναι να επιτρέψουμε στις κινητές συσκευές, όταν αυτό είναι εφικτό, να μεταφορτώνουν ένα μέρος των πιο ενεργοβόρων και υπολογιστικά απαιτητικών εφαρμογών τους σε πλούσιες σε πόρους εξυπηρετητές και να λαμβάνουν τα αποτελέσματα. Η στρατηγική αυτή έχει μελετηθεί εδώ και πολλά χρόνια και στην βιβλιογραφία συχνά αναφέρεται ως υπολογιστική μεταφόρτωση (computational offloading)[25] ή απομακρυσμένη εκτέλεση(remote execution)[12]. Οι τεχνολογίες που έχουν αναπτυχθεί για την εφαρμογή της απομακρυσμένης εκτέλεσης παρουσιάζονται αναλυτικά παρακάτω.

1.1.2 Τεχνολογίες Υπολογιστικού Νέφους

Το Υπολογιστικό Νέφος (Cloud Computing), είναι μία έννοια που την ακούμε όλο και πιο συχνά τα τελευταία χρόνια. Καθημερινά, αποθηκεύουμε πληθώρα δεδομένων μας σε απομακρυ-

σμένους εξυπηρετητές με την βοήθεια διάφορων εφαρμογών υπολογιστικού νέφους. Τυπικά, το Υπολογιστικό Νέφος[28] αποτελεί ένα μοντέλο το οποίο επιτρέπει την πανταχού παρούσα, κατ΄ απαίτηση πρόσβαση σε διαμοιραζόμενες δεξαμενές ρυθμιζόμενων πόρων συστήματος τα οποία μπορούν να παραχωρηθούν γρήγορα, με ελάχιστη υπολογιστική προσπάθεια και αλληλεπίδραση με τον πάροχο.

Το Υπολογιστικό Νέφος [18] εφαρμόζει την υπολογιστική μεταφόρτωση και επιτρέπει στους χρήστες να χρησιμοποιούν πόρους κατ΄ απαίτηση. Οι πόροι αυτοί μπορούν να είναι 1) υποδομές, όπως συσκευές δικτύου, εξυπηρετητές, αποθηκευτικοί χώροι, 2) πλατφόρμες, όπως λειτουργικά συστήματα που περιλαμβάνουν ολοκληρωμένα περιβάλλοντα για ανάπτυξη και έλεγχο εφαρμογών και 3)λογισμικό στην μορφή προγραμμάτων.

Ένα πολύ σημαντικό χαρακτηριστικό του υπολογιστικού νέφους είναι οι υπηρεσίες εικονοποίησης (virtualization) που επιτρέπουν την ταυτόχρονη λειτουργία πολλαπλών λειτουργικών συστημάτων και εφαρμογών, σε ένα μόνο μηχάνημα.
Αν και το Υπολογιστικό Νέφος προσφέρει απεριόριστες υπολογιστικές δυνατότητες στους χρήστες, εισάγεται σημαντική καθυστέρηση, αφού η επικοινωνία των χρηστών με τους εξυπηρετητές απαιτεί την αποστολή δεδομένων μέσω του παγκόσμιου δικτύου.

Μία επέκταση του υπολογιστικού νέφους είναι το Κινητό Υπολογιστικό Νέφος(ΚΥΝ) - (Mobile Cloud Computing). Το Κινητό Υπολογιστικό Νέφος [14] ενσωματώνει το Υπολογιστικό Νέφος στο περιβάλλον των κινητών συσκευών, που περιλαμβάνει γενικότερα κινητούς συνδρομητές και όχι απλά χρήστες έξυπνων κινητών συσκευών. Το Κινητό Υπολογιστικό Νέφος αναφέρεται σε μία υποδομή στην οποία η επεξεργασία και η αποθήκευση των δεδομένων γίνεται εκτός της κινητής συσκευής. Υπηρεσίες Κινητού Νέφους μεταφέρουν τον υπολογισμό από τα κινητά στο νέφος, προκειμένου οι χρήστες κινητών συσκευών να μπορούν να εκμεταλλευτούν πλήρως τα οφέλη του Υπολογιστικού Νέφους.

Τα κύρια μειονεκτήματα του κινητού υπολογιστικού νέφους είναι τα εξής:

- Οι υποδομές που χρησιμοποιούνται στις τεχνολογίες υπολογιστικού νέφους συνήθως απαντώνται σε γεωγραφικά απομακρυσμένα κέντρα δεδομένων νέφους(cloud datacenter)[5]. Όπως είναι αναμενόμενο, η επικοινωνία των χρηστών με τους πάροχους υπηρεσιών νέφους πάσχει από μεγάλες καθυστερήσεις που οφείλονται στη μεγάλη απόσταση που πρέπει να διανύσουν τα δεδομένα μέσα από τον παγκόσμιο δίκτυο και την συμφόρηση του δικτύου που δεν μπορεί να ελεγχθεί. Συνεπώς επιδεινώνεται σημαντικά η ποιότητα εμπειρίας του χρήστη, σε εφαρμογές πραγματικού χρόνου που έχουν υψηλή απαίτηση σε γρήγορη αποκρισιμότητα.

- Η τεχνολογία του υπολογιστικού νέφους είναι συνυφασμένη με την έννοια της κεντροποίησης[18], αφού όλα τα δεδομένα και οι εφαρμογές των χρηστών καταλήγουν στα τεράστια κέντρα δεδομένων νέφους. Αυτού του είδους η κεντροποίηση εγείρει προβληματισμούς σε θέματα που αφορούν την ασφάλεια των δεδομένων των χρηστών.

Για να αντιμετωπιστούν τα παραπάνω, τα τελευταία χρόνια έχει προταθεί η αρχιτεκτονική Υπολογισμού στα Άκρα του Δικτύου(ΥΑΔ) (Edge Computing)[27]. Η έννοια αυτή έχει προσελκύσει μεγάλο επιστημονικό ενδιαφέρον, καθώς υπόσχεται να λύσει τα βασικά προβλήματα από τα οποία πάσχει η Κινητή Υπολογιστική Νέφους πριν την Υπολογιστική στα Άκρα του Δικτύου. Η θεμελιώδης ιδέα στην οποία βασίζεται η ΥΑΔ είναι η μεταφορά της επεξεργαστικής δυνατότητας εγγύτερα στους τελικούς χρήστες. Πιο συγκεκριμένα, η ΥΑΔ αναφέρεται σε όλες τις τεχνολογίες που καθιστούν δυνατή την επεξεργασία στην άκρη του δικτύου. Η έννοια άκρη του δικτύου [41] συνίσταται από τις τελικές συσκευές (π.χ. έξυπνα κινητά τηλέφωνα), τις συσκευές άκρης (οι συσκευές του δικτύου που βρίσκονται στο σύνορο όπως δρομολογητές, μεταγωγείς, σημεία ασύρματης πρόσβασης (wireless access points), σταθμοί βάσης (base stations), τους εξυπηρετητές άκρης καθώς και την διαδρομή από την πηγή δεδομένων προς τα κέντρα δεδομένων νέφους. Όλα αυτά τα συστατικά στοιχεία πρέπει να εφοδιαστούν με τα απαραίτητα εργαλεία προκειμένου να υποστηρίζονται οι λειτουργίες του υπολογισμού στα άκρα του δικτύου.

Συνδυάζοντας τις έννοιες του υπολογιστικού νέφους και του υπολογισμού στα άκρα του δικτύου διάφορες αρχιτεκτονικές έχουν προταθεί. Μεταξύ αυτών, η Κινητή Υπολογιστική στα Άκρα του Δικτύου(ΚΥΑΔ) (Mobile Edge Computing) [1][37] είναι η πιο δημοφιλής πιθανή και υλοποιήσιμη επέκταση. Η ΚΥΑΔ χαίρει μεγάλης δημοτικότητας τα τελευταία χρόνια, καθώς επιτρέπει σε εξυπηρετητές άκρης και κυψελωτούς σταθμούς βάσης (cellular base station) να δουλεύουν συνεργατικά. Η κινητή υπολογιστική στα άκρα του δικτύου μπορεί να συνδυαστεί και να συνδεθεί με απομακρυσμένα κέντρα δεδομένων υπολογιστικού νέφους. Με τον τρόπο αυτό, οι υποδομές και τα πλεονεκτήματα του κορμού του δικτύου συνεχίζουν να χρησιμοποιούνται σαν βοηθητικά μέσα επεξεργασίας όταν είναι απαραίτητο και η φύση των εφαρμογών το επιτρέπει.

Η εξέλιξη των τεχνολογιών ΚΥΑΔ είναι άρρηκτα συνδεδεμένη με την ταυτόχρονη εξέλιξη των δικτύων 5ης γενιάς, στον τομέα των δικτύων επικοινωνιών. Τα δίκτυα 5ης γενιάς, βρίσκονται υπό συνεχή έρευνα προκειμένου να μπορούν να ανταποκριθούν στις απαιτήσεις της τεράστιας αύξησης της κίνησης των δεδομένων [36]. Θεμελιώδεις στόχοι των δικτύων 5ης γενιάς είναι: να αυξήσουν τον χρόνο με τον οποίο μπορούν να αποστείλουν οι κινητές συσκευές δεδομένα, να μειωθεί ο χρόνος απόκρισης παραπάνω από το 1/10 των τιμών που καταφέρνουν τα δίκτυα 4ης γενιάς, και ταυτόχρονα να μειωθεί η κατανάλωση ενέργειας που απαιτείται ώστε οι χρήστες να αποστείλουν ένα bit πληροφορίας. Όλα τα παραπάνω είναι απαραίτητα ώστε να μπορεί να εφαρμοστεί αποτελεσματικά η ΚΥΑΔ.

1.1.3 Το πρόβλημα της Μεταφόρτωσης Υπολογισμών

Η Μεταφόρτωση Υπολογισμών - Computational Offloading, είναι η διαδικασία κατά την οποία, ολόκληρες διεργασίες ή μέρος αυτών, εκτελούνται απομακρυσμένα από την κινητή συσκευή σε υποδομές νέφους[16].

Οι τεχνικές μεταφόρτωσης υπολογισμών μπορούν να κατηγοριοποιηθούν βάσει πολλών διαφορετικών κριτηρίων[5]. Άμα τις διαχωρίσουμε βάσει πρωτοκόλλου που χρησιμοποιείται ώστε κινητές συσκευές να ανταλλάξουν δεδομένα με τον εξυπηρετητή έχουμε τις εξής κατηγορίες [30]:

- Επικοινωνία πελάτη-εξυπηρετητή. Το πρωτόκολλο αυτό προϋποθέτει η εφαρμογή που θέλει να εκτελέσει ο χρήστης, να είναι ήδη εγκατεστημένη στον εξυπηρετητή.
- Εικονοποίηση. Για τις τεχνικές αυτές, απαιτείται ο εξυπηρετητής να υποστηρίζει εικονοποίηση και να μπορεί να ορίζει εικονικά μηχανήματα, στα οποία θα τρέχουν τις εφαρμογές τους οι κινητές συσκευές. Η μέθοδος αυτή εξασφαλίζει απομόνωση και ασφάλεια.
- Πρωτόκολλο Κινητού Πράκτορα. Κινητός πράκτορας είναι ένα πρόγραμμα που μεταφέρεται μέσα στο δίκτυο, μεταφέροντας τον πηγαίο κώδικα και τα απαραίτητα δεδομένα προς επεξεργασία. Οι τεχνικές που βασίζονται στο πρωτόκολλο αυτό προϋποθέτουν από τις κινητές συσκευές να διασπάσουν τις εφαρμογές τους σε ενότητες (modules).

Όπως είδαμε, στη μεταφόρτωση υπολογισμών, συχνά μεταφορτώνεται ένας μέρος της εφαρμογής. Το πρόγραμμα διασπάται σε ανεξάρτητες ενότητες. Κάποιες ενότητες είναι απαραίτητο να εκτελεστούν τοπικά, και για όλες τις υπόλοιπες πρέπει να αποφασιστεί αν θα μεταφορτωθεί, ή όχι. Επιπλέον, ο διαχωρισμός σε ενότητες απαιτεί και διαχωρισμό του κώδικα του προγράμματος.

Όλα τα παραπάνω θέματα είναι αρκετά περίπλοκα και αποτελούν ξεχωριστά ερευνητικά πεδία που συνδέονται με το το πρόβλημα της μεταφόρτωσης. Στο πλαίσιο της παρούσας διπλωματικής, θεωρούμε δεδομένο τον τρόπο με τον οποίο ο χρήστης μεταφορτώνει την εφαρμογή και πώς την διασπά σε επιμέρους ενότητες.

1.2 Συνοπτική Βιβλιογραφική Ανασκόπηση

Στην ενότητα αυτή παρουσιάζονται ενδεικτικά ορισμένες εργασίες των τελευταίων χρόνων, οι οποίες καταπιάνονται με ερευνητικά θέματα στο πεδίο της κινητής υπολογιστικής στα άκρα του δικτύου.

Στις τεχνολογίες ΚΥΑΔ, πολύ σημαντικό ερευνητικό θέμα αποτελούν οι τεχνικές διαχείρισης και κατανομής πόρων στις υποδομές νέφους. Στο [4] μελετάται το πρόβλημα της κατανομής των πόρων στους εξυπηρετητές άκρης και αν αυτοί θα ενεργοποιηθούν, χρησιμοποιώντας θεωρία ελέγχου. Το [7] ασχολείται με την χρονοδρομολόγηση εργασιών σε έναν εξυπηρετητή ΚΥΑΔ, εστιάζοντας στο πώς η αποθήκευσή τους στην κρυφή μνήμη του εξυπηρετητή μπορεί να οδηγήσει σε βελτίωση της επίδοσης. Στο [26] παρουσιάζεται ένα ιεραρχικό πλαίσιο ελέγχου που στοχεύει στον αποδοτικό συντονισμό ανταγωνιστικών στόχων μέσα σε ένα κέντρο δεδομένων νέφους. Στο [47] χρησιμοποιούν ένα πολυδιάστατο μηχανισμό κοστολόγησης, βασισμένο σε οικονομικούς μηχανισμούς για να εξασφαλίσουν την αποδοτική αξιοποίηση των υποδομών ενός cloudlet καθώς και την αποδοτική χρονοδρομολόγηση της μεταφόρτωσης εφαρμογών.

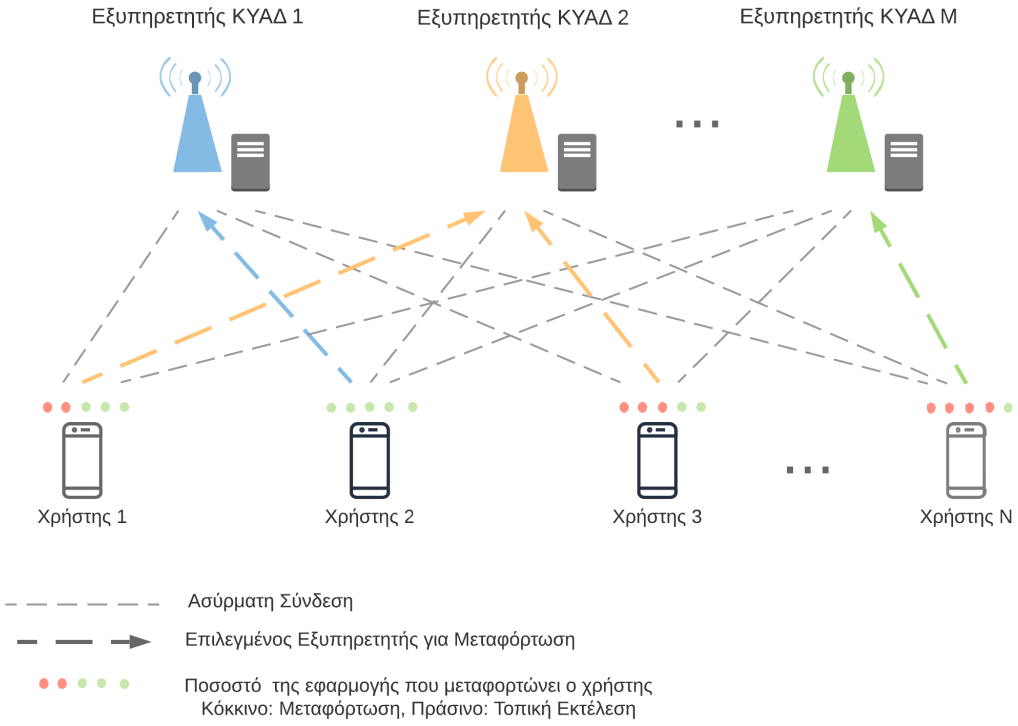
Από την πλευρά του χρήστη, το θέμα της επιλογής καναλιού, σημείου πρόσβασης, σταθμού βάσης ή εξυπηρετητή άκρης για να μεταφορτώσουν ολόκληρες διαδικασίες μελετήθηκαν στα [10][21][51]. Υιοθετήθηκε μία προσέγγιση βασισμένη στην θεωρία παιγνίων και χρησιμοποιώντας αλγορίθμους μάθησης το σύστημα κατέληγε σε ένα αποδοτικό σημείο ισορροπίας.

Στο [39] χρησιμοποιείται παιχνίδι μειονότητας σε συνδυασμό με κάποιο αλγόριθμο μάθησης, αυτή την φορά για να προσδιοριστεί πόσοι χρήστες θα μεταφορτώσουν δεδομένα σε ένα σταθμό βάσης, δεδομένου ότι έχει περιορισμένη χωρητικότητα και δεν μπορεί να τους εξυπηρετήσει όλους. Το [8] προτείνεται ένας αλγόριθμος βασισμένος σε βαθιά νευρωνικά δίκτυα, για να καθοριστεί η βέλτιστη επιλογή σταθμού βάσης για να την υπολογιστική μεταφόρτωση ενός χρήστη. Στο [20] προτείνεται ένας υβριστικός αλγόριθμος για την μεταφόρτωση εργασιών στις υποδομές νέφους ώστε ο παραλληλισμός μεταξύ των χρηστών και του νέφους να μεγιστοποιείται. Στο [3] μελετάται ένα περιβάλλον ΚΥΑΔ στο οποίο οι εξυπηρετητές συμμετέχουν σε ένα παίγνιο μειονότητας για να αποφασίσουν ποιοι από αυτούς θα ενεργοποιηθούν, στην συνέχεια οι χρήστες συμπεριφέρονται σαν αυτόματα μάθησης για να προσδιοριστεί η κατανομή των παικτών στους ενεργούς εξυπηρετητές, ενώ ταυτόχρονα οι χρήστες, συμμετέχουν σε ένα παιχνίδι ικανοποίησης, προσπαθώντας να προσδιορίσουν την ενέργεια μετάδοσης για την μεταφόρτωση των διεργασιών τους στους εξυπηρετητές ώστε να ικανοποιήσουν τους περιορισμούς τους σε ποιότητας υπηρεσίας. Στο [49] οι χρήστες μπορούν να μεταφορτώσουν ολόκληρη την διεργασία τους σε έναν εξυπηρετητή επιλέγοντας μεταξύ δύο σταθμών βάσης, και διάφορων καναλιών ανά σταθμό βάσης, χρησιμοποιούν έναν αλγόριθμο βελτιστοποίησης, με σκοπό να ελαχιστοποιηθεί η κατανάλωση ενέργειας. Στο [33] προτείνεται ένα πλαίσιο(framework) υπολογιστικής μεταφόρτωσης, που υπολογίζει και προβλέπει τον χρόνο εκτέλεσης που συνδέεται με μία απόφαση μεταφόρτωσης, και το συγκρίνει με άλλα υπάρχοντα πλαίσια υπολογιστικής μεταφόρτωσης [12],[11],[22],[46],[24],[40]. Άλλα ερευνητικά θέματα γύρω από τα συστήματα ΚΥΑΔ περιλαμβάνουν την κατανομή ασύρματων ραδιοπόρων μεταξύ των παραδοσιακών υπηρεσιών επικοινωνίας και των υπηρεσιών ΚΥΑΔ σε ένα κοινό δίκτυο ραδιοπρόσβασης(RAN). Με το θέμα αυτό ασχολούνται στα ([9][50][52]).

1.3 Αντικείμενο της διπλωματικής

Στην παρούσα διπλωματική θεωρούμε ένα περιβάλλον ΚΥΑΔ το οποίο περιλαμβάνει πολλαπλούς χρήστες και πολλαπλούς εξυπηρετητές άκρης, οι οποίοι προσφέρουν υπηρεσίες νέφους στους χρήστες. Στο περιβάλλον αυτό, οι χρήστες επιδιώκουν να μεταφορτώσουν ένα μέρος των πιο ενεργοβόρων και υπολογιστικά απαιτητικών τους εφαρμογών, προκειμένου να ικανοποιήσουν περιορισμούς τους στον χρόνο εκτέλεσης και στην κατανάλωση ενέργειας. Στο Σχήμα 1.1 βλέπουμε N χρήστες κινητών συσκευών που μπορούν όλοι να συνδεθούν και με τους M εξυπηρετητές αλλά επιλέγουν έναν μόνο για να μεταφορτώσουν ένα ποσοστό της εφαρμογής που θέλουν να εκτελέσουν.

Θεωρούμε ότι χρησιμοποιώντας μία δεδομένη τεχνική μεταφόρτωσης, οι χρήστες μπορούν να επιλέξουν ποιο ποσοστό της εφαρμογής, και των αντίστοιχων δεδομένων, να μεταφορ-



Σχήμα 1.1: Τοπολογία του συστήματος

τώσουν στον εξυπηρετητή ΚΥΑΔ.

Το πρόβλημα μας, καταρχάς, συνίσταται στην εύρεση κατανομής χρηστών στους εξυπηρετητές. Με άλλα λόγια, από την πλευρά των χρηστών, ο καθένας πρέπει να αποφασίσει σε ποιόν εξυπηρετητή θα μεταφορτώσει τα δεδομένα του. Σε δεύτερο στάδιο, με δεδομένη μία κατανομή, ο κάθε χρήστης πρέπει να αποφασίσει το ποσοστό εκείνο της εφαρμογής του που θα μεταφορτώσει στον εξυπηρετητή που έχει ήδη επιλέξει.

Πρόκειται λοιπόν για ένα περίπλοκο διεπίπεδο πρόβλημα, στο οποίο οι επιλογές των χρηστών αλληλοεπηρεάζονται. Οι χρήστες πρέπει να αποφασίσουν διαθέτοντας πληροφορίες που αφορούν μόνο την δική τους κατάσταση. Αναζητούμε μία κατανεμημένη λύση που να εξασφαλίζει όσο το δυνατόν περισσότερους ικανοποιημένους χρήστες, στην περίπτωση που δεν μπορούν ταυτόχρονα όλοι να ικανοποιήσουν τους περιορισμούς τους.

Για την επίλυση του προβλήματος υιοθετήθηκε μία προσέγγιση που χρησιμοποιεί στοιχεία από την θεωρία παιγνίων και αλγορίθμους από το πεδίο της Ενισχυτικής Μάθησης - Reinforcement Learning. Πιο συγκεκριμένα, στην παρούσα διπλωματική εργασία κάναμε τα εξής:

- Χωρίσαμε το πρόβλημα σε δύο επίπεδα και μοντελοποιήσαμε και τα δύο επίπεδα ως παίγνια.
- Στο πρώτο επίπεδο, που αφορά την επιλογή εξυπηρετητή, ο κάθε χρήστης εκτελεί έναν

αλγόριθμο μάθησης μέχρι το σύστημα να φτάσει σε μία κατάσταση ισορροπίας, σε μία κατάσταση δηλαδή, που κανένας χρήστης δεν θέλει να αλλάξει την επιλογή του εξυπηρετητή.

- Στο δεύτερο επίπεδο χρησιμοποιούμε έναν απλό αλγόριθμο για την απόφαση του ποσοστού που θα μεταφορτώσουν οι χρήστες στους εξυπηρετητές που έχουν ήδη επιλέξει. Σε κάθε ομάδα που αποτελείται από τον εξυπηρετητή και τους χρήστες που τον έχουν επιλέξει, αποδεικνύουμε ότι ο αλγόριθμός μας υπολογίζει ένα αποδοτικό σημείο ισορροπίας ικανοποίησης.

- Τέλος, τα χαρακτηριστικά του αλγοριθμικού πλαισίου που χρησιμοποιήθηκε μελετήθηκαν διεξοδικά μέσα από μία σειρά προσομοιώσεων που πραγματοποιήσαμε. Μελετήθηκε επίσης η κλιμακωσιμότητα του αλγορίθμου μας, καθώς και η επίδοση του αλγορίθμου μας σε σύγκριση με εναλλακτικές στρατηγικές επίλυσης.

1.4 Οργάνωση της Διπλωματικής

Στο κεφάλαιο 2 παρουσιάζεται αναλυτικά το πρόβλημα που πραγματεύεται η διπλωματική και θέματα που αφορούν την μοντελοποίηση του συστήματος. Στο κεφάλαιο 3 παρουσιάζεται συνοπτικά το θεωρητικό υπόβαθρο από την Θεωρία Παιγνίων και την Μηχανική Μάθηση που χρησιμοποιήθηκε. Στην συνέχεια, στο κεφάλαιο 4 περιγράφουμε αναλυτικά όλα τα συστατικά στοιχεία του αλγοριθμικού πλαισίου που χρησιμοποιήθηκε. Τα αποτελέσματα των προσομοιώσεων και πειραμάτων που εκτελέσαμε τα παρουσιάζουμε στο κεφάλαιο 5 και τέλος, στο κεφάλαιο 6 παρουσιάζουμε μία σύνοψη της διπλωματικής εργασίας και αναφέρουμε πιθανές μελλοντικές επεκτάσεις της.

Κεφάλαιο 2

Περιγραφή και Μοντελοποίηση του Προβλήματος

2.1 Εισαγωγή

Στο πλαίσιο της παρούσας διπλωματικής μελετήθηκε ένα σύστημα Κινητής Υπολογιστικής στα Άκρα του δικτύου που περιλαμβάνει:

- Εξυπηρετητές Κινητής Υπολογιστικής στα Άκρα του Δικτύου $\mathbb{M} = \{1, \cdots, j, \cdots, M\}$
- Χρήστες Κινητών Συσκευών $\mathbb{U} = \{1, \cdots, i, \cdots, U\}$

Ο κάθε χρήστης έχει σαν στόχο να εκτελέσει μία διαδικασία T_i εντός ενός χρονικού περιθωρίου και έχει και ένα όριο στην ενέργεια που μπορεί να δαπανήσει. Προκειμένου ο χρήστης να μπορέσει να ικανοποιήσει τους περιορισμούς του, έχει την δυνατότητα να μεταφορτώσει σε κάποιο εξυπηρετητή ΚΥΑΔ ένα ποσοστό a_i των δεδομένων του και τα υπόλοιπα δεδομένα να τα εκτελέσει τοπικά.

Όπως αναφέραμε και στο πρώτο κεφάλαιο, το πρόβλημα που μελετάμε είναι διεπίπεδο. Το πρώτο επίπεδο περιλαμβάνει την κατανομή των χρηστών στους εξυπηρετητές και το δεύτερο επίπεδο αφορά τον καθορισμό του ποσού δεδομένων που θα μεταφορτώσει ο χρήστης στον εξυπηρετητή που έχει επιλέξει στο επίπεδο 1.

Μία δεδομένη κατανομή δημιουργεί συμπλέγματα (clusters) που αποτελούνται από έναν εξυπηρετητή ΚΥΑΔ j και όλους τους χρήστες i που τον έχουν επιλέξει, $i \in \mathbb{U}_j$. Οι χρήστες για να ικανοποιήσουν τους περιορισμούς τους πρέπει να μεταφορτώσουν ένα ποσό των δεδομένων τους στον εξυπηρετητή, ο οποίος έχει συγκεκριμένη χωρητικότητα σε δεδομένα που μπορεί να δεχτεί. Στο επίπεδο αυτό, υπολογίζουμε για κάθε σύμπλεγμα ξεχωριστά, το ποσοστό από τα δεδομένα τους που θα μεταφορτώσει ο κάθε χρήστης, με σκοπό να εξασφαλίσουμε τον μεγαλύτερο αριθμό χρηστών που ικανοποιούν τους περιορισμούς τους μέσα στο σύμπλεγμα.

Πρώτα θα ορίσουμε τις μεταβλητές που αφορούν τις διαδικασίες που πρέπει να εκτελέσουν οι χρήστες, στην συνέχεια θα ορίσουμε το επικοινωνιακό μοντέλο, δηλαδή την ενέργεια και τον χρόνο που δαπανάται προκειμένου να στείλει ο χρήστης τα δεδομένα στον εξυπηρετητή, το

υπολογιστικό μοντέλο, που περιλαμβάνει την ενέργεια και τον χρόνο που δαπανάται από τον χρήστη και από τον εξυπηρετητή για να εκτελέσουν τις διαδικασίες T_i και τέλος υπολογίζουμε τον συνολικό χρόνο και την συνολική ενέργεια του χρήστη που καταναλώνεται.

2.2 Χρήστες Κινητών Συσκευών

Ο κάθε χρήστης i έχει σαν στόχο να εκτελέσει μία διαδικασία T_i. Τα χαρακτηριστικά της διαδικασίας T_i είναι τα εξής:

$$T_i = (I_i, C_i, t_i, e_i)$$

όπου,

- I_i : ο συνολικός όγκος δεδομένων σε bytes που πρέπει να επεξεργαστεί ο χρήστης i
- C_i: οι CPU κύκλοι που απαιτούνται ώστε να εκτελεστεί η διαδικασία T_i
- t_i: είναι το χρονικό όριο του χρήστη για την εκτέλεση της διαδικασίας σε δευτερόλεπτα
- e_i: είναι η μέγιστη ενέργεια που ο χρήστης έχει την δυνατότητα να δαπανήσει για την διαδικασία αυτή

Χρησιμοποιούμε επίσης την βοηθητική μεταβλητή:
λ_i : ορίζεται ως το πηλίκο των απαιτούμενων κύκλων C_i με το συνολικό όγκο δεδομένων I_i. Εκφράζει το πόσο υπολογιστικά βαριά είναι η εν λόγω διαδικασία.

Ο χρήστης επιλέγει να μεταφορτώσει ένα ποσοστό $a_i \in [0, 1]$ από τα δεδομένα του στον εξυπηρετητή ΚΥΑΔ. Ορίζουμε $b_i = a_i \cdot I_i$ το συνολικό αριθμό δεδομένων σε bytes που μεταφορτώνει ο i στον εξυπηρετητή.

2.3 Επικοινωνιακό Μοντέλο

Ο χρήστης i θα επιλέξει ένα ποσό b_i για να μεταφορτώσει στον εξυπηρετητή j.

Ο ρυθμός μετάδοσης με τον οποίο στέλνονται τα δεδομένα από τον χρήστη στον εξυπηρετητή:

$$R_{i,j} = W \cdot \log_2(1 + \frac{p_{i,j} \cdot g_{i,j}}{\sigma_0^2 + \sum_{k \in U_j, a_k \neq 0, k \neq i} p_{k,j} \cdot g_{k,j}})$$

όπου $p_{i,j}, g_{i,j}$ είναι η ισχύς μετάδοσης και το κέρδος καναλιού μεταξύ του χρήστη $i \in \mathbb{U}$ και του εξυπηρετητή j.

Χρόνος μετάδοσης:

$$t_{i,j}^{(tr)} = \frac{b_i}{Ri,j}\text{sec}$$

Ενέργεια που καταναλώνεται στην μετάδοση:

$$e_{i,j}^{(tr)} = \frac{b_i \cdot p_{i,j}}{R_{i,j}} \text{ Joules}$$

2.4 Υπολογιστικό Μοντέλο

Ο χρήστης i θα μεταφορτώσει στον εξυπηρετητή j ένα ποσό b_i και θα εκτελέσει τοπικά τα υπόλοιπα bytes $I_i - b_i$. Ο εξυπηρετητής θα επεξεργαστεί τα b_i δεδομένα με υπολογιστική ικανότητα:

$$f_{i,j}(a_i, a_{-i}) = \frac{\lambda_i}{\sum_{k \in U_j, a_k \neq 0} \lambda_k} \cdot (1 - \frac{\overline{\text{b}_\text{j}}}{B_j}) \cdot F_j$$

όπου,
a_{-i}: είναι οι στρατηγικές των άλλων παικτών που ανήκουν στον εξυπηρετητή j
$\overline{\text{b}_\text{j}}$: είναι το άθροισμα των bytes που έχουν μεταφορτώσει στον εξυπηρετητή j
B_j: είναι το όριο του εξυπηρετητή ΚΥΑΔ ως προς τα συνολικά δεδομένα που μπορεί να αποδεχτεί.
F_j: είναι η υπολογιστική ικανότητα του εξυπηρετητή ΚΥΑΔ σε $\frac{Cycles}{sec}$

Χρόνος εκτέλεσης - Εξυπηρετητής ΚΥΑΔ:

$$t_{i,j}^{(ex,m)} = \frac{\lambda_i \cdot b_i}{f_{i,j}} = b_i \cdot \frac{B_j \cdot \sum \lambda_k}{(B_j - \overline{b_j})F_j} \text{ sec}$$

Χρόνος εκτέλεσης - Τοπικά:

$$t_{i,j}^{(ex,l)} = \frac{\lambda_i \cdot (I_i - b_i)}{lc_i} \text{ sec}$$

Ενέργεια που δαπανάται για την εκτέλεση τοπικά:

$$e_{i,j}^{(ex,l)} = \lambda_i \cdot (I_i - b_i) \cdot le_i \text{ Joules}$$

όπου,
lc_i: είναι η υπολογιστική ικανότητα του χρήστη - συσκευή κινητού i
le_i: είναι η ενεργειακή κατανάλωση του χρήστη σε $\frac{Joules}{Cycle}$

2.5 Συνολικός Χρόνος και Ενέργεια

Σύμφωνα με όλα τα παραπάνω υπολογίζουμε τον συνολικό χρόνο και ενέργεια για έναν χρήστη που έχει επιλέξει έναν εξυπηρετητή j και να μεταφορτώσει ένα ποσοστό a_i από τα δεδομένα του, δεδομένου ότι οι υπόλοιποι χρήστες $k \in \mathbb{U}_j$ έχουν επιλέξει στρατηγικές a_{-i}.

2.5.1 Συνολικός Χρόνος

Η εκτέλεση της διαδικασίας περιλαμβάνει τον χρόνο για την μετάδοση των δεδομένων και τον χρόνο που απαιτείται από τον χρήστη και τον επεξεργαστή ώστε να εκτελέσουν το ποσοστό της διεργασίας που τους έχει ανατεθεί. Ο επεξεργαστής, αφού λάβει τα δεδομένα, επεξεργάζεται παράλληλα με τον χρήστη τα δεδομένα, γι΄ αυτό προκύπτει ο συνολικός χρόνος ως εξής:

$$O_{i,j}^{t}(a_i, a_{-i}) = \max\{t_{i,j}^{tr} + t_{i,j}^{ex,m}, t_{i,j}^{ex,l}\} = \max\{b_i \cdot (\frac{1}{R_{i,j}} + \frac{B_j \cdot \sum \lambda_k}{(B_j - \overline{b_j})F_j}), \frac{\lambda_i \cdot (I_i - b_i)}{lc_i}\}$$

όπου έχουμε θεωρήσει αμελητέο τον χρόνο που δαπανάται από τον εξυπηρετητή για να στείλει τα δεδομένα πίσω στον χρήστη.

2.5.2 Συνολική Ενέργεια για τον χρήστη

Οι περιορισμοί του προβλήματος αφορούν μόνο την κατανάλωση ενέργειας από την πλευρά του χρήστη, γι΄ αυτό δεν λαμβάνουμε υπόψιν την ενέργεια που καταναλώνει ο εξυπηρετητής. Η συνολική ενέργεια είναι η εξής:

$$O_{i,j}^{e}(a_i, a_{-i}) = e_{i,j}^{tr} + e_{i,j}^{ex,l} = b_i \cdot (\frac{p_{i,j}}{R_{i,j}} - \lambda_i \cdot le_i) + \lambda_i \cdot I_i \cdot le_i$$

2.5.3 Περιορισμοί των χρηστών

Έχοντας ορίσει όλα τα παραπάνω, είναι εύκολο να εκφράσουμε μαθηματικά τις απαιτήσεις των χρηστών σε χρόνο και ενέργεια. Πιο συγκεκριμένα, κάθε χρήστης $i \in \mathbb{U}_j$ στοχεύει να βρει μία στρατηγική a_i τέτοια ώστε να ισχύει:

$$O^t_{i,j}(a_i, \mathbf{a}_{-i}) \leq t_i \text{ και } O^e_{i,j}(a_i, \mathbf{a}_{-i}) \leq e_i$$

Οι χρήστες που καταφέρνουν να βρουν τέτοια στρατηγική που να ικανοποιεί ταυτόχρονα και τους δύο περιορισμούς, θα λέμε ότι είναι ικανοποιημένοι. Στην συνέχεια της διπλωματικής, συνδυάζοντας όλα τα παραπάνω με στοιχεία από την θεωρία παιγνίων και αλγορίθμους της ενισχυτικής μάθησης, προσπαθούμε να μεγιστοποιήσουμε τον αριθμό των ικανοποιημένων χρηστών στο σύστημα μας.

Κεφάλαιο 3

Θεωρητικό υπόβαθρο

3.1 Εισαγωγή

Η προσέγγιση της λύσης μας για το πρόβλημα που έχει περιγραφεί περιλαμβάνει στοιχεία από την θεωρία παιγνίων και αλγορίθμους που προέρχονται από το ευρύ πεδίο της μηχανικής μάθησης. Στις επόμενες υποενότητες περιγράφονται τα στοιχεία της θεωρίας που χρησιμοποιήθηκαν στην παρούσα διπλωματική.

3.2 Θεωρία Παιγνίων

Η θεωρία παιγνίων[35] μελετά τις αλληλεπιδράσεις μεταξύ οντοτήτων, των οποίων οι αποφάσεις για τις πράξεις τους επηρεάζουν το αποτέλεσμα, το οποίο αποτελεί το επίκεντρο του ενδιαφέροντος όλων των οντοτήτων[23]. Οι βασικές υποθέσεις που διέπουν την θεωρία παιγνίων είναι ότι οι οντότητες είναι απαραίτητο να έχουν σαφώς ορισμένους στόχους (να είναι λογικοί) και για να αποφασίσουν να λαμβάνουν υπόψιν τους τις γνώσεις τους ή τις προσδοκίες τους για τη συμπεριφορά των υπόλοιπων οντοτήτων (να φέρονται στρατηγικά). Συνεπώς, η θεωρία παιγνίων μελετά τις *στρατηγικές* αλληλεπιδράσεις μέσα σε μία ομάδα οντοτήτων.

Καταστάσεις οι οποίες αφορούν τέτοιου είδους αλληλεπιδράσεις μπορούν να μοντελοποιηθούν και να μελετηθούν σαν ένα παίγνιο. Ένα παίγνιο, ανάλογα με τα χαρακτηριστικά του μπορούμε να το χωρίσουμε στις εξής κατηγορίες:

- Συνεργατικά ή μη-Συνεργατικά Παίγνια, ανάλογα με το αν οι παίκτες του παιγνίου επικοινωνούν και αποφασίζουν συνεργατικά μία λύση, ή όχι.
- Τέλειας ή ατελούς πληροφορίας, ανάλογα με το αν όλοι οι παίκτες γνωρίζουν τις επιλογές που έχουν πραγματοποιήσει οι υπόλοιποι παίκτες του παιγνίου, και αποφασίζουν βάσει αυτών.

3.2.1 Παίγνια σε Κανονική Μορφή Αναπαράστασης

Ορισμός 3.1 (Παίγνια σε Κανονική Μορφή Αναπαράστασης)**.** *Ένα παίγνιο σε κανονική μορφή αναπαράστασης αποτελείται από ένα σύνολο $\mathcal{K} = \{1, \ldots, K\}$ παικτών, έναν πεπερασμένο*

στρατηγικό χώρο $\mathcal{A}_k$ που αποτελείται από τις διαθέσιμες επιλογές του κάθε παίκτη, και μία συνάρτηση ωφέλειας $u_k : \mathcal{A}_k \to \mathbb{R}$ που εκφράζει πόσο ικανοποιημένος είναι ο παίκτης από την έκβαση του παιγνίου [44].

$$G = \{\mathcal{K}, \{\mathcal{A}_k\}_{k\in\mathcal{K}}, \{u_k\}_{k\in\mathcal{K}}\}$$

Ορίζουμε στρατηγικό προφίλ του παιγνίου (action profile) το διάνυσμα που περιλαμβάνει τις K στρατηγικές που έχουν επιλέξει οι παίκτες:

$$\mathbf{a} = (a_1, \ldots, a_K) \in \mathcal{A}$$

όπου $\mathcal{A} = \mathcal{A}_1 \times \cdots \times \mathcal{A}_K$

Ορίζουμε επίσης το σύνολο $\mathbf{a}_{-k} = (a_1, \ldots, a_{k-1}, a_{k+1}, \ldots, a_K)$ που προκύπτει από το $\mathbf{a}$ αφαιρώντας το $k-$οστό στοιχείο. Αντίστοιχα έχουμε και το σύνολο $\mathcal{A}_{-k} = \mathcal{A}_1 \times \cdots \times \mathcal{A}_{k-1} \times \mathcal{A}_{k+1} \times \cdots \times \mathcal{A}_K$ που αντιστοιχεί στο δυναμοσύνολο όλων των πιθανών στρατηγικών προφίλ $\mathbf{a}_{-k}$.

3.2.2 Ισορροπία Νας

Μία από τις βασικότερες έννοιες στην θεωρία των παιγνίων είναι η έννοια της Ισορροπίας. Τα σημεία ισορροπίας είναι καταστάσεις στις οποίες κανείς παίκτης δεν έχει λόγο να αλλάξει στην στρατηγική του, συνεπώς τα σημεία ισορροπίας είναι σταθερές καταστάσεις που αν το παίγνιο φτάσει σε αυτές, κανένας παίκτης δεν αλλάζει την στρατηγική του. Την έννοια της ισορροπίας την οφείλουμε στον Νας και ορίζεται ως εξής:

Ορισμός 3.2 (Ισορροπία Νας[32])**.** *Η ισορροπία Νας, σε ένα παίγνιο σε κανονική μορφή, είναι ένα στρατηγικό προφίλ $\mathbf{a}^* = (a_k^*, \mathbf{a}_{-k}^*) \in \mathcal{A}$, τέτοιο ώστε:*

$$u_k(a_k^*, \mathbf{a}_{-k}^*) \geq u_k(a_k', \mathbf{a}_{-k}^*) \quad \forall a_k' \in \mathcal{A}_k, \forall k \in \mathcal{K}$$

Η φυσική ερμηνεία του παραπάνω ορισμού είναι ότι, με δεδομένο στρατηγικό προφίλ από τους άλλους παίκτες, κανείς παίκτης δεν μπορεί να εξασφαλίσει μεγαλύτερη τιμή ωφέλειας αλλάζοντας μονομερώς την στρατηγική του [43].

3.2.3 Γενικευμένη Ισορροπία Νας

Η θεωρία παιγνίων εδώ και χρόνια αποτελεί ένα πολύ ισχυρό εργαλείο για να μοντελοποιήσει προβλήματα από πολλούς διαφορετικούς κλάδους. Συνεπώς, πολλές διαφορετικές επεκτάσεις έχουν δοθεί στον παραπάνω ορισμό. Για τα προβλήματα στα οποία οι παίκτες έχουν συγκεκριμένες απαιτήσεις για τις τιμές της συνάρτησης ωφέλειας, ο παραπάνω ορισμός ισορροπίας δεν αρκεί για να καλύψει τις απαιτήσεις του προβλήματος. Κατ΄ εξοχήν τέτοια

προβλήματα είναι αυτά που συνδέονται με την ποιότητα Υπηρεσίας και ποιότητα Εμπειρίας των παικτών(QoS, QoE satisfaction). Στην κατεύθυνση αυτή, μελετήθηκε η Γενικευμένη Ισορροπία Νας, η οποία αποτελεί μία πολύ χρήσιμη επέκταση της ισορροπίας Νας για τα παραπάνω προβλήματα.

Για τον ορισμό της γενικευμένης ισορροπίας Νας απαιτείται να επεκτείνουμε τον ορισμό του παιγνίου προκειμένου να συμπεριλάβουμε και τους περιορισμούς που θέλουν να ικανοποιήσουν οι παίκτες. Ορίζουμε το παρακάτω παίγνιο σε μορφή ικανοποίησης (Satisfaction Form) [15].

Ορισμός 3.3. *Ένα παίγνιο σε μορφή ικανοποίησης ορίζεται ως εξής:*

$$G_s = \{\mathcal{K}, \{\mathcal{A}_k\}_{k\in\mathcal{K}}, \{u_k\}_{k\in\mathcal{K}}, \{f_k\}_{k\in\mathcal{K}}\}$$

όπου προσθέσαμε επιπλέον την συνάρτηση f_k. Η συνάρτηση $f_k : \mathcal{A}_{-k} \to 2^{\mathcal{A}_k}$ ονομάζεται ανταπόκριση (correspondence) του παίκτη k και καθορίζει όλα τα δυνατά στρατηγικά προφίλ που ικανοποιούν τους περιορισμούς του παίκτη. Δεδομένου δηλαδή $\mathbf{a}_{-k}$, ο παίκτης k είναι ικανοποιημένος για κάθε $a_k \in f_k(\mathbf{a}_{-k})$.

Ορισμός 3.4 (Γενικευμένη Ισορροπία Νας[13])**.** *Η γενικευμένη ισορροπία Νας, είναι ένα στρατηγικό προφίλ $\mathbf{a}^* = (a_k^*, \mathbf{a}_{-k}^*) \in \mathcal{A}$, τέτοιο ώστε για κάθε k ισχύει:*

$$a_k^* \in f_k(\mathbf{a}_{-k}^*) \quad \textit{και}$$

$$u_k(a_k^*, \mathbf{a}_{-k}^*) \geq u_k(a_k', \mathbf{a}_{-k}^*) \quad \forall a_k' \in f_k(\mathbf{a}_{-k}^*)$$

Στην γενικευμένη ισορροπία Νας οι παίκτες πάλι προσπαθούν να μεγιστοποιήσουν την συνάρτηση ωφέλειας τους, ενώ ταυτόχρονα εξασφαλίζεται ότι ικανοποιούνται οι περιορισμοί τους, κάτι στο οποίο η ισορροπία Νας αποτυγχάνει. Παρολαυτά, από πρακτική άποψη, η γενικευμένη ισορροπία Νας είναι πολύ περιοριστική λύση καθώς περιορίζει το σύνολο των διαθέσιμων στρατηγικών των παικτών.

3.2.4 Ισορροπία Ικανοποίησης και επεκτάσεις

Η Ισορροπία Ικανοποίησης (Satisfaction Equilibrium) είναι μία εφικτή λύση σε προβλήματα που συνδέονται με ικανοποίηση περιορισμών ποιότητας υπηρεσίας, καθώς είναι πολύ λιγότερο περιοριστική από την γενικευμένη ισορροπία Νας.

Ορισμός 3.5 (Ισορροπία Ικανοποίησης[13])**.** *Η ισορροπία ικανοποίησης, είναι ένα στρατηγικό προφίλ $\mathbf{a}^+ = (a_k^+, \mathbf{a}_{-k}^+) \in \mathcal{A}$, τέτοιο ώστε:*

$$\forall k \in \mathcal{K} \quad a_k^+ \in f_k(\mathbf{a}_{-k}^+)$$

Στην ισορροπία αυτή, δεν λαμβάνεται καθόλου υπόψιν η συνάρτηση ωφέλειας u_k, παρά μόνο για να καθοριστούν οι στρατηγικές που ανήκουν στην συνάρτηση f_k. Στα προβλήματα που αναζητούμε ισορροπία ικανοποίησης, απαραίτητη προϋπόθεση είναι οι παίκτες να θέλουν

να ικανοποιήσουν τους περιορισμούς τους, χωρίς να τους ενδιαφέρει η ελαχιστοποίηση ή μεγιστοποίηση κάποιας μετρικής.

Μία επέκταση της Ισορροπίας Ικανοποίησης είναι η Αποδοτική Ισορροπία Ικανοποίησης (Efficient Satisfaction Equilibrium), η οποία επιπλέον εξασφαλίζει ότι οι παίκτες διαλέγουν τη στρατηγική που τους ελαχιστοποιεί μία συνάρτηση κόστους. Ορίζουμε την συνάρτηση κόστους $c_k : \mathcal{A} \to [0, 1]$ ώστε να μοντελοποιήσουμε την έννοια του κόστους ή της προσπάθειας που συνδέεται με μία συγκεκριμένη στρατηγική a_k και ισχύει ότι αν:

$$\forall (a_k, a'_k) \in \mathcal{A}_k^2 : c_k(a_k) < c_k(a'_k)$$

τότε η στρατηγική a_k απαιτεί λιγότερη προσπάθεια - κόστος από την στρατηγική a'_k. Βάση του παραπάνω ορισμού ορίζουμε την Αποδοτική Ισορροπία Ικανοποίησης.

Ορισμός 3.6 (Αποδοτική Ισορροπία Ικανοποίησης[29])**.** *Η αποδοτική ισορροπία ικανοποίησης σε ένα παίγνιο σε μορφή ικανοποίησης με συναρτήσεις κόστους $\{c_k\}_{k\in\mathcal{K}}$, είναι ένα στρατηγικό προφίλ $\mathbf{a}^* = (a_k^*, \mathbf{a}^*_{-k}) \in \mathcal{A}$, τέτοιο ώστε:*

$$\forall k \in \mathcal{K}, \quad a_k^* \in f_k(\mathbf{a}^*_{-k}) \quad \textit{και}$$

$$\forall k \in \mathcal{K}, \quad \forall a_k \in f_k(\mathbf{a}^*_{-k}), \quad c_k(a_k) \geq c_k(a_k^*)$$

Να υπογραμμιστεί ότι συνάρτηση κόστους, σε αντίθεση με την συνάρτηση ωφέλειας που χρησιμοποιείται για την Γενικευμένη Ισορροπία Νας, δεν εξαρτάται από τις στρατηγικές που έχουν επιλέξει οι άλλοι παίκτες. Τις συναρτήσεις κόστους τις καθορίζει ο κάθε παίκτης μεμονωμένα.

Η εύρεση ενός σημείου Ισορροπίας Ικανοποίησης προϋποθέτει την ύπαρξη του. Σε ορισμένα προβλήματα δεν είναι δυνατή η ταυτόχρονη ικανοποίηση των απαιτήσεων όλων των παικτών, και σε τέτοιες περιπτώσεις, λύσεις που ψάχνουν ισορροπία ικανοποίησης αποτυγχάνουν. Για τον λόγο αυτό η έννοια της Ισορροπίας Ικανοποίησης γενικεύτηκε έτσι ώστε να μπορεί να συμπεριλάβει και τις περιπτώσεις αυτές.

Κάθε στρατηγικό προφίλ $\mathbf{a}$ του παιγνίου κατανέμει το σύνολο των παικτών σε δύο υποσύνολα, $\mathcal{K}_s$ και $\mathcal{K}_u$ που απαρτίζονται από τους ικανοποιημένους παίκτες και τους μην ικανοποιημένους παίκτες αντίστοιχα.

Για τους ικανοποιημένους παίκτες ισχύει: $\forall k \in \mathcal{K}_s$, $a_k \in f_k(\mathbf{a}_{-k})$, ενώ για τους μη ικανοποιημένους: $\forall k \in \mathcal{K}_u$, $a_k \notin f_k(\mathbf{a}_{-k})$. Οι ικανοποιημένοι παίκτες δεν έχουν λόγο να αλλάξουν την στρατηγική τους. Αντίθετα, για τους μη ικανοποιημένους, προκειμένου να έχουμε ισορροπία πρέπει να ισχύει ότι καμία διαθέσιμη στρατηγική τους δεν μπορεί να τους ικανοποιήσει με το δεδομένο στρατηγικό προφίλ. Αυτή η έννοια της ισορροπίας, που ονομάζεται Γενικευμένη Ισορροπία Ικανοποίησης (Generalized Satisfaction Equilibrium) ορίζεται ως εξής:

Ορισμός 3.7 (Γενικευμένη Ισορροπία Ικανοποίησης[19])**.** *Ένα στρατηγικό προφίλ $\mathbf{a}$ αποτελεί Γενικευμένη Ισορροπία Ικανοποίησης αν υπάρχει διαμέριση του συνόλου $\mathcal{K}$ σε $\mathcal{K}_s$ και $\mathcal{K}_u$ τέτοια ώστε $\forall k \in \mathcal{K}_s$ ισχύει ότι $a_k \in f_k(\mathbf{a}_{-k})$, και $\forall k \in \mathcal{K}_u$ ισχύει ότι $f_k(\mathbf{a}_{-k}) = \emptyset$*

Η ύπαρξη σημείου Γενικευμένης Ισορροπίας Ικανοποίησης εξαρτάται από τις ιδιότητες των συναρτήσεων $f_1, f_2 \ldots, f_K$.

3.2.5 Συναρτήσεις ευχαρίστησης

Απαραίτητες για την ορθή μοντελοποίηση προβλημάτων με θεωρία παιγνίων είναι οι συναρτήσεις ευχαρίστησης (payoff functions). Οι συναρτήσεις ευχαρίστησης είναι μαθηματικές συναρτήσεις οι οποίες ποσοτικοποιήσουν την ευχαρίστηση των παικτών από μία δεδομένη έκβαση του παιγνίου. Τυπικά οι συναρτήσεις ευχαρίστησης ορίζονται: $F : \mathcal{A} \to \mathbb{R}$

3.3 Αλγόριθμοι Ενισχυτικής Μάθησης για την εύρεση Σημείων Ισορροπίας

3.3.1 Συστήματα με Περιορισμένες Πληροφορίες

Όπως αναφέραμε στην προηγούμενη ενότητα, όταν εφαρμόζουμε την θεωρία παιγνίων για να λύσουμε κάποιο πρόβλημα, οι καταστάσεις ισορροπίας είναι οι επιθυμητές και σταθερές καταστάσεις που θέλουμε να καταλήξουμε κατά τις οποίες κανείς παίκτης δεν θέλει να αλλάξει την στρατηγική του.

Σε πολλά συστήματα οι παίκτες αντιμετωπίζουν περιορισμούς που αφορούν τις πληροφορίες που έχουν για το σύστημα[48]. Πιο συγκεκριμένα οι πληροφορίες που λαμβάνουν ενδέχεται να είναι:

- Ανεπαρκείς. Οι παίκτες ξέρουν προσωπικές τους μετρικές ενώ δεν γνωρίζουν τίποτα για την κατάσταση των άλλων παικτών και τις στρατηγικές που έχουν επιλέξει.
- Δυναμικές. Οι πληροφορίες εξαρτώνται και μεταβάλλονται με τον χρόνο.
- Ασαφείς. Οι τιμές που παρατηρούν οι παίκτες ενδέχεται να μην ταυτίζονται με τις πραγματικές τιμές καθώς επηρεάζονται από θόρυβο και παρεμβολές.

Η εύρεση καταστάσεων ισορροπίας υπό αυτούς τους περιορισμούς αποτελεί μία δύσκολη διαδικασία. Για την άρση αυτών των περιορισμών, η επιστημονική κοινότητα στράφηκε προς το πεδίο της Ενισχυτικής Μάθησης, ώστε οι παίκτες να αποφασίσουν την στρατηγική τους. Ο συνδυασμός της θεωρίας παιγνίων με αλγορίθμους ενισχυτικής μάθησης αποτελεί ένα πολύ ισχυρό εργαλείο για την εύρεση κατανεμημένων λύσεων σε συστήματα με περιορισμένες πληροφορίες.

Η Ενισχυτική Μάθηση (Reinforcement Learning) είναι ένα πεδίο της Μηχανικής Μάθησης, η οποία βασίζεται στην εξής απλή ιδέα[6]: αν μία επιλογή μας συνδέεται με ένα ικανοποιητικό αποτέλεσμα, τότε η τάση να την επαναλάβουμε στο μέλλον ενισχύεται [17]. Η Ενισχυτική Μάθηση συνήθως μοντελοποιείται μαθηματικά σαν ένα πρόβλημα ελαχιστοποίησης, με σκοπό να βρεθεί μία πράξη ή μία στρατηγική επιλογής πράξεων, που να είναι βέλτιστη ως πιο κάποιο σαφώς καθορισμένο κριτήριο. Συνεπώς, έχουμε πράκτορες μάθησης (learning

agents) οι οποίοι πρέπει να ανακαλύψουν, μέσα από μία επαναλαμβανόμενη διαδικασία μάθησης, αλληλεπιδρώντας με το περιβάλλον τους, ποιες πράξεις τους εξασφαλίζουν τη βέλτιστη τιμή κάποιας μετρικής[42, 45].

Θεωρούμε ένα παίγνιο σε κανονική μορφή αναπαράστασης με $\mathcal{N}$ παίκτες. Για να αναλύσουμε την επαναλαμβανόμενη διαδικασία μάθησης θεωρούμε ότι κάθε εποχή που οι παίκτες αποφασίζουν είναι διακριτή και χωρίζεται σε ίσες χρονικές στιγμές (timeslots). Ορίζουμε $a(k) = \{a_1(k), \ldots, a_N(k)\}$ το στρατηγικό προφίλ και $r(k) = \{r_1(k), \ldots, r_N(k)\}$ η λαμβάνουσα τιμή της συνάρτησης ευχαρίστησης του κάθε παίκτη την χρονική στιγμή k. Να σημειωθεί ότι οι λαμβανόμενες τιμές των συναρτήσεων ευχαρίστησης μπορεί να έχουν αλλοιωθεί από θόρυβο. Αφού οι παίκτες λάβουν τις τιμές των συναρτήσεων ευχαρίστησης χρησιμοποιούν ένα κανόνα μάθησης για να ενημερώσουν τις στρατηγικές τους για την επόμενη χρονική στιγμή. Γενικά, η διαδικασία αυτή μπορεί να περιγραφεί ως εξής:

$$a_n(k+1) = F[a_n(1), \ldots, a_n(k); r_n(1), \ldots, r_n(k)]$$

όπου F είναι ο κανόνας μάθησης. Όπως βλέπουμε η διαδικασία αυτή είναι εντελώς κατανεμημένη αφού ο κάθε παίκτης βασίζεται μόνο σε προσωπικές του πληροφορίες.

Παρακάτω παρουσιάζουμε δύο αλγορίθμους μάθησης, οι οποίοι θα χρησιμοποιηθούν και στην διπλωματική εργασία.

3.3.2 Αλγόριθμος Στοχαστικών Αυτόματων Μάθησης

Τα αυτόματα μάθησης (Learning Automata)[31][34], είναι προσαρμοστικές συσκευές που αποφασίζουν, λειτουργούν σε στοχαστικά περιβάλλοντα με ανεπαρκείς πληροφορίες και προοδευτικά βελτιώνουν την απόδοση τους, μέσω μίας διαδικασίας μάθησης. Τα αυτόματα μάθησης, ανά καιρούς, έχουν χρησιμοποιηθεί σε πάρα πολλούς επιστημονικούς κλάδους. Τα χαρακτηριστικά τους, που τους έχουν εξασφαλίσει τόσο μεγάλη δημοτικότητα, είναι η γρήγορη σύγκλισή τους σε σταθερές καταστάσεις με σχετικά μικρή υπολογιστική πολυπλοκότητα.

Η λειτουργία των αυτόματων μάθησης περιγράφεται ως εξής. Το αυτόματο μάθησης (ή αλλιώς στοχαστικό αυτόματο) προσπαθεί να καθορίσει μία βέλτιστη ενέργεια, από ένα πεπερασμένο σύνολο ενεργειών που έχει στην διάθεση του. Ξεκινάει την διαδικασία επιλέγοντας τυχαία μία ενέργεια, παρατηρεί την απόκριση από το περιβάλλον και βάσει αυτής, ανανεώνει τις πιθανότητες που σχετίζονται με την επιλογή των ενεργειών. Η διαδικασία αυτή επαναλαμβάνεται.

Εφαρμόζοντας τον αλγόριθμο των αυτόματων μάθησης σε ένα παίγνιο στην κανονική του μορφή, θεωρούμε ότι ο κάθε παίκτης λειτουργεί σαν ένα αυτόματο που προσπαθεί να επιλέξει την βέλτιστη στρατηγική του με βάση ατομικές του πληροφορίες. Ο επαναληπτικός αλγόριθμος εξελίσσεται σε χρονικές στιγμές (timeslots). Σε κάθε χρονική στιγμή k κάθε παίκτης i επιλέγει την στρατηγική του βάσει της συνάρτησης - κατανομής πιθανότητας $p_i(k) \in \Delta(\mathcal{A}_i)$, όπου $\Delta(\mathcal{A}_i)$ είναι το σύνολο όλων των κατανομών πιθανότητας επί των διαθέσιμων

στρατηγικών $\mathcal{A}_i$.

Κατά την εξέλιξη του αλγορίθμου, σε μία χρονική στιγμή k ο παίκτης επιλέγει μία στρατηγική βάσει της κατανομής πιθανότητας και αν η συνάρτηση ευχαρίστησής του έχει θετική τιμή, τότε η πιθανότητα ο παίκτης να επιλέξει σε επόμενες χρονικές στιγμές την στρατηγική αυτή αυξάνεται. Ο κανόνας που χρησιμοποιείται είναι εντελώς κατανεμημένος και λέγεται κανόνας γραμμικής επιβράβευσης-αδράνειας(linear reward-inaction) και είναι ο εξής [2]:

$$q_{ij}(k+1) = q_{ij}(k) + b\tilde{r}_i(k)(1 - q_{ij}(k)), j = s_i(k)$$

$$q_{ij}(k+1) = q_{ij}(k) - b\tilde{r}_i(k)q_{ij}(k), j \neq s_i(k)$$

όπου το $\tilde{r}_i(k)$ είναι η κανονικοποιημένη τιμή της συνάρτησης ευχαρίστησης στο διάστημα $(0, 1)$ του χρήστη την στιγμή k και b είναι το βήμα μάθησης, με $0 < b < 1$.

Το αυτόματο μάθησης μαθαίνει καλύτερα όσο πιο μικρό είναι το b, αλλά χρειάζεται περισσότερο χρόνο για να καταλήξει σε σύγκλιση.

3.3.3 Αλγόριθμος Binary log-linear

Ένας λιγότερο δημοφιλής κατανεμημένος αλγόριθμος μάθησης είναι ο αλγόριθμος binary log-linear[48]. Η λειτουργία του είναι η εξής: Έστω ότι έχουμε N παίκτες. Σε μία επανάληψη k του αλγορίθμου έχουμε ένα δεδομένο στρατηγικό προφίλ $\{a_1(k), \ldots, a_N(k)\}$ και τις αντίστοιχες τιμές των συναρτήσεων ευχαρίστησης, $\{u_1(k), \ldots, u_N(k)\}$. Ένας παίκτης, έστω i, επιλέγεται τυχαία για να εξερευνήσει και να μάθει. Πιο συγκεκριμένα, επιλέγει τυχαία μία στρατηγική $\hat{a}_i(k) \in \mathcal{A}_i$, με ίσες πιθανότητες $1/|\mathcal{A}_i|$. Με το νέο στρατηγικό προφίλ ο παίκτης i λαμβάνει νέα τιμή της συνάρτησης ικανοποίησης $\hat{u}_i(\kappa)$. Στην $(k+1)$-ιοστή επανάληψη ο παίκτης θα επιλέξει αν θα κρατήσει την στρατηγική που επέλεξε κατά την εξερεύνηση με πιθανότητα:

$$Pr[a_i(k+1) = \hat{a}_i(k)] = \frac{e^{\hat{u}_i\beta}}{e^{\hat{u}_i\beta} + e^{u_i\beta}}$$

ενώ θα την απορρίψει με πιθανότητα:

$$Pr[a_i(k+1) = a_i(k)] = \frac{e^{u_i\beta}}{e^{\hat{u}_i\beta} + e^{u_i\beta}}$$

όπου β είναι η παράμετρος μάθησης. .

Οι στρατηγικές όλων των υπολοίπων παικτών στην $(k+1)$-οστή επανάληψη παραμένουν ανεπηρέαστες, δηλαδή $a_{-i}(k+1) = a_{-i}(k)$. Η παραπάνω διαδικασία επαναλαμβάνεται έως ότου κάποιο κριτήριο σύγκλισης ικανοποιηθεί.

Μία παραλλαγή του αλγορίθμου, που φαίνεται να δίνει γρηγορότερη σύγκλιση, ονομάζεται max-logit. Ο max-logit διαφοροποιείται μόνο στον κανόνα ενημέρωσης, όπου η πιθανότητα ο παίκτης να επιλέξει την στρατηγική εξερεύνησης γίνεται:

$$Pr[a_i(k+1) = \hat{a}_i(k)] = \frac{e^{\hat{u}_i\beta}}{\max\{e^{\hat{u}_i\beta}, e^{u_i\beta}\}}$$

και η αντίστοιχη πιθανότητα να την απορρίψει:

$$Pr[a_i(k+1) = a_i(k)] = \frac{e^{u_i\beta}}{\max\{e^{\hat{u}_i\beta}, e^{u_i\beta}\}}$$

Κεφάλαιο 4

Περιγραφή του ενιαίου αλγοριθμικού πλαισίου

4.1 Εισαγωγή

Στο κεφάλαιο 2 μοντελοποιήσαμε το πρόβλημα και στο κεφάλαιο 3 αναφέραμε κάποια στοιχεία θεωρίας που χρησιμοποιήθηκαν για την λύση του προβλήματος. Στο κεφάλαιο αυτό συνδυάζουμε την γνώση και περιγράφουμε τον ενιαίο αλγόριθμο που χρησιμοποιήσαμε.

4.2 Αλγόριθμος για το Επίπεδο 2

Όπως και στο κεφάλαιο με την περιγραφή του μοντέλου, ξεκινάμε την ανάλυση του αλγορίθμου μας από το επίπεδο 2 το οποίο αφορά την επιλογή του ποσοστού που θα μεταφορτώσουν οι χρήστες, θεωρώντας δεδομένη την κατανομή τους στους εξυπηρετητές. Σε κάθε ένα από τα συμπλέγματα (clusters) που δημιουργούνται, οι χρήστες που ανήκουν σε αυτό, έχουν ως στόχο να εκτελέσουν τις εφαρμογές τους, ικανοποιώντας τους προσωπικούς τους περιορισμούς. Λόγω παρεμβολής και περιορισμένης χωρητικότητας του εξυπηρετητή η επιλογή ενός χρήστη επηρεάζει το αποτέλεσμα ενός άλλου. Πρόκειται για ένα κατ΄ εξοχήν πρόβλημα ικανοποίησης της ποιότητας υπηρεσίας (QoS Satisfaction) το οποίο θα λύσουμε με θεωρία παιγνίων.

4.2.1 Το πρόβλημα σαν Παίγνιο σε Μορφή Ικανοποίησης

Μοντελοποιούμε το πρόβλημα σαν ένα μη συνεργατικό παίγνιο, κατά το οποίο κάθε χρήστης $i \in \mathbb{U}_j$ στοχεύει να βρει μία στρατηγική a_i ώστε να ικανοποιήσει τους περιορισμούς του. Πιο συγκεκριμένα για να είναι ένας παίκτης ικανοποιημένος αρκεί:

$$O^t_{i,j}(a_i, a_{-i}) \leq t_i \quad O^e_{i,j}(a_i, a_{-i}) \leq e_i$$

Το παραπάνω εκφράζεται σαν παίγνιο ικανοποίησης ως εξής:

$$G_s = \{\mathbb{U}_j, \mathcal{A}_i, \gamma_i\}$$

όπου,
$\mathbb{U}_j$: είναι το σύνολο των παικτών που έχουν επιλέξει τον εξυπηρετητή j
$\mathcal{A}_i = \{a_i^{(1)}, \ldots, a_i^{(z)}, \ldots, a_i^{n_i}\}$: είναι το σύνολο των διαθέσιμων στρατηγικών - ποσοστών των δεδομένων του που μπορεί ο χρήστης να επιλέξει να μεταφορτώσει
$\gamma_i(a_{-i}) = \{a_i^z \in \mathcal{A}_i : O_{i,j}^t(a_i, a_{-i}) \leq t_i \;\; O_{i,j}^e(a_i, a_{-i}) \leq e_i\}$: αποτελεί το σύνολο των στρατηγικών του χρήστη i που του ικανοποιούν τους περιορισμούς του, με δεδομένο στρατηγικό προφίλ, a_{-i}.

Στο δεδομένο πρόβλημα δεν γνωρίζουμε αν όλοι οι χρήστες μπορούν να ικανοποιήσουν ταυτόχρονα τους περιορισμούς τους. Για τον λόγο αυτό, θα αναζητήσουμε γενικευμένη ισορροπία ικανοποίησης με στόχο να μεγιστοποιήσουμε τον αριθμό των ικανοποιημένων χρηστών. Στην περίπτωση που μπορούν να ικανοποιηθούν όλοι οι παίκτες τότε έχουμε ισορροπία ικανοποίησης.

Για να προσδιορίσουμε αν ένα παίγνιο κατέχει γενικευμένη ισορροπία ικανοποίησης ή ισορροπία ικανοποίησης, έχει αποδειχθεί ότι είναι NP-hard. Για τον λόγο αυτό, στην βιβλιογραφία συνηθίζεται να υποθέτουν την ύπαρξη του σημείου ισορροπίας, και στην συνέχεια το βρίσκουν. Στην δική μας προσέγγιση, αν το συγκεκριμένο παίγνιο δεν κατέχει σημείο ισορροπίας τότε ο αλγόριθμος μας επιστρέφει μηδενικό στρατηγικό προφίλ, ενώ αν υπάρχει σημείο ισορροπίας τότε συγκλίνει σε αυτό.

Ορίζουμε ως Απόκριση Ικανοποίησης (Satisfaction Response (SR)) ενός χρήστη $i \in \mathbb{U}_j$ την μικρότερη στρατηγική που ικανοποιεί τους περιορισμούς του, δηλαδή

$$SR_i(\mathbf{a_{-i}}) \in \gamma_i(\mathbf{a_{-i}}) \quad \& \quad \forall z \in n_i\text{: } a_i^{(z)} \in \gamma_i(\mathbf{a_{-i}}) \Rightarrow SR_i(\mathbf{a_{-i}}) \leq a_i^{(z)}$$

4.2.2 Αλγόριθμος Επιλογής Μεταφόρτωσης

Για τον υπολογισμού της στρατηγικής μεταφόρτωσης κάθε παίκτη εργαζόμαστε ως εξής: Αρχικά όλοι οι παίκτες μέσα σε ένα σύμπλεγμα επιλέγουν $SR_i = 0$, που σημαίνει ότι δεν μεταφορτώνουν τίποτα στον εξυπηρετητή. Στις επόμενες επαναλήψεις, όλοι οι παίκτες, ο ένας μετά τον άλλο (σε τυχαία σειρά), αρχικά ελέγχουν αν η προηγούμενη τους SR τους ικανοποιεί και την επιλέγουν, αλλιώς ψάχνουν την $a_i \in \mathcal{A}_i$ που αποτελεί SR. Αν καμία στρατηγική δεν τους ικανοποιεί, τότε επιλέγουν $a_i = 0$. Ο αλγόριθμος τερματίζει όταν κανείς παίκτης δεν αλλάζει την στρατηγική του μεταξύ δύο επαναλήψεων.

Algorithm 1 Algorithm - Offloading Decision

Initialize: All Mobile Users start with zero offloading $SR_i^{(0)} = 0, \forall i \in U_j$
$ite = 0$
while Convergence == 0 **do**
  $ite = ite + 1$
  for $i \in \mathbb{U}_j$ **do**
    if $SR_i^{ite-1} \in \gamma_i(\mathbf{a_{-i}})$ **then**
      $SR_i^{(ite)} = SR_i^{(ite-1)}$
    else
      from a_i^0 to $a_i^{n_i}$ find the $SR_i^{(ite)}$ if exists
      if it doesn't exist $SR_i^{(ite)} = 0$
    end if
  end for
end while

Απόδειξη Σύγκλισης Αλγορίθμου Επιλογής Μεταφόρτωσης
Έστω ότι υπάρχει σημείο Ισορροπίας Ικανοποίησης τέτοιο ώστε:

$$\exists \mathbf{a}^+ \in \mathcal{A} = \mathcal{A}_1 \times \cdots \times \mathcal{A}_{|\mathbb{U}_j|} : \forall i \in \mathbb{U}_j \Rightarrow a_i^+ \in \gamma_i(\mathbf{a}_{-i})$$

Την στιγμή $k = 0$ όλοι οι παίκτες έχουν $a_i^{(0)} = 0$. Έστω ότι ο αλγόριθμος για την στιγμή $k = 1$ ξεκινάει από τον παίκτη i, ο οποίος θα έχει απόκριση ικανοποίησης $SR_i(\mathbf{a}_{-i})$. Θα ισχύει ότι $SR_i(\mathbf{a}_{-i}) \leq a_i^+$ αφού είναι ο πρώτος παίκτης που μεταφορτώνει δεδομένα στον εξυπηρετητή και το σύστημα είναι άδειο. Αντίστοιχα, για κάθε παίκτη ισχύει ότι βρίσκει ένα σύστημα λιγότερο συνωστισμένο από το σημείο ισορροπίας ικανοποίησης, εφόσον δεν έχουν μεταφορτώσει όλοι οι παίκτες δεδομένα. Γενικά, για τυχαίο k, έστω ότι παίζει ο παίκτης i και θέλει να καθορίσει την απόκριση ικανοποίησης $SR_i^{(k)}$. Για όλους τους παίκτες που έχουν παίξει πριν από αυτόν ισχύει ότι $\forall l \in \mathbb{U}_j : l < i, SR_l(\mathbf{a_{-l}}) \leq a_l^+$, συνεπώς αυτό εξασφαλίζει ότι ο παίκτης θα μπορεί να προσδιορίσει την απόκριση ικανοποίησης του, αφού το σύστημα είναι λιγότερο συνωστισμένο από το σημείο ισορροπίας ικανοποίησης.

Έστω ότι δεν υπάρχει σημείο ισορροπίας ικανοποίησης τότε ο αλγόριθμος μας, δεν θα αποτύχει αλλά θα βρει κάποιο σημείο που αποτελεί σημείο γενικευμένης ισορροπίας ικανοποίησης, στο οποίο κάποιοι παίκτες είναι ικανοποιημένοι και κάποιοι όχι. Στην περίπτωση που το σύστημα δεν μπορεί να ισορροπήσει σε κάποιο σημείο γενικευμένης ισορροπίας ικανοποίησης τότε ο αλγόριθμος επιστρέφει το $\mathbf{a}^{(0)}$.

4.3 Αλγόριθμος για το Επίπεδο 1

4.3.1 Το πρόβλημα σε μορφή παιγνίου - Συνάρτηση Ευχαρίστησης

Το επίπεδο αυτό αφορά την κατανομή των παικτών στους διαθέσιμους εξυπηρετητές. Για κάθε κατανομή, μετά την εφαρμογή του αλγορίθμου του επιπέδου 2, έχουν προκύψει κάποιοι χρήστες που είναι ικανοποιημένοι και άλλοι που δεν είναι. Για να εκφράσουμε το πόσο ευχαριστημένος είναι ο κάθε παίκτης ορίζουμε μία συνάρτηση ευχαρίστησης, η οποία εξαρτάται από τους περιορισμούς του κάθε παίκτη και τις πραγματικές τιμές που πέτυχε. Αν ένας παίκτης δεν είναι ευχαριστημένος τότε η συνάρτηση αυτή δίνει αρνητική τιμή, ενώ αν είναι ευχαριστημένος θετική. Η συνάρτηση που χρησιμοποιούμε είναι η εξής:

$$z_i(s_i, \mathbf{s_{-i}}) = \begin{cases} (\frac{t_i - O^t_{i,s_i}}{\max\{t_i, O^t_{i,s_i}\}} \cdot \frac{e_i - O^e_{i,s_i}}{\max\{e_i, O^e_{i,s_i}\}})^{0,5} & , \text{αν } O^t_{i,s_i} > t_i \text{ και } O^e_{i,s_i} > e_i \\ -\mid \frac{t_i - O^t_{i,s_i}}{\max\{t_i, O^t_{i,s_i}\}} \cdot \frac{e_i - O^e_{i,s_i}}{\max\{e_i, O^e_{i,s_i}\}} \mid & , \text{αλλιώς} \end{cases} \tag{4.1}$$

όπου στην παραπάνω συνάρτηση το s_i δηλώνει τον εξυπηρετητή που επέλεξε ο παίκτης i ενώ το s_{-i} εκφράζει τις αντίστοιχες επιλογές - στρατηγικές των υπολοίπων παικτών.

Η συνάρτηση αυτή ανήκει πάντα στο διάστημα $[-1, 1]$. Επειδή ο παίκτης διαλέγει πάντα την μικρότερη στρατηγική του που του ικανοποιεί τους περιορισμούς, όταν είναι ικανοποιημένος η τιμή της συνάρτησης ευχαρίστησης είναι πάντα πολύ μικρή. Για τον λόγο αυτό υψώσαμε τον θετικό κλάδο της συνάρτησης στην δύναμη 0,5 ώστε να έχουμε λίγο πιο ομοιόμορφα κατανεμημένες τις θετικές τιμές.

Με την βοήθεια αυτής της συνάρτησης μπορούμε τώρα να μοντελοποιήσουμε και το επίπεδο 1 σε μορφή μη συνεργατικού παιγνίου στην κανονική του μορφή.

$$G = \{\mathbb{U}, S_i, z_i\}$$

όπου,

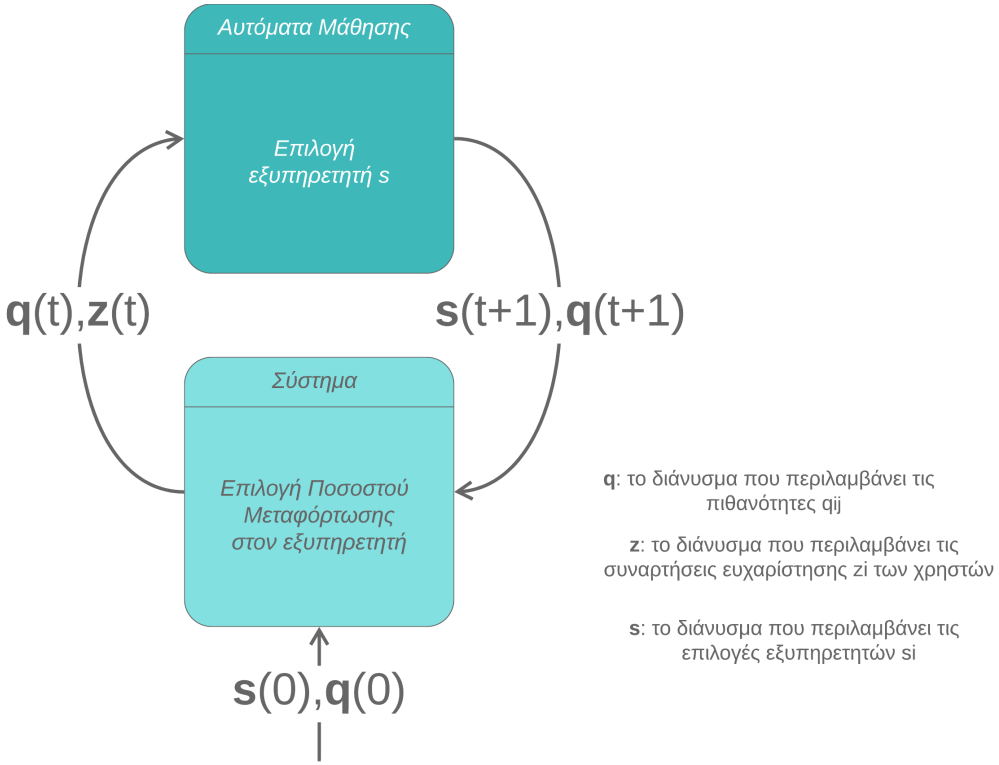
$\mathbb{U}$: είναι το σύνολο όλων των χρηστών στο σύστημα

S_i: είναι το σύνολο των στρατηγικών για τον κάθε χρήστη, συνεπώς το σύνολο των εξυπηρετητών ΚΥΑΔ, $S_i = \{1, \cdots, m\}$

z_i: είναι η συνάρτηση ευχαρίστησης όπως την ορίσαμε παραπάνω.

4.3.2 Εφαρμογή του Αλγορίθμου Αυτόματων Μάθησης στο πρόβλημα μας

Μετά από πειραματική μελέτη, επιλέξαμε τον αλγόριθμο αυτόματων μάθησης για να αντιμετωπίσουμε το πρόβλημα της κατανομής χρηστών στους εξυπηρετητές, για λόγους που θα αναλυθούν στο επόμενο κεφάλαιο.



Σχήμα 4.1: Διάγραμμα Πληροφορίας για τον αλγόριθμο αυτόματων μάθησης

Ορίζουμε $q_{ij}(t)$ την πιθανότητα ο παίκτης i να επιλέξει τον εξυπηρετητή j την χρονική στιγμή t του επαναληπτικού αλγορίθμου αυτόματων μάθησης. Η επιλογή αυτή αρχικά ακολουθεί την διακριτή ομοιόμορφη κατανομή και συνεπώς $\forall i \forall j \quad q_{ij} = \frac{1}{|M|}$. Ορίζουμε επίσης την μετρική συνολική ευχαρίστηση του συστήματος (welfare) ως το άθροισμα των επιμέρους συναρτήσεων ευχαρίστησης όλων των παικτών σε μία στιγμή t του αλγορίθμου αυτόματων μάθησης. Χρησιμοποιούμε την μετρική αυτή για να ελέγξουμε την σύγκλιση του αλγορίθμου αυτόματων μάθησης.

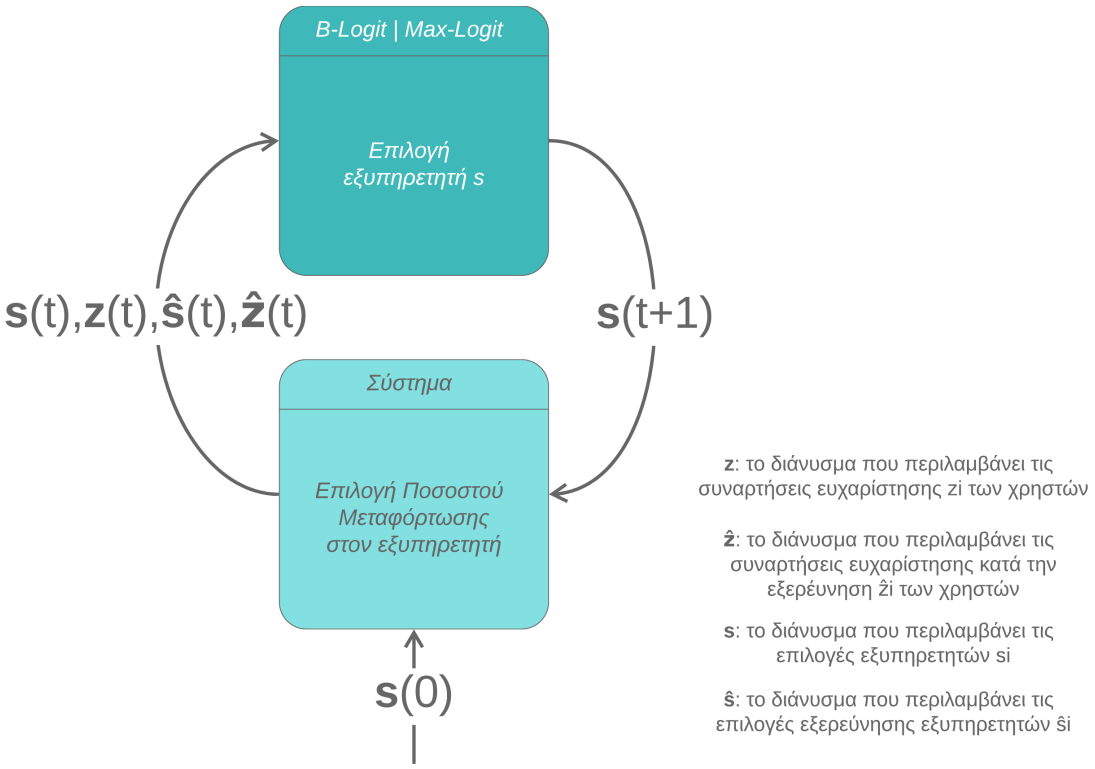
Με βάση όλα τα παραπάνω ο αλγόριθμος του επιπέδου 1 είναι ο εξής:

- Η αρχική κατάσταση από την οποία ξεκινάει ο αλγόριθμος μας είναι τυχαία. Αρχικοποιούμε τις πιθανότητες q_{ij} και βάση αυτών γίνεται γίνεται η αρχική κατανομή των χρηστών σε εξυπηρετητές.
- Στην συνέχεια σε κάθε σύμπλεγμα εφαρμόζουμε τον αλγόριθμο του επιπέδου 2 που περιγράψαμε.
- Ο κάθε χρήστης υπολογίζει την τιμή της συνάρτησης ικανοποίησης, $z_i^{(t)}$
- Αν η τιμή της συνάρτησης ευχαρίστησης ενός παίκτη i, έχοντας επιλέξει τον εξυπηρετητή j, είναι θετική τότε εφαρμόζουμε τον κανόνα ενημέρωσης του αλγορίθμου αυτόματων μάθησης ως εξής:

$$q_{ij}(t+1) = q_{ij}(t) + bz_i^{(t)}(1 - q_{ij}(t)), j = s_i(t)$$
$$q_{ij}(t+1) = q_{ij}(t) - bz_i^{(t)} q_{ij}(t), j \neq s_i(t)$$

- Ο κάθε παίκτης επιλέγει το $s_i(t+1)$, βάση των πιθανοτήτων όπως διαμορφώθηκαν.
- Ο αλγόριθμος τερματίζει όταν η μέση τιμή της συνολικής ευχαρίστησης του συστήματος, μεταξύ δύο παραθύρων διαφέρει λιγότερο μία μικρή τιμή ϵ.

Στο Σχ. 4.1 βλέπουμε την ροή πληροφορίας από το σύστημα προς τον αλγόριθμο αυτόματων μάθησης. Παραθέτουμε παρακάτω και το αντίστοιχο διάγραμμα (σχ. 4.2) για τον αλγόριθμο blogit και maxlogit που χρησιμοποιήσαμε για λόγους σύγκρισης στο κομμάτι των πειραματισμών μας.



Σχήμα 4.2: Διάγραμμα Πληροφορίας για τον αλγόριθμο blogit και maxlogit

Κεφάλαιο 5

Πειραματικά αποτελέσματα

5.1 Εισαγωγή

Στο κεφάλαιο αυτό παρουσιάζονται αναλυτικά όλα τα πειράματα που εκτελέστηκαν προκειμένου να ελεγχθεί η ορθότητα του αλγοριθμικού πλαισίου και την συμπεριφορά του, η δυνατότητα κλιμακωσιμότητας του καθώς και η σύγκριση της απόδοσης του με άλλους αλγορίθμους που θα μπορούσαν να χρησιμοποιηθούν εναλλακτικά.

Στον παρακάτω πίνακα παρουσιάζονται συγκεντρωτικά όλες οι τιμές/εύρος τιμών των μεταβλητών που χρησιμοποιήσαμε στα πειράματα μας.

Πίνακας 5.1: Συμβολισμοί και αριθμητικές τιμές μεταβλητών

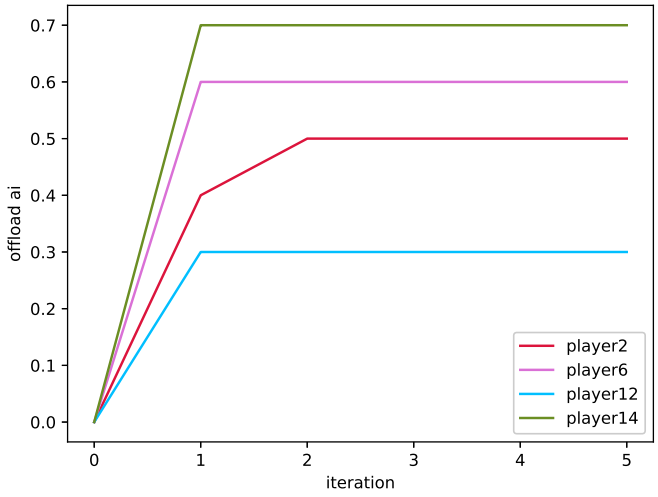
Σύμβολο	Τιμές	Περιγραφή
n	100	αριθμός χρηστών στο σύστημα
m	5	αριθμός εξυπηρετητών ΚΥΑΔ στο σύστημα
I_i	$[1000, 5000]$ KB	μέγεθος της διαδικασίας του χρήστη σε bytes
C_i	$[1000, 5000]$ MegaCycles	κύκλοι που απαιτούνται για την εκτέλεση της διαδικασίας
lc_i	$[0.1, 1]$ GHz	υπολογιστική ικανότητα του χρήστη
le_i	10^{-9} Joules/(CPU-Cycle)	ενεργειακή κατανάλωση
t_i	$[0.6, 0.9] \times (\lambda_i \cdot I_i)/lc_i$	χρονικός περιορισμός του χρήστη
e_i	$[0.6, 0.9] \times (\lambda_i \cdot I_i \cdot le_i)$	ενεργειακός περιορισμός του χρήστη
F_j	$[1, 4] \times 10^3$ GHz	υπολογιστική ικανότητα του εξυπηρετητή
B_j	$[0.6, 1] \times \frac{1}{m} \times \sum I_i$	μέγιστος αριθμός δεδομένων που μπορεί να δεχτεί ο εξυπηρετητής
$d_{i,j}$	$[10, 100]$ m	απόσταση χρήστη i από εξυπηρετητή j
R_j	100 m	η μέγιστη απόσταση που μπορεί να καλύψει ο εξυπηρετητής
$p_{i,j}$	$\frac{d_{i,j}^2}{R_j^2}$	ισχύς μετάδοσης
$g_{i,j}$	$\frac{1}{d_{i,j}^2}$	κέρδος καναλιού
w	5 MHz	εύρος ζώνης
R_{fix}	128 kbps	ρυθμός ανοδικής ζεύξης υπηρεσίας
σ_0^2	10^{-13}	σηματοθορυβικός λόγος
b	0.4	βήμα μάθησης του αλγορίθμου SLA

5.2 Διαγράμματα Λειτουργίας Αλγορίθμου

Αρχικά παρουσιάζουμε τα διαγράμματα που σχετίζουν και μελετούν τα εγγενή χαρακτηριστικά του ενιαίου αλγορίθμου που χρησιμοποιήθηκε.

5.2.1 Σύγκλιση αλγορίθμου Επιλογής Μεταφόρτωσης

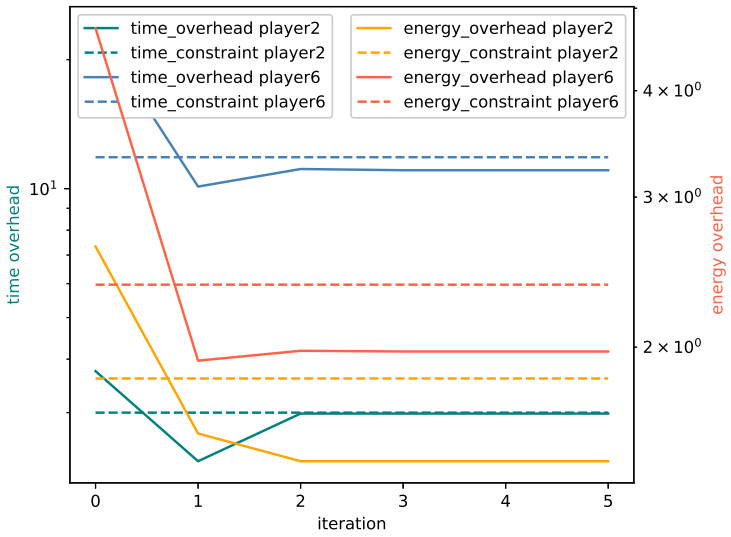
Ο αλγόριθμος του επιπέδου 2 που εφαρμόσαμε, σε όλες τις περιπτώσεις παρουσιάζει πολύ γρήγορη σύγκλιση. Χρειάστηκαν το πολύ 3 επαναλήψεις ώστε ο κάθε παίκτης που βρίσκεται σε ένα σύμπλεγμα να αποφασίσει το ποσοστό a_i που θα μεταφορτώσει. Στο παρακάτω διάγραμμα βλέπουμε τις στρατηγικές a_i 4 παικτών όπως εξελίσσονται όσο περνάνε οι επαναλήψεις (ite) του αλγορίθμου. Για το συγκεκριμένο πείραμα θέσαμε να γίνονται 5 επαναλήψεις, η σύγκλιση όπως φαίνεται προκύπτει νωρίτερα.



Σχήμα 5.1: Σύγκλιση αλγορίθμου Επιλογής Μεταφόρτωσης

5.2.2 Τιμές Ενέργειας και Χρόνου - Αντίστοιχοι Περιορισμοί

Στο διάγραμμα αυτό παρουσιάζονται οι χρονικοί περιορισμοί και οι περιορισμοί στην ενεργειακή κατανάλωση καθώς και οι αντίστοιχες τιμές στον χρόνο και στην ενέργεια που επιτυγχάνουν, δύο από τους τέσσερις χρήστες του διαγράμματος 1. Ο αριστερός άξονας αντιπροσωπεύει τον χρόνο και ο δεξιός την ενέργεια, χρησιμοποιήθηκε λογαριθμική κλίμακα.

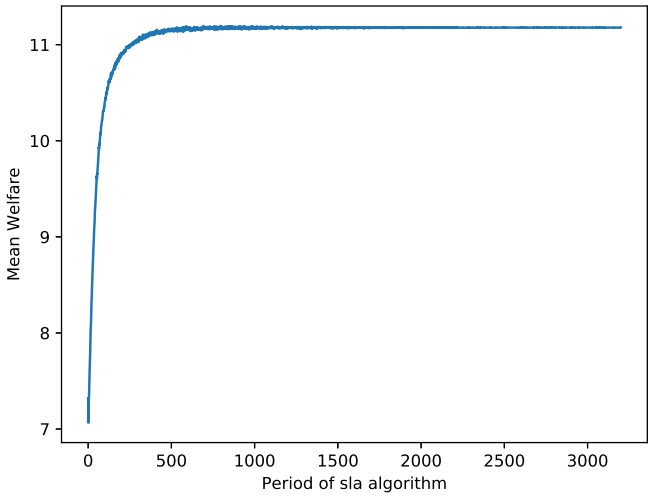


Σχήμα 5.2: Περιορισμοί και μετρούμενες τιμές χρόνου και ενέργειας

Αυτό που αξίζει να σχολιαστεί στο διάγραμμα αυτό είναι ότι, όπως συμβαίνει στις δύο παραπάνω περιπτώσεις, οι χρήστες ικανοποιούν οριακά τους χρονικούς περιορισμούς τους ενώ οι περιορισμοί για την κατανάλωση ενέργειας ικανοποιούνται και με μικρότερες στρατηγικές a_i. Αυτό δείχνει ότι οι χρονικοί περιορισμοί ικανοποιούνται ως επί το πλείστον πιο δύσκολα και αυτοί καθορίζουν την τιμή του a_i.

5.2.3 Σύγκλιση αλγορίθμου αυτόματων μάθησης

Στο επίπεδο 1, για να ελέγξουμε την σύγκλιση του αλγορίθμου χρησιμοποιήσαμε σαν μετρική την συνολική ευχαρίστηση του συστήματος (welfare), το οποίο ορίσαμε στο κεφάλαιο 4. Για να αντιμετωπίσουμε την στοχαστικότητα του αλγορίθμου αυτόματων μάθησης και να έχουμε πιο ομοιόμορφα και έγκυρα αποτελέσματα, χρησιμοποιήσαμε την μέθοδο Monte Carlo.



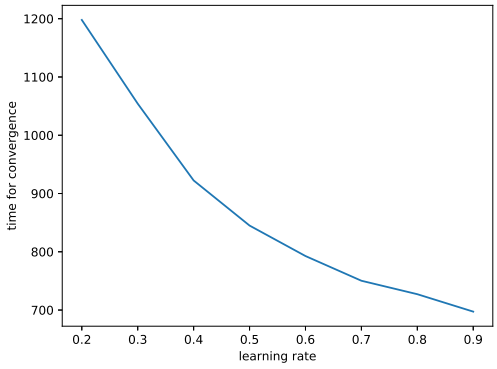
Σχήμα 5.3: Σύγκλιση αλγορίθμου αυτόματων μάθησης

Πιο συγκεκριμένα, τρέξαμε τον ενιαίο αλγόριθμο 1000 φορές, κρατώντας ίδιες αρχικοποιήσεις για τους χρήστες και τους εξυπηρετητές καθώς και την ίδια αρχική κατανομή και πήραμε τον μέσο όρο της συνολικής ευχαρίστησης του συστήματος των 1000 πειραμάτων. Η μέση συνολική ευχαρίστηση του συστήματος κατά την εξέλιξη του αλγορίθμου αυτόματων μάθησης φαίνεται στο παρακάτω διάγραμμα.

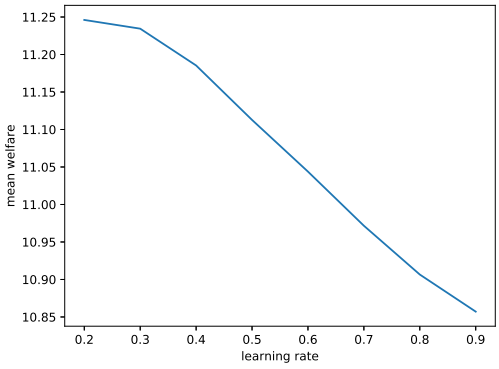
5.2.4 Παράμετρος μάθησης b

Όπως αναφέραμε και στην θεωρία, η παράμετρος μάθησης του αλγορίθμου αυτόματων μάθησης, επηρεάζει τον χρόνο σύγκλισης και την απόδοση του αλγορίθμου. Υπάρχει γενικά trade-off μεταξύ του χρόνου σύγκλισης και της απόδοσης. Με μικρή τιμή στο b ο αλγόριθμος αυτόματων μάθησης μαθαίνει καλύτερα (υψηλότερη απόδοση) αλλά απαιτεί περισσότερο χρόνο.

Για να ελέγξουμε το πώς επηρεάζεται η συμπεριφορά του αλγορίθμου μας από την παράμετρο μάθησης b τρέξαμε το ίδιο πείραμα για διάφορες τιμές του b, χρησιμοποιώντας πάλι την μέθοδο Monte Carlo. Τα διαγράμματα που προέκυψαν είναι τα εξής:



Σχήμα 5.4: Παράμετρος μάθησης b: χρόνος σύγκλισης



Σχήμα 5.5: Παράμετρος μάθησης b: μέση ευχαρίστηση συστήματος

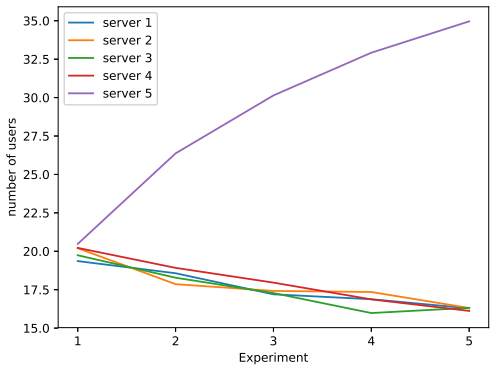
Από τα αποτελέσματα που πήραμε, επαληθεύεται η θεωρία. Στα υπόλοιπα πειράματα χρησιμοποιήθηκε η παράμετρος μάθησης $b = 0.4$ που αποτελεί καλό συμβιβασμό μεταξύ απόδοσης και χρόνου σύγκλισης.

5.2.5 Μελέτη της επίδρασης των χαρακτηριστικών των εξυπηρετητών

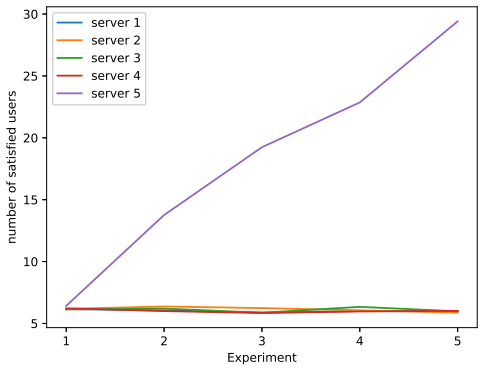
Κάθε εξυπηρετητής διαθέτει ένα ανώτατο όριο στο συνολικό αριθμό από bytes που μπορεί να δεχτεί (B_j) και μία δεδομένη υπολογιστική ικανότητα (F_j). Για να ελέγξουμε το πόσο επηρεάζουν τα χαρακτηριστικά αυτά την επίδοση του κάθε εξυπηρετητή εκτελέσαμε το εξής πείραμα.

Θέσαμε για όλους τους εξυπηρετητές μία μικρή χωρητικότητα και σχετικά μικρή υπολογιστική ικανότητα και σταδιακά κάναμε έναν από τους εξυπηρετητές πιο δυνατό, βελτιώνοντας τα παραπάνω χαρακτηριστικά. Στο πείραμα αυτό θεωρήσαμε όλες τις αποστάσεις μεταξύ χρηστών

και εξυπηρετητών σταθερές και ίσες μεταξύ τους. Για κάθε ένα από τα ζεύγη χωρητικότητας και υπολογιστικής ισχύος που δοκιμάσαμε, μετρήσαμε σε κάθε εξυπηρετητή τον αριθμό των χρηστών που τον επέλεξαν και πόσοι από αυτούς ήταν ικανοποιημένοι.

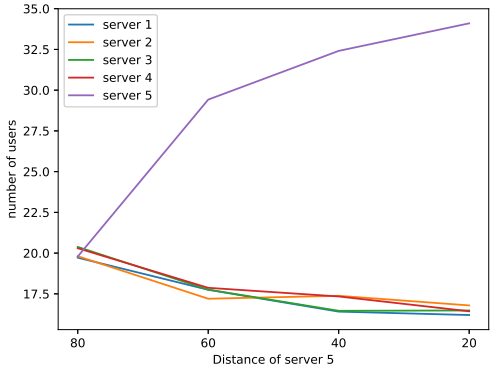


Σχήμα 5.6: Χαρακτηριστικά εξυπηρετητή: χρήστες ανά εξυπηρετητή

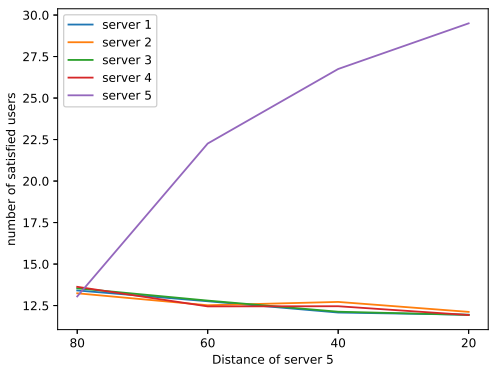


Σχήμα 5.7: Χαρακτηριστικά εξυπηρετητή: ικανοποιημένοι χρήστες ανά εξυπηρετητή

Επιπλέον, ελέγξαμε πώς η απόσταση εξυπηρετητών από τους χρήστες επηρεάζει την επίδοση τους. Κρατώντας τα δύο παραπάνω χαρακτηριστικά σταθερά για όλους τους εξυπηρετητές, για έναν εξυπηρετητή μειώσαμε σταδιακά την απόσταση του από τους χρήστες και μετρήσαμε πάλι τον αριθμό των χρηστών που τους επέλεξαν, και πόσοι από αυτούς ήταν ικανοποιημένοι.



Σχήμα 5.8: Απόσταση εξυπηρετητή: χρήστες ανά εξυπηρετητή



Σχήμα 5.9: Απόσταση εξυπηρετητή: ικανοποιημένοι χρήστες ανά εξυπηρετητή

Σε όλα τα παραπάνω διαγράμματα επιβεβαιώνεται ότι όσο βελτιώνουμε τα χαρακτηριστικά ενός εξυπηρετητή, τόσους περισσότερους χρήστες είναι σε θέση να ικανοποιήσει.

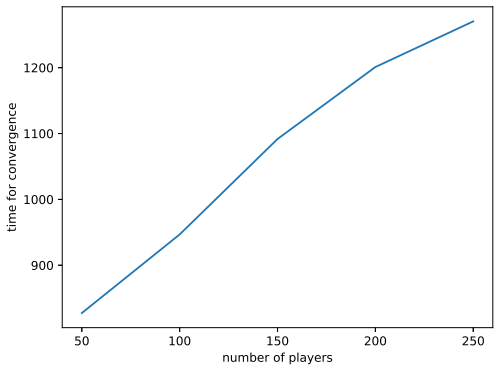
5.3 Διαγράμματα Κλιμακωσιμότητας

Στην κατηγορία αυτή εξετάζουμε την συμπεριφορά του αλγοριθμικού πλαισίου μας όταν αυξάνεται ο αριθμός χρηστών ή εξυπηρετητών. Πραγματοποιήσαμε δύο πειράματα. Στο πρώτο

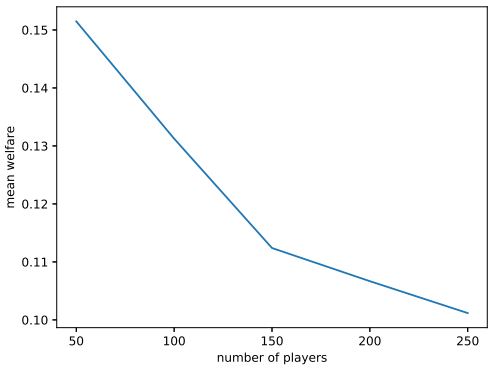
αυξάναμε σταδιακά τους συνολικούς παίκτες του συστήματος με σταθερό αριθμό εξυπηρετητών ενώ στο δεύτερο αυξάναμε σταδιακά τον αριθμό εξυπηρετητών κρατώντας ίδιο αριθμό παικτών. Και στις δύο περιπτώσεις μετρήσαμε τη συνολική ευχαρίστηση του συστήματος και τον χρόνο σύγκλισης.

5.3.1 Χρήστες Κινητών

Με $m = 6$ εξυπηρετητές θέσαμε 50,100,150,200,250 παίκτες. Για τον κάθε εξυπηρετητή κρατήσαμε σταθερό το ποσοστό των συνολικών bytes που μπορεί να δεχτεί αλλά εφόσον αυξανόντουσαν τα συνολικά bytes αυξανόταν ανάλογα και η χωρητικότητα του κάθε εξυπηρετητή. Εργαστήκαμε πάλι με την μέθοδο Monte Carlo και εκτελέσαμε 1000 πειράματα για κάθε αριθμό παικτών. Μετρήσαμε τον μέσο χρόνο σύγκλισης και τη μέση τιμή της συνολικής ευχαρίστησης ανά παίκτη:



Σχήμα 5.10: Κλιμακωσιμότητα χρηστών: χρόνος σύγκλισης

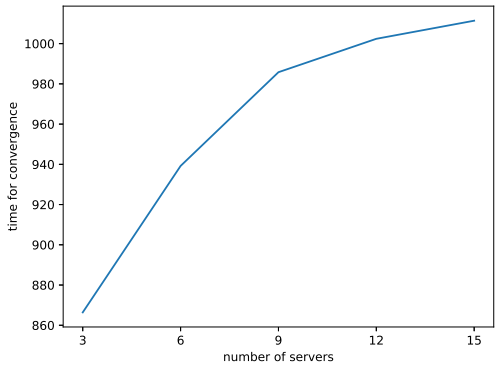


Σχήμα 5.11: Κλιμακωσιμότητα χρηστών: μέση ευχαρίστηση του συστήματος

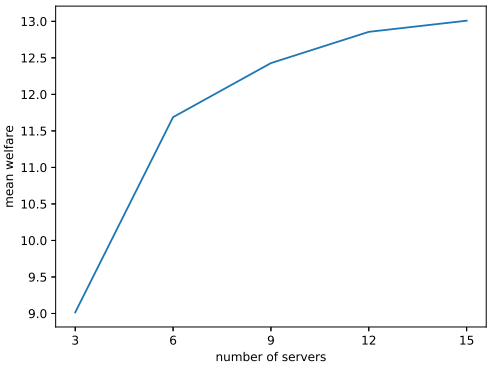
Αυτό που βλέπουμε είναι ότι ο χρόνος σύγκλισης αυξάνεται όσο προσθέτουμε παίκτες, καθώς η διαδικασία κατανομής φορτίου γίνεται πιο περίπλοκη. Επίσης η μέση τιμή της συνολικής ευχαρίστησης για ένα παίκτη πάλι μειώνεται λόγω της συμφόρησης που προκαλείται στους εξυπηρετητές από τους πολλούς χρήστες και τα φαινόμενα παρεμβολής που δημιουργούνται στο δίκτυο επικοινωνίας.

5.3.2 Εξυπηρετητές

Με $n = 100$ παίκτες δοκιμάσαμε 3,6,9,12,15 εξυπηρετητές. Στην περίπτωση αυτή σε κάθε πείραμα είχαμε διαφορετική αρχική κατανομή των χρηστών στους εξυπηρετητές. Εργαστήκαμε πάλι με την μέθοδο Monte Carlo και εκτελέσαμε 1000 πειράματα για κάθε αριθμό εξυπηρετητών. Η χωρητικότητα του κάθε εξυπηρετητή επηρεαζόταν από τον αριθμό m, με τέτοιο τρόπο ώστε να διατηρείται η συνολική χωρητικότητα του συστήματος σταθερή.



Σχήμα 5.12: Κλιμακωσιμότητα εξυπηρετητών: χρόνος σύγκλισης



Σχήμα 5.13: Κλιμακωσιμότητα εξυπηρετητών: μέση ευχαρίστηση του συστήματος

Στα αποτελέσματα αυτά βλέπουμε ότι παρόλο που η συνολική χωρητικότητα του συστήματος παραμένει σταθερή, οι παίκτες εξυπηρετούνται καλύτερα όταν μοιράζονται σε περισσότερους εξυπηρετητές. Αυτό είναι αναμενόμενο αν σκεφτούμε ότι μειώνονται οι παρεμβολές που οφείλονται σε συμφόρηση των χρηστών σε εξυπηρετητές. Από την άλλη, περιμέναμε η σύγκλιση του αλγορίθμου να καθυστερεί προσθέτοντας εξυπηρετητές, ενώ αυτό που βλέπουμε είναι μία μικρή αύξηση στον χρόνο σύγκλισης. Αυτό οφείλεται στο γεγονός ότι η σύγκλιση του αλγορίθμου αυτόματων μάθησης εξαρτάται από τον αριθμό των ικανοποιημένων παικτών, αφού μόνο αυτοί εκτελούν τον κανόνα μάθησης. Οι παίκτες έχουν περισσότερες επιλογές, αλλά ικανοποιούνται πιο εύκολα. Αυτές οι δύο αντίρροπες δυνάμεις προκαλούν την μικρή αύξηση που βλέπουμε στον χρόνο σύγκλισης.

5.4 Συγκριτικά Διαγράμματα

Στα διαγράμματα αυτά εκτελούμε πειράματα για να συγκρίνουμε την επίδοση των αλγορίθμων που χρησιμοποιούμε, με άλλους, εναλλακτικούς για την επίλυση του προβλήματος αλγορίθμους, στα επίπεδα 1 και 2.

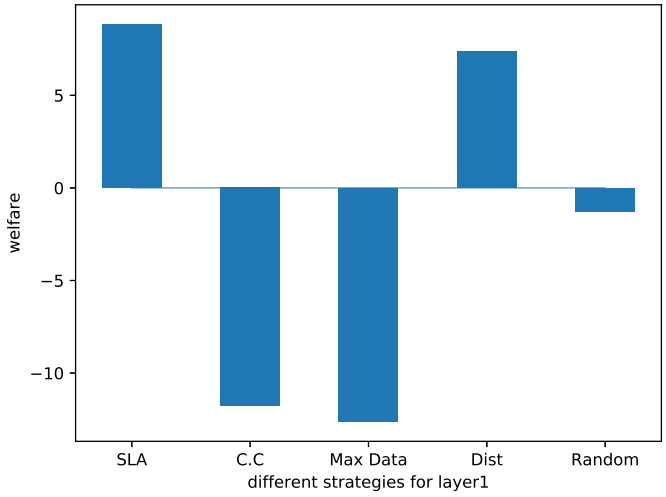
Συγκριτικά διαγράμματα στο Επίπεδο 1

Στο επίπεδο 1, διαλέξαμε τον αλγόριθμο αυτόματων μάθησης από το πεδίο της ενισχυτικής μάθησης για να λύσουμε το πρόβλημα της επιλογής εξυπηρετητή από τους χρήστες. Δοκιμάσαμε εναλλακτικά οι παίκτες να επιλέγουν εξυπηρετητή με τα εξής τέσσερα απλά κριτήρια:

- Ο κάθε χρήστης επιλέγει τον εξυπηρετητή με τη μέγιστη υπολογιστική δυνατότητα (Max Computational Capability - C.C).
- Ο κάθε χρήστης επιλέγει τον εξυπηρετητή με τη μέγιστη χωρητικότητα σε δεδομένα που μπορεί να δεχτεί (Max Data Capacity - Max Data).
- Ο κάθε χρήστης επιλέγει τον εξυπηρετητή με την μικρότερη απόσταση από αυτόν (Min Diastance - Dist).

- Ο κάθε χρήστης επιλέγει τυχαία έναν από τους εξυπηρετητές. Λόγω της στοχαστικότητας αυτού του αλγορίθμου, πήραμε την μέση τιμή από 10000 τυχαίες κατανομές των χρηστών στους εξυπηρετητές (Random).

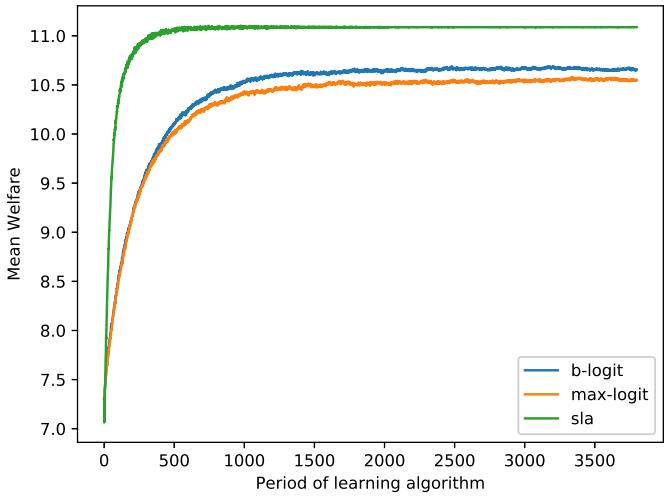
Όλες οι παραπάνω επιλογές είναι ντετερμινιστικές και δεν είναι επαναληπτικές διαδικασίες. Παρακάτω βλέπουμε τη μέση τιμή της συνολικής ευχαρίστησης του συστήματος που μετρήθηκε για κάθε περίπτωση.



Σχήμα 5.14: Συγκριτικό διάγραμμα για τον επίπεδο 1

Αυτό που μπορούμε να σχολιάσουμε είναι ότι η καλύτερη από τις άφελείς'(naive) στρατηγικές που δοκιμάσαμε είναι αυτή που ταιριάζει κάθε χρήστη με τον πιο κοντινό του εξυπηρετητή. Αυτό συμβαίνει διότι, αντίθετα με τις άλλες στρατηγικές, έχουμε μία σχετικά ομοιόμορφη κατανομή των χρηστών στους εξυπηρετητές. Αν στο πείραμα μας, θέταμε κάποιον εξυπηρετητή να είναι σε πιο ΄κεντρικό΄ σημείο, σε σχέση με τους άλλους εξυπηρετητές, και συνεπώς να επιλέγεται από περισσότερους παίκτες τότε δεν θα είχαμε πάλι το φαινόμενο της συμφόρησης και πολύ χειρότερα αποτελέσματα.

Επιπλέον δοκιμάσαμε έναν ακόμα αλγόριθμο από το πεδίο της ενισχυτικής μάθησης, τον b-logit και μία παραλλαγή του, τον max-logit. Οι δύο αυτοί αλγόριθμοι έχουν περιγραφεί εκτενώς στο κεφάλαιο της θεωρίας. Παρακάτω παρουσιάζουμε την σύγκλιση, και την συνολική ευχαρίστηση συστήματος που επιτυγχάνουν οι δύο αυτοί αλγόριθμοι σε σύγκριση με τον αλγόριθμο αυτόματων μάθησης.



Σχήμα 5.15: Σύγκριση SLA με blogit και maxlogit

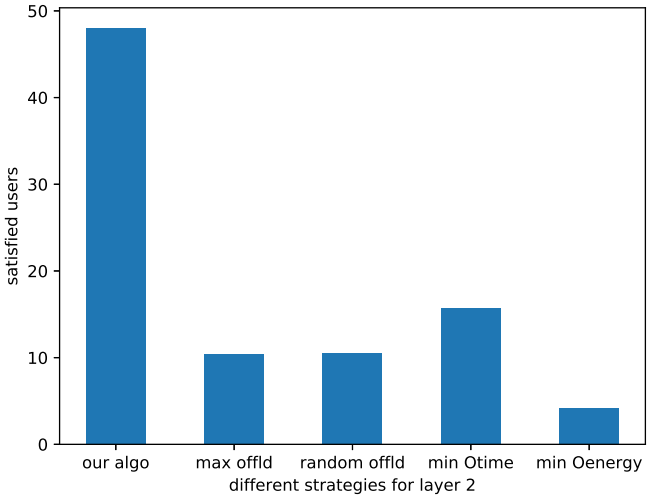
Αυτό που προκύπτει είναι ότι ο αλγόριθμος αυτόματων μάθησης έχει γρηγορότερη σύγκλιση αλλά και καταλήγει σε καλύτερο αποτέλεσμα ως προς τη συνολική ευχαρίστηση του συστήματος. Η παράμετρος μάθησης που χρησιμοποιήθηκε στον b-logit και max-logit είναι $b = 800$.

Συγκριτικά διαγράμματα στο επίπεδο 2

Το επίπεδο 2 αφορά την επιλογή του ποσοστού που θα μεταφορτώσει ο κάθε χρήστης στον εξυπηρετητή που έχει επιλέξει. Δοκιμάσαμε τους εξής τρόπους επιλογής:

- Ο παίκτης ανεβάζει το μεγαλύτερο ποσοστό a_i που μπορεί, εφόσον ο εξυπηρετητής που έχει επιλέξει δεν έχει γεμίσει. (Max Offloading)
- Ο παίκτης ανεβάζει το μικρότερο ποσοστό a_i που μπορεί, εφόσον ο εξυπηρετητής που έχει επιλέξει δεν έχει γεμίσει. (Min Offloading)
- Ο παίκτης ανεβάζει ένα τυχαίο ποσοστό a_i, εφόσον ο εξυπηρετητής που έχει επιλέξει δεν έχει γεμίσει. (Random Offloading)
- Η επιλογή του a_i γίνεται με την μορφή παιγνίου, με τους παίκτες να επιλέγουν την στρατηγική που τους δίνει το μικρότερο συνολικό χρόνο. (min time overhead)
- Η επιλογή του a_i γίνεται με την μορφή παιγνίου, με τους παίκτες να επιλέγουν την στρατηγική που τους δίνει τη μικρότερη ενέργεια. (min energy overhead)

Στο επίπεδο 2 ο αλγόριθμος μας εστιάζει στον αριθμό των ικανοποιημένων παικτών και δεν χρησιμοποιείται κάποια μετρική που να εκφράζει τον βαθμό ικανοποίησης. Για τον λόγο αυτό θεωρήσαμε 10000 τυχαίες κατανομές και μετρήσαμε με κάθε μία από τις παραπάνω στρατηγικές τους ικανοποιημένους παίκτες. Στο παρακάτω διάγραμμα φαίνεται ο μέσος όρος από αυτές τις μετρήσεις.



Σχήμα 5.16: Συγκριτικό διάγραμμα για επίπεδο 2

Αυτό που βλέπουμε είναι ότι ο δικός μας αλγόριθμος δίνει πολύ καλύτερο μέσο αριθμό ικανοποιημένων παικτών σε σχέση με τους υπόλοιπους που δοκιμάσαμε.

Κεφάλαιο 6

Επίλογος

6.1 Σύνοψη και συμπεράσματα

Στην παρούσα διπλωματική μελετήθηκε ένα βασικό πρόβλημα που συνδέεται με την τεχνολογία Κινητής Υπολογιστικής στα Άκρα του Δικτύου (ΚΥΑΔ). Αρχικά ορίστηκε η έννοια Υπολογιστικού Νέφους και πώς αυτή σταδιακά εξελίχθηκε στην ΚΥΑΔ. Διερευνήθηκαν τα διάφορα ερευνητικά προβλήματα που συνδέονται με τα περιβάλλοντα ΚΥΑΔ και καλείται να αντιμετωπίσει η σύγχρονη επιστημονική κοινότητα. Με ένα από αυτά ασχολούμαστε στην παρούσα διπλωματική, το πρόβλημα της μεταφόρτωσης υπολογισμού.

Το πρόβλημα της μεταφόρτωσης υπολογισμού συνδέεται με όλες τις επιλογές που πρέπει να πραγματοποιήσει ένας χρήστης κινητής συσκευής προκειμένου να προσδιοριστεί ο εξυπηρετητής άκρης που θα εκτελέσει μέρος των απαιτητικών εφαρμογών των χρηστών. Στο δικό μας σύστημα, ο χρήστης i μπορεί να επιλέξει μεταξύ M εξυπηρετητών και μεταξύ n_i ποσοστών μεταφόρτωσης της εφαρμογής του. Για την επιλογή του εξυπηρετητή από τους χρήστες υιοθετήθηκε μία λύση βασισμένη στην θεωρία παιγνίων κατά την οποία οι χρήστες συμπεριφέρονται σαν αυτόματα μάθησης, προκειμένου να μεγιστοποιηθεί η συνολική ευχαρίστηση του συστήματος. Σε κάθε εποχή του αλγορίθμου μάθησης, για να προσδιοριστεί το ποσοστό που θα μεταφορτώσουν οι χρήστες στον εξυπηρετητή που έχουν επιλέξει και να καθοριστεί η τιμή στην συνάρτηση ευχαρίστησης τους, αντιμετωπίσαμε το πρόβλημα πάλι σαν ένα παίγνιο και αναζητήσαμε την εύρεση ενός σημείου ισορροπίας ικανοποίησης, ή γενικευμένης ισορροπίας ικανοποίησης, στις περιπτώσεις που δεν ήταν εφικτό όλοι οι παίκτες να ικανοποιηθούν ταυτόχρονα.

Υλοποιήσαμε το παραπάνω ενιαίο αλγοριθμικό πλαίσιο και πραγματοποιήσαμε μία σειρά προσομοιώσεων. Από τα πειράματα μας προέκυψαν τα εξής:

- Ο αλγόριθμος επιλογής ποσοστού μεταφόρτωσης που υλοποιήσαμε καταλήγει πάντα σε σημείο ισορροπίας ικανοποίησης ή γενικευμένης ισορροπίας ικανοποίησης, σε πολύ λίγες επαναλήψεις.

- Ο αλγόριθμος αυτόματων μάθησης συγκλίνει γρήγορα σε μία κατάσταση κατά την οποία η συνολική ευχαρίστηση του συστήματος είναι η καλύτερη που καταφέραμε να

πετύχουμε, συγκρινόμενη με άλλους εναλλακτικούς αλγορίθμους μάθησης και στρατηγικές επιλογής εξυπηρετητή.

- Επαληθεύσαμε πειραματικά τις ιδιότητες της παραμέτρου μάθησης β του αλγορίθμου αυτόματων μάθησης.
- Διαπιστώσαμε ότι το αλγοριθμικό πλαίσιό μας, ανάλογα με τα χαρακτηριστικά των εξυπηρετητών (μέγιστη χωρητικότητα, υπολογιστική ικανότητα και μέση απόσταση από τους χρήστες), κατανείμει περισσότερους χρήστες στους πιο ισχυρούς εξυπηρετητές, αφού έχουν μεγαλύτερη δυνατότητα να τους ικανοποιήσουν.
- Η αύξηση των χρηστών στο σύστημα μας, είχε ως αποτέλεσμα χειρότερη σύγκλιση αλλά και μείωση της μέσης τιμής ευχαρίστησης ανά παίκτη στο σύστημα. Αυτό οφείλεται σε φαινόμενα συμφόρησης στους εξυπηρετητές και παρεμβολής στα κανάλια επικοινωνίας.
- Η αύξηση των εξυπηρετητών στο σύστημα είχε ως αποτέλεσμα μικρή αύξηση στον χρόνο σύγκλισης και αύξηση της συνολικής ευχαρίστησης του συστήματος. Αυτό συμβαίνει διότι η σύγκλιση του αλγορίθμου αυτόματων μάθησης εξαρτάται από τον αριθμό των ικανοποιημένων παικτών, αφού μόνο αυτοί εκτελούν τον κανόνα μάθησης. Οι παίκτες έχουν περισσότερες επιλογές, αλλά ικανοποιούνται πιο εύκολα.
- Πειραματισμοί με εναλλακτικούς αλγόριθμους (blogit,maxlogit και άλλες πιο απλές στρατηγικές επιλογής εξυπηρετητή) έδωσαν χειρότερα αποτελέσματα σε μέση τιμή ευχαρίστησης του συστήματος από τον αλγόριθμο αυτόματων μάθησης που χρησιμοποιήθηκε. Επιπλέον ο αλγόριθμος αυτομάτων μάθησης έδωσε γρηγορότερη σύγκλιση από τους blogit, maxlogit χρησιμοποιώντας το ίδιο κριτήριο σύγκλισης.
- Τέλος, συγκρίναμε τον αλγόριθμο επιλογής ποσοστού μεταφόρτωσης που χρησιμοποιήσαμε με εναλλακτικές στρατηγικές επιλογής ποσοστού και διαπιστώσαμε ότι ο αλγόριθμος που χρησιμοποιήθηκε δίνει πολύ μεγαλύτερο μέσο αριθμό ικανοποιημένων χρηστών στο σύστημα.

6.2 Μελλοντικές επεκτάσεις

Μελλοντικές επεκτάσεις της παρούσας διπλωματικής εργασίας περιλαμβάνουν περαιτέρω μελέτη διαφορετικών ειδών ισορροπιών για το σύστημα και σύγκριση τους με την ισορροπία ικανοποίησης που χρησιμοποιήθηκε. Επιπλέον χρήσιμο κρίνεται να μελετηθούν πειραματικά και άλλοι αλγόριθμοι από το πεδίο της ενισχυτικής μάθησης, όπως μάθηση μέσω δοκιμής και λάθους (trial and error), προκειμένου να συγκριθούν οι χρόνοι σύγκλισης και η συνολική ευχαρίστηση του συστήματος και να επιλεγεί ο καταλληλότερος.

Βιβλιογραφία

[1] A. Ahmed and E. Ahmed. "A survey on mobile edge computing". In: *2016 10th International Conference on Intelligent Systems and Control (ISCO)*. Jan. 2016, pp. 1–8. DOI:

[2] Pavlos Athanasios Apostolopoulos, Eirini Eleni Tsiropoulou, and Symeon Papavassiliou. "Demand response management in smart grid networks: A two-stage game-theoretic learning-based approach". In: *Mobile Networks and Applications* (2018), pp. 1–14.

[3] Pavlos Athanasios Apostolopoulos, Eirini Eleni Tsiropoulou, and Symeon Papavassiliou. "Game-Theoretic Learning-Based QoS Satisfaction in Autonomous Mobile Edge Computing". In: *2018 Global Information Infrastructure and Networking Symposium (GIIS)*. IEEE. 2018, pp. 1–5.

[4] Marios Avgeris et al. "Adaptive resource allocation for computation offloading: A control-theoretic approach". In: *ACM Transactions on Internet Technology (TOIT)* 19.2 (2019), p. 23.

[5] Sergio Barbarossa, Stefania Sardellitti, and Paolo Di Lorenzo. "Communicating while computing: Distributed mobile cloud computing over 5G heterogeneous networks". In: *IEEE Signal Processing Magazine* 31.6 (2014), pp. 45–55.

[6] Andrew G Barto. "Reinforcement learning". In: *Neural systems for control*. Elsevier, 1997, pp. 7–30.

[7] Min Chen et al. "Edge-CoCaCo: Toward joint optimization of computation, caching, and communication on edge cloud". In: *IEEE Wireless Communications* 25.3 (2018), pp. 21–27.

[8] Xianfu Chen et al. "Optimized computation offloading performance in virtual edge computing systems via deep reinforcement learning". In: *IEEE Internet of Things Journal* (2018).

[9] Xianfu Chen et al. "Wireless resource scheduling in virtualized radio access networks using stochastic learning". In: *IEEE Transactions on Mobile Computing* 17.4 (2017), pp. 961–974.

[10] Xu Chen et al. "Efficient multi-user computation offloading for mobile-edge cloud computing". In: *IEEE/ACM Transactions on Networking* 24.5 (2015), pp. 2795–2808.

[11] Byung-Gon Chun et al. "Clonecloud: elastic execution between mobile device and cloud". In: *Proceedings of the sixth conference on Computer systems*. ACM. 2011, pp. 301–314.

[12] Eduardo Cuervo et al. "MAUI: making smartphones last longer with code offload". In: *Proceedings of the 8th international conference on Mobile systems, applications, and services*. ACM. 2010, pp. 49–62.

[13] Gerard Debreu. "A social equilibrium existence theorem". In: *Proceedings of the National Academy of Sciences* 38.10 (1952), pp. 886–893.

[14] Hoang T Dinh et al. "A survey of mobile cloud computing: architecture, applications, and approaches". In: *Wireless communications and mobile computing* 13.18 (2013), pp. 1587–1611.

[15] Michail Fasoulakis, Eirini Eleni Tsiropoulou, and Symeon Papavassiliou. "Satisfy instead of maximize: Improving operation efficiency in wireless communication networks". In: *Computer Networks* 159 (2019), pp. 135–146.

[16] Niroshinie Fernando, Seng W Loke, and Wenny Rahayu. "Mobile cloud computing: A survey". In: *Future generation computer systems* 29.1 (2013), pp. 84–106.

[17] Georgios Fragkos, Pavlos Athanasios Apostolopoulos, and Eirini Eleni Tsiropoulou. "ESCAPE: Evacuation Strategy through Clustering and Autonomous Operation in Public Safety Systems". In: *Future Internet* 11.1 (2019), p. 20.

[18] Pedro Garcia Lopez et al. "Edge-centric computing: Vision and challenges". In: *ACM SIGCOMM Computer Communication Review* 45.5 (2015), pp. 37–42.

[19] Mathew Goonewardena et al. "Generalized satisfaction equilibrium for service-level provisioning in wireless networks". In: *IEEE Transactions on Communications* 65.6 (2017), pp. 2427–2437.

[20] Mike Jia, Jiannong Cao, and Lei Yang. "Heuristic offloading of concurrent tasks for computation-intensive applications in mobile cloud computing". In: *2014 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*. IEEE. 2014, pp. 352–357.

[21] Slacđana Jošilo and György Dán. "A game theoretic analysis of selfish mobile computation offloading". In: *IEEE INFOCOM 2017-IEEE Conference on Computer Communications*. IEEE. 2017, pp. 1–9.

[22] Roelof Kemp et al. "Cuckoo: a computation offloading framework for smartphones". In: *International Conference on Mobile Computing, Applications, and Services*. Springer. 2010, pp. 59–79.

[23] Levent Koçkesen and Efe A Ok. "An introduction to game theory". In: *University Efe A. Ok New York University July* 8 (2007).

[24] Sokol Kosta et al. "Thinkair: Dynamic resource allocation and parallel execution in the cloud for mobile code offloading". In: *2012 Proceedings IEEE Infocom*. IEEE. 2012, pp. 945–953.

[25] Karthik Kumar et al. "A survey of computation offloading for mobile systems". In: *Mobile Networks and Applications* 18.1 (2013), pp. 129–140.

[26] Nikolaos Leontiou et al. "A hierarchical control framework of load balancing and resource allocation of cloud computing services". In: *Computers & Electrical Engineering* 67 (2018), pp. 235–251.

[27] Redowan Mahmud, Ramamohanarao Kotagiri, and Rajkumar Buyya. "Fog computing: A taxonomy, survey and future directions". In: *Internet of everything*. Springer, 2018, pp. 103–130.

[28] Peter Mell, Tim Grance, et al. "The NIST definition of cloud computing". In: (2011).

[29] François Mériaux et al. "Achievability of efficient satisfaction equilibria in self-configuring networks". In: *International Conference on Game Theory for Networks*. Springer. 2012, pp. 1–15.

[30] Bilal Muhsin, Sujin Hwang, and Benjamin C Triman. *Physiological monitor with mobile computing device connectivity*. US Patent 9,877,650. Jan. 2018.

[31] Kumpati S Narendra and Mandayam AL Thathachar. *Learning automata: an introduction*. Courier Corporation, 2012.

[32] John Nash. "Non-cooperative games". In: *Annals of mathematics* (1951), pp. 286–295.

[33] José Leal D Neto et al. "ULOOF: A user level online offloading framework for mobile edge computing". In: *IEEE Transactions on Mobile Computing* 17.11 (2018), pp. 2660–2674.

[34] Mohammad S Obaidat, Georgios I Papadimitriou, and Andreas S Pomportsis. "Guest editorial learning automata: theory, paradigms, and applications". In: *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)* 32.6 (2002), pp. 706–709.

[35] Martin J Osborne and Ariel Rubinstein. *A course in game theory*. 1994.

[36] Charalampos Papadopoulos. *Cloud and Mobile Edge Computing Platforms for 5G Systems*. 2018.

[37] Milan Patel et al. "Mobile-edge computing introductory technical white paper". In: *White paper, mobile-edge computing (MEC) industry initiative* (2014), pp. 1089–7801.

[38] Robert A Powers. "Batteries for low power electronics". In: *Proceedings of the IEEE* 83.4 (1995), pp. 687–693.

[39] Shermila Ranadheera, Setareh Maghsudi, and Ekram Hossain. "Minority games with applications to distributed decision making and control in wireless networks". In: *IEEE Wireless Communications* 24.5 (2017), pp. 184–192.

[40] Cong Shi et al. "Cosmos: computation offloading as a service for mobile devices". In: *Proceedings of the 15th ACM international symposium on Mobile ad hoc networking and computing*. ACM. 2014, pp. 287–296.

[41] Weisong Shi et al. "Edge computing: Vision and challenges". In: *IEEE Internet of Things Journal* 3.5 (2016), pp. 637–646.

[42] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. 2018.

[43] Eirini Eleni Tsiropoulou, Panagiotis Vamvakas, and Symeon Papavassiliou. "Joint utility-based uplink power and rate allocation in wireless networks: a non-cooperative game theoretic framework". In: *Physical Communication* 9 (2013), pp. 299–307.

[44] Eirini Eleni Tsiropoulou et al. "Combined power and rate allocation in self-optimized multi-service two-tier femtocell networks". In: *Computer Communications* 72 (2015), pp. 38–48.

[45] Eirini Eleni Tsiropoulou et al. "On the problem of optimal cell selection and uplink power control in open access multi-service two-tier femtocell networks". In: *International Conference on Ad-Hoc Networks and Wireless*. Springer. 2014, pp. 114–127.

[46] Tim Verbelen et al. "AIOLOS: Middleware for improving mobile application performance through cyber foraging". In: *Journal of Systems and Software* 85.11 (2012), pp. 2629–2639.

[47] Kun Xie et al. "Distributed multi-dimensional pricing for efficient application offloading in mobile cloud computing". In: *IEEE Transactions on Services Computing* (2016).

[48] Yuhua Xu, Jinlong Wang, and Qihui Wu. "Distributed learning of equilibria with incomplete, dynamic, and uncertain information in wireless communication networks". In: *Game Theory Framework Applied to Wireless Communication Networks*. IGI Global, 2016, pp. 63–86.

[49] Ke Zhang et al. "Energy-efficient offloading for mobile edge computing in 5G heterogeneous networks". In: *IEEE access* 4 (2016), pp. 5896–5907.

[50] Pengtao Zhao et al. "Information prediction and dynamic programming-based RAN slicing for mobile edge computing". In: *IEEE Wireless Communications Letters* 7.4 (2018), pp. 614–617.

[51] Jianchao Zheng et al. "Dynamic computation offloading for mobile cloud computing: A stochastic game-theoretic approach". In: *IEEE Transactions on Mobile Computing* 18.4 (2018), pp. 771–786.

[52] Yuchen Zhou et al. "Resource allocation for information-centric virtualized heterogeneous networks with in-network caching and mobile edge computing". In: *IEEE Transactions on Vehicular Technology* 66.12 (2017), pp. 11339–11351.

Γλωσσάριο

Ελληνικός όρος	Αγγλικός όρος
υπολογιστική μεταφόρτωση	computational offloading
υπολογιστικό νέφος	cloud computing
κινητό υπολογιστικό νέφος	mobile cloud computing
κινητή υπολογιστική στα άκρα του δικτύου	mobile edge computing
σταθμοί βάσης	base station
σημείο πρόσβασης	access point
ενότητα	module
σύμπλεγμα	cluster
ισορροπία Νας	Nash Equilibrium
γενικευμένη ισορροπία Νας	generalized Nash equilibrium
ικανοποίηση ποιότητας υπηρεσίας	QoS satisfaction
ικανοποίηση ποιότητας εμπειρίας	QoE satisfaction
ισορροπία ικανοποίησης	satisfaction equilibrium
αποδοτική ισορροπία ικανοποίησης	efficient satisfaction equilibrium
γενικευμένη ισορροπία ικανοποίησης	generalized satisfaction equilibrium
συναρτήσεις ευχαρίστησης	payoff function
ενισχυτική μάθηση	reinforcement learning
πράκτορες μάθησης	learning agent
αυτόματα μάθησης	learning automata
απόκριση ικανοποίησης	satisfaction response
κλιμακωσιμότητα	scalability
ρυθμός ανοδικής ζεύξης υπηρεσίας	service uplink rate