ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ

ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

ΕΥΦΥΗΣ ΔΙΑΧΕΙΡΙΣΗ ΚΥΚΛΟΦΟΡΙΑΣ ΣΕ ΔΙΑΣΤΑΥΡΩΣΗ ΜΕ ΧΡΗΣΗ ΑΣΑΦΩΝ ΣΥΣΤΗΜΑΤΩΝ

ΙΩΑΝΝΗΣ ΚΟΝΤΟΠΟΥΛΟΣ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

ΥΠΕΥΘΥΝΟΣ

Κωνσταντίνος Κολομβάτσος
Αναπληρωτής καθηγητής

Λαμία ..... έτος 2026

ΠΑΝΕΠΙΣΤΗΜΙΟ
ΘΕΣΣΑΛΙΑΣ

ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

ΕΥΦΥΗΣ ΔΙΑΧΕΙΡΙΣΗ ΚΥΚΛΟΦΟΡΙΑΣ ΣΕ ΔΙΑΣΤΑΥΡΩΣΗ ΜΕ ΧΡΗΣΗ ΑΣΑΦΩΝ ΣΥΣΤΗΜΑΤΩΝ

ΙΩΑΝΝΗΣ ΚΟΝΤΟΠΟΥΛΟΣ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

ΥΠΕΥΘΥΝΟΣ

Κωνσταντίνος Κολομβάτσος
Αναπληρωτής καθηγητής

Λαμία ..... έτος 2026

UNIVERSITY OF
THESSALY

SCHOOL OF SCIENCE

DEPARTMENT OF COMPUTER SCIENCE & TELECOMMUNICATIONS

INTELLIGENT TRAFFIC MANAGEMENT AT AN INTERSECTION USING FUZZY SYSTEMS

IOANNIS KONTOPOULOS

FINAL THESIS

ADVISOR

Konstantinos Kolomvatsos
Associate Professor

Lamia ..... year 2026

«Με ατομική μου ευθύνη και γνωρίζοντας τις κυρώσεις [(1)], που προβλέπονται από της διατάξεις της παρ. 6 του άρθρου 22 του Ν. 1599/1986, δηλώνω ότι:

*1. Δεν παραθέτω κομμάτια βιβλίων ή άρθρων ή εργασιών άλλων αυτολεξεί **χωρίς να τα περικλείω σε εισαγωγικά** και χωρίς να αναφέρω το συγγραφέα, τη χρονολογία, τη σελίδα. Η αυτολεξεί παράθεση χωρίς εισαγωγικά χωρίς αναφορά στην πηγή, είναι λογοκλοπή. Πέραν της αυτολεξεί παράθεσης, λογοκλοπή θεωρείται και η παράφραση εδαφίων από έργα άλλων, συμπεριλαμβανομένων και έργων συμφοιτητών μου, καθώς και η παράθεση στοιχείων που άλλοι συνέλεξαν ή επεξεργάσθηκαν, χωρίς αναφορά στην πηγή. Αναφέρω πάντοτε με πληρότητα την πηγή κάτω από τον πίνακα ή σχέδιο, όπως στα παραθέματα.*

*2. Δέχομαι ότι η αυτολεξεί **παράθεση χωρίς εισαγωγικά**, ακόμα κι αν συνοδεύεται από αναφορά στην πηγή σε κάποιο άλλο σημείο του κειμένου ή στο τέλος του, είναι αντιγραφή. Η αναφορά στην πηγή στο τέλος π.χ. μιας παραγράφου ή μιας σελίδας, δεν δικαιολογεί συρραφή εδαφίων έργου άλλου συγγραφέα, έστω και παραφρασμένων, και παρουσίασή τους ως δική μου εργασία.*

3. Δέχομαι ότι υπάρχει επίσης περιορισμός στο μέγεθος και στη συχνότητα των παραθεμάτων που μπορώ να εντάξω στην εργασία μου εντός εισαγωγικών. Κάθε μεγάλο παράθεμα (π.χ. σε πίνακα ή πλαίσιο, κλπ), προϋποθέτει ειδικές ρυθμίσεις, και όταν δημοσιεύεται προϋποθέτει την άδεια του συγγραφέα ή του εκδότη. Το ίδιο και οι πίνακες και τα σχέδια

4. Δέχομαι όλες τις συνέπειες σε περίπτωση λογοκλοπής ή αντιγραφής.

Ημερομηνία: ……/…../20……

Ο – Η Δηλ.

(1) «Όποιος εν γνώσει του δηλώνει ψευδή γεγονότα ή αρνείται ή αποκρύπτει τα αληθινά με έγγραφη υπεύθυνη δήλωση
του άρθρου 8 παρ. 4 Ν. 1599/1986 τιμωρείται με φυλάκιση τουλάχιστον τριών μηνών. Εάν ο υπαίτιος αυτών των πράξεων
σκόπευε να προσπορίσει στον εαυτόν του ή σε άλλον περιουσιακό όφελος βλάπτοντας τρίτον ή σκόπευε να βλάψει άλλον, τιμωρείται με κάθειρξη μέχρι 10 ετών.»

ΠΕΡΙΛΗΨΗ

Σκοπός της παρούσας εργασίας είναι η μελέτη και η υλοποίηση ενός ευφυούς συστήματος διαχείρισης κυκλοφορίας σε αστική διασταύρωση, με χρήση ασαφών συστημάτων (fuzzy systems) και τεχνικών παλινδρόμησης, με στόχο τη βελτιστοποίηση της ροής οχημάτων και τη μείωση της συμφόρησης.

Για την αξιολόγηση της προτεινόμενης προσέγγισης, αναπτύχθηκε ένα περιβάλλον προσομοίωσης σηματοδοτούμενων διασταυρώσεων, στο οποίο το ευφυές σύστημα συγκρίθηκε με απλούστερες στρατηγικές ελέγχου, όπως ο έλεγχος σταθερού χρόνου και ο έλεγχος βασισμένος στο μήκος των ουρών. Η απόδοση των συστημάτων αξιολογήθηκε με βάση μετρικές όπως η συνολική διέλευση οχημάτων, το μήκος των ουρών και ο χρόνος αναμονής.

Τα πειραματικά αποτελέσματα δείχνουν ότι το προτεινόμενο ευφυές σύστημα παρουσιάζει βελτιωμένη συμπεριφορά σε σχέση με τα συστήματα αναφοράς, επιτυγχάνοντας αποδοτικότερη διαχείριση της κυκλοφορίας και πιο ομαλή ροή οχημάτων, ιδιαίτερα σε συνθήκες αυξημένου κυκλοφοριακού φόρτου.

ABSTRACT

The objective of this thesis is to study and implement an intelligent traffic management system for an urban intersection, based on fuzzy systems and regression techniques, aiming to optimize vehicle flow and reduce traffic congestion.

To evaluate the proposed approach, a simulation environment of signalized urban intersections was developed, in which the intelligent system was compared with simpler control strategies, such as fixed-time control and queue-length-based control. The performance of the examined systems was assessed using metrics including total vehicle throughput, queue length, and vehicle waiting time.

The experimental results indicate that the proposed intelligent system exhibits improved performance compared to the baseline approaches, achieving more efficient traffic management and smoother vehicle flow, particularly under high traffic demand conditions.

Table of Contents

ΠΕΡΙΛΗΨΗ I
ABSTRACT III

ΚΕΦΑΛΑΙΟ 1 ΕΙΣΑΓΩΓΗ 2

ΚΕΦΑΛΑΙΟ 2 ΥΠΟΛΟΓΙΣΤΙΚΗ ΝΟΗΜΟΣΥΝΗ ΚΑΙ ΔΙΑΧΕΙΡΙΣΗ ΑΣΑΦΕΙΑΣ 3

2.1 Αβεβαιοτητα και Ασαφεια στα Πραγματικα Συστηματα 4
2.2 Ασαφη Συνολα και Ασαφη Λογικη 5
2.3 Δομη ενος Ασαφους Συστηματος Ελεγχου 6
2.4 Ασαφη Συστηματα στη Διαχειριση Κυκλοφοριας 7
2.5 Συνδυασμος Ασαφων Συστηματων με Τεχνικες Παλινδρομησης 8

ΚΕΦΑΛΑΙΟ 3 ΔΙΑΧΕΙΡΙΣΗ ΟΔΙΚΗΣ ΚΥΚΛΟΦΟΡΙΑΣ 9

3.1 Μεθοδοι Ελεγχου Φωτεινων Σηματοδοτων 10
3.2 Ασαφη Συστηματα στον Έλεγχο Κυκλοφοριας 11
3.3 Συγχρονες Ευφυεις Προσεγγισεις 12
3.4 Εργαλεια και Λογισμικα Προσομοιωσης Κυκλοφοριας 13
3.5 Συνοψη 14

ΚΕΦΑΛΑΙΟ 4 ΠΡΟΤΕΙΝΟΜΕΝΟ ΣΥΣΤΗΜΑ 15

4.1 Περιγραφη του Συστηματος 15
4.2 Δομη και Ροη του Κωδικα 16
4.3 Στοχαστικο Μοντελο Κυκλοφοριας 17
4.4 Ευφυες Συστημα Ελεγχου (Fuzzy logic & Regression) 19
4.5 Υλοποιηση Baseline Συστηματα Ελεγχου 21
4.6 Μηχανισμος Συγκρισης και Συλλογη Μετρικων 22
4.7 Συνοψη 23

ΚΕΦΑΛΑΙΟ 5 ΠΕΙΡΑΜΑΤΙΚΗ ΑΠΟΤΙΜΗΣΗ 24

5.1 Πειραματικη Διαδικασια 24
5.2 Ρυθμισεις και Παραμετροι Πειραματων 24
5.3 Μετρικες Αξιολογησης 25
5.4 Αποτελεσματα Πειραματων 25
5.5 Συζητηση Αποτελεσματων 28
5.6 Συνοψη 28

ΚΕΦΑΛΑΙΟ 6 ΣΥΜΠΕΡΑΣΜΑΤΑ ΚΑΙ ΜΕΛΛΟΝΤΙΚΕΣ ΠΡΟΕΚΤΑΣΕΙΣ 29

6.1 ΣΥΜΠΕΡΑΣΜΑΤΑ **29**
6.2 ΠΕΡΙΟΡΙΣΜΟΥ ΤΗΣ ΠΡΟΣΕΓΓΙΣΗΣ **29**
6.3 ΜΕΛΛΟΝΤΙΚΕΣ ΠΡΟΕΚΤΑΣΕΙΣ **30**

APPENDIX **31**

ΒΙΒΛΙΟΓΡΑΦΙΑ **56**

ΚΕΦΑΛΑΙΟ 1 Εισαγωγή

Η διαχείριση της οδικής κυκλοφορίας αποτελεί ένα από τα σημαντικότερα προβλήματα των σύγχρονων αστικών κέντρων, καθώς η συνεχής αύξηση του αριθμού των οχημάτων οδηγεί σε κυκλοφοριακή συμφόρηση, αυξημένους χρόνους αναμονής, κατανάλωση ενέργειας και περιβαλλοντική επιβάρυνση. Οι παραδοσιακές μέθοδοι ελέγχου φωτεινών σηματοδοτών, όπως τα σταθερά χρονοπρογράμματα , αδυνατούν να προσαρμοστούν δυναμικά στις μεταβαλλόμενες κυκλοφοριακές συνθήκες.

Τα τελευταία χρόνια, η χρήση τεχνικών υπολογιστικής νοημοσύνης έχει προσελκύσει έντονο ερευνητικό ενδιαφέρον στον τομέα της ευφυούς διαχείρισης κυκλοφορίας. Ιδιαίτερα, τα ασαφή συστήματα (fuzzy systems) προσφέρουν τη δυνατότητα μοντελοποίησης της ανθρώπινης λογικής και της αβεβαιότητας επιτρέποντας τη λήψη αποφάσεων με βάση ασαφή και μη γραμμικά δεδομένα, όπως τα μήκη ουρών οχημάτων.

Σκοπός της παρούσας εργασίας είναι η ανάπτυξη και αξιολόγηση ενός ευφυούς συστήματος ελέγχου φωτεινών σηματοδοτών σε διασταύρωση, το οποίο βασίζεται σε συνδυασμό ασαφούς λογικής και γραμμικής παλινδρόμησης. Το προτεινόμενο σύστημα προσαρμόζει δυναμικά τον χρόνο πρασίνου δηματοδότη ανάλογα με την κυκλοφοριακή κατάσταση, με στόχο τη βελτίωση της συνολικής ροής των οχημάτων.

Για την αξιολόγηση της αποτελεσματικότητας του προτεινόμενου συστήματος, υλοποιούνται και συγκρίνονται τρεις διαφορετικές προσεγγίσεις ελέγχου:

- Ένα υβριδικό σύστημα fuzzy logic και regression
- Ένα σύστημα ελέγχου με σταθερό χρόνο πράσινου σηματοδότη
- Ένα απλό ουρο-ευαίσθητο σύστημα χωρίς χρήση τεχνικών υπολογιστικής νοημοσύνης

Η σύγκριση ππραγματοποιείται υπό ίδιες συνθήκες εισόδου, με την χρήση προσομοίωσης, και βασίζεται σε μετρικές όπως ο συνολικός αριθμός διελεύσεων (throughput), το συνολικό μήκος ουρών και η κατανομή του χρόνου πράσινου σηματοδότη. Τα αποτελέσματα καταδεικνύουν την αποδοτικότητα της προτεινόμενης προσέγγισης σε σχέση με τις παραδοσιακές μεθόδους.

ΚΕΦΑΛΑΙΟ 2 Υπολογιστική Νοημοσύνη και Διαχείριση Ασάφειας

Υπολογιστική Νοημοσύνη και Διαχείριση Ασάφειας

Η Υπολογιστική Νοημοσύνη (Computational Intelligence – CI) αποτελεί έναν σύγχρονο και δυναμικά εξελισσόμενο κλάδο της τεχνητής νοημοσύνης, ο οποίος εστιάζει στην επίλυση σύνθετων προβλημάτων μέσω προσεγγίσεων εμπνευσμένων από φυσικές διεργασίες και τον τρόπο με τον οποίο ο άνθρωπος αντιλαμβάνεται και λαμβάνει αποφάσεις. Σε αντίθεση με τις κλασικές αλγοριθμικές μεθόδους, οι οποίες βασίζονται σε αυστηρά μαθηματικά μοντέλα και προϋποθέτουν πλήρη και ακριβή γνώση του προβλήματος, οι τεχνικές υπολογιστικής νοημοσύνης είναι σχεδιασμένες ώστε να λειτουργούν αποδοτικά ακόμη και όταν τα διαθέσιμα δεδομένα είναι ελλιπή, αβέβαια ή ασαφή.

Ένα βασικό χαρακτηριστικό της υπολογιστικής νοημοσύνης είναι η ικανότητά της να προσαρμόζεται σε μεταβαλλόμενες συνθήκες και να προσεγγίζει τη λύση προβλημάτων με μη γραμμική συμπεριφορά, χωρίς την ανάγκη ρητού μοντελοποιημένου κανόνα. Η ιδιότητα αυτή καθιστά τις μεθόδους της ιδιαίτερα κατάλληλες για πραγματικά συστήματα, όπου η πολυπλοκότητα και η αβεβαιότητα αποτελούν εγγενή χαρακτηριστικά.

Οι βασικοί πυλώνες της υπολογιστικής νοημοσύνης περιλαμβάνουν τα τεχνητά νευρωνικά δίκτυα, τους εξελικτικούς αλγορίθμους και τα ασαφή συστήματα. Τα τεχνητά νευρωνικά δίκτυα μιμούνται τη λειτουργία του ανθρώπινου εγκεφάλου και χρησιμοποιούνται κυρίως για μάθηση και πρόβλεψη, οι εξελικτικοί αλγόριθμοι βασίζονται στις αρχές της φυσικής επιλογής για τη βελτιστοποίηση πολύπλοκων προβλημάτων, ενώ τα ασαφή συστήματα επιτρέπουν τη διαχείριση της αβεβαιότητας μέσω λεκτικών κανόνων και βαθμών συμμετοχής.

Οι τεχνικές αυτές έχουν εφαρμοστεί με επιτυχία σε πληθώρα επιστημονικών και τεχνολογικών τομέων, όπως η ρομποτική, η βιομηχανική αυτοματοποίηση, η ανάλυση και εξόρυξη δεδομένων, καθώς και τα συστήματα υποστήριξης αποφάσεων. Ιδιαίτερο ενδιαφέρον παρουσιάζουν οι εφαρμογές τους στη διαχείριση οδικής κυκλοφορίας, όπου η αβεβαιότητα, η δυναμική μεταβολή των συνθηκών και η ανάγκη για προσαρμοστικό έλεγχο καθιστούν τις μεθόδους υπολογιστικής νοημοσύνης ιδιαίτερα αποτελεσματικές.

2.1 Αβεβαιότητα και Ασάφεια στα Πραγματικά Συστήματα

Πολλά πραγματικά προβλήματα του σύγχρονου κόσμου δεν μπορούν να περιγραφούν με ακρίβεια μέσω αυστηρών και ντετερμινιστικών μαθηματικών μοντέλων. Η οδική κυκλοφορία αποτελεί ένα χαρακτηριστικό παράδειγμα τέτοιου προβλήματος, καθώς επηρεάζεται από πλήθος παραγόντων που δεν είναι εύκολο να ποσοτικοποιηθούν με ακρίβεια. Παράγοντες όπως η ανθρώπινη συμπεριφορά των οδηγών, οι καιρικές συνθήκες, τα απρόβλεπτα γεγονότα, καθώς και οι χρονικές μεταβολές της κυκλοφοριακής ζήτησης, συνθέτουν ένα δυναμικό και πολύπλοκο περιβάλλον.

Επιπλέον, τα δεδομένα που συλλέγονται από αισθητήρες ή συστήματα παρακολούθησης κυκλοφορίας συχνά παρουσιάζουν ελλείψεις, θόρυβο ή ανακρίβειες. Ως αποτέλεσμα, η λήψη αποφάσεων με βάση αποκλειστικά ακριβή αριθμητικά δεδομένα καθίσταται δυσχερής. Σε τέτοιες περιπτώσεις, η ανάγκη για μεθόδους που μπορούν να διαχειριστούν την αβεβαιότητα και την ανακρίβεια των πληροφοριών είναι ιδιαίτερα σημαντική.

Η έννοια της ασάφειας (fuzziness) διαφέρει ουσιαστικά από την έννοια της τυχαιότητας. Η τυχαιότητα σχετίζεται με στοχαστικά φαινόμενα, όπου τα αποτελέσματα περιγράφονται με πιθανότητες, ενώ η ασάφεια αναφέρεται στην αδυναμία ακριβούς και σαφούς ορισμού εννοιών. Για παράδειγμα, όροι όπως «χαμηλή κυκλοφορία», «μέτρια συμφόρηση» ή «υψηλή καθυστέρηση» δεν μπορούν να οριστούν με απόλυτα όρια, αλλά εξαρτώνται από το πλαίσιο και την ανθρώπινη αντίληψη.

Για τη διαχείριση τέτοιων καταστάσεων, τα ασαφή συστήματα προσφέρουν ένα ιδιαίτερα αποτελεσματικό και ευέλικτο πλαίσιο. Μέσω της χρήσης ασαφών συνόλων και λεκτικών κανόνων, είναι δυνατή η ενσωμάτωση της ανθρώπινης εμπειρίας και κρίσης στη διαδικασία λήψης αποφάσεων. Η προσέγγιση αυτή επιτρέπει την περιγραφή πολύπλοκων καταστάσεων με τρόπο πιο φυσικό και κατανοητό, καθιστώντας τα ασαφή συστήματα ιδανικά για εφαρμογές σε περιβάλλοντα με υψηλό βαθμό αβεβαιότητας, όπως η διαχείριση της οδικής κυκλοφορίας.

2.2 Ασαφή Σύνολα και Ασαφή Λογική

Η θεωρία των ασαφών συνόλων (Fuzzy Sets), η οποία εισήχθη από τον Lotfi Zadeh, επεκτείνει την κλασική θεωρία συνόλων επιτρέποντας στα στοιχεία να ανήκουν σε ένα σύνολο με βαθμό συμμετοχής που λαμβάνει τιμές στο διάστημα [0,1]. Σε αντίθεση με τα κλασικά σύνολα, όπου η συμμετοχή ενός στοιχείου είναι δυαδική (είτε ανήκει είτε όχι), τα ασαφή σύνολα επιτρέπουν τη μερική συμμετοχή, παρέχοντας έτσι ένα πιο ευέλικτο και ρεαλιστικό μαθηματικό εργαλείο.
Μέσω αυτής της προσέγγισης, έννοιες που δεν διαθέτουν σαφή και αυστηρά όρια μπορούν να αναπαρασταθούν μαθηματικά. Για παράδειγμα, η έννοια της «υψηλής κυκλοφορίας» δεν μπορεί να οριστεί με ένα μοναδικό αριθμητικό κατώφλι, αλλά περιγράφεται πιο ρεαλιστικά μέσω μιας συνάρτησης συμμετοχής που αποδίδει βαθμό αλήθειας σε διαφορετικά επίπεδα φόρτου. Η ιδιότητα αυτή καθιστά τα ασαφή σύνολα ιδιαίτερα κατάλληλα για την αναπαράσταση πραγματικών καταστάσεων που χαρακτηρίζονται από αβεβαιότητα και υποκειμενικότητα.
Η ασαφής λογική (Fuzzy Logic) βασίζεται στη θεωρία των ασαφών συνόλων και επιτρέπει τη λήψη αποφάσεων με τρόπο παρόμοιο με την ανθρώπινη σκέψη και κρίση. Αντί για αυστηρούς δυαδικούς κανόνες της κλασικής λογικής, όπου οι συνθήκες αξιολογούνται αποκλειστικά ως αληθείς ή ψευδείς, η ασαφής λογική χρησιμοποιεί λεκτικούς κανόνες της μορφής:
Αν η κυκλοφορία είναι υψηλή, τότε ο χρόνος πράσινου είναι μεγάλος.
Οι κανόνες αυτοί επιτρέπουν την ενσωμάτωση της ανθρώπινης εμπειρίας στη διαδικασία λήψης αποφάσεων και καθιστούν τα ασαφή συστήματα ιδιαίτερα κατανοητά και διαφανή. Επιπλέον, η δομή τους διευκολύνει την τροποποίηση και προσαρμογή τους σε νέες συνθήκες, ακόμη και από μη ειδικούς, γεγονός που αποτελεί σημαντικό πλεονέκτημα σε εφαρμογές πραγματικού χρόνου, όπως η διαχείριση της οδικής κυκλοφορίας.

2.3 Δομή ενός Ασαφούς Συστήματος Ελέγχου

Ένα τυπικό ασαφές σύστημα ελέγχου αποτελείται από μια ακολουθία διακριτών σταδίων, τα οποία συνεργάζονται για τη μετατροπή ασαφών και αβέβαιων εισόδων σε μια σαφή και αξιοποιήσιμη απόφαση. Τα βασικά δομικά στοιχεία ενός τέτοιου συστήματος περιγράφονται στη συνέχεια.

1. Fuzzification (Ασαφοποίηση):
Κατά το στάδιο της ασαφοποίησης, οι ακριβείς αριθμητικές είσοδοι του συστήματος μετατρέπονται σε ασαφείς τιμές μέσω κατάλληλων συναρτήσεων συμμετοχής. Για παράδειγμα, μια αριθμητική τιμή που εκφράζει τον κυκλοφοριακό φόρτο μπορεί να αντιστοιχιστεί με διαφορετικούς βαθμούς συμμετοχής σε λεκτικές κατηγορίες όπως «χαμηλή», «μέτρια» ή «υψηλή». Η διαδικασία αυτή επιτρέπει στο σύστημα να χειρίζεται τα δεδομένα με τρόπο πιο κοντά στην ανθρώπινη αντίληψη.

2. Inference Mechanism (Μηχανισμός Συμπερασμού):
Στο στάδιο του μηχανισμού συμπερασμού εφαρμόζονται οι ασαφείς κανόνες τύπου IF–THEN, οι οποίοι έχουν οριστεί με βάση την εμπειρία ή τη γνώση του προβλήματος. Οι κανόνες αυτοί συνδυάζουν τις ασαφείς εισόδους και παράγουν ασαφή συμπεράσματα σχετικά με τη συμπεριφορά του συστήματος. Ο μηχανισμός συμπερασμού αποτελεί τον πυρήνα της διαδικασίας λήψης αποφάσεων του ασαφούς συστήματος.

3. Aggregation (Συνάθροιση):
Κατά το στάδιο της συνάθροισης, τα αποτελέσματα όλων των ενεργοποιημένων κανόνων συνδυάζονται σε ένα ενιαίο ασαφές σύνολο εξόδου. Η διαδικασία αυτή επιτρέπει τη συνολική αξιολόγηση της κατάστασης, λαμβάνοντας υπόψη τη συνεισφορά κάθε κανόνα, και διασφαλίζει ότι η τελική απόφαση βασίζεται σε μια ολοκληρωμένη εικόνα των εισόδων.

4. Defuzzification (Αποασαφοποίηση):
Στο τελευταίο στάδιο, η ασαφής έξοδος μετατρέπεται σε μία τελική, αριθμητική τιμή, η οποία μπορεί να χρησιμοποιηθεί άμεσα από το σύστημα ελέγχου. Η αποασαφοποίηση πραγματοποιείται μέσω κατάλληλων μεθόδων, όπως ο υπολογισμός του κέντρου βάρους, και παρέχει μια σαφή απόφαση, για παράδειγμα τον χρόνο πράσινου φωτός που θα εφαρμοστεί σε μια σηματοδοτούμενη διασταύρωση.

Η διαδοχική αυτή διαδικασία καθιστά τα ασαφή συστήματα ελέγχου ιδιαίτερα κατάλληλα για εφαρμογές όπου απαιτείται προσαρμοστικότητα και διαχείριση αβεβαιότητας, όπως η δυναμική διαχείριση της οδικής κυκλοφορίας.

2.4 Ασαφή Συστήματα στη Διαχείριση Κυκλοφορίας

Στον τομέα της διαχείρισης οδικής κυκλοφορίας, τα ασαφή συστήματα έχουν χρησιμοποιηθεί εκτενώς για τον έλεγχο φωτεινών σηματοδοτών, καθώς προσφέρουν τη δυνατότητα δυναμικής προσαρμογής στις εκάστοτε κυκλοφοριακές συνθήκες. Σε αντίθεση με τα παραδοσιακά συστήματα σταθερού χρονοπρογραμματισμού, τα οποία λειτουργούν με προκαθορισμένους κύκλους ανεξάρτητα από τον πραγματικό φόρτο, τα fuzzy συστήματα λαμβάνουν αποφάσεις βασισμένες στην τρέχουσα κατάσταση της κυκλοφορίας.
Συγκεκριμένα, τα ασαφή συστήματα ελέγχου μπορούν να ενσωματώσουν παραμέτρους όπως το μήκος των ουρών, τον χρόνο αναμονής των οχημάτων και τη σχετική συμφόρηση ανά κατεύθυνση, προσαρμόζοντας αντίστοιχα τον χρόνο πράσινου φωτός. Η χρήση λεκτικών μεταβλητών, όπως «χαμηλή», «μέτρια» ή «υψηλή συμφόρηση», επιτρέπει στο σύστημα να προσεγγίζει τη λήψη αποφάσεων με τρόπο παρόμοιο με εκείνον ενός έμπειρου ανθρώπινου χειριστή.
Πλήθος προηγούμενων μελετών στη διεθνή βιβλιογραφία έχουν δείξει ότι η εφαρμογή της ασαφούς λογικής στον έλεγχο φωτεινών σηματοδοτών μπορεί να οδηγήσει σε σημαντική μείωση της συμφόρησης, βελτίωση της συνολικής ροής των οχημάτων και πιο ομαλή λειτουργία των διασταυρώσεων, ιδιαίτερα σε συνθήκες μεταβαλλόμενης ζήτησης. Επιπλέον, τα fuzzy συστήματα εμφανίζουν αυξημένη ανθεκτικότητα σε αβέβαια ή θορυβώδη δεδομένα, γεγονός που τα καθιστά κατάλληλα για πραγματικές αστικές εφαρμογές.
Η ευελιξία, η διαφάνεια και η δυνατότητα ενσωμάτωσης εμπειρικής γνώσης αποτελούν βασικά πλεονεκτήματα των ασαφών συστημάτων. Τα χαρακτηριστικά αυτά καθιστούν τη συγκεκριμένη προσέγγιση ιδιαίτερα ελκυστική για συστήματα διαχείρισης κυκλοφορίας, όπου η προσαρμοστικότητα και η αξιοπιστία αποτελούν κρίσιμους παράγοντες για τη βελτίωση της καθημερινής λειτουργίας των αστικών οδικών δικτύων.

2.5 Συνδυασμός Ασαφών Συστημάτων με Τεχνικές Παλινδρόμησης

Παρότι τα ασαφή συστήματα ελέγχου είναι ιδιαίτερα ισχυρά ως προς τη διαχείριση της αβεβαιότητας και την ενσωμάτωση ανθρώπινης εμπειρίας, σε ορισμένες περιπτώσεις ενδέχεται να παρουσιάζουν περιορισμένη αριθμητική ακρίβεια. Για τον λόγο αυτό, ο συνδυασμός τους με τεχνικές παλινδρόμησης μπορεί να προσφέρει βελτιωμένη σταθερότητα και ακρίβεια στη διαδικασία λήψης αποφάσεων.

Οι μέθοδοι παλινδρόμησης βασίζονται στην ανάλυση ιστορικών δεδομένων και επιτρέπουν την εκτίμηση αριθμητικών μεγεθών μέσω μαθηματικών μοντέλων. Στο πλαίσιο της διαχείρισης κυκλοφορίας, η παλινδρόμηση μπορεί να χρησιμοποιηθεί για την πρόβλεψη μεγεθών όπως ο απαιτούμενος χρόνος πράσινου φωτός, με βάση παραμέτρους όπως το συνολικό μήκος των ουρών ή ο κυκλοφοριακός φόρτος. Η δυνατότητα αυτή συμπληρώνει αποτελεσματικά τη λογική των ασαφών συστημάτων.

Ένας υβριδικός συνδυασμός ασαφούς λογικής και παλινδρόμησης αξιοποιεί τα πλεονεκτήματα και των δύο προσεγγίσεων. Συγκεκριμένα, η ασαφής λογική προσφέρει ευελιξία, ερμηνευσιμότητα και προσαρμοστικότητα σε αβέβαιες συνθήκες, ενώ οι τεχνικές παλινδρόμησης συνεισφέρουν μαθηματική ακρίβεια και δυνατότητα γενίκευσης βάσει δεδομένων. Ο συνδυασμός αυτός επιτρέπει στο σύστημα να λαμβάνει αποφάσεις που είναι ταυτόχρονα κατανοητές και αριθμητικά συνεπείς.

Η υβριδική αυτή προσέγγιση αποτελεί μια σύγχρονη τάση στη σχεδίαση ευφυών συστημάτων ελέγχου, καθώς ανταποκρίνεται στις απαιτήσεις πολύπλοκων και δυναμικών εφαρμογών. Στην παρούσα εργασία, η φιλοσοφία αυτή υιοθετείται με στόχο τη βελτίωση της απόδοσης και της αξιοπιστίας του συστήματος διαχείρισης κυκλοφορίας, αποτελώντας τη θεωρητική βάση της προτεινόμενης υλοποίησης.

ΚΕΦΑΛΑΙΟ 3 Διαχείριση Οδικής Κυκλοφορίας

Διαχείριση Οδικής Κυκλοφορίας

Η διαχείριση οδικής κυκλοφορίας σε αστικά περιβάλλοντα αποτελεί ένα από τα σημαντικότερα και πιο σύνθετα προβλήματα των σύγχρονων μεταφορικών συστημάτων. Η συνεχής αύξηση του αριθμού των οχημάτων, σε συνδυασμό με τους περιορισμένους διαθέσιμους οδικούς πόρους, οδηγεί σε φαινόμενα συμφόρησης, αυξημένες καθυστερήσεις και σημαντική υποβάθμιση της ποιότητας ζωής στις πόλεις. Παράλληλα, η κυκλοφοριακή συμφόρηση συνδέεται άμεσα με αυξημένες εκπομπές ρύπων και κατανάλωση ενέργειας, επιβαρύνοντας τόσο το περιβάλλον όσο και τη δημόσια υγεία.
Ιδιαίτερη σημασία στο πλαίσιο της αστικής κυκλοφορίας έχουν οι σηματοδοτούμενες διασταυρώσεις, οι οποίες αποτελούν κρίσιμα σημεία ελέγχου και συχνά κύριες πηγές συμφόρησης στο οδικό δίκτυο. Η ανεπαρκής ρύθμιση των φωτεινών σηματοδοτών μπορεί να οδηγήσει σε μεγάλους χρόνους αναμονής, συσσώρευση οχημάτων και μη αποδοτική αξιοποίηση της διαθέσιμης οδικής υποδομής.
Για τον λόγο αυτό, η ανάπτυξη αποδοτικών μεθόδων ελέγχου φωτεινών σηματοδοτών αποτελεί βασικό αντικείμενο έρευνας στον τομέα των ευφυών μεταφορικών συστημάτων. Οι σύγχρονες προσεγγίσεις στοχεύουν στη δυναμική προσαρμογή των σηματοδοτικών κύκλων στις τρέχουσες κυκλοφοριακές συνθήκες, με σκοπό τη μείωση της συμφόρησης, τη βελτίωση της ροής των οχημάτων και την ομαλότερη λειτουργία των αστικών οδικών δικτύων [4].

3.1 Μέθοδοι Ελέγχου Φωτεινών Σηματοδοτών

Οι μέθοδοι ελέγχου φωτεινών σηματοδοτών μπορούν γενικά να ταξινομηθούν σε δύο βασικές κατηγορίες: τις μεθόδους σταθερού χρόνου και τις προσαρμοστικές μεθόδους. Η κατηγοριοποίηση αυτή βασίζεται στον τρόπο με τον οποίο καθορίζονται οι χρόνοι του σηματοδοτικού κύκλου και στον βαθμό προσαρμογής τους στις εκάστοτε κυκλοφοριακές συνθήκες.

Οι μέθοδοι σταθερού χρόνου βασίζονται σε προκαθορισμένα χρονοπρογράμματα, τα οποία υπολογίζονται εκ των προτέρων με βάση ιστορικά δεδομένα κυκλοφορίας. Οι χρόνοι πράσινου, κόκκινου και ο συνολικός χρόνος κύκλου παραμένουν σταθεροί κατά τη διάρκεια λειτουργίας του συστήματος και δεν επηρεάζονται από τις στιγμιαίες μεταβολές της κυκλοφορίας. Ένα από τα πλέον κλασικά και ευρέως αναφερόμενα έργα στον τομέα αυτό είναι η μελέτη του Webster, στην οποία παρουσιάζονται μαθηματικές σχέσεις για τον υπολογισμό βέλτιστων χρόνων κύκλου και πράσινης φάσης σε απομονωμένες σηματοδοτούμενες διασταυρώσεις [12].

Οι μέθοδοι αυτές χαρακτηρίζονται από απλότητα, ευκολία υλοποίησης και προβλέψιμη συμπεριφορά, γεγονός που τις καθιστά αξιόπιστες σε περιβάλλοντα με σχετικά σταθερά κυκλοφοριακά πρότυπα. Ωστόσο, παρουσιάζουν σημαντικούς περιορισμούς σε δυναμικά και μεταβαλλόμενα περιβάλλοντα, καθώς αδυνατούν να προσαρμοστούν σε αιφνίδιες μεταβολές της κυκλοφορίας, όπως αυτές που προκαλούνται από ατυχήματα, καιρικές συνθήκες ή ώρες αιχμής.

Ως αποτέλεσμα, η χρήση σταθερών χρονοπρογραμμάτων μπορεί να οδηγήσει σε μη αποδοτική κατανομή του χρόνου πράσινου φωτός, αυξημένους χρόνους αναμονής και συσσώρευση οχημάτων σε συγκεκριμένες κατευθύνσεις. Οι περιορισμοί αυτοί αποτέλεσαν βασικό κίνητρο για την ανάπτυξη προσαρμοστικών και ευφυών μεθόδων ελέγχου, οι οποίες επιδιώκουν τη δυναμική προσαρμογή της σηματοδότησης στις πραγματικές συνθήκες κυκλοφορίας.

3.2 Ασαφή Συστήματα στον Έλεγχο Κυκλοφορίας

Για την αντιμετώπιση των περιορισμών των μεθόδων σταθερού χρόνου, έχουν προταθεί κατά τις τελευταίες δεκαετίες διάφορες ευφυείς τεχνικές ελέγχου φωτεινών σηματοδοτών, με κυρίαρχη προσέγγιση τα ασαφή συστήματα (fuzzy systems). Οι τεχνικές αυτές στοχεύουν στη δυναμική προσαρμογή της σηματοδότησης με βάση την τρέχουσα κατάσταση της κυκλοφορίας, λαμβάνοντας υπόψη πληροφορίες που συχνά είναι αβέβαιες ή ασαφείς.
Τα fuzzy systems επιτρέπουν τη λήψη αποφάσεων με βάση λεκτικές έννοιες, όπως «χαμηλή», «μέτρια» ή «υψηλή» κυκλοφορία, οι οποίες αντικατοπτρίζουν τον τρόπο με τον οποίο ένας ανθρώπινος χειριστής θα αξιολογούσε την κυκλοφοριακή κατάσταση. Μέσω της χρήσης ασαφών κανόνων και συναρτήσεων συμμετοχής, τα συστήματα αυτά μπορούν να προσαρμόζουν τον χρόνο πράσινου φωτός με τρόπο ευέλικτο και κατανοητό, γεγονός που τα καθιστά ιδιαίτερα ελκυστικά για εφαρμογές πραγματικού χρόνου [2].
Στη διεθνή βιβλιογραφία έχει παρουσιαστεί μεγάλος αριθμός fuzzy logic controllers για τον έλεγχο σηματοδοτούμενων διασταυρώσεων. Τα συστήματα αυτά συνήθως χρησιμοποιούν ως εισόδους παραμέτρους όπως το μήκος της ουράς, τον χρόνο αναμονής και τη σχετική συμφόρηση μεταξύ διαφορετικών κατευθύνσεων, προσαρμόζοντας δυναμικά τον χρόνο πράσινου σηματοδότη [5], [13]. Η προσέγγιση αυτή επιτρέπει την καλύτερη εξισορρόπηση της κυκλοφορίας και την αποφυγή υπερβολικής προτεραιότητας σε συγκεκριμένες κατευθύνσεις.
Πλήθος μελετών έχουν δείξει ότι τα ασαφή συστήματα ελέγχου μπορούν να μειώσουν σημαντικά τους χρόνους αναμονής, να περιορίσουν τη συμφόρηση και να βελτιώσουν τη συνολική ροή των οχημάτων σε σύγκριση με τις παραδοσιακές μεθόδους σταθερού χρόνου [3]. Επιπλέον, η διαφάνεια και η ευκολία προσαρμογής των fuzzy controllers επιτρέπουν την ενσωμάτωση εμπειρικής γνώσης και τη συνεχή βελτίωσή τους, καθιστώντας τους μία από τις πλέον διαδεδομένες ευφυείς προσεγγίσεις στη διαχείριση οδικής κυκλοφορίας.

3.3 Σύγχρονες Ευφυείς Προσεγγίσεις

Τα τελευταία χρόνια, η ενισχυτική μάθηση (Reinforcement Learning – RL) και τα πολυ-πρακτορικά συστήματα έχουν προσελκύσει αυξημένο ερευνητικό ενδιαφέρον στον τομέα του ελέγχου οδικής κυκλοφορίας. Οι προσεγγίσεις αυτές βασίζονται στην εκμάθηση πολιτικών ελέγχου μέσω αλληλεπίδρασης με το περιβάλλον, επιτρέποντας στο σύστημα να βελτιστοποιεί τη συμπεριφορά του βάσει ανταμοιβών και ποινών που προκύπτουν από τις αποφάσεις του.
Στο πλαίσιο της διαχείρισης κυκλοφορίας, οι μέθοδοι ενισχυτικής μάθησης έχουν εφαρμοστεί τόσο σε μεμονωμένες σηματοδοτούμενες διασταυρώσεις όσο και σε μεγαλύτερα δίκτυα διασταυρώσεων, συχνά σε πολυ-πρακτορικά περιβάλλοντα όπου κάθε διασταύρωση λειτουργεί ως ανεξάρτητος πράκτορας [6], [7]. Η προσέγγιση αυτή επιτρέπει τον συντονισμένο έλεγχο πολλαπλών σημείων του οδικού δικτύου και μπορεί να οδηγήσει σε ιδιαίτερα αποδοτικές λύσεις σε πολύπλοκα και μεγάλης κλίμακας συστήματα.
Παρά τα πλεονεκτήματά τους, οι μέθοδοι ενισχυτικής μάθησης παρουσιάζουν και σημαντικές προκλήσεις. Η εκπαίδευση των μοντέλων απαιτεί συνήθως υψηλή υπολογιστική ισχύ, μεγάλο όγκο δεδομένων και εκτεταμένο χρόνο εκπαίδευσης, ενώ η συμπεριφορά των εκπαιδευμένων πολιτικών δεν είναι πάντοτε εύκολα ερμηνεύσιμη. Τα χαρακτηριστικά αυτά περιορίζουν την πρακτική τους εφαρμογή σε περιβάλλοντα όπου απαιτείται διαφάνεια, απλότητα και άμεση προσαρμογή.
Για τους λόγους αυτούς, τα ασαφή συστήματα εξακολουθούν να αποτελούν μια ιδιαίτερα ελκυστική λύση για τον έλεγχο φωτεινών σηματοδοτών, ειδικά σε εφαρμογές όπου η ερμηνευσιμότητα των αποφάσεων και η ευκολία ρύθμισης είναι κρίσιμοι παράγοντες. Η δυνατότητα συνδυασμού της ανθρώπινης εμπειρίας με απλές και κατανοητές δομές καθιστά τα fuzzy συστήματα μια πρακτική και αξιόπιστη εναλλακτική προσέγγιση έναντι πιο πολύπλοκων μεθόδων μάθησης.

3.4 Εργαλεία και Λογισμικά Προσομοίωσης Κυκλοφορίας

Η αξιολόγηση αλγορίθμων ελέγχου φωτεινών σηματοδοτών πραγματοποιείται συνήθως μέσω προσομοίωσης, καθώς η δοκιμή νέων μεθόδων σε πραγματικές αστικές συνθήκες είναι συχνά δύσκολη, δαπανηρή ή και πρακτικά αδύνατη. Τα περιβάλλοντα προσομοίωσης επιτρέπουν τη μελέτη διαφορετικών σεναρίων κυκλοφορίας υπό ελεγχόμενες συνθήκες, παρέχοντας τη δυνατότητα αντικειμενικής σύγκρισης εναλλακτικών στρατηγικών ελέγχου.

Ένα από τα πλέον διαδεδομένα εργαλεία στον τομέα αυτό είναι το **Eclipse SUMO** (Simulation of Urban MObility), το οποίο αποτελεί ανοιχτού κώδικα προσομοιωτή μικροσκοπικής κυκλοφορίας. Το SUMO επιτρέπει τη λεπτομερή μοντελοποίηση οχημάτων, διασταυρώσεων και φωτεινών σηματοδοτών, καθώς και την εισαγωγή προσαρμοστικών αλγορίθμων ελέγχου. Λόγω της ευελιξίας και της ακρίβειάς του, έχει χρησιμοποιηθεί εκτενώς στη διεθνή βιβλιογραφία για την ανάλυση και σύγκριση μεθόδων διαχείρισης κυκλοφορίας [9], [10].

Πέραν των κλασικών προσομοιωτών, σύγχρονα περιβάλλοντα όπως το **CityFlow** παρέχουν υποστήριξη για μεγάλης κλίμακας προσομοιώσεις και πολυ-πρακτορικά σενάρια. Τα εργαλεία αυτά έχουν σχεδιαστεί ώστε να διευκολύνουν τη μελέτη σύγχρονων ευφυών μεθόδων ελέγχου, όπως η ενισχυτική μάθηση, επιτρέποντας την ταυτόχρονη προσομοίωση μεγάλου αριθμού διασταυρώσεων και πρακτόρων [8], [11].

Η χρήση τέτοιων προσομοιωτικών εργαλείων συμβάλλει ουσιαστικά στην αντικειμενική αξιολόγηση της απόδοσης διαφορετικών αλγορίθμων, μέσω μετρικών όπως ο χρόνος αναμονής, το μήκος ουράς και η συνολική ροή οχημάτων. Επιπλέον, επιτρέπει την επαναληψιμότητα των πειραμάτων και τη σύγκριση των αποτελεσμάτων με προηγούμενες ερευνητικές εργασίες, καθιστώντας την προσομοίωση αναπόσπαστο μέρος της έρευνας στον τομέα της διαχείρισης οδικής κυκλοφορίας.

3.5 Σύνοψη

Στο παρόν κεφάλαιο παρουσιάστηκαν βασικές προσεγγίσεις και αντιπροσωπευτικές ερευνητικές εργασίες που σχετίζονται με τη διαχείριση οδικής κυκλοφορίας και τον έλεγχο φωτεινών σηματοδοτών σε αστικά περιβάλλοντα. Αρχικά, αναλύθηκαν οι κλασικές μέθοδοι σταθερού χρόνου, καθώς και οι περιορισμοί τους σε δυναμικές και μεταβαλλόμενες κυκλοφοριακές συνθήκες, γεγονός που ανέδειξε την ανάγκη για πιο προσαρμοστικές λύσεις.

Ιδιαίτερη έμφαση δόθηκε στις ευφυείς μεθόδους ελέγχου και ειδικότερα στα ασαφή συστήματα, τα οποία προσφέρουν αποτελεσματική διαχείριση της ασάφειας και τη δυνατότητα ενσωμάτωσης ανθρώπινης εμπειρίας στη διαδικασία λήψης αποφάσεων. Παρουσιάστηκαν επίσης σύγχρονες ερευνητικές τάσεις, όπως οι προσεγγίσεις ενισχυτικής μάθησης και τα πολυ-πρακτορικά συστήματα, καθώς και τα πλεονεκτήματα και οι περιορισμοί τους σε σχέση με την ερμηνευσιμότητα και την υπολογιστική πολυπλοκότητα.

Παράλληλα, έγινε αναφορά σε σύγχρονα εργαλεία και περιβάλλοντα προσομοίωσης που χρησιμοποιούνται ευρέως για την πειραματική αξιολόγηση αλγορίθμων ελέγχου κυκλοφορίας, αναδεικνύοντας τον καθοριστικό ρόλο της προσομοίωσης στην αντικειμενική σύγκριση διαφορετικών μεθόδων.

Στο επόμενο κεφάλαιο παρουσιάζεται το προτεινόμενο σύστημα ελέγχου και η υλοποίησή του, με έμφαση στη δομή, τη λειτουργία και τις βασικές παραμέτρους του, καθώς και η συγκριτική αξιολόγησή του σε σχέση με εναλλακτικές προσεγγίσεις διαχείρισης κυκλοφορίας.

ΚΕΦΑΛΑΙΟ 4 Προτεινόμενο Σύστημα

4.1 Περιγραφή του Συστήματος

Το προτεινόμενο σύστημα αφορά την προσομοίωση και τον έλεγχο της οδικής κυκλοφορίας σε ένα αστικό περιβάλλον τεσσάρων σηματοδοτούμενων διασταυρώσεων, οργανωμένων σε διάταξη πλέγματος 2×2. Η επιλογή της συγκεκριμένης διάταξης επιτρέπει τη μελέτη τόσο της τοπικής συμπεριφοράς κάθε διασταύρωσης όσο και της συνολικής λειτουργίας ενός μικρού δικτύου, το οποίο προσεγγίζει ρεαλιστικά συνθήκες αστικής κυκλοφορίας.
Κάθε διασταύρωση διαθέτει τέσσερις εισόδους, οι οποίες αντιστοιχούν στις κατευθύνσεις Βορράς, Νότος, Ανατολή και Δύση. Η κυκλοφορία σε κάθε διασταύρωση ελέγχεται μέσω φωτεινών σηματοδοτών δύο φάσεων, όπου η πρώτη φάση εξυπηρετεί τις κατευθύνσεις Βορράς–Νότος και η δεύτερη τις κατευθύνσεις Ανατολή–Δύση. Σε κάθε χρονική στιγμή, μόνο μία από τις δύο φάσεις είναι ενεργή, ενώ η εναλλαγή τους καθορίζεται από τον αλγόριθμο ελέγχου που εφαρμόζεται.
Η λειτουργία του συστήματος βασίζεται σε διακριτά χρονικά βήματα προσομοίωσης. Σε κάθε χρονικό βήμα εκτελούνται διαδοχικά συγκεκριμένα στάδια επεξεργασίας. Αρχικά, εισέρχονται νέα οχήματα στο σύστημα στις εισόδους των διασταυρώσεων, σύμφωνα με το στοχαστικό μοντέλο αφίξεων που περιγράφεται σε επόμενη ενότητα. Στη συνέχεια, ενημερώνονται τα μήκη των ουρών για κάθε κατεύθυνση, λαμβάνοντας υπόψη τόσο τις νέες αφίξεις όσο και τα οχήματα που εξυπηρετήθηκαν στο προηγούμενο βήμα.
Με βάση την τρέχουσα κατάσταση της κυκλοφορίας, το σύστημα ελέγχου υπολογίζει τον χρόνο πράσινου φωτός που θα αποδοθεί στην ενεργή φάση της διασταύρωσης. Ο υπολογισμός αυτός εξαρτάται από τη στρατηγική ελέγχου που εφαρμόζεται (ευφυές σύστημα, σύστημα σταθερού χρόνου ή σύστημα βασισμένο στο μήκος ουράς) και αποτελεί τη βασική έξοδο του συστήματος ελέγχου. Ακολούθως, πραγματοποιείται η εκφόρτωση των αντίστοιχων ουρών, δηλαδή η απομάκρυνση οχημάτων από τις κατευθύνσεις που διαθέτουν πράσινο φωτισμό, σύμφωνα με προκαθορισμένο ρυθμό εξυπηρέτησης.
Το σύστημα έχει σχεδιαστεί με τρόπο ώστε να επιτρέπει την αντικειμενική και δίκαιη σύγκριση διαφορετικών στρατηγικών ελέγχου. Για τον σκοπό αυτό, όλες οι στρατηγικές εφαρμόζονται υπό τις ίδιες συνθήκες εισόδου, χρησιμοποιώντας τα ίδια δεδομένα αφίξεων και τις ίδιες παραμέτρους προσομοίωσης. Με τον τρόπο αυτό διασφαλίζεται ότι οι διαφορές στην απόδοση οφείλονται αποκλειστικά στον αλγόριθμο ελέγχου και όχι σε εξωτερικούς παράγοντες.
Η συνολική αρχιτεκτονική του συστήματος επιτρέπει τη συλλογή βασικών μετρικών απόδοσης, όπως ο αριθμός εξυπηρετημένων οχημάτων, τα μήκη ουρών και οι χρόνοι πράσινου φωτός, οι οποίες χρησιμοποιούνται στη

συνέχεια για την πειραματική αποτίμηση και συγκριτική αξιολόγηση των εξεταζόμενων μεθόδων ελέγχου.

4.2 Δομή και Ροή του Κώδικα

Η υλοποίηση του προτεινόμενου συστήματος πραγματοποιήθηκε στη γλώσσα προγραμματισμού Python, η οποία επιλέχθηκε λόγω της ευκολίας ανάπτυξης, της αναγνωσιμότητας του κώδικα και της διαθεσιμότητας βιβλιοθηκών για αριθμητικούς υπολογισμούς και ανάλυση δεδομένων. Η συνολική δομή του συστήματος είναι αρθρωτή και διαχωρίζεται σε τρία βασικά αρχεία, καθένα από τα οποία εξυπηρετεί έναν διακριτό ρόλο στη διαδικασία προσομοίωσης και αξιολόγησης.

Το αρχείο **TrafficTest1.py** υλοποιεί το προτεινόμενο ευφυές σύστημα ελέγχου κυκλοφορίας, το οποίο βασίζεται στον συνδυασμό ασαφούς λογικής και γραμμικής παλινδρόμησης. Στο αρχείο αυτό ορίζονται οι βασικές δομές δεδομένων που περιγράφουν τις διασταυρώσεις, οι μεταβλητές κατάστασης της κυκλοφορίας (μήκη ουρών, ενεργή φάση, χρόνοι πράσινου), καθώς και οι συναρτήσεις που υλοποιούν τον αλγόριθμο λήψης αποφάσεων. Ο ευφυής ελεγκτής δέχεται ως είσοδο την τρέχουσα κατάσταση της κυκλοφορίας και παράγει ως έξοδο τον χρόνο πράσινου φωτός που θα αποδοθεί στην ενεργή φάση.

Το αρχείο **TrafficTest2.py** περιλαμβάνει τα baseline συστήματα ελέγχου, τα οποία χρησιμοποιούνται για λόγους σύγκρισης με το προτεινόμενο ευφυές σύστημα. Συγκεκριμένα, υλοποιούνται απλούστερες στρατηγικές ελέγχου, όπως ο έλεγχος σταθερού χρόνου και ο έλεγχος βασισμένος στο μήκος των ουρών. Οι στρατηγικές αυτές χρησιμοποιούν την ίδια δομή προσομοίωσης και τα ίδια δεδομένα εισόδου, αλλά διαφέρουν αποκλειστικά στον τρόπο υπολογισμού του χρόνου πράσινου φωτός, γεγονός που επιτρέπει την αντικειμενική αξιολόγηση της απόδοσής τους.

Το αρχείο **comparison.py** είναι υπεύθυνο για τον συντονισμό των προσομοιώσεων και τη συλλογή των δεδομένων αξιολόγησης. Στο αρχείο αυτό πραγματοποιείται η εκτέλεση των διαφορετικών σεναρίων ελέγχου, χρησιμοποιώντας κοινές παραμέτρους προσομοίωσης και τα ίδια στοχαστικά δεδομένα αφίξεων. Για κάθε σύστημα ελέγχου καταγράφονται βασικές μετρικές απόδοσης, όπως ο αριθμός εξυπηρετημένων οχημάτων, τα μήκη των ουρών και οι χρόνοι πράσινου φωτός, οι οποίες αποθηκεύονται και χρησιμοποιούνται στη συνέχεια για την πειραματική αποτίμηση.

Η ροή εκτέλεσης του συνολικού συστήματος ξεκινά με την αρχικοποίηση των διασταυρώσεων και των παραμέτρων προσομοίωσης, όπως ο αριθμός των χρονικών βημάτων, οι ρυθμοί αφίξεων οχημάτων και τα όρια των χρόνων πράσινου φωτός. Ακολούθως, για κάθε χρονικό βήμα, ενημερώνεται η κατάσταση της κυκλοφορίας μέσω της προσθήκης νέων οχημάτων στις ουρές

και της απομάκρυνσης οχημάτων από τις κατευθύνσεις που διαθέτουν πράσινο φωτισμό.

Σε κάθε βήμα, ο αντίστοιχος αλγόριθμος ελέγχου καλείται να υπολογίσει τον χρόνο πράσινου φωτός με βάση την τρέχουσα κατάσταση του συστήματος. Η διαδικασία αυτή επαναλαμβάνεται για το σύνολο των χρονικών βημάτων της προσομοίωσης, επιτρέποντας την παρακολούθηση της δυναμικής εξέλιξης της κυκλοφορίας. Με την ολοκλήρωση της προσομοίωσης, τα δεδομένα απόδοσης αποθηκεύονται και καθίστανται διαθέσιμα για περαιτέρω ανάλυση και συγκριτική αξιολόγηση στο επόμενο κεφάλαιο.

Η συγκεκριμένη δομή κώδικα διασφαλίζει την καθαρή διάκριση μεταξύ της υλοποίησης των αλγορίθμων ελέγχου και της διαδικασίας αξιολόγησης, διευκολύνοντας τόσο την επέκταση του συστήματος όσο και την επαναληψιμότητα των πειραμάτων.

4.3 Στοχαστικό Μοντέλο Κυκλοφορίας

Η είσοδος των οχημάτων στο σύστημα μοντελοποιείται με στοχαστικό τρόπο, με στόχο την ρεαλιστική προσομοίωση της τυχαίας φύσης της οδικής κυκλοφορίας. Συγκεκριμένα, οι αφίξεις οχημάτων σε κάθε κατεύθυνση κάθε διασταύρωσης περιγράφονται μέσω κατανομής Poisson, η οποία χρησιμοποιείται ευρέως στη βιβλιογραφία για τη μοντελοποίηση τυχαίων γεγονότων που συμβαίνουν ανεξάρτητα στον χρόνο.

Η κατανομή Poisson χαρακτηρίζεται από μία παράμετρο ρυθμού άφιξης λ, η οποία εκφράζει τον αναμενόμενο αριθμό οχημάτων που εισέρχονται στο σύστημα ανά χρονική μονάδα. Στο πλαίσιο της παρούσας προσομοίωσης, διαφορετικές τιμές του λαποδίδονται σε κάθε διασταύρωση και κατεύθυνση, επιτρέποντας τη δημιουργία ετερογενών κυκλοφοριακών συνθηκών. Με τον τρόπο αυτό, προσομοιώνονται καταστάσεις όπου ορισμένες κατευθύνσεις εμφανίζουν αυξημένη ζήτηση, ενώ άλλες παρουσιάζουν χαμηλότερη κυκλοφορία.

Σε κάθε διακριτό χρονικό βήμα της προσομοίωσης, ο αριθμός των νέων οχημάτων που εισέρχονται σε μία κατεύθυνση προκύπτει ως τυχαία μεταβλητή Poisson με παράμετρο λ. Τα οχήματα αυτά προστίθενται στην αντίστοιχη ουρά, η οποία αναπαριστά τον αριθμό των οχημάτων που αναμένουν την εξυπηρέτησή τους από τον φωτεινό σηματοδότη. Η στοχαστική αυτή διαδικασία επιτρέπει τη δημιουργία ρεαλιστικών διακυμάνσεων στον

κυκλοφοριακό φόρτο και αποτρέπει την εμφάνιση τεχνητά ομαλών ή ντετερμινιστικών μοτίβων.

Πέραν της βασικής στοχαστικής μοντελοποίησης, εφαρμόζεται επιπλέον ένας χρονικός συντελεστής μεταβολής της έντασης κυκλοφορίας, ο οποίος τροποποιεί δυναμικά τις τιμές των ρυθμών άφιξης κατά τη διάρκεια της προσομοίωσης. Ο συντελεστής αυτός λειτουργεί ως πολλαπλασιαστής της παραμέτρου λ και επιτρέπει την προσομοίωση διαφορετικών περιόδων της ημέρας, όπως ώρες αιχμής και περιόδους μειωμένης ζήτησης.

Κατά τις ώρες αιχμής, οι τιμές του συντελεστή αυξάνονται, οδηγώντας σε μεγαλύτερους ρυθμούς αφίξεων και αυξημένη συμφόρηση, ενώ σε περιόδους χαμηλής ζήτησης μειώνονται, προσομοιώνοντας πιο ομαλή ροή κυκλοφορίας. Η χρονική αυτή μεταβολή προσδίδει στο σύστημα τη δυνατότητα να αξιολογηθεί υπό διαφορετικά επίπεδα φόρτου, επιτρέποντας την εξέταση της συμπεριφοράς των αλγορίθμων ελέγχου τόσο σε συνθήκες κανονικής λειτουργίας όσο και σε συνθήκες έντονης συμφόρησης.

Η υιοθέτηση του συγκεκριμένου στοχαστικού μοντέλου αυξάνει τον ρεαλισμό της προσομοίωσης και διασφαλίζει ότι τα αποτελέσματα της πειραματικής αποτίμησης δεν εξαρτώνται από τεχνητές ή ιδανικές συνθήκες. Παράλληλα, επιτρέπει τη δίκαιη και αντικειμενική σύγκριση των εξεταζόμενων συστημάτων ελέγχου, καθώς όλα αξιολογούνται υπό τις ίδιες στοχαστικές εισόδους και χρονικές μεταβολές ζήτησης.

4.4 Ευφυές Σύστημα Ελέγχου (Fuzzy logic & Regression)

Το προτεινόμενο ευφυές σύστημα ελέγχου βασίζεται στον υβριδικό συνδυασμό ασαφούς λογικής και γραμμικής παλινδρόμησης, με στόχο τον προσαρμοστικό και αποδοτικό υπολογισμό του χρόνου πράσινου φωτός σε σηματοδοτούμενες διασταυρώσεις. Η συγκεκριμένη προσέγγιση αξιοποιεί την ικανότητα της ασαφούς λογικής να διαχειρίζεται την αβεβαιότητα και να ενσωματώνει ανθρώπινη εμπειρία, καθώς και τη μαθηματική ακρίβεια των μοντέλων παλινδρόμησης.

Ασαφές Σύστημα Ελέγχου

Το ασαφές σύστημα ελέγχου δέχεται ως εισόδους τα μήκη των ουρών των τεσσάρων κατευθύνσεων κάθε διασταύρωσης (Βορράς, Νότος, Ανατολή και Δύση). Κάθε είσοδος μετατρέπεται σε ασαφή μεταβλητή μέσω της διαδικασίας ασαφοποίησης, χρησιμοποιώντας τρία λεκτικά σύνολα: χαμηλή, μέτρια και υψηλή συμφόρηση. Τα λεκτικά αυτά σύνολα ορίζονται μέσω συναρτήσεων συμμετοχής, οι οποίες καλύπτουν το εύρος τιμών των μηκών ουρών και επιτρέπουν τη μερική συμμετοχή μιας τιμής σε περισσότερα από ένα σύνολα. Η έξοδος του ασαφούς συστήματος είναι ο προτεινόμενος χρόνος πράσινου φωτός, ο οποίος επίσης εκφράζεται αρχικά ως ασαφής μεταβλητή και στη συνέχεια μετατρέπεται σε αριθμητική τιμή μέσω της διαδικασίας αποασαφοποίησης. Οι κανόνες λήψης αποφάσεων εκφράζονται με τη μορφή IF–THEN και βασίζονται σε εμπειρική λογική που προσομοιώνει τη συμπεριφορά ενός ανθρώπινου χειριστή.
Η βάση κανόνων περιλαμβάνει αφενός απλούς κανόνες ανά κατεύθυνση, όπως για παράδειγμα:
Αν η ουρά είναι υψηλή, τότε ο χρόνος πράσινου είναι μεγάλος,
και αφετέρου συνδυαστικούς κανόνες που λαμβάνουν υπόψη ζεύγη κατευθύνσεων, όπως οι κατευθύνσεις Βορράς–Νότος και Ανατολή–Δύση. Οι συνδυαστικοί κανόνες επιτρέπουν την ενίσχυση του χρόνου πράσινου σε περιπτώσεις όπου παρατηρείται ταυτόχρονα αυξημένη συμφόρηση σε περισσότερες από μία κατευθύνσεις της ίδιας φάσης.
Με τον τρόπο αυτό, το ασαφές σύστημα μπορεί να προσαρμόζει δυναμικά τη σηματοδότηση, αποδίδοντας μεγαλύτερο χρόνο πράσινου φωτός στις κατευθύνσεις με αυξημένες ανάγκες εξυπηρέτησης και περιορίζοντας τη σπατάλη χρόνου σε κατευθύνσεις με χαμηλή κυκλοφορία.

Μοντέλο Γραμμικής Παλινδρόμησης

Παράλληλα με το ασαφές σύστημα, χρησιμοποιείται μοντέλο γραμμικής παλινδρόμησης, το οποίο εκτιμά τον χρόνο πράσινου φωτός με βάση το συνολικό μήκος των ουρών της διασταύρωσης. Το μοντέλο αυτό έχει εκπαιδευτεί σε συνθετικά δεδομένα, τα οποία προκύπτουν από διαφορετικούς συνδυασμούς κυκλοφοριακού φόρτου, και αποσκοπεί στην παροχή μιας αριθμητικά συνεπούς εκτίμησης του απαιτούμενου χρόνου εξυπηρέτησης.
Η παλινδρόμηση παρέχει μια συνεχή και ομαλή συνάρτηση που συσχετίζει τον συνολικό φόρτο με τον χρόνο πράσινου, συμβάλλοντας στη σταθερότητα της τελικής απόφασης και περιορίζοντας απότομες μεταβολές στον σηματοδοτικό κύκλο.

Υβριδικός Συνδυασμός και Τελική Απόφαση

Η τελική απόφαση για τον χρόνο πράσινου φωτός προκύπτει ως σταθμισμένος συνδυασμός της εξόδου του ασαφούς συστήματος και της εκτίμησης της γραμμικής παλινδρόμησης. Ο συνδυασμός αυτός επιτρέπει την εκμετάλλευση των πλεονεκτημάτων και των δύο προσεγγίσεων: της ερμηνευσιμότητας και προσαρμοστικότητας της ασαφούς λογικής, καθώς και της αριθμητικής ακρίβειας της παλινδρόμησης.
Επιπλέον, εφαρμόζονται περιορισμοί στα ελάχιστα και μέγιστα επιτρεπτά όρια του χρόνου πράσινου φωτός, ώστε να διασφαλίζεται η ομαλή και ασφαλής λειτουργία του συστήματος. Σε περιπτώσεις εξαιρετικά αυξημένης συμφόρησης, ενεργοποιείται μηχανισμός ενίσχυσης του χρόνου πράσινου, ο οποίος αποτρέπει την υπερβολική συσσώρευση οχημάτων.
Η υβριδική αυτή προσέγγιση επιτρέπει στο σύστημα να λαμβάνει αποφάσεις που είναι ταυτόχρονα κατανοητές, προσαρμοστικές και αριθμητικά συνεπείς, αποτελώντας τον βασικό μηχανισμό ελέγχου της προτεινόμενης λύσης διαχείρισης κυκλοφορίας.

4.5 Υλοποίηση Baseline Συστήματα Ελέγχου

Για λόγους σύγκρισης και αντικειμενικής αξιολόγησης της απόδοσης του προτεινόμενου ευφυούς συστήματος, υλοποιήθηκαν δύο απλούστερα συστήματα ελέγχου φωτεινών σηματοδοτών, τα οποία λειτουργούν ως συστήματα αναφοράς (baseline). Τα συστήματα αυτά δεν χρησιμοποιούν τεχνικές υπολογιστικής νοημοσύνης και βασίζονται σε απλούς, προκαθορισμένους κανόνες λήψης αποφάσεων.

Σύστημα Σταθερού Χρόνου

Το πρώτο baseline σύστημα βασίζεται σε σταθερό χρόνο πράσινου φωτός, ο οποίος παραμένει αμετάβλητος καθ' όλη τη διάρκεια της προσομοίωσης και είναι ανεξάρτητος από την τρέχουσα κυκλοφοριακή κατάσταση. Σε κάθε φάση του σηματοδότη αποδίδεται ένας προκαθορισμένος χρόνος πράσινου φωτός, ο οποίος επιλέγεται ώστε να εξυπηρετεί κατά μέσο όρο τη ροή των οχημάτων.
Η συγκεκριμένη προσέγγιση προσομοιώνει κλασικά συστήματα σταθερού χρονοπρογραμματισμού που χρησιμοποιούνται ευρέως σε αστικά περιβάλλοντα, ιδίως σε περιοχές με σχετικά προβλέψιμα κυκλοφοριακά πρότυπα. Παρότι το σύστημα αυτό χαρακτηρίζεται από απλότητα και προβλέψιμη συμπεριφορά, αδυνατεί να προσαρμοστεί σε αιφνίδιες μεταβολές της κυκλοφορίας, γεγονός που μπορεί να οδηγήσει σε αυξημένους χρόνους αναμονής και μη αποδοτική κατανομή του διαθέσιμου χρόνου πράσινου φωτός.

Σύστημα Βασισμένο στο Μήκος Ουράς

Το δεύτερο baseline σύστημα υλοποιεί μια απλούστερη μορφή προσαρμοστικού ελέγχου, βασισμένη αποκλειστικά στο σχετικό μήκος των ουρών μεταξύ των δύο φάσεων (Βορράς–Νότος και Ανατολή–Δύση). Σε κάθε χρονικό βήμα, το σύστημα υπολογίζει το συνολικό μήκος ουρών για κάθε φάση και προσαρμόζει τον χρόνο πράσινου φωτός αναλογικά, αποδίδοντας μεγαλύτερο χρόνο στην φάση με τη μεγαλύτερη συμφόρηση.
Η προσέγγιση αυτή επιτρέπει μια στοιχειώδη προσαρμογή στις τρέχουσες κυκλοφοριακές συνθήκες, χωρίς όμως να λαμβάνει υπόψη την αβεβαιότητα, τη μη γραμμικότητα ή τη δυναμική εξέλιξη της κυκλοφορίας. Επιπλέον, η απουσία μηχανισμών εξομάλυνσης ή λεκτικής αξιολόγησης μπορεί να οδηγήσει σε απότομες μεταβολές του χρόνου πράσινου φωτός, ιδιαίτερα σε περιπτώσεις έντονων διακυμάνσεων του κυκλοφοριακού φόρτου.

Ρόλος των Baseline Συστημάτων

Τα δύο baseline συστήματα χρησιμοποιούνται αποκλειστικά ως σημεία αναφοράς για τη σύγκριση με το προτεινόμενο ευφυές σύστημα ελέγχου. Η απλότητά τους επιτρέπει την καθαρή απομόνωση της επίδρασης των τεχνικών ασαφούς λογικής και παλινδρόμησης στην απόδοση του συστήματος. Με τον τρόπο αυτό, καθίσταται δυνατή η αξιολόγηση του κατά πόσο η αυξημένη πολυπλοκότητα του ευφυούς συστήματος οδηγεί σε μετρήσιμη βελτίωση της διαχείρισης της κυκλοφορίας.

4.6 Μηχανισμός Σύγκρισης και Συλλογή Μετρικών

Το αρχείο **comparison.py** είναι υπεύθυνο για τον συντονισμό της πειραματικής διαδικασίας και τη συγκριτική αξιολόγηση των διαφορετικών συστημάτων ελέγχου φωτεινών σηματοδοτών. Ο κύριος στόχος του μηχανισμού σύγκρισης είναι η εξασφάλιση δίκαιων, αντικειμενικών και αναπαραγώγιμων αποτελεσμάτων, ώστε οι διαφορές στην απόδοση να αποδίδονται αποκλειστικά στη στρατηγική ελέγχου και όχι σε εξωτερικούς παράγοντες.

Για τον λόγο αυτό, όλα τα συστήματα ελέγχου εκτελούνται υπό κοινές συνθήκες εισόδου. Συγκεκριμένα, χρησιμοποιούνται τα ίδια στοχαστικά δεδομένα αφίξεων οχημάτων, οι ίδιες αρχικές συνθήκες ουρών και οι ίδιες παράμετροι προσομοίωσης, όπως ο αριθμός των χρονικών βημάτων, οι ρυθμοί αφίξεων και τα όρια των χρόνων πράσινου φωτός. Με τον τρόπο αυτό διασφαλίζεται ότι κάθε σύστημα αντιμετωπίζει ακριβώς το ίδιο κυκλοφοριακό σενάριο.

Κατά τη διάρκεια της προσομοίωσης, για κάθε σύστημα ελέγχου συλλέγονται βασικές μετρικές απόδοσης που αντικατοπτρίζουν τη συμπεριφορά και την αποτελεσματικότητά του. Μεταξύ αυτών περιλαμβάνονται ο συνολικός αριθμός διελεύσεων οχημάτων, ο οποίος εκφράζει την ικανότητα εξυπηρέτησης του συστήματος, το συνολικό μήκος των ουρών, που αποτελεί ένδειξη της συμφόρησης, καθώς και οι χρόνοι πράσινου φωτός που αποδόθηκαν σε κάθε φάση κατά τη διάρκεια της προσομοίωσης.

Οι μετρικές αυτές καταγράφονται σε κάθε χρονικό βήμα, επιτρέποντας την παρακολούθηση της δυναμικής εξέλιξης της κυκλοφορίας και τη μελέτη της συμπεριφοράς των συστημάτων σε διαφορετικές συνθήκες φόρτου. Με την ολοκλήρωση της προσομοίωσης, τα δεδομένα αποθηκεύονται σε δομημένη

μορφή και καθίστανται διαθέσιμα για περαιτέρω επεξεργασία και οπτικοποίηση.

Η συγκεντρωτική επεξεργασία των μετρικών πραγματοποιείται στο επόμενο κεφάλαιο, όπου παρουσιάζονται γραφικές παραστάσεις και συγκριτικά αποτελέσματα για τα εξεταζόμενα συστήματα ελέγχου. Ο διαχωρισμός της διαδικασίας σύγκρισης από την παρουσίαση των αποτελεσμάτων συμβάλλει στη σαφή οργάνωση της εργασίας και επιτρέπει την καθαρή διάκριση μεταξύ υλοποίησης και πειραματικής αποτίμησης.

4.7 Σύνοψη

Στο παρόν κεφάλαιο παρουσιάστηκε αναλυτικά η αρχιτεκτονική και η υλοποίηση του προτεινόμενου συστήματος διαχείρισης κυκλοφορίας σε αστικό περιβάλλον. Αρχικά, περιγράφηκε η συνολική δομή του συστήματος και η ροή λειτουργίας του, καθώς και ο τρόπος με τον οποίο σχεδιάστηκε ώστε να επιτρέπει τη δίκαιη και αντικειμενική σύγκριση διαφορετικών στρατηγικών ελέγχου.

Στη συνέχεια, αναλύθηκε η δομή και η ροή του κώδικα, με έμφαση στον αρθρωτό σχεδιασμό και τον διαχωρισμό μεταξύ της υλοποίησης των αλγορίθμων ελέγχου και της διαδικασίας αξιολόγησης. Παρουσιάστηκε επίσης το στοχαστικό μοντέλο κυκλοφορίας που χρησιμοποιείται για την προσομοίωση των αφίξεων οχημάτων, βασισμένο σε κατανομές Poisson και χρονικά μεταβαλλόμενους ρυθμούς, το οποίο προσδίδει ρεαλισμό και δυναμικό χαρακτήρα στην προσομοίωση.

Ιδιαίτερη έμφαση δόθηκε στο προτεινόμενο ευφυές σύστημα ελέγχου, το οποίο βασίζεται στον υβριδικό συνδυασμό ασαφούς λογικής και γραμμικής παλινδρόμησης. Παρουσιάστηκαν αναλυτικά οι είσοδοι, οι κανόνες λήψης αποφάσεων, ο μηχανισμός υπολογισμού του χρόνου πράσινου φωτός και ο τρόπος με τον οποίο επιτυγχάνεται ισορροπία μεταξύ προσαρμοστικότητας και αριθμητικής ακρίβειας. Παράλληλα, περιγράφηκαν τα απλούστερα baseline συστήματα ελέγχου, τα οποία χρησιμοποιούνται ως σημεία αναφοράς για τη συγκριτική αξιολόγηση της απόδοσης.

Τέλος, παρουσιάστηκε ο μηχανισμός σύγκρισης και συλλογής μετρικών, ο οποίος διασφαλίζει την αναπαραγωγιμότητα και τη δικαιοσύνη των πειραμάτων. Τα στοιχεία που παρουσιάστηκαν στο παρόν κεφάλαιο αποτελούν τη βάση για την πειραματική αποτίμηση που ακολουθεί στο επόμενο κεφάλαιο, όπου αναλύονται και συγκρίνονται τα αποτελέσματα των εξεταζόμενων συστημάτων ελέγχου.

ΚΕΦΑΛΑΙΟ 5 Πειραματική Αποτίμηση

5.1 Πειραματική Διαδικασία

Η πειραματική αποτίμηση του προτεινόμενου συστήματος πραγματοποιήθηκε μέσω προσομοίωσης, με στόχο τη σύγκριση της απόδοσής του έναντι απλούστερων στρατηγικών ελέγχου φωτεινών σηματοδοτών. Για λόγους αντικειμενικότητας, όλα τα συστήματα αξιολογήθηκαν υπό τις ίδιες συνθήκες εισόδου και με τις ίδιες παραμέτρους προσομοίωσης.

Συγκεκριμένα, συγκρίθηκαν τα εξής συστήματα:

1. το προτεινόμενο ευφυές σύστημα ελέγχου με ασαφή λογική και παλινδρόμηση,
2. ένα σύστημα σταθερού χρόνου πράσινου φωτός (baseline fixed),
3. ένα σύστημα ελέγχου βασισμένο στο μήκος των ουρών (baseline queue-based).

Κάθε πείραμα εκτελέστηκε για προκαθορισμένο αριθμό χρονικών βημάτων, ώστε να καταγραφεί η συμπεριφορά του συστήματος τόσο σε περιόδους χαμηλής όσο και αυξημένης κυκλοφορίας.

5.2 Ρυθμίσεις και Παράμετροι Πειραμάτων

Οι βασικές παράμετροι των πειραμάτων περιλαμβάνουν:

1. τον συνολικό αριθμό χρονικών βημάτων της προσομοίωσης,
2. τους ρυθμούς άφιξης οχημάτων ανά κατεύθυνση, οι οποίοι μεταβάλλονται χρονικά,
3. τη χρήση κοινού seed για τη στοχαστική γεννήτρια, ώστε τα διαφορετικά συστήματα να λαμβάνουν τα ίδια δεδομένα εισόδου,
4. τα όρια ελάχιστου και μέγιστου χρόνου πράσινου φωτός.

Η παραμετροποίηση αυτή επιτρέπει τη δίκαιη σύγκριση των συστημάτων, καθώς οι διαφοροποιήσεις στην απόδοση οφείλονται αποκλειστικά στη στρατηγική ελέγχου και όχι σε διαφορές στα δεδομένα εισόδου.

5.3 Μετρικές Αξιολόγησης

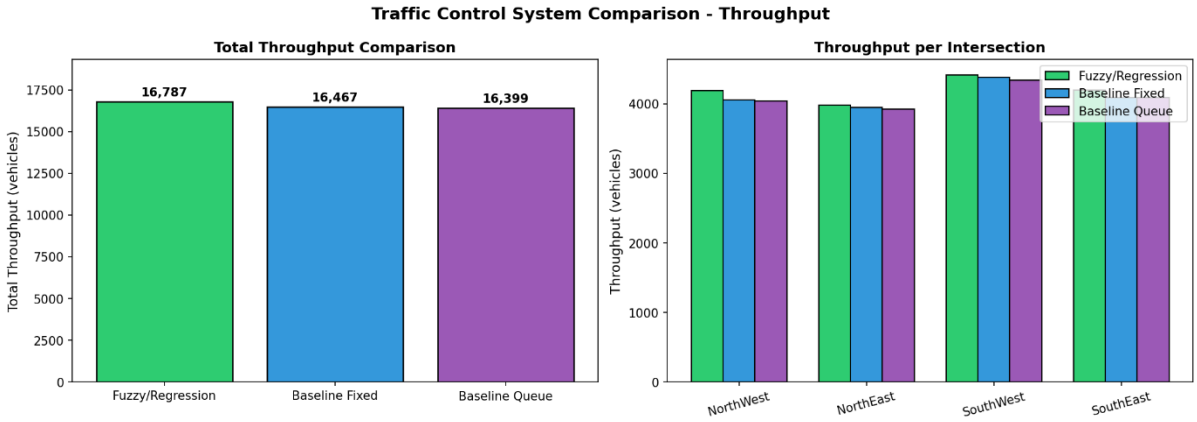
Για την αποτίμηση της απόδοσης των συστημάτων χρησιμοποιήθηκαν οι ακόλουθες μετρικές:

1. **Συνολική διέλευση οχημάτων (Throughput):** αριθμός οχημάτων που διήλθαν επιτυχώς από το δίκτυο κατά τη διάρκεια της προσομοίωσης.
2. **Συνολικό και μέσο μήκος ουρών:** μέτρο της συμφόρησης και της καθυστέρησης στο σύστημα.
3. **Χρόνος αναμονής οχημάτων:** εκτιμάται έμμεσα μέσω της εξέλιξης των ουρών.
4. **Κατανομή χρόνων πράσινου φωτός:** δείχνει τη συμπεριφορά και την προσαρμοστικότητα του ελεγκτή.

Οι παραπάνω μετρικές επιτρέπουν την ολοκληρωμένη αξιολόγηση τόσο της αποδοτικότητας όσο και της σταθερότητας των συστημάτων.

5.4 Αποτελέσματα Πειραμάτων

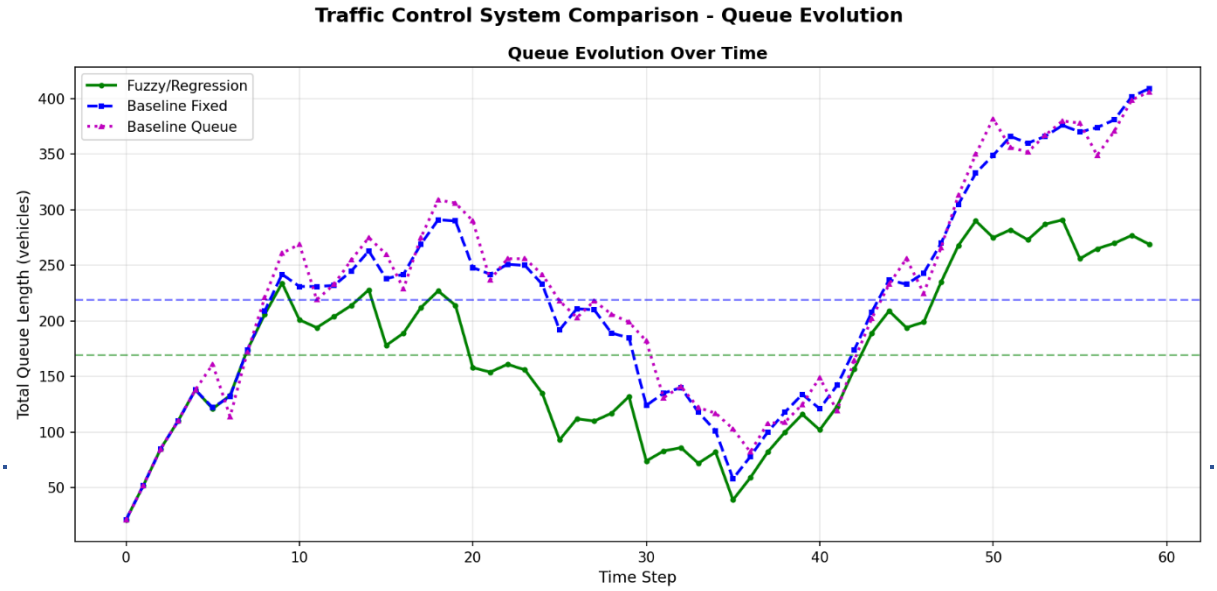
5.4.1 Συνολική Διέλευση Οχημάτων



Σχήμα 5.1 Σύγκριση συνολικού throughput για τα τρία συστήματα ελέγχου

Παρατηρείται ότι το προτεινόμενο ευφυές σύστημα επιτυγχάνει τη μεγαλύτερη συνολική διέλευση οχημάτων. Η βελτίωση σε σχέση με το σύστημα σταθερού χρόνου είναι της τάξης του 1,9%, γεγονός που υποδηλώνει πιο αποδοτική αξιοποίηση του διαθέσιμου χρόνου πράσινου φωτός.

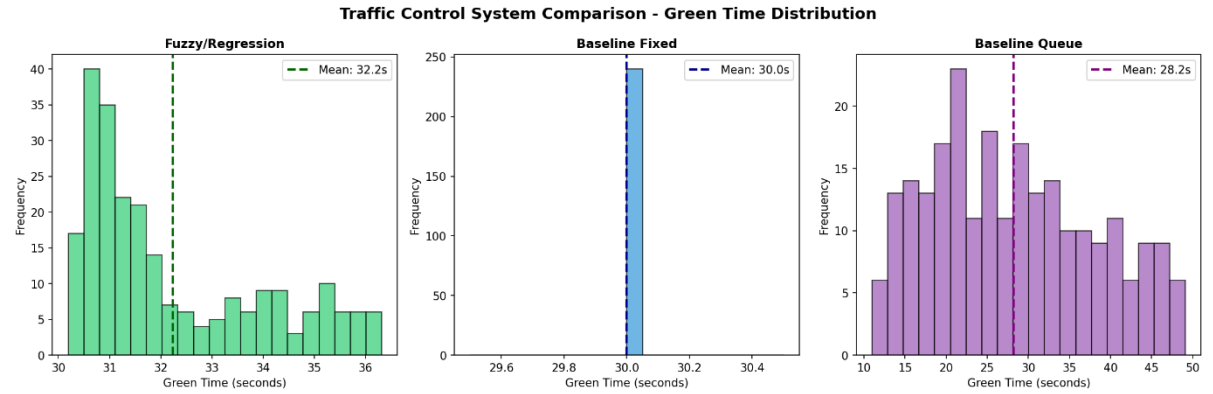
5.4.2 Εξέλιξη Μήκους Ουρών



Σχήμα 5.2 Εξέλιξη συνολικού μήκους ουρών στον χρόνο

Το ευφυές σύστημα εμφανίζει χαμηλότερα επίπεδα συμφόρησης, ιδιαίτερα σε περιόδους αυξημένης κυκλοφορίας. Αντίθετα, το σύστημα σταθερού χρόνου παρουσιάζει εντονότερες διακυμάνσεις, γεγονός που υποδηλώνει μειωμένη προσαρμοστικότητα.

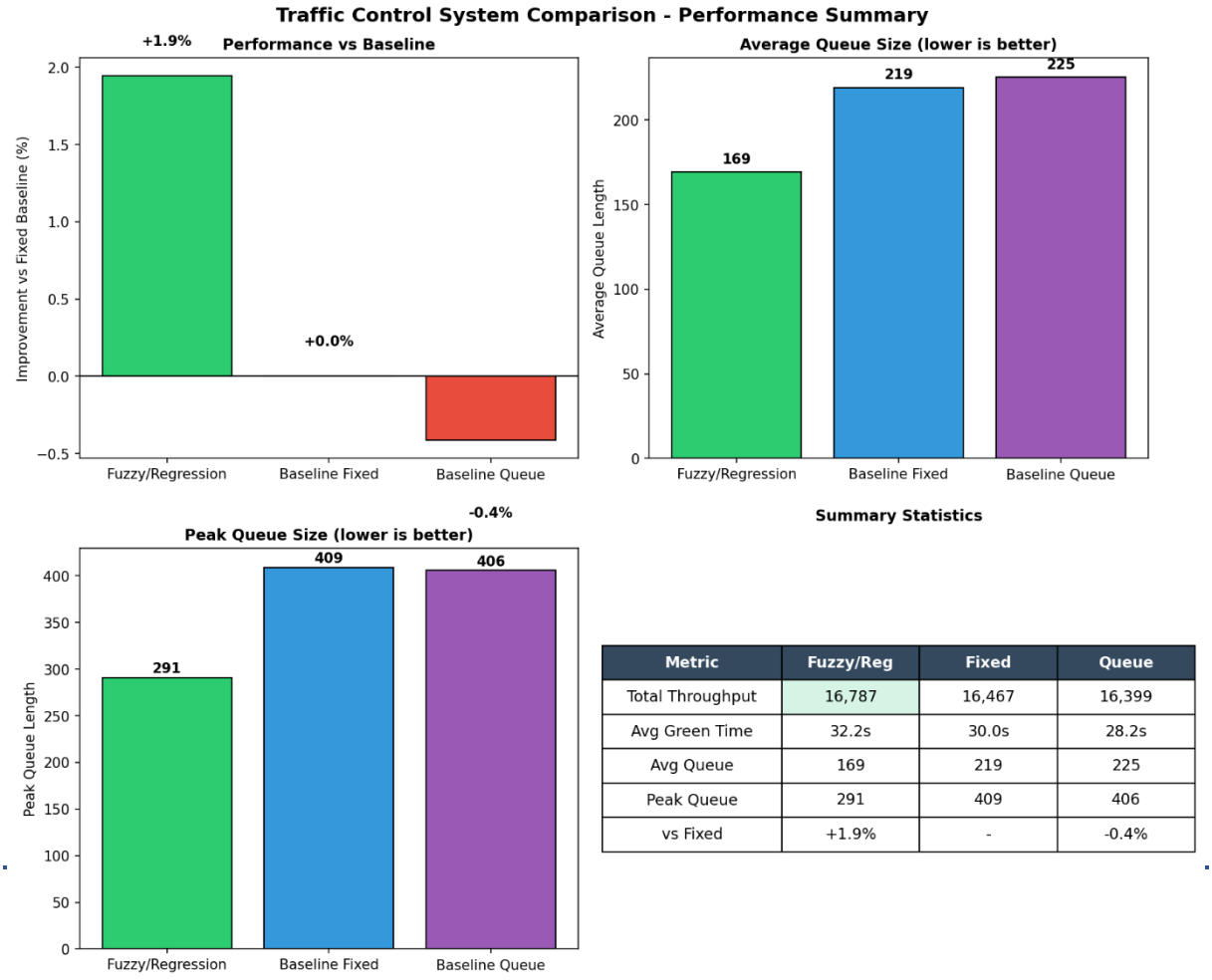
5.4.3 Κατανομή Χρόνου Πράσινου Φωτός



Σχήμα 5.3 Κατανομή χρόνων πράσινου φωτός για τα τρία συστήματα

Το προτεινόμενο σύστημα παρουσιάζει μεγαλύτερη διασπορά στους χρόνους πράσινου, γεγονός που καταδεικνύει τη δυναμική προσαρμογή του στις μεταβαλλόμενες κυκλοφοριακές συνθήκες, σε αντίθεση με το σταθερό σύστημα που διατηρεί αμετάβλητη συμπεριφορά.

5.4.4 Συνοπτική Σύγκριση Μετρικών



Metric	Fuzzy/Reg	Fixed	Queue
Total Throughput	16,787	16,467	16,399
Avg Green Time	32.2s	30.0s	28.2s
Avg Queue	169	219	225
Peak Queue	291	409	406
vs Fixed	+1.9%	-	-0.4%

Σχήμα 5.4 Σύνοψη βασικών μετρικών απόδοσης των συστημάτων ελέγχου

Η συνολική εικόνα επιβεβαιώνει ότι το ευφυές σύστημα υπερέχει τόσο ως προς τη ροή οχημάτων όσο και ως προς τη μείωση της συμφόρησης.

5.5 Συζήτηση Αποτελεσμάτων

Τα πειραματικά αποτελέσματα δείχνουν ότι ο συνδυασμός ασαφούς λογικής και παλινδρόμησης επιτρέπει πιο αποδοτική και ευέλικτη διαχείριση της κυκλοφορίας σε σχέση με τις απλούστερες προσεγγίσεις. Η δυνατότητα του συστήματος να προσαρμόζει δυναμικά τον χρόνο πράσινου φωτός οδηγεί σε καλύτερη εξισορρόπηση μεταξύ διαφορετικών κατευθύνσεων και μειωμένες καθυστερήσεις.

Παρότι οι βελτιώσεις στο throughput είναι σχετικά μικρές, η συνολική συμπεριφορά του συστήματος είναι πιο σταθερή και λιγότερο επιρρεπής σε φαινόμενα συμφόρησης, γεγονός που είναι ιδιαίτερα σημαντικό σε πραγματικές αστικές συνθήκες.

5.6 Σύνοψη

Στο παρόν κεφάλαιο παρουσιάστηκε η πειραματική αποτίμηση του προτεινόμενου συστήματος διαχείρισης κυκλοφορίας. Μέσω προσομοίωσης και σύγκρισης με baseline συστήματα, αποδείχθηκε ότι το ευφυές σύστημα παρουσιάζει βελτιωμένη απόδοση σε βασικές μετρικές, επιβεβαιώνοντας την αποτελεσματικότητα της προτεινόμενης προσέγγισης.

ΚΕΦΑΛΑΙΟ 6 Συμπεράσματα και Μελλοντικές Προεκτάσεις

6.1 Συμπεράσματα

Στην παρούσα πτυχιακή εργασία μελετήθηκε και υλοποιήθηκε ένα ευφυές σύστημα διαχείρισης κυκλοφορίας σε σηματοδοτούμενες αστικές διασταυρώσεις, βασισμένο στον συνδυασμό ασαφούς λογικής και τεχνικών παλινδρόμησης. Το σύστημα σχεδιάστηκε με στόχο τη δυναμική προσαρμογή του χρόνου πράσινου φωτός στις εκάστοτε κυκλοφοριακές συνθήκες.

Μέσω προσομοίωσης, το προτεινόμενο σύστημα συγκρίθηκε με δύο απλούστερες προσεγγίσεις ελέγχου, ένα σύστημα σταθερού χρόνου και ένα σύστημα βασισμένο στο μήκος των ουρών. Τα πειραματικά αποτελέσματα έδειξαν ότι το ευφυές σύστημα παρουσιάζει βελτιωμένη συμπεριφορά ως προς βασικές μετρικές απόδοσης, όπως η συνολική διέλευση οχημάτων και η μείωση της συμφόρησης.

Ιδιαίτερα, παρατηρήθηκε ότι η δυνατότητα του συστήματος να προσαρμόζει δυναμικά τον χρόνο πράσινου φωτός οδηγεί σε πιο ομαλή εξέλιξη των ουρών και καλύτερη αξιοποίηση της υποδομής, σε σύγκριση με τις στατικές ή απλουστευμένες δυναμικές προσεγγίσεις.

6.2 Περιορισμού της Προσέγγισης

Παρότι τα αποτελέσματα είναι ενθαρρυντικά, η παρούσα εργασία παρουσιάζει ορισμένους περιορισμούς. Η αξιολόγηση του συστήματος πραγματοποιήθηκε αποκλειστικά μέσω προσομοίωσης και δεν περιλαμβάνει δεδομένα από πραγματικές κυκλοφοριακές συνθήκες. Επιπλέον, το μοντέλο προσομοίωσης βασίζεται σε απλουστευμένες υποθέσεις σχετικά με τη συμπεριφορά των οδηγών και τη ροή των οχημάτων.

Ακόμη, η αρχιτεκτονική του συστήματος περιορίζεται σε απομονωμένο σύνολο διασταυρώσεων και δεν εξετάζεται ο συντονισμένος έλεγχος σε μεγαλύτερης κλίμακας οδικά δίκτυα.

6.3 Μελλοντικές Προεκτάσεις

Η παρούσα εργασία μπορεί να αποτελέσει βάση για περαιτέρω έρευνα και ανάπτυξη. Μία πιθανή μελλοντική προέκταση είναι η εφαρμογή του προτεινόμενου συστήματος σε μεγαλύτερα και πιο σύνθετα δίκτυα διασταυρώσεων, με στόχο τον συντονισμένο έλεγχο σε επίπεδο περιοχής ή πόλης.

Επιπλέον, θα μπορούσε να εξεταστεί ο συνδυασμός της ασαφούς λογικής με πιο σύγχρονες τεχνικές μηχανικής μάθησης, όπως η ενισχυτική μάθηση, ώστε το σύστημα να μπορεί να προσαρμόζεται αυτόνομα σε μεταβαλλόμενα κυκλοφοριακά πρότυπα. Τέλος, η αξιολόγηση του συστήματος με πραγματικά δεδομένα κυκλοφορίας θα μπορούσε να προσφέρει πιο αξιόπιστα συμπεράσματα σχετικά με την πρακτική του εφαρμογή.

APPENDIX

Appendix 1 – Fuzzy Controller

```
# simulation.py
# -*- coding: utf-8 -*-

# ======================================================================
# Σενάριο προσομοίωσης "έξυπνων" φαναριών σε μικρό δίκτυο 2x2 διασταυρώσεων
#
# Ιδέα:
# - Έχουμε αφίξεις οχημάτων (Poisson) από 4 κατευθύνσεις σε κάθε κόμβο (N/S/E/W).
# - Κάθε κόμβος αποφασίζει ποιο ζεύγος (N/S ή E/W) θα έχει πράσινο.
# - Η διάρκεια πράσινου "υπολογίζεται" με συνδυασμό fuzzy + regression.
# - Όταν ένα ρεύμα έχει πράσινο, οχήματα αποφορτίζονται και είτε:
#   (α) βγαίνουν εκτός δικτύου ή
#   (β) δρομολογούνται σε γειτονική διασταύρωση με turn probabilities,
#       με προστασία spillback (να μη γεμίζει άπειρα η επόμενη ουρά).
# ======================================================================

import sys
import io

# ----------------------------------------------------------------------
# Windows/Terminal fix:
# Σε μερικά Windows terminals τα ελληνικά "σπάνε".
# Τυλίγουμε stdout/stderr σε UTF-8 ώστε τα print να βγαίνουν σωστά.
# ----------------------------------------------------------------------
sys.stdout = io.TextIOWrapper(sys.stdout.buffer, encoding='utf-8', errors='replace')
sys.stderr = io.TextIOWrapper(sys.stderr.buffer, encoding='utf-8', errors='replace')

# ----------------------------------------------------------------------
# Matplotlib backend:
# Agg = χωρίς GUI (βολικό σε servers / headless runs).
# Πρέπει να δηλωθεί ΠΡΙΝ κάνουμε import το pyplot.
# ----------------------------------------------------------------------
import matplotlib
matplotlib.use('Agg')

import argparse
import numpy as np
import pandas as pd  # δεν χρησιμοποιείται εδώ, αλλά κρατιέται για πιθανά exports/analysis
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
import skfuzzy as fuzz
from skfuzzy import control as ctrl
from dataclasses import dataclass

# ----------------------------------------------------------------------
# CLI παράμετρος: seed
# Αν δώσεις seed, η προοομοίωση γίνεται αναπαραγώγιμη (ίδια τυχαία ροή κάθε φορά).
# Αν δεν δώσεις seed, θα παίρνεις "άλλο" αποτέλεσμα σε κάθε run.
# ----------------------------------------------------------------------
parser = argparse.ArgumentParser()
parser.add_argument(
```

```
    "--seed",
    type=int,
    default=None,
    help="Τυχαίος σπόρος για αναπαραγωγιμότητα. Άφησέ το κενό για διαφορετικό αποτέλεσμα κάθε φορά."
)
args = parser.parse_args()

# Ένας κεντρικός RNG για ΟΛΟ το πρόγραμμα.
# Αυτό βοηθάει πολύ όταν κάνεις debug: ένα seed -> ίδια ακριβώς τυχαία συμπεριφορά παντού.
RNG = np.random.default_rng(args.seed)

# ============================================================================
# Σταθερές / παράμετροι προσομοίωσης (μαζεμένες σε dataclass)
# ============================================================================
@dataclass
class SimConfig:
    # Ελάχιστος/μέγιστος αριθμός "steps" που κρατάμε το ίδιο active ζεύγος,
    # για να αποφύγουμε τρεμόπαιγμα και για να υπάρχει δικαιοσύνη.
    MIN_GREEN: int = 5
    MAX_GREEN: int = 10

    # Ρυθμός εκφόρτισης: πόσα οχήματα/δευτ. "περνάνε" όταν μια κατεύθυνση έχει πράσινο.
    DISCHARGE_RATE: float = 1.5

    # Αν μια ουρά ξεπεράσει αυτό το όριο, το θεωρούμε έντονη συμφόρηση.
    # Τότε δίνουμε πιο επιθετικά πράσινο (boost).
    CONGESTION_THRESHOLD: int = 300

    # "Ταβάνι" ουράς ανά κατεύθυνση (spillback / περιορισμός).
    MAX_QUEUE: int = 500

    # Anti ping-pong:
    # για να αλλάξεις νωρίς πράσινο, η άλλη πλευρά πρέπει να είναι SWITCH_RATIO φορές πιο φορτωμένη.
    SWITCH_RATIO: float = 1.2

CFG = SimConfig()

# ============================================================================
# Συντελεστής ώρας: απλό μοντέλο κίνησης μέσα στη μέρα
# (αιχμές πρωί/απόγευμα, ήρεμο μεσημέρι)
# ============================================================================
def time_of_day_multiplier(hour):
    # Πρωινή και απογευματινή αιχμή
    if 7 <= hour < 10 or 17 <= hour < 20:
        return 1.5
    # Μεσημέρι πέφτει η κίνηση
    elif 12 <= hour < 16:
        return 0.7
    # Αλλιώς: "φυσιολογικό"
    else:
        return 1.0

# ============================================================================
# Shared regression model
# Το φτιάχνουμε ΜΙΑ φορά και το μοιράζουμε σε όλες τις διασταυρώσεις,
# για να μη γίνεται training 4 φορές χωρίς λόγο.
```

```
# ======================================================================
def make_shared_regression_model():
    # Synthetic training set:
    # Input: ουρές [N, S, E, W]
    # Output: ένας "λογικός" χρόνος πράσινου 25..60 sec βάσει συνολικού φόρτου.
    X_train, y_train = [], []

    # Sample ανά 50 οχήματα (0..500) για να κρατηθεί το dataset μικρό/γρήγορο.
    for n in range(0, 501, 50):
        for s in range(0, 501, 50):
            for e in range(0, 501, 50):
                for w in range(0, 501, 50):
                    X_train.append([n, s, e, w])

                    # "Truth rule" που βάζουμε εμείς:
                    # Ξεκινάμε από 25s και ανεβαίνουμε με το συνολικό φορτίο μέχρι ~60s.
                    total = n + s + e + w
                    green_time = 25 + (35 * total / 2000)  # 25..60
                    y_train.append(min(60, max(25, green_time)))

    model = LinearRegression()
    model.fit(X_train, y_train)
    return model

REG_MODEL = make_shared_regression_model()

# ======================================================================
# Κλάση Intersection: ένας κόμβος/διασταύρωση
# Κρατά ουρές, αποφασίζει πράσινο, εκφορτίζει και κάνει routing.
# ======================================================================
class Intersection:
    def __init__(self, name, arrival_rate, rng):
        # Όνομα/ID κόμβου (για prints και plots)
        self.name = name

        # Ρυθμοί άφιξης (Poisson λ) ανά κατεύθυνση: [N, S, E, W]
        self.arrival_rate = arrival_rate

        # Τρέχουσες ουρές ανά κατεύθυνση
        self.queues = [0, 0, 0, 0]

        # Μετρικά: πόσα οχήματα εξυπηρετήθηκαν συνολικά σε κάθε κατεύθυνση
        self.total_discharged = [0, 0, 0, 0]

        # Placeholder για πιθανό "history" αφίξεων (δεν το χρησιμοποιούμε ουσιαστικά ακόμα)
        self.arrival_data = [np.zeros(100) for _ in range(4)]

        # Μετρητής βημάτων προσομοίωσης
        self.step_counter = 0

        # Συνδέσεις προς άλλες διασταυρώσεις για routing:
        # [N, S, E, W] -> άλλο Intersection ή None (έξοδος από το δίκτυο)
        self.connected = [None, None, None, None]

        # Κοινός RNG (controlled από seed)
        self.rng = rng

        # Τραβάμε τις ρυθμίσεις από το CFG (πιο καθαρό από "μαγικούς αριθμούς")
```

```
        self.CONGESTION_THRESHOLD = CFG.CONGESTION_THRESHOLD
        self.MAX_QUEUE = CFG.MAX_QUEUE
        self.MIN_GREEN = CFG.MIN_GREEN
        self.MAX_GREEN = CFG.MAX_GREEN
        self.discharge_rate = CFG.DISCHARGE_RATE
        self.switch_ratio = CFG.SWITCH_RATIO

        # Shared regression model
        self.reg_model = REG_MODEL

        # Φτιάχνουμε fuzzy controller για τον κόμβο (μία φορά)
        self.build_fuzzy()

        # active_pair: ποιο ζεύγος έχει πράσινο τώρα
        # 0 = N/S, 1 = E/W
        self.active_pair = 0

        # Πόσα συνεχόμενα steps είμαστε στο ίδιο active_pair
        self.green_steps = 0

        # Πιθανότητες στροφής για οχήματα που φεύγουν από κάθε είσοδο.
        # Format: [straight, left, right]
        self.turn_probs = {
            0: [0.6, 0.2, 0.2],  # από North
            1: [0.6, 0.2, 0.2],  # από South
            2: [0.6, 0.2, 0.2],  # από East
            3: [0.6, 0.2, 0.2],  # από West
        }

    # ------------------------------------------------------------
    # Fuzzy controller:
    # Παίρνει τις ουρές N/S/E/W και βγάζει μια πρόταση για "short/medium/long" green.
    # ------------------------------------------------------------
    def build_fuzzy(self):
        # Inputs 0..500 (όσο και τα queues μας), output 0..60 sec
        traffic = np.arange(0, 501, 1)
        green_range = np.arange(0, 61, 1)

        # 4 είσοδοι (antecedents): ουρές από κάθε κατεύθυνση
        self.vars = [ctrl.Antecedent(traffic, d) for d in ['north', 'south', 'east', 'west']]

        # Μία έξοδος (consequent): πράσινο σε seconds
        self.output = ctrl.Consequent(green_range, 'green')

        # Memberships για ουρές: low/medium/high
        # (χειροκίνητα thresholds ώστε να έχουν νόημα στο 0..500)
        for v in self.vars:
            v['low'] = fuzz.trimf(traffic, [0, 0, 150])
            v['medium'] = fuzz.trimf(traffic, [100, 250, 400])
            v['high'] = fuzz.trimf(traffic, [300, 500, 500])

        # Memberships για output:
        # Τα ranges ξεκινούν από ~25s για να μην κάνει υπερβολικά "κοφτό" πράσινο.
        self.output['short'] = fuzz.trimf(green_range, [25, 32, 42])
        self.output['medium'] = fuzz.trimf(green_range, [38, 48, 56])
        self.output['long'] = fuzz.trimf(green_range, [52, 58, 60])

        rules = []
```

```
        # Βασικοί κανόνες: κάθε κατεύθυνση "τραβάει" το πράσινο προς short/medium/long
        for v in self.vars:
            rules.append(ctrl.Rule(v['low'], self.output['short']))
            rules.append(ctrl.Rule(v['medium'], self.output['medium']))
            rules.append(ctrl.Rule(v['high'], self.output['long']))

        # Συνδυαστικοί: αν και τα δύο ρεύματα ενός ζεύγους είναι φορτωμένα, τότε long/medium
        n, s, e, w = self.vars
        rules.append(ctrl.Rule(n['high'] & s['high'], self.output['long']))
        rules.append(ctrl.Rule(e['high'] & w['high'], self.output['long']))
        rules.append(ctrl.Rule(n['medium'] & s['medium'], self.output['medium']))
        rules.append(ctrl.Rule(e['medium'] & w['medium'], self.output['medium']))

        system = ctrl.ControlSystem(rules)
        self.fuzzy_sim = ctrl.ControlSystemSimulation(system)

    # ----------------------------------------------------------------
    # Πρόβλεψη χρόνου πράσινου (sec):
    # - παίρνουμε αποτέλεσμα από fuzzy
    # - παίρνουμε πρόβλεψη από regression
    # - τα ζυγίζουμε (60/40)
    # - βάζουμε "bonus/boost" αν υπάρχει συμφόρηση
    # - κάνουμε clamp για να μένει σε λογικά όρια
    # ----------------------------------------------------------------
    def predict_green_time(self):
        # Κόβουμε τις ουρές στο MAX_QUEUE, γιατί το σύστημα έχει σχεδιαστεί για
0..MAX_QUEUE.
        clamped_queues = [min(q, self.MAX_QUEUE) for q in self.queues]

        # Fuzzy inference με τωρινές ουρές
        self.fuzzy_sim.input['north'] = clamped_queues[0]
        self.fuzzy_sim.input['south'] = clamped_queues[1]
        self.fuzzy_sim.input['east'] = clamped_queues[2]
        self.fuzzy_sim.input['west'] = clamped_queues[3]
        self.fuzzy_sim.compute()
        fuzzy_result = self.fuzzy_sim.output['green']

        # Regression prediction (με clamp για να μην ξεφύγει)
        reg_result = self.reg_model.predict([clamped_queues])[0]
        reg_result = max(20, min(60, reg_result))

        # Συνδυασμός (λίγο πιο "εμπιστοσύνη" στο fuzzy γιατί πιάνει τοπικούς κανόνες)
        base_time = (fuzzy_result * 0.6) + (reg_result * 0.4)

        # Bonus: αν το ενεργό ζεύγος είναι "τίγκα", δώσε λίγο παραπάνω να ξεμπλοκάρει
        if self.active_pair == 0:
            active_queue = clamped_queues[0] + clamped_queues[1]
        else:
            active_queue = clamped_queues[2] + clamped_queues[3]

        if active_queue > 400:
            base_time += min(10, (active_queue - 400) / 40)

        # Global boost αν κάπου υπάρχει έντονη συμφόρηση
        if any(q > self.CONGESTION_THRESHOLD for q in self.queues):
            base_time *= 1.4
```

```
        # Τελικό clamp: κρατάμε ρεαλιστικό πράσινο [25, 60]
        return max(25, min(base_time, 60))

    # ---------------------------------------------------------------
    # Ένα step προσομοίωσης:
    # 1) arrivals
    # 2) υπολογισμός "ιδανικού" green time
    # 3) απόφαση αλλαγής/παραμονής ενεργού ζεύγους
    # 4) discharge + routing (+spillback)
    # ---------------------------------------------------------------
    def step(self, current_hour):
        # Συντελεστής ώρας: κάνει την κίνηση πιο "ρεαλιστική" μέσα στη μέρα
        multiplier = time_of_day_multiplier(current_hour)

        # Arrivals: Poisson για κάθε κατεύθυνση (με time-of-day multiplier)
        for i in range(4):
            arrivals = self.rng.poisson(self.arrival_rate[i] * multiplier)
            self.queues[i] += arrivals

        # Υπολογίζουμε πόσο πράσινο θα "ήθελε" το σύστημα αυτή τη στιγμή
        green = self.predict_green_time()

        # Decision logic για active_pair:
        # - κάτω από MIN_GREEN: δεν αλλάζουμε (σταθερότητα)
        # - πάνω από MAX_GREEN: αλλάζουμε αναγκαστικά (δικαιοσύνη)
        # - ενδιάμεσα: αλλάζουμε μόνο αν το άλλο ζεύγος είναι ξεκάθαρα πιο φορτωμένο
        if self.green_steps < self.MIN_GREEN:
            pass
        elif self.green_steps >= self.MAX_GREEN:
            self.active_pair = 1 - self.active_pair
            self.green_steps = 0
        else:
            ns = self.queues[0] + self.queues[1]
            ew = self.queues[2] + self.queues[3]
            R = self.switch_ratio

            if self.active_pair == 0 and ew > ns * R:
                self.active_pair = 1
                self.green_steps = 0
            elif self.active_pair == 1 and ns > ew * R:
                self.active_pair = 0
                self.green_steps = 0

        # Μικρό log για να βλέπουμε τι γίνεται στο runtime
        print(f"\n[{self.name}] Step {self.step_counter + 1}, Green Time: {green:.2f}s")

        # Discharge + routing
        for i in range(4):
            # Η κατεύθυνση i έχει πράσινο αν ανήκει στο ενεργό ζεύγος
            is_green_dir = (self.active_pair == 0 and i in [0, 1]) or (self.active_pair == 1 and i in [2, 3])

            if is_green_dir:
                rate = self.discharge_rate
                discharged = min(int(green * rate), self.queues[i])

                # Αφαιρούμε από ουρά και γράφουμε το metric
                self.queues[i] -= discharged
                self.total_discharged[i] += discharged
```

```
            # Αν η έξοδος αυτή οδηγεί σε άλλη διασταύρωση, κάνουμε routing εκεί
            if self.connected[i] and discharged > 0:
                straight, left, right = self.turn_probs[i]

                # "Κανόνας" mapping:
                # - straight: ίδια κατεύθυνση index
                # - left: (i+3) mod 4
                # - right: (i+1) mod 4
                directions = [(i + 0) % 4, (i + 3) % 4, (i + 1) % 4]

                remaining_to_route = discharged

                # Προσπαθούμε να σπρώξουμε οχήματα, αλλά σεβόμαστε το MAX_QUEUE του
επόμενου κόμβου.
                for p, d in zip([straight, left, right], directions):
                    target = self.connected[i].queues[d]
                    capacity_left = self.connected[i].MAX_QUEUE - target
                    if capacity_left <= 0:
                        continue

                    desired = int(discharged * p)
                    desired = min(desired, remaining_to_route)

                    to_push = min(desired, max(0, capacity_left))
                    if to_push > 0:
                        self.connected[i].queues[d] += to_push
                        remaining_to_route -= to_push

                # Ό,τι δεν "χώρεσε" λόγω spillback επιστρέφει στην τωρινή ουρά.
                # Και αφαιρείται από το discharged metric γιατί πρακτικά δεν πέρασε.
                if remaining_to_route > 0:
                    self.queues[i] += remaining_to_route
                    self.total_discharged[i] -= remaining_to_route

            # Αν δεν υπάρχει connection, τα οχήματα θεωρούνται ότι βγαίνουν εκτός δικτύου.
            else:
                discharged = 0

            print(f"  Direction {['N','S','E','W'][i]}: Queue={self.queues[i]} (Discharged={discharged})")

        # Κλείνουμε το step
        self.green_steps += 1
        self.step_counter += 1
        return green

# ============================================================================
# Κατασκευή μικρού grid 2x2 διασταυρώσεων
# ============================================================================
junctions = [[None, None], [None, None]]

# Arrival rates ανά διασταύρωση (N,S,E,W)
rates_grid = [
    [[10, 10, 8, 9],  [9, 8, 11, 10]],
    [[12, 11, 10, 10], [10, 10, 10, 10]]
]

# Δημιουργούμε τους κόμβους (όλοι παίρνουν τον ίδιο RNG)
```

```
junctions[0][0] = Intersection("NorthWest", rates_grid[0][0], RNG)
junctions[0][1] = Intersection("NorthEast", rates_grid[0][1], RNG)
junctions[1][0] = Intersection("SouthWest", rates_grid[1][0], RNG)
junctions[1][1] = Intersection("SouthEast", rates_grid[1][1], RNG)

# ======================================================================
# Συνδέσεις (routing) μέσα στο δίκτυο:
# Για κάθε κόμβο και κατεύθυνση, λέμε σε ποιον κόμβο “βγαίνει”.
# ======================================================================
junctions[0][0].connected[1] = junctions[1][0]  # NW -> South -> SW
junctions[0][0].connected[2] = junctions[0][1]  # NW -> East  -> NE
junctions[0][1].connected[1] = junctions[1][1]  # NE -> South -> SE
junctions[1][0].connected[2] = junctions[1][1]  # SW -> East  -> SE
junctions[1][0].connected[0] = junctions[0][0]  # SW -> North -> NW
junctions[1][1].connected[0] = junctions[0][1]  # SE -> North -> NE
junctions[0][1].connected[3] = junctions[0][0]  # NE -> West  -> NW
junctions[1][1].connected[3] = junctions[1][0]  # SE -> West  -> SW

# ======================================================================
# Κύριος βρόχος προσομοίωσης
# ======================================================================
time_steps = 60  # πόσα βήματα θα τρέξουμε συνολικά

# Για plot: κρατάμε σε κάθε step το άθροισμα ουρών ανά κόμβο
all_data = {j.name: [] for row in junctions for j in row}

for t in range(time_steps):
    # Ξεκινάμε 07:00 και κάθε 4 steps περνάει 1 ώρα
    current_hour = (7 + t // 4) % 24

    for row in junctions:
        for j in row:
            g = j.step(current_hour)
            all_data[j.name].append(sum(j.queues))

# ======================================================================
# Εκτύπωση “τελικού” summary: πόσα οχήματα πέρασαν συνολικά από κάθε κόμβο
# ======================================================================
print("\nΣυνολικός αριθμός οχημάτων που πέρασαν ανά διασταύρωση:")
for row in junctions:
    for j in row:
        print(f"{j.name}: {[int(x) for x in j.total_discharged]} (Σύνολο: {sum(j.total_discharged)})")

# ======================================================================
# Plot: εξέλιξη συνολικής ουράς ανά κόμβο στο χρόνο
# ======================================================================
plt.figure(figsize=(12, 6))
for name, data in all_data.items():
    plt.plot(data, label=name)
plt.legend()
plt.title("Σύνολο Ουρών ανά Διασταύρωση")
plt.xlabel("Χρονικά Βήματα")
plt.ylabel("Οχήματα σε Ουρά")
plt.grid()
plt.show()
```

Appendix 2 – Baseline Systems

```
# simulation_baseline.py
# -*- coding: utf-8 -*-

# ============================================================
# simulation_baseline.py
# Baseline προσομοίωση φωτεινής σηματοδότησης σε grid 2x2.
#
# Τι είναι “baseline” εδώ;
# - Δεν χρησιμοποιούμε Fuzzy Logic ούτε ML.
# - Έχουμε 2 επιλογές controller:
#   1) fixed : σταθερός χρόνος πράσινου (π.χ. 30s)
#   2) queue : απλός “ουρο-ευαίσθητος” (μεγαλύτερο πράσινο όταν το ενεργό ζεύγος έχει μεγαλύτερο φορτίο)
#
# Το αρχείο είναι φτιαγμένο για να τρέχει και σε μη-GUI περιβάλλον (Agg backend),
# και να εμφανίζει σωστά ελληνικά σε Windows terminal.
# ============================================================

import sys
import io

# ------------------------------------------------------------
# Windows console / UTF-8
# ------------------------------------------------------------
# Σε αρκετά Windows terminals, τα ελληνικά μπορεί να “σπάνε”.
# Με αυτό τυλίγουμε stdout/stderr σε UTF-8 ώστε τα prints να βγαίνουν σωστά.
sys.stdout = io.TextIOWrapper(sys.stdout.buffer, encoding='utf-8', errors='replace')
sys.stderr = io.TextIOWrapper(sys.stderr.buffer, encoding='utf-8', errors='replace')

import matplotlib

# ------------------------------------------------------------
# Matplotlib χωρίς GUI
# ------------------------------------------------------------
# Agg backend: χρήσιμο όταν τρέχουμε σε server/WSL/CI ή γενικά χωρίς display.
# ΠΡΟΣΟΧΗ: πρέπει να μπει πριν από το import pyplot.
matplotlib.use('Agg')

import argparse
import numpy as np
import pandas as pd  # (προαιρετικό προς το παρόν, χρήσιμο αν θες export/analysis μετά)
import matplotlib.pyplot as plt
from dataclasses import dataclass

# ------------------------------------------------------------
# CLI flags
# ------------------------------------------------------------
# --seed: για αναπαραγωγιμότητα (ίδια τυχαία ροή)
# --mode: ποιο baseline controller θα χρησιμοποιήσουμε
parser = argparse.ArgumentParser()
parser.add_argument(
    "--seed",
    type=int,
    default=None,
    help="Τυχαίος σπόρος για αναπαραγωγιμότητα. Άφησέ το κενό για διαφορετικό αποτέλεσμα κάθε
```

```
φορά."
)
parser.add_argument(
    "--mode",
    type=str,
    choices=["fixed", "queue"],
    default="fixed",
    help="Baseline ελεγκτής: 'fixed' (σταθερό πράσινο) ή 'queue' (ουρο-ευαίσθητος, χωρίς ML/FIS)."
)
args = parser.parse_args()

# Ένας ενιαίος RNG για όλο το πρόγραμμα.
# Αυτό βοηθάει να είναι deterministic το simulation όταν δώσεις seed.
RNG = np.random.default_rng(args.seed)

# ------------------------------------------------------------
# Κεντρικές ρυθμίσεις προσομοίωσης (μαζεμένες για να αλλάζουν εύκολα)
# ------------------------------------------------------------
@dataclass
class SimConfig:
    # Τα MIN/MAX_GREEN είναι σε steps (όχι σε seconds).
    # Δηλαδή δεν αλλάζουμε "κάθε δευτερόλεπτο", αλλά κάθε βήμα προσομοίωσης.
    MIN_GREEN: int = 5
    MAX_GREEN: int = 10

    # Πόσα οχήματα "μπορούν να περάσουν" ανά δευτερόλεπτο πράσινου.
    DISCHARGE_RATE: float = 1.5

    # Αν κάποια ουρά ξεπεράσει το όριο, θεωρούμε ότι έχουμε συμφόρηση.
    CONGESTION_THRESHOLD: int = 300

    # Ταβάνι ουράς ανά κατεύθυνση (για spillback προστασία).
    MAX_QUEUE: int = 500

    # Anti ping-pong: πόσο πιο μεγάλη πρέπει να είναι η απέναντι πλευρά
    # για να αλλάξουμε πρόωρα ζεύγος φαναριού.
    SWITCH_RATIO: float = 1.2

    # Baseline fixed: σταθερός χρόνος πράσινου σε seconds.
    BASE_FIXED_GREEN: int = 30


CFG = SimConfig()

# ------------------------------------------------------------
# Time-of-day multiplier (ώρες αιχμής / ήρεμες ώρες)
# ------------------------------------------------------------
def time_of_day_multiplier(hour: int) -> float:
    """
    Επιστρέφει έναν συντελεστή που πολλαπλασιάζει τους ρυθμούς αφίξεων.

    Ιδέα:
    - Πρωί και απόγευμα έχουμε αιχμή (περισσότερα αυτοκίνητα).
    - Μεσημέρι (12-16) λίγο πιο "χαλαρά".
    - Υπόλοιπες ώρες = κανονικό.
    """
    if 7 <= hour < 10 or 17 <= hour < 20:
        return 1.5
```

```
    elif 12 <= hour < 16:
        return 0.7
    else:
        return 1.0


# ------------------------------------------------------------
# Intersection: μία διασταύρωση με baseline controller
# ------------------------------------------------------------
class Intersection:
    """
    Μοντελοποιεί μια διασταύρωση με 4 εισόδους/κατευθύνσεις: N, S, E, W.

    Κάθε step:
    1) Γίνονται arrivals (Poisson) στις ουρές.
    2) Επιλέγεται χρόνος πράσινου (baseline controller).
    3) Μένουμε/αλλάζουμε ζεύγος (N/S vs E/W) με MIN/MAX + anti ping-pong.
    4) Γίνεται discharge και routing προς γειτονικές διασταυρώσεις (αν υπάρχουν),
       με spillback προστασία (όριο MAX_QUEUE στο target).
    """

    def __init__(self, name, arrival_rate, rng, mode="fixed"):
        # "Ταυτότητα" διασταύρωσης
        self.name = name

        # Ρυθμοί άφιξης ανά κατεύθυνση [N, S, E, W] (μέσος όρος Poisson)
        self.arrival_rate = arrival_rate

        # Ουρές ανά κατεύθυνση (τρέχουσα κατάσταση)
        self.queues = [0, 0, 0, 0]

        # Μετρικό: πόσα οχήματα "όντως εξυπηρετήθηκαν" ανά κατεύθυνση
        self.total_discharged = [0, 0, 0, 0]

        # Placeholder: αν θες στο μέλλον να κρατάς ιστορικό arrivals κλπ.
        self.arrival_data = [np.zeros(100) for _ in range(4)]

        # Μετρητής steps
        self.step_counter = 0

        # Συνδέσεις προς γειτονικές διασταυρώσεις (N, S, E, W).
        # Αν είναι None σημαίνει "έξοδος από το σύστημα".
        self.connected = [None, None, None, None]

        # Κοινός RNG
        self.rng = rng

        # Επιλογή baseline controller
        self.mode = mode

        # Παίρνουμε configs από CFG (για ευκολία/καθαρότητα)
        self.CONGESTION_THRESHOLD = CFG.CONGESTION_THRESHOLD
        self.MAX_QUEUE = CFG.MAX_QUEUE
        self.MIN_GREEN = CFG.MIN_GREEN
        self.MAX_GREEN = CFG.MAX_GREEN
        self.discharge_rate = CFG.DISCHARGE_RATE
        self.switch_ratio = CFG.SWITCH_RATIO
        self.base_fixed_green = CFG.BASE_FIXED_GREEN
```

```
        # active_pair = ποιο ζεύγος έχει πράσινο:
        # 0 -> N/S, 1 -> E/W
        self.active_pair = 0

        # Πόσα συνεχόμενα steps είμαστε στο ίδιο active_pair
        self.green_steps = 0

        # Πιθανότητες στροφής (straight/left/right) για κάθε εισερχόμενη κατεύθυνση.
        # Χρησιμοποιούνται στο routing προς τον επόμενο κόμβο.
        self.turn_probs = {
            0: [0.6, 0.2, 0.2],  # από North
            1: [0.6, 0.2, 0.2],  # από South
            2: [0.6, 0.2, 0.2],  # από East
            3: [0.6, 0.2, 0.2],  # από West
        }

    # ----------------------------------------------------------
    # Baseline πράσινο: fixed ή "queue-sensitive"
    # ----------------------------------------------------------
    def get_green_time(self) -> float:
        """
        Επιστρέφει χρόνο πράσινου (seconds) χωρίς ML/Fuzzy.

        fixed:
          - πάντα BASE_FIXED_GREEN.

        queue:
          - βλέπει το load (ns vs ew) και δίνει περισσότερο πράσινο στο ενεργό ζεύγος
            όσο μεγαλύτερο είναι το μερίδιό του στο συνολικό φορτίο.
        """
        if self.mode == "fixed":
            base_time = float(self.base_fixed_green)

        else:  # mode == "queue"
            ns = self.queues[0] + self.queues[1]
            ew = self.queues[2] + self.queues[3]

            if self.active_pair == 0:
                active, other = ns, ew
            else:
                active, other = ew, ns

            # Μικρό epsilon για να μη γίνει division-by-zero όταν είναι όλα μηδέν.
            ratio = active / (active + other + 1e-6)

            # Χαρτογράφηση ratio -> [5..60] seconds
            base_time = 5.0 + 55.0 * ratio

        # Αν υπάρχει πολύ βαριά συμφόρηση σε οποιαδήποτε κατεύθυνση,
        # δίνουμε ένα μικρό boost (λίγο πιο ήπιο από "έξυπνο" controller).
        if any(q > self.CONGESTION_THRESHOLD for q in self.queues):
            base_time *= 1.2

        return float(min(base_time, 60.0))

    # ----------------------------------------------------------
    # Ένα βήμα προσομοίωσης
```

```
    # ----------------------------------------------------------
    def step(self, current_hour: int) -> float:
        """
        Εκτελεί ένα χρονικό βήμα:
        - Arrivals (Poisson)
        - Απόφαση green time
        - Απόφαση αλλαγής active_pair (MIN/MAX + anti ping-pong)
        - Discharge + routing με spillback
        """
        multiplier = time_of_day_multiplier(current_hour)

        # 1) Arrivals
        for i in range(4):
            arrivals = self.rng.poisson(self.arrival_rate[i] * multiplier)
            self.queues[i] += arrivals

        # 2) Green time (baseline)
        green = self.get_green_time()

        # 3) Απόφαση εναλλαγής ζεύγους
        # MIN_GREEN: δεν αλλάζουμε για να μη "τρεμοπαίζει"
        # MAX_GREEN: αλλάζουμε υποχρεωτικά για fairness
        if self.green_steps < self.MIN_GREEN:
            pass
        elif self.green_steps >= self.MAX_GREEN:
            self.active_pair = 1 - self.active_pair
            self.green_steps = 0
        else:
            # Early switch μόνο αν το άλλο ζεύγος είναι αρκετά πιο φορτωμένο.
            ns = self.queues[0] + self.queues[1]
            ew = self.queues[2] + self.queues[3]
            R = self.switch_ratio

            if self.active_pair == 0 and ew > ns * R:
                self.active_pair = 1
                self.green_steps = 0
            elif self.active_pair == 1 and ns > ew * R:
                self.active_pair = 0
                self.green_steps = 0

        print(f"\n[{self.name}] Step {self.step_counter + 1}, Green Time: {green:.2f}s
(mode={self.mode})")

        # 4) Discharge + routing
        for i in range(4):
            is_green_dir = (self.active_pair == 0 and i in [0, 1]) or (self.active_pair == 1 and i in [2, 3])

            if is_green_dir:
                discharged = min(int(green * self.discharge_rate), self.queues[i])

                # "Φεύγουν" από την ουρά
                self.queues[i] -= discharged
                self.total_discharged[i] += discharged

                # Αν υπάρχει επόμενος κόμβος, προσπαθούμε να προωθήσουμε οχήματα.
                if self.connected[i] and discharged > 0:
                    straight, left, right = self.turn_probs[i]
```

```
                # Mapping: straight = ίδια διεύθυνση index,
                # left = "+3", right = "+1" (mod 4)
                directions = [(i + 0) % 4, (i + 3) % 4, (i + 1) % 4]

                remaining_to_route = discharged

                # Προώθηση με έλεγχο χωρητικότητας (spillback protection)
                for p, d in zip([straight, left, right], directions):
                    target_queue = self.connected[i].queues[d]
                    capacity_left = self.connected[i].MAX_QUEUE - target_queue
                    if capacity_left <= 0:
                        continue

                    desired = int(discharged * p)
                    desired = min(desired, remaining_to_route)

                    to_push = min(desired, capacity_left)
                    if to_push > 0:
                        self.connected[i].queues[d] += to_push
                        remaining_to_route -= to_push

                # Ό,τι δεν χωράει, "κολλάει" πίσω στην τρέχουσα ουρά
                # και διορθώνουμε το metric γιατί πρακτικά δεν πέρασε.
                if remaining_to_route > 0:
                    self.queues[i] += remaining_to_route
                    self.total_discharged[i] -= remaining_to_route

            else:
                discharged = 0

            print(f"  Direction {['N','S','E','W'][i]}: Queue={self.queues[i]} (Discharged={discharged})")

        # Τέλος step
        self.green_steps += 1
        self.step_counter += 1
        return green


# ------------------------------------------------------------
# Δημιουργία grid 2x2 διασταυρώσεων
# ------------------------------------------------------------
junctions = [[None, None], [None, None]]

# Ρυθμοί άφιξης ανά διασταύρωση (N,S,E,W)
rates_grid = [
    [[10, 10, 8, 9], [9, 8, 11, 10]],
    [[12, 11, 10, 10], [10, 10, 10, 10]],
]

# Όλες οι διασταυρώσεις μοιράζονται τον ίδιο RNG (άρα ένα seed ελέγχει τα πάντα)
junctions[0][0] = Intersection("NorthWest", rates_grid[0][0], RNG, mode=args.mode)
junctions[0][1] = Intersection("NorthEast", rates_grid[0][1], RNG, mode=args.mode)
junctions[1][0] = Intersection("SouthWest", rates_grid[1][0], RNG, mode=args.mode)
junctions[1][1] = Intersection("SouthEast", rates_grid[1][1], RNG, mode=args.mode)

# ------------------------------------------------------------
# Συνδέσεις μεταξύ κόμβων (routing μέσα στο δίκτυο)
# ------------------------------------------------------------
```

```
junctions[0][0].connected[1] = junctions[1][0]  # NW προς South -> SW
junctions[0][0].connected[2] = junctions[0][1]  # NW προς East  -> NE
junctions[0][1].connected[1] = junctions[1][1]  # NE προς South -> SE
junctions[1][0].connected[2] = junctions[1][1]  # SW προς East  -> SE
junctions[1][0].connected[0] = junctions[0][0]  # SW προς North -> NW
junctions[1][1].connected[0] = junctions[0][1]  # SE προς North -> NE
junctions[0][1].connected[3] = junctions[0][0]  # NE προς West  -> NW
junctions[1][1].connected[3] = junctions[1][0]  # SE προς West  -> SW

# ------------------------------------------------------------
# Κύριος βρόχος προσομοίωσης
# ------------------------------------------------------------
time_steps = 60  # πόσα steps θα τρέξει το σενάριο
all_data = {j.name: [] for row in junctions for j in row}  # συνολική ουρά ανά step

for t in range(time_steps):
    # Ξεκινάμε 07:00 και κάθε 4 steps περνάει 1 ώρα.
    current_hour = (7 + t // 4) % 24

    for row in junctions:
        for j in row:
            j.step(current_hour)
            all_data[j.name].append(sum(j.queues))

# ------------------------------------------------------------
# Σύνοψη αποτελεσμάτων
# ------------------------------------------------------------
print("\nΣυνολικός αριθμός οχημάτων που πέρασαν ανά διασταύρωση (baseline):")
for row in junctions:
    for j in row:
        print(f"{j.name}: {[int(x) for x in j.total_discharged]} (Σύνολο: {sum(j.total_discharged)})")

# ------------------------------------------------------------
# Plot: συνολικό μέγεθος ουράς ανά διασταύρωση σε βάθος χρόνου
# ------------------------------------------------------------
plt.figure(figsize=(12, 6))
for name, data in all_data.items():
    plt.plot(data, label=name)
plt.legend()
plt.title(f"Σύνολο Ουρών ανά Διασταύρωση (baseline mode={args.mode})")
plt.xlabel("Χρονικά Βήματα")
plt.ylabel("Οχήματα σε Ουρά")
plt.grid()
plt.show()
```

Appendix 3 – Comparison Code

```
# comparison.py
# -*- coding: utf-8 -*-

"""
Script σύγκρισης μεταξύ:
- "έξυπνου" controller (Fuzzy + Regression)
- baseline fixed controller
- baseline queue-based controller

Τι κάνει πρακτικά:
1) Τρέχει τα scripts μέσω subprocess (με ίδιο seed)
2) Διαβάζει stdout και εξάγει:
   - throughput ανά διασταύρωση + συνολικό throughput
   - εξέλιξη συνολικής ουράς στο χρόνο
   - green times (διανομή/μέσος όρος)
3) Φτιάχνει γραφήματα σύγκρισης και τα αποθηκεύει σε PNG

Σημείωση:
Το parsing βασίζεται σε patterns που τυπώνουν τα άλλα scripts.
Αν αλλάξει το format των print, χρειάζεται update εδώ.
"""

import subprocess
import re
import sys
import os
import io

# Windows console encoding fix (για ελληνικά στα outputs)
sys.stdout = io.TextIOWrapper(sys.stdout.buffer, encoding="utf-8", errors="replace")
sys.stderr = io.TextIOWrapper(sys.stderr.buffer, encoding="utf-8", errors="replace")

import matplotlib
matplotlib.use("Agg")  # Non-GUI backend - must be before pyplot import

import matplotlib.pyplot as plt
import numpy as np

def run_simulation(script_name, seed, mode=None):
    """
    Τρέχει ένα simulation script σαν "ξεχωριστή διεργασία" και επιστρέφει stdout.

    Γιατί subprocess;
    - Θέλουμε ίδιο workflow όπως θα το έτρεχε ένας χρήστης από terminal.
    - Θέλουμε να συγκρίνουμε scripts χωρίς να τα κάνουμε import μεταξύ τους.
    """
    cmd = [sys.executable, script_name, "--seed", str(seed)]
    if mode:
        cmd.extend(["--mode", mode])

    # Στήνουμε περιβάλλον για:
    # - Agg backend ώστε να μη ζητάει GUI
    # - utf-8 ώστε τα ελληνικά prints να μη χαλάνε
    env = os.environ.copy()
    env["MPLBACKEND"] = "Agg"
```

```
    env["PYTHONIOENCODING"] = "utf-8"

    result = subprocess.run(
        cmd,
        capture_output=True,
        text=True,
        env=env,
        cwd=".",
        encoding="utf-8",
        errors="replace",
    )

    if result.returncode != 0:
        print(f"Error running {script_name}:")
        print(result.stderr)

    return result.stdout

def parse_results(output):
    """
    Διαβάζει το stdout της προσομοίωσης και βγάζει χρήσιμα metrics.

    Εξάγουμε:
    - throughput dict (ανά διασταύρωση)
    - total throughput
    - queue evolution (ένα συνολικό νούμερο ανά step)
    - green times + avg green
    """
    # --- throughput ανά διασταύρωση ---
    throughput = {}
    pattern = r"(\w+): \[[\d, ]+\] \(Σύνολο: (\d+)\)"
    for match in re.finditer(pattern, output):
        name, total = match.groups()
        throughput[name] = int(total)

    # --- queues ανά step ---
    # Προσοχή: εδώ κάνουμε parsing από print lines.
    # "Κρατάμε" τις ουρές μέχρι να ξεκινήσει το επόμενο step block.
    queues_per_step = []
    step_pattern = r"\[(\w+)\] Step (\d+), Green Time: ([\d.]+)s"
    queue_pattern = r"Direction [NSEW]: Queue=(\d+)"

    current_step_queues = []
    for line in output.split("\n"):
        # Μόλις βρούμε νέο step, κλείνουμε το προηγούμενο aggregation
        step_match = re.search(step_pattern, line)
        if step_match:
            if current_step_queues:
                queues_per_step.append(sum(current_step_queues))
                current_step_queues = []

        queue_match = re.search(queue_pattern, line)
        if queue_match:
            current_step_queues.append(int(queue_match.group(1)))

    # Κλείσιμο και για το τελευταίο step
    if current_step_queues:
        queues_per_step.append(sum(current_step_queues))
```

```
    # --- green times ---
    green_times = [float(m.group(3)) for m in re.finditer(step_pattern, output)]

    return {
        "throughput": throughput,
        "total_throughput": sum(throughput.values()),
        # Στο stdout υπάρχουν 4 intersections ανά time step -> παίρνουμε ένα sample ανά 4
        "queue_evolution": queues_per_step[::4],
        "green_times": green_times,
        "avg_green_time": np.mean(green_times) if green_times else 0,
    }

def run_comparison(seed=42):
    """
    Εκτελεί όλο το comparison pipeline:
    - τρέξιμο προσομοιώσεων
    - parsing
    - εκτύπωση αποτελεσμάτων
    - παραγωγή plots
    """
    print("=" * 60)
    print("ΣΥΓΚΡΙΣΗ ΣΥΣΤΗΜΑΤΩΝ ΕΛΕΓΧΟΥ ΚΥΚΛΟΦΟΡΙΑΣ")
    print("=" * 60)
    print(f"Seed: {seed}\n")

    # 1) Smart controller
    print("Εκτέλεση Fuzzy/Regression (TrafficTest1)...")
    fuzzy_output = run_simulation("TrafficTest1.py", seed)
    fuzzy_results = parse_results(fuzzy_output)

    # 2) Baseline fixed
    print("Εκτέλεση Baseline Fixed (TrafficTest2 --mode fixed)...")
    fixed_output = run_simulation("TrafficTest2.py", seed, "fixed")
    fixed_results = parse_results(fixed_output)

    # 3) Baseline queue-based
    print("Εκτέλεση Baseline Queue (TrafficTest2 --mode queue)...")
    queue_output = run_simulation("TrafficTest2.py", seed, "queue")
    queue_results = parse_results(queue_output)

    # --------- results printout ----------
    print("\n" + "=" * 60)
    print("ΑΠΟΤΕΛΕΣΜΑΤΑ ΣΥΓΚΡΙΣΗΣ")
    print("=" * 60)

    print("\n1. ΣΥΝΟΛΙΚΗ ΔΙΕΛΕΥΣΗ (THROUGHPUT)")
    print("-" * 40)
    print(f"  Fuzzy/Regression: {fuzzy_results['total_throughput']:,} οχήματα")
    print(f"  Baseline Fixed:   {fixed_results['total_throughput']:,} οχήματα")
    print(f"  Baseline Queue:   {queue_results['total_throughput']:,} οχήματα")

    # % βελτίωση έναντι fixed
    if fixed_results["total_throughput"] > 0:
        fuzzy_vs_fixed = (
            (fuzzy_results["total_throughput"] - fixed_results["total_throughput"])
            / fixed_results["total_throughput"]
            * 100
```

```
        )
        print(f"\n  Fuzzy vs Fixed: {fuzzy_vs_fixed:+.1f}%")
    else:
        fuzzy_vs_fixed = 0
        print("\n  Fuzzy vs Fixed: N/A (no data)")

    # % βελτίωση έναντι queue-based
    if queue_results["total_throughput"] > 0:
        fuzzy_vs_queue = (
            (fuzzy_results["total_throughput"] - queue_results["total_throughput"])
            / queue_results["total_throughput"]
            * 100
        )
        print(f"  Fuzzy vs Queue: {fuzzy_vs_queue:+.1f}%")
    else:
        fuzzy_vs_queue = 0
        print("  Fuzzy vs Queue: N/A (no data)")

    print("\n2. ΜΕΣΟΣ ΧΡΟΝΟΣ ΠΡΑΣΙΝΟΥ")
    print("-" * 40)
    print(f"  Fuzzy/Regression: {fuzzy_results['avg_green_time']:.2f}s")
    print(f"  Baseline Fixed:   30.00s (σταθερό)")
    print("  Baseline Queue:   (δυναμικό)")

    print("\n3. ΔΙΕΛΕΥΣΗ ΑΝΑ ΔΙΑΣΤΑΥΡΩΣΗ")
    print("-" * 40)
    print(f"  {'Διασταύρωση':<12} {'Fuzzy':>10} {'Fixed':>10} {'Queue':>10}")
    print(f"  {'-'*12} {'-'*10} {'-'*10} {'-'*10}")
    for junction in ["NorthWest", "NorthEast", "SouthWest", "SouthEast"]:
        fuzzy_t = fuzzy_results["throughput"].get(junction, 0)
        fixed_t = fixed_results["throughput"].get(junction, 0)
        queue_t = queue_results["throughput"].get(junction, 0)
        print(f"  {junction:<12} {fuzzy_t:>10,} {fixed_t:>10,} {queue_t:>10,}")

    # --------- conclusion ----------
    print("\n" + "=" * 60)
    print("ΣΥΜΠΕΡΑΣΜΑ")
    print("=" * 60)

    best_system = max(
        [
            ("Fuzzy/Regression", fuzzy_results["total_throughput"]),
            ("Baseline Fixed", fixed_results["total_throughput"]),
            ("Baseline Queue", queue_results["total_throughput"]),
        ],
        key=lambda x: x[1],
    )

    print(f"\nΤο καλύτερο σύστημα για throughput: {best_system[0]}")
    print(f"Με {best_system[1]:,} συνολικά οχήματα")

    if fuzzy_results["total_throughput"] >= fixed_results["total_throughput"]:
        print("\n*** Το Fuzzy/Regression σύστημα ξεπερνά ή ισούται με το Baseline Fixed! ***")
    else:
        diff = fixed_results["total_throughput"] - fuzzy_results["total_throughput"]
        print(f"\nΤο Baseline Fixed έχει {diff:,} περισσότερα οχήματα ({-fuzzy_vs_fixed:.1f}%
καλύτερο)")
        print("Πιθανοί λόγοι (γενικά):")
```

```
        print("  - το smart σύστημα μπορεί να κόβει πράσινο σε χαμηλή κίνηση")
        print("  - κρατάει πιο "ισορροπημένες" ουρές αντί να κυνηγάει μόνο throughput")
        print("  - εξαρτάται από τα rates/thresholds που έχεις βάλει")

    # plots
    create_comparison_plots(fuzzy_results, fixed_results, queue_results, seed)

    return fuzzy_results, fixed_results, queue_results

def create_comparison_plots(fuzzy, fixed, queue, seed):
    """
    Παράγει και αποθηκεύει 4 βασικά charts + ένα combined overview.
    Τα κρατάμε σε PNG ώστε να είναι εύκολα να μπουν σε report/εργασία.
    """

    # === CHART 1: Throughput comparisons ===
    fig1, axes1 = plt.subplots(1, 2, figsize=(14, 5))
    fig1.suptitle("Traffic Control System Comparison - Throughput", fontsize=14,
fontweight="bold")

    ax1 = axes1[0]
    systems = ["Fuzzy/Regression", "Baseline Fixed", "Baseline Queue"]
    throughputs = [fuzzy["total_throughput"], fixed["total_throughput"],
queue["total_throughput"]]
    colors = ["#2ecc71", "#3498db", "#9b59b6"]

    bars = ax1.bar(systems, throughputs, color=colors, edgecolor="black", linewidth=1.2)
    ax1.set_ylabel("Total Throughput (vehicles)", fontsize=11)
    ax1.set_title("Total Throughput Comparison", fontsize=12, fontweight="bold")
    ax1.set_ylim(0, max(throughputs) * 1.15)

    for bar, val in zip(bars, throughputs):
        ax1.text(
            bar.get_x() + bar.get_width() / 2,
            bar.get_height() + 200,
            f"{val:,}",
            ha="center",
            va="bottom",
            fontsize=10,
            fontweight="bold",
        )

    ax2 = axes1[1]
    junctions = ["NorthWest", "NorthEast", "SouthWest", "SouthEast"]
    x = np.arange(len(junctions))
    width = 0.25

    fuzzy_vals = [fuzzy["throughput"].get(j, 0) for j in junctions]
    fixed_vals = [fixed["throughput"].get(j, 0) for j in junctions]
    queue_vals = [queue["throughput"].get(j, 0) for j in junctions]

    ax2.bar(x - width, fuzzy_vals, width, label="Fuzzy/Regression", color="#2ecc71",
edgecolor="black")
    ax2.bar(x, fixed_vals, width, label="Baseline Fixed", color="#3498db", edgecolor="black")
    ax2.bar(x + width, queue_vals, width, label="Baseline Queue", color="#9b59b6",
edgecolor="black")

    ax2.set_ylabel("Throughput (vehicles)", fontsize=11)
```

```
    ax2.set_title("Throughput per Intersection", fontsize=12, fontweight="bold")
    ax2.set_xticks(x)
    ax2.set_xticklabels(junctions, rotation=15)
    ax2.legend()

    plt.tight_layout()
    fig1.savefig("chart1_throughput.png", dpi=150, bbox_inches="tight")
    plt.close(fig1)

    # === CHART 2: Queue evolution ===
    fig2, ax3 = plt.subplots(figsize=(12, 6))
    fig2.suptitle("Traffic Control System Comparison - Queue Evolution", fontsize=14,
fontweight="bold")

    steps = range(len(fuzzy["queue_evolution"]))
    ax3.plot(steps, fuzzy["queue_evolution"], "g-", linewidth=2, label="Fuzzy/Regression",
marker="o", markersize=3)
    ax3.plot(steps, fixed["queue_evolution"], "b--", linewidth=2, label="Baseline Fixed",
marker="s", markersize=3)
    ax3.plot(steps, queue["queue_evolution"], "m:", linewidth=2, label="Baseline Queue",
marker="^", markersize=3)

    ax3.set_xlabel("Time Step", fontsize=11)
    ax3.set_ylabel("Total Queue Length (vehicles)", fontsize=11)
    ax3.set_title("Queue Evolution Over Time", fontsize=12, fontweight="bold")
    ax3.legend(loc="upper left")
    ax3.grid(True, alpha=0.3)

    # Μέσοι όροι σαν γραμμές αναφοράς
    ax3.axhline(y=np.mean(fuzzy["queue_evolution"]), color="green", linestyle="--", alpha=0.5,
label="_nolegend_")
    ax3.axhline(y=np.mean(fixed["queue_evolution"]), color="blue", linestyle="--", alpha=0.5,
label="_nolegend_")

    plt.tight_layout()
    fig2.savefig("chart2_queue_evolution.png", dpi=150, bbox_inches="tight")
    plt.close(fig2)

    # === CHART 3: Green time distribution ===
    fig3, axes3 = plt.subplots(1, 3, figsize=(15, 5))
    fig3.suptitle("Traffic Control System Comparison - Green Time Distribution", fontsize=14,
fontweight="bold")

    ax4 = axes3[0]
    ax4.hist(fuzzy["green_times"], bins=20, color="#2ecc71", edgecolor="black", alpha=0.7)
    ax4.axvline(
        x=np.mean(fuzzy["green_times"]),
        color="darkgreen",
        linestyle="--",
        linewidth=2,
        label=f"Mean: {np.mean(fuzzy['green_times']):.1f}s",
    )
    ax4.set_xlabel("Green Time (seconds)", fontsize=10)
    ax4.set_ylabel("Frequency", fontsize=10)
    ax4.set_title("Fuzzy/Regression", fontsize=11, fontweight="bold")
    ax4.legend()

    ax5 = axes3[1]
```

```
    ax5.hist(fixed["green_times"], bins=20, color="#3498db", edgecolor="black", alpha=0.7)
    ax5.axvline(
        x=np.mean(fixed["green_times"]),
        color="darkblue",
        linestyle="--",
        linewidth=2,
        label=f"Mean: {np.mean(fixed['green_times']):.1f}s",
    )
    ax5.set_xlabel("Green Time (seconds)", fontsize=10)
    ax5.set_ylabel("Frequency", fontsize=10)
    ax5.set_title("Baseline Fixed", fontsize=11, fontweight="bold")
    ax5.legend()

    ax6 = axes3[2]
    ax6.hist(queue["green_times"], bins=20, color="#9b59b6", edgecolor="black", alpha=0.7)
    ax6.axvline(
        x=np.mean(queue["green_times"]),
        color="purple",
        linestyle="--",
        linewidth=2,
        label=f"Mean: {np.mean(queue['green_times']):.1f}s",
    )
    ax6.set_xlabel("Green Time (seconds)", fontsize=10)
    ax6.set_ylabel("Frequency", fontsize=10)
    ax6.set_title("Baseline Queue", fontsize=11, fontweight="bold")
    ax6.legend()

    plt.tight_layout()
    fig3.savefig("chart3_green_time_distribution.png", dpi=150, bbox_inches="tight")
    plt.close(fig3)

    # === CHART 4: Performance summary ===
    fig4, axes4 = plt.subplots(2, 2, figsize=(12, 10))
    fig4.suptitle("Traffic Control System Comparison - Performance Summary", fontsize=14,
fontweight="bold")

    ax7 = axes4[0, 0]
    baseline_fixed = fixed["total_throughput"]
    improvements = [
        ((fuzzy["total_throughput"] - baseline_fixed) / baseline_fixed) * 100,
        0,
        ((queue["total_throughput"] - baseline_fixed) / baseline_fixed) * 100,
    ]
    bar_colors = ["#2ecc71" if x >= 0 else "#e74c3c" for x in improvements]
    bars = ax7.bar(systems, improvements, color=bar_colors, edgecolor="black")
    ax7.axhline(y=0, color="black", linewidth=1)
    ax7.set_ylabel("Improvement vs Fixed Baseline (%)", fontsize=10)
    ax7.set_title("Performance vs Baseline", fontsize=11, fontweight="bold")
    for bar, val in zip(bars, improvements):
        ypos = bar.get_height() + 0.2 if val >= 0 else bar.get_height() - 0.5
        ax7.text(bar.get_x() + bar.get_width() / 2, ypos, f"{val:+.1f}%", ha="center", fontsize=10,
fontweight="bold")

    ax8 = axes4[0, 1]
    avg_queues = [
        np.mean(fuzzy["queue_evolution"]),
        np.mean(fixed["queue_evolution"]),
        np.mean(queue["queue_evolution"]),
```

```
    ]
    bars = ax8.bar(systems, avg_queues, color=colors, edgecolor="black")
    ax8.set_ylabel("Average Queue Length", fontsize=10)
    ax8.set_title("Average Queue Size (lower is better)", fontsize=11, fontweight="bold")
    for bar, val in zip(bars, avg_queues):
        ax8.text(bar.get_x() + bar.get_width() / 2, bar.get_height() + 5, f"{val:.0f}", ha="center",
fontsize=10, fontweight="bold")

    ax9 = axes4[1, 0]
    peak_queues = [max(fuzzy["queue_evolution"]), max(fixed["queue_evolution"]),
max(queue["queue_evolution"])]
    bars = ax9.bar(systems, peak_queues, color=colors, edgecolor="black")
    ax9.set_ylabel("Peak Queue Length", fontsize=10)
    ax9.set_title("Peak Queue Size (lower is better)", fontsize=11, fontweight="bold")
    for bar, val in zip(bars, peak_queues):
        ax9.text(bar.get_x() + bar.get_width() / 2, bar.get_height() + 5, f"{val:.0f}", ha="center",
fontsize=10, fontweight="bold")

    ax10 = axes4[1, 1]
    ax10.axis("off")

    table_data = [
        ["Metric", "Fuzzy/Reg", "Fixed", "Queue"],
        ["Total Throughput", f"{fuzzy['total_throughput']:,}", f"{fixed['total_throughput']:,}",
f"{queue['total_throughput']:,}"],
        ["Avg Green Time", f"{fuzzy['avg_green_time']:.1f}s", f"{fixed['avg_green_time']:.1f}s",
f"{queue['avg_green_time']:.1f}s"],
        ["Avg Queue", f"{np.mean(fuzzy['queue_evolution']):.0f}",
f"{np.mean(fixed['queue_evolution']):.0f}", f"{np.mean(queue['queue_evolution']):.0f}"],
        ["Peak Queue", f"{max(fuzzy['queue_evolution'])}", f"{max(fixed['queue_evolution'])}",
f"{max(queue['queue_evolution'])}"],
        ["vs Fixed", f"{improvements[0]:+.1f}%", "-", f"{improvements[2]:+.1f}%"],
    ]

    table = ax10.table(cellText=table_data, loc="center", cellLoc="center", colWidths=[0.3, 0.23,
0.23, 0.23])
    table.auto_set_font_size(False)
    table.set_fontsize(11)
    table.scale(1.2, 1.8)

    # Header styling
    for i in range(4):
        table[(0, i)].set_facecolor("#34495e")
        table[(0, i)].set_text_props(color="white", fontweight="bold")

    # Μικρό highlight (ενδεικτικό)
    table[(1, 1)].set_facecolor("#d5f4e6")
    ax10.set_title("Summary Statistics", fontsize=11, fontweight="bold", pad=20)

    plt.tight_layout()
    fig4.savefig("chart4_performance_summary.png", dpi=150, bbox_inches="tight")
    plt.close(fig4)

    # === COMBINED OVERVIEW ===
    fig_all, axes_all = plt.subplots(2, 2, figsize=(16, 12))
    fig_all.suptitle(f"Traffic Control System Comparison (Seed: {seed})", fontsize=16,
fontweight="bold")
```

```
    ax_a = axes_all[0, 0]
    bars = ax_a.bar(systems, throughputs, color=colors, edgecolor="black", linewidth=1.2)
    ax_a.set_ylabel("Total Throughput (vehicles)")
    ax_a.set_title("Total Throughput", fontweight="bold")
    ax_a.set_ylim(0, max(throughputs) * 1.15)
    for bar, val in zip(bars, throughputs):
        ax_a.text(bar.get_x() + bar.get_width() / 2, bar.get_height() + 200, f"{val:,}", ha="center",
fontweight="bold")

    ax_b = axes_all[0, 1]
    ax_b.plot(steps, fuzzy["queue_evolution"], "g-", linewidth=2, label="Fuzzy/Regression")
    ax_b.plot(steps, fixed["queue_evolution"], "b--", linewidth=2, label="Baseline Fixed")
    ax_b.plot(steps, queue["queue_evolution"], "m:", linewidth=2, label="Baseline Queue")
    ax_b.set_xlabel("Time Step")
    ax_b.set_ylabel("Total Queue Length")
    ax_b.set_title("Queue Evolution Over Time", fontweight="bold")
    ax_b.legend()
    ax_b.grid(True, alpha=0.3)

    ax_c = axes_all[1, 0]
    ax_c.bar(x - width, fuzzy_vals, width, label="Fuzzy/Regression", color="#2ecc71",
edgecolor="black")
    ax_c.bar(x, fixed_vals, width, label="Baseline Fixed", color="#3498db", edgecolor="black")
    ax_c.bar(x + width, queue_vals, width, label="Baseline Queue", color="#9b59b6",
edgecolor="black")
    ax_c.set_ylabel("Throughput (vehicles)")
    ax_c.set_title("Throughput per Intersection", fontweight="bold")
    ax_c.set_xticks(x)
    ax_c.set_xticklabels(junctions, rotation=15)
    ax_c.legend()

    ax_d = axes_all[1, 1]
    bar_colors = ["#2ecc71" if x >= 0 else "#e74c3c" for x in improvements]
    bars = ax_d.bar(systems, improvements, color=bar_colors, edgecolor="black")
    ax_d.axhline(y=0, color="black", linewidth=1)
    ax_d.set_ylabel("Improvement vs Fixed Baseline (%)")
    ax_d.set_title("Performance vs Baseline", fontweight="bold")
    for bar, val in zip(bars, improvements):
        ypos = bar.get_height() + 0.2 if val >= 0 else bar.get_height() - 0.5
        ax_d.text(bar.get_x() + bar.get_width() / 2, ypos, f"{val:+.1f}%", ha="center",
fontweight="bold")

    plt.tight_layout()
    fig_all.savefig("comparison_results.png", dpi=150, bbox_inches="tight")
    plt.close(fig_all)

    print("\nCharts saved:")
    print("  - comparison_results.png (combined overview)")
    print("  - chart1_throughput.png")
    print("  - chart2_queue_evolution.png")
    print("  - chart3_green_time_distribution.png")
    print("  - chart4_performance_summary.png")

if __name__ == "__main__":
    import argparse

    parser = argparse.ArgumentParser(description="Σύγκριση συστημάτων ελέγχου κυκλοφορίας")
    parser.add_argument("--seed", type=int, default=42, help="Random seed για
```

```
αναπαραγωγιμότητα")
    args = parser.parse_args()

    run_comparison(args.seed)
```

ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] L. A. Zadeh, “Fuzzy Sets,” *Information and Control*, vol. 8, no. 3, pp. 338–353, 1965.

[2] C. C. Pappis and E. H. Mamdani, “A Fuzzy Logic Controller for a Traffic Junction,” *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 7, no. 10, pp. 707–717, 1977.

[3] S. Chiu, “Adaptive Traffic Signal Control Using Fuzzy Logic,” *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics*, pp. 312–317, 1992.

[4] M. S. Papageorgiou, “Traffic Signal Control,” in *Concise Encyclopedia of Traffic & Transportation Systems*, Pergamon Press, 1991.

[5] R. C. K. Wong and B. J. F. van Zuylen, “Intelligent Traffic Signal Control by Fuzzy Logic,” *Transportation Research Record*, no. 1727, pp. 1–8, 2000.

[6] A. Khamis and M. Gomaa, “Adaptive Multi-Objective Reinforcement Learning with Fuzzy Inference for Traffic Signal Control,” *Expert Systems with Applications*, vol. 42, no. 9, pp. 4488–4498, 2015.

[7] J. Liu, Y. Wang, and L. Li, “A Survey of Intelligent Traffic Signal Control,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 2, pp. 1–16, 2022.

[8] H. Wei et al., “CityFlow: A Multi-Agent Reinforcement Learning Environment for Large-Scale City Traffic Scenario,” *Proceedings of the World Wide Web Conference (WWW)*, pp. 3620–3624, 2019.

[9] D. Krajzewicz et al., “Recent Development and Applications of SUMO – Simulation of Urban MObility,” *International Journal on Advances in Systems and Measurements*, vol. 5, no. 3–4, pp. 128–138, 2012.

[10] Eclipse Foundation, “SUMO – Simulation of Urban Mobility,” [Online]. Available: https://eclipse.dev/sumo/

[11] CityFlow Project, “CityFlow: Traffic Simulator for Reinforcement Learning,” [Online]. Available: https://cityflow-project.github.io/

[12] P. Webster, “Traffic Signal Settings,” *Road Research Technical Paper No. 39*, Road Research Laboratory, London, 1958.

[13] S. Sharma and A. Jain, “Review of Traffic Signal Control Based on Fuzzy Logic,” *International Journal of Computer Applications*, vol. 145, no. 13, pp. 25–29, 2016.

[14] A. K. Jain et al., “Design of Fuzzy Logic Traffic Controller for Isolated Intersections,” *arXiv preprint*, arXiv:1405.0936, 2014.