ΔΙΕΘΝΕΣ
ΠΑΝΕΠΙΣΤΗΜΙΟ
ΤΗΣ ΕΛΛΑΔΟΣ

ΣΧΟΛΉ ΜΗΧΑΝΙΚΏΝ

ΤΜΉΜΑ ΜΗΧΑΝΙΚΏΝ ΠΛΗΡΟΦΟΡΙΚΉΣ ΚΑΙ ΗΛΕΚΤΡΟΝΙΚΏΝ ΣΥΣΤΗΜΆΤΩΝ

ΠΤΥΧΙΑΚΉ ΕΡΓΑΣΊΑ

Γραφική Επίλυση του Προβλήματος του Περιοδεύοντος Πωλητή με τη χρήση Νευρωνικών Δικτύων τύπου GNN (Graph Neural Networks)

Του φοιτητή
Σεβδάς Σωτήριος
Αρ. Μητρώου: 164739

Επιβλέπων Γεώργιος Κοκκώνης
Βαθμίδα Επίκουρος Καθηγητής

Θεσσαλονίκη 2023

Τίτλος Π.Ε. Γραφική Επίλυση του Προβλήματος του Περιοδεύοντος Πωλητή με τη χρήση Νευρωνικών Δικτύων τύπου GNN
Κωδικός Π.Ε. : 23142
Ονοματεπώνυμο φοιτητή. Σεβδάς Σωτήριος
Ονοματεπώνυμο εισηγητή. Γεώργιος Κοκκώνης
Ημερομηνία ανάληψης Π.Ε. 23-03-2023
Ημερομηνία περάτωσης Π.Ε.

Βεβαιώνω ότι είμαι ο συγγραφέας αυτής της εργασίας και ότι κάθε βοήθεια την οποία είχα για την προετοιμασία της είναι πλήρως αναγνωρισμένη και αναφέρεται στην εργασία. Επίσης, έχω καταγράψει τις όποιες πηγές από τις οποίες έκανα χρήση δεδομένων, ιδεών, εικόνων και κειμένου, είτε αυτές αναφέρονται ακριβώς είτε παραφρασμένες. Επιπλέον, βεβαιώνω ότι αυτή η εργασία προετοιμάστηκε από εμένα προσωπικά, ειδικά ως πτυχιακή εργασία, στο Τμήμα Μηχανικών Πληροφορικής και Ηλεκτρονικών Συστημάτων του ΔΙ.ΠΑ.Ε.

Η έγκριση της πτυχιακής εργασίας από το Τμήμα Μηχανικών Πληροφορικής και Ηλεκτρονικών Συστημάτων του Διεθνούς Πανεπιστημίου της Ελλάδος, δεν υποδηλώνει απαραιτήτως και αποδοχή των απόψεων του συγγραφέα, εκ μέρους του Τμήματος.

Πρόλογος

Η ανάπτυξη της μηχανικής μάθησης έχει επιφέρει κερδοφόρους καρπούς στα κομμάτια της επιστήμης και του αυτοματοποιήσου. Προγράμματα όπως αναγνώριση εικόνας η τα ChatGPT βοηθάν καθημερινά έναν απλό χρήστη, μπαίνοντας σε πιο σύνθετα προβλήματα η μηχανική μάθηση και τα νευρωνικά δίκτυα δεν αποφεύγουν να κάνουν την παρουσία τους αισθητή. Από μοριακές ενώσεις μέχρι και στην δημιουργία φαρμάκων τα νευρωνικά δίκτυα έχουν ξεκινήσει να εφαρμόζονται ολοένα και περισσότερο.

Η παρούσα έρευνα επικεντρώνεται σε μια συγκεκριμένη τεχνολογία στα νευρωνικά δίκτυα, τα Γραφικά Νευρωνικά Δίκτυα η αλλιώς Graph Neural Network (GNN) είναι μια καινοτόμα τεχνολογία που έχει ραγδαία ανάπτυξη πάνω στο τομέα της επιστήμης, των μαθηματικών και των κοινωνικών δικτύων.

Πιο συγκεκριμένα θα αναλυθεί πως τα GNN’s μπορούν να επιλύσουν το κλασικό μαθηματικό πρόβλημα του Περιοδεύοντος Πωλητή η αλλιώς Travelling Salesman Problem (TSP).

Περίληψη

Αντικείμενο της παρακάτω πτυχιακής εργασίας είναι η ανάλυση του Προβλήματος του Περιοδεύοντος Πωλητή (Travelling Salesman Problem, TSP) και η βιβλιογραφική παρουσίαση όλων των λύσεων του Προβλήματος που έχουν προταθεί μέχρι στιγμής με τη χρήση των Γραφικών Νευρωνικών Δικτύων (Graph Neural Network η GNN). Αυτά που θέλουμε να επιτύχουμε με αυτή την πτυχιακή εργασία είναι τα εξής, πρώτον θα θέλαμε να κατανοηθούν πλήρως τα GNN σαν μια νέα τεχνολογία των νευρωνικών δικτύων, δεύτερον είναι η ανάλυση των μέχρι τώρα επιλύσεων στο πρόβλημα TSP και τρίτων να εκπονήσουμε μια από αυτές τις επίλυσης.

Abstract

The subject of the following thesis is the analysis of the Traveling Salesman Problem (TSP) and the literature review of all the solutions proposed for the problem so far using Graph Neural Networks (GNN). The objectives we aim to achieve with this thesis are as follows: firstly, we want to fully comprehend GNN as a new technology in neural networks; secondly, we will analyze the existing solutions to the TSP and thirdly, we will develop one of these solutions.

Περιεχόμενα

Κατάλογος Σχημάτων

Κατάλογος Πινάκων

Συντομογραφίες

ΔΙΠΑΕ	Διεθνές Πανεπιστήμιο Ελλάδος
Π.Ε.	Πτυχιακή Εργασία
TSP	Travelling Salesman Problem
GNN	Graph Neural Network
Gat	Graph Attention Network
OR	Operations Research
MLP	Multilayer Perceptron
CNN	Convolutional Neural Networks
ΚΤΑ	Καθοδηγούμενη Τοπική Αναζήτηση

Κεφάλαιο 1: Εισαγωγή

Τα νευρωνικά δίκτυα θεωρείται μια από της πιο διαδεδομένες μεθόδους προγραμματισμού τα τελευταία χρόνια. Το γεγονός αυτό αποδόθηκε γιατί τα νευρωνικά δίκτυα έχουν την δυνατότητα να λύνουν σύνθετα προβλήματα χωρίς ο προγραμματιστής να προσφέρει στο πρόγραμμα τον τρόπο επίλυσης. Μια τεχνολογία που έχει αναπτυχθεί τα τελευταία χρόνια πάνω στον τομέα τον νευρωνικών δικτύων είναι τα Γραφικά Νευρωνικά Δίκτυα η αλλιώς και Graph Neural Network (GNN), για να μπορέσουμε να εμβαθύνουμε πάνω στα GNN θα πρέπει πρώτα να κατανοήσουμε την λογική και τον τρόπο λειτουργίας τον Νευρωνικών δικτύων ξεχωριστά. Σε αυτό το κεφάλαιο της πτυχιακής θα κάνουμε μια σύντομη αναφορά στη μηχανική μάθηση και στα νευρωνικά δίκτυα έτσι ώστε να γίνει πιο κατανοητή αυτή η Π.Ε και σε αναγνώστες που δεν έχουν ξανά μελετήσει πάνω στα Νευρωνικά Δίκτυα. Συνεχίζοντας στο 2ο κεφάλαιο θα μιλήσουμε για τα GNN's, πιο συγκεκριμένα θα μιλήσουμε για των γενικό σχεδιασμό των GNN's για τον διαχωρισμό σε Κατευθυνόμενα/Μη Κατευθυνόμενα Γραφήματα, Ομογενείς/Ετερογενείς Γραφήματα και Στατικά/Δυναμικά Γραφήματα και άλλα χρήσιμα στοιχεία για να κατανοήσουμε καλύτερα την λογική των GNN. Στο 3ο κεφάλαιο θα κάνουμε μια σύντομη αναφορά πάνω στο ευρέα διαδεδομένο TSP, θα αναφερθούμε στην σύνθεση του προβλήματος τους αλγόριθμους που χρησιμοποιεί και τέλος τις εφαρμογές που έχουν υιοθέτηση αυτό το πρόβλημα σαν βασικό παράγοντα για την επίλυση πιο σύνθετων προβλημάτων. Στο 4ο κεφάλαιο και αφού έχουμε ξεκαθαρίσει τον σκοπό της Π.Ε θα μιλήσουμε και θα περιγράψουμε τις βιβλιογραφικές παρουσίασης των λύσεων του Προβλήματος Του Περιοδεύοντος Πωλητή που έχουν προταθεί με τη χρήση των GNN. Στο 5ο κεφάλαιο αυτής της εργασίας θα μιλήσουμε για την δική μου προσωπική απόπειρα επίλυσης του προβλήματος χρησιμοποιώντας μια από τις υπάρχουσες προαναφερόμενες μεθόδους ως βάση για το δικό μου πρόγραμμα. Στο 6ο και τελευταίο κεφάλαιο θα γίνει μια έρευνα, πάνω στα αποτελέσματα και στο χρόνο εκτέλεσης του προγράμματος σε σχέση με άλλους τρόπους επίλυσης του TSP, στη συνέχεια ένας σύντομος επίλογος και αναφορά για μελλοντικές εργασίες.

1.1 Νευρωνικά Δίκτυα

Τα Νευρωνικά Δίκτυα πήραν την ονομασία τους επειδή είναι ένα δίκτυο συνδεδεμένων στοιχείων [1]. Αυτά τα στοιχεία εμπνεύστηκαν από μελέτες των βιολογικών νευρικών συστημάτων. Με άλλα λόγια, τα νευρωνικά δίκτυα είναι μια προσπάθεια δημιουργίας μηχανών που λειτουργούν με παρόμοιο τρόπο με το ανθρώπινο εγκέφαλο, χρησιμοποιώντας συστατικά που συμπεριφέρονται όπως βιολογικά οι νευρώνες. Η λειτουργία ενός νευρωνικού δικτύου είναι να παράγει ένα πρότυπο εξόδου όταν του παρουσιάζεται ένα πρότυπο εισόδου. Η ταξινόμηση προτύπων είναι η διαδικασία ταξινόμησης των προτύπων σε μια ομάδα ή μια άλλη.

Η ταξινόμηση προτύπων στα νευρωνικά δίκτυα βασίζεται σε δύο σημαντικά χαρακτηριστικά τους που είναι η ικανότητά τους να μάθουν και να γενικεύουν. Καθώς τα νευρωνικά δίκτυα εξελίσσονται και μαθαίνουν από τα δεδομένα, παρέχουν τη δυνατότητα αντιμετώπισης προβλημάτων που προτεινόμενοι αλγόριθμοι δεν μπορούν να αντιμετωπίσουν. Έτσι, τα νευρωνικά δίκτυα αναδεικνύονται ως μια αναπόσπαστη και συναρπαστική συνιστώσα στον κόσμο της τεχνητής νοημοσύνης

1.2 Μηχανική Μάθηση

Η μηχανική μάθηση είναι ένας κλάδος εξελισσόμενων υπολογιστικών αλγορίθμων που φτιάχνονται για να μιμηθούν την ανθρώπινη νοημοσύνη μαθαίνοντας από το περιβάλλον. Τεχνικές που βασίζονται στη μηχανική μάθηση έχουν εφαρμοστεί με επιτυχία σε ποικίλους τομείς, από την αναγνώριση προτύπων, την υπολογιστική όραση, την τεχνολογία διαστημικών σκαφών, την ψυχαγωγία και την υπολογιστική βιολογία έως τις βιοϊατρικές και ιατρικές εφαρμογές [2].

Ένας αλγόριθμος μηχανικής μάθησης είναι μια υπολογιστική διαδικασία που λειτουργεί μέσω εισροές δεδομένων για να πετύχει μια επιθυμητή εργασία χωρίς να είναι κυριολεκτικά προγραμματισμένος (δηλαδή "σκληρός κωδικοποιημένος") για να παράγει ένα συγκεκριμένο αποτέλεσμα. Αυτοί οι αλγόριθμοι είναι σε έναν ευρύτερο βαθμό "μαλακά κωδικοποιημένοι", καθώς αλλάζουν ή προσαρμόζουν αυτόματα την αρχιτεκτονική τους μέσω επανάληψης (δηλαδή εμπειρίας), ώστε να γίνονται ολοένα και καλύτεροι στην επίτευξη της επιθυμητής εργασίας. Η διαδικασία προσαρμογής ονομάζεται εκπαίδευση, στην οποία παρέχονται δείγματα εισροής δεδομένων μαζί με τα επιθυμητά αποτελέσματα. Ο αλγόριθμος προσαρμόζει την αρχιτεκτονική του έτσι ώστε να μπορεί όχι μόνο να παράγει το επιθυμητό αποτέλεσμα όταν του παρουσιάζονται οι εκπαιδευτικές εισροές, αλλά να γενικεύει και να παράγει το επιθυμητό αποτέλεσμα από νέα, προηγουμένως αόρατα δεδομένα. Αυτή η εκπαίδευση αποτελεί το "μάθημα" της μηχανικής μάθησης. Η εκπαίδευση δεν χρειάζεται να περιορίζεται σε μια αρχική προσαρμογή κατά τη διάρκεια ενός πεπερασμένου χρονικού διαστήματος. Όπως και με τους ανθρώπους, ένας καλός αλγόριθμος μπορεί να εξασκηθεί σε "διαρκή" μάθηση καθώς επεξεργάζεται νέα δεδομένα και μαθαίνει από τα λάθη του.

Κεφάλαιο 2: GNN

2.1 Εισαγωγή

Τα γραφήματα είναι ένα είδος δεδομένων που μοντελοποιεί ένα σύνολο αντικειμένων (Nodes) και τις σχέσεις μεταξύ τους (edges). Πρόσφατα, έρευνες στον τομέα γραφημάτων με χρήση μηχανικής μάθησης έχουν δεχτή μεγάλο ενδιαφέρον και προσοχή εξαιτίας των δυνατοτήτων που έχουν τα γραφήματα. Μια από αυτές είναι η χρηστικότητα τους σε διαφόρους τομείς της επιστήμης, παράδειγμα στον τομέα των κοινωνικών επιστημών, των φυσικών επιστημών και κ.ο.κ. Ως μια ξεχωριστή μη ευκλείδεια δομή δεδομένων για μηχανική μάθηση, η ανάλυση γραφημάτων εστιάζει σε εργασίες όπως ταξινόμηση κόμβων, πρόβλεψη συνδέσμων και ομαδοποίηση. Το Graph Neural Network (GNN) είναι βασισμένο στη βαθιά μάθηση (deep learning) που λειτουργεί το τομέα των γραφημάτων. Λόγο της αποτελεσματικότητας του το GNN έχει ευρέος εφαρμοστή στις μεθόδους ανάλυσης γραφημάτων. Συνεχίζοντας θα ήθελα να κάνουμε μια γρήγορη αναφορά σε κάποια από τα πλεονεκτήματα που προσφέρουν τα Graph neural networks.

Το πρώτο ευθύνεται στην μακροχρόνια ιστορία των Graph Neural Networks. Στη δεκαετία το ενενήντα, Recursive Neural Networks ήταν τα πρώτα που χρησιμοποιήθηκαν για Directed Acyclic Graphs (Sperduit and Starita, 1997; Frasconi κ.α., 1998) [3] [4]. Στη συνέχεια, τα επαναλαμβανόμενα νευρωνικά δίκτυα και Τα Feedforward Neural Networks εισάγονται σε αυτή τη βιβλιογραφία αντίστοιχα (Scarselli κ.α., 2009) [5] και (Micheli, 2009) [6] για την αντιμετώπιση κύκλων. Αν και επιτυχής, η γενική ιδέα πίσω από τέτοιες μεθόδους φτιάχνει συστήματα μετάβασης και επαναλαμβάνεται μέχρι τη σύγκλιση, αυτό το γεγονός περιόρισε την επεκτασιμότητα και την ικανότητα αναπαράστασης. Η πρόσφατες αναβαθμίσεις των Deep Neural Networks, πιο συγκεκριμένα των Convolutional Neural Networks (CNNs) είχαν ως αποτέλεσμα την εκ νέου ανακάλυψη GNNs.

Τα εργαλεία του CNN είναι τοπική σύνδεση, κοινά βάρη και χρήση πολλαπλών επιπέδων (multi-layers). Αυτά είναι πολύ βασικά για την επίλυση προβλημάτων σε γραφήματα. Ωστόσο, τα CNN έχουν την ικανότητα και λειτουργούν και σε απλά Ευκλείδεια δεδομένα όπως εικόνες (2D ακολουθίες) και κείμενα (1D ακολουθίες) ενώ αυτές οι δομές δεδομένων μπορούν να θεωρηθούν ως περιπτώσεις γραφημάτων. Επομένως είναι εύκολο να γενικεύσεις τα CNN σε γραφήματα. Όμως είναι δύσκολο να καθοριστούν τοπικά συλλεκτικά φίλτρα και τελεστές συγκέντρωσης, γεγονός που εμποδίζει τη μετατροπή του CNN από Ευκλείδειο τομέα σε μη-Ευκλείδειο τομέα.

Επεκτείνοντας τα μοντέλα deep neural σε μη-Ευκλειδιους τομής, όπου είναι γενικά αναφερόμενη ως γεωμετρική βαθιά μάθηση, έχουν γίνει ένας αναδυόμενος ερευνητικός τομέας. Σύμφωνα με αυτό ο όρος ομπρέλα, βαθιά μάθηση σε γραφήματα λαμβάνει τεράστια προσοχή. Ένας άλλος προέρχεται από το Graph Representaition Learning, οπού μαθαίνει να αναπαριστά κόμβους γραφημάτων, άκρες ή υπό γραφήματα με διανύσματα χαμηλής διάστασης. Στα πεδία της ανάλυσης γραφημάτων, κατά κόρων η απόπειρες μηχανικής μάθησης βασίζονται στο να είναι κατασκευασμένα στο χέρι χαρακτηριστικά και περιορίζονται από το υψηλό κόστος του.

Έχοντας ως γνώμονα την ιδέα της αναπαράστασης μάθησης και την επιτυχία της ενσωμάτωσης λέξεων (Mikolov T., 2013) [7], το DeepWalk (Perozzi B., 2014) [8], που θεωρείται ως η πρώτη μέθοδος ενσωμάτωσης γραφημάτων που βασίζεται στην αναπαράσταση μάθησης, χρησιμοποιεί το μοντέλο SkipGram [7][1] στις παραγόμενες τυχαίες περιπάτους.

Παρόμοιες προσεγγίσεις όπως το node2vec (Grover A.,) [9], το LINE [10] και το TADW (Yang κ.α., 2015) [11] επίσης έχουν κατορθώσει σημαντικά αποτελέσματα. Ωστόσο, αυτές οι μέθοδοι αντιμετωπίζουν δύο σημαντικά μειονεκτήματα (Hamilton κ.α., 2017b) [12]. Πρώτον, δεν διαθέτουν κοινόχρηστες παράμετροι μεταξύ των κόμβων στον κομμάτι της κωδικοποιήσεις, ως αποτέλεσμα οδηγεί σε υπολογιστική ανικανότητα, καθώς σημαίνει ότι ο αριθμός των παραμέτρων αυξάνεται

γραμμικά με τον αριθμό των κόμβων. Δεύτερο μειονέκτημα είναι οι απευθείας μέθοδοι ενσωμάτωσης λείπουν από την ικανότητα γενίκευσης, πράγμα που σημαίνει ότι δεν έχουν την δυνατότητα να αντιμετωπίσουν δυναμικά γραφήματα ή να γενικευτούν σε νέα γραφήματα.

Βασιζόμενοι στα Convolutional Neural Network (CNN) και στην ενσωμάτωση γραφημάτων, προτείνονται παραλλαγές των γραφικών νευρωνικών δικτύων (GNNs) που συλλέγουν συλλογικά πληροφορίες από τη δομή του γραφήματος. Έτσι, κατορθώνουν να μοντελοποιήσουν εισόδους και/ή εξόδους που αποτελούνται από στοιχεία και την εξάρτησή τους. Υπάρχουν πολλές εκτενείς ανασκοπήσεις για γραφικά νευρωνικά δίκτυα. Ο Bronstein κ.α. (2017) [13] προσφέρει μια εκτενή ανασκόπηση της γεωμετρικής βαθιάς μάθησης, η οποία παρουσιάζει τα προβλήματά της, τις δυσκολίες, τις λύσεις, τις εφαρμογές της και τις μελλοντικές κατευθύνσεις. Οι Zhang κ.α. (2019a) [14] συνιστούν μια άλλη εκτενή επισκόπηση των graph convolutional networks. Ωστόσο, δίνουν βάση κυρίως στους τελεστές convolution που ορίζονται σε γραφήματα σε αντίθεση με αυτό το κεφάλαιο που εξετάζονται άλλες μονάδες υπολογισμού στα GNNs, όπως οι συνδέσεις παράλειψης και οι τελεστές συγκέντρωσης.

Οι Wu κ.α. (2020) [15] κατηγοριοποιούν τα GNNs σε τέσσερις ομάδες: επαναληπτικά γραφικά νευρωνικά δίκτυα, convolutional γραφικά νευρωνικά δίκτυα, αυτό-διακοπτόμενα γραφικά νευρωνικά δίκτυα και χώρο χρονικά γραφικά νευρωνικά δίκτυα. Σε αυτό το κεφάλαιο θα επικεντρωθούμε κυρίως στα κλασικά μοντέλα GNN.

2.2 Γενικός Σχεδιασμός Αγωγών GNN

Σε αυτήν την ενότητα, θα αναφερθούμε στον γενικό σχεδιασμό αρχιτεκτονικής ενός μοντέλου GNN για μια συγκεκριμένη εργασία σε έναν συγκεκριμένο τύπο γραφήματος. Γενικά, το κομμάτι του σχεδιασμού περιλαμβάνει τέσσερα βήματα [16]: (1) εύρεση της δομής του γραφήματος, (2) καθορισμός του τύπου και της κλίμακας του γραφήματος, (3) σχεδιασμός της συνάρτησης απώλειας και (4) κατασκευή του μοντέλου χρησιμοποιώντας υπολογιστικές μονάδες. Σε αυτήν την ενότητα θα παρουσιαστούν γενικές αρχές σχεδιασμού και κάποιες βασικές γνώσεις.

2.2.1 Εύρεση Της Δομής Του Γραφήματος

Αρχικά, θα πρέπει να βρεθεί η δομή του γραφήματος στην εφαρμογή. Συνήθως υπάρχουν δύο εκδοχές: δομικά σενάρια και μη δομικά σενάρια. Στα δομικά σενάρια, η δομή του γραφήματος είναι φανερή στις εφαρμογές, όπως εφαρμογές σε μοριακά συστήματα, γνωστικά γραφήματα και άλλα. Στα μη δομικά σενάρια, το γράφημα είναι ασαφής, ως αποτέλεσμα πρέπει πρώτα να κατασκευαστεί το γράφημα από την εργασία, όπως την κατασκευή ενός πλήρως συνδεδεμένου γραφήματος "λέξεων" για κείμενο ή την κατασκευή ενός γραφήματος σκηνής για μια εικόνα. Αφού λάβουμε το γράφημα, η αργότερη διαδικασία σχεδίασης προσπαθεί να βρει ένα βέλτιστο μοντέλο GNN για αυτόν τον συγκεκριμένο γράφημα.

2.2.2 Καθορισμός Τύπου Και Κλίμακας Γραφήματος

Αφού έχει ληφθεί το γράφημα στην εφαρμογή, τότε πρέπει να καθοριστεί ο τύπος και η κλίμακα του γραφήματος.

Τα γραφήματα με πολύπλοκους τύπους μπορούν να παρέχουν περισσότερες πληροφορίες σχετικά με τους κόμβους και τις συνδέσεις τους. Τα γραφήματα κατηγοριοποιούνται συνήθως ως:

- **Κατευθυνόμενα/Μη Κατευθυνόμενα Γραφήματα.** Οι ακμές στα κατευθυνόμενα γραφήματα κατευθύνονται από έναν κόμβο σε έναν άλλο, παρέχοντας περισσότερες πληροφορίες από ό,τι στους μη κατευθυνόμενους γραφήματα. Κάθε ακμή στα μη κατευθυνόμενα γραφήματα μπορεί επίσης να θεωρηθεί ως δύο κατευθυνόμενες ακμές.
- **Ομογενείς/Ετερογενείς Γραφήματα.** Οι κόμβοι και οι ακμές στα ομογενείς γραφήματα έχουν τους ίδιους τύπους, ενώ οι κόμβοι και οι ακμές έχουν διαφορετικούς τύπους στους ετερογενείς γραφήματα. Οι τύποι των κόμβων και των ακμών παίζουν σημαντικό ρόλο στα ετερογενείς γραφήματα και πρέπει να ληφθούν υπόψη περαιτέρω.
- **Στατικά/Δυναμικά Γραφήματα.** Όταν τα χαρακτηριστικά εισόδου ή η τοπολογία του γραφήματος ποικίλλουν με τον χρόνο, το γράφημα θεωρείται ως δυναμικό γράφημα. Οι πληροφορίες χρόνου πρέπει να λαμβάνονται υπόψη με προσοχή στους δυναμικούς γραφήματα.

Να σημειωθεί ότι αυτές οι κατηγορίες είναι ορθογώνιες, γεγονός που σημαίνει ότι αυτοί οι τύποι μπορούν να συνδυαστούν, για παράδειγμα κάποιος μπορεί να αντιμετωπίσει έναν δυναμικό κατευθυνόμενο ετερογενή γράφημα. Υπάρχουν εκτός αυτών και άλλοι τύποι γραφημάτων που έχουν σχεδιαστεί για διάφορες εργασίες, όπως υπέρ γραφήματα και εναλλασσόμενα γραφήματα. Δεν θα γραφούν όλοι οι τύποι εδώ, αλλά το πιο σημαντικό είναι να ληφθεί υπόψη η πρόσθετη πληροφορία που παρέχουν αυτά τα γραφήματα. Όταν καθοριστεί ο τύπος του γραφήματος, η πρόσθετη πληροφορία που παρέχεται από αυτούς τους τύπους γραφημάτων πρέπει να ληφθεί υπόψη στη διαδικασία σχεδίασης.

Όσον αφορά την κλίμακα του γραφήματος, δεν υπάρχει σαφής κατηγοριοποίηση για το μέγεθος στα γραφήματα, δηλαδή "μικρούς" και "μεγάλους". Οι κατηγοριοποίησης κριτηρίων εξακολουθεί να αλλάζει με την ανάπτυξη των υπολογιστικών συσκευών (όπως η ταχύτητα και η μνήμη των GPU). Σε σενάρια τα οποία ο πίνακας γειτνίασης ή ο γραφικός πίνακας Laplacian ενός γραφήματος (η χωρική πολυπλοκότητα είναι O(n^2)) δεν μπορεί να αποθηκευτεί και να επεξεργαστεί από τη συσκευή, τότε θεωρούμε το γράφημα ως μεγάλης κλίμακας γράφημα και θα πρέπει να ληφθούν υπόψη ορισμένες μέθοδοι δειγματοληψίας.

2.2.3 Σχεδίαση Συνάρτησης Απώλειας

Σε αυτήν την ενότητα θα αναφερθούμε για το πως σχεδιάζεται η συνάρτηση απώλειας με βάσει του τύπου εργασίας που έχει το νευρωνικό δίκτυο και των ρυθμίσεων εκπαίδευσης. Για τις εργασίες μάθησης γραφημάτων, υπάρχουν συνήθως τρία είδη εργασιών:

- **Οι εργασίες σε επίπεδο κόμβου** έχουν ως σκοπό να επικεντρωθούν στους κόμβους και περιλαμβάνουν την ταξινόμηση κόμβων, την παλινδρόμηση κόμβων, την ομαδοποίηση κόμβων κ.λ.π. Η ταξινόμηση κόμβων προσπαθεί να κατηγοριοποιήσει τους κόμβους σε διάφορες κλάσεις, ενώ η παλινδρόμηση κόμβων προβλέπει μια συνεχή τιμή για κάθε κόμβο. Η ομαδοποίηση κόμβων έχει ως βασικό σκοπό τη διαμέριση των κόμβων σε αρκετές μη συνεκτικές ομάδες, όπου οι παρόμοιοι κόμβοι θα πρέπει να ανήκουν στην ίδια ομάδα.
- **Οι εργασίες σε επίπεδο ακμής** είναι η ταξινόμηση ακμών και η πρόβλεψη συνδέσμων, όπου απαιτείται από το μοντέλο να ταξινομήσει τους τύπους των ακμών ή να προβλέψει εάν υπάρχει μια ακμή μεταξύ δύο δοθέντων κόμβων.
- **Οι εργασίες σε επίπεδο γραφήματος** εμπεριέχουν την ταξινόμηση γραφημάτων, την παλινδρόμηση γραφημάτων και την αντιστοίχιση γραφημάτων, όλες από τις οποίες έχουν ως απαίτηση από το μοντέλο να μάθει αναπαραστάσεις γραφημάτων.

Όσον αφορά το κομμάτι της επίβλεψης, υπάρχει επίσης κατηγοριοποίηση εργασιών μάθησης γραφημάτων σε τρεις διαφορετικές ρυθμίσεις εκπαίδευσης:

- **Η επίβλεψη** παρέχει επισημασμένα δεδομένα για εκπαίδευση.
- **Η ημι-επιβλεπόμενη ρύθμιση** παρέχει μια μικρή ποσοστό επισημασμένων κόμβων και μια μεγάλη ποσότητα μη επισημασμένων κόμβων για εκπαίδευση. Κατά τη φάση του τεστ, η μεταφορική ρύθμιση απαιτεί από το μοντέλο να προβλέψει τις ετικέτες των δεδομένων χωρίς ετικέτες, ενώ η εισαγωγική ρύθμιση παρέχει νέους μη επισημασμένους κόμβους από την ίδια κατανομή για είσοδο. Οι περισσότερες εργασίες ταξινόμησης κόμβων και ακμών είναι ημι-επιβλεπόμενες.
- **Η μη επιβλέπον ρύθμιση** όπου παρέχει μόνο μη επισημασμένα δεδομένα για το μοντέλο να ανακαλύψει μοτίβα. Η ομαδοποίηση κόμβων είναι μια τυπική μη επιβλέπον εργασία μάθησης.

Έχοντας σε κατοχή τον τύπο της εργασίας και τη ρύθμιση εκπαίδευσης, μπορούμε να σχεδιάσουμε μια συγκεκριμένη συνάρτηση απώλειας για την εργασία. Για παράδειγμα, για μια ημι-επιβλεπόμενη κατηγοριοποίηση κόμβων σε επίπεδο κόμβου, η απώλεια της αναπόδραστης συνάρτησης (cross-entropy loss) μπορεί να χρησιμοποιηθεί για τους επισημασμένους κόμβους στο σύνολο εκπαίδευσης.

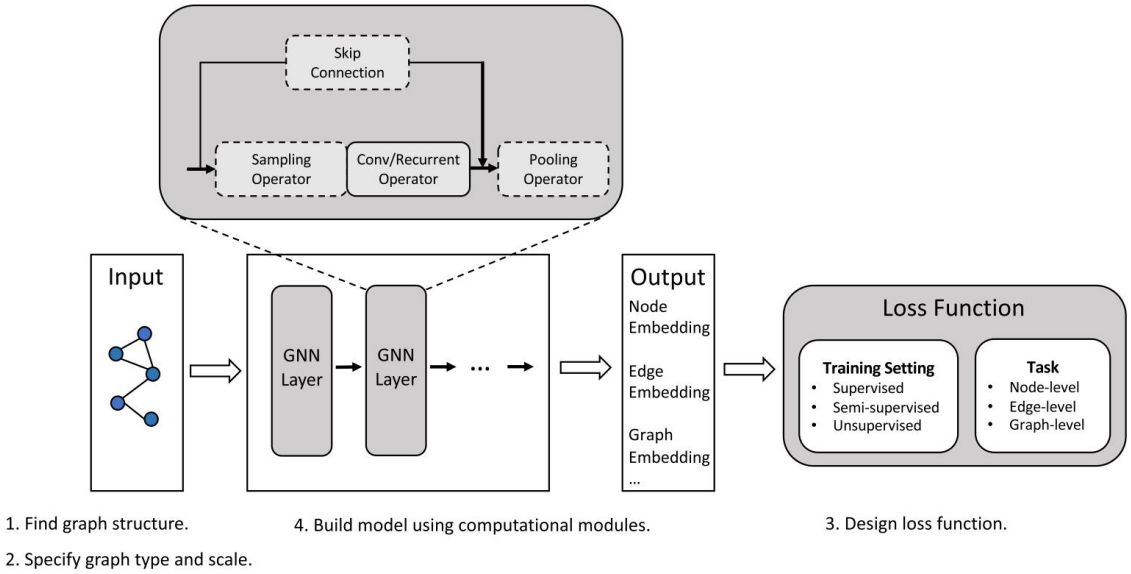
2.2.4 Κατασκευή μοντέλου χρησιμοποιώντας υπολογιστικά μοντέλα

Φτάνοντας σε αυτό το σημείο υπάρχει η δυνατότητα να κατασκευαστεί το μοντέλο χρησιμοποιώντας τα υπολογιστικά μοντέλα. Ορισμένα από τα συνήθως χρησιμοποιούμενα υπολογιστικά μοντέλα είναι:

- **Υπολογιστική Μονάδα Εξάπλωσης.** Η υπολογιστική μονάδα εξάπλωσης η χρήση της είναι να μεταδίδει πληροφορίες μεταξύ των κόμβων, έτσι ώστε οι συγκεντρωμένες πληροφορίες να προσφέρουν τόσο τα χαρακτηριστικά όσο και τις τοπογραφικές πληροφορίες. Στις υπολογιστικές μονάδες εξάπλωσης, οι πράξεις συσχέτισης και επαναληπτικής εξέλιξης, η χρήση τους είναι κατά κόρων για τη συγκέντρωση πληροφοριών από τους γείτονες, ενώ η πράξη σύνδεσης "skip connection" χρησιμοποιείται για τη συλλογή πληροφοριών από τις προηγούμενες αναπαραστάσεις των κόμβων και την αντιμετώπιση του προβλήματος υπερβολικής εξομάλυνσης.
- **Υπολογιστική Μονάδα Δειγματοληψίας.** Όταν τα γραφήματα είναι μεγάλα, συνήθως απαιτούνται μονάδες δειγματοληψίας για την εκτέλεση της εξάπλωσης στα γραφήματα. Η μονάδα δειγματοληψίας συνήθως συνδυάζεται με την υπολογιστική μονάδα εξάπλωσης.
- **Υπολογιστική Μονάδα Συγκέντρωσης.** Όταν χρειαζόμαστε τις αναπαραστάσεις υψηλού επιπέδου υπό γραφημάτων ή γραφημάτων, απαιτούνται μονάδες συγκέντρωσης για την εξαγωγή πληροφοριών από τους κόμβους.

Με τις υπολογιστικές μονάδες που αναφέρθηκαν, ένα τυπικό μοντέλο GNN συνήθως κατασκευάζεται συνδυάζοντάς τις. Μια τυπική αρχιτεκτονική του μοντέλου GNN απεικονίζεται στο κέντρο του Σχήματος 2.1, όπου οι πράξεις συλλογής, επανάληψης, δειγματοληψίας και σύνδεσης "skip connection" χρησιμοποιούνται για τη μετάδοση πληροφοριών σε κάθε μέρος, ενώ η μονάδα συγκέντρωσης διαθέτετε για την εξαγωγή πληροφοριών υψηλού επιπέδου. Αυτά τα στρώματα συνδυάζονται με σκοπό να αποκτήσουν καλύτερες αναπαραστάσεις. Σημαντικό να αναφερθεί είναι ότι αυτή η αρχιτεκτονική μπορεί να γενικευτεί για τα περισσότερα μοντέλα GNN, ενώ υπάρχουν επίσης εξαιρέσεις, για παράδειγμα το NDCN (Zang και Wang, 2020) [17] συνδυάζει συστήματα διαφορικών εξισώσεων και GNNs. Μπορεί να θεωρηθεί ως ένα GNN μοντέλο σε συνεχή χρόνο που ενσωματώνει τα στρώματα GNN σε συνεχή χρόνο χωρίς να προωθείται μέσω διακριτού αριθμού στρωμάτων.

Μια απεικόνιση γενικής διαδικασίας της σχεδίασης παρουσιάζεται στο Σχήμα 2.1.



Σχήμα 2.1: Ο Γενικός Σχεδιασμός Διασωλήνωσης για το μοντέλο GNN [16]

2.3 Παράδειγμα σχεδιασμού μοντέλου GNN

Σε αυτήν την ενότητα, περιγραφεί ένα υπάρχον μοντέλο GNN για να κατανοηθεί η διαδικασία σχεδιασμού. Για παραδείγματος χάριν, χρησιμοποιείται το GPT-GNN (Hu κ.α., 2020b) [18] ως το μοντέλο για να αναλυθεί η διαδικασία σχεδιασμού.

1. Εύρεση της δομής του γραφήματος. Στο άρθρο πάνω στο παράδειγμα που περιγράφεται, επικεντρώνεται σε εφαρμογές στο γνωστικό γράφημα της ακαδημαϊκής κοινότητας και το σύστημα προτάσεων. Στο γνωστικό γράφημα, η δομή του γραφήματος είναι σαφής. Στα συστήματα προτάσεων, οι χρήστες, τα αντικείμενα και οι κριτικές μπορούν να θεωρηθούν ως κόμβοι και οι αλληλεπιδράσεις μεταξύ τους μπορούν να θεωρηθούν ως ακμές, οπότε η δομή του γραφήματος είναι επίσης εύκολο να δημιουργηθεί.

2. Καθορισμός τύπου και κλίμακας του γραφήματος. Οι εργασίες επικεντρώνονται σε ανομοιογενείς γραφήματα, οπότε οι τύποι κόμβων και ακμών θα ήτανε σωστό να ληφθούν υπόψη και να ενσωματωθούν στο τελικό μοντέλο. Επειδή το ακαδημαϊκό γράφημα και το γράφημα προτάσεων περιέχουν εκατομμύρια κόμβους, το μοντέλο πρέπει επίσης να λάβει υπόψη το πρόβλημα της αποδοτικότητας. Συνολικά, το μοντέλο πρέπει να δώσει βάση σε μεγάλης κλίμακας ανομοιογενείς γραφήματα.

3. Σχεδιασμός συνάρτησης απώλειας. Διότι οι κατώτερες εργασίες στο (Hu κ.α., 2020b) [18] είναι εργασίες σε επίπεδο κόμβων (π.χ. πρόβλεψη άρθρου-πεδίου στο ακαδημαϊκό γράφημα), το μοντέλο θα πρέπει να μάθει τις αναπαραστάσεις των κόμβων στο βήμα πριν την εκπαίδευση. Στο βήμα πριν την εκπαίδευση, δεν υπάρχουν διαθέσιμα επισημασμένα δεδομένα, οπότε σχεδιάζεται μια αυτό-επιβλεπόμενη εργασία παραγωγής γραφημάτων για να μάθει τις ενσωματώσεις των κόμβων. Στο βήμα επανεκπαίδευσης, το μοντέλο ξανά εκπαιδεύεται βάσει

των εκπαιδευτικών δεδομένων κάθε εργασίας, επομένως εφαρμόζεται η επιβλεπόμενη απώλεια κάθε εργασίας.

4. Δημιουργία μοντέλου με χρήση υπολογιστικών μονάδων. Τέλος, το μοντέλο δημιουργείται χρησιμοποιώντας υπολογιστικές μονάδες. Για τη μονάδα διάδοσης, οι συντάκτες χρησιμοποιούν έναν τελεστή συσχέτισης HGT (Hu κ.α., 2020a) [19] όπου αναφέρθηκε προηγουμένως. Ο HGT προσθέτει τους τύπους των κόμβων και των ακμών στο βήμα διάδοσης του μοντέλου και προστίθεται επίσης μια σύνδεση παράκαμψης στην αρχιτεκτονική. Για τη μονάδα δειγματοληψίας, χρησιμοποιείται μια ειδικά σχεδιασμένη μέθοδος δειγματοληψίας HGSampling (Hu κ.α., 2020a) [19]. Καθώς το μοντέλο επικεντρώνεται στη μάθηση των αναπαραστάσεων των κόμβων, η μονάδα συγκέντρωσης δεν είναι απαραίτητη. Οι στρώσεις HGT επαναλαμβάνονται πολλαπλές φορές για να μάθουν καλύτερες αναπαραστάσεις κόμβων.

2.4 Εφαρμογές

Τα γραφικά νευρωνικά δίκτυα (GNN) έχουν εξερευνηθεί σε μια ευρεία γκάμα πεδίων στις ρυθμίσεις επιβλεπόμενης, ημι-επιβλεπόμενης, μη επιβλεπόμενης και ενισχυτικής μάθησης. Σε αυτήν την ενότητα, αφιερωθεί στην ομαδοποίηση των εφαρμογών σε δύο σενάρια: (1) Δομικά σενάρια όπου τα δεδομένα έχουν έκφραση δομημένων συσχετίσεων. Αυτά τα σενάρια, από τη μία πλευρά, προέρχονται από επιστημονικές έρευνες, όπως ο εντοπισμός γραφημάτων, η μοντελοποίηση φυσικών συστημάτων και χημικών συστημάτων. Από την άλλη πλευρά, αυτά προκύπτουν από βιομηχανικές εφαρμογές όπως γνωστικά γραφήματα, δίκτυα κυκλοφορίας και συστήματα συστάσεων. (2) Μη-δομικά σενάρια όπου η συσχέτιση δομής είναι ελάχιστη ή απουσιάζει. Αυτά τα σενάρια περιλαμβάνουν γενικά εικόνες (επεξεργασία εικόνας) και κείμενο (επεξεργασία φυσικής γλώσσας), τα οποία είναι δύο από τα πιο ενεργά αναπτυσσόμενα κλαδιά της έρευνας της τεχνητής νοημοσύνης.

2.4.1 Εξόρυξη Γραφημάτων

Η πρώτη εφαρμογή που θα γίνει αναφορά είναι η επίλυση βασικών εργασιών στην εξόρυξη γραφημάτων. Γενικά, οι αλγόριθμοι εξόρυξης γραφημάτων η χρήση τους βασίζεται για την αναγνώριση χρήσιμων δομών για κατώτερες εργασίες. Οι παραδοσιακές προκλήσεις εξόρυξης γραφημάτων περιλαμβάνουν την εξόρυξη συχνών υπό γραφημάτων, τη σύγκριση γραφημάτων, την ταξινόμηση γραφημάτων, την ομαδοποίηση γραφημάτων κ.λ.π. Αν και με τη βαθιά μάθηση, ορισμένες κατώτερες εργασίες μπορούν να επιλυθούν απευθείας χωρίς την εξόρυξη γραφημάτων ως μεσάζοντα, οι βασικές προκλήσεις αξίζουν να μελετηθούν από την προοπτική των GNNs.

Ταίριασμα γραφημάτων. Η πρώτη πρόκληση που θέτετε για επίλυση είναι το ταίριασμα γραφημάτων. Οι παραδοσιακές μέθοδοι για το ταίριασμα γραφημάτων συνήθως αντιμετωπίζουν υψηλή υπολογιστική πολυπλοκότητα ως βασικό πρόβλημα. Η εμφάνιση των GNNs προσφέρει την δυνατότητα στους ερευνητές να αναπαραστήσουν τη δομή των γραφημάτων χρησιμοποιώντας νευρωνικά δίκτυα, δίνοντας έτσι μια άλλη λύση στο πρόβλημα. Ο Riba κ.α. (2018) [20] προτείνει ένα μοντέλο siamese MPNN για τη μάθηση της απόστασης επεξεργασίας γραφημάτων. Το πλαίσιο siamese περιλαμβάνει δύο παράλληλα MPNNs με την ίδια δομή και κοινή χρήση βαρών. Ο στόχος της εκπαίδευσης είναι η ενσωμάτωση ενός ζεύγους γραφημάτων με μικρή απόσταση επεξεργασίας σε κοντινό χώρο χαρακτηριστικών.

Ομαδοποίηση γραφημάτων. Η ομαδοποίηση γραφημάτων, πιο αναλυτικά είναι η ομαδοποίηση των κόμβων ενός γραφήματος σε ομάδες βάσει της δομής του γραφήματος και/ή των χαρακτηριστικών των κόμβων. Διάφορα έργα στην μάθηση αναπαράστασης κόμβων αναπτύσσονται και οι αναπαραστάσεις των κόμβων μπορούν να περάσουν σε παραδοσιακούς αλγορίθμους ομαδοποίησης. Εκτός από την εκμάθηση αναπαραστάσεων κόμβων, η συγκέντρωση γραφημάτων (graph pooling) (Ying κ.α., 2018b) [21] θεωρείται ως μια μορφή ομαδοποίησης.

2.4.2 Φυσικά Συστήματα

Η μοντελοποίηση πραγματικών φυσικών συστημάτων είναι ένα από τα βασικότερα στοιχεία για την κατανόηση της ανθρώπινης νοημοσύνης. Ένα φυσικό σύστημα μπορεί να μοντελοποιηθεί ως τα αντικείμενα στο σύστημα και τις αμοιβαίες αλληλεπιδράσεις μεταξύ των αντικειμένων. Η προσομοίωση στο φυσικό σύστημα έχει ως απαίτηση από το μοντέλο να μάθει το νόμο του συστήματος και να προβλέπει την επόμενη κατάσταση του συστήματος. Με τη μοντελοποίηση των αντικειμένων ως κόμβους και των αμοιβαίων αλληλεπιδράσεων ως ακμές, τα συστήματα μπορούν να απλοποιηθούν ως γραφήματα. Για παράδειγμα, στα συστήματα σωματιδίων, τα σωματίδια μπορούν να αλληλεπιδρούν μεταξύ τους μέσω πολλαπλών αλληλεπιδράσεων, όπως σύγκρουση (Hoshen κ.α, 2017) [22], σύνδεση ελατηρίων, ηλεκτρομαγνητική δύναμη (Kipf κ.α., 2018) [23] κ.λ.π., όπου τα σωματίδια θεωρούνται ως κόμβοι και οι αλληλεπιδράσεις ως ακμές. Ένα άλλο παράδειγμα είναι το ρομποτικό σύστημα, το οποίο αποτελείται από πολλά σώματα (π.χ. βραχίονες, πόδια) που συνδέονται με αρθρώσεις. Τα σώματα και οι αρθρώσεις μπορούν να θεωρηθούν ως κόμβοι και ακμές, αντίστοιχα. Το μοντέλο πρέπει να εκτιμήσει την επόμενη κατάσταση των σωμάτων βάσει της τρέχουσας κατάστασης του συστήματος και των αρχών της φυσικής.

Η εμφάνιση των γραφικών νευρωνικών δικτύων μας δίνει την δυνατότητα να πραγματοποιούμε βασισμένα σε γραφήματα σκέψη σχετικά με αντικείμενα, σχέσεις και φυσική με έναν απλοποιημένο, αλλά αποτελεσματικό τρόπο.

2.4.3 Βιολογικά και Χημικά Συστήματα

Μοριακά αποτυπώματα. Τα μοριακά αποτυπώματα χρησιμοποιούνται για την κωδικοποίηση της δομής των μοριακών ενώσεων. Μια απλή μοριακή απεικόνιση μπορεί να είναι ένα διάνυσμα one-hot, όπου κάθε στοιχείο αντιπροσωπεύει την ύπαρξη ή απουσία μιας συγκεκριμένης υποδομής. Αυτά τα αποτυπώματα μπορούν να χρησιμοποιηθούν στην αναζήτηση μορίων, η οποία αποτελεί ένα βασικό βήμα στον σχεδιασμό φαρμάκων με τη βοήθεια υπολογιστή. Οι συμβατικά μοριακά αποτυπώματα είναι χειροποίητα και σταθερά, όπως το διάνυσμα one-hot. Ωστόσο, μπορούμε να θεωρήσουμε τα μόρια ως γραφήματα, με τα άτομα να είναι οι κόμβοι και τα χημικά δεσίματα να είναι οι ακμές. Επομένως, εφαρμόζοντας γραφικά νευρωνικά δίκτυα (GNNs) σε μοριακούς γραφήματα, μπορούμε να λάβουμε καλύτερα αποτυπώματα.

Οι Duvenaud κ.ά. (2015) [24] προτείνουν τα νευρωνικά γραφικά αποτυπώματα (Neural FPs), τα οποία υπολογίζουν διανυσματικά χαρακτηριστικά υποδομής χρησιμοποιώντας γραφικές συσχετίσεις και συνοψίζουν για να λάβουν συνολικές αναπαραστάσεις. Οι Kearnes κ.ά. (2016) [25] μοντελοποιούν ανεξάρτητα άτομα και ζεύγη ατόμων για να τονίσουν τις αλληλεπιδράσεις μεταξύ τους. Εισάγουν την αναπαράσταση των ακμών et_uv αντί για την συνάθροιση των γειτονικών ατόμων, δηλαδή $ht\_N(v) = \Sigma_{\{u \in N(v)\}} et_{uv}$.

Πρόβλεψη Χημικών Αντιδράσεων. Η πρόβλεψη του προϊόντος χημικής αντίδρασης είναι ένα βασικό θέμα στην οργανική χημεία. Το Graph Transformation Policy Network (Do κ.α., 2019) [26] κωδικοποιεί τις εισερχόμενες μοριακές ενώσεις και έτσι φτιάχνει έναν ενδιάμεσο γράφημα με ένα δίκτυο πρόβλεψης ζεύγους κόμβων και ένα δίκτυο πολιτικής.

Πρόβλεψη Διεπαφής Πρωτεϊνών. Οι πρωτεΐνες έχουν την δυνατότητα και αλληλεπιδρούν μεταξύ τους χρησιμοποιώντας τη διεπαφή, η οποία δημιουργείται από τα αμινοξέα των συμμετεχόντων πρωτεϊνών. Ο στόχος για την πρόβλεψη της διεπαφής των πρωτεϊνών είναι για να προσδιοριστεί εάν συγκεκριμένα αμινοξέα αποτελούν μέρος μιας πρωτεΐνης. Γενικά, η πρόβλεψη για ένα μεμονωμένο αμινοξέα εξαρτάται από άλλα γειτονικά αμινοξέα. Παρέχοντας τα αμινοξέα την δυνατότητα να είναι κόμβοι, οι πρωτεΐνες μπορούν να αναπαρασταθούν ως γραφήματα, τα οποία μπορούν να εκμεταλλευτούν αλγόριθμους μηχανικής μάθησης βασισμένους σε GNNs. Το MR-GNN (Xu κ.α., 2019d) [27] εισάγει μια πολυεπίπεδη προσέγγιση για την εξαγωγή και συνοψίζει τα τοπικά και γενικά χαρακτηριστικά για καλύτερη πρόβλεψη.

Βιοϊατρική Μηχανική. Με το δίκτυο αλληλεπίδρασης πρωτεϊνών-πρωτεϊνών, οι Rhee κ.ά. (2018) [28] εκμεταλλεύονται τη γραφική συσχέτιση και το δίκτυο σχέσεων για την ταξινόμηση των κατώτερων τύπων καρκίνου του μαστού. Οι Zitnik κ.ά. (2018) [29] προτείνουν επίσης ένα μοντέλο βασισμένο σε GCN για την πρόβλεψη παρενεργειών πολυφαρμακίας. Το έργο τους μοντελοποιεί το δίκτυο αλληλεπίδρασης φαρμάκου-πρωτεΐνης και αντιμετωπίζει ξεχωριστά τις ακμές διαφορετικών τύπων.

2.4.4 Γράφημα Γνώσης

Το γράφημα γνώσης (Knowledge Graph, KG) παρουσιάζει μια συλλογή πραγματικών οντοτήτων και των σχέσεων μεταξύ ζευγαριών οντοτήτων. Έχει διάφορους τρόπους εφαρμογής, όπως η απάντηση ερωτήσεων, η ανάκτηση πληροφοριών και η δημιουργία βασισμένη σε γνώση. Οι διάφορες εργασίες πάνω στο KG περιλαμβάνουν τη μάθηση χαμηλής διάστασης ενσωματώσεων που περιέχουν πλούσια σημασιολογία για τις οντότητες και τις σχέσεις, την πρόβλεψη των λειψάνων συνδέσμων μεταξύ των οντοτήτων και την πολλαπλών βημάτων σκέψη πάνω στο γνωστικό γράφημα. Ένα από τους δύο τρόπους προσέγγισης θεωρεί το γράφημα ως μια συλλογή τριάδων και προτείνει διάφορα είδη συναρτήσεων απώλειας για να διαχωρίσει τις σωστές τριάδες από τις ψευδείς τριάδες (Bordes κ.α., 2011) [30]. Η δεύτερη προσέγγιση εκμεταλλεύεται τη γραφική φύση του KG και χειρίζεται μεθόδους που βασίζονται σε γραφικά νευρωνικά δίκτυα (GNNs) για διάφορες εργασίες. Στη περίπτωση που θεωρείται ως γράφημα, το KG μπορεί να θεωρηθεί ως ένα ανομοιογενή γράφημα. Παρόλα αυτά, σε αντίθεση από άλλα ανομοιογενείς γραφήματα όπως οι κοινωνικοί δίκτυα, οι λογικές σχέσεις είναι πιο σημαντικές από την απλή δομή του γραφήματος.

2.4.5 Γενετικά Μοντέλα

Τα γενετικά μοντέλα για γραφήματα του πραγματικού κόσμου έχουν τραβήξει την προσοχή λόγω των σημαντικών εφαρμογών τους, μέσα σε αυτά είναι η μοντελοποίηση κοινωνικών αλληλεπιδράσεων, η ανακάλυψη νέων χημικών δομών και η κατασκευή γραφημάτων γνώσης. Καθώς οι μέθοδοι βαθιάς μάθησης κατορθώνουν να μάθουν την έμμεση κατανομή των γραφημάτων, υπάρχει μια άνοδο στα πρόσφατα νευρωνικά γραφικά γενετικά μοντέλα.

2.4.6 Συνδυαστική Βελτιστοποίηση

Τα προβλήματα συνδυαστικής βελτιστοποίησης πάνω σε γραφήματα αποτελούν ένα σύνολο προβλημάτων NP-hard που προσελκύουν μεγάλο ενδιαφέρον από επιστήμονες όλων των πεδίων. Κάποια συγκεκριμένα προβλήματα όπως το πρόβλημα του περιοδεύοντος πωλητή (TSP) και τα ελάχιστα κατακόρυφα δέντρα (MST) έχουν λάβει διάφορες προσεγγιστικές λύσεις. Πρόσφατα, η χρήση ενός βαθιού νευρωνικού δικτύου για την επίλυση τέτοιων προβλημάτων έχει γίνει αντικείμενο έντονου ενδιαφέροντος, και ορισμένες από τις λύσεις αξιοποιούν περαιτέρω τα γραφικά νευρωνικά δίκτυα λόγω της δομής τους. Το πρόβλημα του περιοδεύοντος πωλητή θα αναφερθεί περαιτέρω στα επόμενα κεφάλαια.

2.4.7 Μη-Δομικά Σενάρια

Σε αυτό το μέρος θα γίνει αναφορά για εφαρμογές σε μη-δομικά σενάρια. Γενικά, υπάρχουν δύο τρόποι προσέγγισης για την εφαρμογή των GNNs σε μη-δομικά σενάρια: (1) Συγχώνευση δομικών πληροφοριών από άλλους τομείς για τη βελτίωση της απόδοσης, παράδειγμα, χρήση πληροφοριών από γνωστικά γραφήματα για την αντιμετώπιση των προβλημάτων μηδενικής εκπαίδευσης σε εικόνες. (2) Εκτίμηση ή υπόθεση της συσχετικής δομής στην εργασία και τέλος εφαρμογή του μοντέλου για την επίλυση των προβλημάτων που ορίζονται σε γραφήματα.

2.4.8 Εικόνες

Few(Zero)-shot Image Classification. Η κατηγοριοποίηση εικόνας είναι μια από τις πιο γνωστές και σημαντικές εργασίες στον τομέα της όρασης υπολογιστών, η οποία έχει μεγάλο ενδιαφέρον και διαθέτει πολλά γνωστά σύνολα δεδομένων όπως το ImageNet (Russakovsky και συν., 2015) [31]. Πρόσφατα, η μάθηση με μηδενικά ή λίγα δείγματα (zero-shot και few-shot learning) έχει γίνει όλο και πιο δημοφιλής στον τομέα της κατηγοριοποίησης εικόνας. Στην N-shot μάθηση, για να γίνουν προβλέψεις για δείγματα δοκιμής σε ορισμένες κατηγορίες, στο σετ εκπαίδευσης παρέχονται μόνο N δείγματα εκπαίδευσης από τις ίδιες κατηγορίες. Επειδή γίνεται αυτό η few-shot μάθηση περιορίζει το N σε μικρές τιμές, ενώ η μηδενική (zero-shot) απαιτεί το N να είναι 0. Τα μοντέλα πρέπει να μάθουν να γενικεύουν από τα περιορισμένα δεδομένα εκπαίδευσης για να κάνουν νέες προβλέψεις για δεδομένα δοκιμής. Από την άλλη, τα γραφικά νευρωνικά δίκτυα μπορούν να βοηθήσουν το σύστημα κατηγοριοποίησης εικόνας σε αυτά τα απαιτητικά σενάρια.

2.4.9 Κείμενο

Τα γραφικά νευρωνικά δίκτυα μπορούν να εφαρμοστούν σε διάφορες εργασίες που έχουν να κάνουν με κείμενα. Μπορούν να εφαρμοστούν τόσο σε εργασίες επιπέδου προτάσεων (π.χ. ταξινόμηση κειμένου), όσο και σε εργασίες επιπέδου λέξεων (π.χ. επισήμανση ακολουθίας).

2.5 Ανοιχτά Προβλήματα

Παρόλο που τα GNN έχουν κατορθώσει μεγάλη επιτυχία σε διάφορους τομείς, είναι σημαντικό να αναφερθεί ότι τα μοντέλα GNN δεν είναι αρκετά ικανοποιητικά για να προσφέρουν ικανοποιητικές λύσεις για οποιοδήποτε γραφήματα σε οποιαδήποτε κατάσταση. Σε αυτήν την ενότητα, τίθενται μερικά ανοιχτά προβλήματα για περαιτέρω έρευνα.

Ανθεκτικότητα. Ως μέλος της οικογένεια των μοντέλων που βασίζονται σε νευρωνικά δίκτυα, τα GNN είναι ευάλωτα σε αντιπαραβολικές επιθέσεις. Συγκρίνοντας τις αντιπαραβολικές επιθέσεις σε εικόνες ή κείμενο που επικεντρώνονται μόνο στα χαρακτηριστικά, οι επιθέσεις στα γραφήματα λαμβάνουν υπόψη τις δομικές πληροφορίες.

Διαφάνεια. Η διαφάνεια είναι επίσης ένας σημαντικός τρόπος έρευνας κατεύθυνσης για τα νευρωνικά μοντέλα. Ωστόσο, τα GNN είναι επίσης μαύρα κουτιά και έχουν έλλειψη εξηγήσεων. Μόνο μερικές μέθοδοι (Baldassarre και Azizpour, 2019) [32] προτάθηκαν για τη δημιουργία εξηγήσεων σε επίπεδο παραδειγμάτων για τα μοντέλα GNN. Είναι σημαντικό τα μοντέλα GNN να τεθούν σε εφαρμογές πραγματικού κόσμου με αξιόπιστες εξηγήσεις. Όπως και στους τομείς της Επεξεργασίας Εικόνας (CV) και της Επεξεργασίας Φυσικής Γλώσσας (NLP), η διαφάνεια στα γραφήματα είναι επίσης μια σημαντική κατεύθυνση για έρευνα.

Προ Εκπαίδευση Γραφημάτων. Τα μοντέλα που βασίζονται σε νευρωνικά δίκτυα απαιτούν πλούσια επισημασμένα δεδομένα και είναι δαπανηρό να αποκτηθούν τεράστια ανθρώπινα επισημασμένα δεδομένα. Προτείνονται μέθοδοι αυτό-εκπαίδευσης για να βοηθήσουν τα μοντέλα να μάθουν από μη επισημασμένα δεδομένα που είναι εύκολο να αποκτηθούν από ιστότοπους ή βάσεις γνώσης. Αυτές οι μέθοδοι έχουν καταφέρει μεγάλη επιτυχία στον τομέα της Επεξεργασίας Εικόνας (CV) και της Επεξεργασίας Φυσικής Γλώσσας (NLP) με την ιδέα της προ εκπαίδευσης (Krizhevsky κ.ά., 2012· Devlin κ.ά., 2019) [33] [34].

Πολύπλοκες Δομές γραφημάτων. Οι δομές γραφημάτων είναι ευέλικτες και πολύπλοκες στις εφαρμογές του πραγματικού κόσμου. Έχουν συζητηθεί διάφορη τρόποι για την αντιμετώπιση πολύπλοκων δομών γραφημάτων, όπως δυναμικά γραφήματα ή ανομοιογενείς γραφήματα. Με την ταχεία ανάπτυξη των κοινωνικών δικτύων στο Διαδίκτυο, σίγουρα υπάρχουν περισσότερα προβλήματα, προκλήσεις και σενάρια εφαρμογών που εμφανίζονται και απαιτούν πιο ισχυρά μοντέλα.

Κεφάλαιο 3 : TSP

3.1 Εισαγωγή

Ένας πλανόδιος πωλητής θέλει να επισκεφτεί ακριβώς μια φορά κάθε μια από τον αριθμό τον πόλεων που έχει επιλέξει και να γυρίσει στην πρωταρχική πόλη που ξεκίνησε. Ένα από τα βασικότερα προβλήματα είναι να βρεθεί η συντομότερη διαδρομή. Το Travel Salesman Problem η αλλιώς TSP, έχει χρησιμοποιηθεί σε διάφορες πτυχές της επιστήμης, όπως μαθηματικά, προγραμματισμός και επιχειρησιακή έρευνα. Heuristics, Linear [35] και branch and bound [35] που εξακολουθούν να θεωρούνται από τα βασικότερα εργαλεία για πετυχημένη απόπειρα λύσης δύσκολων συνδυαστικών προβλημάτων βελτιστοποίησης, πρώτο χρησιμοποιήθηκαν για την επίλυση προβλημάτων TSP το 1954 από τον DANTZIG, Fulkeeson and Johnson [36]. Όταν αναπτύχθηκε η θεωρεία του NP-Hard από τον Karp 1972 [37]. Νέες αλγοριθμικές τεχνικές έχουν αρχικά αναπτυχθεί η τουλάχιστον έχουν εφαρμοστεί στο TSP για να δείξουν την αποτελεσματικότητα τους. Από τους παλιούς τρόπους επίλυσης το TSP αναφερόμαστε στους Hoffman και Wolle (1985) [35]. Οι στατικές αλγοριθμικές προσεγγίσεις έχουν πραγματοποιηθεί πριν 40 χρόνια. Οι αλλαγές μέχρι το 1985 είχαν πραγματοποιηθεί από τους Lawler, Lestra, Rinnooy Kan και τον Shmoys (1985) [38]. Επίσης θα πρέπει να αναφερθεί το γεγονός ότι το TSP ανήκει στην κατηγορία NP-complete. Δηλαδή αν βρεθεί κάποιος αποτελεσματικός αλγόριθμος για το πρόβλημα του πλανόδιου πωλητή, τότε θα μπορέσουν να βρεθούν και άλλοι αποτελεσματική αλγόριθμοι για τα άλλα NP-complete προβλήματα. Από το 1992 έως το 2006 το Concorde, ένα λογισμικό που δημιουργήθηκε από τον D. Οι Applegate, R.E. Bixby, V. Chvátal και W.J. Cook (Applegate κ.α. 1995, 2006) [39], έλυσε (μεταξύ πολλών άλλων) ένα πρόβλημα περιοδεύοντος πωλητή που μοντελοποιεί το παραγωγή πλακετών τυπωμένων κυκλωμάτων με 7.397 οπές (πόλεις), ένα πρόβλημα με τις 13.509 μεγαλύτερες πόλεις στην ΗΠΑ, ένα ακόμα πάνω στις 24.978 πόλεις της Σουηδίας και τέλος ένα πρόβλημα 85.900 πόλης που προκύπτουν από την εφαρμογή VLSI.

3.2 Σύνθεση του προβλήματος

Το πρώτο βήμα για την επίλυση περιπτώσεων μεγάλων TSP πρέπει να είναι η εύρεση ενός καλού τρόπου διατύπωσης του μαθηματικού προβλήματος. Στην περίπτωση του προβλήματος του περιοδεύοντος πωλητή, η μαθηματική δομή είναι ένα γράφημα όπου κάθε πόλη συμβολίζεται με ένα σημείο (ή κόμβο) και οι γραμμές σχεδιάζονται συνδέοντας κάθε δύο κόμβους (που ονομάζονται τόξα ή άκρες). Η συσχέτιση με κάθε γραμμή είναι η απόσταση η αλλιώς το κόστος. Όταν ο περιοδεύοντος πωλητής μπορεί να πάει από μια πόλη σε μια οποιαδήποτε άλλη πόλη απευθείας, τότε το γράφημα θεωρείται ότι είναι πλήρες. Ένα ταξίδι μετ' επιστροφής στις πόλεις αντιστοιχεί σε κάποιο υποσύνολο των γραμμών, και ονομάζεται περιοδεία ή **Hamiltonian cycle [35]** στη θεωρία των γραφημάτων. Το μήκος μιας περιήγησης είναι το άθροισμα των μηκών των γραμμών στο ταξίδι μετ' επιστροφής.

Σε αυτήν την ενότητα θα αναφερθούμε σε πιο σύγχρονους τρόπους επίλυσης. Έστω *Kn = (Vn, En)* είναι το πλήρες μη κατευθυνόμενο γράφημα με *n = [Vn]* κόμβους *και m= [En] = (n/2)* της άκρες. Μια άκρη με τελικές κατευθύνσεις *i* και *j* συμβολίζεται επίσης και με *ij* η αλλιώς (i, j). Συμβολίζουμε με Ren τον χώρο των πραγματικών διανυσμάτων των οποίων τα συστατικά είναι ταξινομημένα σύμφωνα με τα στοιχεία του En. Η συνιστώσα οποιουδήποτε διανύσματος ZεRen ταξινομεί τον κόμβο e=ij που συμβολίζεται μZe,Zij η αλλιώς Z(i,j). Δίνοντας μια αντικειμενική συνάρτηση CεRen,

που συνδέει ένα 'μήκος' Ce με όλα τα άκρα e το Kn, το συμμετρικό TSP εστιάζει στο να βρει ένα κύκλο Hamilton (ένας κύκλος ο οποίος επισκέπτεται όλους τους κόμβους από μια μόνο φορά) έτσι ώστε το c-lenght (το άθροισμα από τα μήκη των ακρών) να είναι όσο μικρότερο γίνεται. Σε μεγάλο ενδιαφέρον είναι η Ευκλείδεια απόσταση του traveling salesmen problem. Σε αυτή την περίπτωση οι κόμβοι λειτουργούν σαν τελείες σε έναν δισδιάστατο πίνακα και η απόσταση μεταξύ δύω κόμβων είναι η ευκλείδεια απόσταση μεταξύ των δύω αυτών κουκκίδων. Ο λόγος για την χρήση ενός πλήρους γραφήματος στον ορισμό του TSP είναι ότι για ένα τέτοιο γράφημα είναι εξασφαλισμένη η ύπαρξη μιας εφικτής λύσης, ενώ για γενικά γραφήματα που καθορίζουν την ύπαρξη ενός κύκλου **Hamilton** είναι ένα πλήρες NP-complete πρόβλημα. Στην πραγματικότητα ο αριθμός των κύκλων **Hamilton** σε Kn, δηλαδή το μέγεθος των εφικτών λύσεων του **TSP** είναι (n-1)!/2.

3.3 Αλγόριθμος

Για να έχουμε μια ακριβείς προσεγγίσει για την επίλυση τέτοιων προβλημάτων υπάρχει η απαίτηση στους αλγόριθμους να υπάρχει ένα κατώτερο όριο και ένα ανώτερο όριο στην πραγματική ελάχιστη τιμή της παρουσίας του προβλήματος. Οποιαδήποτε διαδρομή με το να επιστρέφει στην πρωταρχική πόλη, που περνά κάθε πόλη ακριβώς μια φορά θεωρείται μια εφικτή λύση με δεδομένο κόστος που δεν μπορεί να είναι μικρότερο από το ελάχιστο κόστος περιήγησης. Αλγόριθμοι που κατασκευάζουν μια εφικτή λύση και επομένως δεν ξεπερνούν το ανώτατο όριο για την βέλτιστη λύση ονομάζονται Heuristics. Αυτές οι λύσης συνήθως προσφέρουν την απάντηση αλλά όχι με την εγγύηση με το πόσο κοντά είναι από την βέλτιστη απάντηση. OI αλγόριθμοι Heuristics που βρίσκουν με την πρώτη προσπάθεια μια εφικτή λύση ονομάζονται **Constructive Heuristics**, ενώ αλγόριθμοι που προσπαθούν να βελτιώσουν την λύση τροποποιώντας τα δεδομένα ονομάζονται **Improvement Heuristics**. Όταν η πρωταρχική λύση που διαθέτει το μοντέλο εξαρτάται από την αρχική του θέση, τότε το ίδιο μοντέλο μπορεί να χρησιμοποιηθεί πολλές φορές σε διάφορα τυχαία σημεία ως σημεία εκκίνησης. Συνήθως αν χρειάζεται κάποιος μια γρήγορη λύση προσαρμόζεται και με έναν καλά σχεδιασμένο **Heuristics** αλγόριθμο που δείχνει να φτάνει στη << σχεδόν βέλτιστη>> λύση και πολλά προβλήματα περιηγήσεων TSP. Μια έρευνα από Johnson και McGeoch (2002) και Applegate κ.α. (2006) [39] [40] εξηγεί για αλγορίθμους που επιλύουν εξαιρετικά μεγάλα TSP (προβλήματα με εκατοντάδες χιλιάδες η ακόμα και εκατομμύρια μεταβλητές) εντός 1 η 2% της βέλτιστης λύσης σε λογικούς χρόνους. Ο Heuristic αλγόριθμος των Lin και Kernighan [41] δείχνει να είναι ο πιο αποδοτικός μέχρι και σήμερα από την άποψη της ποιότητας της λύσης, συγκεκριμένα μετά την παραλλαγή που πρότεινε ο Helsgaun (2000) [41], η οποία μπόρεσε να βρει, για πρώτη φορά τη βέλτιστη λύση (αν και χωρίς την εγγύηση της ποιότητας) για πολλές περιπτώσεις του TSPLIB, μια πολύ γνωστής βιβλιοθήκης για τα προβλήματα που έχουν να κάνουν με το TSP περιγράφεται στο Reinelt (1991) [42].

Για να ήμαστε σε θέση να ξέρουμε με σιγουριά την βέλτιστη τιμή του άνω ορίου, θα πρέπει να ξέρουμε με σιγουριά την βέλτιστη τιμή του κάτω ορίου. Αν τα άνω και κάτω όρια συμπίπτουν, επιτυγχάνεται απόδειξη βέλτιστου. Στη περίπτωση που δεν ισχύει, το σχετικό σφάλμα του άνω ορίου παρέχεται από τη διαφορά του άνω και του κάτω όριο διαιρούμενο με το κάτω όριο. Για τον λόγο αυτό χρειαζόμαστε και πάνω και κάτω τεχνικές οριοθέτησης για την εύρεση βέλτιστων λύσεων σε hard combinatorial problems [35] η ακόμα και για την απόκτηση λύσεων με εγγύηση της ποιότητας.

Πως λοιπόν κάποιος έχει την δυνατότητα να διαθέτει και να μπορεί να βελτιώσει το κατώτερο όριο; Μια μέθοδος χαλάρωσης ενός προβλήματος είναι ένα άλλο είδους προβλήματος βελτιστοποίησης του

οποίου το σύνολο των εφικτών λύσεων περιέχει όλες τις εφικτές λύσεις του αρχικού προβλήματος και των οποίων η τιμή συνάρτησης είναι μικρότερη ή ίση με την πραγματική τιμή συνάρτησης για σημεία εφικτά στο αρχικό πρόβλημα. Έτσι, αντικαθιστούμε το "αληθινό" πρόβλημα με ένα που έχει μεγαλύτερη εμβέλεια, αλλά ταυτόχρονα είναι πιο εύκολα επιλύσιμο. Αυτό του είδους χαλάρωσης είναι συνεχής και εξειδικευμένη έτσι ώστε να είναι όσο μικρή η εφικτή περιοχή, έτσι ώστε να είναι όσο πιο κοντά γίνεται στο να μοιάζει με το αληθινό πρόβλημα. Η τυπική τεχνική για την απόκτηση χαμηλότερων ορίων στο πρόβλημα TSP είναι να χρησιμοποιούμε μια τεχνική χαλάρωσης που είναι ευκολότερο να λυθεί από το αρχικό πρόβλημα. Αυτές τις χαλαρώσεις μπορούμε κατέχουμε είτε με διακριτά είτε με συνεχή σύνολα. Έχουν μελετηθεί αρκετές χαλαρώσεις για το TSP. Μεταξύ αυτών είναι η χαλάρωση n-path, η assignment χαλάρωση, η χαλάρωση 2-matching, η χαλάρωση 1-tree και η linear Programming χαλάρωσης. Για τυχαία ασύμμετρα δομημένα TSPs, για προβλήματα που έχουν αριθμό πόλεων έως και 7.500 έχουν επιλυθεί, στις αρχές τις δεκαετίας του ΄90, χρησιμοποιώντας μια χαλάρωση ανάθεσης το οποίο προσθέτει υπό περιγράμματα μέσα σε ένα πλαίσιο διακλάδωσης και δέσμευσης και το οποίο χρησιμοποιεί προσεγγιστική άνω οριοθέτηση βασισμένη σε επιδιόρθωση υπό περιγράμματος (Miller and Pekny, 1991) [43]. Για το συμμετρικό TSP, η χαλάρωση 1-tree και οι 2-matching χαλαρώσεις ήταν πιο επιτυχημένες. Αυτές οι χαλαρώσεις έχουν ενσωματωθεί σε ένα πλαίσιο διακλάδωσης και δέσμευσης.

Η διαδικασία της εύρεσης περιορισμών που παραβιάζονται από μια δεδομένη χαλάρωση ονομάζεται τεχνική κοπής επιπέδων (cutting plane technique) και η επιτυχία σε μεγάλα προβλήματα TSP έχει συνήθως επιτευχθεί μέσω της εφαρμογής της τεχνικής κοπής επιπέδων για την συνεχή σύσφιξη της διατύπωσης του προβλήματος. Για να επιτευχθεί μια ακριβής χαλάρωση, οι ανισότητες που χρησιμοποιούνται ως κοπές επιπέδων σε πολλές υπολογιστικές προσεγγίσεις για το TSP είναι συχνά ανισότητες που καθορίζουν τα περιγράμματα των λύσεων (facet-defining inequalities).

Ένας από τους πιο απλουστευμένους τύπους περιορισμού που αποδεικνύεται ότι καθορίζει τα περιγράμματα του υπό προβλήματος του TSP είναι η κοπή εξάλειψης υποπεριοχών (subtour elimination cut). Εκτός από αυτούς τους τύπους περιορισμού υπάρχουν και άλλοι, μερικοί από αυτούς είναι οι ανισότητες συνδυασμού (comb), ανισότητες δέντρου κλικ (clique tree), ανισότητες μονοπατιού (path), ανισότητες τροχοφόρου και ποδηλάτου (wheelbarrow and bicycle), ανισότητες σκάλας (ladder), ανισότητες στεφάνης (crown), ανισότητες ντόμινο (domino) και πολλοί άλλοι που έχουν αποδειχθεί ότι καθορίζουν περιγράμματα αυτού του πολύτοπου. Η θεωρητική βάση για την παραγωγή περιγραμμάτων στο συμμετρικό πρόβλημα του περιοδεύοντος πωλητή (TSP) παρέχεται από τους Grötschel και Padberg (1985) [44], Jünger Reinelt και Rinaldi (1994) [45] και Naddef (2002) [46], ενώ αντίστοιχα αποτελέσματα για το υπό πρόβλημα ATSP παρέχονται από τους Balas και Fischetti (2002) [47]. Οι αλγοριθμικές λεπτομέρειες για το πώς αυτές οι ανισότητες χρησιμοποιούνται στην προσέγγιση με κοπή επιπέδων.

Οι διαδικασίες κοπής επιπέδων μπορούν στη συνέχεια να ενσωματωθούν σε έναν αλγόριθμο αναζήτηση δέντρου που αναφέρεται ως "branch and cut" και προτάθηκε από τον Padberg και τον Rinaldi (1991) [48], όπου δείχνεται πώς με αυτήν την προσέγγιση ήταν δυνατό να επιλυθούν μερικά ακόμα ανεπίλυτα παραδείγματα μεγέθους έως 2.392 κόμβους. Ένας μεγάλος αριθμός από τα μεγαλύτερα προβλήματα TSP που έχουν επιλυθεί έχουν χρησιμοποιήσει παράλληλη επεξεργασία για να βοηθήσουν στην αναζήτηση της βέλτιστης λύσης. Αυτό συμβαίνει στο λογισμικό Concorde, όπου έχουν εφαρμοστεί προσεκτικά όλες οι γνωστές αλγοριθμικές ιδέες για το TSP (και πολλές νέες). Χρησιμοποιώντας αυτόν τον συγκεκριμένο κώδικα, οι Applegate κ.ά. (2006) [39] είχαν την δυνατότητα να καταφέρουν να επιλύσουν όλα τα προβλήματα της TSPLIB με βέλτιστη λύση. Για το πρόβλημα με τους περισσότερους κόμβους (85.900 nodes), χρησιμοποίησαν 96 σταθμούς εργασίας για συνολικό χρόνο CPU 139 ετών.

Καθώς ο καιρός περνάει και έχουμε μεγαλύτερη κατανόηση για την υποκείμενη μαθηματική δομή του προβλήματος TSP παρέχεται βελτίωση και τη συνεχή πρόοδο στην τεχνολογία υπολογιστών, είναι πιθανό να επιλυθούν πολλά δύσκολα και σημαντικά προβλήματα συνδυαστικής βελτιστοποίησης με τη χρήση συνδυασμού διαδικασιών παραγωγής κοπών επιπέδων, σταθεροποίησης μεταβλητών μέσω λογικών συνεπειών και μειωμένων κοστών, καθώς και αναζήτησης σε δέντρο.

3.4 Εφαρμογές

Σε αυτό το σημείο θα πρέπει να αναφερθεί το γεγονός, γιατί είναι τόσο σημαντικό και τόσο διαχρονικό αυτό το πρόβλημα. Μεγάλο μέρος της προσοχής που έχει λάβει το πρόβλημα οφείλεται στο γεγονός ότι είναι ένα σχετικά απλό πρόβλημα να περιγραφεί, αλλά δύσκολο (από άποψη πολυπλοκότητας) πρόβλημα βελτιστοποίησης να επιλυθεί. Επίσης είναι σημαντικό να μιλήσουμε για τις παραλλαγές που προσφέρει το πρόβλημα για πιο πρακτικά κομμάτια που μπορούν να διατυπωθούν και σαν προβλήματα TSP. Εκτός από τον τρύπημα πλακών εκτυπωμένων κυκλωμάτων που περιγράφηκε παραπάνω, προβλήματα με δομή TSP εμφανίζονται στην ανάλυση της δομής κρυστάλλων [35], στην ανακατασκευή αεροτομών αερίου [35], στην χειρισμό υλικών σε ένα αποθήκευση (Ratliff και Rosenthal, 1981) [49], στα προβλήματα κοπής υλικού (Garfinkel, 1977) [35], στο συμμάζεμα των πινάκων δεδομένων (Lenstra και Rinooy Kan, 1975) [35], στην ακολουθία εργασιών σε μια μονή μηχανή, στην ανάθεση διαδρομών για αεροσκάφη συγκεκριμένου στόλου (Boland, Jones και Nemhauser, 1994) και στον προγραμματισμό ακολουθίας γονιδίων (Ben-Dor και Chor, 1997; Ben-Dor, Chor και Pelleg, 2000) [50] [50][51].

Υπάρχουν εξίσου και άλλα είδη προβλημάτων του περιοδεύοντος πωλητή περιλαμβάνουν το πρόβλημα του περιορισμένου σε πόρους περιοδεύοντος πωλητή, το οποίο κατέληξε στο να εφαρμοστή στον προγραμματισμό με συγκεντρωτική προθεσμία (Pekny και Miller, 1991) [43]. Επίσης σε αυτό το ίδιος προβλήματος περιοδεύοντος πωλητή είναι η συλλογή βραβείων (Balas, 2002) [52] και το πρόβλημα της πεζοπορίας (Golden, Levy και Vohra, 1987; Fischetti, Salazar και Toth, 2002) [53] [45]. Το πρόβλημα του περιοδεύοντος πωλητή κατά κόρον προκύπτει να είναι κατώτερα προβλήματα σε πιο πολύπλοκα συνδυαστικά προβλήματα, ένα από τα πιο γνωστά και σημαντικά από αυτά είναι το πρόβλημα δρομολόγησης οχημάτων. Αυτό είναι το πρόβλημα του να καθοριστεί για μια ομάδα οχημάτων ποιοι πελάτες θα εξυπηρετηθούν από κάθε οχήματα και με ποια σειρά θα επισκεφθεί κάθε όχημα τους εκχωρημένους πελάτες. Για περαιτέρω πληροφορίες, διαβάστε το βιβλίο “The Vehicle Routing Problem” που επιμελήθηκαν οι Toth και Vigo (2001) [54].

Κεφάλαιο 4 : TSP on GNN

4.1 Εισαγωγή

Σε αυτό το κεφάλαιο θα αναλύσουμε για κάθε τεχνολογία που έχει φτιαχτεί για την αντιμετώπιση του κλασικού και διαχρονικού προβλήματος του περιοδεύοντος πολίτη TSP χρησιμοποιώντας τα GNN. Πιο συγκεκριμένα θα αναλύσουμε τους αλγόριθμους και τους διαφορετικούς τρόπους επίλυσης του ίδιου προβλήματος. Στο τέλος κάθε μεθόδου υπάρχουν αναλυτική πίνακες για την απόδοση τους σε σχέση με τις πιο κλασικές μεθόδους επίλυσης του ίδιου προβλήματος.

Στη συνέχεια στο κεφάλαιο θα παρατηρήσουμε ότι υπάρχουν τριών ειδών επίλυσης του TSP με τα GNN.

1. Το μοντέλο μόνο του επιλύει το πρόβλημα του πλανόδιου πωλητή. Δηλαδή το νευρωνικό δίκτυο καθαρά από μόνο του βγάζει ως αποτέλεσμα τη λύση του προβλήματος.
2. Το μοντέλο δίνει πληροφορίες σε ένα OR αλγόριθμο. Έχοντας το νευρωνικό δίκτυο να προσφέρει μεταβλητές που στη συνέχεια θα χρησιμοποιηθούν από έναν αλγόριθμο που θα επίλυση το πρόβλημα.
3. Τέλος το μοντέλο λειτουργεί παράλληλα και παίρνει αποφάσεις με ένα OR αλγόριθμο. Έχοντας το νευρωνικό δίκτυο για παράδειγμα να διαλέγει την επόμενη πόλη με την βοήθεια ενός δέντρου αποφάσεων.

Όλοι αυτοί οι τρόποι επίλυσης θα αναφερθούν παρακάτω.

4.2 Graph Neural Network Για Αποφάσεις TSP

Η βαθιά μάθηση (Deep Learning) έχει κατορθώσει πολλά τα τελευταία χρόνια, επεκτείνοντας σε τομείς όπως η τέχνη η αναγνώριση εικόνας (Krizhevsky, Simonyan και Zisserman 2014, κ.α.) [33] [55], η επεξεργασία φυσικής γλώσσας (Cho κ.ά. 2014b, 2014a· Bahdanau, Cho και Bengio 2014) [56] [57] και η ενίσχυση της μάθησης (reinforcement learning) (Mnih κ.ά. 2013, 2015) [58] [59], η οποία λόγο συνδυασμού με τα βαθιά νευρωνικά δίκτυα κατόρθωσε να μάθει κλασικά παιχνίδια Atari και να επιτύχει ανθρώπινη απόδοση στο κινεζικό επιτραπέζιο παιχνίδι Go (Mnih κ.ά. 2013, 2015) [58] [59]. Ωστόσο, η χρήση της βαθιάς μάθησης απευθείας σε συμβολικά πεδία, σε αντίθεση με τους πράκτορες στην ενισχυμένη μάθηση, βρίσκεται ακόμα σε αρχικά στάδια (Evans και Grefenstette 2018) [60].

Μια πολλά υποσχόμενη μετά-αρχιτεκτονική για την κατασκευή μοντέλων που μαθαίνουν σε συμβολικά πεδία είναι να δημιουργούνται ενότητες νευρώνων και να συναρμολογούνται σε διάφορες διαμορφώσεις, οπού η κάθε μια από αυτές εκδηλώνει μια γραφική αναπαράσταση ενός συγκεκριμένου προβλήματος (Scarselli κ.ά. 2008) [61]. Υπό αυτό το πλαίσιο, οι ενότητες νευρώνων έχουν την δυνατότητα να εκπαιδευτούν να υπολογίζουν μηνύματα που αποστέλλονται μεταξύ κόμβων, παράγοντας έναν ρυθμιζόμενο αλγόριθμο μετάδοσης μηνυμάτων, οι παράμετροι του οποίου βρίσκονται σε θέση να μπορούν να βελτιωθούν μέσω κατεύθυνσης κλίσης. Αυτή η τεχνική έχει χρησιμοποιηθεί με επιτυχία σε αυξανόμενη γκάμα προβλημάτων, αν και διαθέτοντας διαφορετικές ονομασίες. Οι Gilmer κ.ά., που την χρησιμοποιούν σε προβλήματα κβαντικής χημείας, του δίνουν τον

όρο "νευρωνικά μετάδοση μηνυμάτων" (Gilmer κ.ά. 2017) [62], ενώ οι Palm κ.ά. αναφέρονται σε (recurrent relational networks) "επαναληπτικά συστηματικά δίκτυα" σε μια προσπάθεια εκπαίδευσης νευρωνικών δικτύων για την επίλυση γρίφων Sudoku (Palm, Paquet και Winther 2017) [63].

Σε μια πρόσφατη ανασκόπηση σχετικών τεχνικών, επιλέχθηκε ο όρος "δίκτυα γραφημάτων" (Battaglia κ.ά., 2018) [64], αν και θα αναφερθούμε στην προσέγγιση ως "γραφικά νευρωνικά δίκτυα", όπως αρχικά ονομάστηκε από τους Scarselli κ.ά., οι οποίοι ήταν ανάμεσα στους πρωτοπόρους που πρότειναν ένα τέτοιο μοντέλο (Scarselli κ.ά., 2009) [5]. Τα γραφικά νευρωνικά δίκτυα (GNNs) έχουν επιδείξει ελπιδοφόρα αποτελέσματα στην επίλυση συνδυαστικών προβλημάτων NP-Complete, όπως το πρόβλημα αληθοφανούς ικανοποίησης (SAT) (Selsam κ.ά., 2018) [65]. Οι Selsam κ.ά. έδειξαν ότι τα GNNs μπορούν να εκπαιδευτούν για να επιτύχουν ικανοποιητική ακρίβεια (περίπου 85%) σε μικρές περιπτώσεις του SAT, και η απόδοσή τους μπορεί να βελτιωθεί περαιτέρω αυξάνοντας τον αριθμό των επαναλήψεων μετάδοσης μηνυμάτων. Ενδιαφέροντα αποτελέσματα είναι ότι τα εκπαιδευμένα GNNs μπορούσαν να εξάγουν ικανοποιητικές λύσεις χωρίς να έχουν εκπαιδευτεί ρητά για αυτό το σκοπό.

Εμπνευσμένοι από την επιτυχία του NeuroSAT (όπως το ονόμασαν οι συγγραφείς) [65], το οποίο επικεντρώθηκε στο SAT, οι συγγραφείς αυτού του άρθρου σκοπεύουν να διερμηνεύσουν τη δυνατότητα των GNNs [66] στην επίλυση ενός άλλου προβλήματος NP-Complete: την αποφασιστική μορφή του προβλήματος του περιοδεύοντος πωλητή (TSP). Σε αυτό το πρόβλημα, η ερώτηση είναι να καθοριστεί εάν ένα δεδομένο γράφημα επιτρέπει έναν Hamiltonian κύκλο [45] με ένα κόστος μικρότερο ή ίσο από ένα κατώφλι C. Ενώ το πείραμα NeuroSAT έδειξε την ικανότητα των GNNs να υπολογίζουν δύσκολα συνδυαστικά προβλήματα, το SAT είναι κατά έννοια πιο απλό καθώς μπορεί να καθοριστεί αποκλειστικά με βάση λογικές διατυπώσεις. Έτσι, ένα σημαντικό ερευνητικό ερώτημα είναι να διερευνήσουμε εάν τα GNNs μπορούν να εκπαιδευτούν αποτελεσματικά για την επίλυση προβλημάτων NP-Complete που περιλαμβάνουν αριθμητικές πληροφορίες, όπως τα βάρη των ακμών.

Με την εξερεύνηση του πόσου εφικτά είναι τα GNNs στην επίλυση του προβλήματος του περιοδεύοντος πωλητή σε αποφασιστική μορφή, στοχεύετε να συμβάλουν στην κατανόηση της ευρύτερης εφαρμογής των GNNs στην αντιμετώπιση πολύπλοκων συνδυαστικών προβλημάτων που συνδυάζουν συμβολικά και αριθμητικά δεδομένα.

Προσθέτοντας, στην περίπτωση των γραφικών νευρωνικών δικτύων, υπάρχει σημασία στις συμβολικές σχέσεις (ακμές ή συνδέσεις). Το πρόβλημα του περιοδεύοντος πωλητή στη μορφή που παίρνει αποφάσεις (θέτει το ερώτημα εάν το γράφημα G επιτρέπει ένα Hamiltonian [45] μονοπάτι με κόστος < C) αποτελεί έναν ελπιδοφόρο υποψήφιο, διότι απαιτεί τόσο τα βάρη των ακμών W_i όσο και το "στόχο κόστος" C να λαμβάνονται υπόψη για τον υπολογισμό μιας λύσης.

4.2.1 GNN μοντέλο αποφάσεων για το TSP

Τα γραφικά νευρωνικά δίκτυα αναθέτουν μια πολυδιάστατη ενσωμάτωση $\in R^d$ σε κάθε κόμβο του γραφήματος που αναπαριστά τη συγκεκριμένη προβληματική περίπτωση και εκτελούν μια σειρά επαναλήψεων μεταβίβασης μηνυμάτων, όπου ένα νευρωνικό μοντέλο υπολογίζει ένα μήνυμα από

κάθε ενσωμάτωση και το αποστέλλει στους γειτονικούς κόμβους. Κάθε κόμβος συσσωρεύει τα εισερχόμενα μηνύματα, συνδυάζοντάς τα (ή εφαρμόζοντας οποιαδήποτε άλλη λειτουργία πράξης) και το αποτέλεσμα το τροφοδοτεί σε ένα Αναδρομικό Νευρωνικό Δίκτυο (RNN), το οποίο αναλαμβάνει την ενημέρωση της ενσωμάτωσης του κόμβου. Οι εκπαιδευμένες παράμετροι αυτού του μοντέλου περιλαμβάνουν τα μοντέλα υπολογισμού μηνυμάτων και το RNN, επιτρέποντας την εκπαίδευσή του με τη χρήση νευρωνικών δικτύων.

Δεδομένης μιας περίπτωσης TSP $X = (G, C)$ που αποτελείται από ένα γράφημα $G = (V, E)$ και έναν στόχο κόστους $C \in R$, θα μπορούσε να ανατεθεί μια ενσωμάτωση σε κάθε κορυφή και να στείλουμε μηνύματα κατά μήκος των ακμών, αλλά με αυτόν τον τρόπο θα χανόταν πληροφορίες σχετικά με τα βάρη των ακμών. Αντί αυτού, αναθέτετε επιπλέον ενσωμάτωση στις ακμές, οι οποίες μπορούν να τροφοδοτηθούν με τα αντίστοιχα βάρη τους (οι ενσωματώσεις ακμών στα GNN έχουν επιδείξει επιτυχία σε πολλές εφαρμογές (Battaglia κ.α. 2018) [64]). Σε αυτό το πλαίσιο, αντικαθιστάτε ο πίνακας γειτνίασης κορυφής προς κορυφή $A \in \{0, 1\}^{|V|\times|V|}$, με έναν πίνακα γειτνίασης ακμής προς κορυφή $EV \in \{0, 1\}^{|E|\times|V|}$, ο οποίος συνδέει κάθε ακμή e_i = (s, t, w) με τις πηγή και προορισμό κορυφές της. Διότι το μοντέλο χρειάζεται επίσης να γνωρίζει την τιμή του στόχου κόστους, αποφάσισαν να τροφοδοτήσουν το C σε κάθε ενσωμάτωση ακμής μαζί με το αντίστοιχο βάρος της: δεδομένου ενός στόχου κόστους C, για κάθε ακμή ei = (s, t, w) συνενώνεται το w και το C για να λάβουμε έναν 2διάστατο διάνυσμα ∈ R^2. Αυτό το διάνυσμα τροφοδοτείται σε έναν πολυεπίπεδο αντίληψης (MLP) που το μετατρέπει σε $E^{(1)}[i] \in R^d$, την αρχική ενσωμάτωση για την ακμή e_i. Μετά την αρχικοποίηση αυτή, το μοντέλο υποβάλλεται σε έναν συγκεκριμένο αριθμό επαναλήψεων, κατά τις οποίες οι κορυφές και οι ακμές ανταλλάσσουν μηνύματα και βελτιώνουν τις ενσωματώσεις τους, μέχρι τελικά οι βελτιωμένες ενσωματώσεις ακμής να τροφοδοτηθούν σε ένα MLP που υπολογίζει μια λογική πιθανότητα που αντιστοιχεί στην πρόβλεψη του μοντέλου για την απάντηση του προβλήματος απόφασης.

Συνολικά, κατά την εκπαίδευση, το προτεινόμενο μοντέλο μας μαθαίνει επτά εργασίες:

1. Για να φτιαχτεί ένα R^d διάνυσμα, το οποίο θα χρησιμοποιηθεί για να αρχικοποιήσει όλες τις κορυφές
2. Μια μέθοδος Einit : $R^2 \rightarrow R^d$ για τον υπολογισμό της αρχικής ενσωματωμένης ακμής, παίρνοντας ως δεδομένο το βάρος ακμής w και του κόστους διαδρομής C (MLP)
3. Μια μέθοδος Vmsg : $R^d \rightarrow R^d$ για τον υπολογισμό των μηνυμάτων που θα σταλούν από τις ακμές (edges), δεδομένου μιας ενσωμάτωσης κορυφής (MLP)
4. Μια μέθοδος Emsg : $R^d \rightarrow R^d$ για τον υπολογισμό μηνύματος για την αποστολή στης κορυφές (nodes), δεδομένου της ενσωμάτωσης μιας ακμής (MLP)
5. Μια μέθοδος Vu : $R^{2d} \rightarrow R^{2d}$ για να μπορέσει να υπολογίσει και να ενημερώσει την ενσωμάτωση κορυφής (επιπλέον και την ενημερωμένη κατάσταση του κρυφού κόμβου του RNN), έχοντας δεδομένη την τρέχουσα κατάσταση του κρυφού κόμβου του RNN και ενός μηνύματος
6. Μια μέθοδος Eu : $R^{2d} \rightarrow R^{2d}$ για να υπολογίσει και να ενημερώσει της ενσωμάτωσης της ακμής (επιπλέον και την ενημερωμένη κατάσταση του κρυφού κόμβου του RNN), έχοντας δεδομένη την τρέχουσα κατάσταση του κρυφού κόμβου του RNN και ενός μηνύματος
7. Μια μέθοδος Evote : $R^d \rightarrow R^1$ για να υπολογίσει μια πιθανότητα logit δεδομένης της ενσωμάτωσης μιας ακμής (MLP)

4.2.2 Εκπαίδευση του μοντέλου

Για να κατορθωθεί η εκπαίδευση του μοντέλου GNN, θα χρειαστεί να δοθούν στο μοντέλο τέσσερις εισόδους: με τους πίνακες S, T ∈ $\{0, 1\}^{|E|\times|V|}$, τα βάρη των ακμών D, και μια μεταβλητή για τον στόχο κόστους $C \in R$. Το μοντέλο εκπαιδεύεται με την χρήση του αλγόριθμου Stochastic Gradient Descent (SGD) [66], πιο συγκεκριμένα χρησιμοποιώντας την υλοποίηση του TensorFlow's Adam (Kingma και Ba 2014) [67], έτσι ώστε να κατορθωθεί η ελαχιστοποιήσει του σφάλματος της δυαδικής αναπόφευκτης απώλειας μεταξύ της πρόβλεψης του και της αληθινής τιμής (μια δυαδική τιμή που υποδηλώνει αν η απάντηση στο πρόβλημα απόφασης είναι ΝΑΙ ή ΟΧΙ).

Για να κατορθωθεί επιτάχυνση στην εκπαίδευση, θεωρείται πιο βολικό να εκτελείται SGD σε πακέτα με πολλαπλά παραδείγματα. Αυτό μπορεί να επιτευχθεί εκτελώντας την αποσύνδεση της ένωσης μεταξύ όλων των γραφημάτων στο πακέτο, παράγοντας ένα "γράφημα" πακέτου με n αποσυνδεδεμένα γραφήματα. Εξαιτίας αυτού τα μηνύματα δεν θα διασχίσουν οποιονδήποτε από αυτούς και δεν θα υπάρχει καμία αλλαγή στη διαδικασία εμπλουτισμού της ενσωμάτωσης σε σύγκριση με μια μόνη εκτέλεση. Θα υπολογιστούν πιθανότητες logit για κάθε ακμή στο γράφημα πακέτου, οι οποίες μπορούν να διαθέτουν μέσες τιμές ανάμεσα στα ατομικά παραδείγματα για να υπολογιστεί μια πρόβλεψη για καθένα από αυτά. Το δυαδικό σφάλμα αναπόφευκτης απώλειας μπορεί στη συνέχεια να υπολογιστεί μεταξύ αυτών των προβλέψεων και των αντίστοιχων λύσεων του προβλήματος απόφασης.

Φτιάχνονται παραδείγματα εκπαίδευσης παίρνοντας δείγματα $n \sim U(20, 40)$ τυχαίων σημείων σε ένα τετράγωνο από $a\frac{\sqrt{2}}{2} \times \frac{\sqrt{2}}{2}$ και συμπληρώνοντας έναν πίνακα απόστασης D ∈ $R^{n\times n}$ με την ευκλείδεια απόσταση μεταξύ κάθε ζεύγους σημείων. Οι συγκεκριμένες αποστάσεις, κατά κατασκευή, είναι ∈ [0, 1]. Παράγεται επίσης ένας πλήρης πίνακας **γειτνίασης*** A ∈ $\{0, 1\}^{n\times n}$, και με αυτό το τρόπο λύνεται το αντίστοιχο πρόβλημα TSP χρησιμοποιώντας το TSP Concorde (Hahsler και Hornik 2007) για να βρεθεί το βέλτιστο κόστος διαδρομής. Συνολικά παρήχθησαν 2^{20} τέτοια γραφήματα, από τους οποίους επιλέγεται συνολικά 1024 ανά εποχή για να εξασφαλιστεί ότι η πιθανότητα του μοντέλου να δει τον ίδιο αποτέλεσμα δύο φορές κατά την εκπαίδευση είναι χαμηλή.

Τέλος, για κάθε γράφημα G με βέλτιστο κόστος περιήγησης C^*, παράγονται δύο περιπτώσεις απόφασης $X^+ = (G, 1.02C^*)$ και $X^- = (G, 0.98C^*)$ όπου οι απαντήσεις είναι κατασκευασμένες να είναι ΝΑΙ και ΟΧΙ αντίστοιχα. Κατά αυτόν τον τρόπο εκπαιδεύεται το μοντέλο να προβλέπει το πρόβλημα απόφασης με ένα 2% θετική ή αρνητική απόκλιση από το βέλτιστο κόστος περιήγησης.

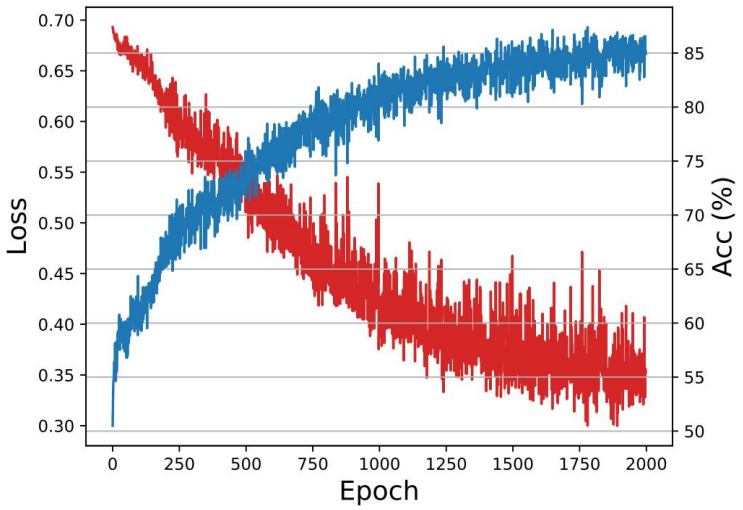
Το μοντέλο αρχίζει με ενσωματώσεις 64 διαστάσεων για τους κόμβους και τις ακμές και πολυεπίπεδα (64,64,64) MLPs με μη γραμμικά ReLU ως ενεργοποιήσεις για όλα τα επίπεδα εκτός από το τελευταίο, το οποίο έχει μια γραμμική ενεργοποίηση. Το μοντέλο εκτελείται για Tmax = 32 βήματα μετάδοσης μηνυμάτων.

4.2.3 Αποτελέσματα Πειραμάτων και Αναλύσεις

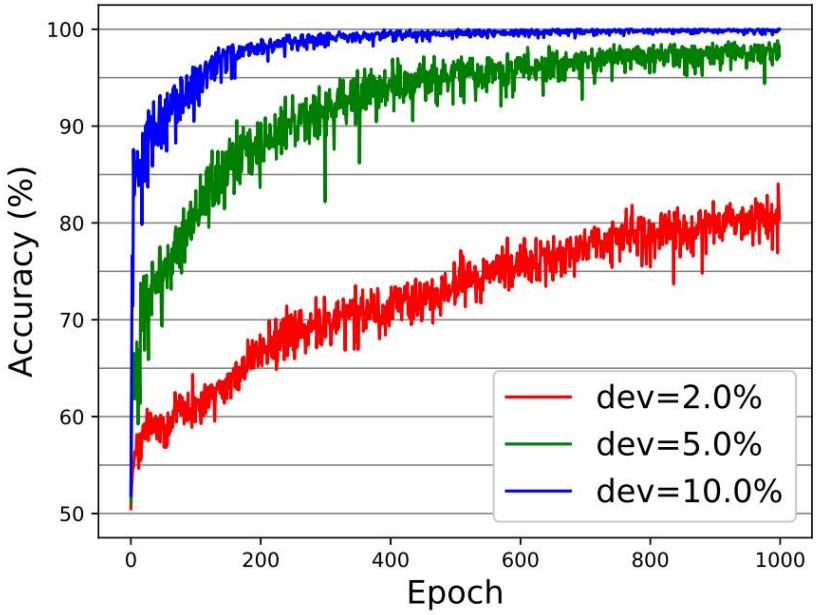
Χρησιμοποιώντας περίπου 2000 εποχές εκπαίδευσης, το μοντέλο κατόρθωσε ακρίβεια *80,16%* σε μέση τιμή πάνω στα 2^{21} παραδείγματα του σετ εκπαίδευσης, έχοντας επίσης κατορθώσει ακρίβεια *80%* μέσα σε ένα σύνολο με *2048* δεδομένα που δεν είχε δει ποτέ πριν. Τα παραδείγματα από τα σετ εκπαίδευσης και ελέγχου παράχθηκαν με την ίδια διαμόρφωση (n ~ U(20, 40) και ποσοστό απόκλισης 2%).

*Στα μαθηματικά και στην επιστήμη υπολογιστών, ο πίνακας γειτνίασης χρησιμοποιείται για να αναπαραστήσει τις κορυφές ενός γραφήματος, οι οποίες συνδέονται με άλλες κορυφές

Το Σχήμα 4.1 δείχνει την εξέλιξη της απώλειας της δυαδικής αντίθεσης εντροπίας και ακρίβειας κατά τη διάρκεια της διαδικασίας εκπαίδευσης. Σημειώστε ότι είναι πολύ πιο εύκολο να εκπαιδεύσετε το μοντέλο με πιο χαλαρές αποκλίσεις από τον βέλτιστο κόστος, όπως δείχνει το Σχήμα 4.2.



Σχήμα 4.1: Εξέλιξη της απώλειας δυαδικής αντίθεσης εντροπίας και ακρίβειας [66]

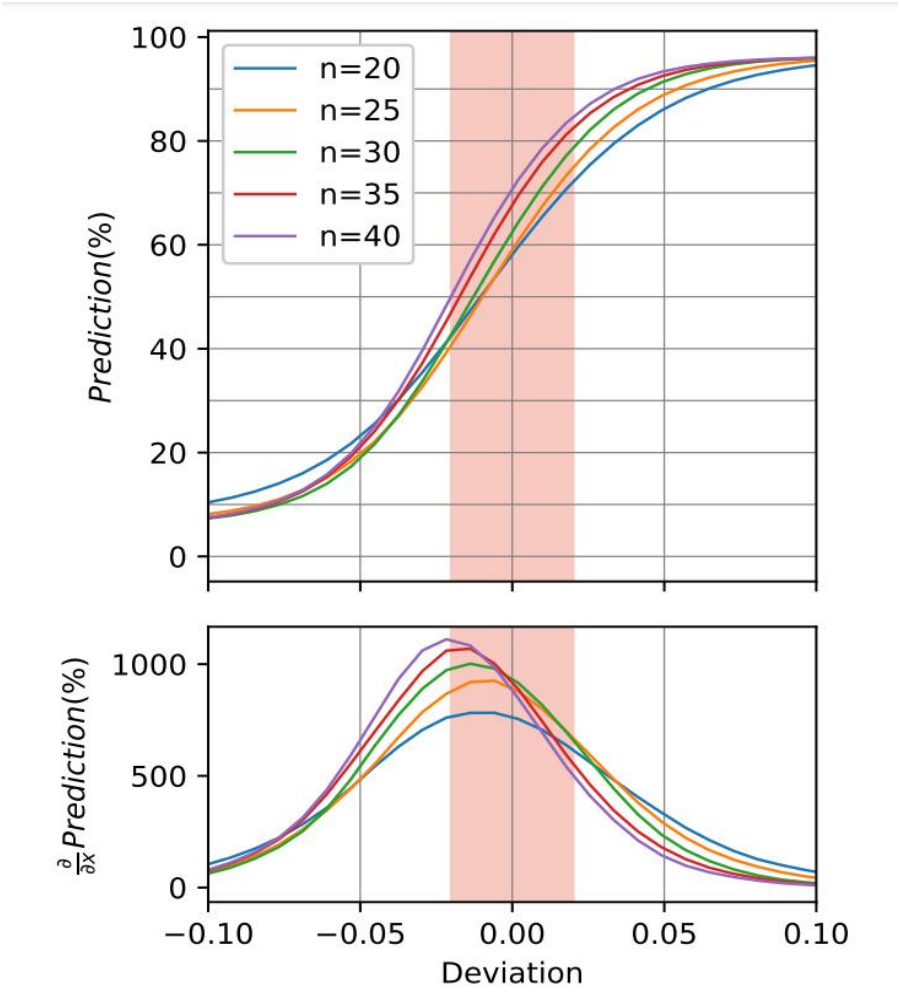


Σχήμα 4.2: Γράφημα χαλαρών αποκλίσεων από τον βέλτιστο κόστος [66]

4.2.4 Εξαγωγή του κόστους διαδρομής

Έχοντας δημιουργήσει μοντέλο για να επιλύει τα προβλήματα απόφασης TSP, υπάρχει η δυνατότητα να χρησιμοποιηθεί για να παραχθούν οι προβλέψεις κόστους διαδρομής με μια λογική απόκλιση από το βέλτιστο κόστος. Το Σχήμα 4.3 δείχνει πώς το μοντέλο λειτουργεί όταν του δίνεται να επιλύσει το

πρόβλημα απόφασης για μεταβλητά κόστη στόχου. Από το χαρακτηριστικό S-σχήμα μπορούμε να καταλάβουμε ότι το μοντέλο με βεβαιότητα αναγνωρίζει ότι διαδρομές με πολύ μικρό κόστος δεν υπάρχουν και επίσης με βεβαιότητα ότι διαδρομές με πολύ μεγάλο κόστος υπάρχουν. Ανάμεσα σε αυτά τα δύο όρια, η πρόβλεψη υποβάλλεται σε μια φάση μετάβασης, με το μοντέλο να γίνεται όλο και πιο αβέβαιο καθώς πλησιάζει σε μηδενική απόκλιση από το βέλτιστο κόστος. Στην πραγματικότητα, αυτή η "καμπύλη αποδοχής" που σχεδιάζεται για μεταβαλλόμενα μεγέθη παραδειγμάτων θυμίζει τις φάσεις μετάβασης σε δύσκολα συνδυαστικά προβλήματα όπως το SAT (Dudek, Meel, και Vardi 2016) [68], μαζί με έναν μεγάλο αριθμό από προβλήματα NP-Complete, το TSP έχει δείξει να εμφανίζει φαινόμενα μετάβασης φάσης (Kirkpatrick και Toulouse 1985; Zhang 2004) [69] [70].



Σχήμα 4.3: Προβλέψεις της απόκλισης μεταξύ του επιθυμητού κόστους και του βέλτιστου κόστους [66]

Το πιο σημαντικό, γνωρίζοντας από τα θεωρητικά αποτελέσματα, ο μέσος όρος από το μήκος του TSP για ένα σύνολο n τυχαίων (ομοιόμορφα κατανεμημένων) σημείων σε ένα επίπεδο είναι ασυμπτωτικά ανάλογο του $\sqrt{n}$, με τη δισδιάστατη "σταθερά TSP" β(2) να την διαθέτουν σαν παράγοντας αναλογίας (Beardwood, Halton, και Hammersley 1959) [71]. Ως αποτέλεσμα, μεγάλα προβλήματα δέχονται αναλογικά πιο μικρές διαδρομές από μικρότερα προβλήματα, ένα γεγονός που θεωρείται, ότι εμφανίζεται στις καμπύλες του Σχήματος 4.3: για παραχθεί κοντά στο μηδέν, το μοντέλο έχει μεγαλύτερη βεβαιότητα ότι υπάρχει μια διαδρομή όσο μεγαλύτερο είναι το μέγεθος του προβλήματος.

Ως συνέπεια, το κρίσιμο σημείο (η απόκλιση στην οποία το μοντέλο αρχίζει να μαντεύει ΝΑΙ) υποστηρίζει μια μετατόπιση προς τα αριστερά όσο αυξάνεται το μέγεθος του προβλήματος, όπως φαίνεται στις παραγώγους των καμπυλών στο Σχήμα 4.3.

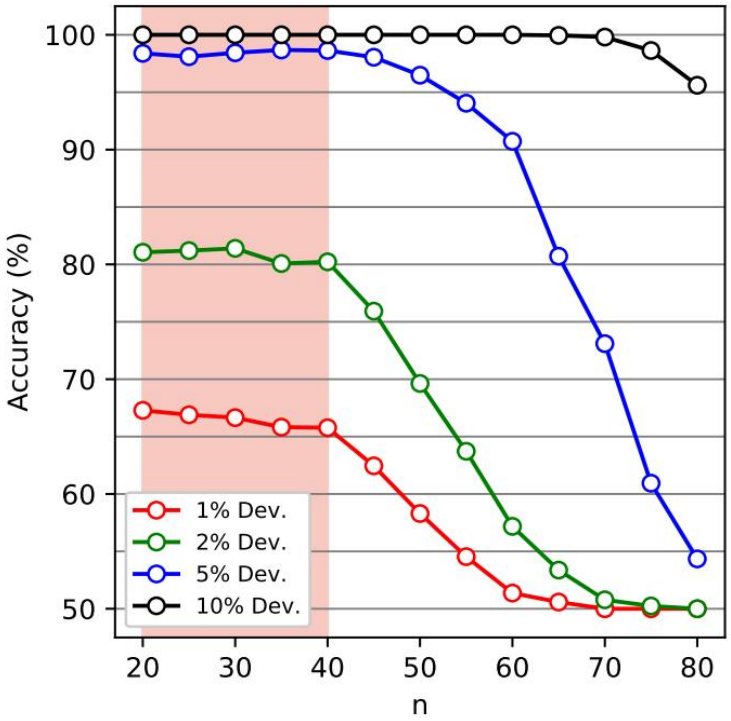
Προσθέτοντας, όλες οι καμπύλες αποδοχής είναι πάνω από τη γραμμή του 50% για απόκλιση = 0, από την οποία εικάζεται ότι το μοντέλο που έχει εκπαιδευτεί, υποθέτει από προεπιλογή ότι υπάρχει μια διαδρομή και προσπαθεί να αποδείξει αυτήν την πρόταση κατά τη διάρκεια των επαναλήψεων της ανταλλαγής μηνυμάτων. Ένα φαινομενικά ενδιαφέρον γεγονός που θα πρέπει να αναφερθεί είναι ότι αυτή η συμπεριφορά είναι αντίθετη από αυτή του GNN SAT-solver NeuroSAT (Selsam κ.α. 2018) [65], εκ του οποίου το μοντέλο υποθέτει από προεπιλογή UNSAT και αλλάζει την πρόβλεψή του μόνο όταν βρίσκει μια ικανοποιητική ανάθεση. Οι παράγοντες που καθορίζουν ποια θα είναι η στρατηγική που το μοντέλο θα μάθει παραμένει ένα ανοιχτό ερώτημα, με προσπάθεια θεωρείται ότι είναι εφικτό να δημιουργηθεί ένα σύνολο εκπαίδευσης που επιβάλλει στο μοντέλο να μάθει ένα αλγόριθμο που είναι αρνητικός από προεπιλογή.

Σύμφωνα με το Σχήμα 4.3 οι καμπύλες πλησιάζουν αυθαίρετα το μηδέν καθώς προχωράει προς μικρότερες αποκλίσεις, αλλά προχωρώντας παρατηρείται ότι το μοντέλο αρχίζει να χάνει την εμπιστοσύνη του ότι υπάρχει μια διαδρομή όταν του δίνονται μεγάλες επιθυμητές τιμές κόστους (≈ 100% απόκλιση). Αυτό απονέμεται πιθανότατα στο γεγονός ότι, εκπαιδευμένο με αποκλίσεις -2%, +2%, το μοντέλο δεν έχει δει ποτέ επιθυμητά κόστη τόσο μεγάλα. Ευτυχώς, αυτό μπορεί να διορθωθεί ξανά εκπαιδεύοντας το για έναν μόνο κύκλο εκπαίδευσης με αποκλίσεις -2%, +2%, +100%, +200%, +1000%, χωρίς σημαντικές επιπτώσεις στη δοκιμαστική ακρίβεια. Με λογική, αν δεν γνωρίζεται τίποτα για το βέλτιστο κόστος, μπορεί να θεωρηθεί ότι είναι πιο κοντά στην τιμή του όταν οι προβλέψεις του μοντέλου είναι κοντά στο 50%. Τότε γίνεται να προβλεφθεί ένα αρχικό κόστος και έτσι στην συνέχεια να εκτελεστεί μια δυαδική αναζήτηση στον άξονα x όπως φαίνεται και στο Σχήμα 4.3.

Με την χρήση του αλγορίθμου από τους M. Prates κ.α. [66], και δίνοντας τιμή στο δ= 0.01 και χρησιμοποιώντας τα βάρη μετά την εκπαίδευση και τον έναν κύκλο εκπαίδευσης για μεγαλύτερες αποκλίσεις, το μοντέλο μπορεί να προβλέψει το κόστος της διαδρομής με μέση απόλυτη απόκλιση 1.5% από το βέλτιστο, εκτελώντας κατά μέσο όρο 8.9 επαναλήψεις στο σύνολο δεδομένων ελέγχου (1024 γραφήματα με n πόλεων, όπου $n \sim U(20, 40)$).

4.2.5 Η απόδοση του μοντέλου σε μεγαλύτερα προβλήματα

Το μοντέλο εκπαιδεύτηκε σε προβλήματα με το πολύ n = 40 πόλεις, αλλά οι συγγραφείς του μοντέλου [66] θέλανε να δουν σε ποιο βαθμό ο μαθών αλγόριθμος γενικεύει σε μεγαλύτερα μεγέθη προβλημάτων. Πήρανε το μέσο όρο της ακρίβειας του εκπαιδευμένου μοντέλου σε σύνολα δοκιμής από 1024 παραδείγματα για διάφορες τιμές του n, όπως φαίνεται στο Σχήμα 4.4. Βρέθηκε ότι το μοντέλο διατηρεί μια ακρίβεια άνω του 80% σε όλο το εύρος των μεγεθών που εκπαιδεύτηκε, αλλά χάνει σταδιακά απόδοση για μεγαλύτερα προβλήματα μέχρι να φτάσει στη βάση του 50%. Επίσης, όπως αναμενόταν από τις καμπύλες αποδοχής στο Σχήμα 4.3, το μοντέλο έχει καλύτερη απόδοση για μεγαλύτερες αποκλίσεις (π.χ. 5%, 10%) και χειρότερη απόδοση για μικρότερες αποκλίσεις (π.χ. 1%). Παρατηρείται όμως ότι ένα πρόβλημα διπλάσιου μεγέθους θα απαιτούσε 2^n φορές περισσότερο χρόνο για υπολογισμό από παραδοσιακούς αλγορίθμους, και συνεπώς αναμένεται μια ταχεία μείωση της ακρίβειας.



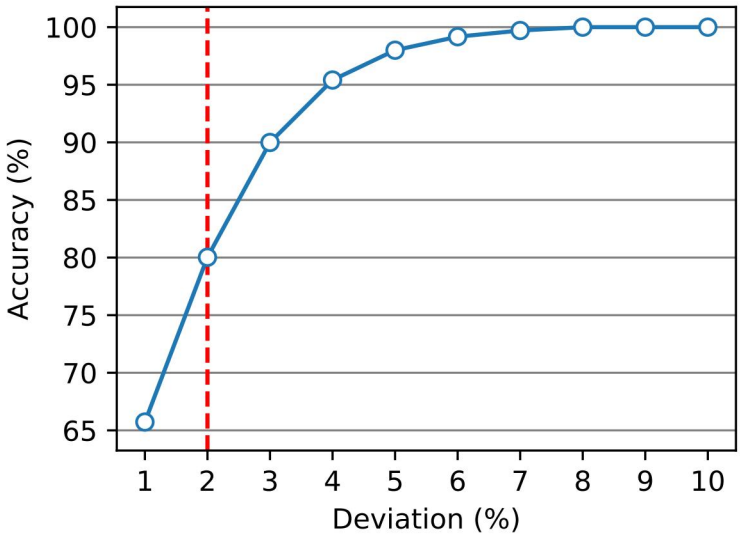
Σχήμα 4.4: Γράφημα Ακρίβειας σε σχέση με τον αριθμό των δεδομένων [66]

4.2.6 Γενικεύοντας τη μεγαλύτερη απόκλιση

Βλέποντας τις καμπύλες αποδοχής στο Σχήμα 4.3 όσο και την ακρίβεια στις καμπύλες στο Σχήμα 4.4 δείχνουν ότι το μοντέλο γενικεύεται σε μεγαλύτερες αποκλίσεις από το βέλτιστο κόστος διαδρομής από το 2% που εκπαιδεύτηκε. Στην πραγματικότητα, αυτές οι καμπύλες υποδεικνύουν ότι το μοντέλο γίνεται όλο και πιο σίγουρο όσο μεγαλύτερη είναι η απόκλιση, αυτό είναι κάτι που δεν θεωρείται απρόσμενο διότι οι αντίστοιχες περιπτώσεις απόφασης είναι σχετικά πιο χαλαρές. Η Σχήμα 4.5 δείχνει πώς η ακρίβεια αυξάνεται μέχρι να σταθεροποιηθεί στο ≈ 100% για αυξανόμενες αποκλίσεις, ενώ βλέποντας τον Πίνακας 4.1 καταλαβαίνουμε ότι απεικονίζει αυτά τα αποτελέσματα στο δοκιμαστικό σύνολο επικύρωσης.

Deviation	Accuracy(%)
1	66
2	80
5	98
10	100

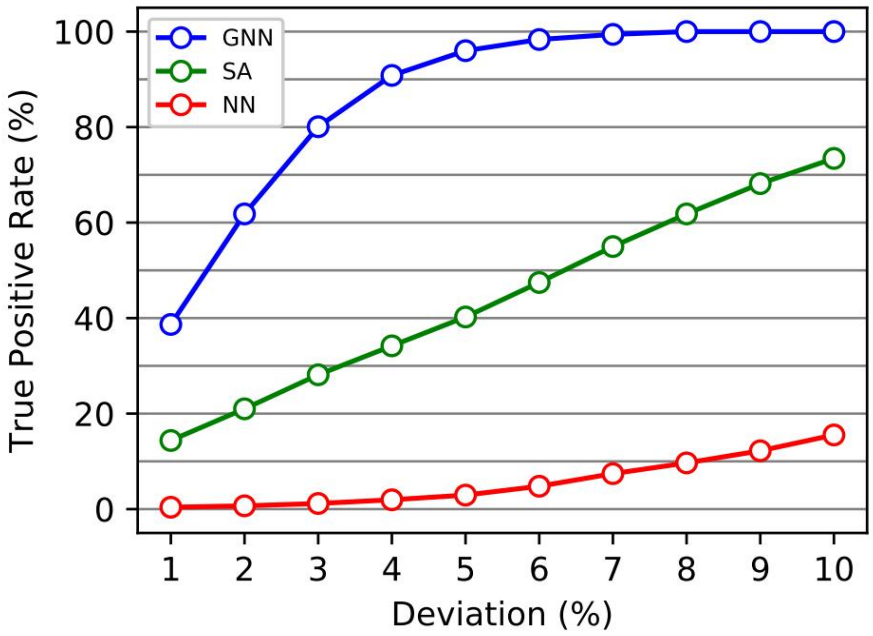
Πίνακας 4.1: Μέσος όρος ακρίβειας σε 1024 αριθμό πόλεων στο κομμάτι του test



Σχήμα 4.5: Γράφημα ακρίβειας εκπαιδευμένου μοντέλου ως προς την απόκλιση [66]

4.2.7 Σύγκριση Βάσης

Σε αυτή την έρευνα αποφάσισαν να εκπαιδεύσουν το μοντέλο με περιπτώσεις αποφάσεων με απόκλιση -2%, +2% από το βέλτιστο κόστος περιήγησης, διότι το 2% είναι η μικρότερη απόκλιση για την οποία το δίκτυο μπορούσε να εκπαιδευτεί μέσα σε λογικό χρόνο (≤ 2000 εποχές). Έχοντας αυτό ως παράγοντα, θα πρέπει να αναφερθεί ότι αυτό το μοντέλο δεν είναι για να βρει την ¨ΒΈΛΤΙΣΤΗ¨ τιμή στο πρόβλημα TSP αλλά για να δείξει πως ένα νευρωνικό δίκτυο είναι δυνατόν να μάθει να λύνει τέτοια προβλήματα με ελάχιστη επίβλεψη (δύο bits: ένα bit για θετική λύση και ένα bit για αρνητική), σε αυτό το κομμάτι της ερευνάς θα δοθεί ως μέτρο σύγκρισης τα μοντέλα (1) "Πλησιέστερου Γείτονα" (Nearest Neighbor) και (2) έναν αλγόριθμο "Προσομοίωσης Ανόδου" (Simulated Annealing) (Kirkpatrick, Gelatt, and Vecchi 1983) [72]. Ο "Πλησιέστερος Γείτονας" είναι ίσως η απλούστερη προσεγγιστική μέθοδος για το TSP, ως αποτέλεσμα παράγει συνήθως λύσεις χαμηλής ποιότητας. Η "Προσομοίωση Ανόδου" μπορεί γενικά να παράγει καλές διαδρομές για το ευκλείδειο TSP, αν οι μετά-παράμετροι είναι καλά βαθμολογημένες. Στο **Σχήμα 4.6** δείχνει το True Positive Rate (TPR) του εκπαιδευμένου μοντέλου με τη συχνότητα με την οποία αυτές οι δύο προσεγγίσεις μπορούν να παράγουν διαδρομές με μια δεδομένη απόκλιση από τη βέλτιστη διαδρομή. Αυτή η συχνότητα θεωρείτε ως το TPR που προκύπτει από τη μετατροπή αυτών των μεθόδων σε προβλεπόμενες για την αποφασιστική εκδοχή του ίδιου προβλήματος (εάν μπορούν τα μοντέλα να αποδείξουν με εποικοδομητικό τρόπο ότι υπάρχει μια διαδρομή με κόστος εντός του στόχου, τότε μαντεύουν NAI, αλλιώς ΟΧΙ). Για το σύνολο τον δεδομένων που μελέτησε το paper [66] (1024 γραφήματα με n πόλεις, όπου n ~ U(20, 40)), ο "Πλησιέστερος Γείτονας" παράγει μέσο όρο διαδρομές που είναι 20,2% μεγαλύτερες από την βέλτιστη, ενώ η "Προσομοίωση Ανόδου" μειώνει αυτόν τον αριθμό στο 6,7%. Ωστόσο, για όλες τις ελεγμένες αποκλίσεις, το εκπαιδευμένο μοντέλο GNN υπερτερεί και των δύο μεθόδων, παρουσιάζοντας TPR > 90% από αποκλίσεις 4% και άνω.



Σχήμα 4.6: Γράφημα True Positive Rate του εκπαιδευμένου μοντέλου ως προς την απόκλιση [66]

4.2.8 Γενικεύοντας σε άλλους τρόπους Διανομής

Αν και το μοντέλο έχει εκπαιδευτή σε δυϊκό γράφημα ευκλείδειας διάστασης, μπορεί να γενικευτεί. Για να αξιολογηθεί η αξιοπιστία σε αυτό το γεγονός, παίρνοντας δύο οικογένειες γραφημάτων που προέρχονται από ομοιόμορφα τυχαίους πίνακες απόστασης: στην πρώτη κατανομή ("Τυχαίο" στον πίνακα 4.2), τα βάρη του μοντέλου επιλέγονται ομοιόμορφα σε τυχαία επιλογή, στη δεύτερη ("Τυχαίος Μετρικός" στον πίνακα 4.2), τα βάρη επιλέγονται αρχικά ομοιόμορφα με τυχαία επιλογή και στη συνέχεια επιβάλλεται η μετρική ιδιότητα αντικαθιστώντας τα βάρη με την ελάχιστη απόσταση μεταξύ των αντίστοιχων κορυφών. Για απόκλιση 2% από το βέλτιστο κόστος διαδρομής, το μοντέλο κατάφερε να επιτύχει ακρίβεια 64% στις τυχαίες μετρικές περιπτώσεις (έναντι 80% στις ευκλείδειες), αλλά η απόδοση γίνεται καλύτερη για πιο χαλαρές αποκλίσεις, με 82% με 5% απόκλιση και 96% με 10% απόκλιση από το βέλτιστο κόστος διαδρομής. Το μοντέλο δεν μπόρεσε να επιτύχει απόδοση πάνω από το 50% βάση για μη-μετρικές περιπτώσεις.

Επίσης, έγινε αξιολόγηση στο μοντέλο με πραγματικές περιπτώσεις από το σύνολο δεδομένων Tsplib95 (Reinelt 1995) [66], για τις οποίες αναφέρονται τα αποτελέσματα που πέτυχε το εκπαιδευμένο μοντέλο με τον Αλγόριθμο 4.2 στον πίνακα 4.3. Γενικά, όταν λαμβάνεται υπόψιν η απόλυτη σχετική απόκλιση, το GNN υπερτερεί την διαδικασία Προσομοίωσης Ανόδου για 6 από τις 9 περιπτώσεις.

Deviation	Accuracy(%)		
	Euc.2D	Rand.Metric	Rand
1	66	57	50
2	80	64	50
5	98	82	50
10	100	96	50

Πίνακας 4.2: Μέσος όρος ακρίβειας σε 1024 αριθμό πόλεων στο κομμάτι του test μαζί με random metric

4.2.9 Εφαρμογή και Αναπαραγωγικότητα

Σε αυτή την ενότητα θα γίνει αναφορά για τις παραμέτρους του μοντέλου. Το μέγεθος των ενσωματώσεων επιλέχθηκε ως d = 64, όλα τα MLP που σε αυτά περνούν μηνύματα έχουν τρία επίπεδα με μεγέθη επιπέδων (64, 64, 64) και μη γραμμικό ReLU ως ενεργοποίηση για όλα τα επίπεδα εκτός από το τελευταίο, διότι το τελευταίο έχει γραμμική ενεργοποίηση. Τα MLP που χρησιμοποιούνται για την αρχικοποίηση των ενσωματώσεων των ακμών έχουν επίσης τρία επίπεδα με μεγέθη επιπέδων (8, 16, 32). Όλοι οι πυρήνες των βαρών w ξεκινούν με τη μέθοδο Xavier του TensorFlow, ενώ οι παράμετροι πόλωσης ξεκινούν με μηδενικά. Η μονάδα που χρησιμοποιεί για επαναληπτική ανανέωση των ενσωματώσεων είναι ένα επίπεδο LSTM με Layer Normalization (Ba, Kiros και Hinton, 2016) [73] έχοντας το ReLU ως ενεργοποίηση και μαζί οι πυρήνες των βαρών και οι παράμετροι πόλωσης ξεκινούν με τον Glorot Uniform Initializer του TensorFlow (Glorot και Bengio, 2010) [74], έχοντας υπόψιν την προσθήκη ότι η παράμετρος πόλωσης της forget gate αυξάνεται κατά 1. Ο αριθμός του χρόνου βημάτων μετάδοσης μηνυμάτων ορίζεται στο tmax = 32. Για κάθε παράδειγμα γραφήματος, δημιουργείται ένα ζευγάρι αποφάσεων: ένα αρνητικό παράδειγμα με στόχο κόστους 2% μικρότερο από το βέλτιστο και ένα θετικό παράδειγμα με στόχο κόστους 2% μεγαλύτερο από το βέλτιστο. Τα παραδείγματα εκπαίδευσης μπορούν να δημιουργηθούν τυχαία, αλλά είναι σημαντικό να παραμένουν αυτά τα ζευγάρια μαζί στο ίδιο πακέτο. Κάθε εποχή εκπαίδευσης αποτελείται από 128 λειτουργίες Stochastic Gradient Descent σε πακέτα των 16 ζευγαριών παραδειγμάτων (με θετική και αρνητική απόκλιση), τα δείγματα επιλέγονται τυχαία από το σύνολο δεδομένων εκπαίδευσης.

Στο πρόγραμμα που δημιούργησαν σε αυτή την έρευνα χρησιμοποιούν δικά τους παραδείγματα εκπαίδευσης και για να μειωθεί η υπέρ-προσαρμογή, δημιούργησαν 2^{20} γραφήματα με n πόλεις, όπου $n \sim U(20, 40)$. Τα παραδείγματα μπορούν να ομαδοποιηθούν μαζί εκτελώντας ένωση σε ένα σύνολο n γραφημάτων, δημιουργώντας ένα γράφημα με n συνδεδεμένα δεδομένα, όπου η ροή πληροφοριών δεν "διαρρέει" από ένα δεδομένο σε ένα άλλο. Τέλος, σε όλα τα πειράματα, εκτέλεσαν κανονικοποίηση σε όλα τα βάρη των ακμών ώστε να είναι στο διάστημα [0, 1], και το στόχο κόστους υποβάλλεται σε κανονικοποίηση πάντα ανάλογα με τον αριθμό των πόλεων n.

Instance	Size	Relative GNN	Deviation(%) SA
Ulyssses16[1]	16	-22.80	+1.94
Ulyssses22[1]	22	-27.20	+1.91
Eil51	51	-18.37	+18.07
Berlin52	52	-8.73	+21.45
St70	70	-11.87	+14.47
Eil76	76	-13.91	+19.24
kroA100	100	-2.00	+30.73
Eil101	101	-9.93	+20.46
Lin105	105	+6.37	+17.77

[1]Αυτές οι περιπτώσεις υπολογίστηκαν με τη χρήση του τύπου Haversine (απόσταση greatcircle) για τον υπολογισμό του πίνακα αποστάσεων.

Πίνακας 4.3: Σχετικές αποκλίσεις από το βέλτιστο κόστος διαδρομής

4.3 BGNN for TSP on arbitrary symmetric graphs

4.3.1 Εισαγωγή

Σε αυτή την ενότητα θα μιλήσουμε για ένα δίκτυο γραφημάτων με διπλή κατεύθυνση (BGNN) [75] για το πρόβλημα του περιοδεύοντος πωλητή (TSP) που είναι εφαρμοσμένα σε αυθαίρετα συμμετρικά γραφήματα. Το BGNN εκπαιδεύεται να προβλέπει σειριακά τον επόμενο κόμβο (πόλη) που πρέπει να επισκεφθεί και για να το πετύχει αυτό χρησιμοποιεί μάθηση με απομίμηση. Στο συγκεκριμένο μοντέλο, κατασκευάζεται ένα επίπεδο αμοιβαίας μετάδοσης μηνυμάτων (BMPL), όπου τα μηνύματα περνούν και στις δύο κατευθύνσεις από τους δύο κόμβους του, ώστε το δυναμικό γράφημα να κωδικοποιείται αποτελεσματικά από τις ακμές και τις μερικές λύσεις. Η εκπαίδευση με μάθηση από απομίμηση αποφεύγει το πρόβλημα της δυσκολίας της ανακάλυψης που αντιμετωπίζει η ενισχυτική μάθηση, αλλά για να το καταφέρει αυτό απαιτεί κοντινά βέλτιστες λύσεις για την εκπαίδευση. Η απαίτηση αυτή μπορεί να πραγματοποιηθεί με τη βοήθεια της αναζήτησης, όπως στον AlphaGo Zero (Silver κ.α., 2017) [76].

Τα πειράματα που έχουν εφαρμόσει σε αυτό τον αλγόριθμο αποδεικνύουν ότι το προτεινόμενο BGNN αποτυπώνει αποτελεσματικά πληροφορίες γραφήματος από τις ακμές και τις μερικές λύσεις και είναι ικανό να βοηθήσει στην παραγωγή κοντινά βέλτιστων λύσεων. Υπάρχουν περιπτώσεις που αποτυγχάνει να παράγει έγκυρους κύκλους αυτό γίνεται όταν το γράφημα γίνεται πιο αραιό, η απλή μέθοδος που παράγει έναν μόνο κόμβο που πρέπει να επισκεφθεί. Για να επιλυθεί αυτό το πρόβλημα, η χρήση μιας απλής αναζήτησης με δέσμη αυξάνει σημαντικά την πιθανότητα παραγωγής έγκυρων κύκλων που είναι κοντά στη βέλτιστη λύση. Αλλιώς, οι προβλέψεις του BGNN θα μπορούσαν να χρησιμοποιηθούν ως κατευθυντήριες για πληροφορημένη αναζήτηση, έτσι ώστε να μειωθεί ο αριθμός των κόμβων που αναζητούνται κατά την παραγωγή μιας περίπου βέλτιστης λύσης. Σε αυτήν την ενότητα θα δούμε [75]:

- Ένα δίκτυο γραφημάτων με διπλή κατεύθυνση (BGNN) για το πρόβλημα TSP σε αυθαίρετα συμμετρικά γραφήματα, το οποίο είναι ένα πιο πρακτικό και σημαντικό μοντέλο που έχει αγνοηθεί από προηγούμενες μελέτες.
- Εισαγωγή μιας νέας αρχιτεκτονικής ενσωμάτωσης γραφήματος (bidirectional message passing layer) που είναι ικανή να ενσωματώσει δυναμικά γραφήματα βασισμένα σε πληροφορίες που υπάρχουν στις ακμές και τις μερικές λύσεις.
- Η εκπαίδευση του BGNN χρησιμοποιώντας μάθηση από απομίμηση επιτρέπει την καταγραφή σημαντικών χαρακτηριστικών λήψης αποφάσεων από τα γραφήματα και την εκμάθηση ισχυρών κατευθυντήριων για την παραγωγή κοντινών βέλτιστων λύσεων.
- Οι πειραματισμοί που συγκρίνουν με μετά-ευρηστικούς αλγορίθμους και προσεγγιστικές προσεγγίσεις δείχνουν την αποδοτικότητα του BGNN.

4.3.2 Προκαταρκτικά

Παίρνοντας τα δεδομένα ενός γραφήματος $G = (V, E)$, όπου $V = \{1,2,..,n\}$ είναι το σύνολο των κόμβων και $|V| = n$ είναι ο αριθμός των κόμβων στο γράφημα, και E είναι ένας $n \times n$ πίνακας που καθορίζει το κόστος ταξιδιού μεταξύ οποιουδήποτε ζευγαριού μεταξύ κόμβων. Η καταχώρηση του

πίνακα E στην i-οστή γραμμή, j-οστή στήλη, συμβολίζεται με $w_{ij} \in R+$, και έτσι η αντιστοιχεί στο κόστος ταξιδιού από τον κόμβο i στον κόμβο j. Για αποσυνδεδεμένους κόμβους i και j, θέτουμε $wij = \infty$. Έστω $X \in \{0,1\}n \times n$ ένας πίνακας μετάθεσης, τότε το πρόβλημα του TSP μπορεί να διατυπωθεί ως (4.1), όπου $[n]$ συμβολίζει το σύνολο $\{1, 2, \ldots, n\}$. Για τον κόμβο i στο γράφημα, ορίζουμε τον επόμενο κόμβο που πρόκειται να επισκεφθούμε ως $\eta(i)$ σύμφωνα με την (4.2α)

$$\boldsymbol{X} \in \min_{\{\mathbf{0},\mathbf{1}\}^{n \times n}} \sum_{i=1}^{n} \boldsymbol{w_{ij} x_{ij}} \tag{4.1}$$

$$\boldsymbol{s.t} \sum_{i=1, j \neq i}^{n} \boldsymbol{x_{ij}} = \mathbf{1}, \sum_{i=1, j \neq i}^{n} \boldsymbol{x_{ij}} = \mathbf{1}, \boldsymbol{j} \in [\boldsymbol{n}]$$

$$\sum_{i \in Q} \sum_{j \in Q} \boldsymbol{x_{ij}} \leq [\boldsymbol{Q}] - \mathbf{1}, \boldsymbol{Q} \subseteq [\boldsymbol{n}], |\boldsymbol{Q}| \geq \mathbf{2}$$

Υποθέτουμε ότι η περιοδεία ξεκινά από τον κόμβο s και η περιοδεία π του TSP μπορεί να κατασκευαστεί μετά από n επαναλήψεις, όπως φαίνεται στο (4.2b), όπου η $\eta^k(s)$ σημαίνει τον αυτό-συνδυασμό της συνάρτησης η k φορές. Ορίζουμε τον κόμβο που επισκέπτεται στο βήμα t ως $\pi(t)$.

$$\eta(j) = arg_i \max x_{ji} \tag{4.2a}$$

$$\pi = \{s, \eta(s), \eta^{o2}(s), \ldots, \eta^{on-1}(s)\} \tag{4.2b}$$

Διαφορές ανάμεσα σε Ευκλείδειο TSP και αραιού TSP. Το ευκλείδειο TSP είναι η πιο απλή εκδοχή, είτε με τη βοήθεια του ανθρώπου είτε με τη βοήθεια του υπολογιστή (Larson και Odoni, 1981, βλ. Ενότητα 6.4.7) [77]. Για ένα d-διάστατο ευκλείδειο TSP με n κόμβους, υπάρχει ένας ντετερμινιστικός αλγόριθμος με πολυπλοκότητα $O\left(n^{d+1}(\log n)^{O(c)}\right)$ για να βρεθεί μια (1 + 1/c)- προσέγγιση για οποιοδήποτε c > 1 (Arora, 1998) [78]. Παρόλα αυτά το TSP σε αραιά γραφήματα είναι αυστηρά NP-hard. Δεν υπάρχει αλγόριθμος πολυωνύμων με προσεγγιστικό λόγο ρ >= 1 εκτός εάν P = NP (Cormen κ.α., 2009, βλ. Θεώρημα 35.3) [79].

Το GNN έχει την δυνατότητα να λειτουργεί με καλές επιδόσεις για πολλές εφαρμογές με γραφικά δεδομένα (Duvenaud κ.α., 2015, Battaglia κ.α., 2016, Wang κ.α., 2018) [80] [81] [82]. Το μοντέλο νευρώνων διαβίβασης μηνυμάτων (MPNN) (Gilmer κ.α., 2017) [62] αφαιρεί τα κοινά στοιχεία από αρκετά μοντέλα νευρώνων για δομημένα δεδομένα γραφημάτων. Παίρνοντας σαν δεδομένο ένα γράφημα $G = (V, E)$, υποθέτουμε ότι κάθε κόμβος $v \in V$ συσχετίζεται με ένα χαρακτηριστικό κόμβου f_v και κάθε ακμή συσχετίζεται με ένα χαρακτηριστικό ακμής euw, το MPNN περνά μηνύματα μεταξύ των κόμβων u και v ακολουθώντας τέσσερα βήματα (4.3): παραγωγή μηνυμάτων, συλλογή, πέρασμα και τέλος ενσωμάτωση του γραφήματος. Στην (4.3) N_u συμβολίζει όλους τους γείτονες του κόμβου u. M και Ψ είναι συναρτήσεις που έχουν αλλάξει από νευρωνικά δίκτυα. Η Φ και η h μπορούν να θεωρηθούν ως συναρτήσεις ενεργοποίησης.

$$m_{um} = M(j_u, j_w, e_{uw}); \; l_u = \Phi_{w \in N_u}(m_{uw}) \tag{4.3}$$

$f_u^{t+1} = \Psi\left(f_u^{t+1}, l_u\right); \quad g^t = h_{u \in V}(f_u^t)$

Το σύστημα εξόρυξης συνθέσεων νευρωνικού δικτύου μετάδοσης μηνυμάτων (CMPNN) (Zhang και Lee, 2019) [83] θεωρείτε ένα παράδειγμα του MPNN. Το CMPNN έχει την δυνατότητα και μειώνει τις υπολογιστικές απαιτήσεις και τις απαιτήσεις μνήμης του MPNN με τη χρήση τοπικών συστατικών προσοχής. Το CMPNN απλοποιεί τη δημιουργία και συλλογή μηνυμάτων όπως φαίνεται στο (4.4). Οι παράμετροι θ_k και θ_m αντιπροσωπεύουν παραμέτρους νευρώνα, το Kuw αναπαριστά τη σημασία της ακμής uw που δημιουργείται από το νευρωνικό δίκτυο K και το $\boldsymbol{mu}$ είναι το μήνυμα από τον κόμβο $\boldsymbol{u}$ που παράγεται από το $\boldsymbol{M}$.

$m_u = M(f_u; \theta_m); \quad K_{uw} = \mathbb{K}\ e_{uw}\theta_k$ (4.4)

$l_u = \Phi_{w \in N_u}(K_{uw} m_w)$

4.3.3 Αρχιτεκτονική του Δικτύου

Για να έρθει εις πέρας η επίλυση ενός αυθαίρετου συμμετρικού TSP, σχεδιάζεται ένα διπλής κατεύθυνσης γραφικό νευρωνικό δίκτυο (BGNN). Το BGNN έχει την δυνατότητα γραφήματα να χωριστεί σε δύο μέρη: κωδικοποίηση και αποκωδικοποίηση. Όταν το πρόγραμμα βρίσκεται σε μια πορεία για την λύση του αυθαίρετα συμμετρικού TSP σε ακολουθία, τα εργασιακά αλλάζουν σε κάθε βήμα. Για να προσδιορίσουν χαρακτηριστικά από τέτοια δυναμικά γραφήματα, η κωδικοποίηση αποτελείται από πολλαπλά στοιβαγμένα στρώματα διπλής κατεύθυνσης μετάδοσης μηνυμάτων όπου τα μηνύματα μεταφέρονται από και προς ένα άκρο και στους δύο κόμβους της ακμής. Συνεχίζοντας με την κωδικοποίηση, χρησιμοποιείται ένα μέρος αποκωδικοποίησης που διαθέτει ένα μόνο στρώμα διπλής κατεύθυνσης μετάδοσης μηνυμάτων και ένα απλό πολυεπίπεδο πεπερασμένου περιβλήματος (MLP) για την πρόβλεψη της πιθανότητας επίσκεψης της επόμενης πόλης. Εκπαιδεύοντας το νευρωνικό δίκτυο BGNN με μάθηση με απομίμηση για να μάθουμε μια ισχυρή προσέγγιση που βελτιώνει την πιθανότητα κατασκευής έγκυρων λύσεων υψηλής ποιότητας.

4.3.4 Αρχικοποίηση

Ξεκινώντας θα πρέπει να τεθεί μια περιγραφή για το πως πραγματοποιείται η αρχικοποίηση των χαρακτηριστικών των κόμβων $\vec{f}_i$ και των χαρακτηριστικών των ακμών $\vec{e}_{ij}$, έτσι ώστε το γραφικό νευρωνικό δίκτυο να μπορεί να αναλύσει τα χαρακτηριστικά που έχουν σχέση με τη λήψη αποφάσεων από το γράφημα του TSP σε αυθαίρετα συμμετρικά γραφήματα.

Στην αρχή της εκπαίδευσης σε κάθε βήμα, τα χαρακτηριστικά των κόμβων f ξεκινούν παίρνοντας πληροφορίες από την τρέχουσα μερική λύση, όπως ο τελευταίος κόμβος μαζί επίσης και τρέχων κόμβο που χρησιμοποιούνται για να δοθεί μια κατεύθυνση επέκτασης της λύσης, οι επισκεπτόμενοι κόμβοι όπου δεν θα επισκεφτούν ξανά και τέλος το κέντρο όπου ο πωλητής χρειάζεται να επιστρέψει. Αν ένας κόμβος είναι κέντρο, το μοντέλο τον αναπαριστά από το (1, 0), αλλιώς από το (0, 1). Παρόμοιες αναπαραστάσεις χρησιμοποιούνται για να προστεθούν πληροφορίες του τελευταίου κόμβου, του τρέχοντος κόμβου και των επισκεπτόμενων κόμβων στα χαρακτηριστικά των κόμβων. Μετά τη συνένωση αυτών των πληροφοριών, το fi μπορεί να παρουσιασθεί από ένα διάνυσμα (1×8), επομένως το f είναι ένας πίνακας διαστάσεων (n×8). Το γράφημα g περιλαμβάνει το πολύ έναν αριθμό k από δείκτες των πλησιέστερων γειτόνων για κάθε κόμβο, ταξινομώντας τον πίνακα κόστους (μόνο για συνδεδεμένα ζεύγη κόμβων) με αύξουσα σειρά. Είναι ένας πίνακας διαστάσεων ($n \times k$), τα αρχικά χαρακτηριστικά των ακμών e δείχνουν το πραγματικό κόστος μεταξύ συνδεδεμένων κόμβων

που συλλέγονται από τον αρχικό πίνακα κόστους. Αντιστοιχεί στο γράφημα g και είναι η κύρια πηγή πληροφοριών του γραφήματος. Τα αρχικά χαρακτηριστικά των ακμών e μπορούν να αναπαρασταθούν από έναν πίνακα διαστάσεων ($n \times k \times 1$).

4.3.5 Επίπεδο διέλευσης μηνυμάτων διπλής κατεύθυνσης

Σε αυτήν την ενότητα, θα τεθεί μια περιγραφή για ένα μοναδικό επίπεδο αμφίδρομης μετάδοσης μηνυμάτων (BMPL) εκ του οποίου τα μηνύματα μεταδίδονται αμφίδρομα μεταξύ κόμβων και ακμών. Το BMPL θα χρησιμοποιηθεί σε ένα μεγάλο κομμάτι της αρχιτεκτονικής του BGNN. Στην είσοδο του BMPL περιλαμβάνονται τρία μέρη: ένα σύνολο χαρακτηριστικών των κόμβων, $f\,(n \times F) = \{\overrightarrow{f_1},\ \overrightarrow{f_2}, .., \overrightarrow{f_n}\}$ με $\overrightarrow{f_i} \in RF \cdot$ η λίστα γειτνίασης $g^{n\times k}\cdot$ τα χαρακτηριστικά ακμών $e^{(n\times k\times E)} = \{\vec{e}_{1,g_{11}}, \dots, \vec{e}_{1,g_{1k}};\ \dots; \vec{e}_{n,g_{n1}}, \dots, \vec{e}_{n,g_{nk}}\}$ που αντιστοιχούν στον γράφημα g. Οι έξοδοι που διαθέτουν δύο μέρη: το χαρακτηριστικό κόμβων $\mathbf{f}'^{(n\times F')}$ με $\overrightarrow{f'_i} \in \mathrm{R}F'$ και το χαρακτηριστικό ακμών $\mathbf{e}'^{(n\times k\times E')}$ με $\vec{e}'^{ij} \in RE'$, το γράφημα g παραμένει σταθερό.

$$\mathrm{h_{ij}} = \mathbb{K}\left(\overrightarrow{\mathbf{f_i}}\,.\,\overrightarrow{\mathbf{f_j}}, \overrightarrow{\mathbf{e_{ij}}}\right) \in \mathbb{R}^{\mathrm{H}} \tag{4.5}$$

Για να είναι εφικτό να αποκτηθούν επαρκής πληροφορίες σχετικά με τις αποφάσεις από τα γραφήματα, τις μερικές λύσεις και να ενσωματωθεί η ολοκληρωμένη πληροφορία στα χαρακτηριστικά των κόμβων και των ακμών, το BMPL χρησιμοποιεί αμφίδρομη μετάδοση μηνυμάτων από και προς μια ακμή και στους δύο κόμβους της. Χρησιμοποιείται εδώ local convolution kernel, ο οποίος υποδεικνύει τη σημασία του ζεύγους (i, j, eij) για τον κόμβο i, σημειωμένο ως h_{ij}, υπολογισμένο με τον τύπο (4.5). Ταυτόχρονα, για να επιτευχθεί η σύνθετη μετάδοση μηνυμάτων, χρειάζεται επίσης να σχεδιαστούν μετασχηματισμοί γραμμικής μάθησης. Η λειτουργία μετάδοσης μηνυμάτων μπορεί να γραφεί όπως το (4.6). Οι παράμετροι θ_f και θ_e αντιστοιχούν στους μετασχηματισμούς γραμμικής μάθησης. Το m_{ij} είναι ένα μήνυμα που θα μεταδοθεί από την ακμή e_{ij} στους κόμβους i, j. Το h_{ij} είναι το αντίστοιχο διάνυσμα σημασίας του m_{ij}. Το *[A;B]* συμβολίζει τη συνένωση του διανύσματος A και B. Το $[A \odot B]$ υποδηλώνει το στοιχειοθετικό πολλαπλασιασμό των διανυσμάτων A και B. Οι συναρτήσεις Ψ και ξ αντιπροσωπεύουν τις συναρτήσεις ενεργοποίησης. Η ζ είναι η συνάρτηση συγκέντρωσης, η οποία μπορεί να είναι max, $mean$ ή sum. Σε αυτό το πρόγραμμα χρησιμοποιείται η συνάρτηση ενεργοποίησης Relu και η συνάρτηση συγκέντρωσης max.

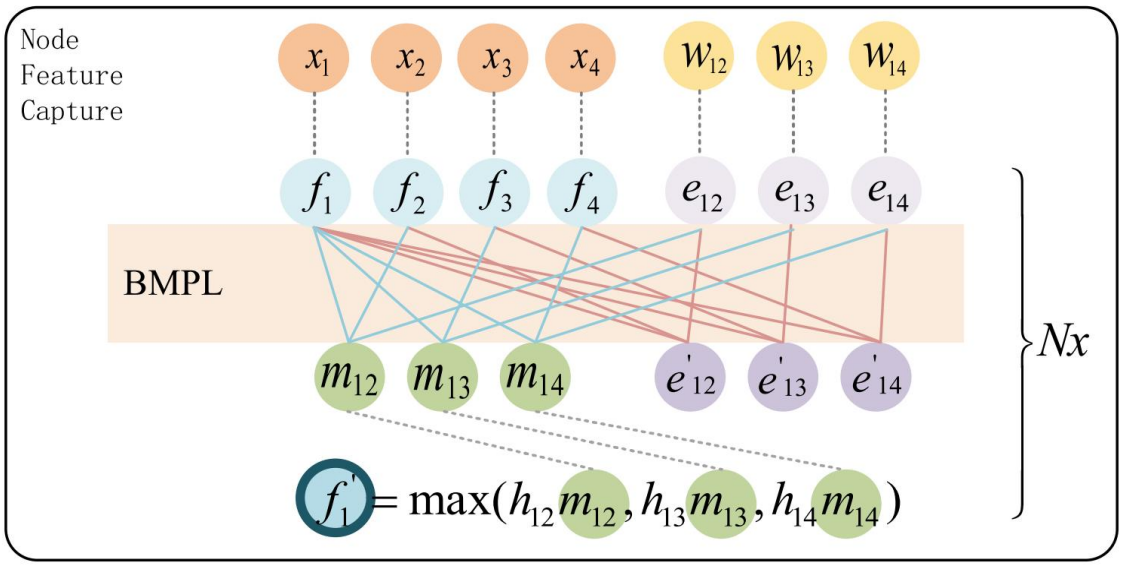
$$m_{il} = \Psi\left(\theta_f^{H\times F'\times 2F}\left[\overrightarrow{f_i}; \overrightarrow{fj}\right] \odot \theta_f^{H\times F'\times E} e_{ij}\right) \tag{4.6}$$

$$\overrightarrow{f'_i} = \zeta_{j\in Ni}\left(h_{ij} M_{ij}\right)$$

$$\overrightarrow{e'_{ij}} = \xi\left(\theta_e^{E'\times 2F}\left[\overrightarrow{f_i}; \overrightarrow{f_j}\right] \odot \theta_e^{E'\times E} e_{ij}\right)$$

Στην Σχήμα 4.7 δείχνει τη διαδικασία αμφίδρομης μετάδοσης μηνυμάτων μέσω ενός παραδείγματος ενημέρωσης των χαρακτηριστικών των κόμβων και των ακμών που σχετίζονται με τον κόμβο 1, όπου x_1, x_2, x_3, x_4 είναι τα χαρακτηριστικά εισόδου των κόμβων και x_2, x_3, x_4 είναι συνδεδεμένα με τον $x1$ με αρχικά βάρη w_{12}, w_{13}, w_{14} αντίστοιχα. Αυτά είναι τα χαρακτηριστικά εισόδου για τον πρώτο

BMPL. Τα εισερχόμενα χαρακτηριστικά των επόμενων στρωμάτων είναι οι έξοδοι του τελευταίου BMPL. Τα μηνύματα m_{12}, m_{13}, m_{14} μεταφέρονται από τις ακμές e_{12}, e_{13}, e_{14} στα (x_1, x_2), (x_1, x_3), (x_1, x_4). Τα χαρακτηριστικά των κόμβων ενημερώνονται αγγίζοντας τα χαρακτηριστικά από τα *($h_{12}m_{12}$, $h_{13}m_{13}$, $h_{14}m_{14}$)*. Αντίστροφα, τα μηνύματα από τους κόμβους προς τις ακμές ενημερώνουν τα χαρακτηριστικά των ακμών σε e'_{12}, e'_{13}, e'_{14}.



Σχήμα 4.7: Παράδειγμα διπλής αναμετάδοσης μηνυμάτων [75]

4.3.6 Παγκόσμια Συγκέντρωση

Στην αρχιτεκτονική PointNet (Qi κ.α., 2017) [84], η παγκόσμια συγκέντρωση έχει έναν πολύ σημαντικό ρόλο. Αυτή έχει την δυνατότητα να καταγράφει τις παγκόσμιες πληροφορίες του γραφήματος με λογαριθμική ανεξαρτησία των εισόδων και θέτει συμμετρικά τον υπολογισμό του δικτύου, και βελτιώνει με επιτυχία την απόδοση σε προβλήματα ταξινόμησης και τμηματοποίηση σε σύνολα σημείων. Το BMPL καταγράφει αποτελεσματικά τα τοπικά χαρακτηριστικά των γραφημάτων. Μπορεί να θεωρείτε ωφέλιμο να μετασχηματιστούν τα παγκόσμια χαρακτηριστικά του γραφήματος σε χαρακτηριστικά κόμβων για τη λήψη αποφάσεων. Εδώ χρησιμοποιείται η μέθοδος max-pooling (Xu κ.α., 2018) [85] για την απόκτηση παγκόσμιων χαρακτηριστικών.

4.3.7 Εκπαίδευση του Μοντέλου

Στο τμήμα αποκωδικοποίησης, ενώνονται τα χαρακτηριστικά των κόμβων και τα παγκόσμια χαρακτηριστικά και στη συνέχεια δίνονται στο BMPL προκειμένου να ενσωματωθούν περαιτέρω τα παγκόσμια και τα τοπικά χαρακτηριστικά. Στη συνέχεια, σχεδιάζεται ένα πολυεπίπεδο πεπερασμένων στοιχείων για τον υπολογισμό των σκορ $S = \{s_1, s_2, \ldots, s_n\}$ για κάθε κόμβο ως επόμενο κόμβο για επίσκεψη. Τέλος, οι πιθανότητες στο βήμα t υπολογίζονται με χρήση της συνάρτησης softmax.

$$\mathcal{P}(\pi(t) = j) = \frac{exp(s_j)}{\sum_{j \in V} exp(s_j)} \tag{4.7}$$

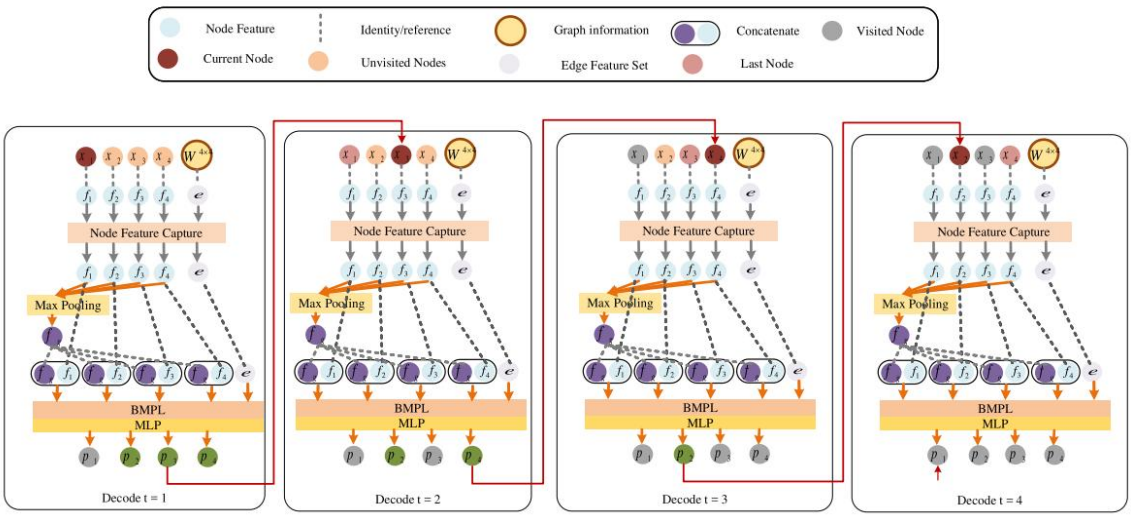
Το δίκτυο χρησιμοποιεί τη συνάρτηση απώλειας cross-entropy σε κάθε βήμα. Δεδομένης της πραγματικής διάταξης $\hat{\pi}$, η συνάρτηση απώλειας στο βήμα t μπορεί να γραφεί ως (4.8). Η διαδικασία της μάθησης με απομίμηση για το πρόβλημα του TSP σε αυθαίρετα συμμετρικά γραφήματα φαίνεται στο Σχήμα 4.8. Ένα αδύναμο σημείο της προσέγγισης που παρουσιάστηκε μέχρι στιγμής είναι ότι το δίκτυο μπορεί να βρίσκεται σε δίλημμα όταν συναντά άλλες βέλτιστες λύσεις για τους ίδιους γραφήματα. Έχοντας δεδομένο ένα κέντρο, η βέλτιστη λύση του προβλήματος TSP σε αυθαίρετα συμμετρικά γραφήματα έχει πάντα μια άλλη ισοδύναμη έκφραση, δηλαδή αν η βέλτιστη λύση είναι $\{A \rightarrow B \rightarrow C \rightarrow D \rightarrow A\}$, τότε η $\{A \rightarrow D \rightarrow C \rightarrow B \rightarrow A\}$ είναι η ισοδύναμη βέλτιστη λύση. Μια σημαντική παρατήρηση η οποία πρέπει να τεθεί είναι ότι η βέλτιστη λύση είναι προκαθορισμένη μόλις δοθεί μια κατεύθυνση. Για παράδειγμα, υποθέστε ότι η τρέχουσα σειρά επίσκεψης είναι $\{A \rightarrow B\}$, η βέλτιστη λύση πρέπει να είναι $\{A \rightarrow B \rightarrow C \rightarrow D \rightarrow A\}$. Ωστόσο, όταν ένα ταξίδι ξεκινά από το κέντρο A, έχει την δυνατότητα να πάει στο B ή το D στο επόμενο βήμα. Για να ξεπεραστεί αυτή η αδυναμία, για το ίδιο γράφημα και δίνοντας το κέντρο, η βέλτιστη λύση που υπολογίζεται από άλλες μεθόδους αντιστρέφεται για να παράξει μια άλλη ισοδύναμη λύση, και στη συνέχεια το δίκτυο εκπαιδεύεται δύο φορές με βάση τις δύο βέλτιστες λύσεις. Με αυτόν τον τρόπο, το δίκτυο είναι σε θέση να διαφοροποιήσει τις βέλτιστες λύσεις και να δίνει περισσότερη σημασία στην κατεύθυνση επίσκεψης, η οποία εκφράζεται από τον τελευταίο κόμβο και τον τρέχοντα κόμβο στα αρχικά χαρακτηριστικά του κόμβου.

$$loss = -ln P\big(\pi(t) = \hat{\pi}(t)\big) \tag{4.8}$$

$$= ln \sum_{j \in V} \exp\big(s_j\big) - s_{\hat{\pi}(t)}$$

4.3.8 Σχεδιασμός Αποκωδικοποίησης

Το BGNN δίνει πιθανότητα σε όλους τους κόμβους να είναι ο επόμενος κόμβος που θα επισκεφθεί βάσει της μερικής λύσης σε κάθε βήμα. Για να κατασκευαστεί μια λύση για το πρόβλημα TSP σε αυθαίρετα συμμετρικά γραφήματα, το BGNN είναι σε θέση να συνδυαστεί με μεθόδους αναζήτησης πληροφορίας, όπως η επιλογή βάσει εγωκεντρικής αναζήτησης, η αναζήτηση με δέσμη (beam search) και η αναζήτηση με βέλτιστη πρώτη επιλογή (best first search). Η ποιότητα των παραγόμενων λύσεων από διάφορα σχήματα αποκωδικοποίησης σχετίζεται άμεσα με τις μάθησης προσεγγιστικών λύσεων του BGNN.



Σχήμα 4.8: Παράδειγμα μίμησης εκμάθησης και κατασκευής περιήγησης [75]

4.3.9 BGNN με εγωκεντρική αναζήτηση

Όταν το BGNN συνδυάζεται με την επιλογή βάσει εγωκεντρικής αναζήτησης (greedy search), είναι δυνατόν να κατασκευαστεί ένα ταξίδι επιλέγοντας επαναληπτικά τον επόμενο κόμβο με την υψηλότερη πιθανότητα επίσκεψης. Ωστόσο, οι κόμβοι δεν είναι πάντα συνδεδεμένοι στο πρόβλημα του TSP σε αυθαίρετους συμμετρικούς γραφήματα, για αυτό υπάρχει η επιλογή να αποκρυφτούν οι κόμβοι που δεν μπορούν να επισκεφθούν από τον τρέχοντα κόμβο κατά την παραγωγή του επόμενου κόμβου προς επίσκεψη. Η κατασκευή του ταξιδιού δεν θα σταματήσει μέχρι να επισκεφθούν όλοι οι κόμβοι ή αν δεν υπάρχουν κόμβοι που να μην έχουν επισκεφτεί που είναι συνδεδεμένοι με τον τρέχοντα κόμβο στη μερική λύση. Το Σχήμα 4.8 δείχνει τη διαδικασία κατασκευής ενός ταξιδιού με την επιλογή βάσει εγωκεντρικής αναζήτησης (greedy decoding).

4.3.10 BGNN με αναζήτηση δέσμης (beam search)

Η αποκωδικοποίηση με εγωκεντρική αναζήτηση (greedy search) βάσει των προβλέψεων του BGNN δεν επιτυγχάνει πάντα να παράγει έγκυρους κύκλους που είναι κοντά στη βέλτιστη λύση, ειδικά όταν το γράφημα γίνεται αραιό. Σε αυτές τις περιπτώσεις, η χρήση της απλής αναζήτησης με δέσμη (beam search) αυξάνει σημαντικά την πιθανότητα λύσεων υψηλής ποιότητας (κοντά στη βέλτιστη). Σε αντίθεση με την εγωκεντρική αναζήτηση, η αναζήτηση με δέσμη (beam search) επιλέγει για επόμενους κόμβους τους υποψήφιους με την υψηλότερη πιθανότητα σύμφωνα με τις προβλέψεις στο τρέχον βήμα.

4.3.11 BGNN με αναζήτηση καλύτερης πρώτης επιλογής

Η αποκωδικοποίηση με εγωκεντρική αναζήτηση (greedy search) και αναζήτηση με δέσμη (beam search) δεν είναι σε θέση να εγγυηθούν την παραγωγή έγκυρου κύκλου με επιτυχία, γι' αυτό συνδυάζεται το BGNN με την αναζήτηση με την καλύτερη πρώτη επιλογή (best-first search) επίσης.

Τα υποδείγματα που καθοδηγούν την αναζήτηση με την καλύτερη πρώτη επιλογή είναι οι προβλέψεις από το BGNN σύμφωνα με τις μερικές λύσεις.

Το BGNN μπορεί να συνδυαστεί και με πολλούς άλλους αλγορίθμους αναζήτησης πληροφοριών, όπως η αναζήτηση κατά βάθος (depth-first search), η αναζήτηση κατά πλάτος (breadth-first search), κ.λ.π.

Πειράματα

Για να γίνει η εξέταση της απόδοσης του προτεινόμενου διπλής κατεύθυνσης γραφικού νευρωνικού δικτύου σε αυθαίρετα συμμετρικά προβλήματα ταξιδιού πωλητή, οι (Yujiao Hu, Zhen Zhang, Yuan Yao κ.α. 2021) [75] διενέργησαν πειράματα σε ευκλείδεια γραφήματα και αραιά συνδεδεμένα συμμετρικά γραφήματα. Το ευκλείδειο TSP θεωρείται ως ένα συγκεκριμένο παράδειγμα του TSP σε αυθαίρετα συμμετρικά γραφήματα. Τα αραιά συνδεδεμένα συμμετρικά γραφήματα αποτελούν έναν γενικό είδος εργασιών για το TSP, όπου κάθε δύο κόμβοι έχουν την δυνατότητα να συνδεθούν ή να αποσυνδεθούν. Ακόμη και τα πλήρες γραφήματα μπορούν να θεωρηθούν ως συγκεκριμένα παραδείγματα αραιά συνδεδεμένων γραφημάτων όταν όλοι οι κόμβοι είναι συνδεδεμένοι. Επομένως, ο TSP σε αραιά συνδεδεμένα συμμετρικά γραφήματα μπορεί να θεωρηθεί ως ένας κλασικός συγκεκριμένος τύπος του TSP, αλλά έχει ελάχιστα μελετηθεί σε προηγούμενες εργασίες.

Ευκλείδειο TSP

Το δικτυακό δίκτυο γραφημάτων με διπλή κατεύθυνση (BGNN) συγκρίνεται με δύο πρόσφατα νευρωνικά δίκτυα, το Graph Attention Network εκπαιδευμένο με πολιτική βελτιστοποίηση (GAT+PG) (Kool κ.α., 2018) [86] και το Structure2Vec εκπαιδευμένο με Deep Q-Learning (S2V+DQN) (Khalil κ.α., 2017)[87], τα οποία αποτελούν δύο από τις τελευταίες καταστάσεις της τέχνης στον τομέα. Πολλές μέθοδοι που βασίζονται σε προσεγγιστική αναζήτηση, όπως οι ORTools, η τυχαία/μακρινή εισαγωγή και οι κοντινότεροι γείτονες, χρησιμοποιούνται επίσης ως αναφορές. Παράλληλα, χρησιμοποιείται το Gurobi για τον υπολογισμό της βέλτιστης διαδρομής. Στην έρευνα έλαβαν υπόψιν μοντέλα με διάφορα μεγέθη γραφημάτων (τρία διαστήματα μεγέθους γραφήματα *[20, 25], [30, 50]* και *[50, 100]*).

Δεδομένα

Οι συντεταγμένες των κόμβων παράγονται ομοιόμορφα τυχαία στο διάστημα $(0, 1)^2$ για να διατηρηθεί η συνέπεια με προηγούμενες εργασίες. Δημιουργούνται 7000 γραφήματα ως σύνολο εκπαίδευσης και 1000 γραφήματα ως σύνολο δοκιμής. Η βέλτιστη διαδρομή αποτελείται χρησιμοποιώντας το Gurobi. Στις πειραματικές δοκιμές, το GAT+PG και το S2V+DQN χρησιμοποιούν τις συντεταγμένες των κόμβων ως αρχικά χαρακτηριστικά των κόμβων. Οι μέθοδοι αυτής της έρευνας απαιτούν τον υπολογισμό του πίνακα ευκλείδειας απόστασης.

Μέτρηση επίδοσης

Τα μεγέθη (Gap & Time) χρησιμοποιούνται για το πρόβλημα του TSP στους ευκλείδειους γραφήματα, όπου Gap = Ltour/Lbest, όπου Ltour είναι το μήκος της διαδρομής που παράγεται από διάφορους αλγορίθμους και Lbest είναι το βέλτιστο μήκος της διαδρομής. Το Time είναι ο μέσος χρόνος εκτέλεσης των αλγορίθμων για μία προσομοίωση.

Αποτελέσματα

Στη φάση των δοκιμών, χρησιμοποιήθηκε η στρατηγική του "greedy decoding" για όλες τις μεθόδους βαθιάς μάθησης, δηλαδή σε κάθε βήμα απλά επιλέγεται η πιο πιθανή πρόβλεψη από το νευρωνικό δίκτυο. Ο Πίνακας 4.4 δείχνει ότι η μέθοδός αυτή υπερτερεί των μεθόδων χωρίς μάθηση και είναι επίσης συγκρίσιμη με τις μεθόδους που βασίζονται στη μάθηση.

	$n \in [20,25]$		$n \in [30,50]$		$n \in [50,100]$	
Method	Time	Gap	Time	Gap	Time	Gap
Gurobi	0.073	1.000	0.287	1.000	3.181	1.000
OR tools	0.012	1.011	0.040	1.034	0.642	1.034
Random Insertion	0.001	1.048	0.002	1.068	0.004	1.087
Farthest Insertion	0.002	1.027	0.003	1.047	0.017	1.079
Nearest neighbor	0.001	1.182	0.003	1.218	0.010	1.356
S2V+DQN	0.041	1.046	0.038	1.045	0.081	1.067
GAT+PG	0.001	1.022	0.001	1.037	0.003	1.049
BGNN	0.005	1.015	0.017	1.021	0.058	1.034

Πίνακας 4.4: Σύγκριση σχετικού κενού και χρόνου εκτέλεσης στο Euclidean TSP

4.3.12 TSP σε αραιά γραφήματα

Για τον συμμετρικό αραιό γράφημα με n κόμβους και m ακμές, αρχίζει με n απομονωμένους κόμβους. Κάθε φορά, τυχαία γίνεται η επιλογή από έναν κύκλο μήκους n και για κάθε ακμή στον κύκλο, αν δεν έχει ήδη ανατεθεί ένα μήκος, τότε του αναθέτουν τυχαία ένα μήκος από την ομοιόμορφη κατανομή $U(0, 1)$. Αυτή η διαδικασία επαναλαμβάνεται μέχρι να ανατεθούν μήκη σε όλες τις m ακμές. Με αυτόν τον τρόπο, μπορεί να δημιουργηθεί ένα συμμετρικό αραιό γράφημα με έναν ή περισσότερους κύκλους Hamiltonian (δηλαδή εφικτές λύσεις του TSP υπάρχουν).

Σε αυτό το πείραμα, εφαρμόζεται η Greedy (Επικερδή), η Beam Search (Αναζήτηση Δέσμης) και η Best First Search (Καλύτερη Πρώτη Αναζήτηση) ως μεθόδους αποκωδικοποίησης, χρησιμοποιώντας τις προβλέψεις (πιθανότητες που υπολογίζονται από την εξίσωση (4.7)) ως προσεγγιστικά. Τα

προσεγγιστικά αυτά δεν θεωρούνται αποδεκτά και χρησιμοποιούνται για να καθοδηγήσουν την αναζήτηση προς την εύρεση μιας λύσης αντί να επιτύχουν την βέλτιστη λύση.

Διότι οι αλγόριθμοι ενδέχεται να μην βρίσκουν λύσεις για το πρόβλημα του TSP σε αραιούς γραφήματα, εισάγετε η πιθανότητα (Prob) για να αντιπροσωπεύει το ποσοστό επιτυχίας στην εύρεση μιας έγκυρης λύσης. Οι τιμές Gap και Time χρησιμοποιούνται μόνο για τις περιπτώσεις πετυχημένων λύσεων. Για να φιλτραριστούν οι λύσεις χαμηλής ποιότητας, εισάγεται ένα αποδεκτό όριο άνω φράγματος για το Gap, το οποίο ορίζεται ως 1.2, και το ποσοστό υψηλής ποιότητας λύσεων σε σχέση με το σύνολο δοκιμών εκφράζεται μέσω του ClipPro. Το sp δείχνει το βαθμό αραιότητας ενός γραφήματος και υπολογίζεται με τον τύπο $sp = esp/ecomp$, όπου esp και $ecomp$ είναι η μεταβλητές που δηλώνουν τον αριθμό των ακμών στο αραιό γράφημα και στον πλήρη γράφημα αντίστοιχα.

Σε αυτό το κομμάτι επιδεικνύονται πειράματα που αποδεικνύουν ότι το ενισχυτικό αντιμετωπίζει προβλήματα αποτελεσματικότητας στην εξερεύνηση. Τροποποιούνται οι κώδικες του GAT+PG (Kool κ.ά., 2018) [86] και του S2V+DQN (Khalil κ.ά., 2017) [87] για να είναι δυνατή η εξαγωγή χαρακτηριστικών από αραιά γραφήματα και η παραγωγή της λύσης του TSP. Αυτά τα δύο έργα καταφέρνουν να επιτύχουν εξαιρετικές επιδόσεις στον Ευκλείδειο TSP με την ενσωμάτωση των χαρακτηριστικών των κόμβων και την ανάπτυξη της λύσης σε μέρος ακολουθίας αποκωδικοποίησης. Τα αρχικά χαρακτηριστικά των κόμβων είναι οι συντεταγμένες με μέγεθος ($n \times 2$). Εδώ τροποποιούνται τα μέρη κωδικοποίησης και αποκωδικοποίησης. Στο μέρος της κωδικοποίησης, για το S2V+DQN, τίθενται οι συντεταγμένες όλων των κόμβων (1, 1) και παίρνουν βάρη στο μέρος της κωδικοποίησης (επειδή το άρθρο περιέχει εξίσωση για ενσωμάτωση χαρακτηριστικών κόμβου που χρησιμοποιεί συντεταγμένες και βάρη). Για το GAT+PG, όρισαν την απόσταση από έναν κόμβο προς οποιονδήποτε άλλον κόμβο ως αρχικά χαρακτηριστικά του κόμβου, δηλαδή το μέγεθος των αρχικών χαρακτηριστικών ενός κόμβου είναι ($1 \times n$), και το μέγεθος των αρχικών χαρακτηριστικών για n κόμβους είναι ($n \times n$). Στο μέρος της αποκωδικοποίησης, η πιθανότητα επίσκεψης ενός μη συνδεδεμένου και ήδη επισκεπτόμενο κόμβο αποκλείεται κατά το μηδέν σε κάθε βήμα. Το κόστος κάθε ολοκληρωμένου δείγματος είναι ίσο με το μήκος της περιήγησης, και το κόστος αρνητικού δείγματος είναι 1000. Κατά την εκπαίδευση, τα αρνητικά δείγματα δημιουργούνται πολύ πιο συχνά από τα θετικά δείγματα.

Ως αποτέλεσμα αυτών των τροποποιήσεων, και οι δύο μέθοδοι που είναι βασισμένες σε ενισχυτική μάθηση είχαν κακή απόδοση σε αραιά γραφήματα σε σύγκριση με τα αρχικά έργα. Οι συγγραφείς εξέτασαν επίσης την απόδοση αλγορίθμων προσεγγιστικής αναζήτησης, όπως ο ORTools, ο Farthest Insertion και ο Nearest Neighbor. Παρόλο που αυτές οι προσεγγιστικές μέθοδοι δημιούργησαν έγκυρους κύκλους για το TSP σε αυθαίρετους συμμετρικούς γραφήματα, οι περιηγήσεις ήταν πολύ μακριά από την βέλτιστη λύση, δείχνοντας ένα υψηλό σχετικό κενό.

Για να επιδείξουν περαιτέρω την αποτελεσματικότητα της προτεινόμενης προσέγγισής τους, οι συγγραφείς συγκρίνουν τον BGNN (Boosted Graph Neural Network) με έναν κανονικό GNN (GNN(Reg)) που χρησιμοποιεί το πλαίσιο CMPNN. Το BGNN χρησιμοποιεί πληροφορίες βοήθειας και τον πίνακα γραφήματος ως αρχικά χαρακτηριστικά κόμβου, ενώ το GNN(Reg) εκπαιδεύεται στον ίδιο σύνολο δεδομένων χρησιμοποιώντας τον μηχανισμό μάθησης μέσω μιμητισμού. Τα αποτελέσματα έδειξαν ότι το GNN(Reg) είχε υψηλότερες πιθανότητες αλλά χαμηλότερη ποιότητα

λύσης σε σύγκριση με το BGNN. Η κλιμακωτή προσέγγιση του BGNN επέτρεψε την εύρεση καλύτερων λύσεων για το TSP.

Συνολικά, η προτεινόμενη μέθοδος ενισχυμένης μάθησης BGNN έδειξε θετικά και υποσχόμενα αποτελέσματα στην επίλυση του προβλήματος του TSP σε αραιούς γραφήματα. Οι συγγραφείς προτείνουν περαιτέρω έρευνα για τη βελτίωση της απόδοσης του BGNN και την εξέταση άλλων παρόμοιων προβλημάτων.

4.3.13 Δοκιμή γενίκευσης

Αναμένεται επίσης το γεγονός ότι το BGNN μπορεί να γενικευθεί σε μεγαλύτερα γραφήματα από το μέγεθος του συνόλου δεδομένων εκπαίδευσης. Για τον λόγο αυτό, σχεδιάστηκε ένα πειραματικό τεστ γενίκευσης. Το μοντέλο BGNN εκπαιδεύεται στο TSP*[70, 75],* αλλά δοκιμάζεται σε μεγαλύτερα γραφήματα έως και 200 πόλεις, περίπου τρεις φορές το μέγεθος του συνόλου δεδομένων εκπαίδευσης. Συνδυάζεται το μοντέλο BGNN με την αναζήτηση Best First Search και τίθεται σύγκριση με το Gurobi και το ORTools όσον αφορά το Gap και τον χρόνο εκτέλεσης (Time). Το Gurobi παράγει ακριβείς λύσεις ως σημείο αναφοράς, ενώ το ORTools θεωρείται ένας αντιπροσωπευτικός εκπρόσωπος των προσεγγιστικών αλγορίθμων, καθώς το ORTools πλεονεκτεί των άλλων δύο προσεγγιστικών αλγορίθμων (Farthest Insertion, Nearest Neighbor) που παρουσιάζονται στο [75]. Τα αποτελέσματα δίνονται στον πίνακα 4.5. Όλες οι μέθοδοι παράγουν επιτυχώς λύσεις στο σύνολο δοκιμής, για αυτό αγνοούμε το Prob. Ο πίνακας δείχνει ότι ο BGNN μπορεί να ισορροπήσει τον χρόνο εκτέλεσης και την απόδοση, καθώς απαιτεί λιγότερο χρόνο για να βρει μια καλή λύση από το Gurobi και παρέχει καλύτερες λύσεις από το ORTools. Ένα πλεονέκτημα είναι επίσης ότι ο BGNN με την αναζήτηση Best First Search είναι σε θέση να παράγει πιο σύντομες διαδρομές διευρύνοντας περισσότερους κόμβους, ενώ είναι δύσκολο για τα αποτελέσματα του ORTools να αποδράσουν από το τοπικό βέλτιστο, καθώς το ORTools χρησιμοποιεί ήδη την τοπική αναζήτηση για να βελτιώνει συνεχώς τη λύση.

		[100-120]		[120-150]		[150-200]	
sp	Method	Gap	Time	Gap	Time	Gap	Time
0.5	Gurobi	1.000	4.59	1.000	7.79	1.000	24.98
	ORTools	1428	0.45	1.465	0.58	1.596	1.05
	BGNN	1.203	2.02	1.255	3.27	1.311	6.58
0.4	Gurobi	1.000	4.26	1.000	7.39	1.000	22.53
	ORTools	1.422	0.45	1.463	0.58	1.438	1.05
	BGNN	1.163	2.21	1.175	3.71	1.225	6.87
0.3	Gurobi	1000	3.78	1.000	6.78	1.000	18.89
	ORTools	1419	0.45	1.477	0.59	1.529	1.03
	BGNN	1.106	2.62	1.135	4.17	1.175	7.89

Πίνακας 4.5: Δοκιμή γενίκευσης νευρωνικού δικτύου αμφίδρομης γραφικής παράστασης (BGNN)

4.4 GRAPH NEURAL NETWORK GUIDED LOCAL SEARCH FOR THE TRAVELING SALESPERSON PROBLEM

4.4.1 Εισαγωγή

Σε αυτή την ενότητα θα αναφερθούμε σε μια προσέγγιση βασισμένη σε δεδομένα για την περίπου επίλυση του Ευκλείδειου προβλήματος του περιπλανώμενου πωλητή (Euclidean TSP) με τη χρήση Γραφικών Νευρωνικών Δικτύων (Graph Neural Networks - GNNs) και Καθοδηγούμενης Τοπικής Αναζήτησης (Guided Local Search - GLS), την οποία παρατηρείται ότι συγκλίνει προς βέλτιστες λύσεις σε έναν ικανοποιητικό χρόνο. Τα συστατικά στοιχεία που παρέχονται είναι τα εξής:

- Ο αλγόριθμος GLS καθοδηγείται από την παγκόσμια λύπη (global regret) κάθε ακμής στο γράφημα του προβλήματος. Αυτό επιτρέπει στον αλγόριθμο να ξεχωρίζει τις ακμές που είναι πολύ δαπανηρές έτσι ώστε να μην συμπεριληφθούν στη λύση από αυτές που δεν είναι τόσο δαπανηρές, είτε ανήκουν στην βέλτιστη λύση είτε όχι. Έτσι, η χρήση της global regret επιτρέπει να βρεθούν γρήγορα λύσεις υψηλής ποιότητας, αντί για βέλτιστες. Οι συγγραφείς [88] είναι οι πρώτοι που χρησιμοποιούν ένα μέτρο παγκόσμιας λύπης, το οποίο θέτετε ως το κόστος επιβολής μιας απόφασης σε σχέση με το κόστος μιας βέλτιστης λύσης.
- Καθιστάτε αυτό υπολογιστικά εφικτό η προσέγγιση της λύπης με ένα μοντέλο που έχει μάθει. Το μοντέλο που βασίζεται σε GNN λειτουργεί στο γράφημα γραμμής του προβλήματος, επιτρέποντάς να δημιουργηθεί ένα δίκτυο χωρίς χαρακτηριστικά κόμβου, επικεντρώνοντας μόνο στο βάρος των ακμών.
- Δείχνονται επίσης πειραματικά αποτελέσματα για την συγκεκριμένη προσέγγισή και για αρκετές προσεγγίσεις που βασίζονται στη μάθηση, οι οποίες συμμορφώνονται με πρόσφατες κατευθυντήριες γραμμές για την υπολογιστική δοκιμή προσεγγίσεων που βασίζονται στη μάθηση για το TSP. Τα πειράματά στην ενότητα **4.4** τονίζουν την αντιστάθμιση μεταξύ ποιότητας λύσης και χρόνου.
- Μειώνεται ο μέσος βαθμός αποδοτικότητας στο σύνολο προβλημάτων με 50 κόμβους από 0,268% σε 0,009%, μια βελτίωση 30 φορές, και στο σύνολο προβλημάτων με 100 κόμβους από 1,534% σε 0,705%, μια βελτίωση 2 φορές. Στο σημείο που γενικεύεται το προβλήματα με 20 κόμβους στο σύνολο προβλημάτων με 100 κόμβους, μειώνεται ο μέσος βαθμός αποδοτικότητας από 18,845% σε 2,622%, μια βελτίωση 7 φορές.

Η κοινότητα της Έρευνας Λειτουργιών (Operations Research - OR) είναι υπεύθυνη για την πλειονότητα των ερευνών πάνω στους αλγορίθμους για την επίλυση προβλημάτων συνδυαστικής βελτιστοποίησης. Το Concorde (Applegate κ.ά., 2006) [39] θεωρείται ευρέως ως ο καλύτερος τρόπος επίλυσης για το πρόβλημα του περιοδεύοντος πωλητή (TSP). Ως καλύτερος τρόπος επίλυσης, μπορεί να εγγυηθεί το επίπεδο αποδοτικότητας των λύσεων που βρίσκει. Χρησιμοποιεί αλγορίθμους κοπής (Dantzig κ.ά., 1954· Padberg & Rinaldi, 1990· Applegate κ.ά., 2003) [36][89][90] και μεθόδους κλαδέματος και κοπής για να περιορίσει επαναληπτικά τον χώρο αναζήτησης που δεν θα περιέχει βέλτιστη λύση. Το LKH-3 (Helsgaun, 2017) [91], καθώς και ο προκάτοχός του, LKH-2 (Helsgaun, 2009) [92], είναι προσεγγιστικοί λύτες για το πρόβλημα του περιοδεύοντος πωλητή και το πρόβλημα δρομολόγησης με χωρητικότητα οχήματος (CVRP) βασισμένοι στην προσέγγιση κ-βέλτιστη μέθοδο (Lin & Kernighan, 1973) [93]. Αν και αυτοί οι λύτες δεν παρέχουν εγγύηση για το αποτέλεσμα τους, πειραματικά έχει αποδειχθεί ότι είναι απίστευτα αποτελεσματικοί. Τα Concorde, LKH-2 και LKH-3

είναι εξειδικευμένοι και αποδοτικοί λύτες που μπορούν να επιλύσουν απαιτητικά προβλήματα TSP σε δευτερόλεπτα.

Ωστόσο, τα προβλήματα δρομολόγησης πραγματικού χρόνου συνήθως επιβάλλουν περιορισμούς πάνω από τους βασικούς ορισμούς του TSP ή του CVRP προβλήματος. Όντας εξειδικευμένοι λύτες είναι δύσκολο να προσαρμοστούν σε αυτούς τους νέους περιορισμούς. Έτσι, υπάρχει ζήτηση για γενικούς αλγοριθμικούς πλαισίου που μπορούν να παράγουν υψηλής ποιότητας λύσεις με ελάχιστο χρόνο υπολογισμού. Για να αντιμετωπιστεί αυτό το ζήτημα, ο Arnold & Sorensen (2019a) [94] δημιούργησαν το KGLS, έναν αλγόριθμο GLS για το CVRP που καθοδηγείται από τρία κατασκευασμένα χαρακτηριστικά. Αποδεικνύουν ότι ο αλγόριθμός τους μπορεί να βρίσκει λύσεις σχεδόν τόσο καλές όσο αυτές που βρίσκονται από τις πιο σύγχρονες μετά-ευριστικές μεθόδους σε ένα κλάσμα του χρόνου. Σε αυτό το σκοπό εστιάζει και αυτός ο αλγόριθμος.

Η δημιουργία μιας επιτυχημένης προσέγγισης μηχανικής μάθησης για το πρόβλημα του περιοδεύοντος πωλητή απαιτεί τη συνδυασμένη χρήση τεχνικών μηχανικής μάθησης και συνδυαστικής βελτιστοποίησης. Οι προσεγγίσεις που βασίζονται σε μηχανική μάθηση προσπαθούν να εκπαιδεύσουν μοντέλα που μπορούν να προβλέψουν καλές λύσεις για το πρόβλημα του περιοδεύοντος πωλητή, ενώ οι τεχνικές συνδυαστικής βελτιστοποίησης χρησιμοποιούνται για τη βελτιστοποίηση και τη βελτίωση αυτών των λύσεων. Τα GNNs έχουν αποδειχθεί αρκετά χρήσιμα για προβλήματα που έχουν γεωμετρική δομή ή δομή γραφημάτων, όπως ο TSP.

Συνοψίζοντας, η επίλυση του προβλήματος του περιοδεύοντος πωλητή με χρήση μηχανικής μάθησης απαιτεί την ανάπτυξη συνδυασμένων προσεγγίσεων που εκπαιδεύουν μοντέλα GNN για την πρόβλεψη καλών λύσεων και χρησιμοποιούν τεχνικές συνδυαστικής βελτιστοποίησης για τη βελτιστοποίηση και τη βελτίωση αυτών των λύσεων.

4.4.2 Εισαγωγή στον Αλγόριθμο

Πρόβλημα του περιοδεύοντος πωλητή. Εστιάζουν στον Ευκλείδειο TSP, αν και η προσέγγισή αυτή μπορεί να εφαρμοστεί και σε άλλα προβλήματα δρομολόγησης, όπως το CVRP. Ένα πρόβλημα με n πόλεις, συνήθως συμβολίζεται ως TSPn, αναπαρίσταται ως ένας πλήρης, μη κατευθυνόμενο, με βάρη γράφημα G = (V, E) με n κόμβους. Οι ακμές E δείχνουν συνδέσεις μεταξύ των πόλεων και έχουν βάρη που ορίζονται από την Ευκλείδεια απόσταση μεταξύ των γειτονικών κόμβων. Μια λύση, ή περιοδεία, είναι ένας Χαμιλτονιανός κύκλος: ένα κλειστό μονοπάτι μέσα στο γράφημα που επισκέπτεται κάθε κόμβο ακριβώς μία φορά. Μια βέλτιστη λύση είναι ένας κύκλος ελάχιστου βάρους.

Regret. Το Regret μετρά το μελλοντικό κόστος μιας ενέργειας που παράγει επίσης άμεση ανταμοιβή και συνήθως χρησιμοποιείται για να καταστήσει λιγότερο μυωπικούς τους αλγόριθμους συνδυασμένης βελτιστοποίησης. Γενικά, το regret υπολογίζεται για ένα σταθερό ορίζοντα. Για παράδειγμα, οι Potvin & Rousseau (1993) [95] αξιολογούν το κόστος εισαγωγής ενός κόμβου στην καλύτερη επιλογή έναντι της δεύτερης καλύτερης επιλογής κατά την κατασκευή μιας λύσης CVRP. Οι Hassin & Keinan (2008) [96] λύνουν το TSP χρησιμοποιώντας Regret, επιτρέποντας σε ένα γειτονικό κατασκευαστικό υπό-αλγόριθμο να αφαιρέσει τον πιο πρόσφατα εισαχθέντα κόμβο. Δεν

είναι υπολογιστικά εφικτό να υπολογίσουμε το regret για έναν παγκόσμιο ορίζοντα, για παράδειγμα, για όλες τις πιθανές ακολουθίες εισαγωγής. Ωστόσο, αν ήταν δυνατό, ένας άπληστος (greedy) αλγόριθμος θα μπορούσε να υπολογίσει μια βέλτιστη λύση επιλέγοντας την απόφαση με τη μικρότερη regret σε κάθε βήμα.

Τοπική Αναζήτηση. Η Τοπική Αναζήτηση (ΤΑ) είναι μια γενική προσέγγιση βελτιστοποίηση. Ξεκινώντας από μια αρχική λύση, η τοπική αναζήτηση μετακινείται επαναληπτικά σε γειτονικές λύσεις που έχουν μικρότερο κόστος από την τρέχουσα λύση σύμφωνα με τη συνάρτηση στόχου g(s). Γειτονικές λύσεις θεωρούνται οι λύσεις που είναι προσβάσιμες από μια δεδομένη λύση εφαρμόζοντας μια συγκεκριμένη συνάρτηση ή τελεστή. Το σύνολο όλων των λύσεων που είναι προσβάσιμες από μια άλλη λύση εφαρμόζοντας έναν τελεστή καθορίζει τη γειτονιά εκείνου του τελεστή. Ο αλγόριθμος τερματίζει όταν όλες οι γειτονικές λύσεις είναι χειρότερες από την τρέχουσα λύση, πράγμα που σημαίνει ότι η αναζήτηση έχει φτάσει σε ένα τοπικό ακέραιο.

Καθοδηγούμενη Τοπική Αναζήτηση. Η Καθοδηγούμενη Τοπική Αναζήτηση (ΚΤΑ) είναι μια μετά-ευρετική που βασίζεται στην ΤΑ και της επιτρέπει να ξεφύγει από τα τοπικά ακόρεστα (Voudouris & Tsang, 1996) [97]. Για να εφαρμοστεί η ΚΤΑ, ο σχεδιαστής πρέπει να καθορίσει ορισμένες πτυχές μιας λύσης. Όταν παγιδευτεί σε ένα τοπικό ακέραιο, ο αλγόριθμος επιβάλλει κυρώσεις σε συγκεκριμένες πτυχές της τρέχουσας λύσης που θεωρούνται μη επιθυμητές. Η υποκείμενη διαδικασία ΤΑ αναζητά χρησιμοποιώντας μια συνάρτηση στόχου που επεκτείνεται με αυτές τις κυρώσεις, έτσι ενθαρρύνεται να αφαιρέσει τις πτυχές που επιβάρυναν σημαντικά από τη λύση. Η επεκτεινόμενη αντικειμενική συνάρτηση h(s) είναι

$$h(s) = g(s) + \lambda \sum_{i=0}^{M} p_i I_i(s) \qquad (4.9)$$

όπου s είναι μια λύση, g(s) είναι η αντικειμενική συνάρτηση, λ είναι ένας παράγοντας κλίμακας, i καθορίζει τις πτυχές, M είναι ο αριθμός των πτυχών, pi είναι ο τρέχων αριθμός των κυρώσεων που έχουν ανατεθεί στην πτυχή i, και Ii είναι μια ένδειξη εάν η λύση s εκδηλώνει την πτυχή i, δηλαδή

$$I_i(s) = \begin{cases} 1 & \text{if s exhibits aspect } i, \\ 0 & otherwise \end{cases} \qquad (4.10)$$

Για το TSP, οι πτυχές της λύσης συνήθως καθορίζονται ως οι ακμές στο γράφημα του προβλήματος. Επομένως, η Ii(s) υποδεικνύει εάν μια ακμή βρίσκεται στη λύση s. Όταν φτάνει σε ένα τοπικό ακέραιο, ποιες πτυχές θα υποβληθούν σε κυρώσεις καθορίζονται από μια ωφέλεια (utility) συνάρτηση. Η ωφέλεια της κυρώσεως της πτυχής i, utili, καθορίζεται ως

$$util_i(s_*) = I_i(s_*) \frac{c_i}{1+p_i}, \qquad (4.11)$$

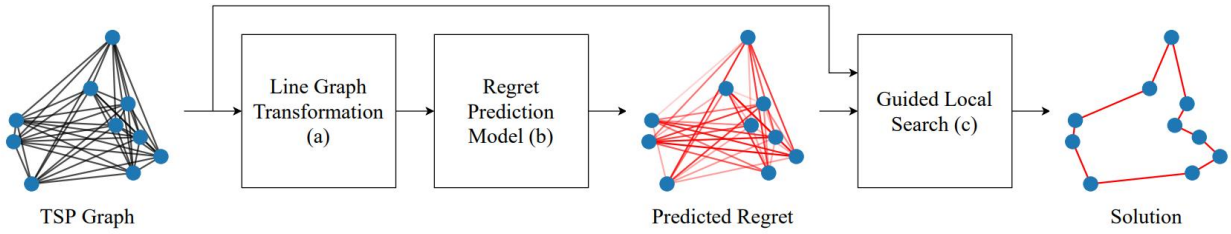
όπου s∗ είναι η λύση σε ένα τοπικό ακέραιο, Ii(s∗) υποδεικνύει εάν η λύση εκδηλώνει την πτυχή i, και ci είναι το κόστος της πτυχής i. Το κόστος μιας πτυχής μετράει πόσο ανεπιθύμητη είναι. Όσο μεγαλύτερο είναι το κόστος, τόσο υψηλότερη είναι η ωφέλεια της κύρωσης της συγκεκριμένης πτυχής. Στο πλαίσιο του TSP, το κόστος μπορεί να είναι το βάρος της ακμής (Voudouris & Tsang, 1999) [98] ή μια συνδυασμένη τιμή διάφορων χαρακτηριστικών (Arnold & Sorensen, ¨ 2019α) [94]. Αντιστρόφως, όσες περισσότερες κυρώσεις δίνονται σε μια πτυχή, τόσο χαμηλότερη είναι η ωφέλεια της να υποστεί ξανά κύρωση. Οι πτυχές με τη μέγιστη ωφέλεια υποβάλλονται πάντα σε κύρωση, γεγονός που σημαίνει αύξηση του pi κατά ένα. Αυτός ο μηχανισμός κύρωσης κατανέμει την

προσπάθεια αναζήτησης στον χώρο αναζήτησης, ευνοώντας περιοχές όπου υπάρχει υπόσχεση (Voudouris & Tsang, 1996)[97].

Σε αυτόν τον αλγόριθμο χρησιμοποιούν μια παραλλαγή του κλασικού αλγορίθμου GLS (βλ. Voudouris & Tsang, 1999) [98] που εφαρμόζει εναλλασσόμενες φάσεις βελτιστοποίησης και αναστάτωσης (βλ. Arnold & Sorensen, 2019α) [94]. Κατά τη περίοδο μιας φάσης βελτιστοποίησης, η διαδικασία τοπικής αναζήτησης καθοδηγείται από την αρχική αντικειμενική συνάρτηση g. Κατά τη περίοδο μιας φάσης αναστάτωσης, καθοδηγείται από την ενισχυμένη αντικειμενική συνάρτηση h. Μετά την κύρωση μιας ακμής, η τοπική αναζήτηση εφαρμόζεται μόνο στην αυτή την ακμή. Δηλαδή, λαμβάνονται υπόψη μόνο λειτουργίες που θα αφαιρέσουν την κυρωμένη ακμή. Αυτό το γεγονός έχει μεγάλη διαφορά από την τοπική αναζήτηση κατά τη φάση βελτιστοποίησης, η οποία λαμβάνει υπόψιν όλες τις λύσεις στη γειτονιά του συγκεκριμένου τελεστή. Η φάση αναστάτωσης συνεχίζεται μέχρι να εφαρμοστούν K τελεστές (τελεστές που βελτιώνουν τη λύση σύμφωνα με την ενισχυμένη αντικειμενική συνάρτηση h) στην τρέχουσα λύση. Αυτοί οι τελεστές αναστατώνουν τη λύση αρκετά ώστε η τοπική αναζήτηση να ξεφύγει από ένα τοπικό ελάχιστο. Οι εναλλασσόμενες φάσεις συνεχίζονται μέχρι να εκπληρωθεί η συνθήκη παύσης.

4.4.3 Μέθοδος

Η υβριδική μέθοδος, που φαίνεται στο Σχήμα 4.8, συνδυάζει ένα μοντέλο μηχανικής μάθησης και μια αναγωγική μέθοδο. Το μοντέλο είναι βασισμένο σε GNN που μαθαίνει μια προσέγγιση της γενικής απώλειας (regret) της συμπερίληψης κάθε ακμής του γραφήματος του προβλήματος στη λύση. Η αναγωγική μέθοδος GLS χρησιμοποιεί αυτήν την εκμάθηση της απώλειας σε συνδυασμό με το αρχικό γράφημα του προβλήματος για να βρει γρήγορα υψηλής ποιότητας λύσεις. Η εκμάθηση της απώλειας επιτρέπει στον αλγόριθμο να ξεχωρίζει μεταξύ ακμών που είναι δαπανηρές για να συμπεριληφθούν στη λύση και αυτών που δεν είναι τόσο δαπανηρές, βελτιώνοντας έτσι την ικανότητά του να κατευθύνει την αρχική διαδικασία τοπικής αναζήτησης έξω από τα τοπικά ελάχιστα και προς προσανατολισμένες περιοχές του χώρου λύσεων.



Σχήμα 4.9: Μοντέλο Μηχανικής Μάθησης και μια Αναγωγική Μέθοδος [88]

4.4.4 Γενική Απώλεια

Ορίζεται η γενική απώλεια (global regret) ως το κόστος που συνεπάγεται η απαίτηση μιας συγκεκριμένης απόφασης να είναι μέρος της λύσης σε σχέση με το κόστος μιας παγκόσμιας βέλτιστης λύσης. Σε αντίθεση με τις προηγούμενες προσεγγίσεις που χρησιμοποιούν την έννοια της απώλειας, υπολογίζοντας το κόστος μιας απόφασης σε σχέση με έναν σταθερό αριθμό εναλλακτικών (για παράδειγμα, την επόμενη καλύτερη ή τις δύο επόμενες καλύτερες επιλογές), η απώλεια μετριέται σε

σχέση με μια παγκόσμια βέλτιστη λύση. Αποφάσεις που είναι μέρος μιας βέλτιστης λύσης έχουν μηδενική απώλεια, ενώ όλες οι άλλες αποφάσεις έχουν θετική απώλεια.

$$r_i = \frac{g(s_i^*)}{g(s^*)} - 1, \tag{4.12}$$

Με μαθηματικούς όρους, όπου r_i είναι η απώλεια της απόφασης i, g είναι η συνάρτηση στόχου, s_i^* είναι μια βέλτιστη λύση με την i να είναι σταθερή, και s^* είναι μια παγκόσμια βέλτιστη λύση. Με την τελειότητα της πληροφορίας, ένας αλγόριθμος αλιεύματος θα μπορούσε να κατασκευάσει μια βέλτιστη λύση επιλέγοντας με ακολουθία τις αποφάσεις με τη χαμηλότερη απώλεια.

Στο πρόβλημα του TSP, οι αποφάσεις αντιστοιχούν στο ποιες ακμές συμπεριλαμβάνονται στη λύση, επομένως η απώλεια ορίζεται ως το κόστος που συνεπάγεται η απαίτηση μιας συγκεκριμένης ακμής να είναι μέρος της λύσης. Στον αλγόριθμο υποστηρίζεται ότι η χρήση της απώλειας είναι προτιμότερη από την άμεση ταξινόμηση των ακμών που ανήκουν στη βέλτιστη λύση. Ενώ η ταξινόμηση μπορεί να παράγει μια πιθανότητα ότι μια ακμή ανήκει στη βέλτιστη λύση, η απώλεια μπορεί να διαφοροποιήσει ανάμεσα σε ακμές που είναι πολύ δαπανηρές για να είναι μέρος της λύσης και αυτές που δεν είναι τόσο δαπανηρές, είτε ανήκουν είτε όχι στη βέλτιστη λύση. Έτσι, η χρήση της απώλειας προωθείτε ο στόχο να βρεθούν λύσεις υψηλής ποιότητας, παρά βέλτιστες, με ελάχιστο χρόνο υπολογισμού.

4.4.5 Υπολογισμός Γενικής Απώλειας

Ο υπολογισμός της γενικής απώλειας μιας ακμής στο γράφημα του TSP απαιτεί την επίλυση του ίδιου του προβλήματος TSP, το οποίο είναι υπολογιστικά απαγορευτικό. Αντί αυτού, στοχεύετε να μαθευτεί μια συνάρτηση $\hat{r}_{ij}$ που προσεγγίζει την απώλεια μιας ακμής r_{ij}. Χρησιμοποιούνται τα GNNs για να επιτευχθεί αυτό, καθώς είναι καλά στο να προσεγγίσουν πανεπιστημιακές συναρτήσεις που λειτουργούν σε δεδομένα με δομή γραφήματος.

Μετατροπή εισόδου: Συνήθως, οι GNNs συγκεντρώνουν μηνύματα και αποθηκεύουν καταστάσεις στους κόμβους ενός γραφήματος (Gilmer κ.α., 2017) [62]. Αντί αυτού, εισάγονται το γράφημα γραμμής του αρχικού προβληματικού γραφήματος στο μοντέλο. Το γράφημα γραμμής L(G) ενός μη κατευθυνόμενου γραφήματος G είναι ένα γράφημα τέτοιο ώστε να υπάρχει ένας κόμβος στο L(G) για κάθε ακμή στο G και ότι για κάθε δύο ακμές στο G που μοιράζονται έναν κόμβο, υπάρχει μια ακμή μεταξύ των αντίστοιχων κόμβων τους στο L(G). Στο Σχήμα 4.9 εξηγείται αυτήν τη μετατροπή για ένα απλό γράφημα. Το αποτέλεσμα είναι ότι το μοντέλο συγκεντρώνει μηνύματα και αποθηκεύει καταστάσεις στις ακμές του προβληματικού γραφήματος (τους κόμβους του γραφήματος γραμμής). Αυτό επιτρέπει να δημιουργούνται μοντέλα χωρίς χαρακτηριστικά κόμβου, το οποίο είναι επιθυμητό καθώς τα βάρη των ακμών, και όχι οι συγκεκριμένες θέσεις των κόμβων, είναι σημαντικοί κατά την επίλυση του TSP. Για ένα πλήρες, μη κατευθυνόμενο γράφημα G με n κόμβους, υπάρχουν $n(n-1)/2$ κόμβοι και $n(n-1)(n-2)$ ακμές στο L(G). Έτσι, παρόλο που το G είναι πλήρες, το L(G) μπορεί να είναι πολύ αραιό.

(a) G (b) $L(G)$

Σχήμα 4.10: Μετατροπή γραφήματος από G σε L(G) [88]

Η αρχιτεκτονική του μοντέλου αποτελείται από ένα επίπεδο ενσωμάτωσης, αρκετά επίπεδα GNN και ένα επίπεδο εξόδου. Το επίπεδο ενσωμάτωσης είναι ένα επίπεδο πλήρως συνδεδεμένων ακμών που υπολογίζει ενσωματώσεις ακμής διαστάσεων dh από χαρακτηριστικά ακμής διαστάσεων dx. Τα χαρακτηριστικά κόμβων, αν χρησιμοποιούνται, συνενώνονται με το σύνολο χαρακτηριστικών των γειτονικών ακμών. Η διαδικασία προώθησης στο επίπεδο ενσωμάτωσης γράφεται ως εξής,

$$h_{ij}^0 = Wx_{ij} + b, \qquad (4.13)$$

όπου h_{ij}^0 είναι η αρχική ενσωμάτωση της ακμής *ij*, W είναι ένας πίνακας εκπαιδευμένων βαρών, x_{ij} είναι τα εισαγόμενα χαρακτηριστικά της ακμής *ij* (συμπεριλαμβανομένων χαρακτηριστικών κόμβου, αν υπάρχουν) και b είναι ένα σύνολο εκπαιδευμένων παραμέτρων. Χρησιμοποιείται dx = 1 και dh = 128, οι ενσωματώσεις των ακμών ενημερώνονται χρησιμοποιώντας T επίπεδα μετάδοσης μηνυμάτων. Κάθε επίπεδο αποτελείται από πολλά υπό επίπεδα. Η προώθηση προς τα εμπρός δίνεται από την ακόλουθη σχέση

$$h_{ij}^{t+1} = f_{BN}^t\left(h_{ij}^t + f_{MHA}^t\left(h_{ij}^t, L(G)\right)\right), \qquad (4.14)$$

$$h_{ij}^{t+1} = f_{BN}^t\left(h_{ij}^{t+1} + f_{FF}^t\left(h_{ij}^{t+1}\right)\right),$$

όπου f_{MHA} είναι ένα πολύ κεφαλικό επίπεδο GAT (Velickovi ć κ.ά., 2017) [99], f_{FF} είναι ένα επίπεδο τροφοδοσίας προς τα εμπρός, f_{BN} είναι η κανονικοποίηση παρτίδας (Ioffe & Szegedy, 2015) [100] και h_u^{t+1} είναι ένα κρυφό κατάστασης. Τα επίπεδα δεν μοιράζονται παραμέτρους. Το επίπεδο GAT χρησιμοποιεί M = 8 κεφάλια και διάστημα $dh/M = 16$, και το επίπεδο FF χρησιμοποιεί ένα κρυφό υπό επίπεδο με διάστημα 512 και ενεργοποίηση ReLU. Τέλος, το επίπεδο εξόδου είναι ένα μοναδικό πλήρως συνδεδεμένο επίπεδο ακμής που υπολογίζει έναν μονοδιάστατο εξαγόμενο αποτέλεσμα από τις dh-διάσταστες ενσωματώσεις κόμβων που υπολογίζονται από το τελικό επίπεδο μετάδοσης μηνυμάτων. Αυτό γράφεται ως

$$\hat{r}_{ij} = Wh_{ij}^T + b, \qquad (4.15)$$

Όπου το $\hat{r}_{ij}$ είναι το αποτέλεσμα για την ακμή ij και h_{ij}^T είναι η τελική ενσωμάτωση αυτής της ακμής.

4.4.6 REGRET-GUIDED Τοπικής Αναζήτησης

Ο αλγόριθμος GUIDED LOCAL SEARCH (GLS) χρησιμοποιώντας τον παράγοντα regret για την επίλυση του προβλήματος του TSP, συμπεριλαμβάνοντας τον τρόπο κατασκευής της αρχικής λύσης, τη διαδικασία τοπικής αναζήτησης και το στρατηγικό ανακάτεμα. Ο αλγόριθμος GLS χρησιμοποιεί εναλλασσόμενες φάσεις βελτιστοποίησης και ανακατέματος. Κατά τη διάρκεια μιας φάσης βελτιστοποίησης, η διαδικασία τοπικής αναζήτησης δέχεται αλλαγές στη λύση μέχρι να φτάσει σε ένα τοπικό ελάχιστο. Κατά τη διάρκεια μιας φάσης ανακατέματος, ο αλγόριθμος τιμωρεί και προσπαθεί να αφαιρέσει ακμές με υψηλό βαθμό regret από την τρέχουσα λύση, επιτρέποντας τη διαφυγή από τα τοπικά ελάχιστα και ταυτόχρονα καθοδηγώντας τον προς ελπιδοφόρες περιοχές του χώρου λύσεων, δηλαδή εκείνες με χαμηλή regret. Αποτελεσματικά, οι προβλέψεις regret που παράγει το μοντέλο μας επιτρέπουν στον αλγόριθμο GLS μας να αναιρεί αποφάσεις που κοστίζουν από τη φιλόδοξη φάση βελτιστοποίησης.

Αρχική λύση: Χρησιμοποιείται ένας εγωιστικό αλγόριθμο κοντινότερου γείτονα για να κατασκευαστή μια αρχική λύση. Ξεκινώντας από τον κόμβο προέλευσης, επιλέγεται επαναληπτικά η ακμή χαμηλότερης τιμής regret που οδηγεί σε ένα μη επισκεπτόμενο κόμβο, μέχρι να έχουν επισκεφθεί όλοι οι κόμβοι.

Γειτονιές τοπικής αναζήτησης: Η διαδικασία τοπικής αναζήτησης χρησιμοποιεί δύο τελεστές λύσης για το TSP, την relocate και την 2-opt. Εναλλάσσεται μεταξύ των δύο τελεστών και χρησιμοποιεί μια στρατηγική "βέλτιστης βελτίωσης", που σημαίνει ότι αναζητεί εξαντλητικά τη γειτονιά που αντιστοιχεί στον τρέχοντα τελεστή και αποδέχεται τη λύση που βελτιώνει την αντικειμενική συνάρτηση περισσότερο πριν συνεχίσει με τον άλλο τελεστή. Ο αλγόριθμος τερματίζει όταν δεν μπορεί να βρει βελτίωση ούτε σε μια από τις δύο γειτονιές.

Ο τελεστής relocate απλά αλλάζει τη θέση ενός μόνο κόμβου στη διαδρομή. Ο τελεστής 2-opt επιλέγει δύο κόμβους στη διαδρομή για ανταλλαγή. Αυτό χωρίζει τη διαδρομή σε τρία τμήματα: ένα αρχικό τμήμα, ένα ενδιάμεσο τμήμα και ένα τελικό τμήμα. Η διαδρομή ξανά συνθέτεται ξεκινώντας με το αρχικό τμήμα, το ενδιάμεσο τμήμα με αντίστροφη σειρά και το τελικό τμήμα. Αυτό είναι μια ειδική περίπτωση του τελεστή κ-opt (Lin & Kernighan, 1973) [93].

Στρατηγική ανακατέματος: Ορίζεται το κόστος μιας ακμής ως το προβλεπόμενο regret $\hat{r}_{ij}$ για εκείνη την ακμή. Το όφελος τιμωρίας της ακμής ij, $util_{ij}$, ορίζεται ως

$$util_{ij}(s_*) = I_{ij}(s_*)\frac{\hat{r}_{ij}}{1+p_{ij}}, \tag{4.16}$$

όπου το s_* είναι η λύση σε ένα τοπικό ελάχιστο όπως προαναφέρθηκε, $I_{ij}(s_*)$ δηλώνει εάν η λύση περιέχει την ακμή ij και p_{ij} είναι ο αριθμός των τιμωριών που έχουν ανατεθεί σε αυτήν την ακμή. Τιμωρούνται οι ακμές με τη μεγαλύτερη ωφέλεια. Στη συνέχεια, εφαρμόζεται η τοπική αναζήτηση μόνο στην τιμωρημένη ακμή. Δηλαδή, λαμβάνονται υπόψη μόνο οι λειτουργίες που θα αφαιρούσαν την τιμωρημένη ακμή.

4.4.7 Το Πείραμα

Εκπαιδεύεται το μοντέλο χρησιμοποιώντας μάθηση με επίβλεψη (supervised learning). Χρησιμοποιείται μια μόνο χαρακτηριστική είσοδος, το βάρος της ακμής (δηλαδή η Ευκλείδεια

απόσταση μεταξύ διπλανών κόμβων), για να μπορέσει να προβλεφθεί το regret των ακμών στο γράφημα του προβλήματος. Ενώ θα μπορούσαν να εξεταστούν περισσότερα χαρακτηριστικά εισόδου, αυτό συνεπάγεται υπολογιστικό κόστος.

Αξιολόγηση: Αξιολογείται η αντίληψη μεταξύ της ποιότητας της λύσης και του χρόνου υπολογισμού κατά την επίλυση τυχαία δημιουργημένων προβλημάτων TSP. Δεν υπάρχει διαθέσιμη δημόσια υλοποίηση του Arnold & Sorensen (2019a) [94], επομένως χρησιμοποιείται η υλοποίηση GLS που περιγράφεται στην ενότητα 4.4.6 με τους οδηγούς που περιγράφονται στο άρθρο τους.

Συγκρίνεται η προσέγγισή με τις παρακάτω μεθόδους, από τις οποίες οι πρώτες έξι δεν βασίζονται σε αλγόριθμους χωρίς μάθηση, ενώ οι άλλες τρεις βασίζονται σε μάθηση:

1. Πλησιέστερος γείτονας
2. Απόμακρος εισαγωγής
3. Τοπική αναζήτηση (όπως περιγράφεται στην ενότητα 4.4.6)
4. Concorde (Applegate κ.α., 2006) [39]
5. LKH-3 (Helsgaun, 2017) [91]
6. Οδηγούμενη αναζήτηση με γνώση για το πρόβλημα δρομολόγησης οχημάτων (Arnold & Sorensen, 2019a) [94]
7. Προσοχή, μάθετε να λύνεται προβλήματα δρομολόγησης! (Kool κ.α., 2018) [86]
8. Μια αποδοτική τεχνική γραφικής επεξεργασίας δικτύου για το πρόβλημα του περιοδεύοντος πωλητή (Joshi κ.α., 2019) [102]
9. Μάθηση μεθόδων 2-opt για το πρόβλημα του περιοδεύοντος πωλητή μέσω ενδυνάμωσης μάθησης βαθιάς μάθησης (da Costa κ.α., 2020) [103]

Ακολουθούνται οι οδηγίες για την υπολογιστική δοκιμή μηχανικής μάθησης προσεγγίσεων σε προβλήματα δρομολόγησης οχημάτων (Accorsi κ.α., 2021) [104] και πραγματοποιούνται δύο είδη πειραμάτων. 1) Αφήνονται τόσο ο χρόνος υπολογισμού όσο και η ποιότητα της λύσης ελεύθερα. Υπολογίζεται ο μέσος χρόνος βελτιστοποίησης και ο χρόνος υπολογισμού ανά πρόβλημα στο σύνολο των προβλημάτων. Χρησιμοποιούνται οι περιγραφόμενες υλοποιήσεις. 2) Σταθεροποιείται ο χρόνος υπολογισμού και μετριέται η ποιότητα της λύσης, σε όρους μέσου χρόνου βελτιστοποίησης και αριθμού προβλημάτων που επιλύθηκαν αριστοτεχνικά. Τροποποιήθηκαν οι υλοποιήσεις ώστε να τρέχουν συνεχώς μέχρι το πέρας του χρόνου υπολογισμού. Επιτρέπεται ένας περιορισμένος αριθμός επαναλήψεων που ορίζονται για να μην περιοριστεί η ικανότητα της αλγοριθμικής μεθόδου για βελτίωση.

Αξιολογούνται τα σετ προβλημάτων μια φορά ανά πρόβλημα, πράγμα που επιτρέπει να μετρηθεί ακριβώς ο χρόνος υπολογισμού ανά πρόβλημα. Επιτρέπεται η παράλληλη επεξεργασία στη μονάδα επεξεργασίας γραφικών (GPU) για να ληφθούν δείγματα πολλαπλών λύσεων για ένα μόνο πρόβλημα ταυτόχρονα, όπου οι υλοποιήσεις έχουν αλλάξει ανάλογα. Χρησιμοποιείται ένα κοινό, μοναδικό σύστημα επεξεργασίας κεντρικής μονάδας (CPU) και μια μοναδική μονάδα επεξεργασίας γραφικών (GPU) για όλα τα πειράματα.

Πραγματοποιούνται και τα δύο είδη πειραμάτων σε τρία σετ προβλημάτων, TSP20, TSP50 και TSP100, που αποτελούνται από χίλια προβλήματα TSP 2D Ευκλείδεια με 20, 50 και 100 κόμβους, αντίστοιχα. Δημιουργούνται τα προβλήματα με την ομοιόμορφη τυχαία δειγματοληψία τοποθεσιών κόμβων στο τετράγωνο μονάδας $[0, 1]^2$, το οποίο είναι σύμφωνο με τις μεθόδους που χρησιμοποιούν οι Kool κ.α. (2018), Joshi κ.α. (2019) και da Costa κ.α. (2020) [86][102][103]. Στο δεύτερο είδος πειραμάτων (σταθερός χρόνος υπολογισμού), επικεντρώνονται στις προσεγγίσεις που βασίζονται στην μάθηση και διατηρούμε το Concorde ως αναφορά. Επιπλέον, αξιολογείται η γενίκευση της απόδοσης των προσεγγίσεων που βασίζονται στη μάθηση από μικρότερα σε μεγαλύτερα σύνολα προβλημάτων και από τα τυχαία δημιουργημένα προβλήματα σε προβλήματα TSPLIB, τα οποία αντιπροσωπεύουν προβλήματα του πραγματικού κόσμου.

Τα αποτελέσματα δείχνουν την απόδοση κάθε προσέγγισης όταν ο χρόνος υπολογισμού και η ποιότητα της λύσης δεν είναι σταθεροποιημένα και φαίνονται στον Πίνακα 4.6. Οι προσεγγίσεις που δεν κυριαρχούν από άλλες (δηλαδή παράγουν λύσεις υψηλότερης ποιότητας ή είναι πιο γρήγορες) είναι έντονα. Αυτές οι τιμές αποτελούν το μέτωπο Pareto για την εξισορρόπηση ανάμεσα στην ποιότητα της λύσης και τον χρόνο υπολογισμού. Οι προσεγγίσεις που δεν βασίζονται στην μάθηση και οι προσεγγίσεις που βασίζονται στην μάθηση αξιολογούνται ξεχωριστά. Η προσέγγισή αυτού το κεφαλαίου βρίσκεται πάντα στο μέτωπο Pareto.

Για να αξιολογηθεί καλύτερα αυτή η εξισορρόπηση, το δεύτερο πείραμα σταθεροποιεί τον χρόνο υπολογισμού και μετρά τη ποιότητα της λύσης καθώς εξελίσσεται στον χρόνο.

Problem	Method	Optimality time (s)	Optimality gap (%)
TSP20	Nearest Neighbor	**0.000±0.000**	**17.448±10.209**
	Farthest Insertion	**0.005±0.000**	**2.242±2.536**
	Local Search	**0.007±0.003**	**1.820±3.254**
	Arnold κ.α.	10.009±0.008	0.000±0.000
	Concorde	0.119±0.083	0.000±0.000
	LKH-3	**0.020±0.019**	**0.000±0.000**
	Kool κ.α.	**0.038±0.005**	**0.069±0.255**
	Joshi κ.α.	1.334±0.106	2.031±2.928
	O. da Costa κ.α.	21.120±0.352	0.001±0.025
	Hudson κ.α.[88]	**10.010±0.008**	**0.000±0.000**
	Kwon κ.α.*	-	0.00
	Fu κ.α.	-	0.000
TSP50	Nearest Neighbor	**0.002±0.000**	**23.230±7.968**
	Farthest Insertion	**0.065±0.001**	**7.263±3.072**
	Local Search	0.101±0.031	3.357±2.603
	Arnold κ.α.	10.035±0.039	0.040±0.180
	Concorde	**0.258±0.197**	**0.000±0.000**
	LKH-3	**0.069±0.030**	**0.000±0.012**
	Kool κ.α.	**0.124±0.006**	**0.494±0.655**
	Joshi κ.α.	3.080±0.182	0.884±1.700
	O. da Costa κ.α.	24.338±0.523	0.136±0.314

	Hudson κ.α.	**10.037±0.039**	**0.009±0.069**
	Kwon κ.α.*	-	0.03
	Fu κ.α.	-	0.0145
TSP100	Nearest Neighbor	**0.010±0.000**	**25.104±6.909**
	Farthest Insertion	0.444±0.010	12.456±2.944
	Local Search	0.727±0.165	4.169±2.047
	Arnold κ.α.	10.128±0.194	1.755±1.240
	Concorde	**0.573±0.771**	**0.000±0.000**
	LKH-3	**0.118±0.022**	**0.011±0.058**
	Kool κ.α.	**0.356±0.007**	**2.368±1.185**
	Joshi κ.α.	6.127±0.081	1.880±4.097
	O. da Costa et al.	30.656±0.734	0.773±0.717
	Hudson κ.α.	**10.108±0.175**	**0.698±0.801**
	Kwon κ.α.*	-	0.14
	Fu κ.α.	-	0.037

Πίνακας 4.6: Ποιότητα λύσης και χρόνος υπολογισμού για διάφορες προσεγγίσεις

Ο Πίνακας 4.7 δείχνει την απόδοση των προσεγγίσεων που βασίζονται στην μάθηση μετά από 10 δευτερόλεπτα χρόνου υπολογισμού ανά πρόβλημα. Η προσέγγισή αυτού το κεφαλαίου βρίσκει καλύτερες λύσεις σε μέσο όρο από τις άλλες προσεγγίσεις για όλα τα σετ προβλημάτων. Ιδιαίτερα, μειώνοντας το μέσο ποσοστό βελτίωσης στο σύνολο προβλημάτων με 50 κόμβους κατά 0,268%

σε 0,009%, μια βελτίωση 30 φορές, και στο σύνολο προβλημάτων με 100 κόμβους από 1,534% σε 0,705%, μια βελτίωση 2 φορές. Η προσέγγισή αυτή βρίσκει πιο βέλτιστες λύσεις για τα σύνολα προβλημάτων με 20 και 50 κόμβους από τις άλλες προσεγγίσεις.

Problem	Method	Optimality gap (%)	Optimality solutions (%)
TSP20	Concorde	0.000±0.000	100.0000
	Kool κ.α.	0.069±0.255	84.6000
	Joshi κ.α.	1.000±12.492	97.4000
	O. da Costa κ.α.	0.002±0.027	99.0000
	Hudson κ.α.[88]	**0.000±0.000**	**100.0000**
TSP50	Concorde	0.000±0.000	100.0000
	Kool κ.α.	0.494±0.655	29.5000
	Joshi κ.α.	1.206±18.066	86.3000
	O. da Costa κ.α.	0.268±0.492	48.7000
	Hudson	**0.009±0.069**	**96.2000**
TSP100	Concorde	0.000±0.000	100.0000
	Kool κ.α.	1.958±1.064	0.3000
	Joshi κ.α.	1.559±4.071	**51.6000**

O. da Costa κ.α.	1.534±1.098	1.4000
Hudson	**0.705±0.806**	20.5000

Πίνακας 4.7: Ποιότητα λύσης μετριέται μετά από 10 δευτερόλεπτα χρόνο υπολογισμού ανά περίπτωση

Αξιολογώντας την απόδοση της προσέγγισής αυτής και άλλων προσεγγίσεων που βασίζονται στην μάθηση όταν γενικεύονται από μικρότερα προβλήματα σε μεγαλύτερα και από τυχαία δημιουργημένα προβλήματα σε πραγματικά προβλήματα TSPLIB που αντιπροσωπεύουν πραγματικές καταστάσεις. Η συγκεκριμένη προσέγγισή γενικεύεται καλά. Ιδιαίτερα, όταν γενικεύεται από προβλήματα με 20 κόμβους στο σύνολο προβλημάτων με 100 κόμβους, μειώνοντας το μέσο ποσοστό βελτιστότητας από 18,845% σε 2,622%, μια βελτίωση 7 φορές.

4.5 Graph Neural Network Assisted Monte Carlo Tree Search Approach to Traveling Salesman Problem

Εκτός από την ενσωμάτωση γραφήματος [87], τα γραφικά νευρωνικά δίκτυα (GNN) χρησιμοποιούνται ευρέως για την επίλυση προβλημάτων βασισμένων σε γραφήματα. Πρόσφατα, κάποιοι ερευνητές παρατήρησαν ότι τα GNN μπορούν να χρησιμοποιηθούν για να ανακαλύψουν χρήσιμα μοτίβα του TSP [66], [105] και άλλων συνδυαστικών προβλημάτων βελτιστοποίησης βασισμένων σε γραφήματα [106],[107], και έδειξαν ότι τα GNN μπορούν να ανακαλύψουν κοινά μοτίβα που υπάρχουν σε γραφήματα διαφορετικής κλίμακας.

Εμπνευσμένοι από την επιτυχία των GNN σε προβλήματα συνδυαστικής βελτιστοποίησης, οι συγγραφείς αυτού του μοντέλου [108] εκπαιδεύουν ένα αναπτυγμένο GNN για να αναπαραστήσει καλύτερα τα χαρακτηριστικά του TSP. Επειδή η έξοδος του MCTS είναι η πληροφορία ανατροφοδότησης που συνδυάζει την προηγούμενη πιθανότητα με την αναζήτηση εξερεύνησης, συνδυάζεται το GNN με το MCTS για να παραχθούν πιο αξιόπιστες αποφάσεις.

4.5.1 Προσέγγιση του μοντέλου

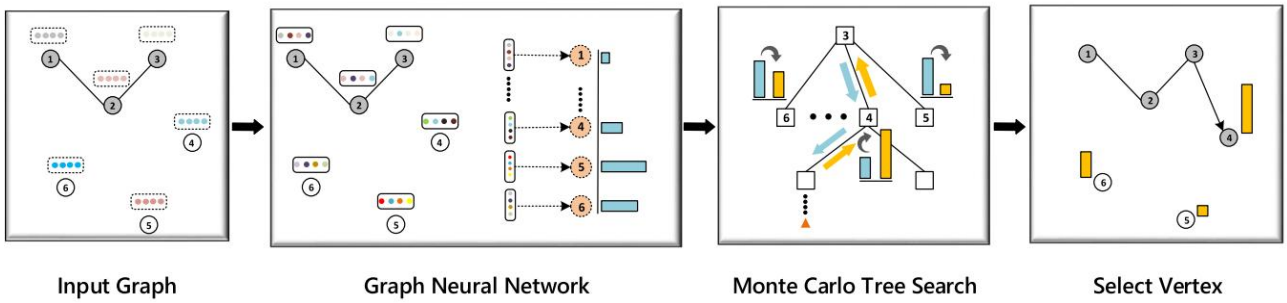
Έστω ότι *G(V, E)* αναπαριστά έναν βαρυσήμαντο γράφημα, όπου *V* είναι το σύνολο των κορυφών και *E* είναι το σύνολο των ακμών. Επίσης, έστω ότι *e(u, v)* είναι το βάρος της ακμής *(u, v)* $\in E$, όπου u, v $\in$ V. Χρησιμοποιούμε το $S = \{v1, v2, \ldots, vi\}$ για να αναπαραστήσουμε μια ακολουθία περιοδείας που ξεκινά με το v1 και τελειώνει με το vi, και $\overline{S}$ = V \ S είναι το σύνολο των υποψήφιων κορυφών για προσθήκη στο S.

Δεδομένου του γράφημα υ *G(V, E)* ή απλά **G**, στόχος αυτής της προσέγγισης είναι να παράγει μια περιοδεία προσθέτοντας την κορυφή $v \in \overline{S}$ στο S με τη σειρά. Μια φυσική προσέγγιση είναι να

εκπαιδεύεται ένα νευρωνικό δίκτυο βαθιάς μάθησης για να προβλέπει ποια κορυφή θα προστεθεί στην μερική ακολουθία περιοδείας σε ένα συγκεκριμένο βήμα. Δηλαδή, το νευρωνικό δίκτυο *f (G|S)* θα λαμβάνει ως είσοδο το γράφημα *G* και τη μερική ακολουθία περιοδείας *S* και θα επιστρέφει πιθανότητες για τις κορυφές που υποδηλώνουν την πιθανότητα κάθε κορυφής να επιλεγεί. Για το TSP, τα δομικά πρότυπα του γραφήματος και η κατάσταση των κορυφών μπορεί να γίνουν πολύπλοκα για να περιγραφούν.

Για να αναπαρασταθεί ένα τέτοιο πολύπλοκο περιβάλλον, θα τεθεί σε χρήση το γραφικό νευρωνικό δίκτυο (GNN) [109] για να ρυθμιστεί το *f (G|S)*. Είναι δυνατόν να χρησιμοποιηθεί απευθείας η προηγούμενη πιθανότητα για μια κορυφή, π.χ. να επιλεχθεί μια κορυφή με τη μεγαλύτερη πιθανότητα, για να δημιουργηθεί η ακολουθία περιοδείας αυξητικά. Ωστόσο, η παραγωγή περιοδειών με αυτόν τον τρόπο μπορεί να μην είναι αξιόπιστη, καθώς ένας αλγόριθμος μάθησης με βάση την μάθηση έχει μόνο μία ευκαιρία για να υπολογίσει την βέλτιστη περιοδεία και δεν επιστρέφει πίσω για να αναστρέψει την απόφαση. Για να αντιμετωπιστεί αυτό το μειονέκτημα, συνδυάζεται το γραφικό νευρωνικό δίκτυο με την αναζήτηση δέντρου Monte Carlo [110],[111] για να υπάρχει μια καλύτερη δομή.

Από τη μια πλευρά, χρησιμοποιείται μια παραλλαγή του PUCT [112] για να ισορροπήσουν την εξερεύνηση (δηλαδή την επίσκεψη σε μια κατάσταση όπως προτείνεται από την προηγούμενη πολιτική) και την εκμετάλλευση (δηλαδή την επίσκεψη σε μια κατάσταση που έχει την καλύτερη τιμή). Χρησιμοποιώντας την έννοια της προηγούμενης πιθανότητας, το χώρο αναζήτησης του δέντρου μπορεί να μειωθεί σημαντικά, επιτρέποντας στην αναζήτηση να εκχωρήσει περισσότερους υπολογιστικούς πόρους στις καταστάσεις που έχουν υψηλότερες τιμές. Από την άλλη πλευρά, μπορεί να δοθεί μια πιο αξιόπιστη πολιτική μετά από ένα μεγάλο αριθμό προσομοιώσεων, καθώς η έξοδος της αναζήτησης δέντρου Monte Carlo λειτουργεί ως πληροφορία ανατροφοδότησης συνδυάζοντας την προηγούμενη πιθανότητα με την αναζήτηση εξερεύνησης. Η συνολική προσέγγιση απεικονίζεται στο Σχήμα 4.11.



Σχήμα 4.11: Επισκόπηση προσέγγισης [108]

Για να κατορθωθεί ένα καλό δίκτυο, απαιτείται πληροφορία για τις δομές του γραφήματος και την συνακόλουθη πληροφορία, δηλαδή την ακολουθία της περιοδείας $S = \{v1, \ldots, vi\}$. Επίσης μειώνεται η κορυφή *v* με *x_v = 1* αν έχει ήδη επισκεφθεί, διαφορετικά *x_v = 0*. Ευθύνη της *f(G|S)* είναι να περιλαμβάνει την κατάσταση ενός τέτοιου "σημαδεμένου" γραφήματος και να παράγει την προηγούμενη πιθανότητα για κάθε κορυφή που πρόκειται να περιληφθεί στο *S*.

Αντίθετα από άλλα προβλήματα συνδυαστικής βελτιστοποίησης βασισμένα σε γραφήματα, όπως το Maximal Independent Set (MIS) και το Minimum Vector Cover (MVC) [106] όπως αναφέρεται στο

[87], δεν αγνοούνται τα χαρακτηριστικά των ακμών καθώς ο στόχος του TSP υπολογίζεται με βάση το κόστος των ακμών, δηλαδή την απόσταση μεταξύ δύο κορυφών. Έτσι, τροποποιείται το βασικό GNN [109], το οποίο ονομάζεται static edge graph neural network (SE-GNN), για να εξαχθούν αποτελεσματικά τα χαρακτηριστικά των κόμβων και των ακμών του TSP.

1. ΑΝΑΠΑΡΆΣΤΑΣΗ ΓΡΑΦΙΚΟΎ ΝΕΥΡΩΝΙΚΟΎ ΔΙΚΤΥΟΥ

Ένα μοντέλο GNN αποτελείται από μια στοίβα T επιπέδων νευρωνικού δικτύου, όπου κάθε επίπεδο συγκεντρώνει την τοπική γειτονιά της πληροφορίας, δηλαδή τα χαρακτηριστικά των γειτόνων κάθε κόμβου, και μεταφέρει αυτήν τη συγκεντρωμένη πληροφορία στο επόμενο επίπεδο. Χρησιμοποιείται το H_t^v για να συμβολίσουν το διάνυσμα πραγματικών αριθμών που σχετίζεται με τον κόμβο *v* στο επίπεδο *t*. Συγκεκριμένα, το βασικό μοντέλο GNN [109] μπορεί να υλοποιηθεί ως εξής. Στο επίπεδο t = 1, 2, ..., T, το νέο χαρακτηριστικό υπολογίζεται όπως παρουσιάζεται στο (4.16)

$$H_v^{t+1} = \sigma\left(H_v^t W_1^t + \sum_{u\in N(v)} H_u^t W_2^t\right) \quad (4.16)$$

Στο (1), *N(v)* είναι το σύνολο των γειτόνων του κόμβου *v*, W_t^1 και W_t^2 είναι οι πίνακες παραμέτρων για το επίπεδο t, και *σ(·)* υποδηλώνει μια συνάρτηση μη γραμμικής συνιστώσας, όπως η σιγμοειδής συνάρτηση ή η συνάρτηση ReLU. Για t = 0, H_0^v υποδηλώνει την αρχικοποίηση των χαρακτηριστικών στο επίπεδο εισόδου.

Όπως παρατηρείται στο (4.16), η πληροφορία των ακμών δεν λαμβάνεται υπόψη. Υπάρχουν πολλοί τρόποι για να ενσωματωθούν τα χαρακτηριστικά των ακμών. Οι Gilmer κ.α. [62] θεώρησαν την πληροφορία της ακμής ως ένα είδος μετρικής για να μετρήσουν την χρησιμότητα των γειτόνων ενός κόμβου, ενώ οι Dai κ.α. [87] και Xie και Grossman [113] θεώρησαν τα χαρακτηριστικά των ακμών ως ανεξάρτητα τμήματα και τα ενσωμάτωσαν με τα χαρακτηριστικά των κόμβων. Επειδή η πληροφορία των ακμών στο TSP είναι σημαντική, ακολουθούν τους τρόπους που παρουσιάζονται στο [87][113]. Σύμφωνα με αυτό, τα χαρακτηριστικά των ακμών μπορούν να ενσωματωθούν με τα χαρακτηριστικά των κόμβων χρησιμοποιώντας το (4.17) [87].

$$\mu_v^{t+1} = \sigma\left(\theta_1 x_v + \theta_2 \sum_{u\in N(v)} \mu_u^t + \theta_3 \sum_{u\in N(v)} \sigma\left(\theta_4 w(v,u)\right)\right) \quad (4.17)$$

Στο (4.17), $\theta_1 \in R^l$, $\theta_2, \theta_3 \in R^{(l\times l)}$ και $\theta_4 \in R^l$ είναι παράμετροι μοντέλου.

Στο (4.16) και (4.17) φαίνεται ότι η μη γραμμική απεικόνιση της συγκεντρωμένης πληροφορίας είναι ενός στρώματος perceptron, το οποίο δεν είναι αρκετό για να απεικονίσει διακριτά πολλαπλά σύνολα σε μοναδικά ενσωματωμένες δομές. Για το λόγο αυτό, σύμφωνα με την πρόταση στο [85], αντικαθιστούν το μονό perceptron με πολυεπίπεδο perceptron. Τελικά, υπολογίζεται το νέο χαρακτηριστικό κόμβου *H* χρησιμοποιώντας το (4.18).

$$H_v^{t+1} = MLP^t\left(H_v^t W_v^t + \sum_{u\in N(v)} H_u^t W_2^t + \sum_{u\in N(v)} e_{v,u} W_3^t\right) \quad (4.18)$$

Στο (4.18), *e(v, u)* είναι το χαρακτηριστικό της ακμής, W_t^3 είναι πίνακες παραμέτρων, και MLP_t είναι το πολυεπίπεδο perceptron για το επίπεδο t.

Να σημειωθεί ότι το SE-GNN διαφέρει από το GEN [87] στις ακόλουθες πτυχές: 1) Το SE-GNN αντικαθιστά το xv στο (4.17) με το H_v, έτσι ώστε το SE-GNN να μπορεί να ενσωματώνει απευθείας το τελευταίο χαρακτηριστικό του ίδιου του κόμβου. 2) Η κάθε διαδικασία ενημέρωσης στο GEN μπορεί να θεωρηθεί ως ένα επίπεδο ενημέρωσης του SE-GNN, δηλαδή κάθε υπολογισμός αντιστοιχεί σε μια προώθηση ένα επίπεδο μπροστά, επομένως χρειάζονται *T* επαναλήψεις για *T* επίπεδα. Οι παράμετροι

κάθε επιπέδου του SE-GNN είναι ανεξάρτητες, ενώ οι παράμετροι του GEN κοινοποιούνται μεταξύ διαφορετικών διαδικασιών ενημέρωσης, πράγμα που περιορίζει την ικανότητα του νευρωνικού δικτύου. 3) Αντικαθιστούν την συνάρτηση στάσης στο (4.17) με το MLP, όπως προτείνεται από το [85], για να βοηθήσουν τα νευρωνικά δίκτυα να απεικονίσουν διακριτά πολλαπλά σύνολα σε μοναδικά ενσωματωμένες δομές.

Η αρχικοποίηση από τα χαρακτηριστικά των κόμβων H^0 ως εξής. Κάθε κορυφή έχει ένα χαρακτηριστικό ετικέτας που είναι ένα διάνυσμα 3 διαστάσεων. Το πρώτο στοιχείο του διανύσματος είναι δυαδικό και είναι ίσο με 1 αν η μερική ακολουθία περιοδείας S περιέχει την κορυφή. Τα δεύτερο και τρίτο στοιχείο της ετικέτας είναι, αντίστοιχα, οι τιμές των συντεταγμένων x και y της κορυφής. Το πρόβλημα είναι να βρεθεί μια διαδρομή από την τελευταία κορυφή στην πρώτη, περνώντας από όλες τις κορυφές που δεν έχουν επισκεφτεί. Για να μαθευτούν οι πρώτες και τελευταίες κορυφές στην μερική ακολουθία περιοδείας *S*, επεκτείνουν τα χαρακτηριστικά των κόμβων H^0 προσθέτοντας τις ετικέτες χαρακτηριστικών αυτών των κορυφών, εκτός από τις βασικές ετικέτες που περιγράφηκαν παραπάνω.

2. ΠΑΡΑΜΕΤΡΟΠΟΊΗΣΗ f(G|S; 2)

Αφού υπολογιστεί το χαρακτηριστικό για κάθε κορυφή μετά την ενημέρωση των T επιπέδων, χρησιμοποιείται το νέο χαρακτηριστικό για τις κορυφές για να οριστεί η συνάρτηση f (G|S; 2), η οποία επιστρέφει την προτεραιότητα πιθανότητας για κάθε κορυφή που υποδεικνύει πόσο πιθανό είναι η κορυφή να ανήκει στην μερική ακολουθία περιοδείας S. Πιο συγκεκριμένα, συνδυάζονται όλα τα χαρακτηριστικά κορυφών H^T_v για να οριστεί η f (G|S; 2) όπως παρουσιάζεται στο (4.19), όπου η συνάρτηση softmax εφαρμόζεται στα άθροισμα των χαρακτηριστικών.

$$f(G|S;\ 2) = \text{softmax}\left(sum\left(H_1^T\right), \ldots, sum\left(H_n^T\right)\right) \quad (4.19)$$

Κατά την εκπαίδευση, ελαχιστοποιείται το σφάλμα της συνάρτησης απώλειας για κάθε δείγμα εκπαίδευσης (Gi, Si) με τη μέθοδο εκπαίδευσης με επίβλεψη όπως φαίνεται στο (4.20),

$$l\left(S_i, F(G_i|S_i;\Theta)\right) = -\sum_{j=1}^{N} y_i \log f(G_i|S_i(1{:}\,j-1);\Theta) \quad (4.20)$$

όπου η συνάρτηση softmax χρησιμοποιείται για την εκτίμηση των πιθανοτήτων. Στο (4.20), η Si αντιπροσωπεύει μια ακολουθία περιοδείας που αποτελεί μια μετάθεση των κορυφών του γραφήματος Gi, ενώ το yj είναι ένα one-hot διάνυσμα του μήκους N, όπου η j-οστή θέση είναι 1.

Όπως υλοποιείται και στην [114], η GNN-MCTS χρησιμοποιεί τις γραφικές δικτυώσεις νευρώνων ως καθοδηγητή του MCTS. Κάθε κόμβος s στο δέντρο αναζήτησης περιέχει ακμές (s, a) για όλες τις έγκυρες ενέργειες a ∈ A(s). Κάθε ακμή αποθηκεύει ένα σύνολο στατιστικών:

$$\{N(s, a), Q(s, a), P(s, a)\}$$

όπου το κόμβος s υποδηλώνει την τρέχουσα κατάσταση του γραφήματος, συμπεριλαμβανομένης της μερικής ακολουθίας περιοδείας S και άλλες πληροφορίες του γραφήματος, ενώ η ενέργεια a υποδηλώνει την επιλογή της κορυφής v από το $\overline{S}$ στο S. Το N(s, a) είναι ο αριθμός επισκέψεων, το Q(s, a) είναι η τιμή της ενέργειας και το P(s, a) είναι η προτεραιότητα πιθανότητας για την επιλογή της ακμής (s, a).

Πρέπει να αναφερθεί ότι υπάρχουν τρεις κύριες διαφορές ανάμεσα στο TSP και το παιχνίδι Go:

- Το παιχνίδι Go παρακολουθεί τον μέσο όρο ρυθμού κέρδους ενός κλαδιού του MCTS για να καθοδηγήσει τις κινήσεις [114]. Ωστόσο, το TSP ενδιαφέρεται να βρει την άκρη, και άρα η μέση τιμή δεν έχει νόημα για αυτό, καθώς μερικές υπό βέλτιστες διαδρομές μπορεί να περιβάλλουν ακόμη και την άκρη της διαδρομής. Αντί να καταγράφει τη μέση τιμή ενέργειας, καταγράφεται η καλύτερη τιμή ενέργειας που βρέθηκε στο υπό δέντρο κάθε κόμβου για να προσδιορίσουν την τιμή εκμετάλλευσης του κόμβου στο δέντρο.
- Στο παιχνίδι Go, είναι συνηθισμένο να χρησιμοποιούμε {0, 0,5, 1} για να υποδείξουν, αντίστοιχα, την ήττα, την ισοπαλία και τη νίκη σε ένα παιχνίδι. Είναι όχι μόνο βολικό, αλλά πληροί και τις απαιτήσεις του UCT [111] εάν το ανταμοιβή βρίσκεται στο εύρος [0, 1]. Στο TSP, μπορεί να επιτευχθεί ένα αυθαίρετο μήκος περιοδείας που δεν ανήκει σε καθορισμένο διάστημα. Μπορεί να επιλυθεί αυτό το θέμα προσαρμόζοντας τις παραμέτρους του UCT έτσι ώστε να είναι εφικτό για ένα συγκεκριμένο διάστημα. Η προσαρμογή παραμέτρων απαιτεί σημαντικές δοκιμές λόγω της αλλαγής στον αριθμό των κορυφών. Αντί αυτού, επιλύουν αυτό το ζήτημα με το να κανονικοποιούν την τιμή ενέργειας του κόμβου n, το οποίος ο γονέας του είναι ο κόμβος p, στο εύρος [0, 1] με χρήση του (4.21).

$$Q_n = \frac{Q_n - w_p}{b_p - w_p} \tag{4.21}$$

- Το AlphaGo χρησιμοποιεί μια μάθησης κριτήριο (critic) v(s, θ) για να εκτιμήσει την πιθανότητα του τρέχοντος παίκτη να κερδίσει από τη θέση s, όπου η παράμετρος θ μαθαίνεται από παρατήρηση (s, π). Ωστόσο, στο TSP, ένα τέτοιο μάθησης κριτήριο θα πρέπει να αντέχει τη μεγάλη αλλαγή της τιμής μεταξύ διαφορετικών λύσεων, ενώ αναμένεται επίσης να διατηρεί την ευαισθησία στην πολύ μικρή αλλαγή τιμής γύρω από τη βέλτιστη λύση. Επομένως, στο GNN-MCTS [108] οι βελτιωμένες τιμές ενέργειας κόμβων υπολογίζονται με την κανονικοποίηση της τιμής ενέργειας κόμβου n, ο γονέας του οποίου είναι ο κόμβος p, στο εύρος [0, 1] χρησιμοποιώντας το (4.21).

Στην άποψή των [108], η παραπάνω αιτία καθιστά το μάθησης κριτήριο τιμών δύσκολο να λειτουργήσει στο TSP. Αντί αυτού, σχεδιάζουν ένα μη μάθησης κριτήριο τιμών h(s) που συνδυάζει το προ-εκπαιδευμένο GNN και την αναζήτηση δέσμης για να αξιολογήσει το μήκος της πιθανής περιοδείας από την τρέχουσα κατάσταση έως την τελική κατάσταση. Καθοδηγούμενο από την έξοδο του προ-εκπαιδευμένου GNN, το κριτήριο τιμών εκτελεί την αναζήτηση δέσμης από το κατάσταση φύλλου που αντιστοιχεί στον κόμβο φύλλου l έως να φτάσει σε μια τελική κατάσταση. Ορίζεται η τιμή του κόμβου φύλλου l ως Vl = −h(κατάσταση φύλλου).

Η GNN-MCTS προχωρά επαναλαμβάνοντας τις τέσσερις φάσεις και στη συνέχεια επιλέγει μια κίνηση για να παιχτεί.

- Στρατηγική Επιλογής. Η πρώτη φάση κάθε rollout αρχίζει στη ρίζα του κόμβου s0 του δέντρου αναζήτησης και ολοκληρώνεται όταν η rollout φτάνει σε έναν κόμβο φύλλου sl στο χρονικό βήμα l. Στο χρονικό βήμα t <l, χρησιμοποιούν μια παραλλαγή του PUCT [115] για να ισορροπήσουν την εξερεύνηση (δηλαδή, να επισκέπτεται τις καταστάσεις όπως προτείνεται από την προηγούμενη πολιτική) και την εκμετάλλευση (δηλαδή, να επισκέπτεται τις καταστάσεις που έχουν την καλύτερη τιμή) ανάλογα με τις στατιστικές στο δέντρο αναζήτησης όπως δίνεται από τις (4.22) και (4.23) αντίστοιχα, όπου cpuct είναι μια σταθερά που ανταλλάσσει μεταξύ εξερεύνησης και εκμετάλλευσης.

$$\text{at} = \arg\max_a \big(Q(s_t, a) + U(s_t, a)\big) \tag{4.22}$$

$$U(s,a) = c_{puct} P(s,a) \frac{\sqrt{\sum_b N(s,b)}}{1+N(s,a)} \quad (4.23)$$

- **Στρατηγική Διασποράς**. Όταν φτάνει ένας κόμβος φύλλου l, αξιολογείται η αντίστοιχη κατάσταση sl για να αποκτηθεί η προτεραιότητα πιθανότητας p των παιδικών κόμβων της. Ο κόμβος φύλλου διευρύνεται και η στατιστική κάθε ακμής (sl, a) αρχικοποιείται σε {N(sl, a) = 0, Q(sl, a) = -∞, 2P(sl, a) = pa}.
- **Στρατηγική Προσομοίωσης**. Αντί να χρησιμοποιεί μια τυχαία στρατηγική, χρησιμοποιούν τη συνάρτηση τιμών h(s) για να αξιολογηθεί το μήκος της περιοδείας που μπορεί να προκύψει από τον κόμβο φύλλου l.
- **Στρατηγική Ανάδρασης**. Για κάθε βήμα t <l, οι στατιστικές των ακμών ενημερώνονται σε αντίστροφη διαδικασία. Οι μετρήσεις επισκέψεων αυξάνονται ως N(st, at) = N(st, at) + 1, και η τιμή δράσης ενημερώνεται στην καλύτερη τιμή ως Q(st, at) = max(Q(st, at), Vl).
- **Παιχνίδι**. Στο τέλος των διαφόρων rollouts, επιλέγεται ο κόμβος με το μεγαλύτερο Pˆ(a|s0) = 1 - P Q(s0,a) / Σ Q(s0,b) ως την επόμενη κίνηση a στην αρχική θέση s0. Το δέντρο αναζήτησης θα επαναχρησιμοποιηθεί σε μελλοντικά βήματα: ο κόμβος παιδί που σχετίζεται με τον επιλεγμένο κόμβο γίνεται ο νέος κόμβος ρίζας, και όλες οι στατιστικές του υπό δέντρου κάτω από αυτόν τον κόμβο παραμένουν.

4.5.2 Πειραματική Διαμόρφωση

1. **ΔΗΜΙΟΥΡΓΊΑ ΠΕΡΙΣΤΆΣΕΩΝ**. Για να αξιολογηθεί η μέθοδός αυτή έναντι άλλων αλγορίθμων προσέγγισης και μεθόδων βασισμένων σε βαθιά μάθηση, χρησιμοποιείται ένας δημιουργός περιστάσεων από τον Διαγωνισμό TSP DIMACS [116] για να δημιουργηθούν δύο είδη Ευκλείδειων περιστάσεων: "τυχαίες" περιστάσεις που αποτελούνται από n σημεία που διασκορπίζονται ομοιόμορφα τυχαία στο τετράγωνο [106, 106], και "συμπυκνωμένες" περιστάσεις που αποτελούνται από n σημεία που μοιράζονται σε n/100 ομάδες. Λαμβάνεται υπόψη τρεις δοκιμαστικές περιπτώσεις, δηλαδή Euclidean TSP20, TSP50 και TSP100.

2. **ΒΑΣΙΚΈΣ ΤΙΜΈΣ.** Για να υπολογιστούν οι βέλτιστες λύσεις, χρησιμοποιούνται δύο προηγμένες μεθόδους επίλυσης, το Concorde 3 [117] και το Gurobi 4 [108]. Συγκρίνονται τα αποτελέσματά της μεθόδου με αυτά των μεθόδων Nearest, Random και Farthest Insertion, καθώς και Nearest Neighbor, που είναι μη μαθησιακοί αλγόριθμοι βάσεις που προκύπτουν και αυτοί από μια περιοδεία με την προσθήκη κόμβων διαδοχικά. Επιπλέον, συγκρίνονται τα αποτελέσματά με αυτά των εξαιρετικών μεθόδων βασισμένων σε βαθιά μάθηση, κυρίως οι μέθοδοι των Vinyals κ.α.[118], Bello κ.α. [119] και Kool κ.α. [86].

3. **ΕΚΠΑΊΔΕΥΣΗ ΚΑΙ ΔΟΚΙΜΉ.** Για να εκπαιδευτεί το SE-GNN, δημιουργούνται 100.000, 40.000 και 20.000 περιστάσεις για τα TSP20, TSP50 και TSP100, αντίστοιχα. Χρησιμοποιούνται δύο προηγμένους τρόπους επίλυσης (Gurobi και Concorde) για να ληφθεί η βέλτιστη περιοδεία για κάθε περίπτωση. Στη συνέχεια δημιουργούνται δείγματα για κάθε περίπτωση σύμφωνα με την βέλτιστη ακολουθία περιοδείας. Διαιρείται το σύνολο δεδομένων σε ένα σύνολο εκπαίδευσης, ένα σύνολο επικύρωσης και ένα σύνολο δοκιμών σε αναλογία 8:1:1. Χρησιμοποιείται το Adam [120] με 512 ελάχιστες παρτίδες και ρυθμό μάθησης 10^{-3}. Η εκπαίδευση πραγματοποιείται για 60 εποχές σε μια μηχανή με κάρτα γραφικών 2080ti. Μετά την εκπαίδευση των μοντέλων για TSP20, TSP50 και TSP100, αντίστοιχα, χρησιμοποιείται το προ-εκπαιδευμένο SE-GNN για να

καθοδηγηθεί το MCTS. Κατά τη δοκιμή, δημιουργούνται τυχαία 1000 περιστάσεις για τα παραπάνω τρία μοντέλα. Οι ρυθμίσεις παραμέτρων του GNN-MCTS που χρησιμοποιήθηκαν στις πειραματικές δοκιμές είναι ως εξής: όρισμα cpuct = 1,3 και πλάτος δέσμης = 1. Θέτονται τα rollouts = 800, 800 και 1200 για τα TSP20, TSP50 και TSP100 αντίστοιχα.

4.5.3 Αποτελέσματα

Εκτός από τους μη μαθησιακούς αλγορίθμους, συγκρίνεται η μέθοδός με εξαιρετικές μεθόδους βασισμένες σε βαθιά μάθηση που προκύπτουν επίσης από κάποιους greedy μηχανισμούς. Τα αποτελέσματα του δικτύου pointer [118] για τυχαίες περιστάσεις TSP20 και TSP50 λαμβάνονται για τοποθέτηση της ευκρίνειας. Επιπλέον, επεκτείνεται το έργο του Li κ.α. [106], όπου το GNN και η αναζήτηση δέντρου σε πλάτος συνδυάστηκαν για την επίλυση του MIS, στο TSP με κάποιες τροποποιήσεις. Επιλέγεται η καλύτερη περιοδεία που καταγράφηκε κατά τη διάρκεια της διαδικασίας αναζήτησης ως τελική περιοδεία. Δεδομένου ότι ο αλγόριθμος τους [106] δεν μπόρεσε να βρει καμία λύση που να είναι εφικτή όταν ο χρόνος εκτέλεσης ήταν ίδιος με αυτόν του GNN-MCTS, αυξάνεται ο χρόνος εκτέλεσής τους κατά 10 φορές και λαμβάνονται μερικές εφικτές λύσεις.

Αντί να αναφερθεί ο λόγος προσέγγισης c/c*, όπου c είναι η τιμή του στόχου της λύσης και c* είναι η καλύτερη γνωστή τιμή στόχου της περίπτωσης, χρησιμοποιείται η μέση τιμή βελτίωσης c-c*/c* = (c/c* - 1) όπως παρουσιάζεται στο [86]. Στον πίνακα 4.8 αναφέρονται οι διαφορές μεταξύ της λύσης κάθε προσέγγισης και της καλύτερης γνωστής λύσης για TSP20, TSP50 και TSP100. Στον πίνακα 4.9 αναφέρονται τα διαστήματα εμπιστοσύνης της μεθόδου μας σε διαφορετικά επίπεδα εμπιστοσύνης.

Τα αποτελέσματα της μεθόδου [108] και εκείνα άλλων μεθόδων βασισμένων σε βαθιά μάθηση δείχνουν ότι η προσέγγισή έχει εξαιρετική απόδοση έως και 100 κόμβους στις "τυχαίες" και "συμπυκνωμένες" περιστάσεις. Δεδομένου ότι η αναζήτηση δέντρου σε πλάτος δεν μπορεί να βρει μια εφικτή περιοδεία σε περιορισμένο χρόνο, η μέθοδος του Li κ.α. [106] αποτυγχάνει στα προβλήματα TSP μεγέθους $n \geq 50$.

	Random						Clustered					
Method	N=20		N=50		N=100		N=20		N=50		N=100	
	Obj.	Gap	Obj.	Gap	Obj.	Gap	Obj.	Gap	Obj.	Gap	Obj.	Gap
Concorde	3.92	0.0%	5.68	0.0%	7.73	0.0%	3.30	0.0%	3.38	0.0%	3.39	0.0%
Gourobi	3.92	0.0%	5.68	0.0%	7.73	0.0%	3.30	0.0%	3.38	0.0%	3.39	0.0%
Nearest Neighbor	4.57	16.50%	7.02	23.44%	9.63	24.58%	3.95	19.48 %	4.18	23.76%	4.23	25.02%
Nearest Insertion	4.40	12.24%	6.77	19.08%	9.48	22.68%	3.66	10.62 %	3.97	17.36%	4.08	20.46%
Random Insertion	4.08	4.12%	6.09	7.12%	8.45	9.24%	3.46	4.67%	3.65	7.93%	3.72	9.70%
Farthest Insertion	4.03	2.73%	5.98	5.29%	8.33	7.78%	3.40	2.87%	3.58	5.91%	3.64	7.55%
Vinyals κ.α.[118]	3.97	1.30%	6.41	18.58%	-		-		-		-	
Bello κ.α. [119]	3.99	1.80%	5.95	4.75%	8.26	6.82%	-		-		-	
Kool κ.α. [86]	3.93	0.36%	5.78	1.75%	8.08	4.57%	-		-		-	
Dai κ.α. [87]	4.03	2.76%	5.98	5.26%	8.33	7.76%	3.37	2.07%	3.58	6.01%	3.66	8.17%
Nowak κ.α. [121]	4.03	2.70%	-	-	-	-	-	-	-	-	-	-
Li κ.α. [106]	7.79	98.56%	-	-	-	-	-	-	-	-	-	-
GNN-MCTS [108]	**3.92**	**0.01%**	**5.92**	**0.20%**	**7.81**	**1.04%**	**3.31**	**0.03 %**	**3.39**	**0.44%**	**3.44**	**1.44%**
MCTS	6.37	62.45%	18.43	224.28%	40.27	420.83 %	9.26	50.69 %	9.26	173.99 %	14.0 0	313.33%
GNN-MCTS-t	3.97	1.25%	5.92	4.31%	8.42	8.95%	3.59	1.44%	3.59	6.09%	3.73	10.08%
GNN-MCTS-p	3.92	0.09%	5.77	1.47%	8.06	4.24%	3.44	0.14%	3.44	1.70%	3.54	4.39%
GNN-MCTS-v	3.92	0.08%	5.77	1.64%	8.06	4.24%	3.42	0.04%	3.42	1.13%	3.58	5.67%

GNN-MCTS$_{ave.}$	3.98	1.62%	6.09	7.21%	8.83	14.23%	3.63	1.25%	3.63	7.27%	3.87	14.11%

Πίνακας 4.8: GNN-MCTS έναντι βασικών γραμμών

Confidence level		90%	95%	99%
Random	n=20	3.922±0.046	3.922±0.054	3.922±0.071
	n=50	5.695±0.040	5.695±0.047	5.695±0.062
	n=100	7.812±0.038	7.812±0.046	7.812±0.060
Clustered	n=20	3.306±0.078	3.306±0.093	3.306±0.122
	n=50	3.394±0.051	3.394±0.061	3.394±0.080
	n=100	3.436±0.036	3.436±0.046	3.436±0.056

Πίνακας 4.9: Διαστήματα σε διαφορετικά επίπεδα εμπιστοσύνης

4.5.4 Γενίκευση Σε Μεγαλύτερα Προβλήματα

Για να γενικευτεί η μέθοδός, εκπαιδεύεται το SE-GNN σε προβλήματα μικρής κλίμακας και δοκιμάζεται το GNN-MCTS σε μεγαλύτερα προβλήματα, συμπεριλαμβανομένων TSP200, TSP300, TSP500 και TSP1000. Συγκρίνεται η εργασία των (link του paper) κυρίως με τις μεθόδους βασισμένες σε βαθιά μάθηση που προτάθηκαν από τους Kool κ.α. [86] και Dai κ.α. [87], οι οποίοι επιτύγχαναν την καλύτερη απόδοση γνωστή μέχρι στιγμής, στα πλαίσια Encoder-Decoder και Graph Embedding αντίστοιχα. Τα πειραματικά αποτελέσματα παρουσιάζονται στον πίνακα 4.10.

Πρώτα εκπαιδεύουν τη μέθοδό τους [108] και τις παραπάνω αναφερόμενες δύο μεθόδους βασισμένες σε βαθιά μάθηση στο TSP100 και στη συνέχεια τις δοκιμάζουν στα TSP200, TSP300 και TSP500. Όλες οι τρεις μέθοδοι μπορούν να λειτουργήσουν σε προβλήματα μεγάλης κλίμακας, αλλά η μέθοδός αυτή παρουσιάζει πολύ καλύτερα αποτελέσματα από τις μεθόδους που προτείνονται από τους Kool κ.α. [86] και Dai κ.α. [87]. Επιπλέον, εκπαιδεύουν τις τρεις μεθόδους στο TSP500 και στη συνέχεια τις δοκιμάζουν στο TSP1000. Είναι αναγκαίο να σημειωθεί ότι η μέθοδος που προτείνεται από τους Kool κ.α. [86] δεν συγκλίνει όταν εκπαιδεύεται στο TSP500, γεγονός το οποίο μπορεί να οφείλεται στο ότι το SE-GNN μπορεί να βρει περισσότερα κοινά γραφικά μοτίβα σε προβλήματα μεγάλης κλίμακας.

Αν και η μέθοδος που προτείνεται από τον Dai κ.α. [87] μπορούσε να λειτουργήσει στο TSP500 και στο TSP1000 όταν εκπαιδεύεται στο TSP500, η απόδοσή της ήταν χειρότερη από αυτή που πέτυχαν εκπαιδεύοντας την στο TSP100. Σε σύγκριση με τον Dai κ.α. [87], η μέθοδός αυτή μπορεί να παρουσιάσει πολύ καλύτερη απόδοση σε προβλήματα μεγάλης κλίμακας, συμπεριλαμβανομένων των TSP500 και TSP1000. Ο λόγος για αυτό μπορεί να είναι ότι η GNN-MCTS μπορεί να παρέχει πιο αξιόπιστες αποφάσεις από το 1-step Q-learning που χρησιμοποιείται στη μέθοδο του Dai κ.α. [87]. Αυτά τα αποτελέσματα δείχνουν ότι η μέθοδός τους [108] μπορεί να γενικευτεί καλύτερα σε μεγαλύτερα προβλήματα από άλλες μεθόδους βασισμένες σε βαθιά μάθηση, ακόμη και αν εκπαιδευτεί σε μικρότερα προβλήματα.

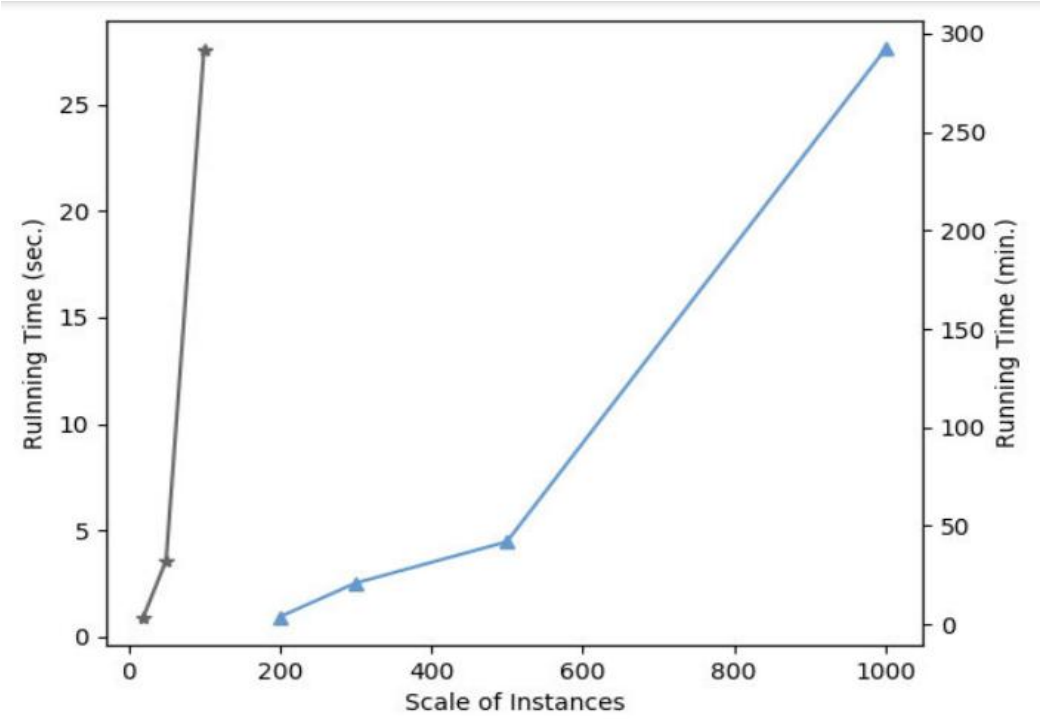
4.5.5 Χρόνοι Εκτέλεσης Και Χαρακτηριστικά Σύγκλισης

Οι χρόνοι εκτέλεσης είναι σημαντικοί, αλλά δύσκολο να συγκριθούν καθώς μπορεί να ποικίλλουν κατά δύο τάξεις μεγέθους λόγω των διαφορών στην υλοποίηση (Python ή C++) και το υλικό (CPU ή GPU). Δοκιμάζοντας τον αλγόριθμο τους [108], το Gurobi και άλλες μεθόδους βασισμένες σε βαθιά μάθηση σε μια μηχανή που διαθέτει 32 εικονικά συστήματα CPU (2 * Xeon(R) E5-2620) και 8 * 2080ti. Σε κάθε εποχή, δοκιμάζονται 32 παραδείγματα παράλληλα. Μετά από 10 εποχές, αναφέρουν τον χρόνο που χρειάστηκε για να λυθεί κάθε παράδειγμα δοκιμής Πίνακας 4.11 Η μέθοδός αυτή είναι πιο αργή από άλλες μεθόδους βασισμένες σε βαθιά μάθηση λόγω της εξερεύνησης που πραγματοποιείται. Ο κώδικάς είναι γραμμένος σε Python και παρατηρείται ότι η ταχύτητα της διαδικασίας MCTS μπορεί να αυξηθεί αν την κωδικοποιηθεί σε C++. Είναι αναγκαίο να σημειωθεί ότι το Gurobi επιλύει τα παραδείγματα του TSP1000 πολύ αργά και δεν μπορεί να δώσει καμία εφικτή λύση ακόμη και μετά από 10 ώρες εκτέλεσης.

Λόγω των GNN και MCTS, είναι δύσκολο να αναλυθεί η χρονική πολυπλοκότητα της GNN-MCTS, γι 'αυτό καταγράφεται ο χρόνος εκτέλεσης της GNN-MCTS σε παραδείγματα διαφορετικού μεγέθους Σχήμα 4.12. Συγκρίνοντας την κατηφόρα των δύο καμπύλων, μπορεί να ανακαλυφθεί ότι η χρονική πολυπλοκότητα της GNN-MCTS είναι χαμηλότερη σε παραδείγματα μεγαλύτερης κλίμακας.

Επιπλέον, αναλύεται ο χρόνος εκτέλεσης κάθε μέρους της GNN-MCTS. Πριν εξαχθεί μια περιήγηση, η μέθοδός χρειάζεται ένα μεγάλο αριθμό περιηγήσεων, όπου κάθε περιήγηση αποτελείται από τέσσερις φάσεις, οι οποίες είναι η επιλογή, η επέκταση, η προσομοίωση και η ανάδραση. Το κόστος χρόνου των τεσσάρων φάσεων σε κάθε περιήγηση παρουσιάζεται στον Πίνακα 4.12. Φαίνεται ότι η φάση προσομοίωσης καταλαμβάνει σχεδόν όλο τον χρόνο.

Ο MCTS έχει αποδειχθεί ότι συγκλίνει σε βέλτιστες λύσεις με υποθέσεις για άπειρη μνήμη και χρόνο υπολογισμού. Σε σύγκριση με το βασικό MCTS, η GNN-MCTS αντικαθιστά την ομοιόμορφη τυχαία στρατηγική επιλογής με μια μάθησης στρατηγικής αξίας στη φάση προσομοίωσης, δηλαδή ενσωματώνοντας εξειδικευμένη γνώση στο MCTS, και η τελευταία προσέγγιση συνήθως επιτρέπει γρηγορότερη σύγκλιση στον τιμολόγηση αντί της απλότητας και της γενικότητας.



Σχήμα 4.12: Ο χρόνος εκτέλεσης του GNN-MCTS σε σχέση με τον αριθμό δεδομένων [108]

$Test/Train$	n=200		n=300		n=500		n=1000	
	Obj.	Gap	Obj.	Gap	Obj.	Gap	Obj.	Gap
Concorde	10.71	0.00%	12.99	0.00%	16.47	0.00%	23.07	0.00%
Gurobi	10.71	0.00%	12.99	0.00%	16.47	0.00%	23.07	0.00%
Kool κ.α.$_{n=100}$	11.59	8.19%	14.59	12.32%	19.83	20.40%	-	-
Dai κ.α. $_{n=100}$	11.90	11.11%	14.51	11.70%	18.42	11.84%	-	-
GNN-MCTS$_{n=100}$	**10.91**	**1.91%**	**13.37**	**2.99%**	**17.19**	**4.37%**	-	-
Kool κ.α. $_{n=500}$	-	-	-	-	-	-	-	-
Dai κ.α. $_{n=500}$	-	-	-	-	18.62	13.05%	26.10	13.14%
GNN-MCTS$_{n=500}$	-	-	-	-	**17.02**	**3.33%**	**24.10**	**4.48%**

Πίνακας 10: GNN-MCTS και άλλες μέθοδοι σε τυχαίες περιπτώσεις

	TSP20	TSP50	TSP100	TSP200	TSP300	TSP500	TSP1000
Kool κ.α.	0.007s	0.018s	0.043s	0.128s	0.243s	0.608s	2.328s
Dai κ.α.	0.062s	0.107s	0.149s	0.270s	0.390s	0.588s	-
Gurobi	0.017s	0.2s	1.9s	0.4min	3.7min	19.4min	-
GNN-MCTS	0.93s	3.58s	27.62s	3.9min	21min	42min	292min

Πίνακας 4.11: Χρόνοι λειτουργίας διαφορετικών μεθόδων

	selection	expansion	simulation	Back-propagation
TSP20	2.91%	3.31%	93.76%	0.02%

TSP50	0.31%	0.53%	99.16%	0.00%
TSP100	0.20%	0.26%	99.54%	0.00%

Πίνακας 4.12: Χρόνος εκτέλεσης τεσσάρων φάσεων σε κάθε κυκλοφορία

4.5.6 Μελέτη αφαιρέσεων

1. ΜΈΘΟΔΟΙ ΜΈΣΟΥ ΈΝΑΝΤΙ ΚΑΛΎΤΕΡΗΣ

Αναλύονται οι επιπτώσεις διαφορετικών στρατηγικών που χρησιμοποιούνται στη διαδικασία GNN-MCTS. Η σύγκριση των δύο στρατηγικών είναι η εξής:

- Καλύτερη (Best): Αντίθετα από το AlphaGo, παρακολουθείται η καλύτερη τιμή δράσης που βρίσκεται στο υπό δέντρο κάθε κόμβου για τον προσδιορισμό της τιμής εκμετάλλευσής του. Στο τέλος πολλών προσομοιώσεων, επιλέγεται ο κόμβος με την καλύτερη (μεγαλύτερη) τιμή δράσης ως την επόμενη κίνηση στην αρχική θέση.
- Μέσος (Average): Όπως και η στρατηγική που χρησιμοποιείται στο AlphaGo, που είναι κοινή σε ένα παιχνίδι δύο παικτών, παρακολουθείται η μέση τιμή δράσης που βρέθηκε στο υπό δέντρο κάθε κόμβου ως την τιμή εκμετάλλευσής του. Αντί να επιλεχθεί ο κόμβος με την καλύτερη (μεγαλύτερη) τιμή δράσης, επιλέγεται ο πιο επισκεπτόμενος κόμβος ως την επόμενη κίνηση στην αρχική θέση.

Χρησιμοποιείται το GNN-MCTS για να αναπαραστήσει την αναζήτηση δέντρου χρησιμοποιώντας τη στρατηγική "Καλύτερη" και το GNN-MCTSave για τη στρατηγική "Μέσος". Ο Πίνακας 4.8 δείχνει τη διαφορά ("Gap") μεταξύ των λύσεων της προσέγγισής μας με τις δύο στρατηγικές και της καλύτερης γνωστής λύσης για τα προβλήματα TSP20, TSP50 και TSP100. Από τα αποτελέσματα του GNN-MCTS και του GNN-MCTSave, παρατηρείται ότι η απόδοση της μεθόδου επηρεάζεται σοβαρά όταν χρησιμοποιείται η στρατηγική "Μέσος". Φαίνεται ότι η απόδοση του GNN-MCTSave μειώνεται επειδή η μέση τιμή δράσης υπό έναν κόμβος δεν είναι μια καλή εκτίμηση αν η βέλτιστη τιμή κάτω από τον κόμβο περιβάλλεται από μερικές κατώτερες τιμές.

2. ΑΝΆΛΥΣΗ ΣΥΝΕΙΣΦΟΡΆΣ ΤΩΝ ΣΥΝΤΟΜΕΎΣΕΩΝ ΤΗΣ GNN-MCTS

Πραγματοποιείται ένας ελεγχόμενος πειραματισμός στο σύνολο δοκιμών για να αναλύσουμε τη συνεισφορά κάθε συνιστώσας στην παρουσιαζόμενη προσέγγιση.

- Αντί να χρησιμοποιηθεί η MCTS, χρησιμοποιείται το SE-GNN για να προκύψουν περιοδείες απευθείας, δηλαδή να επιλέγεται ο κόμβος με τη μεγαλύτερη προτεραιότητα πιθανότητας σε κάθε βήμα. Αυτή την έκδοση αποκαλείται GNN-MCTS-t (χωρίς αναζήτηση δέντρου).
- Αντικαθίσταται η συνάρτηση τιμής h(s) Αλγόριθμο 4.5 με την τυχαία συνάρτηση προσομοίωσης για την αξιολόγηση της κατάστασης κατά τη διάρκεια της διαδικασίας MCTS. Αυτή την έκδοση την αποκαλείται GNN-MCTS-v (χωρίς συνάρτηση τιμής).
- Αφαιρείται η έξοδος του SE-GNN από την εικόνα και ξεκινά η αρχικοποίηση της προτεραιότητα πιθανότητας σε 1 για νεοσύστατους κόμβους. Αυτή την έκδοση την αποκαλείται GNN-MCTS-p (χωρίς προτεραιότητα που παρέχεται από το SE-GNN).
- Μια καθαρή MCTS, η οποία καταργεί την προηγούμενη προτεραιότητα του SE-GNN και αντικαθιστά τη συνάρτηση τιμής h(s) με την τυχαία συνάρτηση προσομοίωσης,

περιλαμβάνεται για σύγκριση. Αυτή την έκδοση την αποκαλείται MCTS (ίση με GNN-MCTS-p,v).

Ο Πίνακας 4.8 δείχνει τη διαφορά ("Gap") μεταξύ της λύσης κάθε προσέγγισης και της καλύτερης γνωστής λύσης για διαφορετικά προβλήματα TSP. Συνολικά, η απόδοση της GNN-MCTS-t μειώνεται πολύ από την απόδοση της GNN-MCTS, πράγμα που δείχνει ότι η αναπτυγμένη MCTS μπορεί αποτελεσματικά να αποτρέψει τον αλγόριθμο από το να πέσει σε κάποιο τοπικό βέλτιστο και παίζει σημαντικό ρόλο στην ενίσχυση της απόδοσης της μεθόδου των [108].

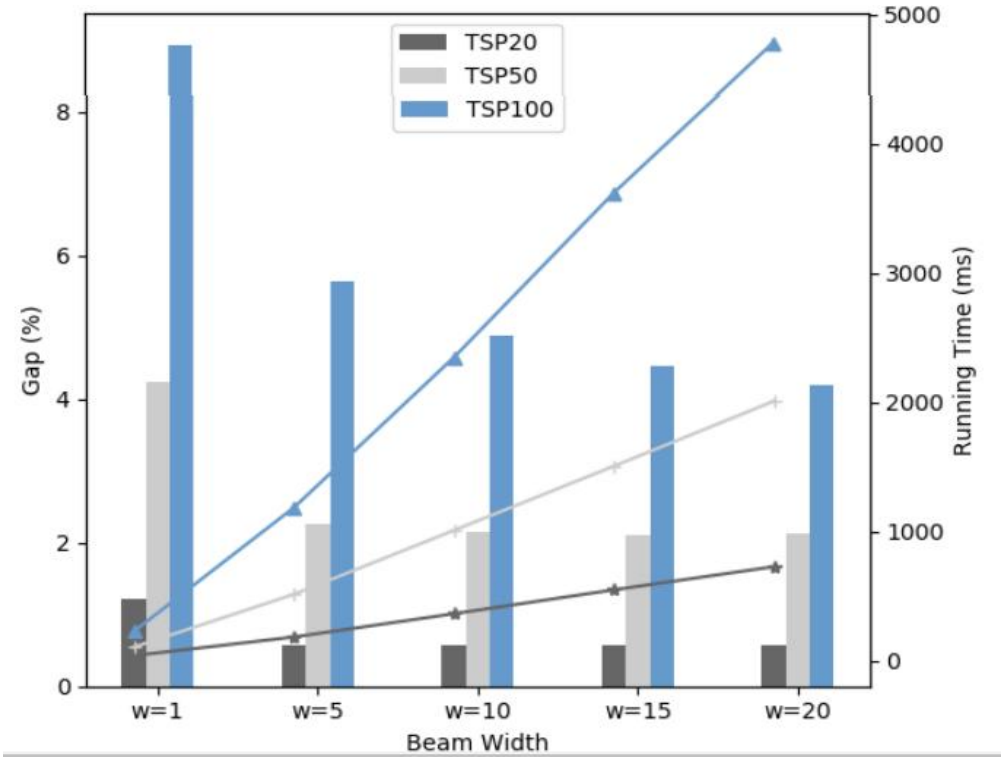
Πραγματοποιείται περαιτέρω ανάλυση των λόγων για τους οποίους βελτιώνεται ο αλγόριθμος. Καταρχάς, συγκρίνοντας τις αποδόσεις των GNN-MCTS-p και GNN-MCTS, μπορούμε να συμπεράνουμε ότι η προηγούμενη προτεραιότητα του SE-GNN μπορεί να βοηθήσει την MCTS να μειώσει αποτελεσματικά το χώρο αναζήτησης, ώστε η MCTS να μπορεί να εκχωρήσει περισσότερους υπολογιστικούς πόρους στις καταστάσεις με υψηλή τιμή δράσης. Δεύτερον, τα αποτελέσματα που έχουν ληφθεί από τα GNN-MCTS-v και GNN-MCTS δείχνουν ότι μια κατάλληλη συνάρτηση τιμής h(s) μπορεί να εκτιμήσει καλά το μήκος της περιήγησης από την κατάσταση του φύλλου στην τελική κατάσταση και να επιτρέπει στην MCTS να αποδίδει καλύτερα από το να χρησιμοποιεί μια απλή συνάρτηση τυχαίας προσομοίωσης. Τέλος, συγκρίνοντας τις αποδόσεις των GNN-MCTS-p, GNN-MCTS-v και άλλων μεθόδων που βασίζονται στη μάθηση, φαίνεται ότι όταν αφαιρείται μια συνιστώσα από την GNN-MCTS, δηλαδή προτεραιότητα του SE-GNN ή συνάρτηση τιμής, η μέθοδός των [108] μπορεί ακόμα να προκύψει μια λογική περιήγηση και να αποδίδει καλύτερα από άλλες μεθόδους που βασίζονται στη μάθηση. Αυτό σημαίνει ότι, όταν είναι δύσκολο να σχεδιαστούν κατάλληλες συνιστώσες για άλλα παρόμοια προβλήματα, η μέθοδός των [108] μπορεί να επιτύχει ικανοποιητικά αποτελέσματα ακόμα και με μόνο μία από αυτές.

Είναι προφανές ότι η μελέτη αυτή βοηθά στο να κατανοηθεί καλύτερα η συνεισφορά των διαφορετικών στρατηγικών και συνιστωσών στη μέθοδό για το πρόβλημα TSP.

3. ΕΠΙΠΛΕΟΝ ΑΝΑΛΥΣΗ ΤΗΣ ΣΥΝΑΡΤΗΣΗΣ ΤΙΜΗΣ

Πραγματοποιούνται πειράματα για να εξερευνήσουν τις επιδράσεις διαφορετικών πλατών δέσμης (beam widths) στην απόδοση της συνάρτησης τιμής. Δεδομένου ότι το πλάτος δέσμης επηρεάζει κυρίως την απόδοση της συνάρτησης τιμής, χρησιμοποιείται το αποτέλεσμα αυτής της συνάρτησης ως μέτρο και καθορίζεται το 'Κενό' (Gap), όπως αναφέρεται στον πίνακα 4.8. Συγκεκριμένα, θέτουν [108] το πλάτος δέσμης στο 1, 5, 10, 15, 20 και δοκιμάζουν την απόδοση της συνάρτησης τιμής σε τυχαίες περιπτώσεις περιηγήσεων που περιλαμβάνουν TSP20, TSP50 και TSP100. Επίσης, καταμετρούν τον χρόνο εκτέλεσης της συνάρτησης τιμής κατά τη διάρκεια της ρύθμισης διαφορετικών πλατών δέσμης. Τα πειραματικά αποτελέσματα παρουσιάζονται στο Σχήμα 4.13.

Όταν το πλάτος δέσμης αυξάνεται από 1 σε 5, η απόδοση της συνάρτησης τιμής βελτιώνεται σημαντικά. Ωστόσο, καθώς το πλάτος δέσμης συνεχίζει να αυξάνεται, η βελτίωση στην απόδοση της συνάρτησης τιμής γίνεται λιγότερο αντιληπτή. Επιπλέον, ο χρόνος εκτέλεσης της συνάρτησης τιμής αυξάνεται περίπου 5 φορές όταν το πλάτος δέσμης αυξάνεται από 1 σε 5. Τα παραπάνω αποτελέσματα δείχνουν ότι πρέπει να υπάρχει ένας συμβιβασμός μεταξύ της απόδοσης και του χρόνου εκτέλεσης της συνάρτησης τιμής.



Σχήμα 4.13: Η απόδοση της συνάρτησης τιμής με διαφορετική δέσμη πλάτους [108]

4. ΣΥΓΚΡΙΣΗ ΜΕ ΆΛΛΑ ΝΕΥΡΩΝΙΚΆ ΔΊΚΤΥΑ ΠΥΡΉΝΑ ΣΥΝΔΕΣΗΣ

Σε σύγκριση με το βασικό GNN [109] και το GCN [122][123], το SE-GNN ενσωματώνει πληροφορίες ακμών για τον υπολογισμό νέας χαρακτηριστικής κόμβου, οπότε θα πρέπει να εξάγει περισσότερες πληροφορίες και να αποδίδει καλύτερα από το βασικό GNN και το GCN. Για να υποστηρίξουν αυτό τον ισχυρισμό, συγκρίνονται οι αποδόσεις του βασικού GNN, GCN και SE-GNN σε τυχαίες περιπτώσεις, συμπεριλαμβανομένων των TSP20, TSP50 και TSP100. Προκύπτουν περιηγήσεις απευθείας χρησιμοποιώντας το νευρωνικό δίκτυο, δηλαδή επιλέγοντας τον κόμβο με τη μεγαλύτερη προηγούμενη πιθανότητα σε κάθε βήμα. Η απόδοση των τριών GNN αναφέρεται στον πίνακα 4.13.

Οι αποδόσεις των GNN, GCN και SE-GNN δείχνουν ότι οι χαρακτηριστικά των ακμών είναι σημαντικά για το TSP. Συμφωνείται ότι, στην αρχή, το νευρωνικό δίκτυο θα έπρεπε να έχει μάθει τις πληροφορίες απόστασης από τις συντεταγμένες των κόμβων. Αυτό είναι ένα απλό πράγμα για τους ανθρώπους, αλλά τα εμπειρικά αποτελέσματά των [108] δείχνουν ότι είναι δύσκολο για το νευρωνικό δίκτυο να μάθει τέτοιες πληροφορίες απόστασης.

Επιπλέον, πραγματοποιείται ένας ελεγχόμενος πειραματισμού για να αναλυθεί γιατί το SE-GNN επιτυγχάνει καλύτερα από άλλα GNN. Σε σύγκριση με τα Crystal Graph Convolutional Neural Networks (CGCNN) [113] και τα Message Passing Neural Networks (MPNN) [62], η μεγαλύτερη βελτίωση του SE-GNN συμβαίνει όταν ο single perceptron αντικαθίσταται με ένα multilayer perceptron (MLP). Για να επαληθεύσουν αυτό το σημείο, δοκιμάζουν την απόδοση του SE-GNNσ, το οποίο αντικαθιστά το MLP στην εξίσωση (4.18) με ένα single perceptron (σ).

Όπως φαίνεται στον πίνακα 4.13, οι αποδόσεις του SE-GNNσ και του CGCNN είναι συγκρίσιμες. Ωστόσο, όταν το single perceptron αντικαθίσταται με MLP, η απόδοση του SE-GNN βελτιώνεται

σημαντικά, ειδικά στη μετρική 'Κενό' (Gap). Αυτά τα αποτελέσματα δείχνουν ότι το MLP, το οποίο χρησιμοποιείται ως μη γραμμική συνάρτηση αντιστοίχισης, παίζει σημαντικό ρόλο στην επίτευξη καλύτερης απόδοσης.

Εκτός από τις διαφορετικές συναρτήσεις αντιστοίχισης, ο μηχανισμός συνένωσης χαρακτηριστικών των ακμών του SE-GNN και άλλων GNN είναι επίσης διαφορετικός. Το MPNN θεωρεί τα χαρακτηριστικά των ακμών ως ένα είδος μετρικής για να μετρήσει τη χρησιμότητα των γειτόνων ενός κόμβου. Ωστόσο, το SE-GNN και το CGCNN θεωρούν τα χαρακτηριστικά των ακμών και των κόμβων εξίσου σημαντικά κατά τη διαδικασία ενημέρωσης. Τα αποτελέσματα του SE-GNN, του CGCNN και του MPNN δείχνουν ότι ο μηχανισμός ολοκλήρωσης που χρησιμοποιείται στο SE-GNN και το CGCNN είναι πιο κατάλληλος για το TSP.

5. ΔΙΑΦΟΡΕΤΙΚΈΣ ΡΥΘΜΊΣΕΙΣ ΤΟΥ SE-GNN

Το SE-GNN έχει T = 3 επίπεδα ενημέρωσης, τα οποία είναι αρκετά βαθιά για έναν κόμβο να συγκεντρώσει πληροφορίες που σχετίζονται με τους γειτονικούς κόμβους του. Δεδομένου ότι το γράφημα εισόδου αποτελείται από κόμβους με ετικέτες 9-διάστατων χαρακτηριστικών, το πλάτος του πρώτου επιπέδου είναι H0 = 9. Το πλάτος των άλλων επιπέδων είναι ίδιο: Ht = 128 για t = 1, 2.

Το SE-GNN έχει μια βαθιά αρχιτεκτονική που αποτελείται από αρκετά επίπεδα ενημέρωσης. Έτσι, καθώς το μοντέλο γίνεται πιο βαθύ με περισσότερα επίπεδα, περισσότερες πληροφορίες μπορούν να συγκεντρωθούν από τους κόμβους. Εκπαιδεύεται το SE-GNN με διαφορετικό αριθμό επιπέδων σε τυχαίες περιπτώσεις περιηγήσεων που περιλαμβάνουν TSP20, TSP50 και TSP100. Χρησιμοποιείται απευθείας η προηγούμενη πιθανότητα για να παραχθεί μια ακολουθία περιηγήσεων, δηλαδή να επιλέγεται ο κόμβος με τη μεγαλύτερη προηγούμενη πιθανότητα σε κάθε βήμα.

Τα αποτελέσματα στον πίνακα 4.14 δείχνουν ότι η απόδοση του SE-GNN βελτιώνεται καθώς αυξάνεται ο αριθμός των επιπέδων του δικτύου. Ωστόσο, τρία επίπεδα είναι αρκετά για το SE-GNN να εξάγει χαρακτηριστικά για το TSP στα πειράματά. Επομένως, το τριώροφο SE-GNN χρησιμοποιείται ως προεπιλογή.

	n=20		n=50		n=100	
	Gap	Acc.	Gap	Acc.	Gap	Acc.
GNN	4.28%	87.68%	15.15%	82.07%	25.93%	78.47%
GCN	3.88%	88.28%	15.33%	83.57%	25.69%	79.97%
GEN	5.24%	86.46%	14.00%	85.23%	16.40%	84.27%
MPNN	5.15%	86.86%	13.06%	86.00%	16.33%	86.22%
CGCNN	3.04%	90.80%	8.80%	90.35%	12.51%	90.30%
SE-GNN	**1.25%**	**93.16%**	**4.31%**	**92.48%**	**8.95%**	**91.88%**
$SE\text{-}GNN_{\sigma}$	2.04%		7.12%	90.42%	11.79%	89.80%

Πίνακας 4.13: Απόδοση διαφορετικών νευρωνικών δικτύων σε τυχαίες περιπτώσεις

	T=1		T=2		T=3		T=4	
	Gap	Acc.	Gap	Acc.	Gap	Acc.	Gap	Acc.
TSP20	6.01%	88.46%	1.90%	91.98%	1.25%	93.16%	1.02%	93.81%

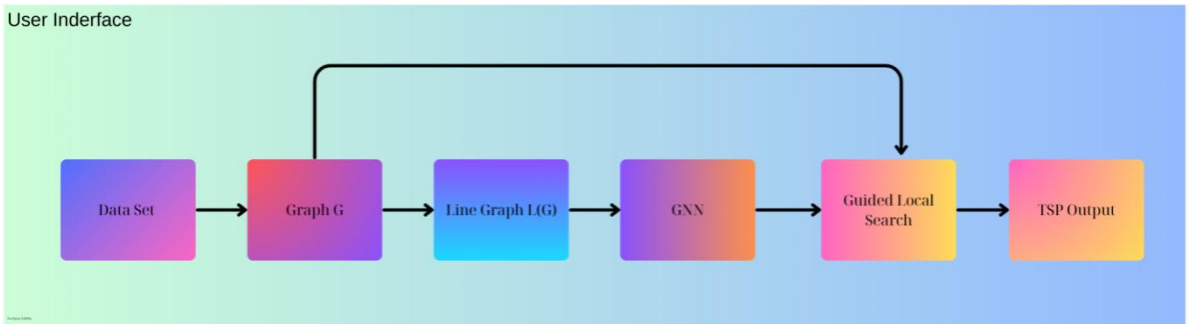
TSP50	11.64%	87.73%	6.60%	91.02%	4.31%	92.48%	3.87%	93.29%
TSP100	15.32%	87.76%	11.16%	90.74%	8.95%	91.88%	7.24%	92.79%

Πίνακας 4.14: Επίδραση του αριθμού των επιπέδων σε τυχαίες περιπτώσεις

Κεφάλαιο 5 : Ο Κώδικας μου

5.1 Εισαγωγή

Στο παρακάτω κεφάλαιο θα αναφερθούμε στη δική μου απόπειρα επίλυσης του προβλήματος χρησιμοποιώντας μια από τις παραπάνω επιλύσεις ως οδηγό. Πρώτα θα γίνει μια ανάλυση της επιλογής αυτής της μεθόδου, στη συνέχεια θα κάνουμε μια ανάλυση του κώδικα που επιλύει το πρόβλημα και τέλος θα αναφερθούμε στα επιπρόσθετα που έβαλα στο κώδικα μου για γραφικό περιβάλλον κ.α.. Όπως αναφερθήκαμε στο κεφάλαιο **4.1**, υπάρχουν τριών ειδών ως τρόποι επίλυσης του προβλήματος σαν επιλογή διάλεξα την δεύτερη (το μοντέλο δίνει πληροφορίες σε ένα OR αλγόριθμο), πιο συγκεκριμένα επέλεξα το **4.4** δηλαδή ένα GNN με καθοδηγούμενη τοπική αναζήτηση για την επίλυση του πλανόδιου πωλητή (TSP). Μελετώντας κάθε μια από τις προσέγγισης αυτή μου φάνηκε η καλύτερη από άποψη ταχύτητας και κατανάλωσης πόρων. Στη παρακάτω ενότητα θα αναφερθούμε στα εργαλεία που χρησιμοποίησα.



Εικόνα 5.1: Στη εικόνα που βλέπουμε είναι η δομή του μοντέλου

5.2 Γλώσσα και εργαλεία προγραμματισμού

Ξεκινώντας θα ήθελα να αναφέρω την γλώσσα προγραμματισμού που χρησιμοποίησα και μερικά πράγματα για αυτήν. Στη συνέχεια θα ήθελα να αναφερθώ σε ποιο πρόγραμμα τρέχω τον κωδικά μου και γιατί.

5.2.1 Python

Η Python είναι μια ισχυρή γλώσσα προγραμματισμού που συνδυάζει την απλότητα με την αποδοτικότητα. Διαθέτει μια αποδοτική και υψηλού επιπέδου δομή δεδομένων, ενώ ταυτόχρονα παρέχει μια απλή, αλλά εξίσου αποδοτική προσέγγιση στον αντικειμενοστραφή προγραμματισμό. Αυτό καθιστά την Python εξαιρετικά ευέλικτη και εύχρηστη, ειδικά για προγραμματιστές που επιθυμούν να αναπτύξουν κώδικα γρήγορα και με ακρίβεια.

Η Python διακρίνεται για την ευανάγνωστη και καθαρή της σύνταξη, η οποία σε συνδυασμό με τις δυνατότητες δυναμικής τυποποίησης (dynamic typing), δημιουργεί μια ιδανική γλώσσα προγραμματισμού τόσο για scripting όσο και για ταχεία ανάπτυξη εφαρμογών. Αυτή η ευκολία χρήσης την έχει καταστήσει δημοφιλή σε πληθώρα τομέων, από την ανάπτυξη web εφαρμογών έως την επιστημονική ανάλυση δεδομένων και την τεχνητή νοημοσύνη.

Ένα από τα μεγαλύτερα πλεονεκτήματα της Python είναι το τεράστιο οικοσύστημα βιβλιοθηκών και εργαλείων που τη συνοδεύει. Από βιβλιοθήκες για αριθμητική υπολογιστική, όπως η NumPy και η SciPy, μέχρι εργαλεία για ανάλυση δεδομένων όπως η pandas και για δημιουργία οπτικών δεδομένων όπως η Matplotlib και η Seaborn, οι δυνατότητες της Python είναι σχεδόν απεριόριστες. Επιπλέον, η γλώσσα προσφέρει εξειδικευμένες βιβλιοθήκες για τεχνητή νοημοσύνη και μηχανική μάθηση, όπως η TensorFlow, η Keras, και η PyTorch, καθιστώντας την κορυφαία επιλογή για ανάπτυξη εφαρμογών σε αυτούς τους τομείς.

Η Python είναι επίσης γνωστή για τη μεγάλη κοινότητα υποστήριξης που διαθέτει. Οι χρήστες μπορούν να βρουν πληθώρα online οδηγών, τεκμηρίωσης και forum που προσφέρουν βοήθεια και παραδείγματα κώδικα. Αυτό διευκολύνει σημαντικά τους νέους προγραμματιστές, επιτρέποντάς τους να μάθουν γρήγορα και να προχωρήσουν σε πιο σύνθετες εφαρμογές χωρίς δυσκολία.

Ένας ακόμη λόγος που η Python ξεχωρίζει είναι η δυνατότητά της να προσαρμόζεται σε διάφορα λειτουργικά συστήματα και πλατφόρμες, γεγονός που επιτρέπει στους προγραμματιστές να αναπτύσσουν κώδικα για πολλές πλατφόρμες χωρίς να χρειάζεται να μάθουν νέες γλώσσες ή να κάνουν σημαντικές αλλαγές στον κώδικα τους. Αυτό την καθιστά εξαιρετικά χρήσιμη για έργα που απαιτούν συμβατότητα με διαφορετικά συστήματα.

Επέλεξα την Python όχι μόνο γιατί είναι η κατάλληλη γλώσσα προγραμματισμού για νευρωνικά δίκτυα και μηχανική μάθηση, αλλά και γιατί προσφέρει βιβλιοθήκες όπως η folium για δημιουργία διαδραστικών χαρτών και η PyQt5 για ανάπτυξη γραφικού περιβάλλοντος εφαρμογών (GUI). Αυτές οι βιβλιοθήκες δίνουν τη δυνατότητα στον προγραμματιστή να δημιουργήσει πολύπλοκα και καλαίσθητα γραφικά περιβάλλοντα εργασίας, επιτρέποντας έτσι την ανάπτυξη ολοκληρωμένων εφαρμογών με προχωρημένες δυνατότητες αλληλεπίδρασης. Οι βιβλιοθήκες που χρησιμοποίησα θα αναλυθούν παρακάτω.

5.2.2 Jupyter Notebook

Το Jupyter Notebook είναι ένα διαδραστικό περιβάλλον που έχει ως βάση το web για σημειωματάρια, κώδικα και δεδομένα. Η ευέλικτη διεπαφή του επιτρέπει στους χρήστες να διαμορφώνουν και να τακτοποιούν ροές εργασίας για τον λόγο αυτό είναι εξαιρετικά χρήσιμο εργαλείο σε επιστήμες όπως data science, scientific computing, computational journalism, και την μηχανική μάθηση.

Το Jupyter Notebook ήταν η κατάλληλη επιλογή όχι μόνο για την ευκολία του στο UI και την δυνατότητα να διαχωρίζεις των κώδικα σε διάφορα μέρη (δηλαδή πιο άμεσα σχόλια και πιο κατανοητά για τον χρήστη όπως πρόσθεση εικόνας και text), αλλά και για την δυνατότητα εμφανίσεων από βιβλιοθήκες όπως η folium. Διότι η εμφάνιση χάρτη σε ένα πρόγραμμα που δεν τρέχει στο web είναι δύσκολο εκτελεστή.

5.3 Το Πρόγραμμα

Παρακάτω θα γίνει ανάλυση το προγράμματος μου, δηλαδή θα αναλύσουμε κατά βήμα τον κώδικα μου και παράλληλα τον οδηγό που είχα από τους [88] και της αλλαγές που χρειάστηκε να κάνω για να εκπληρώσω τα ζητούμενα της πτυχιακής (δηλαδή τα δεδομένα να είναι η γεωγραφικές θέσης των πόλεων της Ελλάδος).

Ξεκινώντας το βασικότερο ζητούμενο στο data science και στη μηχανική μάθηση είναι η καθαρότητα τον δεδομένων. Τα δεδομένα που θα πρέπει να περαστούν από ένα νευρωνικό δίκτυο θα πρέπει να είναι καλά δομημένα και σε συγκεκριμένη μορφή (με άλλα λόγια να μην είναι ατελή δεδομένα με καινά και δηλωμένα σε μορφή που δεν μπορεί να διαβάσει το νευρωνικό δίκτυο). Οπότε επέλεξα να κατεβάσω βια βάση δεδομένων με τις πόλεις της Ελλάδας από την **πηγή** και στη συνέχεια επεξεργαστικά τα δεδομένα σε Excel για να τα προσαρμόσω ανάλογα με τα ζητούμενα μου.

Διέγραφα στήλες όπως πληθυσμό και πρωτεύουσες από την βάση δεδομένων και κράτησα τα ονόματα των πόλεων και τις συντεταγμένες τους δηλαδή το latitude και το longitude. Έπειτα με την βιβλιοθήκη της **Python pandas** χρησιμοποίησα την εντολή **read_excel** για να διαβάσω το αρχείο με τα δεδομένα που είχα αποθήκευση μέσα στο Jupyter. Σύμφωνα με τα ζητούμενα τον [88] τα δεδομένα που θα χρειαστούμε για το νευρωνικό δίκτυο είναι οι χιλιομετρικές αποστάσεις μεταξύ των πόλεων τον οποίον θα τα βρούμε με την βοήθεια των latitude και longitude, τα υπόλοιπα δεδομένα χρειάζονται στο πρόγραμμα για το κομμάτι του UI.

Έτσι στη συνέχεια έφτιαξα ένα αρχείο με όνομα **ApoKomb1** που διαθέτει μέσα 4 μεθόδους που μας επιστρέφουν σημαντικά στοιχεία. Η πρώτη με όνομα **sindet** επιστρέφει τις συντεταγμένες σε έναν πίνακα [2, n] όπου στη πρώτη στήλη επιστρέφει το lat και στη δεύτερη το lng. Η δεύτερη με όνομα apostasis με την βοήθεια της βιβλιοθήκης haversine επιστρέφει έναν πίνακα με της Ευκλείδειες αποστάσεις από κάθε μια συντεταγμένη με την άλλη. Στη Τρίτη μέθοδο με όνομα ***apostasis_test( )*** επιστρέφει της ίδιες αποστάσεις άλλα σε έναν πίνακα [n,n]. Τέλος στη μέθοδο **edge_index** φτιάχνω έναν πίνακα που διαθέτει όλες τις ακμές ανάμεσα στους κόμβους, το γράφημα θα πρέπει να είναι πλήρες συνδεδεμένο.

5.3.1 Προετοιμασία δεδομένων για γράφημα

Στη συνέχεια θα αναφερθούμε στις κλάσεις που βρίσκονται στο αρχείο **main_p,** την ***basiko()*** και την ***graph_to_LineGraph()***. Στη κλάση ***basiko()*** έχουμε τρεις μεθόδους, η πρώτη με όνομα ***main_program()*** καλείτε από το γραφικό περιβάλλον, πιο συγκεκριμένα όταν ο χρήστης πατήσει κάποιο κουμπί από τα τέσσερα που δημιουργούνται με ονόματα 5, 10, 20 και 50 με την βοήθεια της βιβλιοθήκης **PyQt5** πιο συγκεκριμένα της βιβλιοθήκης **PyQt5.QtWidgets** φτιαχνωντε 4 κουμπιά που έχουν onclick event και ανάλογα πιο από τα 4 το πρόγραμμα θα πάρει από τη βάση δεδομένων ένα sample ανάλογο με τον αριθμό από τις πόλεις και τις συντεταγμένες τους (περισσότερα για το γραφικό περιβάλλον θα αναφερθούν σε πιο κάτω κεφάλαιο).

Στη συνέχεια έχουμε την μέθοδο *city_names_and_coords* στην οποία παίρνουμε το sample των πόλεων και χωρίζουμε τις στήλες τον δεδομένων για τις ανάλογες χρήσεις τους. Παίρνοντας τα lat και

lng τα στέλνουμε στη μέθοδο sindet που αναφέραμε στο προηγούμενο κεφάλαιο καλούμε επίσης το *edge_index* για να έχουμε τις ακμές και στη συνέχει την μέθοδο *apostasi*. Έχοντας ολοκληρώσει αυτές τις διαδικασίες, αυτή η μέθοδος καλεί άλλη μία που ανανεώνει τον χάρτη (περισσότερα για τον χάρτη σε επόμενο υποκεφάλαιο) και τέλος επιστρέφει τις καινούργιες τιμές. Τέλος στη κλάση υπάρχει η μέθοδος *map_path* στην οποία αφότου ολοκληρώσουμε και επιλύσουμε το Travel Salesman Problem καλείτε για να βάλει τις γραμμές δηλαδή της ακμές στο χάρτη.

Στη κλάση ***graph_to_LineGraph()*** αφού έχουμε παραλάβει τις τιμές αποστάσεων και τις ακμές σχεδιάζουμε το γράφημα στην ουσία με την βοήθεια της βιβλιοθήκη **networkx** φτιάχνουμε ένα μη κατευθυνόμενο γράφημα που είναι πλήρες συνδεδεμένο και έχει ***edge_features*** τις χιλιομετρικές αποστάσεις. Στη συνέχεια αυτό το γράφημα το μετατρέπω όπως αναφέρονται και οι [88] σε ένα γραμμικό γράφημα έτσι ώστε για κάθε ακμή που έχει το γράφημα να έχει έναν ανάλογο κόμβο το γραμμικό γράφημα και για κάθε δύο ακμές που συνδέονταν στον ίδιο κόμβο στο γράφημα τώρα υπάρχει μια ακμή μεταξύ των αντίστοιχων κόμβων τους στο γραμμικό γράφημα. Με αυτή την αλλαγή επιτρέπεται στο νευρωνικό δίκτυο να παίρνει δεδομένα με στα οποία διαβάζει *node_features* και όχι *edge_features* έτσι γίνεται πιο εύκολη η επεξεργασία των αποτελεσμάτων. Να αναφέρουμε σε αυτό το σημείο ότι με την βιβλιοθήκη **torch** μετατρέπουμε τα *node_features* σε τιμές *tensor* για να μπορούν να διαβαστούν στο νευρωνικό δίκτυο.

5.3.2 Νευρωνικό Δίκτυο

Σε αυτό το σημείο θα αναλύσουμε το νευρωνικό δίκτυο το οποίο μας επιστρέφει την τιμή Regret που θα χρησιμοποιηθεί μετά για την επίλυση του TSP. Ξεκινώντας θα ήθελα να αναφερθώ στη δομή του, το νευρωνικό δίκτυο διαθέτει ένα στρώμα ενσωμάτωσης (embedding) σαν είσοδο στο νευρωνικό δίκτυο στη συνέχεια πολλά GNN στρώματα και ένα στρώμα εξόδου. Για να επιτευχθεί αυτό δημιουργούμε κλάσεις για το κάθε κομμάτι του νευρωνικού δικτύου και στη συνέχεια δημιουργούμε το μοντέλο μας που κληρονομεί και χρησιμοποιεί όλες αυτές της κλάσεις για να δημιουργήσει το νευρωνικό δίκτυο.

1. **Embedding layer**: Το στρώμα ενσωμάτωσης είναι στρώμα στο οποίο δέχεται σαν είσοδο το Γραμμικό γράφημα δηλαδή για κόμβους έχει τις ακμές του αρχικού γραφήματος και τα χαρακτηρίστηκα των κόμβων που είναι η ευκλείδειες αποστάσεις. Στη συνέχεια τα επεξεργάζεται με την βοήθεια κάποιου πίνακα βαρών (Weight Matrix) που μαθαίνεται κατά την εκπαίδευση και αρχικοποιείται τυχαία και την βοήθεια ενός διανύσματος μετατοπίσεων (bias) που επίσης μαθαίνεται και αρχικοποιείται με μηδενικά.

```
1:  class EmbeddingLayer(nn.Module):
2:      def __init__(self, input_dim, output_dim, num_layers=1):
3:              super(EmbeddingLayer, self).__init__()
4:              self.W = nn.Parameter(torch.randn(input_dim,
    output_dim))#LearnableWeightMatrix
5:              self.b = nn.Parameter(torch.zeros(output_dim))  # Learnable biases
6:      def forward(self, x):
```

```
7:              h = x
8:              h = torch.matmul(h, self.W)
9:              h += self.b
10:             return h
```

2. **GNN layers**: Στη περίπτωση των **Graph Neural Network** συνδυάζουμε δύο διαφορετικά νευρωνικά δίκτυα για τα επόμενα στρώματα το ένα είναι το **Graph Attention Network (Gat)** και το άλλο είναι το **FeedForward**. Αφού έχει περαστεί από την πρώτη στρώση του νευρωνικού παίρνει τα αποτελέσματα και τα περνάει στο **Gat,** θέτω το **Gat** ώστε να έχει ως αριθμό κεφαλών προσοχής (attention heads) ίσο με 8 τις οποίες ενώνω στο τέλος με την εντολή concatenated και τέλος αφότου περαστούν από των νευρώνα **Gat** περνάνε από τη λειτουργία ενεργοποίησης **ReLU** στην έξοδο. Μετά έχουμε το **FeedForward** αυτή η κλάση υλοποιεί ένα απλό εμπρός τροφοδοτούμενο νευρωνικό δίκτυο με ένα κρυφό υπόστρωμα με διάσταση 512 και μια λειτουργία ενεργοποίησης **ReLU** στην έξοδο**.**

```
1:  class GATLayer(nn.Module):
2:      def __init__(self, input_dim, output_dim, num_heads=8, concat=True):
3:              super(GATLayer, self).__init__()
4:              self.gat = GATConv(input_dim, output_dim, heads=num_heads,
    concat=concat,                                  dropout=0.6)
5:      def forward(self, x, edge_index):
6:              x = self.gat(x, edge_index)
7:              return F.relu(x)
8:  class FeedForward(nn.Module):
9:      def __init__(self, input_size, output_size, hidden_dim=512):
10:             super(FeedForward, self).__init__()
11:             self.fc = nn.Linear(input_size, hidden_dim)
12:             self.fc2 = nn.Linear(hidden_dim, output_size)
13:     def forward(self, x):
14:             x = self.fc(x)
15:             x = self.fc2(x)
16:             x = F.relu(x)
17:             return x
```

3. **Output Layer**: Τέλος έχουμε ένα στρώμα εξόδου στο οποίο είναι σχεδόν το ίδιο με το στρώμα εισόδου με τη διαφορά ότι η τιμή που δέχεται ως είσοδος είναι τα αποτελέσματα των προηγούμενων νευρώνων.

```
1:  class Output_layer(nn.Module):
2:      def __init__(self, input_dim, output_dim):
3:              super(Output_layer, self).__init__()
4:              self.W = nn.Parameter(torch.randn(input_dim,output_dim))
    #LearnableWeightMatrix
5:              self.b = nn.Parameter(torch.zeros(output_dim))  # Learnable biases
6:      def forward(self, x):
7:              h0ij = torch.matmul(x, self.W)
8:              h0ij += self.b
9:              return h0ij
```

Με την βοήθεια αυτών των κλάσεων φτιάχνεται το νευρωνικό δίκτυο στο οποίο τοποθετώ στην συνάρτηση forward τον κώδικα για να ταυτίζεται με το σχήμα (**4.14**). Σε αυτό το σημείο θα πρέπει να προσθέσω ότι εκτός των προηγούμενων κλάσεων, με την βοήθεια της βιβλιοθήκης **torch** πιο συγκεκριμένα της **torch.nn** προσθέτω **nn.BatchNorm1d** ανάμεσα στο αποτέλεσμα του **Gat** και του **FeedForward** τα οποία κατορθώνουν κοινωνικοποίηση κατά παρτίδες τα οποία βοηθούν στη σταθεροποίηση και την επιτάχυνση της εκπαίδευσης νευρωνικών δικτύων.

```
10: class GNNModel(nn.Module):
11:     def __init__(self, input_dim, hidden_dim, output_dim, num_layers):
12:         super(GNNModel, self).__init__()
13:         self.embedding_layer = EmbeddingLayer(input_dim, hidden_dim)
14:         self.batch_norm1 = nn.BatchNorm1d(hidden_dim)
15:         self.gat_layers = nn.ModuleList([GATLayer(hidden_dim, hidden_dim // 8,
                                                num_heads=8) for _ in range(num_layers)])
16:         self.feedforward = FeedForward(hidden_dim, hidden_dim)  # Adjusted
    output_size
17:         self.batch_norm = nn.BatchNorm1d(hidden_dim)
18:         self.output_layer = Output_layer(hidden_dim, output_dim)
19:     def forward(self, x, edge_index1):
20:         x1 = self.embedding_layer(x)
21:         for layer in self.gat_layers:
22:             x2 = layer(x1, edge_index1)
23:             x1 = x1 + x2
24:         combined1 = x1
25:         bn_output1 = self.batch_norm1(combined1)
26:         ff_output = self.feedforward(bn_output1) # Apply feedforward operation
27:         combined = bn_output1 + ff_output
        #AddTheOriginalInputWithTheFeedForward
28:         bn_output = self.batch_norm(combined) # Apply batch normalization
29:         x = self.output_layer(bn_output)
30:         return x
```

5.3.3 Guided Local Search

Αφότου ολοκληρωθεί η διαδικασία να αποτραβηχτεί η τιμή regret από το νευρωνικό δίκτυο, ερχόμαστε στο σημείο το οποίο θα επιλυθεί το πρόβλημα του περιπλανώμενου πωλητή χρησιμοποιώντας τη καθοδηγούμενη τοπική αναζήτηση σαν τρόπο επίλυσης. Για να κατορθώσω αυτή την επίλυση χρειάστηκε να φτιάξω μια κλάση Guided_local_Search() στην οποία τοποθέτησα τις απαραίτητες μεθόδους που θα μας βοηθήσουν στο να κατορθώσουμε ένα αξιόλογο ποσό για τιμή αποστάσεων, διότι όπως έχει προαναφερθεί το TSP είναι NP-Hard problem δηλαδή είναι δύσκολο έως αδύνατον να βρεθεί η βέλτιστη λύση σε εφικτό χρόνο. Στο σημείο αυτό θα εξηγήσω τις μεθόδους.

1. **make_edge_index_and_regret:** Σε αυτή την μέθοδο παίρνω τις ακμές που έχω δημιουργήσει και τα regret και δημιουργώ τα αντίστροφο τους π.χ. *[α, β]* σε *[β, α]* και παίρνω το regret που αρμόζει για αυτή την ακμή και φτιάχνω πίνακες που διαθέτουν και τις δυο κατευθύνσεις. Αυτό θα χρειαστεί σε μετέπειτα σημείο.

```
1:  def make_edge_index_and_regret(output,edge_index):
2:      output = torch.squeeze(output, 1)
3:      output= output.detach().numpy()
4:      node_index2 = []
5:      node_index1 = []
6:      node_index3 = []
7:      regret_values = []
8:      for i in range(len(edge_index)):
9:              node_index3.append(edge_index[i])
10:             regret_values.append(output1[i])
11:     for i in range(len(edge_index)):
12:             node_index2.append(edge_index[i][1])
13:             node_index1.append(edge_index[i][0])
14:             node_index3.append((node_index2[i], node_index1[i]))
15:             regret_values.append(output1[i])
16:     return node_index3,regret_values
```

2. **calculate_distance:** Υπολογίζει τις αποστάσεις και επιστρέφει το άθροισμα των αποστάσεων της διαδρομής.

```
1:  def calculate_distance(path, graph):
2:      distance = 0
3:      for i in range(len(path) - 1):
4:              distance += graph[path[i]][path[i + 1]]
5:      distance += graph[path[-1]][path[0]]  # Return to starting point
6:      return distance
```

3. **greedy_nearest_neighbor:** Αυτή η μέθοδος δημιουργεί τα θεμέλια της επίλυσης φτιάχνοντας την αρχική λύση, παίρνοντας την τιμή regret σαν οδηγό η μέθοδος διαλέγει βήμα βήμα την ακμή με την μικρότερη τιμή regret και έτσι φτιάχνεται η πρώτη διαδρομή που θα παραδοθεί μετέπειτα στις επόμενες μεθόδους για να βγάλουν καλύτερο αποτέλεσμα.

```
1:  def greedy_nearest_neighbor(regret, edges, distances):
2:      num_cities = distances.shape[0]
3:      visited = [False] * num_cities
4:      tour = [0]  # Start from city 0
5:      current_city = 0
6:      visited[current_city] = True
7:      sum_tour_d = 0
8:      for _ in range(num_cities - 1):
9:              nearest_city = None
10:             min_distance = float('inf')
11:             for next_city in range(len(regret)):
12:                     poli1, poli2 = Guided_local_Search.get_nodes(edges[next_city])
13:                     if visited[poli1] == False and visited[poli2] == True and current_city
                        == poli2:
```

```
14: .                           if regret[next_city] < min_distance:
15:                                 nearest_city = poli1
16:                                 min_distance = regret[next_city]
17:                     elif visited[poli2] == False and visited[poli1] == True and
                        current_city == poli1:
18:                             if regret[next_city] < min_distance:
19:                                 nearest_city = poli2
20:                                 min_distance = regret[next_city]
21:             tour.append(nearest_city)
22:             visited[nearest_city] = True
23:             current_city = nearest_city
24:             sum_tour_d = Guided_local_Search.calculate_distance(tour, distances)
25:     return tour, sum_tour_d
```

4. **two_opt_swap:** Αυτή η μέθοδος είναι βοηθητική για την μετέπειτα μέθοδο που θα αναλυθεί, στην ουσία αυτό που προσφέρει είναι ότι αντιστρέφει ένα συγκεκριμένο σημείο τις διαδρομής για παράδειγμα από το σημείο *[i]* έως το σημείο *[j]* θα αντιστραφεί όλοι αυτή η διαδρομή και θα ξεκινάει από το *[j]* και θα καταλήγει στο σημείο *[i]* και θα συνεχίσει την υπόλοιποι διαδρομή κανονικά.

```
1: def two_opt_swap(path, i, j):
2:     new_path = path[:i] + path[i:j+1][::-1] + path[j+1:]
3:     return new_path
```

5. **two_opt_local_search:** Σε αυτό το σημείο αλλάζουμε για ακόμα μια φορά την διαδρομή χρησιμοποιώντας την τεχνική 2-opt, με άλλα λόγια αλλάζοντας την διαδρομή σε από δύο σημεία, αυτό το κατορθώνουμε κρατώντας την αρχική διαδρομή και την αρχική τιμή της απόστασης σαν οδηγό και ξεκινώντας να αλλάζουμε με την σειρά διαδρομές. Έχοντας ως παράμετρο ασφαλείας να μην αλλάζει την πρωταρχική πόλη, το **two_opt_local_search** καλή την **two_opt_swap** και την **calculate_distance** μέθοδο για να μας βρει καλύτερη διαδρομή. Η διαδικασία διακόπτεται μόλις βρει την μικρότερη διαδρομή που μπορεί να βρε με την μέθοδο **two_opt_local_search.**

```
1:  def two_opt_local_search(path, graph, best_distance):
2:      n = len(path)
3:      best_path = path
4:      improved = True
5:      while improved:
6:      improved = False
7:      for i in range(1, n - 2):
8:              for j in range(i + 1, n):
9:                      if (j - i == 1) and (i !=0) and (j !=0):
10:                         new_path=Guided_local_Search.two_opt_swap(best_path,i,j)
11:                         new_distance =
    Guided_local_Search.calculate_distance(
        new_path, graph)
12:                     if new_distance < best_distance:
13:                         best_distance = new_distance
14:                         best_path = new_path
15:                         improved = True
16:                         break
```

```
                if improved:
                    break
    return best_path, best_distance
```

6. **relocate_algorithm:** Αφότου ολοκληρωθεί η **2-opt** μέθοδος συνεχίζουμε στην επόμενη που είναι η **relocate.** Ουσιαστικά αυτό που κάνει αυτή η μέθοδος είναι να αλλάζει τυχαία μέσα στη διαδρομή μια πόλη σε μια άλλη θέση, μόλις το κάνει αυτό υπολογίζει την απόσταση με την βοήθεια τις μεθόδου **calculate_distance** και αν είναι μικρότερη η διαδρομή τότε αλλάζει την παλιά διαδρομή με την νέα. Αυτή την διαδικασία την κάνει με επανάληψη μέχρι να ολοκληρωθεί ο αριθμός επαναλήψεων.

```
def relocate_algorithm(distance_matrix, old_route, best_distance, max_iterations=1000):
    num_cities = len(old_route)
    current_route = old_route[:]  # Copy the initial route
    current_distance = best_distance
    for _ in range(max_iterations):
        city1, city2 = np.random.choice(range(1, num_cities), size=2, replace=False)
        if city1 > city2:
            city1, city2 = city2, city1
        new_route = current_route[:city1] + current_route[city1+1:]
        new_route = new_route[:city2] + [current_route[city1]] + new_route[city2:]
        if new_route[0] != old_route[0]:
            new_route.remove(old_route[0])
            new_route.insert(0, old_route[0])
        new_distance = Guided_local_Search.calculate_distance( new_route,
    distance_matrix)
        if new_distance < current_distance:
            current_route = new_route
            current_distance = new_distance
    return current_route, current_distance
```

Έχοντας ολοκληρώσει αυτές τις πράξεις το μοντέλο βρίσκει μια διαδρομή, το πρόβλημα σε αυτό το κομμάτι είναι το γεγονός ότι αφήνοντας τον κώδικα σε αυτό το σημείο καταλήγουμε να βρισκόμαστε σε κάποια σενάρια με τοπικά ελάχιστα (συνήθως σε αριθμούς πόλεων από 20 και πάνω). Για να αποφευχθεί αυτό έρχεται να βοηθήσει το νευρωνικό δίκτυο. Πιο συγκεκριμένα η μεταβλητή regret που βρίσκουμε σαν output από το νευρωνικό δίκτυο. Σύμφωνα με τους [88] βλέποντας την διαδρομή που έχει βγει ως αποτέλεσμα, βλέπουμε αν περιέχει ακμές με μεγάλο regret, αυτές η τιμές παίρνουν έναν πόντο ποινής. Οι ακμές με τις μεγαλύτερες τιμές regret το μοντέλο επιχειρεί να τις αλλάξει και έτσι αποφεύγουμε τα τοπικά ελάχιστα. Παρακάτω συνεχίζουμε με τις μεθόδους που κατορθώνουν αυτό το αποτέλεσμα.

7. **penalize_edges:** Παίρνοντας όλους τους κόμβους που φτιάξαμε από το **make_edge_index_and_regret()** και τις τιμές regret που αρμόζουν σε κάθε κόμβο, η μέθοδος αυτή αρχίζει να δίνει ποινές στις ακμές με την βοήθεια τις πράξεις που αναφέρεται στο (**4.11**) αφού κάνει τους υπολογισμούς και δώσει τις ποινές, η μέθοδος επιστρέφει δύο πίνακες έναν πίνακα με τις θέσεις που βρίσκονται οι ακμές που έχουν πάρει ποινή (όσες δεν έχουν πάρει ποινή ο πίνακας έχει στις θέσεις τους τιμή 0) και έναν πίνακα με την ακμή που έχει τις περισσότερες ποινές.

```
def penalize_edges(regret_values, penalties, edges,  node_index3):
```

```
    utilities = np.zeros(len(edges))
    for i in range(len(edges)):
        edge = edges[i]
        edge_idx = Guided_local_Search.find_the_value(edge, node_index3)
        utilities[i] = regret_values[edge_idx] / (1 + penalties[edge_idx])
    max_utility = np.max(utilities)
    max_utility_indices = np.where(utilities == max_utility)[0]
    max_utility_edges = [edges[i] for i in max_utility_indices]
    for edge in max_utility_edges:
        edge_idx = Guided_local_Search.find_the_value(edge, node_index3)
        penalties[edge_idx] += 1
    return penalties, max_utility_edges
```

8. **local_search_on_penalized_edges:** Τέλος αυτή η μέθοδος παίρνει την διαδρομή που έχει βρεθεί από το **relocate_algorithm()** και την τιμή της απόστασης, μαζί με αυτά παίρνει τον πίνακα τον ποινών και τον πίνακα με τις περισσότερες ποινές από το **penalize_edges()** και στην ουσία ξανά εφαρμόζει το **2-opt** και το **relocate** σαν μεθόδους. Η διαφορά με τις προηγούμενες μεθόδους είναι ότι τα εφαρμόζει μόνο στις τιμές που έχουν βρεθεί με μεγάλο αριθμό ποινών. Έτσι το μοντέλο επιχειρεί να αφαιρέσει κοστοβόρες διαδρομές αν το αποτέλεσμα τις αποστάσεις αποδειχθεί μικρότερο.

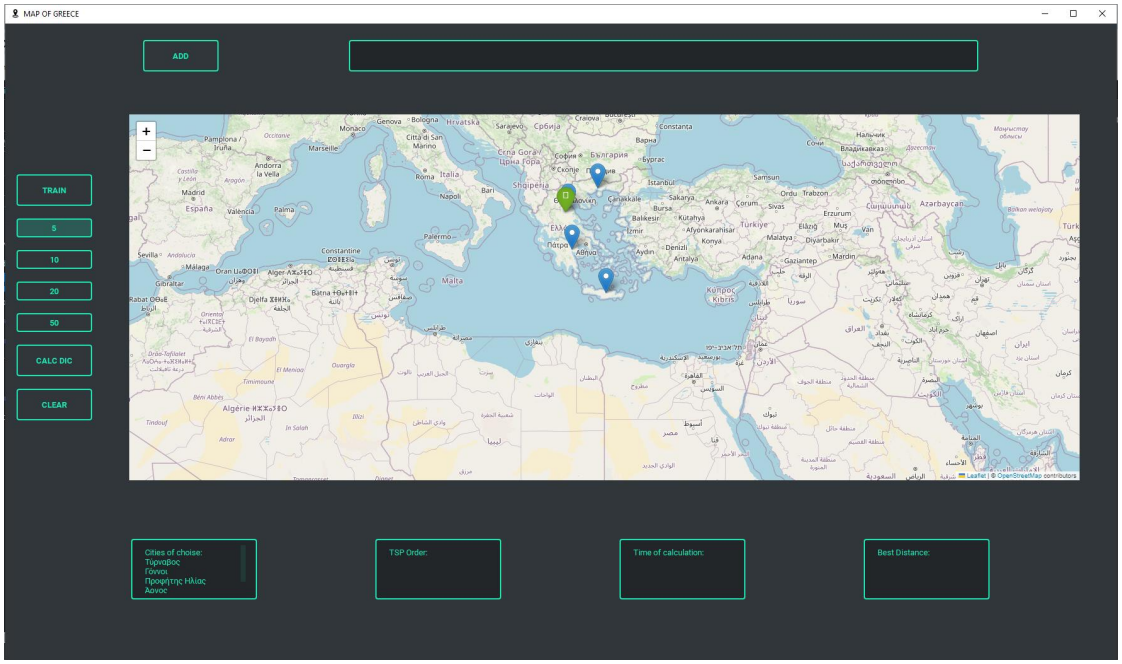
```
def local_search_on_penalized_edges(solution, best_distance, max_utility_edges,
node_index3, regret_values, distances):
    improved_tour = solution[:]
    improved_dis = best_distance
    for edge in max_utility_edges:
        for i in range(1, len(improved_tour) - 1):
            for j in range(i + 1, len(improved_tour)):
                if (improved_tour[i], improved_tour[j]) == edge or
(improved_tour[j], improved_tour[i]) == edge and ((improved_tour[i] != 0 ) and
(improved_tour[j] != 0)):
                    new_path =
Guided_local_Search.two_opt_swap(
    improved_tour, i, j)
                    new_distance =
    Guided_local_Search.calculate_distance( new_path, distances)
                if new_distance < improved_dis:
                    improved_tour = new_path
                    improved_dis = new_distance
    for edge in max_utility_edges:
        for i in range(1, len(improved_tour) - 1):
            if (improved_tour[i] in edge) and (improved_tour[i] != 0):
                for j in range(len(improved_tour)):
                    if j != i and (improved_tour[j], improved_tour[i]) not
in max_utility_edges and (improved_tour[i], improved_tour[j]) not in max_utility_edges
and (improved_tour[j] != 0):
. new_path, new_distance =
    Guided_local_Search.relocate_algorithm(distances, improved_tour, improved_dis,
                                           improved_tour[i])
```

```
19:                                                 if new_distance < improved_dis:
20:                                                     improved_tour = new_path
21:                                                     improved_dis = new_distance
22:         return improved_tour, improved_dis
```

Εικόνα 5.2: Στη εικόνα βλέπουμε την επίλυση του TSP με το GLS.

5.4 Γραφικό Περιβάλλον

Σε αυτό το σημείο θα αναφερθούμε στο γραφικό περιβάλλον η αλλιώς User Interface (UI). Με άλλα λόγια θα αναλύσουμε το πως έφτιαξα το γραφικό περιβάλλον ώστε να μπορεί ο χρήστης να έχει ένα πιο όμορφο πρόγραμμα και πιο διαδραστικό από απλές εμφανίσεις αποτελεσμάτων, επίσης το πως συνδύασα όλες τις μεθόδους και τις κλάσεις μαζί για να μπορεί όχι μόνο με τυχαία επιλογή να παίρνει πόλεις αλλά και με το να πληκτρολογεί ονόματα πόλεων.

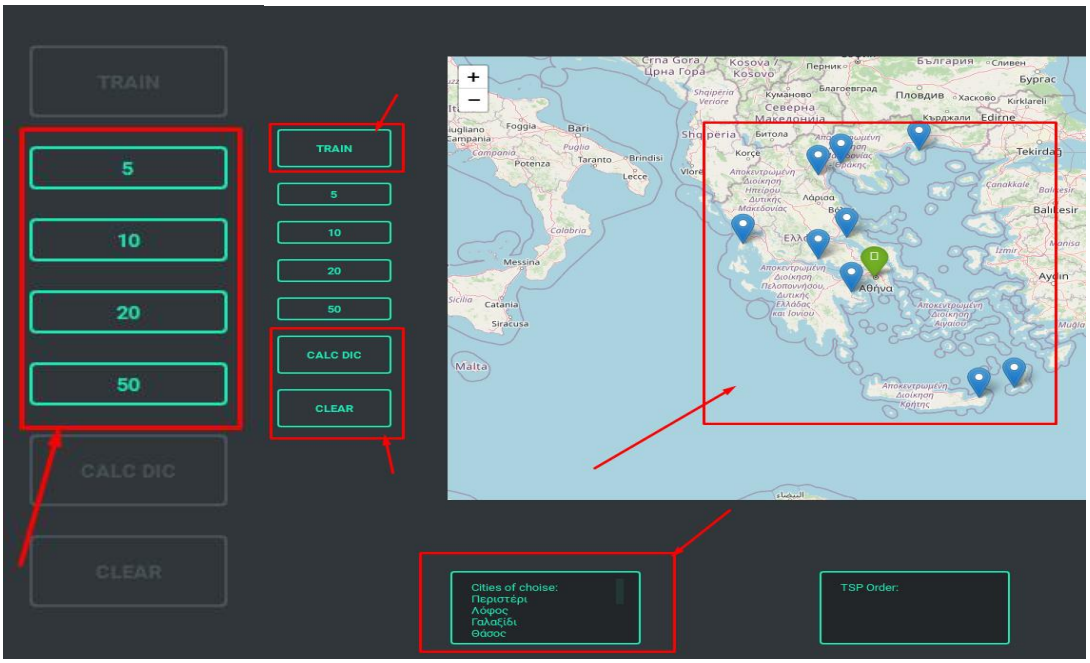


Εικόνα 5.3: Στην εικόνα φαίνεται το UI της εφαρμογής

5.4.1 Δομή Παραθύρου

Χρησιμοποιώντας όπως αναφερθήκαμε στην αρχή του κεφαλαίου το **PyQt5** ως βιβλιοθήκη για το γραφικό περιβάλλον η δομή που υπάρχει είναι πολύ απλή. Η κλάση του UI διαθέτει τις εξής μεθόδους που αφορούν το οπτικό κομμάτι του παραθύρου, έχει την μέθοδο ***initWindow()*** στην οποία αρχικοποιεί την διάσταση του παραθύρου, τον τίτλο και προσθέτω μια εικόνα πάνω αριστερά στο παράθυρο με την εντολή ***setWindowIcon***. Στη συνέχεια έχουμε την μέθοδο ***buttonUI()***στην οποία δημιουργούνται οι θέσεις τον αντικειμένων όπως η θέση των κουμπιών δεξιά από το χάρτη η θέση του χάρτη κ.α. Μαζί με αυτά δημιουργούνται τα κουμπιά και τα textAreas τα οποία έχουν μια περιγραφή μέσα σαν text και τίποτα άλλο. Έπειτα έχουμε την ***init_map()*** στην οποία δημιουργείτε ο χάρτης με τις συντεταγμένες που θέλουμε για να απεικονίζεται στο κέντρο η Ελλάδα και μετά αποθηκεύεται σαν αρχείο **html**, με την εντολή ***setHtml(html)*** της **PyQt5** μπορούμε να εμφανίσουμε τον χάρτη στο παράθυρο. Τέλος έχουμε την μέθοδο ***__init__()*** η οποία είναι η αρχική μέθοδος της κλάσης και εκεί αρχικοποιούνται τα σημαντικότερα κομμάτια στο γραφικό αλλά και στο κομμάτι τον μεταβλητών.

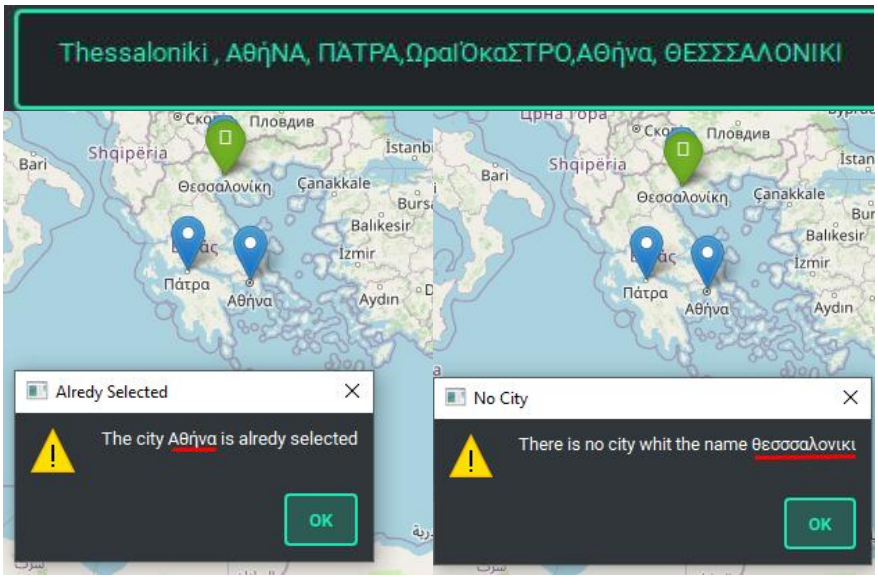
Στη συνέχεια εκτός αυτών των μεθόδων που αφορούν το γραφικό περιβάλλον, θα ήθελα να αναφερθώ στις μεθόδους που αφορούν το κομμάτι του προγράμματος. Με την μέθοδο ***handle_main_program( )*** γίνονται τα εξής, αφού ο χρήστης έχει πατήσει ένα από τα τέσσερα κουμπιά **5, 10, 20, 50** τότε το πρόγραμμα καλεί αυτή την μέθοδο και καλή και αυτή με την σειρά της την μέθοδο ***main_program()*** που αναφερθήκαμε στο 5.3.1, ανάλογα με το κουμπί που έχει πατήσει ο χρήστης θα επιστραφούν ο ανάλογος αριθμός από τυχαίες πόλεις τις Ελλάδας και θα ανανεωθεί ο χάρτης στο παράθυρο. Εκτός από αυτό η μέθοδος ανανεώνει το textArea που έχει σαν περιγραφή ***Cities of choise*** και τοποθετεί από κάτω τις πόλεις, επίσης ενεργοποιεί τα κουμπιά ***TRAIN, Calc Dic*** και το ***Clear*** με την εντολή **setEnabled(True).**



Εικόνα 5.4: Απεικόνιση των κουμπιών και των αλλαγών στο παράθυρο μετά από το κλικ σε κάποιο από τα κουμπιά.

Αφού αναφέραμε την τυχαία τοποθέτηση πόλεων είναι η κατάλληλη στιγμή να αναφερθούμε για την προσθήκη πόλεων με επιλεγμένες πόλεις. Στη θέση πάνω από το χάρτη βρίσκονται τα δύο εργαλεία

που θα μας βοηθήσουν σε αυτό. Ξεκινώντας με το text area, ο χρήστης έχει την δυνατότητα να πληκτρολογήσει ότι θέλει και σε όποια γλώσσα θέλει. Στη συνέχεια πατώντας το κουμπί ADD καλείτε η μέθοδος ***add_coordinates( )*** στην οποία γίνονται τα εξής. Η μέθοδος παίρνει σαν ένα μεγάλο string το ότι έχει γράψει ο χρήστης, το μετατρέπει όλο το κείμενο σε μικρά με την εντολή **lower()** και μετά το διαχωρίζει με το σημείο στίξης να είναι το κόμμα. Έχοντας από ένα sting πλέον ένα πίνακα με strings η μέθοδος αναζητά αν υπάρχει στη βάση δεδομένων μια πόλη με αυτό το όνομα. Σε περίπτωση που υπάρχει η μέθοδος κρατάει τα στοιχεία όπως συντεταγμένες και όνομα και στη συνέχει εμφανίζει στο χάρτη την πόλη, επίσης ενημερώνει το text area με όνομα **Cities of choise** και εμφανίζει την πόλη. Αν δεν υπάρχει αυτό το όνομα (ο χρήστης έκανε κάποιο λάθος) η υπάρχει ήδη σας πόλη επιλεγμένη, τότε το σύστημα καλεί την μέθοδο ***popupMessage( )*** και εμφανίζει ανάλογο μήνυμα.

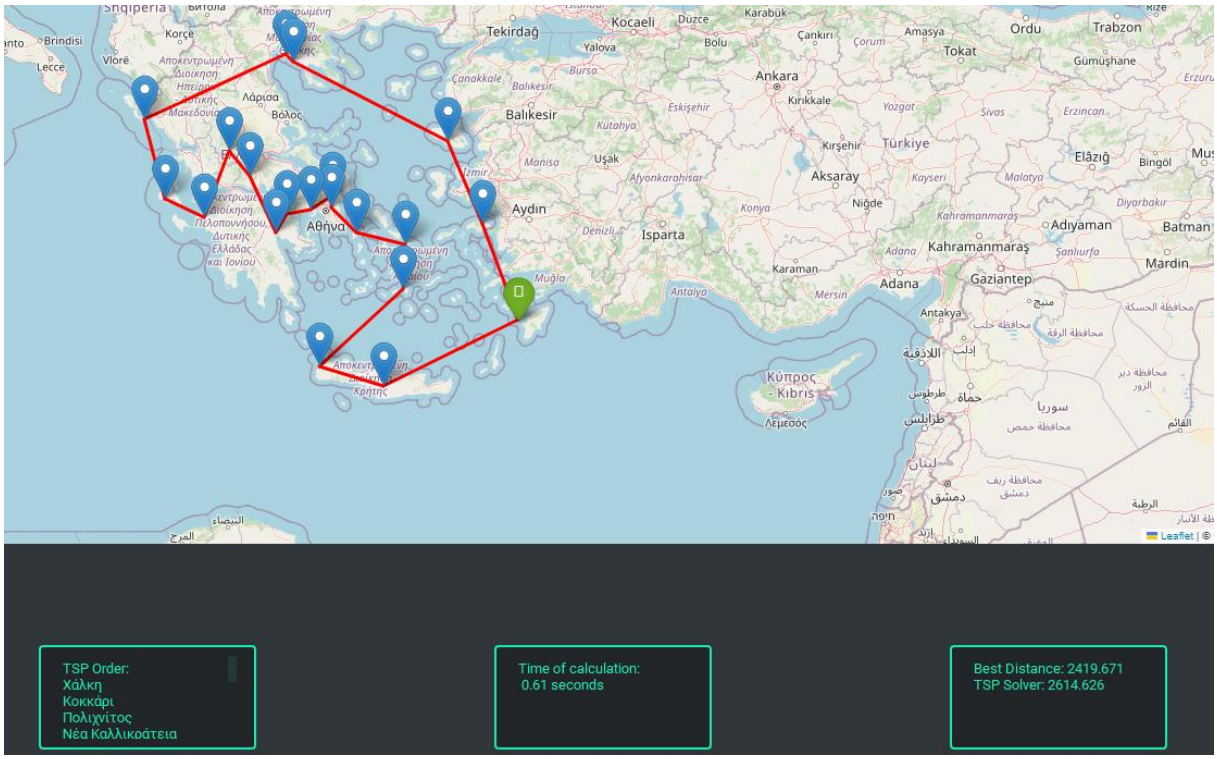


Εικόνα 5.5: Βλέπουμε τα παραδείγματα σε περιπτώσεις λαθών.

Αφότου δοθούν πόλεις από τρεις και πάνω ενεργοποιείται τα κουμπιά ***TRAIN*** και ***Calc Dic.*** Με το πάτημα του κουμπιού ***TRAIN*** εκτελούνται τα εξής γεγονότα. Πρώτα το καλείται η μέθοδος ***trainingClick( )*** στην οποία έχοντας τον αριθμό των πόλεων δημιουργεί πίνακες με τυχαίες τιμές με συντεταγμένες, χιλιομετρικές αποστάσεις αυτών των συντεταγμένων και ακμές για αυτές τις συντεταγμένες. Κάθε πίνακας διαθέτει πίνακες με δείγματα από τυχαίες τιμές και έχουν μέγεθος ανάλογο με των αριθμό των πόλεων που έχουν επιλεχθεί. Στη συνέχεια μετατρέπει τις ακμές σε γράφημα και από γράφημα σε γραμμικό γράφημα για κάθε ένα από τα σενάρια και μετά τα περνάει στο νευρωνικό δίκτυο. Αφότου πάρει τις τιμές regret υπολογίζω για κάθε σενάριο με την βοήθεια της βιβλιοθήκης ***python_tsp.heuristics*** και έχοντας ως οδηγώ το (4.12) δημιουργώ τους στόχους για κάθε δήγμα για τιμές regret που έχει ο κάθε κόμβος. Τέλος έχοντας τα αποτελέσματα από το νευρωνικό και την πραγματική τιμή κάνω εκπαίδευση το μοντέλο μου χρησιμοποιώντας εκπαίδευση με επίβλεψη.

Όταν ο χρήστης πατήσει το κουμπί ***Calc Dic***, καλείτε η μέθοδος ***calculate_distance( )*** στην οποία γίνονται τα εξής γεγονότα. Πρώτα δημιουργεί με την βοήθεια τις βιβλιοθήκης time μια μεταβλητή start_time στην οποία κρατάει τον χρόνο που ξεκίνησε. Μετά με τα είδη αποθηκευμένα δεδομένα (ονόματα πόλεων, συντεταγμένες πόλεων) από το αρχείο **ApoKomb1** καλεί τις απαραίτητες μεθόδους που αναφέραμε στο 5.3, στη συνέχεια καλεί την κλάση από το **main_p, *graph_to_LineGraph()*** και δημιουργεί ένα Line graph. Μετέπειτα δημιουργεί το νευρωνικό μας δίκτυο και τοποθετεί τα απαραίτητα δεδομένα και στη συνέχεια καλεί το Guided_local_Search() για να υπολογίσει τις

αποστάσεις. Στη συνέχεια έχοντας κρατήσει την σειρά με την οποία θα περάσει ο πωλητής τις πόλεις, τις βγάζει με την σειρά αυτή στο textArea με τίτλο TSP Oder:, με σειρά που βρήκαμε εμφανίζουμε και στο χάρτη τις γραμμές ανάμεσα στις πόλεις με την βοήθεια τις folium πιο συγκεκριμένα με την εντολή: ***folium.PolyLine(locations=[location[distances[i]], location[distances[i+1]]], color='red').add_to(self.m).*** Έχοντας ολοκληρώσει τις όλα αυτά στο τέλος ξανά καλούμε την εντολή time για να βρούμε τον χρόνο που τελείωσε το πρόγραμμα και αφαιρούμε το end_time – start time και έτσι βρίσκουμε τον χρόνο που χρειάστηκε να εκτελεστεί.



Εικόνα 5.6: Βλέπουμε με το που πατηθεί το Calc Dic τις ενημερώσεις που κάνει.

Τέλος πατώντας ο χρήστης το κουμπί **Clear** τότε το πρόγραμμα καθαρίζει όλο το UI από εμφανίσεις στο χάρτη η στα text areas και απενεργοποιεί τα κουμπιά **Clear** και **Calc Dic**. Έχοντας ολοκληρώσει και το γραφικό περιβάλλον το πρόγραμμα θεωρείται πλήρες και ως αποτέλεσμα πιο διαδραστικό, στη συνέχεια θα αναφερθούμε σε μελλοντικά έργα πάνω στο πρόγραμμα και τέλος έναν επίλογο.

5.5 Σενάρια Για Το Training

Έπειτα από την ολοκλήρωση του προγράμματος αποφασίσαμε να δοκιμάσουμε την εκπαίδευση του προγράμματος με δύο διαφορετικούς τρόπους. Ο πρώτος ήταν με τυχαίες συντεταγμένες και ο δεύτερος ήταν με συντεταγμένες από την βάση δεδομένων. Ο λόγος που έγινε αυτό ήταν για να δούμε την αντίδραση που θα έχει το σύστημα σε σχέση με τα αποτελέσματα. Στο παρακάτω πίνακα θα δοθούν τα αποτελέσματα από το σύστημα για περιπτώσεις 20, 50 και 100 πόλεων, τα δεδομένα αυτά είναι ο μέσος όρος από των χρόνο εκτέλεσης και την απόκλιση από την λύση που προσφέρεται από το ***python_tsp.heuristics.***

Σενάριο	Μέθοδος	Computation time (s)	Optimality gap (%)

TSP20	Random Data	0.520 sec	0.040
	From The Dataset	**0.468 sec**	**-0.476**
TSP50	Random Data	**4.647 sec**	**-2.100**
	From The Dataset	4.631 sec	-1.657
TSP100	Random Data	40.802 sec	-2.654
	From The Dataset	**41.025 sec**	**-3.038**

Πίνακας 5.1: Στο παραπάνω πίνακα βλέπουμε για το ίδιο πρόγραμμα τα διαφορετικά αποτελέσματα ανάλογα των περιπτώσεων. Το Gap βρίσκετε με των εξής τρόπο (η λύση του μοντέλου μας, μείον την βέλτιστη λύση δια της βέλτιστης λύσης και επί το εκατό.

Από ότι παρατηρείτε στο παραπάνω πίνακα, το πρόγραμμα δεν δείχνει να έχει τεράστιες αλλαγές σε σχέση με τυχαίες συντεταγμένες και έτοιμες αν και έχει μια μικρή βελτίωση στην περίπτωση των σεναρίων που εκπαιδεύεται με τα δικά μας δεδομένα. Τα αποτελέσματα βρέθηκαν από των μέσω όρο σε 100 σενάρια για κάθε περίπτωση πόλεων ξεχωριστά.

Κεφάλαιο 6: Συμπεράσματα

6.1 Εισαγωγή

Σε αυτό το κεφάλαιο, θα πραγματοποιηθεί μια εις βάθος ανάλυση των επιδόσεων του μοντέλου μας, χρησιμοποιώντας έναν συνδυασμό από πίνακες και στατιστικά δεδομένα. Η χρήση αυτών των εργαλείων θα μας επιτρέψει να αποκτήσουμε μια πιο ολοκληρωμένη εικόνα για την αποδοτικότητα και την ακρίβεια του μοντέλου μας, καθώς και να εντοπίσουμε πιθανά σημεία βελτίωσης.

Η αξιολόγηση των επιδόσεων ενός μοντέλου είναι ένας κρίσιμος παράγοντας για την επιτυχία κάθε ανάλυσης δεδομένων ή μηχανικής μάθησης. Μέσα από την παρουσίαση των αποτελεσμάτων σε μορφή πίνακα, στοχεύουμε να προσφέρουμε μια σαφή και κατανοητή απεικόνιση των δεδομένων, επιτρέποντας την ευκολότερη σύγκριση και ερμηνεία τους. Επιπλέον, θα αναλύσουμε τους βασικούς δείκτες απόδοσης, όπως η μέση τιμή της βέλτιστης λύσης και ο χρόνος εκτέλεσης για να προσδιορίσουμε την αποδοτικότητα του μοντέλου να προβλέπει σωστές διαδρομές. Επίσης για να συμβάλει στην καλύτερη κατανόηση της αποδοτικότητας του μοντέλου, θα συμπεριληφθούν κάποιες άλλες μεθόδους επιλύσεις μαζί με κάποια αποτελέσματα από άλλα μοντέλα που επιλύουν το ίδιο πρόβλημα.

Είμαστε πεπεισμένοι ότι η ανάλυση αυτή θα παράσχει πολύτιμες γνώσεις και θα υποστηρίξει περαιτέρω την ανάπτυξη και βελτίωση του μοντέλου μας. Ευελπιστούμε ότι τα ευρήματα που θα παρουσιαστούν σε αυτό το κεφάλαιο θα συμβάλουν στην κατανόηση των επιδόσεων του μοντέλου μας και θα προσφέρουν σημαντικές κατευθυντήριες γραμμές για μελλοντική έρευνα και εφαρμογή.

6.2 Το Πείραμα

Για την επίτευξη της αποδοτικότητας του μοντέλου δημιουργήσαμε σενάρια με 20, 50 και 100 πόλεις από 1000 σενάρια η κάθε μια. Στις περιπτώσεις αυτές ελέγχουμε τον μέσω χρόνο εκτέλεσης για κάθε σενάριο και την μέση τιμή της βέλτιστης λύσης από την λύση που βρίσκουμε με την βοήθεια του ***python_tsp.heuristics.***

Σενάριο	Μέθοδος	Computation time (s)	Optimality gap (%)
TSP20	Nearest Neighbor	0.002 sec	13.043
	Local Search 2-Opt	**0.000 sec**	**0.001**
	Farthest Insertion	0.005 sec	2.536
	GNN-MCTS [108]	0.930 sec	0.010
	Hudson κ.α. [88]	**10.010 sec**	**0.000**
	Το μοντέλο μου	**0.001 sec**	**0.000**
TSP50	Nearest Neighbor	0.028 sec	15.588
	Local Search 2-Opt	0.198 sec	0.449
	Farthest Insertion	0.065 sec	7.263
	GNN-MCTS [108]	**3.580 sec**	**0.200**
	Hudson κ.α. [88]	10.037 sec	0.069
	Το μοντέλο μου	**5.273 sec**	**-1.473**
TSP100	Nearest Neighbor	0.215 sec	15.313
	Local Search 2-Opt	2.969 sec	-0.842
	Farthest Insertion	0.444 sec	12.456
	GNN-MCTS [108]	27.620 sec	1.040

	Hudson κ.α. [88]	**10.108 sec**	**0.801**
	Το μοντέλο μου	**45.958 sec**	**-2.836**

Πίνακας 6.1: Στο παραπάνω πίνακα απεικονίζονται τρία OR συστήματα και τρία μοντέλα με νευρωνικά δίκτυα, απεικονίζεται επίσης ο χρόνος επίλυσης που χρειάστηκαν για να λύσουν και το Optimality gap για τα τρία διαφορετικά σενάρια TSP.

Βλέποντας τα μοντέλα παρατηρούμε ότι στις περιπτώσεις των **20** πόλεων το μοντέλο μου είχε κατά μέσω όρο 0% αποχή από την βέλτιστη λύση που όπως προαναφέρθηκε παίρνουμε από το ***python_tsp.heuristics.*** Στη συνέχεια στις **50** πόλεις με μόλις **5.2** δευτερόλεπτα το μοντέλο είχε κατά μέσο όρο αρνητική απόκλιση **-1.473** αυτό σημαίνει ότι κατά μέσο όρο έβρισκε καλύτερες λύσεις από την βιβλιοθήκη που χρησιμοποιούνταν για βοηθός. Τέλος στις **100** πόλεις και λόγο μεγάλου φόρτου από δεδομένα και λόγο παλιάς κάρτας γραφικών, το μοντέλο χρειάστηκε **46** δευτερόλεπτα κατά μέσο όρο για να βγάλει αποτέλεσμα αλλά είχαμε **-2.836** απόκλιση, με άλλα λόγια το μοντέλο μας κατά μέσο όρο βγάζει κατά **2.8%** καλύτερα αποτελέσματα από μια βιβλιοθήκη που επιλύει το πρόβλημα το περιπλανώμενου πωλητή. Να σημειωθεί ότι τα αποτελέσματα από GNN-MCTS [108] και Hudson κ.α. [88] τα πήραμε από τα papers που παραθέτονται στη βιβλιογραφία.

6.3 Επίλογος

Έχοντας πλέον μια πιο καθαρή εικόνα για τις δυνατότητες των νευρωνικών δικτύων και της μηχανικής μάθησης, μπορούμε να καταλάβουμε το πόσο χρήσιμο μπορεί να γίνει σαν εργαλείο από καθημερινές χρήσεις όπως μετεωρολογικές προβλέψεις έως σε πιο σύνθετα ζητήματα όπως δημιουργία φαρμάκων και των πιθανών παρενεργειών πάνω σε κάποιον άνθρωπο. Βλέποντας όλα αυτά καταλαβαίνουμε ότι τα Graph Neural Networks μπορούν να γίνουν ένας στιβαρός πυλώνας στην επιστήμη και τον προγραμματισμό, έχοντας είδη κάνει δυνατή την παρουσία τους σε ένα σύντομο χρονικό διάστημα.

6.4 Μελλοντικά Έργα

Διερευνώντας το πρόβλημα του πλανόδιου πωλητή και πιο συγκεκριμένα πάνω στα γραφικά νευρωνικά δίκτυα, βρέθηκα στο συμπέρασμα ότι η επίλυση του προβλήματος σε μη ευκλείδειες αποστάσεις είναι μια πρόκληση που ακόμα δεν είναι εφικτή για επίλυση σε με τα εργαλεία που διακατέχω. Σαν μελλοντική εργασία θα ήταν να προσπαθήσω να το επιλύσω με πραγματικά δεδομένα όπως διαδρομές από αυτοκινητόδρομους με συνδυασμό διαδρομών πλοίων κ.α. να παρέχονται σαν παράμετροι στο νευρωνικό δίκτυο για επίλυση σε πραγματικά στοιχεία.

ΒΙΒΛΙΟΓΡΑΦΊΑ

[1] Phil Picto, “Introduction to Neural Networks”, 1994, doi: 10.1007/978-1-349-13530-1, pp 1–12

[2] Issam El N., Martin J. M., 2015, “Machine Learning in Radiation Oncology”,doi:10.1007/978-3-319-18305-3, pp 3–11

[3] Sperduit A., Starita A., “Supervised neural networks for the classification of structures”, May 1997, doi: 10.1109/72.572108, pp 714 – 735

[4] Frasconi, Gori M., Sperduti A., “A general framework for adaptive processing of data structures”, 1998, doi: 10.1109/72.712151, pp 768 – 786

[5] F. Scarselli, Marco G., Ah Chung T., Markus H., Gabriele M., “The Graph Neural Network Model”, December 2008, doi: 10.1109/TNN.2008.2005605, pp 61 – 80

[6] Micheli A., “Neural Network for Graphs: A Contextual Constructive Approach”, February 2009, doi: 10.1109/TNN.2008.2010350, pp 498 - 511

[7] Mikolov T., Kai C., Corrado G., Dean J., Sep 2013, “Efficient Estimation of Word Representations in Vector Space”, doi: 10.48550/arXiv.1301.3781

[8] Perozzi B., Al-Rfou R., Skiena S., August 2014, “DeepWalk: online learning of social representations”, doi: 10.1145/2623330.2623732

[9] Grover A., Leskovec J., August 2016, “node2vec: Scalable Feature Learning for Networks”, doi: 10.1145/2939672.2939754

[10] Tang J., Qu M., Wang M., Yan J., Mei Q., “LINE: Large-scale Information Network Embedding ”, May 2015, doi: 10.1145/2736277.2741093

[11] Cheng Y., Zhiyuan L., Deli Z., Maosong S., Edward Y. C.,“Network Representation Learning with Rich Text Information”,2015, pp. 2111-2117

[12] William L. Hamilton, Rex Y., Jure L., “Representation Learning on Graphs: Methods and Applications”, 2017b, doi: 10.48550/arXiv.1709.05584

[13] Michael M. B., Joan B., Yann L., Arthur S., Pierre V.,“Geometric deep learning: going beyond Euclidean data”, May 2017, doi: 10.48550/arXiv.1611.08097

[14] Wang, Minjie Yu,“Deep Graph Library: towards efficient and scalable deep learning on graphs”, January 2019,

[15] Wu Z., Shirui P., Fengwen C., Guodong L., Chengqi Z., Philip S. Y., “ A Comprehensive Survey on Graph Neural Networks”, March 2020, doi: 10.1109/TNNLS.2020.2978386, pp 4 – 24

[16] Jie Z., Ganqu C., Shengding H., Zhengyan Z., Cheng Y., Zhiyuan L., Lifeng W., Changcheng L., Maosong S.,“Graph neural networks: A review of methods and applications”, 2020, doi: 10.1016/j.aiopen.2021.01.001, pp 57-81

[17] Chengxi Z., Fei W.,“Neural Dynamics on Complex Networks”,2020, doi: 10.1145/3394486.3403132, pp 892–902

[18] Ziniu H., Yuxiao D., Kuansan W., Kai-Wei C.,Yizhou S.,“GPT-GNN: Generative Pre-Training of Graph Neural Networks”, August 2020, doi: 10.1145/3394486.3403237, pp 1857–1867

[19] Ziniu H., Yuxiao D., Kuansan W., Yizhou S., “Heterogeneous Graph Transformer”, April 2020, doi: 10.1145/3366423.3380027, pp 2704–2710

[20] Pau R., Andreas F., Josep L., Alicia F., “Learning Graph Distances with Message Passing Neural Networks”, 2018, doi: 10.1109/ICPR.2018.8545310

[21] Rex Y., Jiaxuan Y., Christopher M., Xiang R., William L. H., Jure L.,“Hierarchical Graph Representation Learning with Differentiable Pooling”, Jun 2018, doi: 10.48550/arXiv.1806.08804

[22] Yedid Hoshen,“VAIN: Attentional Multi-agent Predictive Modeling”, 2017, doi: 10.48550/arXiv.1706.06122, pp 2698 – 2708

[23] Thomas K., Ethan F., Kuan-Chieh W., Max W., Richard Z.,"Neural Relational Inference for Interacting Systems", *Feb 2018*, doi: 10.48550/arXiv.1802.04687
[24] David K. D., Dougal M., Jorge I., Rafael B., Timothy H., Alan Aspuru-G., Ryan P. A.,"Convolutional Networks on Graphs for Learning Molecular Fingerprints",2015, pp 2224-2232
[25] Steven K., Kevin McC., Marc B., Vijay P.,Patrick R.,"Molecular graph convolutions: moving beyond fingerprints", August 2016, doi: 10.1007/s10822-016-9938-8
[26] Kien D., Truyen T., Svetha V.,"Graph Transformation Policy Network for Chemical Reaction Prediction", July 2019, doi: 10.1145/3292500.3330958
[27] Nuo Xu, Pinghui W., Long C., Jing T., Junzhou Z.,"MR-GNN: Multi-Resolution and Dual Graph Neural Network for Predicting Structured Entity Interactions", 2019, doi: doi.org/10.24963/ijcai.2019/551, pp 3968-3974
[28] Sungmin Rhee, Seokjun Seo, Sun Kim,"Hybrid Approach of Relation Network and Localized Graph Convolutional Filtering for Breast Cancer Subtype Classification", 2018, doi: 10.24963/ijcai.2018/490, pp 3527-3534
[29] Marinka Zitnik, Monica Agrawal, Jure Leskovec,"Modeling polypharmacy side effects with graph convolutional networks", July 2018, doi: 10.1093/bioinformatics/bty294, pp i457–i466
[30] Jean-Louis B., Jean-Louis F., Daniel G.,"Multiscale Geomechanics", December 2011, doi: 10.1002/9781118601433.ch1, pp 1-9
[31] Olga R., Jia D., Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg & Li Fei-Fei, "ImageNet Large Scale Visual Recognition Challenge", April 2015, doi: 10.1002/9781118601433, pp 211-252
[32] Federico Baldassarre, Hossein Azizpour,"Explainability Techniques for Graph Convolutional Networks", 2019, doi: 10.48550/arXiv.1905.13686
[33] Alex Krizhevsky, Ilya Sutskever, Geoffrey E. Hinton,"ImageNet Classification with Deep Convolutional Neural Networks", June 2017, doi: 10.1145/3065386, pp 84–90
[34] Jacob Devlin, Ming-Wei Chang, Kenton Lee, Kristina Toutanova,"BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding", June 2019, doi: 10.18653/v1/N19-1423, pp 4171–4186
[35] Karla L. Hoffman, Manfred Padberg,Giovanni Rinaldi," Traveling salesman problem ", 2001, pp 225-330, doi: 10.1007/1-4020-0611-X_1068
[36] Dantzig, G.B., D.R. Fulkerson, S.M. Johnson, "Solution of a large scale traveling-salesman problem", pp 393-410, doi: 10.1287/opre.2.4.393
[37] Michael Held, Richard M. Karp,The traveling-salesman problem and minimum spanning trees: Part II, December 1971, doi: 10.1007/BF01584070
[38] Manfred Padberg, Martin Grötschel, Eugene Lawler, Jan Lenstra, Alexander Kan, David Shmoys, "The Traveling Salesman Problem", 1985, pp. 307-360.
[39] David L. Applegate, Robert E. Bixby, Vašek Chvátal and William J. Cook," The Traveling Salesman Problem", 2006, doi: 10.1515/9781400841103
[40] David S. Johnson, Lyle A. McGeoch,"Experimental Analysis of Heuristics for the STSP", 2002, doi: 10.1007/0-306-48213-4_9, pp 369–443
[41] Helsgun K.,"An effective implementation of the Lin–Kernighan traveling salesman heuristic",2000, doi: 10.1016/S0377-2217(99)00284-2
[42] Reinelt G., "TSPLIB–A traveling salesman library," 1991, doi: 10.1287/ijoc.3.4.376, pp 376–384
[43] Miller D., Pekny J.,"Exact Solution of Large Asymmetric Traveling Salesman Problems", 1991, doi: 10.1126/science.251.4995.754, pp 754-761
[44] Grötschel M., Padberg M.W.," Polyhedral Theory", 1985, pp 251–306.
[45] Michael Jünger, Gerhard Reinelt, Giovanni Rinaldi," Chapter 4 The traveling salesman problem ", 1995, doi: https://doi.org/10.1016/S0927-0507(05)80121-5, pp 225-330
[46] Denis Naddef,"Polyhedral Theory and Branch-and-Cut Algorithms for the Symmetric TSP", 2002, doi: 10.1007/0-306-48213-4_2, pp 29–116
[47] Egon Balas,"The Prize Collecting Traveling Salesman Problem and its Applications", 2002, doi: 10.1007/0-306-48213-4_14, pp 663–695

[48] Padberg, M.W., Rinaldi, G.,“A Branch and Cut Algorithm for the Resolution of Large-scale Symmetric Traveling Salesmen Problems”, 1991, doi: 10.1137/1033004, pp 60–100
[49] H. Donald Ratliff, Arnon S. Rosenthal,“Order-Picking in a Rectangular Warehouse: A Solvable Case for the Traveling Salesman Problem”, June 1983, doi: 10.1287/opre.31.3.507
[50] Ben-Dor, Benny Char, “On Constructing Radiation Hybrid Maps ”, Jan 1997, pp 17-26
[51] Amir Ben-Dor, Benny Chor, Dan Pelleg, “RHO—Radiation Hybrid Ordering”, 2000,pp 365-378
[52] Egon Balas, “The Prize Collecting Traveling Salesman Problem and its Applications”, 2002, doi: 10.1007/0-306-48213-4_14, pp 663-696
[53] Bruce L. Golden, Larry Levy, Rakesh Vohra, “The orienteering problem”, June 1987, doi: 10.1002/1520-6750(198706)34:3<307::AID-NAV3220340302>3.0.CO;2-D
[54] Toth Paolo, Vigo Daniele,“The vehicle routing problem ”, 2001
[55] Karen Simonyan, Andrew Zisserman, “Very Deep Convolutional Networks for Large-Scale Image Recognition”, Sep 2014, doi: 10.48550/arXiv.1409.1556
[56] K. Cho, B. V. Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, Y. Bengio, “Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation”, Sep 2014, doi: 10.48550/arXiv.1406.1078
[57] K. Cho, B. V. Merrienboer, D. Bahdanau, Y. Bengio,“On the Properties of Neural Machine Translation: Encoder-Decoder Approaches”, Oct 2014, doi: 10.48550/arXiv.1409.1259, pp 103–111
[58] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, M. Riedmiller,“ Playing Atari with Deep Reinforcement Learning”, Dec 2013, doi: 10.48550/arXiv.1312.5602
[59] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J.Veness, M. G. Bellemare, A. Graves, M.Riedmiller, A. K. Fidjeland, G. Ostrovski, S.Petersen, C. Beattie, A.Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, D. Hassabis,“Human-level control through deep reinforcement learning”, February 2015, doi: 10.1038/nature14236
[60] Richard Evans, Edward Grefenstette, “Learning Explanatory Rules from Noisy Data”, Jan 2018, doi: 10.1613/jair.5714
[61] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, G. Monfardini, “The Graph Neural Network Model”, December 2008, doi: 10.1109/TNN.2008.2005605, pp 61 – 80
[62] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, G. E. Dahl, “Neural Message Passing for Quantum Chemistry”, August 2017, pp 1263–1272
[63] Rasmus Palm, Ulrich Paquet, Ole Winther, “Recurrent Relational Networks”, 2017, doi: arXiv:1711.08028
[64] P. W. Battaglia, J. B. Hamrick, V. Bapst, A. Sanchez-Gonzalez, V. Zambaldi, M. Malinowski, A. Tacchetti, κ.α., “deep learning, and graph networks”, 2018, doi: 10.48550/arXiv.1806.01261
[65] D. Selsam, M. Lamm, B. Bünz, P. Liang, L. de Moura, D. L. Dill, “Learning a SAT Solver from Single-Bit Supervision”, Feb 2018, doi: 10.48550/arXiv.1802.03685
[66] M. Prates, P. H. C. Avelar, H. Lemos, L. C. Lamb, M. Y. Vardi, “Learning to Solve NP-Complete Problems: A Graph Neural Network for Decision TSP”, 2019, doi: 10.1609/aaai.v33i01.33014731
[67] Diederik P. Kingma, Jimmy Ba, “Adam: A Method for Stochastic Optimization”, Jan 2017, doi: 10.48550/arXiv.1412.6980
[68] Jeffrey M. Dudek, Kuldeep S. Meel, Moshe Y. Vardi, “Combining the k-CNF and XOR Phase-Transitions”, Feb 2017, doi: 10.48550/arXiv.1702.08392
[69] S. Kirkpatrick, G. Toulouse, “Configuration space analysis of travelling salesman problems”, 1985, doi: 10.1051/jphys:019850046080127700, pp 1277 – 1292
[70] W. Zhang, “Phase Transitions and Backbones of the Asymmetric Traveling Salesman Problem”, Apr 2004, doi: 10.1613/jair.1389, pp 471–497
[71] Jillian Beardwood, J. H. Halton, J. M. Hammersley, “The shortest path through many points”, October 1959, doi: 10.1017/S0305004100034095, pp. 299 – 327
[72] S. KIRKPATRICK, C. D. GELATT, JR., M. P. VECCHI, “Optimization by Simulated Annealing”, 1983, doi: 10.1126/science.220.4598.671, pp 671-680

[73] Jimmy Lei Ba, Jamie Ryan Kiros, Geoffrey E. Hinton, “Layer Normalization”, 2016, doi: 10.48550/arXiv.1607.06450
[74] Xavier Glorot, Yoshua Bengio, “Understanding the difficulty of training deep feedforward neural networks”, 2010, pp 249–256
[75] Yujiao Hu, Zhen Zhang , Yuan Yao , Xingpeng Huyan , Xingshe Zhou · Wee Sun Lee , “A bidirectional graph neural network for traveling salesman problems on arbitrary symmetric graphs”, January 2021, doi: 10.1016/j.engappai.2020.104061
[76] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, Y. Chen, T. Lillicrap, F. Hui, L. Sifre, G. van den Driessche, T. Graepel, D. Hassabis, “Mastering the game of Go without human knowledge”, 2017, doi: 10.1038/nature24270, pp 354–359
[77] Larson, R.C., Odoni, A.R., “Combinatorial optimization with graph convolutional networks and guided tree search. In: Advances in Neural Information Processing Systems, 1981, pp 573, http://worldcat.org/isbn/0139394478
[78] Sanjeev Arora, “Polynomial time approximation schemes for Euclidean traveling salesman and other geometric problems”, September 1998, doi: 10.1145/290179.290180, pp 753–782
[79] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, Clifford Stein, “Introduction to Algorithms, fourth edition”, 2009, https://books.google.gr/books?hl=el&lr=&id=RSMuEAAAQBAJ&oi=fnd&pg=PR13&ots=a2o8-V0GWO&sig=I6ZoCnnYRbLKtteSWveNzVbNhes&redir_esc=y#v=onepage&q&f=false
[80] David K. Duvenaud, Dougal Maclaurin, Jorge Iparraguirre, Rafael Bombarell, Timothy Hirzel, Alan Aspuru-Guzik, Ryan P. Adams, “Convolutional Networks on Graphs for Learning Molecular Fingerprints”, 2015, https://proceedings.neurips.cc/paper_files/paper/2015/hash/f9be311e65d81a9ad8150a60844bb94c-Abstract.html
[81] Peter Battaglia, Razvan Pascanu, Matthew Lai, Danilo Jimenez Rezende, koray kavukcuoglu, “Interaction Networks for Learning about Objects, Relations and Physics”, 2016, https://proceedings.neurips.cc/paper_files/paper/2016/hash/3147da8ab4a0437c15ef51a5cc7f2dc4-Abstract.html
[82] Wang, Y., Sun, Y., Liu, Z., Sarma, S.E., Bronstein, M.M., Solomon, J.M., “Dynamic graph CNN for learning on point clouds”,Jan 2018, doi: 10.48550/arXiv.1801.07829
[83] Zhen Zhang, Wee Sun Lee, “Deep Graphical Feature Learning for the Feature Matching Problem”, 2019, pp 5087-5096
[84] Charles R. Qi, Hao Su, Kaichun Mo, Leonidas J. Guibas, “Deep Learning on Point Sets for 3D Classification and Segmentation”, 2017, pp 652-660
[85] Keyulu Xu, Weihua Hu, Jure Leskovec, Stefanie Jegelka, “How Powerful are Graph Neural Networks?”, Oct 2018, dio: 10.48550/arXiv.1810.00826
[86] Wouter Kool, Herke van Hoof, Max Welling, “Attention, Learn to Solve Routing Problems! ”, Mar 2018,doi: 10.48550/arXiv.1803.08475
[87] Elias Khalil, Hanjun Dai, Yuyu Zhang, Bistra Dilkina, Le Song, “Learning Combinatorial Optimization Algorithms over Graphs”, 2017, pp 6348–6358
[88] Benjamin Hudson, Qingbiao Li, Matthew Malencia, Amanda Prorok, “Graph Neural Network Guided Local Search for the Traveling Salesperson Problem”, Oct 2021, doi: 10.48550/arXiv.2110.05291
[89] Manfred Padberg, Giovanni Rinaldi, “A Branch-and-Cut Algorithm for the Resolution of Large-Scale Symmetric Traveling Salesman Problems”,1990, doi: 10.1137/1033004
[90] David Applegate, Robert Bixby, Vašek Chvátal & William Cook, “Implementing the Dantzig-Fulkerson-Johnson algorithm for large traveling salesman problems ”, 2003, doi: /10.1007/s10107-003-0440-4, pp 91–153
[91] Keld Helsgaun, “An Extension of the Lin-Kernighan-Helsgaun TSP Solver for Constrained Traveling Salesman and Vehicle Routing Problems”, December 2017
[92] Keld Helsgaun. General k-opt submoves for the Lin–Kernighan TSP heuristic. Mathematical Programming Computation, 1(2):119–163, 2009

[93] S. Lin, B. W. Kernighan, “An Effective Heuristic Algorithm for the Traveling-Salesman Problem”, Apr 1973, doi: 10.1287/opre.21.2.498
[94] Florian Arnold, Kenneth Sörensen, “Knowledge-guided local search for the vehicle routing problem”, May 2019, doi: 10.1016/j.cor.2019.01.002,pp 32-46
[95] Jean-Yves Potvin, Jean-Marc Rousseau, “A parallel route building algorithm for the vehicle routing and scheduling problem with time windows ”, May 1993, doi: 10.1016/0377-2217(93)90221-8, pp 331-340
[96] Refael Hassin, Ariel Keinan, “Greedy heuristics with regret, with application to the cheapest insertion algorithm for the TSP”, March 2008, doi: 10.1016/j.orl.2007.05.001,Pages 243-246
[97] Chris Voudouris, Edward Tsang, “Partial Constraint Satisfaction Problems and Guided Local Search”, 1996
[98] Christos Voudouris, Edward Tsang, “Guided local search and its application to the traveling salesman problem”, March 1999, doi: 10.1016/S0377-2217(98)00099-X, pp 469-499
[99] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, Yoshua Bengio, “Graph Attention Networks”, Oct 2017, doi: 10.48550/arXiv.1710.10903
[100] Sergey Ioffe, Christian Szegedy, “Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift ”, 2015, pp 448-456
[101] G. A. Croes, “A Method for Solving Traveling-Salesman Problems ”, Dec 1958, doi: 10.1287/opre.6.6.791
[102] Chaitanya K. Joshi, Thomas Laurent, Xavier Bresson, “An Efficient Graph Convolutional Network Technique for the Travelling Salesman Problem ”, Jun 2019, doi: 10.48550/arXiv.1906.01227
[103] Paulo R d O Costa, Jason Rhuggenaath, Yingqian Zhang, Alp Akcay, “Learning 2-opt Heuristics for the Traveling Salesman Problem via Deep Reinforcement Learning ”, 465-480, 2020
[104] Luca Accorsi, Andrea Lodi, Daniele Vigo, “Guidelines for the computational testing of machine learning approaches to vehicle routing problems ”, March 2022, doi: 10.1016/j.orl.2022.01.018, pp 229-234
[105] E. Groshev, A. Tamar, M. Goldstein, S. Srivastava, and P. Abbeel, “Learning Generalized Reactive Policies Using Deep Neural Networks ”, Jun 2018, doi: 10.1609/icaps.v28i1.13872
[106] Z. Li, Q. Chen, and V. Koltun, “Combinatorial optimization with graph convolutional networks and guided tree search ”, 2018
[107] M. Gasse, D. Chételat, N. Ferroni, L. Charlin, and A. Lodi, “Exact Combinatorial Optimization with Graph Convolutional Neural Networks ”, 2019
[108] ZHIHAO XING, SHIKUI TU, “A Graph Neural Network Assisted Monte Carlo Tree Search Approach to Traveling Salesman Problem ”, June 2020, doi: 10.1109/ACCESS.2020.3000236
[109] Will Hamilton, Zhitao Ying, Jure Leskovec, “Inductive Representation Learning on Large Graphs ”, pp 1024–1034
[110] Rémi Coulom, “Efficient Selectivity and Backup Operators in Monte-Carlo Tree Search ”, 2006, doi: 10.1007/978-3-540-75538-8_7,pp 72–83
[111] Levente Kocsis, Csaba Szepesvári, “Bandit Based Monte-Carlo Planning ”, 2006, doi: 10.1007/11871842, pp 282–293
[112] Christopher D. Rosin, “Multi-armed bandits with episode context ”, August 2011, doi: 10.1007/s10472-011-9258-6, pp. 203–230
[113] Tian Xie, Jeffrey C. Grossman, “ Crystal Graph Convolutional Neural Networks for an Accurate and Interpretable Prediction of Material Properties ”, April 2018, doi: 10.1103/PhysRevLett.120.145301
[114] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. van den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, D. Hassabis, “ Mastering the game of go with deep neural networks and tree search ”, Jan 2016, pp. 484–489, doi: 10.1038/nature16961
[115] Christopher D. Rosin, “Multi-armed bandits with episode context ”, Aug 2011, doi: 10.1007/s10472-011-9258-6, pp. 203–230

[116] D. S. Johnson, G. Gutin, L. A. McGeoch, A. Yeo, W. Zhang, A. Zverovitch, “Experimental Analysis of Heuristics for the ATSP”, 2001, doi: 10.1007/0-306-48213-4_10
[117] D. Applegate, R. Bixby, V. Chvatal, and W. Cook., “Concorde TSP Solver ”, 2006, https://www.math.uwaterloo.ca/tsp/concorde/
[118] Oriol Vinyals, Meire Fortunato, Navdeep Jaitly, “Pointer Networks ”, 2015, pp. 2692–2700
[119] Irwan Bello, Hieu Pham, Quoc V. Le, Mohammad Norouzi, Samy Bengio, “Neural Combinatorial Optimization with Reinforcement Learning ”, Jan 2017, doi: 10.48550/arXiv.1611.09940
[120] Diederik P. Kingma, Jimmy Ba, “Adam: A method for stochastic optimization ”, Jan 2017, Doi:10.48550/arXiv.1412.6980
[121] Hanjun Dai, Bo Dai, Le Song, “Discriminative embeddings of latent variable models for structured data”, 2016, PMLR 48:2702-2711
[122] Michaël Defferrard, Xavier Bresson, Pierre Vandergheynst, “Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering ”,2016, pp. 3844–3852
[123] Thomas N. Kipf, Max Welling, “Semi-Supervised Classification with Graph Convolutional Networks ”, Feb 2017, doi: 10.48550/arXiv.1609.02907

Πηγές Εικόνων

- Εικόνα Στη πρώτη σελίδα: https://medium.com/nybles/graph-neural-networks-a-brief-analysis-17d52e546684
- Σχήμα 2.1: https://www.sciencedirect.com/science/article/pii/S2666651021000012
- Σχήματα 4.1 – 4.6: https://ojs.aaai.org/index.php/AAAI/article/view/4399
- Σχήματα 4.7 – 4.8: https://www.sciencedirect.com/science/article/pii/S0952197620303286?casa_token=wkls5i5fSSIAAAAA:XH0wdo0YPnqcgR3o7pG9i_HCzZbWC3_1ZvmqYdsjTiSObO2c3J2VBiSf0_qHtdCccyryDyGp
- Σχήματα 4.9 – 4.10: https://arxiv.org/abs/2110.05291
- Σχήματα 4.11 – 4.13: https://ieeexplore.ieee.org/abstract/document/9109309

Άλλες Χρήσιμες Πηγές

- Το link για την βάση δεδομένων: https://simplemaps.com/data/gr-cities