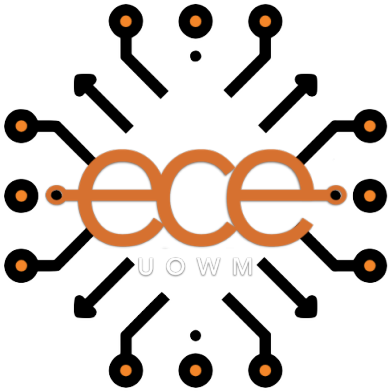


Πανεπιστήμιο Δυτικής Μακεδονίας

Τμήμα Ηλεκτρολόγων Μηχανικών & Μηχανικών Υπολογιστών

Μέθοδοι Μηχανικής Μάθησης για την Επίλυση Προβλημάτων Συνδυαστικής Βελτιστοποίησης (Machine Learning Algorithms for the Solution of Combinatorial Optimization Problems)

Χρήστος Τσακίρογλου (ΑΜ: 919)

Επιβλέπων Καθηγητής: Νικόλαος Πλόσκας

Εργαστήριο Ευφυών Συστημάτων & Βελτιστοποίησης

12 Οκτωβρίου 2021

Περίληψη

Το πρόβλημα του πλανόδιου πωλητή (TSP) είναι ένα από τα πιο ευρέως μελετημένα προβλήματα συνδυαστικής βελτιστοποίησης. Το πρόβλημα μπορεί να διατυπωθεί απλά ως εξής: ένας πλανόδιος πωλητής θέλει να επισκεφτεί κάθε πόλη από μία λίστα πόλεων ακριβώς μία φορά, και μετά να γυρίσει ξανά στην αρχική πόλη. Ποια είναι η μικρότερη πιθανή διαδρομή που μπορεί να ακολουθήσει; Για την επίλυση του προβλήματος έχουν επιστρατευθεί διάφορα είδη αλγορίθμων όπως ακριβείς, προσεγγιστικοί, ευρετικοί, μεθευρετικοί, ακόμα και νευρωνικά δίκτυα. Στην παρούσα διπλωματική γίνεται μελέτη ενός νευρωνικού δικτύου που προτάθηκε από τους Deudon et al. [9], το οποίο προβλέπει διαδρομές για προβλήματα ευκλείδειου tsp. Με στόχο να βελτιωθεί η διαδρομή, εφαρμόστηκαν οι βελτιωτικοί αλγόριθμοι 2opt, 3opt, simulated annealing, variable neighborhood search descent και ο κατασκευαστικός αλγόριθμος nearest neighbor. Μετά από πειράματα σε προβλήματα tsp από τη βιβλιοθήκη TSPLIB, αποδεικνύεται ότι οι βελτιωτικοί αλγόριθμοι επιφέρουν σημαντική βελτίωση στη διαδρομή που δίνει το νευρωνικό, φτάνοντας σε μερικές περιπτώσεις, πολύ κοντά στη βέλτιστη λύση.

Λέξεις κλειδιά: Πρόβλημα πλανόδιου πωλητή, Συνδυαστική βελτιστοποίηση, Νευρωνικά δίκτυα, Python.

Abstract

The travelling salesman problem (TSP) is one of the most widely studied combinatorial optimization problems. The problem can be expressed simply as follows: a salesman wants to visit each city from a list of cities exactly once, and then return to the original city. What is the minimal possible route that can be followed? Various types of algorithms such as exact, approximate, heuristic, meta-heuristic, and even neural networks have been employed to solve the problem. This dissertation studies a neural network proposed by Deudon et al. [9], which provides routes for Euclidean tsp problems. In order to improve the route, the 2opt, 3opt, simulated annealing, variable neighborhood search descent, and the nearest neighbor construction algorithm were implemented. After a computational study on tsp problems from the TSPLIB library, it turns out that the improvement algorithms show a significant improvement of the tour given by the neural network, approaching in some cases even the optimal solution.

Keywords: Traveling salesman problem, Combinatorial optimization, Neural networks, Python.

Δήλωση Πνευματικών Δικαιωμάτων

Δηλώνω ρητά ότι, σύμφωνα με το άρθρο 8 του Ν. 1599/1986 και τα άρθρα 2,4,6 παρ. 3 του Ν. 1256/1982, η παρούσα Διπλωματική Εργασία με τίτλο ”Μέθοδοι Μηχανικής Μάθησης για την Επίλυση Προβλημάτων Συνδυαστικής Βελτιστοποίησης (Machine Learning Algorithms for the Solution of Combinatorial Optimization Problems) ” καθώς και τα ηλεκτρονικά αρχεία και πηγαίοι κώδικες που αναπτύχθηκαν ή τροποποιήθηκαν στα πλαίσια αυτής της εργασίας και αναφέρονται ρητώς μέσα στο κείμενο που συνοδεύουν, και η οποία έχει εκπονηθεί στο Τμήμα Ηλεκτρολόγων Μηχανικών & Μηχανικών Υπολογιστών του Πανεπιστημίου Δυτικής Μακεδονίας, υπό την επίβλεψη του μέλους του Τμήματος κ. Νικολάου Πλόσκα αποτελεί αποκλειστικά προϊόν προσωπικής εργασίας και δεν προσβάλλει κάθε μορφής πνευματικά δικαιώματα τρίτων και δεν είναι προϊόν μερικής ή ολικής αντιγραφής, οι πηγές δε που χρησιμοποιήθηκαν περιορίζονται στις βιβλιογραφικές αναφορές και μόνον. Τα σημεία όπου έχω χρησιμοποιήσει ιδέες, κείμενο, αρχεία ή / και πηγές άλλων συγγραφέων, αναφέρονται ευδιάκριτα στο κείμενο με την κατάλληλη παραπομπή και η σχετική αναφορά περιλαμβάνεται στο τμήμα των βιβλιογραφικών αναφορών με πλήρη περιγραφή.

Απαγορεύεται η αντιγραφή, αποθήκευση και διανομή της παρούσας εργασίας, εξ ολοκλήρου ή τμήματος αυτής, για εμπορικό σκοπό. Επιτρέπεται η ανατύπωση, αποθήκευση και διανομή για σκοπό μη κερδοσκοπικό, εκπαιδευτικής ή ερευνητικής φύσης, υπό την προϋπόθεση να αναφέρεται η πηγή προέλευσης και να διατηρείται το παρόν μήνυμα. Ερωτήματα που αφορούν τη χρήση της εργασίας για κερδοσκοπικό σκοπό πρέπει να απευθύνονται προς τον συγγραφέα. Οι απόψεις και τα συμπεράσματα που περιέχονται σε αυτό το έγγραφο εκφράζουν τον συγγραφέα και μόνο.

Υπογραφή Φοιτητή

Περιεχόμενα

Κατάλογος σχημάτων

Κατάλογος αλγορίθμων

Κατάλογος πινάκων

Κεφάλαιο 1

Εισαγωγή

Σήμερα ένα από τα πιο βασικά προβλήματα που υπάρχει στην επιστήμη των υπολογιστών είναι αυτό της βέλτιστης δρομολόγησης. Η βέλτιστη δρομολόγηση βρίσκει εφαρμογές στους πιο απλούς τομείς, όπως τη διαδρομή που θα ακολουθήσει ένα φορτηγό για τη διανομή προϊόντων γρήγορα και οικονομικά, αλλά και σε πιο σύνθετες εφαρμογές, όπως για παράδειγμα τη δρομολόγηση drones για παράδοση προϊόντων σε πελάτες σε κοντινές περιοχές. Με τη σημερινή τεχνολογία και τη χρήση τεχνητών νευρωνικών δικτύων δίνεται η δυνατότητα επίλυσης πολύπλοκων προβλημάτων δρομολόγησης σε ικανοποιητικό χρόνο.

1.1 Ορισμός του προβλήματος

Ας ξεκινήσουμε με ένα παράδειγμα, με σκοπό να γίνει το πρόβλημα πιο κατανοητό από τον αναγνώστη. Ας πάρουμε το παράδειγμα του παιχνιδιού ”ένωσε τις τελείες”. Σε αυτό το παιχνίδι, ο σκοπός είναι ο παίκτης να ενώσει τις τελείες, περνώντας με το στυλό πάνω από κάθε τελεία, μόνο μια φορά, μέχρι να δημιουργηθεί στο χαρτί το επιθυμητό σχέδιο και να επιστρέψει στην τελεία από την οποία ξεκίνησε. Ας φανταστούμε λοιπόν, ότι αυτές οι τελείες αντιπροσωπεύουν κάτι άλλο, όπως για παράδειγμα πόλεις. Το παιχνίδι πλέον θα αλλάξει, προσθέτοντας κάποια νέα δεδομένα.

Για αρχή κάθε απόσταση μεταξύ των τελειών (πόλεων) παίρνει μια τιμή. Στη συνέχεια, δίνονται οι νέοι κανόνες. Ο πρώτος κανόνας είναι ότι ο παίκτης θα πρέπει να ενώσει τις τελείες περνώντας μόνο μια φορά από κάθε μία και ο δεύτερος, να προσπαθήσει να φτιάξει ένα σχέδιο, το οποίο έχει τη μικρότερη δυνατή τιμή απόστασης. Αυτό είναι ένας πολύ απλός ορισμός στο πρόβλημα το οποίο ονομάζεται

πρόβλημα του πλανόδιου πωλητή.

Η ευκολία στην επίλυση του παραπάνω προβλήματος είναι άμεσα συνδεδεμένη με τον αριθμό από πόλεις που θα είναι διαθέσιμος. Ένας άνθρωπος δε θα δυσκολευτεί να βρει λύση σε ένα σύνολο πέντε πόλεων. Τι γίνεται όμως όταν αυτό το σύνολο γίνει αισθητά μεγαλύτερο, όπως για παράδειγμα δέκα χιλιάδες πόλεις; Εκεί η λύση είναι αδύνατο να βρεθεί από έναν άνθρωπο και χρειάζεται η επιστράτευση ενός συστήματος μεγαλύτερης υπολογιστικής ισχύος (υπολογιστής). Ακόμα όμως και για έναν υπολογιστή δεν είναι εύκολη, και όπως μέχρι τώρα γνωρίζουμε αδύνατη, να βρει την καλύτερη λύση σε τεράστια σύνολα πόλεων (εκατομμύρια πόλεις). Για αυτόν τον λόγο έχουν δημιουργηθεί διάφοροι τρόποι υπολογιστικής επίλυσης του συγκεκριμένου προβλήματος, όπως νευρωνικά δίκτυα για πρόβλεψη, προσεγγιστικοί αλγόριθμοι και ευρετικές μέθοδοι που η σωστή εφαρμογή τους μπορεί να βελτιώσει τη λύση σε πάρα πολύ μεγάλο ποσοστό και να φτάσει το σύστημα, αν όχι στη βέλτιστη, πάρα πολύ κοντά σε αυτήν.

1.2 Κίνητρα και στόχοι υλοποίησης

Το πρόβλημα του πλανόδιου πωλητή (TSP) είναι ένα από τα πιο ευρέως μελετημένα προβλήματα συνδυαστικής βελτιστοποίησης. Το γενικό κίνητρο είναι η δημιουργία ενός αξιόπιστου προγράμματος, ικανού να προβλέψει και να βελτιστοποιήσει τις διαδρομές που θα ακολουθήσει ο πωλητής μεταξύ των σημείων που δίνονται, με σκοπό να κάνει πιο εύκολη και αποτελεσματική την εφαρμογή του προβλήματος στην καθημερινότητα.

Στόχος της παρούσας διπλωματικής εργασίας είναι να μελετήσει τη χρήση τεχνητών νευρωνικών δικτύων για την επίλυση προβλημάτων συνδυαστικής βελτιστοποίησης. Επίσης, στοχεύει στην εκπαίδευση μοντέλων για τη δημιουργία προβλέψεων βέλτιστων διαδρομών και στην εφαρμογή τους σε ευκλείδεια προβλήματα πλανόδιου πωλητή. Στην εργασία αυτή υιοθετούμε την υλοποίηση των Deudon et al. [9], οι οποίοι χρησιμοποιούν μία πρόβλεψη διαδρομής ενός πλανόδιου πωλητή, που δίνει ως έξοδο ένα τεχνητό νευρωνικό δίκτυο, βάση του συνόλου από πόλεις που θα εισάγουμε και τη βελτιώνουν. Ο στόχος της παρούσας διπλωματικής είναι η χρήση αλγορίθμων βελτιστοποίησης πάνω στην πρόβλεψη του νευρωνικού δικτύου με σκοπό τη μείωση της απόστασης διαδρομής που ταξιδεύει ο πωλητής.

Επιπρόσθετα, γίνεται μια ανάλυση αλγορίθμων, ευρετικών και μεθευρετικών μεθόδων συνδυαστικής βελτιστοποίησης, οι οποίοι μετά από μελέτη, εφαρμόζονται στην έξοδο του μοντέλου, με σκοπό την επίτευξη καλύτερου αποτελέσματος. Τέλος, παρουσιάζονται δεδομένα και μετρήσεις της αποτελεσματικότητας της εφαρμογής αυτών των αλγορίθμων στις προβλέψεις του μοντέλου.

1.3 Διάρθρωση κειμένου

Τα υπόλοιπα κεφάλαια της παρούσας διπλωματικής οργανώνονται ως εξής: Στο Κεφάλαιο 2 θα αναπτυχθούν έννοιες της τεχνητής νοημοσύνης, της μηχανικής μάθησης και των τεχνητών νευρωνικών δικτύων, καθώς και τα είδη και οι εφαρμογές των παραπάνω.

Στο Κεφάλαιο 3, θα αναπτυχθεί αρχικά η έννοια της συνδυαστικής βελτιστοποίησης. Στη συνέχεια γίνεται μια πιο ειδική αναφορά στο πρόβλημα του πλανόδιου πωλητή, τη συνεισφορά των νευρωνικών δικτύων και της μηχανικής μάθησης στην επίλυσή του, όπως επίσης αναλύονται και οι πιο συνηθισμένοι αλγόριθμοι που εφαρμόζονται στο πρόβλημα.

Στο Κεφάλαιο 4 θα γίνει η επεξήγηση του νευρωνικού δικτύου που χρησιμοποιήθηκε, η γενική ιδέα του, ο τρόπος μηχανικής μάθησης που εφαρμόστηκε, καθώς και θα επεξηγηθούν οι παράμετροί του. Εξηγούνται οι παράμετροι και τα δεδομένα της εκπαίδευσης που δοκιμάστηκαν. Επίσης, θα γίνει αναλυτική αναφορά στους αλγορίθμους που εφαρμόστηκαν στην έξοδο του νευρωνικού δικτύου, δίνοντας το θεωρητικό υπόβαθρο και εξηγώντας την υλοποίησή τους. Οι αλγόριθμοι αυτοί είναι οι: *2-opt*, *3-opt*, *nearest neighbour*, *variable neighborhood descent search*, *simulated annealing*.

Στο Κεφάλαιο 5 γίνεται ανάλυση των συνόλων δεδομένων, το πώς και από πού αποκτήθηκαν και διάφορες αλλαγές που έχουν υποστεί τα δεδομένα. Δίνονται οι βέλτιστες λύσεις στα προβλήματα και η περιγραφή του προγράμματος που χρησιμοποιήθηκε για την εύρεσή τους. Στο ίδιο κεφάλαιο παρουσιάζονται συγκριτικά στοιχεία απόδοσης και ακρίβειας των αλγορίθμων που μελετάει η παρούσα διπλωματική εργασία.

Τέλος, στο Κεφάλαιο 6, αναφέρονται τα συμπεράσματα που παράχθηκαν από αυτήν τη διπλωματική εργασία.

Κεφάλαιο 2

Έννοιες και ορισμοί

2.1 Τεχνητή νοημοσύνη

Τεχνητή νοημοσύνη είναι η επιστήμη και η τεχνοτροπία που επιτρέπει τη σχεδίαση και υλοποίηση προγραμμάτων και μηχανών που μπορούν να μιμηθούν τις ανθρώπινες γνωστικές ικανότητες, εμφανίζοντας έτσι χαρακτηριστικά που συνήθως αποδίδουμε σε ανθρώπινη συμπεριφορά, όπως η κατανόηση της φυσικής γλώσσας, η μάθηση και η επίλυση προβλημάτων. Ο Alan Turing όρισε την Τεχνητή Νοημοσύνη (Artificial Intelligence - AI) ως εξής: ”Αν υπάρχει μηχανή πίσω από μια κουρτίνα και ένας άνθρωπος αλληλεπιδρά μαζί της (ήχο, μέσω πληκτρολόγησης κα.) και αν αυτός ο άνθρωπος αισθάνεται ότι είναι σαν να αλληλεπιδρά με έναν άλλο άνθρωπο, τότε το μηχάνημα είναι τεχνητά έξυπνο” [23]. Ο ορισμός αυτός έγινε γνωστός με την ονομασία το τεστ του Turing (Turing’s test). Υπάρχουν δύο πτυχές τεχνητής νουμοσήνης που εξετάζονται από την άποψη ανθρώπινης συμπεριφοράς. Η μία είναι ότι η μηχανή είναι έξυπνη και ικανή να επικοινωνήσει αλλά δε διαθέτει μηχανικές πτυχές και η άλλη να έχει φυσική αλληλεπίδραση με το περιβάλλον χρησιμοποιώντας ανθρωπόμορφους μηχανισμούς. Η δεύτερη μορφή κατατάσσεται στην επιστήμη της ρομποτικής.

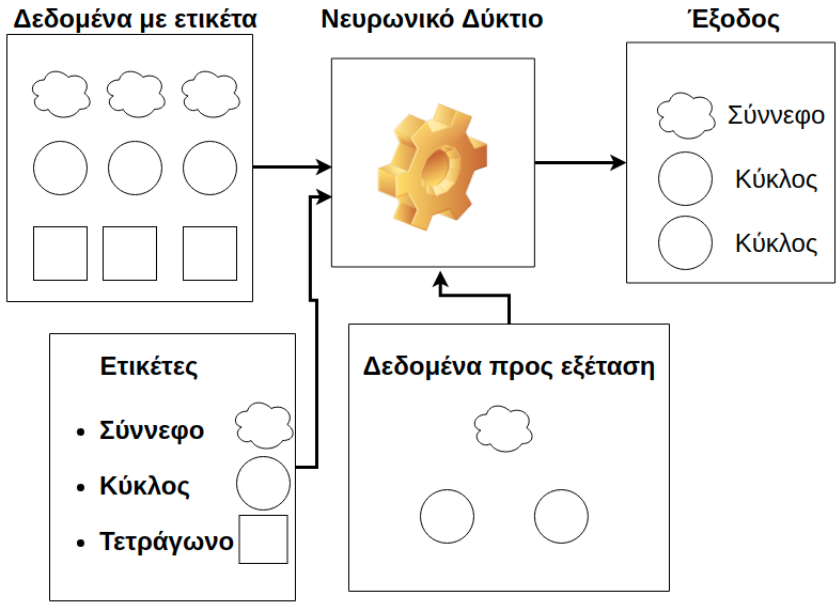
2.2 Τεχνητά νευρωνικά δίκτυα

Τα τεχνητά νευρωνικά δίκτυα [43] προσπαθούν να προσομοιώσουν τη λειτουργία του ανθρώπινου εγκεφάλου και του βιολογικού νευρωνικού δικτύου. Ένα τεχνητό νευρωνικό δίκτυο αποτελείται από ένα σύνολο τεχνητών νευρώνων που αλληλεπιδρούν και συνδέονται μεταξύ τους με συνάψεις. Κάθε ζεύγος νευρώνων έχει

διαφορετικό βαθμό αλληλεπίδρασης που καθορίζεται από τα συναπτικά βάρη τα οποία κατά της αλληλεπίδρασης του δικτύου με το περιβάλλον μεταβάλλονται συνεχώς με αποτέλεσμα να ενδυναμώνεται ή να αποδυναμώνεται η ισχύς του κάθε δεσμού. Όλη η γνώση που λαμβάνει λοιπόν το δίκτυο κωδικοποιείται στα συναπτικά βάρη τα οποία δίνουν στο δίκτυο την ικανότητα εξέλιξης και προσαρμογής στο περιβάλλον.

Οι πιο συχνοί τρόποι με τους οποίους μπορούμε να εκπαιδεύσουμε ένα νευρωνικό δίκτυο είναι με εποπτεία (supervised), χωρίς εποπτεία (unsupervised) και με ενισχυτική μάθηση (reinforcement learning). Όταν έχουμε εκπαίδευση με εποπτεία στο δίκτυο δίνεται ένα σύνολο γνωστών καταστάσεων στις οποίες μπορεί να ανατρέξει το δίκτυο αλλά και των αποτελεσμάτων που θέλουμε να δίνει για τις καταστάσεις αυτές. Για να μπορέσει το δίκτυο να μάθει από αυτά τα παραδείγματα χρησιμοποιούμε έναν αλγόριθμο εκπαίδευσης επιλεγμένο ανάλογα με το πρόβλημα αλλά και τη δομή του δικτύου (Σχήμα 2.1).

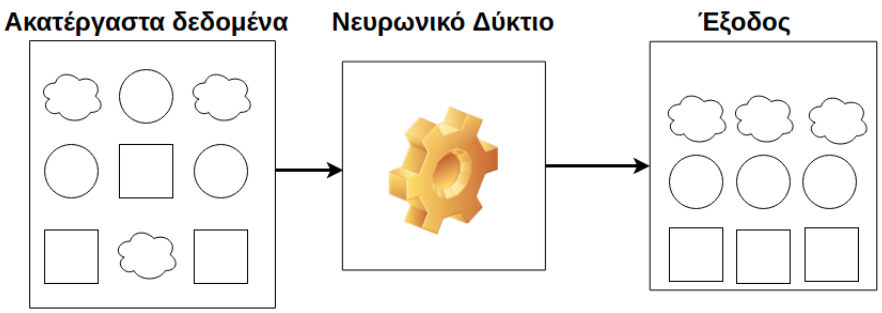
Σχήμα 2.1: Εκπαίδευση με εποπτεία



Στην εκπαίδευση χωρίς εποπτεία τροφοδοτούμε στο δίκτυο δεδομένα και αυτό πρέπει να τα χωρίσει σε ομάδες αναγνωρίζοντας ομοιότητες και μοτίβα στα δεδομένα. Η διαδικασία ολοκληρώνεται όταν δεν υπάρχει μεταβολή στην ταξινόμηση των δεδομένων (Σχήμα 2.2).

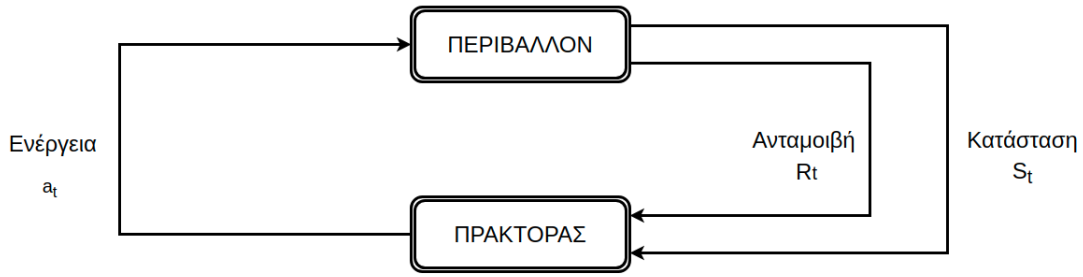
Στην εκπαίδευση με ενισχυτική μάθηση το δίκτυο αλληλεπιδρά με ένα δυναμικό

Σχήμα 2.2: Εκπαίδευση χωρίς εποπτεία



περιβάλλον το οποίο το ανατροφοδοτεί με δεδομένα της μορφής ”επιβράβευσης” και ”τιμωρίας” χωρίς να υπάρχει κάποιος ρητός στόχος από τον επιβλέπων (Σχήμα 2.3).

Σχήμα 2.3: Εκπαίδευση με ενισχυτική μάθηση



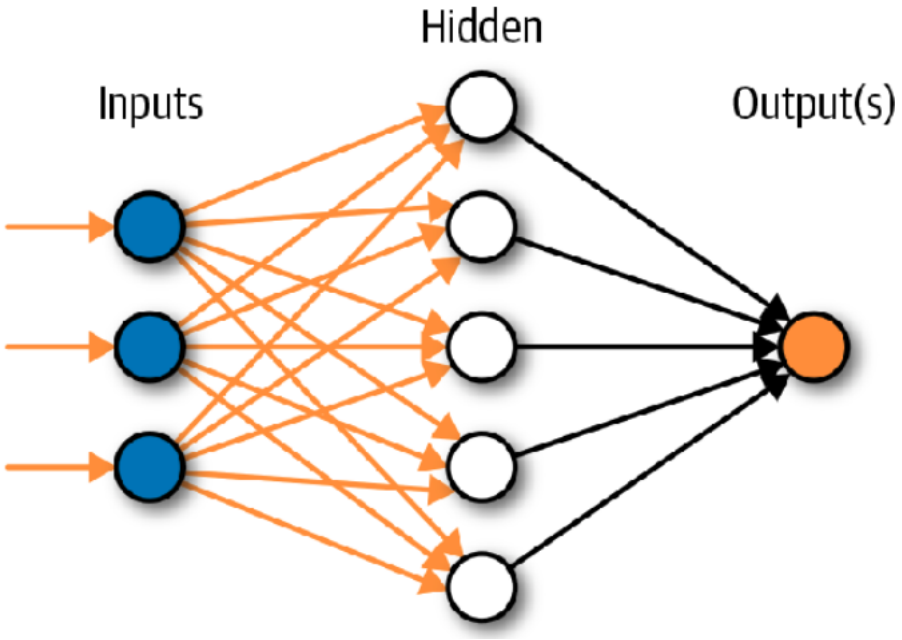
Όπως βλέπουμε και στο Σχήμα 2.4, το νευρωνικό δίκτυο οργανώνεται σε επίπεδα που λειτουργούν παράλληλα. Τα επίπεδα αυτά είναι το επίπεδο εισόδου (input layer) που αντιπροσωπεύει τους δενδρίτες στο νευρικό δίκτυο του ανθρώπινου εγκεφάλου, το κρυφό επίπεδο (hidden layer) που αντιπροσωπεύει το κυτταρικό σώμα και βρίσκεται ανάμεσα στο επίπεδο εισόδου και στο επίπεδο εξόδου (output layer), το οποίο αντιπροσωπεύει τις συναπτικές εξόδους στον εγκέφαλο. Στο κρυφό στρώμα οι τεχνητοί νευρώνες λαμβάνουν ένα σύνολο εισόδων με βάση το συναπτικό βάρος, το οποίο είναι το πλάτος ή η δύναμη μιας σύνδεσης μεταξύ των κόμβων (νευρώνων). Αυτές οι σταθμισμένες είσοδοι παράγουν μια έξοδο μέσω μιας συνάρτησης μεταφοράς στο επίπεδο εξόδου. Η έξοδος h_i του νευρώνα i στο κρυφό επίπεδο

δίνεται από τον μαθηματικό τύπο

$$h_i = \sigma(\sum_{j=1}^{N} V_{ij} x_j + T_i^{hid})$$

όπου $\sigma()$ ονομάζεται η συνάρτηση ενεργοποίησης (ή μεταφοράς), N ο αριθμός νευρώνων εισόδου, V_{ij} τα βάρη, x_j οι είσοδοι στους νευρώνες εισόδου και T_i^{hid} οι όροι του κατωφλιού των κρυφών νευρώνων.

Σχήμα 2.4: Λειτουργία τεχνητού νευρωνικού δικτύου



Χαρακτηριστικά: Μερικά από τα χαρακτηριστικά των νευρωνικών δικτύων είναι:

- Προσαρμοστική μάθηση: διαμορφώνουν μη γραμμικές και πολύπλοκες σχέσεις και βασίζονται σε προηγούμενες γνώσεις.
- Αυτοοργάνωση: Η ικανότητα ομαδοποίησης και ταξινόμησης τεράστιων ποσοτήτων δεδομένων.
- Λειτουργία σε πραγματικό χρόνο: Τα νευρωνικά δίκτυα μπορούν να δώσουν απαντήσεις σε πραγματικό χρόνο
- Πρόγνωση: Η ικανότητα πρόβλεψης του νευρωνικού δικτύου βάσει μοντέλων.

- Ανοχή σε σφάλματα: Όταν χαθούν ή λείπουν σημαντικά τμήματα ενός δικτύου, τα νευρωνικά δίκτυα μπορούν να συμπληρώσουν τα κενά.

Λειτουργίες: Οι λειτουργίες των νευρωνικών δικτύων αποτελούνται από:

- Ταξινόμηση (Classification): Τα νευρωνικά δίκτυα οργανώνουν μοτίβα ή σύνολα δεδομένων σε προκαθορισμένες κλάσεις.
- Πρόβλεψη: Παράγουν την αναμενόμενη έξοδο από από τα δεδομένα εισόδου.
- Ομαδοποίηση (Clustering): Προσδιορίζουν ένα μοναδικό χαρακτηριστικό των δεδομένων και τα ταξινομούν, χωρίς καμία γνώση προηγούμενων δεδομένων.
- Συσχετισμός: Τα νευρωνικά δίκτυα μπορούν να εκπαιδευτούν να ''θυμούνται'' μοτίβα. Όταν εμφανίζεται μια άγνωστη έκδοση ενός μοτίβου, το δίκτυο το συσχετίζει με την πιο κοντινή έκδοση στη μνήμη του και επιστρέφει στην τελευταία.

2.3 Μηχανική μάθηση

Η μηχανική μάθηση (Machine Learning - ML) [46] είναι μία υποκατηγορία της τεχνητής νοημοσύνης για τη δημιουργία αποτελεσματικών αλγορίθμων για πρόβλεψη αποτελεσμάτων. Ο όρος ''μηχανική μάθηση'' μπορεί να οριστεί ως ένα πρόγραμμα υπολογιστή το οποίο δημιουργεί ένα μαθηματικό μοντέλο βασισμένο σε δείγματα δεδομένων, γνωστά ως «δεδομένα εκπαίδευσης», προκειμένου να προβλέψει ή να πάρει αποφάσεις, χωρίς να έχει προγραμματιστεί ρητά το μοντέλο από τον συγγραφέα. Η επιτυχία του αλγορίθμου να μάθει εξαρτάται από τα δεδομένα εκπαίδευσης, αυτό καθιστά τη μηχανική μάθηση άμεσα συνδεδεμένη με την ανάλυση δεδομένων και τη στατιστική. Γενικότερα οι τεχνικές μηχανικής μάθησης είναι καθοδηγούμενες από δεδομένα και συνδυάζουν τις βασικές αρχές της επιστήμης της πληροφορικής με ιδέες από πιθανότητες, στατιστική και βελτιστοποίηση.

2.3.1 Είδη μηχανικής μάθησης

Οι τρόποι που μαθαίνει ένα πρόγραμμα με μηχανική μάθηση κατηγοριοποιούνται στους εξής:

Επιβλεπόμενη μάθηση (Supervised learning): Το πρόγραμμα δέχεται παραδειγματικές εισόδους και επιθυμητά αποτελέσματα από τον επιβλέπων και πρέπει να μάθει έναν γενικό κανόνα που να αντιστοιχεί τις εισόδους με τα αποτελέσματα.

Μη επιβλεπόμενη μάθηση (Unsupervised learning): Το πρόγραμμα τροφοδοτείται από δεδομένα εισόδου χωρίς κάποια προηγούμενη εμπειρία και ως σκοπό έχει να βρει τα μοτίβα μεταξύ των δεδομένων και να τα κατηγοριοποιήσει ανάλογα.

Ημι-επιβλεπόμενη μάθηση (Semi-supervised learning): Το πρόγραμμα δέχεται ένα μικρό σύνολο από παραδειγματικά δεδομένα και τα αποτελέσματά τους και ένα πολύ μεγαλύτερο σύνολο από δεδομένα εισόδου χωρίς καμία συσχέτιση.

Ενισχυτική μάθηση (Reinforcement learning): Το πρόγραμμα αλληλεπιδρά με ένα δυναμικό περιβάλλον το οποίο το ανατροφοδοτεί με δεδομένα της μορφής "επιβράβευσης" και "τιμωρίας" χωρίς να υπάρχει κάποιος ρητός στόχος από τον επιβλέπων. Αυτή η ανατροφοδότηση μεταξύ του συστήματος μάθησης και της εμπειρίας αλληλεπίδρασης είναι χρήσιμη για τη βελτίωση της απόδοσης στην εργασία που μαθαίνεται.

Μάθηση μεταφοράς (Transfer learning): Σε πολλές εφαρμογές στον πραγματικό κόσμο αλλάζει η διανομή των δεδομένων η ακόμα και τα δεδομένα καθίστανται ξεπερασμένα οπότε είναι αναγκαία μια μάθηση μεταφοράς που παίρνει δεδομένα από την πηγή και τα μεταφέρει στον στόχο.

2.3.2 Εφαρμογές μηχανικής μάθησης

Με μηχανική μάθηση δίνεται λύση σε πολλά προβλήματα και έχει πολυάριθμες πρακτικές εφαρμογές, κάποιες από αυτές είναι οι ακόλουθες [35]:

- Κατηγοριοποίηση κειμένου ή αρχείων: κατηγοριοποιεί αν κάποια σελίδα είναι ακατάλληλη, εντοπίζει ανεπιθύμητα (spam) email.
- Εφαρμογές επεξεργασίας ομιλίας: Περιέχει αναγνώριση φωνής, σύνθεση ομιλίας, φωνητική επαλήθευση και άλλα.

- Εφαρμογές όρασης υπολογιστών: Περιέχει αναγνώριση και ταυτοποίηση αντικειμένων, αναγνώριση προσώπου.
- Υπολογιστικές βιολογικές εφαρμογές: Περιέχει ανάλυση γονιδιακών και πρωτεϊνικών δικτύων, πρόβλεψη λειτουργίας πρωτεϊνών.
- Διάφορες άλλες εφαρμογές, όπως να μάθει να παίζει παιχνίδια (σκάκι), να αναγνωρίζει απάτες, οδήγηση δίχως επιβλέπων, μηχανές αναζήτησης και πολλά άλλα.

Κεφάλαιο 3

Το πρόβλημα του πλανόδιου πωλητή

3.1 Συνδυαστική βελτιστοποίηση

Πολλά προβλήματα πρακτικής αλλά και θεωρητικής σημασίας αφορούν την εύρεση της καλύτερης σύνθεσης ή το καλύτερο σύνολο παραμέτρων για να επιτευχθεί ένας στόχος. Τις τελευταίες δεκαετίες έχουν προκύψει αρκετά τέτοια προβλήματα, με αντίστοιχη συλλογή από τρόπους και τεχνικές επίλυσης. Κάθε πρόβλημα που καλείται να επιλέξει μια λύση από ένα πεπερασμένο σύνολο από πιθανές λύσεις ονομάζεται συνδυαστικό. Τα προβλήματα βελτιστοποίησης γενικά διατυπώνονται ως:

$$minimize f(x)$$

$$x \in F$$

όπου f μια συνάρτηση στόχου, F η εφικτή περιοχή που ικανοποιεί όλους τους περιορισμούς που δόθηκαν και μια λύση $x \in F$ εφικτή λύση. Αν η F έχει συνδυαστικά χαρακτηριστικά τότε το πρόβλημα ονομάζεται πρόβλημα συνδυαστικής βελτιστοποίησης [44]. Η συνδυαστική βελτιστοποίηση μπορεί να θεωρηθεί ως η μελέτη προβλημάτων βελτιστοποίησης, μόνο με πεπερασμένο αριθμό πιθανών λύσεων.

Για ορισμένους μαθηματικούς, η μείωση ενός άπειρου προβλήματος σε ένα πεπερασμένο, είναι το μόνο σημαντικό βήμα. Τα υπόλοιπα είναι "απλώς συνδυαστικά", επιλύσιμα με την αρχή της εξαντλητικής αναζήτησης, άρα μη μαθηματικού ενδιαφέροντος. Ωστόσο, αν και η εξαντλητική αναζήτηση είναι όντως περιορισμένου μαθηματικού ενδιαφέροντος, είναι επίσης και περιορισμένης χρήσης, εκτός εάν ο πεπερασμένος αριθμός των πιθανών λύσεων που εξαντλούμε είναι αρκετά μικρός.

Αν κάποιος θέλει να συνοψίσει τη μελέτη της συνδυαστικής βελτιστοποίησης σε δύο μόνο ερωτήσεις, αυτές είναι οι εξής: Είναι δυνατόν να βρεθούν βέλτιστες λύσεις χωρίς να χρησιμοποιηθεί ο εκθετικός (χειρότερος) χρόνος που απαιτείται από την εξαντλητική αναζήτηση και αν ναι, πώς. Οι απαντήσεις σε αυτές τις ερωτήσεις δεν είναι μόνο θέμα προγραμματιστικής τεχνικής αλλά περιλαμβάνουν και κρίσιμους μαθηματικούς τρόπους. Μέχρι πρόσφατα ωστόσο, αγνοούνταν ως επί το πλείστων στα παραδοσιακά προγράμματα σπουδών μαθηματικών, αντίθετα είχαν αφεθεί σε δύο σχετικά νέους κλάδους: την Επιχειρησιακή Έρευνα (Operations Research - OR) και την Επιστήμη των Υπολογιστών (Computer Science - CS). Αυτοί οι δύο κλάδοι πλησίασαν τη συνδυαστική βελτιστοποίηση από διαφορετικές κατευθύνσεις.

Στην επιστήμη των υπολογιστών έχουμε τη θεωρία της πληρότητας-NP που αναπτύχθηκε από τους Steven Cook και Richard Karp στις αρχές της δεκαετίας του 1970. Αυτή η θεωρία ασχολείται με την εκ φύσεως δυσκολία τον προβλημάτων και βασίζεται σε μαθηματικές τεχνικές για την επίδειξη της υπολογιστικής ισοδυναμίας τους. Εντοπίζει τα προβλήματα για τα οποία η εξαντλητική αναζήτηση είναι αναπόφευκτη (NP-Hard), σε αντίθεση με αυτά για τα οποία μπορούν να δημιουργηθούν ουσιαστικές βελτιώσεις στην εξαντλητική αναζήτηση, δηλαδή που μπορούν να επιλυθούν με αλγορίθμους που εκτελούνται εγγυημένα σε χρόνο περιορισμένο, από μία πολυωνυμική συνάρτηση (και όχι εκθετική) του μεγέθους του παραδείγματος [38]. Ένα είδος NP-Hard προβλήματος είναι και το πρόβλημα του πλανόδιου πωλητή που εξετάζεται στην παρούσα διπλωματική.

3.2 Ορισμός

Το πρόβλημα του πλανόδιου πωλητή (Traveling Salesman Problem - TSP) [24] λέγεται ότι πρωτομελετήθηκε από τον Karl Menger στην Βιέννη και στο Harvard και αργότερα προωθήθηκε από τους Hassler, Whitney και Merrill σε ένα σεμινάριο στο Πανεπιστήμιο του Princeton.

Το πρόβλημα μπορεί να διατυπωθεί με λίγα λόγια ως εξής: εάν ένας πλανόδιος πωλητής θέλει να επισκεφτεί κάθε πόλη από μία λίστα c πόλεων ακριβώς μία φορά, και μετά να γυρίσει ξανά στην αρχική πόλη, ποια είναι η μικρότερη πιθανή διαδρομή που μπορεί να ακολουθήσει; Το πρόβλημα αυτό αντιπροσωπεύει μια κλάση προβλημάτων που είναι γνωστά ως προβλήματα συνδυαστικής βελτιστοποίησης και

ανήκει στην κατηγορία NP-complete [37] και όπως αποδείχτηκε από τον Karp το 1972 NP-Hard [25].

Το πρόβλημα διαχωρίζεται σε τρεις βασικές κατηγορίες το συμμετρικό, το ασύμμετρο και των πολλαπλών πωλητών. Στο συμμετρικό, η απόσταση μεταξύ δύο κορυφών πρέπει να ισούται αμφίδρομα $d_{ij} = d_{ji}$. Εάν έστω και μία απόσταση μεταξύ κορυφών είναι διαφορετική $d_{ij} \neq d_{ji}$ τότε είναι ασύμμετρο. Στην περίπτωση των πολλαπλών πωλητών έχουμε πολλούς πωλητές τοποθετημένους σε μία πόλη-αρχή και πρέπει να βρούμε ένα μονοπάτι για τον καθένα, έτσι ώστε να περάσει από ενδιάμεσες πόλεις και να φτάσει στον τερματισμό.

Ως μαθηματικό μοντέλο θεωρούμε ένα γράφημα $G = (V, E)$ όπου $V = v_1, v_2, ..., v_n$ ένα σύνολο πόλεων, $E = (i, j) : i, j \in V$ ένα σύνολο ακμών και F την οικογένεια των κύκλων Χάμιλτον στο G. Για κάθε κορυφή $e \in E$ ορίζεται ένα κόστος. Για το πρόβλημα αυτό, χρειάζεται να βρεθεί μια διαδρομή (κύκλος Χάμιλτον), τέτοια ώστε το άθροισμα των κοστών των κορυφών να είναι το ελάχιστο δυνατό. Αν το πρόβλημα είναι συμμετρικό, οι πόλεις δίνονται από τις συντεταγμένες τους και η απόσταση μεταξύ δύο πόλεων είναι η ευκλείδεια τότε έχουμε ένα ευκλείδειο TSP. Θεωρώντας d_{ji} την απόσταση της j-πολης με την απόσταση της i-πολης προσδιόρισε το x_{ji}, $j = 1, 2, ...n$, $i = 1, 2, ...n$ τέτοιο ώστε

$$\left.\begin{array}{ll} \text{minimize} & Z = \sum_{j=1}^{n} \sum_{i=1}^{n} x_{ji} d_{ji} \\ \text{s.t.} & \sum_{j=1}^{n} x_{ji} = 1, \text{ για } i = 1, 2, ...n \\ & \sum_{i=1}^{n} x_{ji} = 1, \text{ για } j = 1, 2, ...n \end{array}\right\}$$

όπου $x_{ji} = 1$ αν ο πωλητής ταξιδέψει από την πόλη j στην i αλλιώς $x_{ji} = 0$.

Έστω $(x_1, x_2, ...x_n, x_1)$ μια ολοκληρωμένη διαδρομή του πωλητή όπου $x_j \in 1, 2, ...n$ και όλα τα x_j είναι διακριτά, δηλαδή $(x_1, x_2, ...x_n, x_1)$ είναι η ακολουθία των πόλεων που επισκέφθηκε ο πωλητής το μοντέλο μειώνεται σε [26]:

$$\left.\begin{array}{ll} \text{Καθόρισε μια πλήρη διαδρομή} & (x_1, x_2, ...x_n, x_1) \\ \text{για να ελαχιστοποιήσει} & \sum_{j=1}^{n} (d_{x_j} d_{x_{j+1}} + d_{x_n} d_{x_1}) \end{array}\right\}$$

Το πρόβλημα βρίσκει εφαρμογή και σε άλλους τομείς όπως τη δρομολόγηση οχημάτων, τις αποστολές παραγγελιών από αποθήκες, τη βέλτιστη διαδρομή μιας

στρατιωτικής αποστολής, ακόμα και τη σχεδίαση παγκόσμιου δορυφορικού συστήματος για δίκτυα τοπογράφησης [33].

3.3 Τρόποι επίλυσης

Όπως προαναφέρθηκε, το πρόβλημα του πλανόδιου πωλητή ανήκει στην κλάση προβλημάτων συνδυαστικής βελτιστοποίησης και είναι στην κατηγορία NP-Hard. Οι πιο συνηθισμένοι τρόποι για επίλυση προβλημάτων NP-Hard είναι οι εξής:

- Σχεδιάζοντας ακριβείς αλγορίθμους που δουλεύουν ικανοποιητικά μόνο για μικρού μεγέθους προβλήματα.
- Σχεδιάζοντας προσεγγιστικούς αλγόριθμους.
- Σχεδιάζοντας υποβέλτιστους ή αλλιώς ευρετικούς αλγορίθμους (heuristic) οι οποίοι παρέχουν προσεγγιστικές λύσεις.
- Εύρεση ειδικών ''υποπροβλημάτων'' για τα οποία είναι καλύτεροι ή ακριβέστεροι οι ευρετικοί αλγόριθμοι (μεθευρετικοί-metaheuristic).

3.3.1 Ακριβείς αλγόριθμοι

Η πιο άμεση λύση θα ήταν να γίνει δοκιμή όλων των συνδυασμών και να βρεθεί αυτή με το μικρότερο κόστος χρησιμοποιώντας αναζήτηση *ωμής βίας (brute-force search)*. Η χρονική πολυπλοκότητα μίας τέτοιας μεθόδου είναι $O(n!)$, όμως το 1961 ένας αλγόριθμος δυναμικού προγραμματισμού που αναπτύχθηκε από τον Held και Karp [18] καταφέρνει να τρέξει με χρονική πολυπλοκότητα $O(2^n n^2)$.

Επιπλέον κάποιες άλλες προσεγγίσεις περιλαμβάνουν διάφορους αλγόριθμους διακλάδωσης και οριοθέτησης *(bracnh and bound)*, προοδευτικούς αλγορίθμους βελτίωσης που χρησιμοποιούν τεχνικές που βασίζονται στον γραμμικό προγραμματισμό *(linear programming)* και υλοποιήσεις του αλγορίθμου branch and bound με γενική κοπή για συγκεκριμένα προβλήματα *(branch and cut)*.

Ο στόχος ενός αλγόριθμου branch and bound είναι να βρει μια τιμή x που μεγιστοποιεί ή ελαχιστοποιεί την τιμή μιας αντικειμενικής συνάρτησης $f(x)$, μεταξύ ορισμένων συνόλων S αποδεκτών ή υποψήφιων λύσεων. Μία τέτοια υλοποίηση ενός

αλγορίθμου branch and bound έγινε απο τους *Jagiełło και Grymin* [14] και περιέχει, δύο τροποποιήσεις του χαμηλού ορίου (*lower bound*) σε δέντρα ελάχιστης επέκτασης (*minimal spanning trees*), βασισμένες στο γεγονός ότι οι κορυφές στη βέλτιστη περιήγηση δεν μπορούν ποτέ να διασταυρωθούν στο ευκλείδειο TSP και την παραλληλοποίηση του σχήματος Branch and Bound. Η υλοποίηση αυτή βελτιώνει σημαντικά τις δυο τροποποιήσεις συγκριτικά με το βασικό minimal spanning tree που στη συνέχεια συνδυάστηκαν με τον branch and bound και επέφεραν επιπλέον βελτιώσεις. Οι αλγόριθμοι που συγκρίθηκαν είναι οι εξής (Σχήμα 3.1):

- shortest feasible connections (BB(sfc), BB(sfc, i))
- minimum spanning tree (BB(mst), BB(mst, i))
- minimum spanning tree – 1η τροποποίηση (BB(mstm), BB(mstm, i))
- minimum spanning tree – 2η τροποποίηση (BB(mstg), BB(mstg, i))
- Τον δυναμικό αλγόριθμο Bellman-Held-Karp (Bellman)
- Brute force χωρίς και με έλεγχο διασταυρώσεων (BF, BF(i))

Σχήμα 3.1: Χρόνος εκτέλεσης αλγορίθμων των Jagiełło και Grymin [14]

	Time[s]			
Instance	**BB(sfc)**	**BB(sfc,i)**	**BB(mst)**	**BB(mst,i)**
burma14	21.5	5.4	0.8	0.4
ulysses16	125930.3	4703.4	375.9	54.2

	Time[s]			
Instance	**BB(mstm)**	**BB(mstm,i)**	**BB(mstg)**	**BB(mstg,i)**
burma14	0.3	0.2	0.3	0.2
ulysses16	138.8	26.9	117.2	23.3

	Time[s]		
Instance	**Bellman**	**BF**	**BF(i)**
burma14	0.5	500.6	18.6
ulysses16	7.9	109855.2	423.3

3.3.2 Προσεγγιστικοί αλγόριθμοι

Όπως προαναφέρθηκε οι ακριβείς αλγόριθμοι βρίσκουν τη βέλτιστη λύση σε προβλήματα πλανόδιου πωλητή με μικρό αριθμό πόλεων. Σε περίπτωση που ο αριθμός αυτός αυξηθεί ο όγκος των υπολογισμών για τους ακριβείς αλγορίθμους αυξάνεται δραματικά. Στην προσπάθεια να ξεπεραστεί αυτό το πρόβλημα αναζητήθηκαν αλγόριθμοι που δεν απαιτούν τη βέλτιστη λύση αλλά μία σχετικά καλή λύση που προσεγγίζει τη βέλτιστη σε πολύ καλύτερο χρόνο. Για να μετρηθεί η απόδοση ενός τέτοιου αλγορίθμου υπάρχει ως έννοια ο λόγος προσέγγισης που δίνεται από τη σχέση:

$$\varrho(n) = \frac{C_{arp}}{C_{opt}}$$

όπου n ο αριθμός πόλεων, C_{arp} το κόστος της προσεγγιστικής λύσης και C_{opt} το κόστος της άριστης λύσης. Για προβλήματα ελαχιστοποίησης όπως το tsp ο λόγος είναι μεγαλύτερος της μονάδας [30].

3.3.3 Ευρετικοί - μεθευρετικοί αλγόριθμοι

Οι ευρετικοί αλγόριθμοι για το TSP μπορούν να ταξινομηθούν σε *Αλγόριθμους κατασκευής (Construction algorithms)*, *Αλγόριθμους βελτίωσης (Improvement algorithms)* και *Υβριδικούς αλγόριθμους (Hybrid algorithms)*.

Αλγόριθμοι κατασκευής: Στους αλγόριθμους κατασκευής η διαδρομή σχηματίζεται προσθέτοντας κορυφές στην ήδη υπάρχουσα διαδρομή συνήθως μια τη φορά. Ακολουθούν κάποια παραδείγματα κατασκευαστικών αλγορίθμων. Ο κατασκευαστικός *άπληστος αλγόριθμος (Greedy algorithm)* [15] παίρνει μία κορυφή ως αρχική και υπολογίζει όλες τις αποστάσεις με τις υπόλοιπες ανεξερεύνητες κορυφές, διαλέγοντας αυτή με τη μικρότερη απόσταση για να μεταβεί και να επαναλάβει τη διαδικασία. Σε αυτό το μοτίβο κινείται και ο αλγόριθμος του *πιο κοντινού γείτονα (Nearest neighbor algorithm)*.

Αλγόριθμοι βελτίωσης: Στους αλγόριθμους βελτίωσης, μια δεδομένη αρχική λύση βελτιώνεται, αν είναι δυνατόν, μεταφέροντας δύο ή περισσότερες κορυφές στην αρχική διαδρομή. Ακολουθούν κάποια παραδείγματα αλγορίθμων βελτίωσης.

Ο *k-opt* είναι ένας ευρετικός αλγόριθμος που διαγράφει k αμοιβαία αποσυνδεδεμένες κορυφές και συνδέει τις υπόλοιπες μεταξύ τους, αποκλείοντας τις k κορυφές μέχρι να μη μείνει κάτι αποσυνδεδεμένο.

Ο *γενετικός αλγόριθμος (Genetic algorithm - GA)* είναι ένας πιθανοτικός αλγόριθμος προτεινόμενος από τον Holland [19], του οποίου η ιδέα προέρχεται από την εξέλιξη. Ο GA εφαρμόζει επανειλημμένα λειτουργίες όπως *διασταύρωση (crossover)*, *μετάλλαξη (mutation)* και *επιλογή (selection)* στο σύνολο των υποψήφιων λύσεων P. Ξεκινάει απο την αρχική υποψήφια λύση P και δημιουργεί καινούριες λύσεις $S \subset N(P)$, όπου $N(P)$ είναι οι λύσεις που μπορεί να πάρει εφαρμόζοντας τις λειτουργίες διασταύρωσης και μετάλλαξης. Στη συνέχεια διαλέγει ενα $P' \subset P \subset S$ βάση των κανόνων επιλογής και θέτει $P = P'$. Ένας τελεστής διασταύρωσης δημιουργεί μία ή περισσότερες λύσεις συνδυάζοντας δύο ή περισσότερες υποψήφιες λύσεις και ένας τελεστής μετάλλαξης δημιουργεί μια λύση διαταράσσοντας ελαφρώς μια υποψήφια λύση.

Ο αλγόριθμος *αποικίας μυρμηγκιών (ant colony optimization)* [45] είναι ένας μεθευρετικός αλγόριθμος και βασίζεται στη συμπεριφορά των μυρμηγκιών που ψάχνουν για τροφή. Στο πρόβλημα tsp το τεχνητό μυρμήγκι είναι ένας πράκτορας που πάει από μια πόλη σε μια άλλη διαλέγοντας την πόλη προορισμού βάση μιας πιθανοτικής συνάρτησης. Με αυτήν την πιθανότητα το τεχνητό μυρμήγκι επιλέγει πόλεις βάση των κορυφών πιο κοντά στο μυρμήγκι και με συσσωρευμένη φερομόνη στο μονοπάτι. Οι τρεις ιδέες από τη συμπεριφορά των μυρμηγκιών που μεταφέρονται στον αλγόριθμο είναι η επιλογή διαδρομής με υψηλό επίπεδο φερομόνης, μιας και το ίχνος χρησιμοποιείται ως μηχανισμός επικοινωνίας μεταξύ των μυρμηγκιών και γίνεται επιλογή βάση του υψηλότερου ποσοστού φερομόνης σε μικρότερη διαδρομή [17].

Επιπλέον ευρετικοί-μεθευρετικοί αλγόριθμοι για επίλυση του προβλήματος του πλανόδιου πωλητή είναι οι *Simulated annealing* [10], *Multi-start local search*, [5], *Tabu search* [13], *Tree physiology optimization* [16], *Artificial bee colony* [32] και πολλοί ακόμα. Μια αναλυτικότερη οπτική στους αλγορίθμους βελτιστοποίησης ταξινομημένους σε κατηγορίες δίνουν οι Halim & Ismail [17].

Υβριδικοί αλγόριθμοι: Οι υβριδικοί αλγόριθμοι χρησιμοποιούν αλγόριθμους κατασκευής για να αποκτήσουν μια αρχική λύση και στη συνέχεια να τη βελτιώσουν

χρησιμοποιώντας έναν αλγόριθμο βελτίωσης.

3.3.4 Νευρωνικά δίκτυα για επίλυση του προβλήματος του πλανόδιου πωλητή

Εκτός από τους αλγορίθμους που προαναφέρθηκαν, τα τεχνητά νευρωνικά δίκτυα χρησιμοποιούνται για την επίλυση προβλημάτων συνδυαστικής βελτιστοποίησης και εξού και του προβλήματος του πλανόδιου πωλητή. Υπάρχουν κυρίως δύο τύποι νευρωνικών δικτύων για το TSP: το νευρωνικό δίκτυο τύπου *HopfIeld* [20] και τα νευρωνικά δίκτυα τύπου *αυτοοργανώσιμου χάρτη (self-organizing map, SOM)* [29].

Τα νευρωνικά δίκτυα τύπου HopIeld κάνουν περιοδείες αναζητώντας τις καταστάσεις ισορροπίας ενός δυναμικού συστήματος που αντιστοιχεί στο υπό εξέταση TSP. Τα νευρωνικά δίκτυα τύπου αυτοοργανώσιμου χάρτη λύνουν το TSP μέσω μάθησης χωρίς επίβλεψη. Το δίκτυο επιθεωρεί τις πόλεις εισόδου για κανονικότητες και μοτίβα και στη συνέχεια προσαρμόζει τους νευρώνες του ώστε να ταιριάζουν στην είσοδο συνεργατικά. Στη συνέχεια βρίσκει την απόκριση στην είσοδο και έτσι παραπέμπει στη γειτονιά των πόλεων. Αυτός ο χάρτης που διατηρεί τη γειτονιά οδηγεί στη συνέχεια σε μια περιοδεία. Από κάθε πόλη, η περιοδεία που προκύπτει προσπαθεί να επισκεφθεί την πλησιέστερη πόλη της.

Οι Durbin και Wilshaw [11] προτείνουν το ελαστικό δίκτυο, στο οποίο οι νευρώνες κινούνται σύμφωνα με δύο δυνάμεις που αναγκάζουν το δίκτυο να επεκταθεί ως ελαστική ζώνη, έως ότου όλες οι πόλεις καλυφθούν από τους νευρώνες σχηματίζοντας μια διαδρομή TSP. Αυτές οι δυνάμεις δρουν στους νευρώνες με στόχο την ελαχιστοποίηση του μεγέθους της ζώνης. Το ελαστικό δίχτυ χρησιμοποιήθηκε για την επίλυση περιπτώσεων 50 πόλεων του προβλήματος του πλανόδιου πωλητή με αποτελέσματα 2% χειρότερα από αυτά που λήφθηκαν με τον simulated annealing.

Οι Leung et al. [31] προτείνουν ένα νέο νευρωνικό δίκτυο που μοιάζει με SOM, που ονομάζεται επεκτεινόμενο SOM (ESOM). Σε κάθε επανάληψη εκμάθησης, το ESOM προσελκύει τους διεγερμένους νευρώνες κοντά στην πόλη εισόδου και ταυτόχρονα τους ωθεί προς το κυρτό κύτος των πόλεων συνεργατικά. Διεξάγοντας πειράματα σε τυχαία δεδομένα και δεδομένα από το TSPLIB αποδείχτηκε ότι το προτεινόμενο ESOM υπερτερεί συγκριτικά με αρκετά τυπικά νευρωνικά δίκτυα που μοιάζουν με SOM, όπως το SOM του Budinich [6], το ενισχυμένο κυρτό ελαστικό

δίχτυ [1] και οι αλγόριθμοι KNIES [2].

Η Skubalska-Rafajlowicz [39] παρουσιάζει μια προσέγγιση για την επίλυση του ευκλείδειου tsp χρησιμοποιώντας χάρτες SOM Kohonen [2] με τοπολογία αλυσίδας. Ο κανόνας μάθησης Kohonen χρησιμοποιείται με τυχαίες παραμέτρους που παρέχουν διαφορετικές θέσεις νευρώνων και κάθε θέση δίνει μια διαφορετική λύση στο πρόβλημα. Μετά από εξέταση του δικτύου σε προβλήματα αναφοράς από το TSPLIB αποδείχτηκε ότι σε κάθε επανάληψη είχε αυξομειώσεις στη λύση αλλά το γενικό μοτίβο είναι προς τα κάτω. Για παράδειγμα στο πρόβλημα pbc1173 η βέλτιστη λύση είναι $56,892$ και μετά από $800,000$ επαναλήψεις το δίκτυο έδωσε $62,901.2$, δηλαδή 1.127 φορές μεγαλύτερη από τη βέλτιστη.

Οι Créput και Koukam [8] λύνουν το ευκλείδειο tsp με τον υβριδισμό του SOM σε έναν εξελικτικό αλγόριθμο. Η εξελικτική δυναμική περιλαμβάνει την αλληλεπίδραση της εκτέλεσης του SOM με χειριστή χαρτογράφησης, αξιολόγηση καταλληλότητας και χειριστή επιλογής. Το δίκτυο ονομάζεται *memetic SOM* και εξετάζεται σε ένα σύνολο από προβλήματα tsp. Αποδεικνύεται ότι αποδίδει καλύτερα από το ESOM και eISOM σε προβλήματα με μικρό αριθμό πόλεων. Στα μεγαλύτερα προβλήματα με περίπου 12% λιγότερο χρόνο εκτέλεσης, το memetic SOM αποδίδει καλύτερα ποιοτικά αποτελέσματα κατά μέσο όρο από το Co-Adaptive Net [7].

Οι Jin et al. [22] προτείνουν έναν ολοκληρωμένο SOM, που ονομάζεται *ISOM*, με έναν κανόνα εκμάθησης που ενσωματώνει τους τρεις κανονισμούς από τη βιβλιογραφία των SOM. Αυτό σημαίνει ότι μέσα σε ένα βήμα εκμάθησης ο διεγερμένος νευρώνας σύρεται πρώτα προς την πόλη εισόδου, στη συνέχεια ωθείται στο κυρτό κύτος του TSP και τελικά τραβιέται προς το μεσαίο σημείο των δύο γειτονικών νευρώνων του. Στη συνέχεια γίνεται επέκταση του στο *eISOM* προσδιορίζοντας ένα γενετικό αλγόριθμο που καθορίζει τον συντονισμό των κανόνων και τις μεταβλητές τους. Ο eISOM στη συνέχεια εξετάζεται σε διάφορα προβλήματα tsp. Σε σύγκριση με την προσέγγιση simulated annealing, μπορεί να πραγματοποιήσει διαδρομές περίπου 3% μικρότερες έχοντας λιγότερο χρόνο εκτέλεσης. Έχει βελτιώσει τουλάχιστον 1% τη διαδρομή σε σχέση με το SOM που αναπτύχθηκε από τον Budinich [6], το ESOM [31] και το κυρτό ελαστικό δίχτυ [11].

Οι Cochrane & Beasley [7] προτείνουν ένα νευρωνικό δίκτυο τύπου SOM για την επίλυση του ευκλείδειου tsp, που δε χρησιμοποιεί μόνο μάθηση χωρίς επίβλεψη για

την εκπαίδευση των νευρώνων αλλά επιτρέπει στους νευρώνες να συνεργάζονται και να ανταγωνίζονται μεταξύ τους. Μετά από πειράματα συγκρίναν το co-op net με άλλου τύπου νευρωνικά δίκτυα όπως το elastic net, Hopfield και SOM και αποδεικνύεται ότι η πρότασή τους υπερτερεί και σε ποιότητα αποτελεσμάτων αλλά και σε χρόνο εκτέλεσης. Επίσης, όταν δημοσιεύτηκε, ήταν η μεγαλύτερη υπολογιστική μελέτη για ένα νευρωνικό δίκτυο εξετάζοντας 91 προβλήματα, με το μεγαλύτερο να έχει $85,900$ πόλεις.

Οι Al-Mulhem & Al-Maghrabi [1], προτείνουν έναν υβριδικό αλγόριθμο για την επίλυση του ευκλείδειου tsp, που συνδυάζει το *convex-elastic net (CEN)* με τον αλγόριθμο της μη ντετερμινιστικής επαναληπτικής βελτίωσης *(nondeterministic iterative improvement (NII))* και ονομάζεται *Efficient Convex-Elastic Net (ECEN)*. Ο ECEN παίρνει το αποτέλεσμα από τον CEN και εφαρμόζει δύο τελεστές αναδιάταξης για να βελτιώσει την αρχική περιήγηση που δίνει ο αλγόριθμος CEN. Μετά από πειραματική διαδικασία σε τυχαία προβλήματα αλλά και σε γνωστά προβλήματα αποδείχτηκε ότι η πρότασή τους παράγει καλύτερα αποτελέσματα από άλλα νευρωνικά, όπως τα Hopfield model, guilty net και elastic net, και κλιμακώνεται καλά με το μέγεθος του προβλήματος.

Κεφάλαιο 4

Υλοποίηση βελτιωτικών αλγόριθμων βελτιστοποίησης

Η παρούσα διπλωματική εργασία χρησιμοποιεί και εκπαιδεύει ένα νευρωνικό δίκτυο που έχει δημιουργηθεί από τους Deudon et al. [9], το οποίο προβλέπει διαδρομές σε ευκλείδεια προβλήματα πλανόδιου πωλητή και στη συνέχεια βελτιώνει αυτήν τη διαδρομή, χρησιμοποιώντας τον αλγόριθμο *2opt*. Στο πλαίσιο της βελτίωσης της διαδρομής που δίνεται ως έξοδος από το νευρωνικό δίκτυο, έχουν υλοποιηθεί και εφαρμοστεί οι βελτιωτικοί αλγόριθμοι *3opt*, *Simulated annealing*, *Variable neighborhood descent search* και ο κατασκευαστικός αλγόριθμος *Nearest neighbor*.

4.1 Το νευρωνικό δίκτυο

Το νευρωνικό δίκτυο μαθαίνει τις παραμέτρους θ μιας στοχαστικής πολιτικής (*stochastic policy*) για τις μεταθέσεις των πόλεων $p\theta(\pi|s)$ χρησιμοποιώντας ενισχυτική μάθηση σε νευρωνικά δίκτυα και $policygradient$. Δίνοντας ένα σύνολο σημείων s, η βασική ιδέα είναι να εκχωρηθεί υψηλότερη πιθανότητα σε "καλές" διαδρομές π^+ και μικρότερη πιθανότητα σε "ανεπιθύμητες" διαδρομές π^-. Ακολουθεί τη γενική ιδέα κωδικοποιητή-αποκωδικοποιητή. Ο κωδικοποιητής αντιστοιχίζει ένα σύνολο εισόδου $I = (i_1.....i_n)$ σε ένα σύνολο συνεχών αναπαραστάσεων $Z = (z_1.....z_n)$. Δεδομένου του , ο αποκωδικοποιητής δημιουργεί μια ακολουθία συμβόλων εξόδου $O = (o_1.....o_n)$, ένα στοιχείο τη φορά. Σε κάθε βήμα στο μοντέλο γίνεται αυτόματη παλινδρόμηση, χρησιμοποιώντας τα σύμβολα που δημιουργήθηκαν προηγουμένως, ως πρόσθετη είσοδο κατά τη δημιουργία του επόμενου συμβόλου. Στις συντεταγμένες εισόδου χρησιμοποιείται $Principal\ Component\ Analysis\ (PCA)$.

4.1.1 Κωδικοποιητής

Ο σκοπός του κωδικοποιητή είναι να αποκτήσει μια αναπαράσταση για κάθε πόλη, δεδομένου του πλαισίου της. Η έξοδος είναι ένα σύνολο διανυσμάτων $A = (a_1.....a_n)$ που το καθένα αντιπροσωπεύει μια πόλη που αλληλεπιδρά με άλλες πόλεις. Χρησιμοποιείται ο κωδικοποιητής που αναφέρεται στο [42] που βασίζεται σε μηχανισμούς προσοχής αντί των παραδοσιακών συσπειρώσεων ή υποτροπών και όπως στο [42], ο ηθοποιός και ο κριτής χρησιμοποιούν μηχανισμούς νευρικής προσοχής για να κωδικοποιήσουν τις πόλεις ως σύνολο και όχι ως ακολουθία. Πιο συγκεκριμένα, ο κωδικοποιητής λαμβάνει ως είσοδο ένα ενσωματωμένο και κανονικοποιημένο σύνολο από πόλεις $s = (city_i)$ (d-διάστατος χώρος). Συνολικά αποτελείται από μια στοίβα πανομοιότυπων επιπέδων, όπως φαίνεται στο Σχήμα 4.1 που κάθε επίπεδο έχει δύο υποεπίπεδα.

Το πρώτο υποεπίπεδο $Multi - head\ Attention$ είναι ένας μηχανισμός νευρικής προσοχής που επιτρέπει στα queries να αλληλεπιδρούν με ζεύγη κλειδιών-τιμών. Για το TSP, τα queries και τα ζεύγη κλειδιών-τιμών $q_i, k_i, v_i \in \mathbb{R}^d$ λαμβάνονται με τον γραμμικό μετασχηματισμό της κάθε πόλης $city_i \in \mathbb{R}^d$ και την εφαρμογή μιας μη γραμμικής $ReLu$.

Ακολουθώντας το [42] ο μηχανισμός ορίζεται ως εξής:

$$Attention(Q; K; V) = softmax(\frac{QK^T}{\sqrt{d}})$$

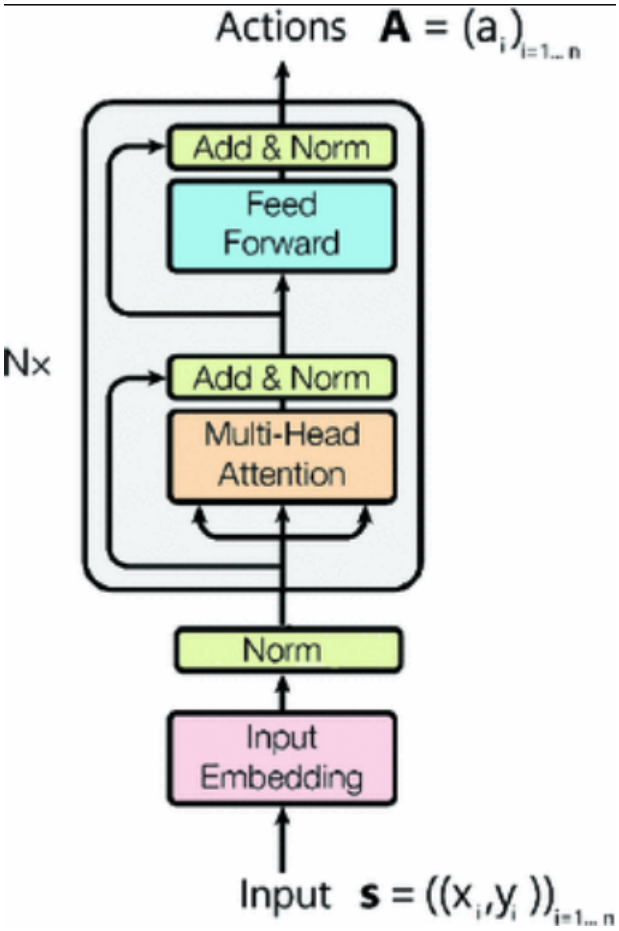
όπου $Q = [q_1.....q_n]$, $K = [k_1.....k_n]$, $V = [v_1.....v_n]$.

Το υποεπίπεδο $Multi - Head\ Attention$ εξάγει μια νέα αναπαράσταση για κάθε πόλη, υπολογιζόμενη ως σταθμισμένο άθροισμα των τιμών της πόλης, όπου τα αντίστοιχα βάρη ορίζονται από μια συνάρτηση συνάφειας μεταξύ των queries και των κλειδιών των πόλεων. Όπως προτείνεται στο [42], τα queries, τα κλειδιά και οι τιμές προβάλλονται γραμμικά σε h διαφορετικούς υποχώρους (εξού και το όνομα Multi Head). Στη συνέχεια, εφαρμόζουμε τον μηχανισμό προσοχής σε κάθε ένα από αυτά τα νέα σύνολα αναπαραστάσεων για να λάβουμε h d_h-διάστατες τιμές εξόδων για κάθε πόλη, οι οποίες συνδέονται με τις τελικές τιμές.

Το δεύτερο υποεπίπεδο Feed-Forward αποτελείται από δύο γραμμικούς μετασχηματισμούς κατά θέση με ενδιάμεση ενεργοποίηση ReLU. Η έξοδος κάθε υποεπι-

πέδου δίνεται από τον τύπο $LayerNorm(x + Sublayer(x))$, όπου το $Sublayer(x)$ είναι η συνάρτηση που υλοποιείται από το ίδιο το υποεπίπεδο και το $LayerNorm()$ που είναι η κανονικοποίηση επιπέδου [21].

Σχήμα 4.1: Νευρωνικός κωδικοποιητής [9]



4.1.2 Αποκωδικοποιητής

Ακολουθώντας το [3], η αρχιτεκτονική του νευρωνικού δικτύου, χρησιμοποιεί τον κανόνα της αλυσίδας για να παραγοντοποιήσει την πιθανότητα μιας διαδρομής ως:

$$p\theta(\pi|s) = \prod_{t=1}^{n} p\theta(\pi(t)|\pi(< t), s)$$

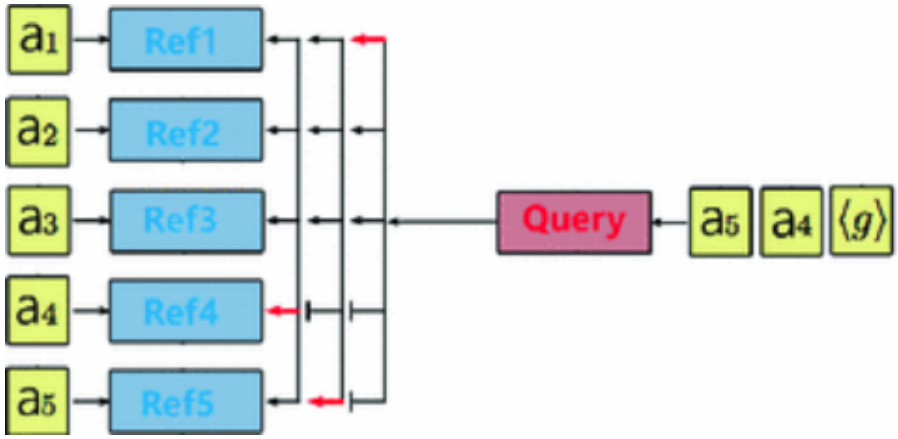
Κάθε όρος στη δεξιά πλευρά της εξίσωσης υπολογίζεται διαδοχικά με μονάδες softmax. Σε αντίθεση με το [3] που συνοψίζει όλες τις προηγούμενες ενέργειες σε ένα διάνυσμα σταθερού μήκους, το μοντέλο ξεχνά ρητά μετά από $K = 3$ βήματα, καθιστώντας μη αναγκαία τη χρήση δυκτίων $Long\ Short\ Term\ Memory$ $(LSTM)$.

Σε κάθε χρόνο εξόδου t, αντιστοιχίζονται οι τρεις τελευταίες ενέργειες (πόλεις που επισκέφτηκαν) στο ακόλουθο διάνυσμα:

$$q_t = ReLu(W_1 a_{\pi(t-1))} + W_2 a_{\pi(t-2))} + W_3 a_{\pi(t-3))} \in \mathbb{R}^{d'}$$

Το παραπάνω διάνυσμα q_t αλληλεπιδρά με ένα σύνολο n διανυσμάτων για να ορίσει μια κατανομή κατάδειξης στον χώρο ενεργειών. Μόλις γίνει δειγματοληψία της επόμενης πόλης, η τροχιά $q_{t+!}$ ενημερώνεται με το επιλεγμένο διάνυσμα δράσης και η διαδικασία τελειώνει όταν ολοκληρωθεί η διαδρομή (Σχήμα 4.2).

Σχήμα 4.2: Νευρωνικός Αποκωδικοποιητής [9]



Μηχανισμός Κατάδειξης Ο μηχανισμός κατάδειξης που χρησιμοποιείται για να προβλέψει μια κατανομή σε πόλεις, δεδομένων κωδικοποιημένων ενεργειών (πόλεις) και μιας αναπαράστασης κατάστασης (διάνυσμα query), είναι ίδιος με αυτόν του [3]. Η κατάδειξη μιας συγκεκριμένης θέσης στην ακολουθία εισόδου, επιτρέπει την προσαρμογή του ίδιου πλαισίου σε διαδρομές μεταβλητού μήκους. Ακολουθώντας το [3], ο μηχανισμός κατάδειξης έχει ως παραμέτρους δύο πίνακες προσοχής $W_{ref} \in \mathbb{R}^{dd"}$, $W_q \in \mathbb{R}^{d'd"}$ και ένα διάνυσμα προσοχής $v \in \mathbb{R}^{d"}$:

$$\forall i \leq n, u_i^t = \begin{cases} v^T \tanh(W_{ref} a_i + W_q q_t), & \text{αν } i \in \{\pi(0), ..., \pi(t-1)\} \\ -\infty, & \text{σε όποια άλλη περίπτωση} \end{cases}$$

$$p\theta(\pi(t)|\pi(<t), s) = softmax(C \tanh v^t / T)$$

Ο όρος $p\theta(\pi(t)|\pi(<t), s)$ προβλέπει μια κατανομή στο σύνολο των n διανυσμάτων δράσης, δεδομένου ενός query q_t. Οι πιθανότητες καταγραφής των πόλεων που έχουν

ήδη εμφανιστεί στη διαδρομή ορίζεται σε $-\infty$. Η διαδικασία αυτή διασφαλίζει ότι το μοντέλο μας εξάγει έγκυρες μεταθέσεις της εισόδου. Όπως αναφέρεται στο [3], η αποκοπή των πιθανοτήτων καταγραφής σε $[-C, +C]$ είναι ένας τρόπος ελέγχου της εντροπίας. Ο όρος T είναι μια υπερπαράμετρος θερμοκρασίας που χρησιμοποιείται για τον έλεγχο της βεβαιότητας της δειγματοληψίας και έχει οριστεί ως $T = 1$ κατά τη διάρκεια της εκπαίδευσης και $T > 1$ κατά τη διάρκεια των συμπερασμάτων.

4.2 Αλγόριθμοι βελτιστοποίησης

4.2.1 2opt

Στον Αλγόριθμο 1 δίνεται ο ευρετικός αλγόριθμος 2-opt [36] που προτάθηκε αρχικά από τον Croes το 1958. Σαν βασική ιδέα ο αλγόριθμος διαγράφει 2 συνδέσεις μεταξύ κορυφών της διαδρομής tour και επανασυνδέει τις κορυφές εξαιρώντας τις προηγούμενες συνδέσεις, έτσι δημιουργεί 2 εναλλακτικές διαδρομές. Ο αλγόριθμος τελειώνει όταν εξεταστεί κάθε πιθανός συνδυασμός. Παρόλο που θεωρητικά δεν υπάρχουν πολλές γνώσεις για αυτόν τον ευρετικό αλγόριθμο σε πολλές παραλλαγές του προβλήματος με ευκλείδεια απόσταση έχει δειχτεί ότι έχει χρονική πολυπλοκότητα $O(n^3)$ [40].

Αναλύοντας την παρακάτω υλοποίηση του αλγόριθμου παρατηρούμε ότι αποτελείται από μία συνάρτηση swap2opt, η οποία παίρνει σαν όρισμα μια ακολουθία από πόλεις $tsp\_sequence$ και ανταλλάσσει τις πόλεις που βρίσκονται στην θέση i με τη θέση j στην ακολουθία. Στη συνέχεια έχουμε μία συνάρτηση step2opt η οποία αντιπροσωπεύει το βήμα του 2opt και έχει ως όρισμα μια ακολουθία tsp πόλεων.

Στις πρώτες εντολές βρίσκουμε το μήκος της ακολουθίας, δηλαδή πόσες πόλεις υπάρχουν, και το μήκος της διαδρομής που ακολουθούμε. Για κάθε ζευγάρι πόλεων εφαρμόζουμε τη συνάρτηση swap2opt και παίρνουμε το καινούριο μήκος από τη διαδρομή. Εάν αυτό το μήκος είναι πιο σύντομο από το προηγούμενο επιστρέφουμε τη βελτιωμένη ακολουθία tsp και το βελτιωμένο μήκος, σε αντίθετη περίπτωση επιστρέφει τα ήδη υπάρχοντα, μήκος και ακολουθία. Αφού ολοκληρωθούν οι συναρτήσεις περνάμε στο βασικό μέρος που παίρνει σαν όρισμα μια ακολουθία tsp και τον μέγιστο αριθμό επαναλήψεων ($iterations$), ορίζουμε ως καλύτερη διαδρομή την τωρινή ακολουθία και ως καλύτερο μήκος, το μήκος της ακολουθίας

αυτής. Μετέπειτα επαναλαμβάνουμε το βήμα του 2opt με τη συνάρτηση step2opt για όσες φορές έχουν δοθεί σαν όρισμα. Μέσα σε αυτήν την επανάληψη εξετάζουμε αν η καινούρια διαδρομή είναι πιο σύντομη από την καλύτερη και αν είναι, θα γίνει αυτή η καλύτερη.

Function `swap2opt`($tsp\_sequence$, i, j):
  $new\_tsp\_sequence = copy(tsp\_sequence)$;
  $new\_tsp\_sequence[i : j + 1] = swap(new\_tsp\_sequence[i : j + 1])$;
  return $new\_tsp\_sequence$;
end
Function `step2opt`($tsp\_sequence$):
  $seq\_length \leftarrow$ *length of the sequence list*;
  $distance \leftarrow$ *reward of the sequence*;
  for $i \leftarrow 1$ **to** $seq\_length - 1$ **do**
    for $j \leftarrow i + 1$ **to** seq_length **do**
      perform a city swap in the sequence with the swap2opt function;
      $new\_distance \leftarrow$ *reward of the improved tsp sequence*;
      if $new\_distance < distance$ **then**
        return the improved sequence and the improved distance;
      return this sequence and distance;
    end
  end
end
Input: A TSP sequence $tsp\_sequence$ and a number of max iterations max_iter
Result: The best solution found
$best\_reward \leftarrow$ *reward of this tsp sequence*;
$new\_tsp\_sequence \leftarrow$ *copies this tsp sequence*;
for $i \leftarrow 1$ **to** max_iter **do**
  $new\_reward, new\_tsp\_sequence \leftarrow$ *results of the step2opt function*;
  if *new_reward* $<$ *best_reward* **then**
    $new\_reward \leftarrow best\_reward$
  else
    break;
  end
end

Αλγόριθμος 1: Αλγόριθμος 2opt

4.2.2 3opt

Στον Αλγόριθμο 2 δίνεται ο αλγόριθμος 3-opt [36] που ακολουθεί την ίδια λογική με τον 2-opt και γενικά όλους τους k-opt αλγορίθμους. Δηλαδή διαγράφει 3 συνδέσεις μεταξύ κορυφών της διαδρομής tour και επανασυνδέει τη διαδρομή

εξαιρώντας τις προηγούμενες συνδέσεις. Δημιουργεί έτσι 7 πιθανούς τρόπους επανασύνδεσης της διαδρομής, οι οποίοι εξετάζονται για να βρεθεί η βέλτιστη επανασύνδεση. Αυτή η διαδικασία επαναλαμβάνεται για διαφορετικές τριάδες μέχρι να εξεταστούν όλοι οι πιθανοί συνδυασμοί. Συγκριτικά με τον αλγόριθμο 2-opt, ο 3-opt είναι αρκετά πιο αργός αλλά δίνει καλύτερες λύσεις. Μια εφαρμογή του 3opt έχει χρονική πολυπλοκότητα $O(n^3)$ [4]. Η επαναληπτική εφαρμογή έχει υψηλότερη χρονική πολυπλοκότητα.

Η παρακάτω υλοποίηση του αλγορίθμου αποτελείται για αρχή από μία συνάρτηση *swap3opt* που παίρνει σαν ορίσματα μια ακολουθία tsp πόλεων ($tsp\_sequence$), μέσα στη συνάρτηση δημιουργούνται οι 7 πιθανοί τρόποι επανασύνδεσης μεταξύ των τριών κορυφών και συγκρίνονται μεταξύ τους επιστρέφοντας αυτή με το μικρότερο μήκος διαδρομής. Στη συνέχεια έχουμε τη συνάρτηση *step3opt* που δέχεται σαν όρισμα μια ακολουθία πόλεων tsp. Η λειτουργία της είναι ότι βρίσκει το μήκος της ακολουθίας και το μήκος της διαδρομής και τα εισάγει σε αντίστοιχες μεταβλητές, δημιουργεί μια εμφωλευμένη επανάληψη για πάρει τις 3 κορυφές και μέσα της βρίσκει τη βελτιωμένη ακολουθία για τις 3 κορυφές που πήρε καλώντας τη συνάρτηση swap3opt. Από τη νέα ακολουθία βρίσκει το μήκος της διαδρομής και το συγκρίνει με το παλιό μήκος, αν αυτό είναι μικρότερο τότε επιστρέφει τη βελτιωμένη ακολουθία και τη βελτιωμένη διαδρομή αλλιώς επιστρέφει τις ήδη υπάρχουσες τιμές.

Αφού ολοκληρωθούν οι συναρτήσεις, περνάμε στο βασικό μέρος το οποίο παίρνει σαν όρισμα μια ακολουθία tsp και τον μέγιστο αριθμό επαναλήψεων ($iterations$), ορίζουμε ως καλύτερη διαδρομή την τωρινή ακολουθία και ως καλύτερο μήκος, το μήκος της ακολουθίας. Έπειτα έχουμε μια επανάληψη για όσες φορές πήραμε ως όρισμα, στην οποία καλούμε τη συνάρτηση step3opt και εισάγουμε τα αποτελέσματα που επιστρέφονται σε αντίστοιχες μεταβλητές. Συγκρίνουμε το μήκος της διαδρομής που επέστρεψε η συνάρτηση με το καλύτερο μήκος διαδρομής και εφόσον είναι μικρότερο το καλύτερο μήκος διαδρομής, παίρνει την τιμή της επιστροφής αλλιώς παραμένει ίδιο και ξαναπάει στην αρχή της επανάληψης.

```
Function swap3opt(tsp_sequence, i, j, k):
    tsp0 ← tsp_sequence;
    tsp1 ← swap city i with city j in tsp sequence;
    tsp2 ← swap city i with city k in tsp sequence;
    tsp3 ← swap city j with city k in tsp sequence;
    // flip reverses the order of elements
    tsp4 ← swap 3 cities in the tsp sequence;
    tsp5 ← swap another combination of 3 cities in the tsp sequence;
    tsp6 ← swap another combination of 3 cities in the tsp sequence;
    tsp7 ← swap another combination of 3 cities in the tsp sequence;
end
return The tsp that has the minimum distance
Function step3opt(tsp_sequence):
    seq_length ← length of the tsp sequence;
    distance ← distance of the tsp sequence;
    for i ← 0 to seq_length − 3 do
        for j ← 0 to seq_length − 2 do
            for k ← 0 to seq_length − 1 do
                new_tsp_sequence ← swap3opt(tsp_sequence, i, j, k);
                new_distance = reward(new_tsp_sequence);
                if new_distance < distance then
                    return new_tsp_sequence, new_distance
            end
        end
    end
    return tsp_sequence , distance
end
Input: A tsp sequence tsp_sequence and number of max iterations max_iter
Result: The best solution found
new_tsp_sequence ← tsp_sequence;
best_reward ← distance of the tsp sequence;
for i ← 0 to max_iter do
    new_tsp_sequence, new_reward ← step3opt(new_tsp_sequence);
    if new_reward < best_reward then
        best_reward ← new_reward;
    else
        break;
    end
end
```

Αλγόριθμος 2: Αλγόριθμος 3opt

4.2.3 Simulated annealing

Στον Αλγόριθμο 3 δίνεται ο αλγόριθμος simulated annealing [10], ο οποίος είναι μια πιθανοτική μέθοδος που βρίσκει το ολικό ελάχιστο μιας συνάρτησης κόστους που μπορεί να έχει αρκετά τοπικά ελάχιστα. Η διαδικασία είναι βασισμένη στο ξε-

πύρωμα στερεών στη μεταλλουργία, η οποία είναι μια τεχνική που περιλαμβάνει τη θέρμανση και την ελεγχόμενη ψύξη ενός υλικού, έτσι ώστε η δομή του να σταθεροποιηθεί, αυτό γίνεται στην ελάχιστη σύνθεση ενέργειας. Σε κάθε βήμα ο αλγόριθμος εξετάζει κάποια γειτονική κατάσταση (s*) της τωρινής (s) και βάση πιθανοτήτων αποφασίζει εάν θα μεταφέρει το σύστημα στη γειτονική ή θα παραμείνει στην ίδια. Στο συγκεκριμένο πρόβλημα οι γείτονες δημιουργούνται με την αλλαγή δύο τυχαίων πόλεων.

Η υλοποίηση του αλγορίθμου περιέχει μία συνάρτηση $acceptancepossibility$ και το κύριο μέρος του αλγορίθμου. Η συνάρτηση με ορίσματα την υποψήφια διαδρομή $candidate\_tour$, την τωρινή διαδρομή $current\_tour$ και τη θερμοκρασία , επιστρέφει την πιθανότητα με την οποία γίνεται η αλλαγή κατάστασης. Η πιθανότητα δίνεται από τον τύπο $\mid (candidate\_tour - current\_tour)/T \mid$, έτσι ώστε η πιθανότητα αποδοχής να μικραίνει όσο η διαφορά $candidate\_tour - current\_tour$ αυξάνεται.

Στο κύριο μέρος έχουμε ως ορίσματα μια ακολουθία πόλεων $tsp\_sequence$, τον ρυθμό με τον οποίο θα γίνεται η μείωση της θερμοκρασίας $alpha$, την αρχική θερμοκρασία , τον αριθμό θερμοκρασίας που όταν επιτευχθεί θεωρείται οτι έγινε η ψύξη και σταματάει ο αλγόριθμος $stopping\_temperature$ και τον μέγιστο αριθμό επαναλήψεων εάν δεν επιτευχθεί η μείωση της θερμοκρασίας $stopping\_iter$. Αρχικοποιούμε ως τωρινή λύση την ακολουθία που είχαμε ως όρισμα, ως τωρινή διαδρομή το μήκος της τωρινής ακολουθίας και ως βέλτιστη λύση την τωρινή διαδρομή. Στη συνέχεια έχουμε μια επανάληψη για όσο η θερμοκρασία δε φτάσει την τιμή του ορίσματος και οι επαναλήψεις δε ξεπεράσουν τις μέγιστες. Μέσα στην επανάληψη παίρνουμε δύο τυχαίες πόλεις από την ακολουθία i, j και τις αλλάζουμε θέση μεταξύ τους δημιουργώντας μια υποψήφια ακολουθία αλλά και διαδρομή. Εάν αυτή η υποψήφια διαδρομή είναι μικρότερη από την τωρινή, η τωρινή διαδρομή και η τωρινή λύση παίρνουν την τιμή της υποψήφιας, εάν είναι μικρότερη και από την καλύτερη διαδρομή τότε και η καλύτερη διαδρομή και η καλύτερη λύση παίρνουν την τιμή της υποψήφιας. Στην αντίθετη περίπτωση που δεν είναι μικρότερη της τωρινής καλούμε τη συνάρτηση $acceptancepossibility$ και αν η πιθανότητα ικανοποιείται η τωρινή διαδρομή και η τωρινή λύση παίρνουν την τιμή της υποψήφιας. Αφού ολοκληρωθούν αυτοί οι έλεγχοι μειώνουμε τη θερμοκρασία βάση της μεταβλητής $alpha$ και αυξάνουμε τον αριθμό των επαναλήψεων κατά 1. Όταν δεν ικανοποιείται πλέον

η συνθήκη της επανάληψης, αυτή τελειώνει και ο αλγόριθμος επιστέφει τη βέλτιστη διαδρομή και λύση.

Function `acceptance possibility`($candidate\_tour$, $current\_tour$, T):
  return the absolute value of $(candidate\_tour - current\_tour) \div T$
end
Input: A TSP sequence $tsp\_sequence$, the tempo of temperature decreace $alpha$, the temperature T, the number of stoπping temperature $stopping\_temperature$, and the number of stoπping iterations $stopping\_iter$
Result: The best solution found
$current\_solution \leftarrow tsp\_sequence$;
$current\_route \leftarrow$ *reward of the tsp sequence*;
$best\_route \leftarrow current\_route$;
while *temperature* $<$ $stopping\_temperature$ *and iteration* $<$ $stopping\_iter$ **do**
  $l, i \leftarrow$ *two random cities*;
  $candidate \leftarrow$ *swap city l with city i in the tsp sequence*;
  $candidate\_route \leftarrow$ *reward of candidate tsp* ;
  if *candidate_route* $<$ *current_route* **then**
    $current\_route \leftarrow candidate\_route$;
    $current\_solution \leftarrow candidate$;
    if *candidate_route* $<$ *best_route* **then**
      $best\_route \leftarrow candidate\_route$;
      $best\_solution \leftarrow candidate$;
  else
    if *acceptance_possibility is satisfied* **then**
      $current\_route \leftarrow candidate\_route$;
      $current\_solution \leftarrow candidate$;
    $T = T * alpha$;
    $iteration = iteration + 1$;
  end
  return best_route, best_solution;
end

Αλγόριθμος 3: Αλγόριθμος simulated annealing

4.2.4 Nearest neighbor

Στον Αλγόριθμο 4 δίνεται ο αλγόριθμος Nearest Neighbor (NN) [28], ο οποίος ανήκει στην κατηγορία αλγορίθμων κατά προσέγγιση για την επίλυση του προβλήματος του πλανόδιου πωλητή, αυτό σημαίνει ότι δε βρίσκει τη βέλτιστη λύση αλλά μια κοντά σε αυτή. Στον συγκεκριμένο αλγόριθμο δίνεται ένα σύνολο από πόλεις $cities(x, y)$, έπειτα επιλέγει μία τυχαία πόλη c για να ξεκινήσει τη διαδρομή και επισκέπτεται την πιο κοντινή σε αυτή βάση της ευκλείδειας απόστασης. Στα επό-

μενα βήματα επισκέπτεται την πιο κοντινή ανεξερεύνητη πόλη και επαναλαμβάνει τη διαδικασία μέχρι να επισκεφθεί όλες τις πόλεις. Αφού επισκεφτεί και την τελευταία ανεξερεύνητη πόλη επιστρέφει στην αρχική c. Ο αλγόριθμος αυτός έχει χρονική πολυπλοκότητα $O(dN)$, όπου είναι το πλήθος των cities και d είναι η διάσταση του χώρου.

Αναλύοντας την παρακάτω υλοποίηση του αλγόριθμου παρατηρούμε ότι αποτελείται από μία συνάρτηση distance, η οποία έχει σαν όρισμα δύο πόλεις x, y και επιστρέφει την ευκλείδεια απόσταση μεταξύ τους. Στη συνέχεια έχουμε τη συνάρτηση Nearest neighbour που δέχεται σαν όρισμα μία πόλη $city$ και μία λίστα απο πόλεις, οι οποίες είναι ανεξερεύνητες $list\_of\_cities$. Εξετάζει ποια από τις ανεξερεύνητες πόλεις είναι πιο κοντά στην πόλη city σύμφωνα με τη συνάρτηση distance και την επιστρέφει.

Στο κύριο μέρος του αλγορίθμου παίρνουμε σαν όρισμα μία ακολουθία πόλεων, αρχικοποιούμε την αφετηρία $start$ ως την πρώτη πόλη της ακολουθίας και την προσθέτουμε στη διαδρομή $tour$. Όλες οι υπόλοιπες πόλεις προστίθενται σε μία λίστα ως ανεξερεύνητες $unvisited$. Έπειτα για όσο η λίστα $unvisited$ δεν είναι άδεια, δηλαδή υπάρχουν πόλεις που δεν έχουν εξερευνηθεί βρίσκουμε την πιο κοντινή στην τωρινή πόλη ($start$) χρησιμοποιώντας τη συνάρτηση $nearest\_neighbor$ και τη θέτουμε ως τωρινή. Αφού βρεθεί η πόλη, την προσθέτουμε στο $tour$ και τη διαγράφουμε από τη λίστα με τις ανεξερεύνητες πόλεις. Όταν η επανάληψη φτάσει στο τέλος της και έχουν εξερευνηθεί όλες οι πόλεις βρίσκουμε το μήκος της διαδρομής, το εισχωρούμε στην $best\_route$ και επιστρέφουμε τη διαδρομή ($tour$) και το μήκος της ($best\_route$).

4.2.5 Variable neighborhood descent search

Στον Αλγόριθμο 5 δίνεται ο αλγόριθμος variable neighborhood search [34], ο οποίος είναι ένας μεθευρετικός αλγόριθμος για επίλυση προβλημάτων συνδυαστικής βελτιστοποίησης του οποίου η βασική ιδέα είναι η συστηματική αλλαγή γειτονιάς μέσα σε μία τοπική αναζήτηση. Διάφορες παραλλαγές του αλγορίθμου αναφέρονται στο [34]. Εδώ χρησιμοποιούμε μια παραλλαγή η οποία κάνει πολλές καταβάσεις σε διαφορετικές γειτονιές μέχρι το τοπικό ελάχιστο να βρεθεί για όλες. Αυτή η παραλλαγή ονομάζεται variable neighborhood descent (VND). Ο VND περιλαμβάνει την επιλογή μιας αρχικής θέσης x, βρίσκει μια κατεύθυνση για απότομη κατάβαση

Function `distance`(x, y):
  return $math.sqrt((x[0] - y[0]) * *2 + (x[1] - y[1]) * *2)$;
end
Function `Nearest neighbour`($city$, $list\_of\_cities$):
  return $min(cities, key = lambda\ c : distance(city, list\_of\_cities))$;
end
Input: A TSP sequence $tsp\_sequence$
Result: The best solution found
$start \leftarrow$ *the first city of the sequence*;
$tour \leftarrow$ *the tour begining from the first city*;
$unvisited \leftarrow$ *all the cities that are still unvisited*;
while *unvisited is not empty* **do**
  $c \leftarrow nearest\_neighbor(start, unvisited)$;
  $start \leftarrow c$;
  Add c to the tour;
  Delete c from unvisited;
end
$best\_route \leftarrow$ *distance of the tour*;
return tour, best_route;

Αλγόριθμος 4: Αλγόριθμος nearest neighbour

από την x μέσα σε μια γειτονιά (x) και μεταφέρεται στην ελάχιστη $f(x)$ μέσα στην (x), αν δεν υπάρχει κατεύθυνση για κατάβαση, ο αλγόριθμος τερματίζει ή επαναλαμβάνεται.

Αναλύοντας την παρακάτω υλοποίηση του αλγόριθμου παρατηρούμε ότι αποτελείται από μία συνάρτηση $bestneighbour$ που έχει σαν ορίσματα μια ακολουθία πόλεων $tsp\_sequence$ και το μέγεθος της γειτονιάς $neighbourhood\_size$. Αρχικοποιείται ως λύση $solution$ η ακολουθία που πήρε σαν όρισμα $tsp\_sequence$. Μετά, μέσα σε μια επανάλληψη, για να βρεθεί ο καλύτερος γείτονας στην γειτονιά, η συνάρτηση διαλέγει δύο τυχαίες πόλεις και τις αλλάζει θέση στην ακολουθία εισχορόντας την καινούρια ακολουθία στη μεταβλητή $candidate$ δημιουργόντας έτσι έναν γείτονα . Αν η ακολουθία $candidate$ έχει μικρότερη διαδρομή από την ακολουθία $solution$, η $solution$ παίρνει την τιμή της $candidate$. Στο τέλος της συνάρτησης επιστρέφεται ο καλύτερος γείτονας που βρέθηκε.

Στο κύριο μέρος του αλγορίθμου παίρνουμε σαν ορίσματα μια ακολουθία πόλεων $tsp\_sequence$, τον μέγιστο αριθμό προσπαθειών βελτίωσης πριν την αλλαγή γειτονιάς, $max\_attempts$, το μέγεθος της γειτονιάς, $neighbourhood\_size$ και τον αριθμό επαναλήψεων $iterations$. Μέσα στη συνάρτηση δημιουργούμε μία μεταβλητή $count$ και την αρχικοποιούμε στο 0 για να γίνεται η σύγκριση με τον αριθμό επανα-

λήψεων, μια μεταβλητή $k = 1$ για να γίνεται η σύγκριση με τον μέγιστο αριθμό προσπαθειών βελτίωσης πριν την αλλαγή γειτονιάς και αρχικοποιούμε επίσης ως λύση $best\_solution$ την ακολουθία που πήραμε σαν όρισμα $tsp\_sequence$. Όσο το $count$ είναι μικρότερο από τον αριθμό επαναλήψεων ξεκινάει μια επανάληψη για όσο το k είναι μικρότερο απο τον μέγιστο αριθμό προσπαθειών αρχικοποιείται μια μεταβλητή $solution$ με την τιμή του καλύτερο γείτονα της ακολουθίας $best\_solution$. Αν η διαδρομή της ακολοθίας $solution$ είναι μικρότερη αυτής της $best\_solution$ η μεταβλητή $best\_solution$ της παίρνει την τιμή της $solution$ και το $k = 1$ αφού βρέθηκε καλύτερη λύση. Σε αντίθετη περίπτωση αθξάνεται το k κατά 1. Μετά το τέλος αυτής της επανάληψης το $count$ αυξάνετε κατά 1 για να ξανακάνει την επανάληψη. Στο τέλος ο αλγόριθμος επιστρέφει την καλύτερη λύση που βρήκε.

Function `Best neighbour`($tsp\_sequence$, $neighbourhood\_size$):
    $solution \leftarrow tsp\_sequence$;
    for $x \leftarrow 0$ **to** $neighbourhood\_size$ **do**
        $i, j \leftarrow$ *two random cities*;
        $candidate \leftarrow$ *swaped cities i,j in tsp sequence*;
        if *candidate_route_distance* $<$ *solution_route_distance* **then**
            $solution \leftarrow candidate$;
    end
    return solution;
end
Input: A TSP sequence $tsp\_sequence$, $max\_attempts$, $neighbourhood\_size$, $iterations$
Result: The best solution found
$best\_solution \leftarrow tsp\_sequence$;
$count \leftarrow 0$;
$k \leftarrow 1$;
while $count < iteration$ **do**
    while $k < max\_attempts$ **do**
        $solution \leftarrow$ *the best neighbor in best solution*;
        if *tour distance solution* $<$ *tour distance of best solution* **then**
            $best\_solution \leftarrow solution$;
            $k = 1$;
        else
            $k = k + 1$;
        end
    end
    $count \leftarrow count + 1$;
end

Αλγόριθμος 5: Αλγόριθμος variable neighborhood search

Κεφάλαιο 5

Υπολογιστική μελέτη

5.1 Εκπαίδευση του δικτύου

Στην εκπαίδευση του δικτύου χρησιμοποιείται ένα σύνολο από $1,000$ ευκλείδειους tsp γράφους, με τα σημεία να σχεδιάζονται ομοιόμορφα και τυχαία μέσα στο μοναδιαίο τετράγωνο. Χρησιμοποιούνται μικρές παρτίδες 32 ακολουθιών μεγέθους $n = 140$ πόλεων, σε αντίθεση με το [9] που έχει παρτίδες 256 ακολουθιών, μεγέθους $n = 100$ πόλεων. Αυτές οι αλλαγές έγιναν λόγο περιορισμών του συστήματος και ανικανότητας του να επεξεργαστεί παρτίδες 256 ακολουθιών. Ο ηθοποιός και κριτής ενσωματώνουν κάθε πόλη σε χώρο 128 διαστάσεων. Ο κωδικοποιητής αποτελείται από τρεις στοίβες με $h = 16$ παράλληλες κεφαλές και $d = 128$ κρυφές διαστάσεις. Σε κάθε κεφαλή αντιστοιχούν $d_h = d/h = 8$ διαστάσεις.

Η υποστοιβάδα του τροφοδοτικού νευρωνικού δίκτυου *Feedforward neural network (FFN)* έχει διαστάσεις εισόδου και εξόδου d και η εσωτερική στοιβάδα έχει διάσταση $4d = 512$. Ο μηχανισμός προσοχής έχει queries που είναι διανύσματα 360 διαστάσεων ($d' = 360$) και υπολογίζεται σε χώρο 256 διαστάσεων (d" $= 256$). Τα στρώματα τροφοδοσίας του κριτή αποτελούνται από 256 και 1 κρυφές μονάδες. Οι παράμετροι του θ αρχικοποιούνται με το *GlorotUniform initializer* γνωστό και ως *xavier initializer* [12] για να αποφευχθεί ο κορεσμός των μη γραμμικών συναρτήσεων ενεργοποίησης και να διατηρηθεί η κλίμακα των κλίσεων σε όλα τα στρώματα.

Η πιθανότητα καταγραφής του μηχανισμού κατάδειξης οριοθετείται στο διάστημα $[-10, 10]$ και η θερμοκρασία ορίζεται σε $T = 1$ κατά τη διάρκεια της εκπαίδευσης. Χρησιμοποιείται ο *Adam optimizer* [1] [27] για το *Stochastic Gradient Descent*

[1] `https://keras.io/api/optimizers/adam/`

SGD με μεταβλητές $\beta_1 = 0.9$, $\beta_2 = 0.99$, $\epsilon = 10^{-9}$. Ο ρυθμός εκμάθησης, αρχικά είναι 10^{-3} και μειώνεται κατά 0.96 κάθε $5,000$ βήματα. Η εκπαίδευση του μοντέλου έγινε σε μία κάρτα NVIDIA GEFORCE GTX 1050 και διήρκεσε περίπου 3 ώρες. Τα πειράματα διεξάγονται χρησιμοποιώντας το Tensorflow 2.2.0.

5.2 Το σύνολο δεδομένων

Τα δεδομένα που χρησιμοποιήθηκαν για τη δημιουργία των προβλέψεων από το νευρωνικό δίκτυο, είναι διαθέσιμα στο symmetric traveling salesman problem από τη βιβλιοθήκη TSPLIB[2], η οποία είναι μια βιβλιοθήκη με δείγματα του προβλήματος του πλανόδιου πωλητή και άλλα σχετικά προβλήματα, τα οποία προήλθαν από διάφορες πηγές και έχουν διάφορους τύπους. Ο τύπος δεδομένων που υποστηρίζει ο κώδικας για κάθε πόλη $city_i$ είναι της μορφής δισδιάστατων συντεταγμένων x_i, y_i, όπου $x_i, y_i \in \mathbb{R}$. Οπότε η επιλογή των δεδομένων από τη βιβλιοθήκη έγινε σύμφωνα με τις απαιτήσεις του κώδικα. Λόγω χρονικής πολυπλοκότητας των αλγορίθμων και περιορισμένων δυνατοτήτων της συσκευής στην οποία λειτουργούσε, έγινε επιλογή δεδομένων με μέγιστο αριθμό τις 160 πόλεις. Κάποια από τα προβλήματα που επιλέχθηκαν είχαν μεγαλύτερο αριθμό πόλεων, σε αυτά τα προβλήματα επιλέχθηκε τυχαία ο αριθμός πόλεων για να είναι κατάλληλος για τις απαιτήσεις που προαναφέρθηκαν αλλά και το ποιες συντεταγμένες πόλεων θα χρησιμοποιηθούν. Τα αρχεία τα οποία επιλέχθηκαν βρίσκονται στον Πίνακα 5.1.

5.3 Βέλτιστες λύσεις

Οι βέλτιστες λύσεις για κάθε ένα από τα προβλήματα tsp, έχουν βρεθεί χρησιμοποιώντας το λογισμικό Concorde[3], το οποίο είναι ένας κώδικας γραμμένος στην ANSI C γλώσσα προγραμματισμού και δίνει λύση για το συμμετρικό πρόβλημα του πλανόδιου πωλητή και ορισμένα σχετικά προβλήματα βελτιστοποίησης. Το λογισμικό αυτό έχει χρησιμοποιηθεί για να παρέχει τις βέλτιστες λύσεις για κάθε ένα από τις εκατόν δέκα περιπτώσεις που προσφέρει το TSPLIB με τη μεγαλύτερη να περιέχει $85,900$ πόλεις. Τα προβλήματα από το TSPLIB για τα οποία χρειάστηκε

[2] `http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/`
[3] `https://www.math.uwaterloo.ca/tsp/concorde.html`

Πρόβλημα	**Αριθμός πόλεων**	**Πλήθος πόλεων**	**Βέλτιστη λύση**
pr76	76	Όλες	108,159
st70	70	Όλες	675
pr654	73	Οι πρώτες 73	19,143
eil101	101	Όλες	629
fl417	109	Οι πρώτες 109	7,380
fl1400	123	1-32 ,180-264, 1382-1400	12,294
pr264	97	Οι πρώτες 97	17,274
fl1577	94	Οι πρώτες 94	1,215
pr136	86	Οι πρώτες 86	65,377
att532	105	Οι πρώτες 105	15,482
a280	40	Οι πρώτες 40	553
pr144	47	Οι πρώτες 47	26,057
pr152	50	Οι πρώτες 50	36,176
pr299	55	Οι πρώτες 55	13,790
pr439	45	Οι πρώτες 45	12,003

Πίνακας 5.1: Δεδομένα

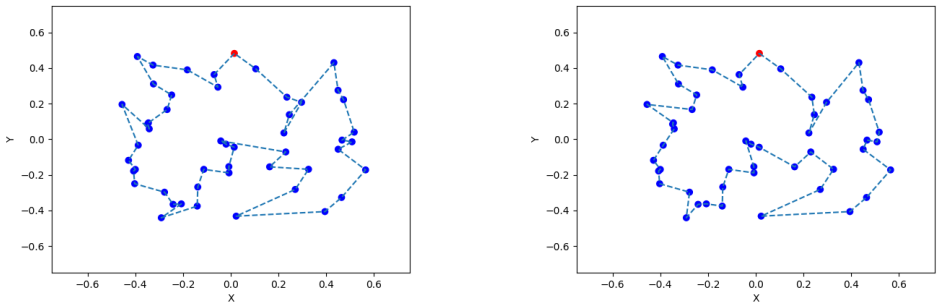
να γίνει περικοπή αριθμού πόλεων και επιλογή τυχαίων συντεταγμένων, έχουν περαστεί στο Concorde και έχει βρεθεί η καινούρια βέλτιστη λύση σύμφωνα με το νέο πρόβλημα που δημιουργήθηκε.

5.4 Προδιαγραφές συστήματος

Οι προδιαγραφές του συστήματος που έγινε η εκπαίδευση του δικτύου αλλά και η εξέταση των δεδομένων για τη δημιουργία προβλέψεων είναι οι εξής:

- Κάρτα γραφικών: GeForce GTX 1050
- Επεξεργαστής: Intel® Core™ i7-8750H CPU @ 2.20GHz × 12
- Λειτουργικό σύστημα: Ubuntu 18.04.5 LTS
- Τύπος λειτουργικού: 64-bit

Η γλώσσα προγραμματισμού που χρησιμοποιήθηκε για την υλοποίηση του νευρωνικού δικτύου αλλά και των αλγορίθμων είναι η *Python* [41].



(α′) Πρόβλεψη δικτύου (β′) Με χρήση 2opt

Σχήμα 5.1: Πρόβλεψη και βελτιστοποίηση 50 πόλεων με το νευρωνικό δίκτυο

5.5 Αποτελέσματα νευρωνικού δικτύου

Το εκπαιδευμένο νευρωνικό δίκτυο με τις μεταβλητές που δίνονται στην Ενότητα 5.1, εξετάζεται με τον τρόπο που προτείνεται στο [9]. Το δίκτυο παίρνει σαν είσοδο, ένα τυχαία κατανεμημένο, στο μοναδιαίο τετράγωνο, σύνολο 50 πόλεων. Τα αποτελέσματα τα οποία δίνει, είναι ο μέσος όρος 10 δοκιμών και είναι τα εξής:

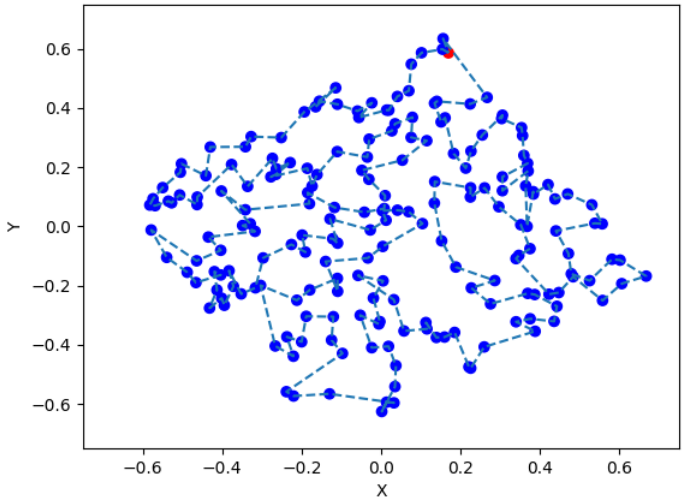
- Πρόβλεψη νευρωνικού δικτύου: 6.48
- Βελτιστοποίηση με 2opt: 6.05

Στο Σχήμα 5.1 δίνονται οι γράφοι των διαδρομών για τις παραπάνω τιμές. Ως συμπέρασμα ο αλγόριθμος 2opt βελτιώνει την πρόβλεψη του νευρωνικού κατά 7.1%

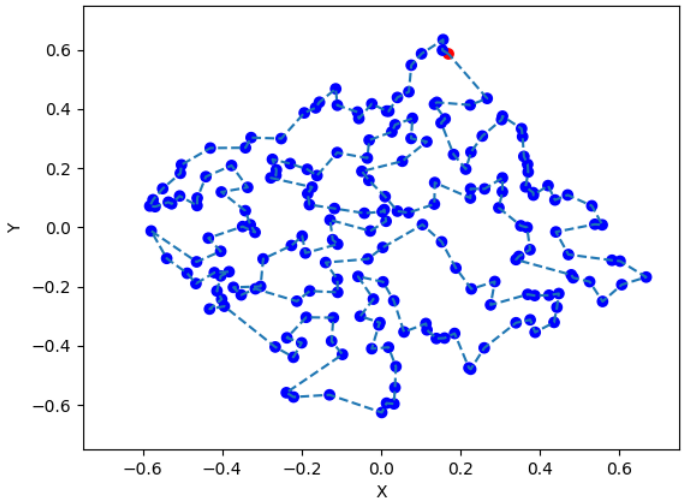
Στο επόμενο πείραμα αυξάνουμε το σύνολο στις 200 πόλεις και πάλι τα αποτελέσματα τα οποία δίνει, είναι ο μέσος όρος 10 δοκιμών και είναι τα εξής:

- Πρόβλεψη νευρωνικού δικτύου: 13.22
- Βελτιστοποίηση με 2opt: 11.66

Στο Σχήμα 5.2 δίνονται οι γράφοι των διαδρομών για τις παραπάνω τιμές. Ως συμπέρασμα ο αλγόριθμος 2opt βελτιώνει την πρόβλεψη του νευρωνικού κατά 13.77%.



(α′) Πρόβλεψη δικτύου



(β′) Με χρήση 2opt

Σχήμα 5.2: Πρόβλεψη και βελτιστοποίηση 200 πόλεων με το νευρωνικό δίκτυο

5.6 Αποτελέσματα αλγορίθμων βελτιστοποίησης

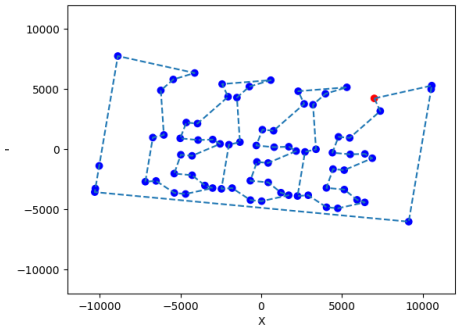
Τα αποτελέσματα της διαδρομής και του χρόνου που δίνονται στον Πίνακα 5.2 έχουν υπολογιστεί τρέχοντας κάθε αλγόριθμο (ακριβώς μετά την εφαρμογή του νευρωνικού δικτύου) σε κάθε πρόβλημα πέντε φορές και υπολογίζοντας τον μέσο όρο από τις εξόδους, διαδρομής και χρόνου. Οι μεταβλητές που έτρεξε ο κάθε αλγόριθμος είναι:

- 2opt: $iterations = 4,000$
- 3opt: $iterations = 4,000$
- Simulated annealing: $alpha = 0.99999$, $T = 10$, $stopping\_temperature = 1e-8$, $stopping\_iter = 500,000$
- Variable neighborhood search descent: $max\_attempts = 1,000$, $neighbourhood\_size = 20$, $iterations = 100,000$

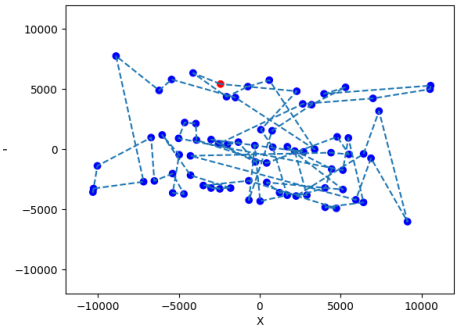
Ως παράδειγμα, στο Σχήμα 5.3 δίνεται η διαδρομή σε γράφους για μια εκτέλεση του προβλήματος pr76 και στο Σχήμα 5.4 η διαδρομή σε γράφους για μια εκτέλεση του προβλήματος st70.

Για να εξεταστεί η σταθερότητα στις προβλέψεις του νευρωνικού και στη βελτίωση των αλγορίθμων έχει επεξεργαστεί το σύνολο των πέντε δεδομένων διαδρομής από δύο προβλήματα, το eil101 και το a280, ως θηκόγραμμα (box plot) στο Σχήμα 5.5. Όπως παρατηρείται, οι προβλέψεις του νευρωνικού δικτύου έχουν τη μεγαλύτερη απόκλιση μεταξύ τους και στα δύο προβλήματα. Επίσης μεγάλη απόκλιση έχει και ο nearest neighbor, ενώ οι βελτιωτικοί αλγόριθμοι έχουν πιο σταθερά αποτελέσματα βελτίωσης.

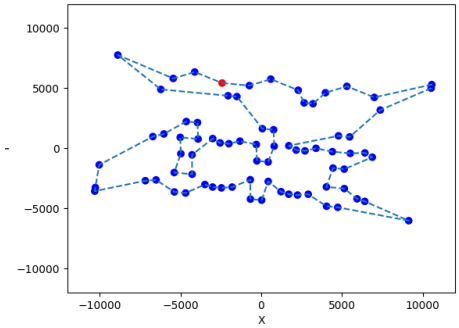
Τα αποτελέσματα χρόνου για τον αλγόριθμο nearest neighbor που στον πίνακα εμφανίζονται με 0.00 δεν είναι μηδενικά αλλά είναι μικρότερα από ακρίβεια δύο δεκαδικών ψηφίων. Το ίδιο ισχύει και για τα αποτελέσματα της απόκλισης μερικών από τους βελτιωτικούς αλγορίθμους. Η διαδρομή που επιστρέφεται σαν αποτέλεσμα είναι τόσο κοντά στη βέλτιστη που η απόκλιση είναι μικρότερη από δύο δεκαδικά ψηφία.



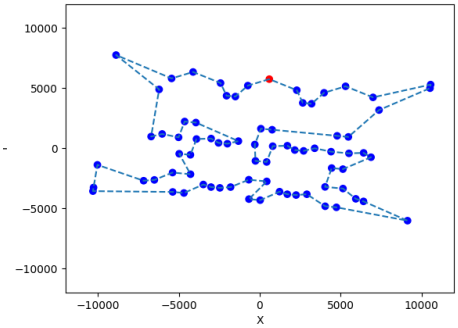
(α΄) Δεδομένα



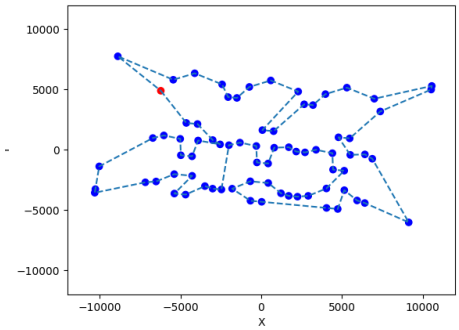
(β΄) Πρόβλεψη νευρωνικού



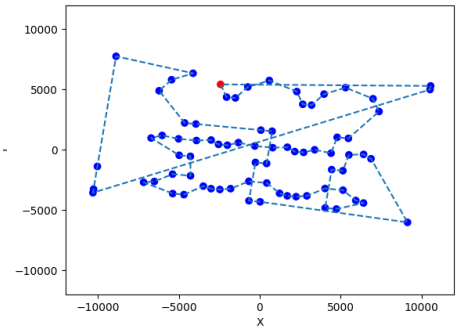
(γ΄) 2opt



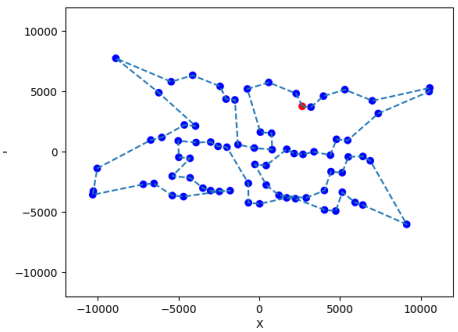
(δ΄) 3opt



(ε΄) SA

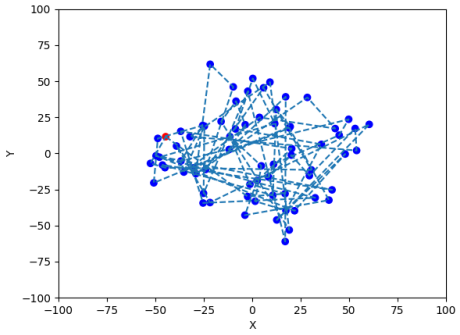


(στ΄) NN

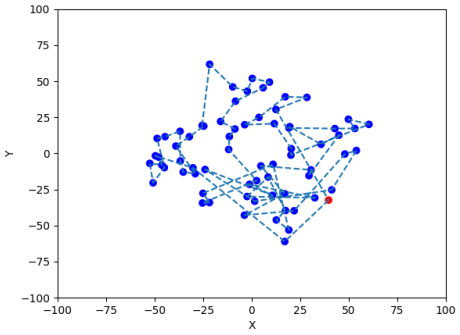


(ζ΄) VND

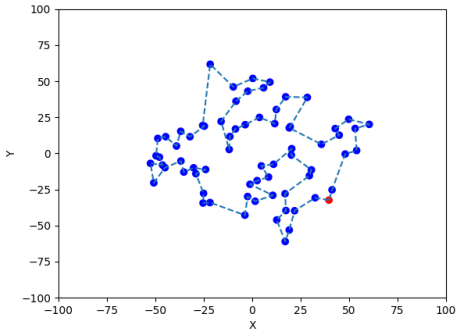
Σχήμα 5.3: Γράφοι για το πρόβλημα pr76



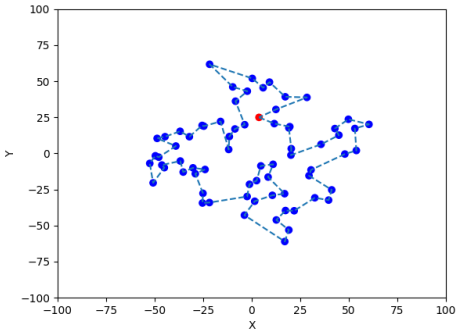
(α΄) Δεδομένα



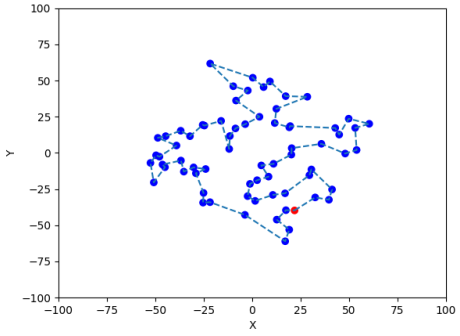
(β΄) Πρόβλεψη νευρωνικού



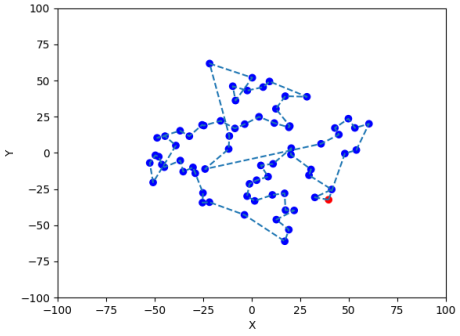
(γ΄) 2opt



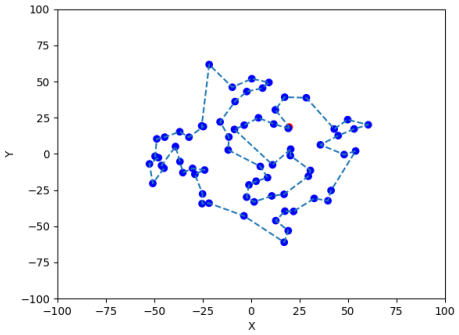
(δ΄) 3opt



(ε΄) SA

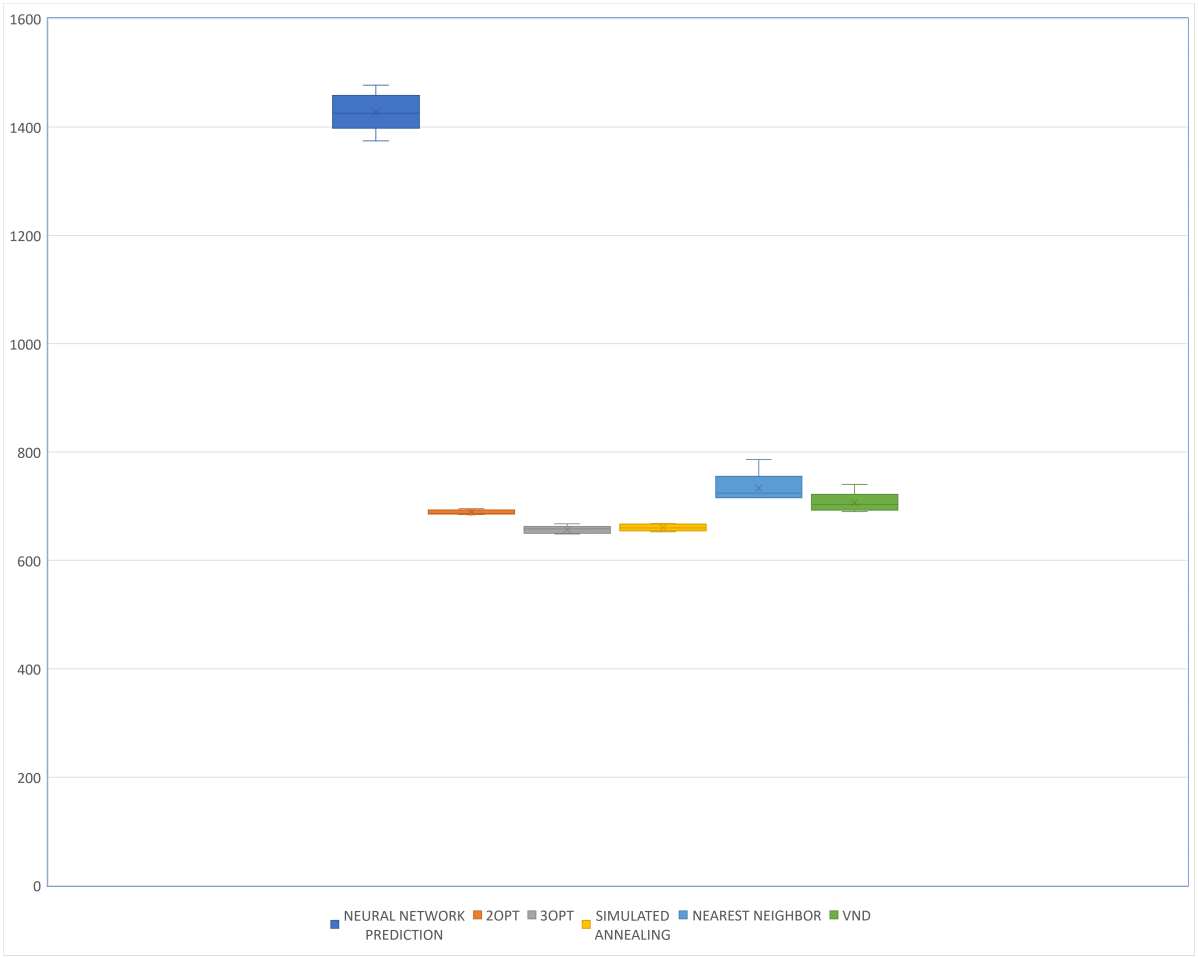


(στ΄) NN

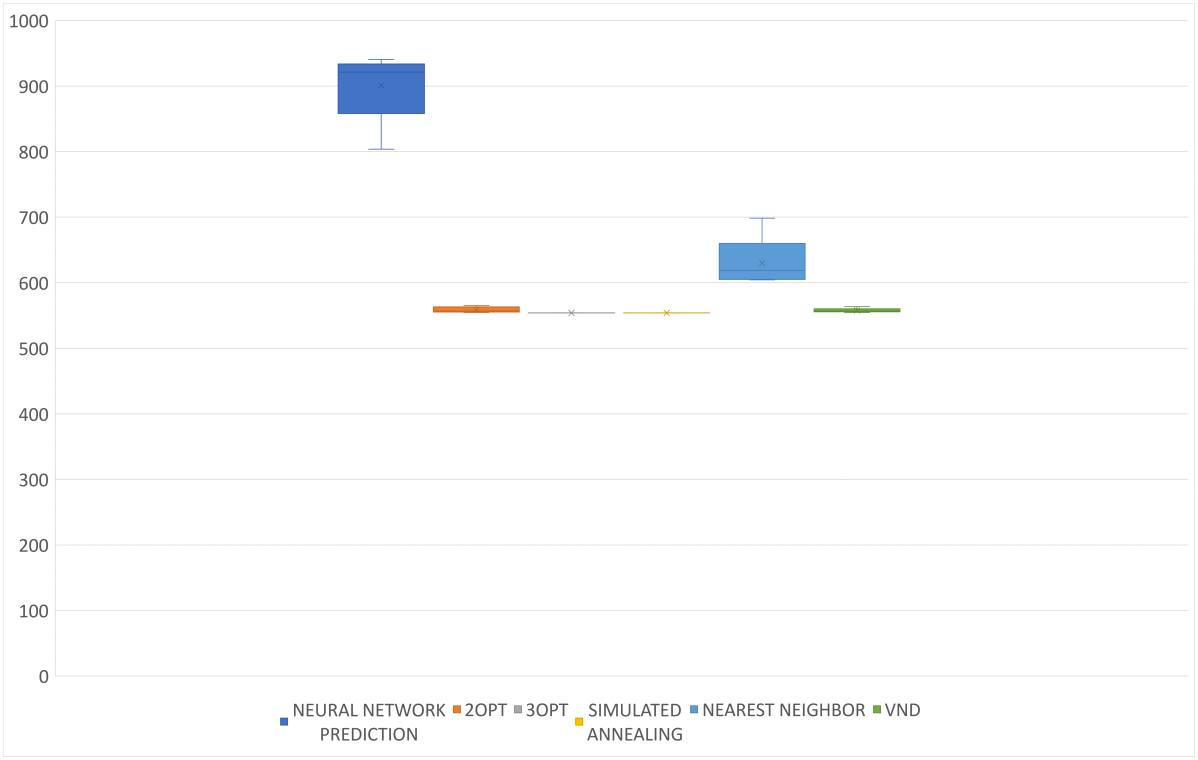


(ζ΄) VND

Σχήμα 5.4: Γράφοι για το πρόβλημα st70



(α′) eil101



(β′) a280

Σχήμα 5.5: Διακύμανση αποτελεσμάτων

Πίνακας 5.2: Αποτελέσματα αλγορίθμων

Πρόβλημα	Αλγόριθμος	Αποτελέσματα	Χρόνος (sec)	Βέλτιστη λύση	Απόκλιση
pr76	Νευρωνικό	256,219.05	-	108,159	100%
	2OPT	113,683.41	2.33		5%
	3OPT	110,292.41	220.80		2%
	SA	115,509.90	33.44		7%
	NN	143,329.00	0.01		33%
	VND	118,069.43	2.68		9%
st70	Νευρωνικό	1,375.79	-	675	100%
	2OPT	725.37	1.86		7%
	3OPT	688.91	82.83		2%
	SA	688.28	36.62		2%
	NN	822.22	0.01		22%
	VND	732.21	2.34		8 %
pr654	Νευρωνικό	35,726.45	-	19,143	87%
	2OPT	19,581.29	2.09		2%
	3OPT	19,334.98	85.30		1%
	SA	19,559.90	34.26		2%
	NN	23,249.03	0.01		21%
	VND	19,502.38	2.77		2%
eil101	Νευρωνικό	1,427.46	-	629	100%
	2OPT	688.87	8.78		10%
	3OPT	656.81	490.50		4%
	SA	660.70	44.00		5%
	NN	733.12	0.02		17%
	VND	706.58	4.29		12%
fl417	Νευρωνικό	16,196.29	-	7,380	100%
	2OPT	7,829.62	8.40		6%
	3OPT	7,492.82	531.37		2%
	SA	7,821.47	126.88		6%
	NN	8,707.96	0.02		18%
	VND	7,945.12	4.15		8%
fl1400	Νευρωνικό	27,808.38	-	12,294	100%
	2OPT	13,022.84	17.68		6%
	3OPT	12,414.39	585.65		1%
	SA	12,900.86	38.44		5%
	NN	15,882.74	0.02		29%
	VND	12,977.86	6.52		6%

pr264	Νευρωνικό	34,171.38	-	17,274	98%
	2OPT	17,606.42	4.65		2%
	3OPT	17,362.48	465.71		1%
	SA	17,906.92	81.99		4%
	NN	19,539.89	0.01		13%
	VND	18,108.72	3.72		5%
fl1577	Νευρωνικό	2,496.14	-	1,215	100%
	2OPT	1,305.46	5.20		7%
	3OPT	1,228.38	416.83		1%
	SA	1,249.89	21.47		3%
	NN	1,722.83	0.01		42%
	VND	1,293.81	3.60		6%
pr136	Νευρωνικό	131,784.64	-	65,377	100%
	2OPT	68,529.47	3.00		5%
	3OPT	66,183.56	190.70		1%
	SA	69,326.51	22.92		6%
	NN	81,317.21	0.02		24%
	VND	68,919.61	3.48		5%
att532	Νευρωνικό	39,644.30	-	15,482	100%
	2OPT	15,996.20	11.23		3%
	3OPT	15,646.58	373.23		1%
	SA	16,286.33	25.72		5%
	NN	18,541.91	0.01		20%
	VND	16,536.96	4.75		7%
a280	Νευρωνικό	901.00	-	553	63%
	2OPT	558.80	0.23		1%
	3OPT	553.92	5.81		0%
	SA	582.80	17.85		5%
	NN	629.79	0.00		0.14%
	VND	557.78	1.34		1%
pr144	Νευρωνικό	45,126.47	-	26,057	73%
	2OPT	28,664.73	0.30		10%
	3OPT	26,097.93	16.93		0%
	SA	26,892.73	18.53		3%
	NN	28,622.09	0.00		10%
	VND	27,365.90	1.58		5%
	Νευρωνικό	66,661.67	-		84%
	2OPT	36,779.50	0.72		2%

pr152	3OPT	36,176.32	22.28	36,176	0%
	SA	36,866.16	15.24		2%
	NN	38,802.21	0.00		7%
	VND	36,690.91	1.77		1%
pr299	Νευρωνικό	20,182.54	-	13,790.00	46%
	2OPT	13,819.46	0.78		0%
	3OPT	13,790.21	12.61		0%
	SA	13,800.31	15.05		0%
	NN	14,807.20	0.00		7%
	VND	13,825.35	1.93		0%
pr439	Νευρωνικό	15,990.62	–	12,003.00	33%
	2OPT	12,246.62	0.33		2%
	3OPT	12,003.70	20.43		0%
	SA	12,192.96	15.22		2%
	NN	13,725.82	0.00		14%
	VND	12,164.57	1.35		1%
Γεωμετρικός μέσος προβλημάτων	Νευρωνικό	17,111.07	–	-	91%
	2OPT	9,197.06	2.18		3%
	3OPT	8,891.39	104.22		1%
	SA	9,128.60	29.44		3%
	NN	10,474.51	0.01		17%
	VND	9,246.81	2.77		4%

Οι βελτιωτικοί αλγόριθμοι που εξετάστηκαν καταφέρνουν όλοι να βελτιώσουν την έξοδο του νευρωνικού δικτύου που τους δόθηκε σαν αρχική λύση, με διαφορετική αποτελεσματικότητα ο καθένας. Επίσης, και ο κατασκευαστικός αλγόριθμος *nearest neighbor* δίνει μια λύση διαδρομής, η οποία είναι καλύτερη από αυτή του νευρωνικού δικτύου.

Όπως παρατηρείται, από τους βελτιωτικούς αλγόριθμους, αυτός που έχει τα καλύτερα αποτελέσματα διαδρομής σε όλα τα προβλήματα, εκτός του *st70*, είναι ο *3opt*, αλλά έχει επίσης και τον μεγαλύτερο χρόνο εκτέλεσης, εκτός από τα προβλήματα a280 και pr144 που έχει τον δεύτερο μεγαλύτερο μετά από του simulated annealing. Εξετάζοντας τα αποτελέσματα όταν το σύνολο εισόδου είναι κάτω των 60 πόλεων (προβλήματα a280, pr144, pr152, pr299, pr439), ο αλγόριθμος 3opt βελ-

τιώνει την έξοδο σχεδόν βέλτιστα έχοντας τιμή απόκλισης, που τείνει στο 0. Στα προβλήματα αυτά επίσης ο χρόνος εκτέλεσης του αλγορίθμου έχει μειωθεί αισθητά.

Αν ο χρόνος εκτέλεσης είναι σημαντικός περιορισμός, τις επόμενες καλύτερες λύσεις στα προβλήματα που ο αλγόριθμος 3opt έχει μεγάλο χρόνο εκτέλεσης, τις δίνει ο αλγόριθμος 2opt στα προβλήματα pr76, pr264, pr136 και att532, ο αλγόριθμος simulated annealing στα προβλήματα eil101, fl417, fl1400 και fl1577 και ο αλγόριθμος VND στα προβλήματα pr654, pr152 και pr439. Το πρόβλημα pr144 έχει παρόμοιο χρόνο εκτέλεσης στους αλγορίθμους 3opt και simulated annealing. Οι αλγόριθμοι αυτοί δίνουν τα δύο καλύτερα αποτελέσματα στο πρόβλημα αλλά αν επιθυμείται επιπλέον μείωση χρόνου εκτέλεσης το αμέσως καλύτερο αποτέλεσμα το δίνει ο αλγόριθμος VND. Τέλος, στο πρόβλημα a280, ο 3opt έχει μικρό χρόνο εκτέλεσης και την καλύτερη βελτίωση, αλλά αν και εκεί επιθυμείτε περαιτέρω μείωση του χρόνου, την αμέσως καλύτερη βελτίωση την έχει ο αλγόριθμος VND.

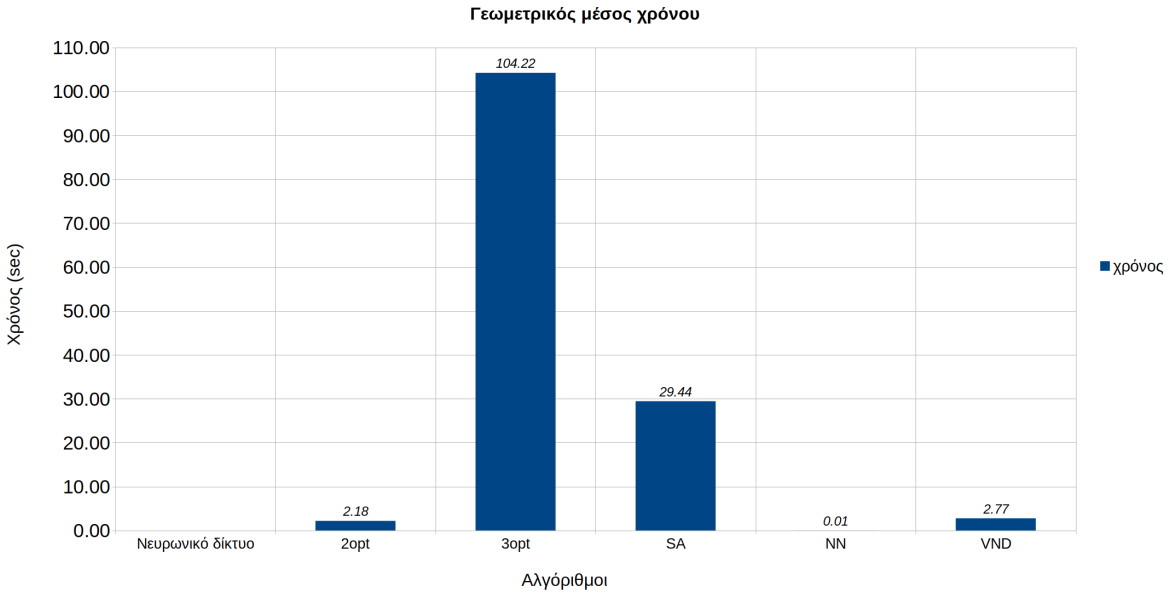
Στο πρόβλημα st70 τα καλύτερα αποτελέσματα διαδρομής τα έχει ο αλγόριθμος *simulated annealing* με διαδρομή 688.28 και εκτελείται σε χρόνο 36.62sec, σε αντίθεση με τον δεύτερο καλύτερο, ο οποίος είναι ο 3opt με διαδρομή 688.91 και χρόνο εκτέλεσης 82.83sec. Σε αυτό το πρόβλημα και οι δύο αυτοί αλγόριθμοι έχουν απόκλιση από τη βέλτιστη λύση 2%. Στο πρόβλημα pr299 παρατηρείται ότι όλοι οι βελτιωτικοί αλγόριθμοι έχουν βρει μια λύση τόσο κοντά στη βέλτιστη, που για όλους η απόκλιση τείνει στο 0%.

Ο κατασκευαστικός αλγόριθμος *nearest neighbor*, παρόλο που έχει τον μικρότερο χρόνο εκτέλεσης, έχει τα χειρότερα αποτελέσματα διαδρομής από όλους τους υπόλοιπους αλγορίθμους πέρα από την έξοδο του νευρωνικού σε όλα τα προβλήματα tsp.

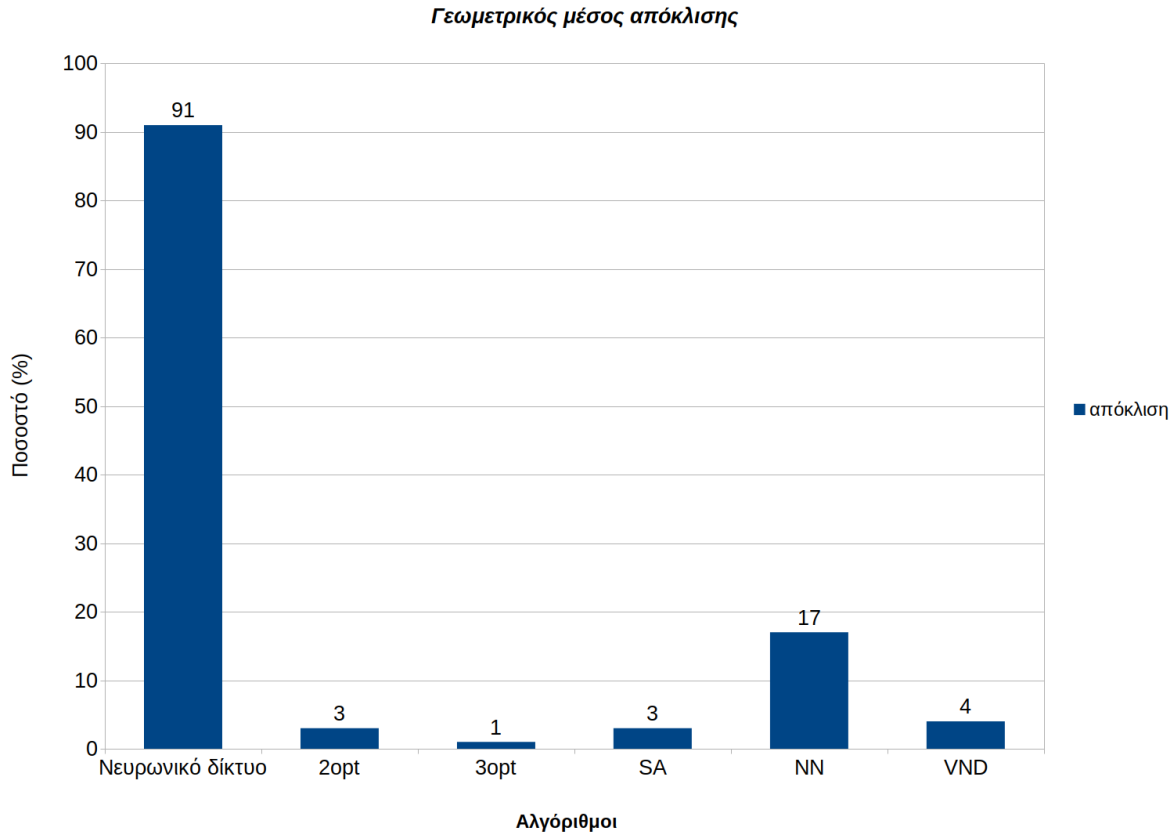
Παρατηρείται επίσης ότι οι προβλέψεις του νευρωνικού δικτύου απέχουν αρκετά από τις βέλτιστες λύσεις σε μεγαλύτερα προβλήματα tsp (πάνω από 60 πόλεις). Τα προβλήματα pr76, st70, eil101, fl417, fl1400, fl1577, pr136, att532 έχουν απόκλιση από τη βέλτιστη 100%. Ενώ το πρόβλημα pr654 έχει απόκλιση 87% και το pr264 98%. Αντίθετα σε προβλήματα με μικρότερο σύνολο πόλεων (κάτω από 60 πόλεις) το νευρωνικό δίκτυο κάνει καλύτερες προβλέψεις και φτάνει ακόμα και την απόκλιση στο 33% στο πρόβλημα pr439.

Οι γεωμετρικοί μέσοι του χρόνου και της απόκλισης από τον Πίνακα 5.2 πα-

ρουσιάζονται σαν γραφήματα (Σχήμα 5.6) και δίνουν μια καλύτερη εικόνα από τη συνολική αποτελεσματικότητα του κάθε αλγορίθμου για όλα τα προβλήματα.



(α΄) Γεωμετρικός μέσος χρόνου



(β΄) Γεωμετρικός μέσος απόκλισης

Σχήμα 5.6: Γεωμετρικοί μέσοι

Κεφάλαιο 6

Συμπεράσματα

Σε αυτή τη διπλωματική, μελετήθηκε η επίλυση του προβλήματος του πλανόδιου πωλητή με χρήση νευρωνικών δικτύων και αλγορίθμους μηχανικής μάθησης. Έγινε ανάλυση και εκπαίδευση του νευρωνικού δικτύου που προτάθηκε από τους Deudon et al. [9] με παρτίδες 32 ακολουθιών με μέγεθος πόλεων $n = 140$ και εφαρμόστηκαν οι βελτιωτικοί αλγόριθμοι 2opt, 3opt, simulated annealing, variable neighborhood descent και ο κατασκευαστικός αλγόριθμος nearest neighbor ακριβώς μετά την εφαρμογή του νευρωνικού δικτύου.

Στην υπολογιστική μελέτη του νευρωνικού δικτύου παρατηρούμε ότι το δίκτυο εμφανίζει καλά αποτελέσματα, όταν το σύνολο των πόλεων που παίρνει σαν είσοδο, είναι τυχαίες πόλεις κατανεμημένες ομοιόμορφα στο μοναδιαίο τετράγωνο. Αντίθετα, τα αποτελέσματα των προβλέψεων του στα προβλήματα του TSPLIB δεν έχουν την ίδια αποδοτικότητα αφού έχει γεωμετρικό μέσο για όλα τα προβλήματα 91% απόκλιση από τη βέλτιστη λύση.

Παρόλο που το δίκτυο δεν αποδίδει και πολύ καλά παρατηρείται ότι οι βελτιωτικοί αλγόριθμοι καταφέρνουν να βελτιώσουν την έξοδο του νευρωνικού σε αρκετά καλό βαθμό, με τον αλγόριθμο *3opt* να έχει συνολικά την καλύτερη βελτίωση, αγγίζοντας τη βέλτιστη διαδρομή με μέση απόκλιση 1%, αλλά έχοντας επίσης και τον μεγαλύτερο μέσο χρόνο εκτέλεσης 104.22 δευτερόλεπτα. Ο 2opt και simulated annealing δίνουν μέση απόκλιση της κλίμακας του 3% με τον 2opt να έχει συγκριτικά μικρότερο μέσο χρόνο εκτέλεσης 2.18 δευτερόλεπτα έναντι 29.44 δευτερόλεπτα.

Αμέσως μετά ο αλγόριθμος VND έχοντας καλύτερο μέσο χρόνο εκτέλεσης από τον 3opt και simulated annealing στα 2.77 δευτερόλεπτα δίνει μέση απόκλιση 4%. Παρατηρείται επίσης ότι ο μοναδικός κατασκευαστικός αλγόριθμος έχει τον μικρό-

τερο μέσο χρόνο εκτέλεσης της τάξης του 0.01%, και καταφέρνει να κατασκευάσει διαδρομή μικρότερη από αυτή του νευρωνικού με μέση απόκλιση 4%.

Βιβλιογραφία

[1] Muhammed Al-Mulhem and Tareq Al-Maghrabi. Efficient convex-elastic net algorithm to solve the euclidean traveling salesman problem. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 28(4):618–620, 1998.

[2] Necati Aras, B John Oommen, and İK Altınel. The kohonen network incorporating explicit statistics and its application to the travelling salesman problem. *Neural Networks*, 12(9):1273–1284, 1999.

[3] Irwan Bello, Hieu Pham, Quoc V Le, Mohammad Norouzi, and Samy Bengio. Neural combinatorial optimization with reinforcement learning. *arXiv preprint arXiv:1611.09940*, 2016.

[4] Andrius Blazinskas and Alfonsas Misevicius. combining 2-opt, 3-opt and 4-opt with k-swap-kick perturbations for the traveling salesman problem. *Kaunas University of Technology, Department of Multimedia Engineering, Studentu St*, pages 50–401, 2011.

[5] Geir Brønmo, Marielle Christiansen, Kjetil Fagerholt, and Bjørn Nygreen. A multi-start local search heuristic for ship scheduling—a computational study. *Computers & Operations Research*, 34(3):900–917, 2007.

[6] Marco Budinich. A self-organizing neural network for the traveling salesman problem that is competitive with simulated annealing. *Neural Computation*, 8(2):416–424, 1996.

[7] EM Cochrane and John E Beasley. The co-adaptive neural network approach to the euclidean travelling salesman problem. *Neural Networks*, 16(10):1499–1525, 2003.

[8] Jean-Charles Créput and Abderrafiaa Koukam. A memetic neural network for the euclidean traveling salesman problem. *Neurocomputing*, 72(4-6):1250–1264, 2009.

[9] Michel Deudon, Pierre Cournut, Alexandre Lacoste, Yossiri Adulyasak, and Louis-Martin Rousseau. Learning heuristics for the tsp by policy gradient. In *International Conference on the Integration of Constraint Programming, Artificial Intelligence, and Operations Research*, pages 170–181. Springer, 2018.

[10] Kathryn Anne Dowsland and Jonathan Thompson. Simulated annealing. *Handbook of Natural Computing*, pages 1623–1655, 2012.

[11] Richard Durbin and David Willshaw. An analogue approach to the travelling salesman problem using an elastic net method. *Nature*, 326(6114):689–691, 1987.

[12] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the 13th International Conference on Artificial Intelligence and Statistics*, pages 249–256. JMLR Workshop and Conference Proceedings, 2010.

[13] Fred Glover and Manuel Laguna. Tabu search. In *Handbook of Combinatorial Optimization*, pages 2093–2229. Springer, 1998.

[14] Radosław Grymin and Szymon Jagiełło. Fast branch and bound algorithm for the travelling salesman problem. In Khalid Saeed and Władysław Homenda, editors, *Computer Information Systems and Industrial Management*, pages 206–217, Cham, 2016. Springer International Publishing.

[15] Gregory Gutin and Anders Yeo. The greedy algorithm for the symmetric tsp. *Algorithmic Operations Research*, 2(1):33–36, 2007.

[16] A Hanif Halim and I Ismail. Nonlinear plant modeling using neuro-fuzzy system with tree physiology optimization. In *2013 IEEE Student Conference on Research and Developement*, pages 295–300. IEEE, 2013.

[17] A Hanif Halim and IJAoCMiE Ismail. Combinatorial optimization: comparison of heuristic algorithms in travelling salesman problem. *Archives of Computational Methods in Engineering*, 26(2):367–380, 2019.

[18] Michael Held and Richard M Karp. A dynamic programming approach to sequencing problems. *Journal of the Society for Industrial and Applied Mathematics*, 10(1):196–210, 1962.

[19] John Henry Holland. *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence*. MIT press, 1992.

[20] John J Hopfield and David W Tank. "neural" computation of decisions in optimization problems. *Biological Cybernetics*, 52(3):141–152, 1985.

[21] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning*, pages 448–456. PMLR, 2015.

[22] Hui-Dong Jin, Kwong-Sak Leung, Man-Leung Wong, and Z-B Xu. An efficient self-organizing map designed by genetic algorithms for the traveling salesman problem. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 33(6):877–888, 2003.

[23] Ameet V. Joshi. *Introduction to AI and ML*, pages 3–7. Springer International Publishing, Cham, 2020.

[24] Michael Jünger, Gerhard Reinelt, and Giovanni Rinaldi. The traveling salesman problem. *Handbooks in Operations Research and Management Science*, 7:225–330, 1995.

[25] Richard M Karp. Reducibility among combinatorial problems. In *Complexity of Computer Computations*, pages 85–103. Springer, 1972.

[26] Indadul Khan, Manas Kumar Maiti, and Manoranjan Maiti. Coordinating particle swarm optimization, ant colony optimization and k-opt algorithm for traveling salesman problem. In *International Conference on Mathematics and Computing*, pages 103–119. Springer, 2017.

[27] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[28] Gözde Kizilateş and Fidan Nuriyeva. On the nearest neighbor algorithms for the traveling salesman problem. In *Advances in Computational Science, Engineering and Information Technology*, pages 111–118. Springer, 2013.

[29] Teuvo Kohonen and Manfred Schroeder. *Self-Organizing Maps*. 01 2001.

[30] Παναγιώτης Γ Λαμπάτος. προσεγγιστικοί αλγόριθμοι του προβλήματος του περιοδεύοντος πωλητή. Master's thesis, 2014.

[31] Kwong-Sak Leung, Hui-Dong Jin, and Zong-Ben Xu. An expanding self-organizing neural network for the traveling salesman problem. *Neurocomputing*, 62:267–292, 2004.

[32] Li Li, Yurong Cheng, Lijing Tan, and Ben Niu. A discrete artificial bee colony algorithm for tsp problem. In *International Conference on Intelligent Computing*, pages 566–573. Springer, 2011.

[33] Rajesh Matai, Surya Prakash Singh, and Murari Lal Mittal. Traveling salesman problem: an overview of applications, formulations, and solution approaches. *Traveling Salesman Problem, Theory and Applications*, 1, 2010.

[34] Nenad Mladenović and Pierre Hansen. Variable neighborhood search. *Computers & Operations Research*, 24(11):1097–1100, 1997.

[35] Mehryar Mohri, Afshin Rostamizadeh, and Ameet Talwalkar. *Foundations of machine learning*. MIT press, 2018.

[36] Luc Muyldermans, Patrick Beullens, Dirk Cattrysse, and Dirk Van Oudheusden. Exploring variants of 2-opt and 3-opt for the general routing problem. *Operations Research*, 53(6):982–995, 2005.

[37] Christos H Papadimitriou. Np-completeness: A retrospective. In *International Colloquium on Automata, Languages, and Programming*, pages 2–6. Springer, 1997.

[38] Christos H Papadimitriou and Kenneth Steiglitz. Combinatorial optimization: algorithms and complexity. 1998.

[39] Ewa Skubalska-Rafajłowicz. Exploring the solution space of the euclidean traveling salesman problem using a kohonen som neural network. In *International Conference on Artificial Intelligence and Soft Computing*, pages 165–174. Springer, 2017.

[40] Jan van Leeuwen and Anneke A Schoone. *Untangling a traveling salesman tour in the plane*, volume 80. Technical report, 1980.

[41] Guido Van Rossum et al. Python programming language. In *USENIX Annual Technical Conference*, volume 41, 2007.

[42] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008, 2017.

[43] Sun-Chong Wang. Artificial neural network. In *Interdisciplinary Computing in Java Programming*, pages 81–100. Springer, 2003.

[44] Mutsunori Yagiura and Toshihide Ibaraki. On metaheuristic algorithms for combinatorial optimization problems. *Systems and Computers in Japan*, 32(3):33–55, 2001.

[45] Xin-She Yang. *Nature-inspired metaheuristic algorithms*. Luniver press, 2010.

[46] Xian-Da Zhang. Machine learning. In *A Matrix Algebra Approach to Artificial Intelligence*, pages 223–440. Springer, 2020.