ΑΛΕΞΑΝΔΡΕΙΟ ΤΕΧΝΟΛΟΓΙΚΟ ΕΚΠΑΙΔΕΥΤΙΚΟ ΙΔΡΥΜΑ ΘΕΣΣΑΛΟΝΙΚΗΣ
ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΚΩΝ ΕΦΑΡΜΟΓΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Πτυχιακή Εργασία

Επίλυση και Απεικόνιση στο Χάρτη του Προβλήματος του Περιοδεύοντος Αντιπροσώπου (Traveling Salesman Problem) με τη χρήση Νευρωνικού Δικτύου

Του φοιτητή
Τσαμσακίζογλου Χρήστου

Αρ. Μητρώου: 04/2494

Επιβλέπων καθηγητής
Γουλιάνας Κωνσταντίνος

Μάρτιος 2010

Περίληψη

Τα Τεχνητά Νευρωνικά Δίκτυα είναι ένας κλάδος ο οποίος άρχισε να αναπτύσεται ραγδαία τις τελευταίες δεκαετίες, και ανήκει στο γενικότερο χώρο της επιστήμης της τεχνητής νοημοσύνης. Επειδή ο τρόπος λειτουργίας τους προσπαθεί να προσομοιάσει αυτόν του ανθρώπινου εγκεφάλου, η ανάπτυξη τους άνοιξε καινούργιους δρόμους στην επίλυση δύσκολων προβλημάτων.

Ένα τέτοιο πρόβλημα είναι αυτό του περιοδεύοντος πωλητή. Το πρόβλημα αυτό είναι ένα πρόβλημα βελτιστοποίησης το οποίο λόγο των ιδιαιτεροτήτων του χρησιμοποιήται για τη συγκριτική μελέτη πολλών αλγορίθμων. Αφορά την εύρεση του συντομότερου δρόμου απο ένα πλήθος πόλεων περνώντας από κάθε πόλη ακριβώς μια φορά και η ιδιαιτερότητα του είναι ότι αν και η επίλυση του χρειάζεται πολύ υπολογιστικό χρόνο η επαλύθευση της λύσης είναι πολύ γρήγορη.

Στην τρέχουσα πτυχιακή γίνεται μια προσπάθεια να κατασκευαστεί ένας αλγόριθμος βελτιστοποίησης με βάση το νευρωνικό δίκτυο Hopfield, συνδιάζοντας υπάρχοντα μοντέλα, καθώς και μια γραφική διεπαφή ώστε να είναι εύκολη η κατανόηση και η παρατήρηση των αποτελεσμάτων.

Οι παρατηρήσεις των πειραμάτων αναφέρονται και καταγράφονται αναλυτικά.

Alexander Technological Institute of Thessaloniki
Information Technology Faculty
Department of Technological Applications

Abstract

Tsamsakizoglou Christos

Artificial neural Networks is a scientific branch which began to develop rapidly in the recent decades, and belongs to the general science of artificial intelligence. Because the way the neural networks work is by simulating the human brain their development has opened new ways of solving difficult problems.
Such a problem is the Traveling Salesman Problem (TSP). This is an optimization problem and is one of the most intensively studied problems in optimization and it is used as a benchmark for many optimization methods. Its definition is: *Given a list of cities and their pairwise distances, find the shortest possible tour that visits each city exactly once*. Even though the problem is computationally difficult, the final solution can be checked quickly. In this thesis we try to create a Hopfield type network for solving the TSP by combining existing models. Furthermore we are going to design a graphical interface for observation and capturing the results.
The experiments will be recorded and analyzed.

Περιεχόμενα

Κατάλογος σχημάτων

Κεφάλαιο 1

Τεχνητά νευρωνικά δίκτυα

1.1 Εισαγωγή

Τα νευρωνικά δίκτυα αποτελούν μια σχετικά νέα περιοχή στις τεχνολογικές επιστήμες, καθώς έχουν αναπτυχθεί κατά τις τελευταίες δεκαετίες. Οι αρχές λειτουργίας τους προέρχονται από το νευρικό σύστημα των ζωντανών οργανισμών και έχουν προσαρμοστεί κατάλληλα ώστε να μπορούν να λύσουν κάθε είδους πρόβλημα μέσω του ηλεκτρονικού υπολογιστή. Η φιλοσοφία τους είναι διαφορετική από τον τρόπο με τον οποίο δουλεύουν οι κλασικοί υπολογιστές. Η λειτουργία τους προσπαθεί να συνδιάσει τον τρόπο σκέψης του ανθρώπινου εγκεφάλου με τον αφηρημένο μαθηματικό τρόπο σκέψης. Γι'αυτό θα συναντήσουμε πολύ συχνά έννοιες όπως π.χ. ένα δίκτυο μαθαίνει, εκπαιδεύεται, θυμάται, που μέχρι τώρα τα αποδίδαμε μόνο στην ανθρώπινη σκέψη. Το εύρος των γνώσεων για την ανάπτυξη των νευρωνικών δικτύων είναι πολύ μεγάλο. Απαιτούνται ταυτόχρονα γνώσεις σε θέματα από πολλές περιοχές (Ιατρική, Χημεία, Μαθηματικά, επιστήμη των Υπολογιστών, Ηλεκτρολογία), ενώ το ίδιο ισχύει και για τις τεχνικές και τις μεθόδους που χρησιμοποιούνται. Πολύ λίγες επιστήμες σήμερα συνδιάζουν με τόσο άμεσο τρόπο γνώσεις που προέρχονται από τόσο διαφορετικές περιοχές. Η έμπνευση για τα νευρωνικά δίκτυα, όπως αναφέραμε παραπάνω, ξεκινά από την βιολογία.

1.2 Ο Νευρώνας

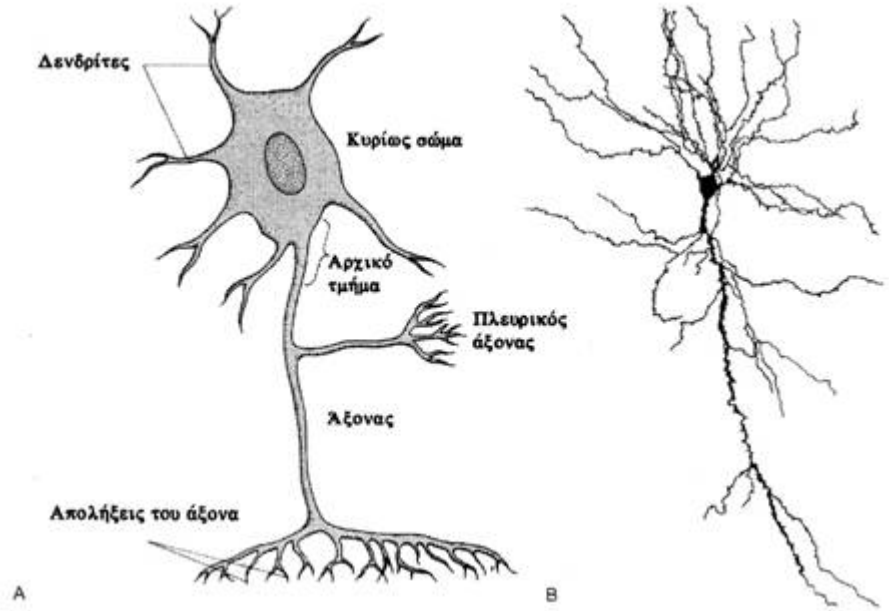
Οι ζώντες οργανισμοί, από τους πιο απλούς μέχρι τον άνθρωπο, έχουν ένα νευρικό σύστημα το οποίο είναι υπεύθυνο για ένα πλήθος από διεργασίες, όπως είναι η επαφή με τον εξωτερικό κόσμο, η μάθηση, η μνήμη κτλ. Το νευρικό σύστημα των οργανισμών αποτελείται από πολλά νευρωνικά δίκτυα τα οποία είναι εξειδικευμένα στις διεργασίες αυτές. Η κεντρική μονάδα του νευρικού συστήματος είναι ο εγκέφαλος, ο οποίος επίσης αποτελείται από νευρωνικά δίκτυα. Κάθε νευρωνικό δίκτυο αποτελείται από ένα μεγάλο αριθμό μονάδων, που λέγονται νευρώνες (neurons). Ο νευρώνας, η μικρότερη ανεξάρτητη μονάδα του δικτύου, δέχεται, επεξεργάζεται και στέλνει πληροφορίες μέσω ηλεκτρικών σημάτων. Αποτελείται απο τα εξής βασικά τμήματα:

- τους δενδρίτες: Τα αισθητήρια όργανα του νευρώνα
- το σώμα: Ο εγκέφαλος του νευρώνα και το σημείο όπου συγκεντρώνονται όλες οι πληροφορίες
- τον άξονα: Η πύλη εξόδου του νευρώνα
- τις συνάψεις: Τα σημεία ένωσης μεταξύ του άξονα και των άλλων νευρώνων

1.2.1 Λειτουργία του Νευρώνα

Σε ένα βιολογικό νευρώνα η πληροφορία μεταφέρετε μέσω μιας αλληλουχίας παλμών. Ένας νευρώνας μπορεί να έχει μέχρι και 2000 δενδρίτες, οι οποίοι μεταφέρουν τα σήματα που παίρνουν από τους άλλους νευρώνες με τους οποίους είναι συνδεδεμένοι, στο σώμα όπου και γίνεται η επεξεργασία των πληροφοριών. Οι πληροφορίες που λαμβάνονται αθροίζονται και αν το συνολικό άθροισμα είναι πάνω από ένα συγκεκριμένο κατώφλι τότε λέμε ότι ο νευρώνας πυροβολεί, δηλαδή παράγει ηλεκτρικούς παλμούς, αλλιώς μένει αδρανής. Εαν αποφασίσει να πυροβολήσει τότε οι ηλεκτρικοί παλμοί μεταφέρονται μέσω του άξονα στις συνάψεις. Ο άξονας μοιάζει με μια νηματοειδείς επέκταση και το μήκος του μπορεί να ξεπράσει ακόμα και το 1 μέτρο σε μερικούς νευρώνες. Σε έναν άξονα υπάρχουν διάφορες διακλαδώσεις διότι οι παλμοί μεταφέρονται σε πολλούς άλλους νευρώνες οι οποίοι μπορεί να βρίσκονται σε κοντινή ή μακρινή απόσταση. Η μετάδοση των πληροφοριών απο ένα νευρώνα σε έναν άλλο γίνεται μέσω των συνάψεων. Οι συνάψεις είναι κύστες με ηλεκτροχημικό υλικό, ιόντα, κυρίως Νατρίου

και Καλίου (Na^+, K^+). Όταν ο νευρώνας αποφασίσει να πυροβολήσει τότε οι συνάψεις μεταδίδουν την ηλεκτρική δραστηριότητα στους δενδρίτες-παραλήπτες. Το πλάτος της σύναψης, η απόσταση της από τον δενδρίτη αλλά και η πυκνότητα του ηλεκτροχημικού υλικού επηρεάζουν το ποσοστό της ηλεκτρικής δραστηριότητας που θα μεταδοθεί. Το ποσοστό αυτό λέγεται συναπτικό βάρος και καθορίζει το πόσο έντονα θα συμμετάσχει το συγκεκριμένο φορτίο ηλεκτρικής ενέργειας στο συνολικό άρθροισμα.



Σχήμα 1.1: Η δομή ενός νευρώνα

1.3 Τεχνητά νευρωνικά δίκτυα

Τα τεχνητά νευρωνικά δίκτυα (ΤΝΔ) ή εν συντομία νευρωνικά δίκτυα είναι ένα μαθηματικό μοντέλο για την επεξεργασία της πληροφορίας με τρόπο παρόμοιο αυτού του εγκεφάλου. Η αρχιτεκτονική των νευρωνικών δικτύων είναι πολύ διαφορετική από αυτήν των παραδοσιακών υπολογιστών που περιέχουν έναν επεξεργαστή. Οι γνωστοί υπολογιστές δουλεύουν σειριακά, σύμφωνα με τις πρώτες ιδέες του von Neumann, και έχουν την ικανότητα να επιτελούν μερικές εκατοντάδες εντολές, όπως αριθμητικές πράξεις κτλ. Από τη φύση τους τα νευρωνικά δίκτυα δεν λειτουργούν σειριακά, αλλά με τρόπο που μοιάζει πιο πολύ σε παράλληλο τρόπο λειτουργίας, διότι μία εργασία μοιράζεται σε όλους τους επιμέρους νευρώνες. Έτσι μπορούμε να πούμε ότι τα νευρωνικά δίκτυα είναι συστήματα «παράλληλων κατανεμημένων διεργασιών». Αυτός ο τρόπος λειτουργίας μας παρέχει μεγάλες ταχύτητες, διότι είναι σαν να έχουμε ταυτόχρονα

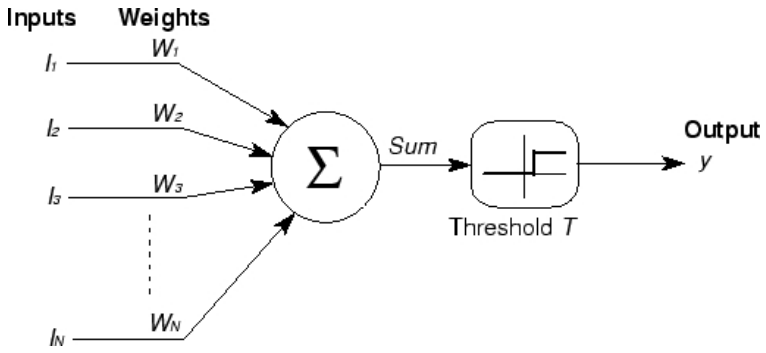
πολλούς επεξεργαστές στη διάθεσή μας. Στην πραγματικότητα όμως η αρχιτεκτονική των νευρωνικών δικτύων διαφέρει από αυτή των παραλλήλων επεξεργαστών, για το λόγο ότι οι νευρώνες έχουν ένα πολύ μεγάλο αριθμό διασυνδέσεων μεταξύ τους, ο οποίος συνολικά είναι πολύ μεγαλύτερος από τον αριθμό των νευρώνων. Στους παράλληλους υπολογιστές, οι επεξεργαστές είναι συνήθως περισσότεροι από τις διασυνδέσεις μεταξύ τους. Από την άλλη στα νευρωνικά δίκτυα ο κάθε νευρώνας επιτελεί πολύ απλές λειτουργίες, π.χ. αθροίζουν τα σήματα εισόδου και τροποποιούν τα βάρη των διασυνδέσεων, σε αντίθεση με τους επεξεργαστές οι οποίοι έχουν τη δυνατότητα να εκτελούν πολύπλοκες αριθμητικές πράξεις. Μια άλλη σημαντική διαφορά είναι ότι οι νευρώνες λειτουργούν ανεξάρτητα ο ένας από τον άλλο και δεν χρειάζονται συγχρονισμό. Αυτό είναι που δίνει στα νευρωνικά δίκτυα το κύριο χαρακτηριστικό τους, την ανοχή σε σφάλματα. Όταν κάποιο κομμάτι του δικτύου καταστραφεί το υπόλοιπο δίκτυο συνεχιζει να λειτουργεί, έστω και με μικρό σφάλμα. Κοιτάζοντας από άλλη οπτική γωνία θα μπορούσαμε να πούμε ότι όταν τα δεδομένα ενός προβλήματος είναι ελλειπή το δίκτυο θα μπορέσει να δώσει ικανοποιητική απάντηση, πάλι όμως με ένα μικρό σφάλμα. Η σκέψη για τη δημιουργία τεχνητών νευρωνικών δικτύων άρχισε στα τέλη του 1800 σαν μια προσπάθεια να περιγραφεί ο τρόπος με τον οποίο συμπεριφέρεται ο εγκέφαλος. Η αρχή έγινε στα μέσα της δεκαετίας του 1940 από τον Alan Turing ο οποίος αναφέρθηκε σε μηχανές, οι οποίες δεν ήταν σχεδιασμένες για να εκτελούν συγκεκριμένες πράξεις, τις οποίες ονόμασε ανοργάνωτες μηχανές. Ο Turing εμπνεύσμένος από το ανθρώπινο νευρικό σύστημα συνέλαβε την ιδέα για τη δημιουργία μηχανών δομημένων με τυχαίο τρόπο οι οποίες όμως, έχουν τη δυνατότητα να εκπαιδευτούν (οργανωθούν) με την βοήθεια μιας εξωτερικής διέγερσης, με την έννοια του δασκάλου. Την ίδια περίπου περίοδο (1943) δύο επιστήμονες, ο Warren McCulloch και ο Walter Pitts δημιουργούν το πρώτο μοντέλο τεχνητού νευρώνα.

1.3.1 Το μοντέλο McCulloch-Pitts

Το μοντέλο των McCulloch-Pitts δεν είναι ένα πλήρες μοντέλο νευρικού κυττάρου αλλά προσομοιάζει ένα νευρώνα με πολλές εισόδους, I_1, I_2, ... , I_n και μια έξοδο y η οποία περιγράφεται από ένα δυαδικό αριθμό και μπορεί να πάρει δυο καταστάσεις y=0 και y=1. Η μορφή του φαίνεται στο παρακάτω σχήμα και οι εξισώσεις που περιγράφουν το μοντέλο είναι οι εξής:

$$sum = \sum_{i=1}^{n} I_i W_i \,, \tag{1.1}$$

$$y = f(sum - \theta) \tag{1.2}$$



Σχήμα 1.2: Το μοντέλο των McCulloch-Pitts

Τα συναπτικά βάρη W_1, W_2, ... , W_n είναι πραγματικοί αριθμοί, θετικοί ή αρνητικοί ανάλογα με τον τύπο των συνάψεων. Η κάθε σύναψη μπορέι να είναι είτε ενισχυτική (δηλαδή να ωθεί το νευρώνα να αποκριθεί στη διέγερση) είτε ανασταλτική (δηλαδή να αποτρέπει το νευρώνα να παράγει μια απόκριση). Στη συνέχεια η μονάδα άθροισης του νευρώνα αθροίζει τα σήματα εισόδου πολλαπλασιαζόμενα με τα αντίστοιχα βάρη των συνάψεων. Αν το άθροισμα αυτό είναι μικρότερο από το κατώφλι (Threshold) Θ τότε ο νευρώνας μένει αδρανής, αλλιώς πυροβολεί. Όπως εύκολα διαπιστώνουμε από τον τύπο (1.2) το κατώφλι Θ μπορεί να θεωρηθεί ως ένα επιπλέον συναπτικό βάρος το οποίο ονομάζεται πόλωση και συνδέεται με μια σταθερή είσοδο W_0 η οποία έχει τιμή ίση με -1. Έτσι οι (1.1) και (1.2) μετασχηματίζονται αντίστοιχα σε

$$sum = \sum_{i=0}^{n} I_i W_i \,, \tag{1.3}$$

$$y = f(sum) \tag{1.4}$$

Η συνάρτηση f λέγεται συνάρτηση ενεργοποίησης και συνήθως είναι η βηματική συνάρτηση 0/1

$$f(u) = \begin{cases} 0, & \text{αν } u \leq 0 \\ 1, & \text{αν } u > 0 \end{cases} \tag{1.5}$$

Υπάρχουν διάφορες παραλαγές του μοντέλου των McCulloch-Pitts. Η κυριότερη διαφορά εντοπίζεται στις συναρτήσεις ενεργοποίησης που χρησιμοποιούνται. Αυτές μπορεί να είναι:

Βηματική-1/1

$$f(u) = \begin{cases} -1, & \text{αν } u \leq 0 \\ 1, & \text{αν } u > 0 \end{cases} \tag{1.6}$$

Σιγμοειδής

$$f(u) = 1/(1 + e^{-u}) \tag{1.7}$$

Υπερβολική εφαπτομένη

$$f(u) = tanh(u) \tag{1.8}$$

Συνάρτηση κατωφλίου

$$f(u) = \begin{cases} 0, & \text{αν } u \leq 0 \\ u, & \text{αν } 0 < u < 1 \\ 1, & \text{αν } u \geq 1 \end{cases} \tag{1.9}$$

Γραμμική

$$f(u) = u \tag{1.10}$$

1.4 Εκπαίδευση στα ΤΝΔ

Η εκπαίδευση στα νευρωνικά δίκτυα είναι μια πολύ σημαντική έννοια. Ο κύριος σκοπός ενός νευρωνικού δικτύου είναι να επιλύει συγκεκριμένα προβλήματα. Για να είναι αυτό εφικτό θα πρέπει πρώτα να το εκπαιδεύσουμε. Με τον όρο εκπαίδευση εννοούμε την κατάλληλη χρήση πληροφοριών για την βελτίωση της απόκρισης του συστήματος. Οι πληροφορίες αυτές όπως και στα βιολογικά δίκτυα αντιστοιχούν σε εισόδους οι οποίες δίνουν συγκεκριμένες εξόδους. Οταν λέμε εισόδους/εξόδους εννοούμε ότι παρουσιάζονται στο δίκτυο κάποια σήματα με τρόπο ώστε να μπορεί να τα καταλάβει λ.χ θα μπορούσε να είναι κάποιος δυαδικός αριθμός αποτελούμενος από 0 και 1. Οι αριθμοί που δίνονται στην είσοδο του δικτύου αποτελούν κάποιο πρότυπο. Σε κάθε πρότυπο αντιστοιχεί και μία σωστή απάντηση, η οποία είναι το σήμα που πρέπει να

πάρουμε στην έξοδο ή αλλιώς στόχος. Η εκπαίδευση γίνεται με το να παρουσιάσουμε μια ομάδα από τέτοια πρότυπα στο δίκτυο. Το δίκτυο χρησιμοποιεί τα πρότυπα και εφόσον γνωρίζει την επιθυμητή έξοδο τροποποιεί τις τιμές των βαρών των συνδέσεων των νευρώνων ώστε να μπορεί να αντιστοιχίσει τα πρότυπα με τις εξόδους τους. Ενώ αρχικά ξεκινάει με τιμές στα βάρη που είναι τυχαίες, κατά την διάρκεια της εκπαίδευσης μεταβάλλει τις τιμές αυτές, μέχρι να εκπαιδευθεί πλήρως. Αφού εκπαιδευθεί θα μπορεί να βρίσκει τις εξόδους άλλων προτύπων τα όποια δεν έχει δει προηγουμένως. Παρακάτω θα αναλύσουμε τη διαδικασία εκπαίδευσης ενός νευρώνα. Τα βιολογικά συστήματα όπως ο εγκέφαλος έχουν την δυνατότητα να μαθαίνουν λόγο της τρομερής τους πολυπλοκότητας. Σε έναν ανθρώπινο εγκέφαλο για παράδειγμα υπάρχουν περίπου 100 δισεκατομμύρια νευρώνες, ο καθένας από τους οποίους έχει κατά μέσο όρο 1000 συνάψεις. Ένα τέτοιο σύστημα είναι αρκετά δύσκολο να κατασκευαστεί με τεχνητά μέσα ώστε να είναι το ίδιο αποδοτικό και γρήγορο, διότι θα χρειάζόταν πολλά χρήματα και τεχνολογία που ακόμα δεν έχει ανακαλυφθεί. Παρόλα αυτά σήμερα υπάρχουν πολλές εφαρμογές που βασίζονται πάνω σε τέτοια συστήματα μικρότερης κλίμακας. Μεταξύ αυτών περιλαμβάνονται:

- Προβλήματα αναγνώρισης (π.χ. φωνής, εικόνας)
- Αποθήκευση πληροφοριών - μνήμη
- Προβλήματα ελέγχου με ελλειπή δεδομένα
- Η εύρεση λύσης σε προβλήματα με βάση διάφορα κριτήρια κόστους (π.χ. δρομολόγηση, αναζήτηση)
- Κωδικοποίηση - Συμπίεση δεδομένων
- Ανάγνωση κειμένου
- Ταξινόμηση σε κατηγορίες
- Υπολογισμός συναρτήσεων
- Αυτόματος έλεγχος

1.5 Κατηγορίες Νευρωνικών Δικτύων

Από την εποχή που δημιουργήθηκε ο πρώτος τεχνητός νευρώνας μέχρι και σήμερα πολλοί επιστήμονες έχουν κατασκευάσει πολλά μοντέλα νευρωνικών δικτύων που αποτελούνται από περισσότερους του ενός νευρώνες και έχουν διαφορετικούς τρόπους συμπεριφοράς. Τα νευρωνικά δίκτυα ταξινομούνται σε 2 κύριες κατηγορίες ανάλογα με τον τρόπο εκπαίδευσης των βαρών τους. Αν κατά την εκπαίδευση των βαρών χρησιμοποιούμε κάποιον εξωτερικό "δάσκαλο" που μας δίνει για κάθε πρότυπο εκπαίδευσης το επιθυμητό αποτέλεσμα τότε λέμε ότι η εκπαίδευση είναι με επίβλεψη. Σε αντίθετη περίπτωση όπου μας δίνονται μόνο τα πρότυπα χωρίς τις εξόδους τους, τότε λέμε ότι η εκπαίδευση είναι χωρίς επίβλεψη. Κατα την εκπαιδευση με επίβλεψη σκοπός μας είναι να ελαχιστοποιήσουμε τη διαφορά μεταξύ της δοσμένης εξόδου και της τρέχουσας τιμής της εξόδου μεταβάλλοντας τα συναπτικά βάρη. Στην εκπαίδευση χωρις επίβλεψη απλά προβάλλουμε τα πρότυπα στο δίκτυο και αυτό "αυτοοργανώνεται" συνήθως διορθώνοντας τα σφάλματα με ένα μηχανισμό ανάδρασης. Επίσης μπορούμε να χωρίσουμε τα νευρωνικά δίκτυα σε αυτά με εκπαιδευόμενα βάρη και σε αυτά με σταθερά βάρη. Όπως φαίνεται και από το όνομα τους τα δίκτυα με εκπαιδευόμενα βάρη αλλάζουν τα συναπτικά βάρη τους ώστε να εκπαιδευτούν, ενώ σε αυτά με σταθερά βάρη, τα βάρη υπολογίζονται συνήθως από μια συνάρτηση και παραμένουν σταθερά. Μια τρίτη κατηγοριοποίηση είναι ανάλογα με το αν οι νευρώνες είναι διατεταγμένοι σε στρώματα. Ανάλογα με την τοπολογία διασύνδεσης τα χωρίζουμε σε, α) δίκτυα πρόσθιας τροφοδότησης (Feedforward networks) στα οποία η πληροφορία μεταδίδεται ιεραρχικά από στρώμα σε στρώμα, β) δίκτυα με ανατροφοδότηση στα οποία σε αντίθεση με τα δίκτυα πρόσθιας τροφοδότησης η πληροφορία μεταδίδεται και σε προηγούμενα στρώματα

1.5.1 Perceptron

Το δίκτυο perceptron είναι το πιο απλό δίκτυο που μπορούμε να κατασκευάσουμε και αποτελείται από ένα μόνο νευρώνα. Το μοντέλο του νευρώνα είναι ίδιο με αυτό των McCulloch-Pitts. Το δίκτυο αυτό μπορεί να ξεχωρίσει ορθά δύο πρότυπα ύστερα από κατάλληλη εκπαίδευση. Η εκπαίδευση γίνεται με επίβλεψη, δηλαδή υπάρχει κάποιος που μας δίνει για κάθε πρότυπο τη σωστή τιμή της εξόδου. Ο κανόνας εκπαίδευσης είναι σχετικά απλός. Εισάγουμε ένα ένα τα γνωστά πρότυπα στο δίκτυο και αυτό προσαρμόζει τα βάρη των συνάψεων κατάλληλα ώστε η έξοδος του να ταιριάζει με αυτήν

που του δίνουμε. Μετά από κάποιο πλήθος εποχών, αποδεικνύεται, ότι εαν το πρόβλημα είναι γραμμικά διαχωρίσημο, το δίκτυο θα συγκλήνει.

1.5.2 Perceptron πολλών στρωμάτων

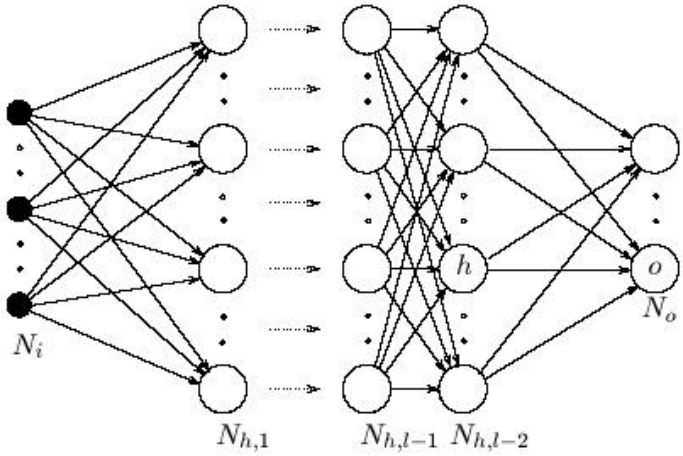
Τα δίκτυα πολλών στρωμάτων MLP (MultiLayer Perceptron), είναι δίκτυα που χρησιμοποιούν πολλούς νευρώνες και πολλά στρώματα ώστε να πετύχουν μεγαλύτερο εύρος διαχωρισμού. Το βασικό θεώρημα για αυτά τα δίκτυα λέει ότι δίκτυα τέτοιας μορφής μπορούν να προσεγγίσουνν οποιαδήποτε ομαλή συνάρτηση όσο κοντά επιθυμούμε. Για το λόγο αυτό καλούνται "Καθολικοί προσεγγιστές" (Universal Approximators) [Διαμαντάρας 2007]. Αποτελούνται από 3 τύπους στρωμάτων. Το στρώμα εισόδου, το στρώμα εξόδου, και το κρυφό στρώμα. Στο κρυφό στρώμα μπορούμε να έχουμε παραπάνω του ενός στρώματα, αν και στις περισσότερες περιπτώσεις το 1 ειναι αρκετό. Όλοι οι νευρώνες του ενός επιπέδου συνδέονται με τους νευρώνες του επομένου επιπέδου. Στα δίκτυα πολλών στρωμάτων ο κυριότερος και πιο γνωστός αλγόριθμος εκπαίδευσης είναι ο αλγόριθμος Back-Propagation. Ονομάστηκε έτσι διότι κατά τη διαδικασία της εκπαίδευσης και αφού υπολογιστούν οι έξοδοι, ακολουθούμε την αντίστροφη φορά διορθώνοντας τα βάρη σε κάθε στρώμα. Έχοντας ως δεδομένα ότι L=πλήθος στρωμάτων, P ζεύγη διανυσμάτων εισόδων, και τα βάρη $w_{ij}(l)$, η μορφή του αλγορίθμου Back-Propagation για την εκπαιδευση είναι η παρακάτω:

Επανέλαβε {
    Για κάθε πρότυπο p=1, ..., P {
        Υπολόγισε τις εξόδους a από το στρώμα 1 μέχρι και το στρώμα L
        Υπολόγισε τα σφάλματα δ_i από το στρώμα L έως και το στρώμα 1
        Ενημέρωσε τα βάρη όλων των στρωμάτων
    }
} (Όσο το σφάλμα είναι μικρότερο από το όριο ε που έχουμε ορίσει)

Κατά την ανάκληση σε ένα δίκτυο MLP η διαδικασία είναι σχεδόν παρόμοια με την εκπαίδευση χωρίς φυσικά τη διόρθωση των βαρών.

Για κάθε στρώμα l=1, ..., L {
    Υπολόγισε τις εξόδους a όλων των νευρώνων

}



Σχήμα 1.3: Το δίκτυο πολλών στρωμάτων (MLP)

1.5.3 SOM

Το δίκτυο SOM (Self Organizing topographic Map) σχεδιάστηκε από τον Teuvo Kohonen και ανήκει στα δίκτυα χωρίς επίβλεψη. Το μοντέλο SOM χρησιμοποιείται κυρίως σε περιπτώσεις που δεν υπάρχει εξωτερικός "δάσκαλος" και είναι ιδιαίτερα χρήσιμο όταν δε γνωρίζουμε εξαρχής τις κατηγορίες των προτύπων. Στα δίκτυα αυτά το ίδιο το δίκτυο αποφασίζει ποιες είναι οι κατηγορίες και αντιστοιχίζει κατάλληλα τα δείγματα εισόδου. Η δομή αυτών των δικτύων περιγράφεται ως μια δισδιάστατη ή μονοδιάστατη συστοιχία νευρώνων, όπου κύριο ρόλο παίζει η θέση του νευρώνα καθώς σε αυτά τα δίκτυα προσομοιάζετε η τοπογραφική οργάνωση του εγκεφάλου. Η εκπαίδευση γίνεται με διαφορετικό τρόπο από αυτά που έχουμε αναφέρει νωρίτερα καθώς οι νευρώνες "ανταγωνίζονται" για το ποιός θα ταιριάξει καλύτερα στο πρότυπο εισόδου. Με αυτόν τον τρόπο δημιουργούνται διαφορετικές περιοχές στο πλέγμα, που αντιστοιχούν σε διαφορετικές κλάσεις εισόδων.

1.5.4 RBF

Τα δίκτυα RBF (Radial Basis Functions) είναι παρόμοια με τα MLP. Έχουν δύο στρώματα και ισχύει και γι'αυτά το θεώρημα του καθολικού προσεγγιστή. Η διαφορά τους με τα MLP είναι ότι τα πρώτα δε χρησιμοποιούν τη σιγμοειδή συνάρτηση σα συνάρτηση

ενεργοποίησης, αλλά κάποια συνάρτηση ακτινικής βάσης. Τέτοιες συναρτήσεις μπορεί να είναι οι:

Συνάρτηση Gauss,

$$f(x) = e^{-\frac{\|x-c\|^2}{\sigma^2}} \tag{1.11}$$

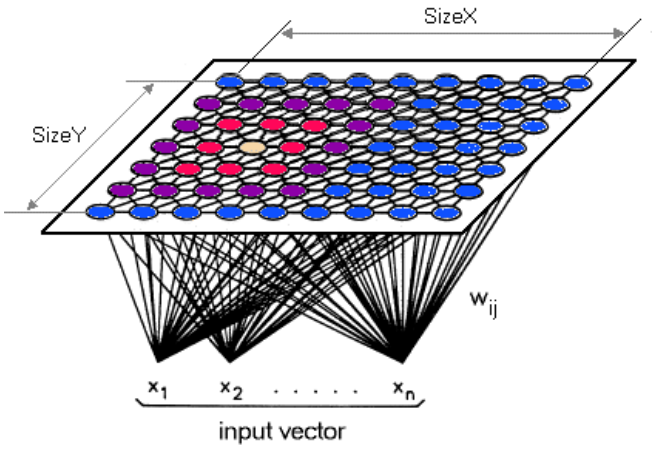
Πολυτετραγωνική συνάρτηση,

$$f(x) = (\| x - c \|^2 + \sigma^2)^{1/2} \tag{1.12}$$

Συνάρτηση Cauchy,

$$f(x) = \frac{(\| x - c \|^2 + \sigma^2)^{-1}}{\sigma} \tag{1.13}$$

όπου c είναι ένα διάνυσμα που αποκαλείται κέντρο, x η απόσταση από το c και σ το εύρος συνάρτησης.



Σχήμα 1.4: Το μοντέλο SOM του Kohonen

Κεφάλαιο 2

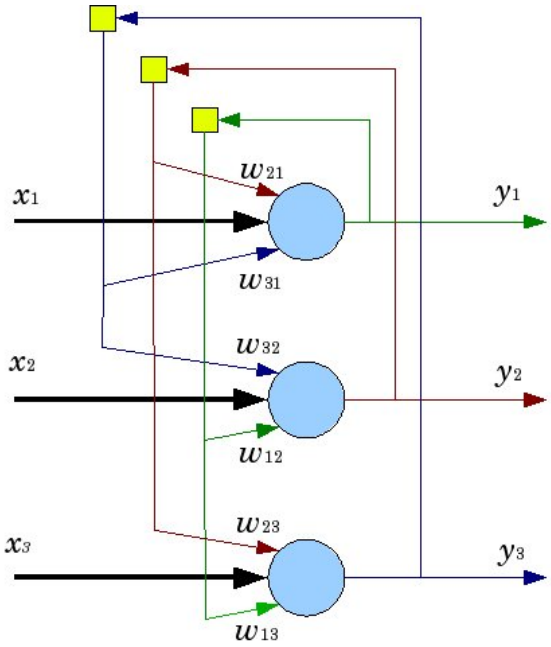
Το δίκτυο Hopfield

Στο πρόγραμμα μας θα χρησιμοποιήσουμε το δίκτυο Hopfield το οποίο ανήκει στην κατηγορία των δικτύων με σταθερά βάρη. Το δίκτυο αυτό επινοήθηκε από τον John Hopfield στις αρχές της δεκαετίας του 1980. Χρησιμοποιείται κυρίως για την αναγνώριση προτύπων με ελλειπή δεδομένα και σε προβλήματα βελτιστοποίησης. Σε αντίθεση με τα δίκτυα πολλών στρωμάτων, το Hopfield έχει μόνο ένα στρώμα το οποίο αποτελείται από πολλούς νευρώνες. Εδώ δεν έχουμε ξεχωριστό στρώμα εισόδου και εξόδου. Κάθε νευρώνας δέχεται σήματα από το περιβάλλον και έχει εξόδους προς αυτό. Η εσωτερικές συνδέσεις του δικτύου, είναι και προς τα εμπρός αλλά και προς τα πίσω, συνδέοντας τις εξόδους του με τις εισόδους του. Γι'αυτόν το λόγο ονομάζεται αναδρομικό δίκτυο. Η διαδικασία εκμάθησης συνδέεται άμεσα με την παρουσία αυτών των αναδρομικών βρόγχων.

2.1 Εκπαίδευση

Οι νευρώνες στο δίκτυο Hopfield είναι σε διάταξη μονοδιάστατου πλέγματος, όπου καθένας δέχεται στην είσοδο του τις εξόδους όλων των άλλων νευρώνων και σε μερικές περιπτώσεις ακόμα και του εαυτού του. Αυτό φαίνεται ξεκάθαρα στο σχήμα 2.1. Η έξοδος κάθε νευρώνα μπορεί να έχει δύο διακριτές τιμές, 1 ή -1 οι οποίες δίνονται από τον τύπο:

$$y_i = f(\sum_{j=0}^{n} w_{ij} y_j) \tag{2.1}$$



Σχήμα 2.1: Το δίκτυο Hopfield

Τα συναπτικά βάρη των νευρώνων αποθηκεύονται σε ένα δισδιάστατο πίνακα μεγέθους n*n (n το μέγεθος των νευρώνων) αντιπροσωπεύοντας τα βάρη. Με βάση αυτές τις τιμές είναι δυνατή η ανάκληση ενός προτύπου. Ο όρος εκπαίδευση δεν μπορούμε να πούμε ότι ταιριάζει απόλυτα στο δίκτυο Hopfield. Η διαδικασία της εκπαίδευσης δεν ακολουθεί την κλασική διαδικασία εκπαίδευσης των άλλων δικτύων, δεν υπάρχει δηλαδή κάποιος αναδρομικός κανόνας για την εύρεση των βαρών, καθώς αυτά υπολογίζονται κατευθείαν. Ο τρόπος υπολογισμού είναι αρκετά εύκολος. Έστω ότι έχουμε ένα πρότυπο X = (x_1, x_2, ..., x_n) όπου τα x_i παίρνουν τιμές -1 ή 1. Αν σκεφτούμε το πρότυπο σαν ένα μονοδιάστατο πίνακα με μορφή

x_1	x_2	...	x_n

τότε παίρνοντας τον ανάστροφο του έχουμε:

x_1
x_2
...
x_n

Στη σύνέχεια αρκεί να πολλαπλασιάσουμε τους δύο πίνακες μεταξύ τους των οποίων το αποτέλεσμα θα μας δώσει τον δισδιάστατο πίνακα με τα βάρη. Επειδή όμως είπαμε δεν υπάρχει άμεση σύνδεση ανάμεσα σε ένα νευρώνα και στον εαυτό του (εκτός ορισμένων περιπτώσεων) βάζουμε στα στοιχεία της πάνω αριστερά διαγωνίου X(i, i) την

τιμή 0. Αν θέλουμε να αποθηκέυσουμε περισσότερα πρότυπα του ενός αυτό γίνεται με άθροιση των πινάκων των βαρών. Στη γενική περίπτωση ενός δικτύου hopfield οι συνδέσεις μεταξύ των νευρώνων είναι αμφίδρομες. Αυτό σημαίνει ότι υπάρχει και w_{ij} και w_{ji} τα οποία όμως θα πρέπει να είναι ίσα ώστε να μπορέσει το δίκτυο να συγκλήνει σύμφωνα με το θεώρημα των Cohen και Grossberg. Το θεώρημα αυτό αποδεικνύει ότι τα αναδρομικά δίκτυα συγκλήνουν εαν η μήτρα των βαρών είναι συμμετρική με 0 στην κύρια διαγώνιο δηλαδή $w_{ii} = 0$ για όλα τα i, και $w_{ij} = w_{ji}$ για κάθε $i \neq j$.

Πίνακας 2.1: Πίνακας συναπτικών βαρών

	1	2	$\cdots$	n
1	0	$w_{1,2}$	$\cdots$	$w_{1,n}$
2	$w_{2,1}$	0	$\cdots$	$w_{2,n}$
3	$w_{3,1}$	$w_{3,2}$	$\cdots$	$w_{3,n}$
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
n	$w_{n,1}$	$w_{n,2}$	$\cdots$	0

2.2 Ανάκληση

Έχοντας υπολογίσει τα βάρη για κάθε σύνδεση των νευρώνων μπορούμε να βρούμε τις εξόδους y για τις οποίες ικανοποιείται η σχέση (2.1), για όλα τα i = 1, 2, 3, $\cdots$, n. Παρατηρούμε ότι ο άγνωστος y εμφανίζεται και στις δύο πλευρές το οποίο σημαίνει ότι η σχέση είναι αναδρομική. Έστω ότι δίνουμε κάποιες αρχικές τιμές στα y, τότε χρησιμοποιώντας τη συνάρτηση (2.1), θα πάρουμε κάποια νέα y. Μπορούμε να συνεχίσουμε τη διαδικασία της αναδρομής με δύο τρόπους, το σύγχρονο και τον ασύγχρονο.

- Κατά το σύγχρονο τρόπο υπολογίζονται πρώτα όλες οι νέες τιμές των y και μετά ανατροφοδοτούνται στο δεξί σκέλος για να πάρουμε τις επόμενες τιμές.
- Κατά τον ασύγχρονο τρόπο κάθε φορά που υπολογίζουμε μια νέα τιμή του y τη χρησιμοποιούμε για τον επόμενο νευρώνα.

Η διαδικασία αυτή του υπολογισμού των y συνεχίζεται για όλους τους νευρώνες. Όταν τελειώσουν όλοι οι νευρώνες τότε ακολουθούμε την ίδια διαδικασία μέχρι να μην υπάρχει καμιά αλλαγή μεταξύ των παλαιών y και των καινούργιων. Την κατάσταση αυτή την ονομάζουμε κατάσταση ισορροπίας. Γενικά στην κατάσταση ισορροπίας ικανοποιείται

η σχέση:

$$y_i = f(\sum_{j=0}^{n} w_{ij} y_j) \tag{2.2}$$

2.3 Συνάρτηση Ενέργειας

Το δίκτυο Hopfield χαρακτιρίζεται ως δυναμικό σύστημα (dynamical system). Δυναμικά συστήματα αποκαλούνται αυτά για τα οποία μπορούμε να ορίσουμε μια τιμή για κάθε χρονική κατάσταση που βρίσκονται. Στο δίκτυο Hopfield είναι η συνάρτηση ενέργειας E (συνάρτηση Lyapunov) η οποία μας δίνει αυτήν την τιμή.

$$E = -\frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} w_{ij} y_i y_j - \sum_{i=1}^{n} I_i y_i \tag{2.3}$$

όπου n ο αριθμός των νευρώνων, w_{ij} το συναπτικό βάρος μεταξύ του νευρώνα i και j, y_i η έξοδος του νευρώνα i και I_i η εξωτερική διέγερση. Η συνάρτηση E σε κάθε κατάσταση του δικτύου μειώνεται μέχρι που συγκλίνει σε ένα σημείο όπου δε μεταβάλλεται άλλο. Σε αυτό το σημείο λέμε ότι έχουμε φτάσει σε κατάσταση ισορροπίας.

2.4 Συσχετιστική μνήμη

Σε αντίθεση με τα δίκτυα πολλών στρωμάτων ένα δίκτυο hopfield δεν μεταβάλλει τις εξόδους του ώστε να δείξει σε ποιά κάτηγορία ανήκει το πρότυπο που εισάγαμε αλλά, παίρνει σαν είσοδο το “κατεστραμένο πρότυπο” και προσπαθεί να το ανακατασκευάσει ώστε να ταιριάξει ή να μοιάξει με κάποιο από τα υπάρχοντα πρότυπα. Με άλλα λόγια το δίκτυο hopfield μπορεί να λειτουργήσει σαν συσχετιστική μνήμη. Η ιδιότητα αυτή είναι πολύ χρήσιμη κυρίως σε προβλήματα του καθημερινού κόσμου όπου υπάρχει πολύς θόρυβος με αποτέλεσμα τα δεδομένα να μην είναι συγκεκριμένα. Για παράδειγμα ο ανθρώπινος εγκέφαλος μόλις ακούει το όνομα ενός ανθρώπου φέρνει στη μνήμη του περιστατικά και πληροφορίες που σχετίζονται με αυτόν. Στο επίπεδο του υπολογιστή η λειτουργία είναι παρόμοια αν και μικρότερης κλίμακας. Η πιο συνηθησμένη εφαρμογή είναι η αναγνώριση προτύπων τα οποία είναι ελαφρώς κατεστραμένα. Σε κάθε κατεστραμένο πρότυπο υπάρχει ένα ποσοστό της αρχικής πληροφορίας από την οποία το δίκτυο μπορεί να συσχετίσει αυτήν την πληροφορία με το πρότυπο που βρίσκεται στη μνήμη και να το ανακαλέσει. Φυσικά εδώ υπάρχει το ερώτημα εως τι ποσοστό

του αρχικού δεδομένου μπορούμε να έχουμε θόρυβο ώστε το δίκτυο να βρει τη σωστή απάντηση. Δεν υπάρχει κάποιο συγκεκριμένο όριο αλλά εμπειρικά εκτιμάται ότι το ποσοστό αυτό είναι γύρω στο 10-15 και θα πρέπει να χρησιμοποιείται με προσοχή.

2.5 Μέθοδος βελτιστοποίησης

Τα προβλήματα βελτιστοποίησης είναι πολύ συχνά στην επιστήμη των υπολογιστών. Το πρόβλημα του περιοδεύοντος πωλητή, το πρόβλημα του ζυγισμένου ταιριάσματος, αλλά και η επίλυση ενός Sudoku puzzle μπορούν να χαρακτηριστούν ως προβλήματα βελτιστοποίησης, εφόσον ο σκοπός τους είναι να βρούμε τη βέλτιστη λύση από ένα πλήθος πεπερασμένων λύσεων. Τέτοιου είδους προβλήματα μπορούν να επιλυθούν με τη βοήθεια του δικτύου hopfield, αρκεί να μετασχηματιστούν κατάλληλα ώστε η βελτιστοποίηση να αντιστοιχεί στην ελαχιστοποίηση της συνάρτησης ενέργειας του δικτύου.

2.6 Περιορισμοί των δικτύων Hopfield

Το δίτκυο Hopfield θα συγκλίνει πάντα σε μια σταθερή κατάσταση (κατάσταση ισορροπίας), αλλά δε μας εξασφαλίζει ότι η αυτή η κατάσταση θα είναι η βέλτιστη αν μιλάμε για προβλήματα βελτιστοποίησης, ή ένα από τα αποθηκευμένα πρότυπα αν μιλαμε για προβλήματα αναγνώρισης προτύπων. Αν η αρχική κατάσταση είναι κοντά σε ένα τοπικό ελάχιστο τότε το δίκτυο θα συγκλίνει σε αυτό, χωρίς να σημαίνει ότι αυτό το σημείο είναι η βέλτιστη λύση. Ένα άλλο πρόβλημα είναι η αποθηκευτική χωριτικότητα των δικτύων. Ο μέγιστος αριθμός C_{max} των προτύπων που μπορούν να αποθηκευτούν και να ανακληθούν χωρίς σφάλμα σε ένα δίκτυο Hopfield με n νευρώνες είναι:

$$C_{max} = \frac{n}{4 \ln n} \tag{2.4}$$

ενώ σε περίπτωση που δε μας ενδιαφέρει εαν η ανάκληση γίνει σωστά τότε έχουμε:

$$C_{max} = 0.15n \tag{2.5}$$

Κεφάλαιο 3

Το πρόβλημα του περιοδεύοντος πωλητή

3.1 Εισαγωγή

Το πρόβλημα του περιοδεύοντος πωλητή (Traveling Salesman Problem - TSP) πρωτοδιατυπώθηκε σαν μαθηματικό πρόβλημα το 1930 και αποτέλεσε ένα από τα πιο ενδιαφέροντα και γνωστά προβλήματα βελτιστοποίησης. Θα μπορούσαμε να δώσουμε τον ορισμό του προβλήματος ως εξής : *Δοσμένου ενός πλήθους πόλεων και το κόστος μεταξύ τους να βρούμε το φθηνότερο τρόπο να επισκεφτούμε όλες τις πόλεις και να επιστρέψουμε στην αρχική με τον περιορισμό να μην επισκεφτούμε μια πόλη πάνω από δυο φορές*.

Το κόστος μεταξύ δύο πόλεων μπορεί να εξαρτάται από διάφορους παράγοντες όπως το μήκος της μεταξύ τους διαδρομής, διάφορα εμπόδια που τυχόν υπάρχουν στη διαδρομή, το κόστος των διοδίων αν υπάρχουν κ.α. και είναι συμμετρικό, με την έννοια ότι αν πούμε ότι το κόστος για να πάμε από την πόλη X στην πόλη Y είναι Z, τότε και για να πάμε από την πόλη Y στην πόλη X το κόστος θα είναι πάλι Z. Η άλλη παραλλαγή είναι, να έχουμε ασυμμετρικά κόστη. Σε αυτήν την περίπτωση θα πρέπει να ξεχωρίσουμε την κάθε διαδρομή από την αντίθετη της.

Το TSP βρίσκει διάφορες εφαρμογές σε προβλήματα. Ενδεικτικά μπορούμε να αναφέρουμε μερικές:

- Προγραμματισμός των σχολικών λεωφορείων για να πάρουν τα παιδιά προσπαθώντας να κάνουν την μικρότερη δυνατή διαδρομή.
- Διανομή προιόντων από εργοστάσια σε super market.
- Δρομολόγια αεροπλάνων.
- Διατρητικές μηχανές. Οι διατρητικές μηχανές είναι αυτόματες μηχανές οι οποίες προγραμματίζονται ώστε να ανοίγουν τρύπες σε τυπωμένα κυκλώματα. Οι τρύπες παίζουν το ρόλο των πόλεων και ο χρόνος που κάνει η μηχανή για να μεταφερθεί από το ένα σημείο στο άλλο είναι το κόστος. Όσο πιο μικρή διαδρομή κάνει η μηχανή σύμφωνα πάντα με τις συντεταγμένες, τόσο λιγότερος χρόνος χρειάζεται για να ανοιχτούν οι τρύπες στην πλακέτα με αποτέλεσμα να αυξάνεται η παραγωγή.

3.2 Τρόποι λύσης

Το πρόβλημα του περιοδεύοντος πωλητή λόγο της απλότητας του ως πρόβλημα και της πολυπλοκότητας του ως επίλυση, έγινε και είναι ακόμα αντικείμενο μελέτης από πολλούς επιστήμονες. Έχει εξακριβωθεί ότι ο αριθμός των πιθανών λύσεων αυξάνεται εκθετικά όσο αυξάνονται και οι πόλεις, σε βαθμό που γίνεται αδύνατος ο υπολογισμός της βέλτιστης λύσης. Γι'αυτόν το λόγο δίνεται περισσότερο βάση σε ευρετικούς αλγόριθμους οι οποίοι βρίσκουν σχεδόν-βέλτιστη λύση αλλά σε μικρό χρόνο. Μπορούμε να χωρίσουμε τους αλγόριθμους σε κατηγορίες ανάλογα με τον τρόπο επίλυσης. Έτσι μπορούμε να έχουμε:

- Κατασκευαστικούς αλγόριθμους
- Αλγόριθμους βελτίωσης
- Υβριδικούς αλγόριθμους

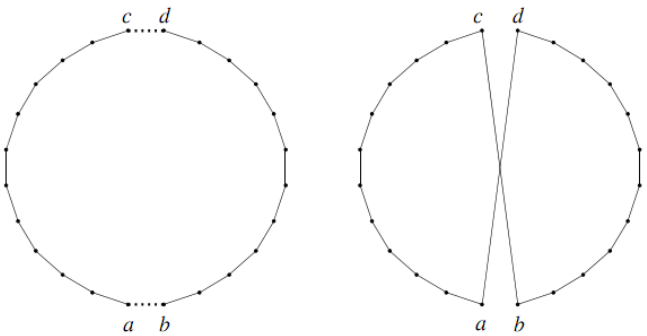
Σκοπός των κατασκευαστικών αλγορίθμων είναι να δημιουργήσουν τη λύση από την αρχή, συνήθως τοποθετόντας μια πόλη κάθε φορά μέχρι να δημιουργηθεί μια ολοκληρωμένη λύση. Οι αλγόριθμοι βελτίωσης παίρνουν μια αρχική λύση την οποία βελτιώνουν βηματικά. Τέλος οι υβριδικοί αλγόριθμοι δέχονται μια αρχικά αποδεκτή λύση από κάποιο κατασκευαστικό αλγόριθμο την οποία βελτιώνουν με βάση κάποιο αλγόριθμο βελτίωσης. Παρακάτω θα αναφερθούμε σε μερικούς από τους πιο σημαντικούς αλγόριθμους.

3.2.1 Αλγόριθμος πλησιέστερου γείτονα

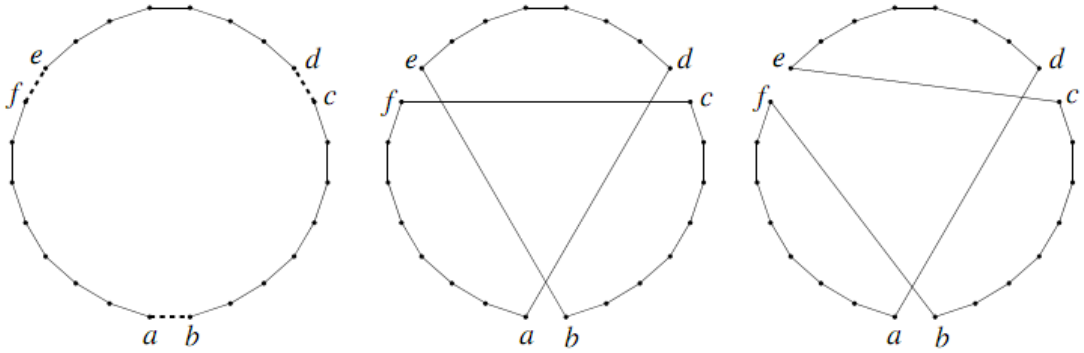
Ο αλγόριθμος του πλησιέστερου γείτονα ανήκει στους κατασκευαστικούς αλγόριθμους και είναι ίσως ο πιο απλός αλγόριθμος σε ότι αφορά την κατανόηση και την κατασκευή. Ο τρόπος λειτουργίας του μιμείται έναν πωλητή ο οποίος σε κάθε βήμα του πηγαίνει στην πιο κοντινή πόλη την οποία δεν έχει επισκεφτεί. Αν και ο χρόνος που χρειάζεται για να τρέξει είναι ο πιο μικρός από όλους τους αλγόριθμους ($O(N^2)$ όπου N είναι το πλήθος των πόλεων) η ποιότητα των λύσεων είναι ικανοποιητική μόνο για μικρά σετ πόλεων.

3.2.2 2-Opt, 3-Opt

Οι αλγόριθμοι αυτοί ανήκουν στην κατηγορία των αλγόριθμων βελτίωσης. Επειδή μοιάζουν μεταξύ τους θα τους αναφέρουμε στην ίδια παράγραφο. Οι αλγόριθμοι αυτοί δέχονται σαν είσοδο μια αρχική λύση η οποία μπορεί να έχει δημιουργηθεί με όποιο τρόπο εμείς θέλουμε (π.χ. με του πλησιέστερου γείτονα). Στη σύνέχεια αυτό που κάνουν είναι να προσπαθούν σε κάθε βήμα να βελτιώνουν αυτήν τη λύση έως ότου φτάσουν σε σημείο να μην μπορεί να βελτιωθεί άλλο, φτάνουμε δηλαδή σε τοπικό ελάχιστο. Ο τρόπος λειτουργίας τους είναι ο εξής: Σε κάθε βήμα του ο 2-Opt διαγράφει δύο ακμές χωρίζοντας τη διαδρομή σε δύο και στη συνέχεια τις ενώνει με τον αντίθετο τρόπο. Εαν η καινούργια διαδρομή είναι μικρότερη από την παλιά τότε την κρατάει, αλλιώς συνεχίζει με τις επόμενες. Στον 3-Opt η διαδικασία είναι παρόμοια με αυτήν του 2-Opt εκτός του ότι σε κάθε βήμα χωρίζουμε τη διαδρομή σε 3 διαδρομές και τις ενώνουμε πάλι ώστε να μικρίνει η συνολική διαδρομή. Εάν το τοπικό ελάχιστο (τελική λύση) είναι ταυτόχρονα και ολικό ελάχιστο τότε η λύση είναι η βέλτιστη.



Σχήμα 3.1: 2-Opt. Διάσπαση και ένωση δύο ακμών



Σχήμα 3.2: 3-Opt. Πιθανές εκδοχές ένωσης ακμών

Στην κατηγορία των αλγορίθμων βελτίωσης ανήκουν εν μέρη, και οι γενετικοί αλγόριθμοι, οι αποικίες μυρμηγκιών και τα νευρωνικά δίκτυα. Τα μοντέλα αυτά δείχνουν πως μπορούν διαδικασίες από την πραγματική ζωή με κατάλληλη μετατροπή να χρησιμοποιηθούν ως τρόποι επίλυσης σε προβλήματα υπολογιστών, και θα τα αναφέρουμε ξεχωριστά.

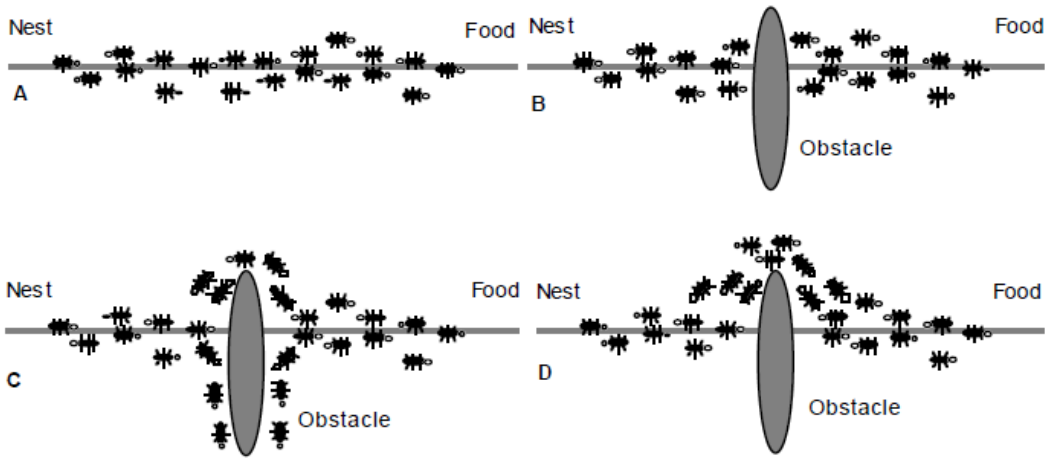
3.2.3 Γενετικοί αλγόριθμοι (ΓΑ)

Οι ΓΑ είναι ευρετικοί αλγόριθμοι βελτιστοποίησης, που βασίζονται σε ένα γενικό πληθυσμό. Ο αλγόριθμος ξεκινάει με ένα πλήθος από λύσεις (χρωμοσώματα) που αποτελούν τον πληθυσμό. Στη συνέχεια σε κάθε γενιά παίρνουμε λύσεις από προηγούμενες γενιές και δημιουργούμε έναν καινούργιο πλυθησμό. Στη διαδικασία αυτή χρησιμοποιούμε μηχανισμούς, εμπνευσμένους από τη βιολογική εξέλιξη, όπως η αναπαραγωγή, η μετάλλαξη, ο ανασυνδυασμός και η επιλογή, με σκοπό η καινούργια γενιά να είναι καλύτερη από την προηγούμενη. Οι λύσεις κάθε γενιάς περνάνε από μια συνάρτηση αξιολόγησης έτσι ώστε αυτές με τις καλύτερες τιμές να έχουν περισσότερες πιθανότητες να επιλεχτούν για γονείς. Σκοπός είναι στο τέλος να “επιβιώσει” το καλύτερο άτομο δηλαδή να βρεθεί η καλύτερη λύση. Για το πρόβλημα του περιοδεύοντος πωλητή, το χρωμόσωμα κωδικοποιείται σαν ένας μονοδιάστατος πίνακας ο οποίος περιέχει όλες τις πόλεις που πρέπει να επισκεφτούμε. Η συνάρτηση αξιολόγησης μετράει το μήκος της διαδρομής και δίνει στο κάθε χρωμόσωμα, ανάλογα με το πόσο μικρή ή μεγάλη είναι, την κατάλληλη τιμή. Οι νέες γενιές δημιουργούνται χρησιμοποιώντας ανασυνδυασμό δυο σημείων. Οι γενετικοί αλγόριθμοι δεν εγγυώνται καλά αποτελέσματα αλλά κατά ένα μεγάλο ποσοστό τα αποτελέσματα τους είναι αρκετά καλά. Μειονέκτημα τους είναι ότι δεν μπορούμε να προβλέψουμε το χρόνο που χρειάζονται για να

συγκλίνουν σε μια λύση, γι'αυτό σε μεγάλα προβλήματα χρησιμοποιούνται διάφορες μέθοδοι και από άποψη υλικού (πολυνημάτωση, παράλληλη επεξεργασία), και από άποψη λογισμικού (βελτίωση αρχικού πλυθυσμού, ευρετικές μέθοδοι) για να επιταχύνουν τη διαδικασία.

3.2.4 Αποικία μυρμηγκιών

Τα μυρμήγκια στην πραγματική ζωή είναι πολύ ικανά στο να βρίσκουν την κοντινότερη διαδρομή από τη φωλιά στο φαγητό. Ακόμα και στην περίπτωση που η υπάρχουσα διαδρομή καταστραφεί, τα μυρμήγκια μπορούν και προσαρμόζονται ώστε να βρίσκουν την επόμενη καλύτερη διαδρομή. Αυτό είναι που εκμεταλέυεται αυτή η μέθοδος η οποία χρησιμοποιεί τεχνητά μυρμήγκια. Ένα τεχνητό μυρμήγκι κινείται από πόλη σε πόλη ανάλογα με τη μεταξύ τους απόσταση και το ίχνος της φερορμόνης που έχουν αφήσει τα προηγούμενα μυρμήγκια. Σε κάθε μετακίνηση αφήνουν το δικό τους ίχνος ανάλογα με την ποιότητα της μετακίνησης. Όσο προχωράει η διαδικασία η καλύτερη διαδρομή θα αρχίζει να προσελκύει περισσότερα μυρμήγκια και τελικά θα φτάσει σε σημείο όπου όλα τα μυρμήγκια θα ακολουθούν την ίδια διαδρομή.



Σχήμα 3.3: (A) Αρχική διαδρομή. (B) Η διαδρομή κλείνεται από ένα εμπόδιο. (C) Η φερορμόνη μαζεύτεται πιο γρήγορα στην καλύτερη διαδρομή. (D) Όλα τα μυρμήγκια επιλέγουν την καλύτερη διαδρομή.

Κεφάλαιο 4

Λύνοντας το πρόβλημα του περιοδεύοντος πωλητή με δίκτυο Hopfield

Όπως αναφέραμε πιο πάνω το δίκτυο hopfield έχει την δυνατότητα να ελαχιστοποιεί τη συνάρτηση ενέργειας. Αν πάρουμε ως αναφορά αυτό τότε αν καταφέρουμε να σχηματίσουμε μια συνάρτηση ενέργειας δευτέρου βαθμού που να περιγράφει το πρόβλημα μας μπορούμε να χρησιμοποιήσουμε ένα δίκτυο hopfield για να ελαχιστοποιήσουμε τη συνάρτηση ενέργειας και να πετύχουμε τη βελτιστοποίηση της λύσης του προβλήματος. Η πρώτη προσπάθεια επίλυσης του TSP με τη χρήση του δικτύου Hopfield παρουσιάστηκε στην εργασία των Hopfield και Tank [Hopfield . Tank 1985]. Στην εργασία τους αυτή πρότειναν τη χρήση ενός δικτύου hopfield το οποίο δε θα ψάχνει τη βέλτιστη από ένα σύνολο γνωστών διαδρομών αλλά θα προσπαθεί να την κατασκευάσει. Κάθε λύση του προβλήματος αποτελεί ένα τοπικό ελάχιστο της συνάρτησης ενέργειας, γι'αυτό και το μόνο που έχουμε να κάνουμε είναι να σχηματίσουμε μια συνάρτηση ενέργειας δευτέρου βαθμού που να περιγράφει το πρόβλημα μας. Στο πρόγραμμα μας θα χρησιμοποιήσουμε ένα τροποποιημένο μοντέλο των Hopfield - Tank.

4.1 Δημιουργία λύσης

Μελετώντας τα δεδομένα του προβλήματος βλέπουμε ότι μας δίνονται N πόλεις τις οποίες θα πρέπει να επισκεφτούμε και κάθε διαδρομή είναι πιθανή. Μια πρώτη προσέγγιση είναι να αναπαραστήσουμε τη θέση μιας πόλης στη διαδρομή σαν A = (0, 0, 0, 1, 0, 0) όπου το 1 στο παράδειγμα μας σημαίνει ότι η πόλη A θα είναι τέταρτη σε σειρά επίσκεψης. Αν το κάνουμε αυτό για κάθε πόλη, μας οδηγεί στο να αναπαραστήσουμε μια ολόκληρη διαδρομή με ένα πίνακα N*N όπου οι γραμμές αναπαριστούν τις πόλεις και οι στήλες τη σειρά επίσκεψης της κάθε πόλης.

Πίνακας 4.1: Πίνακας πόλεων - επισκέψεων για N = 4

	1	2	3	4
πόλη A	0	1	0	0
πόλη B	0	0	1	0
πόλη Γ	1	0	0	0
πόλη Δ	0	0	0	1

Όπως μπορούμε να δούμε κάθε πίνακας αντιπροσωπεύει μια λύση. Για να είναι η λύση αποδεκτή θα πρέπει να ισχύουν 3 πράγματα.

1. Η κάθε γραμμή να έχει έναν άσο: Να επισκεπτόμαστε κάθε πόλη μια μόνο φορά.
2. Η κάθε στήλη να έχει έναν άσο. Να επισκεπτόμαστε μία πόλη σε κάθε βήμα.
3. Ο αριθμός όλων των άσων στον πίνακα να είναι ίσος με τον αριθμό των πόλεων. Να επισκεπτούμε δηλαδή όλες τις πόλεις χωρίς να παραλείψουμε καμία.

Κατασκευάζοντας το δίκτυο μας θα πρέπει καταρχήν να προσδιορίσουμε τον αριθμό των νευρώνων που θα έχει. Σύμφωνα με τη μορφή της λύσης θέτουμε κάθε κελί του πίνακα ως ένα νευρώνα ο οποίος θα παίρνει τις τιμές 0 έως 1 στη έξοδο του. Η έξοδος κάθε νευρώνα θα ανατροφοδοτείται σε όλους τους άλλους νευρώνας εκτός από τον εαυτό του. Στην εργασία τους οι Hopfield και Tank χρησιμοποίησαν το συνεχές μοντέλο του δικτύου, δηλαδή η έξοδος κάθε νευρώνα δε θα είναι 0 ή 1 αλλά θα καθορίζεται από μια συνάρτηση η οποία είναι συνεχείς. Η συνάρτηση που προτιμήθηκε ήταν η υπερβολική εφαπτομένη.

$$V = f(u) = \frac{1}{2}\left(1 + tanh\left(\frac{u}{u_0}\right)\right) \tag{4.1}$$

όπου το u είναι η έισοδος του νευρώνα,

$$u_{xi} = u_{xi} + \Delta\tau \sum_{j=0}^{n-1} w_{xi,yj} v_{yj} + I_{xi} - \frac{u_{xi}}{\rho} \tag{4.2}$$

το u_0 μια σταθερά που καθορίζει την κλίση της υπερβολικής εφαπτομένης, και το V η έξοδος του νευρώνα. Τους νευρώνες για ευκολία θα τους προσπελάυνουμε με δύο δείκτες x, i, οι οποίοι καθορίζουν σε ποια πόλη αναφερόμαστε και σε ποιά θέση. Έτσι ο νευρώνας V_{xi} θα είναι ο νευρώνας που αντιστοιχεί στην πόλη x στη i-στη θέση επίσκεψης.

Το επόμενο βήμα είναι να σχηματίσουμε τη συνάρτηση ενέργειας η οποία περιγράφει το πρόβλημα και που στην μικρότερη τιμή της (ολικό ελάχιστο) αντιστοιχεί στην καλύτερη διαδρομή. Για να συμβεί αυτό θα πρέπει εκτός από τους περιορισμούς που καθορίσαμε πιο πάνω η συνάρτηση ενέργειας να ευνοεί και τις λύσεις με τη μικρότερη διαδρομή. Έτσι καταλήγουμε σε μια συνάρτηση ενέργειας που έχει την παρακάτω μορφή.

$$E = E_1 + E_2 + E_3 + E_4 \tag{4.3}$$

$$E_1 = \frac{A}{2} \sum_{x=1}^{N} \left(\sum_{i=1}^{N} V_{xi} - 1 \right)^2 \tag{4.4}$$

$$E_2 = \frac{B}{2} \sum_{i=1}^{N} \left(\sum_{x=1}^{N} V_{xi} - 1 \right)^2 \tag{4.5}$$

$$E_3 = \frac{C}{2} \sum_{x=1}^{N} \sum_{i=1}^{N} V_{xi}(1 - V_{xi}) \tag{4.6}$$

$$E_4 = \frac{D}{2} \sum_{x=1}^{N} \sum_{y=1}^{N} \sum_{i=1}^{N} d_{xy} V_{xi}(V_{y,i+1} + V_{y,i-1}) \quad x \neq y \tag{4.7}$$

Ο κάθε παράγοντας της συνάρτησης αυτής ορίζει και έναν περιορισμό. Έτσι το πρώτο άρθροισμα είναι 0 αν και μόνο αν κάθε γραμμή του πίνακα περιέχει ένα 1 (επισκεπτόμαστε κάθε πόλη μια φορά), το δεύτερο άρθροισμα είναι 0 αν και μόνο αν κάθε στήλη του πίνακα περιέχει ένα 1 (επισκεπτόμαστε μια πόλη κάθε φορά), και το τρίτο άρθροισμα είναι 0 αν και μόνο αν έχουμε τόσους άσους στον πίνακα όσες και οι πόλεις (δεν αφήσαμε καμία πόλη χωρίς να την επισκεπτούμε). Το τελευταίο άθροισμα δεν έχει να κάνει τόσο με την ορθότητα της λύσης όσο με την ποιότητα της. Το άθροισμα αυτό

κρατάει τη συνολική διαδρομή και δίνει μικρή ή μεγάλη ποινή στις λύσεις ανάλογα με το μήκος της διαδρομής.

Παίρνοντας υπόψη τις εξισώσεις (4.3) και (2.3), ο τύπος για τον υπολογισμό του πίνακα των βαρών είναι ο εξής:

$$w_{xi,yj} = -A_{xy} - B_{ij} + C_{xyij} - Dd_{xy}(_{j,i+1} + _{j,i-1}) \tag{4.8}$$

όπου $w_{xi,yj}$ είναι το βάρος της σύνδεσης μεταξύ του νευρώνα xi και yj, d_{xy} είναι η απόσταση μεταξύ των πόλεων x, y και

$$\delta_{ij} = \begin{cases} 1, & \text{αν } i = j \\ 0, & \text{αν } i \neq j \end{cases} \tag{4.9}$$

Για ευκολία στην κωδικοποίηση αλλά και στην κατανόηση μπορούμε να σπάσουμε την παραπάνω συνάρτηση των βαρών σε τρεις περιπτώσεις ανάλογα με τους νευρώνες που εμπλέκονται.

- Για νευρώνες που βρίσκονται στην ίδια γραμμή έχουμε: $w_{xi,xj} = -A - C$
- Για νευρώνες στην ίδια στήλη: $w_{xi,yi} = -B - C$
- Για όλους τους άλλους νευρώνες ισχύει: $w_{xi,yj} = -C - Dd_{xy}$

Τα A, B, C, D είναι παράμετροι που καθορίζουν σε μεγάλο βαθμό τη σύγκλιση του αλγορίθμου ενώ παράλληλα μας επιτρέπουν να δώσουμε προτεραιότητα σε λύσεις ανάλογα με το τί θέλουμε. Η επιλογή τους είναι αρκετά δύσκολη και συνήθως γίνεται με τη μέθοδο της δοκιμής. Κρατώντας τις τιμές των A, B, C σε πιο μεγάλα επίπεδα από αυτή της D, ευνοούμε την εύρεση αποδεκτών λύσεων οι οποίες όμως δύσκολα θα είναι οι βέλτιστες. Από την άλλη αυξάνοντας την τιμή της D πάνω απο των άλλων δίνουμε ένα πλεονέκτημα σε λύσεις με μικρό κόστος διαδρομής, αλλά δεν είναι σίγουρο ότι θα είναι αποδεκτές.

Κεφάλαιο 5

Υλοποίηση

5.1 Απαιτήσεις εφαρμογής

Η εφαρμογή μας υλοποιεί ένα δίκτυο hopfield, του οποίου η δομή είναι αυτή που περιγράψαμε στο παραπάνω κεφάλαιο, σε ένα εύχρηστο γραφικό περιβάλλον. Βασική απαίτηση από την εφαρμογή είναι να φαίνονται, σε κάθε βήμα του αλγόριθμου, οι πόλεις που επισκεπτόμαστε σε ένα χάρτη καθώς και η συνολική απόσταση που διανύουμε. Για να γίνει η εφαρμογή μας πιο λειτουργική και πιο γενικευμένη προσθέσαμε επιπλέον τις παρακάτω απαιτήσεις:

- Ο χρήστης θα μπορεί να αλλάζει και να προσθέτει πόλεις δυναμικά καθώς και τις αποστάσεις μεταξύ τους.
- Ο χάρτης δε θα είναι στατικός (εικόνα). Γι'αυτόη το λόγο χρησιμοποιήθηκε η βιβλιοθήκη JXMapViewer η οποία προσθέτει επιπλέον λειτουργικότητα από έναν απλό χάρτη.
- Οι περισσότεροι παράμετροι του δικτύου θα είναι παραμετροποιήσιμοι.
- Ο αλγόριθμος θα έχει τρείς καταστάσεις. Γρήγορη, κανονική, πλήρης. Στη γρήγορη θα εμφανίζεται στο τέλος μόνο η διαδρομή στο χάρτη μαζί με τις αποστάσεις. Στην κανονική θα εμφανίζονται απλά οι πόλεις και οι συνδέσεις μεταξύ τους. Στην πλήρης θα εμφανίζεται σε κάθε βήμα η κατάσταση των εξόδων, η ενέργεια του δικτύου και επιπλέον πληροφορίες τις οποίες θα επιλέγει ο χρήστης.
- Κάθε τρέξιμο θα αποθηκεύεται προσωρινά ώστε να μπορούμε να συγκρίνουμε τα αποτελέσματα.

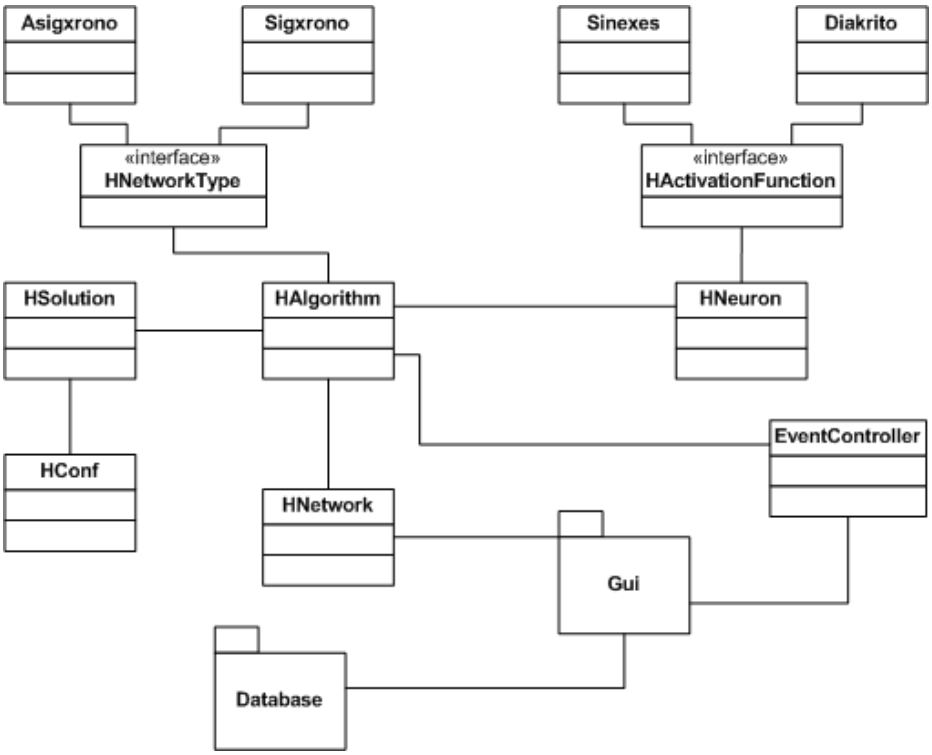
- Ο χρήστης θα έχει τη δυνατότητα να αποθηκέυει την κατάσταση και τις παραμέτρους ενός τρεξίματος ώστε να μπορεί να αναπαράγει το ίδιο αποτέλεσμα όποια στιγμή θέλει.
- Μαζική εκτέλεση πολλών δοκιμών και εύρεση της καλύτερης λύσης.
- Τέλος επειδή η εφαρμογή θα είναι ανοιχτού κώδικα, όλα τα εργαλεία και οι βιβλιοθήκες που θα χρησιμοποιθούν θα πρέπει να είναι ανοιχτού κώδικα ή ελεύθερα (άδειες GLP, LGPL, LPPL κτλ.).

Για τη κατασκευή της εφαρμογής μας χρησιμοποιήσαμε τη γλώσσα προγραμματισμού Java. Η Java είναι μια γλώσσα υψηλού επιπέδου με πολύ πλούσια βιβλιοθήκη. Βασικό προτέριμα της είναι ότι μπορεί και τρέχει ανεξαρτήτως λειτουργικού συστήματος (Windows, Linux, Macintosh) και πλατφόρμας (Intel x86, IBM, Sun SPARC, Motorola) χωρίς να χρειάζεται επαναμεταγλώττιση, χάρη στην εικονική της μηχανή (Virtual Machine ή VM) η οποία μεταφράζει τα μεταγλωττισμένα αρχεία του προγράμματος στην κατάλληλη κάθε φορά γλώσσα μηχανής. Για τη συγγραφή του κώδικα χρησιμοποιήθηκε το πρόγραμμα NetBeans IDE. Το NetBeans είναι ένα ολοκληρωμένο περιβάλλον ανάπτυξης, ανοιχτού κώδικα, το οποίο περιλαμβάνει επεξεργαστή πηγαίου κώδικα, μεταγλωττιστή, αποσφαλματωτή και άλλα εργαλεία που μας βοηθάνε στη δημιουργία και την επίβλεψη μεγάλων προγραμμάτων.

5.2 Κατασκευή δικτύου και αλγόριθμος λειτουργίας

Η δομή του δικτύου και τα επιμέρους στοιχεία του φαίνονται στο παρακάτω σχήμα. Κάθε φορά πριν αρχίσει μια εκτέλεση του αλγορίθμου δημιουργείται ένα αντικείμενο (HConf) το οποίο κρατάει όλες τις απαραίτητες πληροφορίες για να είναι δυνατή η εκτέλεση του. Επίσης είναι αυτό στο οποίο φορτώνονται όλες οι παράμετροι όταν ανοίγουμε ένα αρχειο. Οι πληροφορίες αυτές περιλαμβάνουν τις διάφορες παραμέτρους που ορίζουμε μέσω των παραθύρων διαλόγου, καθώς και τις πόλεις και τις αποστάσεις μεταξύ τους, που συμμετέχουν στο προβλημα. Για κάθε εκτέλεση υπάρχει ένα αντικείμενο (HSolution) το οποίο δημιουργήται στην αρχή της κάθε εκτέλεσης και ενημερώνεται σε κάθε εποχή. Το αντικείμενο αυτό περιέχει πληροφορίες σχετικά με τη λύση. Τέτοιες πληροφορίες είναι η συνολική απόσταση της διαδρομής, ποιές πόλεις συνδέονται με

ποιές, εαν η λύση είναι αποδεκτή η όχι, η ενέργεια του δικτύου καθώς και η παραμετροποίηση που χρησιμοποιείται. Το αντικείμενο αυτό χρησιμοποιείται από διάφορα μέρη του προγράμματος ώστε να ενημερωθεί με τις νέες τιμές το γραφικό περιβάλλον. Αυτό γίνεται είτε κατά τη φάση της εκτέλεσης όπου ενημερώνεται το γράφημα της ενέργειας και η περιοχή πληροφοριών, είτε και μετά, στο ιστορικό όπου ενημερώνεται ο χάρτης με την τρέχουσα λύση. Ο πυρήνας του δικτύου μας είναι ο τεχνητός νευρώνας (HNeuron) και αποτελείται από έναν πίνακα με τα συναπτικά βάρη προς τους υπόλοιπους νευρώνες, την έξοδο του νευρώνα, τη διέγερση, και τον τύπο της συνάρτησης ενεργοποίησης. Οι νευρώνες μπορούν να έχουν δυο διαφορετικούς τρόπους υπολογισμού της εξόδου τους, είτε με τη συνάρτηση της υπερβολικής εφαπτομένης (συνεχές), είτε με τη βηματική συνάρτηση (διακριτός). Όλοι οι νευρώνες δημιουργούνται από το αντικείμενο HAlgorithm το οποίο παράλληλα είναι υπεύθυνο για τη διαχείριση τους και την εκτέλεση του αλγορίθμου. Ο σύνδεσμος μεταξύ του γραφικού περιβάλλοντος και του δικτύου είναι το αντικείμενο HNetwork στο οποίο υλοποιούνται οι λειτουργίες τις έναρξης, πάυσης, διακοπής, αλλά και τις μαζικής εκτέλεσης.



Σχήμα 5.1: Διάγραμμα κλάσεων - Πακέτο HPackage

5.2.1 Χάρτης

Για την προβολή και των πόλεων στο χάρτη προτιμήσαμε στη θέση μιας στατικής εικόνας να χρησιμοποιήσουμε ένα δυναμικό χάρτη όπως αυτόν του google maps και της microsoft. Τη δυνατότητα αυτή μας την προσέφερε η βιβλιοθήκη swingx-ws. Η βιβλιοθήκη αυτή προσφέρει ένα πλήθος απο JavaBeans με τα οποία μπορούμε να αλληλεπιδράσουμε με διάφορες διαδικτυακές υπηρεσίες, όπως τη μηχανή αναζήτησης του Yahoo, και της Google. Το πρόγραμμα μας χρησιμοποιεί το JXMApViewer, ένα κομμάτι της βιβλιοθήκης swingx-ws, για να προβάλλει έτοιμες εικόνες από διάφορους map servers σε ένα δυναμικό χάρτη με επιλογές zooming. Για τις ανάγκες τις εφαρμογής μας έπρεπε να κάνουμε μερικές ρυθμίσεις ώστε να προσφέρουμε τη δυνατότητα στο χρήστη να φορτώσει τις εικόνες τόσο από το διαδύκτιο όσο και από το τοπικό του δίσκο (σε περίπτωση που δεν υπάρχει διαθέσιμη σύνδεση). Ο προεπιλεγμένος map server είναι ο OpenStreetMaps ο οποίος παρέχει γεωγραφικά δεδομένα όπως χάρτες δρόμων, ελέυθερα χωρίς περιορισμούς. Παράλληλα επιτρέπει σε εγγεγραμμένους χρήστες να προσθέτουν και να τροποποιούν χάρτες, δημιουργώντας έτσι μια πολύ μεγάλη βάση δεδομένων. Στο χάρτη μπορούμε εκτός από το να εμφανίσουμε τις πόλεις που φορτώνουμε από τη βάση, να προσθέσουμε δικές μας και να αφαιρέσουμε ήδη υπάρχοντες. Οι αποστάσεις μεταξύ των πόλεων υπολογίζονται είτε από τα δεδομένα που έχουμε στη βάση (αν δεν έχει οριστεί κάποια απόσταση τότε παίρνει τη τιμή 0), είτε δυναμικά με βάση το γεωγραφικό μήκος και το γεωγραφικό πλάτος των πόλεων. Για τον υπολογισμό αυτόν χρησιμοποιούμε τη φόρμουλα του Thaddeus Vincenty [cite]η οποία επιτρέπει εύρεση της απόστασης με πολύ μεγάλη ακρίβεια.

5.2.2 Αρχικοποίηση δεδομένων

Σε κάθε εκτέλεση του νευρωνικού δικτύου πρέπει να ορίσουμε ένα πλήθος πόλεων που θα πάρουν μέρος στη λύση. Επειδή όμως ειναι δύσκολο κάθε φορά να βρούμε στο χάρτη τις πόλεις που είχαμε ορίσει τις προηγούμενες φορές, μπορούμε να τις αποθηκεύσουμε σε μια βάση δεδομένων ώστε να τις έχουμε διαθέσιμες όποτε θέλουμε. Στη βάση αυτή αποθηκεύονται και οι αποστάσεις μεταξύ των πόλεων. Θεωρούμε ότι η απόσταση μεταξύ της πόλης Α και της πόλης Β είναι ίδια με αυτή, της πόλης Β με της πόλης Α (συμμετρικό TSP). Λόγο της απλότητας των πινάκων και του μικρού μεγέθους της βάσης προτιμήσαμε να χρησιμοποιήσουμε τη Microsoft Access σαν βάση δεδομένων. Επίσης η Microsoft Access έχει το πλεονέκτημα ότι όλοι οι πίνακες της

βάσης και τα δεδομένα βρίσκονται σε ένα αρχείο, το οποίο μπορει να μεταφερθεί εύκολα από υπολογιστή σε υπολογιστή χωρίς να χρειάζεται καμία εγκατάσταση ή άλλες ρυθμίσεις. Κάθε αρχείο βάσης της Microsoft Access μπορεί να χρησιμοποιηθεί αρκεί να έχει τουλάχιστον τους 2 παρακάτω πίνακες με τις ανάλογες στήλες:
“Πόλεις” [ID, Πόλη, Γεωγραφικό πλάτος, Γεωγραφικό μήκος]
“Αποστάσεις” [ID, Πόλη από, Πόλη προς, Απόσταση]
Η εισαγωγή, τροποποίηση και διαγραφή εγγραφών στη βάση γίνεται μέσα από το πρόγραμμα από την καρτέλα “Διαχείριση βάσης” εύκολα και γρήγορα χωρίς να χρειάζεται ιδιαίτερες γνώσεις.

5.2.3 Καταστάσεις εκτέλεσης

Όπως αναφέραμε παραπάνω, υπάρχουν τρεις καταστάσεις εκτέλεσης. Αυτό έγινε για δύο λόγους. Πρώτον διότι σε τέτοιου είδους προβλήματα δεν είναι γνωστός ο χρόνος για την εύρεση της λύσης ο οποίος μπορεί να είναι απαγορευτικά μεγάλος, και δεύτερον επειδή το πρόγραμμα φτιάχτηκε όχι μόνο για να εμφανίζει την τελική λύση αλλά και να προσφέρει ένα περιβάλλον μελέτης του αλγορίθμου κατα τη διάρκεια εκτέλεσης του. Έτσι ανάλογα με την προτίμηση μας μπορούμε να επιλέξουμε:

- τη γρήγορη κατάσταση και παράλληλα να ορίσουμε αρκετές εκτελέσεις ώστε το πρόγραμμα να βρει την καλύτερη. Αυτό μας διευκολύνει στις προσομοιώσεις και στα πειράματα μας όταν θέλουμε να βγάλουμε το μέσο όρο των αποτελεσμάτων και πρέπει να τρέξουμε το ίδιο προβλημα με τις ίδιες παραμέτρους πολλές φορές.
- την κανονική όπου απλά θα φαίνεται η διαδρομή σε κάθε εποχή,
- ή την πλήρη, στην οποία εκτός από τη διαδρομή βλέπουμε και διάφορες άλλες πληροφορίες. Στην καρτέλα πληροφορίες υπάρχει ένα διάγραμμα με τη ενέργεια του δικτύου, καθώς και ένας χώρος όπου μπορούμε να ορίσουμε εμείς τις πληροφορίες που θέλουμε (εξόδους νευρώνων, ενέργεια δικτύου, εποχές κ.α.). Επίσης αυτό που βρήκαμε χρήσιμο από άλλα παραδείγματα είναι να εμφανίζουμε με γραφικό τρόπο την κατάσταση των εξόδων των νευρώνων, ώστε να αντιλαμβανόμαστε καλύτερα πως δημιουργείται η λύση.

5.2.4 Αποθήκευση κατάστασης δικτύου

Μετά από κάθε επιτυχή εκτέλεση του αλγόριθμου έχουμε τη δυνατότητα να αποθηκεύσουμε την τρέχουσα κατάσταση του δικτύου σε ένα αρχείο της επιλογής μας ώστε να μπορούμε όποτε θελήσουμε να το φορτώσουμε και να αναπαράγουμε το ίδιο αποτέλεσμα. Η αποθήκευση γίνεται σε αρχεία xml και οι πληροφορίες που γράφουμε είναι οι βασικές παράμετροι του δικτύου (A, B, C, D, I, uo, DT), ο τύπος του δικτύου, ο τύπος της συνάρτησης ενεργοποίησης, οι επιλεγμένες πόλεις και οι αποστάσεις μεταξύ τους και οι αρχικές διεγέρσεις των νευρώνων. Η επιλογή αυτών των παραμέτρων έγινε με σκοπό να είναι ανεξάρτητη η εκτέλεση του αλγορίθμου από άλλα στοιχεία όταν φορτώνουμε ένα τέτοιο αρχείο καθώς ότι χρειαζόμαστε περιέχεται μέσα στο xml.

5.2.5 Ιστορικό

Επειδή δίνεται η δυνατότητα στο χρήστη να τρέξει πολλές φορές τον αλγόριθμο και να πάρει πολλά και διαφορετικά αποτελέσματα, σκεφτήκαμε ότι θα ήταν καλό να κρατάμε και ένα ιστορικό των εκτελέσεων. Η εμφάνιση του ιστορικού γίνεται σε έναν πίνακα όπου αναφέρεται το τελικό αποτέλεσμα της λύσης, οι παράμετροι που χρησιμοποιήθηκαν καθώς επίσης επιλέγοντας μια λύση μας εμφανίζεται η διαδρομή στο χάρτη ώστε να μπορούμε να ανατρέξουμε σε μια παλιότερη εκτέλεση και να τη συγκρίνουμε με μια πιο πρόσφατη. Με αυτόν τον τρόπο αποφεύγεται η διαδικασία αποθήκευσης των αποτελεσμάτων για προσωρινές εκτελέσεις κάνοντας το πρόγραμμα πιο εύχρηστο. Παράλληλα με το ιστορικό, ενσωματώσαμε ένα γράφημα διασποράς για να υπάρχει μια συνολική εικόνα των αποτελεσμάτων. Το διάγραμμα αυτό δείχνει την κατανομή των λύσεων και είναι αρκετά βολικό όταν ορίζουμε πολλές επαναλήψεις εκτέλεσης του αλγορίθμου.

5.3 Προσομοίωση

Για πειραματική εκτέλεση του αλγορίθου, την εύρεση αποτελεσμάτων και την εξαγωγή συμπερασμάτων θα χρησιμοποιηθεί ένας σταθερός αριθμός πόλεων, οι οποίες ειναι αποθηκευμένες στη βάση. Οι δοκιμές θα χωριστούν σε δύο βασικές κατηγορίες. Στη μια όλες οι πόλεις συνδέονται μεταξύ τους ενώ στην άλλη οι αποστάσεις είναι πραγματικές (υπάρχουν πόλεις που δε συνδέονται μεταξύ τους άμεσα). Για την επιλογή των

παραμέτρων A, B, C, D, I χρησιμοποιήθηκαν αρχικά σταθερές τιμές, αλλά οι λύσεις που έδιναν τις περισσότερες φορές ήταν μη αποδεχτές. Γιαυτό το λόγο χρησιμοποιήσαμε δυναμικά υπολογίσιμες παραμέτρους. Τα δυο μοντέλα υπολογίζουν τις παραμέτρους ως εξής:
Μοντέλο Talavan Yanez [Talavan . Yanez 2002]

$$D = \frac{1}{d_{max}} \tag{5.1}$$

$$B = 3d_{max} + C \tag{5.2}$$

$$A = B - Dd_{min} \tag{5.3}$$

$$I = n + \frac{3Dd_{max}}{C} \tag{5.4}$$

Μοντέλο Tan, Tang, Ge [Tan . 2005]

$$D = \frac{C}{10d_{max}} \tag{5.5}$$

$$A = \frac{C}{2} - \frac{Dd_{min}}{10} \tag{5.6}$$

$$B = A + Dd_{min} \tag{5.7}$$

$$I = A + B - \frac{C}{2} \tag{5.8}$$

όπου d_{max} και d_{min} η μεγαλύτερη, και η μικρότερη απόσταση αντίστοιχα και n ο αριθμός των νευρώνων. Η παράμετρος C είναι η μόνη που ρυθμίζουμε εμείς και μπορεί να πάρει οποιεσδήποτε τιμές. Στη δικιά μας προσομοίωση χρησιμοποιούμε τις τιμές 1, 10. Στο δίκτυο οι νευρώνες ενημερώνονται είτε σύγχρονα είτε ασύγχρονα, και η συνάρτηση ενεργοποίησης των νευρώνων είναι είτε συνεχείς, είτε παίρνει 2 τιμές (0, 1). Για κάθε μια περίπτωση θα γίνουν 1000 επαναλήψεις από τις οποίες θα καταγραφεί η καλύτερη και η μέση λύση, και ο μέσος αριθμός εποχών που σταμάτησε ο αλγόριθμος.

Ο αλγόριθμος που ακολουθείται για την εύρεση της λύσης είναι ο εξής:
Βήμα 1: Κανονικοποιούμε τις αποστάσεις στο διάστημα [0, 1]. Οι πόλεις που έχουν μεταξύ τους απόσταση ίση με 0 τοτε παίρνουν την τιμή 1. Υπολογίζουμε τα βάρη για κάθε νευρώνα σε συνάρτηση με τους υπόλοιπους. Αρχικοποιούμε τις αρχικές εισόδους των νευρώνων σε τυχαίες τιμές μεταξύ του -0.1u_0 και του 0.1u_0. Δε θέλουμε να έχουν όλοι νευρώνες τις ίδιες αρχικές τιμές, ώστε να μπορέσει το δίκτυο να συγκλινει.
Βήμα 2: Βρίσκουμε την διέγερση του πρώτου νευρώνα σύμφωνα με τη (4.1)
Βήμα 3: Εκτελούμε το βήμα 2 για όλους τους υπόλοιπους νευρώνες με τη σειρά. Εαν

το δίκτυο είναι σύγχρονο υπολογίζονται ταυτόχρονα και οι έξοδοι από τη (4.1).

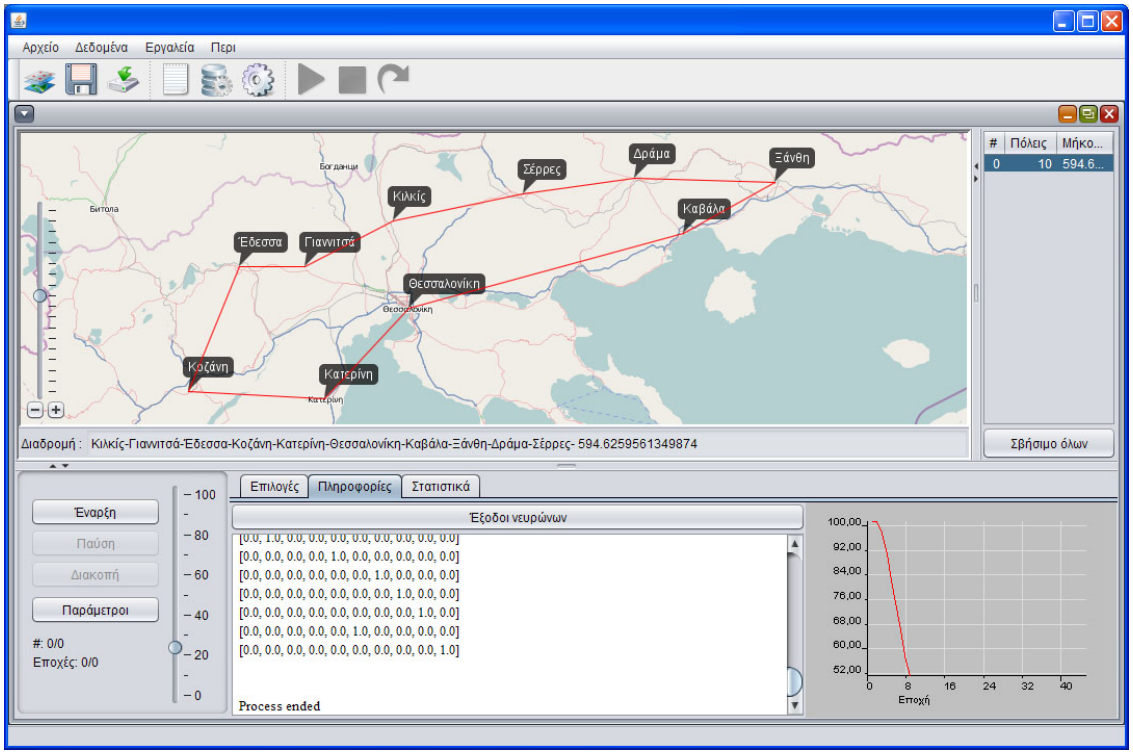
Βήμα 4: Υπολογίζουμε την ενέργεια του δικτύου (4.3)

Βήμα 5: Εκτελούμε τα βήματα 2-4 μέχρι να φτάσουμε τον ζητούμενο αριθμό εποχών ή μέχρι η ενέργεια του δικτύου να μείνει σταθερή για κάποιο συγκεκριμένο αριθμό εποχών

Τα αποτελέσματα στη δεύτερη περίπτωση δεν είναι και τόσο ικανοποιητικά καθώς τις περισσότερες φορές το δίκτυο συγκλίνει σε μη αποδεκτές λύσεις, ενώνοντας πόλεις μεταξύ των οποίων δεν υπάρχει σύνδεση. Στον παρακάτω πίνακα φαίνονται τα αποτελέσματα για την πρώτη περίπτωση όπου όλες οι πόλεις συνδέονται μεταξύ τους. Εδώ το δίκτυο τα πήγε αρκετά καλά βρίσκοντας λύσεις αρκετά κοντά στη βέλτιστη. Παρατηρούμε ότι χρησιμοποιώντας τη συνεχή συνάρτηση για τις εξόδους των νευρώνων τα αποτελέσματα είναι καλύτερα καθώς έτσι το δίκτυο συγκλίνει πιο σταθερά. Μεταβάλλοντας την τιμή του C η σύγκλιση γίνεται σε πιο λίγες εποχές.

Πίνακας 5.1: Αποτελέσματα πειραμάτων

C	Παράμετροι	Τύπος δικτύου	Εν. βαρών	Διαδρομή		Μέσ. Αρ. Εποχών
				Βέλτιστη	Μέση	
1	1	Σύγχρονο	Συνεχή	656.89	1147.03	1474
1	1	Σύγχρονο	Συνεχή	0	0	0
1	1	Ασύγχρονο	Διακριτό	658,85	1159,34	1459
1	1	Ασύγχρονο	Διακριτό	0	0	0
1	2	Σύγχρονο	Συνεχή	0	0	0
1	2	Σύγχρονο	Συνεχή	0	0	0
1	2	Ασύγχρονο	Διακριτό	0	0	0
1	2	Ασύγχρονο	Διακριτό	0	0	0
10	1	Σύγχρονο	Συνεχή	691,355	1165,23	98
10	1	Σύγχρονο	Συνεχή	710,316	1156,153	80
10	1	Ασύγχρονο	Διακριτό	666,616	1066,59	83
10	1	Ασύγχρονο	Διακριτό	966,298	966,298	30
10	2	Σύγχρονο	Συνεχή	666,859	1170,466	58
10	2	Σύγχρονο	Συνεχή	689,287	1168,16	69
10	2	Ασύγχρονο	Διακριτό	594,625	992,886	47
10	2	Ασύγχρονο	Διακριτό	647,266	989,7	24



Σχήμα 5.2: Βέλτιστη λύση

Παράρτημα Α΄

Εγχειρίδιο χρήσης

Αν και στο πρόγραμμα υπάρχει κομμάτι βοήθειας όπου αναφέρονται βασικές οδηγίες για τη χρήση του, θεωρήσαμε σκόπιμο να αναφέρουμε και εδώ μερικές σημαντικές λειτουργίες του προγράμματος πιο αναλυτικά. Οι κύριες λειτουργίες στις οποίες θα αναφερθούμε είναι:

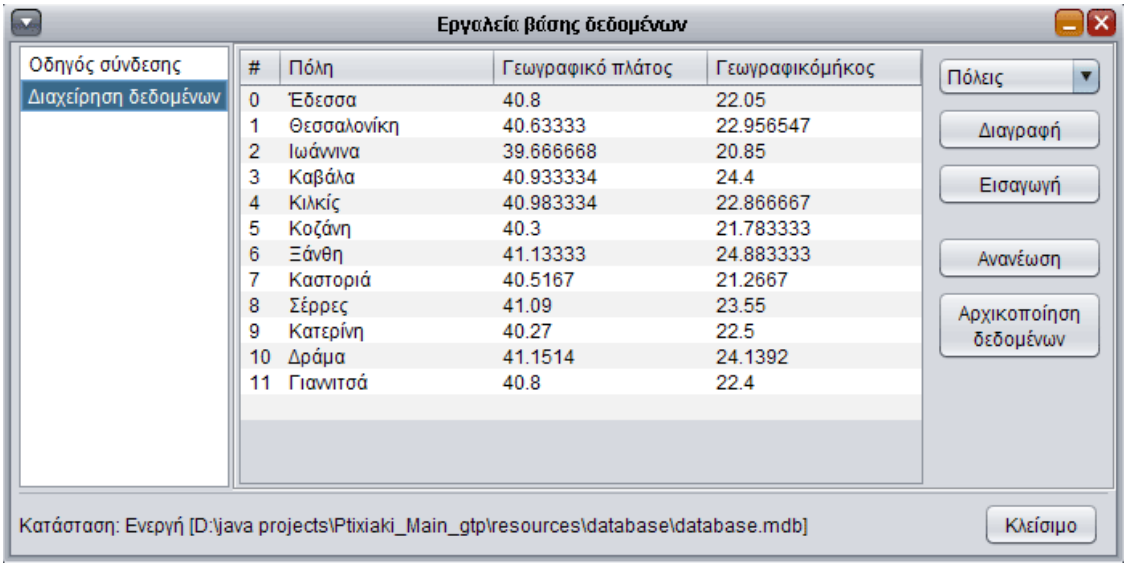
- Ρυθμίσεις βάσης
- Ρυθμίσεις χάρτη
- Παραμετροποίηση και εκτέλεση του αλγορίθμου

Α΄.1 Ρυθμίσεις βάσης

Μπορούμε να χρησιμοποιήσουμε ότι βάση θέλουμε αρκεί να έχει τους κατάλληλους πίνακες. Για την διασύνδεση του προγράμματος με τη βάση θα πρέπει να έχουμε και το κατάλληλο driver. Στην εφαρμογή μας προς το παρόν υπάρχουν drivers για MySQL, και PostgreSQL. Το παράθυρο με τις ρυθμίσεις ανοίγει από το μενού *Δεδομένα->Σύνδεση σε Βάση δεδομένων*. Από το τη λίστα επιλέγουμε το κατάλληλο driver για τη βάση μας και στη συνέχεια γράφουμε τη διαδρομή του αρχείου της βάσης στο επόμενο πλαίσιο κειμένου. Σε περίπτωση που η βάση έχει προστατευθεί με κωδικό, το επισημαίνουμε στα δυο επόμενα πλαίσια κειμένου. Όταν τελειώσουμε πατάμε το κουμπί σύνδεση. Αν η σύνδεση είναι επιτυχής κάτω στην μπάρα κατάστασης θα βγει το μύνημα επιτυχής σύνδεση, και η κατάσταση από ανενεργή θα γίνει ενεργή.

Για τη διαχείριση των δεδομένων της βάσης δε χρειάζεται να γνωρίζουμε τη χρήση του προγράμματος Microft Access ή των άλλων προγραμμάτων. Επιλέγοντας από τη λίστα την *Διαχείριση δεδομένων*, μας εμφανίζεται μια καρτέλα απ'όπου μπορούμε να κάνουμε όποιες αλλαγές, προσθήκες χρειαζόμαστε. Από την δεξιά λίστα επιλέγουμε σε πιο πίνακα θέλουμε να δουλέψουμε, (θα πρέπει φυσικά να έχουμε κάνει σύνδεση σε κάποια βάση για να μπορούμε να έχουμε πρόσβαση στα δεδομένα της) και στον πίνακα εμφανίζονται οι εγγραφές που υπάρχουν στη βάση. Για τη μεταβολή μιας εγγραφής χρησιμοποιούμε το ποντίκι ή το πληκτρολόγιο για να εισάγουμε τις νέες τιμές που θέλουμε. Για τη προσθήκη μιας νέας εγγραφής αρκεί να εισάγουμε τις νέες τιμές στην τελευτάια σειρά η οποία είναι κενή και έχοντας επιλεγμένο το τελευταίο κελί να πατήσουμε το κουμπί *Εισαγωγή* ή το πλήκτρο *enter*.

Από το κουμπί *Αρχικοποίηση Δεδομένων* φορτώνουμε τα δεδομένα της βάσης στη μνήμη, ώστε να χρησιμοποιηθούν από την εφαρμογή.



Σχήμα Α΄.1: Διαχείριση βάσης

Α΄.2 Ρυθμίσεις χάρτη

Ο χάρτης που χρησιμοποεί το πρόγραμμα μπορεί να δεχτεί εικόνες είτε από το δίκτυο από κάποιο map-server είτε από φάκελο του τοπικού δίσκου. Για να ρυθμίσουμε το χάρτη πρέπει να πάμε από το μενού *Εργαλεία* στης *Ρυθμίσεις* και από κει στην καρτέλα *Χάρτης*.

- Διεύθυνση: Η διεύθυνση στο δίκτυο όπου βρίσκονται οι εικόνες.

- Διαδρομή στο δίσκο: Ο φάκελος στον τοπικό δίσκο που έχουμε αποθηκεύσει τις εικόνες.
- Ζουμ: Ο μέγιστος βαθμός μεγέθυνσης που μπορεί να έχει ο χάρτης. Θα πρέπει να είναι θετικός και μεγαλύτερος ή ίσος του 4.
- Αντιστροφή: Όταν είναι επιλεγμένο τότε οι τιμές για το ζουμ αντιστρέφονται. Συνήθωσε είναι ανάλογα με τα ονόματα των εικόνων που έχουμε.
- Μέγεθος εικόνων: Το μέγεθος των εικόνων απο τις οποίες αποτελείται ο χάρτης. Θα πρέπει να έχουν ίσες διαστάσεις.
- Μορφη εικόνων: Ο τρόπος σχηματισμού των εικόνων από τις τρεις μεταβλητές z (zoom), x (x axis), y (y axis). Μπορούμε να φανταστούμε ένα χάρτη σαν μια πυραμίδα από εικόνες όπου στο επίπεδο 0 έχουμε μια εικόνα στο επίπεδο 1 τέσσερις στο επίπεδο 2 οχτώ κτλ. Με βάση αυτή τη μορφή και το που βρίσκονται οι εικόνες, θα σχηματίσουμε την τελική διαδρομή της εικόνας.

Οι προεπιλεγμένες ρυθμίσεις που χρησιμοποιούνται είναι:

- Διεύθυνση: http://tile.openstreetmap.org/
- Ζουμ: 17
- Αντιστροφή: όχι
- Μέγεθος εικόνων: 256
- Μορφή εικόνων: z/x/y.png

Α΄.3 Παραμετροποίηση και εκτέλεση του αλγορίθμου

Η διαδικασία εκτέλεσης του αλγορίθμου ακολουθεί τα παρακάτω βήματα:

- Επιλογή πόλεων και αποστάσων που θα συμμετέχουν στη λύση
- Ρύθμιση παραμέτρων
- Επισκόπηση αποτελεσμάτων

Α΄.3.1 Επιλογή πόλεων και αποστάσων

Η επιλογή των πόλεων μπορεί να γίνει με τρείς τρόπους. Ο πρώτος τρόπος είναι να συνδεθούμε με μια βάση και να φορτώσουμε τις πόλεις που είναι αποθηκευμένες σε αυτή. Ο δεύτερος είναι επιλέγοντας με το ποντίκι πάνω στο χάρτη το σημείο που θέλουμε και ο τρίτος, αν ξέρουμε τις συντεταγμένες, από το κουμπί προσθήκη που βρίσκεται δίπλα στον πίνακα των πόλεων. Κάθε γραμμή στο πίνακα των πόλεων αντιπροσωπεύει ένα σημείο στο χάρτη. Για αλλαγή του ονόματος ή των συντεταγμένων ενός σημείου μπορούμε να κάνουμε διπλό κλικ πάνω στην αντίστοιχη γραμμή στον πίνακα και να κάνουμε τις επιθυμητές αλλαγές. Όλα τα σημεία που θα λάβουν μέρος στο σχηματισμό της τελικής λύσης έχουν επιλεγμένο το κουτάκι στη στήλη εμφάνιση. Εάν θέλουμε να βγάλουμε ένα σημείο προσωρινά τότε απλά αποεπιλέγουμε τη στήλη εμφάνιση. Αν πάλι θέλουμε να διαγράψουμε τελέιως το σημείο από τον πίνακα και το χάρτη τοτε έχοντας επιλεγμένη την αντίστοιχη γραμμή πατάμε το κουμπί *Διαγραφή(-)*. Οι αποστάσεις μεταξύ των πόλεων υπολογίζονται είτε αυτόματα επιλέγοντας το κουμπί “ράδιο” *Γεωγραφικές*, είτε με βάση αυτές που βρίσκονται στη βάση δεδομένων που έχουμε φορτώσει. Φυσικά υπάρχει η δυνατότητα να ορίζουμε κάποια δικιά μας τιμή σε μια απόσταση κάνοντας διπλό κλικ στο κατάλληλο κελί στον πίνακα των αποστάσεων.

Α΄.3.2 Ρύθμιση παραμέτρων

Η ρύθμιση των παραμέτρων του αλγορίθμου γίνεται από το παράθυρο διαλόγου από το κουμπί *Παράμετροι*. Επειδή η επιλογή των σωστών τιμών μερικών παραμέτρων είναι αρκετά δύσκολη υπάρχουν 2 προεπιλεγμένες παραμετροποιήσεις στις οποίες αναφερθήκαμε στο κεφάλαιο 4. Όπου οι τιμές είναι -1 σημαίνει ότι οι τιμές των παραμέτρων εξαρτώνται από άλλες μεταβλητές και η πραγματική τους τιμή θα μπει αφού ξεκινήσει η εκτέλεση του αλγορίθμου. Επιλέγοντας το κουμπί “ράδιο” *Τιμές* έχουμε τη δυνατότητα να ορίσουμε δικές μας τιμές. Οι *Συνθήκες τερματισμού* αναφέρονται στα κριτήρια με βάση τα οποία θα τελειώσει η εκτέλεση του αλγορίθμου, και το *Πλήθος εκτελέσεων* στο πόσες διαφορετικές φορές θα εκτελεστεί ο αλγόριθμος. Συνήθως παίρνει την τμή 1 εκτός και αν θέλουμε να βρούμε κάποιο μέσο όρο ή να κάνουμε πολλές προσομοιώσεις ταυτόχρονα. Με το κουμπί ελέγχου *Όλοι οι συνδιασμοί* ο αλγόριθμος τρέχει με όλους τους δυνατούς συνδιασμούς των παραμέτρων. Οι συνδιασμοί αυτοί είναι προκαθορισμένοι εξαρχής. Στο παράρτημα Β υπάρχουν οδηγίες για τη δημιουργία νέων σετ παραμέτρων.

Α΄.3.3 Καταστάσεις εκτέλεσης

Όταν τελειώσουμε με την επιλογή των πόλεων και τη ρύθμιση των παραμέτρων μπορούμε να αρχίσουμε την εκτέλεση του αλγορίθμου. Πατώντας το κουμπί *Έναρξη* μας εμφανίζεται ένα παράθυρο όπου πρέπει να επιλέξουμε την κατάσταση εκτέλεσης του αλγορίθμου. Εαν στις παραμέτρους που ορίσαμε παραπάνω έχουμε βάλει παραπάνω από μία επαναλήψεις εκτελέσων, όλες θα τρέξουν με την ίδια κατάσταση που επιλέξαμε στην αρχή.

Α΄.3.4 Εκτέλεση

Κατά τη διάρκεια της εκτέλεσης ανάλογα με την κατάσταση που έχουμε επιλέξει θα βλέπουμε και διαφορετικά αντικείμενα να ενημερώνονται. Στη γρήγορη κατάσταση ο αλγόριθμος τρέχει στη μέγιστη ταχύτητα δείχνοντας μόνο την τελική λύση στο χάρτη. Στην κανονική κατάσταση μπορούμε να επιλέξουμε την ταχύτητα εκτέλεσης από την μπάρα κύλισης με το 0 να ειναι το πιο γρήγορο και το 100 το πιο αργό. Επίσης σε κάθε εποχή ανανεώνεται και ο χάρτης δείχνοντας την τρέχουσα λύση. Τέλος στην πλήρη κατάσταση έχουμε διαγραμματική εμφάνιση της ενέργειας του δικτύου σε κάθε εποχή, εμφάνιση διαφόρων πληροφοριών που καθορίζουμε εμείς, καθώς και αναπαράσταση του πίνακα των βαρών των νευρώνων.

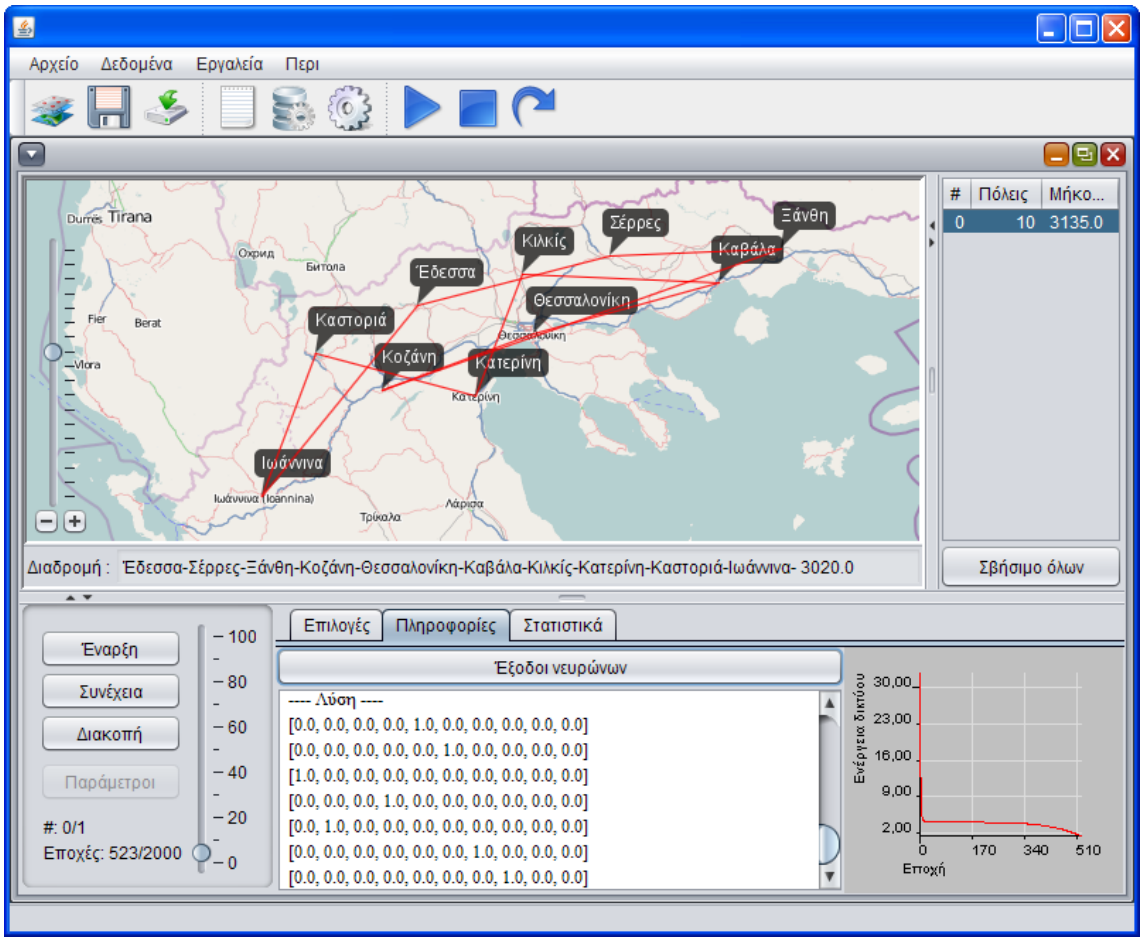
Α΄.3.5 Αποθήκευση Αποτελεσμάτων

Για την αποθήκευση των αποτελεσμάτων θα πρέπει ο πίνακας αποτελεσμάτων που βρίσκεται στα δεξιά του χάρτη να μην είναι άδειος. Επιλέγοντας μια γραμμή μπορούμε από το μενού *Αρχείο->Αποθήκευση παραμετροποίησης* να την αποθηκεύσουμε με κατάληξη xml. Το πρόγραμμα αυτόματα στο όνομα που δώσαμε προσθέτει και το αποτέλεσμα της λύσης που είναι το μήκος της διαδρομής.

Α΄.3.6 Άνοιγμα και Εκτέλεση από αρχείο

Το πρόγραμμα έχει τη δυνατοτητα να τρέξει τον αλγόριθμο από τα αρχεία που έχουμε αποθηκεύσει τα οποία περιέχουν όλες τις πληροφορίες ώστε κάθε φορά το αποτέλεσμα να είναι το ίδιο. Επίσης μπορεί να ανοίξει αυτά τα αρχεία για επισκόπηση ή και

αλλαγή, από τον ενσωματωμένο κειμενογράφο. Από το μενού *Αρχείο->Άνοιγμα παραμετροποίησης*, επιλέγουμε πώς θέλουμε να ανοίξουμε το αρχείο. Από το *Άνοιγμα για εκτέλεση*, εαν ο χάρτης δεν είναι ανοιχτός, ανοίγει με τις πόλεις τοποθετημένες στο χάρτη. Αν ο χάρτης είναι ανοιχτός τότε οι πόλεις που έχουμε αντικαθίστανται με αυτές από το αρχείο.



Σχήμα Α΄.2: Παράδειγμα εκτέλεσης

Παράρτημα Β΄

Παραμετροποιήσεις

Κατά την επιλογή των παραμέτρων έχουμε τη δυνατότητα να επιλέξουμε να τρέξει ο αλγόριθμος με πολλές παραμετροποιήσεις. Αυτές θα πρέπει να έχουν πρώτα δημιουργηθεί και να βρίσκονται στο φάκελο *configurations*. Για τις ανάγκες των πειραμάτων που κάναμε δημιουργήσαμε 16 αρχεία διαφορετικών παραμετροποιήσεων. Ένα παράδειγμα τέτοιου αρχείου με όλες τις παραμέτρους είναι το παρακάτω.

```
import Gui.comp.NetworkParameters;
import HPackage.configs.Aconf;
import HPackage.configs.Iconf;

public class conf extends Aconf implements Iconf{

        public conf_1() {
                super();
                netparam.setA(-1);
                netparam.setB(-1);
                netparam.setC(1);
                netparam.setD(-1);
                netparam.setI(-1);
                netparam.setDT(0.01);
                netparam.setUo(0.1);
                netparam.setSFALMA(0.000000001);
                netparam.setEpanalipsi_sfalmatos(20);
                netparam.setPlithos_ekteleseon(10);
                netparam.setEpoxes(2000);
                netparam.setTipos_diktiou(NetworkParameters.DIAKRITO);
                netparam.setTipos_enimerosis_baron(NetworkParameters.ASIGXRONOS);
                netparam.setSchema(NetworkParameters.DEFAULT_1);
        }
```

```
        public NetworkParameters getNetParam() {
                return netparam;
        }
}
```

Listing Β΄.1: Παράδειγμα αρχείου παραμετροποίησης

Οι τιμές που έχουμε παραπάνω είναι οι προεπιλεγμένες. Σε περίπτωση που δε θέλουμε να μεταβάλλουμε μια τιμή απλά σβήνουμε ή σχολιάζουμε την αντίστοιχη γραμμή. Το αρχείο μας θα πρέπει να κληρονομεί την αφηρημένη κλάση Aconf και να υλοποιεί τη διεπαφή Iconf. Για την εισαγωγή των κλάσεων *NetworkParameters*, *Aconf*, *Iconf* θα πρέπει κατά τη μεταγλώττιση να ορίσουμε στο classpath το αρχείο της εφαρμογής.

```
public interface Iconf {
        public NetworkParameters getNetParam();
}
```

Listing Β΄.2: Διεπαφή Iconf

Βιβλιογραφία

[weba] *Artificial neural network*. http://en.wikipedia.org/wiki/Artificial_neural_network.

[webb] *Biological neural network*. http://en.wikipedia.org/wiki/Biological_neural_network.

[webc] *Travelling salesman problem*. http://en.wikipedia.org/wiki/Traveling_salesman_problem.

[Κινγκ 1998] Κινγκ, . . (1998). *Υπολογιστική νοημοσύνη στον έλεγχο συστημάτων*. Αθήνα: Π. Τραυλός.

[Αργυράκης 2001] Αργυράκης, . (2001). *Νευρωνικά Δίκτυα και Εφαρμογές - Τόμος Β'*. Πάτρα: Ελληνικό Ανοιχτό Πανεπιστήμιο.

[Διαμαντάρας 2007] Διαμαντάρας, . (2007). *Τεχνητά Νευρωνικά Δίκτυα*. Αθήνα: Κλειδάριθμος.

[Λυκοθανάσης 2007] Λυκοθανάσης, . (2007). *Συμπληρωματικές σημειώσεις για το μάθημα "Υπολογιστική Νοημοσύνη 1"*. ., Πανεπιστήμιο Πατρών Πολυτεχνική σχολή.

[Abramson . 1998] Abramson, D., Smith, K., Logothetis, P., und Duke, D. (1998). *FPGA Based Implementation of a Hopfield Neural Network for Solving Constraint Satisfaction Problems*. . *EUROMICRO '98: Proceedings of the 24th Conference on EUROMICRO*, . 20688, Washington, DC, USA: IEEE Computer Society.

[Dorigo . Gambardella 1997] Dorigo, M. und Gambardella, L. M. (1997). *Ant Colonies for the Traveling Salesman Problem*.

[Feng . Douligeris 2000] Feng, G. und Douligeris, C. (2000). *Using Hopfield Networks to Solve Traveling Salesman Problems Based on Stable State Analysis Technique*. Neural Networks, IEEE - INNS - ENNS International Joint Conference on, 6, 6521.

[Grätzer 1996] Grätzer, G. (1996). *Math into LaTeX : an introduction to LaTeX and AMS-LaTeX*: Birkhäuser Verlag.

[Hagan . 2002] Hagan, M. T., Demuth, H. B., und Beale, M. H. (2002). *Neural Network Design*: Martin Hagan.

[Hahsler . Hornik 2009] Hahsler, M. und Hornik, K. (2009). *Introduction to TSP - Infrastructure for the Traveling Salesperson Problem*.

[Hopfield . Tank 1985] Hopfield, J. und Tank, D. (1985). *Neural Computation of Decisions in Optimization Problems*. Biological Cybernetics, 52, 141–152.

[Hopfield] Hopfield, J. J. *Hopfield network*. `http://www.scholarpedia.org/article/Hopfield_network`.

[Huajin . 2007] Huajin, T., Chen, T. K., und Zhang, Y. (2007). *Neural Networks: Computational Models and Applications*: Springer.

[Johnson . Mcgeoch 1997] Johnson, D. S. und Mcgeoch, L. A. (1997). *The Traveling Salesman Problem: A Case Study in Local Optimization*.

[Mandziuk 1996] Mandziuk, J. (1996). *Solving the Travelling Salesman Problem with a Hopfield - type neural network*. Demonstratio Mathematica, 29, 219–231.

[Matsuda 2002] Matsuda, S. (2002). *"Optimal" Neural Representation of Higher Order for Traveling Salesman Problems*. Electronics and Communications in Japan (Part II: Electronics), 85(9), 32–42.

[Nagórny 2007] Nagórny, A. K. . Z. (2007). *Modified Hopfield Neural Network for Travelling Salesman Problem*. . *Proc. of the 2nd Conference Tools of Information Technology*, Rzeszów.

[Oetiker 2008] Oetiker, T. (2008). *The Not So Short Introduction to LaTeX2ε*.

[Rojas 1996] Rojas, R. (1996). *Neural Networks - A Systematic Introduction*, . 337–372. New York: Springer-Verlag, Fourth .

[Saiko 2005] Saiko, D. (2005). *Traveling Salesman Problem Java Genetic Algorithm Solution*.

[Schwaiger . 1999] Schwaiger, R., August, M., Gerald, W., und Markus, S. (1999). *Applications of Hopfield Networks*. ., University of Salzburg, Institute for Computer Science.

[Talavan . Yanez 2002] Talavan, P. M. und Yanez, J. (2002). *Parameter setting of the Hopfield network applied to TSP*. Neural Networks, 15(3), 363 – 373.

[Tan . 2005] Tan, K., Tang, H., und Ge, S. (2005). *On parameter settings of Hopfield networks applied to traveling salesman problems*. IEEE Transactions on Circuits and Systems I: Regular Papers, 52(5), 994– 1002.

[Tsang . Voudouris 1998] Tsang, E. und Voudouris, C. (1998). *Constraint Satisfaction in Discrete Optimisation*. . *Proceedings, UNICOM Seminar on Constraint Satisfaction and Discrete Optimisation*.

[Veness] Veness, C. *Vincenty formula for distance between two Latitude/Longitude points*. `http://www.movable-type.co.uk/scripts/latlong-vincenty.html`.

[Yang 2004] Yang, H. W. . Y. (2004). *Application of continuous Hopfield network to solve the TSP*. . *ICARCV*, . 2258–2263.

[Zeng . Martinez 1999] Zeng, X. und Martinez, T. R. (1999). *A New Activation Function in the Hopfield Network for Solving Optimization Problems*. . *Fourth International Conference on Artificial Neural Networks and Genetic Algorithms (ICANNGA'99*.

[Zhao . 2003] Zhao, X., Wang, X., Tang, Z., Tamura, H., Ishii, M., und Zeng, G. (2003). *A fast adaptive algorithm for Hopfield neural network*. SICE 2003 Annual Conference, 1, 638– 642.