ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΤΟΜΕΑΣ ΕΠΙΚΟΙΝΩΝΙΩΝ, ΗΛΕΚΤΡΟΝΙΚΗΣ ΚΑΙ ΣΥΣΤΗΜΑΤΩΝ
ΠΛΗΡΟΦΟΡΙΚΗΣ

Ευφυής Προσδιορισμός Βέλτιστων Διαδρομών βάσει Μεθόδων Μηχανικής Μάθησης

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Γεώργιος Β. Σομπόνης

Επιβλέπων : Μιχαήλ Ε. Θεολόγου
Καθηγητής Ε.Μ.Π.

Αθήνα, Ιούλιος 2012

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΤΟΜΕΑΣ ΕΠΙΚΟΙΝΩΝΙΩΝ, ΗΛΕΚΤΡΟΝΙΚΗΣ ΚΑΙ ΣΥΣΤΗΜΑΤΩΝ
ΠΛΗΡΟΦΟΡΙΚΗΣ

Ευφυής Προσδιορισμός Βέλτιστων Διαδρομών βάσει Μεθόδων Μηχανικής Μάθησης

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Γεώργιος Β. Σομπόνης

Επιβλέπων : Μιχαήλ Ε. Θεολόγου
Καθηγητής Ε.Μ.Π.

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή τη 12η Ιουλίου 2012.

.....	.....	.....
Μιχαήλ Ε. Θεολόγου	Ευστάθιος Δ. Συκάς	Γεώργιος Ι. Στασινόπουλος
Καθηγητής Ε.Μ.Π.	Καθηγητής Ε.Μ.Π.	Καθηγητής Ε.Μ.Π.

Αθήνα, Ιούλιος 2012

.....
Γεώργιος Β. Σομπόνης

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

Πρόλογος

Η παρούσα διπλωματική εργασία εκπονήθηκε στα πλαίσια της ερευνητικής δραστηριότητας του εργαστήριου Δικτύων Κινητών και Προσωπικών Επικοινωνιών της σχολής Ηλεκτρολόγων Μηχανικών και Μηχανικών Η/Υ του Εθνικού Μετσόβιου Πολυτεχνείου και πραγματεύεται τη χρήση των Νευρωνικών Δικτύων, και συγκεκριμένα του δικτύου Hopfield, για τη δρομολόγηση οχημάτων σε οδικά δίκτυα.

Στις επόμενες σελίδες παρουσιάζονται και αναλύονται τόσο οι δημοφιλέστεροι υπολογιστικοί αλγόριθμοι δρομολόγησης, όσο και τα Νευρωνικά Δίκτυα, με ιδιαίτερη έμφαση στο δίκτυο Hopfield, μια τροποποιημένη εκδοχή του οποίου χρησιμοποιείται, στη συνέχεια, για την εύρεση της βέλτιστης διαδρομής σε οδικά δίκτυα. Η εργασία ολοκληρώνεται με την έκθεση των αποτελεσμάτων και των συμπερασμάτων, που προέκυψαν, ύστερα από πειραματικές μελέτες. Για την πραγματοποίηση αυτών των μελετών οι εξεταζόμενοι αλγόριθμοι υλοποιήθηκαν στο περιβάλλον προγραμματισμού MATLAB.

Κλείνοντας τον σύντομο αυτό πρόλογο θα ήθελα να ευχαριστήσω θερμά τον επιβλέποντα καθηγητή κ. Μιχαήλ Θεολόγου για την ευκαιρία, που μου προσέφερε, να εργαστώ πάνω σε ένα σύγχρονο και ενδιαφέρον θέμα καθώς και τον υποψήφιο διδάκτορα κ. Μιχαήλ Μασίκο για την αμέριστη βοήθεια και την άψογη συνεργασία.

Γεώργιος Β. Σομπόνης
Αθήνα, Ιούλιος 2012

Περίληψη

Η παρούσα διπλωματική εργασία ασχολείται με τη δρομολόγηση οχημάτων σε οδικά δίκτυα χρησιμοποιώντας το Νευρωνικό Δίκτυο Hopfield. Για λόγους σαφήνειας και πληρότητας αρχικά περιγράφονται οι δημοφιλέστεροι «συμβατικοί» αλγόριθμοι δρομολόγησης. Στη συνέχεια παρουσιάζονται οι βασικότερες αρχές των Νευρωνικών Δικτύων ούτως ώστε να καταστεί κατανοητή η λειτουργία του δικτύου Hopfield, η ανάλυση της οποίας έπεται. Έχοντας, πλέον, τα απαραίτητα γνωστικά εφόδια, η διπλωματική προχωράει στην επεξήγηση του προτεινόμενου δικτύου Hopfield με τροποποιημένη συνάρτηση ενέργειας, που καλείται να επιλύσει το πρόβλημα της δρομολόγησης μεταξύ δύο δοσμένων κόμβων σε οδικά δίκτυα. Η εφαρμογή του νευρωνικού αυτού δικτύου σε ένα τμήμα του οδικού δικτύου της Αθήνας και η αντίστοιχη προσομοίωση προσφέρουν σημαντικά αποτελέσματα, τα οποία επεξηγούνται και αναλύονται. Στα πλαίσια της πειραματικής ανάλυσης υλοποιήθηκε στο MATLAB μία εφαρμογή δρομολόγησης, που υπολογίζει τις βέλτιστες ή σχεδόν βέλτιστες διαδρομές μεταξύ δύο κόμβων κάνοντας χρήση είτε του προτεινόμενου δικτύου Hopfield, είτε του αλγόριθμου Dijkstra.

Λέξεις κλειδιά

Δίκτυο Hopfield, νευρωνικό δίκτυο, μηχανική μάθηση, αλγόριθμος δρομολόγησης, οδικό δίκτυο, βέλτιστη διαδρομή, συνάρτηση ενέργειας, αλγόριθμος Dijkstra

Abstract

This diploma thesis deals with the vehicle routing problem. For clarity and completeness purposes the most popular conventional routing algorithms are initially described. The basic principles of Neural Networks are then presented in order to make intelligible the operation of Hopfield network, whose analysis follows. Having now the necessary cognitive skills, the thesis moves on explaining the proposed Hopfield network with a modified energy function, that is called to solve the routing problem between two given nodes in road networks. The implementation of the proposed network into a section of the road network of the city of Athens and the corresponding simulation provide significant results, which are explained and analyzed. The experimental analysis is performed with a MATLAB application that calculates the optimal or near optimal path between two nodes using either the proposed Hopfield network or Dijkstra's Algorithm.

Keywords

Hopfield network, neural network, machine learning, routing algorithm, road network, optimal path, energy function, Dijkstra's algorithm

Πίνακας περιεχομένων

1 *Εισαγωγή*

Ο σχεδιασμός διαδρομής ή η δρομολόγηση είναι μια διαδικασία, που αποτελεί θεμελιώδες πρόβλημα στον τομέα της πλοήγησης οχημάτων και μπορεί να ταξινομηθεί σε δρομολόγηση πολλαπλών οχημάτων (*multivehicle*), η οποία περιλαμβάνει τον υπολογισμό διαδρομών πολλαπλών προορισμών (*multidestination*) για ό λα τα οχήματα σε ένα συγκεκριμένο οδικό δίκτυο και σε δρομολόγηση ενός οχήματος (*singlevehicle*), η οποία σχεδιάζει μια ενιαία διαδρομή για ένα και μόνο όχημα, σύμφωνα με την τρέχουσα θέση αυτού και έναν καθορισμένο προορισμό [3].

Στην επιστήμη των υπολογιστών η εύρεση μίας διαδρομής από το σημείο Α στο σημείο Β συχνά αναφέρεται ως πρόβλημα συντομότερης διαδρομής (*shortest path problem - SP*). Για την επίλυση του προβλήματος αυτού έχουν αναπτυχθεί πολλοί αλγόριθμοι, οι οποίοι αντιμετωπίζουν καταστάσεις ανάλογες με τον σχεδιασμό διαδρομής ενός ή πολλαπλών οχημάτων.

Η κατασκευή φορητών συσκευών, για την πλοήγηση οχημάτων, δημιούργησε την ανάγκη εύρεσης αποδοτικότερων μεθόδων για τον σχεδιασμό της διαδρομής σε πραγματικό χρόνο, με κύριο γνώμονα την άμεση απόκριση και το χαμηλό κόστος. Στην πράξη χρησιμοποιούνται κυρίως υπολογιστικοί αλγόριθμοι, σε διάφορες παραλλαγές, ούτως ώστε να ικανοποιούν τις παραπάνω απαιτήσεις. Ωστόσο, το ερευνητικό ενδιαφέρον έχει πλέον στραφεί προς την ευφυή πλοήγηση με εφαρμογές Τεχνητής Νοημοσύνης και Μηχανικής Μάθησης.

Στην παρούσα διπλωματική εργασία προτείνεται μία ευφυής μέθοδος δρομολόγησης, η οποία βασίζεται σε ένα νευρωνικό δίκτυο Hopfield με τροποποιημένη συνάρτηση ενέργειας.

1.1 Διάρθρωση

Η διπλωματική εργασία αναπτύσσεται σε οκτώ κεφάλαια. Τα τέσσερα πρώτα, συνιστούν το θεωρητικό μέρος και προσφέρουν το απαραίτητο υπόβαθρο για την κατανόηση των υπολοίπων, ενώ τα επόμενα τέσσερα αποτελούν το πρακτικό τμήμα.

Στο Κεφάλαιο 2 γίνεται μια συνοπτική παρουσίαση της απαραίτητης ορολογίας και κατόπιν περιγράφονται οι κυριότεροι «συμβατικοί» υπολογιστικοί αλγόριθμοι, οι οποίοι χρησιμοποιούνται στην πλοήγηση των οχημάτων.

Το Κεφάλαιο 3 αποτελεί εισαγωγή σ τα Νε υρωνικά Δίκτυα αρχίζοντας από μια σύντομη ιστορική αναδρομή και συνεχίζοντας με την ανάλυση των μοντέλων του βιολογικού και τεχνητού νευρώνα, τις αρχιτεκτονικές των νευρωνικών δικτύων και τις τεχνικές μάθησης. Το κεφάλαιο ολοκληρώνεται με μία επιγραμματική αναφορά στις εφαρμογές των νευρωνικών δικτύων.

Στο Κεφάλαιο 4 παρουσιάζονται εκτενέστερα τόσο το διακριτό όσο και το αναλογικό δίκτυο Hopfield, το οποίο θα χρησιμοποιήσουμε στην παρούσα εργασία. Η παρουσίαση αρχίζει με μία επισκόπηση της αρχιτεκτονικής, ενώ κατόπιν εμβαθύνει στη δυναμική συμπεριφορά του. Στη συνέχεια, αναλύεται η υλοποίηση του εν λόγω δικτύου με αναλογικά κυκλώματα και κλείνοντας αναφέρονται ορισμένα παραδείγματα προβλημάτων, στην επίλυση των οποίων είναι ιδιαίτερα αποδοτικό.

Στο Κεφάλαιο 5 προτείνεται ένα δίκτυο Hopfield με τροποποιημένη συνάρτηση ενέργειας για τη δρομολόγηση οχημάτων σε οδικό δίκτυο.

Στο Κ εφάλαιο 6 παρουσιάζεται η εφαρμογή, που υλοποιεί το προτεινόμενο δίκτυο Hopfield και παρέχει μία χρηστική γραφική διεπαφή (Graphical User Interface - GUI), ούτως ώστε να μπορεί ο χρήστης να υπολογίζει τις βέλτιστες διαδρομές μεταξύ δύο κόμβων στο οδικό δίκτυο της Αθήνας.

Το Κεφάλαιο 7 αποτελεί την πειραματική μελέτη του προτεινόμενου δικτύου Hopfield, το οποίο χρησιμοποιείται για τον υπολογισμό διαδρομών σε ένα τμήμα του οδικού δικτύου της Αθήνας και στη συνέχεια εκθέτονται τα αποτελέσματα.

Στο Κ εφάλαιο 8 συνοψίζονται τα συμπεράσματα, τα οποία προέκυψαν από την ανάλυση, υλοποίηση και προσομοίωση της προτεινόμενης μεθόδου δρομολόγησης σε οδικά δίκτυα, ενώ σκιαγραφούνται οι μελλοντικές εξελίξεις στον τομέα αυτό.

Τέλος, στο Παράρτημα παρατίθεται ο κώδικας της εφαρμογής, που παρουσιάστηκε στο Κεφάλαιο 6, εμπλουτισμένος με αναλυτικά σχόλια για είναι πλήρως κατανοητός από τον αναγνώστη, καθώς και η διαδικασία, η οποία πρέπει να ακολουθηθεί, για να χρησιμοποιηθεί η εν λόγω εφαρμογή σε άλλα οδικά δίκτυα.

2 *Αλγόριθμοι δρομολόγησης*

Για την εύρεση της διαδρομής μεταξύ δύο σημείων υπάρχουν πολλοί αλγόριθμοι, που κάνουν χρήση υπολογιστικών μεθόδων και βρίσκουν εφαρμογή σε δίκτυα επικοινωνιών, δίκτυα υπολογιστών, οδικά δίκτυα κ.τ.λ.. Στο κεφάλαιο αυτό παρουσιάζουμε και αναλύουμε τους σημαντικότερους αλγόριθμους δρομολόγησης, δίνοντας ιδιαίτερη έμφαση σε εκείνους, που χρησιμοποιούνται για τον σχεδιασμό διαδρομής ενός οχήματος και την πλοήγηση οχημάτων σε οδικά δίκτυα. Η δρομολόγηση πολλαπλών οχημάτων είναι πιο σύνθετη διαδικασία και δεν θα αναλυθεί περαιτέρω.

2.1 Βασικές έννοιες

Η ουσιαστικότερη κατανόηση του προβλήματος δρομολόγησης και των τεχνικών επίλυσης του, προϋποθέτει τη γνώση ορισμένων εννοιών. Για λόγους πληρότητας παραθέτουμε τις βασικότερες:

Γράφος: Ένας γράφος $\mathrm{G} = (\mathrm{V}, \mathrm{E})$ ορίζεται από ένα μη - κενό, πεπερασμένο σύνολο κορυφών ή κόμβων V και ένα σύνολο ακμών ή τόξων ή συνδέσμων $E \leq \{(u, v)\ \text{όπου}\ u, v \in V\ \text{και}\ u \neq v\}$. Θεωρούμε ότι ο γράφος αποτελείται από n κόμβους και s συνδέσμους. Κάθε κόμβος ή σύνδεσμος ενός γράφου έχει ένα μοναδικό όνομα, που ονομάζεται επιγραφή ή ετικέτα (*label*). Κάθε σύνδεσμος

ορίζεται από ένα ζεύγος κόμβων, ενώ αν ο γράφος είναι κατευθυνόμενος, τότε το ζεύγος αυτό είναι διατεταγμένο.

Δίκτυο: Ως δίκτυο $N = (G, w)$ ορίζεται ένας κατευθυνόμενος γράφος μαζί με μία πραγματική συνάρτηση βάρους $w: E \to R$, η οποία αντιστοιχεί σε κάθε σύνδεσμό του από έναν αριθμό, που ονομάζεται βάρος συνδέσμου. Σε ένα οδικό δίκτυο κάθε κόμβος του γράφου είναι μια διασταύρωση ή ένα αδιέξοδο και κάθε σύνδεσμος είναι ένας δρόμος. Το βάρος κάθε συνδέσμου αποτελεί το κόστος ταξιδιού (*travel cost*) και εξαρτάται από πολλούς παράγοντες όπως η απόσταση, ο χρόνος, η ταχύτητα, ο αριθμός των στροφών και των φαναριών, και η κίνηση.

Μονοπάτι: Ως μονοπάτι από τον κόμβο αφετηρίας *s (source)* προς τον κόμβο προορισμού *d (destination)* στον γράφο *G* ορίζεται μία πεπερασμένη α κολουθία $p = (s = n_1, n_2, \dots, n_k = d)$ κόμβων του *G* για τις οποίες ισχύει $(n_i, n_{i+1}) \in E \; για \; i = 1,2, \dots, k-1, όπου \; \kappa > 0$. Αν κάθε κόμβος εμφανίζεται το πολύ μία φορά στο *p* το μονοπάτι λέγεται στοιχειώδες. Αν ο κόμβος *s* ταυτίζεται με τον κόμβο *d* το μονοπάτι ονομάζεται κύκλος. Ως εκ τούτου, για οποιουσδήποτε κόμβους *s, d* υπάρχει ένα μονοπάτι στον *G* αν και μόνο αν υπάρχει ένα στοιχειώδες μονοπάτι από τον *s* στον *d* στον *G*.

Βάρος μονοπατιού: Το βάρος ή το κόστος του μονοπατιού *p* ισούται με το άθροισμα των βαρών των συνδέσμων, που το αποτελούν :

$$w(p) = \sum_{i=1}^{k-1} w(n_i, n_{i+1}) \qquad (2.1)$$

Συντομότερο μονοπάτι: Ως συντομότερο μονοπάτι από τον *s* στον *d* ορίζεται ένα μονοπάτι *p* μεταξύ των δύο κόμβων με βάρος $w(s, d)$, που δίνεται από τη σχέση:

$$w(s,d) = \begin{cases} \min[w(p): s \to d], & αν \; υπάρχει \; το \; μονοπάτι \; από \; τον \; s \; στον \; d \\ \infty, & διαφορετικά \end{cases} \qquad (2.2)$$

Άμεση συνέπεια του παραπάνω ορισμού είναι ότι υπάρχει συντομότερο μονοπάτι από τον *s* στον *d* αν και μόνο αν υπάρχει στοιχειώδες μονοπάτι από τον *s* στον *d* [1].

Μπορούμε, πλέον, να διατυπώσουμε με σαφήνεια το πρόβλημα, στο οποίο επικεντρώνεται η παρούσα διπλωματική εργασία:

"Δεδομένου ενός δικτύου $N = (G, w)$ να βρεθεί το συντομότερο μονοπάτι από τον κόμβο αφετηρίας s προς τον κόμβο προορισμού d."

Το πρόβλημα αυτό αποτελεί υποπερίπτωση του παρακάτω προβλήματος:

"Δεδομένου ενός δικτύου $N = (G, w)$ να βρεθεί το συντομότερο μονοπάτι από τον κόμβο αφετηρίας s προς κάθε άλλο κόμβο του N."

Τέλος, για λόγους πληρότητας αναφέρουμε και τον ορισμό του γενικότερου προβλήματος:

"Δεδομένου ενός δικτύου $N = (G, w)$ να βρεθεί το συντομότερο μονοπάτι μεταξύ κάθε ζεύγους κόμβων s, d."

Κατά την επίλυση του προβλήματος εύρεσης της συντομότερης διαδρομής σε ένα οδικό δίκτυο μια συνάρτηση αξιολόγησης καλείται να ελαχιστοποιήσει το κόστος (*travel cost*) ενός μονοπατιού p μεταξύ του κόμβου αφετηρίας και του κόμβου προορισμού. Μερικοί οδηγοί μπορεί να προτιμούν τη μικρότερη απόσταση, ενώ άλλοι μπορεί να προτιμούν τον μικρότερο χρόνο ταξιδιού κ.τ.λ.. Ως εκ τούτου, η συνάρτηση αξιολόγησης, που έχει επιλεχθεί για να ελαχιστοποιήσει το κόστος αυτό, εξαρτάται από τον σχεδιασμό του συστήματος και τις προτιμήσεις του χρήστη.

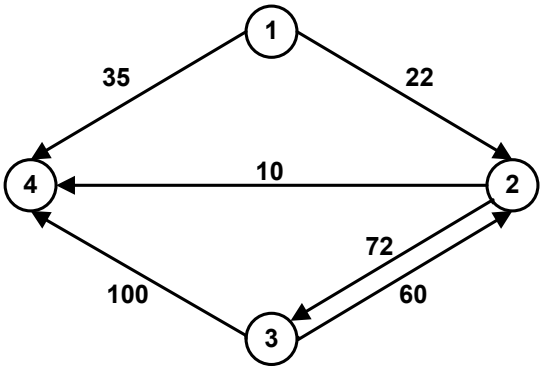
2.2 Αλγόριθμοι συντομότερης διαδρομής

Οι συγκεκριμένοι αλγόριθμοι υπολογίζουν όλα τα πιθανά μονοπάτια μεταξύ δύο κόμβων και επιλέγουν το συντομότερο. Στην παράγραφο αυτή αρχικά θα μελετήσουμε τον δημοφιλέστερο αλγόριθμο υπολογισμού της συντομότερης διαδρομής και κατόπιν θα ασχοληθούμε με τροποποιήσεις αυτού.

2.2.1 Ο αλγόριθμος Dijkstra

Ένας από τους πιο αντιπροσωπευτικούς αλγόριθμους της κατηγορίας αυτής είναι ο αλγόριθμος Dijkstra [2], ο οποίος κάνει χρήση μίας τεχνικής, που απαντάται συχνά σε προβλήματα βελτιστοποίησης. Με βάση την τεχνική αυτή ο αλγόριθμος σε κάθε βήμα

λαμβάνει τις βέλτιστες αποφάσεις σε τοπικό επίπεδο με την ελπίδα πως αυτές οι επιλογές θα οδηγήσουν σε μια συνολικά βέλτιστη λύση. Με δοσμένο έναν σταθμισμένο και κατευθυνόμενο γράφο, παρόμοιο με αυτόν του σχήματος 2.1, ο αλγόριθμος Dijkstra διατηρεί ένα σύνολο *S* από κόμβους, των οποίων τα ελάχιστα βάρη σε σχέση με τον κόμβο προέλευσης είναι ήδη γνωστά.



Σχήμα 2.1: Κατευθυνόμενος γράφος με βάρη

Αρχικά, αυτό το σύνολο περιέχει μόνο τον κόμβο εκκίνησης, ενώ όλοι οι άλλοι αποτελούν το σύνολο *V-S*. Σε κάθε βήμα προσθέτουμε στο σύνολο *S* έναν εναπομείναντα κόμβο, του οποίου το βάρος σχέση με τον κόμβο προέλευσης είναι το μικρότερο δυνατό. Υποθέτοντας ότι όλοι οι σύνδεσμοι έχουν μη-αρνητικά βάρη, μπορούμε πάντα να βρούμε τη συντομότερη διαδρομή από την αφετηρία μέχρι οποιονδήποτε άλλο κόμβο, που να διέρχεται μόνο από κόμβους, οι οποίοι ανήκουν στο σύνολο *S*, και σε κάθε βήμα, χρησιμοποιούμε έναν πίνακα για να την καταγράφουμε. Όταν το σύνολο *S* περιλαμβάνει όλους τους κόμβους, ο πίνακας θα περιέχει τη συντομότερη απόσταση από τον κόμβο εκκίνησης προς κάθε κόμβο. Ο αλγόριθμος αυτός είναι παρόμοιος με τους αλγόριθμους αναζήτησης κατά πλάτος, που είναι κατά προσέγγιση ανάλογοι της επιπεδοδιατεταγμένης διάσχισης ενός δέντρου.

Η μετατροπή ενός οδικού δικτύου σε έναν κατευθυνόμενο γράφο παρόμοιο με αυτόν του σχήματος 2.1. είναι ιδιαίτερα εύκολη. Μια απαραίτητη συνθήκη για τη χρήση του αλγόριθμου αυτού είναι η απαίτηση το βάρος για κάθε σύνδεσμο να είναι μη-αρνητικό, το οποίο ισχύει πάντα στην πλοήγηση οχημάτων. Εφ' όσον η παραπάνω συνθήκη ικανοποιείται, ο αλγόριθμος του Dijkstra θα βρει τις βέλτιστες λύσεις από τον κόμβο εκκίνησης προς κάθε άλλο κόμβο. Στο παράδειγμα μας, αυτό σημαίνει πως

ο αλγόριθμος θα βρει τις διαδρομές ελάχιστου κόστους από τον κόμβο 1 προς τους κόμβους 2,3, και 4 αντίστοιχα.

Έστω *n* ο συνολικός αριθμός των κόμβων και *s* ο συνολικός αριθμός των συνδέσμων σε ένα οδικό δίκτυο. Η πολυπλοκότητα του αλγόριθμου Dijkstra είναι $O(n^2 + s) = O(n^2)$, δηλαδή τετραγωνική.

Παραπάνω είδαμε ότι ο αλγόριθμος Dijkstra επιλέγει τον κόμβο με το ελάχιστο βάρος και σχέση με τον κόμβο προέλευσης και τον κάνει μέλος του συνόλου *S*. Με τον τρόπο αυτό χτίζεται σταδιακά το συντομότερο μονοπάτι από την αφετηρία προς όλους τους υπόλοιπους κόμβους. Στον αρχικό αλγόριθμο οι κόμβοι του συνόλου *V-S* δεν είναι ταξινομημένοι. Αυτό πρακτικά σημαίνει ότι για να βρούμε τον κόμβο με το ελάχιστο βάρος πρέπει να ελέγχουμε τα βάρη όλων των κόμβων σε κάθε επανάληψη, πράγμα το οποίο είναι ασύμφορο. Μία βελτίωση του αρχικού αλγορίθμου Dijkstra θα ήταν η αποθήκευση των κόμβων σε μία δομή δεδομένων με τέτοιο τρόπο ώστε οι κόμβοι να είναι ταξινομημένοι με αύξουσα διάταξη των βαρών τους.

Αυτό είναι εφικτό όταν για τον *παράγοντα διακλάδωσης b,* ο οποίος ορίζεται ως ο μέσος αριθμός των συνδέσμων**,** που ξεκινούν από κάθε κόμβο, ισχύει $b << n$. Σε προβλήματα πλοήγησης οχημάτων η συνθήκη $b << n$ ικανοποιείται εύκολα επειδή ο αριθμός των διασταυρώσεων στην περιοχή του προβλήματος είναι συνήθως πολύ μεγαλύτερος από τον μέσο αριθμό των δρόμων, που αποτελούν την κάθε διασταύρωση. Συνεπώς, είναι πρακτική η οργάνωση των κόμβων σε μια ουρά προτεραιότητας με μία δυαδική σωρό**.** Η πολυπλοκότητα αυτού του αλγόριθμου είναι $O((n + s)logn)$ ή $O(slogn)$ στην περίπτωση, που όλοι οι κόμβοι είναι προσβάσιμοι από την αφετηρία. Μια άλλη τροποποίηση είναι να εφαρμόσουμε μια σειρά προτεραιότητας με σωρό Fibonacci**.** Αυτό θα έχει ως αποτέλεσμα η πολυπλοκότητα αυτής της έκδοσης του αλγόριθμου Dijkstra να γίνει $O(nlogn + s) = O(nlogn)$. Με άλλα λόγια, η χρήση καλύτερων δομών δεδομένων μειώνει την τετραγωνική πολυπλοκότητα του αλγόριθμου Dijkstra σε λογαριθμική. Περαιτέρω μείωση της πολυπλοκότητας μπορεί να επιτευχθεί με χρήση και άλλων δομών δεδομένων, όπως η σωρός *radix* [3].

2.2.2 Τροποποιημένος αλγόριθμος Dijkstra

Για την πλοήγηση οχημάτων δεν είναι απαραίτητο να βρεθεί η βέλτιστη διαδρομή από τον κόμβο εκκίνησης προς όλους τους άλλους κόμβους στο οδικό δίκτυο. Ο αλγόριθμος Dijkstra μπορεί να τροποποιηθεί, ώστε να τερματίζει όταν βρει τη βέλτιστη διαδρομή μόνο προς τον κόμβο προορισμού. Παρακάτω παρατίθεται μια τροποποίηση του αλγόριθμου Dijkstra με μία διπλά συνδεδεμένη λίστα. Κάθε κόμβος στη λίστα έχει έναν δείκτη προς τον πρόγονο του και έναν ή περισσότερους δείκτες προς τους απογόνους του ή το κενό. Παρόλο, που αυτός ο αλγόριθμος δεν είναι τόσο δημοφιλής στον σχεδιασμό διαδρομής, είναι απαραίτητο να τον αναλύσουμε έτσι ώστε ο αναγνώστης να μπορεί να τον συγκρίνει με τους επόμενους και να εξοικειωθεί με τη σχετική ορολογία. Οι αλγόριθμοι, που θα αναλυθούν στη συνέχεια, αποτελούν περαιτέρω βελτιώσεις αυτού. Να σημειωθεί πως γενικά υπάρχει μια πληθώρα υλοποιήσεων κάθε αλγόριθμου. Στη διπλωματική αυτή δίνεται έμφαση στην παρουσίαση των βασικών χαρακτηριστικών καθενός και στην κατανόηση του τρόπου, με τον οποίο μπορεί να χρησιμοποιηθεί για να λύσει το πρόβλημα του σχεδιασμού διαδρομής, και όχι σε κάποια συγκεκριμένη τεχνική υλοποίησης.

Το οδικό δίκτυο αναπαριστάται με μία ψηφιακή βάση των δεδομένων του χάρτη . Ο αλγόριθμος πραγματοποιεί αναζήτηση μέσα σε αυτό το οδικό δίκτυο, όπου κάθε κόμβος αντιπροσωπεύει μία διασταύρωση ή ένα αδιέξοδο. Υποθέτουμε ότι κατά την εκτέλεση του αλγόριθμου μετατρέπουμε κάθε κόμβο της βάσης δεδομένων σε μία δομή δεδομένων, που περιέχει τουλάχιστον τις συντεταγμένες x, y, κάποιες ιδιότητες του δρόμου, μια ένδειξη της προοπτικής του ως ένα ενδιάμεσο σημείο δρομολόγησης, έναν σύνδεσμο, που δείχνει πίσω στον κόμβο από τον οποίο προέκυψε και έναν σύνδεσμο, που δείχνει στον επόμενο κόμβο σε μία από τις δύο συνδεδεμένες λίστες, OPEN και CLOSED:

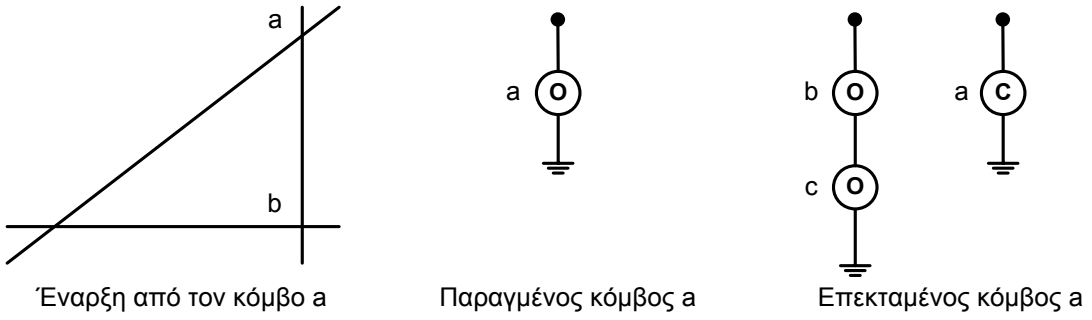
OPEN: λίστα κόμβων, που έχουν παραχθεί αλλά δεν έχουν ακόμα επεκταθεί.

CLOSED: λίστα κόμβων, που έχουν επεκταθεί.

Ο όρος *παραχθεί* σημαίνει να δημιουργηθεί η δομή δεδομένων, που αφορά έναν συγκεκριμένο κόμβο και ο όρος *επεκταθεί* σημαίνει να παραχθούν όλοι οι απόγονοί του. Αυτή η δομή δεδομένων πρέπει να περιέχει όλη την απαραίτητη πληροφορία για τον σχεδιασμό της διαδρομής, όπως περιγράφηκε σύντομα στην προηγούμενη παράγραφο. Στο σχήμα 2.2 παρουσιάζεται ένα παράδειγμα παραγμένων και

επεκταμένων κόμβων στις αντίστοιχες λίστες, όπου ο b και c είναι διάδοχοι του a. Μία μαύρη κουκκίδα υποδεικνύει την κορυφή της κάθε λίστας, ενώ η ηλεκτρική γείωση το τέλος της. Στην πραγματική υλοποίηση χρησιμοποιείται ένας δείκτης προς το κενό.

Για να απλοποιηθεί το σχήμα, οι κόμβοι αναπαρίστανται με μικρούς κύκλους, ενώ έχουν αγνοηθεί οι δείκτες και η δομή δεδομένων, που σχετίζονται με τον εκάστοτε κόμβο. Τα γράμματα μέσα σε κάθε κύκλο υποδεικνύουν το όνομα της κάθε λίστας, OPEN ή CLOSED.



Σχήμα 2.2: Συνδεδεμένες λίστες

Μία συνάρτηση αξιολόγησης (συνάρτηση κόστους) χρησιμοποιείται για να υπολογίσει την αξία κάθε κόμβου, που δημιουργείται. Αυτή η συνάρτηση κατευθύνει τον αλγόριθμο, ώστε να αναζητήσει πρώτα τους πιο «ελπιδοφόρους» κόμβους. Η συνάρτηση αξιολόγησης ορίζεται για τον κόμβο n ως $g(n)$, όπου $g(n)$ είναι ένα μέτρο του πραγματικού κόστους της μετάβασης από τον τρέχοντα κόμβο στον κόμβο n. Ο καθορισμός της συνάρτησης αξιολόγησης, που θα χρησιμοποιηθεί, είναι στη διακριτική ευχέρεια του σχεδιαστή ή του χρήστη του συστήματος, όπως αναφέρθηκε προηγουμένως. Ακολουθεί ο ψευδοκώδικας για τον τροποποιημένο αλγόριθμο Dijkstra:

1. Έστω μία λίστα OPEN, που περιέχει μόνο τον κόμβο αφετηρίας με μηδενικό κόστος (τιμή της g) και μία λίστα CLOSED, που είναι άδεια. Έστω ότι τα κόστη όλων των υπόλοιπων κόμβων είναι ίσα με το άπειρο.

2. Εάν δεν υπάρχει κανένας κόμβος στη λίστα OPEN, ανέφερε αποτυχία. Διαφορετικά επέλεξε τον κόμβο από τη λίστα OPEN με το

μικρότερο κόστος (τιμή της g) και ονόμασε τον ως BEST. Αφαίρεσε τον από τη λίστα OPEN και τοποθέτησε τον στη λίστα CLOSED. Έλεγξε αν ο BEST είναι ο κόμβος προορισμού. Εάν είναι, πήγαινε στο βήμα 3, διαφορετικά παρήγαγε τους διαδόχους του BEST. Για κάθε διάδοχο κόμβο n, ακολούθησε τα παρακάτω βήματα:

2α. Υπολόγισε το κόστους του n: g(n) = κόστος του BEST + κόστος από τον BEST στον n.

2b. Εάν ο n ταυτίζεται με κάποιον κόμβο, που βρίσκεται ήδη στη λίστα OPEN, έλεγξε αν ο κόμβος n έχει μικρότερο κόστος (τιμή g). Εάν ισχύει, αντικατέστησε το κόστος του ταυτιζόμενου κόμβου με το κόστους του κόμβου n και όρισε τον δείκτη του να δείχνει στον BEST.

2c. Εάν ο n ταυτίζεται με κάποιον κόμβο, που βρίσκεται ήδη στη λίστα CLOSED, έλεγξε αν ο κόμβος n έχει μικρότερο κόστος (τιμή g). Εάν ισχύει, αντικατέστησε το κόστος του ταυτιζόμενου κόμβου με το κόστους του κόμβου n, όρισε τον δείκτη του να δείχνει στον BEST και μετέφερε τον στη λίστα OPEN.

2d. Εάν ο n δεν είναι ήδη ούτε στη λίστα OPEN ούτε στη λίστα CLOSED, όρισε τον δείκτη του να δείχνει στον BEST και τοποθέτησε τον στη λίστα OPEN. Επανέλαβε το βήμα 2.

3. Από τον BEST διέσχισε τους δείκτες προς τα πίσω μέχρι τον κόμβο αφετηρίας και ανέφερε τη λύση της διαδρομής.

Ο παραπάνω αλγόριθμος θα βρει τη βέλτιστη διαδρομή από μία δοσμένη αφετηρία προς έναν δοσμένο προορισμό. Οι τεχνικές, που χρησιμοποιήθηκαν στον αρχικό αλγόριθμο Dijkstra, μπορούν να χρησιμοποιηθούν και εδώ για να μειωθεί η πολυπλοκότητά του. Μια ενδιαφέρουσα διαπίστωση είναι ότι αν όλοι οι σύνδεσμοι έχουν ίσα κόστη, ο αλγόριθμος μετατρέπεται σε μία απλή αναζήτηση κατά πλάτος.

Σε αυτόν τον αλγόριθμο ελέγχεται κάθε απόγονος αν βρίσκεται ήδη στη λίστα OPEN ή CLOSED, για να αποφευχθεί η περίπτωση ένας κόμβος να εμφανίζεται παραπάνω από μία φορά. Είναι πιθανό περισσότερες από μία διαδρομές να φτάνουν στον προορισμό διερχόμενες από τον ίδιο κόμβο ή διασταύρωση. Το γεγονός ότι ένας κόμβος απαντήθηκε περισσότερες από μία φορές κατά τη διάρκεια της αναζήτησης,

π.χ. ένας πρόσφατος διάδοχος είναι ήδη στη λίστα OPEN ή CLOSED, αντικατοπτρίζει στην πραγματικότητα ότι πολλαπλές διαδρομές διέρχονται από αυτόν. Μία από αυτές πρέπει να έχει το ελάχιστο κόστος. Ο έλεγχος αυτός εγγυάται ότι η βέλτιστη διαδρομή είναι πάντα αποθηκευμένη στη μνήμη [3].

Μία άλλη τεχνική, συχνά χρησιμοποιούμενη στον σχεδιασμό διαδρομής, είναι να περιλαμβάνονται απαγορευμένες καταστάσεις στη δο μή δεδομένων κάθε κόμβου. Γενικά, ένα οδικό δίκτυο περιέχει έναν σημαντικό αριθμό από αδιέξοδα, κλειστούς δρόμους και μονόδρομους, που δε μπορούν να χρησιμοποιηθούν στον σχεδιασμό της διαδρομής. Το ίδιο ισχύει και για τη διάσχιση ενός μονόδρομου προς λάθος κατεύθυνση. Ένας απλός τρόπος αντιμετώπισης αυτού του προβλήματος είναι η δυναμική ενσωμάτωση της πληροφορίας αυτής σε απαγορευμένες καταστάσεις στη δομή δεδομένων του αντίστοιχου κόμβου. Οι απαγορευμένοι κόμβοι δε μπορούν να χρησιμοποιηθούν για την κατασκευή του δέντρου αναζήτησης. Συνεπώς το σύστημα αυτομάτως τους αποφεύγει. Μια εναλλακτική είναι η αποφυγή δημιουργίας απογόνου ενός υπάρχοντος κόμβου της λίστας OPEN, εάν αυτός είναι συνδεδεμένος με ένα απαγορευμένο τμήμα. Η ιδέα αυτή μπορεί να επεκταθεί γ ια τη βελτίωση του σχεδιασμού της διαδρομής. Για παράδειγμα, ένας χρήστης επιθυμεί να αποφύγει τους αυτοκινητόδρομους σε ένα συγκεκριμένο ταξίδι. Εάν ο σχεδιασμός του συστήματος επιτρέπει στον χρήστη να πραγματοποιήσει την παραπάνω επιλογή πριν την εκτέλεση του αλγόριθμου δρομολόγησης, τότε όλοι οι αυτοκινητόδρομοι θα επισημανθούν ως απαγορευμένοι ούτως ώστε να ικανοποιηθεί η απαίτηση του χρήστη [4].

Στις εφαρμογές πλοήγησης οχημάτων υπάρχει μία διαφορά μεταξύ της χρήσης του αλγόριθμου Dijkstra και της χρήσης της τροποποιημένης έκδοσης αυτού, που περιγράφηκε παραπάνω. Ο αλγόριθμος Dijkstra χρησιμοποιείται για να βρει λύση σε έναν δοσμένο γράφο ή για να υπολογίσει εκ των προτέρων διαδρομές *off-line*, οι οποίες και αποθηκεύονται στη μνήμη. Εν αντιθέσει, η τροποποιημένη έκδοση χρησιμοποιείται όταν είναι απαραίτητη η πλοήγηση πραγματικού χρόνου. Όταν ένα σύστημα πλοήγησης είναι σε λειτουργία ο χρήστης αναμένει την άμεση αναπαράσταση της βέλτιστης λύσης. Παρόλο, που μπορούμε να απεικονίσουμε ένα μεγάλο, λεπτομερές οδικό δίκτυο σαν έναν γράφο, από την οπτική γωνία της πλοήγησης οχημάτων, δεν είναι πρακτικό, με βάση την παρούσα τεχνολογία, να φορτώσουμε ολόκληρο το δίκτυο στη μνήμη για τον σχεδιασμό της διαδρομής. Η συνήθης πρακτική είναι να φορτώσουμε, κατ' απαίτηση, στη μνήμη το τμήμα του

δικτύου, που είναι πιο πιθανό να χρησιμοποιηθεί κατά τον σχεδιασμό της διαδρομής. Δεν χρειαζόμαστε ολόκληρο τον γράφο για να εργαστούμε, ειδικά στον υπολογισμό πραγματικού χρόνου. Συνεπώς, μπορούμε να φανταστούμε ότι η αναζήτηση πραγματοποιείται σε ένα δέντρο, το οποίο έχει ρίζα τον κόμβο εκκίνησης και φύλλα τους υπόλοιπους παραγμένους κόμβους. Αυτό έχει σαν αποτέλεσμα να είναι απαραίτητο ένα διαφορετικό μέτρο για την αξιολόγηση της πολυπλοκότητας του αλγόριθμου.

Έστω ότι d είναι το βάθος της αναζήτησης της συντομότερης λύσης από τον κόμβο εκκίνησης προς τον κόμβο προορισμού. Ως βάθος αναζήτησης ορίζεται ο αριθμός των επιπέδων ενός δέντρου, που η διαδικασία αναζήτησης χρειάζεται να διασχίσει για να βρει μία λύση. Για παράδειγμα, αν θεωρήσουμε τη ρίζα ως επίπεδο 0, όλοι οι απόγονοι της θα βρίσκονται στο επίπεδο 1. Με αυτήν την υπόθεση, η πολυπλοκότητα του τροποποιημένου αλγόριθμου Dijkstra γίνεται $O(b^d)$. Κατά την ολοκλήρωση του σχεδιασμού της διαδρομής, ο συνολικός αριθμός των κόμβων του δέντρου πρέπει να είναι περίπου ίσος με b^d.

Ο τροποποιημένος αλγόριθμος Dijkstra απαιτεί περισσότερο χρόνο και χώρο για να εκτελεστεί σε σχέση με τους αλγόριθμους ευριστικής αναζήτησης, που θα αναλυθούν παρακάτω.

2.3 Ευριστική αναζήτηση

Η ευριστική αναζήτηση είναι μια πληροφορημένη στρατηγική αναζήτησης, γεγονός το οποίο σημαίνει ότι έχει πληροφορίες σχετικά με τον αριθμό των βημάτων ή το κόστος από την αρχική και την παρούσα κατάσταση μέχρι την τελική. Οι πληροφορίες αυτές χρησιμοποιούνται για τη βελτίωση της αποδοτικότητας της διαδικασίας αναζήτησης, θυσιάζοντας ενδεχομένως αξιώσεις πληρότητας. Παραδείγματα ενημερωμένων μεθόδων αναζήτησης αποτελούν η *best-first* αναζήτηση, η *memory bounded* αναζήτηση και οι αλγόριθμοι επαναληπτικής βελτίωσης, όπως ο *hill-climbing search* και ο *simulated annealing*.

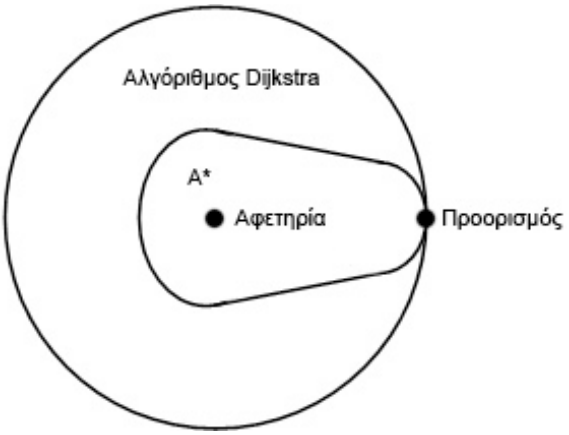
Ο λόγος, που πρέπει να πραγματοποιήσουμε μια αναζήτηση, είναι το ότι δεν ξέρουμε προς ποια κατεύθυνση ακριβώς να κινηθούμε. Αν το γνωρίζαμε εξ αρχής, το πρόβλημα θα ήταν πολύ πιο απλό. Η ευριστική πληροφορία βοηθάει στον καθορισμό

του περισσότερο «υποσχόμενου» κόμβου, ποιοι διάδοχοι θα παραχθούν και ποια άσχετα κλαδιά αναζήτησης θα περικοπούν με αποτέλεσμα η αναζήτηση να είναι πιο αποδοτική. Ο σκοπός της ευριστικής πληροφορίας είναι η παροχή μιας εκτίμησης της απόστασης του κόμβου προορισμού από τον τρέχοντα κόμβο, έτσι ώστε το σύστημα να μπορεί να καθορίσει πόσο πιθανό είναι ο συγκεκριμένος κόμβος να περιλαμβάνεται σ τη βέλτιστη λύση. Οι ευριστικές πληροφορίες είναι αποτελεσματικές στον βαθμό, που δείχνουν προς τη σωστή κατεύθυνση, αλλά μπορούν προσωρινά ή περιστασιακά να μας οδηγήσουν σε αναζήτηση ενός ακατάλληλου κόμβου, όπως ένα αδιέξοδο, για μια συγκεκριμένη διαδρομή. Παρά το προσωρινό αυτό πρόβλημα, η διαδικασία θα οδηγήσει στον κόμβο προορισμού, αν αυτός υπάρχει. Η χρήση καλών ευριστικών προσφέρει τη δυνατότητα εύρεσης καλής, αν όχι βέλτιστης, λύσης σε λιγότερο χρόνο και με λιγότερη κατανάλωση μνήμης σε σχέση με τον απλό ή τροποποιημένο αλγόριθμο Dijkstra.

*2.3.1 Αλγόριθμος A**

Η πιο δημοφιλής ευριστική αναζήτηση, που εφαρμόζεται ευρέως στον σχεδιασμό διαδρομής, είναι ο αλγόριθμος A* [5], ο οποίος χρησιμοποιεί την *best-first* μέθοδο. Ο αλγόριθμος A* έχει χρησιμοποιηθεί και σε άλλα πεδία εκτός της πλοήγησης οχημάτων, όπως στη ρο μποτική και σε περιοχές ενδιαφέροντος όπου απαιτούνται λύσεις ελάχιστου κόστους.

Ο αλγόριθμος A* είναι μια εξαιρετικά επιτυχημένη εφαρμογή της *best-first* αναζήτησης σε οδικά δίκτυα. Εάν η επιλεγμένη συνάρτηση εκτίμησης ικανοποιεί μια συγκεκριμένη συνθήκη, που θα αναλυθεί παρακάτω, ο αλγόριθμος A* εγγυάται ότι θα βρει μία βέλτιστη διαδρομή, εάν αυτή υπάρχει, ενώ η αναζήτηση θα πραγματοποιηθεί σε λιγότερο χώρο τόσο από τον αρχικό όσο και από τον τροποποιημένο αλγόριθμο Dijkstra. Για να γίνει κ ατανοητή η εξοικονόμηση αυτή χρησιμοποιούμε έναν κύκλο για να παραστήσουμε τον χώρο στον οποίο γίνεται η αναζήτηση στον απλό και τροποποιημένο αλγόριθμο Dijkstra και μία έλλειψη για τον αντίστοιχο χώρο στον αλγόριθμο A*, όπως φαίνεται στο σχήμα 2.3.



Σχήμα 2.3: Οι χώροι αναζήτησης στον αλγόριθμο A και τον αλγόριθμο Dijkstra*

Ένας καλός τρόπος για τη χρήση ευριστικής πληροφόρησης είναι ο υπολογισμός μιας ευριστικής συνάρτησης, η οποία αξιολογεί κάθε κόμβο, που δημιουργείται, για να προσδιορίσει αν είναι καλός ή κακός. Με τον τρόπο αυτό, έχοντας τ ην ευριστική συνάρτηση να προτείνει ποια διαδρομή να ακολουθηθεί πρώτα, όταν περισσότερες από μία είναι διαθέσιμες, γίνεται η αναζήτηση όσο το δυνατόν πιο αποτελεσματική. Στον αλγόριθμο A* αυτή η ευριστική συνάρτηση αξιολόγησης επιτρέπει στον αλγόριθμο να αναζητήσει πρώτα τους πιο «ελπιδοφόρους» κόμβους. Η συνάρτηση αξιολόγησης $f'(n)$ για τον κόμβο *n* ορίζεται ως εξής:

$$f'(n) = g(n) + h'(n) \qquad (2.3)$$

όπου *g(n)* είναι το πραγματικό κόστος μετάβασης από τον κόμβο προέλευσης στον τρέχοντα κόμβο *n* και $h'(n)$ είναι μία εκτίμηση του ελάχιστου κόστους μετάβασης από τον τρέχοντα κόμβο *n* στον κόμβο προορισμού. Συγκρίνοντας τις δύο εξισώσεις αξιολόγησης, που συζητήθηκαν μέχρι τώρα, παρατηρούμε ότι όταν $h'(n) = 0$ η εξίσωση (2.3) ταυτίζεται με αυτήν, που χρησιμοποιήθηκε στον τροποποιημένο αλγόριθμο Dijkstra.

Ο καθορισμός του κόστους, που θα υπολογίζεται από τη συνάρτηση αξιολόγησης εξαρτάται από τον σχεδιαστή του συστήματος ή τον χρήστη. Για παράδειγμα, εάν ο συντομότερος χρόνος ταξιδιού είναι προτιμότερος, μια λύση είναι η υιοθέτηση της παρακάτω εξίσωσης για την ευριστική συνάρτηση αξιολόγησης :

$$f'(n) = t'(n) = g(n) + h'(n) = \sum_{i=1}^{n} \frac{d_i(n)}{v_i(n)} + \frac{d'(n)}{v'} \qquad (2.4)$$

όπου $t'(n)$ είναι ο χρόνος ταξιδιού, $d_i(n)$ είναι η πραγματική απόσταση για το τμήμα i και $v_i(n)$ είναι η μέγιστη ταχύτητα (όριο ταχύτητας) για αυτό το τμήμα. Ως $g(n)$ ορίζεται ο χρόνος μετάβασης από τον κόμβο εκκίνησης στον τρέχον κόμβο και υπολογίζεται για κάθε τμήμα ως ο λόγος της απόστασης του εκάστοτε τμήματος προς τη μέγιστη ταχύτητα. Η εκτίμηση $h'(n)$ είναι ο λόγος της ευκλείδειας απόστασης $d'(n)$ μεταξύ του τρέχοντα κόμβου n και ενός δοσμένου κόμβου προορισμού προς την εκτιμώμενη μέγιστη ταχύτητα v'. Να σημειωθεί πως χρησιμοποιήθηκε η ευκλείδεια απόσταση στην εκτίμηση $h'(n)$, που είναι και η μικρότερη απόσταση μεταξύ του τρέχοντα κόμβου και του κόμβου προορισμού. Εάν η εκτιμώμενη μέγιστη ταχύτητα είναι μεγαλύτερη ή ίση της πραγματικής ταχύτητας σε κάθε τμήμα, ο χρόνος ταξιδίου πάντα θα υποτιμάται. Εν συντομία, ο σκοπός του αλγόριθμου είναι να ελαχιστοποιήσει το $f'(n)$ και εν προκειμένω το $t'(n)$, δηλαδή τον χρόνο ταξιδίου.

Η λειτουργία του αλγόριθμου προχωράει σταδιακά, επεκτείνοντας έναν κόμβο (με αρχή έναν δοσμένο κόμβο εκκίνησης) σε κάθε βήμα, μέχρι να προκύψει ο κόμβος, που αντιστοιχεί στον δοσμένο προορισμό. Σε κάθε βήμα μόνο ο πιο «ελπιδοφόρος» κόμβος, δηλαδή ο κόμβος με το μικρότερο $f'(n)$, παράγει απογόνους. Κατά τη διάρκεια της διαδικασίας αυτής οι κόμβοι, που σχετίζονται με περιορισμούς (π.χ. το τμήμα έχει λάθος κατεύθυνση ή καταλήγει σε αδιέξοδο), εξαιρούνται έτσι ώστε οι κόμβοι, οι οποίοι οδηγούν σε παράνομες στροφές ή σε αδιέξοδα να μην λαμβάνονται υπόψη κατά την αναζήτηση και να μην περιέχονται στην τελική διαδρομή. Η ευριστική συνάρτηση εφαρμόζεται για να αξιολογήσει πόσο «ελπιδοφόρος» είναι κάθε κόμβος, αφού πρώτα ελέγξει αν ο κόμβος αυτός έχει παραχθεί ξανά. Ο έλεγχος αυτός εγγυάται ότι κάθε κόμβος βρίσκεται μόνο σε μία από τις δύο συνδεδεμένες λίστες. Η παραπάνω διαδικασία επαναλαμβάνεται, μέχρις ότου παραχθεί ο κόμβος προορισμού. Η προκύπτουσα λίστα των κόμβων και των συναφών τμημάτων συχνά αναφέρεται και ως λίστα ελιγμών, επειδή το όχημα αργότερα θα χρειαστεί να ακολουθήσει αυτή τη λίστα, προκειμένου να ελιχθεί μέσα από το οδικό δίκτυο και να φτάσει στον επιθυμητό προορισμό.

Εάν η ευριστική συνάρτηση $h'(n)$ δεν υπερεκτιμά το πραγματικό κόστος για την επίτευξη του κόμβου προορισμού, ο αλγόριθμος A* θα καταλήξει στη βέλτιστη λύση, εάν αυτή υπάρχει. Αυτό ονομάζεται ως ιδιότητα της παραδεκτότητας (*admissibility property)*. Όπως είναι προφανές από τη συνάρτηση αξιολόγησης, που παρατέθηκε προηγουμένως, η υπερεκτίμηση του κόστους αποφεύγεται γιατί η συνάρτηση $h'(n)$

το υποτιμά για κάθε κόμβο *n*. Συνεπώς, δεδομένου ότι η ευριστική συνάρτηση έχει επιλεχθεί προσεκτικά, ισχύει η ιδιότητα της παραδεκτότητας. Παρακάτω παρουσιάζεται ο ψευδοκώδικας για τον αλγόριθμο A*:

1. Έστω μία λίστα OPEN, που περιέχει μόνο τον κόμβο αφετηρίας με μηδενικό κόστος (τιμή της g) και μία λίστα CLOSED, που είναι άδεια. Έστω ότι τα κόστη όλων των υπόλοιπων κόμβων είναι ίσα με το άπειρο.

2. Εάν δεν υπάρχει κανένας κόμβος στη λίστα OPEN, ανέφερε αποτυχία. Διαφορετικά επέλεξε τον κόμβο από τη λίστα OPEN με το μικρότερο κόστος (τιμή της f') και ονόμασε τον ως BEST. Αφαίρεσε τον από τη λίστα OPEN και τοποθέτησε τον στη λίστα CLOSED. Έλεγξε αν ο BEST είναι ο κόμβος προορισμού. Εάν είναι, πήγαινε στο βήμα 3, διαφορετικά παρήγαγε τους διαδόχους του BEST. Για κάθε διάδοχο κόμβο n, ακολούθησε τα παρακάτω βήματα:

   2α. Υπολόγισε το κόστους του n: g(n) = κόστος του BEST + κόστος από τον BEST στον n.

   2b. Εάν ο n ταυτίζεται με κάποιον κόμβο, που βρίσκεται ήδη στη λίστα OPEN, έλεγξε αν ο κόμβος n έχει μικρότερο κόστος (τιμή g). Εάν ισχύει, αντικατέστησε το κόστος του ταυτιζόμενου κόμβου με το κόστους του κόμβου n και όρισε τον δείκτη του να δείχνει στον BEST.

   2c. Εάν ο n ταυτίζεται με κάποιον κόμβο, που βρίσκεται ήδη στη λίστα CLOSED, έλεγξε αν ο κόμβος n έχει μικρότερο κόστος (τιμή g). Εάν ισχύει, αντικατέστησε το κόστος του ταυτιζόμενου κόμβου με το κόστους του κόμβου n, όρισε τον δείκτη του να δείχνει στον BEST και μετέφερε τον στη λίστα OPEN.

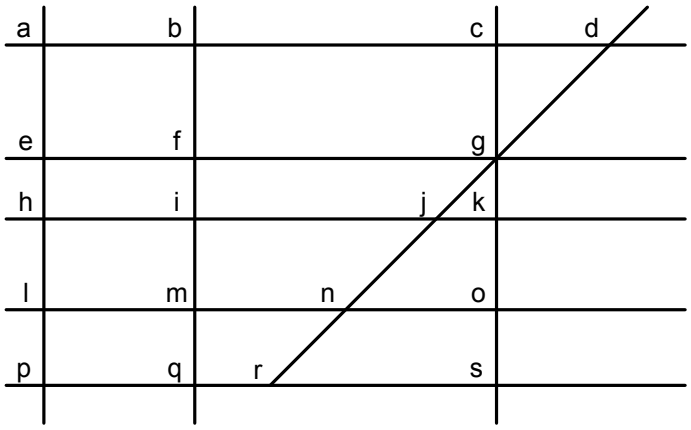
   2d. Εάν ο n δεν είναι ήδη ούτε στη λίστα OPEN ούτε στη λίστα CLOSED, όρισε τον δείκτη του να δείχνει στον BEST, τοποθέτησε τον στη λίστα OPEN και υπολόγισε την ευριστική συνάρτηση αξιολόγησης για τον κόμβο n :

   $$f'(n) = g(n) + h'(n)$$

   Επανέλαβε το βήμα 2.
3. Από τον BEST διέσχισε τους δείκτες προς τα πίσω μέχρι τον κόμβο αφετηρίας και ανέφερε τη λύση της διαδρομής.

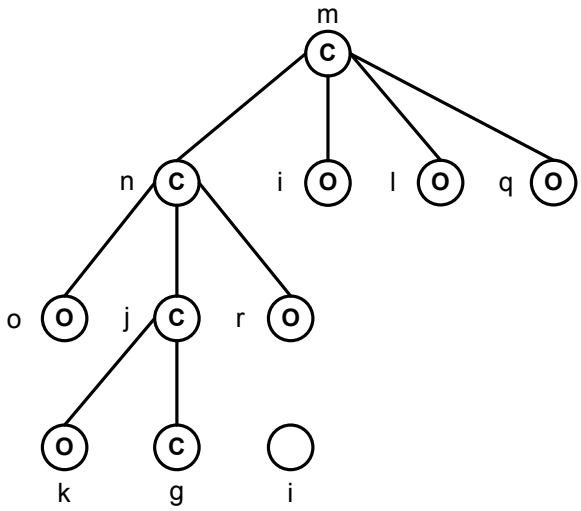
Εξετάζοντας προσεκτικά τον αλγόριθμο A*, γίνεται προφανές ότι η μόνη διαφορά μεταξύ αυτού και των αλγόριθμων υπολογισμού της συντομότερης διαδρομής είναι ότι η συνάρτηση αξιολόγησης είναι ευριστική. Ο τροποποιημένος αλγόριθμος Dijkstra είναι ουσιαστικά μια ειδική περίπτωση του αλγόριθμου A* με $h'(n) = 0$.

Για την καλύτερη κατανόησή του ακολουθεί ένα παράδειγμα με το πώς ο αλγόριθμος A* δημιουργεί ένα δέντρο αναζήτησης για το απλό οδικό δίκτυο του σχήματος 2.4, όπου κάθε γράμμα αντιστοιχεί σε μία διασταύρωση, με τον κόμβο αφετηρίας να είναι στο *m* και ο προορισμός στο *g*.



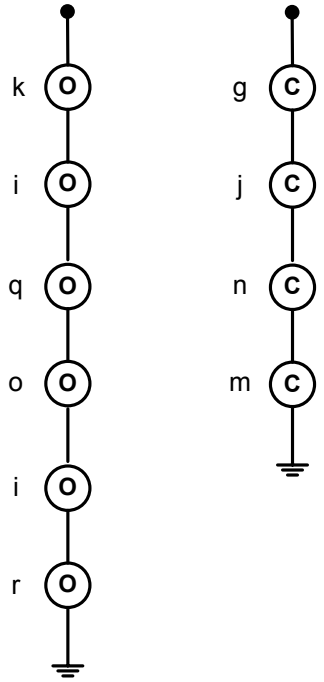
Σχήμα 2.4: Ένα απλό οδικό δίκτυο

Με την ολοκλήρωση του σχεδιασμού της διαδρομής το τελικό δέντρο αναζήτησης, που προκύπτει, φαίνεται στο σχήμα 2.5. Όπως και πριν, χρησιμοποιούμε ένα μικρό κύκλο για να παραστήσουμε κάθε κόμβο και μια ευθεία γραμμή για να παραστήσουμε τους δείκτες από έναν κόμβο προς τον πρόγονο του και τους απογόνους του. Τα γράμματα μέσα στους κύκλους δείχνουν ότι ένας κόμβος του δέντρου αναζήτησης ανήκει στη λίστα OPEN ή στη λίστα CLOSED. Παρόλο, που κάθε κόμβος στο δέντρο μπορεί να εμφανίζεται περισσότερες από μία φορά ως απόγονος, δύναται να απαντάται μόνο μία φορά και σε μία μόνο λίστα, αφού ο έλεγχος στα βήματα 2b και 2c εξαλείφει τους περιττούς κόμβους. Η πολλαπλή εμφάνιση κάποιων κόμβων στο δέντρο δεν είναι δύσκολο να γίνει κατανοητή, εφόσον ενδέχεται να υπάρχουν περισσότερες από μία διαδρομές προς τον προορισμό, που να περνούν από την ίδια διασταύρωση. Να σημειωθεί πως αυτό δε συνέβη στο συγκεκριμένο παράδειγμα, επειδή απλά ο αλγόριθμος δεν έλεγξε ποτέ τον τελικό απόγονο *i*, γιατί είχε βρει ήδη τον προορισμό g και ως καλύτερη διαδρομή προέκυψε η *m-n-j-g*.



Σχήμα 2.5: Δέντρο αναζήτησης

Οι τελικές λίστες OPEN και CLOSED για το συγκεκριμένο παράδειγμα απεικονίζονται στο σχήμα 2.6. Η μαύρη κουκκίδα υποδεικνύει την αρχή της λίστας, ενώ η ηλεκτρική γείωση το τέλος της. Η λίστα OPEN είναι διατεταγμένη με τέτοιο τρόπο, ώστε ο κόμβος με το μικρότερο κόστος να είναι πάντα στην κορυφή.



Σχήμα 2.6: Οι λίστες OPEN και CLOSED

Θεωρώντας πάλι τον *παράγοντα διακλάδωσης b,* ο οποίος ορίζεται ως ο μέσος αριθμός των συνδέσμων**,** που ξεκινούν από κάθε κόμβο, και το *βάθος αναζήτησης d* της συντομότερης διαδρομής από τον κόμβο αφετηρίας στον κόμβο προορισμού, προκύπτει ότι η πολυπλοκότητα του αλγόριθμου A* είναι $O(b^d)$. Με μία αρχική ματιά παρατηρούμε ότι ο χρόνος εκτέλεσης του αλγόριθμου A* είναι ίδιος με αυτόν του τροποποιημένου αλγόριθμου Dijkstra. Στην πραγματικότητα όμως η ευριστική πληροφορία, που χρησιμοποιεί ο αλγόριθμος A*, τείνει να ελαττώσει τον χρόνο αυτό, γιατί η τιμή του *b* είναι μικρότερη από ότι στον τροποποιημένο αλγόριθμο Dijkstra. Γι' αυτό τον λόγο άλλωστε χρησιμοποιείται η ευριστική προσέγγιση. Όπως, έχει ήδη αναφερθεί προηγουμένως, η πολυπλοκότητα του αλγόριθμου A* μπορεί να μειωθεί περαιτέρω με τη χρήση καλύτερων δομών δεδομένων [3].

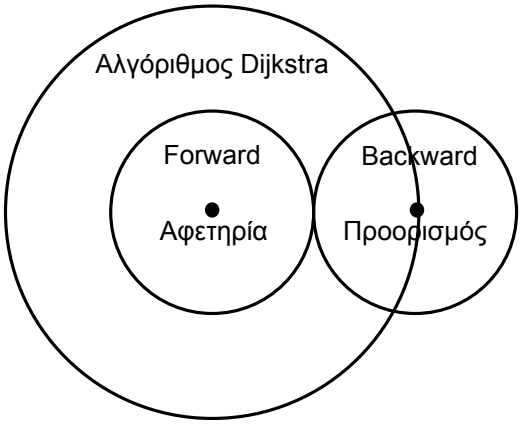
Εξαιτίας της αυξανόμενης τάσης για έρευνα και ανάπτυξη στον τομέα της πλοήγησης οχημάτων, έχουν πραγματοποιηθεί πολλές μελέτες για διάφορους αλγόριθμους σχεδιασμού της διαδρομής. Δεν υπάρχει αμφιβολία πως αυτές οι μελέτες θα συνεχιστούν, γιατί ο σχεδιασμός της διαδρομής είναι ένα από τα στοιχεία κλειδιά των συστημάτων πλοήγησης.

2.4 Αμφίδρομη αναζήτηση

Η κατανόηση των αλγορίθμων συντομότερης διαδρομής και ευριστικής αναζήτησης οδήγησε στη γενίκευση αυτών και στη δημιουργία αλγόριθμων αμφίδρομης αναζήτησης [6]. Στη μελέτη των προηγούμενων αλγορίθμων η αναζήτηση της βέλτιστης διαδρομής πραγματοποιήθηκε από τον δοσμένο κόμβο αφετηρίας προς τον κόμβο προορισμού. Αυτή η αναζήτηση ονομάζεται *αναζήτηση προς τα εμπρός (forward search)*. Μία *αναζήτηση προς τα πίσω (backward search)* στην πραγματικότητα θα έχει το ίδιο αποτέλεσμα, εφόσον το κόστος κάθε συνδέσμου είναι ίδιο. Αυτό οδηγεί στην ιδέα μιας αμφίδρομης αναζήτησης.

Η βασική διαδικασία της αμφίδρομης αναζήτησης έγκειται στον ταυτόχρονο υπολογισμό της καλύτερης διαδρομής από τον κόμβο αφετηρίας προς τον κόμβο προορισμού (*forward search*) και από τον κόμβο προορισμού προς τον κόμβο αφετηρίας (*backward search*). Εάν ο αλγόριθμος αναζήτησης εκτελείται σε έναν μόνο επεξεργαστή, ο αλγόριθμος πρέπει να εναλλάσσεται γρήγορα μεταξύ των *forward* και

backward αναζητήσεων. Ο αμφίδρομος αλγόριθμος αναζήτησης ενδέχεται να μειώσει τον χώρο αναζήτησης κατά τουλάχιστον 50%, έτσι ώστε ο χρόνος εκτέλεσης του αλγόριθμου αυτού να είναι αρκετά μικρότερος σε σχέση με τη μονόδρομη αναζήτηση. Μια γενική αναπαράσταση του χώρου, που εξετάζεται, στον απλό και τροποποιημένο αλγόριθμο Dijkstra και στον αλγόριθμο αμφίδρομης αναζήτησης απεικονίζεται στο σχήμα 2.7.



Σχήμα 2.7: Οι χώροι αναζήτησης στον αλγόριθμο Dijkstra και στον αλγόριθμο αμφίδρομης αναζήτησης

Η επιτυχής εφαρμογή του αλγόριθμου αμφίδρομης αναζήτησης απαιτεί δύο επιπλέον συνθήκες (υποθέτοντας ότι χρησιμοποιείται ένας μόνο επεξεργαστής) : Ένα κριτήριο για τον τερματισμό της αναζήτησης και ένα κριτήριο για την εναλλαγή μεταξύ των *forward* και *backward* αναζητήσεων. Για παράδειγμα, μπορούμε να καθορίσουμε ένα κριτήριο τερματισμού όταν ένας κόμβος προσεγγισθεί και από τις δύο κατευθύνσεις αναζήτησης και ένα στατικό κριτήριο εναλλαγής όταν πραγματοποιηθούν δέκα επαναλήψεις σε κάθε κατεύθυνση. Εάν τα παραπάνω κριτήρια έχουν επιλεχθεί κατάλληλα, η αμφίδρομη αναζήτηση συνήθως θα εξετάσει λιγότερους κόμβους και θα καταλήξει σε μία λύση σε λιγότερο χρόνο συγκριτικά με τη μονόδρομη αναζήτηση.

Ακολουθεί ο ψευδοκώδικας για τον αλγόριθμο αμφίδρομης αναζήτησης:

1. Δημιούργησε μία λίστα forward OPEN, που περιέχει μόνο τον κόμβο αφετηρίας με μηδενικό κόστος (τιμή της g) και μία λίστα forward CLOSED, που είναι άδεια. Τα κόστη όλων των υπόλοιπων κόμβων όρισε τα ίσα με το άπειρο.

2. Δημιούργησε μία λίστα backward OPEN, που περιέχει μόνο τον κόμβο προορισμού με μηδενικό κόστος (τιμή της g) και μία λίστα backward CLOSED, που είναι άδεια. Τα κόστη όλων των υπόλοιπων κόμβων όρισε τα ίσα με το άπειρο.

3. Ενάλλασσε τη forward και backward αναζήτηση με βάση ένα στατικό ή δυναμικό κριτήριο. Εάν δεν υπάρχει κανένας κόμβος στη λίστα OPEN της επιλεγμένης κατεύθυνσης αναζήτησης δοκίμασε τη λίστα της αντίθετης κατεύθυνσης. Εάν δεν υπάρχει κανένας κόμβος και στις δύο λίστες OPEN, ανέφερε αποτυχία. Διαφορετικά επέλεξε τον κόμβο από τη λίστα OPEN με το μικρότερο κόστος (τιμή της g) και ονόμασε τον ως BEST. Αφαίρεσε τον από τη λίστα OPEN και τοποθέτησε τον στη λίστα CLOSED. Έλεγξε αν ο BEST ικανοποιεί κάποιο κριτήριο τερματισμού . Εάν ναι , πήγαινε στο βήμα 4, διαφορετικά παρήγαγε τους διαδόχους του BEST. Για κάθε διάδοχο κόμβο n, ακολούθησε τα παρακάτω βήματα:

   3α. Υπολόγισε το κόστους του n: g(n) = κόστος του BEST + κόστος από τον BEST στον n.

   3b. Εάν ο n ταυτίζεται με κάποιον κόμβο, που βρίσκεται ήδη στη λίστα OPEN, έλεγξε αν ο κόμβος n έχει μικρότερο κόστος (τιμή g). Εάν ισχύει, αντικατέστησε το κόστος του ταυτιζόμενου κόμβου με το κόστους του κόμβου n και όρισε τον δείκτη του να δείχνει στον BEST.

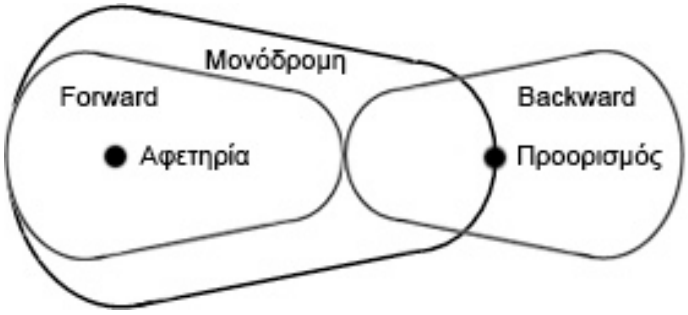
   3c. Εάν ο n ταυτίζεται με κάποιον κόμβο, που βρίσκεται ήδη στη λίστα CLOSED, έλεγξε αν ο κόμβος n έχει μικρότερο κόστος (τιμή g). Εάν ισχύει, αντικατέστησε το κόστος του ταυτιζόμενου κόμβου με το κόστους του κόμβου n, όρισε τον δείκτη του να δείχνει στον BEST και μετέφερε τον στη λίστα OPEN.

   3d. Εάν ο n δεν είναι ήδη ούτε στη λίστα OPEN ούτε στη λίστα CLOSED, όρισε τον δείκτη του να δείχνει στον BEST και τοποθέτησε τον στη λίστα OPEN. Επανέλαβε το βήμα 3.

4. Από τον BEST διέσχισε τους δείκτες προς τα πίσω μέχρι τον κόμβο αφετηρίας και προς τα εμπρός μέχρι τον κόμβο προορισμού, ένωσε τις δύο λίστες και ανέφερε τη λύση της διαδρομής.

Όπως αναφέρθηκε νωρίτερα η πολυπλοκότητα της μονόδρομης αναζήτησης (τροποποιημένος αλγόριθμος Dijkstra) είναι $O(b^d)$. Η αμφίδρομη αναζήτηση έχει πολυπλοκότητα $O\left(2b^{d/2}\right) = O(b^{d/2})$. Από αυτόν τον υπολογισμό παρατηρούμε ότι η αμφίδρομη αναζήτηση θα εξετάσει λιγότερους κόμβους από τη μονόδρομη αναζήτηση, όπως φαίνεται και στο σχήμα 2.7, κάτι, που οδηγεί σε γρηγορότερο υπολογισμό της λύσης.

Ομοίως, μία ευριστική αμφίδρομη αναζήτηση μπορεί να χρησιμοποιηθεί για την εύρεση της καλύτερης διαδρομής. Στο σχήμα 2.8 φαίνονται οι χώροι αναζήτησης για τη μονόδρομη και αμφίδρομη ευριστική αναζήτηση.



Σχήμα 2.8: Χώροι αναζήτησης στη μονόδρομη και αμφίδρομη ευριστική αναζήτηση.

Παρακάτω παρατίθεται ο ψευδοκώδικας για την αμφίδρομη ευριστική αναζήτηση:

1. Δημιούργησε μία λίστα forward OPEN, που περιέχει μόνο τον κόμβο αφετηρίας με μηδενικό κόστος (τιμή της g) και μία λίστα forward CLOSED, που είναι άδεια. Τα κόστη όλων των υπόλοιπων κόμβων όρισε τα ίσα με το άπειρο.

2. Δημιούργησε μία λίστα backward OPEN, που περιέχει μόνο τον κόμβο προορισμού με μηδενικό κόστος (τιμή της g) και μία λίστα backward CLOSED, που είναι άδεια. Τα κόστη όλων των υπόλοιπων κόμβων όρισε τα ίσα με το άπειρο.

3. Ενάλλασσε τη forward και backward αναζήτηση με βάση ένα στατικό ή δυναμικό κριτήριο. Εάν δεν υπάρχει κανένας κόμβος στη λίστα OPEN της επιλεγμένης κατεύθυνσης αναζήτησης δοκίμασε τη λίστα της αντίθετης κατεύθυνσης. Εάν δεν υπάρχει κανένας κόμβος και στις δύο λίστες OPEN, ανέφερε αποτυχία. Διαφορετικά επέλεξε τον κόμβο από τη λίστα OPEN με το μικρότερο κόστος (τιμή της

f') και ονόμασε τον ως BEST. Αφαίρεσε τον από τη λίστα OPEN και τοποθέτησε τον στη λίστα CLOSED. Έλεγξε αν ο BEST ικανοποιεί κάποιο κριτήριο τερματισμού . Εάν ναι πήγαινε στο βήμα 4, διαφορετικά παρήγαγε τους διαδόχους του BEST. Για κάθε διάδοχο κόμβο n, ακολούθησε τα παρακάτω βήματα:

3α. Υπολόγισε το κόστους του n: g(n) = κόστος του BEST + κόστος από τον BEST στον n.

3b. Εάν ο n ταυτίζεται με κάποιον κόμβο, που βρίσκεται ήδη στη λίστα OPEN, έλεγξε αν ο κόμβος n έχει μικρότερο κόστος (τιμή g). Εάν ισχύει, αντικατέστησε το κόστος του ταυτιζόμενου κόμβου με το κόστους του κόμβου n και όρισε τον δείκτη του να δείχνει στον BEST.

3c. Εάν ο n ταυτίζεται με κάποιον κόμβο, που βρίσκεται ήδη στη λίστα CLOSED, έλεγξε αν ο κόμβος n έχει μικρότερο κόστος (τιμή g). Εάν ισχύει, αντικατέστησε το κόστος του ταυτιζόμενου κόμβου με το κόστους του κόμβου n, όρισε τον δείκτη του να δείχνει στον BEST και μετέφερε τον στη λίστα OPEN.

3d. Εάν ο n δεν είναι ήδη ούτε στη λίστα OPEN ούτε στη λίστα CLOSED, όρισε τον δείκτη του να δείχνει στον BEST, τοποθέτησε τον στη λίστα OPEN και υπολόγισε την ευριστική συνάρτηση αξιολόγησης για τον κόμβο n :

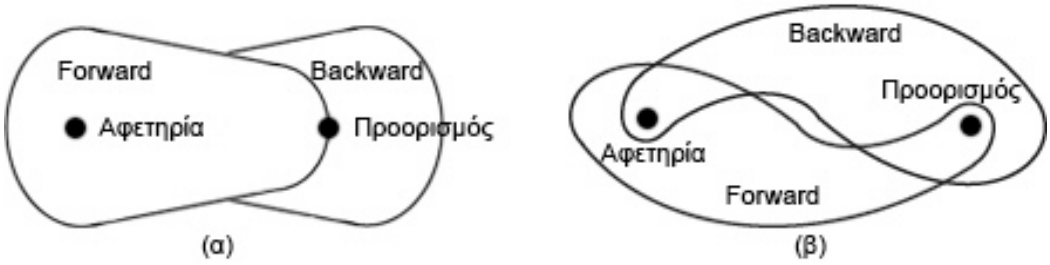
$$f'(n) = g(n) + h'(n)$$

Επανέλαβε το βήμα 3.

4. Από τον BEST διέσχισε τους δείκτες προς τα πίσω μέχρι τον κόμβο αφετηρίας και προς τα εμπρός μέχρι τον κόμβο προορισμού, ένωσε τις δύο λίστες και ανέφερε τη λύση της διαδρομής.

Στην ιδανική περίπτωση οι αναζητήσεις προς τις δύο κατευθύνσεις πρέπει να συναντηθούν στη μέση (σχήματα 2.7 και 2.8). Εάν υπάρχουν πολλαπλές διαφορετικές διαδρομές από τον κόμβο αφετηρίας προς τον κόμβο προορισμού, οι αναζητήσεις μπορεί να μη συναντηθούν στη μέση, όπως φαίνεται στο σχήμα 2.9. Καθεμία από τις δύο αναζητήσεις μπορούν να βρουν σχεδόν ολοκληρωμένες διαδρομές ή η μία να βρει ολοκληρωμένη λύση πριν να διασταυρωθούν. Στη χειρότερη περίπτωση (π.χ. όταν το κριτήριο τερματισμού της ευριστικής συνάρτησης αξιολόγησης λαμβάνει

υπόψη μόνο το εκτιμώμενο κόστος διαδρομής) η αμφίδρομη ευριστική αναζήτηση μπορεί να κάνει διπλάσια δουλειά σε σχέση με τη μονόδρομη ευριστική αναζήτηση. Συνεπώς, η επιλογή των κριτήριων τερματισμού και της κατάλληλης ευριστικής συνάρτησης αξιολόγησης είναι πολύ σημαντική για τη σωστή εφαρμογή της αμφίδρομης ευριστικής αναζήτησης.



Σχήμα 2.9: Οι χώροι αναζήτησης αλγορίθμων αμφίδρομης ευριστικής αναζήτησης με α) ακατάλληλο κριτήριο τερματισμού και β) ακατάλληλη ευριστική συνάρτηση αξιολόγησης.

Όπως αναφέρθηκε στην παράγραφο 2.3 αναφορικά με την ευριστική αναζήτηση, ο αλγόριθμος A* καταλήγει στη βέλτιστη διαδρομή εάν ικανοποιείται η ιδιότητα της παραδεκτότητας. Μία αμφίδρομη ευριστική αναζήτηση πρέπει να ελέγξει σχεδόν τους διπλάσιους κόμβους σε σχέση με μία μονόδρομη ευριστική αναζήτηση για να προσδιορίσει τη βέλτιστη διαδρομή. Αυτό συμβαίνει επειδή η αμφίδρομη αναζήτηση πρέπει να συγκρίνει όλες τις ενωμένες διαδρομές κατά την αναζήτηση για να εξασφαλίσει πως η διαδρομή είναι βέλτιστη. Στην πραγματικότητα, οι άνθρωποι σπάνια χρειάζονται να αναζητήσουν τη βέλτιστη λύση για τις καθημερινές τους δραστηριότητες. Μία σχεδόν βέλτιστη λύση είναι συχνά ικανοποιητική για πολλά προβλήματα. Εάν δεν είναι απαραίτητη μία βέλτιστη λύση, η αμφίδρομη αναζήτηση είναι ένας καλός υποψήφιος για τον σχεδιασμό διαδρομής. Με άλλα λόγια, όταν χρησιμοποιείται ένας αμφίδρομος αλγόριθμος αναζήτησης, οι περιορισμοί, που θέτει η βελτιστοποίηση, πρέπει να χαλαρώσουν. Στην περίπτωση αυτή η αμφίδρομη αναζήτηση θα βρει μια σχεδόν βέλτιστη διαδρομή πριν βρεθεί η συνολικά βέλτιστη λύση [3].

Υπάρχουν πολλές διαφορετικές παραλλαγές του αμφίδρομου αλγόριθμου αναζήτησης, που μπορούν να χρησιμοποιηθούν για να επιταχύνουν τη διαδικασία αναζήτησης. Στους αλγόριθμους, που παρουσιάστηκαν προηγουμένως, χρησιμοποιείται το κόστος της διαδρομής από τον κόμβο αφετηρίας ή προορισμού

για να εξακριβωθεί ποιοι κόμβοι χρειάζονται να επεκταθούν περαιτέρω, ενώ μία άλλη μέθοδος χρησιμοποιεί τον αριθμό των κόμβων, που επεκτείνονται, για να λάβει αυτές τις αποφάσεις. Η τελευταία μέθοδος έχει σαν αποτέλεσμα να επιλέγει διαδρομές, που εκκινούν από τον κόμβο αφετηρίας, οι οποίες έχουν περίπου τον ίδιο αριθμό κόμβων με τις διαδρομές, που εκκινούν από τον κόμβο προορισμού. Άλλη μέθοδος περιλαμβάνει τη χρήση ενός παράγοντα βαρύτητας για κάθε τμήμα δρόμου κατά τη διάρκεια της αναζήτησης, έτσι ώστε ένας μεγαλύτερος παράγοντας βαρύτητας να χρησιμοποιείται για τμήματα κοντά στην ευθεία γραμμή μεταξύ του κόμβου αφετηρίας και προορισμού και ένας μικρότερος για τμήματα μακριά από αυτή τη γραμμή. Συνεπώς, ο αλγόριθμος αυτός ενισχύει τη διαδικασία αναζήτησης να επιλέξει μια διαδρομή, που είναι κοντά στην ευθεία γραμμή, η οποία ενώνει τους δύο κόμβους εκκίνησης. Μια άλλη παραλλαγή έγκειται στη χρήση κατηγοριών δρόμων για τη μείωση της περιττής αναζήτησης. Αυτό λειτουργεί με την υπόθεση ότι μόνο στις περιοχές κοντά στους κόμβους αφετηρίας και προορισμού είναι απαραίτητο να ελεγχθούν όλοι οι δρόμοι, ενώ εκτός αυτών των περιοχών, ελέγχονται μόνο οι μεγάλοι δρόμοι. Σαφέστατα, αυτή η λειτουργία μειώνει τον συνολικό αριθμό των κόμβων, που εξετάζονται.

2.5 Ιεραρχική αναζήτηση

Υπάρχει μία φυσική ιεραρχία όσον αφορά τους δρόμους, με τους αυτοκινητόδρομους να βρίσκονται στην κορυφή. Αυτό οδηγεί στην ιδέα ενός πιο αποτελεσματικού αλγόριθμου ιεραρχικής αναζήτησης για ένα δοσμένο οδικό δίκτυο. Η βασική ιδέα είναι η αναζήτηση να πραγματοποιείται αρχικά σε έναν «αφαιρετικό» χώρο και όχι σε ολόκληρο τον χώρο του προβλήματος [7]. Ένας «αφαιρετικός» χώρος είναι μία απλοποιημένη παρουσίαση του χώρου του προβλήματος, στην οποία αγνοούνται οι ασήμαντες λεπτομέρειες. Κατόπιν, η αναζήτηση συμπληρώνεται με τις λεπτομέρειες από τον αρχικό χώρο με βάση τα αποτελέσματα, που προέκυψαν από την αναζήτηση στον «αφαιρετικό» χώρο. Η επιλογή μιας απλοποιημένης αναπαράστασης αυτού του τύπου οδηγεί σε σημαντική αύξηση της αποτελεσματικότητας και της ικανότητας επίλυσης του προβλήματος.

Οι άνθρωποι γνωρίζουν καλά την αξία της αφαίρεσης. Για παράδειγμα, όταν επιλύουν ένα σύνθετο πρόβλημα, σ την αρχή συχνά αγνοούν τις λεπτομέρειες και επικεντρώνονται στα βασικά χαρακτηριστικά του προβλήματος. Στη συνέχεια συμπληρώνουν τις λεπτομέρειες. Αυτή η ιδέα μπορεί εύκολα να γενικευτεί σε πολλά ιεραρχικά επίπεδα αφαίρεσης, έτσι ώστε κάθε επίπεδο να περιέχει ένα διαφορετικό επίπεδο λεπτομέρειας. Επειδή οι υπολογιστές αναπτύχθηκαν για να εξυπηρετούν τους ανθρώπους, είναι φυσιολογικό το πρόβλημα της αφαίρεσης να έχει αντιμετωπιστεί στον τομέα της επιστήμης των υπολογιστών, ειδικά στην τεχνητή νοημοσύνη.

Ο βασικός λόγος, που η αφαίρεση, μειώνει την πολυπλοκότητα των προβλημάτων είναι ότι η συνολική πολυπλοκότητα ισούται με το άθροισμα των πολυπλοκοτήτων των ανεξάρτητων αναζητήσεων και όχι με το γινόμενο αυτών. Αυτή η μέθοδος έχει επαληθευτεί πειραματικά ότι είναι ιδιαίτερα αποτελεσματική στη μείωση του χώρου αναζήτησης σε σύνθετα προβλήματα. Για να αναλυθεί ευκολότερα η ιεραρχική αναζήτηση, υποθέτουμε αρχικά ότι η πολυπλοκότητα για έναν δοσμένο αλγόριθμο είναι $O(b^d)$. Με τη χρήση ενός επιπέδου αφαίρεσης η ιεραρχική αναζήτηση μπορεί να μειώσει την πολυπλοκότητα του αλγόριθμου αυτού σε $O(b^{d/2})$. Στην περίπτωση χρήσης πολλαπλών επιπέδων αφαίρεσης, η βέλτιστη ιεραρχία, για έναν χώρο με b^d καταστάσεις (στην περίπτωση μας, κόμβους), θα έχει lnb^d επίπεδα αφαίρεσης. Ο λόγος για αλλεπάλληλα επίπεδα αφαίρεσης είναι *e*. Αυτός ο τύπος ιεράρχησης μπορεί να μειώσει την πολυπλοκότητα του αλγόριθμου αναζήτησης από $O(b^d)$ σε $O(dlogb)$. Με άλλα λόγια, έχει αποδειχθεί ότι η βέλτιστη ιε ράρχηση έχει λογαριθμικό αριθμό επιπέδων με σταθερό λόγο και μειώνει την πολυπλοκότητα από εκθετική, ως προς τις καταστάσεις του χώρου, σε γραμμική. Ορισμένοι αλγόριθμοι, παρουσιάζουν προβληματική απόδοση επειδή ο αριθμός των κόμβων, που πρέπει να επεκταθούν σε ένα οδικό δίκτυο, συχνά αυξάνονται εκθετικά. Από την παραπάνω ανάλυση διαπιστώνουμε ότι τα πολλαπλά επίπεδα αφαίρεσης μπορούν να μειώσουν την εκθετική αύξηση σε γραμμική. Συνεπώς, είναι προφανές ότι η ιεραρχική αναζήτηση μπορεί δυνητικά να μειώσει τον χρόνο εκτέλεσης ενός αλγόριθμου αναζήτησης.

Για να εφαρμοστεί η ιεραρχική αναζήτηση και να απλοποιηθεί η διαδικασία σχεδιασμού της διαδρομής, πρέπει η βάση δεδομένων του χάρτη να κατασκευαστεί ιεραρχικά. Όσο ψηλότερα βρίσκεται το επίπεδο αφαίρεσης στη βάση δεδομένων τόσο λιγότερες λεπτομέρειες περιέχει, δηλαδή λιγότερους κόμβους και συνδέσμους. Η

αναζήτηση βέβαια σε ένα τέτοιο επίπεδο μπορεί να καταλήξει σε μη ολοκληρωμένη λύση (αλλά με μειωμένο χρόνο αναζήτησης) γι' αυτό είναι προφανές ότι κατά τη διάρκεια της αναζήτησης πρέπει να γίνεται εναλλαγή μεταξύ των διαφόρων επιπέδων. Η αποφυγή επιβάρυνσης κατά την εναλλαγή των επιπέδων και ο καθορισμός της χρονικής στιγμής της μετάβασης σε διαφορετικό επίπεδο, διαδραματίζουν σπουδαίο ρόλο στην αποτελεσματικότητα του αλγόριθμου [3].

Στην πράξη, είναι πολύ δύσκολο να κατασκευαστεί μια βάση δεδομένων ή να σχεδιαστεί ένας αλγόριθμος, ο οποίος να ικανοποιεί τις συνθήκες της βέλτιστης ιεραρχικής αφαίρεσης, που παρουσιάστηκε. Στην πλοήγηση οχημάτων, χρησιμοποιούνται κλάσεις δρόμων για τον καθορισμό μια ιεραρχίας στον σχεδιασμό της διαδρομής. Παρά τη δυσκολία επίτευξης της θεωρητικά βέλτιστης ιεραρχίας, η αμφίδρομη ευριστική ιεραρχική αναζήτηση αποτελεί τον πιο δημοφιλή αλγόριθμο για τον σχεδιασμό διαδρομής σε ένα μεγάλο οδικό δίκτυο.

2.6 Άλλοι αλγόριθμοι

Η χάραξη διαδρομής μπορεί επίσης να πραγματοποιηθεί με τη χρήση μιας μεθόδου «διαίρει και βασίλευε». Η βασική ιδέα έγκειται στη διαίρεση του οδικού δικτύου σε επιμέρους περιοχές και στον υπολογισμό εκ των προτέρων των βέλτιστων τοπικών διαδρομών για κάθε περιοχή. Στη συνέχεια οι τοπικές αυτές διαδρομές ενώνονται για τον σχηματισμό μιας ολοκληρωμένης διαδρομής από τον κόμβο αφετηρίας προς τον κόμβο προορισμού. Εκτός από τον υπολογισμό και την αποθήκευση, εκ των προτέρων, των τοπικών διαδρομών στο πρώτο στάδιο, στον αλγόριθμο πρέπει να προστεθεί και η ιδιότητα να καθορίζει τους κόμβους εισόδου και εξόδου των διαδρομών σε κάθε περιοχή και ακολούθως να τις ενώνει σε μία τελική λύση στο δεύτερο στάδιο. Επειδή όλες οι βέλτιστες διαδρομές σε μία περιοχή πρέπει να αποθηκευτούν εκ των προτέρων στη μνήμη, συνήθως γίνεται χρήση του αλγόριθμου Dijkstra για τον υπολογισμό τους. Για το δεύτερο στάδιο μπορεί να χρησιμοποιηθεί διαφορετικός αλγόριθμος. Ανάλογα με τον τρόπο, που το οδικό δίκτυο χωρίζεται σε περιοχές, και τον τρόπο, που η διαθέσιμη πληροφορία χρησιμοποιείται στον υπολογισμό των τοπικών διαδρομών, μπορεί να γίνει χρήση διάφορων τεχνικών για τη σύνθεση της τελικής λύσης. Αυτή η μέθοδος μπορεί επίσης να θεωρηθεί σαν μια προσέγγιση για την ιεραρχική ανακατασκευή του δικτύου και κατόπιν τον

υπολογισμό όλων των ζευγαριών κόμβου αφετηρίας και προορισμού στο δίκτυο. Η ιδέα αυτή μπορεί να επεκταθεί στον σχεδιασμό διαδρομής για πολλαπλά οχήματα σε ιεραρχικά δομημένες βάσεις δεδομένων πολλαπλών επιπέδων.

Οι αλγόριθμοι ευριστικής αναζήτησης πραγματικού χρόνου μπορούν να τροποποιηθούν έτσι ώστε σε ορισμένες περιπτώσεις να βρίσκουν μια απλά ικανοποιητική διαδρομή, σε αντίθεση με τον αλγόριθμο A*, που βρίσκει πάντα τη βέλτιστη λύση. Για ένα σύστημα πραγματικού χρόνου, ενσωματωμένο σε όχημα, με πολύ μεγάλη βάση δεδομένων, οι χρονικοί περιορισμοί συνήθως θέτουν ένα όριο στον αριθμό των κόμβων, που μπορούν να εξερευνηθούν. Για παράδειγμα, ένας υπολογιστής, διαδοχικής επεξεργασίας, ίσως είναι απαραίτητο να μεταβεί σε μία πιο κρίσιμη εργασία πριν μπορέσει να ολοκληρωθεί η αναζήτηση. Αυτό το γεγονός αποτελεί τον λεγόμενο *ορίζοντα αναζήτησης*. Στον αλγόριθμο ευριστικής αναζήτησης πραγματικού χρόνου ως ορίζοντας ορίζεται το βάθος αναζήτησης, που καθορίζεται από τον διαθέσιμο χρόνο για αναζήτηση στο οδικό δίκτυο. Ο περιορισμένος ορίζοντας αναζήτησης έχει σαν αποτέλεσμα η ευριστική αναζήτηση πραγματικού χρόνου να λαμβάνει αποφάσεις βασισμένες σε τοπικά βέλτιστες διαδρομές. Το βάθος της αναζήτησης από τον τρέχοντα κόμβο καθορίζεται από τους διαθέσιμους υπολογιστικούς πόρους, δηλαδή, την πληροφορία, η οποία παρέχεται από μία πρόσφατη ενημέρωση της δυναμικής βάσης δεδομένων (μπορεί να περιλαμβάνει και ανανεωμένες πληροφορίες για την κυκλοφορία), την υπολογιστική δύναμη της CPU ή τον χρόνο, που είναι διαθέσιμος για τον σχεδιασμό. Εάν διατεθεί μια ισχυρότερη CPU, το βάθος αναζήτησης μπορεί εύκολα να αυξηθεί. Συνεπώς, ο ορίζοντας αναζήτησης προσαρμόζεται σύμφωνα με το υπάρχον υλικό και λογισμικό. Εν αντιθέσει, κανένας από τους αλγόριθμους, που αναλύθηκαν σε αυτό το κεφάλαιο, δεν περιορίζει τον ορίζοντα αναζήτησης, έτσι ώστε να τιθεται κάποιο όριο στη χρήση της CPU ή της μνήμης. Αυτό μπορεί να προκαλέσει προβλήματα, όπως υπερβολικά εκτεταμένες αναζητήσεις. Για παράδειγμα, μια διαδικασία αναζήτησης ενδέχεται να αποτύχει να αναφέρει λύση εξαιτίας περιορισμένων πόρων, παρόλο, που υπάρχει τουλάχιστον μία λύση.

Οι αλγόριθμοι, που αναπτύχθηκαν για να επιλύουν το πρόβλημα του πλανόδιου πωλητή, μπορούν να χρησιμοποιηθούν για να σχεδιάσουν ένα ταξίδι με πολλές στάσεις. Παρόλο, που υπάρχει ένας μεγάλος αριθμός διαφοροποιήσεων, το βασικό πρόβλημα περιλαμβάνει ένα όχημα, το οποίο πρέπει να σταματήσει σε πολλαπλούς

ενδιάμεσους σταθμούς πριν φτάσει στον τελικό προορισμό του ή έναν αριθμό πελατών, οι οποίοι πρέπει να εξυπηρετηθούν από ένα όχημα ή από έναν στόλο οχημάτων από μία μόνο αποθήκη. Για κάποιες υπηρεσίες κάθε πελάτης μπορεί να απαιτήσει μία ορισμένη ποσότητα αγαθών. Υπάρχουν περιορισμοί στη χωρητικότητα του οχήματος, τη μέγιστη απόσταση, που μπορεί να διανύσει το συγκεκριμένο όχημα κτλ.. Εν συντομία, ο στόχος του αλγόριθμου είναι να βρει ένα σύνολο διαδρομών, το οποίο θα ελαχιστοποιεί το συνολικό κόστος ταξιδιού. Το πρόβλημα δρομολόγησης οχημάτων είναι γνωστό ως *NP-complete*, δηλαδή δεν υπάρχει αλγόριθμος πολυωνυμικής πολυπλοκότητας, που να το επιλύει. Ιστορικά, η έρευνα είχε επικεντρωθεί σε μεθόδους παραγωγής μίας εφικτής λύσης, η οποία μετά μπορούσε να βελτιωθεί με τη χρήση τοπικής αναζήτησης. Πρόσφατα, δημοφιλείς έγιναν οι γενετικοί αλγόριθμοι, *tabu searches* και *robust* αλγόριθμοι. Κάποιοι από αυτούς έχουν επεκταθεί για να συμπεριλάβουν και προβλήματα δρομολόγησης πολλαπλών οχημάτων.

Τα τελευταία χρόνια, έχει αυξηθεί το ενδιαφέρον για δυναμικούς *on-line* αλγόριθμους επίλυσης του προβλήματος συντομότερης διαδρομής. Οι αλγόριθμοι αυτοί μπορούν να τροποποιηθούν για να χρησιμοποιηθούν στη δρομολόγηση οχημάτων. Ο κύριος στόχος τους είναι ο σχεδιασμός αποτελεσματικών δομών δεδομένων και προσφέρουν γρήγορες λύσεις, που μπορούν να λειτουργήσουν σε δυναμικά περιβάλλοντα, όπου, για παράδειγμα, τα δεδομένα εισόδου μεταβάλλονται. Εν αντιθέσει, οι αλγόριθμοι, που συζητήθηκαν ως τώρα, είναι ικανοί να διαχειριστούν μόνο στατικά δεδομένα. Λαμβάνοντας υπόψη τις δυνατότητες αυτών των δυναμικών αλγορίθμων, δε πρέπει να είναι δύσκολο να προσαρμοστούν για τη δυναμική πλοήγηση οχημάτων, όταν οι συνθήκες οδήγησης (τα βάρη των τμημάτων των δρόμων) ενημερώνονται είτε από ένα κέντρο πληροφόρησης για την κυκλοφορία είτε από ένα κέντρο διαχείρισης της κυκλοφορίας [3].

3 Νευρωνικά Δίκτυα

Με τον όρο *τεχνητό νευρωνικό δίκτυο* (ΤΝΔ) εννοούμε κάθε αρχιτεκτονική υπολογισμού, η οποία περιλαμβάνει έναν μεγάλο αριθμό διασυνδεδεμένων νευρωνικών επεξεργαστών και απομιμείται τη λειτουργία και τις ιδιότητες του ανθρώπινου εγκεφάλου. Συνεπώς, ένα τεχνητό νευρωνικό δίκτυο ή απλά ένα νευρωνικό δίκτυο (ΝΔ) έχει την ικανότητα να μαθαίνει από εμπειρίες, να γενικεύει την υπάρχουσα γνώση και να εκτελεί λογικές αφαιρέσεις [8].

Τα ΝΔ είναι συστήματα μεγάλης κλίμακας, τα οποία περιέχουν έναν μεγάλο αριθμό μη γραμμικών επεξεργαστών ειδικού τύπου, τα *νευρώνια*. Κάθε ΝΔ χαρακτηρίζεται από μία κατάσταση, ένα σύνολο εισόδων με βάρη, που προέρχονται από άλλα νευρώνια, και μία εξίσωση, η οποία περιγράφει τη δυναμική λειτουργία του. Τα βάρη του ΝΔ ανανεώνονται μέσω μιας διαδικασίας μάθησης (εκπαίδευσης), η οποία πραγματοποιείται με την ελαχιστοποίηση κάποιας συνάρτησης κόστους. Οι βέλτιστες τιμές των βαρών αποθηκεύονται ως δυνάμεις μεταξύ των νευρωνίων και χρησιμοποιούνται κατά την εκτέλεση της εργασίας για την οποία προορίζεται το ΝΔ.

Τα ΝΔ εκτελούν μη συμβολική επεξεργασία πληροφορίας, η οποία βασίζεται στη λειτουργία του ανθρώπινου εγκεφάλου και επικαλούνται την ιδέα της μοντελοποίησης *μαύρου – κουτιού* χρησιμοποιώντας μοντέλα, που εμπνέονται από τη βιολογία και τη νευροφυσιολογία. Για τη χρήση των μοντέλων αυτών διατίθενται μέθοδοι (εργαλεία), που υλοποιούν πολύπλοκες συναρτήσεις και λειτουργίες. Η

εφαρμογή των μεθόδων αυτών δεν απαιτεί ρητή γνώση σε αντίθεση με ότι ισχύει κατά την εφαρμογή συμβολικών μεθόδων τεχνητής νοημοσύνης, οι οποίες στηρίζονται στη λογική. Στα συμβολικά συστήματα της τεχνητής νοημοσύνης η γνώση αναπαριστάται ρητά με κανόνες παραγωγής. Στη μη συμβολική προσέγγιση δε δίνεται η υπό εξέταση σχέση ρητά αλλά, κωδικοποιείται στη δομή του ΝΔ.

Σκοπός του κεφαλαίου αυτού είναι να παρουσιάσει συνοπτικά τις βασικές έννοιες των ΝΔ αρχίζοντας με μία σύντομη ιστορική αναδρομή. Κατόπιν, αναλύονται τα μοντέλα βιολογικού και τεχνητού νευρωνίου, οι αρχιτεκτονικές των νευρωνικών δικτύων και οι τύποι της νευρωνικής μάθησης.

3.1 Ιστορική αναδρομή

Το ανθρώπινο μυαλό αποτελεί αντικείμενο έρευνας εδώ και χιλιάδες χρόνια. Με την ανάπτυξη όμως των σύγχρονων ηλεκτρονικών ο άνθρωπος προσπάθησε να μιμηθεί τον ανθρώπινο εγκέφαλο και τις νοητικές του διεργασίες. Η πρώτη αναφορά στα νευρωνικά δίκτυα χρονολογείται το 1936, όταν ο *Alan Turing*, ένας Βρετανός μαθηματικός, διατύπωσε τις αρχές για τον προσδιορισμό του τι είναι και τι δεν είναι υπολογίσιμο και έγραψε για τον τρόπο, που πραγματοποιεί ο ανθρώπινος εγκέφαλος υπολογισμούς με τη χρήση των νευρώνων [9]. Ένας ακόμα μαθηματικός, ο *John von Neumann*, επίσης αναφέρθηκε στα νευρωνικά δίκτυα, ως έναν τρόπο πραγματοποίησης υπολογισμών, ο οποίος προκύπτει από τη δομή του δικτύου αυτή καθαυτή.

Ο *Norbert Wiener*, το όνομα του οποίου συνδέεται με τον ορισμό της κυβερνητικής, σκέφτηκε ότι η βασική δομή για την κατανόηση των μαθηματικών υπολογισμών τόσο στον ανθρώπινο εγκέφαλο, όσο και στη μηχανή, είναι ένας συνδυασμός λογικής και δικτυακής δομής. Όμως, λόγω του ότι η αλγοριθμική μέθοδος ως τότε θεωρούταν αποτελεσματική, η θεωρία για τα νευρωνικά δίκτυα έμεινε αδρανής για μεγάλο χρονικό διάστημα [10].

Το πρώτο βήμα για την ανάπτυξη των ΝΔ έγινε το 1943, από τον νευροφυσιολόγο *Warren McCulloch* και τον μαθηματικό *Walter Pitts*, οι οποίοι πρότειναν ένα μοντέλο για το θεμελιώδες κύτταρο του εγκεφάλου, τον νευρώνα, και κατασκεύασαν ένα πρωταρχικό ΝΔ, χρησιμοποιώντας απλά ηλεκτρικά κυκλώματα, αποδεικνύοντας έτσι

ότι ένα δίκτυο νευρώνων μπορεί να πραγματοποιήσει υπολογισμούς, με άλλα λόγια, να μιμηθεί οποιαδήποτε υπολογιστική μηχανή [11].

Άλλο ένα σημαντικό βήμα για την εξέλιξη των ΝΔ πραγματοποιήθηκε το 1949 από τον *Donald Hebb* με τη δημοσίευση του βιβλίου του *«The Organization of Behaviour*» [12], στο οποίο προτείνει έναν συγκεκριμένο μηχανισμό μάθησης για τα βιολογικά νευρωνικά δίκτυα. Σύμφωνα με τον *Hebb,* η μάθηση πραγματοποιείται μέσω της τροποποίησης των βαρών των συναπτικών συνδέσεων μεταξύ των νευρώνων, ούτως ώστε όταν δύο νευρώνες τείνουν να ενεργοποιηθούν ταυτόχρονα, το βάρος της μεταξύ τους σύναψης πρέπει να αυξηθεί. Αυτός ο κανόνας μάθησης αποτελεί ακόμα τη βάση για την εκπαίδευση ορισμένων απλών ΝΔ.

Το πρώτο νευρωνικό δίκτυο, γνωστό ως *Perceptron,* κατασκευάστηκε το 1958 από τον *Frank Rosenblatt* και βασιζόταν στο νευρωνικό μοντέλο των *McCulloch – Pitts*. Αποτελείτο από έναν πίνακα φωτουποδοχέων, οι οποίες δρούσαν ως εξωτερικές είσοδοι, και χρησιμοποιούσε μηχανοκίνητα ποτενσιόμετρα για να παρέχει προσαρμοζόμενες συναπτικές συνδέσεις, που είχαν τη δυνατότητα να διατηρήσουν μία γνωστή ρύθμιση. Σε πολλές περιπτώσεις το *Perceptron* μπορούσε να εκπαιδευτεί κατάλληλα, ώστε να διακρίνει χαρακτήρες και σχήματα, τα οποία παρουσιάζονταν στην είσοδο. Παρόμοια δίκτυα μελετήθηκαν από τον *Widrow*, ο οποίος ανέπτυξε το *ADALINE (ADAptive LINear Element*) και μία αντίστοιχη διαδικασία εκμάθησης, τον κανόνα μάθησης *Widrow – Hoff*, ο οποίος χρησιμοποιείται ακόμα και σήμερα για την ακύρωση της ηχούς σε τηλεφωνικά καλώδια μεγάλων αποστάσεων [13].

Τη δεκαετία του ’70 υπήρξε μεγάλη ερευνητική δραστηριότητα πάνω στα νευρωνικά δίκτυα, μεγάλο μέρος της οποίας χαρακτηριζόταν από έλλειψη αυστηρότητας, περίσσια αλχημεία, καθώς και υπερβολικές απαιτήσεις σε σχέση με τις δυνατότητες και τη βραχυπρόθεσμη προοπτική της τεχνολογίας. Παρά τις αρχικές επιτυχίες, η δυναμική στον τομέα άρχισε να υποχωρεί προς το τέλος της δεκαετίας, καθώς προέκυψε μια σειρά δύσκολων προβλημάτων, τα οποία δεν μπορούσαν να επιλυθούν με τους διαθέσιμους αλγόριθμους. Επιπρόσθετα, τα νευρωνικά δίκτυα δέχτηκαν σκληρή κριτική, με επίκεντρο το βιβλίο «*Perceptrons*» [14], των *Minsky* και *Papert*, από υποστηρικτές του τομέα της Τεχνητής Νοημοσύνης, που προσπαθούσε να δώσει λύσεις για την αναγνώριση προτύπων και σε παρόμοια προβλήματα με τη χρήση ρητών κανόνων. Η κριτική στόχευε σε μία κατηγορία προβλημάτων, τα οποία ονομάζονται *γραμμικά μη – διαχωρίσιμα* και δε μπορούσαν να λυθούν με νευρωνικά

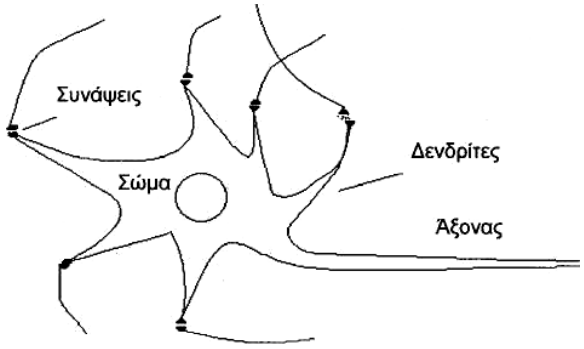
δίκτυα, όπως το *Perceptron* και το *ADALINE*, με αποτέλεσμα να χαθεί το ενδιαφέρον προς αυτά, ενώ ελάχιστοι ερευνητές παρέμειναν ενεργοί.

Τη δεκαετία του 1980 το ενδιαφέρον για τα νευρωνικά δίκτυα αναζωπυρώθηκε κυρίως λόγω της θεωρίας του *John Hopfield* ότι τα νευρωνικά δίκτυα μπορούν να χρησιμοποιηθούν για την κατασκευή προηγμένων μνημών για τους υπολογιστές. Ένας δεύτερος λόγος ήταν η ανακάλυψη του αλγόριθμου *ανάστροφης μετάδοσης λάθους (back propagation algorithm),* ο οποίος ξεπέρασε τους βασικούς περιορισμούς των προηγούμενων ΝΔ, όπως το απλό *Perceptron* [13].

Σήμερα, τα νευρωνικά δίκτυα χρησιμοποιούνται ήδη με επιτυχία σε ένα ευρύ φάσμα εφαρμογών, ενώ ταυτόχρονα λειτουργούν πολλά ερευνητικά προγράμματα για την περαιτέρω ανάπτυξη τους.

3.2 Μοντέλο βιολογικού νευρώνα

Η ικανότητα του ανθρώπου να σκέφτεται, να θυμάται και να επιλύει προβλήματα εντοπίζεται στον εγκέφαλό του. Όπως είναι γνωστό από τη Βιολογία, η δομική μονάδα του εγκεφάλου είναι ο *νευρώνας.* Ένας τυπικός *βιολογικός νευρώνας* (σχήμα 3.1) αποτελείται από το *σώμα,* που αποτελεί τον πυρήνα του, τους *δενδρίτες* μέσω των οποίων λαμβάνει σήματα από γειτονικούς νευρώνες (σημεία εισόδου) και τον *άξονα,* που είναι η έξοδος του νευρώνα και το μέσο σύνδεσής του με άλλους νευρώνες. Σε κάθε δενδρίτη υπάρχει ένα απειροελάχιστο κενό, που ονομάζεται *σύναψη.* Οι *συνάψεις* μέσω χημικών διαδικασιών επιταχύνουν ή επιβραδύνουν τη ροή ηλεκτρικών φορτίων προς το σώμα του νευρώνα. Η ικανότητα μάθησης και μνήμης, που παρουσιάζει ο εγκέφαλος οφείλεται στην ικανότητα των συνάψεων να μεταβάλουν την αγωγιμότητά τους. Τα ηλεκτρικά σήματα, που εισέρχονται στο σώμα μέσω των δενδριτών, συνδυάζονται και, αν το αποτέλεσμα ξεπερνά κάποια τιμή κατωφλίου, το σήμα διαδίδεται, με τη βοήθεια του άξονα, προς τους άλλους νευρώνες.



Σχήμα 3.1: Αναπαράσταση βιολογικού νευρώνα.

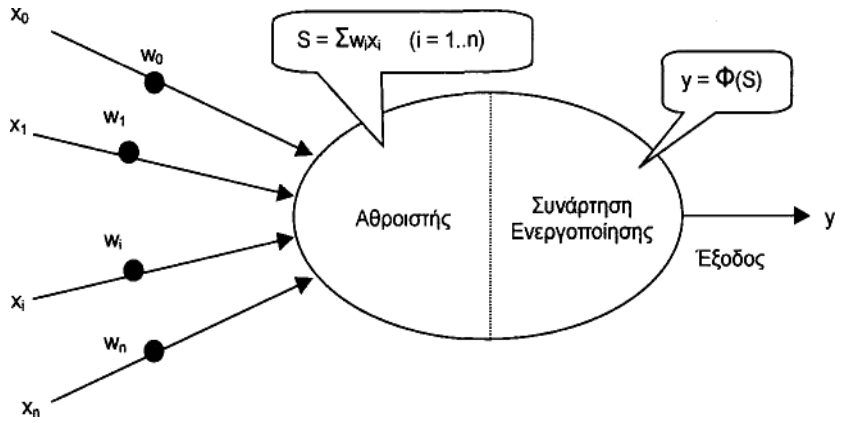
Ο εγκέφαλος ενός νεογέννητου ανθρώπου αποτελείται από περίπου 100 δισεκατομμύρια νευρώνες, κάθε ένας από τους οποίους συνδέεται, κατά μέσο όρο, με άλλους 1000. Αυτό πραγματοποιείται μέσω του άξονα κάθε νευρώνα, στον οποίο καταλήγουν ισάριθμοι δενδρίτες άλλων νευρώνων. Δεδομένου ότι κάθε τέτοια σύνδεση περιλαμβάνει και μία σύναψη, προκύπτει ότι υπάρχουν περίπου 100 τρισεκατομμύρια συνάψεις, οι οποίες και επηρεάζουν τη λειτουργία του εγκεφάλου. Είναι προφανές ότι κάθε προσπάθεια να αντιγραφεί η δομή και η λειτουργία του εγκεφάλου σε τέτοια κλίμακα είναι αδύνατη. Στην πραγματικότητα, τα μοντέλα, τα οποία κατασκευάζονται, περιλαμβάνουν μερικές χιλιάδες τεχνητούς νευρώνες, έχουν το πολύ ένα εκατομμύριο τεχνητές συνάψεις και παρουσιάζουν πολύ περιορισμένη λειτουργικότητα.

Αν και ο χρόνος απόκρισης των βιολογικών νευρώνων είναι της τάξης των χιλιοστών του δευτερολέπτου (*msec*), ο εγκέφαλος είναι σε θέση να λαμβάνει πολύπλοκες αποφάσεις, εκπληκτικά γρήγορα. Κατά μία άποψη, αυτό οφείλεται στο ότι η υπολογιστική ικανότητα του εγκεφάλου, καθώς και η πληροφορία, που περιέχει, είναι διαμοιρασμένα σε όλο του τον όγκο. Πρόκειται δηλαδή για ένα παράλληλο και, κατανεμημένο υπολογιστικό σύστημα. Αυτά τα χαρακτηριστικά είναι και το κυριότερο κίνητρο πίσω από την επιθυμία να μοντελοποιηθεί ο ανθρώπινος εγκέφαλος με τα *τεχνητά νευρωνικά δίκτυα* [15].

3.3 Μοντέλο Τεχνητού Νευρώνα

Ο *τεχνητός νευρώνας (artificial neuron)* είναι ένα υπολογιστικό μοντέλο, τα μέρη του οποίου αντιστοιχίζονται άμεσα με αυτά του βιολογικού νευρώνα. Όπως απεικονίζεται στο σχήμα 3.2, ένας τεχνητός νευρώνας δέχεται κάποια *σήματα εισόδου* $x_0, x_1, \dots, x_n$ τα οποία, σε αντίθεση με τους ηλεκτρικούς παλμούς του εγκεφάλου, αντιστοιχούν σε

συνεχείς μεταβλητές. Κάθε τέτοιο σήμα εισόδου μεταβάλλεται από μια *τιμή βάρους* w_i *(weight)* ο ρόλος της οποίας είναι αντίστοιχος της σύναψης του βιολογικού εγκεφάλου. Η τιμή βάρους μπορεί να είναι θετική ή αρνητική, σε αντιστοιχία με την επιταχυντική ή επιβραδυντική λειτουργία της σύναψης. Το σώμα του τεχνητού νευρώνα χωρίζεται σε δύο μέρη, τον *αθροιστή* (*sum*), ο οποίος προσθέτει τα επηρεασμένα από τα βάρη σήματα εισόδου και παράγει την ποσότητα S, και τη *συνάρτηση ενεργοποίησης* ή *κατωφλίου* (*activation* ή *threshold function*), ένα μη γραμμικό φίλτρο το οποίο διαμορφώνει την τελική τιμή του σήματος εξόδου *y,* συναρτήσει της ποσότητας S.



Σχήμα 3.2: Μοντέλο τεχνητού νευρώνα.

Υπάρχουν τρεις τυπικές περιπτώσεις για τη συνάρτηση ενεργοποίησης :

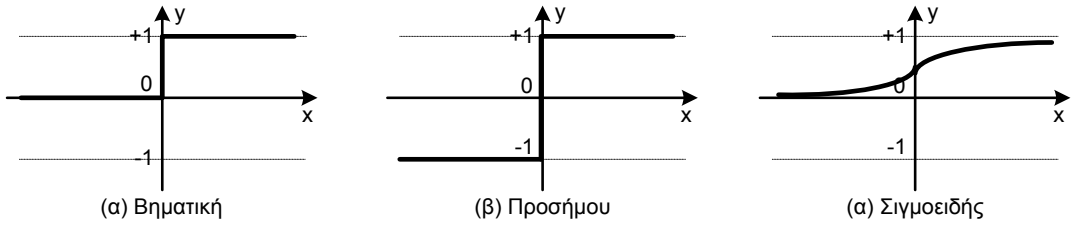
- Η *βηματική (step)* συνάρτηση (σχήμα 3.3α), η οποία δίνει στην έξοδο αποτέλεσμα (συνήθως 1) μόνο αν η τιμή, που υπολογίζει ο αθροιστής είναι μεγαλύτερη από μία τιμή κατωφλίου Τ.

- Η συνάρτηση *πρόσημου (sign)* (σχήμα 3.3β), η οποία δίνει στην έξοδο αρνητική (ή θετική) πληροφορία αν η τιμή, που υπολογίζει ο αθροιστής είναι μικρότερη (ή μεγαλύτερη) από μία τιμή κατωφλίου Τ.

- Η *σιγμοειδής (sigmoid)* συνάρτηση η οποία εκφράζεται από τη γενική σχέση:

$$\Phi(S) = \frac{1}{1+e^{-aS}} \qquad (3.1)$$

όπου a είναι ένας συντελεστής ρύθμισης της ταχύτητας μετάβασης μεταξύ των δύο ασυμπτωτικών τιμών. Η σιγμοειδής συνάρτηση (σχήμα 3.3γ) είναι ιδιαίτερα σημαντική γιατί παρέχει μη γραμμικότητα στο ν νευρώνα, κάτι που είναι απαραίτητο στη μοντελοποίηση μη γραμμικών φαινομένων.



Σχήμα 3.3 *Γραφικές παραστάσεις συναρτήσεων ενεργοποίησης.*

Οι τεχνητοί νευρώνες δίνουν τη δυνατότητα υλοποίησης απλών αλγεβρικών συναρτήσεων, καθώς και των λογικών συναρτήσεων AND, OR και NOT. Για παράδειγμα, στην υλοποίηση του NOT χρησιμοποιείται ως συνάρτηση ενεργοποίησης η βηματική με κατώφλι $T = -0.5$. Οι τιμές εισόδου μπορούν να κυμαίνονται από 0 (ψευδές) έως 1 (αληθές). Αν η είσοδος του νευρώνα είναι 0, τότε πολλαπλασιαζόμενη με το βάρος $w = -1$ δίνει $S = 0$. Η τιμή αυτή ξεπερνά το κατώφλι του -0.5 οπότε στην έξοδο παράγεται 1. Στην περίπτωση, που η τιμή εισόδου είναι 1, τότε $S = -1$, τιμή η οποία βρίσκεται κάτω του κατωφλίου , με αποτέλεσμα να παράγεται στην έξοδο 0 [15].

3.4 Βασικές ιδιότητες

Υπάρχουν τέσσερεις ιδιότητες, που είναι άρρηκτα συνδεδεμένες με τα Τεχνητά Νευρωνικά Δίκτυα (ΤΝΔ). Αυτές είναι:

- Η ικανότητά τους να μαθαίνουν μέσω παραδειγμάτων (*learn by example*).
- Η δυνατότητα θεώρησής τους ως κατανεμημένη μνήμη (*distributed memory*) και ως μνήμη συσχέτισης (*associative memory*).
- Η μεγάλη τους ανοχή σε σφάλματα (*fault-tolerant*).
- Η ικανότητά τους για αναγνώριση προτύπων (*pattern recognition*).

Αν και τα ΝΔ δεν είναι τα μόνα συστήματα με ικανότητα μάθησης μέσω παραδειγμάτων, εντούτοις διακρίνονται για την ικανότητά τους να οργανώνουν την πληροφορία των δεδομένων εισόδου σε χρήσιμες μορφές. Αυτές οι μορφές αποτελούν στην ουσία ένα μοντέλο, το οποίο αναπαριστά τη σχέση, που ισχύει, μεταξύ των δεδομένων εισόδου και εξόδου.

Ο χαρακτηρισμός των ΝΔ ως κατανεμημένη μνήμη πηγάζει από το ότι η πληροφορία, που κωδικοποιούν, είναι κατανεμημένη σε όλα τα βάρη της συνδεσμολογίας τους. Για τον ίδιο λόγο τα ΝΔ χαρακτηρίζονται και ως μνήμες συσχέτισης. Μια μνήμη συσχέτισης αποθηκεύει πληροφορία συσχετίζοντας αποθηκευμένα δεδομένα μεταξύ τους. Η ανάκληση (*recall*) της πληροφορίας γίνεται με βάση το περιεχόμενο και όχι τη διεύθυνση, όπως δηλαδή συμβαίνει και με τον ανθρώπινο εγκέφαλο. Η παραπάνω οργάνωση κάνει ορισμένα είδη ΝΔ να είναι πολύ ανεκτικά σε μικροαλλαγές στα σήματα εισόδου, δηλαδή είναι σε θέση να παράγουν τη σωστή έξοδο ακόμη κι αν τα δεδομένα εισόδου είναι λίγο διαφορετικά ή και ελλιπή.

Τα ΝΔ έχουν μεγάλη ανοχή σε δομικά σφάλματα. Αυτό σημαίνει ότι η κακή λειτουργία ή η καταστροφή ενός νευρώνα ή κάποιων συνδέσεων δεν είναι ικανή να διαταράξει σημαντικά τη λειτουργία τους, καθώς, όπως αναφέρθηκε, η πληροφορία, που εσωκλείουν, δεν είναι εντοπισμένη σε συγκεκριμένο σημείο αλλά διάχυτη σε όλο το δίκτυο. Γενικά, το μέγεθος του σφάλματος λόγω «*δομικών αστοχιών*» είναι ανάλογο του ποσοστού των κατεστραμμένων συνδέσεων. Αυτό κάνει τα ΝΔ, και συγκεκριμένα τις υλοποιήσεις τους σε κύκλωμα, ιδανικά για χρήση σε αυτοματισμούς, που θα λειτουργήσουν σε αντίξοες συνθήκες (π.χ. διάστημα, χώρους με ραδιενέργεια, πεδίο μάχης κτλ).

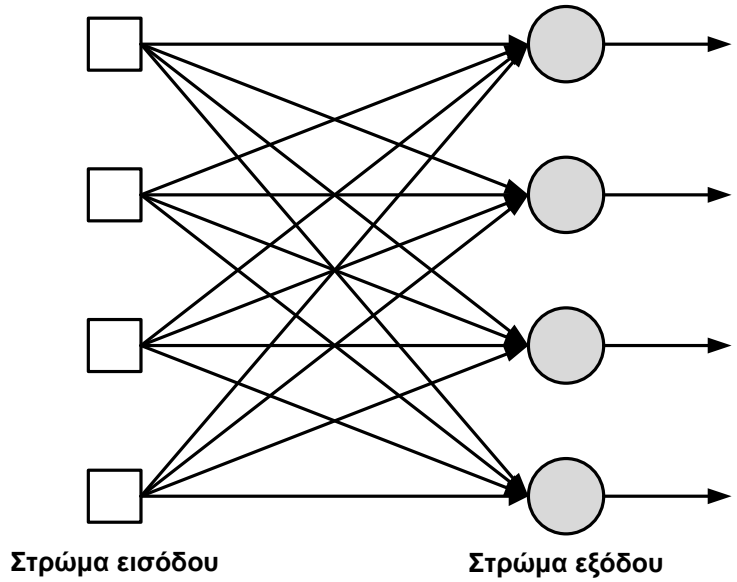
Τέλος, τα ΝΔ έχουν εξαιρετική ικανότητα αναγνώρισης προτύπων, καθώς δεν επηρεάζονται από ελλιπή ή και με θόρυβο δεδομένα. Από τη στιγμή, που ένα ΝΔ εκπαιδευτεί στο να αναγνωρίζει συνθήκες και καταστάσεις, απαιτείται ένας μόνο κύκλος λειτουργίας για να προσδιορίσει μία συγκεκριμένη κατάσταση [15].

3.5 Αρχιτεκτονικές Νευρωνικών δικτύων

Η τοπολογική δομή είναι το κύριο χαρακτηριστικό των ΝΔ και αναφέρεται στην αρχιτεκτονική, στην οποία διευθετούνται τα πολλαπλά νευρώνια. Οι δύο βασικές ιδιότητες, που καθορίζουν την αρχιτεκτονική ενός ΝΔ, είναι το πλήθος των στρωμάτων (*layers)* και οι συνδέσεις ανάμεσα στα νευρώνια.

3.5.1 Νευρωνικά δίκτυα προσοτροφοδότησης

Στην απλούστερη περίπτωση ένα διαστρωματωμένο ΝΔ έχει ένα στρώμα εισόδου από *κόμβους πηγής (source nodes),* το οποίο προβάλλεται πάνω σε ένα στρώμα νευρωνίων εξόδου (*κόμβων υπολογισμού),* αλλά όχι αντίστροφα. Ένα τέτοιο ΝΔ είναι αυστηρά τύπου *προσοτροφοδότησης (feed forward)* και καλείται *μονοστρωματικό ΝΔ προσοτροφοδότησης (*σχήμα 3.4*)*, όπου το μοναδικό στρώμα είναι το στρώμα νευρωνίων εξόδου. Αυτό σημαίνει ότι το στρώμα των κόμβων εισόδου δεν προσμετράται, γιατί δεν λαμβάνει χώρα κανένας υπολογισμός σε αυτό.



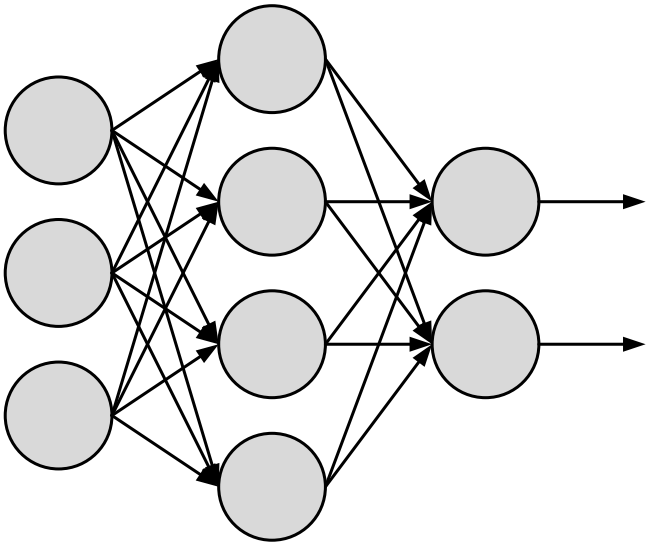
Σχήμα 3.4: Μονοστρωματικό ΝΔ προσοτροφοδότησης

Ένα τυπικό παράδειγμα μονοστρωματικού ΝΔ προσοτροφοδότησης είναι η *γραμμική συσχετιστική μνήμη,* η οποία αντιστοιχεί (συσχετίζει) ένα διάνυσμα (πρότυπο) εξόδου σε ένα διάνυσμα εισόδου και αποθηκεύει την πληροφορία ως αλλαγές στα συναπτικά βάρη.

Στη γενική περίπτωση, ένα ΝΔ προσοτροφοδότησης περιέχει ένα η περισσότερα *κρυμμένα (hidden)* στρώματα, των οποίων οι υπολογιστικοί κόμβοι είναι γνωστοί ως *κρυμμένα νευρώνια* και παρεμβαίνουν μεταξύ των εξωτερικών εισόδων και εξόδων. Στα δίκτυα αυτά, που ονομάζονται *πολυστρωματικά ΝΔ προσοτροφοδότησης* οι κόμβοι πηγής στο στρώμα εισόδου παρέχουν τα στοιχεία του προτύπου δράσης (εισόδου), τα οποία εισέρχονται ως είσοδοι στο πρώτο κρυμμένο στρώμα υπολογιστικών κόμβων. Ομοίως, οι έξοδοι του πρώτου κρυμμένου στρώματος

εισέρχονται ως είσοδοι στους κόμβους του δεύτερου κρυμμένου στρώματος κ.ο.κ.. Το τελικό στρώμα κόμβων (στρώμα εξόδου) δίνει τη συνολική απόκριση στα διανύσματα εισόδου, δηλαδή στα πρότυπα εξωτερικής δράσης.

Ένα παράδειγμα ΝΔ προσοτροφοδότησης με ένα κρυμμένο στρώμα τεσσάρων κόμβων, τρεις κόμβους πηγής στο στρώμα εισόδου και δύο κόμβους στο στρώμα εξόδου εικονίζεται στο σχήμα 3.5. Το δίκτυο αυτό αναφέρεται συμβολικά ως ΝΔ προσοτροφοδότησης 3-4-2.



Σχήμα 3.5: Πολυστρωματικό ΝΔ προσοτροφοδότησης 3-4-2 (με ένα κρυμμένο στρώμα)

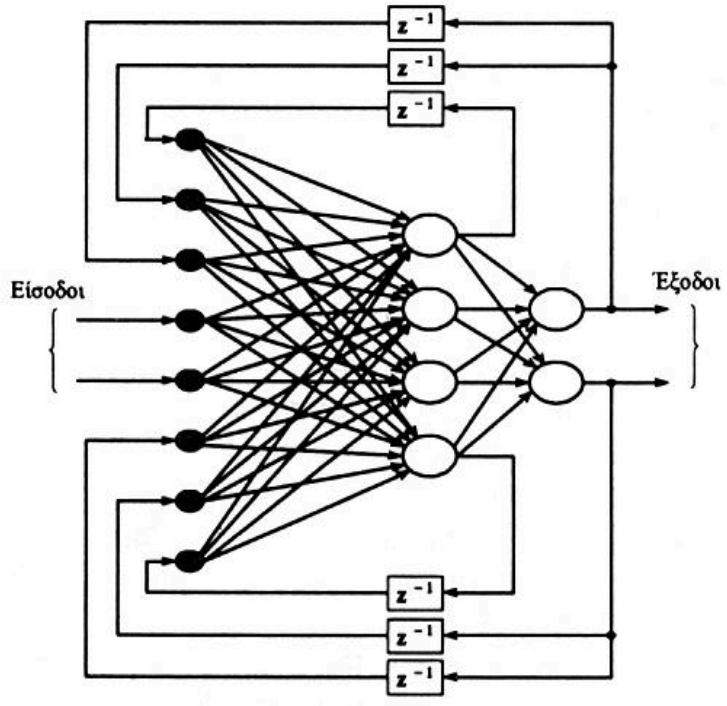
Το νευρωνικό δίκτυο του σχήματος 3.5 ονομάζεται *πλήρως διασυνδεδεμένο* γιατί κάθε κόμβος, οποιουδήποτε στρώματος, συνδέεται με όλους τους κόμβους του επόμενου στρώματος. Εάν αυτό δεν ισχύει, δηλαδή εάν λείπουν μερικοί σύνδεσμοι επικοινωνίας (συναπτικές συνδέσεις), τότε το ΝΔ χαρακτηρίζεται *μερικώς διασυνδεδεμένο* [8].

3.5.2 Νευρωνικά δίκτυα ανατροφοδότησης

Εάν το ΝΔ περιέχει τουλάχιστον έναν βρόχο ανατροφοδότησης, ο οποίος ανακυκλώνει πληροφορία μέσω του ιδίου ή προηγούμενων στρωμάτων, τότε ονομάζεται *αναδρομικό ΝΔ* ή *ΝΔ ανατροφοδότησης*. Το αποτέλεσμα της ανατροφοδότησης είναι ότι όταν ένα διάνυσμα (πρότυπο) εισόδου εισέρχεται στο

αναδρομικό ΝΔ, δεν παράγει ένα πρότυπο εξόδου σε πεπερασμένο αριθμό χρονικών βημάτων, αλλά δρα με κυκλικό τρόπο, όπου τα ίδια στρώματα ενεργοποιούνται επαναληπτικά. Εάν το ΝΔ είναι αφ' εαυτού ευσταθές, πιθανώς να ταλαντωθεί για κάποιο χρονικό διάστημα προτού φτάσει σε μία σταθερή κατάσταση, στην οποία οι νευρωνικές ενεργοποιήσεις θα σταματήσουν να αλλάζουν με αποτέλεσμα να παραχθεί μια σταθερή έξοδος. Διαφορετικά, εάν το ΝΔ δεν είναι ευσταθές, οι ταλαντώσεις θα συνεχίσουν αδιάκοπα. Συνεπώς, όταν εκπαιδεύουμε ένα αναδρομικό ΝΔ είναι σημαντικό να βρούμε το σύνολο των συναπτικών βαρών, που του επιτρέπουν να σταθεροποιηθεί στις επιθυμητές τιμές εξόδου.

Ένα παράδειγμα μονοστρωματικού αναδρομικού ΝΔ απεικονίζεται στο σχήμα 3.6. Οι βρόχοι ανατροφοδότησης κλείνουν μέσω μοναδιαίων καθυστερήσεων, οι οποίες συμβολίζονται με z^{-1}, όπου $z^{-1} \cdot y(k) = y(k-1)$ και k παριστά διακριτό χρόνο. Στη θεωρία συστημάτων το z^{-1} ονομάζεται τελεστής μοναδιαίας καθυστέρησης [8].



Σχήμα 3.6. Πλήρως διασυνδεδεμένο ΝΔ ανατροφοδότησης

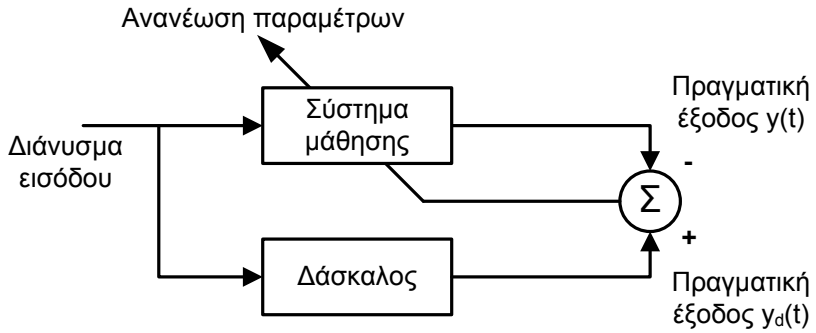
3.6 Μάθηση των Νευρωνικών Δικτύων

Τα ΝΔ πραγματοποιούν δύο βασικές λειτουργίες: τη μάθηση (*learning*) και την ανάκληση (*recall*). Μάθηση είναι η διαδικασία τροποποίησης της τιμής των

βαρών του δικτύου , ώστε δοθέντος συγκεκριμένου διανύσματος εισόδου να παραχθεί συγκεκρι μένο διάνυσμα εξόδου . Η διαδικασία αυτή ονομάζεται επίσης και *εκπαίδευση (training)*. *Ανάκληση (recall)* είναι η διαδικασία υπολογισμού ενός διανύσματος εξόδου για συγκεκριμένο διάνυσμα εισόδου και συγκεκριμένες τιμές βαρών [15]. Ο γενικός τρόπος , με τον οποίο γίνεται η τροποποίηση των βαρών ενός ΝΔ κατά την εκπαίδευσή του , επιτρέπει τη διάκριση τριών ειδών μάθησης, που παρουσιάζονται παρακάτω.

3.6.1 Επιβλεπόμενη μάθηση

Η δομή της επιβλεπόμενης ή ενεργού μάθησης (*supervised learning*) έχει τη γενική μορφή του σχήματος 3.7 και περιλαμβάνει δύο κύριες συνιστώσες, τον δάσκαλο και το σύστημα μάθησης.



Σχήμα 3.7: Δομή της επιβλεπόμενης μάθησης

Το κύριο χαρακτηριστικό της επιβλεπόμενης μάθησης είναι η ύπαρξη του εξωτερικού δασκάλου, ο οποίος με βάση τη γνώση και εμπειρία του είναι ικανός να διδάξει στο ΝΔ τις επιθυμητές εξόδους για ένα σύνολο δεδομένων εκπαίδευσης. Όταν ο δάσκαλος και το ΝΔ λαμβάνουν ένα διάνυσμα εισόδου εκπαίδευσης, ο δάσκαλος δίνει στο ΝΔ μια επιθυμητή έξοδο, η οποία αναπαριστά τη βέλτιστη ενέργεια (δράση), που πρέπει να κάνει το ΝΔ. Οι παράμετροι του ΝΔ ανανεώνονται με βάση τόσο το διάνυσμα εκπαίδευσης όσο και τ ου σήματος σφάλματος, δηλαδή της διαφοράς μεταξύ της πραγματικής και επιθυμητής απόκρισης [8].

3.6.2 Ενισχυτική μάθηση

Στον τύπο αυτό της μάθησης το ΝΔ τροφοδοτείται και πάλι με δειγματικά πρότυπα εισόδου, χωρίς να τροφοδοτείται όμως με τις επιθυμητές αποκρίσεις σε αυτές τις εισόδους. Εν αντιθέσει, χρησιμοποιείται ένα συνολικό μέτρο επάρκειας της προκύπτουσας απόκρισης (*δείκτης συμπεριφοράς*), το οποίο μπορεί να οδηγήσει το δίκτυο στην επιθυμητή συμπεριφορά. Το μέτρο αυτό είναι γνωστό ως ενισχυτικό σήμα (*reinforcement signal*) και ανατροφοδοτείται στο ΝΔ έτσι ώστε να επιβραβεύσει τις ορθές και να τιμωρεί τις λανθασμένες συμπεριφορές.

Η διαφορά ανάμεσα στην ενισχυτική και την επιβλεπόμενη μάθηση είναι ότι στην πρώτη το σύστημα βελτιώνεται χρησιμοποιώντας ένα κριτήριο συμπεριφοράς, οι τιμές του οποίου δίνονται από το περιβάλλον, ενώ στη δεύτερη το κριτήριο συμπεριφοράς καθορίζεται εσωτερικά με βάση τις επιθυμητές αποκρίσεις. Συνεπώς ένα σύστημα ενισχυτική μάθησης μπορεί να θεωρηθεί ως ένα σύστημα *αξιολόγησης και ανατροφοδότησης.* Με την ενισχυτική μάθηση μπορούν να εκπαιδευτούν τόσο τα ΝΔ προσοτροφοδότησης όσο και τα αναδρομικά [8].

3.6.3 Μη επιβλεπόμενη μάθηση

Στη μη επιβλεπόμενη μάθηση, που καλείται και αυτό-οργανούμενη μάθηση (*self-organized learning*), δε χρησιμοποιείται εξωτερικός δάσκαλος για να επιβλέψει την εκπαίδευση, αλλά το ΝΔ, αντί να μάθει συγκεκριμένα παραδείγματα (ζεύγη) εισόδου – εξόδου, μαθαίνει ένα *ανεξάρτητο καθηκόντων μέτρο* της ποιότητας της παράστασης. Οι προς επιλογή, ελεύθερες παράμετροι του δικτύου προσαρμόζονται έτσι ώστε να βελτιστοποιηθεί το μέτρο αυτό. Πρακτικά, το μόνο που χρειάζεται ένα ΝΔ μη επιβλεπόμενης μάθησης είναι να συντονιστεί στις στατιστικές ο μαλότητες των δεδομένων εισόδου και μετά να μπορέσει να δημιουργήσει εσωτερικές παραστάσεις για την κωδικοποίηση των ιδιοτήτων τους [8].

Στην πράξη , στις περισσότερες εφαρμογές ΤΝΔ χρησιμοποιείται μάθηση υπό επίβλεψη (*supervised learning*), για την οποία υπάρχουν αρκ ετοί αλγόριθμοι. Στον αλγόριθμο, που βασίζε ται στον *κανόνα Δέλτα (Delta rule learning),* η διαφορά μεταξύ πραγματικής και επιθυμητής εξόδου ελαχιστοποιείται μέσω μιας

διαδικασίας ελαχίστων τετραγώνων . Στον αλγόριθμο *ανάστροφης μετάδοσης λάθους (back propagation)* η μεταβολή των βαρών βασίζεται στον υπολογισμό της συνεισφοράς κάθε βάρους στο συνολικό σφάλμα. Στην *ανταγωνιστική μάθηση (competitive learning)* οι τεχνητοί νευρώνες συναγωνίζονται, κατά κάποιο τρόπο, μεταξύ τους και μόνο αυτός με τη μεγαλύτερη απόκριση σε δοθείσα είσοδο τροποποιεί τα βάρη του . Τέλος , στην *τυχαία μάθηση (random learning)*, οι μεταβολές στα βάρη εισάγονται τυχαία και ανάλογα με το αν η έξοδος βελτιώνεται ή όχι με βάση κάποια προκαθορισμένα από τον χρήστη κριτήρια, οι μεταβολές αυτές υιοθετούνται ή απορρίπτονται [15].

3.7 Συσχετιστικές μνήμες

Η συσχετιστική μνήμη είναι γνώριμη σε μας γιατί αντιστοιχεί στον τρόπο με τον οποίο λειτουργεί η ανθρώπινη μνήμη. Έτσι λ.χ. σχηματίζουμε συσχετίσεις (αντιστοιχίες) ανάμεσα σε τόπους και γεγονότα, μουσικά κομμάτια με ανθρώπους κ.ο.κ. Στους συνήθεις, μη νευρωνικούς υπολογιστές, η μνήμη δεν λειτουργεί με συσχετιστικό τρόπο, γιατί κάθε στοιχείο πληροφορίας αποθηκεύεται στη δική του περιοχή μνήμης και η πρόσβαση σε αυτό γίνεται χρησιμοποιώντας τη διεύθυνσή του.

Στις συσχετιστικές μνήμες η παρουσίαση ενός ερεθίσματος εισόδου (*προτύπου*, *pattern*) ακολουθείται από ανάκληση ενός άλλου συνόλου προτύπων, που είναι αποθηκευμένα στη μνήμη. Συνεπώς, η λειτουργία μιας συσχετιστικής μνήμης περιλαμβάνει δύο φάσης: τη φάση μάθησης και τη φάση ανάκλησης (ανάκτησης). Κατά τη φάση μάθησης το πρότυπο, που παρουσιάζεται, μετασχηματίζεται σε ένα αποθηκευμένο πρότυπο, ενώ κατά τη φάση της ανάκτησης η μνήμη δέχεται μια παραμορφωμένη ή ατελή παραλλαγή του αποθηκευμένου προτύπου και δίνει ως έξοδο το αντίστοιχο ακριβές πρότυπο. Για αυτόν τον λόγο οι συσχετιστικές μνήμες ονομάζονται και *μνήμες με διευθύνσεις περιεχομένου (CAM: Content Addressable Memory)*. Επειδή δεν υπάρχει άμεση αντιστοιχία ανάμεσα στην είσοδο και την έξοδο, η έξοδος δε καθορίζεται από μια απλή θέση της μνήμης, αλλά μάλλον συμμετέχει ολόκληρη η μήτρα θέσεων αυτής. Συνεπώς μια συσχετιστική μνήμη χαρακτηρίζεται από μη - εντοπισμένη παράσταση, είναι δηλαδή *κατανεμημένη μνήμη*.

Η συσχέτιση διακρίνεται σε *αυτοσυσχέτιση* και *ετεροσυσχέτιση*. Στην πρώτη, ένα πρότυπο αντιστοιχείται στη μνήμη με τον εαυτό του, οπότε οι χώροι των δεδομένων

εισόδου και εξόδου έχουν την ίδια διάσταση. Στην ετεροσυσχέτιση, αυθαίρετα διανύσματα εισόδου αντιστοιχούνται με άλλα αυθαίρετα αποθηκευμένα διανύσματα της μνήμης, που μπορεί να έχουν την ίδια ή διαφορετική διάσταση [8].

3.8 Εφαρμογές Νευρωνικών Δικτύων

Τα πλεονεκτήματα και τα μειονεκτήματα των ΝΔ είναι συχνά συμπληρωματικά με εκείνα των συμβατικών τεχνικών ανάλυσης δεδομένων, ενώ θα πρέπει να χρησιμοποιούνται για την επίλυση προβλημάτων, τα οποία έχουν ένα ή και περισσότερα από τα παρακάτω χαρακτηριστικά [13]:

- Υπάρχει μεγάλο πλήθος δεδομένων για την εκπαίδευση του δικτύου
- Είναι δύσκολο να βρεθεί μια απλή και μοντελοποιημένη λύση, η οποία να είναι επαρκής
- Η επεξεργασία των νέων δεδομένων πρέπει να είναι ταχύτατη, είτε γιατί υπάρχει μεγάλος, προς ανάλυση, αριθμός δεδομένων, είτε επειδή η ανάλυση πραγματοποιείται σε πραγματικό χρόνο
- Η μέθοδος επεξεργασίας των δεδομένων πρέπει να είναι αποτελεσματική στην αντιμετώπιση του θορύβου στα δεδομένα εισόδου.

Τα ΝΔ είναι κατάλληλα για προβλήματα, στα οποία ο συνήθης υπολογισμός δεν είναι αποδοτικός, όπως μηχανική όραση, αναγνώριση προτύπων, αναγνώριση φωνής, μη γραμμική πρόβλεψη, ελεύθερη μοντέλου αναγνώριση συστημάτων, αυτόματο έλεγχο, ρομποτική, επιχειρησιακά προβλήματα κ.α.

Τα ΝΔ έχουν χρησιμοποιηθεί σε πλήθος εφαρμογών, όπως είναι ο σχεδιασμός ενεργειών (*planning*), ο χρονοπρογραμματισμός (*scheduling*), η διάγνωση λαθών σε δορυφορικές επικοινωνίες, η αναγνώριση υπογραφών, η βραχυπρόθεσμη πρόβλεψη τιμών μετοχών, κτλ.. Επίσης, τα περισσότερα σύγχρονα προγράμματα οπτικής αναγνώρισης χαρακτήρων χρησιμοποιούν νευρωνικά δίκτυα. Επιπλέον, νευρωνικά δίκτυα χρησιμοποιούνται και σε υπολογιστές παλάμης, που δέχονται εντολές χειρόγραφα (*palmtop*).

Άλλες εφαρμογές των ΝΔ είναι στην Ιατρική (π.χ. ανάλυση ηλεκτροκαρδιογραφήματος, ιατρική διάγνωση και επεξεργασία ιατρικής εικόνας), σε αμυντικά συστήματα (π.χ. υποβρύχια ανίχνευση ναρκών) στην οικονομία (π.χ.

ανάλυση αγοράς μετοχών, ασφάλεια συναλλαγών, εκτίμηση φερεγγυότητας δανειζόμενου πελάτη, εκτίμηση ακίνητης περιουσίας), στη σχεδίαση, έλεγχος και αναζήτηση (π.χ. παράλληλη υλοποίηση NP-πλήρων προβλημάτων).

Τα τελευταία χρόνια τα ΝΔ χρησιμοποιούνται σε συστήματα ελέγχου, που βασίζονται στην ασαφή λογική *(neurofuzzy systems),* με κύριο ρόλο τον υπολογισμό της συνάρτησης συγγένειας.

Ένα γνωστό εμπορικό σύστημα που κάνει χρήση ΝΔ είναι το *PAPNET,* το οποίο κάνει διάγνωση σε αποτελέσματα "τεστ Παπανικόλαου ". Μια ψηφιοποιημένη εικόνα μικροσκοπίου δίνεται σαν είσοδος στο νευρωνικό δίκτυο , το οποίο αναλαμβάνει να κρίνει αν υπάρχει ανωμαλία ή όχι . Το εν λόγω σύστημα επιταχύνει σημαντικά τη χρονοβόρα διαδικασία εξέτασης των δεδομένων της εξέτασης από τον ειδικό ιατρό [15].

4 *Νευρωνικό Δίκτυο Hopfield*

Το νευρωνικό δίκτυο Hopfield είναι ένα αναδρομικό, ασύγχρονο δίκτυο, μη επιβλεπόμενης μάθησης και αποτελείται από ένα μοναδικό στρώμα πλήρως διασυνδεδεμένων κόμβων με συμμετρικές συνδέσεις ανατροφοδότησης. Αυτοβρόχοι, δηλαδή ανατροφοδοτήσεις από κάποιον κόμβο στον εαυτό του, δεν υπάρχουν.

Το δίκτυο Hopfield έχει αυτοσχετιστικές ιδιότητες και μπορεί να ανακτήσει (ανακαλέσει) ένα πρότυπο, που έχει αποθηκευμένο στη μνήμη του, όταν του παρουσιαστεί μια διαταραγμένη ή αλλοιωμένη παραλλαγή του προτύπου αυτού στην είσοδό του.

Το δίκτυο αυτό προέκυψε από την ακόλουθη παρατήρηση του *John Hopfield* σχετικά με τα φυσικά συστήματα :

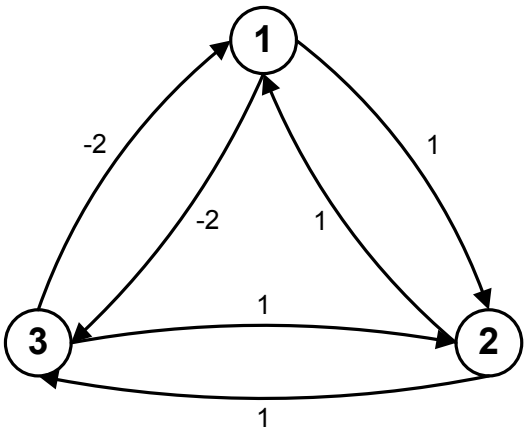
«Οποιοδήποτε φυσικό σύστημα, του οποίου η δυναμική στον χώρο των φάσεων κυριαρχείται από έναν αξιόλογο αριθμό τοπικά ευσταθών καταστάσεων προς τις οποίες ελκύεται, μπορεί να θεωρηθεί ως μια μνήμη με διευθύνσεις περιεχομένου (content addressable memory). Το φυσικό αυτό σύστημα θα είναι μία χρήσιμη μνήμη εάν, επιπλέον, οι καταστάσεις σε κάθε καθορισμένο σύνολο αυτού, μπορούν να ελεγχθούν και να γίνουν ευσταθείς καταστάσεις.»

Συνεπώς, το δίκτυο Hopfield αποτελεί μια αυτοσυσχετιστική μνήμη και ένα εργαλείο επίλυσης προβλημάτων βελτιστοποίησης [8].

4.1 Αρχιτεκτονική

Το 1982 ο Αμερικανός φυσικός *John Hopfield* πρότεινε ένα αναδρομικό, ασύγχρονο μοντέλο νευρωνικού δικτύου, που είχε άμεσο αντίκτυπο στον τομέα της Τεχνητής Νοημοσύνης [16].

Το δίκτυο Hopfield αποτελείται από ένα μοναδικό στρώμα με N πλήρως διασυνδεδεμένους νευρώνες, καθένας εκ των οποίων είναι συνδεδεμένος με όλους τους υπόλοιπους εκτός του εαυτού του (σχήμα 4.1). Το δίκτυο είναι συμμετρικό γιατί, το βάρος w_{ij} της σύνδεσης μεταξύ του νευρώνα i και του νευρώνα j είναι ίσο με το βάρος w_{ij} της σύνδεσης μεταξύ του νευρώνα i και του νευρώνα j. Αυτό μπορεί να ερμηνευτεί με την έννοια ότι υπάρχει μια αμφίδρομη σύνδεση μεταξύ των δύο νευρώνων. Η απουσία διασύνδεσης κάθε νευρώνα με τον εαυτό του αποτρέπει τη μόνιμη ανατροφοδότηση της δικής του κατάστασης [17].



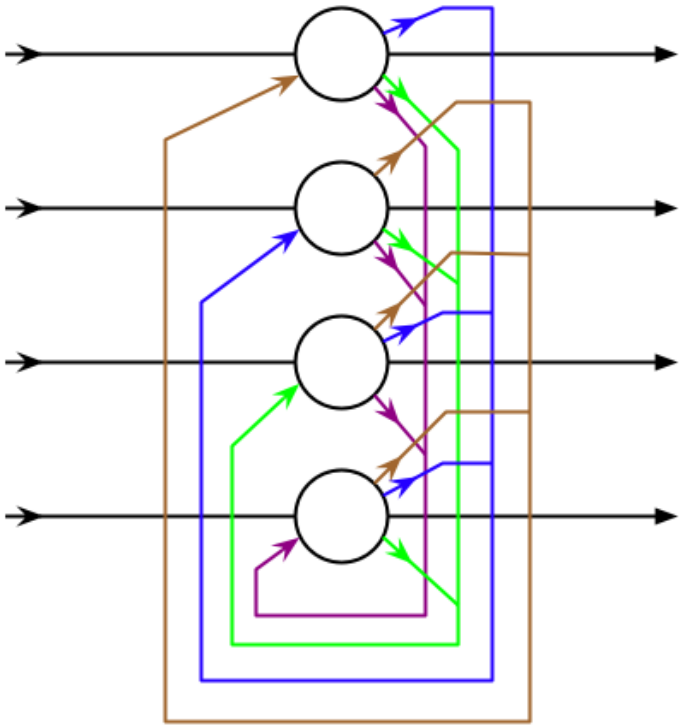
Σχήμα 4.1: Δίκτυο Hopfield με τρεις νευρώνες

Τα συναπτικά βάρη ενός δικτύου Hopfield με N νευρώνες μπορούν να αναπαρασταθούν με έναν N x N συμμετρικό πίνακα $\boldsymbol{W} = w_{ij}$ με μηδενικά τα στοιχεία της διαγωνίου. Η συμμετρία και η μηδενική διαγώνιος του πίνακα των συναπτικών βαρών αποτελούν ικανές και αναγκαίες συνθήκες για τη σύγκλιση του δικτύου σε μία ευσταθή κατάσταση.

Στους νευρώνες του δικτύου Hopfield μπορεί να ανατεθεί μία τιμή κατωφλίου θ διάφορη του μηδενός. Στην περίπτωση αυτή ο νευρώνας, που έχει επιλεχθεί για ενημέρωση, πυροδοτείται αν η συνολική του διέγερση υπερβεί την τιμή αυτή.

4.2 Δυναμική συμπεριφορά

Στην παρούσα παράγραφο παρουσιάζεται μια ανάλυση της δυναμικής συμπεριφοράς, δηλαδή της λειτουργίας, τόσο του διακριτού, όσο και του αναλογικού δικτύου Hopfield τεσσάρων νευρώνων, που απεικονίζεται στο σχήμα 4.2. Παρατηρώντας το σχήμα διαπιστώνουμε ότι όλοι οι κόμβοι του δικτύου είναι ισοδύναμοι, πράγμα που αποτελεί το κύριο χαρακτηριστικό του δικτύου Hopfield και καθορίζει τον τρόπο λειτουργίας του. Οι είσοδοι εφαρμόζονται ταυτόχρονα και κατόπιν το δίκτυο λειτουργώντας αυτόνομα περνάει από διάφορους κύκλους λειτουργίας έως ότου συγκλίνει σε μια ευσταθή κατάσταση, στην οποία ελαχιστοποιείται μια συνάρτηση κόστους (ενέργειας). Η έξοδος του δικτύου αποτελείται από τις τιμές ό λων των κόμβων όταν δεν υπάρχει πλέον καμία αλλαγή από κύκλο σε κύκλο. Αποδεικνύεται ότι κάθε ασύγχρονο δίκτυο Hopfield με N νευρώνες και οποιαδήποτε δοσμένη αρχική κατάσταση θα μεταβεί, μετά από πεπερασμένο αριθμό επαναλήψεων, σε ευσταθή κατάσταση, η οποία θα αντιστοιχεί σε κάποιο τοπικό ελάχιστο της συνάρτησης ενέργειας [16].



Σχήμα 4.2: Δίκτυο Hopfield τεσσάρων νευρώνων

4.2.1 Διακριτό δίκτυο Hopfield

Στο διακριτό δίκτυο Hopfield ο χρόνος είναι διακριτός και οι τιμές των κόμβων είναι διπολικές (-1,1) ή δυαδικές (0,1). Η δυναμική συμπεριφορά του εκφράζεται με τις παρακάτω σχέσεις :

$$u_i(t+1) = \sum_{j=1}^{N} w_{ij} v_j\,(t) + \theta_i \qquad (4.1)$$

$$v_i(t+1) = f\big(u_i(t+1)\big) \qquad (4.2)$$

Όπου : u_i το σήμα εισόδου, v_i το σήμα εξόδου, θ_i το εξωτερικό κατώφλι στο νευρώνα *i,* και *f* η συνάρτηση ενεργοποίησης.

Αναφορικά με τη συνάρτηση *f,* ο *Hopfield*, στο αρχικό δίκτυό του, χρησιμοποίησε νευρώνες *McCulloch – Pitts*, με δυαδική συνάρτηση ενεργοποίησης αλλά αργότερα χρησιμοποιήθηκαν σιγμοειδείς συναρτήσεις όλων των τύπων.

Η συνάρτηση κόστους ή ενέργειας, που καλείται να ελαχιστοποιήσει το διακριτό δίκτυο Hopfield είναι :

$$E = -\frac{1}{2}\sum_{i=1}^{N} w_{ij}\, v_i v_j - \sum_{i=1}^{N} \theta_i v_i \qquad (4.3)$$

Η συνάρτηση ενέργειας έχει τετραγωνική μορφή και το δίκτυο Hopfield βρίσκει πάντα ένα τοπικό ελάχιστό της. Ο παράγοντας ½ χρησιμοποιείται γιατί στη διπλή σειρά εμφανίζεται τόσο ο όρος $w_{ij}v_iv_j$ όσο και ο όρος $w_{ji}v_jv_i$, οι οποίοι είναι όμοιοι.

4.2.2 Αναλογικό δίκτυο Hopfield

Στο αναλογικό δίκτυο Hopfield ο χρόνος είναι συνεχής και οι κόμβοι παίρνουν συνεχείς τιμές στο διάστημα [-1,1] ή [0,1]. Η δυναμική συμπεριφορά του εκφράζεται με τις παρακάτω σχέσεις :

$$\frac{du_i}{dt} = -u_i + \sum_{j=1}^{N} w_{ij} v_j + \theta_i \qquad (4.4)$$

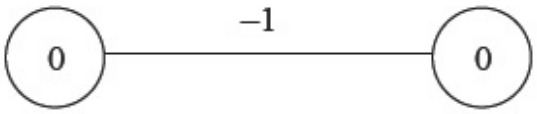
$$v_i = f(u_i) \qquad (4.5)$$

Όπου : u_i το σήμα εισόδου, v_i το σήμα εξόδου, θ_i το εξωτερικό κατώφλι στον νευρώνα *i,* και *f* η συνάρτηση ενεργοποίησης.

Η συνάρτηση κόστους ή ενέργειας, που καλείται να ελαχιστοποιήσει το αναλογικό δίκτυο Hopfield είναι :

$$E = -\frac{1}{2} \cdot \sum_{i=1}^{N} w_{ij}\, v_i v_j - \sum_{i=1}^{N} \theta_i v_i + \sum_{i=1}^{N} \int_0^{v_i} f^{-1}(v) dv \qquad (4.5)$$

Στο σχήμα 4.3 απεικονίζεται ένα δίκτυο με μόνο δύο νευρώνες, μηδενική τιμή κατωφλίου και $w_{12} = w_{21} = -1$.

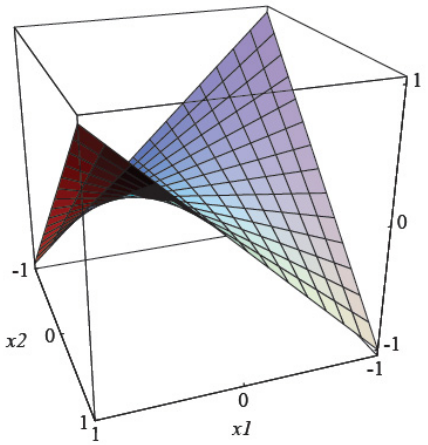


Σχήμα 4.3: Δίκτυο Hopfield με δύο νευρώνες

Η συνάρτηση ενέργειας του παραπάνω δικτύου δίνεται από τη σχέση:

$$E = x_1 x_2$$

Όπου x_1, x_2 είναι οι τιμές του πρώτου και του δεύτερου νευρώνα αντίστοιχα. Στο σχήμα 4.4 απεικονίζεται η συνάρτηση ενέργειας.

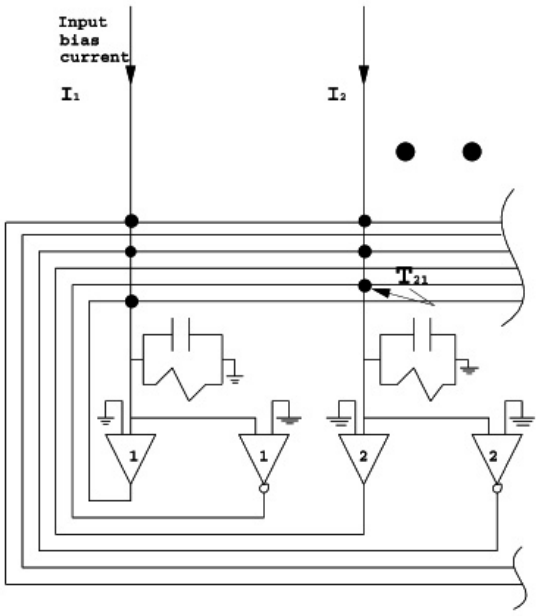


Σχήμα 4.4: Συνάρτηση ενέργειας

Παρατηρούμε ότι η συνάρτηση ενέργειας παρουσιάζει δύο τοπικά ελάχιστα για $(x_1, x_2) = (1, -1)$ και $(x_1, x_2) = (-1,1)$, που αποτελούν τις ευσταθείς καταστάσεις, στις οποίες και ισορροπεί το δίκτυο.

4.3 Υλοποίηση δικτύου Hopfield με αναλογικά κυκλώματα

Η υλοποίηση ενός δικτύου Hopfield με απλά αναλογικά κυκλώματα φαίνεται στο σχήμα 4.4 [18]. Το δίκτυο αυτό έχει τρεις βασικούς τύπους παράλληλης οργάνωσης, που εντοπίζονται στα νευρωνικά συστήματα: παράλληλα κανάλια εισόδου, παράλληλα κανάλια εξόδου και πλήρης διασύνδεση μεταξύ των νευρωνικών επεξεργαστών. Οι νευρωνικοί επεξεργαστές μοντελοποιούνται με ενισχυτές σε συνδυασμό με κυκλώματα ανάδρασης, που αποτελούνται από καλώδια, αντιστάσεις και πυκνωτές, έτσι ώστε να προσομοιώνουν τα πιο βασικά υπολογιστικά χαρακτηριστικά των νευρώνων, δηλαδή τους άξονες, τους δενδρίτες και τις συνάψεις. Η σχέση εισόδου – εξόδου των ενισχυτών είναι μονότονη σιγμοειδής και οι χρονικές σταθερές τους θεωρούνται αμελητέες. Κάθε ενισχυτής, σε αναλογία με τον βιολογικό νευρώνα, όπου υπάρχει αντίσταση εισόδου, που προκαλείται από τη μεμβράνη του κυττάρου, έχει μία αντίσταση ρ_j, η οποία οδηγεί σε γείωση, και έναν πυκνωτή C_j. Αυτά τα στοιχεία καθορίζουν μερικώς τις χρονικές σταθερές των νευρώνων και παρέχουν ολοκληρωτική αναλογική άθροιση των συναπτικών ρευμάτων εισόδου από τους υπόλοιπους νευρώνες στο δίκτυο.



Σχήμα 4.4: Αναλογικό κύκλωμα δικτύου Hopfield

Προκειμένου να υπάρξουν τόσο διεγερτικές όσο και ανασταλτικές συναπτικές συνδέσεις μεταξύ των νευρώνων, κάθε ενισχυτής έχει δύο εξόδους, μία κανονική με τιμές μεταξύ 0 και 1, και μία ανεστραμμένη με τιμές μεταξύ -1 και 0. Μια σύναψη

μεταξύ δύο νευρώνων καθορίζεται από την αγωγιμότητα T_{ij}, που συνδέει μία από τις δύο εξόδους του ενισχυτή *j* στην είσοδο του ενισχυτή *i,* και αποτελείται από μία αντίσταση $R_{ij} = 1/|T_{ij}|$. Αν η σύναψη είναι διεγερτική ($T_{ij} > 0$), συνδέεται στην κανονική έξοδο του ενισχυτή *j,* ενώ αν είναι ανασταλτική ($T_{ij} < 0$), συνδέεται στην ανεστραμμένη έξοδο. Όπως φαίνεται στο σχήμα 4.4, το κύκλωμα περιέχει και εξωτερικά ρεύματα εισόδου, τα οποία μπορούν να χρησιμοποιηθούν για να καθορίσουν την τιμή κατωφλίου ή για την απευθείας καθοδήγηση ορισμένων νευρώνων.

Από τη θεωρία των κυκλωμάτων οι εξισώσεις, που περιγράφουν το κύκλωμα είναι :

$$C_i \frac{du_i}{dt} = -\frac{u_i}{R_i} + \sum_{j=1}^{N} T_{ij} v_j + I_i \qquad (4.6)$$

$$v_i = f(u_i)$$

Όπου R_i είναι το άθροισμα των παράλληλων αντιστάσεων ρ_i και R_{ij} :

$$\frac{1}{R_i} = \frac{1}{\rho_i} + \sum_{j=1}^{N} \frac{1}{R_{ij}} \qquad (4.7)$$

Για λόγους απλότητας, θεωρούμε ότι $R_i = R$, $C_i = C$, ανεξάρτητα του *i.* Διαιρώντας με *C* και θέτοντας $T_{ij} = T_{ij}/C$ και $I_i = I_i/C$ οι εξισώσεις γίνονται :

$$\frac{du_i}{dt} = -\frac{u_i}{\tau} + \sum_{j=1}^{N} T_{ij} v_j + I_i \qquad (4.8)$$

$$v_i = f(u_i) \qquad (4.9)$$

όπου $\tau = RC$.

Η εξισώσεις (4.8) και (4.9) περιγράφουν πλήρως τη χρονική εξέλιξη της κατάστασης των νευρώνων και η υλοποίηση τους με τη βοήθεια υπολογιστή επιτρέπει την προσομοίωση οποιουδήποτε υποθετικού δικτύου.

Η συνάρτηση ενέργειας δίδεται από τη σχέση :

$$E = -\frac{1}{2} \cdot \sum_{i=1}^{N} T_{ij}\, v_i v_j - \sum_{i=1}^{N} v_i I_i + \sum_{i=1}^{N} \int_0^{v_i} f^{-1}(v) dv \qquad (4.10)$$

Όταν οι ενισχυτές λειτουργούν με υψηλό κέρδος η συνάρτηση απλοποιείται ως εξής:

$$E = -\frac{1}{2} \cdot \sum_{i=1}^{N} T_{ij}\, v_i v_j - \sum_{i=1}^{N} v_i I_i \qquad (4.11)$$

Τέλος, εύκολα μπορεί να αποδειχθεί ότι ισχύει:

$$\frac{du_i}{dt} = -\frac{u_i}{\tau} - \frac{\partial E}{\partial v_i} \qquad (4.12)$$

Ο χώρος κατάστασης, στον οποίο λειτουργεί το κύκλωμα, είναι το εσωτερικό ενός N-διάστατου υπερκύβου. Όταν, ό μως, ισχύει η συνθήκη υψηλού κέρδους, τα τοπικά ελάχιστα βρίσκονται στις γωνίες αυτού. Συνεπώς, οι ευσταθείς καταστάσεις αντιστοιχούν στις 2^N γωνίες του υπερκύβου.

4.4 Εφαρμογές

Το δίκτυο Hopfield μπορεί να λειτουργήσει αποδοτικά ως αυτοσυσχετιστική μνήμη ή στην επίλυση προβλημάτων βελτιστοποίησης. Στις επόμενες παραγράφους παρατίθενται δύο χαρακτηριστικά παραδείγματα εφαρμογών.

4.4.1 Αναγνώριση προτύπων

Το δίκτυο Hopfield αναπτύσσει συνήθως έναν αριθμό ευσταθών σημείων στον χώρο κατάστασης. Επειδή η δυναμική του δικτύου τείνει να ελαχιστοποιήσει την ενέργεια, τα άλλα σημεία του χώρου κατάστασης διοχετεύονται στα ευσταθή σημεία, που καλούνται *ελκυστές* και αποτελούν τοπικά ενεργειακά ελάχιστα. Η λειτουργία του δικτύου ως αυτοσυσχετιστική μνήμη έγκειται στο ότι νέα πρότυπα παρουσιαζόμενα στην είσοδο συνδέονται (συσχετίζονται) με τα κατάλληλα πρότυπα, τα οποία είναι αποθηκευμένα ως ευσταθείς καταστάσεις. Αποδεικνύεται ότι ο αριθμός των προτύπων *P*, που μπορούν να αποθηκευτούν είναι ανάλογος του αριθμού των νευρώνων *N* και συγκεκριμένα ισχύει :

$$P \leq 0{,}138N$$

Σε περίπτωση, που ο αριθμός των προτύπων ξεπεράσει αυτό το όριο αυξάνεται η πιθανότητα εσφαλμένης ανάκλησης προτύπου.

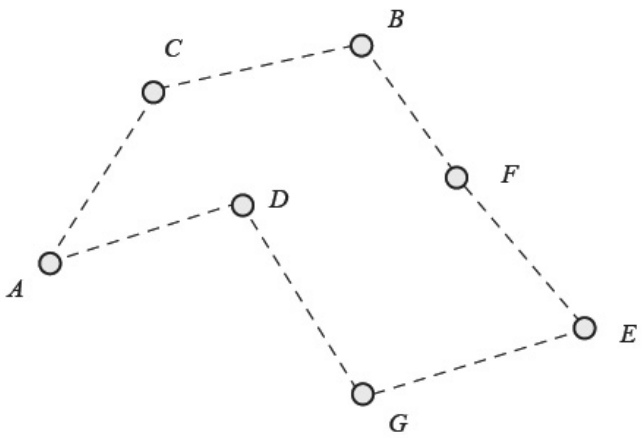
Λόγω της ανεκτικότητας τους στις παραμορφώσεις των προτύπων εισόδου τα δίκτυα Hopfield μπορεί να χρησιμοποιηθούν για την αναγνώριση προτύπων. Στην είσοδο παρουσιάζεται μια αλλοιωμένη εικόνα και το δίκτυο δίνει στην έξοδο τ η σωστή (εικόνα 4.1).

Εικόνα 4.1: Παράδειγμα ανάκτησης εικόνας

4.4.2 Πρόβλημα Περιοδεύοντος Πωλητή (Travel Salesman Problem)

Το *Πρόβλημα του Περιοδεύοντος Πωλητή*, εν συντομία TSP, είναι πιθανώς το πιο δημοφιλές στον χώρο της συνδυαστικής βελτιστοποίησης. Είναι απλό στην περιγραφή, μαθηματικώς καλά ορισμένο, και αποτελεί χρήσιμο μέσο στην ανάλυση της χρήσης του δικτύου Hopfield για την επίλυση δύσκολων προβλημάτων βελτιστοποίησης. Ένα σύνολο από N πόλεις A, B, C… έχουν μεταξύ τους αποστάσεις $d_{AB}, d_{AC}, \dots, d_{BC}$ … ($\sigma\chi\acute{\eta}\mu\alpha$ 4.5). Το πρόβλημα έγκειται στην εύρεση μιας κλειστής διαδρομής, που επισκέπτεται κάθε πόλη μόνο μία φορά, επιστρέφει στην αρχική πόλη και έχει το ελάχιστο μήκος. Η λύση αποτελείται από μία σειρά (π.χ. B, F, E, G,…W), με την οποία ο πωλητής επισκέπτεται τις πόλεις και το συνολικό μήκος της διαδρομής, που δίνεται από τη σχέση :

$$d = d_{BF} + d_{FE} + d_{EG} + \cdots + d_{WB}$$



Σχήμα 4.5: Ένα TSP πρόβλημα και η λύση του

Η εύρεση της βέλτιστης λύσης είναι υπολογιστικά δύσκολη. Το TSP ανήκει στα *np-complete* προβλήματα και ο χρόνος, που απαιτείται για την επίλυσή του σε οποιοδήποτε υπολογιστή αυξάνεται εκθετικά με την αύξηση του αριθμού των πόλεων.

Οι *Hopfield* και *Tank* [18], με την επιλογή της κατάλληλης συνάρτησης ενέργειας, απέδειξαν πως το δίκτυο Hopfield μπορεί να λύσει το TSP πρόβλημα και, λόγω της παράλληλης λειτουργίας των νευρώνων του, σε μικρότερο χρόνο συγκριτικά με τις «συμβατικές» μεθόδους. Έκτοτε, πολλοί ερευνητές ασχολήθηκαν με το πεδίο αυτό βελτιώνοντας περαιτέρω την αποτελεσματικότητα και την ταχύτητα υπολογισμού.

5 *Δίκτυο Hopfield και δρομολόγηση σε οδικά δίκτυα*

Οι συνεχώς αυξανόμενες απαιτήσεις για ταχεία και ακριβή επίλυση του προβλήματος της δρομολόγησης έχει οδηγήσει τους ερευνητές στην αναζήτηση νέων μεθόδων, πέρα των «συμβατικών» υπολογιστικών αλγορίθμων, που παρουσιάστηκαν στο πρώτο κεφάλαιο. Τα αποτελέσματα των σύγχρονων τεχνικών επίλυσης, συμπεριλαμβανομένων των Γενετικών αλγόριθμων, της Τεχνητής νοημοσύνης και των Νευρωνικών δικτύων, είναι αρκετά ελπιδοφόρα.

Η εφαρμογή των ΝΔ στο πρόβλημα δρομολόγησης ενισχύθηκε από το γεγονός ότι όταν αυτά υλοποιηθούν σε υλικό, μπορούν να επιτύχουν υψηλές ταχύτητες απόκρισης [19]. Οι απαιτήσεις από οποιοδήποτε ΝΔ είναι [20]:

- Ο υπολογισμός πρέπει να πραγματοποιείται σε πραγματικό χρόνο.
- Ο αλγό ριθμος πρέπει να προσαρμόζεται στις αλλαγές των βαρών των συνδέσμων.
- Η επιβάρυνση της μνήμης και του επεξεργαστή πρέπει να είναι όσο το δυνατόν μικρότερη.
- Ο αλγόριθμος πρέπει να προσαρμόζεται στις αλλαγές της τοπολογίας του δικτύου.

Σε πολλές περιπτώσεις μία σχεδόν βέλτιστη λύση, όταν υπολογίζεται σε μικρό χρονικό διάστημα, είναι προτιμότερη από τη βέλτιστη λύση. Αυτό μπορεί να επιτευχθεί με ένα ΝΔ, που αποτελείται από αναλογικά υπολογιστικά κυκλώματα. Η απόκριση είναι σχεδόν ακαριαία γιατί όλοι οι νευρώνες αλλάζουν συνεχώς και παράλληλα την κατάστασή τους.

Στο προηγούμενο κεφάλαιο, κατά τη λεπτομερή ανάλυση του δικτύου Hopfield, αναφέρθηκε ότι μπορεί να χρησιμοποιηθεί για την επίλυση προβλημάτων βελτιστοποίησης, συνεπώς και για την εύρεση της συντομότερης διαδρομής. Η χρήση του δικτύου Hopfield για την εύρεση της συντομότερης διαδρομής μεταξύ δύο κόμβων ενός δικτύου επικοινωνιών προτάθηκε αρχικά από τους *Rauch* και *Winarske* [21]. Η προσέγγιση τους περιοριζόταν από την απαίτηση πρωτύτερης γνώσης του αριθμού των κόμβων, που συνθέτουν τη διαδρομή, κάτι το οποίο στην πλειονότητα των περιπτώσεων δεν είναι γνωστό. Για την άρση του παραπάνω περιορισμού οι *Zhang* και *Thomopoulos* [22] πρότειναν μία νέα μέθοδο. Όμως, με τη μέθοδο αυτή, για κάθε διαφορετικό ζεύγος αφετηρίας – προορισμού απαιτείται τροποποίηση των ρυθμίσεων του νευρωνικού δικτύου κάτι το οποίο δεν είναι κατάλληλο για πρακτικά προβλήματα δρομολόγησης. Οι *Ali* και *Kamoun* [23] εισήγαγαν έναν αλγόριθμο δρομολόγησης, προσαρμοζόμενο σε διαφορετικές τοπολογίες, ο οποίος, για δίκτυα μικρής κλίμακας, παρουσιάζει πολύ καλή σύγκλιση με χαμηλή προγραμματιστική πολυπλοκότητα. Ωστόσο, η σύγκλιση και η ποιότητα των λύσεων για μεγαλύτερα δίκτυα ($N > 20$) δεν είναι ικανοποιητική εξαιτίας των βρόχων και των διαμερίσεων. Για την αντιμετώπιση αυτού του προβλήματος ένας νέος όρος προστέθηκε στη συνάρτηση ενέργειας από τους *Park* και *Choi* [24], όμως η απόδοση του συγκεκριμένου αλγόριθμου εξαρτάται έντονα από την τοπολογία του δικτύου, ενώ ή ταχύτητα σύγκλισης χρίζει περαιτέρω βελτίωσης. Οι *Ahn* και *Pamakrishna* [25] προσέθεσαν δύο επιπλέον όρους στην συνάρτηση ενέργειας των *Park* και *Choi* για να επιταχύνουν τη σύγκλιση και ταυτόχρονα να βελτιώσουν την ποιότητα της λύσης χωρίς όμως ικανοποιητικά αποτελέσματα για δίκτυα μεγαλύτερα των 20 κόμβων.

Στο κεφάλαιο αυτό θα αναλυθεί η δυνατότητα χρήσης του δικτύου Hopfield για τη δρομολόγηση σε οδικά, πλέον, δίκτυα, αφού πρότινος οριστεί επαρκώς το πρόβλημα.

5.1 Ορισμός του προβλήματος

Παρ' ότι δεν υπάρχει σχετική βιβλιογραφία για τη χρήση των δικτύων Hopfield στον υπολογισμό της συντομότερης διαδρομής σε οδικά δίκτυα, λόγω των κοινών χαρακτηριστικών τους με τα δίκτυα επικοινωνιών μπορεί να υποτεθεί ότι αυτό είναι εφικτό.

Ωστόσο, το πρόβλημα, που ανακύπτει, είναι ότι ο αριθμός των κόμβων ενός οδικού δικτύου είναι πολύ μεγάλος, ενώ οι μέχρι τώρα έρευνες έχουν αποδείξει ότι η απόδοση του δικτύου Hopfield επηρεάζεται αρνητικά για μεγάλα *N*. Για να ξεπεραστεί αυτό, μπορούμε να υιοθετήσουμε την ιδέα της ιεραρχικής αναζήτησης, που παρουσιάστηκε στην παράγραφο 2.5. Συνεπώς, χωρίζοντας τους δρόμους σε κλάσεις, χρησιμοποιούμε το δίκτυο Hopfield για τον υπολογισμό της διαδρομής στο κυρίως οδικό δίκτυο, ενώ για τις διαδρομές από και προς αυτό, μπορεί να γίνει χρήση οποιουδήποτε «συμβατικού» υπολογιστικού αλγόριθμου.

Για παράδειγμα, στην περίπτωση της Αθήνας, μιας πόλης ιδιαίτερα εκτεταμένης, χρησιμοποιώντας την παραπάνω λογική οι κόμβοι, που προκύπτουν από τους κεντρικούς δρόμους και αυτοκινητόδρομους, περιορίζονται στους 200.

Σύμφωνα με τους ορισμούς της παραγράφου 2.1, θεωρούμε ένα οδικό δίκτυο R = (G, W), όπου G, κατευθυνόμενος μερικώς διασυνδεδεμένος γράφος και $\boldsymbol{W} = [w_{ij}]$, ο πίνακας, που περιέχει τα βάρη των συνδέσμων. Το βάρος κάθε συνδέσμου είναι ένας μη αρνητικός αριθμός, ενώ όταν ο σύνδεσμος από τον κόμβο *i* στον *j* δεν υπάρχει ισούται με το μηδέν. Για τον κατάλληλο προσδιορισμό του προβλήματος είναι απαραίτητο να βρεθεί ένα σύστημα αντιστοίχισης, έτσι ώστε το συντομότερο μονοπάτι να αποκωδικοποιείται από την ευσταθή κατάσταση του δικτύου Hopfield. Έστω ότι το δίκτυο αποτελείται από *N* κόμβους. Θεωρούμε έναν *N* x *N* πίνακα με μηδενικά τα στοιχεία της διαγωνίου. Κάθε στοιχείο του πίνακα αντιστοιχεί σε έναν νευρώνα, ο οποίος συμβολίζεται με δύο δείκτες *i*, *j*. Δεδομένου ότι τα στοιχεία της διαγωνίου είναι μηδενικά, ο αριθμός των νευρώνων, που απαιτούνται, είναι *N* x (*N* – 1). Η κατάσταση ενός νευρώνα, v_{ij}, ορίζεται ακολούθως :

$$v_{ij} = \begin{cases} 1, & \textit{εάν ο σύνδεσμος από τον κόμβο i στον j περιέχεται στη λύση} \\ 0, & \textit{διαφορετικά} \end{cases}$$

Με τη χρήση των παραπάνω ορισμών το πρόβλημα της συντομότερης διαδρομής από τον κόμβο αφετηρίας *s* προς τον κόμβο προορισμού *d*, μπορεί να εκφραστεί με τις παρακάτω λογικές αλγεβρικές εκφράσεις:

$$Ελαχιστοποίησε\ το \sum_{i=1}^{N}\sum_{j=1}^{N} w_{ij}v_{ij} \qquad (5.1)$$

$$Όταν\ ισχύει \left(\sum_{\substack{j=1\\ j\neq i}}^{N} v_{ij} - \sum_{\substack{j=1\\ j\neq i}}^{N} v_{ji}\right) = \begin{cases} 1, & εάν\ i = s \\ -1, & εάν\ i = d \\ 0, & διαφορετικά \end{cases} \qquad (5.2)$$

5.2 Συνάρτηση ενέργειας

Για την επίλυση του προβλήματος συντομότερης διαδρομής, όπως έχει ήδη αναφερθεί, έχουν προταθεί διάφορες συναρτήσεις ενέργειας. Προκειμένου να βελτιωθεί η σύγκλιση και η ποιότητα της λύσης στην παρούσα εργασία προτείνεται η παρακάτω τροποποιημένη συνάρτηση ενέργειας :

$$E = \frac{a_1}{2}\sum_{\substack{i=1\\ i\neq d}}^{N}\sum_{\substack{j=1\\ j\neq i,s}}^{N} w_{ij}v_{ij} + \frac{a_2}{2}\sum_{i=1}^{N}\left(\sum_{\substack{j=1\\ j\neq i}}^{N} v_{ij} - \sum_{\substack{j=1\\ j\neq i}}^{N} v_{ji}\right)^2 + \frac{a_3}{2}\sum_{i=1}^{N}\sum_{\substack{j=1\\ j\neq i}}^{N} v_{ij}(1 - v_{ij})$$

$$+ \frac{a_4}{2}(1 - v_{ds}) + \frac{a_5}{2}\sum_{i=1}^{N}\sum_{\substack{j=1\\ j\neq i}}^{N}\left(v_{ij}\sum_{\substack{k=1\\ k\neq i}}^{N} v_{ki}\right)$$

$$+ \frac{a_6}{2}\sum_{i=1}^{N}\sum_{\substack{j=1\\ j\neq i}}^{N}\left(\frac{v_{ij}^2}{2} - v_{ij}\sum_{\substack{k=1\\ k\neq j}}^{N} v_{jk}\right) \qquad (5.3)$$

Οι όροι a_1, a_2, a_3, a_4 έχουν προκύψει από τη συνάρτηση ενέργειας των *Ali* και *Kamoun* [23]. Ο πρώτος όρος ελαχιστοποιεί το συνολικό κόστος της διαδρομής, λαμβάνοντας υπόψη το κόστος των υπαρχόντων συνδέσμων. Ο όρος a_2 μηδενίζεται όταν ο αριθμός των εισερχόμενων συνδέσμων σε έναν κόμβο ισούται με τον αριθμό των εξερχόμενων κάτι το οποίο είναι απαραίτητο για να είναι έγκυρο το μονοπάτι. Αυτό είναι εφικτό γιατί η συνάρτηση ενέργειας δεν έχει οριστεί με τέτοιο τρόπο, ώστε να βρίσκει ένα μονοπάτι από την αφετηρία στον προορισμό, αλλά μία κυκλική διαδρομή, χρησιμοποιώντας έναν υποθετικό σύνδεσμο από τον κόμβο προορισμού στον κόμβο αφετηρίας για να την ολοκληρώσει. Προκειμένου να έχει νόημα το

αποτέλεσμα του δικτύου Hopfield είναι επιθυμητό η ευσταθής κατάσταση να είναι τέτοια, ώστε να μπορεί να αποκωδικοποιηθεί, δηλαδή οι τιμές των νευρώνων να είναι 0 ή 1. Ο όρος a_3 πραγματοποιεί ακριβώς αυτό κατευθύνοντας το νευρωνικό δίκτυο να συγκλίνει σε μία από τις $2^{N(N-1)}$ γωνίες του υπερκύβου, που ορίζεται από το $v_{ij} \in \{0,1\}$. Είναι απαραίτητο να περιέχονται στη λύση τόσο ο κόμβος αφετηρίας *s* όσο και ο κόμβος προορισμού *d*. Αυτό επιτυγχάνεται μέσω του όρου α_4, ο οποίος μηδενίζεται όταν η έξοδος του νευρώνα (*d, s*) τείνει στη μονάδα. Ο σύνδεσμος από τον *d* στον *s* είναι υποθετικός, έτσι η τελική λύση θα είναι πάντα ένας βρόχος, που θα περιέχει τους κόμβους *s* και *d*. Ο όρος α_5 έχει προταθεί από τους *Park Dong-Chul* και *Keum Kyo-Reen* [26] έτσι ώστε το διάνυσμα ροής $\boldsymbol{V} = [v_{ij}]$ να έχει μια κατεύθυνση για κάθε προορισμό. Αυτό σημαίνει πως εάν υπάρχει ένα μονοπάτι από τον κόμβο *i* στον *j*, τότε ο όρος α_5 αποτρέπει κάθε μονοπάτι προς τον *i*, από όλους τους γειτονικούς του κόμβους συμπεριλαμβανομένου του *j*, δηλαδή αποτρέπει την ύπαρξη βρόγχων ή διαμερίσεων στην προτεινόμενη διαδρομή. Τέλος, στην παρούσα εργασία εισάγουμε τον όρο α_6, ο οποίος, όταν η λύση τείνει να συμπεριλάβει τον σύνδεσμο από τον κόμβο *j* προς έναν κόμβο *k,* ενισχύει την ενεργοποίηση του νευρώνα (*i, j*) ούτως ώστε η διαδρομή, που θα προκύψει, να είναι συνεχής.

5.3 Μετασχηματισμός συνάρτησης ενέργειας

Η συνάρτηση ενέργειας της προηγούμενης παραγράφου πρέπει να εκφραστεί στα συναπτικά βάρη και τα ρεύματα εισόδου του δικτύου Hopfield. Ξαναγράφοντας τις εξισώσεις (4.8), (4.9) και (4.12) χρησιμοποιώντας διπλούς δείκτες, αφού στην περίπτωση μας κάθε νευρώνας αναπαριστάται με έναν διπλό δείκτη (*i,j*), έχουμε :

$$\frac{du_{ij}}{dt} = -\frac{u_{ij}}{\tau} + \sum_{x=1}^{N}\sum_{\substack{y=1 \\ y\neq x}}^{N} T_{ij,xy} v_{ij} + I_{ij} \qquad (5.4)$$

$$\frac{du_{ij}}{dt} = -\frac{u_{ij}}{\tau} - \frac{\partial E}{\partial v_{ij}} \qquad (5.5)$$

$$v_{ij} = f_{ij}\left(u_{ij}\right) \qquad (5.6)$$

Αντικαθιστώντας τη συνάρτηση ενέργειας της εξίσωσης (5.3) στην εξίσωση (5.5) λαμβάνουμε:

$$\frac{du_{ij}}{dt} = -\frac{u_{ij}}{\tau} - \frac{a_1}{2} w_{ij}\left(1 - \delta_{id}\delta_{js}\right) - a_2 \sum_{\substack{k=1 \\ k\neq i}}^{N} (v_{ik} - v_{ki}) + a_2 \sum_{\substack{k=1 \\ k\neq i}}^{N} \left(v_{jk} - v_{kj}\right)$$

$$- \frac{a_3}{2}\left(1 - 2v_{ij}\right) + \frac{a_4}{2}\delta_{id}\delta_{js} - \frac{a_5}{2} \sum_{\substack{k=1 \\ k\neq i}}^{N} v_{ki} - \frac{a_6}{2}\left(v_{ij} - \sum_{\substack{k=1 \\ k\neq i}}^{N} v_{jk} \right) \quad (5.7)$$

Όπου δ είναι η συνάρτηση δέλτα του Kronecker :

$$\delta_{ij} = \begin{cases} 1, & \gamma\iota\alpha\ i = j \\ 0, & \gamma\iota\alpha\ i \neq j \end{cases} \qquad (5.8)$$

Συγκρίνοντας τις εξισώσεις (5.4) και (5.7) προκύπτουν οι σχέσεις για τα συναπτικά βάρη και τα ρεύματα εισόδου ως εξής :

$$T_{ij,xy} = -\,a_2\delta_{ix} + a_2\delta_{jx} + a_2\delta_{yi} - a_2\delta_{jy} + a_3\delta_{ix}\delta_{jy} - \frac{a_5}{2} \sum_{\substack{k=1 \\ k\neq i}}^{N} \delta_{ki}$$

$$- \frac{a_6}{2}\left(\delta_{ix}\delta_{jy} - \sum_{\substack{k=1 \\ k\neq i}}^{N} \delta_{jk} \right) \qquad (5.9)$$

$$I_{ij} = -\,a_1 w_{ij}\left(1 - \delta_{id}\delta_{js}\right) - \frac{a_3}{2} + \frac{a_4}{2}\delta_{id}\delta_{js} \qquad (5.10)$$

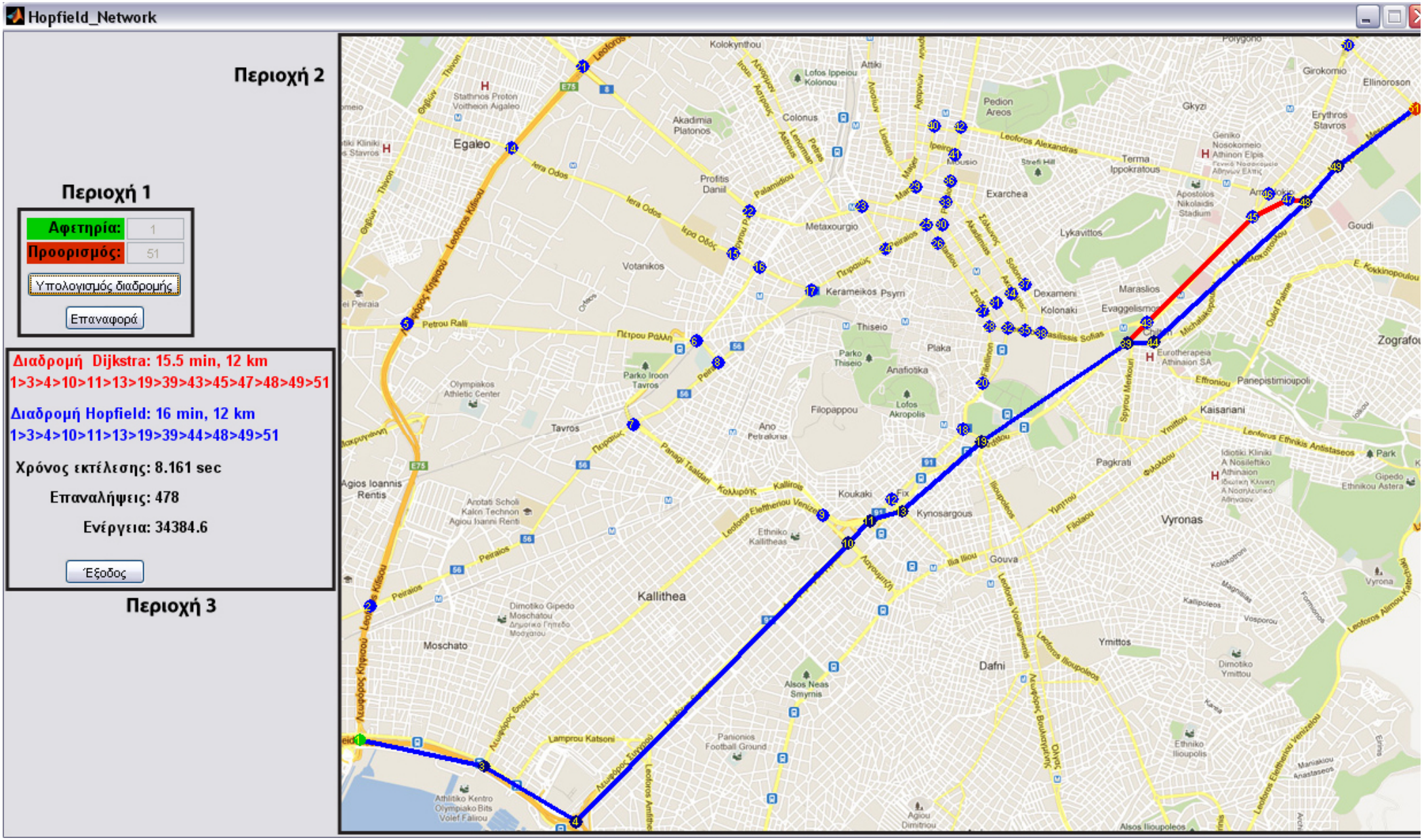
Παρατηρώντας τις εξισώσεις (5.9) και (5.10) διαπιστώνουμε ότι τα συναπτικά βάρη είναι ανεξάρτητα του κόστους των συνδέσμων, το οποίο εκφράζεται στα ρεύματα εισόδου. Το πλεονέκτημα του μοντέλου αυτού είναι πολύ σημαντικό, διότι το κόστος των σ υνδέσμων και η τοπολογία του δικτύου μπορούν να προσαρμοστούν στην τρέχουσα κατάσταση του οδικού δικτύου απλά τροποποιώντας τα ρεύματα εισόδου, χωρίς να είναι απαραίτητη η μεταβολή των εσωτερικών παραμέτρων του νευρωνικού δικτύου, όπως είναι τα συναπτικά β άρη, γεγονός, που το καθιστά ικανό να λειτουργήσει σε προβλήματα πραγματικού χρόνου. Επιπρόσθετα, στην περίπτωση υλοποίησης του προτεινόμενου δικτύου Hopfield σε υλικό, το ολοκληρωμένο κύκλωμα είναι ουσιαστικά ένα, ανεξάρτητο της τοπολογίας του δικτύου, του κόστους των συνδέσμων καθώς και του αλγόριθμου.

6 *Εφαρμογή δρομολόγησης σε οδικό δίκτυο*

Για την πειραματική μελέτη, καθώς και για την οπτική παρουσίαση των διαδρομών και τ ων αποτελεσμάτων, του προτεινόμενου δικτύου Hopfield αναπτύχθηκε, στο περιβάλλον προγραμματισμού MATLAB, μία εφαρμογή, που παρέχει μία γραφική διεπαφή (Graphical User Interface - GUI) μέσω της οποίας ο χρήστης μπορεί να υπολογίζει τις βέλτιστες διαδρομές μεταξύ δύο κόμβων σε ένα τμήμα του οδικού δικτύου της Αθήνας.

Το GUI, το οποίο απεικονίζεται στην εικόνα 6.1, αποτελείται ουσιαστικά από τρεις διαφορετικές περιοχές. Στην περιοχή 1, ο χρήστης μπορεί να εισάγει τους κόμβους αφετηρίας και προορισμού για τον υπολογισμό της διαδρομής ή να επαναφέρει το δίκτυο Hopfield στην αρχική του κατάσταση. Στην περιοχή 2 απεικονίζεται ο χάρτης του οδικού δικτύου, στον οποίο απαριθμούνται οι κυριότεροι κόμβοι, ενώ εκεί σχεδιάζονται και οι λύσεις, που θα προκύψουν. Τέλος, στην περιοχή 3 εμφανίζονται τα όποια αριθμητικά αποτελέσματα.

Για τον υπολογισμό της διαδρομής μεταξύ δύο κόμβων ο χρήστης εισάγει στα αντίστοιχα πεδία της περιοχής 1 τους αριθμούς αυτών και κατόπιν επιλέγει το κουμπί «Υπολογισμός διαδρομής». Στην εικόνα 6.1 το GUI έχει χρησιμοποιηθεί για την εύρεση της διαδρομής από τον κόμβο 1 προς τον κόμβο 51. Οι τιμές των κόμβων αφετηρίας και προορισμού έχουν κλειδώσει, ενώ στην περιοχή 2 έχει σχεδιαστεί με κόκκινο χ ρώμα η βέλτιστη διαδρομή, όπως προκύπτει από τον αλγόριθμο Dijkstra, και με μπλε χρώμα η διαδρομή, που δίνει ως λύση το δίκτυο Hopfield, καθώς και το



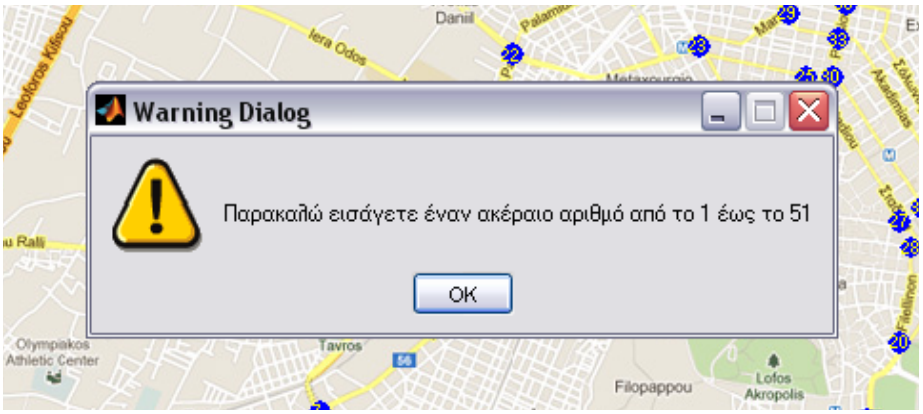
Εικόνα 6.1: GUI

κοινό τους τμήμα. Επίσης οι κόμβοι αφετηρίας και προορισμού, καθώς και κόμβοι από τους οποίους διέρχεται η διαδρομή Hopfield σημειώνονται με διαφορετικό χρώμα, σε σχέση με τους υπόλοιπους. Στην περιοχή 3, εμφανίζεται ο εκτιμώμενος χρόνος και η απόσταση, τόσο για τη βέλτιστη διαδρομή, όσο και για τη διαδρομή του δικτύου Hopfield. Σε κάθε περίπτωση περιλαμβάνεται και η αλληλουχία των κόμβων, που συνθέτουν την υπολογιζόμενη διαδρομή. Στο συγκεκριμένο παράδειγμα παρατηρούμε πως, ενώ οι δύο διαδρομές ταυτίζονται μέχρι τον κόμβο 39, στη συνέχεια διαφοροποιούνται για να συναντηθούν πάλι στον κόμβο 48. Τέλος, ο χρόνος και οι επαναλήψεις, που απαιτήθηκαν, για να συγκλίνει το νευρωνικό δίκτυο σε μία ευσταθή κατάσταση, καθώς και η τελική τιμή της συνάρτησης ενέργειας συμπληρώνουν τα αποτελέσματα, τα οποία αποτελούν την περιοχή 3.

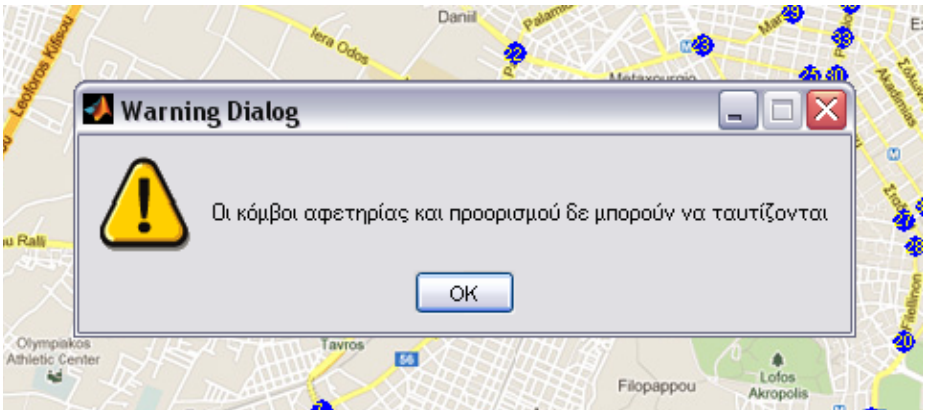
Εικόνα 6.2: Αρχικοποιημένο GUI

Για να καταστεί δυνατή η εισαγωγή από τον χρήστη ενός νέου ζευγαριού κόμβων, αφού έχει προηγηθεί κάποιος υπολογισμός διαδρομής, είναι απαραίτητη η επιλογή του κουμπιού «Επαναφορά». Αυτό έχει σαν αποτέλεσμα τη διαγραφή των προηγούμενων κόμβων αφετηρίας και προορισμού και το ξεκλείδωμα των αντίστοιχων πεδίων. Επιπρόσθετα, σβήνονται όλα τα αποτελέσματα καθώς και η σχεδιασμένη διαδρομή, ενώ το δίκτυο Hopfield επανέρχεται στην αρχική του κατάσταση (εικόνα 6.2). Τέλος, το κλείσιμο της εφαρμογής πραγματοποιείται μέσω του κουμπιού «Έξοδος».

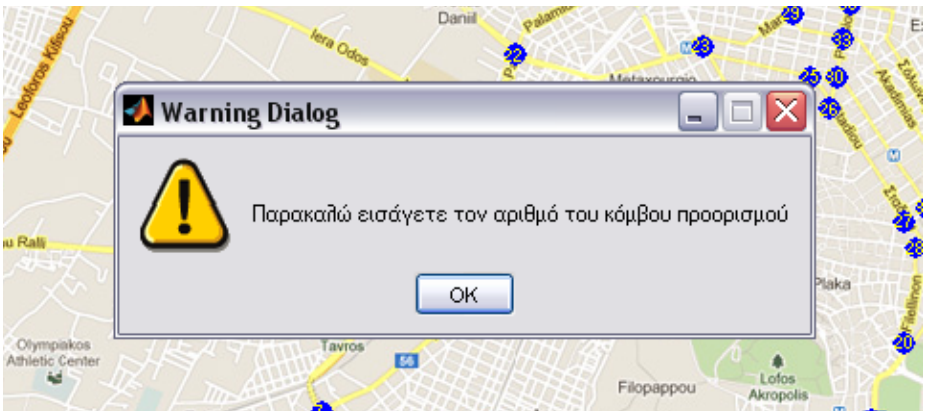
Κατά την εισαγωγή των δεδομένων πραγματοποιείται έλεγχος για την ορθότητα τους και ενημερώνεται ο χρήστης με κατάλληλα μηνύματα στην περίπτωση, που είναι λανθασμένα. Αυτό συμβαίνει όταν ο αριθμός κάποιου κόμβου δεν είναι ακέραιος στο διάστημα [1,51] (εικόνα 6.3α), οι κόμβοι αφετηρίας και προορισμού ταυτίζονται (εικόνα 6.3β) ή δεν έχει συμπληρωθεί κάποιο πεδίο (εικόνα 6.3γ).



(α)

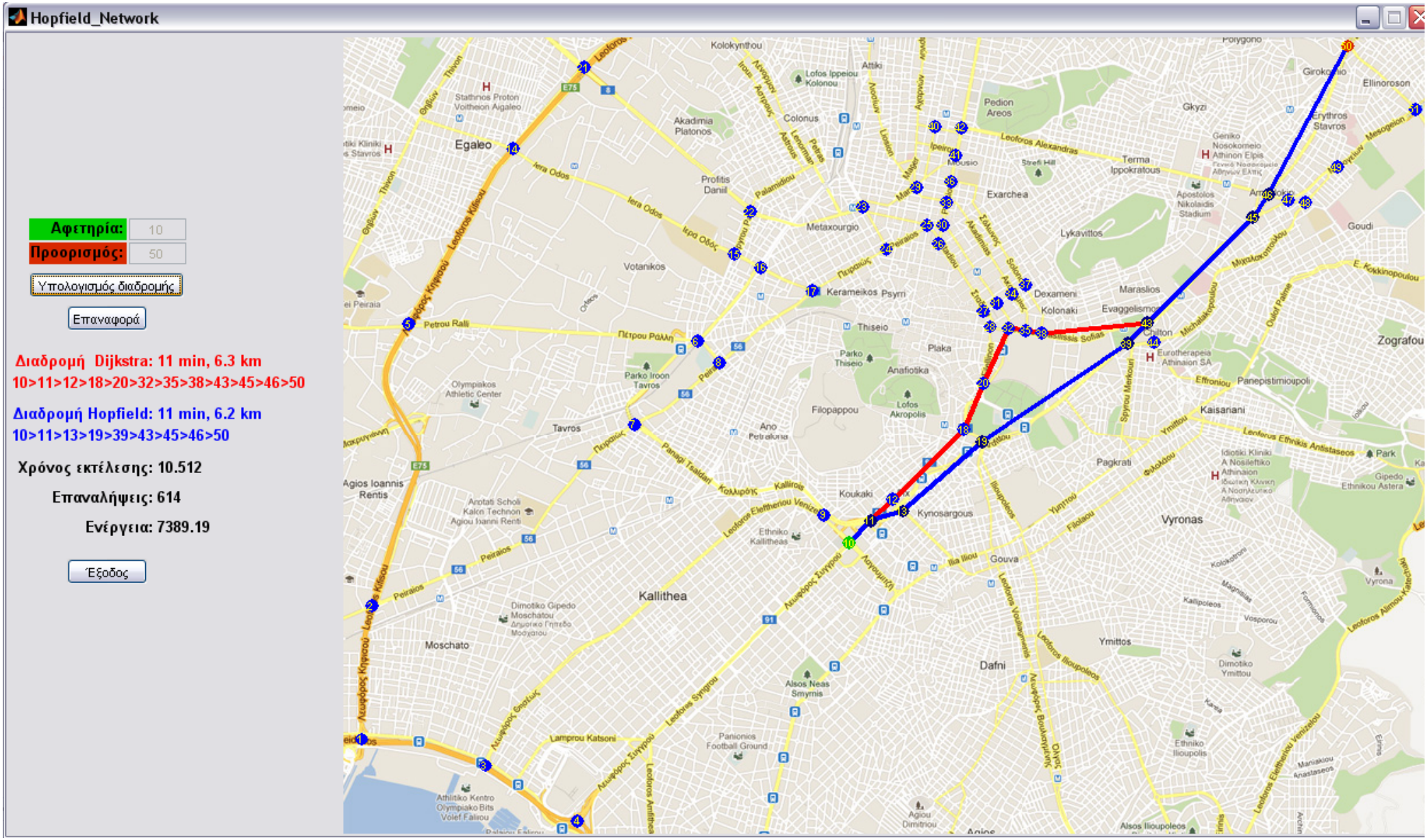


(β)



(γ)

Εικόνα 6.3: Μηνύματα σφάλματος



Εικόνα 6.4: Παράδειγμα υπολογισμού με δύο βέλτιστες διαδρομές

Η βελτιστότητα των διαδρομών, που υπολογίζει η συγκεκριμένη εφαρμογή, αφορά την ελαχιστοποίηση του χρόνου ταξιδίου, και όχι της απόστασης. Η τιμή της απόστασης, η οποία εμφανίζεται στα αποτελέσματα, προκύπτει μετά την εύρεση της διαδρομής, αθροίζοντας τις αποστάσεις των επί μέρους τμημάτων, που την απαρτίζουν. Για ένα δεδομένο ζεύγος (*s,d*) οι δύο προκύπτουσες διαδρομές, ενδέχεται να έχουν τον ίδιο χρόνο ταξιδίου αλλά διαφορετική αλληλουχία κόμβων. Αυτό συμβαίνει στο παράδειγμα υπολογισμού της εικόνας 6.4, όπου, αν και οι διαδρομές διαφέρουν, ο εκτιμώμενος χρόνος είναι ίδιος. Στην περίπτωση αυτή, προφανώς υπάρχουν παραπάνω από μία βέλτιστες διαδρομές μεταξύ των κόμβων αυτών και τυγχάνει το δίκτυο Hopfield να επιλέξει διαφορετική από αυτήν που έχει προκύψει με βάση τον αλγόριθμο Dijkstra.

Αξίζει, επίσης, να σημειωθεί πως οι βέλτιστες διαδρομές του αλγόριθμου Dijkstra και το κόστος τους, έχουν υπολογιστεί εκ των προτέρων, ούτως ώστε να μην επιβαρύνουν την εφαρμογή, ενώ ο χρόνος σύγκλισης του ΝΔ αφορά σε σειριακή, και όχι παράλληλη ενημέρωση των νευρώνων.

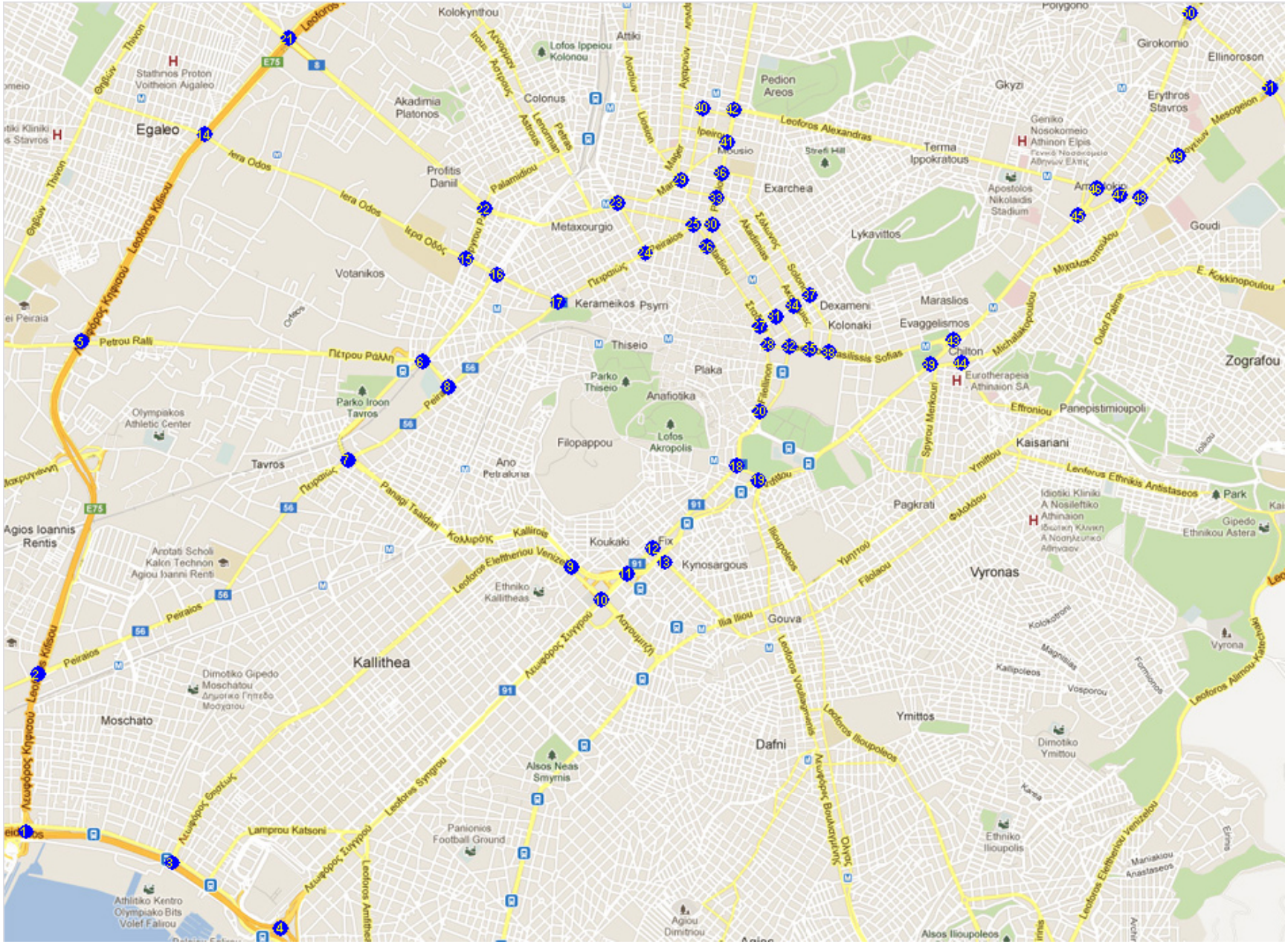
Η παραπάνω εφαρμογή, με τις κατάλληλες τροποποιήσεις, μπορεί να χρησιμοποιηθεί για τον υπολογισμό διαδρομών και σε άλλα οδικά δίκτυα, πέραν της Αθήνας. Ο αντίστοιχος κώδικας παρατίθεται στο Παράρτημα εμπλουτισμένος με αναλυτικά σχόλια για την πληρέστερη και ευκολότερη κατανόησή του, ενώ ο κώδικας, που χρησιμοποιείται, για τον υπολογισμό των βέλτιστων διαδρομών και του κόστους τους, με βάση τον αλγόριθμο Dijkstra μπορεί να βρεθεί εδώ [27].

7 *Πειραματική μελέτη και αποτελέσματα*

Η πειραματική μελέτη της προτεινόμενης ευφυούς μεθόδου δρομολόγησης πραγματοποιείται στο τμήμα του οδικού δικτύου της Αθήνας, της εικόνας 7.1, με την βοήθεια της εφαρμογής του Κεφαλαίου 6. Στην εν λόγω εικόνα έχουν αριθμηθεί οι κόμβοι, οι οποίοι προκύπτουν από τους κεντρικούς δρόμους και αυτοκινητόδρομους, με βάση τη λογική της παραγράφου 5.1. Κάθε τμήμα μεταξύ δύο κόμβων *i, j* έχει ένα κόστος w_{ij}, που προκύπτει από τον χρόνο, ο οποίος απαιτείται για τη διάσχισή του και είναι κανονικοποιημένο στο διάστημα [0, 1]. Όταν δεν υπάρχει σύνδεσμος μεταξύ δύο κόμβων, τότε το κόστος του, w_{ij}, λαμβάνει μια πολύ μεγάλη τιμή, ούτως ώστε να μην περιέχεται στη λύση. Ο στόχος είναι να βρεθεί η βέλτιστη διαδρομή για το εκάστοτε ζεύγος αφετηρίας – προορισμού.

Με γνωστές τις αρχικές εισόδους των νευρώνων u_{ij} για τη χρονική στιγμή $t = 0$, είναι δυνατή η προσομοίωση του νευρωνικού μοντέλου, με τη βοήθεια υπολογιστή, λύνοντας αριθμητικά την εξίσωση (5.4). Με μαθηματικούς όρους, η προσομοίωση αντιστοιχεί στην επίλυση ενός συστήματος *n x (n – 1)* μη γραμμικών διαφορικών εξισώσεων με άγνωστες μεταβλητές τις εξόδους των νευρώνων v_{ij}. Για την επίλυση του συστήματος αυτού εφαρμόζουμε τη μέθοδο του Euler στην εξίσωση (5.4) :

$$u_{ij}(t + \Delta t) = u_{ij}(t) + \Delta t \left(-\frac{u_{ij}}{\tau} + \sum_{x=1}^{N} \sum_{\substack{y=1 \\ y \neq x}}^{N} T_{ij,xy} v_{ij} + I_{ij} \right) \quad (5.11)$$



Εικόνα 7.1: Τμήμα του οδικού δικτύου της Αθήνας

Όπου $T_{ij,xy}$ και I_{ij} οι τιμές των συναπτικών βαρών και των αρχικών ρευμάτων, όπως προκύπτουν από τις εξισώσεις (5.9) και (5.10) αντίστοιχα.

Τέλος, ως συνάρτηση ενεργοποίησης, επιλέγεται η ακόλουθη:

$$v_{ij} = \frac{1}{1 + e^{-\lambda_{ij} u_{ij}}} \quad (5.12)$$

Η προσομοίωση αποτελείται από την παρατήρηση και ενημέρωση των τιμών εξόδου των νευρώνων σε στοιχειώδη χρονικά διαστήματα Δt. Δεν θεωρούμε μηδενικές τις αρχικές τιμές εισόδου των νευρώνων κατά την έναρξη του υπολογισμού, αλλά ενσωματώνουμε μικρή ποσότητα τυχαίου θορύβου :

$$-0.001 \leq \delta u_{ij} \leq +0.001$$

Ο θόρυβος αυτός βοηθάει στη διάσπαση της συμμετρίας και συνιστάται στη βιβλιογραφία των νευρωνικών δικτύων. Η προσομοίωση διακόπτεται όταν το σύστημα φτάσει σε μια ευσταθή κατάσταση, η οποία θεωρούμε ότι επέρχεται όταν οι έξοδοι όλων των νευρώνων δεν μεταβάλλονται περισσότερο από μία οριακή τιμή ΔV, μεταξύ των ενημερώσεων. Στην ευσταθή κατάσταση θεωρούμε ότι ένας νευρώνας είναι ενεργός, όταν η τιμή εξόδου του είναι μεγαλύτερη του 0.5. Σε αντίθετη περίπτωση ο νευρώνας θεωρείται ανενεργός.

7.1 Ρύθμιση παραμέτρων

Κατά τον σχεδιασμό ενός δικτύου Hopfield η σωστή επιλογή των παραμέτρων είναι εξέχουσας σημασίας, διότι η απόδοσή του είναι ιδιαίτερα ευαίσθητη στις τιμές αυτών. Παρά την πιθανότητα, η κατάσταση του δικτύου να πέσει σε τοπικό ελάχιστο, η κατάλληλη ρύθμιση των παραμέτρων μπορεί να εξαλείψει ή τουλάχιστον να μειώσει αυτό το ενδεχόμενο.

Αρχικά επιλέγουμε μια συγκριτικά μεγάλη τιμή για την παράμετρο α_1. Η αύξηση της τιμή της παραμέτρου αυτής θα βελτιώσει σταδιακά την ποιότητα της λύσης, αλλά η υπέρβαση ενός συγκεκριμένου ορίου οδηγεί σε μη έγκυρες διαδρομές. Αυτό συμβαίνει διότι παρακάμπτει τον όρο α_2, μια υψηλή τιμή του οποίου είναι απαραίτητη για να είναι η λύση αποδεκτή. Μια μεγάλη τιμή του όρου α_3 βοηθάει στη γρήγορη σύγκλιση, αλλά αυξάνει την πιθανότητα να παγιδευτεί το νευρωνικό δίκτυο

σε ένα τοπικό ελάχιστο. Όπως αναφέρθηκε στην παράγραφο 5.3, ο όρος α_4 χρησιμεύει για να υπάρχουν οι κόμβοι αφετηρίας και προορισμού στη λύση, συνεπώς πρέπει να είναι αρκετά μεγάλος, ώστε από τα πρώτα στάδια του υπολογισμού ο νευρώνας *(d,s)* να είναι ενεργός. Σύμφωνα με τους *Park Dong-Chul* και *Keum Kyo-Reen,* ο όρος α_5 αποτρέπει τον σχηματισμό βρόχων και διαμερίσεων, όμως μια πολύ μεγάλη τιμή του θα αποτρέψει και τον σχηματισμό της διαδρομής αυτής καθαυτής. Από την άλλη, ο όρος α_6, που προτείνεται σε αυτή την εργασία, βοηθάει στον σχηματισμό της διαδρομής, αλλά η υπερβολική αύξησή του μπορεί να οδηγήσει στην εμφάνιση βρόχων και διαμερίσεων.

Όσον αφορά τις λοιπές παραμέτρους της προσομοίωσης μπορούμε να θέσουμε ίση με τη μονάδα τη χρονική σταθερά τ κάθε νευρώνα χωρίς βλάβη της γενικότητας, ενώ επιλέγοντας το στοιχειώδες χρονικό διάστημα *Δt* και το κριτήριο διακοπής *ΔV* ίσα με 10^{-4} λαμβάνουμε ικανοποιητικά αποτελέσματα. Τέλος, δεδομένου ότι η τιμή της παραμέτρου λ στη συνάρτηση ενεργοποίησης, είναι ανάλογη της ταχύτητας σύγκλισης και αντιστρόφως ανάλογη της ποιότητας της λύσης επιλέξαμε μία χαμηλή τιμή, ίση με τη μονάδα, για να επιτευχθούν καλύτερα αποτελέσματα, αλλά με κόστος τον υψηλότερο χρόνο εκτέλεσης. Γενικά, η παράμετρος αυτή πρέπει να ρυθμίζεται με βάση τις απαιτήσεις κάθε εφαρμογής σε ταχύτητα και ακρίβεια.

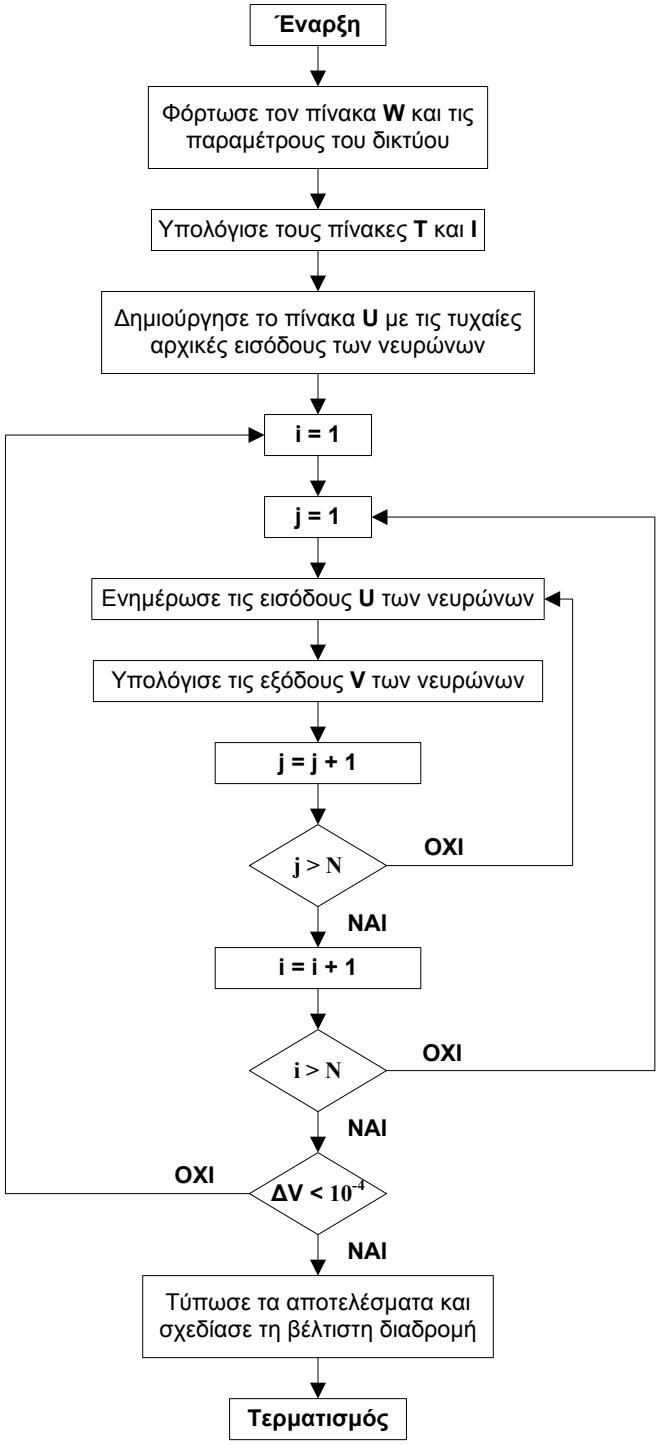
Λαμβάνοντας υπόψη τις παραπάνω γενικές αρχές και μετά από δοκιμές (*trial and error*) προέκυψαν οι ακριβείς τιμές των παραμέτρων, που επιλέχθηκαν για την προσομοίωση, οι οποίες φαίνονται στον Πίνακα 7.1.

Παράμετρος	**Σύμβολο**	**Τιμή**
Χρονική σταθερά	τ	1
Κλίση συνάρτησης ενεργοποίησης	λ	1
Στοιχειώδες χρονικό διάστημα	Δt	10^{-4}
Κριτήριο διακοπής	ΔV	10^{-4}
Συντελεστές βαρύτητας	a_1	1380
	a_2	26550
	a_3	560
	a_4	10000
	a_5	520
	a_6	600

Πίνακας 7.1 : Παράμετροι προσομοίωσης

7.2 Αλγόριθμος

Το διάγραμμα ροής του αλγόριθμου, που χρησιμοποιήθηκε στην προσομοίωση για την εύρεση της συντομότερης διαδρομής απεικονίζεται στο σχήμα 7.1. Από το διάγραμμα διαπιστώνουμε ότι οι νευρώνες ενημερώνονται με τη σειρά έως ότου επιτευχθεί ΔV μικρότερο του 10^{-4} για κάθε έναν από αυτούς.



Σχήμα 7.1: Διάγραμμα ροής αλγόριθμου

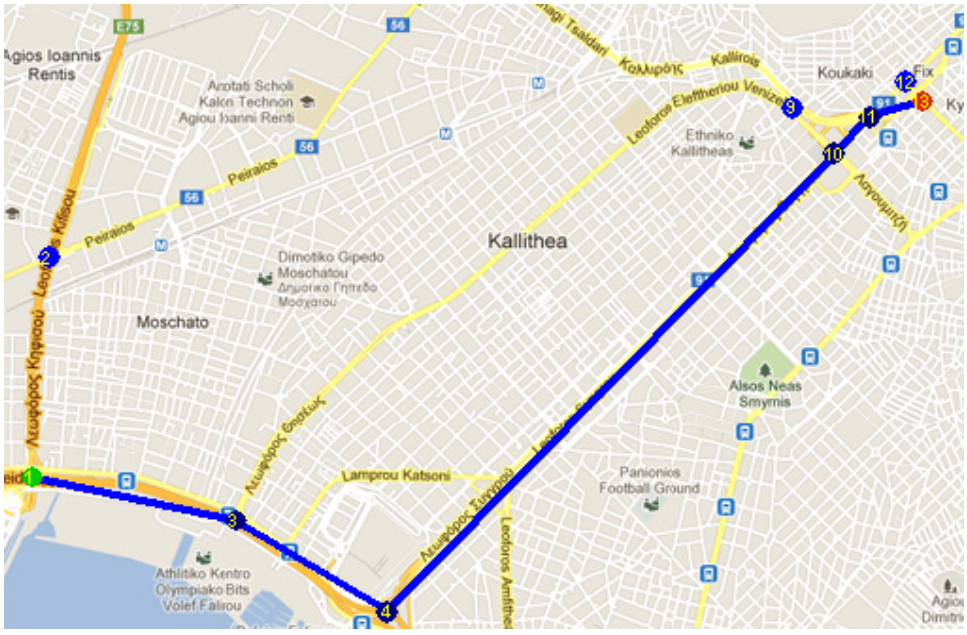
7.3 Αποτελέσματα προσομοίωσης

Κατά την προσομοίωση εκτελέσαμε τον προτεινόμενο αλγόριθμο για κάθε ζεύγος αφετηρίας - προορισμού στο τμήμα του οδικού δικτύου της Αθήνας της εικόνας 7.1, δηλαδή για συνολικά 2550 διαδρομές.

Στον πίνακα 7.2 φαίνεται η έξοδος του δικτύου Hopfield για τη διαδρομή από τον κόμβο 1 προς τον κόμβο 13, ενώ στην εικόνα 7.2 η λύση αυτή, που εν προκειμένω είναι η βέλτιστη, απεικονίζεται γραφικά, αφού πρώτα έχει αποκωδικοποιηθεί.

	1	2	3	4	5	6	7	8	9	10	11	12	13	...
1	0	0	1	0	0	0	0	0	0	0	0	0	0	...
2	0	0	0	0	0	0	0	0	0	0	0	0	0	...
3	0	0	0	1	0	0	0	0	0	0	0	0	0	...
4	0	0	0	0	0	0	0	0	0	1	0	0	0	...
5	0	0	0	0	0	0	0	0	0	0	0	0	0	...
6	0	0	0	0	0	0	0	0	0	0	0	0	0	...
7	0	0	0	0	0	0	0	0	0	0	0	0	0	...
8	0	0	0	0	0	0	0	0	0	0	0	0	0	...
9	0	0	0	0	0	0	0	0	0	0	0	0	0	...
10	0	0	0	0	0	0	0	0	0	0	1	0	0	...
11	0	0	0	0	0	0	0	0	0	0	0	0	1	...
12	0	0	0	0	0	0	0	0	0	0	0	0	0	...
13	1	0	0	0	0	0	0	0	0	0	0	0	0	...
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...

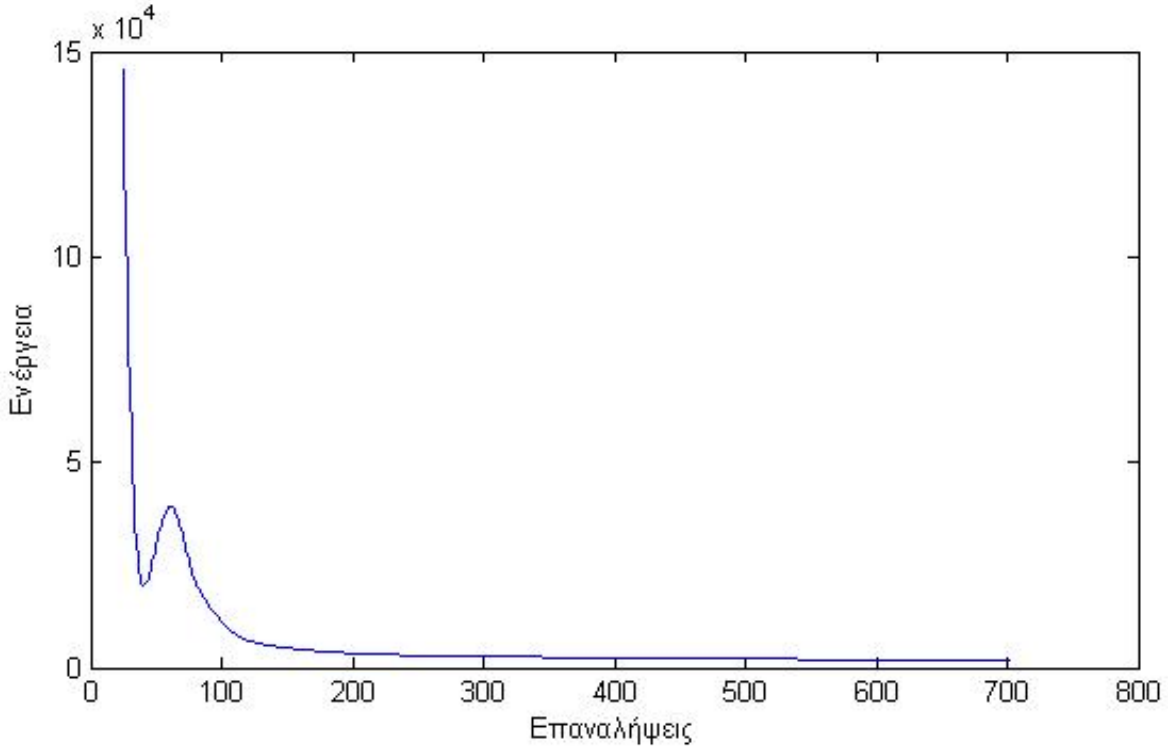
Πίνακας 7.2: Έξοδος δικτύου Hopfield



Εικόνα 7.2: Γραφική αναπαράσταση της εξόδου του δικτύου Hopfield

Στον πίνακα 7.2 παρατηρούμε ότι για κάθε οδικό τμήμα, που συμμετέχει στη λύση, ο αντίστοιχος νευρώνας ενεργοποιείται λαμβάνοντας την τιμή 1, ενώ οι υπόλοιποι είναι ανενεργοί με έξοδο ίση με το 0. Οι τιμές των νευρώνων, που δεν φαίνονται λόγω έλλειψης χώρου, θεωρούνται ίσες με το μηδέν.

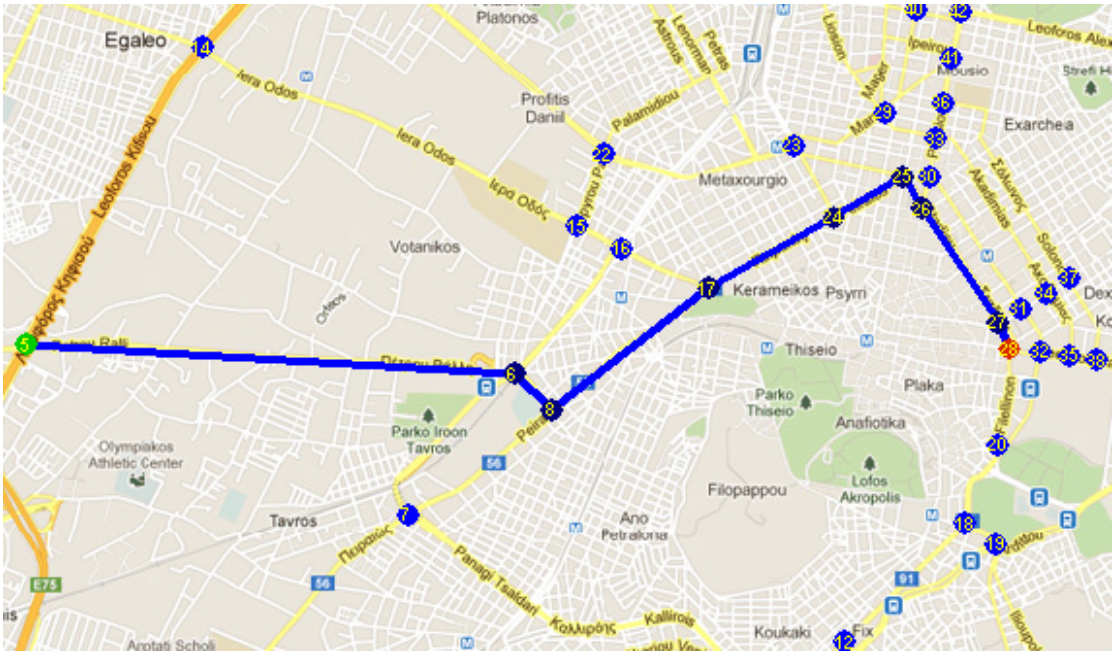
Η ενέργεια του δικτύου συναρτήσει του αριθμού επαναλήψεων του αλγόριθμου ενημέρωσης των νευρώνων απεικονίζεται στο σχήμα 7.2. Παρατηρώντας το διάγραμμα διαπιστώνουμε πως η συνάρτηση ενέργειας ακολουθεί μια φθίνουσα πορεία, ενώ καταφέρνει να απεγκλωβιστεί από το τοπικό ελάχιστο, στο οποίο πέφτει στους πρώτους κύκλους λειτουργίας του νευρωνικού δικτύου.



Σχήμα 7.2: Διάγραμμα ενέργειας συναρτήσει του αριθμού επαναλήψεων

Στην εικόνα 7.3 απεικονίζεται ένα ακόμα παράδειγμα υπολογισμού από τον κόμβο 5 προς τον κόμβο 28.

Ο αλγόριθμος κατάφερε να συγκλίνει σε μια ευσταθή κατάσταση επιστρέφοντας έγκυρη διαδρομή για όλα τα *(s, d)* μετά από, κατά μέσο όρο, 472 επαναλήψεις και 9,2 δευτερόλεπτα. Στον πίνακα 7.3 παρουσιάζονται αναλυτικά τα αποτελέσματα της προσομοίωσης όσον αφορά το υπολογιστικό κόστος του αλγόριθμου.



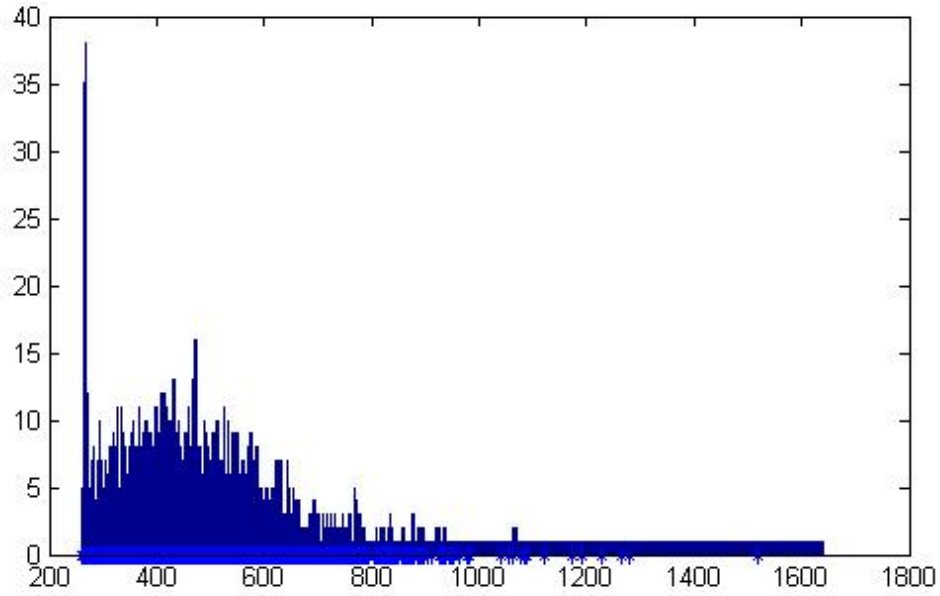
Εικόνα 7.3: Παράδειγμα υπολογισμού από τον κόμβο 5 προς τον κόμβο 28

	Μέσος όρος	**Τυπική απόκλιση**	**Max**	**Min**
Επαναλήψεις	472	153	1520	267
Χρόνος εκτέλεσης (s)	9,2	3	29,9	4,6

Πίνακας 7.3: Υπολογιστικό κόστος

Στον παραπάνω πίνακα παρατηρούμε, ότι για να συγκλίνει το δίκτυο σε μια ευσταθή κατάσταση, στο 68% των διαδρομών απαιτήθηκαν 472 ± 153 επαναλήψεις και 9,2 ± 3 δευτερόλεπτα, ενώ στο 95% των διαδρομών 507 ± 306 επαναλήψεις και 9,2 ± 6 δευτερόλεπτα. Οι, όποιες, ακραίες τιμές εντοπίζονται κυρίως στο δεξί άκρο της κατανομής, που απεικονίζεται στο σχήμα 7.3, και οφείλονται περισσότερο στη δυσκολία του νευρωνικού, στην προκειμένη περίπτωση, να επιλέξει μεταξύ δύο πιθανών διαδρομών και λιγότερο στο μήκος της διαδρομής μεταξύ της αφετηρίας και του προορισμού.

Συγκρίνοντας τη διαδρομή, την οποία δίνει ως έξοδο το δίκτυο Hopfield, με τη βέλτιστη διαδρομή, όπως προκύπτει από τον αλγόριθμο Dijkstra, διαπιστώνουμε την ταύτιση των διαδρομών για το 82,2% των ζευγών *(s, d)*. Στον πίνακα 7.4 εκτίθενται τα αποτελέσματα της σύγκρισης.



Σχήμα 7.3: Κατανομή αριθμού επαναλήψεων

	Ποσοστό (%)
Βέλτιστη	82,2
ΔΤ < 1 min	5,1
1 min < ΔΤ < 2 min	5,1
2 min < ΔΤ < 3 min	4,2
3 min < ΔΤ < 4 min	2,0
4 min < ΔΤ < 5 min	0,5
ΔΤ > 5 min	0,9

Πίνακας 7.4: Σύγκριση της εξόδου του δικτύου Hopfield με τη βέλτιστη διαδρομή

Από τον πίνακα 7.4 διαπιστώνουμε ότι η χρονική διαφορά μεταξύ της λύσης, που προτείνει το δίκτυο Hopfield, όταν αυτή δεν είναι η βέλτιστη, και της λύσης του αλγόριθμου Dijkstra, στην πλειονότητα των περιπτώσεων (14,4%), δεν ξεπερνά τα 3 λεπτά. Ειδικότερα, χρονικές διαφορές μεγαλύτερες των 5 λεπτών απαντώνται μόνο στο 0,9% των περιπτώσεων.

8 *Συμπεράσματα και μελλοντικές επεκτάσεις*

Στην παρούσα διπλωματική εργασία προτάθηκε μία ευφυής μέθοδος για τον υπολογισμό της συντομότερης διαδρομής σε οδικά δίκτυα, η οποία βασίζεται σε ένα νευρωνικό δίκτυο Hopfield με τροποποιημένη συνάρτηση ενέργειας. Στη συνέχεια πραγματοποιήθηκαν προσομοιώσεις για όλες τις διαδρομές σε ένα τμήμα του οδικού δικτύου της Αθήνας, που περιλάμβανε 51 κεντρικούς κόμβους. Τα αποτελέσματα δείχνουν πως ο αλγόριθμος συγκλίνει, σε όλες τις περιπτώσεις, σε μια έγκυρη διαδρομή, ενώ, επί των πλείστων, αυτή η διαδρομή είναι και η βέλτιστη.

Ο χρόνος εκτέλεσης του αλγόριθμου είναι σχετικά υψηλός, αλλά αυτό οφείλεται στο ότι εκτελείται σειριακά σε υπολογιστή και δεν εκμεταλλευόμαστε την ταχύτητα, που προσφέρει η παράλληλη λειτουργία των νευρώνων. Συγκεκριμένα, κατά την προσομοίωση, σε έναν κύκλο λειτουργίας ενημερώνονται με τη σειρά $N\,x\,(N-1) = 2550$ νευρώνες, κάτι το οποίο είναι ιδιαίτερα χρονοβόρο. Στην περίπτωση όμως, που το νευρωνικό δίκτυο υλοποιηθεί σε VLSI, όλοι οι νευρώνες θα ενημερώνονται παράλληλα με αποτέλεσμα η ταχύτητα απόκρισης να περιορίζεται μόνο από την ταχύτητα διάδοσης των σημάτων στο υλικό και να είναι δυνατή η χρήση του σε εφαρμογές πραγματικού χρόνου.

Κατά την υλοποίηση του νευρωνικού δικτύου σε υλικό απαιτείται χώρος αποθήκευσης μόνο για έναν πίνακα *N x N,* έτσι ώστε να ελέγχεται πότε συγκλίνει σε μία ευσταθή κατάσταση. Επιπρόσθετα, γ ια την εκτέλεση του αλγόριθμου δεν χρειάζεται CPU, διότι το ΝΔ υπολογίζει τη λύση με έναν πλήρως κατανεμημένο

τρόπο. Μάλιστα, ούτε η χρήση ενός Νευρωνικού Επεξεργαστή θα συνεισφέρει ιδιαίτερα, αφού ακόμα και το αναλογικό μοντέλο των *Hopfield* και *Tank* συγκλίνει σε μία ευσταθή κατάσταση σε χρόνο συγκρίσιμο των χρονικών σταθερών του κυκλώματος. Συνεπώς, το προτεινόμενο δίκτυο Hopfield δεν είναι απαιτητικό όσον αφορά την πολυπλοκότητα του υλικού, συγκρινόμενο με τους «συμβατικούς» αλγόριθμους, των οποίων η ταχύτητα απόκρισης έχει άμεση σχέση με την υπολογιστική δύναμη της CPU και τη χωρητικότητα της μνήμης.

Ο πίνακας των συναπτικών βαρών του συγκεκριμένου δικτύου Hopfield είναι ανεξάρτητος της αφετηρίας και του προορισμού, καθώς και του κόστους των συνδέσμων και της τοπολογίας του δικτύου. Αυτό έχει σαν αποτέλεσμα να μην απαιτούνται δυναμικές αλλαγές στην αρχιτεκτονική του ΝΔ, το οποίο μπορεί να προσαρμοστεί στις τροποποιήσεις τόσο του κόστους των συνδέσμων, όσο και της τοπολογίας του δικτύου απλά μεταβάλλοντας τα αρχικά ρεύματα εισόδου. Ως εκ τούτου, δύναται να χρησιμοποιηθεί και σε σενάρια, στα οποία το κόστος διάσχισης των οδικών τμημάτων μεταβάλλεται. Αυτό μπορεί να συμβαίνει στην περίπτωση, που υπάρχει ζωντανή αναμετάδοση των συνθηκών κυκλοφορίας (*Live traffic*) ή το κόστος κάθε οδικού τμήματος υπολογίζεται μέσω συνάρτησης, η οποία λαμβάνει υπόψη της και μεταβλητές παραμέτρους (ημέρα, ώρα, καιρός κ.τ.λ.)

Με βάση τη ν παραπάνω ανάλυση συμπεραίνουμε πως το προτεινόμενο δίκτυο Hopfield χωρίς να υστερεί σε ταχύτητα και ευελιξία σε σχέση με τους «συμβατικούς» υπολογιστικούς αλγόριθμους, υπερτερεί στο κόστος κατασκευής, ενώ ικανοποιεί όλες τις προϋποθέσεις, που αναφέρθηκαν νωρίτερα και απαιτούνται, για να εφαρμοστεί στη δρομολόγηση οχημάτων σε οδικά δίκτυα σε πραγματικό χρόνο και με χαμηλό κόστος.

Καθώς το εν λόγω δίκτυο Hopfield υπολογίζει τις διαδρομές μόνο μεταξύ των κυριότερων κόμβων του οδικού δικτύου, θα πρέπει μελλοντικά να συμπληρωθεί με έναν κατάλληλο υπολογιστικό αλγόριθμο για την εύρεση των διαδρομών από την ακριβή θέση του οχήματος προς τον κεντρικό οδικό άξονα και από τον κεντρικό οδικό άξονα προς τον ακριβή προορισμό. Επίσης, η ύπαρξη ενός κριτηρίου για την εναλλαγή μεταξύ των δύο αυτών μεθόδων είναι απαραίτητη για την αποδοτική λειτουργία του συστήματος.

Η σύγκλιση του δικτύου Hopfield, καθώς και η βελτιστότητα των διαδρομών, μειώνονται με την αύξηση των κόμβων του δικτύου. Σε ένα οδικό δίκτυο, παρά τη διατήρηση μόνο των κυριότερων κόμβων, ο αριθμός τους είναι πολύ μεγάλος. Συνεπώς, το οδικό δίκτυο θα πρέπει να διαιρεθεί σε περιοχές, στα όρια των οποίων η απόκριση του νευρωνικού είναι ικανοποιητική. Επιπρόσθετα, στην περίπτωση, που η αφετηρία και ο προορισμός βρίσκονται σε διαφορετικές περιοχές, θα πρέπει το σύστημα να έχει την ικανότητα να επιλέγει τους κατάλληλους κόμβους εξόδου και εισόδου και να ενώνει τις επιμέρους διαδρομές σε μία συνολική λύση.

Κλείνοντας, οφείλουμε, για μία ακόμα φορά, να τονίσουμε πως η υλοποίηση του δικτύου Hopfield σε υλικό είναι επιβεβλημένη για την προώθηση της ερευνητικής δραστηριότητας στον τομέα αυτό και για την ενδελεχή μελέτη της προτεινόμενης μεθόδου δρομολόγησης.

9 *Παράρτημα*

Ο κώδικας για την εφαρμογή, που παρουσιάστηκε στο Κεφάλαιο 6 χωρίζεται σε 2 αρχεία. Στο hopfield_run.m πραγματοποιούνται οι βασικές λειτουργίες, όπως υπολογισμός της εξόδου του δικτύου Hopfield, σχεδιασμός της διαδρομής κ.τ.λ., ενώ το Hopfield_Network.m αποτελεί την οπτικοποίηση των αποτελεσμάτων. Ακολουθεί ο κώδικας και για τα δύο αρχεία εφοδιασμένος με αναλυτικά σχόλια.

9.1 Hopfield_run.m

```
% hopfield_run.m
%
% Λαμβάνει ως είσοδο από το GUI τους κόμβους αφετηρίας και προορισμού
% και χρησιμοποιεί ένα δίκτυο Hopfield για να υπολογίσει τη βέλτιστη
% διαδρομή
%
%--------------------------------------------------------------------%

% Καλεί το GUI για να εισάγει ο χρήστης τους κόμβους αφετηρίας και
% προορισμού
sd = Hopfield_Network;
s = sd(1);
d = sd(2);
```

```
% Φόρτωση παραμέτρων και τοπολογίας δικτύου
load('A.mat')
load('W.mat')

N = length(W); % Αριθμός κόμβων

% Υπολογισμός των αρχικών τιμών εισόδου των νευρώνων
U = (2*rand(N)/1000 - ones(N)/1000);
U = U - diag(diag(U));

% Αρχικοποιεί τον πίνακα με τις τιμές εξόδου
V = zeros(N);
V_old = V;

dt = 0.0001; % Παράμετρος Δt της μεθόδου Euler
flag = 1; % Χρησιμοποιείται για τη διακοπή του βρόγχου while
n = 0; % Αριθμός επαναλήψεων
tic; % Έναρξη χρονομέτρου

% Υπολογισμός τιμών εξόδου με χρήση της μεθόδου Euler
while flag
    flag = 0;
    for i=1:N
        for j=1:N
            if i~=j
                A_2_1 = 0;
                A_2_2 = 0;
                A_5 = 0;
                A_6 = 0;
                % Υπολογισμός των αθροισμάτων
                for x=1:N
                    if x~=i
                        A_2_1 = A_2_1 + (V(i,x) - V(x,i));
                        A_5 = A_5 + V(x,i);
                    end
                    if x~=j
                        A_2_2 = A_2_2 + (V(j,x) - V(x,j));
                        A_6 = A_6 + V(j,x);
                    end
                end
```

```
                % Ενημέρωση της τιμής εισόδου του νευρώνα (i,j)
                U(i,j) = U(i,j) + dt*(-U(i,j) - A(1)*W(i,j)*(1 -
                KronD(i,d)*KronD(j,s))/2 - A(2)*A_2_1 + A(2)*A_2_2 -
                A(3)*(1 - 2*V(i,j))/2 + A(4)*KronD(i,d)*KronD(j,s)/2
                - A(5)*A_5/2 - A(6)*(V(i,j) - A_6)/2);

                % Υπολογισμός της τιμής εξόδου του νευρώνα (i,j)
                V(i,j) = 1/(1 + exp(-1*U(i,j)));

                % Έλεγχος αν ΔV < 0.0001
                if abs(V(i,j) - V_old(i,j)) > 0.0001
                      flag = 1;
                end
            end
        end
    end
    V_old = V;
    n = n + 1;
end

time = toc; % Διακοπή χρονομέτρου

%-------------------------------------------------------------------------%
%-------------------------------------------------------------------------%
% Σχεδίαση διαδρομής
load('dijkstra_paths') % Φόρτωση των διαδρομών του αλγόριθμου Dijkstra

% Φόρτωση τοπολογίας δικτύου
load('g.mat')
load('WX.mat')
load('WY.mat')

% Σχεδίαση διαδρομής Dijkstra
dijkstra_path = cell2mat(dijkstra_paths(s,d));

for i=1:length(dijkstra_path)-1
    line( [WX(dijkstra_path(i)) WX(dijkstra_path(i+1))], ...
          [WY(dijkstra_path(i)) WY(dijkstra_path(i+1))], ...
          'Color', [1.0 0.0 0.0], 'LineWidth', 3 );
end
```

```
% Αποκωδικοποίηση της κατάστασης του δικτύου Hopfield για τον
% σχηματισμό της διαδρομής
l = 1;
hopfield_path = s;
i = s;

while i~=d
     for j=1:N
         if V(i,j) > 0.5
             l = l + 1;
             hopfield_path(l)=j;
             i=j;
             break;
         end
     end
end

% Σχεδίαση διαδρομής Hopfield
for i=1:length(hopfield_path)-1
    line( [WX(hopfield_path(i)) WX(hopfield_path(i+1))], ...
          [WY(hopfield_path(i)) WY(hopfield_path(i+1))], ...
          'Color', [0.0 0.0 1.0], 'LineWidth', 3 );
end

% Σχεδίαση των κόμβων του δικτύου
for i=1:N
    rectangle('Position',[WX(i)-5.5,WY(i)-5.5,11,11],...
              'Curvature',[1,1],...
              'FaceColor','b','EdgeColor', 'none')

    text( 'Position', [WX(i) WY(i) 0], ...
          'HorizontalAlignment','center', ...
          'VerticalAlignment','middle', ...
          'String', i , ...
          'FontSize',7,'FontWeight','bold', ...
          'Color', [1 1 0] );
end
```

```
% Σχεδίαση των κόμβων, που αποτελούν τη διαδρομή του δικτύου Hopfield
for i=1:length(hopfield_path)

    if i == 1
            rectangle('Position',[WX(hopfield_path(i))-5.5,
            WY(hopfield_path(i)) - 5.5,11,11],...
            'Curvature',[1,1],...
            'FaceColor',[0 0.8 0],'EdgeColor', 'none')

    elseif i == length(hopfield_path)
            rectangle('Position',[WX(hopfield_path(i))-5.5,
            WY(hopfield_path(i))-5.5,11,11],...
            'Curvature',[1,1],...
            'FaceColor',[0.847 0.161 0.0],'EdgeColor', 'none')

    else
             rectangle('Position',[WX(hopfield_path(i))-5.5,
             WY(hopfield_path(i))-5.5,11,11],...
             'Curvature',[1,1],...
             'FaceColor',[0 0 0.5],'EdgeColor', 'none')
    end

    text( 'Position', [WX(hopfield_path(i)) WY(hopfield_path(i)) 0],...
          'HorizontalAlignment','center', ...
          'VerticalAlignment','middle', ...
          'String', hopfield_path(i) , ...
          'FontSize',7,'FontWeight','bold', ...
          'Color', [1 1 0] );
end


%--------------------------------------------------------------------%
%--------------------------------------------------------------------%
% Υπολογισμός ενέργειας και κόστος διαδρομών

% Φόρτωση κόστους διαδρομής Dijkstra
load('dijkstra_costs')
load('D')
```

```
dijkstra_length = 0;

for i=1:length(dijkstra_path) - 1
     dijkstra_length = dijkstra_length +
     D(dijkstra_path(i),dijkstra_path(i+1));
end

% Υπολογισμός κόστος διαδρομής Hopfield
hopfield_cost = 0;
hopfield_length = 0;

for i=1:length(hopfield_path) - 1
     hopfield_cost = hopfield_cost +
     W(hopfield_path(i),hopfield_path(i+1));

     hopfield_length = hopfield_length +
     D(hopfield_path(i),hopfield_path(i+1));
end

% Υπολογισμός ενέργειας
A1 = 0;
A2 = 0;
A3 = 0;
A5 = 0;
A6 = 0;
B2 = 0;
C2 = 0;
B5 = 0;
B6 = 0;

for i=1:N
    for j=1:N
        if i==d && j==s
        elseif i~=j
        A1 = A1 + W(i,j)*V(i,j);
        end
```

```
        if i~=j
            B2 = B2 + V(i,j);
            C2 = C2 + V(j,i);
            A3 = A3 + V(i,j)*(1-V(i,j));
            B5 = 0;
            for k=1:N
                if k~=i
                   B5 = B5 + V(k,i);
                end
            end
            B6 = 0;
            for k=1:N
                if k~=j
                   B6 = B6 + V(j,k);
                end
            end
            A5 = A5 + V(i,j)*B5;
            A6 = A6 + (V(i,j)^2/2 - V(i,j)*B6);
        end
    end
    A2 = A2 + (B2 - C2)^2;
end

E = A(1)*A1/2 + A(2)*A2/2 + A(3)*A3/2 + A(4)*(1 - V(d,s))/2 +
A(5)*A5/2 + A(6)*A6/2;


%-------------------------------------------------------------------------%
%-------------------------------------------------------------------------%
% Εκτύπωση αποτελεσμάτων

% Κόστος διαδρομής Dijkstra
dijkstraCost = uicontrol('Style','Text', ...
    'Position', [132 409 100 15], ...
    'BackgroundColor',[0.878 0.875 0.89], ...
    'ForegroundColor', [1 0 0], ...
     'String',sprintf('%g min, %g km',5*dijkstra_costs(s,d),
     dijkstra_length), ...
    'FontSize',10, 'FontWeight', 'bold',...
    'HorizontalAlignment', 'left');
```

```
% Μονοπάτι διαδρομής Dijkstra
dijkstraPath = uicontrol('Style','Text', ...
    'Position', [5 390 (7*length(sprintf('%d',dijkstra_path(1)))) 15], ...
    'BackgroundColor',[0.878 0.875 0.89], ...
    'ForegroundColor', [1 0 0], ...
    'String',sprintf('%d',dijkstra_path(1)), ...
    'FontSize',10, 'FontWeight', 'bold',...
    'HorizontalAlignment', 'left');

x = -3;
for i=2:length(dijkstra_path)
    x = x + 8 + 7*length(sprintf('%d',dijkstra_path(i-1)));

    dijkstraPath = uicontrol('Style','Text', ...
    'Position', [x 390 (8+7*length(sprintf('%d',dijkstra_path(i)))) 15], ...
    'BackgroundColor',[0.878 0.875 0.89], ...
    'ForegroundColor', [1 0 0], ...
    'String',sprintf('>%d',dijkstra_path(i)), ...
    'FontSize',10, 'FontWeight', 'bold',...
    'HorizontalAlignment', 'left');
end

% Κόστος διαδρομής Hopfield
hopfieldCost = uicontrol('Style','Text', ...
    'Position', [132 363 100 15], ...
    'BackgroundColor',[0.878 0.875 0.89], ...
    'ForegroundColor', [0 0 1], ...
     'String',sprintf('%g min, %g km',5*hopfield_cost,
     hopfield_length), ...
    'FontSize',10, 'FontWeight', 'bold',...
    'HorizontalAlignment', 'left');

%Μονοπάτι διαδρομής Hopfield
hopfieldPath = uicontrol('Style','Text', ...
'Position', [5 344 (7*length(sprintf('%d',hopfield_path(1)))) 15], ...
'BackgroundColor',[0.878 0.875 0.89], ...
'ForegroundColor', [0 0 1], ...
'String',sprintf('%d',hopfield_path(1)), ...
'FontSize',10, 'FontWeight', 'bold',...
'HorizontalAlignment', 'left');
```

```
x = -3;
for i=2:length(hopfield_path)
    x = x + 8 + 7*length(sprintf('%d',hopfield_path(i-1)));

    hopfieldPath = uicontrol('Style','Text', ...
    'Position', [x 344 (8+7*length(sprintf('%d',hopfield_path(i)))) 15], ...
    'BackgroundColor',[0.878 0.875 0.89], ...
    'ForegroundColor', [0 0 1], ...
    'String',sprintf('>%d',hopfield_path(i)), ...
    'FontSize',10, 'FontWeight', 'bold',...
    'HorizontalAlignment', 'left');
end

% Χρόνος εκτέλεσης αλγόριθμου δικτύου Hopfield
elapsedTime = uicontrol('Style','Text', ...
    'Position', [132 316 60 15], ...
    'BackgroundColor',[0.878 0.875 0.89], ...
    'ForegroundColor', [0 0 0], ...
    'String',sprintf('%.3f sec',time), ...
    'FontSize',10, 'FontWeight', 'bold',...
    'HorizontalAlignment', 'left');

% Αριθμός επαναλήψεων
iterations = uicontrol('Style','Text', ...
    'Position', [132 290 40 15], ...
    'BackgroundColor',[0.878 0.875 0.89], ...
    'ForegroundColor', [0 0 0], ...
    'String',sprintf('%d',n), ...
    'FontSize',10, 'FontWeight', 'bold',...
    'HorizontalAlignment', 'left');

% Τελική ενέργεια δικτύου Hopfield
energy = uicontrol('Style','Text', ...
    'Position', [132 264 60 15], ...
    'BackgroundColor',[0.878 0.875 0.89], ...
    'ForegroundColor', [0 0 0], ...
    'String',sprintf('%g',E), ...
    'FontSize',10, 'FontWeight', 'bold',...
    'HorizontalAlignment', 'left');
```

9.2 Hopfield_Network.m

```
% Hopfield_Network.m
% GUI, που αποτελείται από το χάρτη με τους κόμβους, πεδία για την
% εισαγωγή των κόμβων αφετηρίας και προορισμού, κουμπιά λειτουργιών
% και χώρο απεικόνισης των αποτελεσμάτων.

function varargout = Hopfield_Network(varargin)

% Έναρξη κώδικα αρχικοποίησης
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @Hopfield_Network_OpeningFcn, ...
                   'gui_OutputFcn',  @Hopfield_Network_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% Τέλος κώδικα αρχικοποίησης


% --- Εκτελείται πριν εμφανιστεί το GUI
function Hopfield_Network_OpeningFcn(hObject, eventdata, handles,
varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to Hopfield_Network (see  VARARGIN)

% Choose default command line output for Hopfield_Network
handles.output = hObject;
```

```
% Δέσμευση χώρου για την αποθήκευση των  κόμβων αφετηρίας και
% προορισμού
handles.sd = zeros(1,2);

% Φορτώση της τοπολογίας του δικτύου
load('W.mat')
load('WX.mat')
load('WY.mat')

handles.N = length(W); % Αριθμός κόμβων

guidata(hObject, handles);

% Σχεδίαση χάρτη
imshow('map.jpg')

% Σχεδίαση κόμβων
for i=1:handles.N
    rectangle('Position',[WX(i)-5.5,WY(i)-5.5,11,11],...
              'Curvature',[1,1],...
              'FaceColor','b', 'EdgeColor', 'none')

    text( 'Position', [WX(i) WY(i) 0], ...
          'HorizontalAlignment','center', ...
          'VerticalAlignment','middle', ...
          'String', i , ...
          'FontSize',7,'FontWeight','bold', ...
          'Color', [1 1 0] );
end

% Αναμονή μέχρι ο χρήστης να εισάγει τους κόμβους αφετηρίας και
% προορισμού
uiwait(handles.figure1);



% Οι τιμές εξόδου της παρακάτω συνάρτησης επιστρέφονται στο hopfield_run
function varargout = Hopfield_Network_OutputFcn(hObject, eventdata,
handles)
% varargout  cell array for returning output args (see VARARGOUT);
```

```
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

varargout{1} = handles.sd; % Έξοδος = (s,d)


function source_Callback(hObject, eventdata, handles)
% hObject    handle to source (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Ανάγνωση του κόμβου αφετηρίας, που εισάγει ο χρήστης
source_node = str2double(get(hObject,'String'));

% Έλεγχος ορθότητας του κόμβου αφετηρίας
if source_node > handles.N || source_node < 1 || mod(source_node,1) ~= 0
   uicontrol(handles.source);
    warndlg(sprint('Παρακαλώ εισάγετε έναν ακέραιο αριθμό από το 1
    έως το %d', handles.N))
end


function source_CreateFcn(hObject, eventdata, handles)
% hObject    handle to source (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


function destination_Callback(hObject, eventdata, handles)
% hObject    handle to destination (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

```
% Ανάγνωση του κόμβου προορισμού, που εισάγει ο χρήστης
destination_node = str2double(get(hObject,'String'));

% Έλεγχος ορθότητας του κόμβου προορισμού
if destination_node > handles.N || destination_node < 1 ||
mod(destination_node,1) ~= 0
   uicontrol(handles.destination);
   warndlg(sprint('Παρακαλώ εισάγετε έναν ακέραιο αριθμό από το 1 έως
    το %d', handles.N))
end



function destination_CreateFcn(hObject, eventdata, handles)
% hObject    handle to destination (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



% --- Εκτελείται με το πάτημα του κουμπιού "Υπολογισμός διαδρομής"
function find_route_Callback(hObject, eventdata, handles)
% hObject    handle to find_route (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

s = str2double(get(handles.source,'string'));
d = str2double(get(handles.destination,'string'));

% Έλεγχος αν έχουν εισαχθεί οι κόμβοι αφετηρίας και προορισμού
if isnan(s)
    uicontrol(handles.source);
    warndlg('Παρακαλώ εισάγετε τον αριθμό του κόμβου αφετηρίας')
```

```
elseif isnan(d)
    uicontrol(handles.destination);
    warndlg('Παρακαλώ εισάγετε τον αριθμό του κόμβου προορισμού')
elseif s==d
    uicontrol(handles.destination);
     warndlg('Οι κόμβοι αφετηρίας και προορισμού δε μπορούν να
     ταυτίζονται')
else
    handles.sd=[s,d]; % Η τιμή αυτή επιστρέφεται στο hopfield_run
    guidata(hObject, handles);

    % Απενεργοποίηση των πεδίων εισόδου
    set(handles.source,'Enable','off');
    set(handles.destination,'Enable','off');

    % Λήξη της αναμονής από την εντολή uiwait
    uiresume
end



% --- Εκτελείται με το πάτημα του κουμπιού "Επαναφορά"
function reset_gui_Callback(hObject, eventdata, handles)
% hObject    handle to reset_gui (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Ενεργοποίηση των πεδίων εισόδου
set(findobj('style','edit'), 'String', {''}, 'Enable', 'on');

% Διαγραφή των αποτελεσμάτων
set(findobj('style','text'), 'String', {''});
set(handles.source_text, 'String', {'Αφετηρία: '});
set(handles.destination_text, 'String', {'Προορισμός: '});
set(handles.dijkstra_text, 'String', {'Διαδρομή  Dijkstra: '});
set(handles.hopfield_text, 'String', {'Διαδρομή Hopfield: '});
set(handles.time_text, 'String', {'Χρόνος εκτέλεσης: '});
set(handles.iterations_text, 'String', {'Επαναλήψεις: '});
set(handles.energy_text, 'String', {'Ενέργεια: '});

hopfield_run; % Καλεί το hopfield_run
```

```
% --- Εκτελείται με το πάτημα του κουμπιού "Έξοδος"
function exit_Callback(hObject, eventdata, handles)
% hObject    handle to exit (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

close(gcf) % Κλείσιμο GUI
```

9.3 Διαδικασία επέκτασης εφαρμογής

Όπως αναφέρθηκε στο Κεφάλαιο 6, η συγκεκριμένη εφαρμογή, με τις κατάλληλες τροποποιήσεις, μπορεί να χρησιμοποιηθεί για τον υπολογισμό των διαδρομών σε οποιαδήποτε πόλη και σε οποιοδήποτε οδικό δίκτυο. Συγκεκριμένα, για να συμβεί αυτό πρέπει να πραγματοποιηθούν τα ακόλουθα:

- Επιλογή των κυριότερων οδικών κόμβων
- Ενημέρωση του πίνακα ***W*** με το κόστος των συνδέσμων του νέου δικτύου.
- Ενημέρωση των πινάκων ***WX*** και ***WY*** με τις συντεταγμένες των κόμβων.
- Ενημέρωση του πίνακα ***g***, που καθορίζει την ύπαρξη ή όχι συνδέσμου μεταξύ δύο κόμβων.
- Ενημέρωση του πίνακα ***D***, με τις αποστάσεις μεταξύ των κόμβων.
- Υπολογισμός και αποθήκευση των βέλτιστων διαδρομών και του κόστους τους με βάση τον αλγόριθμο Dijkstra.
- Αλλαγή της εικόνας του χάρτη.

Για λόγους ευκολίας και φορητότητας επιλέχθηκε, κατά την υλοποίηση της εφαρμογής, όλες οι παραπάνω αλλαγές να αφορούν αρχεία εκτός του κώδικα ούτως ώστε η τροποποίησή τους να είναι απλή.

Βιβλιογραφία

[1] Wagner D., Willhalm T., Speed-Up Techniques for Shortest-Path Computations, Springer-Verlag, Lecture Notes in Computer Science, vol. 4393, pp. 23-36, 2007.

[2] Dijkstra, E.W., A note on two problems in connection with graphs, Numerische Mathematik 1, pp. 269–271, 1959.

[3] Yilin Zhao, Vehicle Location and Navigation Systems: Intelligent Transportation Systems, Artech House, 1997.

[4] Gao Yang, An Improved Shortest Route Algorithm in Vehicle Navigation System, 3rd International Conference on Advanced Computer Theory and Engineering (ICACTE), vol. 2, pp. 363-366, 2010.

[5] Hart, E.P., Nilsson, J.N., Bertram R., A formal basis for the heuristic determination of minimum cost paths, IEEE Transactions on System Science and Cybernetics, vol. 4, pp. 100–107, 1968.

[6] Pohl Ira, Bi-directional Search, Machine Intelligence, Edinburgh University Press, vol. 6, pp. 127–140, 1971.

[7] Kleinrock L., Kamoun F., Hierarchical Routing for Large Networks, North-Holland Publishing Company, Computer Networks 1, pp. 155-174, 1977.

[8] Τζαφέστας, Γ. Σ., Υπολογιστική νοημοσύνη, Τόμος Α: Μεθοδολογίες, Αθήνα, 1992.

[9] Igor Aleksander, An Introduction to Neural Computing. Chapman & Hall, 1993.

[10] Roland Schwaiger, Applications of Hopfield Networks, PS Introduction to Bioinformatics, Salzburg, 1999.

[11] Pitts, W., McCulloch, W. S., A logical calculus of the ideas immanent in nervous activity. Bull. Math. Biophys., vol. 5, 115–133, 1943.

[12] Hebb, D.O., The organization of behavior, Wiley, New York, 1949.

[13] Bishop, C.M., Neural networks and their applications, American Institute of Physics, Rev. Sci. Instrum., vol. 65, no. 6, pp. 1803-1832, 1994.

[14] Minsky, L.M., Papert, A.S., Perceptrons, MIT Press, 1959.

[15] Βλαχάβας Ι., Κεφάλας Π., Βασιλειάδης Ν., Ρεφανίδης Ν., Κόκκορας Φ., Σακελλαρίου Η., Τεχνητή Νοημοσύνη, Εκδόσεις Γαργατάνη, Θεσσαλονίκη, 2002.

[16] Hopfield, J.J., Neural networks and physical systems with emergent collective computational abilities, Proc. Natl. Acad. Sci., vol. 79, pp. 2554-2558, USA, 1982.

[17] Rojas, R., Neural Networks, Springer-Verlag, Berlin, 1996.

[18] Hopfield, J. J., Tank, D. W., Neural computation of decisions in optimization problems. Biological Cybernetics, vol. 52, pp. 141-152, 1985.

[19] Park, Y.K., Lee, G., Applications of neural networks in high speed communication networks, IEEE Communications Magazine, vol. 33, pp. 68–74, 1995.

[20] Pallapa Venkataram, Sudip Ghosal, B.P. Vijay Kumar, Neural network based optimal routing algorithm for communication networks, Neural Networks 15, Pergamon, pp. 1289-1298, 2002.

[21] Rauch, H.E., Winarske, T., Neural networks for routing communication traffic, IEEE control systems magazine, vol. 8, pp. 26–30, 1988.

[22] Zhang, L., Thomopoulos, S.C.A., Neural network implementation of the shortest path algorithm for traffic routing in communication networks, Proceedings of the international joint conference on neural networks, 1989.

[23] Ali, M.K., Kamoun, F., Neural networks for shortest path computation and routing in computer networks, IEEE transactions on neural networks, vol. 4, pp. 941–953, 1993.

[24] Park, D.C., Choi, S.E., A neural network based multi-destination routing algorithm for communication network, Proceedings of joint conference on neural networks, pp. 1673–1678, 1998.

[25] Ahn, C.W., Ramakrishna, R.S., Neural network based near-optimal routing algorithm, Proceedings of international conference on neural information processing, vol. 4, pp. 1771–1776, 2002.

[26] Park Dong-Chul, Keum Kyo-Reen, A shortest path routing algorithm using Hopfield neural network with an improved energy function, International Journal of General Systems, vol. 38, pp. 777-791, 2009.

[27] Joseph Kirk, Advanced Dijkstra's Minimum Path Algorithm, Mathworks, Matlab Central, File Exchange, 2009.
http://www.mathworks.com/matlabcentral/fileexchange/20025