ΑΝΩΤΑΤΟ ΕΚΠΑΙΔΕΥΤΙΚΟ ΙΔΡΥΜΑ ΠΕΙΡΑΙΑ

Α.Ε.Ι. ΠΕΙΡΑΙΑ Τ.Τ.

ΑΛΓΟΡΙΘΜΟΙ ΕΥΡΕΣΗΣ ΔΙΑΔΡΟΜΗΣ

ΓΚΑΒΟΓΙΑΝΝΗΣ ΛΑΜΠΡΟΣ

ΓΡΗΓΟΡΗΣ ΝΙΚΟΛΑΟΥ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΚΩΝ ΕΦΑΡΜΟΓΩΝ

ΤΜΗΜΑ ΑΥΤΟΜΑΤΙΣΜΟΥ

ΜΑΙΟΣ 2017

ΤΜΗΜΑ ΑΥΤΟΜΑΤΙΣΜΟΥ Π. Ράλλη & Θηβών 250,
12244 Αιγάλεω , Αθήνα – Ελλάδα Τηλ. 210-5381488

ΠΕΡΙΛΗΨΗ

Η πτυχιακή αυτή εργασία πραγματεύεται αλγορίθμους εύρεσης διαδρομής. Σκοπός της εργασίας αυτής είναι η μελέτη των αλγορίθμων αυτών. Η εφαρμογή των αλγορίθμων γίνεται σε διάφορες εφαρμογές όπως δίκτυα ή ρομποτικά συστήματα. Οι αλγόριθμοι αναζήτησης αφορούν το σχεδιασμό κατάλληλων ενεργειών με στόχο την άφιξη ενός ελέγξιμου συστήματος σε μία αποδεκτή τελική κατάσταση, εκκινώντας από κάποια προκαθορισμένη αρχική κατάσταση. Υπάρχει ποικιλία αλγορίθμων που χρησιμοποιούνται σήμερα για τέτοιες εφαρμογές αλλά στην συγκεκριμένη εργασία θα αναλυθούν μόνο τέσσερις. Οι αλγόριθμοι που επιλέχθηκαν να αναλυθούν είναι ο A* (Α-άστρο), Dijkstra, Particle Filters και οι Bug Algorithms, στους οποίους ανήκει ένα εύρος αλγορίθμων. Καθένας από αυτούς τους αλγορίθμους έχει πλεονεκτήματα και μειονεκτήματα και όλα αυτά παρουσιάζονται στην εργασία αυτή. Στην αρχή της παρούσας εργασίας παρουσιάζεται η έννοια και η χρήση του αλγορίθμου. Αναλύεται επίσης ο ορισμός των γράφων καθώς είναι απαραίτητοι για την εκτέλεση του Dijkstra αλγορίθμου. Στο υπόλοιπο τμήμα της εργασίας παρουσιάζονται οι επιλεγμένοι αλγόριθμοι και αφιερώνεται ένα κεφάλαιο για τον καθένα. Σε κάθε κεφάλαιο από αυτά υπάρχει υλοποίηση και επεξήγηση των αλγορίθμων μέσω απλών παραδειγμάτων μέσα από εικόνες και πίνακες. Σε κάποια σημεία των κεφαλαίων που περιγράφουν τους αλγορίθμους υπάρχει σύγκριση μεταξύ ορισμένων και εξηγείται για πιο λόγο προτιμάται ο καθένας.

ABSTRACT

In the present thesis we discuss about path finding algorithms. The main focus here is to study those algorithms. Their implementation can be applied in a variety of applications such as networks or robotic systems. They are used to plan actions that will result in having a controllable system in the desired form whilst started in a predefined form. There is a huge variety of those algorithms, however we will be analyzing four of them. These four are A* (A star), Dijkstra, Particle Filters and Bug Algorithms. Bug Algorithms contain different types of algorithms. Every single one of those four has advantages and disadvantages that we are going to talk about. In the beginning of this thesis we are going to talk about the definition and the use of them. Furthermore, we will be defining the graphs as they are an important part of Dijkstra. Throughout the rest of the thesis a unique chapter will be used to analyze each one of those four. The implementation and definition are going to be explained by using examples with images and drawings. In some cases there are going to be comparisons between them.

ΔΗΛΩΣΗ ΣΥΓΓΡΑΦΕΑ ΠΤΥΧΙΑΚΗΣ ΕΡΓΑΣΙΑΣ

Ο κάτωθι υπογεγραμμένος Λάμπρος Γκαβογιάννης του Βαΐου, με αριθμό μητρώου 39777 φοιτητής του Τμήματος Μηχανικών Αυτοματισμού Τ.Ε του Α.Ε.Ι Πειραιά Τ.Τ πριν αναλάβω την εκπόνηση της Πτυχιακής Εργασίας μου, δηλώνω ότι ενημερώθηκα για τα παρακάτω:

<<Η Πτυχιακή Εργασία (Π.Ε.) αποτελεί προϊόν πνευματικής ιδιοκτησίας τόσο του συγγραφέα, όσο και του ιδρύματος και θα πρέπει να έχει μοναδικό χαρακτήρα και πρωτότυπο περιεχόμενο.

Απαγορεύεται αυστηρά οποιοδήποτε κομμάτι κειμένου της να εμφανίζεται αυτούσιο ή μεταφρασμένο από κάποια άλλη δημοσιευμένη πηγή. Κάθε τέτοια πράξη αποτελεί προϊόν λογοκλοπής και εγείρει θέμα Ηθικής Τάξης για τα πνευματικά δικαιώματα του άλλου συγγραφέα. Αποκλειστικός υπεύθυνος είναι ο συγγραφέας της Π.Ε., ο οποίος φέρει και την ευθύνη των συνεπειών, ποινικών και άλλων, αυτής της πράξης.

Πέραν των όποιων ποινικών ευθυνών του συγγραφέα σε περίπτωση που το Ίδρυμα του έχει απονείμει Πτυχίο, αυτό ανακαλείται με απόφαση της Συνέλευσης του Τμήματος. Η Συνέλευση του Τμήματος με νέα απόφασης της, μετά από αίτηση του ενδιαφερόμενου, του αναθέτει εκ νέου την εκπόνηση της Π.Ε. με άλλο θέμα και διαφορετικό επιβλέποντα καθηγητή. Η εκπόνηση της εν λόγω Π.Ε. πρέπει να ολοκληρωθεί εντός τουλάχιστον ενός ημερολογιακού δμήνου από την ημερομηνία ανάθεσης της. Κατά τα λοιπά εφαρμόζονται τα προβλεπόμενα στο άρθρο 18, παρ. 5 του ισχύοντος Εσωτερικού Κανονισμού.>>

Επίσης δηλώνω υπεύθυνα ότι έχω παρακολουθήσει το σεμινάριο συγγραφής και εκπόνησης πτυχιακής εργασίας που διοργανώνεται από το Τμήμα Μηχανικών Αυτοματισμού Τ.Ε. κατά το Χειμερινό/Εαρινό Εξάμηνο του Ακαδημαϊκού Έτους 2015-2016.

Ο Δηλών

Ημερομηνία
16/05/2017

Περιεχόμενα

Κατάλογος Εικόνων

Κατάλογος Πινάκων

Αυτή η σελίδα είναι σκόπιμα κενή

ΕΥΧΑΡΙΣΤΙΕΣ

Θα ήθελα να ευχαριστήσω θερμά τον καθηγητή μου Γρηγόρη Νικολάου για τη δυνατότητα που μου έδωσε να εκπονήσω την παρούσα πτυχιακή εργασία σε έναν ενδιαφέροντα τομέα των αλγορίθμων, καθώς επίσης και για την πολύτιμη βοήθεια και συμπαράστασή του σε όλες τις φάσεις της εκπόνησης της πτυχιακής εργασίας. Τέλος θα ήθελα να ευχαριστήσω την οικογένεια μου για την στήριξη τους σε όλη τη διάρκεια των σπουδών μου.

ΚΕΦΑΛΑΙΟ 1

ΕΙΣΑΓΩΓΗ

1.1 Αντικείμενο και στόχος της πτυχιακής

Η παρούσα πτυχιακή εργασία έχει ως αντικείμενο την επεξήγηση των αλγορίθμων αναζήτησης διαδρομής (pathfinding) και την ανάπτυξη και υλοποίηση κάποιων από αυτών μέσω απλών παραδειγμάτων. Στόχος της εργασίας είναι από τη μία η πλήρης περιγραφή των συγκεκριμένων αλγορίθμων, η επίλυση κάποιων αντιπροσωπευτικών προβλημάτων και από την άλλη η δημιουργία συμπερασμάτων σχετικά με την αποδοτικότητά τους και τον τρόπο εκτέλεσης και λειτουργία τους. Επίσης υπάρχουν και συγκρίσεις μεταξύ κάποιων από αυτούς.

1.2 Ορισμός και ιστορική διαδρομή του αλγορίθμου

Ως αλγόριθμος ορίζεται μία πεπερασμένη σειρά ενεργειών, αυστηρά καθορισμένων και εκτελέσιμων σε πεπερασμένο χρόνο, που στοχεύουν στην επίλυση ενός προβλήματος. Πιο απλά αλγόριθμο ονομάζουμε μία σειρά από εντολές που έχουν αρχή και τέλος, είναι σαφείς και εκτελέσιμες και σκοπό έχουν την επίλυση κάποιου προβλήματος. Η λέξη αλγόριθμος προέρχεται από μία μελέτη του Πέρση μαθηματικού του 8ου αιώνα μ.Χ. Αλ Χουαρίζμι (Mohammed ibn Musa Al-Khwarizmi), η οποία περιείχε συστηματικές τυποποιημένες λύσεις αλγεβρικών προβλημάτων και αποτελεί ίσως την πρώτη πλήρη πραγματεία άλγεβρας. Πέντε αιώνες αργότερα η μελέτη μεταφράστηκε στα Λατινικά και άρχιζε με τη φράση "Algorithmus dixit ..." (ο Αλγόριθμος είπε ...). Έτσι η λέξη αλγόριθμος καθιερώθηκε αργά τα επόμενα χίλια χρόνια με την έννοια <<συστηματική διαδικασία αριθμητικών χειρισμών>>. Τη σημερινής της σημασία την οφείλει στη γρήγορη ανάπτυξη των ηλεκτρονικών υπολογιστών στα μέσα του 20ου αιώνα [1].

1.3 Σχεδιασμός αλγορίθμων

Ο σχεδιασμός και η ανάλυση αλγορίθμων αποτελεί τον πλέον θεμελιώδη τομέα της Επιστήμης των Υπολογιστών. Το πρώτο στάδιο για την επίλυση κάθε προβλήματος με τη χρήση υπολογιστή είναι η εύρεση μίας μεθόδου που να επιλύει το πρόβλημα. Η ιδέα του σχεδιασμού αλγορίθμων χρονολογείται από την εποχή των αρχαίων Ελλήνων και Κινέζων μαθηματικών, αλλά η συστηματική μελέτη και ανάλυση αλγορίθμων ως προς την αποδοτικότητά τους άρχισε σχετικά πρόσφατα (1970-1980). Αποδοτικότητα είναι η δυνατότητα επίλυσης ενός προβλήματος με τη χρήση των διαθέσιμων πόρων (π.χ. , χρόνος επεξεργασίας, χώρος αποθήκευσης στη μνήμη). Ο σχεδιασμός αποδοτικών αλγόριθμων δεν έχει μόνο θεωρητική, αλλά και πολύ μεγάλη πρακτική σημασία. Η συνεχώς εξελισσόμενη τεχνολογία των υπολογιστών δεν μπορεί πάντα να αντιμετωπίζει με αποδοτικό τρόπο έναν ιδιαίτερα μεγάλο ρυθμό αύξησης στην κατανάλωση πόρων, που απαιτούνται για κάποιον

υπολογισμό. Για παράδειγμα, στην περίπτωση ενός εκθετικού ρυθμού αύξησης, αν με τη σημερινή τεχνολογία μπορούμε να επιλύσουμε στιγμιότυπα ενός προβλήματος μεγέθους N σε χρόνο T (για οποιεσδήποτε τιμές των N και T) τότε, ακόμη και αν η μελλοντική τεχνολογία μας δώσει υπολογιστές 100 φορές πιο γρήγορους από τους σημερινούς, θα μπορούμε να επιλύουμε στον ίδιο χρόνο στιγμιότυπα του ίδιου προβλήματος μεγέθους το πολύ N+7, δηλαδή ένα εκατονταπλάσιο άλμα στην ταχύτητα των υπολογιστών μπορεί να επιλύσει στιγμιότυπα που είναι μόνο κατά 7 μονάδες μεγαλύτερα.

1.4 Αλγοριθμικές ιδέες

Οι αλγοριθμικές ιδέες είναι διεισδυτικές και η επίδρασή τους είναι εμφανής σε παραδείγματα τόσο από την επιστήμη των υπολογιστών, όσο και πέρα από αυτή. Κάποιες από τις σημαντικότερες αλλαγές στα πρότυπα δρομολόγησης στο Διαδίκτυο μπορούν να θεωρηθούν διαμάχες σχετικά με τις ανεπάρκειες ενός αλγορίθμου συντομότερης διαδρομής και τα σχετικά πλεονεκτήματα ενός άλλου. Οι βασικές έννοιες που χρησιμοποιούν οι βιολόγοι για να εκφράσουν τις ομοιότητες μεταξύ γονιδίων και γονιδιωμάτων έχουν αλγοριθμικούς ορισμούς. Οι προβληματισμοί που εκφράζονται από τους οικονομολόγους σχετικά με την υλοποιησιμότητα των συνδυαστικών πλειστηριασμών στην πράξη πηγάζουν κατά κύριο λόγο από το γεγονός ότι αυτοί οι πλειστηριασμοί περιλαμβάνουν ως ειδικές περιπτώσεις υπολογιστικά δυσεπίλυτα προβλήματα αναζήτησης. Επιπρόσθετα, οι αλγοριθμικές έννοιες δεν περιορίζονται σε γνωστά και μακροχρόνια προβλήματα. Μπορούμε να δούμε τις αντανακλάσεις αυτών των ιδεών σε καθημερινή βάση, σε καινούρια ζητήματα τα οποία ανακύπτουν σε μεγάλο εύρος τομέων.

Το θέμα δεν είναι μόνο το ότι οι αλγόριθμοι έχουν πολλές εφαρμογές. Το βαθύτερο ζήτημα είναι ότι το αντικείμενο των αλγορίθμων είναι ένας ισχυρός φακός μέσα από τον οποίο μπορεί κανείς να δει γενικότερα το πεδίο της επιστήμης υπολογιστών. Τα αλγοριθμικά προβλήματα σχηματίζουν τον πυρήνα της επιστήμης των υπολογιστών, όμως πολύ σπάνια εμφανίζονται με τη μορφή των ξεκάθαρα διατυπωμένων και με μαθηματική ακρίβεια ερωτημάτων. Αντίθετα, τείνουν να ανακύπτουν μαζί με διάφορες λεπτομέρειες που αφορούν την κάθε συγκεκριμένη εφαρμογή, κάποιες από τις οποίες είναι απαραίτητες ενώ άλλες είναι περιττές. Κατά συνέπεια ο αλγοριθμικός τομέας αποτελείται από δύο θεμελιώδεις συνιστώσες, την εργασία διατύπωσης του καθαρού μαθηματικού πυρήνα του προβλήματος, και μετά την εργασία του προσδιορισμού της κατάλληλης τεχνικής σχεδιασμού αλγορίθμων με βάση τη δομή του προβλήματος. Αυτές οι δύο συνιστώσες αλληλεπιδρούν μεταξύ τους, όσο πιο πολύ εξοικειώνεται κανείς με την πλήρη εργαλειοθήκη τεχνικών σχεδιασμού, τόσο

πιο πολύ αρχίζει να αναγνωρίζει τις καθαρές διατυπώσεις που κρύβονται πίσω από πολύπλοκα προβλήματα του πραγματικού κόσμου. Έτσι, στην πιο αποδοτική μορφή τους, οι αλγοριθμικές ιδέες δεν παρέχουν απλώς λύσεις σε ξεκάθαρα διατυπωμένα προβλήματα, αλλά σχηματίζουν τη γλώσσα που μας επιτρέπει να εκφράσουμε τις υποκείμενες ερωτήσεις.

1.5 Τυποποιημένοι αλγόριθμοι

Οι αλγόριθμοι είναι σημαντικοί γιατί σχετίζονται άμεσα με τον τρόπο με τον οποίο οι υπολογιστές επεξεργάζονται δεδομένα και παράγουν πληροφορίες. Ένα πρόγραμμα υπολογιστών είναι ουσιαστικά ένας αλγόριθμος που λέει στον υπολογιστή ποια συγκεκριμένα βήματα να εκτελέσει (σε ποια συγκεκριμένη σειρά) προκειμένου να επιτευχθεί ένας συγκεκριμένος στόχος, όπως π.χ. ο υπολογισμός των μισθών των υπαλλήλων ή η εκτύπωση των έλεγχων των μαθητών. Κατά συνέπεια, ένας αλγόριθμος μπορεί να θεωρηθεί οποιαδήποτε ακολουθία εντολών που μπορεί να εκτελεσθεί από μια υπολογιστική μηχανή (turing).

Χαρακτηριστικά, όταν ένας αλγόριθμος συνδέεται με την επεξεργασία πληροφοριών, τα δεδομένα διαβάζονται από μια συσκευή εισόδου, γράφονται σε μια συσκευή εξόδου, και/ή αποθηκεύονται για την περαιτέρω χρήση. Τα αποθηκευμένα στοιχεία θεωρούνται ως τμήμα της εσωτερικής κατάστασης του συστήματος που εκτελεί τον αλγόριθμο.

Για οποιαδήποτε τέτοια υπολογιστική διαδικασία, ο αλγόριθμος πρέπει να είναι ορισμένος για όλες τις πιθανές περιστάσεις που θα μπορούσαν να προκύψουν. Δηλαδή οποιαδήποτε υπό όρους βήματα πρέπει να εξεταστούν συστηματικά, και σε κάθε περίπτωση τα κριτήρια πρέπει να είναι σαφή και υπολογίσιμα.

Επειδή ένας αλγόριθμος είναι ένας ακριβής κατάλογος βημάτων ακριβείας, η σειρά του υπολογισμού θα είναι σχεδόν πάντα κρίσιμη για τη λειτουργία του αλγορίθμου. Οι εντολές συνήθως απαριθμούνται ρητά, και περιγράφονται σαν να ξεκινούν "από την κορυφή" και πηγαίνοντας "προς στο κατώτατο σημείο", μία ιδέα που περιγράφεται τυπικά με τον όρο της "ροής ελέγχου".

Μέχρι τώρα, σε αυτήν η συζήτηση για την τυποποίηση του αλγορίθμου, έχουμε δεχθεί ως βάση τον διαδικαστικό προγραμματισμό. Αυτή είναι και η πιο κοινή αντίληψη, η οποία προσπαθεί να περιγράψει ένα έργο με διακεκριμένα, "μηχανικά" μέσα. Μοναδικός σε αυτήν την αντίληψη των αλγόριθμων είναι ο ρόλος της λειτουργίας ανάθεσης (ο καθορισμός της τιμής μιας μεταβλητής), ο οποίος προέρχεται από τη ιδέα "της μνήμης" σαν πρόχειρο τετράδιο.

1.6 Εφαρμογή αλγορίθμων

Οι αλγόριθμοι μπορούν να υλοποιηθούν από προγράμματα ηλεκτρονικών υπολογιστών, μολονότι συχνά σε περιορισμένες μορφές. Ένα λάθος στον σχεδιασμό ενός αλγόριθμου για την λύση ενός προβλήματος μπορεί να οδηγήσει σε αποτυχίες/βλάβες στο εφαρμοσμένο πρόγραμμα.

Οι αλγόριθμοι δεν υλοποιούνται μόνο ως προγράμματα υπολογιστών, αλλά συχνά επίσης και με άλλα μέσα, όπως π.χ. σε ένα βιολογικό νευρικό δίκτυο, ή σε ένα ηλεκτρονικό κύκλωμα, ή σε μια μηχανική συσκευή.

Η ανάλυση και η μελέτη των αλγορίθμων είναι ένας τομέας της επιστήμης της πληροφορικής, και ασκείται συχνά αφαιρετικά (χωρίς τη χρήση μίας συγκεκριμένης γλώσσας προγραμματισμού ή άλλη εφαρμογή). Από αυτή την άποψη, μοιάζει με άλλους μαθηματικούς τομείς, συγκεκριμένα στο ότι η εστίαση της ανάλυσης είναι πάνω στις βασικές αρχές του αλγορίθμου, και όχι σε οποιαδήποτε ιδιαίτερη εφαρμογή του. Ένας τρόπος απεικόνισης ενός αλγορίθμου είναι το γράψιμο του ψευδοκώδικα. Άλλοι τρόποι είναι το ελεύθερο κείμενο, με φυσική γλώσσα περιγράφοντας τα βήματα και με λογικό διάγραμμα.

1.7 Αλγόριθμοι αναζήτησης διαδρομής

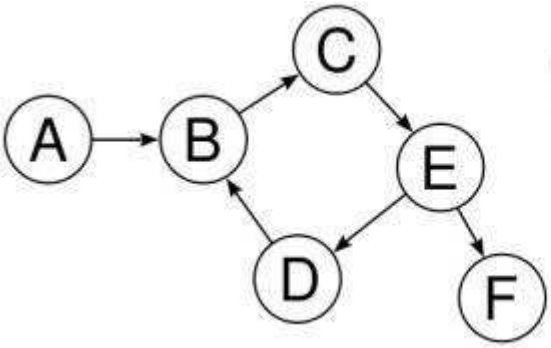
Οι αλγόριθμοι λαμβάνουν ως είσοδο το δοθέν πρόβλημα και επιστρέφουν ως έξοδο μία λύση σε αυτό, αφού αξιολογήσουν πρώτα μία ομάδα υποψηφίων λύσεων. Πρόκειται για ένα θεμελιώδες γνωστικό πεδίο της Τεχνητής Νοημοσύνης το οποίο γνώρισε μεγάλη ανάπτυξη από τη δεκαετία του 1950. Αποτέλεσε μία προσπάθεια αλγοριθμικής εξομοίωσης της διαδικασίας της σκέψης, ενώ με τον καιρό ενσωμάτωσε μεθοδολογίες από τη θεωρία βελτιστοποίησης και τη θεωρία γράφων.

ΚΕΦΑΛΑΙΟ 2

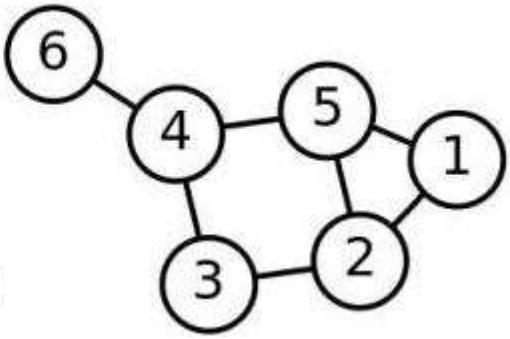
ΓΡΑΦΟΙ

2.1 Ορισμός του Γράφου

Οι παρακάτω εικόνες ορίζουν τους γράφους, οι οποίοι απαρτίζονται από οντότητες συνδεδεμένες μεταξύ τους:



Εικόνα 1: Γράφος ή Δίκτυο



Εικόνα 2: Γράφος ή Δίκτυο

Γράφος ή Δίκτυο ονομάζεται η παραπάνω σχηματική διάταξη. Ένα βασικό αντικείμενο της μελέτης των διακριτών μαθηματικών είναι οι γράφοι. Μαθηματικά εργαλεία χρησιμοποιούνται από την επιστήμη της θεωρίας Γράφων για την ανάλυση αυτών.

Μία οπτική αναπαράσταση των σχέσεων που υπάρχουν μεταξύ ορισμένων οντοτήτων είναι ένας τυπικός ορισμός γράφου. Πρόκειται για σημεία (οντότητες) και καμπύλες (σχέσεις), όλα αυτά σε σχέση με ένα σύστημα. Σύμφωνα με έναν άλλον ορισμό, ο οποίος όμως κινείται στο ίδιο εννοιολογικό πλαίσιο της οπτικής αναπαράστασης, ο γράφος αναγνωρίζεται ως μία απεικόνιση αποτελούμενη από ένα σύνολο σημείων (κορυφών ή κόμβων) που συνδέονται με γραμμές (ακμές).

Για να θεωρηθεί ένας γράφος χρειάζεται ένα ζεύγος G=(V, E), το οποίο αποτελείται από ένα σύνολο V(G) κόμβων και ένα σύνολο E(G) ακμών, όπου το E(G) είναι ένα σύνολο ζευγών με στοιχεία από το σύνολο V (μία ακμή μπορεί να θεωρηθεί σαν ένα ζεύγος κόμβων).

Ως άκρα ακμής ορίζονται οι κόμβοι που ανήκουν σε μία ακμή. Υπάρχει περίπτωση να υπάρχει ένας κόμβος στο γράφο αλλά να μην ανήκει σε μία ακμή. Το |V(G)| (αριθμός των κόμβων) ορίζει την τάξη του γράφου και το |E(G)| (αριθμός των ακμών) το μέγεθός του. Ο αριθμός των ακμών που ανήκει ο κόμβος αυτός ορίζει την τάξη ενός κόμβου και επίσης μετράται δύο φορές μία ακμή που έχει ως άκρα της τον ίδιο κόμβο (βρόχος). Μπορούμε να γράψουμε για συντομία μία ακμή ως uv, εάν γνωρίζουμε ότι έχει άκρα τους κόμβους u, v. Τα σύνολα V(G) και E(G) μπορούν να ορίσουν ένα γράφο G με την χρήση μίας συνάρτησης fG, στην οποία ένα ζευγάρι κόμβων του G, σχετίζεται με μία ακμή του συνόλου E(G). Εάν έχουμε μία ακμή e και δύο κόμβους v1, v2 και ισχύει ότι fG(e)=v1v2 τότε μπορούμε να πούμε ότι η e ενώνει τα v1, v2, που ορίζονται ως άκρα της e. Γνωρίζουμε δύο κατηγορίες

γράφων, τον κατευθυνόμενο ή προσανατολισμένο και τον μη κατευθυνόμενο γράφο. Στον πρώτο (Εικόνα 1) οι ακμές του έχουν κατεύθυνση επειδή είναι διανύσματα. Άρα σε μία ακμή με άκρα δύο κόμβους v1, v2, ο ένας από τους δύο θα είναι η αρχή του διανύσματος και ο άλλος το τέλος. Άρα για κάθε τέτοιου τύπου γράφου θα ισχύει
(fG(e1)=v1v2)≠(fG(e2)=v2v1). Όσο για τους μη κατευθυνόμενους γράφους (Εικόνα 2) οι ακμές δεν έχουν κατεύθυνση, οπότε θα ισχύει (fG)(e)=v1v2=v2v1). Για να θεωρείται ένας γράφος Πλήρης θα πρέπει να έχει ακμές για κάθε πιθανό ζεύγος μέσα από το σύνολο των κόμβων του. Πυκνός κόμβος ορίζεται αυτός που έχει πολλές ακμές ενώ αραιός το αντίθετο. Στην περίπτωση που υπάρχουν δύο ακμές με έναν κοινό κόμβο τότε λέμε ότι οι μη κοινοί κόμβοι συνδέονται μέσω του κοινού κόμβου. Έστω ότι έχουμε μία ακμή αβ και μία βγ. Κοινός κόμβος είναι ο β, άρα ο α συνδέεται με τον γ μέσω του κόμβου β. Συνεχίζοντας τον παραπάνω ορισμό καταλήγουμε στο συμπέρασμα ότι για να υπάρξει διαδρομή ανάμεσα σε δύο κόμβους του γράφου πρέπει αυτοί οι δύο κόμβοι να συνδέονται μέσω άλλων κόμβων.

Συνεκτικός ή μη συνδεδεμένος γράφος ορίζεται ως ο γράφος, στον οποίο δεν υπάρχει μονοπάτι για τουλάχιστον έναν κόμβο προς έναν άλλον. Τα παρακάτω ισχύουν για ένα προσανατολισμένο γράφο:

- Στην περίπτωση που δεν υπάρχει διαδρομή μεταξύ δύο κόμβων είτε από τον έναν προς τον άλλον είτε αντίστροφα, τότε ο γράφος ονομάζεται συνεκτικός ή μη συνδεδεμένος.
- Στην περίπτωση που υπάρχει μονοπάτι από έναν κόμβο προς κάθε άλλο κόμβο, τότε ο γράφος ονομάζεται ισχυρά συνεκτικός, ή ισχυρά συνδεδεμένος.
- Στην περίπτωση που ισχύει η προηγούμενη περίπτωση αλλά για τουλάχιστον ένα ζεύγος κόμβων δεν υπάρχει διαδρομή, τότε ο γράφος ονομάζεται ασθενώς συνεκτικός, ή ασθενώς συνδεδεμένος.

Όταν θέλουμε να χρησιμοποιήσουμε τον γράφο για να επεξεργαστούμε κάποια πραγματική οντότητα, τότε οι ακμές του φέρουν τιμές, οι οποίες υποδηλώνουν κάποιο φυσικό ή τεχνικό μέγεθος. Ένα τέτοιο παράδειγμα είναι ένα δίκτυο υπολογιστών, όπου οι τιμές εκφράζουν ταχύτητες ζεύξεων. (Τσατσάνης, 2014)

2.2 Εφαρμογές Γράφων

Μοντελοποίηση πολλών ειδών σχέσεων καθιστούν αναγκαία την χρήση γράφων σε συστήματα βιολογικά [2] κοινωνικά και πληροφοριακά. Οι γράφοι μπορούν να αναπαραστήσουν πολλά πρακτικά προβλήματα. Τέτοια προβλήματα μπορούν να ανήκουν

στην επιστήμη των υπολογιστών για την αναπαράσταση δομών δεδομένων, υπολογιστικών μηχανημάτων κ.λπ. Ένα τέτοιο παράδειγμα είναι η αναπαράσταση μίας ιστοσελίδας ως ένας κατευθυνόμενος γράφος. Στον γράφο αυτό οι κόμβοι θεωρούνται οι σελίδες και οι ακμές τα links από την μία σελίδα σε μία άλλη. Επιπλέον προβλήματα που μπορούν να προσεγγιστούν παρόμοια ανήκουν στον τομέα της βιολογίας, στον σχεδιασμό ηλεκτρονικών κυκλωμάτων κα. Μεγάλο ενδιαφέρον παρουσιάζουν για την επιστήμη των υπολογιστών η ανάπτυξη γράφων που διαχειρίζονται τέτοιους γράφους.

Έχει αποδειχθεί ότι διάφοροι μέθοδοι θεωρίας γράφων είναι χρήσιμες στη γλωσσολογία και στις φυσικές γλώσσες εφόσον χρησιμοποιούνται διακριτές δομές. Στην επιστήμη της χημείας και της φυσικής η μελέτη των μορίων και τον ατόμων γίνεται με την βοήθεια της θεωρίας των γράφων. Οι γράφουν μπορούν και συλλέγουν ιδιότητες σχετικές με την τοπολογία των ατόμων μέσω της μελέτης, εφόσον είναι ικανοί να αναπαραστήσουν πολύπλοκες και ατομικές δομές.

Επίσης στην κοινωνιολογία οι γράφοι χρησιμοποιούνται για την παράσταση σχέσεων μεταξύ συνόλων, τα οποία σύνολα είναι στατιστικά δείγματα από διάφορες μελέτες [3]. Στον τομέα της βιολογίας η χρησιμότητα των γράφων είναι η αναπαράσταση περιοχών με συγκεκριμένα ήδη ζώων. Αυτό είναι δουλειά των κόμβων καθώς οι ακμές μπορούν να αναπαραστήσουν πιθανές κινήσεις μεταξύ αυτών των περιοχών.

Στον τομέα των μαθηματικών η γεωμετρία και η τοπολογία καθιστούν αναγκαία την χρήση γράφων και στην άλγεβρα, η θεωρία συνόλων. (Τσατσάνης, 2014)

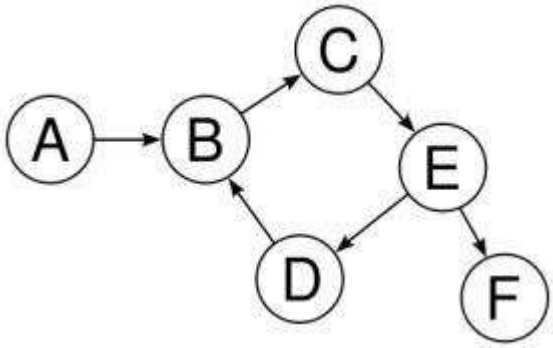
2.3 Δομές παράστασης γράφων σε υπολογιστικά συστήματα

Οι τρόποι για να αποθηκευτεί ένας γράφος σε ένα υπολογιστικό σύστημα ποικίλουν. Ο αλγόριθμος που θα χρησιμοποιηθεί για την επεξεργασία του γράφου, αλλά και ο ίδιος ο γράφος καθορίζουν την δομή δεδομένων. Οι δομές παράστασης χωρίζονται σε τρεις κατηγορίες, η λίστα, ο πίνακας ή ο συνδυασμός αυτών των δύο. Όταν απαιτείται γρήγορο προσπέλαση του γράφου, είναι αναγκαία η χρήση δομών πινάκων, αλλά καταναλώνουν πολλή μνήμη. Σε περιπτώσεις που ο γράφος είναι πιο αραιός, χρησιμοποιούνται δομές λιστών και απαιτούν και λιγότερη μνήμη.

Μία λίστα από ζεύγη κόμβων, δηλαδή ένας πίνακας συνδέσεων (connectivity table) και μία λίστα που περιλαμβάνει για κάθε κόμβο τους κόμβους που συνδέονται με αυτόν (γείτονες), δηλαδή μία λίστα γειτνίασης (adjacency list) συγκροτούν τις δομές λίστας.

Ο πίνακας γειτνίασης (adjacency matrix) αποτελεί τις δομές πινάκων. Αυτός ο πίνακας αποτελείται από γραμμές και στήλες, που είναι οι δείκτες του κάθε κόμβου. Ο πίνακας είναι

γεμάτος με στοιχεία 0 ή 1. Το 0 δηλώνει ότι δεν υπάρχει ακμή που να συνδέει τους κόμβους i, j, ενώ το 0 δηλώνει το αντίθετο. Δηλαδή αν υπάρχει ή όχι κόμβος i στην γραμμή i, κόμβος j στην στήλη j. Επίσης αντί για τον πίνακα αυτό μπορεί να χρησιμοποιηθεί ένας παρόμοιος πίνακας, ο οποίος ονομάζεται πίνακας κόστους (distance matrix). Σε αυτόν τον πίνακα τα στοιχεία i, j αντί για 0 ή 1 έχουν τα αντίστοιχα βάρη των ακμών. Στην παρακάτω εικόνα βλέπουμε ένα παράδειγμα προσανατολισμένου γράφου και των δομών αναπαράστασής του από ένα υπολογιστικό σύστημα.



Εικόνα 3: Προσανατολισμένος γράφος (Τσατσάνης, 2014)

Ο πίνακας γειτνίασης του παραπάνω γράφου θα είναι:

Πίνακας 1: Πίνακας γειτνίασης γράφου (Τσατσάνης, 2014)

	A	**B**	**C**	**D**	**E**	**F**
A	0	1	0	0	0	0
B	0	0	1	0	0	0
C	0	0	0	0	1	0
D	0	1	0	0	0	0
E	0	0	0	1	0	1
F	0	0	0	0	0	0

Η λίστα συνδέσεων είναι:

και η λίστα γειτνίασης είναι:

Για γράφους με βάρη στις ακμές τους ανάλογη πληροφορία μπορεί να προστίθεται στις παραπάνω δομές ώστε να είναι πλήρεις.(Τσατσάνης, 2014)

ΚΕΦΑΛΑΙΟ 3

DIJKSTRA

3.1 Γενικά

Ο Edsger W. Dijkstra δημιούργησε τον αλγόριθμο Dijkstra το 1956 και τον έκδωσε το 1959 [5]. Εφαρμόζεται για να λύνει προβλήματα συντομότερου μονοπατιού μίας πηγής για έναν γράφο με μη αρνητικά κόστη στις ακμές του, αφού είναι ένας αλγόριθμος αναζήτησης γράφου. Για να φτάσει στην επίλυση πρέπει να δημιουργήσει ένα δέντρο σύντομου μονοπατιού. Πολλές φορές άλλοι αλγόριθμοι γράφων χρησιμοποιούν τον Dijkstra ως υπορουτίνα για δρομολόγηση. Το χαρακτηριστικό του αλγορίθμου είναι, ότι μπορεί και βρίσκει το μονοπάτι με το ελάχιστο κόστος μεταξύ του κόμβου-πηγή και οποιουδήποτε άλλου κόμβου στο γράφο. Επίσης υπολογίζει την απόσταση από έναν απλό κόμβο προς έναν άλλο απλό κόμβο, μέχρι ο αλγόριθμος εντοπίσει τον κόμβο προορισμού. Σε μία εφαρμογή όπου οι κόμβοι του γράφου αναπαριστούν πόλεις και τα κόστη των ακμών αναπαριστούν αποστάσεις μεταξύ δύο πόλεων που συνδέονται μέσω δρόμου, ο αλγόριθμος Dijkstra χρησιμοποιείται για την εύρεση της συντομότερης διαδρομής μεταξύ μίας πόλης και όλων των υπολοίπων πόλεων. Πρωτόκολλα δρομολόγησης δικτύων όπως IS-IS (Intermediate System to Intermediate System) και OSPF (Open Shortest Path First) χρησιμοποιούν τον αλγόριθμο αυτό. [3]

3.2 Περιγραφή

Για να εφαρμοστεί ο αλγόριθμος Dijkstra πρέπει όλα τα βάρη των ακμών να είναι θετικά, άρα να ισχύει w(u,v) 0 για κάθε ακμή (u,v) E και έστω s η ακμή-αφετηρία. Δημιουργείται ένα σύνολο S από τις ακμές που ήδη έχουν συνδεθεί προς την ακμή s με το συντομότερο μονοπάτι. Η ακμή s ανήκει στο σύνολο S και για κάθε άλλη ακμή v S, ισχύει d[v]=δ(s,v). Η διαδικασία αυτή επαναλαμβάνεται και ο αλγόριθμος τοποθετεί την ακμή u στο S όσο ισχύει u V-S και έπειτα εφαρμόζεται η διαδικασία σάρωσης. Κάποιες φορές τυχαίνει να υπάρχει περισσότερες από μία ακμές που δεν ανήκουν στο S, οπότε η επιλογή ανάμεσά τους είναι τυχαία αρκεί να έχουν ελάχιστη ετικέτα βάρους.

3.3 Σκοπός και λειτουργία του αλγορίθμου

Ο αλγόριθμος αυτός έχει εφαρμογή σε διάφορους τομείς. Έστω ότι έχουμε ένα δίκτυο όπου ο Dijkstra βρίσκει εφαρμογή. Σκοπός του είναι η εύρεση της διαδρομής με το μικρότερο μήκος, δηλαδή τον ταχύτερο δρόμο για μία κλήση σε ένα δίκτυο.

Χρησιμοποιούνται δύο πίνακες για την λειτουργία αυτή. Ο ένας έχει την ονομασία προσωρινή ετικέτα (temporary label) d, ενώ ο δεύτερος την ονομασία μόνιμη ετικέτα (permanent label) r. Ορίζεται ένας πίνακας με την ονομασία πίνακας κόστους, ο οποίος

περιέχει δεδομένα. Τα δεδομένα αυτά αναφέρονται σε στοιχεία που εκφράζουν όρια ή ικανότητες συστήματος όπως είναι η απόσταση, το κόστος μεταφοράς, η χωρητικότητα κα. Πριν ξεκινήσει οποιαδήποτε λειτουργία πρέπει να οριστούν δύο κόμβοι. Ο κόμβος έναρξης και ο κόμβος τερματισμού και παίρνουμε ως δεδομένο ότι δεν υπάρχει σύνδεση μεταξύ δύο διαφορετικών κόμβων, οπότε και η αντίστοιχη τιμή τους στον πίνακα κόστους θα είναι άπειρη και θα παριστάνεται από μία μεγάλη τιμή (π.χ. 1000).

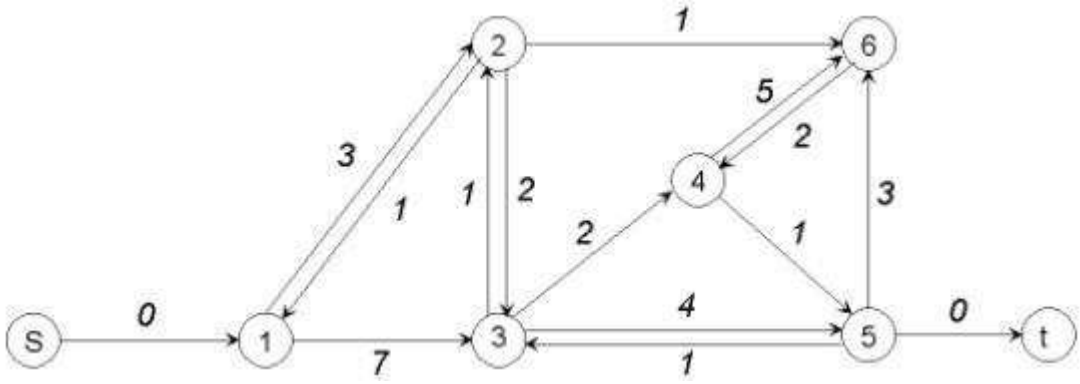
Αφού τοποθετήσουμε άπειρες τιμές στον μόνιμο πίνακα, εκτελούμε μία συγκεκριμένη διαδικασία έως ότου το στοιχείο που αντιστοιχεί στον τερματισμό της διαδρομής να έχει διάφορη τιμή του απείρου. Ο αλγόριθμος θέτει μία ετικέτα μόνιμη ή προσωρινή σε όλους τους κόμβους. Όλοι όμως οι κόμβοι εκτός του αρχικού δέχονται μία προσωρινή ετικέτα, η οποία παριστάνει την απόσταση μεταξύ του κόμβου έναρξης και του συγκεκριμένου κόμβου. Ο αρχικός κόμβος παίρνει μόνιμη τιμή το μηδέν. Οι κόμβοι που δεν συνδέονται απευθείας με τον αρχικό κόμβο αποκτούν προσωρινή ετικέτα με άπειρη τιμή, ενώ όσοι συνδέονται αποκτούν προσωρινή ετικέτα c_{sj},j s. Για να αποκτήσει ένας κόμβος μόνιμη ετικέτα θα πρέπει να ανήκει στη μικρότερη διαδρομή. Μετατρέποντας τη μικρότερη από τις προσωρινές ετικέτες σε μόνιμη, μπορούμε και βρίσκουμε τον κόμβο που είναι πιο κοντά στον αρχικό κόμβο. Ακολουθεί μία διαδικασία επανάληψης δύο σταδίων μέχρι η ετικέτα του τελικού κόμβου γίνει μόνιμη.

1. Αναζητώ τους κόμβους που απέμειναν με προσωρινή ετικέτα.. Συγκρίνω κάθε προσωρινή ετικέτα με το άθροισμα της τελευταίας μόνιμης ετικέτας και της απευθείας απόστασης του κόμβου, με την προσωρινή ετικέτα, από τον υπό εξέταση κόμβο. Η ελάχιστη από τις αποστάσεις γίνεται μόνιμη για τον συγκεκριμένο κόμβο.

2. Διαλέγω την μικρότερη εναπομείνουσα ετικέτα και την μετατρέπω σε μόνιμη. Αν η ετικέτα του κόμβου τερματισμού γίνει μόνιμη ο αλγόριθμος λαμβάνει τέλος διαφορετικά επιστρέφω στο βήμα 1.

3.3.1 Παράδειγμα 1

Ο σκοπός αυτού του παραδείγματος είναι η εφαρμογή του αλγορίθμου του Dijkstra στο δίκτυο που ακολουθεί. Το s παριστάνει τον κόμβο έναρξης της δρομολόγησης του πακέτου, το t το τέλος, και το c_{ij} είναι η χωρητικότητα μεταξύ των i,j συνδέσμων του δικτύου. Ο αλγόριθμος ξεκινάει θέτοντας τη μόνιμη ετικέτα $r_s=0$ του κόμβου s, και τις προσωρινές ετικέτες $d_j=c_{sj}$, j=1, 2, ..., 6, t. Αυτό σημαίνει ότι $d_1=0$ και $d_1=0$ και $d_j=$, j=2, ..., t. Επειδή το $d_1=0$ είναι η μικρότερη ετικέτα από τις προσωρινές, ο κόμβος 1 λαμβάνει μόνιμη ετικέτα με τιμή 0.



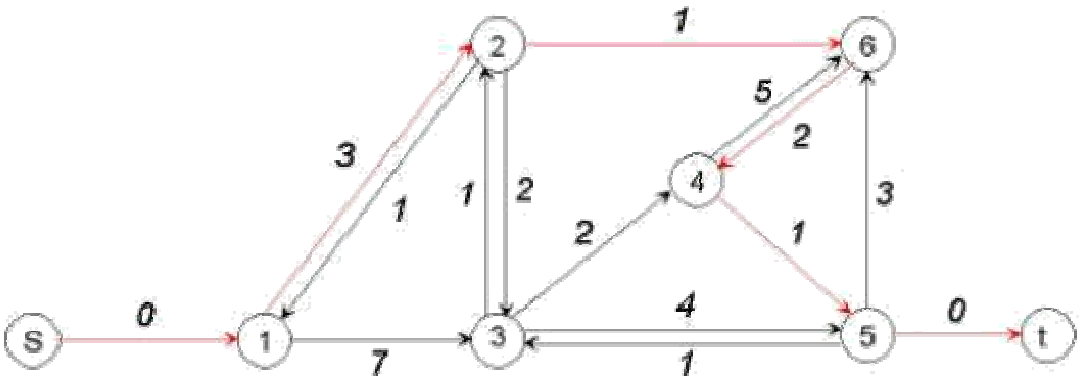
Εικόνα 4: Εφαρμογή του αλγορίθμου Dijkstra στο συγκεκριμένο δίκτυο

Οι κόμβοι 2 και 3 συνδέονται απευθείας με τον 1, που είναι ο τελευταίος κόμβος που απέκτησε μόνιμη ετικέτα. Σημειώνεται ότι $d_1+c_{12}=0+3<$ και $d_1+c_{13}=0+7<$, επομένως οι προσωρινές ετικέτες των κόμβων 2 και 3 γίνονται $d_2=3$ και $d_3=7$ αντίστοιχα. Αφού $d_2<d_3$ η ετικέτα του κόμβου 2 γίνεται μόνιμη, $r_2=3$. Οι κόμβοι 3 και 6 συνδέονται απευθείας με τον 2. Ακόμα $d_2+c_{23}=3+2=5<7$ και $d_2+c_{26}=3+1=4<$. Συνεπώς έχουμε $d_3=5$ και $d_6=4$. Η ετικέτα του κόμβου 6 γίνεται μόνιμη επειδή $d_6>d_3$, άρα $r_6=4$.Ο τελευταίος κόμβος στον οποίο δόθηκε μόνιμη ετικέτα είναι ο 6ος ο οποίος συνδέεται απευθείας μόνο με τον 4. Ισχύει ότι $d_6+c_{64}=4+2=6<$, άρα $d_4=6$. Τώρα επειδή οι μόνιμες ετικέτες που απέμειναν είναι οι $d_3=5$, $d_4=6$, $d_5=dt=$ προκύπτει ότι $d_3=\min\{d_3,d_4,d_5,d_6\}$, έτσι ο κόμβος 3 μονιμοποιείται λαμβάνοντας την τιμή $r_3=5$. Ακολουθώντας αυτή τη διαδικασία το πρόβλημα τελειώνει όταν ο κόμβος t αποκτήσει μια μόνιμη ετικέτα.

Πίνακας 2: Υπολογισμοί που έγιναν για το παράδειγμα της μεθόδου Dijkstra. Με κόκκινο παριστάνονται οι μόνιμες ετικέτες και με μαύρο οι προσωρινές

Βήμα	Κόμβοι	s	1	2	3	4	5	6	T
0		0	∞	∞	∞	∞	∞	∞	∞
1		0	0	∞	∞	∞	∞	∞	∞
2		0	0	∞	∞	∞	∞	∞	∞
3		0	0	3	7	∞	∞	∞	∞
4		0	0	3	7	∞	∞	∞	∞
5		0	0	3	5	∞	∞	4	∞
6		0	0	3	5	∞	∞	4	∞
7		0	0	3	5	6	∞	4	∞
8		0	0	3	5	6	∞	4	∞
9		0	0	3	5	6	9	4	∞
10		0	0	3	5	6	9	4	∞
11		0	0	3	5	6	7	4	∞
12		0	0	3	5	6	7	4	∞
13		0	0	3	5	6	7	4	7
14		0	0	3	5	6	7	4	7

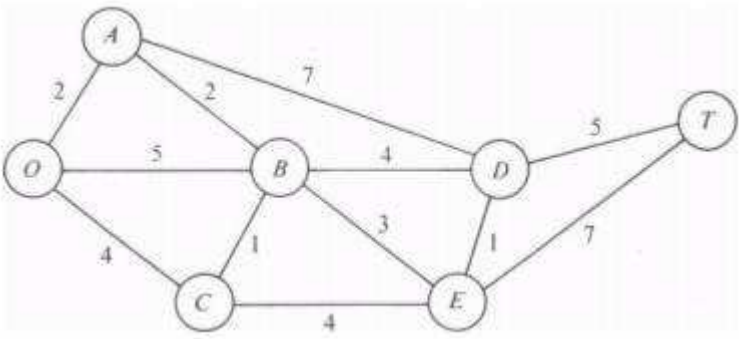
Για να βρούμε το δρόμο που τελικά ακολούθησε το πακέτο εργαζόμαστε ως εξής. Επειδή $r_t=7$ αυτό σημαίνει ότι το πιο σύντομο μονοπάτι απέχει μόλις 7. Για να είναι οι κόμβοι i,j μέσα στο μονοπάτι πρέπει να ισχύει ότι $r_j=r_i+c_{ij}$. Ελέγχουμε αν ισχύει αυτή η σχέση ξεκινώντας από το τέλος, δηλαδή από τον κόμβο t. Μόλις βρεθεί ο κόμβος μέσω του οποίου πάμε στον t επαναλαμβάνουμε τη διαδικασία μέχρι να φτάσουμε στον κόμβο s. Για το συγκεκριμένο παράδειγμα βρίσκουμε ότι η πιο σύντομη διαδρομή είναι s-1-2-6-4-5-t.



Εικόνα 5: Διαδρομή που ακολουθεί το πακέτο μέσω του Dijkstra

3.3.2 Παράδειγμα 2

Η διοίκηση του πάρκου το οποίο απεικονίζεται στο παρακάτω δίκτυο θέλει να βρει τη συντομότερη διαδρομή από την είσοδο του πάρκου (κόμβος O) μέχρι την κορυφή T διαμέσου του οδικού συστήματος. Οι αριθμοί σε κάθε κλάδο δίνουν το μήκος του κλάδου αυτού.



Εικόνα 6: Εφαρμογή του αλγορίθμου Dijkstra στο συγκεκριμένο δίκτυο

Εφαρμόζοντας τον παραπάνω αλγόριθμο στο πρόβλημα παίρνουμε τα αποτελέσματα του επόμενου πίνακα.

Πίνακας 3: Πίνακας αποτελεσμάτων σύμφωνα με τον αλγόριθμο Dijkstra

n	Λυμένοι κόμβοι άμεσα συνδεδεμένοι με άλυτους κόμβους	Πλησιέστερος συνδεδεμένος άλυτος κόμβος	Συνολική Απόσταση	n-οστός πλησιέστερος κόμβος	Ελάχιστη Απόσταση	Τελευταία Σύνδεση
1	O	A		A	2	OA
2	O A	C B		C B	4 4	OC AB
4	A B C	D E E		E	7	BE
5	A B E	D D D		D D	8 8	BD ED
6	D E	T T		T	13	DT

Προσέχουμε ότι η ισοβάθμηση στη δεύτερη επανάληψη μας δίνει τη δυνατότητα να προσθέσουμε δύο νέους λυμένους κόμβους, οπότε στην επόμενη επανάληψη υπολογίζουμε τον τέταρτο πλησιέστερο κόμβο. Η συντομότερη διαδρομή από τον προορισμό προς την αρχή μπορεί να βρεθεί από την τελευταία στήλη του πίνακα και είναι ή η T→D→E→B→A→O ή η T→D→B→A→O. Έτσι οι δύο εναλλακτικές λύσεις της συντομότερης διαδρομής από την αρχή μέχρι τον προορισμό είναι η O→A→B→E→D→T και η O→A→B→D→T, με συνολική απόσταση 13 η καθεμιά.

Πριν τελειώσουμε με την παρουσίαση του προβλήματος της συντομότερης διαδρομής θα πρέπει να υπογραμμίσουμε ένα ακόμα σημείο. Το πρόβλημα μέχρι τώρα έχει αντιμετωπιστεί ως το πρόβλημα ελαχιστοποίησης της απόστασης μεταξύ ενός κόμβου προέλευσης και ενός κόμβου προορισμού ενός δικτύου. Όμως αυτό δε σημαίνει ότι οι τιμές των κλάδων πρέπει οπωσδήποτε να είναι αποστάσεις. Για παράδειγμα, οι κλάδοι μπορεί να αντιστοιχούν σε δραστηριότητες κάποιου είδους, όπου η τιμή για κάθε κλάδο είναι το κόστος της

δραστηριότητας. Σε αυτή την περίπτωση, το πρόβλημα είναι να βρεθεί η ακολουθία των δραστηριοτήτων που ελαχιστοποιεί το συνολικό κόστος. Ακόμη, η τιμή ενός κλάδου μπορεί να είναι ο χρόνος που χρειάζεται για την εκτέλεση της δραστηριότητας. Στην περίπτωση αυτή, το πρόβλημα είναι να βρεθεί η ακολουθία των δραστηριοτήτων που ελαχιστοποιεί το συνολικό χρόνο. Έτσι, στο κεφάλαιο αυτό η λέξη <<απόσταση>> μπορεί να θεωρηθεί και ως κόστος ή χρόνος ή κάποια άλλη ποσότητα.

3.4 Αναλυτικό Παράδειγμα

Ο αλγόριθμος χρειάζεται να συσχετιστεί με μία πρακτική εφαρμογή για αυτό το λόγο όροι όπως ακμή, κόμβος και γράφος αντικαθίστανται από διασταύρωση, δρόμος και χάρτης. Με αυτόν τον τρόπο ο αλγόριθμος θα κατανοηθεί ευκολότερα.

Το πρόβλημα είναι η εύρεση διαδρομής ανάμεσα σε ένα σημείο αφετηρίας και ένα σημείο προορισμού, τα οποία ορίζουν δύο διασταυρώσεις πάνω σε ένα χάρτη μίας πόλης. Η πρώτη κίνηση που κάνουμε είναι να σημειώσουμε ότι δεν έχουμε ακόμα επισκεφτεί κάθε διασταύρωση. Αυτό γίνεται μαρκάροντας την απόσταση κάθε διασταύρωσης στο χάρτη από την αρχή στο άπειρο. Υπό άλλες συνθήκες και σε διαφορετικές εφαρμογές αυτές οι διασταυρώσεις θα μπορούσαν να θεωρηθούν και χωρίς ετικέτα. Θα ακολουθήσουμε μία σειρά επαναλήψεων, όπου η πρώτη επανάληψη έχει ως αρχικό σημείο την τρέχουσα διασταύρωση και απόσταση ίση με το μηδέν. Θα επιλέγεται η τρέχουσα διασταύρωση για κάθε επανάληψη. Για τις υπόλοιπες επαναλήψεις θα θεωρούμε ως την διασταύρωση που δεν έχουμε ακόμα επισκεφτεί, την τρέχουσα και σε σχέση με όλες τις αποστάσεις των διασταυρώσεων που δεν έχουμε επισκεφτεί θα έχει την μικρότερη απόσταση από το αρχικό σημείο.

Σε κάθε διασταύρωση θα ενημερώνεται η απόσταση της κάθε γειτονικής διασταύρωσης που δεν έχουμε ακόμα επισκεφτεί, πρόκειται για μία διασταύρωση η οποία συνδέεται με την τρέχουσα. Έτσι καταφέρνουμε να καθορίσουμε την απόσταση των κοντινών διασταυρώσεων που δεν έχουμε ακόμα επισκεφτεί. Οπότε εάν υπολογιστεί η απόσταση μίας διασταύρωσης από το αρχικό σημείο και είναι μικρότερη από μία απόσταση που έχει υπολογιστεί σε προηγούμενη επανάληψη τότε την αντικαθιστά. Όταν γίνεται αυτή η αντικατάσταση, σημειώνεται και μία επιπρόσθετη πληροφορία για την συγκεκριμένη διασταύρωση, η οποία θα είναι η τρέχουσα. Αυτό συμβαίνει επειδή είναι αναγκαίο να γνωρίζουμε για μία διασταύρωση, εάν θέλουμε να διανύσουμε τη συντομότερη διαδρομή, ποια είναι η προηγούμενη από αυτήν. Εν ολίγοις όταν βρεθεί συντομότερο μονοπάτι τότε αντικαθιστά το προηγούμενο. Εφόσον έχουμε σημειώσει όλες τις αποστάσεις των γειτονικών

διασταυρώσεων για μία διασταύρωση τότε συνεχίζουμε στην επόμενη. Η νέα διασταύρωση θα έχει την μικρότερη απόσταση από το αρχικό σημείο και θα είναι η πρώτη φορά που θα την επισκεφτούμε.

Η διαδικασία αυτή επαναλαμβάνεται μέχρι να σημειωθεί ότι έχουμε επισκεφτεί τη διασταύρωση η οποία είναι το σημείο προορισμού και μέχρι να γίνει αυτό, σημειώνονται κάθε φορά οι αποστάσεις των γειτόνων της τρέχουσας διασταύρωσης. Κάνοντας οπισθοδρόμηση έχουμε τη δυνατότητα να βρούμε την συντομότερη διαδρομή από το αρχικό προς το τελικό σημείο. Αρχίζουμε από τον προορισμό και κοιτάμε ποια είναι η προηγούμενη σημειωμένη σε αυτόν διασταύρωση.

Είναι εντυπωσιακό το γεγονός ότι ο αλγόριθμος δεν κινείται από το αρχικό προς το τελικό σημείο, αντιθέτως η απόσταση της διασταύρωσης από το αρχικό σημείο είναι το χαρακτηριστικό εκείνο που την καθορίζει ως επόμενη τρέχουσα. Άρα ο αλγόριθμος κινείται γύρω από το αρχικό σημείο μέχρι να εντοπίσει το τελικό. Οπότε περιμένουμε ο κύκλος να έρθει σε επαφή με το σημείο προορισμού, μεγαλώνοντας την ακτίνα ενός κύκλου με κέντρο το αρχικό σημείο. Άρα είναι ξεκάθαρο ότι ο αλγόριθμος καταφέρνει και βρίσκει το συντομότερο μονοπάτι ανάμεσα σε δύο σημεία, αλλά γίνεται εμφανές ένα από τα μειονεκτήματα του αλγορίθμου που είναι η βραδύτητά του για συγκεκριμένες εφαρμογές. (Τσατσάνης, 2014)

3.5 Βήματα Αλγορίθμου

Πρώτα θα πρέπει να θέσουμε έναν κόμβο ως κόμβο αρχή ή κόμβο-πηγή και θα είναι το σημείο εκκίνησης του αλγορίθμου. Ως απόσταση ενός κόμβου Α θεωρούμε την απόσταση του από τον αρχικό κόμβο. Η δουλειά του αλγορίθμου Dijkstra είναι να ενημερώνει αυτές τις αποστάσεις αφού πρώτα τις αρχικοποιήσει.

Βήμα 1: Θέτουμε την απόσταση του κάθε κόμβου ίση με άπειρο και την απόσταση του κόμβου-πηγή ίση με μηδέν.

Βήμα 2: Δημιουργούμε ένα σύνολο το οποίο θα περιέχει τους κόμβους τους οποίους ο αλγόριθμος δεν έχει ακόμα επισκεφτεί. Στο βήμα αυτό δεν έχουμε επισκεφτεί κανέναν κόμβο άρα το σύνολο αυτό θα περιέχει όλους τους κόμβος του γράφου. Θέτουμε τον αρχικό κόμβο ως τρέχον κόμβο.

Βήμα 3: Για τον τρέχον κόμβο, βλέπουμε όλους τους γείτονες του που ο αλγόριθμος δεν έχει ακόμα επισκεφτεί και υπολογίζουμε τις αποστάσεις. Συγκρίνουμε την καινούρια απόσταση

με την προηγούμενη και αν είναι μικρότερη αντικαθιστούμε την παλιά με την νέα. Για παράδειγμα, αν σε έναν τρέχον κόμβο Α έχει σημειωθεί ότι η απόστασή του είναι 6 και η ακμή που συνδέει αυτόν με έναν γείτονα του Β έχει μήκος 2 τότε η απόσταση του Β μέσω του Α θα είναι 6+2=8. Εάν ο Β πριν τον υπολογισμό αυτόν είχε απόσταση μεγαλύτερη του 8 τότε η απόσταση του Β αντικαθίσταται και γίνεται 8, αλλιώς διατηρεί την παλιά της τιμή.

Βήμα 4: Όταν για όλους τους γείτονες του τρέχοντος κόμβου έχει ενημερωθεί η απόστασή τους, σημειώνουμε τον τρέχον κόμβο ότι ο αλγόριθμος τον έχει επισκεφτεί και τον αφαιρούμε από το σύνολο των κόμβων που δεν έχουμε επισκεφτεί. Έτσι, ένας κόμβος που ο αλγόριθμος έχει επισκεφτεί δεν θα προσπελαστεί ξανά.

Βήμα 5: Ο αλγόριθμος τερματίζει για τις δύο παρακάτω περιπτώσεις:

Περίπτωση 1: αν ο αλγόριθμος εκτελείται για την εύρεση της συντομότερης διαδρομής ανάμεσα σε δύο συγκεκριμένους κόμβους, τερματίζει όταν ο κόμβος προορισμού σημειωθεί ότι τον έχουμε επισκεφτεί.

Περίπτωση 2: αν ο αλγόριθμος εκτελείται για την εύρεση των αποστάσεων όλων των κόμβων του γράφου από τον κόμβο-πηγή, τότε ο αλγόριθμος τερματίζει όταν η μικρότερη απόσταση ανάμεσα στους κόμβους του συνόλου των κόμβων που δεν έχουμε επισκεφτεί είναι άπειρη. Αυτό σημαίνει ότι δεν υπάρχει σύνδεση ανάμεσα στον κόμβο-πηγή και στους εναπομείναντες κόμβους.

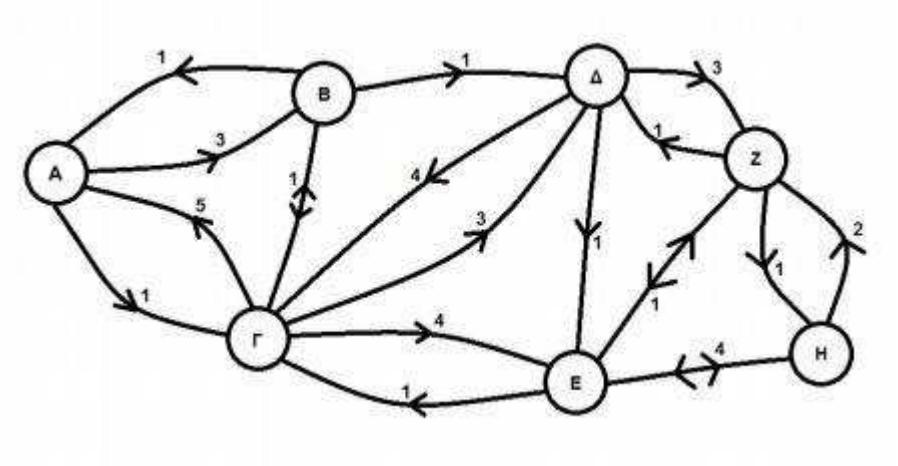
Βήμα 6: Διαλέγουμε από το σύνολο των κόμβων που δεν έχουμε επισκεφτεί τον κόμβο με την μικρότερη απόσταση και τον θέτουμε ως τρέχον κόμβο. Στη συνέχεια, επιστρέφουμε στο βήμα 3. [4]

3.6 Πολυπλοκότητα

Ο πρωτότυπος αλγόριθμος Dijkstra εκτελείται σε $O(V^2)$ (όπου V είναι ο αριθμός των κόμβων). Η υλοποίηση βασίζεται στην ουρά προτεραιότητας υλοποιημένη με Fibonacci Heap και τρέχει σε χρόνο O(E+VlogV) (όπου E ο αριθμός των ακμών) (Fredman & Tarjan, 1984). Αυτός είναι ασυμπτωτικά ο γρηγορότερος γνωστός αλγόριθμος εύρεσης συντομότερου μονοπατιού μόνης πηγής για αφηρημένους κατευθυνόμενους γράφους με μη αρνητικά βάρη. (Τσατσάνης, 2014)

3.7 Εκτέλεση Παραδείγματος

Ο παρακάτω κατευθυνόμενος γράφος θα είναι το παράδειγμα μας για την εκτέλεση του αλγορίθμου Dijkstra. Το παράδειγμα θα λυθεί εξηγώντας βήμα-βήμα την διαδικασία επίλυσης.

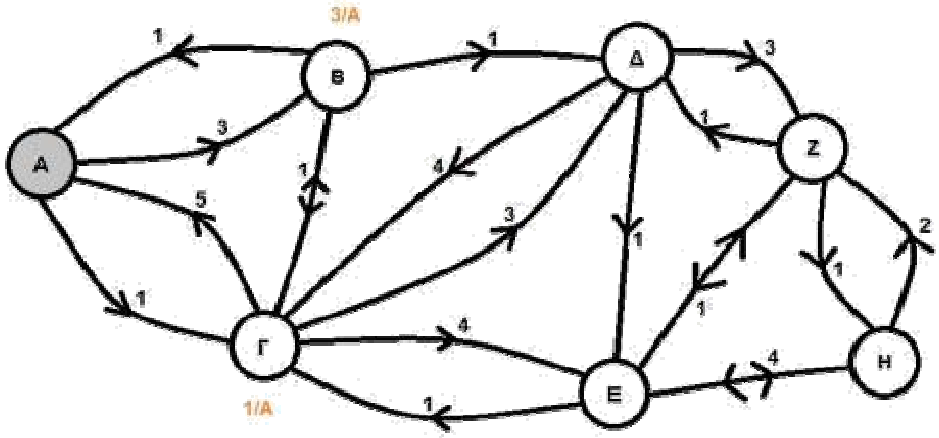


Εικόνα 7: Κατευθυνόμενος γράφος (Τσατσάνης, 2014)

Το ζητούμενο του προβλήματος αυτού είναι να βρεθεί η συντομότερη διαδρομή από τον κόμβο Α μέχρι τον κόμβο Η.

Ξεκινώντας από τον κόμβο Α θα κοιτάξουμε τους γειτονικούς κόμβους του. Ως γειτονικός κόμβος σε έναν κατευθυνόμενο γράφο ενός κόμβου Κ, θεωρείται ο κόμβος που συνδέεται άμεσα με μια ακμή που επιτρέπει την μετάβαση από τον κόμβο Κ στον γειτονικό. Εάν η ακμή που συνδέει τον κόμβο Κ με έναν άλλο κόμβο Ζ επιτρέπει μόνο την μετάβαση από τον Ζ στον Κ τότε ο Κ είναι γειτονικός του Ζ αλλά ο Ζ δεν είναι γειτονικός του Κ.

Παρακάτω φαίνεται πως ο κόμβος Α συνδέεται με τον Β με κόστος 3 και με τον Γ με κόστος 1. Τα παραπάνω δεδομένα σημειώνονται πάνω στους γειτονικούς κόμβους του Α.

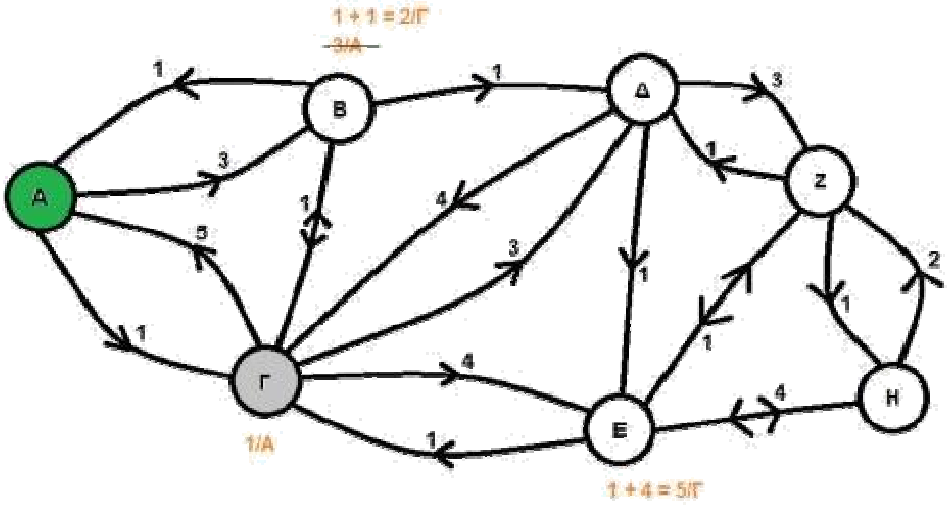


Εικόνα 8: Σύνδεση κόμβου Α με Β (Τσατσάνης, 2014)

Επίσης σε κάθε γειτονικό κόμβο σημειώνεται ότι προς το παρόν για να φτάσουμε σε αυτόν ο προηγούμενος κόμβος στο μονοπάτι θα είναι ο Α.

Στο επόμενο βήμα εφόσον εξετάσαμε όλους τους γείτονες του Α θα σημειώσουμε τον Α ως επεξεργασμένο κόμβο (visited) έτσι ώστε να μην χρειαστεί να τον επισκεφθούμε ξανά στο μέλλον. Ο επόμενος κόμβος για τον οποίο θα εξετασθούν οι γείτονές του θα είναι ο κόμβος με την μικρότερη απόσταση από τον κόμβο Α. Υπενθυμίζουμε πως για όλους τους κόμβους που δεν έχουμε επισκεφθεί ακόμα η απόσταση από τον Α θεωρείται άπειρη. Εάν κρατήσουμε ένα διάνυσμα με στοιχεία τους κόμβους και την απόστασή τους από τον Α (κόμβο-πηγή) προς το παρόν θα έχουμε το εξής:

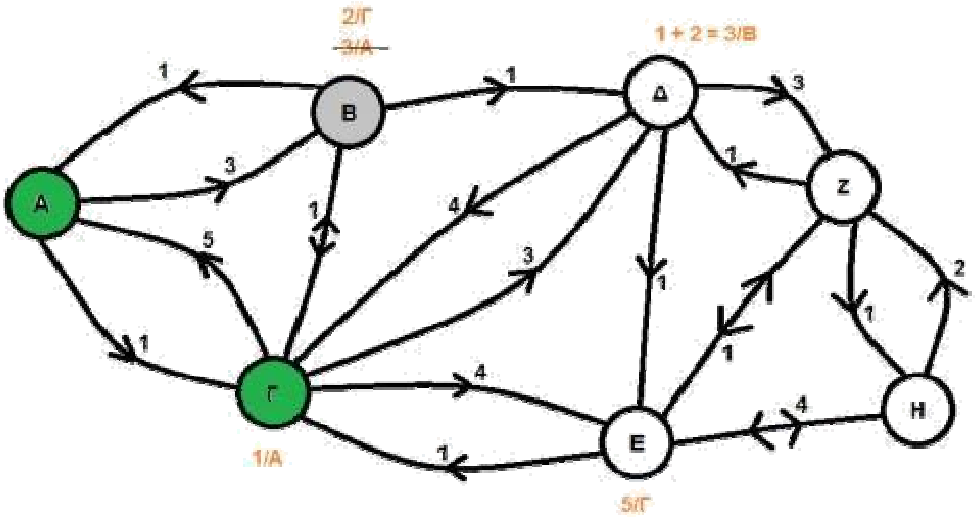
Επομένως ο επόμενος κόμβος θα είναι ο Γ αφού έχει την μικρότερη απόσταση από τον Α, δηλαδή 1. Ο Α θεωρείται επεξεργασμένος κόμβος και δεν λαμβάνεται υπόψη στην σύγκριση της απόστασης. (Τσατσάνης, 2014)



Εικόνα 9: Ανάλυση κόμβου Γ (Τσατσάνης, 2014)

Όπως φαίνεται και στην παραπάνω εικόνα οι μη επεξεργασμένοι γείτονες του Γ είναι ο Β και ο Ε. Τον Ε αφού τον επισκεπτόμαστε πρώτη φορά σημειώνουμε την απόστασή του από τον Α ως το άθροισμα της απόστασης από τον Γ στον Ε συν της απόστασης του Γ από τον Α. Επίσης σημειώνουμε πως προς το παρόν στον Ε πηγαίνουμε με προηγούμενο κόμβο τον Γ.
Για τον κόμβο Β αφού έχουμε σημειώσει στο παρελθόν κάποια απόσταση (εδώ 3) συγκρίνουμε την νέα απόσταση με την παλιά. Η νέα απόσταση είναι το άθροισμα της απόστασης από τον Γ στον Β συν την απόσταση του Γ από τον Α, δηλαδή 1+1=2. Εφόσον 2<3 αντικαθιστούμε την παλιά σημειωμένη απόσταση με την νέα. Διαφορετικά θα αφήναμε την παλιά όπως έχει. Επίσης σημειώνουμε ότι προς το παρόν για να φτάσουμε στον κόμβο Β ο προηγούμενος κόμβος θα είναι ο Γ αφού αν ήταν ο Α (όπως είχαμε σημειώσει προηγουμένως θα διανύσουμε μεγαλύτερη απόσταση). Αφού σημειώσουμε τον Γ ως επεξεργασμένο κόμβο (visited) το διάνυσμα αποστάσεων θα είναι τώρα:

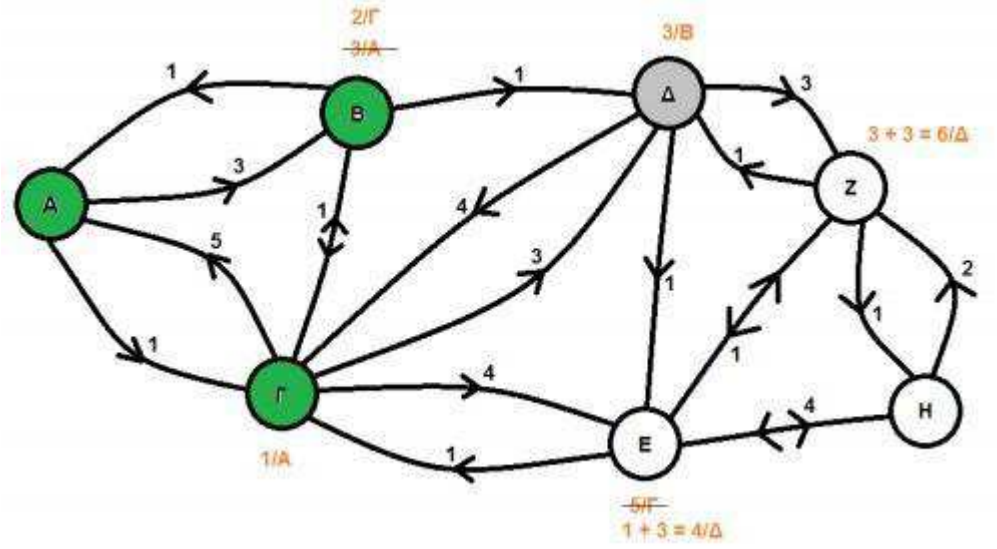
Άρα ο επόμενος κόμβος θα είναι ο Β ως ο μη επεξεργασμένος κόμβος με την μικρότερη απόσταση από τον Α.



Εικόνα 10: Ανάλυση κόμβου Β (Τσατσάνης, 2014)

Ο μοναδικός γείτονας του Β είναι ο . Σημειώνουμε την απόστασή του από τον Α ως το άθροισμα της απόστασης από τον Β στον συν την απόσταση του Β από τον Α. Σημειώνουμε τον Β ως επεξεργασμένο κόμβο (visited) και έτσι το διάνυσμα αποστάσεων θα γίνει:

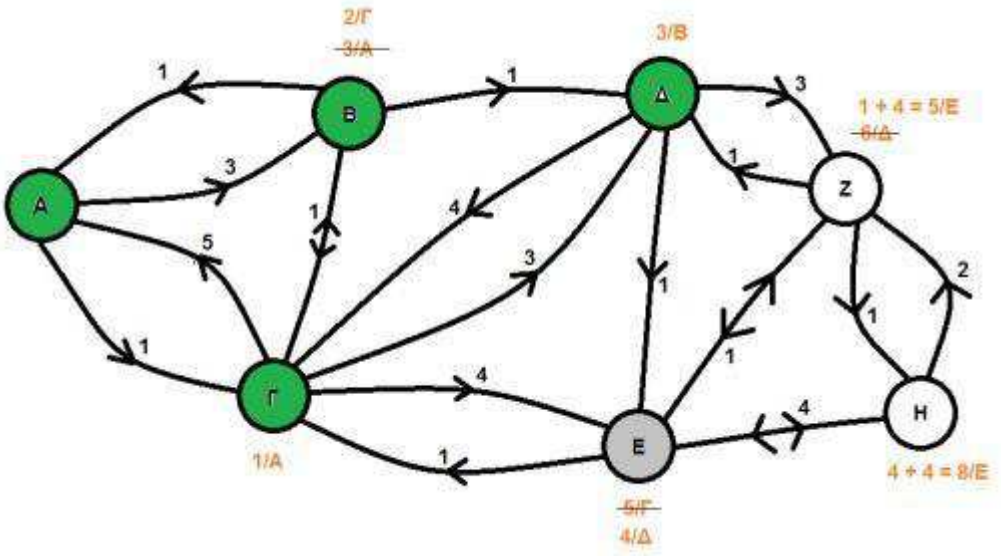
Άρα ο επόμενος κόμβος θα είναι ο .



Εικόνα 11: Ανάλυση κόμβου (Τσατσάνης, 2014)

Οι γείτονες του είναι ο Ζ και ο Ε. Καθώς τον Ζ τον επισκεπτόμαστε πρώτη φορά σημειώνουμε την απόσταση του από τον Α κατά τα γνωστά. Για τον Ε συγκρίνουμε τη νέα απόσταση με την προηγούμενή του. Εφόσον η νέα είναι μικρότερη αντικαθιστά την προηγούμενη και σημειώνουμε πως για να πάμε στον Ε ξεκινώντας από τον Α ο προηγούμενος κόμβος θα είναι ο και όχι ο Γ. Το διάνυσμα αποστάσεων θα είναι τώρα:

Επόμενος κόμβος θα είναι ο Ε.

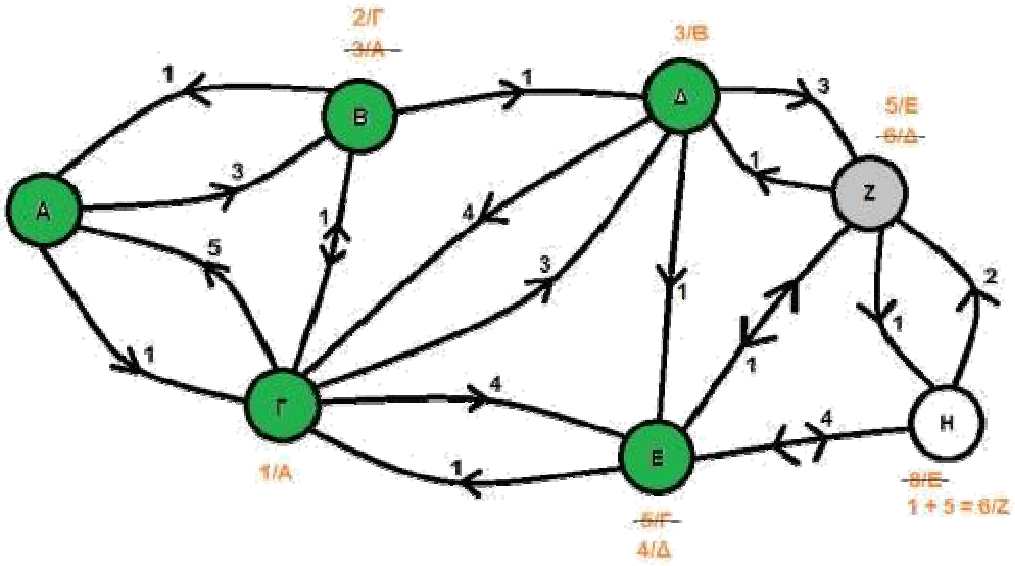


Εικόνα 12: Ανάλυση κόμβου Ε (Τσατσάνης, 2014)

Ενημερώνουμε την απόσταση του Ζ αφού τυχαίνει να είναι μικρότερη μέσω του Ε από ότι μέσω του και σημειώνουμε και την απόσταση του Η. Προσοχή, ο αλγόριθμος δεν τερματίζει αφού επισκεφθήκαμε τον τελικό κόμβο Η και βρήκαμε μια απόσταση του από τον Α. Πρέπει στο διάνυσμα απόστασης να έχει και την μικρότερη απόσταση σε σχέση με τους υπόλοιπους μη επεξεργασμένους κόμβους. Το διάνυσμα απόστασης τώρα είναι:

[(Α):0/Α,(Β):2/Γ, (Γ):1/Α,( ):3/Β,(Ε):4/ , Ζ:5/Ε,Η:8/Ε]

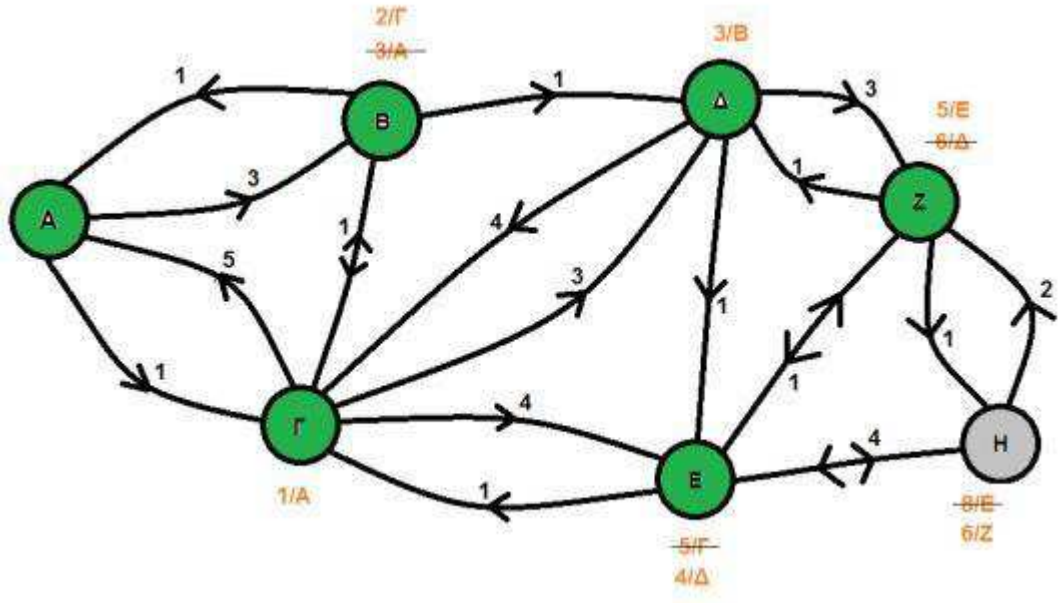
Άρα ο επόμενος κόμβος θα είναι ο Ζ.



Εικόνα 13: Ανάλυση κόμβου Z (Τσατσάνης, 2014)

Ενημερώνουμε την απόσταση του μοναδικού γείτονα του κόμβου Z. Το διάνυσμα απόστασης θα είναι τώρα:

Άρα ο επόμενος κόμβος θα είναι ο Η και αφού είναι ο τελικός κόμβος ο αλγόριθμος τερματίζει.



Εικόνα 14: Ανάλυση κόμβου Η (Τσατσάνης, 2014)

Το συντομότερο μονοπάτι από τον Α στον Η βρίσκεται ως εξής:

Ξεκινάμε από τον Η και βλέπουμε ποιος είναι ο προηγούμενος σημειωμένος κόμβος. Εδώ είναι ο Ζ. Πηγαίνουμε στο Ζ και βλέπουμε ποιος είναι ο προηγούμενος από αυτόν σημειωμένος κόμβος. Εδώ είναι ο Ε. Ακολουθώντας την λογική αυτή της οπισθοδρόμησης κάποια στιγμή θα εντοπίσουμε τον κόμβο Α και θα έχουμε σημειώσει το μονοπάτι που ζητείται. Στο παράδειγμα θα είναι:

$$A \rightarrow \Gamma \rightarrow B \rightarrow \Delta \rightarrow E \rightarrow Z \rightarrow H$$

(Τσατσάνης, 2014)

3.8 Απόδειξη ορθότητας αλγορίθμου Dijkstra

Ο αλγόριθμος του Dijkstra είναι ένας άπληστος (greedy) αλγόριθμος, δηλαδή σε κάθε βήμα υπολογίζει την τοπικά βέλτιστη λύση. Στο τέλος, συνθέτει τις τοπικές λύσεις και επιστρέφει μια συνολική. Όμως, το γεγονός ότι οι τοπικές λύσεις είναι βέλτιστες δεν εγγυάται λογικά ότι και η σύνθεσή τους θα είναι μια συνολικά βέλτιστη λύση. Για το λόγο αυτόν είναι απαραίτητη μια απόδειξη ορθότητας του αλγορίθμου.

Αρχικά ορίζεται το σύνολο S, το οποίο περιέχει τους κόμβους του γράφου που αλγόριθμος έχει ήδη προσπελάσει. Αυτό σημαίνει ότι όσο ο αλγόριθμος εκτελείται, το σύνολο S θα μεγαλώνει. Ο αλγόριθμος μόλις προσπελάσει έναν κόμβο u θα γνωρίζει το μονοπάτι από τον αρχικό κόμβο s προς τον κόμβο u.

Ορίζουμε διαδρομή $P_{i,j}$ ως μια διαδρομή (μονοπάτι) του γράφου που ξεκινά από τον κόμβο i και καταλήγει στον κόμβο j. Ως $|P_{i,j}|$ ορίζεται ο αριθμός των κόμβων που αποτελούν το μονοπάτι. Ορίζουμε την απόσταση R_{ij} ως το κόστος της ακμής που ενώνει τους κόμβους i και j.

Έστω $P_{s,v}$ η διαδρομή (μονοπάτι) που έχει σημειώσει ο αλγόριθμος από τον αρχικό κόμβο s στον κόμβο v. Θα δείξουμε ότι για κάθε κόμβο v του συνόλου S, σε κάθε στάδιο της εκτέλεσης του αλγορίθμου, η $P_{s,v}$ είναι η ελάχιστη διαδρομή. Αυτό μας δείχνει άμεσα ότι ο αλγόριθμος επιστρέφει τη βέλτιστη λύση, αφού στο τελευταίο βήμα το σύνολο S θα περιέχει τον κόμβο προορισμού t και επομένως θα γνωρίζει την διαδρομή $P_{s,t}$ η οποία θα είναι και η ελάχιστη. Η απόδειξη είναι επαγωγική ως προς το μέγεθος του συνόλου S (των κόμβων που έχουν μέχρι εκείνο το βήμα προσπελαστεί).

Για $|S|=1$, έχουμε ότι $S=\{s\}$ και το κόστος της διαδρομής $P_{s,s}$ είναι 0. Αφού δεν υπάρχουν αρνητικά βάρη, αυτή είναι όντως η συντομότερη διαδρομή, Έστω ότι για $|S|=k$, με $k>1$, ο ισχυρισμός ισχύει. Μένει να δείξουμε ότι ο ισχυρισμός ισχύει και για $|S|=k+1$, έχοντας αυξήσει το σύνολο των επεξεργασμένων κόμβων κατά 1, με την προσθήκη ενός νέου κόμβου,

έστω του v. Έστω ότι ο κόμβος u είναι ο αμέσως προηγούμενος του v στη διαδρομή $P_{s,v}$ που έχει σημειώσει ο αλγόριθμος. Λόγω της υπόθεσης της επαγωγής, η διαδρομή $P_{s,u}$ είναι η συντομότερη από τον αρχικό κόμβο στον u.

Έστω ότι στο σύνολο S οι κόμβοι x_i, είναι οι μόνοι κόμβοι όπου έχουν γειτονικούς κόμβους εκτός του συνόλου S. Προφανώς ο κόμβος u θα είναι ένας από τους κόμβους x_i.
Τώρα έστω ένα σύνολο Z με κόμβους y_j που είναι εκτός του συνόλου S και είναι γειτονικοί με τουλάχιστον έναν κόμβο x_i του S. Κατά την επανάληψη k+1 του αλγορίθμου, εξετάζονται όλες οι αποστάσεις $R_{xi,yj}$ (έστω οι κόμβοι u και v αντίστοιχα) και προσθέτει τον κόμβο v στο σύνολο S. Στη συνέχεια αποθηκεύει το μονοπάτι $P_{s,v}$ ως την ένωση $P_{s,u}$ και $P_{u,v}$ με $|P_{u,v}|=2$. Επομένως αφού η διαδρομή $P_{s,u}$ είναι η συντομότερη από την υπόθεση και η $P_{u,v}$ είναι η συντομότερη σε σχέση με όλες τις άλλες $P'_{u,v}$ αφού ο αλγόριθμος την εντόπισε τοπικά σημαίνει ότι η $P_{s,v}$ είναι η συντομότερη. (Τσατσάνης, 2014)

3.9 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 μία τροποποίηση του αλγόριθμου Dijkstra με μία διπλά συνδεδεμένη λίστα. Κάθε κόμβος στη λίστα έχει έναν δείκτη προς τον πρόγονό του και έναν η 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 σχετικά ορολογία. Οι αλγόριθμοι, που θα αναλυθούν στην συνέχεια, αποτελούν περαιτέρω βελτιώσεις αυτού. Να σημειωθεί πως γενικά υπάρχει μία πληθώρα υλοποιήσεων κάθε αλγορίθμου. Στην εργασία 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 αλγορίθμου μετατρέπουμε κάθε κόμβο της βάσης δεδομένων σε μία δομή δεδομένων, που περιέχει τουλάχιστον τις συντεταγμένες x, y, κάποιες ιδιότητες του δρόμου, μία 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:

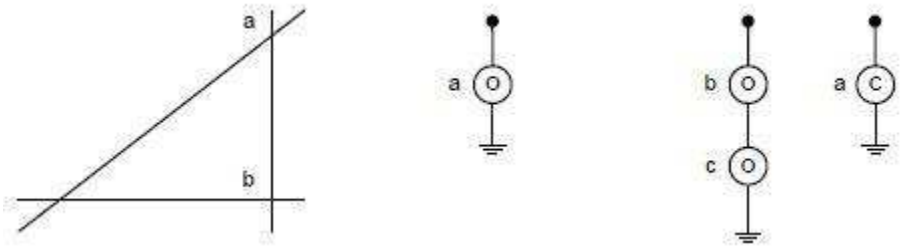
OPEN και CLOSED:

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. Στην εικόνα 15 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 κύκλος, 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.



Έναρξη από τον κόμβο a Παραγμένος κόμβος a Επεκταμένος κόμβος a

Εικόνα 15: Συνδεδεμένες λίστες (Σομπόνης, 2012)

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. (Σομπόνης, 2012)

3 .10 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 (τιμής 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:

   I. Υπολόγισε το κόστος του n: g(n)=κόστος του BEST+κόστος από τον BEST στον n.
   II. 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 κόστος του κόμβου n και όρισε τον δείκτη του να δείχνει στον BEST.
   III. 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 κόστος του κόμβου n, όρισε τον δείκτη του να δείχνει στον BEST και μετέφερε τον στη λίστα OPEN.
   IV. 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. Μία 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 [7].

Μία άλλη τεχνική, συχνά χρησιμοποιούμενη στον σχεδιασμό διαδρομής, είναι να περιλαμβάνονται απαγορευμένες καταστάσεις στη δομή δεδομένων κάθε κόμβου. Γενικά, ένα οδικό δίκτυο περιέχει έναν σημαντικό αριθμό από αδιέξοδα, κλειστούς δρόμους και μονόδρομους, που δεν 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 δεν μπορούν να χρησιμοποιηθούν για την κατασκευή του δέντρου αναζήτησης. Συνεπώς το σύστημα αυτομάτως τους αποφεύγει. Μία 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 για 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 αλγορίθμου δρομολόγησης, τότε όλοι οι αυτοκινητόδρομοι θα επισημανθούν ως απαγορευμένοι ούτως ώστε να ικανοποιηθεί η απαίτηση του χρήστη [8].

Στις εφαρμογές πλοήγησης οχημάτων υπάρχει μία διαφορά μεταξύ της χρήσης του αλγορίθμου 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, κατά 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 έναν 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 αλγορίθμου.

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 απόγονοί της θα βρίσκονται στο επίπεδο 1. Με αυτήν την υπόθεση, η πολυπλοκότητα του τροποποιημένου αλγορίθμου 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. (Σομπόνης 2012)

ΚΕΦΑΛΑΙΟ 4

ΑΛΓΟΡΙΘΜΟΣ Α*(Α-ΑΣΤΡΟ)

4.1 Ευριστική αναζήτηση

Εάν μία αναζήτηση περιέχει πληροφορίες για τον αριθμό των βημάτων ή το κόστος από την αρχική και την παρούσα κατάσταση μέχρι την τελική τότε αυτή η αναζήτηση είναι μία πληροφορημένη στρατηγική αναζήτησης και ονομάζεται ευριστική αναζήτηση. Η αποδοτικότητα της διαδικασίας αναζήτησης βελτιώνεται χρησιμοποιώντας τις πληροφορίες αυτές, θυσιάζοντας όμως αξιώσεις πληρότητας. Τέτοια παραδείγματα ενημερωμένων μεθόδων αναζήτησης είναι: η best-first αναζήτηση, η memory bounded αναζήτηση και οι αλγόριθμοι επαναληπτικής βελτίωσης, όπως ο hill-climbing search και ο simulated annealing.

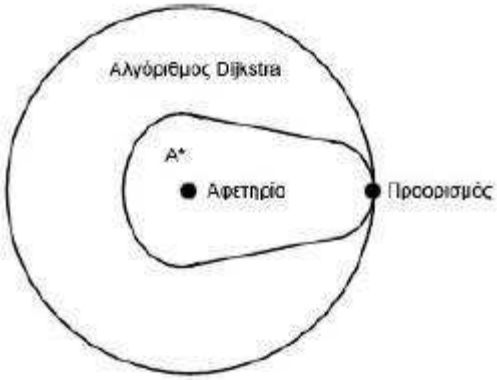
Όταν δεν γνωρίζουμε προς ποια κατεύθυνση πρέπει να κινηθούμε τότε θα πρέπει να πραγματοποιήσουμε μία αναζήτηση. Αν το γνωρίζαμε εξ αρχής, το πρόβλημα θα ήταν πολύ πιο απλό. Η ευριστική πληροφορία βοηθάει στον καθορισμό του περισσότερου "υποσχόμενου" κόμβου, ποιοι διάδοχοι θα παραχθούν και ποια άσχετα κλαδιά αναζήτησης θα περικοπούν με αποτέλεσμα η αναζήτηση να είναι πιο αποδοτική. Ο σκοπός της ευριστικής πληροφορίας είναι η παροχή μίας 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 στη 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. (Σομπόνης 2012)

4.2 Αλγόριθμος A*

Η πιο δημοφιλής ευριστική αναζήτηση, που εφαρμόζεται ευρέως στον σχεδιασμό διαδρομής, είναι ο αλγόριθμος A* [9], ο οποίος χρησιμοποιεί την best-first μέθοδο. Ο αλγόριθμος A* έχει χρησιμοποιηθεί σε πεδία όπως η πλοήγηση οχημάτων, η ρομποτική 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 μια 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 κατανοητή 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 στη παρακάτω εικόνα 16.



Εικόνα 16: Οι χώροι αναζήτησης στον αλγόριθμο A* και τον αλγόριθμο Dijkstra (Σομπόνης, 2012)

Ένας καλός τρόπος για τη χρήση ευριστικής πληροφόρησης είναι ο υπολογισμός μίας ευριστικής συνάρτησης, η οποία αξιολογεί κάθε κόμβο, που δημιουργείται, για να προσδιορίσει αν είναι καλός ή κακός. Με τον τρόπο αυτό, έχοντας την 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:

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.

Ο καθορισμός του κόστος, 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, μία λύση είναι η υιοθέτηση της παρακάτω εξίσωσης για την ευριστική συνάρτηση αξιολόγησης:

—— —— (4.2)

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 ταξιδιού πάντα θα υποτιμάται. Εν συντομία, ο σκοπός του αλγόριθμου είναι να ελαχιστοποιήσει το f '(n) και εν προκειμένω το t'(n), δηλαδή τον χρόνο ταξιδιού.

Η λειτουργία του αλγορίθμου 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. (Σομπόνης, 2012)

4.3 Πολυπλοκότητα

Η πολυπλοκότητα χρόνου του αλγόριθμου A* εξαρτάται από την ευρετική συνάρτηση. Στην χειρότερη περίπτωση, ο αριθμός των κόμβων εξελίσσεται εκθετικά κατά την διάρκεια της λύσης, αλλά η εξέλιξη γίνεται πολυωνυμική όταν ο χώρος αναζήτησης είναι δέντρο, όπου υπάρχει μία τελική κατάσταση και η ευρετική συνάρτηση h συναντά την παρακάτω προϋπόθεση:

Όπου h*(x) είναι η βέλτιστη ευρετική συνάρτηση, δηλαδή το ακριβές κόστος για να φτάσουμε από τον x στον στόχο. Με άλλα λόγια, το σφάλμα της h δεν θα μεγαλώνει γρηγορότερα από τον λογάριθμο της βέλτιστης ευρετικής h* η οποία επιστρέφει την πραγματική απόσταση από τον x προς τον στόχο.

4.4 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:
    a. Υπολόγισε το κόστος του n: g(n)=κόστος του BEST+κόστος από τον BEST στον n.
    b. 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 κόστος του κόμβου n και όρισε τον δείκτη του να δείχνει στον BEST.
    c. 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 κόστος του κόμβου n, όρισε τον δείκτη του να δείχνει στον BEST και μετέφερε τον στη λίστα OPEN.

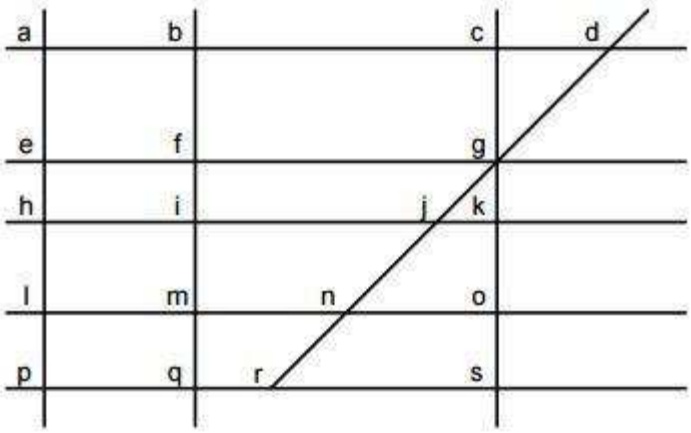
d. Εάν ο n δεν είναι ήδη ούτε στη λίστα OPEN ούτε στη λίστα CLOSED, όρισε τον δείκτη του να δείχνει στον BEST, τοποθέτησε τον στη λίστα OPEN και υπολόγισε την ευριστική συνάρτηση αξιολόγησης για τον κόμβο n:

Επανέλαβε το βήμα 2.

3) Από τον BEST διέσχισε τους δείκτες προς τα πίσω μέχρι τον κόμβο αφετηρίας και ανέφερε τη λύση της διαδρομής. (Σομπόνης, 2012)

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.

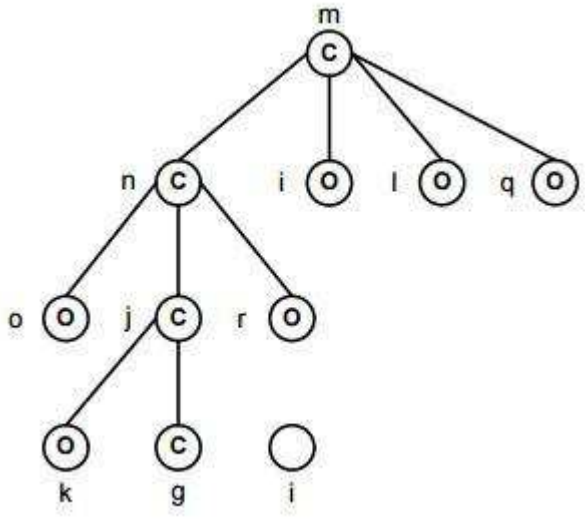
Για την καλύτερη κατανόησή του ακολουθεί ένα παράδειγμα με το πως ο αλγόριθμος A* δημιουργεί ένα δέντρο αναζήτησης για το απλό οδικό δίκτυο της εικόνας 17, όπου κάθε γράμμα αντιστοιχεί σε μία διασταύρωση, με τον κόμβο αφετηρίας να είναι στο m και ο προορισμός στο g.



Εικόνα 17: Ένα απλό οδικό δίκτυο (Σομπόνης, 2012)

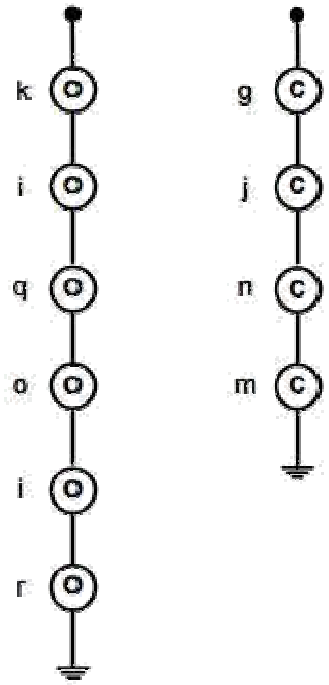
Με την ολοκλήρωση του σχεδιασμού της διαδρομής το τελικό δέντρο αναζήτησης, που προκύπτει, φαίνεται στην εικόνα 18. Όπως και πριν, χρησιμοποιούμε ένα μικρό κύκλο για να παραστήσουμε κάθε κόμβο και μια ευθεία γραμμή για να παραστήσουμε τους δείκτες από έναν κόμβο προς τον πρόγονό του και τους απογόνους 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. Αυτό όμως δεν 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.



Εικόνα 18: Δέντρο αναζήτησης (Σομπόνης, 2012)

Οι τελικές λίστες OPEN και CLOSED για το συγκεκριμένο παράδειγμα απεικονίζονται στην εικόνα 19. 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.



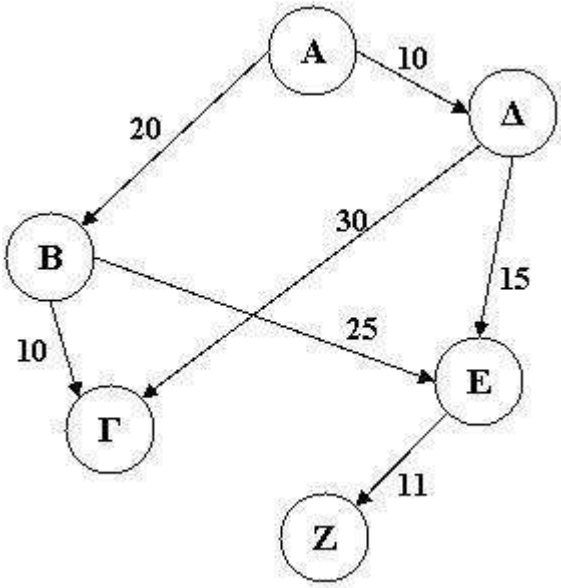
Εικόνα 19: Οι λίστες OPEN και CLOSED (Σομπόνης, 2012)

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 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. Για 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 [10]. 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 από τα στοιχεία κλειδιά των συστημάτων πλοήγησης. (Σομπόνης, 2012)

4.4.1 Παράδειγμα 1

Παρακάτω βλέπουμε έναν γράφο έξι πόλεων. Πάνω στις ακμές φαίνονται οι πραγματικές αποστάσεις των πόλεων που συνδέονται άμεσα μεταξύ τους. Θεωρούμε ως πρόβλημα την εύρεση διαδρομής από την πόλη A στην πόλη Z. Επίσης θεωρούμε ευριστική συνάρτηση η

οποία εκτιμά ως απόσταση δύο πόλεων την ευθεία απόστασής τους και στον πίνακα έχουμε τις ευθείες αποστάσεις όλων τον πόλεων από την πόλη Z.



Εικόνα 20: Γράφος έξι πόλεων

Πίνακας 4: Πίνακας αποστάσεων από την πόλη Z

Πόλη	**Ευθεία απόσταση από την Z**
A	30
B	18
Γ	10
	20
E	11

Ο αλγόριθμος A* βαθμολογεί τις καταστάσεις με το άθροισμα της απόστασής τους από την αρχική κατάσταση (η οποία είναι γνωστή) και της εκτίμησης της απόστασής τους από την κοντινότερη τελική κατάσταση. Στη συνέχεια, επιλέγει από το μέτωπο αναζήτησης την κατάσταση εκείνη με τη μικρότερη "βαθμολογία". Εφαρμόζοντας λοιπόν τον αλγόριθμο A* στο συγκεκριμένο πρόβλημα γράφου παίρνουμε τον παρακάτω πίνακα.

Πίνακας 5: Πίνακας επίλυσης γράφου

Μέτωπο αναζήτησης	Κλειστό σύνολο	Τρέχουσα κατάσταση	Παιδιά
A^{30}	-	A^{30}	AB^{38}, $A\Delta^{30}$
$A\Delta^{30}$,AB^{38}	A	$A\Delta^{30}$	$A\Delta\Gamma^{50}$, $A\Delta E^{36}$
$A\Delta E^{36}$, $A\Delta\Gamma^{50}$, AB^{38}	A, AΔ	$A\Delta E^{36}$	$A\Delta EZ^{36}$
$A\Delta EZ^{36}$, $A\Delta\Gamma^{50}$, AB^{38}	A, AΔ, AΔE	$A\Delta EZ^{36}$	Λύση

Ο αλγόριθμος A* εξασφαλίζει ότι η πρώτη λύση που θα βρει, θα είναι και η βέλτιστη στην περίπτωση που η ευριστική συνάρτηση που χρησιμοποιεί δίνει πάντα υποεκτιμήσεις, δηλαδή εκτιμήσεις μικρότερες ή ίσες της πραγματικής απόστασης κάθε κατάστασης από την κοντινότερη τελική. Μια τέτοια ευριστική συνάρτηση ονομάζεται αποδεκτή.

Παρατηρούμε ότι ο αλγόριθμος A* βρήκε καλύτερη λύση, την ΑΔΕΖ με συνολικό μήκος 36. Όπως μάλιστα είναι εύκολο να διαπιστωθεί ότι αυτή είναι η καλύτερη λύση για το συγκεκριμένο πρόβλημα. Αυτό συνέβη γιατί η ευριστική συνάρτηση που χρησιμοποιήθηκε παράγει πάντα υποεκτιμήσεις, οπότε σύμφωνα με ότι είπαμε παραπάνω, σε τέτοιες περιπτώσεις ο αλγόριθμος A* εξασφαλίζει ότι η πρώτη λύση που θα βρει θα είναι και η βέλτιστη.

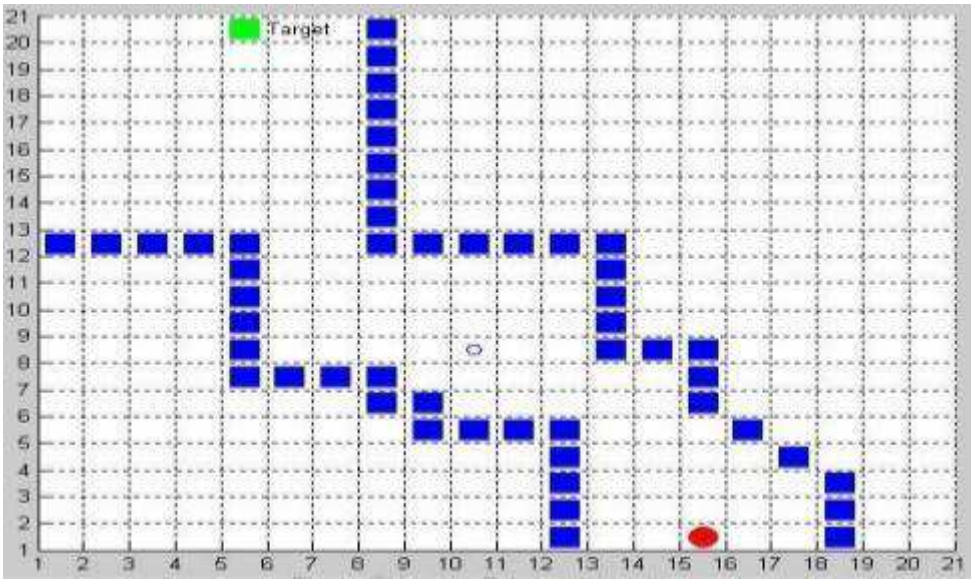
4.4.2 Παράδειγμα 2

Οι ElHalawany, Abdel-Kader,TagEldeen, Elsayed (2013) παρουσίασαν στην εργασία τους μία τροποποιημένη έκδοση του αλγορίθμου A*. Αυτή η έκδοσή του λαμβάνει υπόψη περισσότερους παράγοντες οπότε καταλήγει και σε πιο σωστές λύσεις εύρεσης διαδρομής [21].

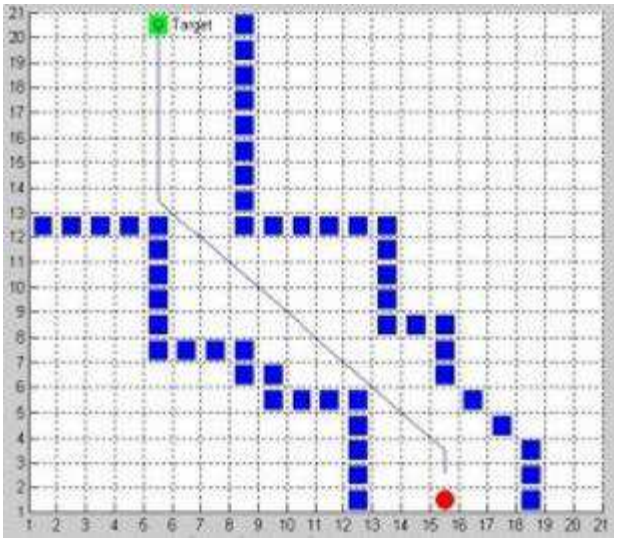
Ας υποθέσουμε ότι έχουμε μία ρομποτική πλατφόρμα, η οποία πρέπει να μεταφερθεί από την αρχική θέση (Start) στον τελικό προορισμό (Target). Στην περιοχή αυτή υπάρχουν εμπόδια, αντικείμενα, τοίχοι και ακίνητες επιφάνειες από τα οποία το ρομπότ δεν μπορεί να περάσει κατευθυνόμενο προς τον προορισμό του.

Για να λύσουμε το πρόβλημα αυτό χρειάζεται να κάνουμε συγκεκριμένες διαδικασίες. Αρχικά απλοποιούμε την περιοχή αυτή χωρίζοντάς την σε τετράγωνα. Κάθε στοιχείο του πίνακα αναπαριστά καθένα από τα τετράγωνα όπως φαίνεται στην εικόνα 21. Κάθε τετράγωνο λαμβάνει μία κατάσταση ως προς την οποία μπορεί να μετακινηθεί ή όχι. Για να βρεθεί η λύση θα πρέπει να καθοριστούν τα τετράγωνα που θα ακολουθήσει το ρομπότ για να φτάσει από την αρχή (κύκλος) στον τελικό προορισμό του (τετράγωνο). Αφού καθοριστούν

τα τετράγωνα, το ρομπότ είναι έτοιμο να ξεκινήσει την πορεία του προς τον προορισμό (Target). Θα μετακινείται κάθε φορά στο κέντρο κάθε νέου τετραγώνου και θα περιστρέφεται με τέτοιο τρόπο ώστε να αντικρίζει το επόμενο τετράγωνο στο οποίο θα μετακινηθεί. Ένα τέτοιο παράδειγμα φαίνεται στην εικόνα 22. Τα κεντρικά σημεία των τετραγώνων τα ονομάζουμε κόμβους.



Εικόνα 21: Περιοχή δοκιμής για τον A*, το τετράγωνο πάνω είναι ο προορισμός, ο κύκλος το ρομπότ και τα υπόλοιπα μαύρα τετράγωνα τα εμπόδια [21]



Εικόνα 22: Η λύση του παραδείγματος [21]

4.5 Αξιολόγηση του βασικού A* αλγορίθμου

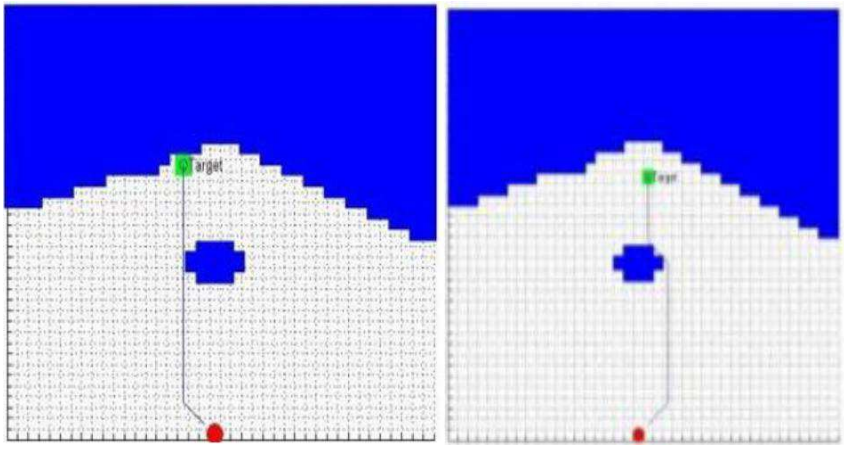
Εάν εκτελέσουμε διάφορα παραδείγματα του αλγορίθμου κάθε φορά με διαφορετικό σενάριο, θα συμπεράνουμε κάποια μειονεκτήματα. Στις παρακάτω εικόνες βλέπουμε ότι ο

αλγόριθμος δημιουργεί ένα μονοπάτι από την αφετηρία μέχρι τον προορισμό με σκοπό να το ακολουθήσει το ρομπότ όπως φαίνεται στις εικόνες (23) και (24). Το μονοπάτι αυτό είναι αποτελεσματικό από άποψη κόστους (ως κόστος υπολογίζεται το μήκος του μονοπατιού).

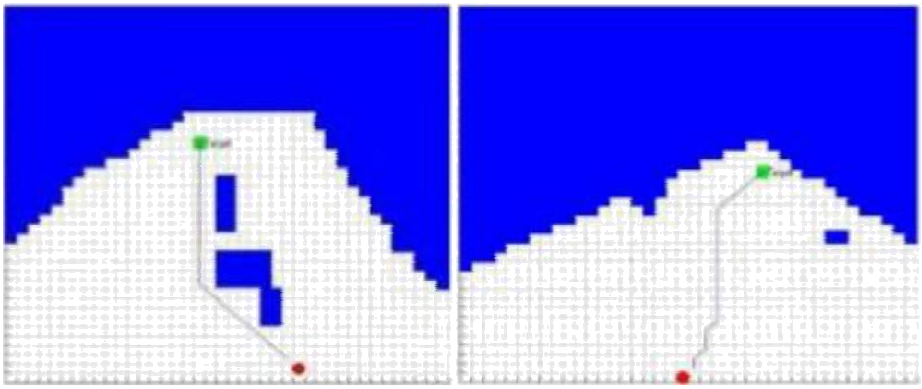
Ο αλγόριθμος A* μας παρέχει την συντομότερη και μικρότερου κόστους διαδρομή αλλά αυτό απαραίτητα δεν σημαίνει ότι είναι και η καταλληλότερη. Θα πρέπει να υπάρχει ένας συσχετισμός μεταξύ της τρέχουσας πορείας με αυτήν της επόμενης κίνησης ώστε να βελτιωθεί το αποτέλεσμα.

Τέτοιου είδους περιπτώσεις φαίνονται στις παρακάτω εικόνες 23 όπου υπάρχουν πολλά εμπόδια στην πορεία προς τον τελικό προορισμό. Ακόμα και εάν η διαδρομή δεν περιέχει εμπόδια στην πορεία προς τον στόχο, είναι αδύνατο να σχεδιαστεί μια ευθεία γραμμή ως λύση λόγο της φύσης λειτουργίας του αλγορίθμου (23.δ, 24.γ). Επομένως οι περιστροφές της ρομποτικής κεφαλής σε κάθε βήμα μπορούν να ληφθούν υπόψη για τον σχεδιασμό μίας καλύτερης διαδρομής.

Υπάρχουν περιπτώσεις στις οποίες ο αλγόριθμος A* δημιουργεί διαδρομές ανάμεσα από δύο εμπόδια χωρίς να υπολογίσει το μέγεθος του ρομπότ, το οποίο μπορεί να είναι αρκετά μεγάλο για να καταφέρει να περάσει ανάμεσα (23.α), (24.β). Αυτό φυσικά μπορεί να είναι καταστροφικό σε περίπτωση επαφής με το εμπόδιο.



α) β)



γ) δ)

Εικόνα 23: Οι παραπάνω εικόνες εμφανίζουν παραδείγματα του αλγορίθμου A* [21]

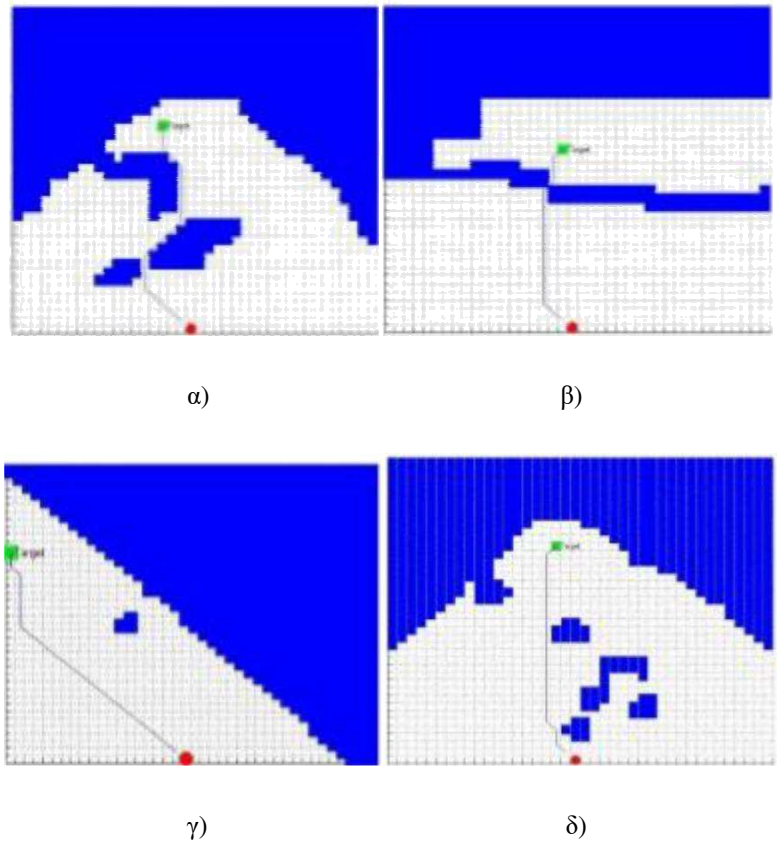
4.6 Τροποποιημένος A* αλγόριθμος

Λόγο των μη ασφαλών διαδρομών που δημιουργεί σε συγκεκριμένες περιπτώσεις ο A* όπως είδαμε προηγουμένως, προτείνεται και υλοποιείται μία τροποποιημένη μορφή του αλγορίθμου αυτού για την επίτευξη καλύτερων διαδρομών. Η κεντρική ιδέα αυτής της τροποποίησης είναι η αλλαγή στον τρόπο με τον οποίο ο αλγόριθμος A* επιλέγει τον επόμενο κόμβο.

Η απλή μορφή του αλγορίθμου A* επιλέγει υποψήφιους κόμβους εξετάζοντας όλους τους κοντινούς κόμβους με δύο κριτήρια. Το πρώτο κριτήριο είναι ότι ο υποψήφιος κόμβος δεν είναι γόνος του τρέχοντος κόμβου και το δεύτερο είναι ότι ο υποψήφιος κόμβος δεν περιέχει κάποιου είδους εμπόδιο όπως φαίνεται στην εικόνα 25. Παρακάτω αναλύονται κάποια προβλήματα και τροποποιήσεις για τον A*.

4.6.1 Επικίνδυνες διαγώνιες κινήσεις και απότομες στροφές

Η νέα τροποποιημένη έκδοση του αλγορίθμου χρησιμοποιεί τα ίδια κριτήρια με αυτά του απλού. Η διαφορά τους είναι ότι πλέον δεν εξετάζεται ο κάθε υποψήφιος κόμβος χωριστά, αλλά εξετάζονται πάνω από ένας ταυτόχρονα ανάλογα με την πιθανή πορεία του ρομπότ. Αυτό εξυπηρετεί στον έλεγχο των επικίνδυνων διαγωνίων κινήσεων όπως φαίνεται στην εικόνα (24.β). Στον πίνακα 6 φαίνεται πως υπολογίζονται οι κατάλληλοι κόμβοι γύρω από τον τρέχων.



α) β)

γ) δ)

Εικόνα 24: Οι παραπάνω εικόνες εμφανίζουν παραδείγματα του αλγορίθμου A* [21]

Εικόνα 25: Με καφέ χρώμα φαίνονται οι υποψήφιοι κόμβοι, ενώ με κίτρινο ο γόνος [21]

Στον πίνακα 6 βλέπουμε την διαδικασία κανόνων που ακολουθεί ο τροποποιημένος αλγόριθμος A* για να ελέγξει ποιος από τους κοντινούς υποψήφιους κόμβους είναι ο καταλληλότερος για την αποφυγή των διαγωνίων επικίνδυνων κινήσεων. Για παράδειγμα για να θεωρηθεί ως υποψήφιος κόμβος, ο κόμβος (1) θα πρέπει να ελεγθεί το κόστος της διαδρομής και επίσης δεν θα πρέπει να είναι γόνος του τρέχοντος κόμβου. Αυτά είναι τα τυπικά κριτήρια του απλού αλγορίθμου. Ακόμα οι κόμβοι (2) και (4) θα πρέπει να μην

περιέχουν κάποιου είδους εμπόδιο για να επιτρέψουν στο ρομπότ να κάνει διαγώνια κίνηση από τον τρέχοντα κόμβο στον κόμβο (1). Το ίδιο γίνεται και για τις υπόλοιπες πιθανές διαγώνιες κινήσεις.

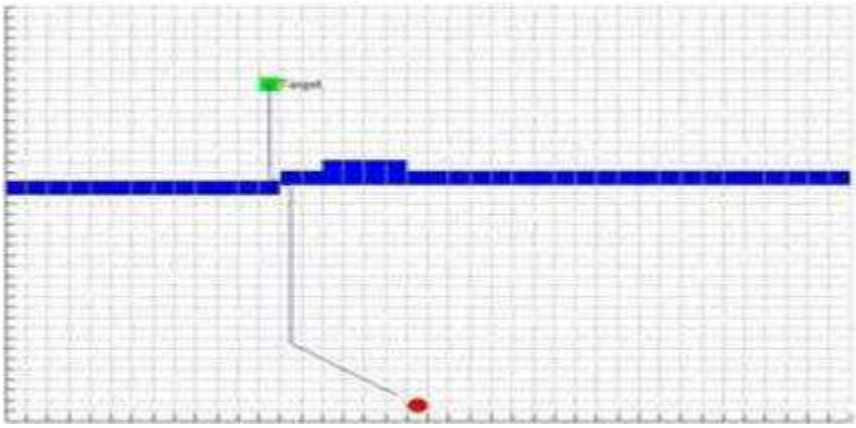
Στην εικόνα 26 βλέπουμε το μονοπάτι που δημιουργεί ο απλός αλγόριθμος A*, ενώ στην εικόνα 27 βλέπουμε την ενέργεια που πραγματοποιεί ο τροποποιημένος αλγόριθμος A* για την αποφυγή οποιασδήποτε επικίνδυνης κίνησης.

Πίνακας 6: Υποψήφιοι κόμβοι

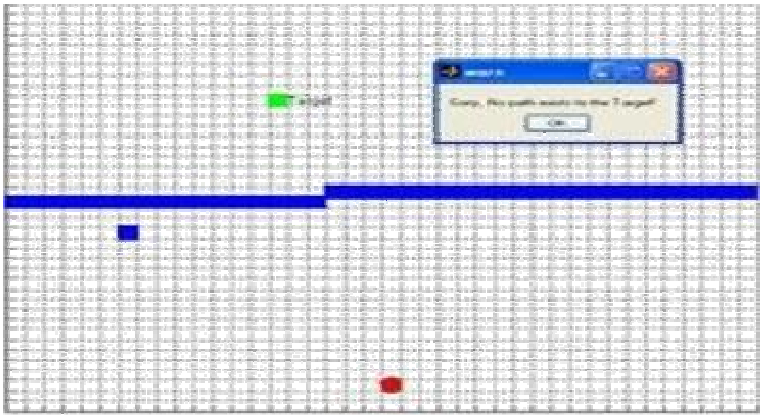
1	2	3
4	X	5
6	7	8

Πίνακας 7: Κόμβοι για έλεγχο διαγωνίων κινήσεων

Υποψήφιος κόμβος	**Κόμβοι για έλεγχο**
1	1,2 & 4
3	3,2 & 5
6	6,4 & 7
8	8,5 & 7



Εικόνα 26: Χαρακτηριστική επικίνδυνη πορεία του απλού αλγορίθμου A* [21]



Εικόνα 27: Αναφορά του τροποποιημένου αλγορίθμου ότι δεν υπάρχει ιδανική διαδρομή [21]

Τα νέα κριτήρια του τροποποιημένου αλγορίθμου A* μας παρέχουν επίσης επιπλέον ασφάλεια αποφεύγοντας τις απότομες στροφές γύρο από εμπόδια με επίκινδυνες γωνίες. Η διαφορά αυτή φαίνεται στις εικόνες 28.α και 28.β.

4.6.2 Υπολογισμός του μεγέθους του ρομπότ

Όπως αναφέρθηκε και παραπάνω ο απλός αλγόριθμος A* δεν υπολογίζει το μέγεθος της ρομποτικής πλατφόρμας κατά την δημιουργία της διαδρομής. Εδώ λοιπόν θα αναλυθεί ότι το μέγεθος πρέπει να λαμβάνεται υπόψη και να χρησιμοποιείται ως παράμετρος όταν επίλεγουμε τους υποψήφιους κόμβους.

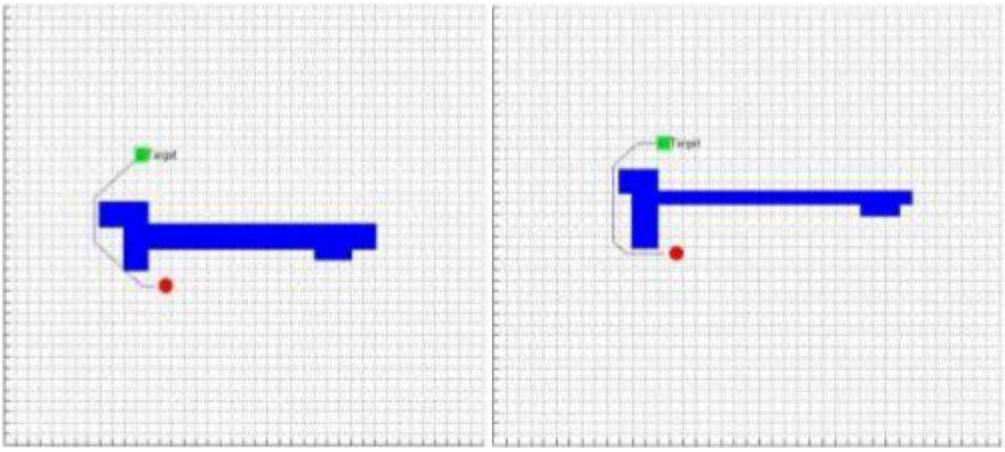
Η κετρνική ιδέα είναι να χρησιμοποιηθεί ένας αριθμός κόμβων αντίστοιχος του μεγέθους του ρομπότ κατά τον έλεγχο κόμβων. Αυτή η ενέργεια εξασφαλίζει ότι κόμβοι έχουν αρκετό χώρο για να επιτρέψουν στο ρομπότ να κινηθεί. Ο πίνακας 8 δείχνει την σειρά κανόνων που χρησιμοιποιούνται για να επιλεγέι ο κόμβος με την προϋπόθεση ότι το μέγεθος του ρομπότ είναι αντίστοιχο με δύο τετράγωνα.

Η εικόνα 24.α δείχνει πως ο απλός αλγόριθμος A* κινείται ανάμεσα από ομάδες εμποδίων για να φτάσει στον προορισμό. Εάν το μέγεθος είναι μεγαλύτερο προφανώς η λύση αυτή είναι ακατάλληλη. Η εικόνα 29 δέιχνει την λύση χρησιμοποιώντας τροποποιημένο αλγόριθμο A* και ρομπότ μεγέθους δύο τετραγώνων.

Η εικόνα 29.α δείχνει ότι υπάρχει ένα κενό ανάμεσα από τα εμπόδια για να κινηθεί το ρομπότ, αλλά ο αλγόριθμος υπολογίζει το μέγεθος του ρομπότ και επειδή το κενό δεν είναι αρκετά μεγάλο δεν επιτρέπει την κίνησή του.

Στην εικόνα 29.β φαίνεται το ίδιο πρόβλημα αλλά στην συγκεκριμένη περίπτωση υπάρχει κενό δύο τετραγώνων, το οποίο είναι αρκετό για να σχεδιαστεί η διαδρομή προς τον προορισμό. Στην εικόνα 29.γ υπάρχει ελεύθερη ευθεία που συνδέει το ρομπότ με τον προορισμό, αλλά το πλάτος είναι μόνο ενός τετραγώνου οπότε δεν είναι αρκετό για το

συγκεκριμένο μέγεθος του ρομπότ. Για αυτόν το λόγο σχεδιάζεται νέα πορεία πιο κατάλληλη. Στην εικόνα 29.δ φαίνεται ένα πιο περιέργο σενάριο, στο οποίο ο αλγόριθμος μπορεί και φτάνει στον προορισμό υπολογίζοντας σωστά και το μέγεθος του ρομπότ.

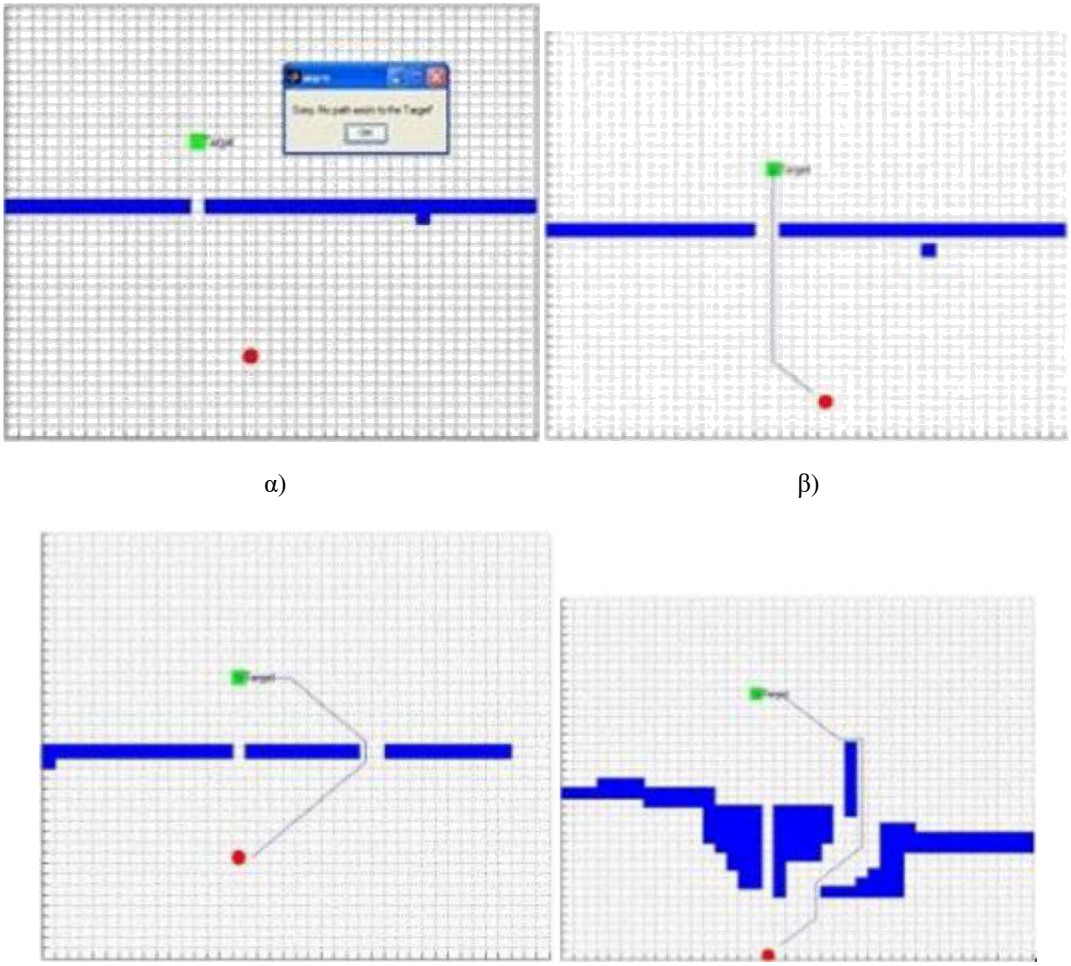


α) β)

Εικόνα 28: Αποφυγή επικίνδυνων γωνιών από τον τροποποιημένο A* [21]

Πίνακας 8: Κόμβοι για έλεγχο με ρομπότ μεγέθους δύο τετραγώνων

Υποψήφιος κόμβος	**Κόμβοι για έλεγχο**
1	(1,2 & 4) χωρίς εμπόδια
2	(2 & 3) ή (2& 1) χωρίς
3	(3,2 & 5) χωρίς
4	(1 & 4) ή (4 & 6) χωρίς
5	(5 & 3) ή (5 & 8) χωρίς
6	(4,6 & 7) χωρίς
7	(7 & 6) ή (7 & 8) χωρίς
8	(8,5 & 7) χωρίς



α) β)

γ) δ)

Εικόνα 29: Παραδείγματα A* αλγορίθμου στα οποία το μέγεθος του ρομπότ ισοδυναμεί με δύο τετράγωνα [21]

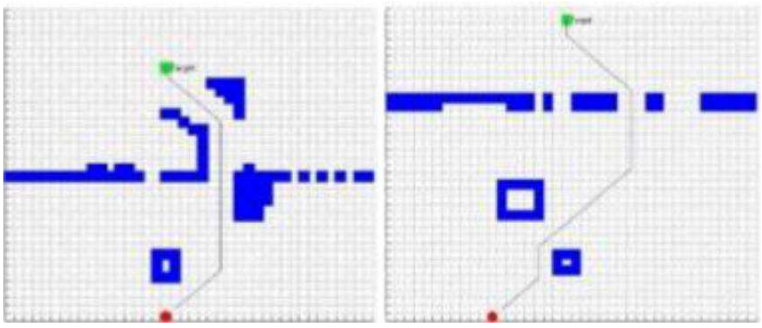
Είναι προφανές ότι ο τροποποιημένος A* αλγόριθμος είναι πιο ασφαλής από τον απλό. Η ίδια μεθοδολογία θα μπορούσε να χρησιμοποιηθεί για ποικίλα μεγέθη. Ο πίνακας 9 δείχνει την σειρά κανόνων που ισχύουν στην περίπτωση που το ρομπότ έχει μέγεθος τριών τετραγώνων. Στην πίνακα 10 βλέπουμε πως επιλέγονται οι κοντινοί κόμβοι και στην εικόνα 30 τα αποτελέσματα της επιλογής.

Πίνακας 9: Έλεγχος κόμβων

Υποψήφιος κόμβος	**Κόμβοι για έλεγχο**
1	[(4,2,D) ή (4,2,J)] & (H,I,C) & (F,1,2)
2	(1,2,3)
3	[(2,5,K) ή (2,5,B)] & (G,3,2) & (C,3,I)
4	(I,4,6) ή (1,4,B) ή (4,6,M)
5	(3,5,8) ή (5,8,O) ή (5,3,D)
6	[(4,7,O) ή (4,7,F)] & (H,6,N) & (J,6,7)
7	(6,7,8)
8	[(5,7,M) ή (5,7,G)] & (I,8N) & (7,8,K)

Πίνακας 10: Επιλογή κόμβων

A	B	C	D	E
F	1	2	3	G
H	4	X	5	I
J	6	7	8	K
L	M	N	O	P



α) β)

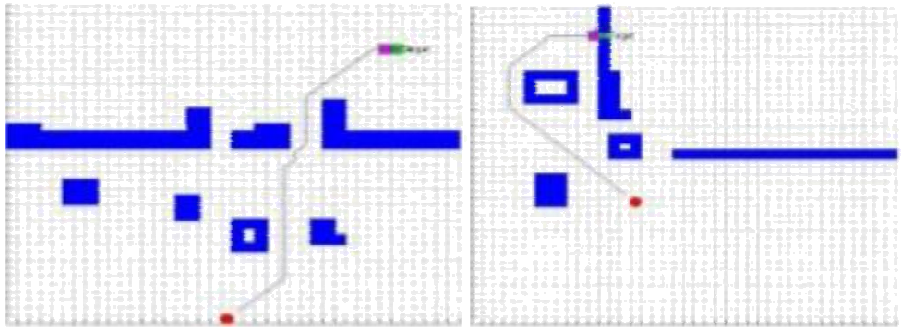
Εικόνα 30: Παραδείγματα με μέγεθος 3 τετραγώνων [21]

4.6.3 Ορισμός κατεύθυνσης του ρομπότ

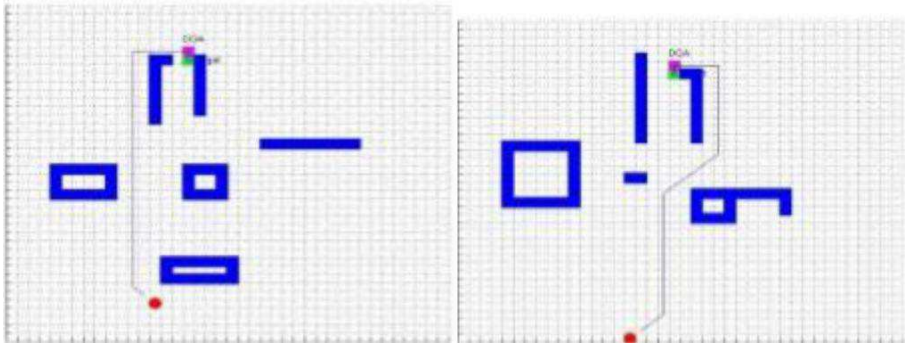
Σε ορισμένες εφαρμογές το ρομπότ πρέπει να φτάσει στον προορισμό με συγκεκριμένη γωνία ή κατεύθυνση. Υπάρχουν διάφορες περιπτώσεις που το καθιστούν αναγκαίο. Ένα τέτοιο παράδειγμα είναι η φόρτιση της μπαταρίας του ρομπότ, θα πρέπει να συνδεθεί αυτόματα στην απαιτούμενη έξοδο, οπότε το ρομπότ θα πρέπει να είναι στραμμένο στην σωστή κατεύθυνση. Προσθέτοντας αυτήν την παράμετρο στον τροποποιημένο A*, ο νέος αλγόριθμος θεωρεί τον γείτονα κόμβο δίπλα στον κόμβο προορισμού (υπολογίζοντας και την απαραίτητη γωνία) ως τον αρχικό προορισμό και μετά ολοκληρώνει την πορεία του προς τον στόχο. Στην εικόνα 31 φαίνονται τα αποτελέσματα με την εφαρμογή της νέας παραμέτρου, το τετράγωνο με το όνομα DOA (direction of arrival) είναι ο κόμβος που ορίζει την τελική κατεύθυνση του ρομπότ στον στόχο.

4.7 Σύγκριση απλού με τροποποιημένο αλγόριθμο A*

Στις παραπάνω παραγράφους, υλοποιήσαμε, δοκιμάσαμε και αξιολογήσαμε τον απλό αλγόριθμο A*. Παρατηρήσαμε κάποια μειονεκτήματα κατά την δημιουργία διαδρομών για την κίνηση ρομποτικών πλατφορμών. Για τον λόγο αυτό προτείνεται μία τροποποιημένη έκδοση του A* αλγορίθμου ώστε να λυθούν να εξαλειφθούν αυτά τα μειονεκτήματα. Στον πίνακα 11 βλέπουμε μία σύγκριση μεταξύ αυτών των αλγορίθμων.



α) β)



γ) δ)

Εικόνα 31: Παραδείγματα με την παράμετρο DOA. Το μέγεθος παραμένει 3 τετράγωνα [21]

Πίνακας 11: Σύγκριση μεταξύ των δύο αλγορίθμων

	Απλός A*	**Τροποποιημένος A***
Επικίνδυνες Διαγώνιες Κινήσεις	Το επιτρέπει	Δεν το επιτρέπει
Γωνιακές στροφές γύρο από γωνίες	Το επιτρέπει	Δεν το επιτρέπει
Μέγεθος του ρομπότ	Δεν υπολογίζεται	Χρησιμοποιείται ως παράμετρος
Τελική διαδρομή	Δεν είναι η ιδανικότερη	Δεν είναι η ιδανικότερη
DOA	Δεν υπολογίζεται	Χρησιμοποιείται ως παράμετρος

4.8 Συμπέρασμα

Σε ένα δυναμικό περιβάλλον, ο υπολογιστικός χρόνος είναι κρίσιμος. Ο υπολογιστικός χρόνος του A* ποικίλει όσο ποικίλει και ο εναέριος χώρος. Η περιπλοκότητα του A* μπορεί να μειωθεί σχεδιάζοντας μία καλή ευρετική λειτουργία έτσι ώστε ο A* να έχει μεγαλύτερη ευελιξία να προσαρμόσει διάφορα σενάρια. Το πλεονέκτημα του A* είναι ότι μπορεί να είναι ευέλικτος επιτρέποντας σε διαφορετικές ευρετικές λειτουργίες να χειριστούν διαφορετικές καταστάσεις. Ο A* είναι ένας καλός υποψήφιος για την εκτέλεση διαδρομής σχεδιασμένη σε πραγματικό χρόνο.

ΚΕΦΑΛΑΙΟ 5

PARTICLE FILTERS
(ΦΙΛΤΡΑ ΣΩΜΑΤΙΔΙΩΝ)

5.1 Γενικά για τα Particle Filters

Το Particle Filtering αποτελεί μία υπό βέλτιστη λύση. Για συστήματα δηλαδή που δεν είναι γραμμικά και δε μπορούμε να πάρουμε μία βέλτιστη λύση ελαχιστοποίησης του σφάλματος, όπως αυτή που δίνει το απλό φίλτρο Kalman, τα φίλτρα αυτά προσεγγίζουν τη βέλτιστη Bayesian λύση. Η έρευνα για τη μέθοδο αυτή έχει προχωρήσει αρκετά τα τελευταία κυρίως χρόνια και όλα τα ερευνητικά αποτελέσματα έχουν αποδείξει ότι αποτελούν μία πολύ ικανοποιητική λύση για εφαρμογές πραγματικού χρόνου σε συστήματα όπου οι αυστηρές παραδοχές για Gaussian μοντέλα θορύβου και γραμμικά μοντέλα κίνησης δεν μπορούν να εφαρμοστούν [11]. Το κύριο πλεονέκτημα του Particle Filtering είναι η δυνατότητα χειρισμού οποιασδήποτε μη γραμμικότητας και οποιασδήποτε κατανομής θορύβου. Όσο πιο μη γραμμικό είναι το μοντέλο ή όσο πιο μη-Gaussian είναι ο θόρυβος, τόσο μεγαλύτερες δυνατότητες έχει το Particle Filtering.

Όπως θα γίνει σαφές και παρακάτω το Particle Filtering βασίζεται στη διαδοχική εκτέλεση Monte Carlo εκτιμήσεων. Η βασική ιδέα της μεθόδου είναι η χρήση δειγμάτων (particles) για την αναπαράσταση του μοντέλου του συστήματος αντί για χρήση Gaussian κατανομών ή οποιουδήποτε άλλου μοντέλου και η διάδοση μόνο των επικρατέστερων δειγμάτων.

Εκτός από τη παρακολούθηση στόχου (Target Tracking) η μέθοδος του particle filtering βρίσκει εφαρμογή σε ποικίλες άλλες ερευνητικές περιοχές, όπως είναι οι: Εκτίμηση θέσης, Πλοήγηση, Εντοπισμός θέσης κινούμενου ρομπότ (mobile robot localization κα.). (Γεωργιάδου 2009)

5.2 Αλγόριθμος SIS

Ο αλγόριθμος SIS είναι ο βασικός αλγόριθμος του Particle Filtering. Το ακρωνύμιο SIS-Sequential Importance Sampling μεταφράζεται στα ελληνικά σαν δειγματοληψία ακολουθιακής σημασίας και αποτελεί μία μέθοδο Monte Carlo (MC ή Sequential MC). Η βασική ιδέα του αλγορίθμου είναι η αναπαράσταση της απαιτούμενης συνάρτησης πυκνότητας πιθανότητας μέσω τυχαίων δειγμάτων με τα κατάλληλα βάρη και η πραγματοποίηση εκτιμήσεων βάσει αυτών των δειγμάτων. Καθώς ο αριθμός αυτών των δειγμάτων μεγαλώνει, αυτή η μέθοδος γίνεται ισοδύναμη με την συναρτησιακή περιγραφή της <<εκ των υστέρων>> πυκνότητα και το SIS φίλτρο προσεγγίζει τη βέλτιστη Bayesian λύση.

Έστω $X_k=\{x_j,j=0, ..., k\}$ όλες οι καταστάσεις του στόχου έως τη χρονική στιγμή k. Και ακόμα $Z_k=\{z_j=0, ..., k\}$ όλες οι μετρήσεις που είναι διαθέσιμες μέχρι τη χρονική στιγμή k. Η από κοινού <<εκ των υστέρων>> πυκνότητα έως τη χρονική στιγμή k είναι η $p(X_k|Z_k)$ και η

$p(X_k|Z_k)$ είναι η οριακή της τιμή. Θεωρούμε μια τυχαία μέτρηση όπου είναι ένα σύνολο σημείων με αντίστοιχα βάρη . Τα βάρη θεωρούνται κανονικοποιημένα, δηλαδή . Τότε η $p(X_k|Z_k)$ την στιγμή k μπορεί να προσεγγιστεί ως:

Αυτή είναι μια προσέγγιση διακριτών βαρών. Σκοπός μας τώρα είναι να εξάγουμε την σχέση ενημέρωσης των βαρών. Η επιλογή των βαρών w γίνεται με βάση την αρχή δειγματοληψίας σημαντικότητας (importance sampling). Εδώ πρέπει να γίνει μια μικρή αναφορά στην αρχή αυτή.

Έστω ότι έχουμε μία συνάρτηση πυκνότητας πιθανότητας p(x) από την οποία είναι δύσκολο να πάρουμε δείγματα (μόρια) αλλά μπορούμε να την προσεγγίσουμε με μία άλλη παρόμοια συνάρτηση πυκνότητας πιθανότητας q(x) και να πάρουμε δείγματα από εκεί κατά αντιστοιχία. Ονομάζουμε την q(x) συνάρτηση σημαντικότητας .

Για να πάρουμε μία Monte Carlo εκτίμηση του ολοκληρώματος :

Επιλέγουμε N>>1 δείγματα που κατανέμονται σύμφωνα με την συνάρτηση πυκνότητας πιθανότητας q(x) και φτιάχνουμε το άθροισμα:

είναι τα βάρη σημαντικότητας.

Κάνοντας λοιπόν εφαρμογή αυτών στη σχέση (5.1) θεωρούμε ότι τα δείγματα προέκυπταν από μια πυκνότητα q(x) τότε σύμφωνα με την (5.3):

Θεωρώντας τώρα ότι τη χρονική στιγμή k-1 έχουμε δείγματα που αποτελούν μια προσέγγιση της $p(X_{k-1}|Z_{k-1})$ μετά τη λήψη της μέτρησης z_k την στιγμή k, επιθυμούμε να προσεγγίσουμε την $p(X_k|Z_k)$. Αν η πυκνότητα σημαντικότητας q(x) επιλέγεται έτσι ώστε:

τότε μπορούμε να δημιουργήσουμε τα νέα δείγματα μεταβάλλοντας κάθε ένα απόταυπάρχοντα δείγματα με τηνέακατάσταση . Για να εξάγουμε την σχέση ενημέρωσης των βαρών, η συνάρτηση πυκνότητας πιθανότητας πρέπει πρώτα να εκφραστεί ως:

Αντικαθιστώντας την (5.6) και την (5.5) στην (5.4), η εξίσωση ενημέρωσης των βαρών μπορεί να εκφραστεί ως εξής:

Επιπλέον, αν ισχύει $q(x_k|X_{k-1}, Z_k)=q(x_{k-1},z_k)$, τότε η πυκνότητα σημαντικότητας εξαρτάται μόνο από τα x_{k-1} και z_k.

Έτσι για το φίλτρο δε χρειάζεται να αποθηκεύσουμε τα αλλά μόνο το και ιστορικό των μετρήσεων. Έτσι τα νέα βάρη δίνονται από τη σχέση:

και η a posteriori συνάρτηση πυκνότητας πιθανότητας προσεγγίζεται από την:

όπου τα βάρη ορίζονται στην (5.7). Για N τείνει στο άπειρο αποδεικνύεται ότι η προσέγγιση (5.8) πλησιάζει στην πραγματική.

Συνοψίζοντας ο αλγόριθμος SIS αποτελεί μια αναδρομική διάδοση των βαρών και των σημείων καθώς κάθε νέα μέτρηση φτάνει στο σύστημα. (Γεωργιάδου 2009)

5.3 Πρόβλημα Εκφυλισμού και Επαναδειγματοληψία

Ένα μειονέκτημα του αλγορίθμου SIS είναι το φαινόμενο εκφυλισμού. Πρακτικά αυτό σημαίνει ότι μετά από ένα συγκεκριμένο αριθμό επαναλήψεων, όλα τα μόρια (particles), εκτός από ένα, έχουν αμελητέο βάρος (πρακτικά μηδενικό). Αυτός ο εκφυλισμός είναι αδύνατο να ξεπεραστεί από τον αλγόριθμο SIS. Ένα κατάλληλο μέγεθος μέτρησης του εκφυλισμού είναι το Neff: αριθμός των αποτελεσματικών δειγμάτων που ορίζεται ως εξής:

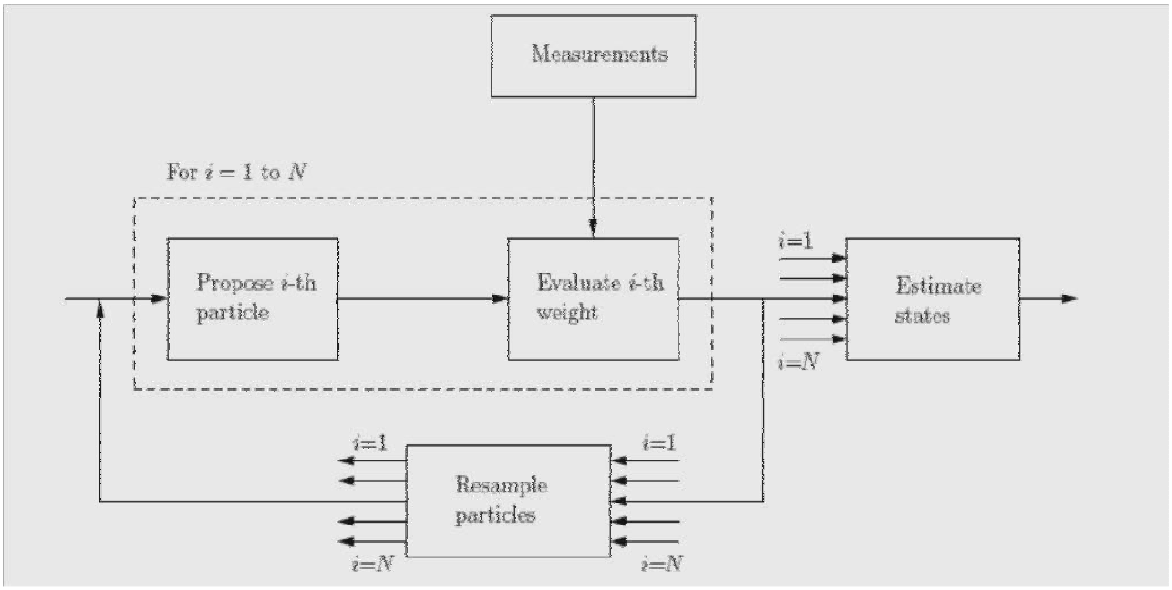
$$\frac{\qquad\qquad}{}$$

Συνεπώς, μικρός αριθμός Neff σημαίνει ισχυρός εκφυλισμός και αντίστροφα. Για την αντιμετώπιση του προβλήματος αυτού χρησιμοποιείται η μέθοδος της επαναδειγματοληψίας (resampling). Σε κάθε σάρωση δηλαδή υπολογίζεται το Neff και όταν βρίσκεται μικρότερο από κάποιο κατάλληλα επιλεγμένο κατώφλι γίνεται resampling στον αλγόριθμος SIS. Η βασική ιδέα έγκειται στην εξάλειψη των δειγμάτων με αμελητέο βάρος και στην αύξηση εκείνων με σημαντικά βάρη.

Το νέο σετ των τυχαίων δειγμάτων παράγεται με την δειγματοληψία (με αντικατάσταση) N φορές από μια προσεγγιστική διακριτή αναπαράσταση της $p(x_k|Z_k)$ που δίνεται από την:

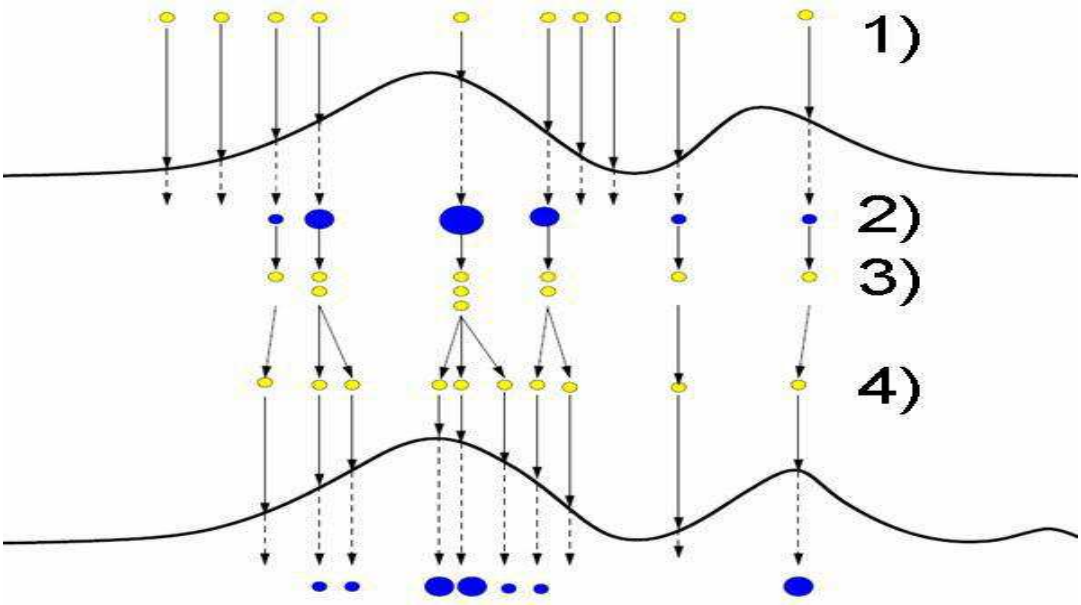
έτσι ώστε και τα βάρη ουσιαστικά επαναρυθμίζονται στη τιμή 1/N.

Στο παρακάτω σχήμα φαίνεται ένα block διάγραμμα του αλγορίθμου SIS με resampling.



Εικόνα 32: Block διάγραμμα Particle Filter [12]

Για να γίνουν πιο κατανοητά τα όσα έχουν αναφερθεί μέχρι στιγμής παρουσιάζεται ένα παράδειγμα εφαρμογής του αλγορίθμου με δειγματοληψία:



Εικόνα 33: Βασική ιδέα Particle Filter (Γεωργιάδου 2009)

- Αρχικοποίηση: Στο πρώτο βήμα επιλέγονται N δείγματα ισοβαρή με βάρος (1/N)
- Αναπροσαρμογή βαρών: Στο δεύτερο βήμα ορίζονται τα βάρη για τα δείγματα αναλογικά με την συνάρτηση πυκνότητας πιθανότητας που έχει προκύψει από τη μοντελοποίηση του συστήματος και τις μετρήσεις.
- Επαναδειγματοληψία: Υπολογίζεται η τιμή Neff και αν είναι απαραίτητο στο τρίτο βήμα εκτελείται επαναδειγματοληψία επαναλαμβάνοντας τα δείγματα με μεγάλα βάρη περισσότερες φορές και αγνοώντας τα δείγματα με μικρά βάρη.
- Τέλος: Στο τέταρτο βήμα κανονικοποιούνται τα βάρη που προέκυψαν και επιλέγονται τελικά τα καινούρια δείγματα και ξαναρχίζει η διαδικασία αναδρομικά.

Παρότι, όμως, η επαναδειγματοληψία περιορίζει δραστικά τα αποτελέσματα του εκφυλισμού, εισάγει άλλα πρακτικά προβλήματα. Το σημαντικότερο είναι ότι τα μόρια με μεγάλα βάρη επιλέγονται στατιστικά πολλές φορές. Αυτό οδηγεί σε απώλεια της ποικιλίας μεταξύ των μορίων καθώς το δείγμα που προκύπτει θα περιέχει πολλά επαναλαμβανόμενα σημεία. Αυτό το πρόβλημα, που είναι γνωστό ως εξασθένιση των μορίων, είναι πολύ σοβαρό στην περίπτωση όπου ο θόρυβος διαδικασίας είναι μικρός. Οδηγεί σε μια κατάσταση όπου όλα τα μόριά θα καταλήξουν σε ένα και μόνο στοιχείο μετά από λίγες επαναλήψεις. Μια λύση αυτού του προβλήματος είναι η διαδικασία της κανονικοποίησης (regularization). (Γεωργιάδου 2009)

5.4 Αλγόριθμος SIR

Το φίλτρο αυτό προτάθηκε αρχικά με το όνομα <<bootstrap>>. Ο αλγόριθμος SIR (sampling importance resampling) προκύπτει από τον SIS, διαλέγοντας την πυκνότητα σημαντικότητας να είναι η <<εκ των προτέρων>> συνάρτηση μετάβασης και πραγματοποιώντας το βήμα επαναδειγματοληψίας σε κάθε χρονικό βήμα (σάρωση). Λόγω της παραπάνω επιλογής η 5.10 γίνεται:

Επίσης εφόσον εφαρμόζεται δειγματοληψία κάθε φορά, θα ισχύει $\forall$.

Άρα η σχέση 5.11 γίνεται:

Επειδή η πυκνότητα σημαντικότητας για αυτό το φίλτρο είναι ανεξάρτητη της μέτρησης z_k, η εκτίμηση κατάστασης πραγματοποιείται χωρίς τη γνώση των παρατηρήσεων. Για αυτό το

φίλτρο μπορεί να είναι ιδιαιτέρως ευαίσθητο σε εξωγενείς παράγοντες. Επιπλέον, καθώς η επαναδειγματοληψία εφαρμόζεται σε κάθε σάρωση, μπορούμε να οδηγηθούμε σε απώλεια της ανομοιότητας (ποικιλίας) των μορίων πολύ γρήγορα. Παρόλα αυτά, αυτή η μέθοδος έχει το πλεονέκτημα ότι τα βάρη μπορούν εύκολα να υπολογιστούν βάσει της 5.12 και η πυκνότητα σημαντικότητας μπορεί εύκολα να δειγματοληπτηθεί. (Γεωργιάδου 2009)

5.5 Αλγόριθμος auxiliary SIR (ASIR)

Το βοηθητικό SIR αποτελεί μια παραλλαγή του φίλτρου SIR. Η βασική ιδέα είναι να πραγματοποιηθεί το βήμα της επαναδειγματοληψίας την χρονική στιγμή k-1 (χρησιμοποιώντας τη μέτρηση z_k) προτού αναπαραχθούν τα μόρια την χρονική στιγμή k. Εισάγεται η χρήση μιας νέας πυκνότητας σημαντικότητας $q(x_k,i|Z_k)$ η οποία δειγματοληπτεί το ζεύγος όπου το i^j αναφέρεται στο δείκτη του μορίου την στιγμή k-1.
Εφαρμόζοντας τον κανόνα του Bayes προκύπτει:

Το φίλτρο ASIR λειτουργεί βγάζοντας ένα δείγμα από την από κοινού πυκνότητα $p(x_k|i,Z_k)$ και εν συνεχεία απαλείφοντας τους δείκτες i στο ζεύγος (x_k,i) για να παραχθεί ένα δείγμα από την οριακή πυκνότητα $p(x_k|Z_k)$. Η πυκνότητα σημαντικότητας που χρησιμοποιείται για να εξάγουμε ένα δείγμα πρέπει να ικανοποιεί την αναλογικότητα:

όπου το αποτελεί ένα χαρακτηριστικό του x_k με δεδομένο το . Πρακτικά θα μπορούσε να είναι ο μέσος όρος του.

Γράφοντας:

Προκύπτει από την (5.14):

Τα βάρη που ανατίθενται στα δείγματα θα δίνονται λοιπόν από τη σχέση:

Το βασικό πλεονέκτημα του ASIR είναι ότι παράγει σημεία από το δείγμα την στιγμή k-1, τα οποία, διαμορφωμένα από την τρέχουσα μέτρηση, είναι πιο πιθανό να παράγουν ακριβή αποτελέσματα. Εάν ο θόρυβος διαδικασίας είναι μικρός, έτσι ώστε η να χαρακτηρίζεται επαρκώς από το , τότε το φίλτρο είναι λιγότερο ευαίσθητο σε εξωγενείς παράγοντες και τα βάρη είναι πιο ομαλά. Αν όμως ο θόρυβος διαδικασίας είναι μεγάλος, ένα και μόνο σημείο δεν μπορεί να χαρακτηρίσει επαρκώς την , και άρα το φίλτρο επαναδειγματοληπτεί βασισμένο σε μια <<κακή>> προσέγγιση της . Σε τέτοιες περιπτώσεις, η χρήση του ASIR μπορεί να είναι προβληματική. (Γεωργιάδου 2009)

5.6 PF με βελτιωμένη ανομοιότητα δειγμάτων (Regularized PF)

Όπως αναφέρθηκε και παραπάνω το πρόβλημα της απώλειας ποικιλίας μεταξύ των μορίων που επιφέρει το resampling είναι δυνατόν να επιλυθεί με κανονικοποίηση. Αυτό το πρόβλημα επιλύει ο αλγόριθμος RPF εισάγοντας ένα επιπλέον βήμα κανονικοποίησης που έχει σαν αποτέλεσμα την βελτίωση της ανομοιότητας μεταξύ των μορίων.

Το RPF κάνει επαναδειγματοληψία από μια συνεχή προσέγγιση της $p(x_k|Z_k)$. Πιο συγκεκριμένα, στο RPF, τα δείγματα παράγονται από την προσέγγιση:

όπου

είναι η πυκνότητα φλοιού , το n_x είναι η διάσταση του διανύσματος κατάστασης και το είναι τα κανονικοποιημένα βάρη. Η πυκνότητα φλοιού είναι μια συμμετρική συνάρτηση πυκνότητας πιθανότητας, τέτοια ώστε να ισχύει:

Το K και το h επιλέγονται με τέτοιο τρόπο ώστε να ελαχιστοποιείται το μέσο τετραγωνικό σφάλμα ολοκλήρωσης (MISE) μεταξύ της πραγματικής και της που προκύπτει από την 5.18. Αξίζει, εδώ, να σημειωθεί ότι αυτή η προσέγγιση γίνεται κατά πολύ λιγότερο κατάλληλη καθώς το αυξάνει. Στην ειδική περίπτωση ενός δείγματος με ίσα βάρη, η βέλτιστη επιλογή φλοιού είναι ο φλοιός Epanechikov:

Επιπλέον, αν η πυκνότητα είναι γκαουσσιανή, τότε η βέλτιστη λύση για το εύρος είναι :

(Γεωργιάδου 2009)

5.6.1 Ψευδοκώδικας

Παρακάτω βλέπουμε μια πιο αναλυτική περιγραφή του αλγόριθμου, σε ψευδοκώδικα.
Αλγόριθμος Particle Filter

for m = 1 to M do

πάρε δειγματοληπτικά

endfor

for m = 1 to M do

πάρε το I από το με πιθανότητα

πρόσθεσε το στο

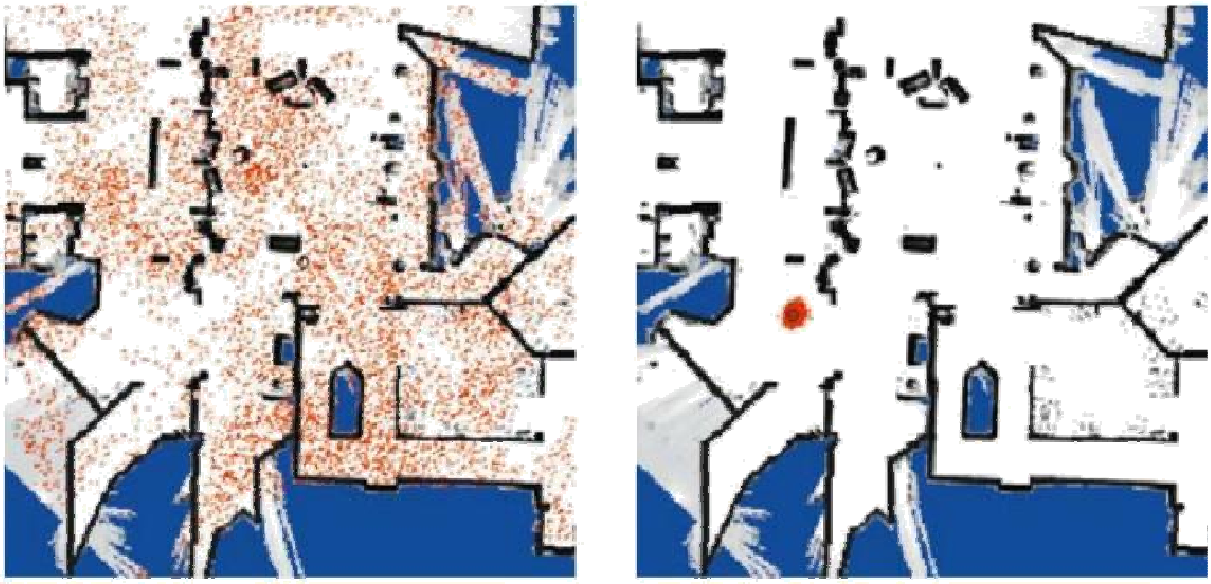
endfor

return

Όπως βλέπουμε παραπάνω, η είσοδος στον αλγόριθμο είναι το στιγμιότυπο του particle μια χρονική στιγμή πριν, μαζί με τον τρέχοντα έλεγχο u_t και την πιο πρόσφατη παρατήρηση z_t. Στη συνέχεια ο αλγόριθμος κατασκευάζει ένα πρωταρχικό σύνολο , που αναπαριστά την τιμή $bel(x_t)$.Το πρωταρχικό σύνολο αυτό είναι ίσο το σύνολο των particles (που έχουμε από την προηγούμενη χρονική στιγμή) και σε αυτό προσαρτώνται οι νέες καταστάσεις· μια νέα πιθανή θέση για κάθε particle. Συγκεκριμένα, στη γραμμή 5 έχουμε τη νέα θέση, που προκύπτει από το μοντέλο κίνησης του ρομπότ. Αυτό παίρνει το δείκτη m για να δηλώνει ότι προέκυψε από το m particle και από την κατάσταση X_{t-1}. Σε αυτή τη φάση επιλέγουμε από την κατανομή $p(x_t|u_t, x_{t-1}^{[m]})$. Στη γραμμή 6 υπολογίζεται η τιμή του βάρους w, που προκύπτει από το μοντέλο παρατήρησης. Στις γραμμές 9-11 βλέπουμε την ενέργεια, που κάνει τη μέθοδο του φίλτρου σωματιδίων να ξεχωρίζει και να υπερτερεί σε σύγκριση με άλλους αλγόριθμους. Στις γραμμές αυτές πραγματοποιείται η επαναδειγματοληψία με βάση τα βάρη. Ο αλγόριθμος παίρνει με αντικατάσταση τα M particles από το πρωταρχικό σύνολο . Η πιθανότητα να πάρουμε ένα particle είναι ανάλογη της τιμής του βάρους, όπως έχει διαμορφωθεί για το συγκεκριμένο particle. Ύστερα από την επαναδειγματοληψία θα προκύψει ένα σύνολο από particles ιδίου μεγέθους με το αρχικό, που όμως θα περιέχει κυρίως τα particles που πλησιάζουν περισσότερο στην επιθυμητή-πραγματική τιμή. Αυτό σημαίνει ότι στο νέο σύνολο θα υπάρχουν πολλά διπλά από τα πιο πιθανά particle, θα υπάρχουν όμως και κάποια λιγότερο πιθανά ώστε να έχουμε μια κατανομή, από όλες τις πιθανές τιμές, κάνοντάς το σύνολό μας πιο αντιπροσωπευτικό και ανεξάρτητο από τις εκάστοτε συνθήκες.

Το βήμα της επαναδειγματοληψίας αναγκάζει τα particles να κατανεμηθούν σύμφωνα με την εκ των υστέρων κατανομή πιθανότητας $bel(x_t)$. Μια διαφορετική έκδοση του αλγόριθμου θα μπορούσε να μη χρησιμοποιεί επαναδειγματοληψίας. Ο αλγόριθμος αυτός συγκλίνει δυσκολότερα στην επιθυμητή κατάσταση και αυτό κυρίως για μεγάλο αριθμό particles. Η μέθοδος της δειγματοληψίας είναι συνέπεια της θεωρίας του Δαρβίνου περί επικράτησης του ισχυρότερου: υπερτερούν τα καλύτερα particle. Έτσι, επικεντρώνεται στις περιοχές του χώρου κατάστασης που είναι πιο σημαντικές.

Σημαντικό ρόλο για τη διεξαγωγή του αλγόριθμου παίζουν το Μοντέλο Κίνησης και το Μοντέλο Αντίληψης.



Εικόνα 34: Παράδειγμα εφαρμογής του φίλτρου particle, για τον προσδιορισμό της θέσης του ρομπότ (Παπανικολάου 2008)

Παραπάνω βλέπουμε δύο παραδείγματα του φίλτρου particle για τον προσδιορισμό της θέσης του ρομπότ.

I. Αρχικά, έχουμε ενσωματώσει στο φίλτρο λίγες πληροφορίες και η αβεβαιότητα είναι μεγάλη
II. Αφού, έχουν ενσωματωθεί πολλές πληροφορίες το φίλτρο συγκλίνει σε μία μοναδική εκ των υστέρων πιθανότητα. (Παπανικολάου 2008)

5.7 Κριτήριο Επαναδειγματοληψίας

Το ζήτημα της συχνότητας δειγματοληψίας είναι κρίσιμο, διότι σκοπός του φίλτρου σωματιδίων είναι να υπάρχει μια ποικιλομορφία στα χαρακτηριστικά τους, η οποία στο τέλος της διαδρομής θα μας βοηθήσει να προσδιορίσουμε την πραγματική θέση του ρομπότ, χωρίς μεγάλο σφάλμα (δηλαδή θα πρέπει να τείνει στην πραγματική).

Έτσι, για να έχουμε καλή απόδοση (ποικιλομορφία και στο τέλος η λύση να τείνει στην πραγματική) εισάγουμε το μέγεθος:

$$\_\_\_\_\_$$

Η ιδέα για το Neff έχει ως εξής: Αν τα particles είχαν ληφθεί από την πραγματική κατανομή, τότε τα βάρη τους θα ήταν όλα ίσα. Όσο περισσότερο αξίζει η προσέγγιση/εκτίμηση, τόσο υψηλότερη είναι η ποικιλότητα στα particles. Από τη στιγμή που

το Neff μπορεί να θεωρηθεί ως μέτρο της διασποράς των βαρών, θα είναι ένα χρήσιμο μέτρο για να αποτιμήσει πόσο καλά το σύνολο των particles πλησιάζει στην πραγματική κατανομή.

5.8 Διαδικασία Επαναδειγματοληψίας

Για να μειώσουμε το σφάλμα δειγματοληψίας, δειγματοληπτούμε με τη μέθοδο <<low-variance sampling>>. Παρακάτω βλέπουμε αναλυτικά τον αλγόριθμο. Η βασική ιδέα είναι ότι αντί να επιλέγουμε ανεξάρτητα μμεταξύ τους στη διαδικασία επαναδειγματοληψίας, η επιλογή θα περιλαμβάνει μια ακολουθιακή, στοχαστική διαδικασία.

Αντί να διαλέγουμε M τυχαίους αριθμούς και να επιλέγουμε τα αντίστοιχα particles, ο αλγόριθμος υπολογίζει μόνο ένα τυχαίο αριθμό και επιλέγει δείγματα με βάση αυτόν τον αριθμό, αλλά με μια πιθανότητα ανάλογη στο βάρος του δείγματος. Έτσι, επιλέγουμε έναν τυχαίο αριθμό στο διάστημα [0,M-1], όπου M είναι ο αριθμός των particles. Ύστερα επιλέγουμε particle αθροίζοντας επανειλημμένα το σταθερό αριθμό M-1 στο r και επιλέγουμε το particle που αντιστοιχεί σε αυτόν τον αριθμό. Κάθε αριθμός U στο διάστημα [0,1] δείχνει σε ένα μόνο particle, που είναι το particle i για το οποίο ισχύει:

Το πλεονέκτημα του αλγορίθμου low-variance sampler είναι τριπλό. Αρχικά καλύπτει το διάστημα των δειγμάτων πολύ πιο συστηματικά σε σχέση με την τυχαία δειγματοληψία.
Αυτό είναι προφανές από το γεγονός ότι ο εξαρτημένος δειγματολήπτης <<κυκλοφορεί>>
μέσα στα particles συστηματικά. Δεύτερον, αν όλα τα δείγματα έχουν τους ίδιους παράγοντες σημαντικότητας, το ενδιάμεσο σύνολο που προκύπτει είναι ίδιο με το X_t, έτσι ώστε να μην χάνεται κανένα δείγμα και να επαναδειγματοληπτίσουμε χωρίς να έχουμε προσαρτήσει μια παρατήρηση στο X_t. Τρίτον, αυτός ο αλγόριθμος έχει πολυπλοκότητα O(M). Το να φτάσουμε την πολυπλοκότητα αυτή με την τυχαία δειγματοληψία είναι πολύ δύσκολο, με πιο συνηθισμένο το όριο O(M log M).

5.8.1 Ψευδοκώδικας

Αλγόριθμος Resampling (,W_t)

If Neff<m/3

Low_varience_sampler (,W_t)

return X_t

Αλγόριθμος Low_varience_sampler (X_t',W_t)

r=rand (0,M-1)

c=

i=1

for m=1 to M do

U=r+(m-1)*M-1

while U>c

i=i+1

c=c+

end while

πρόσθεσε στο X_t

endfor

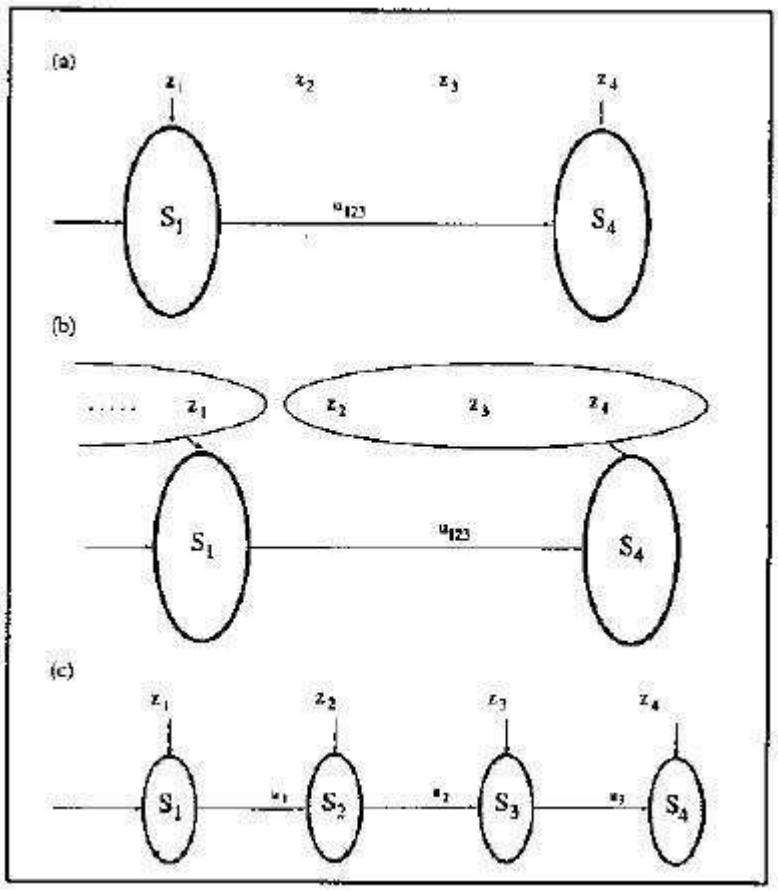
return X_t

5.9 Περιορισμένη υπολογιστική ισχύς

Μια σημαντική προϋπόθεση για τη λειτουργία του φίλτρου σωματιδίων είναι τα N δείγματα να μπορούν να ανανεωθούν πριν ληφθούν νέες πληροφορίες από τον αισθητήρα. Ωστόσο, στα πρώτα στάδια του καθολικού εντοπισμού θέσης είναι πολύ πιθανό να μην έχει ολοκληρωθεί η ανανέωση όταν φτάσει η νέα μέτρηση.

Υποθέτουμε πως οι παρατηρήσεις φτάνουν σε συγκεκριμένα χρονικά διαστήματα. Έστω N ο αριθμός των δειγμάτων που απαιτούνται από το φίλτρο σωματιδίων. Παράθυρο εκτίμησης είναι ο χρόνος που απαιτείται για να ανανεωθούν και τα N δείγματα. Μετράμε τη διάρκεια ή το μέγεθος του παραθύρου εκτίμησης από τον αριθμό των παρατηρήσεων που φτάνουν κατά

τη διάρκεια του παραθύρου. Επομένως, μέγεθος παραθύρου k σημαίνει ότι k παρατηρήσεις φτάνουν κατά τη διάρκεια ανανέωσης των απαιτούμενων N δειγμάτων.



Εικόνα 35: Διάφορες τεχνικές για να αντιμετωπιστεί το πρόβλημα της περιορισμένης υπολογιστικής δυνατότητας.

Τα φίλτρα σωματιδίων επίσημα υποθέτουν ότι όλα τα δείγματα μπορούν να επεξεργαστούν μεταξύ δύο παρατηρήσεων, δηλαδή έχουμε παράθυρο εκτίμησης μεγέθους ένα. Στην εικόνα 33 εμφανίζονται διαφορετικές προσεγγίσεις για να αντιμετωπιστούν περιπτώσεις με μέγεθος παραθύρου μεγαλύτερο από ένα. Όλες οι προσεγγίσεις επεξεργάζονται το ίδιο αριθμό δειγμάτων ανά διάστημα εκτίμησης (windows size=3).

1. Παράλειψη παρατηρήσεων, για παράδειγμα ενσωματώνω μόνο κάθε τρίτη παρατήρηση.
2. Αθροίζω τις παρατηρήσεις και τις ενσωματώνω σε ένα βήμα.
3. Μειώνω το μέγεθος του συνόλου δειγμάτων ώστε να ενσωματώνεται κάθε παρατήση.

Η πιο απλή και κοινή προσέγγιση είναι η (α). Εδώ, οι παρατηρήσεις που φτάνουν κατά τη διάρκεια της ανανέωσης του συνόλου των δειγμάτων απορρίπτονται. Το μειονέκτημα είναι ότι μπορεί έτσι να χαθούν πολύτιμες πληροφορίες από τον αισθητήρα.

Η προσέγγιση (β) ξεπερνά αυτό το πρόβλημα αθροίζοντας πολλαπλές παρατηρήσεις σε μία και στη συνέχεια ολοκληρώνοντας αυτήν την παρατήρηση. Αυτή η τεχνική αποφεύγει την απώλεια πληροφοριών από τον αισθητήρα και θα πρέπει να εφαρμόζεται όποτε είναι δυνατόν. Δυστυχώς, βασίζεται στην υπόθεση πως οι παρατηρήσεις μπορούν να αθροιστούν βέλτιστα και πως η ολοκλήρωση μιας αθροιστικής παρατήρησης μπορεί να γίνει το ίδιο αποτελεσματικά με την ολοκλήρωση ανεξάρτητων παρατηρήσεων. Ενώ οι υποθέσεις αυτές είναι λογικές για γραμμικά μοντέλα αισθητήρων, δεν ισχύουν για αυθαίρετα δυναμικά συστήματα.

Η τρίτη προσέγγιση (γ) σταματά την παραγωγή νέων δειγμάτων κάθε φορά που γίνεται μια παρατήρηση. Επομένως, για μέγεθος παραθύρου k, κάθε σύνολο δειγμάτων περιέχει μόνο N/k δείγματα. Ενώ αυτή η προσέγγιση εκμεταλλεύεται τις δυνατότητες του φίλτρου σωματιδίων κάθε χρονική στιγμή, είναι αμφίβολη ως προς την απόκλιση του φίλτρου εξαιτίας του ανεπαρκούς αριθμού των δειγμάτων. Θα πρέπει να σημειώσουμε πως το N έχει επιλεγεί ως ο αριθμός των δειγμάτων που απαιτούνται για επιτυχημένο filtering.

5.10 Cluster particle filter (CPF)

Η ύστερη κατανομή πιθανότητας (posterior distribution) της ρομποτικής θέσης στο πρόβλημα καθολικού εντοπισμού θέσης (global localization) είναι συνήθως πολυτροπική (multimodal) εξαιτίας της συμμετρίας του περιβάλλοντος και των ασαφών χαρακτηριστικών (features) που ανιχνεύονται. Προτείνεται, λοιπόν το cluster particle filter (CPF) για τη βελτίωση της ακρίβειας και της ευστάθειας (Guanghui et al., 2008).

Το CPF χωρίζει τα δείγματα (particles) σε ομάδες (clusters) ανάλογα με την χωρική ομοιότητά τους. Κάθε cluster θεωρείται ως μια υποθετική θέση και όλα τα clusters εντοπίζονται ανεξάρτητα κατά τη διάρκεια του global localization. Ταυτόχρονα, υπολογίζουμε τις μέσης μη-κανονικοποιημένες πιθανότητες σε κάθε cluster. Εάν η τιμή είναι μικρότερη από ένα προκαθορισμένο όριο, αποκόπτουμε δυναμικά το cluster. Αυτό σημαίνει πως θα επιζήσει το cluster με τα μεγαλύτερα μέσα βάρη.

Έστω ότι το σύνολο των particles διαιρείται σε K clusters και ότι το c_j δηλώνει το j-οστό cluster, όπου j = 1,2,...,K. Το σύνολο των particles είναι N και η πιθανότητα του j cluster τη χρονική στιγμή t είναι $B_{j,t}$.

Χρησιμοποιούμε το k_t για να αναπαραστήσουμε την κατανομή πιθανότητας στα clusters τη χρονική στιγμή t, το z_t για τα δεδομένα από τις μετρήσεις του αισθητήρα τη χρονική στιγμή t και το u_t για τα δεδομένα που λαμβάνουμε από την οδομετρία του ρομπότ τη χρονική στιγμή t:

Επειδή ο αριθμός των particles είναι ορισμένος, μπορούμε να έχουμε τη διακριτή μορφή:

Θα πρέπει να σημειωθεί πως τα particles δεν μπορούν να αλλάξουν cluster οποιαδήποτε κι αν είναι η κίνηση του ρομπότ. Δηλαδή, κάθε particle ξεκινά σε ένα cluster και παραμένει σε αυτό. Προφανώς, θα έχουμε:

Επομένως,

Ταυτόχρονα, θα πρέπει να προσέξουμε πως κάθε cluster αποτελείται από ένα σύνολο particles. Η κατανομή του cluster θα πρέπει να είναι το άθροισμα των κατανομών όλων των particles που ανήκουν σε αυτό το cluster. Το x είναι το σύνολο που περιλαμβάνει τα particles. Άρα:

Από τις δύο προηγούμενες εξισώσεις, λαμβάνουμε την τελική μορφή της εξίσωσης ανανέωσης βάρους του cluster:

Εδώ το είναι παράγοντας κανονικοποίησης. Σε κάθε βήμα, η πιθανότητα κάθε cluster δηλώνεται από την εξίσωση:

Μπορούμε να χρησιμοποιήσουμε το για την εξέλιξη του cluster.

5.11 Προσαρμοστικό φίλτρο σωματιδίων (adaptive particle filter)

Αυτή η προσέγγιση προσαρμόζει τον αριθμό των δειγμάτων σύμφωνα με την πιθανότητα των παρατηρήσεων (Fei et al., 2006). Χρησιμοποιείται η απόσταση Kullback Leibler (K-L) για να δηλώσει τη διαφορά μεταξύ της πραγματικής κατανομής πιθανότητας p(x) και της στοχευόμενης κατανομής πιθανότητας q(x).

Χωρίζουμε το χάρτη σε κ διαφορετικά περιοχές. Έστω το διάνυσμα $X = [X_1, X_2, ..., X_\kappa]$ που δηλώνει τον αριθμό των δειγμάτων που σχεδιάζεται σε κάθε περιοχή και $p = [p_1, p_2, ..., p_\kappa]$ η πιθανότητα σε κάθε περιοχή. Η μέγιστη εκτίμηση πιθανότητας του p δίνεται από τη σχέση $p'=n^{-1}X.$, όπου n είναι το πλήθος των δειγμάτων. Το στατιστικό αναλογίας πιθανοτήτων των λ_n για τον έλεγχο του p είναι:

Όταν το n πάει στο άπειρο το συγκλίνει σε μία τετραγωνική κατανομή:

Η πιθανότητα της διαφοράς μεταξύ του p και του p' να είναι μικρότερη του ε:

Ισχύει :

Εάν επιλέξουμε n ώστε το 2nε να ισούται με , λαμβάνουμε:

(5.37)

Τότε μπορούμε να εγγυηθούμε ότι με πιθανότητα 1-δ, η απόσταση Kullback Leibler (K-L) είναι μικρότερη από ε. Μια καλή προσέγγιση για τον καθορισμό του n είναι:

όπου το άνω 1-δ τέταρτο της τυπικής κανονικής κατανομής N(0,1).

Ο χάρτης που χρησιμοποιείται σε αυτήν την προσέγγιση χωρίζεται σε ίσες περιοχές. Ο αριθμός και το μέγεθός τους επηρεάζει την αποδοτικότητα και το αποτέλεσμα αυτής της προσέγγισης.

ΚΕΦΑΛΑΙΟ 6

BUG ALGORITHMS

6.1 Εισαγωγή

Σκοπός των αλγορίθμων αυτών είναι να ολοκληρώσουν μία διαδρομή από ένα αρχικό σημείο προς ένα τελικό χωρίς να έρθουν σε άμεση επαφή με οποιοδήποτε εμπόδιο. Οι bug algorithms είναι οι απλούστεροι αλγόριθμοι εύρεσης διαδρομής. Χρησιμοποιούνται εκεί όπου δεν υπάρχει πληροφορία για το περιβάλλον και οι θέσεις αλλά και τα σχήματα των όποιων εμποδίων είναι άγνωστα. Τα μόνα στοιχεία που δέχονται αυτοί οι αλγόριθμοι είναι μέσω των αισθητήρων τους. Έτσι οι αλγόριθμοι μπορούν και υποθέτουν πληροφορίες σχετικά με το περιβάλλον και τον προορισμό τους. Εάν δεν υπάρχει προσβάσιμο μονοπάτι, τότε ο αλγόριθμος μπορεί και το αναγνωρίζει και ενημερώνει ότι δεν μπορεί να φτάσει στον ζητούμενο προορισμό, αποφεύγοντας άσκοπες διαδρομές.

Το συγκεκριμένο μοντέλο αλγορίθμου κάνει τρείς υποθέσεις για το ρομπότ ώστε να μπορέσει να λειτουργήσει σωστά. Αυτές είναι οι παρακάτω:

1) Το ρομπότ βρίσκεται στην θέση του αντικειμένου.
2) Το ρομπότ έχει τέλεια ικανότητα εντοπισμού.
3) Το ρομπότ έχει πλήρως λειτουργικούς αισθητήρες.

Αυτές οι τρείς υποθέσεις είναι μη-ρεαλιστικές για τα αληθινά ρομπότ, για αυτόν το λόγο οι αλγόριθμοι αυτοί συνήθως δεν χρησιμοποιούνται σε πρακτικά έργα πλοήγησης, αλλά θεωρούνται ως υψίστου επιπέδου εργαλεία εποπτικών συστημάτων που πληρούν τις παραπάνω υποθέσεις. Οι bug algorithms είναι το πρώτο λογικό βήμα για την επίλυση ενός ρομποτικού προβλήματος δύο διαστάσεων.

Κάποιοι από τους αλγορίθμους που ανήκουν στην κατηγορία των bug algorithms είναι οι εξής: Bug1, Bug2, Alg1, Alg2, Distbug, Class1, Rev1, Rev2, Onebug, VisBug-21, VisBug-22, HD-1, Ave, RoverBug, WedgeBug, CautiousBug, Leavebug, TangentBug.

6.2 Αλγόριθμος Bug1

Ο αλγόριθμος Bug1 είναι ο πρώτος αλγόριθμος στην οικογένεια των bug algorithms με δημιουργούς τους Lumelsky και Stepanov [15,16,17]. Είναι γενικότερα ο πρώτος στην κατηγορία των αλγορίθμων αποφυγής εμποδίων. Είναι εύχρηστος και εύκολος στην χρήση, αλλά κάποιες φορές κατευθύνει το ρομπότ πολύ μακριά από τον στόχο [18]. Σε αυτόν τον αλγόριθμο, το ρομπότ αφού εντοπίσει ένα εμπόδιο αρχίζει και κινείται γύρω από αυτό μέχρι να φτάσει στο σημείο που το εντόπισε για πρώτη φορά. Ταυτόχρονα υπολογίζει την

απόσταση από την τρέχουσα θέση μέχρι τον τελικό προορισμό και εν τέλει αποθηκεύει το σημείο, το οποίο βρίσκεται στην μικρότερη απόσταση από τον στόχο. Αφού ολοκληρώσει έναν κύκλο το ρομπότ, το σημείο αυτό ορίζεται ως το σημείο από το οποίο το ρομπότ θα ξεκινήσει την διαδρομή του. Οπότε το ρομπότ ξεκινά πάλι τον κύκλο γύρω από το εμπόδιο μέχρι να φτάσει στο σημείο αυτό. Αφού αφήσει πίσω του το εμπόδιο, το ρομπότ υπολογίζει τη νέα διαδρομή από το σημείο αυτό (x_1,y_1) μέχρι τον στόχο (x_2,y_2) χρησιμοποιώντας εξίσωση ευθείας γραμμής. Ο συντελεστής διεύθυνσης m και το σημείο τομής c με τον άξονα y δίνονται από τους παρακάτω τύπους.

———

Το ρομπότ ακολουθεί την ευθεία γραμμή μέχρι να φτάσει στον στόχο ή άλλο εμπόδιο. Ένα από τα γνωστά μειονεκτήματα του Bug1 είναι. Υπάρχει περίπτωση ένα δεύτερο εμπόδιο να βρίσκεται πολύ κοντά στο πρώτο. Σε αυτό το πιθανό σενάριο το ρομπότ καθώς κάνει τον κύκλο γύρω από το πρώτο μπορεί να έρθει σε επαφή με το δεύτερο, επειδή είτε τα δύο εμπόδια βρίσκονται πολύ κοντά είτε το μέγεθος του ρομπότ είναι αρκετά μεγάλο για να συμβεί αυτό. Αυτό είναι ένα γνωστό μειονέκτημα του Bug1 αλγορίθμου.

6.2.1 Ψευδοκώδικας

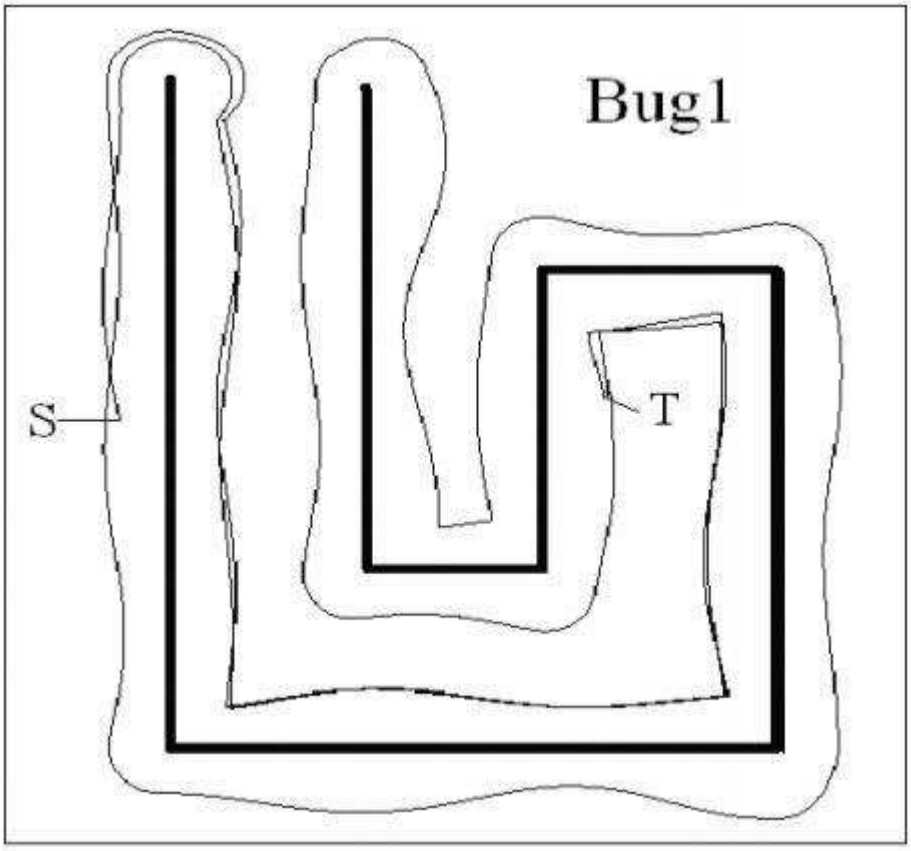
Παρακάτω παρουσιάζεται ένα απλό παράδειγμα ψευδοκώδικα του αλγορίθμου Bug1.

1) Κατευθύνσου προς τον τελικό προορισμό, μέχρις ότου:
    - Φτάσεις στον προορισμό. Ο αλγόριθμος σταματάει.
    - Βρεις εμπόδιο. Όρισε το σημείο επαφής H_j. Πήδα στο βήμα 2.
2) Κινήσου κυκλικά του σημείου επαφής σύμφωνα με την πορεία του ρολογιού και όσο κινείσαι ανανέωνε συνεχώς το πρόγραμμα με την πιο κοντινή απόσταση στον προορισμό μέχρι εκείνη τη στιγμή, μέχρις ότου:
    - Φτάσεις στον προορισμό. Ο αλγόριθμος σταματάει.
    - Έρθεις σε επαφή με το σημείο επαφής H_j ξανά. Έλεγξε αν το εμπόδιο H_j βρίσκεται στην γραμμή που συνδέει την πιο κοντινή στον προορισμό θέση με την τελική:
        - Εάν ισχύει, τότε ο στόχος δεν είναι προσβάσιμος. Ο αλγόριθμος σταματάει.

- Εάν δεν ισχύει, τότε γύρισε στην τελευταία πιο κοντινή στον προορισμό θέση χαράζοντας την πιο κοντινή πορεία είτε αριστερόστροφη είτε δεξιόστροφη. Πήδα στο βήμα 1.

6.2.2 Συμπέρασμα

Συνοπτικά λοιπόν, ο αλγόριθμος Bug1 σαρώνει το κάθε εμπόδιο που εντοπίζει για το σημείο που βρίσκεται πιο κοντά στον προορισμό. Όταν αυτή η θέση οριστεί, το ρομπότ ελέγχει αν μπορεί να συνεχίσει προς τον προορισμό ή όχι. Εάν δεν μπορεί, τότε ο προορισμός δεν είναι προσβάσιμος. Εάν μπορεί, τότε το ρομπότ γνωρίζει πως όταν φύγει από αυτό το συγκεκριμένο σημείο δεν θα ξαναέρθει σε επαφή με το συγκεκριμένο εμπόδιο. Στην παρακάτω εικόνα βλέπουμε ένα απλό παράδειγμα του αλγορίθμου.



Εικόνα 36: Παράδειγμα αλγορίθμου Bug1

6.3 Αλγόριθμος Bug2

Όπως ο αλγόριθμος Bug1 έτσι και αυτός είναι δημιούργημα των Lumelsky και Stepanov [17,18,19]. Είναι λίγο πιο εξελιγμένος (συντηρητικός) από τον Bug1 επειδή το ρομπότ λόγο της M-γραμμής μπορεί και αποφεύγει την άσκοπη περιστροφή γύρω από το εμπόδιο όταν βρει το πιο κοντινό σημείο στον προορισμό. Ο Bug2 αλγόριθμος δημιουργεί την αρχική διαδρομή από την εκκίνηση έως τον προορισμό και αποθηκεύει τον συντελεστή κατεύθυνσης κάθε φορά καθώς κινείται. Η συμπεριφορά του αλγορίθμου αλλάζει όταν βρίσκει εμπόδιο. Πραγματοποιεί τον γύρο του εμποδίου και υπολογίζει την κλίση της ευθείας από την τρέχουσα θέση μέχρι τον στόχο. Όταν η κλίση γίνει ίδια με την κλίση του αρχικού υπολογισμού, το ρομπότ αλλάζει συμπεριφορά και μετακινείται στον στόχο. Έτσι το ρομπότ καταφέρνει και ακολουθεί μη επαναλαμβανόμενη διαδρομή κατά την διάρκεια της κίνησης του. Ο Bug2 αλγόριθμος είναι πιο αποτελεσματικός από τον Bug1, επειδή επιτρέπει στο ρομπότ να φτάσει στον στόχο σε λιγότερο χρόνο.

Αμφότεροι οι παραπάνω αλγόριθμοι απαιτούν ελάχιστες απαιτήσεις συστήματος. Ωστόσο, δεν έχουν την δυνατότητα να κάνουν βέλτιστη χρήση των αισθητήρων για την δημιουργία μικρότερων διαδρομών.

6.3.1 Ψευδοκώδικας

Παρακάτω παρουσιάζεται ένα απλό παράδειγμα ψευδοκώδικα του αλγορίθμου Bug2.

1) Σχεδίασε μια φανταστική γραμμή M, από την αρχική θέση μέχρι τον τελικό προορισμό και θέσε το i=0.
2) Αύξησε το i κατά 1 και ακολούθησε την γραμμή μέχρι τον τελικό προορισμό, μέχρις ότου:
    - Φτάσεις στο προορισμό. Ο αλγόριθμος σταματάει.
    - Βρεις εμπόδιο. Όρισε το σημείο επαφής H_i. Πήδα στο βήμα 3.
3) Κινήσου αριστερόστροφα γύρω από το εμπόδιο, ακολουθώντας τα όρια του, μέχρις ότου:
    - Φτάσεις στο προορισμό. Ο αλγόριθμος σταματάει.
    - Βρεις σημείο κατά μήκος της M-γραμμής, όπου ισχύει $d(x,T)<d(H_i,T)$. Εάν το ρομπότ μπορεί να κινηθεί προς τον προορισμό, όρισε αυτό το σημείο ως L_i. Πήδα στο βήμα 2. Αλλιώς, αντικατέστησε το $d(H_i,T)$ με το $d(x,T)$.

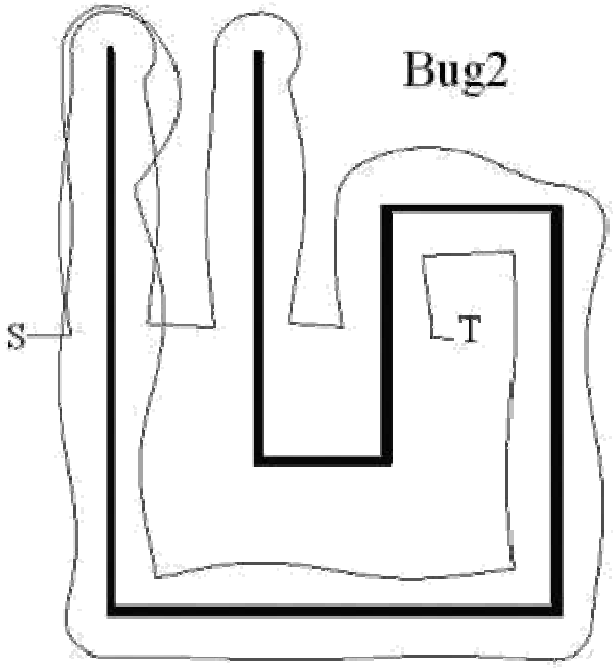
- Επιστρέψεις στο H_i. Ο προορισμός δεν είναι προσβάσιμος. Ο αλγόριθμος σταματάει.

6.3.2 Διευκρινήσεις

Υπήρχαν κάποια προβλήματα σχετικά με τις συνθήκες που επέτρεπαν στον Bug2 αλγόριθμο να αποφεύγει την άσκοπη περιστροφή γύρω από το εμπόδιο. Για αυτόν τον λόγο ο Antich και ο Ortiz πρότειναν έναν νέο αλγόριθμο τον Bug2+, ο οποίος ξεκαθάριζε όλα αυτά τα προβλήματα, αλλά ο Sankar και ο Noborio είχαν ήδη συμπεριλάβει αυτές τις τροποποιήσεις στους αλγόριθμους που δημιούργησαν και είναι όμοιοι του Bug2. Έτσι ναι μεν θα αναφερόμαστε στον Bug2+ αλλά στην ουσία εννοούμε τον Bug2 επειδή όλες αυτές οι αλλαγές συμπεριλαμβάνονται στον Bug2 απλά ο δημιουργός του Lumelsky ξέχασε να τις αναφέρει.

Ο αρχικός κανόνας έλεγε πως όταν το ρομπότ φτάσει σε ένα σημείο όπου η απόσταση του μέχρι τον προορισμό είναι μικρότερη από την απόσταση από το αρχικό σημείο, τότε ορίζει αυτή τη θέση ως σημείο διαφυγής L_j. Αυξάνει το i κατά ένα και συνεχίζει από το βήμα 2. Μία τόσο αυστηρή λεπτομέρεια δίνει την δυνατότητα στον αλγόριθμο Bug2 να ορίσει ένα σημείο διαφυγής, το οποίο έτσι και αλλιώς θα όριζε το ρομπότ όταν θα εκτελούσε το βήμα 2 και δεν θα είχε κινηθεί καθόλου. Ωστόσο αυτό δεν είναι και τόσο λογικό, δηλαδή το ρομπότ να μπορεί να φύγει σε περίπτωση που δεν κινείται απευθείας προς τον προορισμό. Το ρομπότ του επιτρέπεται να φύγει μόνο αν επρόκειτο να κατευθυνθεί προς τον προορισμό.

Επίσης, αν το ρομπότ δεν του επιτρέπεται να φύγει, επειδή δεν μπορεί να κινηθεί προς τον προορισμό, τότε πρέπει το $d(H_i,T)$ να αντικατασταθεί με το $d(x,T)$. Προφανώς, καταλήγουμε στο συμπέρασμα, πως όταν το ρομπότ δεν έχει την άδεια να φύγει επειδή δεν μπορεί να φτάσει στον προορισμό, τότε αυτό σημαίνει πως πάνω στην M-γραμμή υπάρχει μια θέση που βρίσκεται πιο κοντά στον προορισμό. Σε κάθε περίπτωση, εάν η αρχική έκδοση του αλγορίθμου του Lumelsky ακολουθούσε αυστηρά αυτό το μονοπάτι τότε είναι ακριβώς ίδιος με τον Bug2+, επειδή το ρομπότ θα αντικαταστήσει το $d(H_i,T)$ όταν εκτελεστεί το βήμα 2. Στην παρακάτω εικόνα βλέπουμε ένα απλό παράδειγμα του αλγορίθμου.



Εικόνα 37: Παράδειγμα αλγορίθμου Bug2

6.4 Αλγόριθμος Alg1

Ο αλγόριθμος Alg1 είναι μία επέκταση του Bug2 και δημιουργοί του είναι οι Sankaranarayanan και Vidyasagar [22]. Η αδυναμία του Bug2 είναι, ότι μπορεί να εντοπίσει το ίδιο μονοπάτι δύο φορές και έτσι να δημιουργήσει μεγαλύτερες διαδρομές. Ο Alg1 το διορθώνει αυτό καθώς θυμάται τις προηγούμενες θέσεις (hit και leave points) και τις χρησιμοποιεί για να δημιουργήσει μικρότερα μονοπάτια.

6.4.1 Ψευδοκώδικας

Παρακάτω παρουσιάζεται ένα απλό παράδειγμα ψευδοκώδικα του αλγορίθμου Alg1.

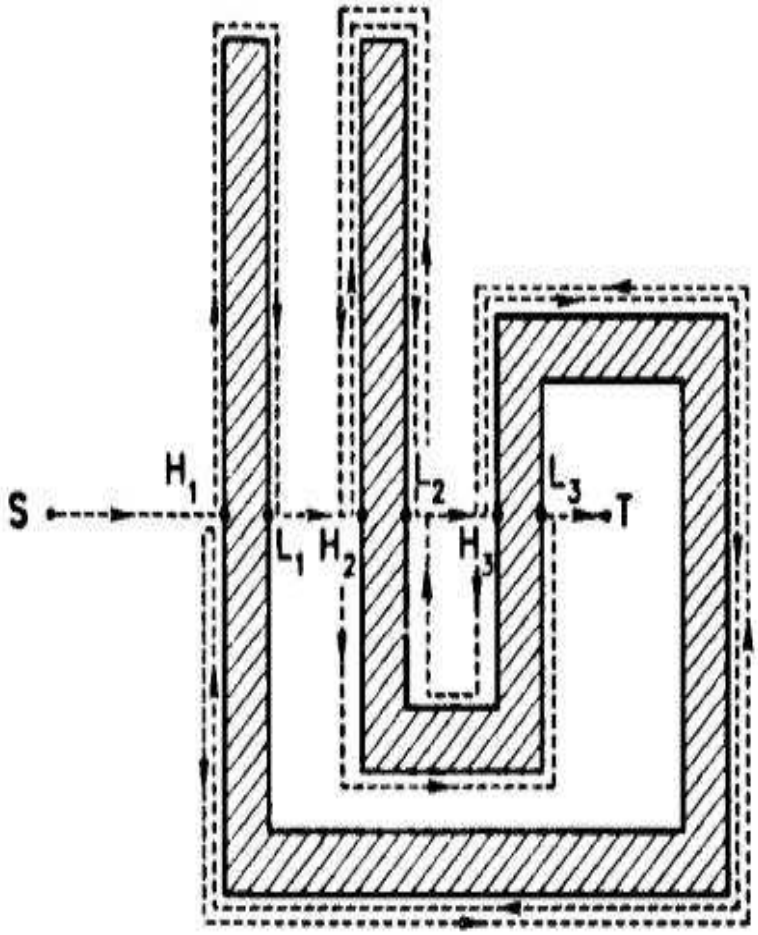
1) Σχεδίασε μια φανταστική γραμμή M, από την αρχική θέση μέχρι τον τελικό προορισμό και θέσε το i=0.
2) Αύξησε το i κατά 1 και ακολούθησε την γραμμή μέχρι τον τελικό προορισμό, μέχρις ότου:
    - Φτάσεις στο προορισμό. Ο αλγόριθμος σταματάει.
    - Βρεις εμπόδιο. Όρισε το σημείο επαφής H_i. Πήδα στο βήμα 3.
3) Κινήσου αριστερόστροφα γύρω από το εμπόδιο, ακολουθώντας τα όρια του, μέχρις ότου:
    - Φτάσεις στο προορισμό. Ο αλγόριθμος σταματάει.
    - Βρεις σημείο y:

- το οποίο να βρίσκεται πάνω στην M-γραμμή.
- να ισχύει ότι d(y,T)<d(x,T) για όλα τα x που ανήκουν στην M-γραμμή και έχει ήδη επισκεφτεί το ρομπότ .
- το ρομπότ να είναι σε κατάλληλη θέση, ώστε να μπορεί να κινηθεί προς τον προορισμό.

Όρισε αυτό το σημείο ως L_i. Πήδα στο βήμα 2.

- Φτάσεις σε μία προηγούμενη θέση H_j ή L_j, όπου ισχύει j<i. Γύρνα πίσω στο H_i. Όταν φτάσεις στο H_i, ακολούθησε αριστερόστροφα τα όρια του εμποδίου. Αυτός ο κανόνας δεν μπορεί να ισχύσει ξανά άμα δεν οριστεί το L_i.
- Επιστρέψεις στο H_i. Ο προορισμός δεν είναι προσβάσιμος. Ο αλγόριθμος σταματάει.

Στην παρακάτω εικόνα βλέπουμε ένα απλό παράδειγμα του αλγορίθμου.



Εικόνα 38: Παράδειγμα αλγορίθμου Alg1 [22]

6.5 Αλγόριθμος Alg2

Ο αλγόριθμος Alg2 είναι μία βελτιωμένη έκδοση του Alg1 με δημιουργούς και πάλι τους Sankaranarayanan και Vidyasagar [22]. Το ρομπότ παύει να λειτουργεί βάση της M-γραμμής και πλέον υιοθετείται νέα συνθήκη.

6.5.1 Ψευδοκώδικας

Παρακάτω παρουσιάζεται ένα απλό παράδειγμα ψευδοκώδικα του αλγορίθμου Alg2.

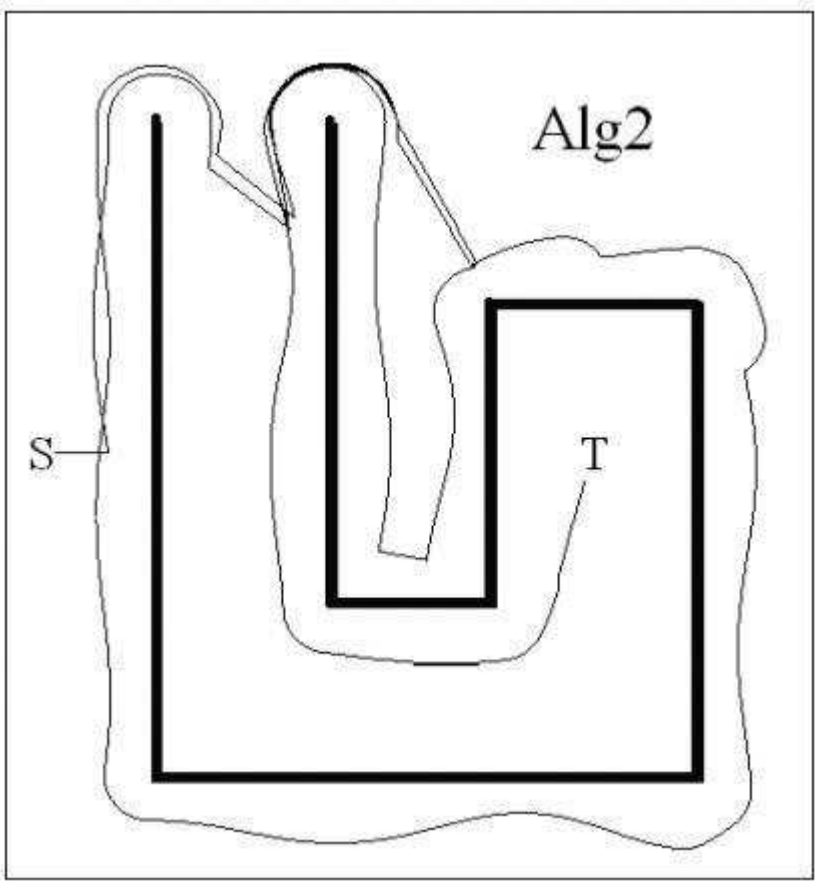
1) Θέσε Q=d(S,T) και i=0.
2) Αύξησε το i κατά ένα και προχώρησε προς τον προορισμό ανανεώνοντας συνεχώς το Q με d(x,T) εάν ισχύει ότι Q<d(x,T). Άρα τώρα το Q συμβολίζει την κοντινότερη απόσταση που έχει βρεθεί το ρομπότ από τον προορισμό του. Επανάλαβε αυτό το βήμα μέχρις ότου:

   I. Φτάσεις στο προορισμό. Ο αλγόριθμος σταματάει.
   II. Βρεις εμπόδιο. Όρισε το σημείο επαφής H_i. Πήδα στο βήμα 3.

3) Κινήσου αριστερόστροφα γύρω από το εμπόδιο, ακολουθώντας τα όρια του και ανανέωνε συνεχώς το Q με d(x,T) εάν ισχύει ότι Q<d(x,T), μέχρις ότου:
   - Φτάσεις στο προορισμό. Ο αλγόριθμος σταματάει.
   - Βρεις σημείο y:
     - Να ισχύει, ότι d(y,T)<d(Q,T) και
     - το ρομπότ να είναι σε κατάλληλη θέση, ώστε να μπορεί να κινηθεί προς τον προορισμό.
   - Όρισε αυτό το σημείο ως L_i. Πήδα στο βήμα 2.
   - Φτάσεις σε μία προηγούμενη θέση H_j ή L_j, όπου ισχύει j<i. Γύρνα πίσω στο H_i. Όταν φτάσεις στο H_i, ακολούθησε αριστερόστροφα τα όρια του εμποδίου. Αυτός ο κανόνας δεν μπορεί να ισχύσει ξανά άμα δεν οριστεί το L_i.
   - Επιστρέψεις στο H_i. Ο προορισμός δεν είναι προσβάσιμος. Ο αλγόριθμος σταματάει.

6.5.2 Συμπέρασμα

Η νέα συνθήκη αυτού του αλγορίθμου είναι μία ιδανική βελτίωση, επειδή το ρομπότ δεν χρειάζεται να βρίσκεται στην M-γραμμή για να αποφύγει το εμπόδιο. Το γεγονός αυτό βελτιώνει την εκτέλεση και είναι υπολογιστικά πιο φιλικό. Ωστόσο τέτοιες βελτιώσεις απαιτούν μία μέθοδο για να αποφύγουν την περίπτωση Class1. Στην παρακάτω εικόνα βλέπουμε ένα απλό παράδειγμα του αλγορίθμου.



Εικόνα 39: Παράδειγμα αλγορίθμου Alg2

6.6 Αλγόριθμος Class1

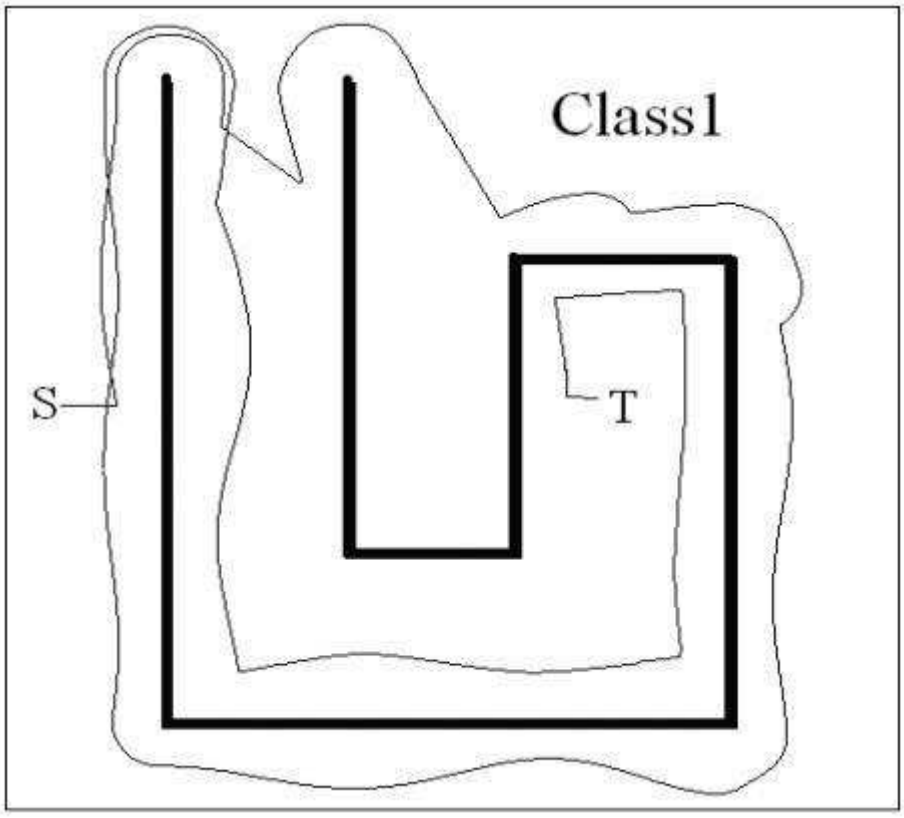
Ο αλγόριθμος Class1 δεν ανήκει επίσημα στους Bug Algorithms και δεν εγγυάται πεπερασμένο τερματισμό. Χρησιμοποιείται για να μας δείχνει τι συμβαίνει όταν το ρομπότ έχει την άδεια να φύγει αν βρίσκεται ήδη στο κοντινότερο σημείο στον προορισμό και μπορεί να κατευθυνθεί προς αυτόν. Ο Class1 γενικότερα χρησιμοποιείται για την ανάπτυξη των Bug Algorithms και για να δικαιολογήσει την χρήση των ειδικών κανόνων που επιτρέπουν στο ρομπότ να φύγει από την εκάστοτε θέση που βρίσκεται.

6.6.1 Ψευδοκώδικας

Παρακάτω παρουσιάζεται ένα απλό παράδειγμα ψευδοκώδικα του αλγορίθμου Class1.

- Κινήσου προς τον προορισμό, μέχρις ότου:
  - Φτάσεις στο προορισμό. Ο αλγόριθμος σταματάει.
  - Βρεις εμπόδιο. Ακολούθησε αριστερόστροφα τα όρια του εμποδίου. Πήδα στο βήμα 2.
- Φύγε εάν το ρομπότ μπορεί να κατευθυνθεί προς τον προορισμό και είναι πιο κοντά σε αυτόν από τα τις προηγούμενες θέσεις που πέρασε. Πήδα στο βήμα 1.

Στην παρακάτω εικόνα βλέπουμε ένα απλό παράδειγμα του αλγορίθμου.



Εικόνα 40: Παράδειγμα αλγορίθμου Class1

6.7 Dist-Bug Algorithm

Ο αλγόριθμος αυτός είναι η πιο βελτιωμένη έκδοση των Bug Algorithms. Είναι ικανός να διανύσει πολύ μικρότερες αποστάσεις με αποτέλεσμα το ρομπότ να φτάσει γρηγορότερα στον προορισμό του. Συμπεριφέρεται διαφορετικά στον τρόπο αποφυγής εμποδίων σε σύγκριση με ότι είχαμε δει μέχρι τώρα. Όταν το ρομπότ συναντά ένα εμπόδιο στην πορεία του, καθώς

πραγματοποιεί τον γύρο από αυτό υπολογίζει και αποθηκεύει συνεχώς την τρέχουσα απόσταση από τον στόχο. Αυτό που τον κάνει μοναδικό είναι, ότι μπορεί και κάνει ταυτόχρονα το ίδιο για την επόμενη θέση που θα βρεθεί. Για να συνεχίσει την πορεία του το ρομπότ προς τον στόχο, θα πρέπει η απόσταση προς τον στόχο της επόμενης θέσης που θα βρεθεί να είναι μεγαλύτερη από αυτήν της τρέχουσας θέσης προς τον στόχο ($d_{next}>d_{current}$). Διαφορετικά το ρομπότ συνεχίζει την προηγούμενη λειτουργία του.

ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] Α. Βακάλη, Η. Γιαννόπουλος, Ν. Ιωαννίδης, Χ. Κοίλιας, Κ. Μαλάμας, Ι. Μανωλόπουλος, Π. Πολίτης (2010 – Έκδοση ΙΑ). *Ανάπτυξη Εφαρμογών σε Προγραμματιστικό Περιβάλλον*. Υπουργείο Εθνικής Παιδείας και Θρησκευμάτων – Παιδαγωγικό Ινστιτούτο, σελ 25-27.

[2] A.R. Mashaghi, A. Ramezanpour, V. Karimipour (2004). *Investigation of a protein complex network*. European Physical Journal B41 (1): 113-121.

[3] Rosen, Kenneth H. *Discrete mathematics and its applications (7th edition)*. New York: McGraw-Hill.

[4] Αθανάσιος Τσατσάνης (2014). *Εφαρμογή Βασικών Αλγορίθμων Εύρεσης Βέλτιστης Διαδρομής σε Δίκτυα Μεταφορών*. Εθνικό Μετσόβιο Πολυτεχνείο.

[5] Dijstra Edsger, Thomas J. Misa, Editor (August 2010). *An interview with Edsger W. Dijkstra*. Communications of the ACM 53 (8): 41-47.

[6] Γεώργιος Β. Σομπόνης (2012). *Ευφυής Προσδιορισμός Βέλτιστων Διαδρομών βάσει Μεθόδων Μηχανικής Μάθησης*, Εθνικό Μετσόβιο Πολυτεχνείο.

[7] Yilin Zhao (1997). *Vehicle Location and Navigation Systems: Intelligent Transportation Systems*. Artech House.

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

[9] Hart, E.P., Nilsson, J.N., Bertram R (1968). *A formal basis for the heuristic determination of minimum cost paths*. IEEE Transactions on System Science and Cybernetics, vol. 4, σελ. 100-107.

[10] Dijkstra, E.W (1959). *A note on two problems in connection with graphs*. Numerische Mathematik 1, σελ. 269-271.

[11] Arulampalam, M., Maskell, S., Gordon, S.& Clapp, T.(2002), *A tutorial on particle filters for online nonlinear/non-Gaussian Bayesian tracking*, IEEE Transactions on Signal Processing, Volume 50, Issue 2, pp. 174 – 188.

[12] Velmurugan R. (2007). *Implementation Strategies for Particle Filter Based Target* Diploma Thesis, Georgia Institute of Technology.

[13] Ελευθερία Ε. Γεωργιάδου (2009). *Έλεγχος Οπτικής Οδήγησης Ρομποτικών Χειριστών με εφαρμογή φίλτρων Kalman*. Εθνικό Μετσόβιο Πολυτεχνείο.

[14] Επιστήμη Π. Παπανικολάου (2008). *Ταυτόχρονος Προσδιορισμός Θέσης Κινούμενου Ρομπότ και Χαρτογράφηση*. Εθνικό Μετσόβιο Πολυτεχνείο.

[15] V. Lumelsky and A. Stepanov (1987). *Path planning strategies for a point mobile automaton moving amidst unknown obstacles of arbitrary shape*. Algorithmica, vol. 2, pp. 403-430.

[16] V. Lumelsky and P. Stepanov (1984). *Effect of Uncertainty on Continuous Path Planning for an Autonomous Vehicle*. Proceedings of the 23rd Conference on Decision and Control, pp. 1616-1621.

[17] V. Lumelsky and A. Stepanov (1986). *Dynamic Path Planning For A Mobile Automaton With Limited Information On The Environment*. IEEE Transactions On Automatic Control, Vol. AC.31, No. 11, Nov.

[18] Yufka, A. , Parlaktuna, O (2009). *Performance Comparison of Bug Algorithms for Mobile Robots*. 5th International Advanced Technologies Symposium. Karabuk, Turkey.

[19] James Ng (2010). *An Analysis of Mobile Robot Navigation Algorithms in Unknown Environments*. School of Electrical, Electronic and Computer Engineering.

[20] Muhammad Zohaib, Syed Mustafa Pasha, Nadeem Javaid, Jamshed Iqbal. *Intelligent Bug Algorithm (IBA): A Novel Strategy to Navigate Mobile Robots Autonomously. Department of Electrical Engineering*, COMSATS Institute of Information Technology, Islamabad, Pakistan.

[21] Basem M. ElHalawany, Hala M. Abdel-Kader, Adly TagEldeen, Alaa Eldeen Elsayed. *Modified A* Algorithm for Safer Mobile Robot Navigation*, Electronics & Communications Department, Shoubra Faculty of Engineering Benha University Cairo, Egypt.

[22] Sankaranarayanan A., Vidyasagar, M. (1990), *A new path planning algorithm for moving a point object amidst unknown obstacles in a plane*, IEEE conference on robotics and automation, pp. 1930-1936.