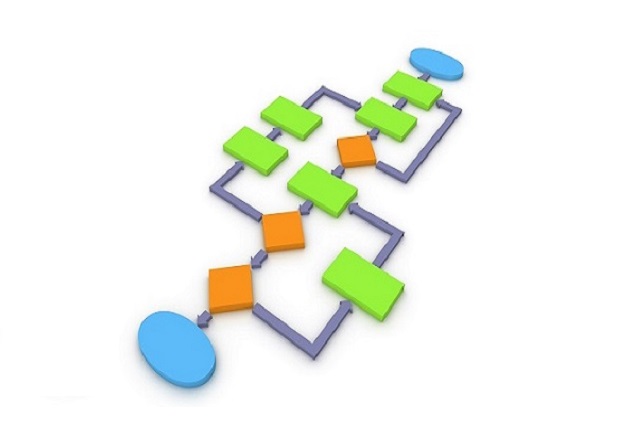
****



**ΤΕΧΝΗΤΗ ΝΟΗΜΟΣΥΝΗ-ΕΜΠΕΙΡΑ ΣΥΣΤΗΜΑΤΤΑ 2024-2025**

**Εκπώνηση εργασίας:**

**Χαρίδης Μάριος-Χρήστος Α.Μ. ΜΠΠΛ2336**

**Καθηγητής:**

## **Παναγιωτόπουλος Θεμιστοκλής**

**Ημερομηνία: 10/2/2025**

***Αφιερωμένο στη μνήμη του καθηγητή Παναγιωτόπουλου Θεμιστοκλή,  
ο οποίος με το πάθος του για τη γνώση και τη διδασκαλία ενέπνευσε εμένα και τόσους άλλους να εξερευνήσουμε τον κόσμο της επιστήμης. Ήταν πάντα στο πλευρό των μαθητών του, πρόθυμος να βοηθήσει και να συμβάλει με την εμπειρία του. Η συνεισφορά του στην εκπαίδευση και η αφοσίωσή του στους μαθητές του θα παραμείνουν ανεξίτηλες στη μνήμη μας.***

**Πίνακας Περιεχομένων**

1. Εισαγωγή στην Τεχνητή Νοημοσύνη και Αναζήτηση Διαδρομής
   1. Σκοπός της εργασίας
   2. Περιγραφή του προβλήματος

*1.2.1* Περιγραφή του Περιβάλλοντος

*1.2.2* Προκλήσεις και Περιορισμοί

*1.2.3* Χρήση του Αλγόριθμου Α\*

* 1. Εφαρμογές του αλγορίθμου Α\*

1. **Εισαγωγή στους Ευριστικούς Αλγορίθμους Αναζήτησης**

*2.1* Η ιστορία των Ευριστικών  
*2.2* Ο ρόλος της Τεχνητής Νοημοσύνης στην αναζήτηση διαδρομής  
*2.3* Τι είναι οι ευριστικοί αλγόριθμοι  
*2.4* Γενική αρχή λειτουργίας των αλγορίθμων αναζήτησης

1. Εισαγωγή στον αλγόριθμο A\*
   1. Σύγκριση A\* με άλλες μεθόδους αναζήτησης (BFS, DFS)
   2. Βασική Ιδέα και Ψευδοκώδικας του Α\*
   3. Η έννοια του Κόστους Διαδρομής (g, h, f functions)
   4. Υλοποίηση με χρήση JavaScript και p5.js
   5. Δημιουργία html σελίδας για οπτικοποίηση του A\*
   6. Ανάλυση ιστοσελίδας / Εμπειρία Χρήστη
   7. Συμπεράσματα και Μελλοντικές Προεκτάσεις

*3.7.1* Πλεονεκτήματα και Μειονεκτήματα του Α\*

*3.7.2* Προτάσεις για βελτιώσεις και μελλοντικές έρευνες

*3.7.3* Σύνδεσμοι

**Κεφάλαιο 1: Εισαγωγή στην Τεχνητή Νοημοσύνη και Αναζήτηση Διαδρομής**

## ***Σκοπός της Εργασίας***

Ο σκοπός αυτής της εργασίας είναι η ανάλυση, υλοποίηση και οπτικοποίηση του αλγορίθμου **A\***, ενός από τους πιο διαδεδομένους αλγορίθμους εύρεσης της βέλτιστης διαδρομής. Στόχος είναι να κατανοήσουμε τόσο τη θεωρητική του βάση όσο και την πρακτική του εφαρμογή μέσα από μία διαδραστική οπτικοποίηση σε **JavaScript** χρησιμοποιώντας τη βιβλιοθήκη **p5.js**.

Η εργασία επιδιώκει να απαντήσει σε ερωτήματα όπως:

* Πώς λειτουργεί ο A\* και ποια είναι η θεωρητική του ανάλυση;
* Πώς συγκρίνεται με άλλους αλγορίθμους αναζήτησης διαδρομής;
* Πώς επηρεάζουν οι παράμετροί του (ευρετική συνάρτηση, επιτρεπόμενες κινήσεις) την απόδοσή του;
* Πώς μπορούμε να τον οπτικοποιήσουμε και να παρέχουμε ένα διαδραστικό περιβάλλον χρήστη;

### ***1.2 Περιγραφή του Προβλήματος***

Η αναζήτηση της βέλτιστης διαδρομής σε έναν χώρο αποτελεί ένα από τα πιο θεμελιώδη προβλήματα στην επιστήμη των υπολογιστών, με εφαρμογές σε πολλούς τομείς, όπως η τεχνητή νοημοσύνη, η ρομποτική και τα βιντεοπαιχνίδια. Το πρόβλημα αφορά τον εντοπισμό της πιο αποδοτικής διαδρομής από ένα αρχικό σημείο προς έναν προορισμό, λαμβάνοντας υπόψη εμπόδια, κόστος μετακίνησης και πιθανές εναλλακτικές πορείες.

#### **1.2.1 Περιγραφή του Περιβάλλοντος**

Το περιβάλλον στο οποίο εκτελείται η αναζήτηση μπορεί να είναι:

* **Δισδιάστατος ή τρισδιάστατος χώρος** (π.χ. χάρτης μιας πόλης, λαβύρινθος, πίστα σε βιντεοπαιχνίδι).
* **Δικτυωτή αναπαράσταση (grid-based ή graph-based)**, όπου οι κόμβοι και οι ακμές αντιστοιχούν σε σημεία και διαδρομές μεταξύ τους.
* **Δυναμικό ή στατικό περιβάλλον**, δηλαδή αν τα εμπόδια και οι συνθήκες μετακίνησης παραμένουν σταθερά ή αλλάζουν με τον χρόνο.

#### **1.2.2 Προκλήσεις και Περιορισμοί**

Η εύρεση της βέλτιστης διαδρομής δεν είναι πάντα απλή, καθώς υπάρχουν διάφοροι παράγοντες που δυσκολεύουν την αναζήτηση:

* **Υπολογιστική πολυπλοκότητα**: Ο χρόνος εκτέλεσης αυξάνεται εκθετικά με το μέγεθος του χώρου.
* **Διαχείριση εμποδίων**: Πρέπει να λαμβάνονται υπόψη τείχη, κλειστές διαδρομές ή μεταβλητές συνθήκες.
* **Ισορροπία μεταξύ ταχύτητας και ακρίβειας**: Ένας αλγόριθμος μπορεί να είναι ταχύς αλλά να μη βρίσκει πάντα τη βέλτιστη διαδρομή.
* **Δυναμικές αλλαγές**: Σε ένα περιβάλλον όπου τα εμπόδια ή οι συνθήκες μετακίνησης αλλάζουν, ο αλγόριθμος πρέπει να είναι ικανός να προσαρμόζεται σε νέα δεδομένα.

#### **1.2.3 Χρήση του Αλγορίθμου A\***

Ο αλγόριθμος A\* (A-star) χρησιμοποιείται ευρέως για την επίλυση αυτού του προβλήματος επειδή:

* **Συνδυάζει ταχύτητα και ακρίβεια**, χρησιμοποιώντας ευρετική αναζήτηση.
* **Είναι εφαρμόσιμος σε διάφορους τύπους περιβαλλόντων**, είτε πρόκειται για στατικούς χάρτες είτε για δυναμικά περιβάλλοντα.
* **Ελαχιστοποιεί το συνολικό κόστος διαδρομής**, αξιολογώντας τόσο την απόσταση που έχει ήδη διανυθεί όσο και την εκτιμώμενη απόσταση προς τον στόχο.

Συνοψίζοντας, το πρόβλημα της εύρεσης βέλτιστης διαδρομής είναι ιδιαίτερα σημαντικό στην επιστήμη των υπολογιστών και η λύση του απαιτεί έξυπνους αλγορίθμους, όπως ο A\*, οι οποίοι μπορούν να προσφέρουν αποδοτικές και ακριβείς διαδρομές, ακόμα και σε πολύπλοκα περιβάλλοντα.

## ***1.3 Εφαρμογές του Αλγορίθμου A\****

Ο αλγόριθμος **A\*** βρίσκει εφαρμογή σε πολλούς τομείς της πληροφορικής και της μηχανικής:

### **✔️ Τεχνητή Νοημοσύνη (AI) και Video Games**

* **Pathfinding σε 2D/3D περιβάλλοντα**: Οι χαρακτήρες σε παιχνίδια χρησιμοποιούν τον A\* για να βρίσκουν το δρόμο τους στο χάρτη.
* **Στρατηγικά παιχνίδια (RTS)**: Π.χ. το **StarCraft** και το **Age of Empires** χρησιμοποιούν τον A\* για τον έλεγχο των μονάδων.
* **Αποφυγή εμποδίων**: Οι NPCs (non-player characters) προσαρμόζουν τις κινήσεις τους δυναμικά με χρήση του αλγορίθμου.

### **✔️ Ρομποτική και Αυτοματισμοί**

* Αυτόνομα ρομπότ (π.χ. **Roomba**) χρησιμοποιούν τον A\* για να αποφεύγουν εμπόδια και να κινούνται σε εσωτερικούς χώρους.
* Τα drones χαρτογραφούν περιοχές και βρίσκουν την καλύτερη διαδρομή χωρίς να συγκρούονται με εμπόδια.

### **✔️ GPS και Πλοήγηση Οχημάτων**

* Ο A\* χρησιμοποιείται σε **GPS εφαρμογές (Google Maps, Waze)** για να βρίσκει τη βέλτιστη διαδρομή με βάση την κυκλοφορία.
* Χρησιμοποιείται σε αυτόνομα οχήματα για τη λήψη αποφάσεων πλοήγησης.

### **✔️ Δίκτυα Υπολογιστών & Data Routing**

* Εφαρμόζεται σε **δικτυακούς αλγορίθμους δρομολόγησης** (routing), όπως το πρωτόκολλο **OSPF** (Open Shortest Path First).
* Βοηθά στον σχεδιασμό των πιο αποδοτικών διαδρομών μεταφοράς δεδομένων στο **Internet**.

**Κεφάλαιο 2. Εισαγωγή στους Ευριστικούς Αλγορίθμους Αναζήτησης**

### *Η Ιστορία των Ευριστικών*

Η λέξη «ευριστικός» προέρχεται απ΄ το ελληνικό ρήμα ευρίσκειν, που σημαίνει βρίσκω, ανακαλύπτω. Λέγεται ότι ο Αρχιμήδης έτρεχε γυμνός στο δρόμο φωνάζοντας «Εύρηκα» ανακαλύπτοντας τη βασική αρχή της άνωσης.

Η τεχνική σημασία των «Ευριστικών» έχει περάσει από αρκετές αλλαγές στην ιστορία της τεχνητής νοημοσύνης. Το 1957, ο George Polya έγραψε ένα βιβλίο με τίτλο *Πώς να το λύσεις*. Χρησιμοποιώντας την «ευριστική» για να αναφερθεί στη μελέτη μεθόδων επίλυσης προβλημάτων, ιδιαίτερα προβλημάτων που χρειάζονται μαθηματικές αποδείξεις.

Πολλοί χρησιμοποιούν τα ευριστικά σαν το αντίθετο των αλγορίθμων. Για παράδειγμα, οι Newell Saw και Simon δήλωσαν το 1963, «Μια διαδικασία που μπορεί να λύσει ένα δοσμένο πρόβλημα, αλλά δεν προσφέρει εγγυήσεις ότι θα το κάνει, λέγεται ευριστική για αυτό το πρόβλημα». Αλλά σημειώστε ότι δεν υπάρχει τίποτα το τυχαίο ή το μη-ντετερμινιστικό σ΄ έναν αλγόριθμο ευριστικής αναζήτησης: προχωράει με αλγοριθμικά σήματα προς το αποτέλεσμα. Σε κάποιες περιπτώσεις, δεν υπάρχει εγγύηση για το πόσο θα κρατήσει η αναζήτηση, και σε κάποιες περιπτώσεις δεν εγγυάται η ποιότητα της λύσης.

Οι ευριστικές τεχνικές είχαν κυριαρχήσει στις πρώτες εφαρμογές της τεχνητής νοημοσύνης. Το πρώτο εργαστήριο «εμπείρων συστημάτων» ονομαζόταν Project Ευριστικού Προγραμματισμού (HPP). Tα ευριστικά χρησιμοποιήθηκαν ευρέως για να παράγουν καλές λύσεις χωρίς εξαντλητική αναζήτηση. Στην αρχή είχαν ενσωματωθεί απ΄ ευθείας στη δομή των προγραμμάτων, αλλά τελικά αποδείχτηκαν όχι τόσο προσαρμόσιμα όταν χρειαζόταν μεγάλος αριθμός τους. Σταδιακά, σχεδιάζοντας συστήματα που μπορούσαν να δεχτούν ευριστική πληροφορία με τη μορφή «κανόνων», κι έτσι γεννήθηκαν τα συστήματα που βασίζονται σε κανόνες.

Πρόσφατα, χρησιμοποιούνται τα ευριστικά σαν αντικείμενο που βοηθάει το μέσο όρο απόδοσης. Στον τομέα των αλγορίθμων αναζήτησης, αναφέρεται σε μια συνάρτηση που παρέχει μια εκτίμηση του κόστους της λύσης.

#### **2.2 Ο ρόλος της Τεχνητής Νοημοσύνης στην αναζήτηση διαδρομής**

Η **Τεχνητή Νοημοσύνη (AI)** έχει φέρει επανάσταση σε πολλούς τομείς της πληροφορικής, επιτρέποντας στα συστήματα να λαμβάνουν αποφάσεις βασισμένες σε δεδομένα και ευρετικές στρατηγικές. Ένας από τους κλάδους που επωφελείται σημαντικά από την AI είναι η **αναζήτηση διαδρομής**, η οποία χρησιμοποιείται για την εύρεση της βέλτιστης πορείας από ένα σημείο Α σε ένα σημείο Β μέσα σε ένα συγκεκριμένο περιβάλλον.

Οι εφαρμογές της αναζήτησης διαδρομής είναι εκτενείς και περιλαμβάνουν:

* **Συστήματα πλοήγησης και GPS**, όπου οι αλγόριθμοι βρίσκουν την ταχύτερη ή συντομότερη διαδρομή.
* **Βιντεοπαιχνίδια**, όπου οι μηχανισμοί τεχνητής νοημοσύνης ελέγχουν την κίνηση χαρακτήρων (π.χ. NPCs).
* **Ρομποτική**, όπου οι ρομποτικές πλατφόρμες σχεδιάζουν τη βέλτιστη πορεία προς έναν στόχο, αποφεύγοντας εμπόδια.
* **Δίκτυα και επικοινωνίες**, όπου χρησιμοποιούνται αλγόριθμοι για τη βελτιστοποίηση της δρομολόγησης πακέτων δεδομένων.

Η επιλογή της σωστής μεθόδου αναζήτησης είναι κρίσιμη, καθώς επηρεάζει τόσο την **ακρίβεια** όσο και τον **χρόνο εκτέλεσης** της διαδικασίας εύρεσης διαδρομής.

#### 2.3 **Τι είναι οι ευριστικοί αλγόριθμοι**

Οι **ευριστικοί αλγόριθμοι** είναι μέθοδοι αναζήτησης και επίλυσης προβλημάτων που χρησιμοποιούν **ευρετικές συναρτήσεις** για να καθοδηγήσουν την αναζήτηση προς την πιο υποσχόμενη κατεύθυνση.

Σε αντίθεση με τους **αποδεδειγμένα βέλτιστους αλγορίθμους** (όπως ο Dijkstra), οι ευριστικοί αλγόριθμοι επιδιώκουν να εξισορροπήσουν την **ακρίβεια** και την **υπολογιστική απόδοση**. Με άλλα λόγια, δεν εγγυώνται πάντα τη βέλτιστη λύση, αλλά την βρίσκουν **γρηγορότερα** από μεθόδους εξαντλητικής αναζήτησης.

Μια **ευρετική συνάρτηση** είναι ένας μαθηματικός τύπος που εκτιμά το κόστος μετάβασης από μια τρέχουσα κατάσταση στην τελική κατάσταση. Καλά σχεδιασμένες ευρετικές συναρτήσεις μπορούν να βελτιώσουν δραστικά την αποδοτικότητα της αναζήτησης.

#### **2.4 Γενική αρχή λειτουργίας των αλγορίθμων αναζήτησης**

Οι αλγόριθμοι αναζήτησης διαδρομής λειτουργούν σε περιβάλλοντα που μπορούν να αναπαρασταθούν ως **γραφικές δομές (graphs)**.

Γενικά, η διαδικασία περιλαμβάνει τα εξής στάδια:

1. **Ορισμός του προβλήματος** (αρχικό σημείο, στόχος, πιθανά μονοπάτια).
2. **Αναπαράσταση του περιβάλλοντος** ως γράφημα με κόμβους και ακμές.
3. **Εφαρμογή μιας στρατηγικής αναζήτησης** (π.χ. BFS, DFS, A\*).
4. **Αναγνώριση του βέλτιστου μονοπατιού** με βάση κάποιο κριτήριο (κόστος, απόσταση, χρόνος).
5. **Επιστροφή της διαδρομής στον χρήστη ή στον αυτοματοποιημένο πράκτορα (agent)**.

Υπάρχουν δύο κύριες προσεγγίσεις αναζήτησης:

* **Άπληστη αναζήτηση (Greedy Search):** Βασίζεται αποκλειστικά στην ευρετική συνάρτηση και επιλέγει τον κόμβο με το χαμηλότερο εκτιμώμενο κόστος προς τον στόχο.
* **Βέλτιστη αναζήτηση (Optimal Search):** Συνδυάζει πραγματικό κόστος και ευρετική εκτίμηση, όπως κάνει ο A\*.

#### **Κεφάλαιο 3 Εισαγωγή στον αλγόριθμο A**\*

Ο A\* είναι ένας από τους πιο διάσημους και αποτελεσματικούς **ευριστικούς αλγορίθμους αναζήτησης**, ο οποίος εγγυάται την εύρεση του **βέλτιστου μονοπατιού**. Σε αυτή την ενότητα, θα αναλύσουμε τη βασική του ιδέα, τη μαθηματική του θεμελίωση, καθώς και μια λεπτομερή παρουσίαση της υλοποίησης του σε **JavaScript** και **p5.js**.

Η βασική ιδέα του A\* είναι ο υπολογισμός μιας **συνδυαστικής συνάρτησης κόστους**:

*f(n)=g(n)+h(n)*

όπου:

* g(n) = το κόστος μετάβασης από την αρχική θέση στον κόμβο n.
* h(n) = η ευρετική εκτίμηση του κόστους από τον κόμβο n μέχρι τον στόχο.

Ο αλγόριθμος προχωρά προς τον στόχο επεκτείνοντας τον κόμβο με τη μικρότερη τιμή f(n), επιτυγχάνοντας έτσι την **ισορροπία μεταξύ ακρίβειας και απόδοσης**.

*3.1 Σύγκριση A\* με άλλες μεθόδους αναζήτησης*

Ο A\* συγκρίνεται συχνά με άλλους αλγόριθμους αναζήτησης:

| **Αλγόριθμος** | **Πλεονεκτήματα** | **Μειονεκτήματα** |
| --- | --- | --- |
| **Breadth-First Search (BFS)** | Εγγυάται τη μικρότερη διαδρομή σε μη σταθμισμένα γραφήματα | Υπερβολική κατανάλωση μνήμης |
| **Depth-First Search (DFS)** | Απλός στην υλοποίηση, λειτουργεί καλά σε δέντρα | Μπορεί να χαθεί σε ατέρμονους κύκλους |
| **A**\* | Βρίσκει πάντα τη βέλτιστη διαδρομή, συνδυάζει ταχύτητα και ακρίβεια | Υψηλότερη κατανάλωση μνήμης και υπολογιστικής ισχύος |

Σε σύγκριση με το **BFS**, ο **A**\* είναι πιο αποδοτικός καθώς λαμβάνει υπόψη του το κόστος της διαδρομής και δεν επεκτείνει αχρείαστα κόμβους. Αντίθετα με το **DFS**, ο **A**\* δεν κινείται τυφλά μέσα στο γράφημα, αλλά χρησιμοποιεί ευρετικές συναρτήσεις για να κατευθύνει την αναζήτηση προς τη σωστή κατεύθυνση, εξοικονομώντας χρόνο και πόρους.

Ο αλγόριθμος A\* ξεχωρίζει επειδή καταφέρνει να εξισορροπήσει την ακρίβεια και την αποδοτικότητα στην αναζήτηση διαδρομής.

* Χρησιμοποιεί Ευρετική Στρατηγική  
  Αντί να ψάχνει τυφλά, ο A\* καθοδηγείται από μια ευριστική συνάρτηση (h(n)) που προβλέπει πόσο κοντά βρίσκεται στον στόχο.
* Συνδυάζει το Κόστος και την Απόσταση  
  Λαμβάνει υπόψη τόσο το πραγματικό κόστος μετάβασης (g(n)) όσο και την ευριστική εκτίμηση (h(n)), επιτρέποντας μια ισορροπημένη αναζήτηση:

*f(n)=g(n)+h(n)*

Αυτό τον καθιστά πιο αποδοτικό από το BFS (που αναζητά εξαντλητικά) και πιο βέλτιστο από το DFS (που μπορεί να ακολουθήσει λάθος μονοπάτια).

* Προσαρμόζεται σε Δυναμικά Περιβάλλοντα  
  Ο A\* μπορεί να υπολογίσει διαδρομές εκ νέου όταν αλλάζουν οι συνθήκες, καθιστώντας τον ιδανικό για ρομπότ, βιντεοπαιχνίδια και GPS.
* Υποστηρίζει Διαφορετικές Ευριστικές Συναρτήσεις  
  Ανάλογα με την εφαρμογή, μπορεί να χρησιμοποιηθεί Ευκλείδεια, Manhattan ή άλλη ευριστική συνάρτηση για καλύτερη απόδοση.

Συμπερασματικά το πρόβλημα της αναζήτησης διαδρομής είναι σύνθετο, αλλά ο αλγόριθμος A\* προσφέρει μια ισορροπημένη λύση ανάμεσα στην ταχύτητα και την ακρίβεια, υπερτερώντας των απλούστερων αλγορίθμων όπως το DFS και BFS. Οι ευρετικές στρατηγικές του επιτρέπουν να προσαρμόζεται σε διαφορετικά περιβάλλοντα, κάνοντάς τον ιδανικό για πραγματικές εφαρμογές.

*3.2 Βασική Ιδέα και Ψευδοκώδικας του A\**

Ο A\* βασίζεται στην ιδέα ότι κάθε κόμβος έχει ένα συνολικό κόστος **f(n)**, το οποίο αποτελείται από δύο τιμές:

* **g(n):** Το πραγματικό κόστος από την αρχή μέχρι τον κόμβο **n**
* **h(n):** Η εκτιμώμενη (ευριστική) απόσταση από τον κόμβο **n** μέχρι τον στόχο

Ο αλγόριθμος διατηρεί δύο λίστες:

* **Open Set**: Οι κόμβοι που πρόκειται να εξεταστούν.
* **Closed Set**: Οι κόμβοι που έχουν ήδη εξεταστεί.

Ο παρακάτω ψευδοκώδικας περιγράφει τον αλγόριθμο:



Πηγή Wikipedia: [Bibliographic details for "A\* search algorithm"](https://en.wikipedia.org/w/index.php?title=A*_search_algorithm&oldid=1273031415)

Σε αυτόν τον ψευδοκώδικα, εάν ένας κόμβος προσεγγιστεί από ένα μονοπάτι, αφαιρεθεί από το openSet και στη συνέχεια φτάσει από μια φθηνότερη διαδρομή, θα προστεθεί ξανά στο openSet. Αυτό είναι απαραίτητο για να διασφαλιστεί ότι η διαδρομή που επιστρέφεται είναι βέλτιστη εάν η ευρετική συνάρτηση είναι αποδεκτή αλλά όχι συνεπής. Εάν το ευρετικό είναι συνεπές, όταν ένας κόμβος αφαιρείται από το openSet, η διαδρομή προς αυτόν είναι εγγυημένη ότι είναι βέλτιστη.

*3.3 Η Έννοια του Κόστους Διαδρομής (g, h, f functions)*

Ο **A\*** βασίζεται στον υπολογισμό τριών συναρτήσεων για να καθορίσει ποιον κόμβο θα εξετάσει στη συνέχεια.

**🡪 g(n) – Κόστος μετάβασης**  
Το κόστος g(n) υπολογίζεται με βάση την απόσταση από τον κόμβο εκκίνησης μέχρι το σημείο **n**.

* Σε **ορθογώνιο πλέγμα**, η απόσταση κάθε μετακίνησης είναι **1** (ή √2 για διαγώνιες κινήσεις αν επιτρέπονται).
* Η νέα τιμή g(n) ενημερώνεται ως εξής:

g(n)=g(parent)+costg(n) = g(parent) + costg(n)=g(parent)+cost

🡪 **h(n) – Ευριστική συνάρτηση (Heuristic Function)**  
Είναι μια εκτίμηση της απόστασης του κόμβου **n** από τον στόχο. Χρησιμοποιούνται διάφορες προσεγγίσεις:

* **Manhattan Distance** (|x1-x2| + |y1-y2|) αν επιτρέπονται μόνο οριζόντιες/κάθετες κινήσεις.
* **Euclidean Distance** αν επιτρέπονται διαγώνιες κινήσεις.

🡪 **f(n) – Συνολικό κόστος**  
Ο A\* εξετάζει κάθε κόμβο με βάση τη συνάρτηση:

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

Ο κόμβος με το μικρότερο f(n) προτιμάται για αναζήτηση.

*3.4 Υλοποίηση με Χρήση JavaScript και p5.js*

Παρακάτω παρουσιάζεται η υλοποίηση του αλγορίθμου A\* σε **JavaScript με p5.js**.

* **Δήλωση μεταβλητών**

A screen shot of a computer

Description automatically generated

* **Ορισμός των συναρτήσεων κόστους Συνάρτηση *heuristic()***

A computer screen with white text

Description automatically generated

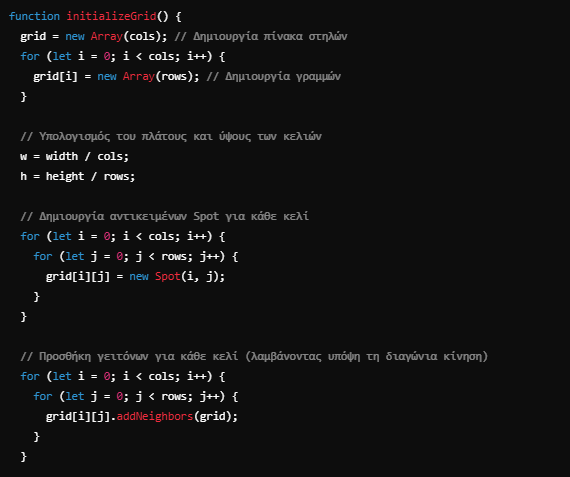
* **Αρχικοποίηση του Πλέγματος Συνάρτηση *setup()***

A screen shot of a computer program

Description automatically generated

* **Δημιουργία πλέγματος Συνάρτηση *initializeGrid()***

Η συνάρτηση ***initializeGrid()*** δημιουργεί το πλέγμα (grid), ορίζει το μέγεθος των κελιών και προσθέτει τους γειτονικούς κόμβους.



A screenshot of a computer screen

Description automatically generated

Η ***initializeGrid()*** εκτελείται όταν αρχίζει ή επαναφέρεται η αναζήτηση διαδρομής. Με αποτέλεσμα η συνάρτηση αυτή να πραγματοποιεί τις παρακάτω ενέργειες:

* + Δημιουργεί το πλέγμα **cols x rows**
  + Ορίζει το **start** και το **end**
  + Αποτρέπει τα εμπόδια στο σημείο εκκίνησης και τερματισμού
  + Προσθέτει τους γείτονες των κόμβων
  + Επαναφέρει τα σύνολα **openSet** και **closedSet**
  + Ξεκινά ξανά την αναζήτηση καλώντας **loop()**
* **Εκτέλεση του A\* συνάρτηση *Draw()***

A computer screen with white text

Description automatically generated

Η υλοποίηση με **JavaScript** και **p5.js** δείχνει πώς ο A\* μπορεί να εφαρμοστεί για αναζήτηση διαδρομής σε ένα δυναμικό πλέγμα. Ο κώδικας περιλαμβάνει όλα τα απαραίτητα στοιχεία: **υπολογισμό κόστους, επεξεργασία γειτονικών κόμβων και ενημέρωση του μονοπατιού**.

*3.5 Δημιουργία html σελίδας για οπτικοποίηση του A\**

Για την οπτικοποίηση του αλγορίθμου A\*, έχει δημιουργηθεί μια ξεχωριστή σελίδα astar.html, η οποία περιέχει το απαραίτητο περιβάλλον για την εκτέλεση του αλγορίθμου.

Η **astar.html** περιλαμβάνει:

* **Βιβλιοθήκες p5.js** για τη σχεδίαση του καμβά
* **Εξωτερικά αρχεία JavaScript** με τις συναρτήσεις του αλγορίθμου (**astarFunctions.js**, **spot.js**)
* **Ρυθμίσεις ελέγχου** (πλήκτρο για νέα δημιουργία πλέγματος, επιλογή διαγώνιας κίνησης, ποσοστό εμποδίων)
* **JavaScript κώδικα** που διαχειρίζεται το πλέγμα και επικοινωνεί με την index.html

**Κύρια σημεία του κώδικα της astar.html:**

* Αρχικοποίηση των μεταβλητών **cols, rows, grid, openSet, closedSet, path, start, end**
* Συνάρτηση initializeGrid() για τη δημιουργία του πλέγματος
* Δυνατότητα **διαγώνιας κίνησης** με το diagonalToggle
* **Διαδραστικός έλεγχος** εμποδίων μέσω του wallSlider
* **Λήψη μηνυμάτων** από τη index.html μέσω του window.addEventListener("message", κλπ.)

Η **astar.html** καλείται μέσα στην **index.html** μέσω ενός **iframe**, επιτρέποντας τη διαχείριση του αλγορίθμου **A\*** από το κύριο interface της εφαρμογής.

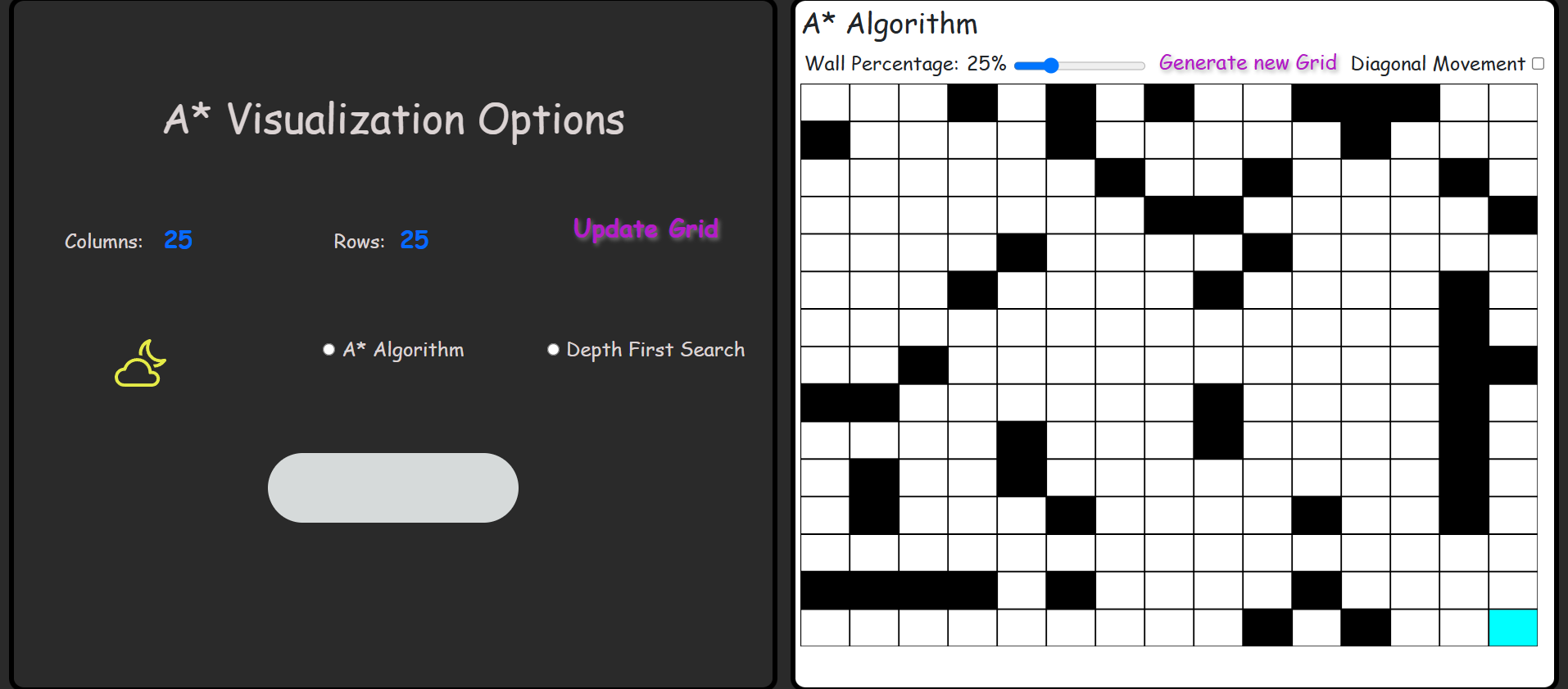


Με αυτόν τον τρόπο, η **astar.html** εκτελείται ως ξεχωριστή οντότητα, αλλά μπορεί να επικοινωνεί με την **index.html** μέσω window.postMessage(), επιτρέποντας την αλλαγή του μεγέθους του πλέγματος ή άλλων παραμέτρων.

Συμπερασματικά η **astar.html** αποτελεί τη βασική σελίδα οπτικοποίησης του **A\***, ενώ η **index.html** λειτουργεί ως κεντρικό περιβάλλον ελέγχου μέσω iframe. Αυτή η διάκριση επιτρέπει την καλύτερη οργάνωση του κώδικα και την εύκολη ενσωμάτωση διαφορετικών αλγορίθμων.

*3.6 Ανάλυση ιστοσελίδας / Εμπειρία Χρήστη*

Παρακάτω παρουσιάζεται η αρχική οθόνη κατά την είσοδο του χρήστη στην ιστοσελίδα/ ανοίγοντας την index.html.



(Κεντρική οθόνη με μαύρο θέμα)

* Στα αριστερά της οθόνης παρουσιάζονται οι επιλογές όπου δίνονται στο

χρήστη για χειρισμό της ιστοσελίδας καθώς και του αλγορίθμου.

Στη πρώτη σειρά μας δίνεται η επιλογή *Columns* (Στήλες) και *Rows* (Σειρές) όπου έχουν αρχικοποιηθεί σε 25 επί 25, ωστόσο μπορούμε να επιλέξουμε μεταξύ ενός εύρους 10 έως 50. Αφότου ο χρήστης ορίσει την αναλογία της επιλογής του, τότε επιλέγει το κουμπί *Update Grid* για ανανεωθεί το πλέγμα στην οθόνη.

Στη δεύτερη σειρά μας δίνεται η επιλογή αλλαγής θέματος σε λευκό επιλέγοντας το αντίστοιχο κουμπί (Σύννεφο).

A screenshot of a crossword puzzle

Description automatically generated

(Κεντρική οθόνη με λευκό θέμα)

Στη συνέχεια μπορούμε να επιλέξουμε τον αλγόριθμο όπου θέλουμε να χρησιμοποιήσουμε. Εδώ θα πρέπει να αναφερθεί πως στα πλαίσια της εργασίας πραγματοποιήθηκε και υλοποίηση αλγορίθμου *Depth First Search (DFS)* με γνώμονα τη σύγκριση του με τον αλγόριθμο *A\**.

Τέλος μας δίνεται ένα textarea στο οποίο αναγράφεται το μήνυμα “Algorithm completed successfully!” όταν ο αλγόριθμος κατάφερε να φτάσει το στο σημείο όπου επιλέξαμε ως στόχο και “No solution!” σε περίπτωση όπου ο αλγόριθμο δε μπόρεσε να φτάσει το στόχο.

* Στα βλέπουμε τη σελίδα *astar.html* *(πλέγμα)* όπου καλούμε μέσω του iframe.

A screenshot of a crossword puzzle

Description automatically generated

Αρχικά με μπλε χρώμα μας παρουσιάζεται το τελικό σημείο όπου έπειτα με ο χρήστης μπορεί να επιλέξει το τελικό προορισμό της επιλογής του.

Έπειτα πάνω από το πλέγμα παρουσιάζονται οι παρακάτω ιδιότητες:

🡪 Wall percentage: Με την επιλογή αυτή αναπαριστούμε τα εμπόδια (μαύρα κουτιά) και διαθέτει ένα slider με εύρος 0-100%.

🡪 Generate new grid: Με το κουμπί αυτό μας δίνεται η δυνατότητα να κάνουμε ανανέωση του grid με αποτέλεσμα να αλλάζει η θέσεις όπου έχουν τοποθετηθεί τα εμπόδια (μαύρα κουτιά) στο πλέγμα.

🡪 Diagonal Movement: Με την επιλογή αυτή επιτρέπουμε στον αλγόριθμο την διαγώνια κίνηση.

Στη παρακάτω εικόνα μπορούμε να δούμε ένα στιγμιότυπο από τη χρήση του προγράμματος με τον αλγόριθμο να φτάνει στο στόχο όπου έχουμε ορίσει.

A screenshot of a crossword puzzle

Description automatically generated

Κατά την ολοκλήρωση του αλγόριθμου στην οθόνη μας εμφανίζεται με μπλε χρώμα η βέλτιστη διαδρομή όπου εντόπισε ο αλγόριθμος έτσι ώστε να φτάσει στο σημείο στόχο (Το ίδιο ισχύ και για την υλοποίηση του *DFS*).  
  
*3.7 Συμπεράσματα και Μελλοντικές Προεκτάσεις*

Η εργασία αυτή επικεντρώθηκε στην ανάλυση και υλοποίηση του αλγορίθμου **A\*** για την εύρεση βέλτιστης διαδρομής. Ο **A\*** αποδείχθηκε ένας ισχυρός και αποτελεσματικός αλγόριθμος αναζήτησης, ο οποίος συνδυάζει το κόστος της ήδη εξερευνημένης διαδρομής και μια ευριστική εκτίμηση του υπολειπόμενου κόστους για να εξασφαλίσει μια αποδοτική αναζήτηση. Παρακάτω παρατίθενται μερικά πλεονεκτήματα και μειονεκτήματα του αλγόριθμου Α\*.

*3.7.1 Πλεονεκτήματα και Μειονεκτήματα του Α\**

**Πλεονεκτήματα**

🡪 **Εγγυημένη βέλτιστη λύση**: Εάν η ευριστική είναι συνεπής και αποδεκτή, ο **A\*** βρίσκει πάντα τη βέλτιστη διαδρομή.  
🡪 **Προσαρμοστικότητα**: Μπορεί να χρησιμοποιηθεί σε διάφορα περιβάλλοντα   
🡪 **Αποδοτικότητα**: Σε σχέση με απλές αναζητήσεις όπως BFS και DFS, ο **A\*** είναι πολύ πιο γρήγορος και στοχευμένος.

**Μειονεκτήματα/ Περιορισμοί**

🡪 **Υψηλή υπολογιστική πολυπλοκότητα**: Αν το πλέγμα είναι πολύ μεγάλο, η μνήμη που απαιτείται για το openSet μπορεί να αυξηθεί σημαντικά.

🡪 **Εξάρτηση από την ευριστική**: Η απόδοση του αλγορίθμου εξαρτάται από την επιλογή της **h(n)** συνάρτησης.

*3.7.2 Προτάσεις για βελτιώσεις και μελλοντικές έρευνες*

🡪 **Δυναμική αναπροσαρμογή της διαδρομής**: Ο αλγόριθμος θα μπορούσε να επεκταθεί ώστε να προσαρμόζεται σε αλλαγές στο περιβάλλον (π.χ. κινούμενα εμπόδια).  
🡪 **Χρήση πιο αποδοτικών ευριστικών**: Πειραματισμός με άλλες συναρτήσεις **h(n)**, όπως Machine Learning μοντέλα, που προβλέπουν καλύτερες διαδρομές.

🡪 Οπτικοποίηση της διαδρομής όπου διάνυσε ο αλγόριθμος και τα βήματα όπου χρειάστηκε ως ότου φτάσει στο στόχο

🡪 Προσθήκη περαιτέρω εμποδίων με χρήση διαφορετικών βαρών ανά εμπόδιο έτσι ώστε να παρουσιάσουμε μεγαλύτερη ποικιλία εμποδίων και της δυνατότητας του αλγορίθμου.

*3.7.3 Σύνδεσμοι*

Ο αλγόριθμος θα είναι διαθέσιμος για όλους στους παρακάτω συνδέσμου:

* Github link για λήψη του αλγόριθμου: [github.com/mariosch93/A\_\_Star](https://github.com/mariosch93/A__Star)
* Άμεση προβολή του με άνοιγμα μέσω [a-star-marios-chs-projects.vercel.app/](https://a-star-marios-chs-projects.vercel.app/)