Εργασία

Συστήματα Ευφυών Πρακτόρων

Αλέξανδρος Γαϊτάνης, ΑΜ: 63

ΠΜΣ στην Τεχνητή Νοημοσύνη

Εαρινό εξάμηνο 2021 - 2022

Τμήμα Πληροφορικής

Αριστοτέλειο Πανεπιστήμιο Θεσσαλονίκης

**1. Εισαγωγή**

Στην παρούσα εργασία μελετήθηκαν maze solving αλγόριθμοι. Ένα ρομπότ βρίσκεται σε ένα λαβύρινθο και προσπαθεί να αποδράσει χωρίς ξέρει που βρίσκεται. Οι αλγόριθμοι που υλοποιήθηκαν είναι οι εξής: Wall Follower, Random Mouse και Tremaux. Η υλοποίηση έγινε στην πλατφόρμα Jason.

**2. Περιβάλλον**

Ως περιβάλλον χρησιμοποιήθηκαν δύο λαβύρινθοι ο Α και ο Β *(Εικόνα 1, Εικόνα 2)*.

A screenshot of a computer

Description automatically generated with low confidence

Εικόνα : Λαβύρινθος Α

A screenshot of a computer

Description automatically generated with low confidence

Εικόνα 2: Λαβύρινθος Β

Το περιβάλλον είναι ένα grid 20x20 και περιέχει:

* Τοίχους που ζωγραφίζονται με σκούρο χρώμα
* Διαδρόμους που ζωγραφίζονται με ανοιχτό χρώμα
* Μία είσοδο που ζωγραφίζεται με κόκκινο χρώμα
* Μία έξοδο που ζωγραφίζεται με πράσινο χρώμα
* Το ρομπότ που ζωγραφίζεται ως ένας μπλε κύκλος με έναν χαρακτήρα (<, ^, >, v) που δείχνει τον προσανατολισμό του

Στο παράθυρο υπάρχει:

* Ένας slider που ρυθμίζει την ταχύτητα ανανέωσης του περιβάλλοντος
* Ένα label που δείχνει τον αριθμό των κελιών από τα οποία πέρασε το ρομπότ

Το ποιος λαβύρινθος θα χρησιμοποιηθεί ορίζεται στο *maze.mas2j* ως η 1η παράμετρος του *environment: maze.MazeEnv*. Οι τιμές που μπορεί να πάρει είναι *map\_a* ή *map\_b.*

**3. Ενέργειες**

Οι ενέργειες που μπορεί να κάνει το ρομπότ είναι οι εξής:

* *move\_fwd*: Να προχωρήσει μπροστά κατά ένα κελί.
* *turn\_left*: Να στρίψει αριστερά
* *turn\_right*: Να στρίψει δεξιά
* *mark\_cell*: Να μαρκάρει το κελί πάνω στο οποίο βρίσκεται
* *mark\_back\_cell*: Να μαρκάρει το κελί το οποίο βρίσκεται από πίσω του

Οι τελευταίες 2 ενέργειες χρησιμοποιούνται μόνο στον αλγόριθμο Tremaux.

**4. Αντιληπτικά δεδομένα**

Τα αντιληπτικά δεδομένα που έχει διαθέσιμα το ρομπότ είναι τα εξής:

* *cell(exit)*: Το κελί στο οποίο βρίσκεται το ρομπότ είναι η έξοδος
* *cell(front,obstacle), cell(left,obstacle), cell(right,obstacle)*: Το κελί το οποίο βρίσκεται μπροστά/αριστερά/δεξιά του ρομπότ είναι εμπόδιο
* *cell(front,entrance), cell(left,entrance), cell(right,entrance)*: Το κελί το οποίο βρίσκεται μπροστά/αριστερά/δεξιά του ρομπότ είναι η είσοδος
* *cell(front,exit), cell(left,exit), cell(right,exit)*: Το κελί το οποίο βρίσκεται μπροστά/αριστερά/δεξιά του ρομπότ είναι η έξοδος
* *cell(front,marked\_once), cell(left,marked\_once), cell(right,marked\_once)*: Το κελί το οποίο βρίσκεται μπροστά/αριστερά/δεξιά του ρομπότ είναι μαρκαρισμένο μια φορά
* *cell(front,marked\_twice), cell(left,marked\_twice), cell(right,marked\_twice)*: Το κελί το οποίο βρίσκεται μπροστά/αριστερά/δεξιά του ρομπότ είναι μαρκαρισμένο δύο φορές

Τα αντιληπτικά δεδομένα που περιέχουν τα marked\_once και marked\_twice χρησιμοποιούνται μόνο στον αλγόριθμο Tremaux.

**5. Κανόνες**

Για ευκολία στην υλοποίηση των πλάνων χρησιμοποιήθηκαν οι παρακάτω κανόνες:

Text

Description automatically generated

1. *left\_cell\_free*: Αληθές όταν μόνο το αριστερό κελί είναι ελεύθερο
2. *front\_cell\_free*: Αληθές όταν μόνο το μπροστινό κελί είναι ελεύθερο
3. *right\_cell\_free*: Αληθές όταν μόνο το δεξιό κελί είναι ελεύθερο
4. *left\_front\_right\_cell\_free*: Αληθές όταν το αριστερό, το μπροστινό και το δεξιό κελί είναι ελεύθερο
5. *left\_front\_cell\_free*: Αληθές όταν μόνο το αριστερό και το μπροστινό κελί είναι ελεύθερο
6. *front\_right\_cell\_free*: Αληθές όταν μόνο το μπροστινό και το δεξιό κελί είναι ελεύθερο
7. *left\_right\_cell\_free*: Αληθές όταν μόνο το αριστερό και το δεξιό κελί είναι ελεύθερο
8. *in\_junction*: Αληθές όταν ισχύουν ένα από τα 5, 6, 7, δηλαδή το ρομπότ βρίσκεται σε διασταύρωση

Ένα κελί θεωρείται ελεύθερο όταν δεν έχει εμπόδιο, δηλαδή τοίχο, δεν είναι η είσοδος και δεν έχει μαρκαριστεί δύο φορές στον αλγόριθμο Tremaux.

**6. Αλγόριθμοι**

Το ποιος αλγόριθμος θα χρησιμοποιηθεί ορίζεται στο *maze.mas2j* ως η 2η παράμετρος του *environment: maze.MazeEnv*. Οι τιμές που μπορεί να πάρει είναι *wall\_follower, random\_mouse* ή *tremaux.*

**6.1 Wall Follower**

Σε αυτόν τον αλγόριθμο το ρομπότ προσπαθεί να έχει στα δεξιά του ή στα αριστερά του τοίχο ώσπου να βρει τον στόχο. Είναι αποδοτικός σε περιπτώσεις που ο στόχος είναι δίπλα σε τοίχο, όπως είναι η έξοδος στο περιβάλλον που χρησιμοποιήθηκε. Υπάρχουν δύο τύποι του αλγορίθμου, ο δεξιόστροφος και ο αριστερόστροφος. Ανάλογα με το ποιον τύπο επιλέγουμε μπορεί να προκύψουν πολύ διαφορετικά αποτελέσματα. Στην συγκεκριμένη εργασία χρησιμοποιήθηκε ο δεξιόστροφος. Όταν το ρομπότ βρίσκεται σε διασταύρωση στρίβει δεξιά, αν υπάρχει δεξιά του εμπόδιο τότε προχωράει μπροστά, και αν υπάρχει μπροστά και δεξιά του εμπόδιο τότε στρίβει αριστερά.

Παρακάτω φαίνεται ένας πίνακας που αντιστοιχεί τα αντιληπτικά δεδομένα του ρομπότ με τις ενέργειες του:

|  |  |  |
| --- | --- | --- |
| Μπροστά | Δεξιά | Ενέργεια |
| 0 | 0 | *turn\_right, move\_fwd* |
| 0 | 1 | *move\_fwd* |
| 1 | 0 | *turn\_right, move\_fwd* |
| 1 | 1 | *turn\_left* |

0 = Δεν εντοπίστηκε εμπόδιο, 1 = Εντοπίστηκε εμπόδιο

Ο αλγόριθμος αυτός δεν χρησιμοποιεί μνήμη και δεν είναι εγγυημένο ότι θα βρει πάντα την έξοδο. Στην περίπτωση που ο λαβύρινθος είναι απλά συνδεδεμένος, δηλαδή όλοι οι τοίχοι είναι συνδεδεμένοι μεταξύ τους ή με το εξωτερικό του λαβυρίνθου *(Εικόνα 1)*, τότε οι τοίχοι μπορούν να παραμορφωθούν σε έναν κύκλο, οπότε το πρόβλημα απλοποιείται στο να κινηθεί το ρομπότ γύρω από έναν κύκλο. Αν ο λαβύρινθος δεν είναι απλά συνδεδεμένος, δηλαδή η είσοδος ή η έξοδος βρίσκεται κάπου στο εσωτερικό του και είναι περικυκλωμένη από κυκλικό διάδρομο *(Εικόνα 2)*, τότε το ρομπότ μπορεί να μην φτάσει στον στόχο. Τότε επιστρέφει στην είσοδο και ξεκινά από την αρχή τις ίδιες κινήσεις χωρίς να βρίσκει ποτέ τον στόχο του.

**6.2 Random mouse**