Πανεπιστήμιο Πειραιώς
Σχολή Τεχνολογιών Πληροφορικής και Τηλεπικοινωνίων
Τμήμα Ψηφιακών Συστημάτων

Επίπεδο: Προπτυχιακό Πρόγραμμα Σπουδών

Μάθημα – Συστήματα Ευφυών Πρακτόρων

Τίτλος - 1η ομαδική εργασία 2 ατόμων (2023-2024)

Επιβλέπον Καθηγητής: Γεώργιος Βούρος, Ναταλία Κόλιου

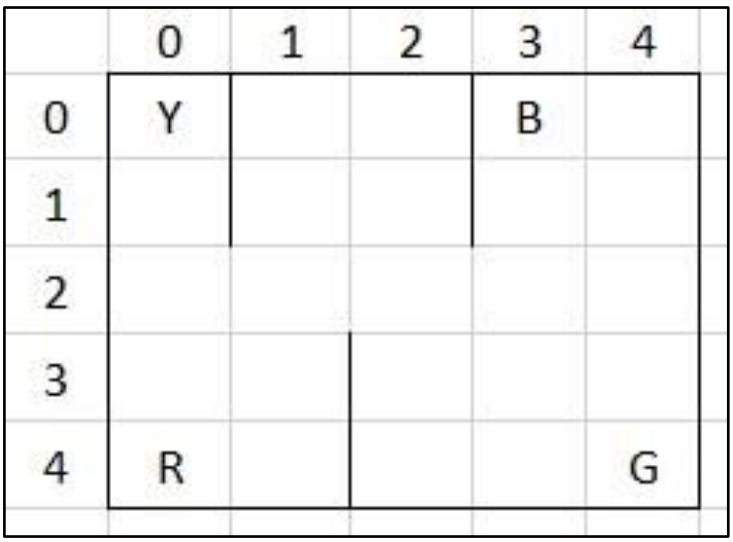
Ονοματεπώνυμο	E-mail	A.M.
Απόστολος Χασιώτης	apostolis10@pm.me	E20183
Σωτήριος Δημητρακουλάκος	sdimitrakoulakos@gmail.com	E20040

Πειραιάς
11/12/2024

Abstract

Περιεχόμενα

Αρχιτεκτονική



Λειτουργίες

MarsModel consructor

Θέτει τις αρχικές τοποθεσίες του agent 0 (taxi), του agent 1 (destination) και κάνει τις διαδικασίες αρχικοποίησης defineCosts(), createWalls(), simulateWalls().

updatePercepts

Κάθε φορά που εκτελείται, προσθέτει perception στο μοντέλο για τις τοποθεσίες των δύο πρακτόρων (taxi/0 και d/1(destination)), αλλά και το αν ισχύουν τα percepts gr1 (customer(taxi) – ο πελάτης είναι στο ταξί) και το percept gd (customer(d) – ο πελάτης είναι στο destination).

generateRandomCustomer

Η συνάρτηση αυτή δημιουργεί κάθε φορά που εκτελείται έναν πελάτη, στον οποίο θα αντιστοιχεί ένα τυχαίο location και ένα τυχαίο destination (τυχαία επιλογή ανάμεσα στις τοποθεσίες R, Y, G, B στον χάρτη). Καλεί την getRandomLocation (με όρισμα 1 για να επιλέξει μία από τις 4 τοποθεσίες-γράμματα / επεξήγηση στο κομμάτι για την αντίστοιχη μέθοδο) και αναθέτει τυχαίες συντεταγμένες λοιπόν στην κάθε μεταβλητή. Ύστερα, ελέγχει από την λίστα με τους υπάρχοντες customers, αν η τοποθεσία καποιανού από αυτούς τους πελάτες, είναι ταυτόσημη με την τοποθεσία που μόλις πήραμε τυχαία. Αν ναι, τότε κάνει return και δεν δημιουργείται πελάτης σε αυτήν την τοποθεσία, αφού δεν πρέπει να υπάρχει παραπάνω από ένας πελάτης τη φορά, σε ένα συγκεκριμένο κελί-τοποθεσία. Αν όμως δεν υπάρχει ήδη πελάτης σε αυτήν την τοποθεσία, απλά βάζει την οντότητα του πελάτη στον χάρτη και καλεί την addCustomerToWorld, η οποία στην ουσία ανανεώνει την λίστα των customers και της προσθέτει τον καινούργιο πελάτη με τις συντεταγμένες του.

checkForWall

Παίρνει από το walls για τον κάθε τοίχο, το πού είναι ο τοίχος και το direction (E, K, A, D) προς στο οποίο υπάρχει τοίχος. Αν σε ένα από τους τοίχους, τα location και direction (μεριά του κελιού που υπάρχει ο τοίχος) ταυτίζονται με τα location και direction τους ταξί, τότε σημαίνει ότι πρόκειται να κοπανήσει το ταξί σε τοίχο. Ελέγχει για ποια ακριβώς κατεύθυνση έχουμε (E, K, A, D) και βάζει στον costs, στις αντίστοιχες συντεταγμένες που βρίσκεται το wall (x, y) και για την αντίστοιχη μεριά d (0,1,2,3 / E, K, A, D), το αντίστοιχο κόστος της κατεύθυνσης αυτού του κελιού, που θα είναι 101. Έτσι, ο πράκτορας μας μαθαίνει πού υπάρχουν τοίχοι. Π.χ. αν το κελί [0][0] έχει τοίχο στα δεξιά του, το κόστος του θα

βρίσκεται στον costs σε μορφή int [0][0][3] = 101, όπου το 3 αντιστοιχεί σε D (δεξιά).

createWalls

Η συνάρτηση αυτή πολύ απλά κάνει hard code στη λίστα walls που υπάρχει στο περιβάλλον μας, όλα τα κελιά που έχουν τοίχους γύρω τους, καθώς και την κατεύθυνση στην οποία υπάρχει ο τοίχους (E(επάνω), K(κάτω), D(δεξιά), A(αριστερά)).

defineCosts

Η συνάρτηση αυτή όταν εκτελεστεί, αρχικοποιεί όλα τα elements (ένα για κάθε κελί στον χάρτη).

getRandomLocation

Η συνάρτηση αυτή δέχεται σαν όρισμα έναν αριθμό (0 ή 1) που αντιπροσωπεύει τον τύπο της τυχαίας τοποθεσίας που θα παραχθεί. Αν το type είναι 1, τότε επιλέγεται τυχαία μία από τις 4 τοποθεσίες με γράμματα στον πίνακα (R, Y, G, B), αλλιώς αν είναι 0, επιλέγεται μια τυχαία τοποθεσία γενικά στον κόσμο του ταξί. Αυτή η διάκριση γίνεται διότι την πρώτη φορά που θα εμφανιστεί το ταξί, πρέπει να εμφανιστεί σε μια τυχαία τοποθεσία γενικά, ενώ η τοποθεσία του customer και του destination του όταν δημιουργηθεί, θα πρέπει να είναι μία από τις 4 τοποθεσίες με γράμματα (R, Y, G, B).

LocationToArray

Μέθοδος που δέχεται ένα όρισμα τύπου Location και το μετατρέπει σε Array ακεραίων, όπου το πρώτο στοιχείο της είναι η συντεταγμένη x και το δεύτερο η συντεταγμένη y. Επιστρέφει το αποτέλεσμα.

manhattan

Συνάρτηση που δέχεται 2 Array ακεραίων (το ένα με συντεταγμένες για το αρχικό position και το άλλο για το τελικό), υπολογίζει την απόσταση Μανχάταν μεταξύ αυτών των δύο κελιών και την επιστρέφει.

getNextBestCustomer

Η μέθοδος αυτή έχει ως σκοπό να βρίσκει ποιος από τους πελάτες βρίσκεται εκείνη την στιγμή πιο κοντά στο ταξί και να τον κάνει current customer. Αρχικά, παίρνει την τωρινή τοποθεσία του πράκτορα ταξί και αρχικοποιεί τα δεδομένα του ελαχίστου (ελάχιστη τοποθεσία και αντίστοιχος πελάτης που την έχει (συντεταγμένες αυτού)). Παίρνοντας έναν-έναν τους πελάτες και υπολογίζοντας την απόσταση Μανχάταν του ταξί από την τοποθεσία του, βρίσκει εν τέλει τον πελάτη με την μικρότερη απόσταση από το ταξί, με τον απλό, κλασσικό αλγόριθμο εύρεσης ελαχίστου. Αν τώρα βρεθούν αποστάσεις που έχουν και οι δύο την ελάχιστη τιμή, ο αλγόριθμος θα κοιτάξει ένα βήμα παραπέρα (αυξάνεται κατά ένα το βάθος της αναζήτησης) και θα λάβει υπόψιν ποιος από τους δύο πελάτες έχει το λιγότερο μακρινό destination, από το location του. Έτσι υπολογίζει καινούργιες Μανχάταν αποστάσεις και διαλέξει σαν min_customer, αυτόν που έχει την μικρότερη. Αν και αυτές αποστάσεις είναι ίδιες, διαλέγει αυθαίρετα ένα από τους 2. Αφού βρέθηκε ποιος θα είναι ο επόμενος πελάτης που θα εξυπηρετηθεί (και αυτός δεν είναι null σε περίπτωση που δεν υπήρχε καν πελάτης να εξυπηρετηθεί στην customers), τότε θέτεται ως current_customer αυτός ο πελάτης (min_customer) και ο πράκτορας 1 (d/destination), τοποθετείται στις συντεταγμένες του καταχωρημένου προορισμού που έχει ο πελάτης που μόλις επιλέχθηκε. Αν επιλέχθηκε όντως πελάτης προς εξυπηρέτηση, γίνεται return true, αλλιώς false.

performActionToArray

Παίρνει ως όρισμα ένα array με δύο ακέραιους, όπου ο καθένας αντιστοιχεί σε μια από τις συντεταγμένες x, y και έναν χαρακτήρα. Άρα το array περιέχει μια τοποθεσία για ένα κελί (σε νούμερα) πάνω στον χάρτη μας και έναν χαρακτήρα που υποδηλώνει στην ουσία ποιον γείτονα αυτού του κελιού θέλουμε να επιστρέψουμε μετά (τον πάνω (E), τον κάτω (K), τον δεξιά (D) ή τον αριστερά (A)). Έτσι δημιουργεί ένα αντίγραφο αυτής της τοποθεσίας και αν ο χαρακτήρας που δεχτήκαμε είναι E, θα μειώσει το y κατά 1 (η αρίθμηση του πίνακα πάει από 0 έως 4, από πάνω προς τα κάτω), αν δεχτεί K θα το αυξήσει κατά 1, αν δεχθεί D θα αυξήσει το x του κατά 1 και αν δεχθεί A θα το μειώσει. Τέλος, επιστρέφει το μετατοπισμένο αντίγραφο.

nextSlot

Αν δεν υπάρχει current_customer (είναι null), τότε καλείται η συνάρτηση που βρίσκει τον πελάτη που πρέπει να εξυπηρετήσει τώρα και τον κάνει current_customer. Αρχικοποιεί την αρχική τοποθεσία του ταξί στην r1 και την αποθηκεύει σε μια integer list, με όνομα old_location. Καλείται η μέθοδος που επιστρέφει μια λίστα με πρώτο στοιχείο της το σημείο που είναι το βέλτιστο να πάει μετά το ταξί (το βάζει στο res_pos / περιέχει τις συντεταγμένες του), δεύτερο στοιχείο την κατεύθυνση που βρίσκεται αυτό το σημείο σε σχέση με το προηγούμενο (το βάζει στο move) και σαν τρίτο το κόστος της μετάβασης στο σημείο αυτό. Το αντικείμενο r1 τύπου Location πια παίρνει σαν συντεταγμένες αυτές που περιέχει το res_po και τοποθετείται το ταξί στην τοποθεσία r1 (άρα στο επόμενο καλύτερο κελί σε σχέση με αυτό που ήταν πριν). Καλείται μια συνάρτηση (εξηγείται παραπάνω) για να γίνει ο έλεγχος για το αν ανάμεσα στη μετάβαση αυτή υπήρχε τοίχος. Αν ναι ο πράκτορας επιστρέφει στο old_location.

Τελικός έλεγχος είναι εάν η τοποθεσία του ταξί ταυτίζεται με την τοποθεσία του πελάτη. Σε τέτοια περίπτωση πρέπει να ελέγξουμε μερικές περιπτώσεις. Εάν λοιπόν βρεθεί καταχωρημένη διαδρομή, αυτό σημαίνει πως φθάσαμε στον προορισμό μας χρησιμοποιώντας την ή δημιουργήσαμε την διαδρομή οδηγούμενοι στα τυφλά βάση απόστασης Μανχάταν. Εάν πρόκειται για δημιουργία νέας διαδρομής, τότε η εν λόγω διαδρομή θα ενταχθεί στο dictionary για να χρησιμοποιηθεί στο μέλλον. Τελικό βήμα είναι η εξαγωγή του τωρινού πελάτη από την λίστα των πελατών, ώστε την επόμενη φορά να είναι σε θέση να επιλέξει κάποιον νέο.
Καλείται από την executableAction.

pickCust

Η μέθοδος αυτή εκτελεί την ενέργεια «εκφόρτωσης» του πελάτη στο ταξί. Αφού ελέγξει αν ο agent 0 (ταξί) έχει τον πελάτη, θα επιχειρήσει το πολύ MErr φορές και εν τέλει θα πάρει τον πελάτη και η boolean μεταβλητή που δείχνει αν το ταξί έχει τον πελάτη, θα γίνει true.Καλείται από την executableAction.

doBestNextMove

Αρχικό βήμα είναι ο έλεγχος εάν υπάρχει διαδρομή στην μνήμη ώστε να εκτελεστεί αυτομάτως το επόμενο βήμα. Βάση του τελευταίου χαρακτήρα ξέρουμε εάν πρέπει να διαβάσουμε ή να γράψουμε (τελευταίος χαρακτήρας ’.’).
Διαφορετικά πρέπει να εκτελεστεί η εύρεση όλων των νέων τοποθεσιών του ταξί από τις 4 δυνατές κινήσεις (Ε, Κ, Α, Δ).
Έπειτα, να υπολογιστούν όλες οι αποστάσεις Μανχάτταν που υπολογίστηκαν . Σε αυτές τις αποστάσεις προστίθεται το κόστος μετάβασης που θα είναι είτε 1 είτε 101 (τοίχος) είτε ένα νέο κόστος που ορίστηκε για ένα κελί και μία διαδρομή, για την συγκεκριμένη εκτέλεση

της επόμενης ενέργειας. Από τις αποστάσεις αυτές βρίσκουμε όλες τις ελάχιστες τιμές , δηλαδή τιμές που είναι ισάξιες και τυχαία επιλέγουμε μία από αυτές. Αυτό γίνεται για να οριστεί η έννοια της πλάγιας κίνησης σε περίπτωση που ο πράκτορας κολλήσει σε 2 κινήσεις πχ πάνω-κάτω.
Εφόσον η ελάχιστη απόσταση ευρεθεί, ορίζεται και η ενέργεια που πρέπει να γίνει. Επόμενο βήμα είναι η ανανέωση των κόστων κατά 1 και στο τωρινό κελί προς το επόμενο για την δεδομένη ενέργεια, αλλά και το κόστος από το νέο κελί προς το τωρινό με την αντίστροφη ενέργεια.
Τέλος, εάν υποτίθεται πως ψάχνουμε διαδρομή, κρατάμε την ενέργεια που αποφασίσαμε να κάνουμε, ώστε στο τέλος η συνολική διαδρομή να ενσωματωθεί στο dictionary των συνολικών routes.

moveTowards

Η συγκεκριμένη function εκτελείται κάθε φορά που το ταξί κάνει pickup τον πελάτη. Ουσιαστικά, λόγω της φύσης του προβλήματος κάνει τις ίδιες ενέργειες με την nextSlot, οπότε ο τρόπος λειτουργίας είναι ίδιος.

dropCust

Η μέθοδος αυτή εκτελεί την ενέργεια «αποφόρτισης» του πελάτη από το ταξί. Αφού ελέγξει αν η boolean μεταβλητή που δείχνει αν το ταξί έχει τον πελάτη, είναι true, τότε θα την κάνει false και θα βάλει τον πελάτη θα τοποθετηθεί στην τοποθεσία που βρίσκεται αυτή τη στιγμή το ταξί. Καλείται από την executableAction.

removeCust

Η μέθοδος αυτή, αφού ελέγξει αν το ταξί (taxi agent 0) και το destination (d agent 1), έχουν τις ίδιες συντεταγμένες τοποθεσίας και αν ο agent 1 (d) έχει τον πελάτη, τότε μόνο θα τον κάνει remove από αυτόν και θα εξαφανιστεί. Καλείται από την executableAction.

Βιβλιογραφία

- Συστήματα Ευφυών Πρακτόρων: Διαφάνειες και Διαλέξεις μαθήματος
- Συστήματα Ευφυών Πρακτόρων: Διαφάνειες και Διαλέξεις Εργαστηρίου μαθήματος
- https://github.com/nataliakoliou/AgentSpeak-Programming-using-Jason/tree/main/examples/marsbots
- https://github.com/nataliakoliou/AgentSpeak-Programming-using-Jason/blob/main/slides.pdf
- «Εισαγωγή στα Πολυπρακτορικά συστήματα»: Michael Wooldridge – Εκδόσεις Κλειδάριθμος

ΑΠΟΣΤΟΛΟΣ ΧΑΣΙΩΤΗΣ

ΣΩΤΗΡΙΟΣ ΔΗΜΗΤΡΑΚΟΥΛΑΚΟΣ