Αναφορά 1ης Προγραμματιστικής Εργασίας

Γιώργος Δούκας

ΑΜ: 2016030032

Σκοπός αυτής της εργασίας ήταν η λύση του προβλήματος εύρεσης βέλτιστης διαδρομής που αντιμετωπίζει ένα όχημα, το οποίο πρέπει να μεταβεί από μία αρχική θέση σε ένα τελικό στόχο. Το όχημα λύνει το πρόβλημα offline πριν την αναχώρησή του, και ακολουθεί το πλάνο διαδρομής που του προτείνει το πρόγραμμά. Το πρόγραμμά αυτό υπολογίζει πολλαπλά πλάνα διαδρομής, ένα πλάνο διαδρομής ανά συνδυασμό παραλλαγής αλγορίθμου και ευρετικής συνάρτησης, χρησιμοποιώντας:
(α) Παραλλαγές του Weighted A*, με βάρη $w_1 = 0$, $w_2 = 1$ ... (όπου w_i ακέραιος)
(β) Τον αλγόριθμο IDA*

Το πρόγραμμά δέχεται ως είσοδο ένα αρχείο σχετικό με ένα scenario που περιλαμβάνει πληροφορίες για την αρχική θέση του οχήματος, των ορίων του δρόμου, τα εμπόδια και έναν τελικό στόχο, και καθοδηγεί το όχημα στον τελικό προορισμό αποφεύγοντας να χτυπήσει κάποιο εμπόδιο ή να βρεθεί εκτός των ορίων του δρόμου.

main()

Στο αρχείο main.py γίνεται η επιλογή του scenario, της ευρετικης μέσω του heuristic_option (1 -> Euclidean, 2 -> Manhattan) και του βάρους w της ευρετικης μέσω του heuristic_weight. Καλείται η execute_search() για τον κάθε αλγόριθμο με τα κατάλληλα ορίσματα και επιστρέφει:
- τον αριθμό κόμβων που επεκτείνονται (Visited nodes number),
- το ακριβές δρομολόγιο (ακολουθία οδών) που προτείνει ο αλγόριθμος να ακολουθήσει το όχημα (Path),
-τη τιμή της ευρετικής που επιλέχθηκε από τον αρχικό κόμβο,
-το εκτιμώμενο από τον αλγόριθμο συνολικό κόστος g(n) της τελικής διαδρομής.

Καλώντας την print_results() εμφανίζει στην οθόνη αλλά και παράγει ένα αρχείο εξόδου (*results.txt)*, όπου αναφέρονται για κάθε αλγόριθμο και για κάθε διαφορετικό scenario τα παραπάνω αποτελέσματα. Επίσης μετά από κάθε επιτυχές τρέξιμο ενός αλγορίθμου, αποθηκεύεται τελικό στιγμιότυπο στον φάκελο Figures, τα οποία είναι διαθέσιμα στον αντίστοιχο φάκελο.

Weighted A*

Η υλοποίηση του weighted A*αλγορίθμου πληροφορημένης αναζήτησης γίνεται εντός της execute_search(). Για w=1 είναι ο κλασσικός A* και για την υλοποίηση του έχει χρησιμοποιηθεί μια λίστα queue όπου αποθηκεύονται οι ανοιχτοί κομβόι και μέχρι να εξαντληθεί αυτή η λίστα ή να βρεθεί ο στόχος συνεχίζει η αναζήτηση. Ξεκινώντας από τον αρχικό κόμβο (node_initial) παίρνει τα παιδιά του και για όσα δεν είναι εμπόδια ή δεν προκαλούν σύγκρουση, υπολογίζει το κόστος *f* και *h* και τα προσθέτει στην λίστα ανοιχτών κόμβων. Εδώ χρησιμοποιήθηκε μια παραλλαγή της take_step(), η successor_to_node(), όπου κάνει ότι και η take_step αλλά χωρίς να γίνεται update_visuals(), έτσι ώστε να πάρουμε την απαραίτητη πληροφορία για τα κόστη των κόμβων παιδιών του τωρινού κόμβου χωρίς να επισκεπτόμαστε τον κόμβο αυτόν. Η λίστα ταξινομείται με βάση το κόστος f, και σε περίπτωση ιδίου f επιλέγεται ο κόμβος με το μικρότερο ευρετικο κόστος h. Σε περίπτωση ισοπαλίας και στο h, χρησιμοποιήθηκε μια ακόμα μεταβλητή (tie_breaker), ώστε να επιλεχθεί κάποιος κόμβος.

Αφού προστεθούν οι κόμβοι παιδιά, βγαίνει από την λίστα ο successor με το μικρότερο κόστος f και κάνει το βήμα καλώντας την take_step(). Ελέγχει αν είναι ο στόχος και τότε σταματάει η αναζήτηση. Αν δεν είναι στόχος και η λίστα με τους ανοιχτούς κόμβους δεν άδειασε, ο τωρινός κόμβος γίνεται αυτός του παιδιού successor και συνεχίζει η αναζήτηση.

Να σημειωθεί ότι για weight w=0 τότε η συνάρτηση εκτίμησης f(n) = g(n) + w * h(n) παίρνει την μορφή f(n) = g(n) και μιλάμε για τον αλγόριθμο απληροφόρητης αναζήτησης Uniform Cost Search (UCS)

Iterative Deepening A*

Παρόμοια υλοποίηση με αυτή του A* μόνο που η αναζήτηση συνεχίζει μέχρι ένα όριο κόστους f_{max} κάθε φορά. Αν ο κόμβος που βγαίνει από την λίστα έχει $f > f_{max}$ τότε η αναζήτηση σταματάει, το νέο f_{max} είναι αυτό του κόμβου και γίνεται reset ο αλγόριθμος IDA* ξεκινώντας από την αρχή με το καινούργιο αυτό όριο. Για να μην χαθεί ο αριθμός των κόμβων που πραγματικά επισκέφτηκε αυτός ο αλγόριθμος και εμφανίζεται ο αριθμός κόμβων μόνο της τελευταίας κλήσης αυτού, κρατείται στην μεταβλητή counter το πλήθος κόμβων που έχει επισκεφτεί και ανανεώνεται κάθε φορά πριν ξανακαλεστεί ο αλγόριθμος.

Heuristics

Έστω (node_x, node_y) οι συντεταγμένες του τωρινού κόμβου και (goal_x, goal_y) οι συντεταγμένες του στόχου. Ορίζουμε ως distance_x = |goal_x – node_x| την απόσταση του κόμβου από τον στόχο ως προς τον άξονα x. Ομοίως για τον άξονα y.

- **Euclidean**

$$h = \sqrt{distance\_x^2 + distance\_y^2}$$

  Η τιμή αυτής της ευρετικης είναι η απόσταση της ευθείας γραμμής μεταξύ των δυο σημείων.

- **Manhattan variation**

$$h = distance\_x + distance\_y + \sqrt{2} * min(distance\_x , distance\_y)$$

Η ιδέα για αυτή την ευρετικη προήλθε έπειτα από πειραματισμούς με την Manhattan ευρετικη

$$h = distance\_x + distance\_y$$

και την παραλλαγή της Diagonal , Octile.

$$h = |distance\_x - distance\_y| + \sqrt{2} * min(distance\_x , distance\_y)$$

Η Octile και η Manhattan επισκέπτονταν οριακά λιγότερους κόμβους από την Euclidean, σε scenario χωρίς πολλά εμπόδια (scenario 1), οπότε σε τέτοια περίπτωση δεν είχε ιδιαίτερη διαφορά. Όμως στα αλλά δυο σενάρια , η Octile και η Manhattan είχαν καλύτερη εκτίμηση του πραγματικού κόστους g και ειδικά η Manhattan, από την Euclidean. Έτσι βασιζόμενος στην εκτίμηση της Manhattan και προσθέτοντας ένα ακόμα όρο ($\sqrt{2}$ * min(distance_x , distance_y)) για την προσομοίωση των εμποδίων που μπορεί να συναντήσει ο αλγόριθμος και των παρακάμψεων που θα κάνει από την ευθεία απόσταση, υλοποιήθηκε η παραπάνω παραλλαγή της Manhattan ευρετικης. Η οποία είναι παραδεκτή και προσεγγίζει αρκετά η εκτίμηση της το πραγματικό κόστος της διαδρομής, χωρίς να κάνει υπερεκτιμήσεις.

Αποτελέσματα

Όπου h->E είναι για την ευρετικη Euclidean, και h->M για την παραλλαγή της Manhattan. Επίσης το μονοπάτι κόμβων Path αναγράφεται στο αρχείο *results.txt* εκτός από τις περιπτώσεις όπου path = None που σημαίνει ότι ο αλγόριθμος δεν βρήκε λύση.

SCENARIO 1

	DFS	***A* w =0 h->E***	***A* w =1 h->E***	***A* w =1 h->M***	***A* w =2 h->E***	***A* w =2 h->M***	***A* w =5 h->E***	***A* w =5 h->M***	***IDA* w =1 h->E***	***IDA* w =1 h->M***
Visited Nodes number	231	298	28	27	16	29	16	12	304	168
Path	-	-	-	-	-	-	-	-	-	-
Heuristic Cost (initial node)	0.00	32.50	32.50	32.64	32.50	32.64	32.50	32.64	32.50	32.64
Estimated Cost	36.00	31.50	31.50	31.50	31.50	31.50	31.50	31.50	31.50	31.50

Απληροφόρητη αναζήτηση:

Ο DFS αν και επισκέπτεται λιγότερους κόμβους από τον A* w=0, το μονοπάτι που διαλέγει έχει μεγαλύτερο κόστος.

Πληροφορημένη αναζήτηση:

- A*

Και οι δυο ευρετικες σε αυτό το σενάριο κάνουν μεγαλύτερη εκτίμηση από το πραγματικό κόστος της διαδρομής, παρόλα αυτά προσφέρουν παρόμοια μονοπάτια.
Για w=1 καλύτερη είναι η παραλλαγή της Manhattan επισκεπτόμενη ένα λιγότερο κόμβο.
Για w=2 η μικρή υπερεκτίμηση που προσφέρει η Manhattan έχει σαν αποτέλεσμα να προσπεραστεί ο στόχος και να επισκεφτεί παραπάνω κόμβους.
Για w=5 αυτή η μικρή υπερεκτίμηση έχει σαν αποτέλεσμα να φτάσει πιο γρηγορά στον στόχο επισκεπτόμενος 4 κόμβους λιγότερους.

- IDA*

Η διαφορά μεταξύ των επαναλήψεων είναι ξεκάθαρη, με την ευρετικη που υλοποιήθηκε να δίνει σχεδόν τους μισούς κόμβους που επισκέφτηκαν.

SCENARIO 2

	DFS	***A* w =0 h->E***	***A* w =1 h->E***	***A* w =1 h->M***	***A* w =2 h->E***	***A* w =2 h->M***	***A* w =5 h->E***	***A* w =5 h->M***	***IDA* w =1 h->E***	***IDA* w =1 h->M***
Visited Nodes number	253	1145	204	163	113	53	190	72	11328	5266
Path	-	-	-	-	-	-	-	-	-	-
Heuristic Cost (initial node)	0.00	52.50	52.50	52.66	52.50	52.66	52.50	52.66	52.50	52.66
Estimated Cost	54.00	54.00	54.00	54.00	54.00	54.00	54.00	54.00	54.00	54.00

Απληροφόρητη αναζήτηση:
Ο DFS επισκέπτεται λιγότερους κόμβους από τον A* w=0 και το μονοπάτι που διαλέγει έχει ίδιο κόστος.

Πληροφορημένη αναζήτηση:

- A*

  Σε αυτό το σενάριο η ευρετικη που υλοποιήθηκε προσφέρει καλύτερη χρονικά υλοποίηση, καθώς ο αλγόριθμος επισκέπτεται κάθε φορά σχεδόν τους μισούς κόμβους από ότι η Euclidean. Παρόλα αυτά και οι δυο προσφέρουν το ίδιο κόστος για τη τελική διαδρομή.

- IDA*

  Όπως και στον A* με τη χρήση της Manhattan παραλλαγής από την Euclidean ο αλγόριθμος επισκέπτεται τους μισούς κόμβους.

SCENARIO 3

	DFS	***A* w =0 h->E***	***A* w =1 h->E***	***A* w =1 h->M***	***A* w =2 h->E***	***A* w =2 h->M***	***A* w =5 h->E***	***A* w =5 h->M***	***IDA* w =1 h->E***	***IDA* w =1 h->M***
Visited Nodes number	850	575	346	107	49	49	49	49	30370	1389
Path	-	-	-	-	None	None	None	None	-	-
Heuristic Cost (initial node)	0.00	57.54	57.54	62.49	57.54	62.49	57.54	62.49	57.54	62.49
Estimated Cost	72.00	67.50	67.50	67.50	0.00	0.00	0.00	0.00	67.50	67.50

Απληροφόρητη αναζήτηση:
Σε αυτό το σενάριο ο A* w=0 επισκέπτεται λιγότερους κόμβους από τον DFS και το μονοπάτι που διαλέγει έχει μικρότερο κόστος.

Πληροφορημένη αναζήτηση:

- A*

Στο σενάριο αυτό επειδή τα εμπόδια είναι στημένα έτσι, ο αλγόριθμος A* βρίσκει λύση μόνο για w=1. Στις άλλες περιπτώσεις που έχουν επιλεγεί βάρη w>1 ο αλγόριθμος υπερεκτιμάει τον στόχο με αποτέλεσμα να μην προσπερνάει το εμπόδιο και να εξαντλείται η λίστα ανοιχτών κόμβων. Έτσι ο αλγόριθμος τερματίζει χωρίς να βρει λύση.
Για w=1 λοιπόν, η ευρετικη Manhattan κάνει καλύτερη εκτίμηση h=62.49 του αρχικού κόμβου σε σχέση με το συνολικό κόστος της τελικής διαδρομής g=67.50, καθώς η ευκλιδειανή εκτιμάει h=57.54. Ακόμη με την Manhattan επισκέπτεται 107 κόμβους ενώ με την Euclidean σχεδόν τους τριπλάσιους, 346 κόμβους.

- IDA*

Στο συγκεκριμένο αλγόριθμο η επίσκεψη λιγότερων κόμβων μέσω της Manhattan ευρετικης είναι καθοριστική στην σύγκριση, καθώς με κάθε επανάληψη αυξάνεται εκθετικά ο αριθμός των κόμβων και αυτό έχει ως συνέπεια αργή υλοποίηση. Με την Euclidean ευρετικη λοιπόν, θα επισκεφτεί 30,370 κόμβους, ενώ με την Manhattan μόνο 1,389.

Συμπεράσματα

Στα παραπάνω σενάρια ο A* αλγόριθμος είναι πιο γρήγορος από τον IDA*, καθώς ο IDA* επαναλαμβάνεται κάθε φορά που αλλάζει το όριο. Σε αυτά τα σενάρια ο IDA* δεν έχει καμία διαφορά από τον A* στα αποτελέσματα του. Θα μπορούσε σε κάποιο άλλο σενάριο να βγάλει πιο σύντομο μονοπάτι από τον A* καθώς με την επαναληπτική εκβάθυνση δεν προσπερνάει την βέλτιστη λύση. Παρόλα αυτά έχει αργή πρόοδο λόγω της αργής αύξησης του ορίου αποκοπής και διότι δεν ανιχνεύει επαναλαμβανόμενες καταστάσεις.
Συγκρίνοντας τα αποτελέσματα, ο A* δίνει καλύτερες λύσεις, όταν χρησιμοποιείται η ευρετική που υλοποιήθηκε (Manhattan variation), και όταν w=1 έτσι ώστε να είναι παραδεκτή η ευρετικη και να μην κάνει απαισιόδοξες προσεγγίσεις του στόχου.

Πηγές:

Πηγές πληροφόρησης:

http://theory.stanford.edu/~amitp/GameProgramming/Heuristics.html

https://www.geeksforgeeks.org/a-search-algorithm/

https://www.eclass.tuc.gr/courses/HMMY226/