7/2/2014

ΥΣ02 Τεχνητή Νοημοσύνη

ΠΡΩΤΗ ΣΕΙΡΑ ΑΣΚΗΣΕΩΝ (2ο μέρος)

ΟΝΟΜΑ: Κωνσταντίνα Γαλούνη

Α.Μ. : 1115201000034

Πρόβλημα 2

δ) Άπληστη Αναζήτηση πρώτα στον καλύτερο

α. Ο κόμβος στον οποίο φτάνει πρώτα ο αλγόριθμος είναι ο G1

β. Από τη λίστα «σύνορο» οι κόμβοι βγαίνουν με την εξής σειρά:

S, A, B, G1

- Fringe: (S, 5, None)
  Closed: –
- Fringe: (A, 2, S), (C, 3, S)
  Closed: S
- Fringe: (C, 3, S), (B, 1, A), (F, 6, A)
  Closed: S, A
- Fringe: (C, 3, S), (F, 6, A), (G1, 0, B), (C, 3, S), (D, 2, B)
  Closed: S, A, B
- Εξαγωγή κόμβου G1 από το σύνορο → Κόμβος Στόχος

ε) A^*

α. Ο κόμβος που φτάνει πρώτα ο αλγόριθμος είναι ο G2.

β. Από τη λίστα «σύνορο» οι κόμβοι βγαίνουν με την εξής σειρά:

S, A, B, D, C, D, G2

- Fringe: (S, 5, None)
  Closed: –
- Fringe: (A, 4, S), (C, 6, S)
  Closed: S
- Fringe: (C, 6, S), (B, 4, A), (F, 16, A)
  Closed: S, A

- Fringe: (C, 6, S), (F, 16, A), (G1, 7, B), (C, 7, B), (D, 6, B)
  Closed: S, A, B
- Fringe: (F, 16, A), (G1, 7, B), (C, 7, B), (D, 6, B), (D, 6, C), (G2, 8, C)
  Closed: S, A, B, C
- Fringe: (F, 16, A), (G1, 7, B), (C, 7, B), (D, 6, C), (G2, 8, C), (G2, 6, D)
  Closed: S, A, B, C, D
- Fringe: (F, 16, A), (G1, 7, B), (C, 7, B), (G2, 8, C), (G2, 6, D)
  Closed: S, A, B, C, D
- Εξαγωγή κόμβου G2 από το σύνορο → Κόμβος στόχος

Πρόβλημα 3

1. Η αλλαγή του αλγορίθμου που προτείνεται δε θα είναι αποδοτική για τη γενική περίπτωση των προβλημάτων. Συγκεκριμένα, παρατηρούμε πως θα έχουμε με αυτόν τον τρόπο ελαχιστοποίηση της απαιτούμενης μνήμης, αφού δε θα εισάγουμε στο σύνορο τους κόμβους που έχουν ήδη επεκταθεί (expand). Ωστόσο, πρέπει να λάβουμε υπόψην και τη χρονική πολυπλοκότητα αυτής της υλοποίησης. Για μικρές διαστάσεις του προβλήματος (όπου κάθε κόμβος δε θα έχει μεγάλο πλήθος απογόνων) ο αλγόριθμος αυτός θα είναι καλύτερος. Αντίθετα, για μεγάλο πλήθος απογόνων σε ένα κόμβο N, όπου θα πρέπει να ελέγξουμε όλους τους απογόνους του, αυξάνεται υπερβολικά ο χρόνος εκτέλεσης. Αν για παράδειγμα επεκταθούν n κόμβοι με k απογόνους ο καθένας (για k μεγάλο αριθμό), τότε θα πρέπει για κάθε έναν από τους n κόμβους να τσεκάρουμε k άλλους. Συνεπώς, δε θα ήταν καλή ιδέα να αλλάξουμε τον αλγόριθμο GRAPH-SEARCH.

2. Ο αλγόριθμος A^* επιλέγει ποιος από τους κόμβους του συνόρου θα γίνει expand με βάση την τιμή της ευρετικής συνάρτησης για τον κόμβο αυτό, αλλά και του κόστους του μονοπατιού μέχρι τον κόμβο αυτό. Σε περίπτωση που αλλάζαμε τον αλγόριθμο και ελέγχαμε και τους απογόνους ενός κόμβου N που είχε επιλεγεί για επέκταση, θα αγνοούσαμε το κόστος της μετάβασης από τον κόμβο N στο στόχο.

7/2/2014

ΥΣ02 Τεχνητή Νοημοσύνη

ΠΡΩΤΗ ΣΕΙΡΑ ΑΣΚΗΣΕΩΝ (2ο μέρος)

ΟΝΟΜΑ: Κωνσταντίνα Γαλαούνη

Α.Μ.: 1115201000034

Πρόβλημα 3 (συνέχεια)

2. Ένα παράδειγμα, που αναδεικνύει πως αυτή η αλλαγή κάποιες φορές έχει ως αποτέλεσμα μία μη βέλτιστη λύση, είναι ο γράφος του Προβλήματος 2. Συγκεκριμένα, αν εκτελούσαμε το νέο αλγόριθμο, το αποτέλεσμα θα ήταν να φτάσουμε πρώτα στον κόμβο G1 με κόστος 7, ακολουθώντας το μονοπάτι S→A→B→G1. Άρα, καταλήγουμε στο συμπέρασμα πως η αλλαγή αυτή στον αλγόριθμο A^* είναι κακή.

Πρόβλημα 1

ε) Για το πρόβλημα AlienTiles γνωρίζουμε τα εξής:

1. Κάθε tile επηρεάζει τη γραμμή και τη στήλη του
2. Κάθε tile αλλάζει χρώμα κυκλικά με τη σειρά: R→G→B→P

Οπότε αφαιρώντας κανέναν, έναν ή και τους 2 παραπάνω περιορισμούς καταλήγουμε στις παρακάτω ευρετικές συναρτήσεις.

h_1: Η h_1 επιστρέφει 0 αν είμαστε στην κατάσταση στόχου και 1 σε οποιαδήποτε άλλη περίπτωση, δηλαδή δε λαμβάνει υπόψην της κανέναν από τους παραπάνω περιορισμούς. Η ευρετική αυτή είναι συνεπής και παραδεκτή, αφού ποτέ δεν υπερεκτιμά και ικανοποιεί την τριγωνική ανισότητα $h_1(n) \leq c(n,a,n') + h_1(n')$. Ωστόσο, παρά το ότι ο αλγόριθμος A^* βρίσκει τη βέλτιστη λύση χρησιμοποιώντας την h_1, δεν είναι καθόλου αποδοτική σε σχέση με το χρόνο. Αυτό συμβαίνει γιατί σε κάθε περίπτωση μη κατάστασης στόχου επιστρέφεται ο ίδιος αριθμός, επομένως για ένα κόμβο N που γίνεται expand, θα εξεταστούν όλοι οι απόγονοί του, εξαιτίας των ισοπαλιών. Τελικά, παρατηρούμε ότι με αυτήν την ευρετική ο A^* λειτουργεί όπως ο BFS αλγόριθμος.

h_2: Η ευρετική αυτή βασίζεται στην παραδοχή ότι ενώ τα χρώματα αλλάζουν κυκλικά και κάθε tile επηρεάζει τη γραμμή και τη στήλη του, μόλις ένα tile φτάσει σε χρώμα στόχου, πάει να επηρεάζεται από τα υπόλοιπα. Γνωρίζοντας πως η επιλογή ενός tile θα αλλάξει χρώμα σε 2n-1 tiles (για διάσταση n) και θεωρώντας distance: την απόσταση του μακρινότερου χρώματος από το χρώμα στόχο και number: τον αριθμό των tiles που έχουν το χρώμα που αντιστοιχεί σε αυτήν την απόσταση έχουμε:

```
if number >= 2n-1 :
    if (number % 2n-1) == 0 :
        return (number / 2n-1) * distance
    else :
        return (number / 2n-1) * distance + distance
else :
    return distance.
```

h_3: Η h_3 είναι μια παραλλαγή της h_2 στην προσπάθεια να περιορίσουμε λίγο περισσότερο το πρόβλημα. Συγκεκριμένα, είναι και αυτή συνεπής και παραδεκτή και ελέγχει για κάποιες περιπτώσεις αν θα αλλάξει το χρώμα σε tiles που βρίσκονται στο χρώμα-στόχο, οπότε η νέα μέγιστη απόσταση θα είναι 3. (Οπότε απαιτούνται κατ' ελάχιστο 3 κινήσεις ακόμα για την κατάσταση στόχου). Τελικά προκύπτει το παρακάτω:

```
if distance == 0 :
    return 0
if number == 2n-1 :
    return distance
elif number > 2n-1 :
    if (number % 2n-1) == 0 :
        return (number/2n-1) * distance
    else :
        return (number/2n-1) * distance + distance
else :
    if "στον πίνακα υπάρχουν tiles με 2 διαφορετικά χρώματα και το ένα είναι σίγουρα το goal"
        return 4
    else :
        return distance
```

7/2/2014

ΥΣ02 Τεχνητή Νοημοσύνη

ΠΡΩΤΗ ΣΕΙΡΑ ΑΣΚΗΣΕΩΝ (2ο μέρος)

ΟΝΟΜΑ: Κωνσταντίνα Γαλαούη

Α.Μ. : 1115201000034

Σημείωση: Οι χρόνοι και οι κόμβοι έχουν μετρηθεί από εκτελέσεις σε Unix μέσω Putty

Πρόβλημα 1 (Συνέχεια)

δ) Μια εύκολη προς επίλυση είσοδος είναι:

Για διάσταση $n=3$ και goal = Green:

G	G	R
G	G	R
R	R	R

h_1: expand: 9 κόμβοι, χρόνος: < 1sec

h_2: expand: 1 κόμβος, χρόνος: < 1sec

h_3: expand: 1 κόμβος, χρόνος: < 1sec

Παρόμοια κατάσταση: $n=3$ και goal = Green:

G	B	G
B	B	B
G	B	G

h_1: expand: 59 κόμβοι, χρόνος: < 1 sec

h_2: expand: 3 κόμβοι, χρόνος: < 1 sec

h_3: expand: 3 κόμβοι, χρόνος: < 1 sec

Μια ακόμα ενδεικτική είσοδος: $n=3$ και goal = Blue:

R	R	R
R	R	R
R	R	R

h_1: expand: 2820 κόμβοι, χρόνος: 3 min και 2 sec

h_2: expand: 271 κόμβοι, χρόνος: 2,63 sec

h_3: expand: 245 κόμβοι, χρόνος: 2,38 sec

Είσοδος που βρίσκει δύσκολα λύση:

Για $n=3$ και goal = Purple:

R	G	R
R	G	R
G	G	P

h_1: expand: 6948 κόμβοι, χρόνος: 10 min και 50 sec

h_2: expand: 1536 κόμβοι, χρόνος: 1 min και 3 sec

h_3: expand: 1505 κόμβοι, χρόνος: 1 min και 1 sec

Μια ακόμα είσοδος που είναι δύσκολο να βρεθεί λύση, επειδή στην υλοποίηση δεν εκμεταλλευόμαστε τις συμμετρίες είναι: $n=4$ και goal = Purple:

P	R	G	R
R	G	G	G
G	G	G	R
R	G	R	P

h_1: expand: - , χρόνος: -

h_2: expand: 1143 , χρόνος: 2 min και 51 sec

h_3: expand: 1143 , χρόνος: 2 min και 51 sec

Πρόβλημα 4

β) Για το πρόβλημα των κύβων γνωρίζουμε ότι:

Μπορούμε να μετακινήσουμε έναν ελεύθερο κύβο από έναν άλλο κύβο στο τραπέζι ή από ένα αντικείμενο (κύβο ή τραπέζι) σε έναν ελεύθερο κύβο.

h_1: Η h_1 είναι όμοια με την h_1 για το πρόβλημα Alien tiles. Αυτό είναι εφικτό, αφού η ευρετική αυτή είναι ανεξάρτητη από οποιαδήποτε παράμετρο του προβλήματος. Οπότε, τα χαρακτηριστικά της είναι ακριβώς τα ίδια με πριν.

h_2: Επιστρέφει: Πόσοι κύβοι είναι σε λάθος θέση (σε σχέση με τον κύβο ή το τραπέζι που βρίσκεται κάτω από τον καθένα).
Σε αυτό το πρόβλημα η ευρετική είναι συνεπής και παραδεκτή. Ως προς την αποδοτικότητά της, όμως, παρατηρούμε ότι δε λαμβάνει υπόψην ότι κάποιοι κύβοι που βρίσκονται σε λάθος θέση και πρέπει να μετακινηθούν, έχουν από πάνω τους άλλους κύβους. Άρα η εκτίμηση αυτής της συνάρτησης είναι αρκετές φορές υπερβολικά αισιόδοξη, αφού κάθε κύβος μετακινείται μόνο εάν είναι ελεύθερος.

h_3: Η ευρετική αυτή υπολογίζει τον αριθμό των κύβων που βρίσκονται σε λάθος θέση (σε σχέση με τον κύβο ή το τραπέζι που βρίσκεται κάτω από τον καθένα), καθώς επίσης και για κάθε έναν από τους παραπάνω, πόσοι κύβοι βρίσκονται από πάνω του και θα έπρεπε να είναι εκεί, δηλαδή είναι σωστοί ως προς αυτόν που έχουν από κάτω. Συμβολίζουμε τους πρώτους με wrong_position και τους άλλους που αναφέρθηκαν με top. Τότε, η h_3 επιστρέφει wrong_position + 2*top, γιατί για κάθε έναν από τους wrong position, οι top που του αντιστοιχούν πρέπει να μεταφερθούν προσωρινά σε λάθος θέση και όταν ο wrong position πάρει τη σωστή θέση του, να επιστρέψουν από πάνω του. Όπως θα δούμε παρακάτω στα πειραματικά αποτελέσματα, η συνάρτηση αυτή δεν είναι καθόλου αποδοτική, γιατί προσθέτει μεγάλη πολυπλοκότητα.

7/2/2014

ΥΣ02 Τεχνητή Νοημοσύνη

ΠΡΩΤΗ ΣΕΙΡΑ ΑΣΚΗΣΕΩΝ (2ο μέρος)

ΟΝΟΜΑ: Κωνσταντίνα Γαλαίνη

Α.Μ. : 1115201000034

Πρόβλημα 4 (συνέχεια)

δ) Μια εύκολη προς επίλυση είσοδος είναι:

Για διάσταση $n=5$:

Είσοδος	Στόχος
B	
A C D E	A B C D E

h_1: expand: 1 , χρόνος: < 1 sec

h_2: expand: 1 , χρόνος: < 1 sec

h_3: expand: 4 , χρόνος: < 1 sec

Μια ακόμα είσοδος

Για διάσταση $n=5$:

Είσοδος	Στόχος
B D	E C
A C E	A B D

h_1: expand: 149 , χρόνος: 1 sec

h_2: expand: 11 , χρόνος: < 1 sec

h_3 expand: 299 , χρόνος: < 1 sec

Η είσοδος που δίνεται στο παράδειγμα της άσκησης είναι:

Για διάσταση $n=5$:

Είσοδος	Στόχος
A	
B D	E C
C E	A B D

h_1 : expand: 177 , χρόνος : < 1 sec

h_2 : expand : 11 κόμβοι , χρόνος : < 1 sec

h_3 : expand : 341 κόμβοι , χρόνος : 1 sec