Εργασία

Συστήματα Ευφυών Πρακτόρων

Αλέξανδρος Γαϊτάνης, ΑΜ: 63

ΠΜΣ στην Τεχνητή Νοημοσύνη

Εαρινό εξάμηνο 2021 - 2022

Τμήμα Πληροφορικής

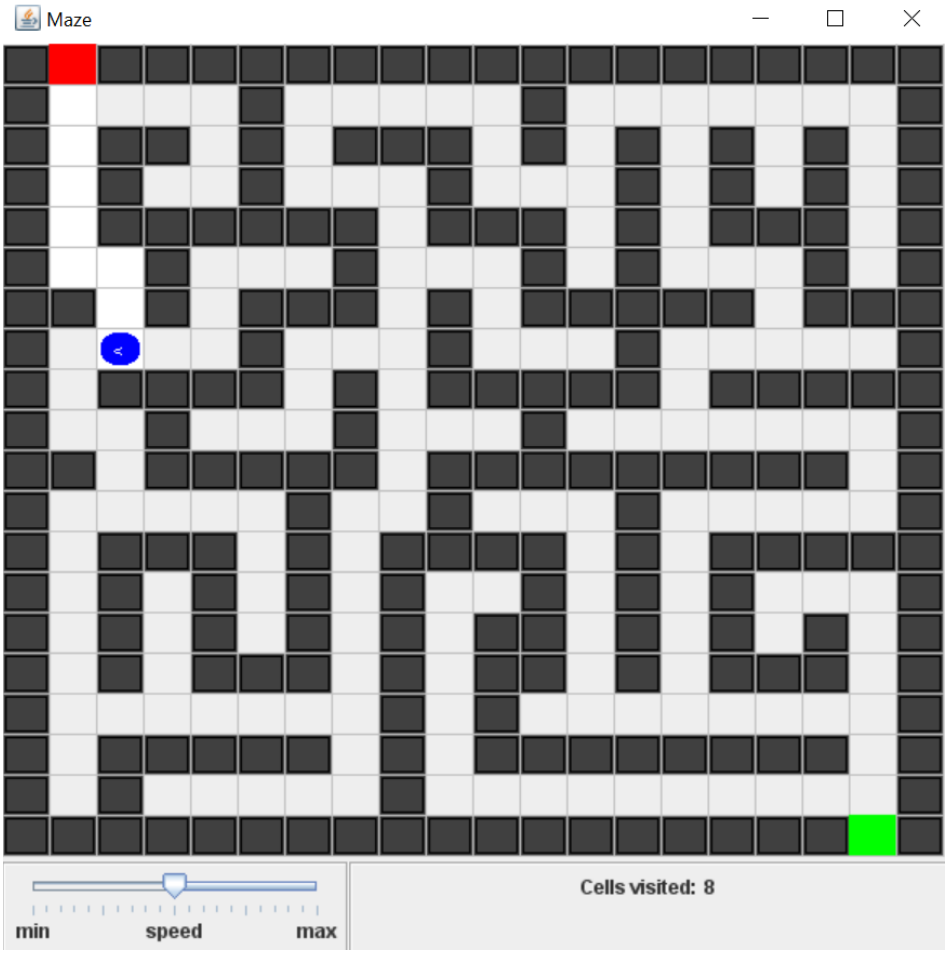
Αριστοτέλειο Πανεπιστήμιο Θεσσαλονίκης

1. Εισαγωγή

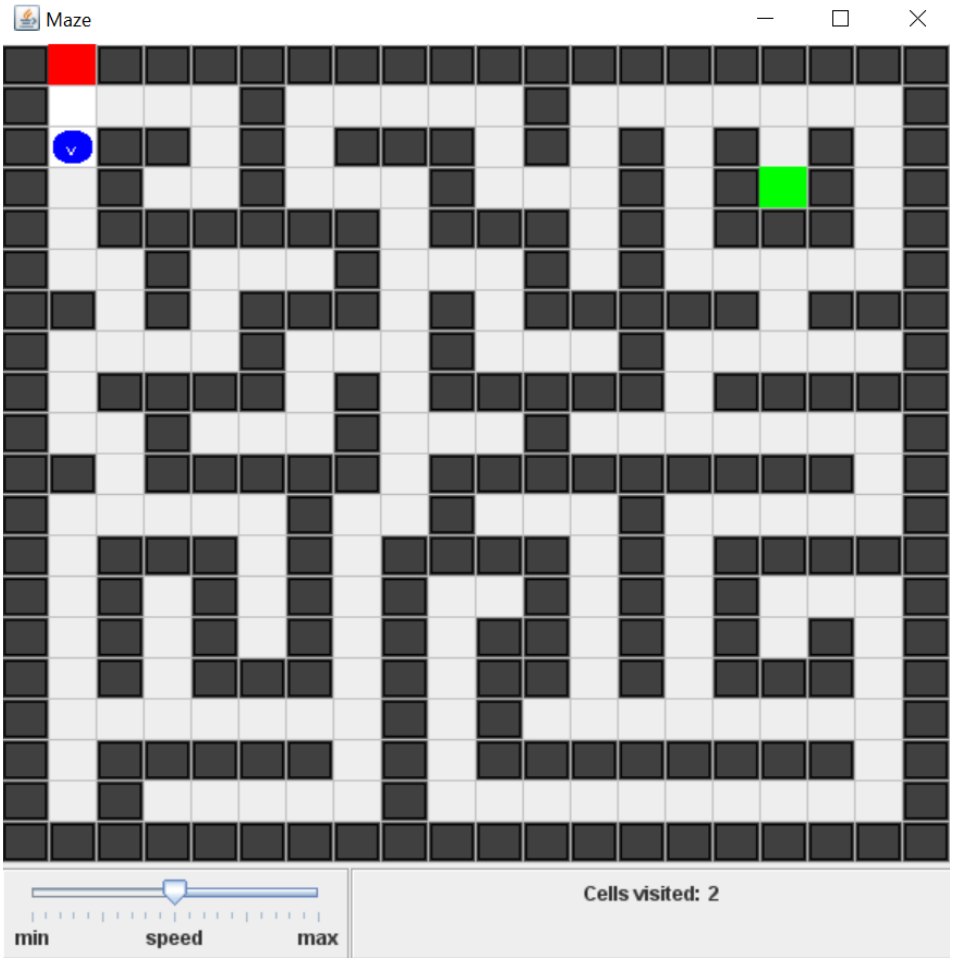
Στην παρούσα εργασία μελετήθηκαν maze solving αλγόριθμοι. Ένα ρομπότ βρίσκεται σε ένα λαβύρινθο και προσπαθεί να αποδράσει χωρίς ξέρει που βρίσκεται. Οι αλγόριθμοι που υλοποιήθηκαν είναι οι εξής: Wall Follower, Random Mouse και Tremaux. Η υλοποίηση έγινε στην πλατφόρμα Jason.

2. Περιβάλλον

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



Εικόνα 1: Λαβύρινθος Α



Εικόνα 2: Λαβύρινθος B

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:

```
/* Rules */
left_cell_free :- not cell(left,obstacle) & not cell(left,entrance) & not cell(left,marked_twice).
front_cell_free :- not cell(front,obstacle) & not cell(front,entrance) & not cell(front,marked_twice).
right_cell_free :- not cell(right,obstacle) & not cell(right,entrance) & not cell(right,marked_twice).
left_front_right_cell_free :- left_cell_free & front_cell_free & right_cell_free.
left_front_cell_free :- left_cell_free & front_cell_free & not right_cell_free.
front_right_cell_free :- not left_cell_free & front_cell_free & right_cell_free.
left_right_cell_free :- left_cell_free & not front_cell_free & right_cell_free.
in_junction :- left_front_right_cell_free | left_front_cell_free | front_right_cell_free | left_right_cell_free.
```

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 $2^{η}$ 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 στον λαβύρινθο Α *(Εικόνα 1)*. 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, τότε οι τοίχοι μπορούν να παραμορφωθούν σε έναν κύκλο, οπότε το πρόβλημα απλοποιείται στο να κινηθεί το ρομπότ γύρω από έναν κύκλο και υπάρχει πάντα λύση *(Εικόνα 1)*. Αν ο λαβύρινθος δεν είναι απλά συνδεδεμένος, δηλαδή η είσοδος ή η έξοδος βρίσκεται κάπου στο εσωτερικό του και είναι περικυκλωμένη από κυκλικό διάδρομο, τότε το ρομπότ μπορεί να μην φτάσει στον στόχο *(Εικόνα 2)*. Τότε επιστρέφει στην είσοδο και ξεκινά από την αρχή τις ίδιες κινήσεις χωρίς να βρίσκει ποτέ τον στόχο του.

6.2 Random mouse

Ο αλγόριθμος αυτός είναι πολύ απλός στη λογική του και δεν απαιτεί μνήμη. Συγκεκριμένα κάθε φορά που το ρομπότ βρίσκεται σε διασταύρωση επιλέγει τυχαία μια κατεύθυνση. Το ρομπότ κάποια στιγμή θα φτάσει τον στόχο του αλλά ο αλγόριθμος μπορεί να είναι πάρα πολύ αργός.

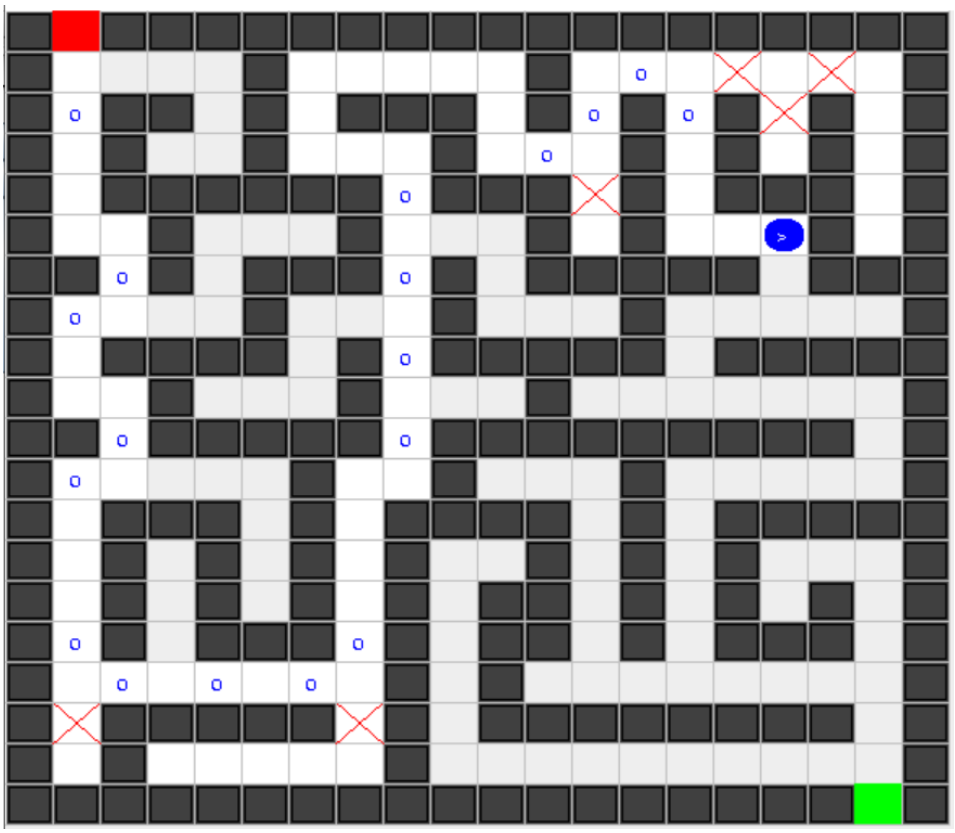
Όταν το ρομπότ βρίσκεται σε διασταύρωση παράγεται ένας τυχαίος αριθμός μεταξύ του 0 και του 1. Αν η διασταύρωση έχει τρεις δυνατές επιλογές, δηλαδή είναι αληθές το *left_front_right_cell_free,* τότε το ρομπότ στρίβει αριστερά, προχωράει μπροστά ή στρίβει δεξιά αν ο τυχαίος αριθμός είναι μεταξύ του 0 και του 1/3, μεταξύ του 1/3 και 2/3 ή μεταξύ του 2/3 και 1 αντίστοιχα. Αν η διασταύρωση έχει δύο δυνατές επιλογές, δηλαδή είναι αληθές ένα από τα *left_front_cell_free*, *front_right_cell_free*, *left_right_cell_free,* τότε το ρομπότ ακολουθεί μια από τις ελεύθερες διαδρομές αν ο τυχαίος αριθμός είναι μεταξύ του 0 και του 1/2 ή μεταξύ του 1/2 και του 1.

6.3 Tremaux

Ο αλγόριθμος αυτός προσφέρει μια αποδοτική μέθοδο στο να βρεθεί η έξοδος από έναν λαβύρινθο και απαιτεί μνήμη, αλλά δεν είναι εγγυημένο ότι θα βρει την συντομότερη διαδρομή.

Κάθε διαδρομή μαρκάρεται μία ή δύο φορές. Κάθε φορά που εισέρχεται το ρομπότ σε μια διαδρομή, τότε την μαρκάρει. Διαδρομή ορίζεται ως όλα τα κελιά από μια διασταύρωση σε μια άλλη. Αν μια διαδρομή έχει μαρκαριστεί 2 φορές το ρομπότ δεν την ακολουθεί. Όταν το ρομπότ βρίσκεται σε μια διασταύρωση τότε επιλέγει την διαδρομή που δεν είναι μαρκαρισμένη, αλλιώς επιλέγει αυτήν που είναι μαρκαρισμένη μία φορά.

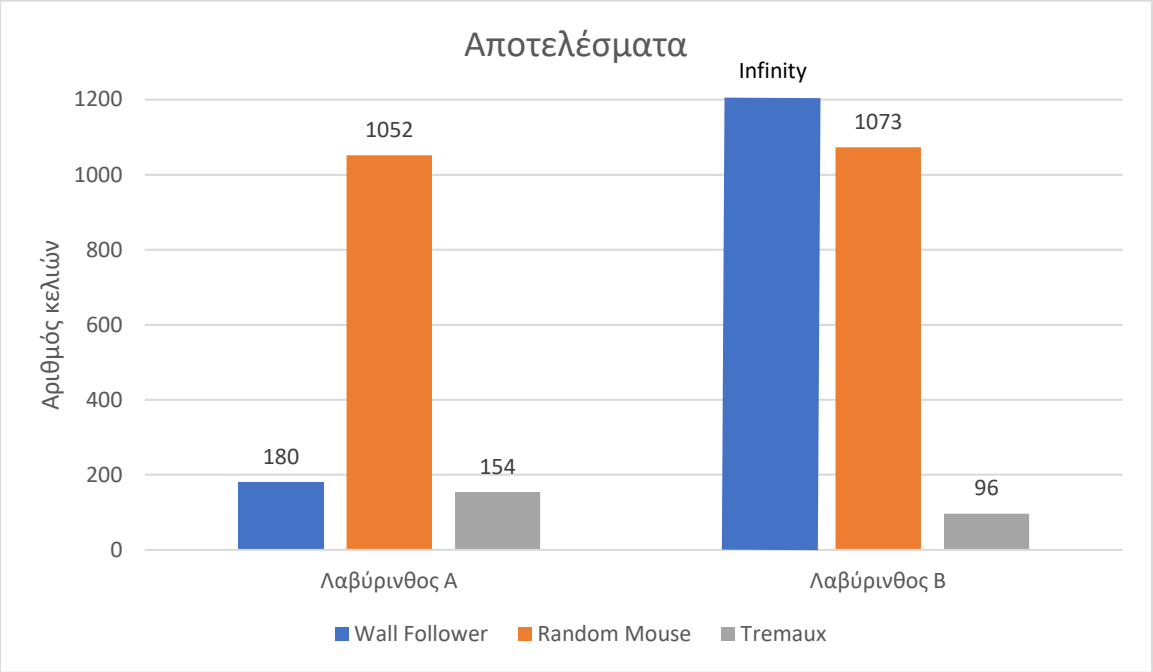
Ένα παράδειγμα της μεθόδου φαίνεται παρακάτω, όπου οι διαδρομές που είναι μαρκαρισμένες μία φορά έχουν στην αρχή και στο τέλος τους ένα μπλε Ο και διαδρομές που είναι μαρκαρισμένες δύο φορές έχουν στην αρχή και το τέλος τους ένα κόκκινο Χ *(Εικόνα 3)*. Παρατηρούμε ότι κάποια αδιέξοδα έχουν μαρκαριστεί δύο φορές και το ρομπότ δεν θα ξαναπεράσει από εκεί.



Εικόνα 3

7. Αποτελέσματα

Παρακάτω φαίνονται τα αποτελέσματα των πειραμάτων *(Εικόνα 4)*.



Εικόνα 4

Στον λαβύρινθο Α ο αλγόριθμος Wall Follower τα πηγαίνει αρκετά καλά, ενώ στον λαβύρινθο Β δεν μπορεί να βρει τον στόχο επειδή, όπως αναλύθηκε παραπάνω, η έξοδος βρίσκεται στο εσωτερικό και είναι περικυκλωμένη από κυκλικό διάδρομο.

Για τον αλγόριθμο Random Mouse δίνεται ο μέσος όρος από 10 διαφορετικά τρεξίματα. Και στους δύο λαβυρίνθους, ο αλγόριθμος ξοδεύει πολύ χρόνο για να βρει τον στόχο αφού σε κάθε διασταύρωση διαλέγει τυχαία μια διαδρομή. Παρατηρείται ότι πηγαίνει πάρα πολλές φορές πίσω μπρος, φτάνοντας ακόμα και στην είσοδο όπου ξεκινάει πάλι από την αρχή. Και στις δύο περιπτώσεις όμως κάποια στιγμή καταφέρνει και βρίσκει τον στόχο σε αντίθεση με τον Wall Follower.

Ο αλγόριθμος Tremaux καταφέρνει και στις δύο περιπτώσεις να βρει την έξοδο, έχοντας επισκεφτεί τον μικρότερο αριθμό κελιών σε σχέση με τους άλλους δύο. Αποδεικνύεται ο πιο αποδοτικός καταφέρνοντας να αποκλείσει αδιέξοδα και διαδρομές τις οποίες τις έχει ξαναδοκιμάσει χωρίς αποτέλεσμα.