Τεχνητή Νοημοσύνη

Στρατηγικές Πληροφορημένης Αναζήτησης

Δρ. Δημήτριος Κουτσομητρόπουλος

Αλγόριθμοι αναζήτησης λύσης

Αλγόριθμοι Πληροφορημένης Αναζήτησης

- **Best-First Search(BestFS)**
  Αναζήτηση Πρώτα στο Καλύτερο
- **Αλγόριθμος A***
  IDA*, MA*/SMA*

(Μεθ-) Ευρετικές Μέθοδοι

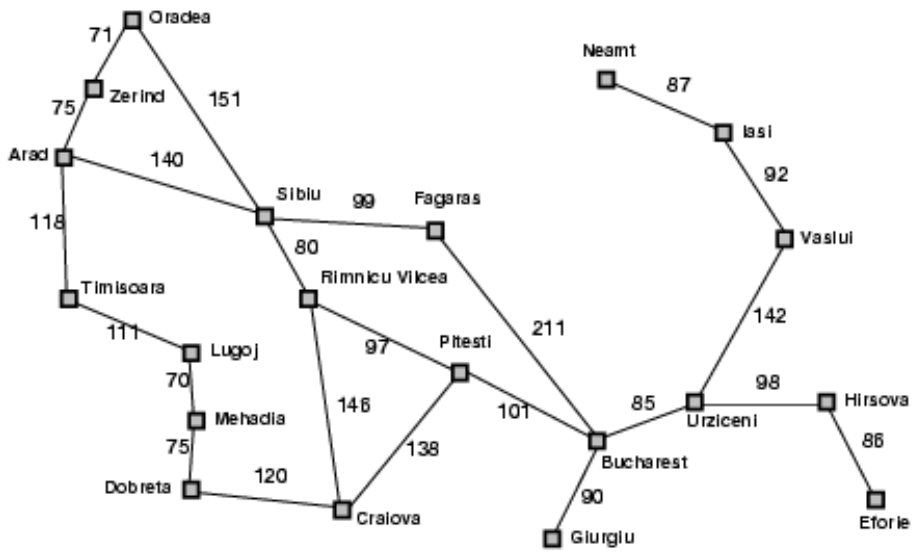
- **Hill Climbing, Local Search** Αναρρίχηση Λόφων ή Τοπική αναζήτηση
- **Simulated Annealing** Προσομοιωμένη Ανόπτηση
- **Local Beam Search** Τοπική Ακτινική Αναζήτηση
- **Γενετικοί Αλγόριθμοι**
- **Αναζήτηση Tabu**
- ***PSO, ACO, BCO, …***

Πληροφορημένη (ευριστική) αναζήτηση

- **Πληροφορία:** Μια **εκτίμηση** ή **μαντεψιά** του πόσο κοντά στο στόχο βρίσκεται μια κατάσταση
  - Συνάρτηση **αξιολόγησης *f(n)***, *n* μια κατάσταση
    - Σε BFS, DFS *f(n)* το βάθος του κόμβου
  - Μπορεί να εκληφθεί ως *κόστος*
    - *Σε UCS f(n) = g(n),* το κόστος του μονοπατιού μέχρι τον κόμβο.
  - Η *f(n)* περιλαμβάνει μια ***ευρετική συνάρτηση h(n)*** που αναπαριστά αυτήν την πληροφορία
- (greedy) **Best-First Search** (BestFS)
  - Αναζήτηση Πρώτα στο Καλύτερο
  - *h(n)* = εκτίμηση του κόστους του φθηνότερου μονοπατιού από τον κόμβο *n* στο στόχο
  - Στον BestFS απλώς ***f(n) = h(n)***
  - *(Στον A*: f(n) = g(n) + h(n))*

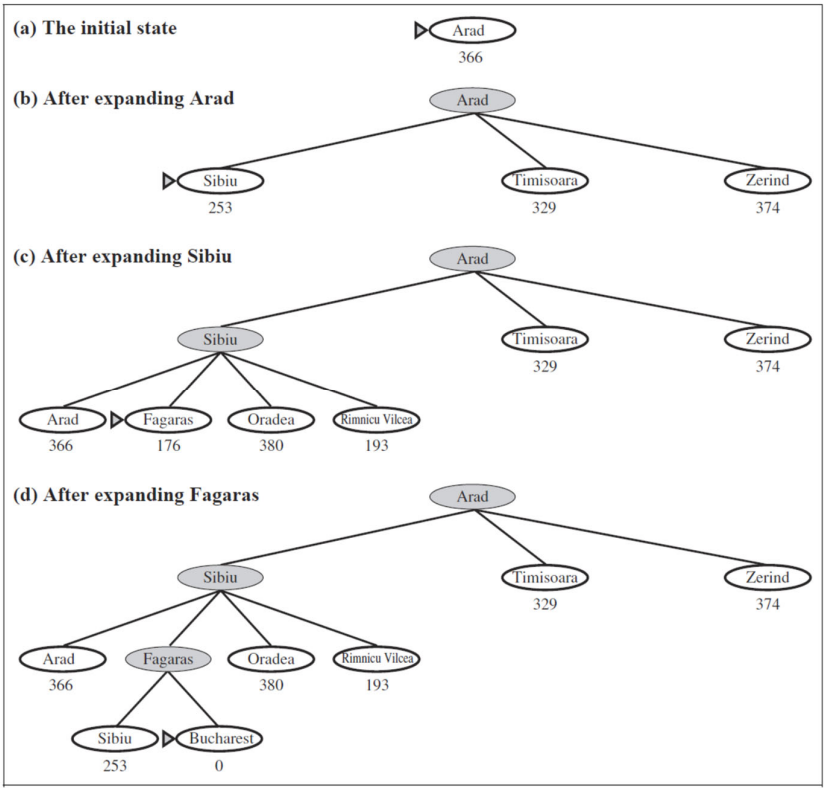
Αναζήτηση Πρώτα στο Καλύτερο (Best-First Search)

- Η επιλογή της ευριστικής καθορίζει τη σειρά επέκτασης των κόμβων στο μέτωπο αναζήτησης
- Π.χ. μια εκτίμηση στο πρόβλημα της διαδρομής είναι η ***απόσταση σε ευθεία γραμμή*** των πόλεων, h_{SLD}
- Κατά τ'άλλα, **στιγμιότυπο του UCS (στιγμιότυπο του BFS)**



Straight–line distance to Bucharest	
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	176
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	10
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

BestFS παράδειγμα



Ιδιότητες BestFS

- Βέλτιστος; Όχι
  - Π.χ. μπορεί να υπάρχει συντομότερο μονοπάτι
  - ***Άπληστος*** αλγόριθμος: τα *άμεσα* φθηνότερα βήματα δεν είναι πάντα τα καλύτερα
- Πλήρης; Ναι – αν Graph-Search
  Όχι – αν Tree-Search, μπορεί να εγκλωβιστεί σε κυκλικά μονοπάτια,
  - Πχ διαδρομή Iasi-Fagaras
  - *Iasi → Neamt → Iasi → Neamt → ….*
- Χρόνος; $O(b^m)$ – χειρότερη περίπτωση
  - *m* το μέγιστο βάθος του χώρου αναζήτησης
  - Μια καλή ευρετική μπορεί να επιφέρει μεγάλη βελτίωση
- Χώρος; $O(b^m)$ – όλοι οι κόμβοι στη μνήμη

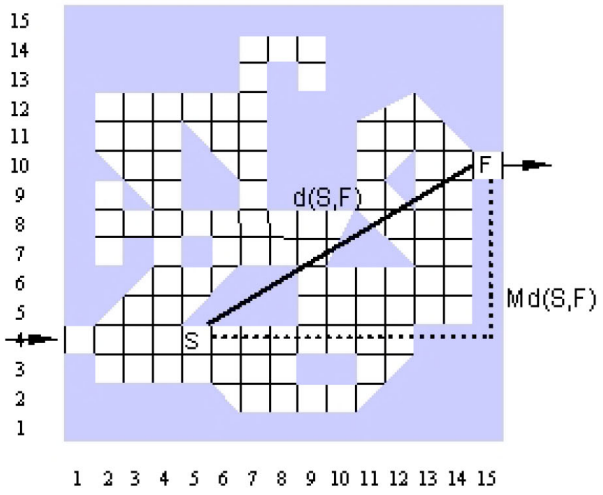
Ευριστικές συναρτήσεις στο Πρόβλημα του Λαβυρίνθου

- Ευκλείδεια απόσταση:

$$d(S,F) = \sqrt{(X_S - X_F)^2 + (Y_S - Y_F)^2}$$

- Απόσταση Manhattan:

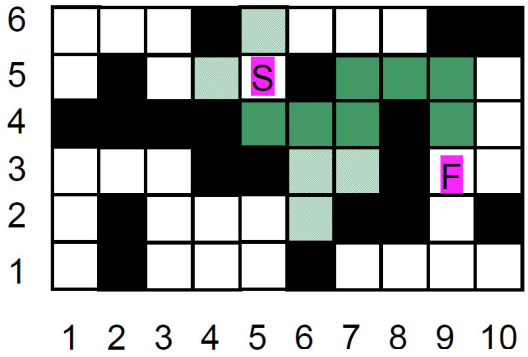
$$Md(S,F) = |X_S - X_F| + |Y_S - Y_F|$$



$d(S,F) = \sqrt{(5-15)^2 + (4-10)^2} = \sqrt{(100+36)} = 11{,}6$

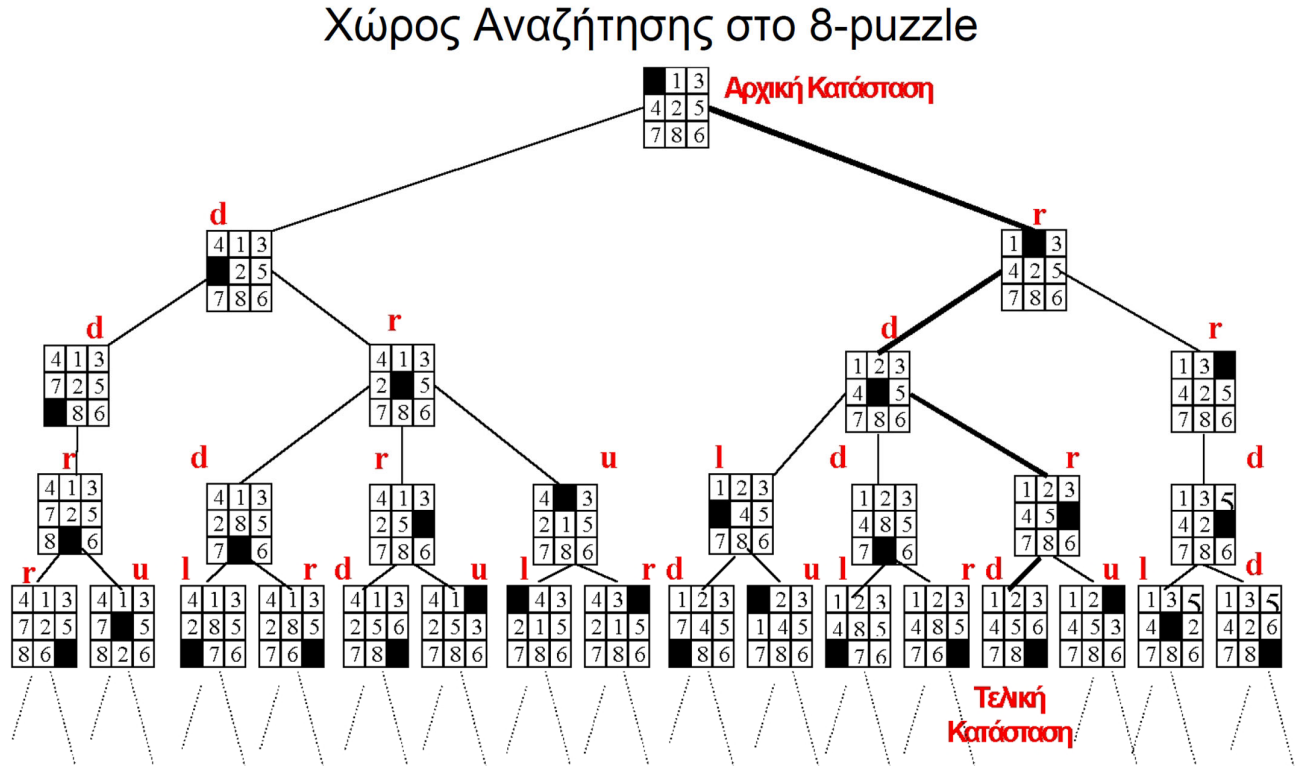
Md(S,F)=|5-15| + |4-10|=10+6=16.

Εφαρμογή BestFS στο πρόβλημα του λαβυρίνθου



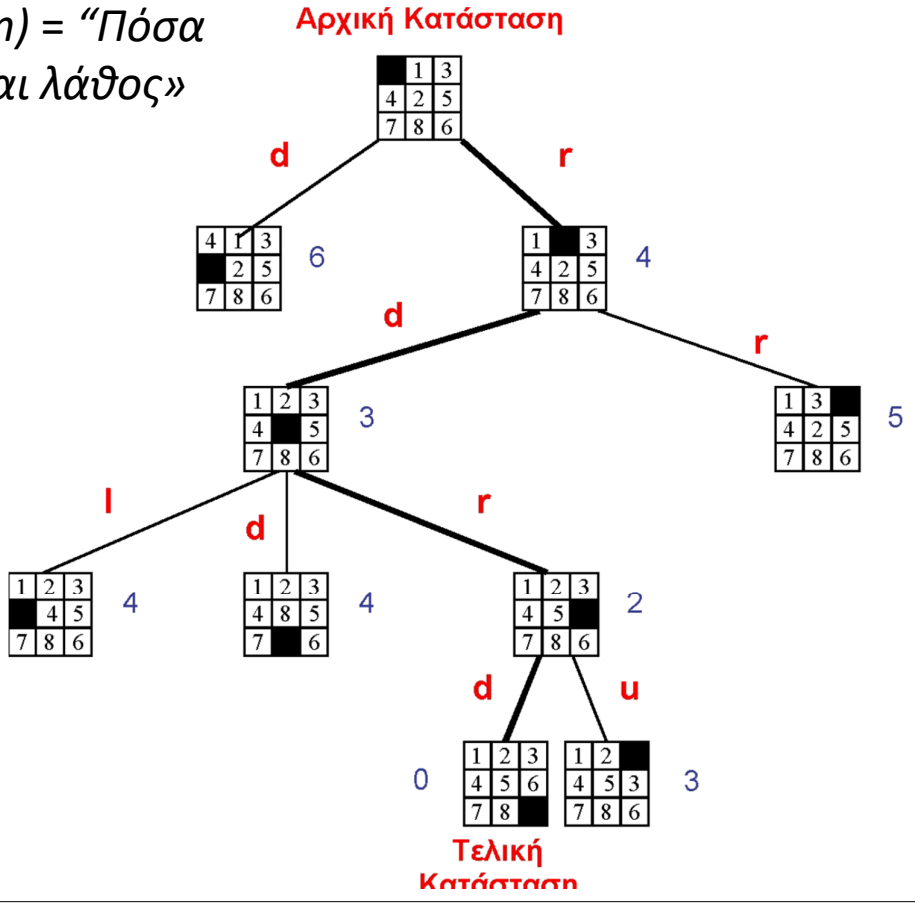
Μέτωπο Αναζήτησης	Κλειστό Σύνολο	Κατάσταση	Παιδιά
<5-5>	<>	5-5	5-4^5,5-6^7,4-5^7
<5-4^5,5-6^7,4-5^7>	<5-5>	5-4	5-5^6,6-4^4
<6-4^4,5-5^6,5-6^7,4-5^7>	<5-5,5-4>	6-4	5-4^7,6-3^3,7-4^3
<6-3^3,7-4^3,5-5^6,5-6^7,...>	<5-5,5-4,6-4>	6-3	6-4^4,6-2^3,7-3^2
<7-3^2,6-2^3,7-4^3,6-4^4,5-5^6,...>	<5-5,5-4,...>	7-3	6-3^3,6-4^4
<6-3^3,6-2^3,7-4^3,6-4^4,5-5^6,...>	<...,6-3,...>	6-3	Βρόχος
<6-2^3,7-4^3,6-4^4,5-5^6,5-6^7,...>	<...>	6-2	5-2^5,6-3^3
<7-4^3,6-4^4,5-2^5,...>	<...>	7-4	7-5^4,6-4^4,7-3^2
<7-3^2,7-5^4,6-4^4,5-2^5,...>	<..,7-3,..>	7-3	Βρόχος
<4-5^4,6-4^4,5-2^5,...>	<...>	7-5	7-4^3,8-5^3,7-6^5
<8-5^3,7-4^3,6-4^4,...>	<...>	8-5	8-6^4,7-5^4,9-5^2
<9-5^2,7-4^3,6-4^4,8-6^4,...>	<...>	9-5	8-5^3,9-4^1
<9-4^1,8-5^3,7-4^3,...>	<...>	9-4	9-3^0,9-5^2,10-4^2
<9-3^0,9-5^2,10-4^2,...>	<...>	9-3	ΤΕΛΙΚΗ ΚΑΤΑΣΤΑΣΗ

Εφαρμογή BestFS στο 8-puzzle



Εφαρμογή BestFS στο 8-puzzle

- Ευριστική *h(n)* = *"Πόσα πλακίδια είναι λάθος»*



Αλγόριθμος αναζήτησης A*

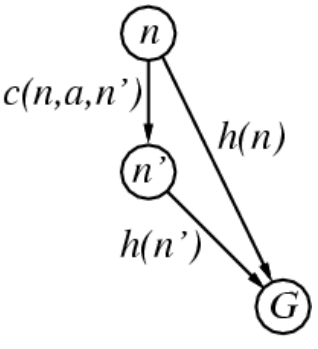
- Ιδέα: Ελαχιστοποίηση του ***συνολικού*** *εκτιμώμενου* κόστους
- Παραλλαγή BestFS, UCS
- Συνάρτηση αξιολόγησης *f(n)*:

$$f(n) = g(n) + h(n)$$

- *g(n)*: Το πραγματικό κόστος μέχρι τον κόμβο *n* στη διαδρομή
- *h(n)*: Εκτίμηση του κόστους του φθηνότερου μονοπατιού από τον *n* μέχρι τον στόχο (**π.χ.** h_{SLD})
- Κριτήρια για την ευριστική συνάρτηση h(n):
- **Παραδεκτή** (admissible) (← για Tree-Search)
  - Δεν υπερεκτιμά το κόστος: ∀n, $h(n) \leq h^*(n)$, όπου $h^*(n)$ το πραγματικό κόστος προς τον στόχο από την n.
  - *h(n) κάτω όριο* του πραγματικού κόστους
- **Συνεπής ή μονότονη** (consistent or monotonic) (← για Graph-Search)
  - $h(n) \leq c(n, a, n') + h(n')$, *n΄ διάδοχος του n* **(τριγωνική ανισότητα)**
  - *Αυστηρότερο* κριτήριο που συνεπάγεται το 1ο
- Τότε ο A* είναι και **βέλτιστος και πλήρης**

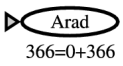
Αλγόριθμος αναζήτησης A* - κριτήρια

- Κριτήρια για την ευριστική συνάρτηση h(n):
- **Παραδεκτή** (admissible) (← για Tree-Search)
  - Δεν υπερεκτιμά το κόστος: ∀n, $h(n) \leq h^*(n)$, όπου $h^*(n)$ το πραγματικό κόστος προς τον στόχο από την n.
  - *h(n) κάτω όριο* του πραγματικού κόστους
- **Συνεπής ή μονότονη** (consistent or monotonic) (← για Graph-Search)
  - $h(n) \leq c(n, a, n') + h(n')$, *n΄ διάδοχος του n* **(τριγωνική ανισότητα)**
  - *Αυστηρότερο* κριτήριο που συνεπάγεται το 1ο
- Τότε ο A* είναι και **βέλτιστος και πλήρης**

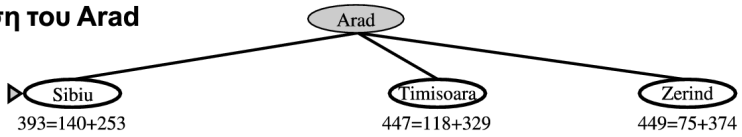


Παράδειγμα A*

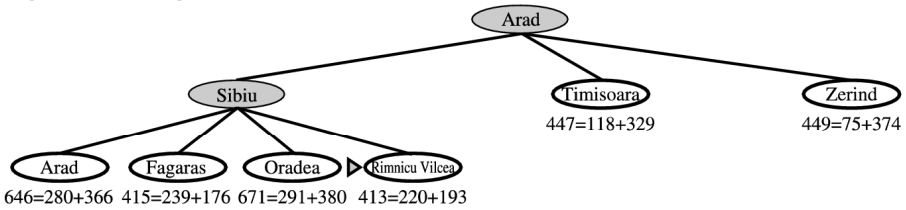
(α) Η αρχική κατάσταση



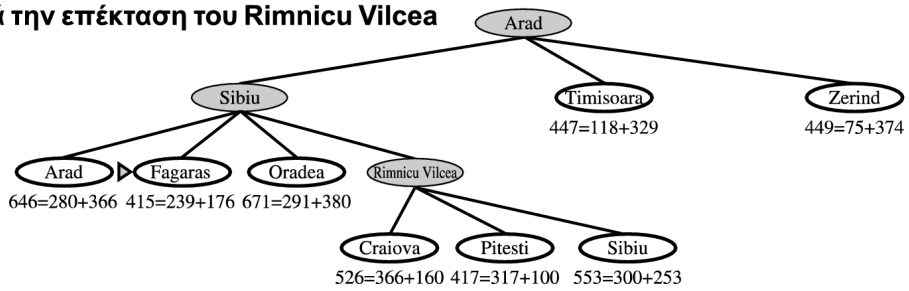
(β) Μετά την επέκταση του Arad



(γ) Μετά την επέκταση του Sibiu

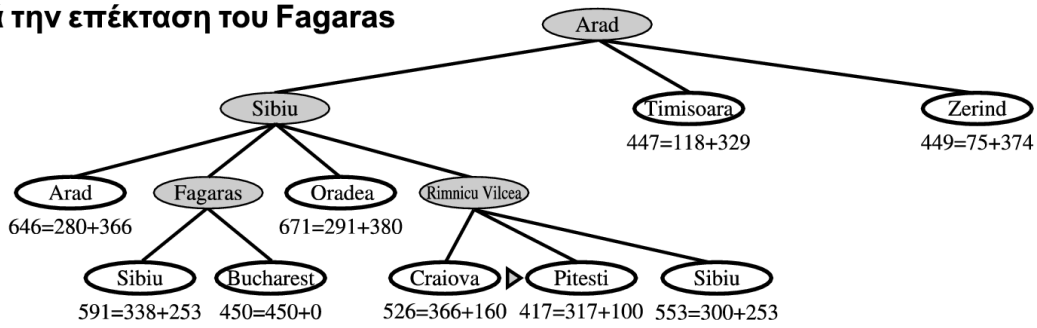


(δ) Μετά την επέκταση του Rimnicu Vilcea

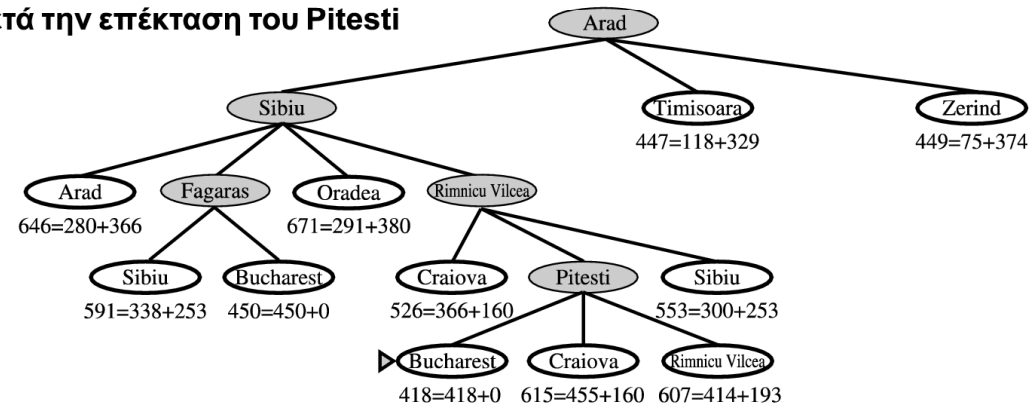


Παράδειγμα A*

(ε) Μετά την επέκταση του Fagaras



(στ) Μετά την επέκταση του Pitesti



Πληρότητα A*

- Είναι πλήρης και στις δύο εκδοχές (tree-search και graph-search)
  - Προϋπόθεση: Να μην υπάρχουν άπειροι κόμβοι με εκτίμηση κόστους $f(n) <= C^*$. Αρκεί:
    - Κόστος βήματος $c > \varepsilon$, ε θετική σταθερά (όπως UCS)
    - b πεπερασμένο (όπως UCS, BFS)
  - Ακόμα και σε κυκλικά μονοπάτια (Tree-search);
    - Ναι, γιατί κάποια στιγμή μια επανάληψη ενός κύκλου θα γίνει ακριβότερη ($c > \varepsilon$)
    - Όμως: περισσότερος χρόνος

Πληρότητα A*, βέλτιστη αποδοτικότητα

- Υπάρχει περίπτωση να επεκτείνει κόμβο με $f(n) > C^*$;
  - Τότε σίγουρα ο n δεν είναι πάνω στο βέλτιστο μονοπάτι
    - Αν ήταν, δεν μπορεί f(n) > C* γιατί h **παραδεκτή**
  - Άρα υπάρχει βέλτιστο μονοπάτι που δεν περνά από τον n
  - Έστω n' κόμβος του βέλτιστου μονοπατιού. Θα είναι $f(n') < C^* < f(n)$
  - Άρα όλοι οι n' θα επεκταθούν πριν από τον n, και θα βρεθεί η λύση χωρίς ποτέ να επεκταθεί ο n !
  - Οι καταστάσεις που είναι σίγουρα μη-βέλτιστες **κλαδεύονται** (pruned), π.χ. *Timisoara, Zerind, Oradea*
- Ο αλγόριθμος είναι **βέλτιστα αποδοτικός** (optimally efficient)
  - Ο A* επεκτείνει όλους τους κόμβους με $f(n) \leq C^*$
  - Δεν υπάρχει άλλος βέλτιστος αλγόριθμος, με την ίδια ***h(n)***, που να επεκτείνει λιγότερους κόμβους
  - Αν επέκτεινε λιγότερους από όσους ισχύει $f(n) \leq C^*$ μπορεί να έχανε τον βέλτιστο

Βελτιστότητα A*

Βελτιστότητα A* Tree-Search

- Υπάρχει περίπτωση να βρει πρώτα το στόχο από μη-βέλτιστο μονοπάτι;
  - Εφόσον μη-βέλτιστο $f(G) > C^*$
  - Δεν θα επιλεγεί πότε για επέκταση, ώστε να ελεγχθεί
  - Οι κόμβοι n' του βέλτιστου μονοπατιού θα έχουν επεκταθεί νωρίτερα, αφού $f(n') < C^*$ (***h* παραδεκτή**, όπως πριν)
  - **Η παραδεκτότητα εγγυάται ότι θα βρούμε το συντομότερο μονοπάτι στο στόχο**

Όμως μπορεί να χρειαστεί να ξαναεπισκεφθούμε έναν κόμβο

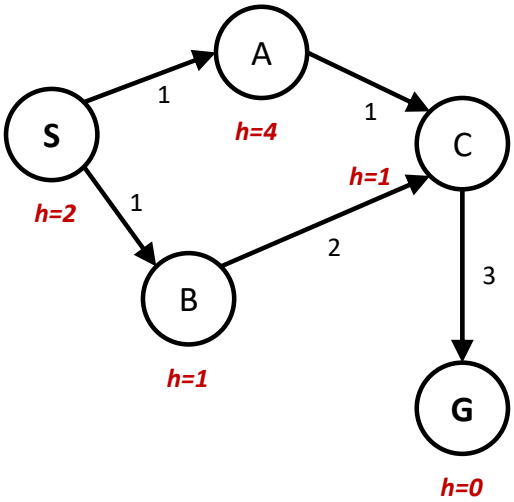
Βελτιστότητα A* Graph-Search

- Υπάρχει περίπτωση ένας κόμβος n που επεκτείνεται να μη βρίσκεται στο βέλτιστο μονοπάτι του;
  - Εφόσον μη-βέλτιστο, θα υπάρχει άλλος κόμβος n' που θα οδηγεί στον n
  - Όμως $f(n') < f(n)$ (***f* συνεπής**, $f(n) = g(n)+h(n) = g(n')+\mathbf{c(n',n)+h(n)} > g(n')+\mathbf{h(n')} = f(n')$
  - Άρα ο n' θα είχε επεκταθεί νωρίτερα
  - **Η συνέπεια εγγυάται επιπλέον ότι όταν ένας κόμβος επιλέγεται για επέκταση, έχουμε βρει το βέλτιστο μονοπάτι του**

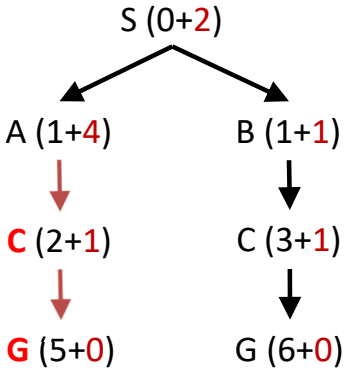
Δεν χρειάζεται να ξαναεπισκεφθούμε κανέναν άλλο κόμβο

A* Graph-Search: Γιατί χρειάζεται η συνέπεια;

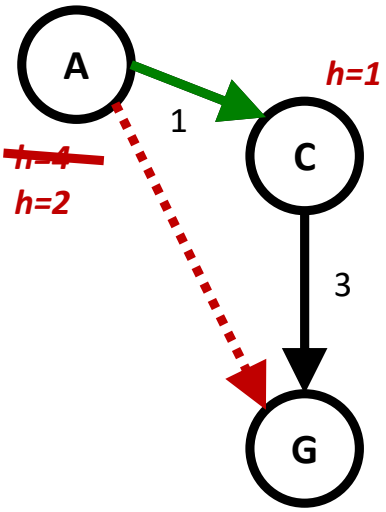
Χώρος καταστάσεων



Δέντρο αναζήτησης (graph search)



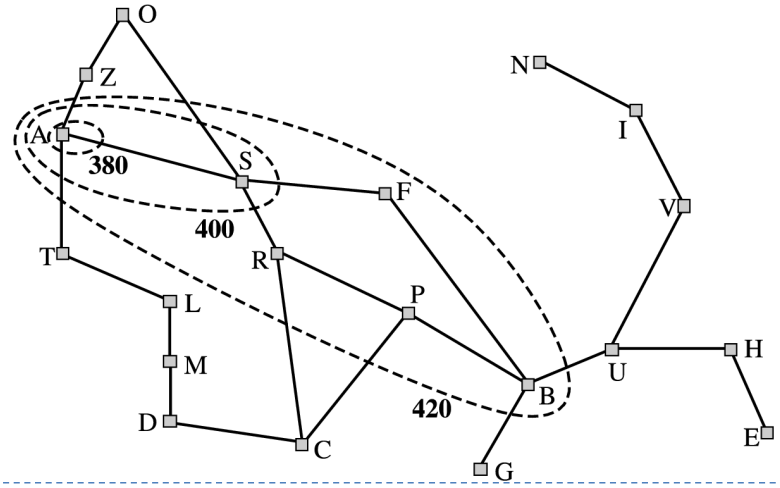
A* Graph-Search: Γιατί χρειάζεται η συνέπεια;



- *h* είναι παραδεκτή
  - Ποτέ δεν υπερεκτιμά το κόστος
- Αν *h(A)* = 4, η *h* δεν είναι συνεπής!
  - *h(A)* ***μεγαλύτερο από*** *c(A,C) + h(C) = 1 + 1 = 2*
  - Προσθέτουμε ακμή (*c(A,C)*) και η εκτίμηση καλυτερεύει!!
- *h* συνεπής, αν π.χ. h(A) = 2
  - *h(A) ≤ c(A,C) + h(C)*
  - Ποτέ δεν μειώνεται το εκτ. κόστος *f* όταν προσθέτουμε ακμές

Ισοϋψείς καμπύλες (contours)

- Αν η *h(n)* είναι συνεπής, τότε οι τιμές της *f(n)* πάνω σε οποιαδήποτε διαδρομή είναι ***μη φθίνουσες***.
- Ο A* σταδιακά ξεκινά από το f(n) της ρίζας και αυξάνει την **ακτίνα αναζήτησης**
- Οι κόμβοι που βρίσκονται μέσα στην ακτίνα έχουν το ίδιο ή μικρότερο *f-κόστος*
- Στον UCS *f(n) = g(n)* : *η ακτίνα ταυτίζεται με την απόσταση από τη ρίζα* (ομόκεντρες ζώνες ως προς την απόσταση)
- Στον A*η ακτίνα καμπυλώνει προς το βέλτιστο μονοπάτι, όσο πιο ακριβής είναι η *h(n)* (ομόκεντρες ζώνες ως προς *f(n)*)



Ιδιότητες A*

- Βέλτιστος; Ναι
  - Η ευρετική *h(n)* να είναι *παραδεκτή*
  - Η ευρετική *h(n)* να είναι *συνεπής* (χωρίς επανεξέταση κόμβων)
- Πλήρης; Ναι
- Χρόνος; $O(b^d)$ – χειρότερη περίπτωση
  - *d* το βάθος της λύσης (όχι αναγκαστικά της *ρηχότερης*)
  - Π.χ. η ευρετική δεν είναι καλή ή ίδια για όλους ή h(n) = 0 **(UCS)**
  - Μπορεί όμως και *πολυωνυμική*, λόγω κλαδέματος και μη ύπαρξης κύκλων. Εξαρτάται πολύ από την *h(n)*
- Χώρος; $O(b^d)$ – όλοι οι κόμβοι στη μνήμη *μειονέκτημα*
  - Αν Tree-Search; Λιγότερη μνήμη αλλά αύξηση χρόνου
  - Συχνή επέκταση των ίδιων (επαναλαμβανόμενων) καταστάσεων
  - Το πρόβλημα μπορεί να γίνει *δυσεπίλυτο* (intractable)
  - Υπάρχει έντονο *tradeoff* ανάμεσα σε χρόνο/χώρο

A* με όριο μνήμης

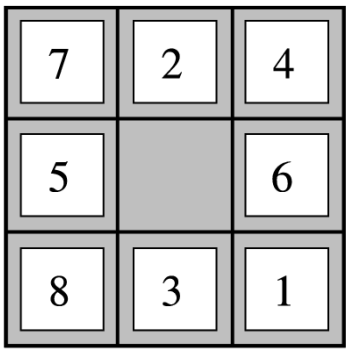
- IDA* (Iterative Deepening A*)
  - Όπως στην *επαναληπτική εκβάθυνση* (ID), αλλά κάθε φορά αυξάνουμε το μέγιστο επιτρεπτό *f(n)*, όχι το βάθος, μέχρι να βρεθεί λύση
  - Θα αποφύγει την επαναληπτική επέκταση των ίδιων καταστάσεων, όμως έχει το overhead των επαναληπτικών εκτελέσεων
  - Μπορεί να ξανα-δημιουργηθούν υπερβολικά πολλοί κόμβοι
- SMA* (Simplified Memory-bounded A*)
  - Κρατά τους κόμβους στη μνήμη (*graph-search*), αλλά μέχρι ένα όριο
  - Όταν γεμίσει, διαγράφει αυτόν με το μεγαλύτερο *f(n)* και άρα όλο το υποδένδρο του
  - Αποθηκεύει την τιμή του *f(n)* στον γονέα του
  - Θα ξαναδημιουργήσει το υποδένδρο μόνο όταν τα άλλα μονοπάτια αποδειχθούν χειρότερα
  - Είναι βέλτιστος και πλήρης, αν το βάθος *d* της βέλτιστης λύσης χωρά στη μνήμη
  - Αν όχι, μπορεί να ταλαντώνεται ανάμεσα στα υποδένδρα

Ευριστικές συναρτήσεις για το 8-puzzle

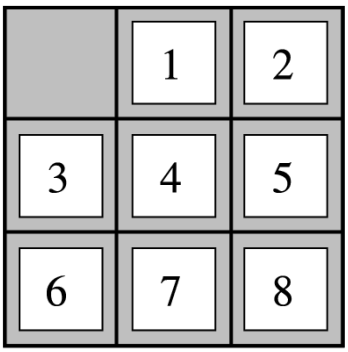
h_1=8

h_2=18

h*=26



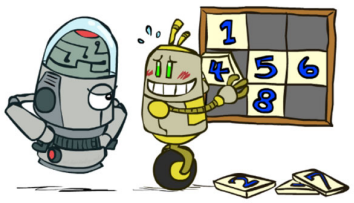
Αρχική κατάσταση



Κατάσταση στόχου

- h_1 = ο αριθμός των πλακιδίων που είναι λάθος (Απόσταση Hamming)
- h_2 = το άθροισμα των αποστάσεων των πλακιδίων από τις θέσεις προορισμού τους (Απόσταση Manhattan)
- Παραδεκτές και οι δύο (γιατί;)
- Η h_2 **κυριαρχεί** της h_1. Πάντα $h_2(n) \geq h_1(n)$
  - Η h_2 θα επεκτείνει λιγότερους κόμβους
  - Επεκτείνονται όλοι με $f(n) = g(n) + h(n) \leq C^*$
  - Υπάρχει χώρος για περισσότερους αν $h(n)$ μικρότερο

Ευριστικές και χαλαρωμένα προβλήματα



- Πώς προκύπτουν οι δύο προηγούμενες συναρτησεις;
- Είναι **ακριβή κόστη** για χαλαρωμένες εκδοχές του 8-puzzle
  - h_1: Επιτρέπεται μετακίνηση σε οποιαδήποτε θέση σε ένα βήμα
  - h_2: Επιτρέπεται μετακίνηση σε οποιαδήποτε γειτονική θέση
- Ο χαλαρωμένος χώρος καταστάσεων είναι υπερ-γράφος του αρχικού
  - Έχει περισσότερες ακμές/συνδέσεις
  - Μια βέλτιστη λύση του αρχικού είναι επίσης λύση στο χαλαρωμένο (οι στόχοι είναι ίδιοι)
  - Αλλά μπορεί να έχει και καλύτερες λόγω νέων συντομότερων μονοπατιών/γεφυρών
  - ***Άρα το κόστος μιας βέλτιστης λύσης του χαλαρωμένου είναι παραδεκτή ευριστική για το αρχικό*** (μικρότερο κόστος)
  - *Και επειδή είναι ακριβές κόστος ικανοποιεί την τριγωνική ανισότητα και **άρα είναι και συνεπής***

 Πηγές: S. Russell, P. Norvig "Artificial Intelligence: A modern Approach" 3d ed., Ι. Βλαχάβας κ.α. «Τεχνητή Νοημοσύνη» 4η έκδοση, ai.berkeley.edu