Τεχνητή Νοημοσύνη

Ενότητα 3: Αναζήτηση

Αν. καθηγητής Στεργίου Κωνσταντίνος
kstergiou@uowm.gr

Τμήμα Μηχανικών Πληροφορικής και Τηλεπικοινωνιών

BY NC ND
Ευρωπαϊκή Ένωση
ΕΠΙΧΕΙΡΗΣΙΑΚΟ ΠΡΟΓΡΑΜΜΑ ΕΚΠΑΙΔΕΥΣΗ ΚΑΙ ΔΙΑ ΒΙΟΥ ΜΑΘΗΣΗ
ΥΠΟΥΡΓΕΙΟ ΠΑΙΔΕΙΑΣ & ΘΡΗΣΚΕΥΜΑΤΩΝ, ΠΟΛΙΤΙΣΜΟΥ & ΑΘΛΗΤΙΣΜΟΥ
ΕΙΔΙΚΗ ΥΠΗΡΕΣΙΑ ΔΙΑΧΕΙΡΙΣΗΣ
Με τη συγχρηματοδότηση της Ελλάδας και της Ευρωπαϊκής Ένωσης
ΕΣΠΑ 2007-2013

Άδειες Χρήσης

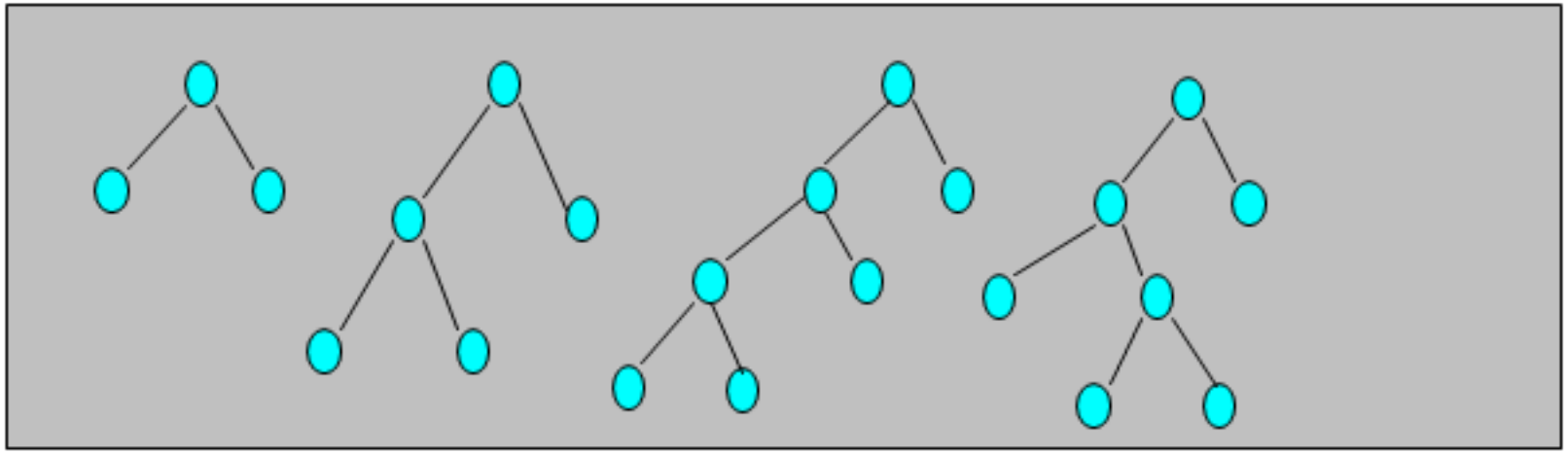
- Το παρόν εκπαιδευτικό υλικό υπόκειται σε άδειες χρήσης Creative Commons.
- Για εκπαιδευτικό υλικό, όπως εικόνες, που υπόκειται σε άλλου τύπου άδειας χρήσης, η άδεια χρήσης αναφέρεται ρητώς.

Χρηματοδότηση

- Το παρόν εκπαιδευτικό υλικό έχει αναπτυχθεί στα πλαίσια του εκπαιδευτικού έργου του διδάσκοντα.
- Το έργο «**Ανοικτά Ψηφιακά Μαθήματα στο Πανεπιστήμιο Δυτικής Μακεδονίας**» έχει χρηματοδοτήσει μόνο τη αναδιαμόρφωση του εκπαιδευτικού υλικού.
- Το έργο υλοποιείται στο πλαίσιο του Επιχειρησιακού Προγράμματος «Εκπαίδευση και Δια Βίου Μάθηση» και συγχρηματοδοτείται από την Ευρωπαϊκή Ένωση (Ευρωπαϊκό Κοινωνικό Ταμείο) και από εθνικούς πόρους.

Ευρωπαϊκή Ένωση
Ευρωπαϊκό Κοινωνικό Ταμείο
ΕΠΙΧΕΙΡΗΣΙΑΚΟ ΠΡΟΓΡΑΜΜΑ
ΕΚΠΑΙΔΕΥΣΗ ΚΑΙ ΔΙΑ ΒΙΟΥ ΜΑΘΗΣΗ
επένδυση στην κοινωνία της γνώσης
ΥΠΟΥΡΓΕΙΟ ΠΑΙΔΕΙΑΣ ΚΑΙ ΘΡΗΣΚΕΥΜΑΤΩΝ
ΕΙΔΙΚΗ ΥΠΗΡΕΣΙΑ ΔΙΑΧΕΙΡΙΣΗΣ
ΕΣΠΑ 2007-2013
πρόγραμμα για την ανάπτυξη
ΕΥΡΩΠΑΪΚΟ ΚΟΙΝΩΝΙΚΟ ΤΑΜΕΙΟ
Με τη συγχρηματοδότηση της Ελλάδας και της Ευρωπαϊκής Ένωσης

Αναζήτηση (Search)



Ευριστικοί Αλγόριθμοι Αναζήτησης

- **Ευριστικοί Μηχανισμοί** *(Heuristics).*
- **Αναζήτηση Πρώτα στο Καλύτερο** *(Best-First Search).*
- **Αλγόριθμος Α*.**
- **Ιδιότητες Ευριστικών Συναρτήσεων.**
- **Αλγόριθμοι Επαναληπτικής Βελτίωσης** *(Iterative Improvement Algorithms).*
- **Τοπική Αναζήτηση** *(Local Search):*
  - ✓ **Hill-climbing.**
  - ✓ **Simulated Annealing.**
  - ✓ **Local Beam Search.**
  - ✓ **Genetic Algorithms.**

Ευριστικοί Μηχανισμοί

- Όλοι οι αλγόριθμοι τυφλής αναζήτησης που συζητήσαμε έχουν χρονική πολυπλοκότητα της τάξης του $O(b^d)$ ή κάτι παρόμοιο.
  - Και η απόδοση τους δεν είναι αποδεκτή σε πραγματικά προβλήματα!
- Σε μεγάλους χώρους αναζήτησης μπορούμε να πετύχουμε πολύ καλύτερη απόδοση αν εκμεταλλευτούμε **γνώση σχετικά με το συγκεκριμένο πρόβλημα** που επιλύουμε *(domain specific knowledge).*
- Οι **Ευριστικοί μηχανισμοί *(heuristics)*** είναι κανόνες βασισμένοι σε τέτοιου είδους γνώση σύμφωνα με τους οποίους επιλέγουμε τον επόμενο κόμβο για επέκταση κατά τη διάρκεια της αναζήτησης.

Αναζήτηση Πρώτα στο Καλύτερο

- Ένας αλγόριθμος τυφλής αναζήτησης θα μπορούσε να βελτιωθεί αν ξέραμε κάθε στιγμή ποιος είναι ο καλύτερος κόμβος για επέκταση *(ή ποιος "φαίνεται" να είναι ο καλύτερος).*

function BestFirstSearch (problem, Eval-Fn)
returns a solution sequence or failure
    Queuing-Fn ← a function that orders nodes in ascending
        order of Eval-Fn
    return TreeSearch (problem, Queuing-Fn)

- Η συνάρτηση Eval-Fn ονομάζεται **συνάρτηση αποτίμησης** *(evaluation function).*

Συναρτήσεις Αποτίμησης & Ευριστικές Συναρτήσεις

- Υπάρχει μια ολόκληρη οικογένεια από αλγόριθμους best-first αναζήτησης με διαφορετικές συναρτήσεις αποτίμησης.
- Ένα βασικό συστατικό αυτών των αλγορίθμων είναι μια **ευριστική συνάρτηση** *(heuristic function) h*, τέτοια ώστε:
  - $h(n)$ είναι το εκτιμώμενο κόστος του φτηνότερου μονοπατιού από την κατάσταση στον κόμβο *n* ως μια κατάσταση στόχου.
- Η *h* μπορεί να είναι οποιαδήποτε συνάρτηση τέτοια ώστε $h(n) = 0$ αν το *n* αντιστοιχεί σε κατάσταση στόχου.
- Για να βρούμε όμως μια καλή ευριστική συνάρτηση χρειαζόμαστε **πληροφορία σχετικά με το συγκεκριμένο πρόβλημα.**

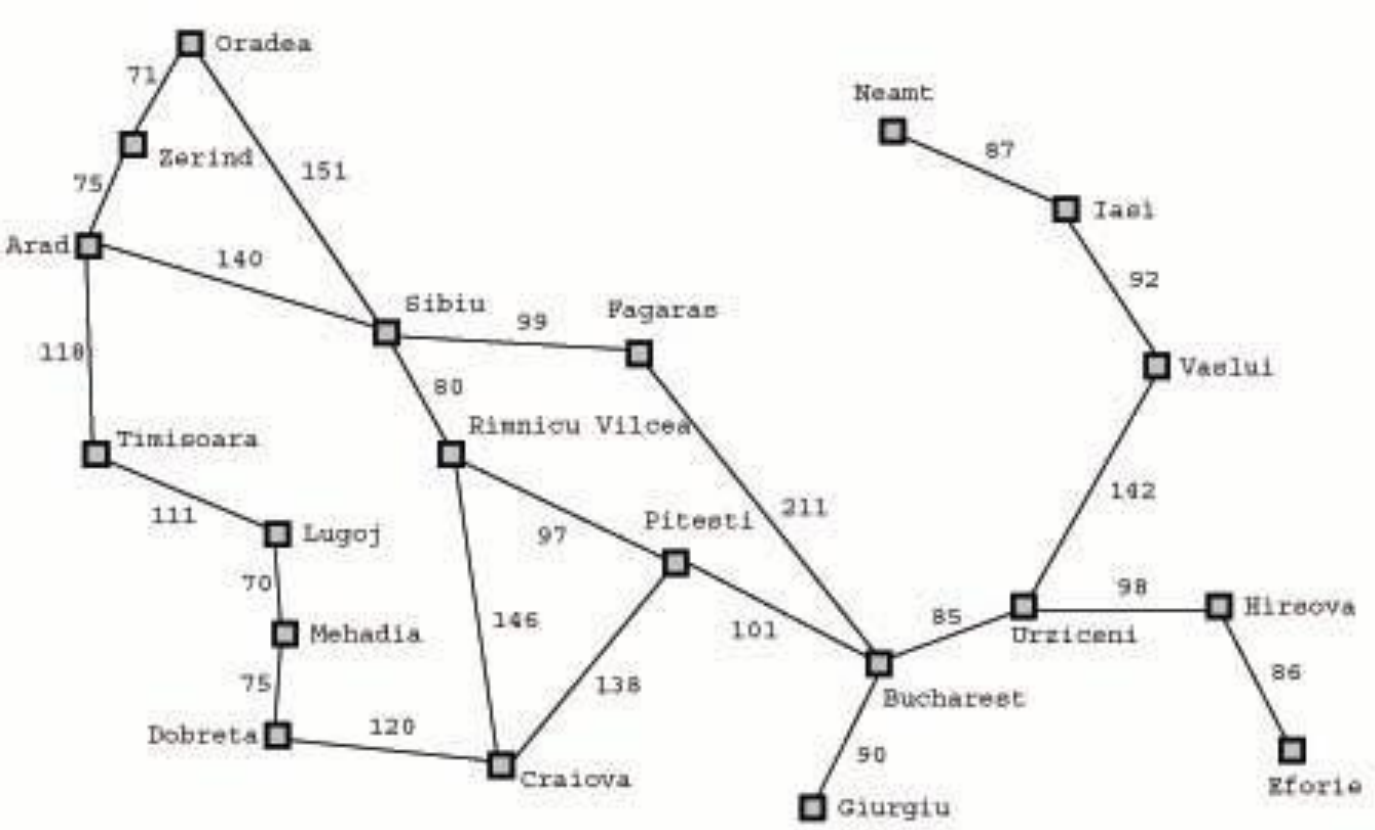
Άπληστη Αναζήτηση Πρώτα στο Καλύτερο (1/2)

- Στην **άπληστη αναζήτηση πρώτα στο καλύτερο** *(greedy best-first search)* οι κόμβοι αποτιμώνται χρησιμοποιώντας μόνο την ευριστική συνάρτηση, δηλ. $f(n) = h(n)$.

function GreedyBestFirstSearch (problem)
returns a solution sequence or failure
    return TreeSearch (problem, h)

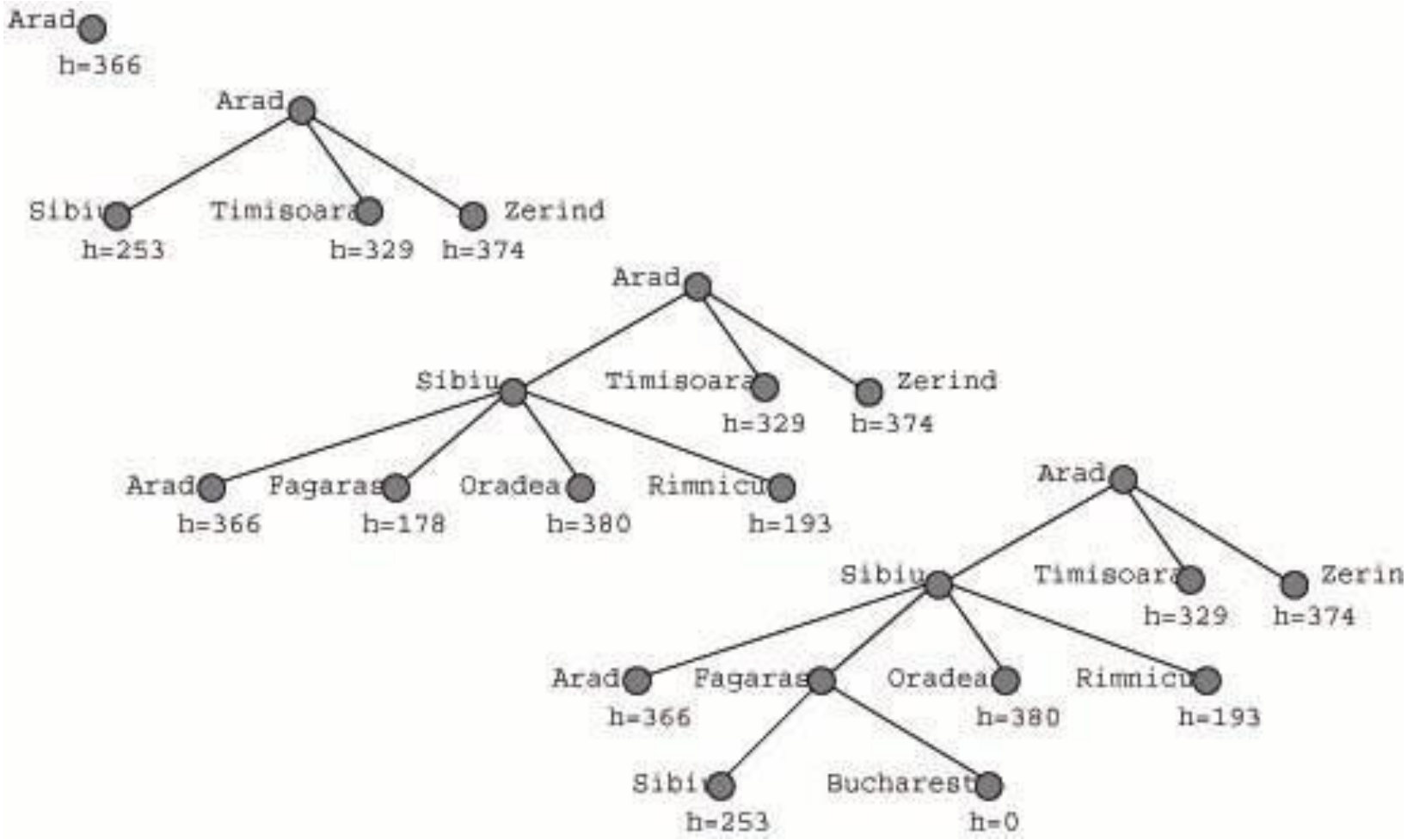
- Ο αλγόριθμος αυτός ονομάζεται **άπληστος** γιατί προσπαθεί πάντα να μειώσει όσο το δυνατό περισσότερο το κόστος που απομένει μέχρι να φτάσουμε στο στόχο.

Παράδειγμα (1/2)



Straight-line dis 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	98
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

Παράδειγμα (2/2)



Άπληστη Αναζήτηση
Πρώτα στο Καλύτερο (2/2)

- Αξιολόγηση:
  - **Πλήρης;** Όχι.
    - Σκεφτείτε το πρόβλημα να πας από το Iasi στο Fagaras.
  - **Χρονική πολυπλοκότητα** – $O(b^m)$.
    - m είναι το μέγιστο βάθος του δέντρου αναζήτησης.
  - **Χωρική πολυπλοκότητα** – $O(b^m)$.
  - **Βρίσκει την βέλτιστη λύση;** Όχι πάντα.

- Μια καλή επιλογή ευριστικής συνάρτησης *h* μπορεί όμως να μειώσει το χρόνο και το χώρο δραστικά.

Ο Αλγόριθμος Αναζήτησης A* (1/2)

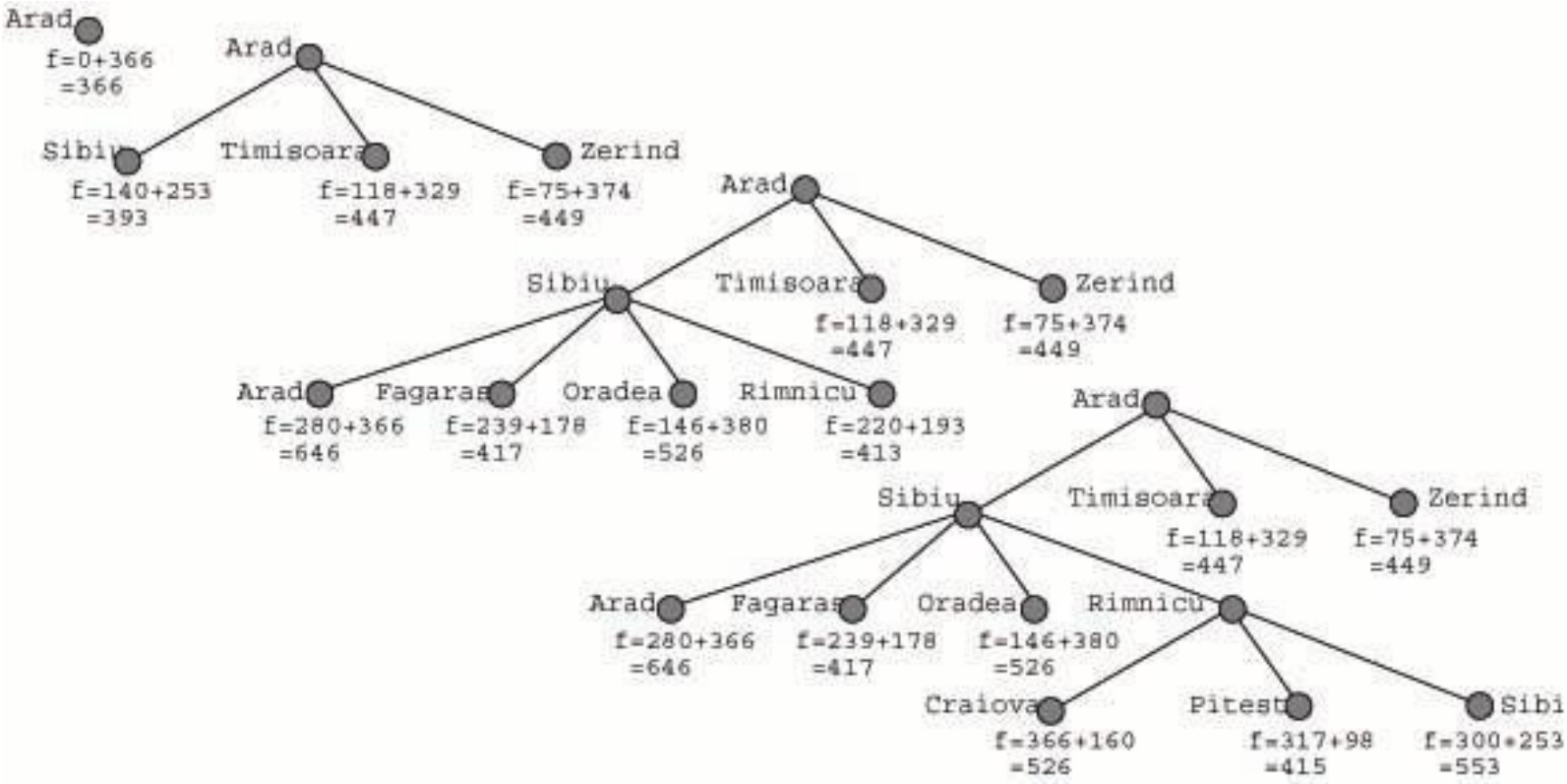
- **Greedy Best-First Search:**
  - Αναζητά προσπαθώντας να ελαχιστοποιήσει το εκτιμώμενο κόστος $h(n)$ ως τον στόχο.
  - Δεν είναι πλήρης και δεν εγγυάται ότι θα βρει τη βέλτιστη λύση.
- **Uniform Cost Search:**
  - Αναζητά προσπαθώντας να ελαχιστοποιήσει το κόστος του μονοπατιού $g(n)$ από τη ρίζα ως τον τρέχων κόμβο.
  - Είναι πλήρης και εγγυάται ότι θα βρει τη βέλτιστη λύση.
- Μπορούμε να συνδυάσουμε τους δύο αλγόριθμους;

Ο Αλγόριθμος Αναζήτησης A* (2/2)

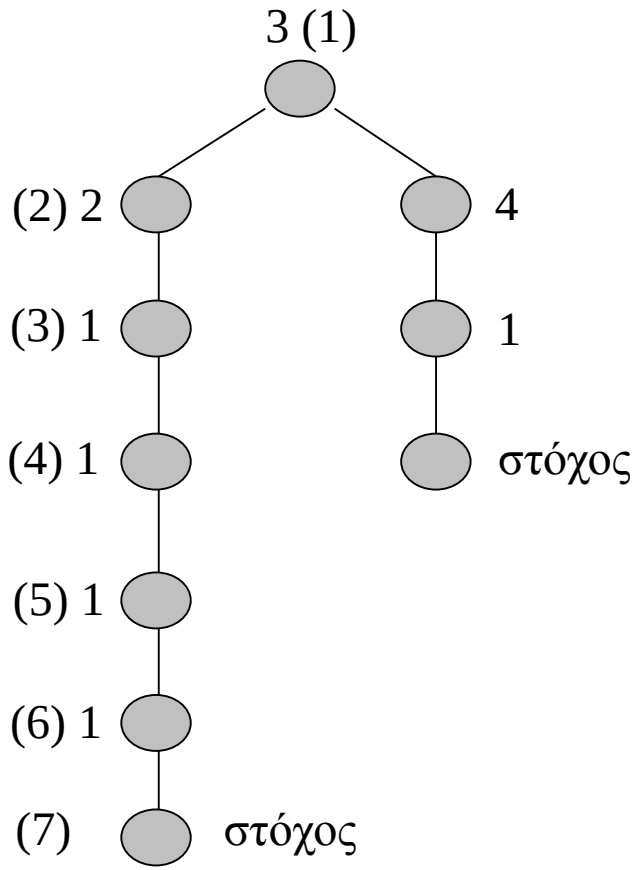
- Ο A* είναι ένας best-first αλγόριθμος αναζήτησης με συνάρτηση αποτίμησης $f(n) = g(n) + h(n)$.
- Σε αυτή την περίπτωση $f(n)$ είναι το εκτιμώμενο κόστος της φτηνότερης λύσης που περνάει από το *n*.

function A*Search (problem)
returns a solution sequence or failure
    return BestFirstSearch (problem, g+h)

Ο A* πάει στο Βουκουρέστι

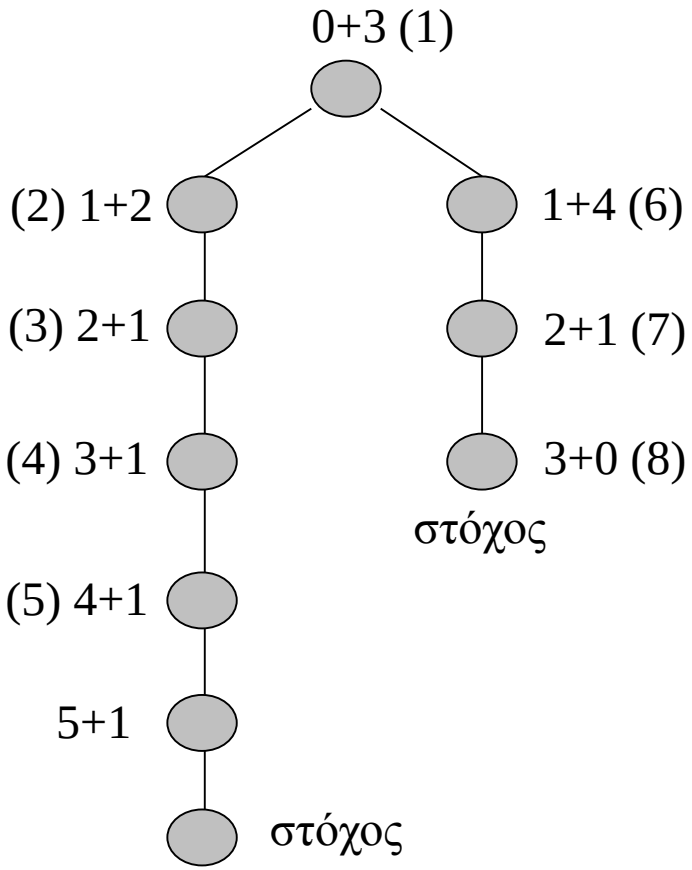


A* - Παράδειγμα (1/2)



- Ο αριθμός δίπλα σε κάθε κόμβο είναι η τιμή της ευριστικής συνάρτησης.
- Οι αριθμοί σε παρενθέσεις δείχνουν τη σειρά επίσκεψης των κόμβων από τον αλγόριθμο best-first search.

A* - Παράδειγμα (2/2)



- Ο πρώτος αριθμός στο άθροισμα είναι η απόσταση από τον αρχικό κόμβο και ο δεύτερος η τιμή της ευριστικής συνάρτησης (εκτιμώμενη απόσταση από το στόχο).

- Οι αριθμοί σε παρενθέσεις δείχνουν τη σειρά επίσκεψης των κόμβων από τον αλγόριθμο A*.

Ο Αλγόριθμος Α* (1/2)

- Ας υποθέσουμε ότι :
  - Η συνάρτηση *h* είναι επιλεγμένη έτσι ώστε ποτέ να μην υπερεκτιμάει το κόστος της αναζήτησης μέχρι το στόχο.
    - ✓ Μια τέτοια ευριστική συνάρτηση λέγεται **αποδεκτή ευριστική συνάρτηση** *(admissible heuristic)*.
    - ✓ Αν η *h* είναι αποδεκτή τότε η f(*n*) ποτέ δεν υπερεκτιμάει το πραγματικό κόστος της καλύτερης λύσης που περνάει από το *n.*
  - Ο παράγοντας διακλάδωσης *b* είναι πεπερασμένος.
  - Κάθε ενέργεια κοστίζει τουλάχιστον $\delta > 0$.

Ο Αλγόριθμος A* (2/2)

- **Αξιολόγηση (με τις προηγούμενες παραδοχές):**
  - **Πλήρης;** Ναι.
  - **Χρονική πολυπλοκότητα** – Εκθετική.
    - Όμως ο A* προσφέρει **τεράστια βελτίωση** συγκριτικά με τυφλή αναζήτηση.
  - **Χωρική πολυπλοκότητα** – $O(b^d)$.
    - Το βασικό μειονέκτημα του A*.
  - **Βρίσκει την βέλτιστη λύση;** Ναι.

Ο A* είναι Βέλτιστος και Πλήρης (1/4)

- **Ο A* βρίσκει πάντα την βέλτιστη λύση**
- **Απόδειξη:**
  - Ας υποθέσουμε ότι το κόστος της βέλτιστης λύσης είναι C* και μη-βέλτιστος κόμβος-λύση G_2 υπάρχει στο μέτωπο αναζήτησης. Επειδή ο G_2 είναι μη-βέλτιστος και $h(G_2) = 0$, έχουμε: $\boldsymbol{f(G_2) = g(G_2) + h(G_2) = g(G_2) > C^*}$.
  - Τώρα θεωρήστε έναν κόμβο *n* στο μέτωπο αναζήτησης που ανήκει στο βέλτιστο μονοπάτι. Επειδή η *h* δεν υπερεκτιμά το κόστος ως τον στόχο, έχουμε:

    $$\boldsymbol{f(n) = g(n) + h(n) \leq C^*}$$

  - Άρα ο G_2 δεν θα επιλεχθεί για επέκταση!

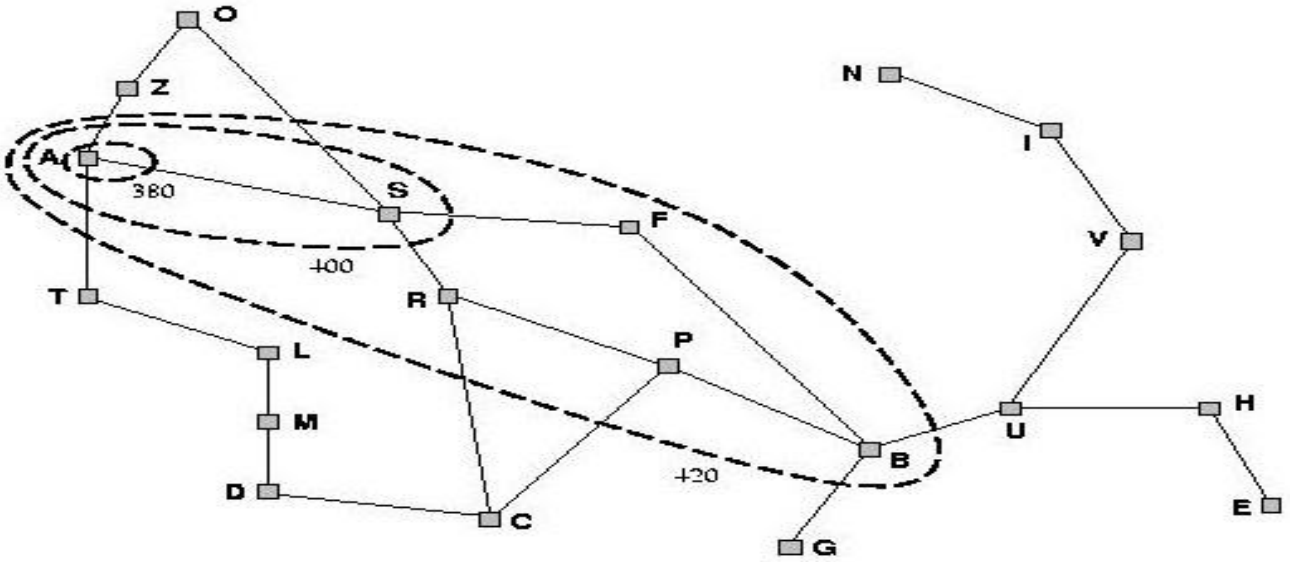
Ο A* είναι Βέλτιστος και Πλήρης (2/4)

- Γενικά για να εγγυηθούμε την εύρεση της βέλτιστης λύσης (στην περίπτωση ύπαρξης επαναλαμβανόμενων καταστάσεων) πρέπει η ευριστική συνάρτηση *h* να είναι **συνεπής** (consistent) (ή αλλιώς **μονοτονική** (monotonic)).

- **Ορισμός:**
  - Ένα heuristic *h* ονομάζεται συνεπές αν και μόνο αν για όλους τους κόμβους *n*, *n'*, όπου ο *n'* παράγεται από τον *n* με μια πράξη *a:* $h(n) \leq c(n,a,n') + h(n')$.
- **Θεώρημα:**
  - Κάθε συνεπές heuristic είναι και αποδεκτό.
    - Τα περισσότερα αποδεκτά heuristics είναι και συνεπή.

Ο A* είναι Βέλτιστος και Πλήρης (3/4)

- Αν το *h* είναι συνεπές τότε οι τιμές της *f* δε μειώνονται ποτέ καθώς ακολουθούμε οποιοδήποτε μονοπάτι.
- Επίσης η ακολουθία των κόμβων που εξερευνά ο A* είναι σε αύξουσα σειρά ανάλογα με την τιμή της *f.*

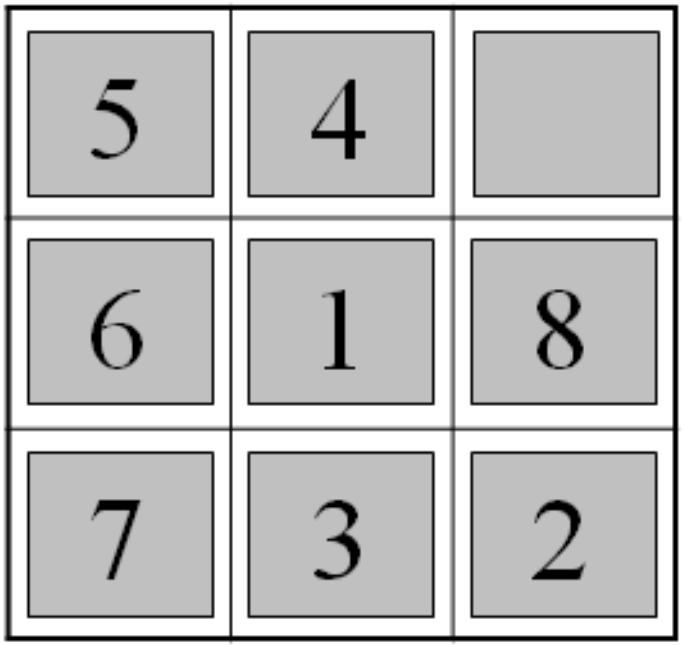


Ο A* είναι Βέλτιστος και Πλήρης (4/4)

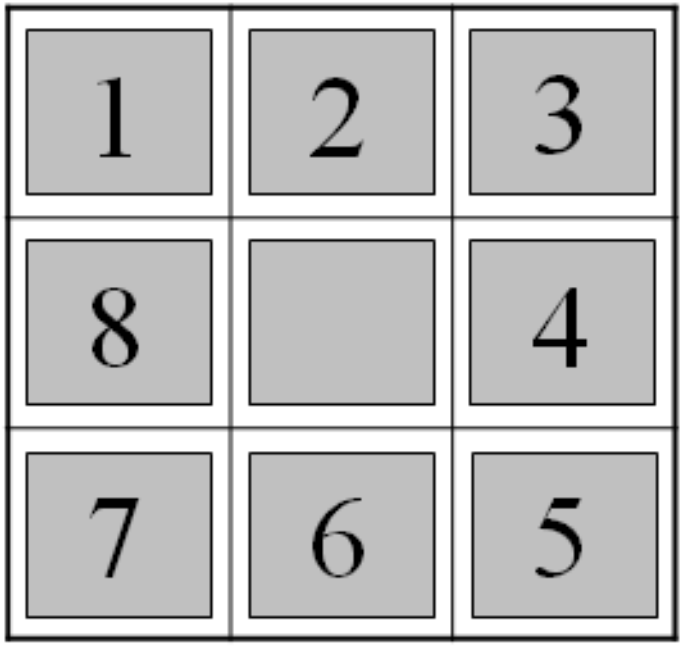
- **Ο A* είναι πλήρης:**
  - Καθώς προσθέτουμε ισοϋψείς καμπύλες αυξανόμενου *f* θα φτάσουμε κάποια στιγμή σε ισοϋψή καμπύλη όπου το *f* είναι ίσο με το κόστος ενός μονοπατιού ως τον στόχο.
- Στην πράξη ο A* λειτουργεί ως εξής:
  - Επεκτείνει όλους τους κόμβους με $f(n) < C^*$, όπου C* είναι το κόστος της βέλτιστης λύσης.
  - Μετά μπορεί να επισκεφτεί κάποιους από τους κόμβους για τους οποίους $f(n) = C^*$ ώσπου τελικά θα βρει τη λύση.
  - Ο A* ποτέ δεν επισκέπτεται κόμβους με κόστος $f(n) > C^*$.
- Δεν υπάρχει άλλος βέλτιστος αλγόριθμος που να εγγυάται ότι θα επισκεφτεί λιγότερους κόμβους από ότι ο A*.

Ευριστικές Συναρτήσεις

Ποιο heuristic είναι καλό για το πρόβλημα 8-puzzle;



Αρχική Κατάσταση



Κατάσταση Στόχου

Το 8-puzzle

- Επίσημη περιγραφή:
  - **Καταστάσεις.**
- Η κάθε κατάσταση περιγράφεται προσδιορίζοντας την τοποθεσία κάθε αριθμού καθώς και του κενού.
  - **Ενέργειες.**
- Το κενό μετακινείται Π (πάνω), Κ (κάτω), Δ (δεξιά), Α (αριστερά).
  - **Κατάσταση Στόχου.**
- Το κενό στη μέση, οι αριθμοί σε διάταξη σύμφωνα με τους δείκτες του ρολογιού.
  - **Κόστος Μονοπατιού.**
- Το μήκος του μονοπατιού, δηλ. πόσες μετακινήσεις γίνονται.
  - **Μέσο κόστος λύσης περίπου 22 βήματα.**
    - παράγοντας διακλάδωσης περίπου 3.
    - άρα 3^{22} καταστάσεις για τυφλή αναζήτηση!
    - με αποφυγή επαναλαμβανόμενων μειώνεται σε περίπου 170.000.
      - ✓ Για 15-puzzle ο αριθμός είναι $10^{13.}$

Ευριστικές Συναρτήσεις (1/3)

- **Heuristics για το 8-puzzle:**
  - ✓ h_1 = το πλήθος των αριθμών που είναι σε λάθος θέση.
  - ✓ h_2= το άθροισμα των οριζόντιων και κάθετων αποστάσεων όλων των αριθμών από τις σωστές τους θέσεις ***(απόσταση Manhattan)***.

- Και τα δύο heuristics είναι αποδεκτά.
  - Γιατί;

- **Ποιο είναι το καλύτερο;**
  - h_1 = 7.
  - h_2 = 18.

Ευριστικές Συναρτήσεις (2/3)

- Ένας τρόπος για να καθοριστεί η ποιότητα ενός heuristic είναι μέσω του **δραστικού παράγοντα διακλάδωσης** (effective branching factor) **b***.
- Αν το συνολικό πλήθος κόμβων που επισκέπτεται ο A* σε ένα συγκεκριμένο πρόβλημα είναι N, και το βάθος όπου βρίσκεται η λύση είναι d τότε b* είναι ο παράγοντας διακλάδωσης που πρέπει να έχει ένα ομοιόμορφο δέντρο με βάθος d για να περιέχει N+1 κόμβους. Δηλαδή:

$$N + 1 = 1 + b^* + (b^*)^2 + \ldots + (b^*)^d$$

- Συνήθως το b* παραμένει σταθερό για μεγάλο εύρος στιγμιότυπων ενός προβλήματος
  - **Η τιμή του b* για ένα καλό heuristic είναι γύρω στο 1.**

Σύγκριση A* και IDS

- Μέσοι όροι 100 8-puzzle προβλημάτων.

	Search Cost			Effective Branching Factor		
d	IDS	A*(h_1)	A*(h_2)	IDS	A*(h_1)	A*(h_2)
2	10	6	6	2.45	1.79	1.79
4	112	13	12	2.87	1.48	1.45
6	680	20	18	2.73	1.34	1.30
8	6384	39	25	2.80	1.33	1.24
10	47127	93	39	2.79	1.38	1.22
12	364404	227	73	2.78	1.42	1.24
14	3473941	539	113	2.83	1.44	1.23
16	–	1301	211	–	1.45	1.25
18	–	3056	363	–	1.46	1.26
20	–	7276	676	–	1.47	1.27
22	–	18094	1219	–	1.48	1.28
24	–	39135	1641	–	1.48	1.26

Ευριστικές Συναρτήσεις (3/3)

- Αν $h_2(n) \geq h_1(n)$ για όλους τους κόμβους n τότε το h_2 είναι **πιο ισχυρό** (ή **πιο ενημερωμένο**) από το $h_{1:}$
  - Για παράδειγμα στο 8-puzzle το h_2 είναι πιο ισχυρό από το $h_{1.}$
- **Θεώρημα:**
  - Αν ένα heuristic h_2 είναι πιο ισχυρό από ένα heuristic h_1 τότε ο A* με χρήση του h_2 θα επισκεφτεί λιγότερους κόμβους από τον A* με χρήση του $h_{1.}$
- **Συμβουλή:**
  - Είναι πάντα καλύτερο να διαλέγουμε την ευριστική συνάρτηση που δίνει μεγαλύτερες τιμές (δηλ. πιο κοντά στο πραγματικό κόστος) χωρίς να το υπερεκτιμάει.

Ευριστικές Συναρτήσεις: Πως τις βρίσκουμε; (1/3)

- Είναι πιθανό να βρούμε ευριστικές συναρτήσεις αν σκεφτούμε **χαλαρωμένες** *(relaxed)* εκδόσεις του προβλήματος που θέλουμε να λύσουμε.
- Το κόστος της βέλτιστης λύσης στο χαλαρωμένο πρόβλημα είναι ένα αποδεκτό heuristic στο αρχικό πρόβλημα.
- Χαλαρωμένα εκδοχές προβλημάτων μπορούν μερικές φορές να παραχθούν αυτόματα.
  - Οπότε και τα heuristics μπορούν να βρεθούν αυτόματα.
  - Αλλιώς πρέπει να σκεφτούμε προσεκτικά το πρόβλημα και να χρησιμοποιήσουμε το μυαλό και τη φαντασία μας!

Ευριστικές Συναρτήσεις: Πως τις βρίσκουμε; (2/3)

- Στο παράδειγμα του 8-puzzle τα δύο heuristics μπορούν να επινοηθούν με βάση χαλαρωμένες εκδόσεις του προβλήματος.
- h_1=το πλήθος των αριθμών που είναι σε λάθος θέση:
  - στο χαλαρωμένο πρόβλημα όπου επιτρέπεται οποιαδήποτε κίνηση δίνει τον ακριβή αριθμό κινήσεων για τη λύση.
- h_2=το άθροισμα των οριζόντιων και κάθετων αποστάσεων όλων των αριθμών από τις σωστές τους θέσεις:
  - στο χαλαρωμένο πρόβλημα όπου επιτρέπεται η κίνηση προς οποιοδήποτε διπλανό τετράγωνο, ακόμα κι αν είναι κατειλημμένο, δίνει τον ακριβή αριθμό κινήσεων για τη λύση.

Ευριστικές Συναρτήσεις: Πως τις βρίσκουμε; (3/3)

- Αν έχουμε ένα πλήθος από αποδεκτά heuristics $h_1, h_2, \ldots, h_m$ τέτοια ώστε κανένα να μην είναι πιο ισχυρό από κανένα άλλο, μπορούμε να διαλέξουμε για heuristic το h=**max** ($h_1, h_2, \ldots, h_m$).
- Κατά τη διαδικασία επιλογής ενός heuristic δεν πρέπει να ξεχνάμε το κόστος εφαρμογής του heuristic σε όλους τους κόμβους του δέντρου:
  - Αν είναι πολύ μεγάλο τότε ένα λιγότερο καλό αλλά πιο φτηνό heuristic ίσως είναι καλύτερη επιλογή.

Επεκτάσεις του A*

- Το βασικό πρόβλημα του A* είναι η υπερβολική χρήση μνήμης σε μεγάλα και δύσκολα προβλήματα.
- Έχουν προταθεί διάφοροι αλγόριθμοι που αντιμετωπίζουν αυτό το πρόβλημα:
  - **A* Επαναληπτικής Εκβάθυνσης** *(Iterative Deepening A* - IDA*).*
    - Παρόμοιο με το IDS μόνο που εδώ το όριο σε κάθε επανάληψη βασίζεται στο κόστος.
  - **Απλοποιημένος A* με Όριο Μνήμης** *(Simplified Memory-Bounded A* - SMA*).*
    - Χρησιμοποιεί όλη τη διαθέσιμη μνήμη.

Τοπική Αναζήτηση *(Local Search)*

- **Hill-climbing** *(αναρρίχηση λόφων).*
- **Simulated Annealing** *(προσομοιωμένη εξέλιξη).*
- **Genetic Algorithms** *(γενετικοί αλγόριθμοι).*

Αλγόριθμοι Τοπικής Αναζήτησης (1/3)

- Οι αλγόριθμοι αναζήτησης που είδαμε εξερευνούν χώρους αναζήτησης συστηματικά:
  - Διατηρούν μια ή περισσότερες διαδρομές στη μνήμη κι όταν βρεθεί μια κατάσταση στόχου, η ***διαδρομή*** προς αυτή την κατάσταση είναι ***λύση.***
- Σε πολλά προβλήματα βελτιστοποίησης η ***διαδρομή*** ως την κατάσταση στόχου δεν έχει σημασία:
  - Το μόνο που μας ενδιαφέρει είναι να βρούμε μια κατάσταση στόχου και να προσδιορίσουμε δηλ. τη μορφή της.
- **Παραδείγματα:**
  - Εύρεση κάποιας διάταξης που ικανοποιεί ορισμένους περιορισμούς (π.χ. 8-queens, κατασκευή προγράμματος μαθημάτων).
  - χρονοπρογραμματισμός εργασιών σε εργοστάσια.

Αλγόριθμοι Τοπικής Αναζήτησης (2/3)

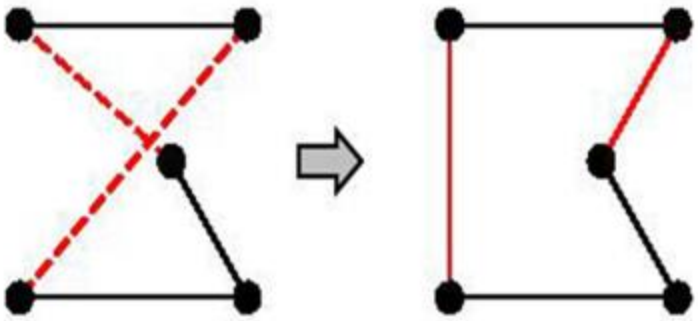
- Σε τέτοιες περιπτώσεις *(και όχι μόνο)* μπορούμε να χρησιμοποιήσουμε αλγόριθμους που δεν ενδιαφέρονται για διαδρομές και βασίζονται στην **επαναληπτική βελτίωση** *(iterative improvement):*
  - ξεκίνα με μια αρχική κατάσταση και προσπάθησε να τη βελτιώσεις.
- Οι αλγόριθμοι που θα περιγράψουμε ονομάζονται αλγόριθμοι **τοπικής αναζήτησης (local search):**
  - λειτουργούν χρησιμοποιώντας **μια μόνο** *(τρέχουσα)* **κατάσταση** *(αντί για πολλαπλά μονοπάτια)* και γενικά μετακινούνται μόνο σε γειτονικές της καταστάσεις,
  - οι διαδρομές που ακολουθούνται από την αναζήτηση δεν διατηρούνται στην μνήμη.

Αλγόριθμοι Τοπικής Αναζήτησης (3/3)

- Σε κάθε βήμα ενός αλγόριθμου τοπικής αναζήτησης έχουμε μια **πλήρη αλλά ατελή "λύση"** του προβλήματος:
  - προηγούμενοι αλγόριθμοι που εξετάσαμε (π.χ. A*) έχουν μια **μερική λύση** και προσπαθούν να την επεκτείνουν σε πλήρη.
- Καλές ιδιότητες αλγορίθμων τοπικής αναζήτησης:
  - ***Σταθερός χώρος.***
  - ***Ταχύτητα*** (κατάλληλοι για on-line και off-line αναζήτηση).
- Οι αλγόριθμοι αυτοί είναι κατάλληλοι και για την επίλυση **προβλημάτων βελτιστοποίησης.**
  - σε αυτά τα προβλήματα στόχος είναι να βρεθεί η καλύτερη κατάσταση σύμφωνα με μια **αντικειμενική συνάρτηση.**

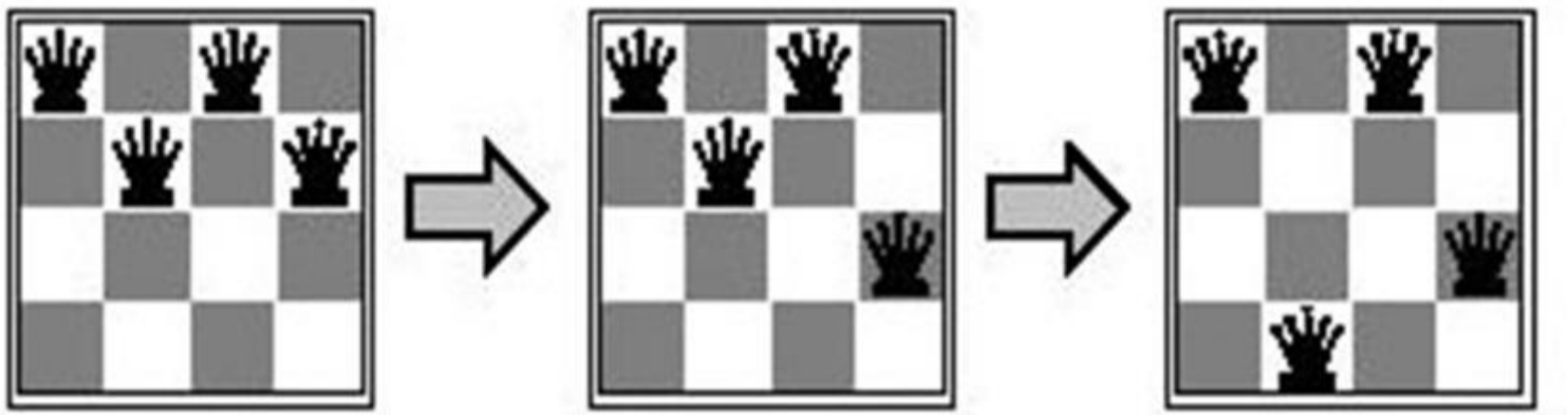
Παράδειγμα – Πλανόδιος Πωλητής (TSP)

- **Πρόβλημα:**
  - Υπάρχει ένα σύνολο πόλεων που συνδέονται με δρόμους. Ο πλανόδιος πωλητής ξεκινάει από μια πόλη και πρέπει να τις επισκεφτεί όλες, μια φορά την καθεμία και να επιστρέψει στην αρχική έχοντας διανύσει τη μικρότερη δυνατή απόσταση.
- **Ιδέα:**
  - Ξεκίνα με ένα τυχαίο πλήρες μονοπάτι. Μετά κάνε τοπικές αλλαγές διαδρομών μειώνοντας τη συνολική απόσταση του μονοπατιού.



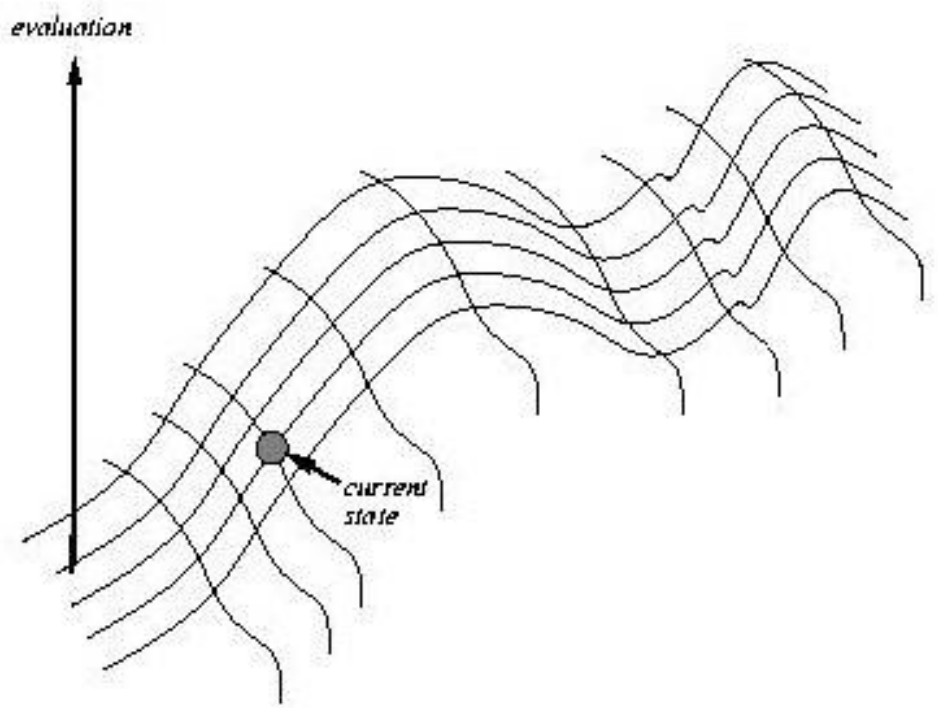
Παράδειγμα – 4-Queens

- **Ιδέα:** Ξεκίνα με 4 βασίλισσες τυχαία τοποθετημένες στη σκακιέρα. Μετά επαναληπτικά μετακίνησε βασίλισσες ώσπου να μη μπορεί καμία να επιτεθεί σε άλλη.



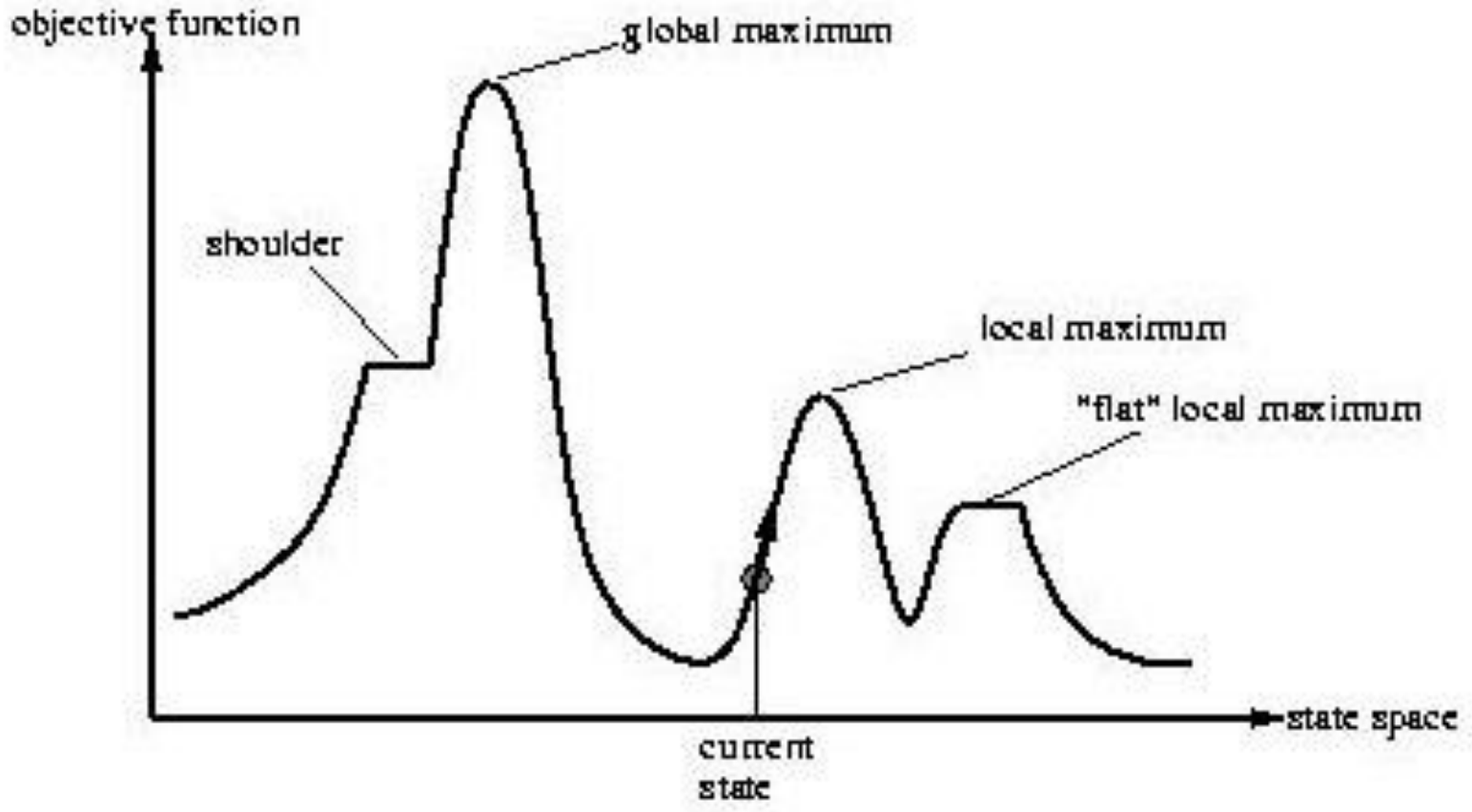
Αλγόριθμοι Επαναληπτικής Βελτίωσης

- **Ιδέα:** Ξεκίνα με μια πλήρη “λύση” και κάνε μετατροπές μέχρι να βρεις μια πραγματική λύση.



Τοπίο του χώρου καταστάσεων

Το Τοπίο του Χώρου Καταστάσεων



Αναρρίχηση Λόφων *(Hill-Climbing Search)*

```
function Hill_Climbing (problem, Eval-fn)
returns a state that is a local maximum
inputs: problem, a search problem
    Eval-fn, an evaluation function
local variables: current, next, nodes
                 neighbors, a set of nodes
    current ← MakeNode(InitialState[problem])
    loop do
        neighbors ← SuccessorStates(current)
        if neighbors = ∅ return current
        Eval-fn(neighbors)
        next ← a highest-valued node of neighbors
        if the value of next is less or equal to the value of current
                return current
    end
```

Παράδειγμα -8-Queens (1/3)

Περιγραφή Προβλήματος:

- **Καταστάσεις:**
  - Κάθε τοποθέτηση 8 βασιλισσών στη σκακιέρα όπου η καθεμία είναι σε διαφορετική στήλη.
- **Ενέργειες:**
  - Μετακίνησε μια βασίλισσα μέσα στη στήλη της.
- **Τεστ Στόχου:**
  - 8 βασίλισσες στη σκακιέρα και καμία δε μπορεί να επιτεθεί σε άλλη.
- **Συνάρτηση Αξιολόγησης *(κόστος):***
  - Το πλήθος των ζευγαριών βασιλισσών όπου επιτίθεται η μια στην άλλη.
- **Πρόβλημα ελαχιστοποίησης.**

Παράδειγμα -8-Queens (2/3)

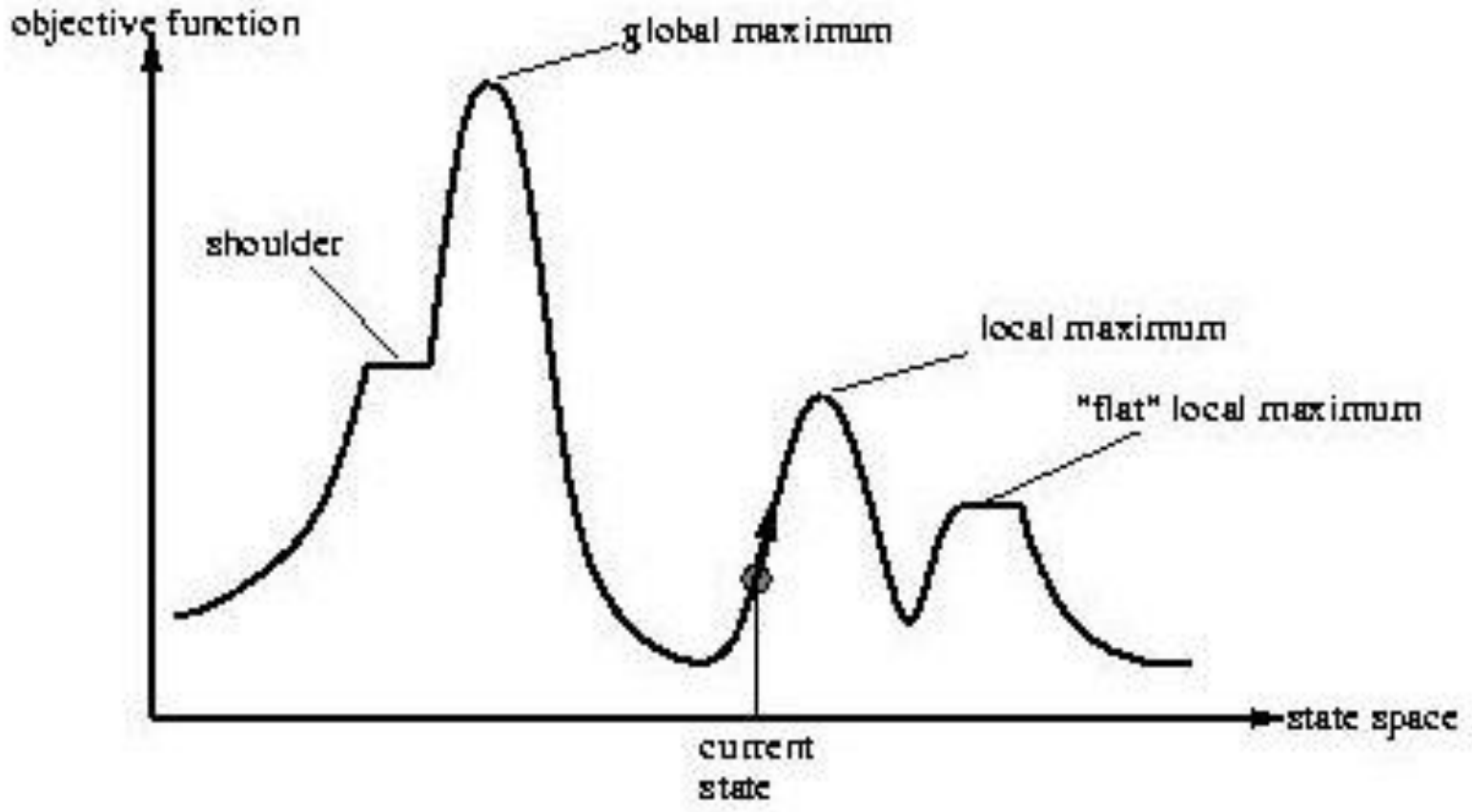
- Το κόστος της **τρέχουσας κατάστασης** είναι 17:
  - Οι τιμές σε κάθε τετράγωνο δείχνουν το κόστος των **γειτονικών καταστάσεων**:

  Δηλ. το κόστος αν η αντίστοιχη βασίλισσα μετακινηθεί σε αυτό το τετράγωνο.

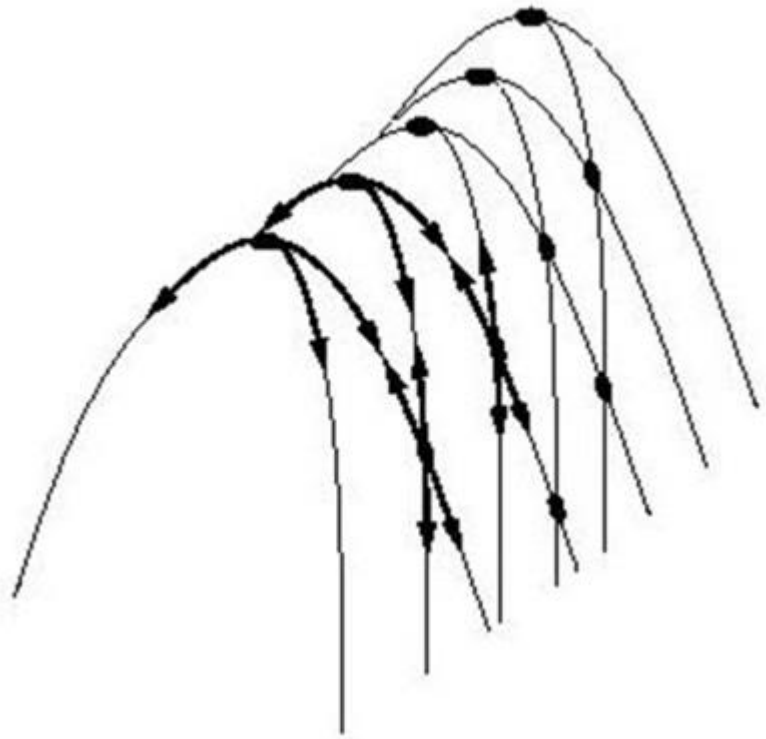
Hill-Climbing – Προβλήματα (1/2)

- **Τοπικά Βέλτιστα** *(local optima):*
  - Η αναζήτηση μπορεί να κολλήσει σε μια κατάσταση της οποίας όλες οι γειτονικές έχουν χειρότερο κόστος αλλά δεν είναι η βέλτιστη.
- **Οροπέδια** *(plateaus):*
  - Αν όλα τα γειτονικά σημεία έχουν το ίδιο κόστος η επιλογή γίνεται τυχαία.
- **Κορυφογραμμές** *(ridges):*
  - Οι κορυφογραμμές είναι ψηλά και εντοπίζονται εύκολα, όμως από εκεί και πέρα η αναζήτηση είναι πολύ αργή γιατί αποτελούνται από μια ακολουθία τοπικών βέλτιστων.
- **Πως αντιμετωπίζονται αυτά τα προβλήματα;**
  - εξαρτάται από το συγκεκριμένο πρόβλημα.

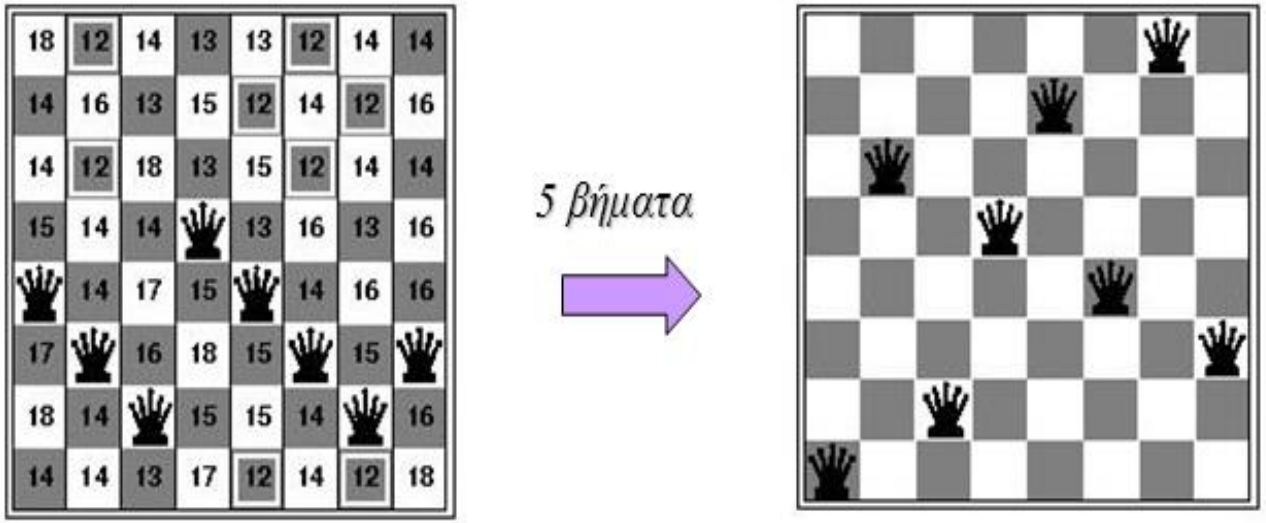
Hill-Climbing – Προβλήματα (2/2)



Παράδειγμα – Κορυφογραμμές



Παράδειγμα -8-Queens (3/3)



- Όμως το κόστος της κατάστασης είναι 1. Όλες οι γειτονικές καταστάσεις έχουν κόστος > 1:
  - άρα έχουμε τοπικό βέλτιστο.

Hill-Climbing – Απόδοση

- Ας υποθέσουμε ότι ξεκινάμε με μια τυχαία τοποθέτηση 8 βασιλισσών στη σκακιέρα.
- Η απόδοση του hill-climbing έχει ως εξής:
  - Λύνει *(κατά μέσο όρο)* το 14% των προβλημάτων μέσα σε 4 βήματα.
  - Κολλάει σε τοπικά βέλτιστα στο 86% των προβλημάτων *(κατά μέσο όρο)* μέσα σε 3 βήματα.
- Το μέγεθος του χώρου καταστάσεων είναι $8^8 \approx 17$ εκατομμύρια καταστάσεις.

Πλάγιες Κινήσεις *(Sideways Moves)*

- Όταν το hill-climbing φτάνει σε οροπέδιο και δεν υπάρχουν γειτονικές κινήσεις προς τα επάνω, κολλάει.
- Εναλλακτικά θα μπορούσε να κάνει μια **πλάγια κίνηση:**
  - Δηλ. κίνηση προς κατάσταση που έχει το **ίδιο κόστος** με την τωρινή.
  - Χρειάζεται προσοχή γιατί αν το οροπέδιο είναι τοπικό βέλτιστο, ο αλγόριθμος θα πέσει σε ατέρμονο loop.
  - Μια λύση για αυτό το πρόβλημα είναι να βάλουμε ένα όριο στο πλήθος των πλάγιων κινήσεων.
- Αν επιτρέψουμε πλάγιες κινήσεις στο 8-queens πρόβλημα και βάλουμε όριο 100 κινήσεων, μπορούμε να λύσουμε το 94% των προβλημάτων σε 21 βήματα κατά μέσο όρο.

Παραλλαγές του hill-climbing

- **First-choice hill-climbing:**
  - Παράγει γειτονικές καταστάσεις τυχαία μέχρι να παραχθεί κάποια με καλύτερο κόστος από την τωρινή.
    - Καλή στρατηγική όταν υπάρχει μεγάλο πλήθος γειτόνων (π.χ. χιλιάδες).
- **Stochastic hill-climbing:**
  - Διαλέγει τυχαία από τις διαθέσιμες κινήσεις προς τα πάνω:
    - Η πιθανότητα επιλογής εξαρτάται από την ποιότητα της κάθε κίνησης.
    - Γενικά είναι πιο αργή από το κλασσικό hill-climbing αλλά σε πολλές περιπτώσεις βρίσκει καλύτερη λύση.
- Πως αποφεύγουμε τα τοπικά βέλτιστα;
  - **Random restarts** *(τυχαίες επανεκκινήσεις).*
  - **Simulated annealing** *(προσομοιωμένη ανόπτυση).*

Hill-climbing με random restarts

- **Ιδέα:** Αν δεν πετύχεις με την πρώτη, προσπάθησε ξανά και ξανά!
- Το **hill-climbing με τυχαίες επανεκκινήσεις** πραγματοποιεί μια σειρά από hill-climbing αναζητήσεις από τυχαία παραγόμενες αρχικές καταστάσεις και σταματάει όταν βρεθεί ο στόχος:
  - Με πιθανότητα 1 θα παραχθεί κάποια στιγμή ο στόχος ως αρχική κατάσταση.
  - Αν κάθε αναζήτηση έχει πιθανότητα επιτυχίας p, τότε ο αναμενόμενος αριθμός επανεκκινήσεων που απαιτούνται για να βρεθεί λύση είναι $1/p$.
- Στο πρόβλημα 8-queens $p \approx 0.14$, οπότε χρειαζόμαστε περίπου 7 επανεκκινήσεις (6 αποτυχίες και 1 επιτυχία).

Προσομοιωμένη Ανόπτηση *(Simulated Annealing)*

- Ένας hill-climbing αλγόριθμος που δεν κάνει ποτέ κινήσεις προς τα κάτω *(προς καταστάσεις με χειρότερο κόστος)* δεν είναι πλήρης.
- Το τυχαίο βάδισμα ***(random walk)*** όπου η επόμενη κατάσταση επιλέγεται τυχαία από το σύνολο των γειτόνων είναι πλήρης *(γιατί;)* αλλά καθόλου πρακτικό.
- Πως μπορούμε να τα συνδυάσουμε;
- Υπάρχει μια ανταλλαγή ***(trade-off)*** μεταξύ της εξερεύνησης του χώρου αναζήτησης και της εκμετάλλευσης της τρέχουσας ατελούς «λύσης»:
  - ή θα προσπαθήσουμε να εκμεταλλευτούμε όσο γίνεται τη «λύση» *(μη πλήρης αλγόριθμος)* ή θα κάνουμε πολύ εξερεύνηση.

Simulated Annealing (1/4)

Physical analogue: Annealing of metals is the process used to temper or harden metals by heating them to a high temperature and then gradually cooling them, thus allowing the material to coalesce into a low-energy crystalline state.

The discovery of the simulated annealing algorithm is an instance of the use of ideas from **statistical mechanics** (an area of condensed matter physics) to solving **large and complex optimization problems**.

Statistical mechanics concentrates on analyzing aggregate properties of large numbers of atoms to be found in samples of liquid or solid matter. See the paper on simulated annealing by Kirkpatrick et. al. in Science, Volume 220, Number 4598, May 1983.

Simulated Annealing (2/4)

- Το **simulated annealing** επιλύει τη σύγκρουση μεταξύ εξερεύνησης και εκμετάλλευσης της τρέχουσας κατάστασης ως εξής:
  - Σε κάθε επανάληψη επιλέγεται μια τυχαία κίνηση.
  - Αν βελτιώνει την κατάσταση τότε η κίνηση γίνεται δεκτή.
  - Αλλιώς η κίνηση γίνεται δεκτή με κάποια πιθανότητα < 1.
- Η πιθανότητα αποδοχής μειώνεται εκθετικά σε σχέση με το πόσο κακή είναι η κίνηση:
  - Επίσης μειώνεται σε σχέση με μια **παράμετρο θερμοκρασίας *T.***
- Το simulated annealing ξεκινάει με υψηλή τιμή για το *T* η οποία σταδιακά μειώνεται:
  - Για μεγάλες τιμές του *T* το simulated annealing μοιάζει με random walk.
  - Για μικρές τιμές του *T* το simulated annealing μοιάζει hill-climbing.

Simulated Annealing (3/4)

function Simulated_Annealing (problem, schedule, Eval-fn)
returns a solution state
inputs: problem, a search problem
schedule, a mapping from time to “temperature”
local variables: current, next: nodes
T, a “temperature” controlling the probability of downward moves
current ← MakeNode(InitialState[problem])
for t ←1 to ∞ **do**
T ← schedule[t]
if T=0 **then return** current
next ← a randomly selected successor of current
ΔE ← Eval-fn(next) - Eval-fn(current)
if ΔE > 0 **then** current ← next
else current ← next only with probability $e^{-\Delta E/T}$
return current
end

Simulated Annealing – Παράδειγμα

- Αν υποθέσουμε ότι το επόμενο σημείο στο χώρο αναζήτησης έχει μικρότερη τιμή *(σύμφωνα με τη συνάρτηση εκτίμησης)* από το τωρινό κατά 13 μονάδες
  - Δηλ. ΔΕ = -13

T	$e^{\Delta E/T}$
1	0.000002
5	0.0743
10	0.2725
20	0.52
50	0.77
10^{10}	0.9999…

Simulated Annealing (4/4)

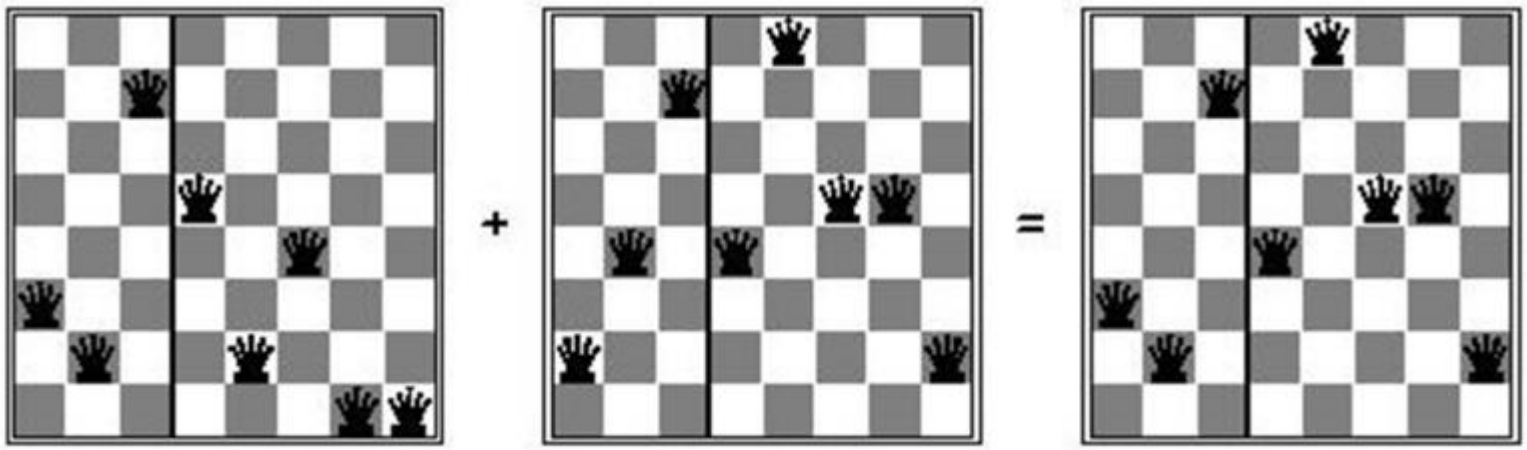
- Το simulated annealing βρίσκει το συνολικό βέλτιστο με πιθανότητα κοντά στο 1 αν το *T* μειώνεται αρκετά αργά.
- Το ακριβές όριο για το *t* και το πρόγραμμα μείωσης του *T* *(schedule)* εξαρτάται πολύ από το συγκεκριμένο πρόβλημα:
  - Οπότε πρέπει να κάνουμε εκτεταμένα πειράματα σε κάθε καινούργιο πρόβλημα για να βρούμε αν το simulated annealing είναι αποδοτικό.
- Γενικά το simulated annealing είναι πολύ δημοφιλής αλγόριθμος και έχει χρησιμοποιηθεί με επιτυχία σε πολλά προβλήματα:
  - job-shop scheduling, VLSI layout.

Γενετικοί Αλγόριθμοι (1/5)

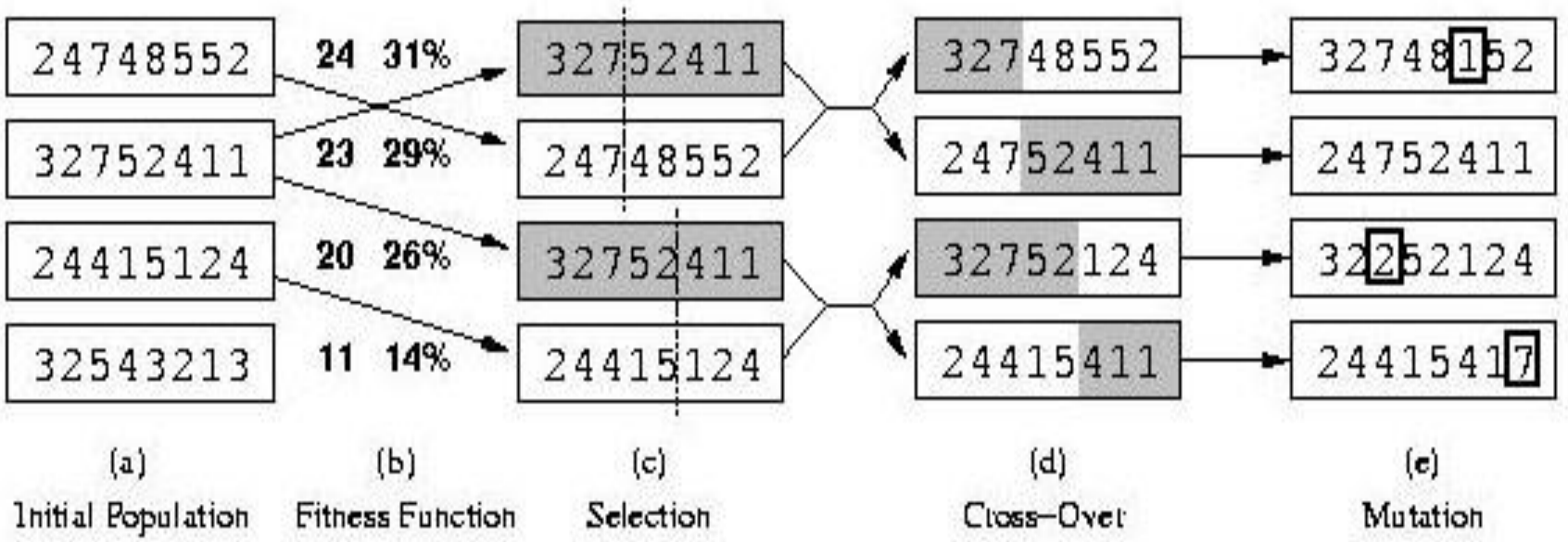
- Οι **γενετικοί αλγόριθμοι** *(genetic algorithms)* είναι παραλλαγές στοχαστικής αναζήτησης όπου η επόμενη κατάσταση παράγεται συνδυάζοντας δύο γονικές καταστάσεις *(αναπαραγωγή).*
- Βασικές Έννοιες:
  - Τα **άτομα** *(individuals)* αντιπροσωπεύουν καταστάσεις. Αναπαριστώνται με συμβολοσειρές ενός αλφαβήτου *(συνήθως {0,1})*:
    - ✓ χρωμοσώματα, γονίδια.
  - Ένας **πληθυσμός** *(population)* είναι ένα σύνολο ατόμων.
  - Υπάρχει μια **συνάρτηση καταλληλότητας** *(fitness function)* των ατόμων.

Γενετικοί Αλγόριθμοι (2/5)

- Λειτουργίες:
  - **Αναπαραγωγή** *(reproduction):* Ένα νέο άτομο γεννιέται συνδυάζοντας δύο γονείς.
  - **Μετάλλαξη** *(mutation):* Ένα άτομο μεταβάλλεται ελαφρώς.



Γενετικοί Αλγόριθμοι (3/5)



Γενετικοί Αλγόριθμοι (4/5)

function Genetic_Algorithm (population, Fitness-fn)
returns an individual
inputs: population, a set of individuals
Fitness-fn, a function that measures the fitness of an individual
repeat new_population ← 0
for 1 **to** Size(population) **do**
x ← Random_selection(population,Fitness-fn)
y ← Random_selection(population,Fitness-fn)
child ← Reproduce(x,y)
with small random probability child ← Mutate(child)
add child to new_population
population ← new_population
until some individual is fit enough, or enough time has elapsed
return the best individual in population according to Fitness-fn

Γενετικοί Αλγόριθμοι (5/5)

- Η δύναμη των γενετικών αλγορίθμων προκύπτει από τη διασταύρωση γονέων με υψηλή τιμή καταλληλότητας.
- Το βασικό ερώτημα έχει να κάνει με την κωδικοποίηση προβλημάτων ώστε να μπορούν να λυθούν αποδοτικά με γενετικούς αλγόριθμους:
  - Επιλογή γονιδίων.
  - Επιλογή συνάρτησης καταλληλόλητας.
  - κτλ.
- **Συμπέρασμα:** Χρησιμοποιείστε τους με επιφύλαξη!

Αλγόριθμοι Αναζήτησης

- Αλγόριθμοι Τυφλής Αναζήτησης:
  - Depth-First Search (DFS).
  - Breadth-First Search (BFS).
  - Uniform Cost Search (UCS).
  - Iterative Deepening Search (IDS).
- Αλγόριθμοι Ευριστικής Αναζήτησης:
  - Best-First Search.
  - A*.
- Αλγόριθμοι Τοπικής Αναζήτησης:
  - Hill Climbing.
  - Simulated Annealing.
  - Genetic Algorithms.

Τέλος Ενότητας

Σημείωμα Αναφοράς

- Copyright Πανεπιστήμιο Δυτικής Μακεδονίας, Τμήμα Μηχανικών Πληροφορικής και Τηλεπικοινωνιών, Στεργίου Κωνσταντίνος. «**Τεχνητή Νοημοσύνη**». Έκδοση: 1.0. Κοζάνη 2015. Διαθέσιμο από τη δικτυακή διεύθυνση:

  https: //eclass.uowm.gr/courses/ICTE103/

Σημείωμα Αδειοδότησης

Το παρόν υλικό διατίθεται με τους όρους της άδειας χρήσης Creative Commons Αναφορά, Όχι Παράγωγα Έργα Μη Εμπορική Χρήση 4.0 [1] ή μεταγενέστερη, Διεθνής Έκδοση. Εξαιρούνται τα αυτοτελή έργα τρίτων π.χ. φωτογραφίες, διαγράμματα κ.λ.π., τα οποία εμπεριέχονται σε αυτό και τα οποία αναφέρονται μαζί με τους όρους χρήσης τους στο «Σημείωμα Χρήσης Έργων Τρίτων».

[1] h t t p ://creativecommons.org/licenses/by-nc-nd/4.0/

Ως Μη Εμπορική ορίζεται η χρήση:

- που δεν περιλαμβάνει άμεσο ή έμμεσο οικονομικό όφελος από την χρήση του έργου για το διανομέα του έργου και αδειοδόχο
- που δεν περιλαμβάνει οικονομική συναλλαγή ως προϋπόθεση για τη χρήση ή πρόσβαση στο έργο
- που δεν προσπορίζει στο διανομέα του έργου και αδειοδόχο έμμεσο οικονομικό

Διατήρηση Σημειωμάτων

Οποιαδήποτε αναπαραγωγή ή διασκευή του υλικού θα πρέπει να συμπεριλαμβάνει:

- το Σημείωμα Αναφοράς
- το Σημείωμα Αδειοδότησης
- τη δήλωση Διατήρησης Σημειωμάτων
- το Σημείωμα Χρήσης Έργων Τρίτων (εφόσον υπάρχει)

μαζί με τους συνοδευόμενους υπερσυνδέσμους.

Σημείωμα Χρήσης Έργων Τρίτων

Το Έργο αυτό κάνει χρήση των ακόλουθων έργων:

Εικόνες/Σχήματα/Διαγράμματα/Φωτογραφίες

- Τεχνητή Νοημοσύνη, Μια σύγχρονη προσέγγιση, S. Russel, P. Norvig, Εκδόσεις Κλειδάριθμος