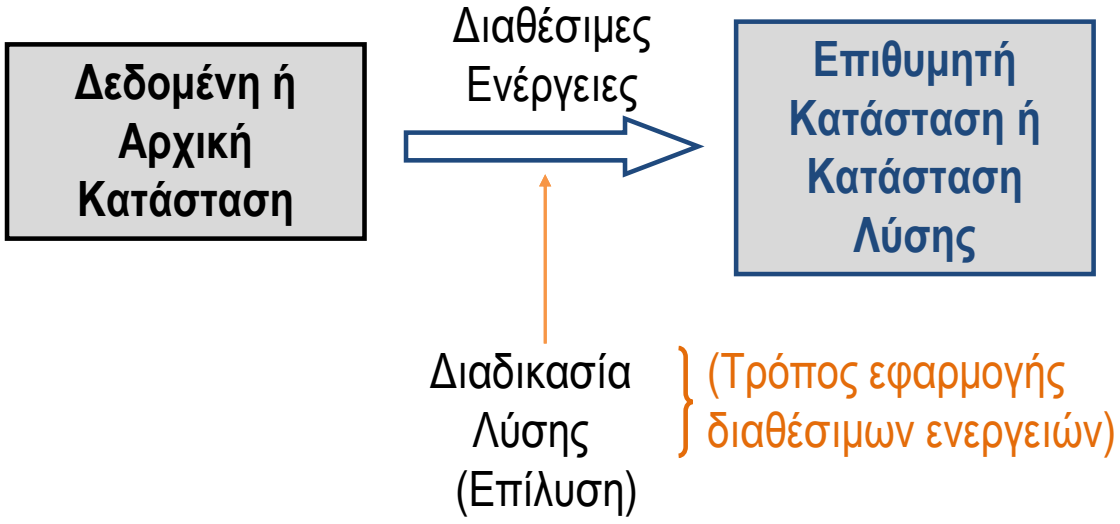
Τεχνητή Νοημοσύνη

Επίλυση Προβλημάτων & Αναζήτηση

Δρ. Δημήτριος Κουτσομητρόπουλος

Πρόβλημα-
Επίλυση Προβλήματος



Προβλήματα αναζήτησης στον πραγματικό κόσμο

- Εύρεση Διαδρομής (route finding),
- Καθοδήγηση Robot (robot navigation),
- Σχεδιασμός Πρωτεϊνών (protein design),
- VLSI σχεδιασμός,
- Ανάθεση εργασιών/πόρων σε εργαζόμενους (resource allocation),
- Βελτιστοποίηση Επερωτήσεων σε ΣΔΒΔ (query optimization),
- Αναζήτηση στο διαδίκτυο (internet search engines),
- Κατασκευή χρονοδιαγράμματος (timetabling).

Κύριες κατηγορίες προβλημάτων

- Προβλήματα που είναι πλήρως γνωστές οι τελικές καταστάσεις.
  - **Προβλήματα σχεδιασμού ενεργειών και προβλήματα πλοήγησης.**
- Προβλήματα που είναι πλήρως γνωστές οι τελικές καταστάσεις αλλά ζητείται η βέλτιστη.
  - **Προβλήματα βελτιστοποίησης.**
- Προβλήματα που είναι γνωστές κάποιες ιδιότητες μόνο των τελικών καταστάσεων και επιδιώκεται η εύρεση ενός πλήρους στιγμιότυπου τελικής κατάστασης.
  - **Προβλήματα χρονοπρογραμματισμού γνωστά ως προβλήματα ικανοποίησης περιορισμών.**

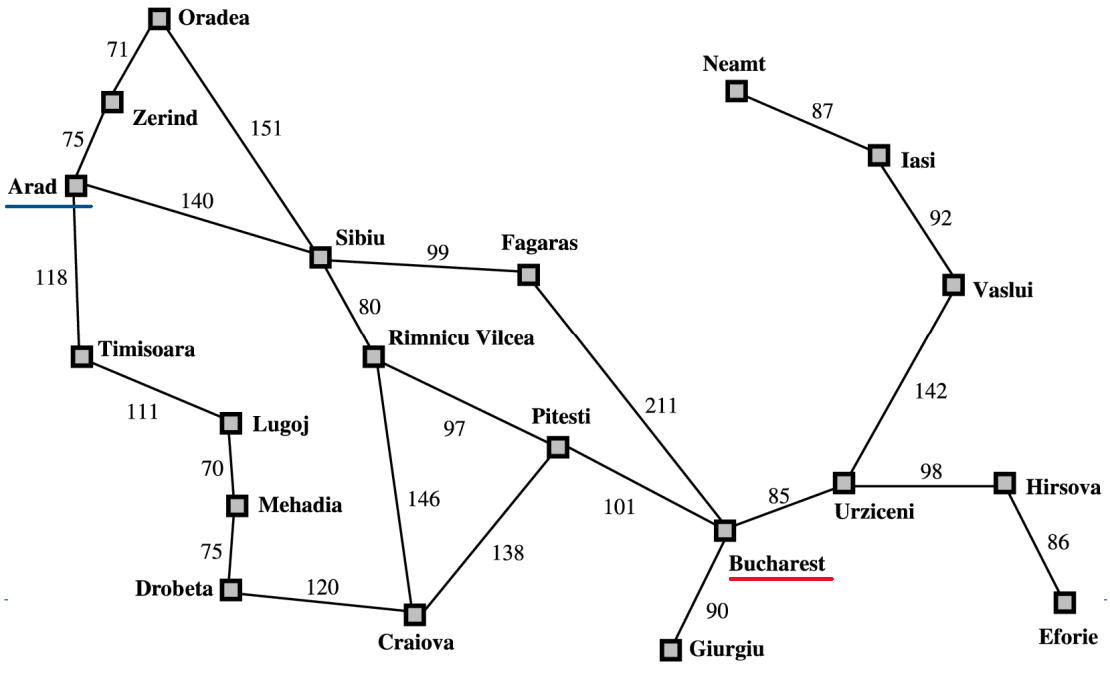
Κόσμος ενός Προβλήματος

Κόσμος Προβλήματος (problem's world): ένα υποσύνολο του πραγματικού κόσμου που περιέχει μόνο:

- τα **αντικείμενα** (στοιχεία) που έχουν άμεση σχέση με το πρόβλημα,
- τις **ιδιότητές** τους και
- τις **σχέσεις** που τα συνδέουν

Πράκτορες επίλυσης προβλημάτων

- Διατύπωση στόχου: υποσύνολο καταστάσεων του κόσμου
- Διατύπωση προβλήματος
  - Επιλογή επιπέδου λεπτομέρειας (*αφαίρεση*)



Κατάσταση προβλήματος

- **Κατάσταση (state):** Στιγμιότυπο μιας συγκεκριμένης στιγμής της εξέλιξης του κόσμου του προβλήματος. Μια κατάσταση πρέπει να περιγράφει **ολόκληρο** τον κόσμο ενός προβλήματος **μια συγκεκριμένη στιγμή**.
- **Χώρος καταστάσεων (state space)** είναι το σύνολο των ΕΓΚΥΡΩΝ καταστάσεων που μπορεί να βρεθεί ένα πρόβλημα κατά την εξέλιξη του κόσμου του.
- Υπάρχουν κλειστοί χώροι και ανοιχτοί χώροι καταστάσεων

Μοντελοποίηση Προβλήματος

- Συνιστώσες προβλήματος:
  - **Αρχική κατάσταση** (initial state)
    - π.χ. *Εντός*(*Arad*)
  - **Ενέργειες** (actions) ή **τελεστές μετάβασης**, συνάρτηση διαδόχων Successor-Fn(*x*) **(μοντέλο μετάβασης)**
    - π.χ. Successor-Fn(*Arad*) = { ⟨*Μετάβαση*(*Sibiu*), *Εντός*(*Sibiu*)⟩, ⟨*Μετάβαση*(*Timisoara*), *Εντός*(*Timisoara*)⟩, ⟨*Μετάβαση*(*Zerind*), *Εντός*(*Zerind*)⟩ }
    - **Χώρος καταστάσεων, αναπαράσταση με γράφημα, διαδρομές**
  - **Έλεγχος στόχου** (goal test)
    - Ρητή απαρίθμηση καταστάσεων ή περιγραφή ιδιοτήτων
  - **Κόστος διαδρομής** (path cost)
    - Κόστος βήματος, *c*(*x*,*a*,*y*)
- **Λύση**: Διαδρομή από αρχική κατάσταση σε κατάσταση στόχου
  - Βέλτιστη λύση

Τυποποιημένος Ορισμός προβλήματος

P = (I, G, T, S)

Ένα πρόβλημα ορίζεται από:

- την αρχική κατάσταση **(I)**
- το σύνολο των τελικών καταστάσεων **(G)**
- το σύνολο των τελεστών μετάβασης **(T)** και από
- το χώρο των καταστάσεων **(S)**.

Τελεστής μετάβασης είναι μια από τις ενέργειες που μπορούν να γίνουν κατά τη διάρκεια της επίλυσης ενός προβλήματος έτσι ώστε η κατάσταση K_n του προβλήματος να εξελιχθεί σε μια νέα κατάσταση K_{n+1}

Χώρος αναζήτησης

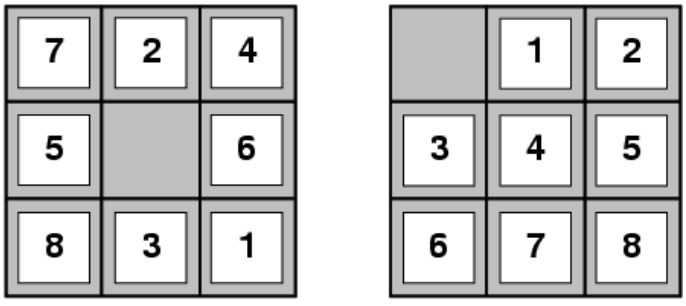
- Δοθέντος ενός προβλήματος **P = (I, G, T, S)** ο **χώρος αναζήτησης (search space**) είναι το σύνολο όλων των καταστάσεων που είναι προσβάσιμες από την αρχική κατάσταση
- Μια κατάσταση s χαρακτηρίζεται ως **προσβάσιμη** αν υπάρχει μια ακολουθία τελεστών μετάβασης που να οδηγεί από την αρχική κατάσταση στην κατάσταση s.
- Ο χώρος αναζήτησης είναι υποσύνολο του χώρου καταστάσεων
- Ο χώρος αναζήτησης μπορεί να αναπαρασταθεί ως γράφημα

Κατηγορίες προβλημάτων στην ΤΝ
(βάσει πολυπλοκότητας επίλυσης)

- Απλά προβλήματα (toy problems):
  - *Κύβοι,*
  - *N-puzzle,*
  - *Λαβύρινθος,*
  - *Πύργοι του Ανόι,*
  - *Αγρότης ή Κανίβαλοι,*
  - *Ποτήρια νερού,*
  - *Τρίλιζα*

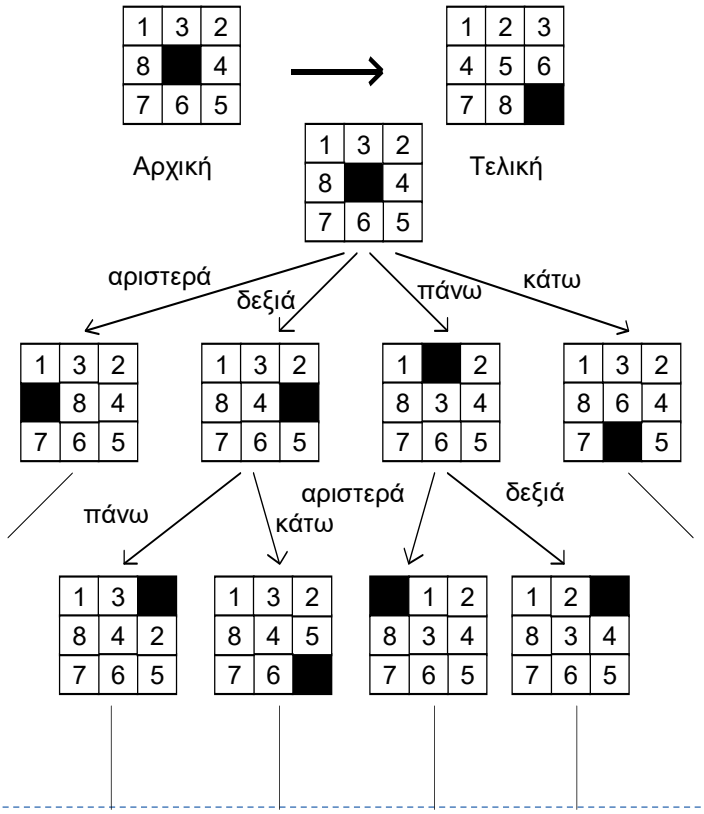
- Πραγματικά και πολύπλοκα προβλήματα (real word problems):
  - *Σκάκι,*
  - *Περιοδεύων πωλητής,*
  - *N-βασίλισσες*

Παράδειγμα: Το 8-puzzle

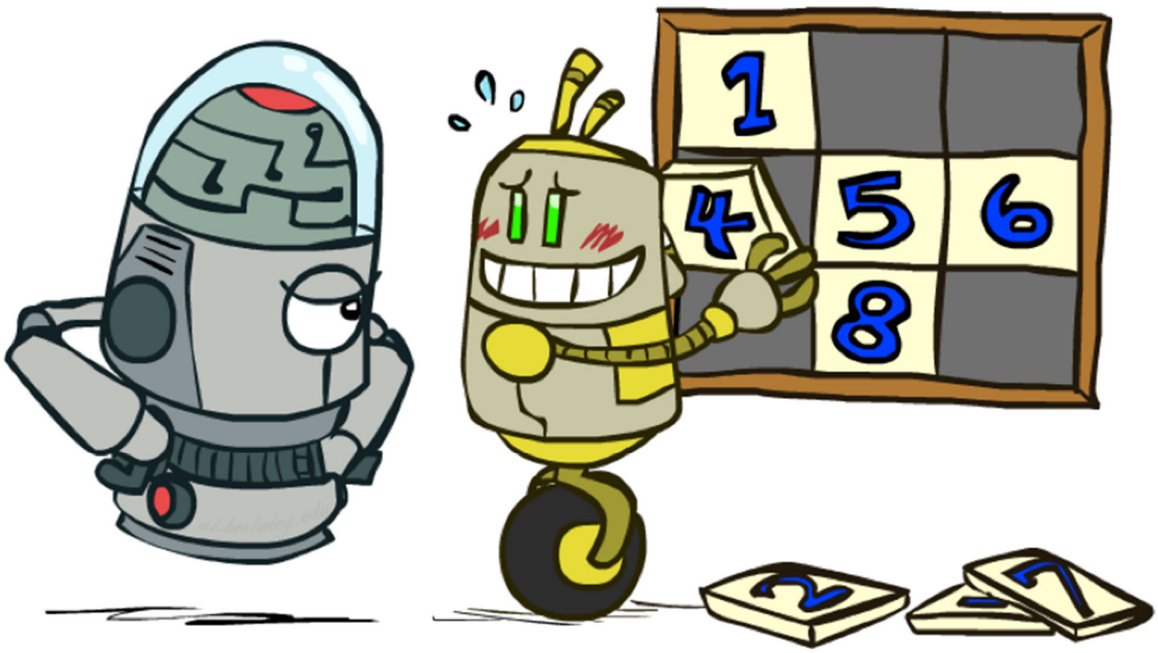


- **states?** Οι θέσεις των πλακιδίων
- **actions?** Μετακίνησε το κενό αριστερά, δεξιά, πάνω, κάτω
- **goal test?** = κατάσταση-στόχος (δοσμένη)
- **path cost?** 1 ανά κίνηση
- 8-puzzle: 9!/2 = 181.440 προσβάσιμες καταστάσεις
- n-puzzle: **NP-complete**
  - *tetris, candy crush, ναρκαλιευτής….*

8-puzzle: Χώρος καταστάσεων

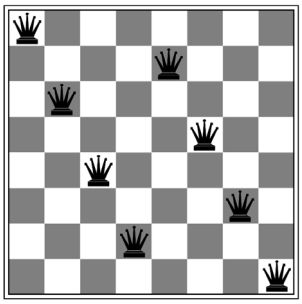


8-puzzle



Πηγή: ai.berkeley.edu

Το πρόβλημα των 8 βασιλισσών



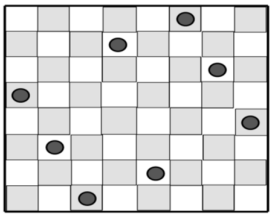
- **states?**: κάθε διάταξη από 0-8 βασίλισσες πάνω στο ταμπλό (αρχική: 0)
- **actions?**: Προσθήκη βασίλισσας σε ένα άδειο τετράγωνο
- **goal test?**: 8 βασίλισσες στο ταμπλό, χωρίς επίθεση
- **path cost?**: αδιάφορο, μόνο η τελική κατάσταση μετράει

- Αναπαράσταση καταστάσεων με ζεύγη συντεταγμένων
  - Π.χ. <(α, 1,1), (β, 3,2), …., (η, 8,8)>
  - Κατασκευαστική (constructive) προσέγγιση: Τοποθέτηση βασιλισσών μία-μία
- Δυνατές καταστάσεις: 64*63…*57 = $1.8X10^{14}$

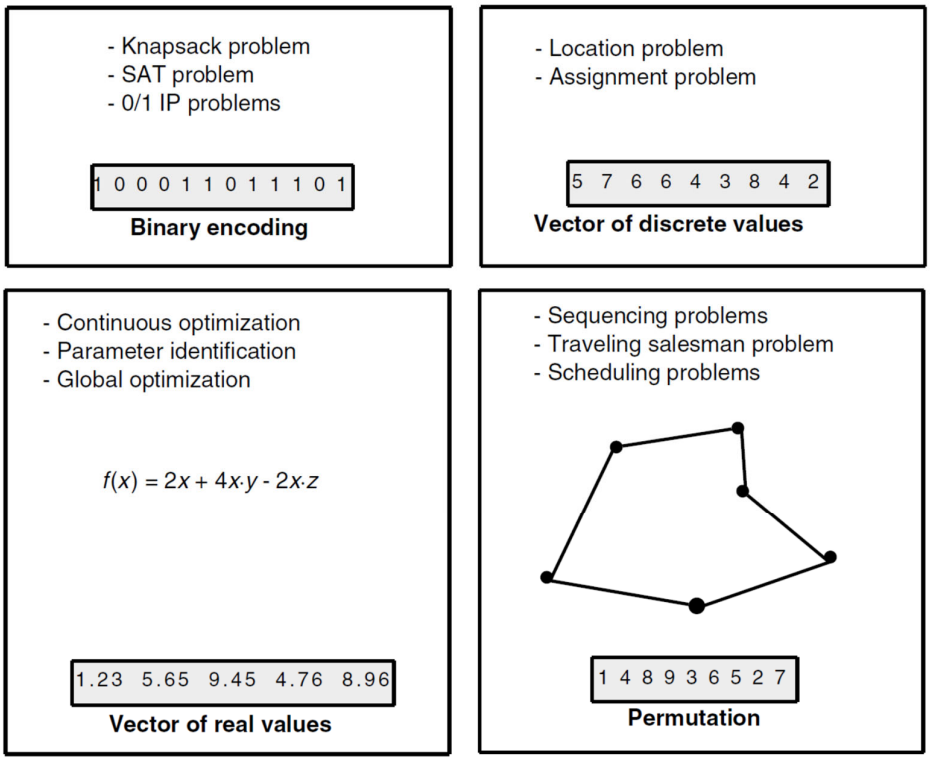
Η σημασία της αναπαράστασης

- Κατασκευαστική (constructive) προσέγγιση:
  - Τοποθέτηση βασιλισσών μία-μία
  - Πρόκειται συνήθως για *άπληστους (greedy)* αλγορίθμους
    - Η βασίλισσα τοποθετείται στην 1η νόμιμη θέση που θα βρεθεί
- **ΌΜΩΣ**
  - Δύο βασίλισσες δεν μπορεί να βρίσκονται στην ίδια γραμμή: 8^8=16.777.216 καταστάσεις
  - Δύο βασίλισσες δεν μπορεί να βρίσκονται στην ίδια στήλη: 8! = 40.320 καταστάσεις
  - Αντί για ζεύγη συντεταγμένων, μια κατάσταση μπορεί να αντιστοιχηθεί σε μία ***διάταξη* (permutation):**

<6, 4, 7, 1, 8, 2, 5, 3>



Παραδείγματα αναπαράστασης



Ιεραπόστολοι και Κανίβαλοι

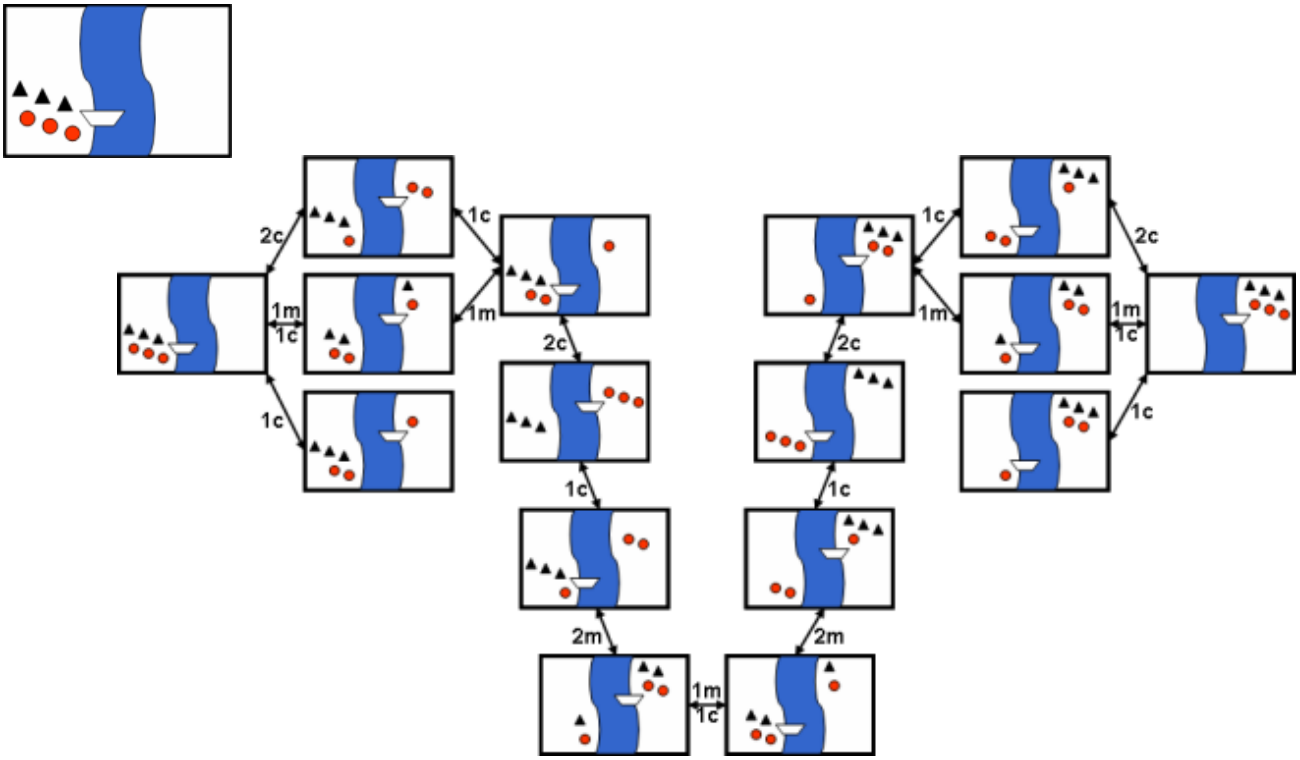
Στην όχθη ενός ποταμού βρίσκονται τρεις ιεραπόστολοι και τρεις κανίβαλοι. Αντικειμενικός σκοπός είναι και τα 6 άτομα -κανίβαλοι και ιεραπόστολοι- να φτάσουν στην άλλη όχθη.

Υπάρχει μια βάρκα που μπορούν χρησιμοποιήσουν, υπό την προϋπόθεση ότι σε κάθε διέλευση του ποταμού θα πρέπει να ***επιβαίνει τουλάχιστον ένα άτομο και το πολύ δύο.***

Δεν επιτρέπεται σε καμία περίπτωση σε μια από τις δυο όχθες ***να βρεθούν περισσότεροι κανίβαλοι από ιεραποστόλους.***

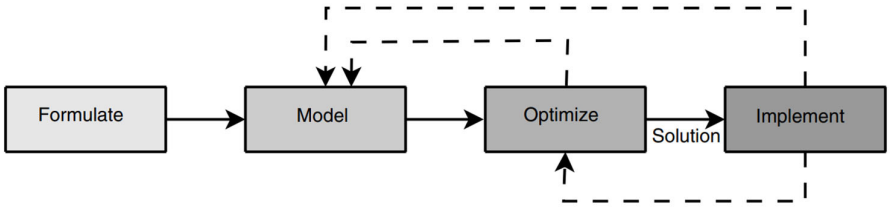
Με ποιο τρόπο μπορούν οι τρεις ιεραπόστολοι και οι τρεις κανίβαλοι να μεταβούν στην άλλη όχθη;

Ιεραπόστολοι και Κανίβαλοι: Χώρος καταστάσεων



Πηγή: http://www.aiai.ed.ac.uk/

Προβλήματα Βελτιστοποίησης



- Ορισμός (πρόβλημα *ελαχιστοποίησης*) : δυάδα ***(S, f)***
  - Δεδομένου του ***χώρου καταστάσεων S*** που αναπαριστά όλες τις έγκυρες λύσεις και
  - Δεδομένης μιας ***αντικειμενικής συνάρτησης*** (objective function) ***f: S → R***
  - Βρες **s* ∈ S** τέτοιο ώστε:
  - **f(s*) ≤ f (s)** **∀s ∈ S**
  - **s*** : ***Ολικό βέλτιστο***

Πρόβλημα βελτιστοποίησης: TSP

- Traveling Salesman Problem
  - Εύρεση του συντομότερου Χαμιλτονιανού Κύκλου
  - Είναι NP-hard (vs. NP-complete)
  - Optimization problem (vs. decision problem: ναι/όχι)
  - Συμμετρικό TSP: (n-1)!/2 διαδρομές
  - Συνδυαστική έκρηξη (combinatorial explosion)

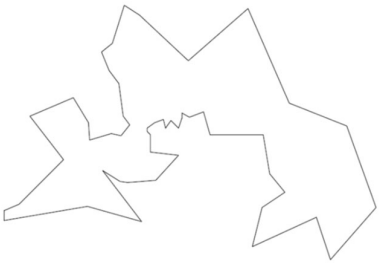
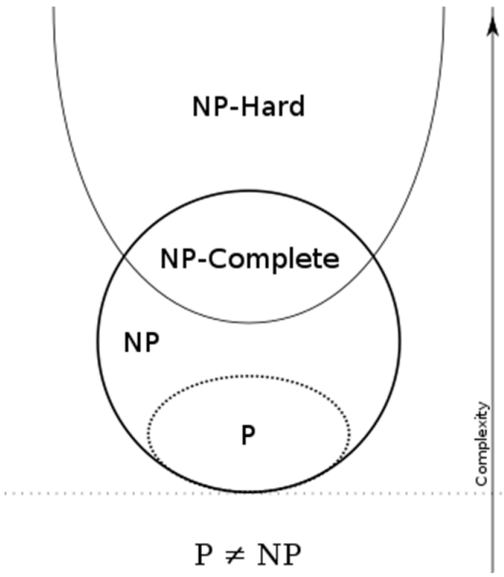


FIGURE 1.4 TSP instance with 52 cities.

Number of Cities n	Size of the Search Space
5	120
10	3, 628, 800
75	2.5×10^{109}

FIGURE 1.5 TSP instance with 24,978 cities.

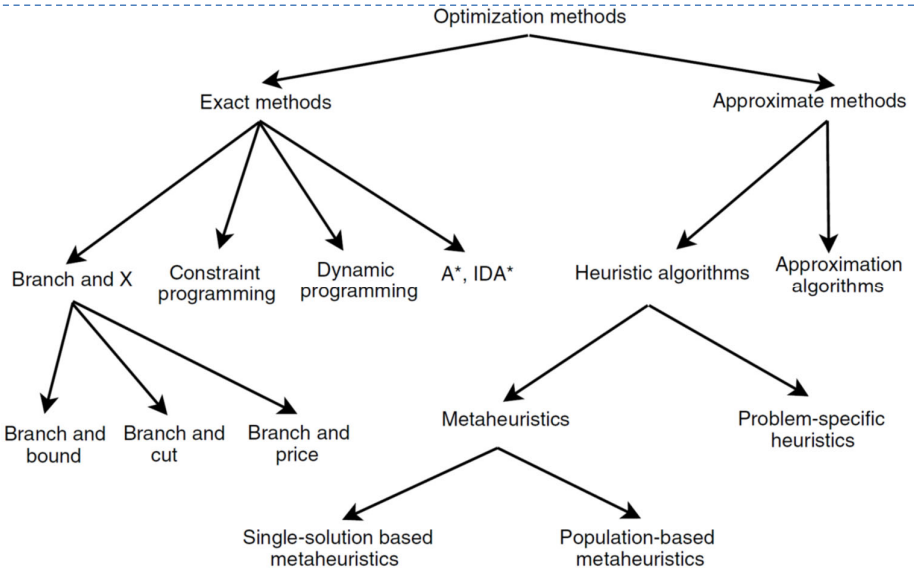
Πολυπλοκότητα



- Γνωστά NP-hard προβλήματα:
  - SAT, 3-SAT, k-SAT
  - TSP
  - Προβλήματα χρονοπρογραμματισμού
  - Το πρόβλημα της ανάθεσης
  - Το πρόβλημα του σάκου
  - ...

Complexity	Size = 10	Size = 20	Size = 30	Size = 40	Size = 50
$O(x)$	0.00001 s	0.00002 s	0.00003 s	0.00004 s	0.00005 s
$O(x^2)$	0.0001 s	0.0004 s	0.0009 s	0.0016 s	0.0025 s
$O(x^5)$	0.1 s	0.32 s	24.3 s	1.7 mn	5.2 mn
$O(2^x)$	0.001 s	1.0 s	17.9 mn	12.7 days	35.7 years
$O(3^x)$	0.059 s	58.0 mn	6.5 years	3855 centuries	2×10^8 centuries

Μέθοδοι Επίλυσης



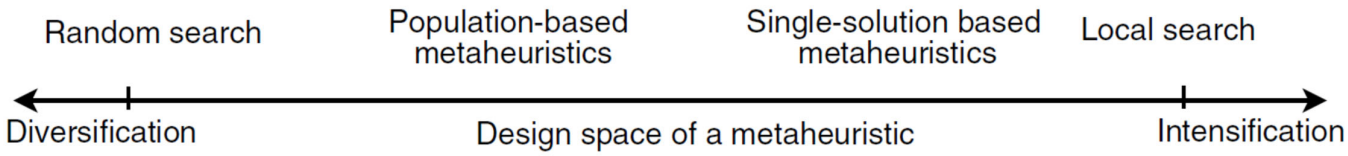
- **Ακριβείς μέθοδοι**: Εγγυημένα βέλτιστες λύσεις
- **Προσεγγιστικές μέθοδοι**: Εγγυημένα «καλές» λύσεις σε λογικό χρόνο, αλλά όχι βέλτιστες
  - **Ευριστικές (heuristics):** Εύρεση νέων στρατηγικών (κανόνων) για την επίλυση του προβλήματος

Μέθοδοι επίλυσης και πολυπλοκότητα

- **Ακριβείς μέθοδοι**
  - Εφαρμογή σε **μικρά στιγμιότυπα δύσκολων προβλημάτων**
  - Μικρό μέγεθος χώρου καταστάσεων
  - TSP: βέλτιστη λύση για 13.509 πόλεις (*(n-1)!/2*)
  - Πολυπλοκότητα $O(n^{2}*2^{n})$
- Η δυσκολία εξαρτάται και από τη δομή του προβλήματος
  - Όχι μόνο από το μέγεθος
  - *Μετάβαση φάσης* (phase transition): Η δυσκολία ενός προβλήματος αυξάνει μέχρι ένα μέγεθος *n* και μετά *μειώνεται*.
  - SAT, number partitioning…

Εξερεύνηση/Εκμετάλλευση

- **Εξερεύνηση** του χώρου καταστάσεων (diversification)
- **Εκμετάλλευση** των καλύτερων λύσεων που έχουν βρεθεί (intensification)



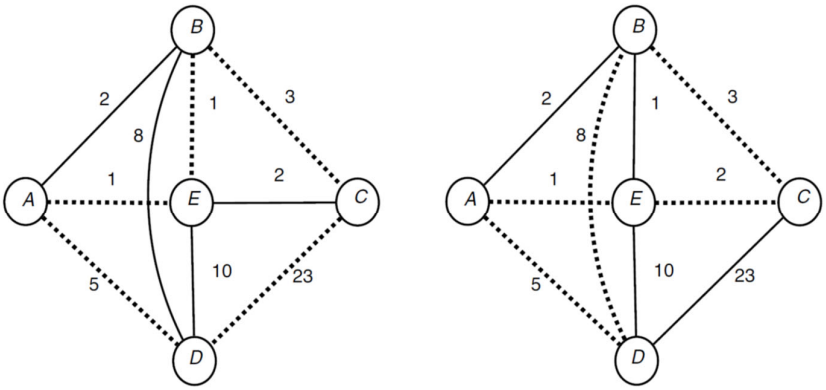
Αναζήτηση και βελτιστοποίηση

- Κατασκευαστικοί αλγόριθμοι
  - Ξεκινούν από μια κενή λύση και σε κάθε βήμα αναθέτουν και από μία μεταβλητή απόφασης του προβλήματος, μέχρι να φτιαχτεί μια πλήρης λύση
    - Π.χ. προσθέτουν βήματα σε μια διαδρομή
  - Είναι συνήθως *άπληστοι* (greedy)
- Επαναληπτικοί αλγόριθμοι
  - Ξεκινούν από μια πλήρη λύση *s* (οποιαδήποτε)
  - Μετασχηματίζουν τη λύση αυτή σε ένα σύνολο από νέες, *N(s) <- γειτονιά της s*
  - Επιλέγουν κάποια από αυτές, *s΄* και επαναλαμβάνουν, ώσπου να βρεθεί (ικανοποιητική) λύση
  - *Η κατάσταση του προβλήματος αντιστοιχεί σε μία αποδεκτή λύση, όχι μόνο σε ένα στάδιο αυτής*

$s = s_0$; /* Generate an initial solution s_0 */
While not Termination_Criterion **Do**
  Generate ($N(s)$) ; /* Generation of candidate neighbors */
  If there is no better neighbor **Then** Stop ;
  $s = s'$; /* Select a better neighbor $s' \in N(s)$ */
Endwhile
Output Final solution found (local optima).

Κατασκευαστικός (άπληστος) αλγόριθμος για το TSP

- Ξεκινά από μία πόλη και προσθέτει την επόμενη στη διαδρομή *(βλ. κ. 8 βασίλισσες)*
- Ευρετική για την επιλογή της επόμενης πόλης:
  - Κοντινότερος Γείτονας (Nearest Neighbor)



Διαχείριση περιορισμών

- Αναζητούμε βέλτιστες λύσεις που όμως να ικανοποιούν και κάποιους περιορισμούς
  - Π.χ. Knapsack
  - Καταστάσεις-λύσεις που δεν ικανοποιούν τους περιορισμούς είναι *μη νόμιμες*
  - Μπορεί όμως να προκύψουν στον χώρο καταστάσεων!
- Στρατηγικές διαχείρισης περιορισμών
  - **Απόρριψη** λύσης (και αντικατάσταση)
  - **Ανάθεση ποινής** (π.χ. αύξηση κόστους)
  - **Επιδιόρθωση** λύσης
  - **Αποκωδικοποίηση**
    - Ορίζεται συνάρτηση που *μετασχηματίζει* κάθε κατάσταση-λύση σε έγκυρη κατάσταση (π.χ. κανονικοποίηση στο [0,1])
  - **Ενσωμάτωση**
    - η αναπαράσταση και οι τελεστές δεν δημιουργούν μη-νόμιμες λύσεις

Επίλυση προβλημάτων με αναζήτηση

Αναζήτηση

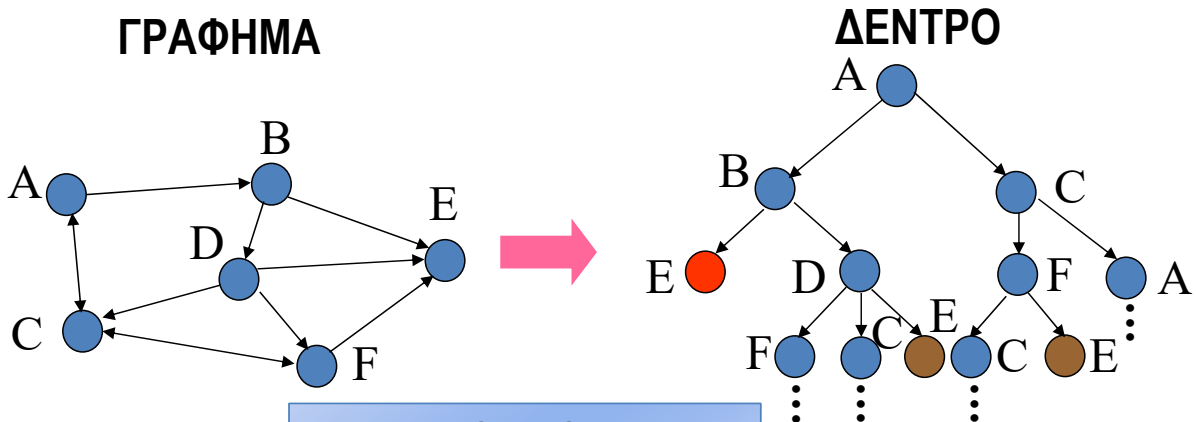
- **ΔΙΑΔΙΚΑΣΙΑ ΑΝΑΖΗΤΗΣΗΣ**
  1. **Ανάπτυξη καταστάσεων (states expansion)** με εφαρμογή των τελεστών μετάβασης και κατάλληλη στρατηγική αναζήτησης (search strategy)
  2. **Έλεγχος** για τελική κατάσταση (κατάσταση στόχου)
- **ΤΥΠΟΙ ΑΝΑΖΗΤΗΣΗΣ**
  - **Εξαντλητική Αναζήτηση (Exhaustive Search)**
    Ή τυφλή (blind) ή απληροφόρητη (uninformed) αναζήτηση
  - Αναπτύσσεται όλος ο χώρος αναζήτησης λύσεων
  - **Ευριστική (ή Ευρετική) Αναζήτηση (Heuristic Search)**
    Ή πληροφορημένη (informed) αναζήτηση
  - Αναπτύσσεται μέρος του χώρου αναζήτησης λύσεων με βάση κάποιο ευρετικό (= έξυπνο τρόπο)

Γράφημα αναζήτησης

- Αναπαράσταση του χώρου αναζήτησης ενός προβλήματος ως γράφημα
- Βασικά στοιχεία
  - **κόμβοι** (nodes) ↔ καταστάσεις (states)
  - **ακμές/κλάδοι** (edges/branches) ↔ τελεστές (operators)
  - **διαδρομή** (path)
    - ακολουθία κόμβων που συνδέονται με διαδοχικές ακμές/κλάδους

Κάθε γράφημα αναζήτησης μπορεί να διαπεραστεί με ένα αντίστοιχο δέντρο αναζήτησης, το οποίο όμως μπορεί να έχει μονοπάτια άπειρου μήκους

Δέντρο Αναζήτησης



Ορολογία Δέντρων

- ρίζα (root) → A
- κόμβοι (nodes) → A, B, C, …
- ενδιάμεσοι κόμβοι → B, C, …
- κλάδοι (branches) → A-B, B-D, …
- φύλλα (leaves) → E

- απόγονος (descendant)
- πρόγονος (ancestor)
- παιδιά (children)
- γονέας (parent)
- αδέλφια (siblings)

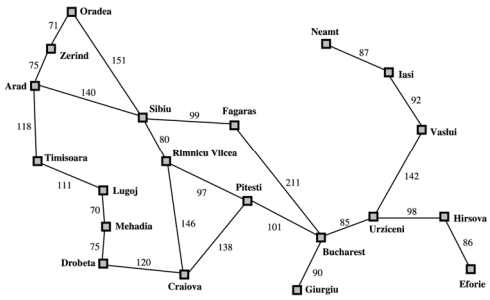
Δέντρο Αναζήτησης

- **ΚΟΜΒΟΙ (NODES)**
  - **Μη τερματικοί**
    - ανοικτοί (open) (μη αναπτυγμένοι) → μέτωπο αναζήτησης
    - κλειστοί (closed) (αναπτυγμένοι) → κλειστό σύνολο
  - **Τερματικοί**
    - στόχου (goal)
    - αδιέξοδοι (dead-ends)
- **ΔΙΑΔΡΟΜΕΣ (PATHS)**
  - **Λύσης (solution paths)**
  - **Αδιέξοδοι (dead-end paths)**
  - **Ατέρμονες (infinite paths)**
  - **Κυκλικές (cyclic paths)**

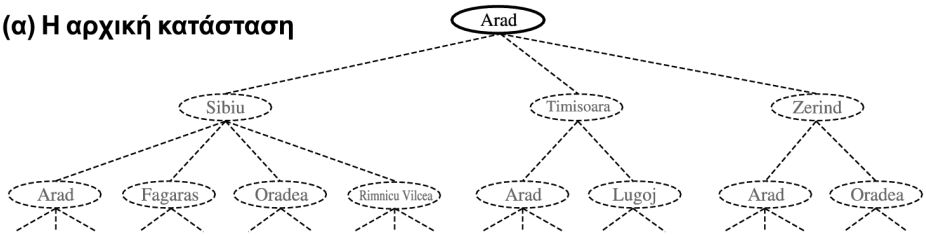
Αλγόριθμος Tree-Search

function TREE-SEARCH(*problem*) **returns** a solution, or failure
  initialize the frontier using the initial state of *problem*
  loop do
    if the frontier is empty **then return** failure
    choose a leaf node and remove it from the frontier
    if the node contains a goal state **then return** the corresponding solution
    expand the chosen node, adding the resulting nodes to the frontier

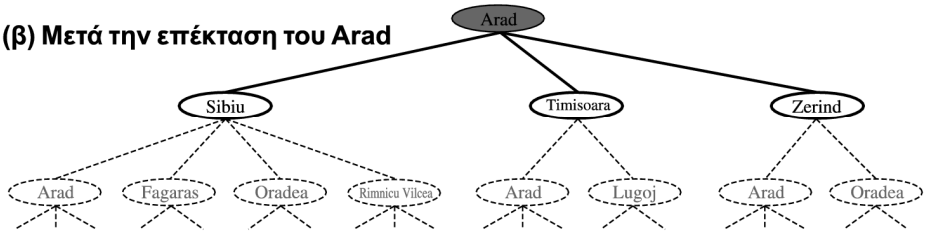
Δένδρο Αναζήτησης-Παράδειγμα



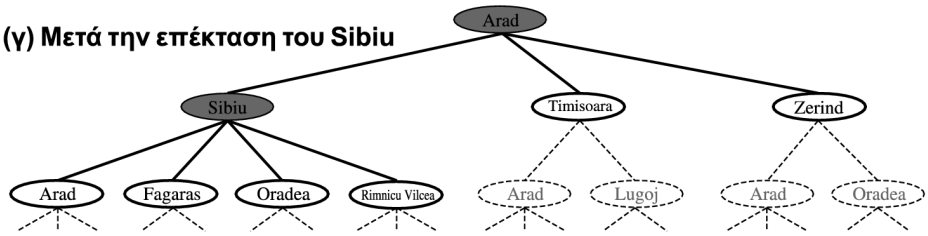
(α) Η αρχική κατάσταση



(β) Μετά την επέκταση του Arad

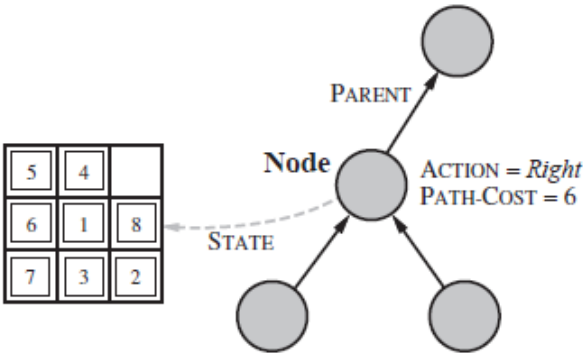


(γ) Μετά την επέκταση του Sibiu



Κόμβος: Δομή δεδομένων

- Κόμβος αναζήτησης
  - ⟨State, Parent, Action, Path-Cost⟩



- Επέκταση τρέχουσας κατάστασης → παραγωγή νέων καταστάσεων
- Στρατηγική αναζήτησης

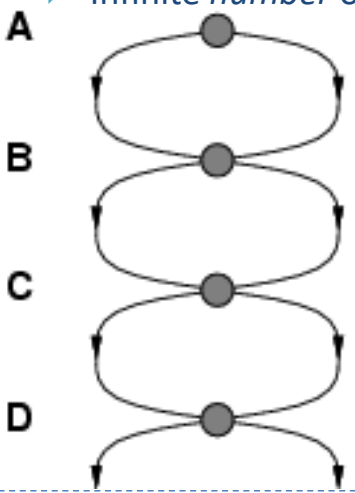
Μέτωπο αναζήτησης

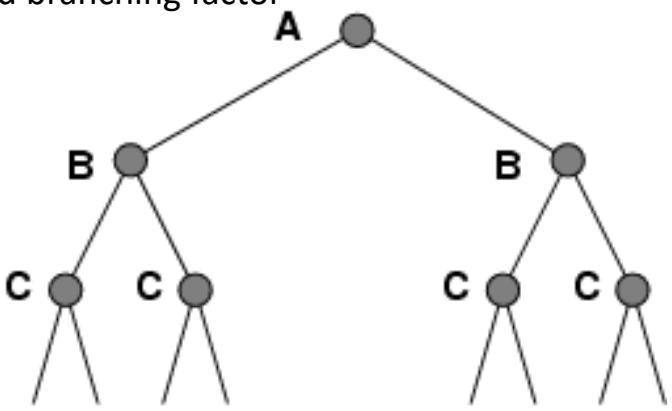
- **Μέτωπο αναζήτησης** ή **Σύνορο** (fringe): Η λίστα των καταστάσεων που έχουν παραχθεί, αλλά δεν έχουν ακόμα επεκταθεί.
  - Υλοποίηση με **ουρά (FIFO)**
  - Υλοποίηση με **στοίβα (LIFO)**
  - Υλοποίηση με ουρά προτεραιότητας: Εφαρμόζεται ***συνάρτηση διάταξης*** στα στοιχεία της λίστας

- Βασικές λειτουργίες στο μέτωπο αναζήτησης:
  - bool EMPTY? (queue)
  - node POP (queue)
  - void INSERT(queue)

Επαναλαμβανόμενες καταστάσεις

- **Repeated states** (επαναλαμβανόμενες καταστάσεις), προκαλούνται:
  - Redundant paths (πλεονάζοντα μονοπάτια)
  - Loopy paths (κυκλικά μονοπάτια): **Είναι** Redundant, **Είναι** Infinite
- **Infinite Search Tree** (άπειρο δέντρο αναζήτησης), προκαλείται:
  - Infinite paths, go arbitrarily deep
  - Infinite *number* of paths, unbounded branching factor





Αλγόριθμος Graph-Search

function TREE-SEARCH(*problem*) **returns** a solution, or failure
  initialize the frontier using the initial state of *problem*
  loop do
    if the frontier is empty **then return** failure
    choose a leaf node and remove it from the frontier
    if the node contains a goal state **then return** the corresponding solution
    expand the chosen node, adding the resulting nodes to the frontier

function GRAPH-SEARCH(*problem*) **returns** a solution, or failure
  initialize the frontier using the initial state of *problem*
  initialize the explored set to be empty
  loop do
    if the frontier is empty **then return** failure
    choose a leaf node and remove it from the frontier
    if the node contains a goal state **then return** the corresponding solution
    add the node to the explored set
    expand the chosen node, adding the resulting nodes to the frontier
      only if not in the frontier or explored set

- **Μνήμη:** Ο αλγόριθμος θυμάται αυτά που έχει επισκεφθεί
- Explored Set = ***κλειστό σύνολο***

Απόδοση αλγορίθμων αναζήτησης

- **Πληρότητα (Completeness)**
- **Βέλτιστη συμπεριφορά (Optimality)**
- **Χρονική πολυπλοκότητα (Time complexity)**
- **Χωρική πολυπλοκότητα (Space complexity)**
- Μέγεθος προβλήματος: Καθορίζεται από το μέγεθος του γραφήματος αναζήτησης
  - **Παράγοντας διακλάδωσης, *b*** (branching factor)
  - **Βάθος, *d*** (depth) του πιο ρηχού κόμβου-στόχου
  - **Μέγιστο μήκος, *m*,** οποιασδήποτε διαδρομής του χώρου καταστάσεων.

- **Αποτελεσματικότητα αλγορίθμου**
- **κόστος αναζήτησης** (που εξαρτάται από την πολυπλοκότητα) ή και
- **κόστος λύσης** (δηλ. το κόστος διαδρομής της λύσης που βρέθηκε)

Ολικό κόστος = κόστος αναζήτησης+ κόστος λύσης

- Εξαρτάται από την εφαρμογή:
  - Άλλοτε θέλουμε μια πολύ καλή (ή και τη βέλτιστη) λύση χωρίς να μας απασχολεί ο χρόνος που θα ξοδέψουμε για να τη βρούμε.
  - Άλλοτε θέλουμε μια (οποιαδήποτε) λύση γρήγορα.

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