

Αλγόριθμοι Τοπικής Αναζήτησης

- Αναζήτηση με αναρρίχηση λόφων
- Προσομοιωμένη απόπτηση
- Τοπική ακτινική αναζήτηση
- Γενετικοί αλγόριθμοι

Επαναληπτική Βελτίωση

Σε πολλά προβλήματα αναζήτησης, δεν μας ενδιαφέρει η διαδρομή προς μια κατάσταση στόχου αλλά η ίδια η κατάσταση στόχου.

Παράδειγμα:

- Έυρεση λύσης που να ικανοποιεί συγκεκριμένους περιορισμούς, π.χ. το πρόβλημα των 8 βασιλισσών ή το πρόβλημα του χρονοπρογραμματισμού εργασιών.

Σε τέτοιες περιπτώσεις, μπορούμε να χρησιμοποιήσουμε μια τεχνική που λέγεται **επαναληπτική βελτίωση (iterative improvement)**: ξεκινάμε με μια μοναδική τρέχουσα κατάσταση, και προσπαθούμε να τη βελτιώσουμε.

Το ίδιο πλαίσιο εργασίας μπορεί να εφαρμοστεί και σε **προβλήματα βελτιστοποίησης** όπου μας ενδιαφέρει η λύση που βελτιστοποιεί μια δοσμένη αντικειμενική συνάρτηση.

Αλγόριθμοι Τοπικής Αναζήτησης

Οι αλγόριθμοι **τοπικής αναζήτησης (local search)** χρησιμοποιούν την τεχνική της επαναληπτικής βελτίωσης και λειτουργούν ως εξής:

- Διαλέγουμε μια 'λύση' από το χώρο αναζήτησης και την **αποτιμούμε**. Ονομάζουμε αυτή τη λύση **τρέχουσα**.
- Εφαρμόζουμε ένα **μετασχηματισμό** στην τρέχουσα λύση για να παράγουμε μια νέα λύση. Αξιολογούμε τη νέα λύση.
- Αν η νέα λύση είναι καλύτερη από την τρέχουσα, τότε την ανταλλάσσουμε με την τρέχουσα λύση· διαφορετικά απορρίπτουμε τη νέα λύση.
- Επαναλαμβάνουμε τα παραπάνω βήματα μέχρι κανένας μετασχηματισμός να μη βελτιώνει άλλο την τρέχουσα λύση.

Τοπική Αναζήτηση ή Εξαντλητική Αναζήτηση;

Οι αλγόριθμοι **τοπικής αναζήτησης** λειτουργούν βελτιώνοντας μια **μοναδική τρέχουσα** κατάσταση και γενικά μετακινούνται μόνο σε **γειτονικές** της καταστάσεις. Σε κάθε βήμα ενός αλγόριθμου τοπικής αναζήτησης έχουμε μια **πλήρη αλλά ατελή** λύση στο δοσμένο πρόβλημα αναζήτησης. Μπορεί να μην βρούμε την βέλτιστη λύση αλλά μια 'αρκετά καλή' λύση σε πολύ μικρό χρόνο.

Οι αλγόριθμοι **εξαντλητικής αναζήτησης** που είδαμε νωρίτερα (π.χ., ο A^*) λειτουργούν με πολλές **μερικές λύσεις** και τις επεκτείνουν σε πλήρεις. Υπάρχουν τέτοιοι αλγόριθμοι που βρίσκουν πάντα την βέλτιστη λύση.

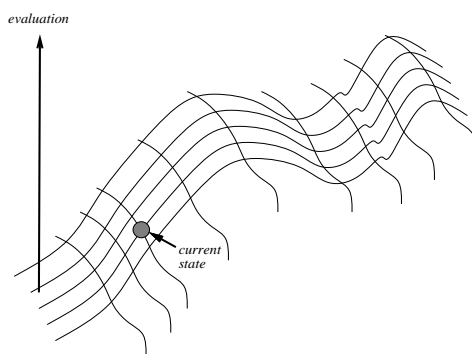
Αλγόριθμοι Τοπικής Αναζήτησης

Καλές ιδιότητες των αλγορίθμων τοπικής αναζήτησης:

- Χρειάζονται σταθερό χώρο.
- Είναι κατάλληλοι για **on-line** καθώς και για **off-line** προβλήματα.
- Μπορούν να βρουν **ικανοποιητικές λύσεις σε μεγάλους ή και άπειρους χώρους λύσεων** όπου η εξαντλητική αναζήτηση θα αποτύγχανε.

Αλγόριθμοι Τοπικής Αναζήτησης

Ιδέα: Ξεκινάμε με μια 'λύση' και κάνουμε μετασχηματισμούς μέχρι να βρούμε μια λύση. Γραφικά:



Παράδειγμα: Το Πρόβλημα του Πλανόδιου Πωλητή (TSP)

TSP: Έστω G ένας (κατευθυνόμενος ή μη) γράφος με n κόμβους και μη αρνητικά βάρη σε κάθε ακμή. Να βρεθεί ένα μονοπάτι στον G που έχει το χαμηλότερο κόστος, περνάει από κάθε κόμβο μόνο μια φορά, και επιστρέφει σ' ένα δοθέντα αρχικό κόμβο.

Σύνολο λύσεων: Το σύνολο των διατάξεων (**permutations**) των n πόλεων.

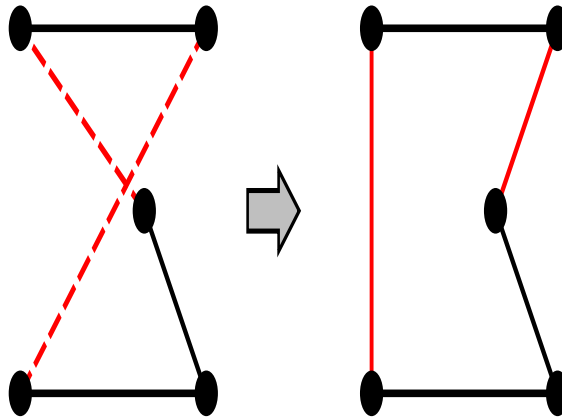
Κάθε διάταξη αντιστοιχεί σε μια ταξινομημένη λίστα των πόλεων που θα επισκεφθεί ο πωλητής: ξεκινάει από την αφετηρία, συνεχίζει μέχρι τον τελευταίο σταθμό και γυρίζει πίσω στην αφετηρία.

Ο Αλγόριθμος 2-Opt

1. Ξεκινάμε με μια τυχαία πλήρη διαδρομή T (δηλαδή, μια τυχαία διάταξη).
2. Ορίζουμε ως **γειτονιά (neighbourhood)** της T , το σύνολο όλων των διαδρομών που μπορούν να κατασκευαστούν με μια **εναλλαγή 2 ακμών (2-interchange move)** της T .
3. **Αναζητούμε** στη γειτονιά της T μια νέα διαδρομή T' . Αν αυτή η διαδρομή είναι καλύτερη από την T (δηλαδή, **έχει χαμηλότερο κόστος**), τότε αντικαθιστούμε την T με την T' (σταματάμε στην πρώτη τέτοια διαδρομή που θα βρούμε). Αν δεν μπορούμε να βρούμε μια καλύτερη διαδρομή, **τερματίζουμε** τον αλγόριθμο.

Η προκύπτουσα διάταξη ονομάζεται **2-βέλτιστη (2-optimal)**.

Εναλλαγή 2 Ακμών



Αυτή η κίνηση **διαγράφει** δύο μη γειτονικές ακμές, διασπώντας έτσι τη διαδρομή σε δύο μικρότερες διαδρομές, και στη συνέχεια **επανασυνδέει** αυτές τις διαδρομές με τον μόνο εναλλακτικό δυνατό τρόπο.

Ο Αλγόριθμος 2-Opt

Ο αλγόριθμος 2-Opt είναι ένας απλός αλγόριθμος τοπικής αναζήτησης για το πρόβλημα του πλανόδιου πωλητή. Όμοια μπορούμε να ορίσουμε τον αλγόριθμο k -Opt για $k > 2$. Θα είναι καλύτερος από τον 2-Opt;

Ο καλύτερος γνωστός αλγόριθμος τοπικής αναζήτησης για το TSP είναι ο αλγόριθμος των **Lin-Kernighan** που είναι αποδοτικός ακόμα και για προβλήματα με 10^6 πόλεις και παράγει σχεδόν βέλτιστες λύσεις (π.χ., 2% χειρότερες από την βέλτιστη):

- Βασίζεται στον k -Opt αλλά επιτρέπει στο k να είναι διαφορετικό σε κάθε επανάληψη.
- Δεν επιλέγει τον πρώτο καλύτερο γείτονα, αλλά τον καλύτερο από όλους τους γείτονες.



Αναζήτηση με Αναρρίχηση Λόφων

```

function HILL-CLIMBING(problem)
returns a state that is a local maximum
  inputs: problem, a problem
  local variables: current, a node
                   neighbour, a node
  current ← MAKENODE(RANDOMSTATE[problem])
  loop do
    neighbour ← a highest-valued successor of current
    if VALUE[neighbour] ≤ VALUE[current] then return current
    current ← neighbour
  end

```

Αναζήτηση με Αναρρίχηση Λόφων

- Ο αλγόριθμος αναζήτησης με αναρρίχηση λόφων είναι ο απλούστερος αλγόριθμος τοπικής αναζήτησης.
- Η έκδοση του αλγόριθμου που παρουσιάσαμε είναι η έκδοση της **πλέον απότομης ανάβασης (steepest ascent)** και χρησιμοποιείται για προβλήματα μεγιστοποίησης.
- Οι διάδοχες καταστάσεις (successors) αναζητώνται με **συστηματικό** τρόπο.
- Η επιλογή μεταξύ ισοδύναμων διαδόχων καταστάσεων γίνεται **τυχαία**.
- Αν έχουμε ένα **πρόβλημα ελαχιστοποίησης**, απλά αλλάζουμε στον αλγόριθμο το 'highest-valued' σε 'lowest-valued' και το \leq σε \geq , και έχουμε την έκδοση της **πλέον απότομης κατάβασης (steepest descent)**.

Παράδειγμα: Το Πρόβλημα των 8 Βασιλισσών

Τυπικός Ορισμός:

- **Καταστάσεις:** Οποιαδήποτε διάταξη με 8 βασίλισσες στη σκακιέρα.
- **Ενέργειες:** Μετακίνηση μιας βασίλισσας στη στήλη της.
Σε κάθε βήμα έχουμε $8 \cdot 7 = 56$ δυνατές επόμενες καταστάσεις.
- **Έλεγχος στόχου:** 8 βασίλισσες στη σκακιέρα, από τις οποίες καμιά δεν απειλείται.
- **Συνάρτηση αξιολόγησης (κόστος):** Αριθμός ζευγαριών βασιλισσών που αλληλοαπειλούνται.

Έτσι έχουμε ένα **πρόβλημα ελαχιστοποίησης**: θέλουμε να βρούμε μια κατάσταση που ελαχιστοποιεί τη συνάρτηση αξιολόγησης.

Παράδειγμα Κατάστασης

18	12	14	13	13	12	14	14
14	16	13	15	12	14	12	16
14	12	18	13	15	12	14	14
15	14	14	♠	13	16	13	16
♠	14	17	15	♠	14	16	16
17	♠	16	18	15	♠	15	♠
18	14	♠	15	15	14	♠	16
14	14	13	17	12	14	12	18

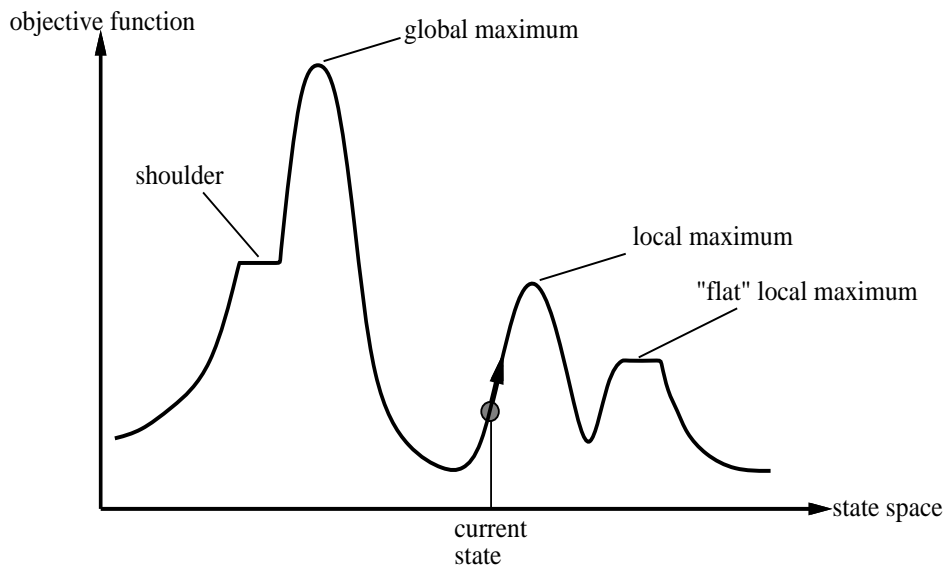
Η τιμή του κόστους για την παραπάνω κατάσταση είναι 17. Οι αριθμοί στα τετράγωνα δείχνουν τα νέα κόστη αν μια βασίλισσα μετακινηθεί σε κάποιο άλλο τετραγωνάκι στη στήλη της.

Προβλήματα της Αναζήτησης με Αναρρίχηση Λόφων

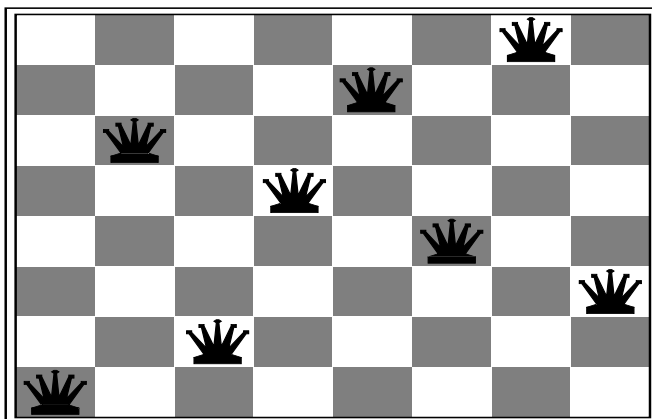
- Τοπικά βέλτιστα (local optimums).
- Οροπέδια (plateaux):
 - Επίπεδα τοπικά βέλτιστα
 - Όμοι (shoulders)
- Κορυφογραμμές (ridges).

Πώς μπορούμε να αντιμετωπίσουμε αυτά τα προβλήματα; Η κατάλληλη επιλογή μπορεί να εξαρτάται από το πρόβλημα.

Προβλήματα της Αναζήτησης με Αναρρίχηση Λόφων

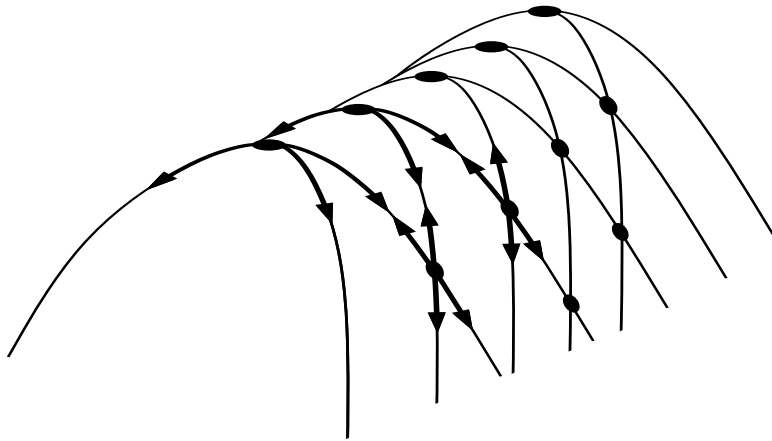


Τοπικά Ελάχιστα



Η τιμή του κόστους για την παραπάνω κατάσταση είναι 1. Όλοι οι γείτονες της κατάστασης αυτής έχουν κόστος > 1 , άρα έχουμε ένα τοπικό ελάχιστο (local minimum).

Κορυφογραμμές



Εδώ έχουμε μια ακολουθία από τοπικά μέγιστα που είναι δύσκολο να τα διασχίσουμε με τοπική αναζήτηση.

Αναρρίχηση Λόφων για τις 8 Βασίλισσες

Ας ξεκινήσουμε με μια τυχαία παραγόμενη κατάσταση στο πρόβλημα των 8 βασίλισσών. Τότε η εφαρμογή του αλγόριθμου αναρρίχησης λόφων θα έχει τα εξής αποτελέσματα:

- Επιλύει το 14% των προβλημάτων με 4 βήματα κατά μέσο όρο.
- Κολλάει σε τοπικά βέλτιστα ή οροπέδια στο 86% των περιπτώσεων με 3 βήματα κατά μέσο όρο.

Υπενθύμιση: Συνολικός χώρος καταστάσεων:

$$8^8 \approx 17 \text{ εκατομμύρια καταστάσεις}$$

Πως Αποφεύγουμε τα Οροπέδια;

Όταν ο αλγόριθμος αναρρίχησης λόφων φτάνει σε ένα **οροπέδιο** και δεν υπάρχουν ανοδικές κινήσεις, τότε σταματάει.

Τότε μπορούμε να καταφύγουμε σε μια **πλάγια κίνηση (sideways move)**, δηλαδή μια κίνηση προς μια κατάσταση η οποία έχει την **ίδια τιμή** με την τρέχουσα, με την ελπίδα ότι το οροπέδιο είναι ώμος.

Προσοχή: πρέπει να είμαστε προσεκτικοί έτσι ώστε να μην μπούμε σε έναν **ατέρμονα βρόχο** (όταν είμαστε σε ένα οροπέδιο που δεν είναι ώμος). Μια συνηθισμένη λύση για το πρόβλημα αυτό είναι **να περιορίσουμε τον αριθμό των διαδοχικών πλάγιων κινήσεων**.

Παράδειγμα: Αν περιορίσουμε τον αριθμό των διαδοχικών πλάγιων κινήσεων σε 100 στο πρόβλημα των 8 βασιλισσών, το ποσοστό των προβλημάτων που καταφέρνουμε να λύσουμε ανεβαίνει στο 94%.

Πως Αποφεύγουμε τα Τοπικά Βέλτιστα;

Θα παρουσιάσουμε δύο αλγόριθμους που αποφεύγουν τα τοπικά βέλτιστα:

- **Αναρρίχηση λόφων με τυχαίες επανεκκινήσεις (random-restart hill-climbing)**
- **Προσομοιωμένη ανόπτωση (simulated annealing)**

Αναρρίχηση Λόφων με Τυχαίες Επανεκκινήσεις

Ιδέα: Αν δεν καταφέρουμε να βρούμε την κατάσταση στόχου την πρώτη φορά, ας προσπαθήσουμε ξανά!

Η αναρρίχηση λόφων με τυχαίες επανεκκινήσεις (**random restart hill climbing**) πραγματοποιεί μια σειρά από αναζητήσεις με αναρρίχηση λόφων αρχίζοντας από τυχαία παραγόμενες αρχικές καταστάσεις και σταματά όταν βρεθεί ένας στόχος.

Η αναρρίχηση λόφων με τυχαίες επανεκκινήσεις είναι **πλήρης** επειδή, με πιθανότητα που τείνει στο 1, θα παράγουμε τελικά μια κατάσταση στόχου ως αρχική κατάσταση.

Αν κάθε αναζήτηση με αναρρίχηση λόφων έχει πιθανότητα επιτυχίας p , τότε ο αναμενόμενος αριθμός επανεκκινήσεων που απαιτούνται για να φτάσουμε σε μια λύση είναι $1/p$.

Αναρρίχηση Λόφων με Τυχαίες Επανεκκινήσεις

Παράδειγμα: Το πρόβλημα των 8 βασιλισσών

Όπως είδαμε προηγουμένως, $p \approx 0.14$.

Σε αυτή την περίπτωση χρειαζόμαστε περίπου 7 επαναλήψεις (6 αποτυχίες και 1 επιτυχία).

Αναμενόμενος αριθμός βημάτων: Αριθμός βημάτων μιας επιτυχημένης επανάληψης συν $(1/p) - 1$ φορές τον αριθμό των βημάτων μιας αποτυχημένης επανάληψης. Αυτά είναι περίπου $6 \cdot 3 + 4 = 22$ βήματα στην περίπτωση μας (χρησιμοποιώντας τον αριθμό των βημάτων που υπολογίστηκαν παραπάνω για τις επιτυχίες/αποτυχίες).

Η αναρρίχηση λόφων με τυχαίες επανεκκινήσεις είναι **πολύ αποτελεσματική** για το πρόβλημα των 8 βασιλισσών. Ακόμα και για **τρία εκατομμύρια βασίλισσες**, μπορεί να βρει λύση σε λιγότερο από ένα λεπτό.

Αναρρίχηση Λόφων με Τυχαίες Επανεκκινήσεις

Η επιτυχία της αναρρίχησης λόφων με τυχαίες επανεκκινήσεις εξαρτάται σημαντικά από το **σχήμα του τοπίου του χώρου καταστάσεων**. Αν υπάρχουν **λίγα τοπικά μέγιστα ή οροπέδια**, η αναρρίχηση λόφων με τυχαίες επανεκκινήσεις θα βρει μια καλή λύση πολύ γρήγορα.

Πολλά πρακτικά προβλήματα έχουν χώρους καταστάσεων που δεν έχουν αυτή την ιδιότητα: μπορεί να έχουν **εκθετικά μεγάλο αριθμό τοπικών μεγίστων** στα οποία μπορεί να παγιδευτεί ένας αλγόριθμος αναρρίχησης λόφων.

Συζήτηση

Ένας αλγόριθμος αναρρίχησης λόφων που δεν κάνει ποτέ **κατηφορικές κινήσεις** προς καταστάσεις μικρότερης αξίας μπορεί να είναι μη πλήρης.

Ένας **τυχαίος περίπατος (random walk)**, δηλαδή, μετακίνηση προς μια διάδοχη κατάσταση που επιλέγεται ομοιόμορφα στην τύχη από το σύνολο των διαδόχων καταστάσεων, είναι πλήρης (απόδειξη;) αλλά έχει εξαιρετικά χαμηλή απόδοση.

Πώς μπορούμε να συνδυάσουμε αυτά τα δύο;

Αυτό είναι ένα κλασσικό δίλημμα ανάμεσα στην **εξερεύνηση (exploration)** του χώρου αναζήτησης και στην **εκμετάλλευση (exploitation)** της ατελούς λύσης που έχουμε στα χέρια μας. Πώς λύνουμε το δίλημμα αυτό; Μια απάντηση δίνεται από τον αλγόριθμο **προσομοιωμένης ανόπτωσης (simulated annealing)**.

Τι είναι Ανόπτηση;

Η **ανόπτηση (annealing)** των μετάλλων είναι η διαδικασία που χρησιμοποιείται στη μεταλλουργία για να μαλακώσουμε ή να σκληρύνουμε μέταλλα και γυαλί θερμαίνοντάς τα σε υψηλή θερμοκρασία και στη συνέχεια ψύχοντάς τα σταδιακά, επιτρέποντας έτσι στο υλικό να στερεοποιηθεί σε μια κρυσταλλική κατάσταση χαμηλής ενέργειας.

Ο Αλγόριθμος της Προσομοιωμένης Ανόπτησης

Η ανακάλυψη του αλγορίθμου προσομοιωμένης ανόπτησης (simulated annealing) είναι ένα στιγμιότυπο της χρήσης των ιδεών της **στατιστικής μηχανικής (statistical mechanics)** – μια περιοχή της φυσικής συμπυκνωμένης ύλης – για τη χρήση **μεγάλων και πολύπλοκων προβλημάτων βελτιστοποίησης**.

Η στατιστική μηχανική επικεντρώνεται στην ανάλυση συγκεντρωτικών ιδιοτήτων μεγάλου πλήθους ατόμων που βρίσκονται σε δείγματα υγρής ή στερεής ύλης.

Η αρχική δημοσίευση για την προσομοιωμένη ανόπτηση είναι:

S. Kirkpatrick, C. D. Gelatt Jr., and M. P. Vecchi.

“Optimization by Simulated Annealing”. Science, Volume 220, Number 4598, May 1983.

Στατιστική Μηχανική και Βελτιστοποίηση

Φυσικό Σύστημα	Πρόβλημα Βελτιστοποίησης
κατάσταση	εφικτή λύση
ενέργεια	συνάρτηση αξιολόγησης
βασική κατάσταση	βέλτιστη λύση
σβήσιμο	τοπική αναζήτηση
θερμοκρασία	παράμετρος ελέγχου T
ανόπτηση	προσομοιωμένη ανόπτηση

Προσομοιωμένη Ανόπτηση

Η προσομοιωμένη ανόπτηση επιλύει το δίλημμα ανάμεσα στην **εξερεύνηση** και την **εκμετάλλευση** με τον εξής τρόπο.

Σε κάθε επανάληψη, επιλέγεται μια **τυχαία** κίνηση. Αν βελτιώνει την κατάσταση τότε η κίνηση είναι αποδεκτή, διαφορετικά γίνεται αποδεκτή με κάποια πιθανότητα μικρότερη από 1.

Η πιθανότητα **μειώνεται εκθετικά** ως προς την **ακαταλληλότητα της κίνησης**.

Μειώνεται επίσης σύμφωνα με μία **παράμετρο θερμοκρασίας T** . Η προσομοιωμένη ανόπτηση ξεκινά με μια μεγάλη τιμή της T και στη συνέχεια η T μειώνεται σταδιακά. Σε μεγάλες τιμές της T , η προσομοιωμένη ανόπτηση μοιάζει με **καθαρή τυχαία αναζήτηση**. Κατά το τέλος του αλγόριθμου, όταν οι τιμές της T είναι αρκετά μικρές, η προσομοιωμένη ανόπτηση μοιάζει με τη συνηθισμένη **αναρρίχηση λόφων**.

Προσομοιωμένη Ανόπτηση

```

function SIMULATED-ANNEALING(problem, schedule)
returns a solution state
  inputs: problem, a problem
           schedule, a mapping from time to “temperature”
  local variables: current, a node  next, a node  T, the temperature
  current ← MAKE-NODE(RANDOM-STATE[problem])
  for t ← 1 to ∞ do
    T ← schedule[t]
    if T = 0 then return current
    next ← a randomly selected successor of current
    ΔE ← VALUE[next] − VALUE[current]
    if ΔE > 0 then current ← next
    else current ← next only with probability  $e^{\Delta E/T}$ 

```

Προσομοιωμένη Ανόπτηση

Η γραμμή κώδικα

current ← *next* only with probability $e^{\Delta E/T}$

υλοποιείται σαν

if $\text{random}[0, 1) < e^{\Delta E/T}$ **then** *current* ← *next*

Η παράσταση $e^{\Delta E/T}$ προέρχεται από την θεωρία της στατιστικής μηχανικής:

Η πιθανότητα να βρούμε ένα φυσικό σύστημα σε μια κατάσταση ενέργειας E είναι ανάλογη της συνάρτησης των Gibbs-Boltzmann $e^{-E/(kT)}$, όπου $T > 0$ είναι η θερμοκρασία και $k > 0$ είναι μια σταθερά.

Παράδειγμα

Ας υποθέσουμε ότι $\Delta E = \text{Value}[\text{next}] - \text{Value}[\text{current}] = -13$. Τότε:

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

Έτσι, σε μεγάλες τιμές του T , η προσομοιωμένη ανόπτηση συμπεριφέρεται σαν τυχαίος περίπατος: σε χαμηλές τιμές του T , συμπεριφέρεται σαν αναρρίχηση λόφων.

Προσομοιωμένη Ανόπτηση

Ο αλγόριθμος προσομοιωμένης ανόπτησης βρίσκει ένα **ολικό βέλτιστο (global optimum)** με πιθανότητα που τείνει στο 1 αν το χρονοδιάγραμμα (schedule) μειώνει τη θερμοκρασία T αρκετά αργά.

Το ακριβές όριο για την παράμετρο T και το χρονοδιάγραμμα για το T εξαρτάται συνήθως από το πρόβλημα. Γι' αυτό πρέπει να **πειραματιστούμε αρκετά** πάνω σε κάθε πρόβλημα που τίθεται για να δούμε αν η προσομοιωμένη ανόπτηση θα κάνει τη διαφορά.

Η προσομοιωμένη ανόπτηση είναι ένας **πολύ δημοφιλής αλγόριθμος** και έχει χρησιμοποιηθεί με επιτυχία για να λυθούν διάφορα ενδιαφέροντα προβλήματα βελτιστοποίησης (π.χ., προβλήματα σχεδίασης VLSI, προβλήματα χρονοπρογραμματισμού εργασιών κ.λ.π.)

Τοπική Ακτινική Αναζήτηση (Local Beam Search)

Ιδέα: Σ' ένα αλγόριθμο τοπικής αναζήτησης, γιατί να μην κρατάμε περισσότερες από μια καταστάσεις (π.χ., k) στη μνήμη;

Σε κάθε επανάληψη, παράγουμε όλες τις διάδοχες καταστάσεις των k καταστάσεων. Αν κάποια από αυτές είναι λύση, τότε σταματάμε. Διαφορετικά, επιλέγουμε τις k καλύτερες καταστάσεις από αυτές και η διαδικασία επαναλαμβάνεται.

Η **ποικιλία (diversity)** των διαδόχων καταστάσεων είναι σημαντική έτσι ώστε να μην κολλάμε σε ακατάλληλες περιοχές του χώρου αναζήτησης.

Αυτό μπορεί να επιτευχθεί με την **στοχαστική ακτινική αναζήτηση (stochastic beam search)**: επιλέγουμε k τυχαίες διάδοχες καταστάσεις, με την πιθανότητα επιλογής ενός διαδόχου να είναι μια αύξουσα συνάρτηση της τιμής του.

Γενετικοί Αλγόριθμοι

Ένας **γενετικός αλγόριθμος (genetic algorithm)** είναι μια παραλλαγή της στοχαστικής ακτινικής αναζήτησης στην οποία οι διάδοχες καταστάσεις παράγονται **συνδυάζοντας δύο καταστάσεις - προγόνους** (φυλετική αναπαραγωγή - sexual reproduction).

Έννοιες:

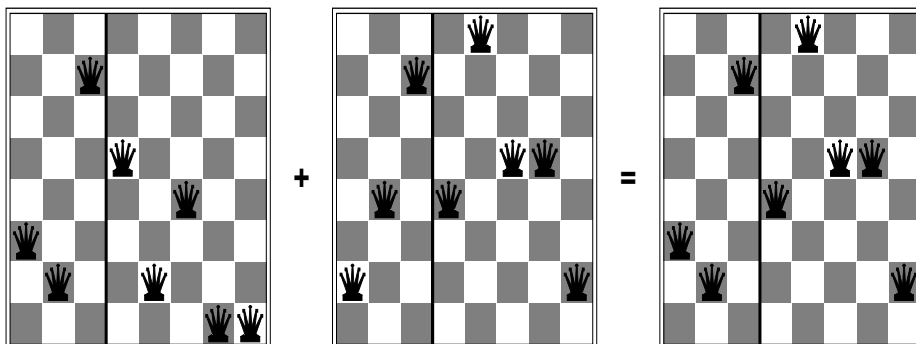
- Τα **άτομα (individuals)** αναπαριστούν καταστάσεις. Συμβολίζονται με συμβολοσειρές κάποιου αλφάβητου, συνήθως το $\{0, 1\}$.
- Οι **πληθυσμοί (populations)** είναι σύνολα ατόμων.
- Η **συνάρτηση καταλληλότητας (fitness function)** είναι μια συνάρτηση εκτίμησης για τη βαθμολόγηση κάθε ατόμου.

Γενετικοί Αλγόριθμοι

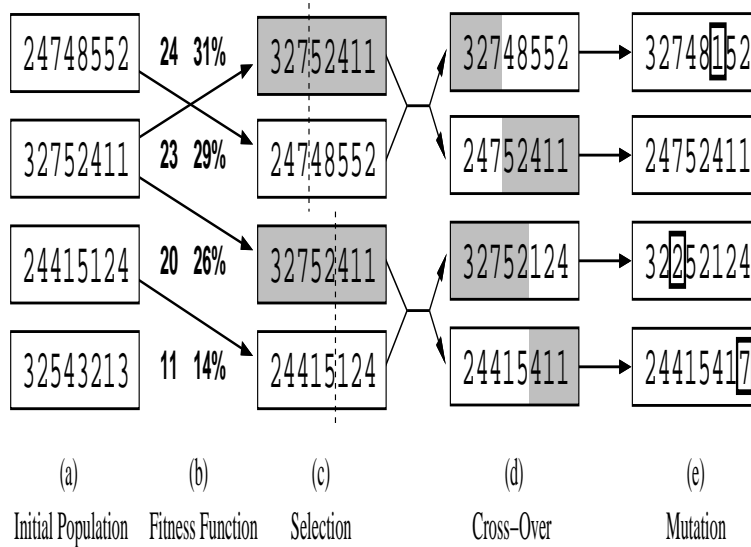
Επιτρεπόμενες Ενέργειες:

- **Αναπαραγωγή (reproduction):** ένα νέο άτομο παράγεται με συνδυασμό δύο γονέων.
- **Μετάλλαξη (mutation):** ένα νέο άτομο αλλάζει σε περιορισμένο βαθμό.

Παράδειγμα Αναπαραγωγής



Παράδειγμα: Το Πρόβλημα των 8 Βασιλισσών



Ένας Γενετικός Αλγόριθμος

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 $\leftarrow \emptyset$

loop for *i* **from** 1 **to** SIZE(*population*) **do**

x \leftarrow RANDOM-SELECTION(*population*, FITNESS-FN)

y \leftarrow RANDOM-SELECTION(*population*, FITNESS-FN)

child \leftarrow REPRODUCE(*x*, *y*)

if (small random probability) **then** *child* \leftarrow MUTATE(*child*)

add *child* to *new_population*

population \leftarrow *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

Διαισθητικά το πλεονέκτημα των γενετικών αλγορίθμων προέρχεται από την ικανότητα της **διασταύρωσης (crossover)** να συνδυάζει μεγάλα τμήματα από γράμματα που έχουν εξελιχθεί αυτόνομα προκειμένου να παράγει χρήσιμες λειτουργίες.

Υπάρχει ακόμα πολλή έρευνα που πρέπει να γίνει για να καταλάβουμε υπό ποιες συνθήκες οι γενετικοί αλγόριθμοι λειτουργούν πολύ καλά.

Οι γενετικοί αλγόριθμοι έχουν εφαρμοστεί με επιτυχία σε πολλά προβλήματα βελτιστοποίησης π.χ., το πρόβλημα του χρονοπρογραμματισμού εργασιών.

Μελέτη

- Κεφάλαιο 4, Ενότητα 4.3 του βιβλίου AIMA.
- (Προαιρετικά!) Τμήματα των Ενότητων 3 και 5 από το βιβλίο:
Z. Michalewicz and D. B. Fogel. *How to Solve it: Modern Heuristics*. Springer, 2000.