Αλγόριθμοι και Πολυπλοκότητα

Αντιμετώπιση NP-Πληρότητας

Δημήτρης Μιχαήλ

Τμήμα Πληροφορικής και Τηλεματικής
Χαροκόπειο Πανεπιστήμιο

Στρατηγικές Αντιμετώπισης NP-πληρότητας

Πάρα πολλά προβλήματα που προκύπτουν είναι NP-πλήρη. Τι κάνουμε λοιπόν όταν χρειαζόμαστε μια λύση ενός τέτοιου προβλήματος;

Τεχνικές.

1. ειδικές περιπτώσεις εισόδων και εύκολα στιγμιότυπα
2. προσεγγιστικοί αλγόριθμοι (approximation algorithms)
3. ευφυής εκθετική αναζήτηση
   - υπαναχώρηση (backtracking)
   - διακλάδωση και οριοθέτηση (branch and bound)
4. τοπική αναζήτηση (local search)

Ειδικές Περιπτώσεις Εισόδων

Πολλές φορές το πρόβλημα που έχουμε να λύσουμε είναι μια ειδική περίπτωση ενός προβλήματος που ξέρουμε πως είναι δύσκολο.

Είναι πιθανό αυτή η ειδική περίπτωση να λύνεται σε πολυωνυμικό χρόνο.

π.χ

- πολλά δύσκολα προβλήματα γραφημάτων λύνονται σε πολυωνυμικό χρόνο σε δέντρα
- το πρόβλημα 2-SAT ανήκει στο $\mathcal{P}$ ενώ το 3-SAT είναι NP-Complete.

2-SAT σε Πολυωνυμικό Χρόνο

Ένα στιγμιότυπο 2-SAT έχει την εξής μορφή:

$$(x \vee y) \wedge (x \vee \bar{y}) \wedge (y \vee \bar{z}) \wedge (z \vee \bar{x}).$$

Πρόκειται για ένα *λογικό τύπο* (Boolean formula) σε *συζευκτική κανονική μορφή* (conjunctive normal form - CNF) όπου κάθε όρος (clause) έχει ακριβώς δύο στοιχεία (literals).

2-SAT σε Πολυωνυμικό Χρόνο

Κοιτάμε έναν όρο της μορφής $(x \vee y)$ ως δύο κανόνες

- $\bar{x} \Rightarrow y$
- $\bar{y} \Rightarrow x$

Για να εκφράσουμε αυτούς τους κανόνες για όλο τον λογικό τύπο, φτιάχνουμε ένα γράφημα $G(V, E)$ όπου υπάρχουν

- για κάθε στοιχείο x δύο κόμβοι, για το στοιχείο x και την άρνηση του $\bar{x}$,
- για κάθε όρο της μορφής $(x \vee y)$ δύο ακμές $(\bar{x}, y)$ και $(\bar{y}, x)$.

2-SAT σε Πολυωνυμικό Χρόνο

Κοιτάμε έναν όρο της μορφής $(x \vee y)$ ως δύο κανόνες

- $\bar{x} \Rightarrow y$
- $\bar{y} \Rightarrow x$

Για να εκφράσουμε αυτούς τους κανόνες για όλο τον λογικό τύπο, φτιάχνουμε ένα γράφημα $G(V, E)$ όπου υπάρχουν

- για κάθε στοιχείο x δύο κόμβοι, για το στοιχείο x και την άρνηση του $\bar{x}$,
- για κάθε όρο της μορφής $(x \vee y)$ δύο ακμές $(\bar{x}, y)$ και $(\bar{y}, x)$.

Στο γράφημα που προκύπτει μπορούμε να αποφασίσουμε την ύπαρξη ικανοποιούσας ανάθεσης εαν υπάρχει διαδρομή που να συνδέει ένα στοιχείο x με την άρνηση του $\bar{x}$.

Προσεγγιστικοί Αλγόριθμοι

Προσεγγιστικός Αλγόριθμος. Είναι ένας αλγόριθμος ο οποίος τρέχει σε πολυωνυμικό χρόνο και βρίσκει λύσεις που είναι εγγυημένα κοντά στη βέλτιστη λύση.

Ακριβώς επειδή δεν αναζητούμε την βέλτιστη λύση, γίνεται εφικτό το να στοχεύσουμε σε ένα πολυωνυμικό χρόνο εκτέλεσης.

Δυσκολία. Για να αποδείξουμε μια εγγύηση προσέγγισης πρέπει να συγκρίνουμε τη λύση μας με μια βέλτιστη λύση που υπολογιστικά είναι πολύ δύσκολο να βρεθεί.

Ορισμός.
Έστω P ένα πρόβλημα ελαχιστοποίησης, και I ένα στιγμιότυπο του P. Έστω επίσης A ένα αλγόριθμος που βρίσκει μια εφικτή λύση σε στιγμιότυπα του P, $A(I)$ το κόστος της λύσης που βρίσκει ο A και $OPT(I)$ το κόστος της βέλτιστης λύσης για το στιγμιότυπο I.

Λέμε πως ο A είναι ένας α**-προσεγγιστικός** αλγόριθμος (α-approximation algorithm) για το πρόβλημα P αν,

$$\text{για κάθε στιγμιότυπο } I, \quad A(I) \leq \alpha \cdot OPT(I)$$

όπου $\alpha \geq 1$.

Αφού το P είναι πρόβλημα ελαχιστοποίησης, $A(I) \geq OPT(I)$.

Προσεγγιστικός Αλγόριθμος για Κάλυψη Κορυφών

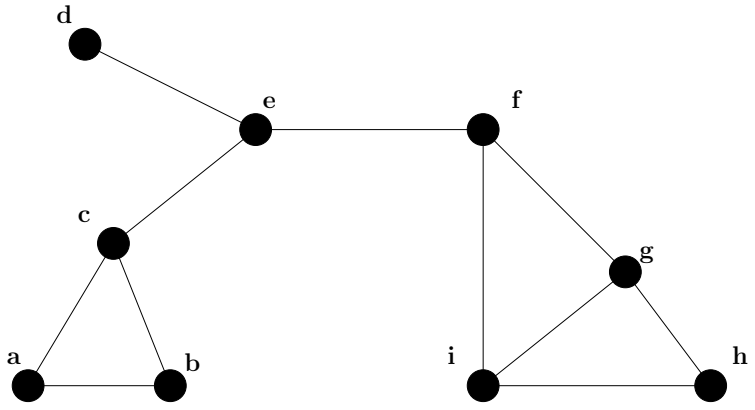
Κάλυψη Κορυφών. Δεδομένου ενός γραφήματος $G(V, E)$ βρείτε ένα ελάχιστο υποσύνολο κόμβων $S \subseteq V$ που να ”καλύπτει” όλες τις κορυφές.

Δύσκολο Πρόβλημα. Το πρόβλημα κάλυψης κορυφών είναι NP-δύσκολο.

Στόχος. Θέλουμε να σχεδιάσουμε έναν αλγόριθμο που να βρίσκει μια κάλυψη κορυφών η οποία για κάθε στιγμιότυπο του προβλήματος να μην απέχει πολύ από την βέλτιστη λύση.

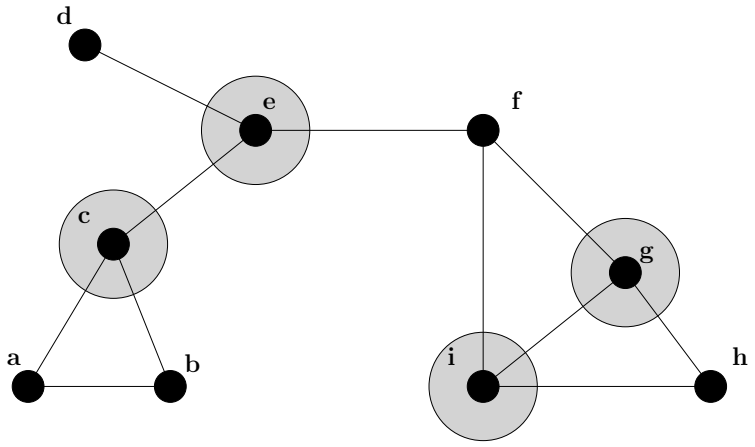
Κάλυψη Κορυφών

Κάλυψη Κορυφών. Δεδομένου ενός γραφήματος $G(V, E)$ βρείτε ένα ελάχιστο υποσύνολο κόμβων $S \subseteq V$ που να "καλύπτει" όλες τις κορυφές.



Κάλυψη Κορυφών

Κάλυψη Κορυφών. Δεδομένου ενός γραφήματος $G(V, E)$ βρείτε ένα ελάχιστο υποσύνολο κόμβων $S \subseteq V$ που να "καλύπτει" όλες τις κορυφές.



Προσεγγιστικός Αλγόριθμος για Κάλυψη Κορυφών

APPROX-VERTEX-COVER(G)

Input: Γράφημα $G(V, E)$
Output: Σύνολο $C \subseteq V$ που είναι μια Κάλυψη Κορυφών

```
1 C ← ∅
2 while E ≠ ∅ do
3 |   διάλεξε μια ακμή e = (u, v) ∈ E
4 |   C ← C ∪ {u, v}
5 |   σβήσε όλες τις προσκείμενες ακμές στο u και στο v
6 end
7 return C
```

Ο αλγόριθμος τρέχει σε χρόνο $\mathcal{O}(m + n)$.

Προσεγγιστικός Αλγόριθμος για Κάλυψη Κορυφών

Κάλυψη Κορυφών. Είναι εύκολο να δούμε πως ο αλγόριθμος επιστρέφει μια κάλυψη κορυφών.

Θεώρημα

Έστω OPT η βέλτιστη λύση για ένα στιγμιότυπο του προβλήματος. Ο αλγόριθμος επιστρέφει $|C| \leq 2 \cdot OPT$.

Προσεγγιστικός Αλγόριθμος για Κάλυψη Κορυφών

Κάλυψη Κορυφών. Είναι εύκολο να δούμε πως ο αλγόριθμος επιστρέφει μια κάλυψη κορυφών.

Θεώρημα

Έστω OPT η βέλτιστη λύση για ένα στιγμιότυπο του προβλήματος. Ο αλγόριθμος επιστρέφει $|C| \leq 2 \cdot OPT$.

Απόδειξη

Έστω M το σύνολο των ακμών που διαλέγονται στην γραμμή 3 του αλγορίθμου.

Οι ακμές του συνόλου M δεν έχουν κοινές κορυφές, λόγω της γραμμής 5 του αλγορίθμου. Άρα έχουμε $|OPT| \geq |M|$.

Ο αλγόριθμος μας υπολογίζει μια λύση όπου $|C| = 2 \cdot |M|$.

Καταλήγουμε πως $|C| = 2|M| \leq 2 \cdot |OPT|$. □

Η υπαναχώρηση (backtracking) βασίζεται στην παρατήρηση ότι συχνά είναι δυνατό να απορρίψουμε μια λύση εξετάζοντας απλώς ένα μικρό τμήμα της.

Η υπαναχώρηση (backtracking) βασίζεται στην παρατήρηση ότι συχνά είναι δυνατό να απορρίψουμε μια λύση εξετάζοντας απλώς ένα μικρό τμήμα της.

Εαν π.χ ένα στιγμιότυπο του SAT περιέχει τον όρο

$$(x_1 \vee x_2)$$

τότε όλες οι αναθέσεις με $x_1 = x_2 = 0$ (false) μπορούν αμέσως να απορριφθούν.

Η υπαναχώρηση (backtracking) βασίζεται στην παρατήρηση ότι συχνά είναι δυνατό να απορρίψουμε μια λύση εξετάζοντας απλώς ένα μικρό τμήμα της.

Εαν π.χ ένα στιγμιότυπο του SAT περιέχει τον όρο

$$(x_1 \vee x_2)$$

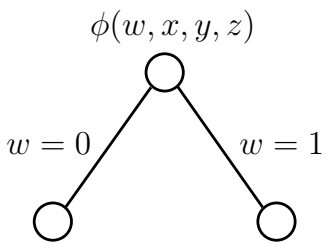
τότε όλες οι αναθέσεις με $x_1 = x_2 = 0$ (false) μπορούν αμέσως να απορριφθούν.

Με άλλα λόγια, με τον γρήγορο έλεγχο και την αμφισβήτηση αυτής της *μερικής* ανάθεσης μπορούμε να μειώσουμε τον χώρο αναζήτησης κατά το ένα τέταρτο.

Έστω ο λογικός τύπος $\phi(w, x, y, z)$ ο οποίος ορίζεται ως

$$(w \vee x \vee y \vee z) \wedge (w \vee \bar{x}) \wedge (x \vee \bar{y}) \wedge (y \vee \bar{z}) \wedge (z \vee \bar{w}) \wedge (\bar{w} \vee \bar{z}).$$

Θα φτιάξουμε σταδιακά ένα δέντρο μερικών λύσεων. Ξεκινάμε με διακλάδωση σε μία μεταβλητή, έστω την w:

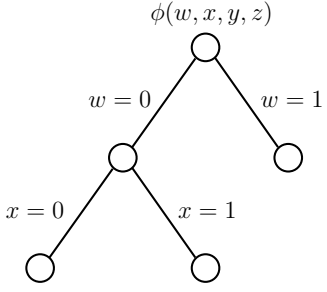


Έστω ο λογικός τύπος $\phi(w, x, y, z)$ ο οποίος ορίζεται ως

$$(w \vee x \vee y \vee z) \wedge (w \vee \bar{x}) \wedge (x \vee \bar{y}) \wedge (y \vee \bar{z}) \wedge (z \vee \bar{w}) \wedge (\bar{w} \vee \bar{z}).$$

Κανένας όρος δεν παραβιάζεται άμεσα και επομένως καμία από τις μερικές αναθέσεις δεν μπορεί να απαλειφθεί. Συνεχίζουμε λοιπόν την διακλάδωση.

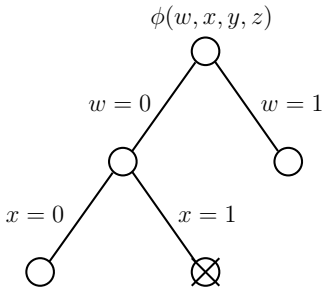
Μπορούμε να επεκτείνουμε έναν από τους δύο διαθέσιμους κόμβους, σε οποιαδήποτε μεταβλητή της επιλογής μας.



Έστω ο λογικός τύπος $\phi(w, x, y, z)$ ο οποίος ορίζεται ως

$$(w \vee x \vee y \vee z) \wedge (w \vee \bar{x}) \wedge (x \vee \bar{y}) \wedge (y \vee \bar{z}) \wedge (z \vee \bar{w}) \wedge (\bar{w} \vee \bar{z}).$$

Αυτή την φορά είμαστε τυχεροί. Η μερική ανάθεση $w = 0, x = 1$ παραβιάζει τον όρο $(w \vee \bar{x})$, και μπορεί να τερματιστεί, με συνέπεια την απαλοιφή ενός μεγάλου τμήματος του χώρου αναζήτησης.



Υπαναχωρούμε από αυτό το αδιέξοδο και συνεχίζουμε την εξερεύνηση μας στον έναν από τους δύο εναπομείναντες ενεργούς κόμβους.

Η υπαναχώρηση εξερευνά το χώρο των αναθέσεων, αναπτύσσοντας το δέντρο μόνο σε εκείνους τους κόμβους στους οποίους υπάρχει αβεβαιότητα όσον αφορά το αποτέλεσμα, και σταματά αν σε οποιοδήποτε στάδιο συναντήσει ικανοποιούσα ανάθεση.

Στην περίπτωση του SAT, κάθε κόμβος του δέντρου αναζήτησης μπορεί να περιγραφεί

- είτε με μια μερική ανάθεση
- είτε με τους όρους οι οποίοι απομένουν όταν αυτές οι τιμές τεθούν στον αρχικό τύπο.

π.χ αν $w = 0$ και $x = 0$ τότε

- ένας όρος με $\bar{w}$ ή $\bar{x}$ ικανοποιείται αμέσως και αφαιρείται
- ένα στοιχείο w ή x δεν ικανοποιείται και μπορεί να αφαιρεθεί

Έστω ο λογικός τύπος $\phi(w, x, y, z)$ ο οποίος ορίζεται ως

$$(w \vee x \vee y \vee z) \wedge (w \vee \bar{x}) \wedge (x \vee \bar{y}) \wedge (y \vee \bar{z}) \wedge (z \vee \bar{w}) \wedge (\bar{w} \vee \bar{z}).$$

Η μερική ανάθεση $w = 0$ και $x = 0$ αφήνει τον τύπο

$$(y \vee z) \wedge (\bar{y}) \wedge (y \vee \bar{z}).$$

Έστω ο λογικός τύπος $\phi(w, x, y, z)$ ο οποίος ορίζεται ως

$$(w \vee x \vee y \vee z) \wedge (w \vee \bar{x}) \wedge (x \vee \bar{y}) \wedge (y \vee \bar{z}) \wedge (z \vee \bar{w}) \wedge (\bar{w} \vee \bar{z}).$$

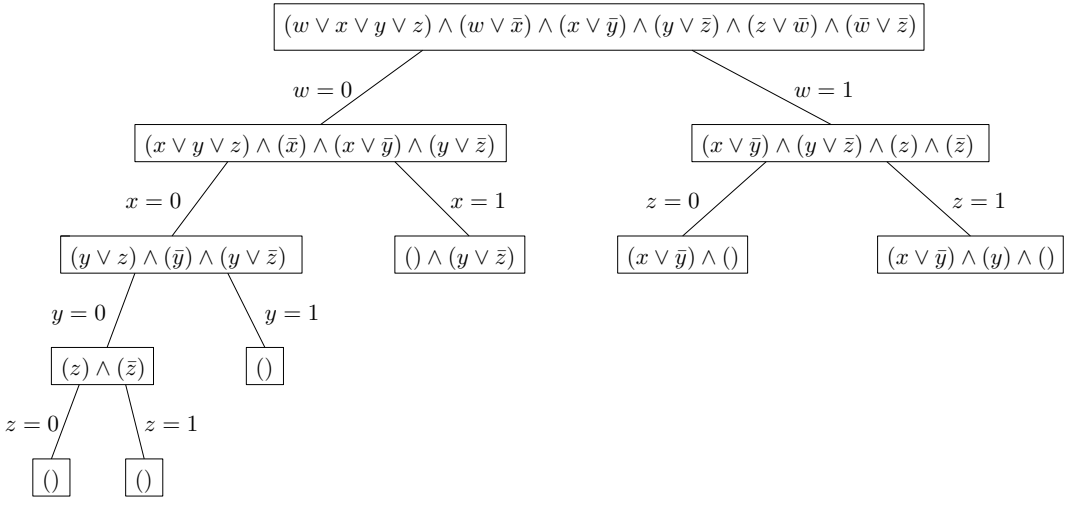
Η μερική ανάθεση $w = 0$ και $x = 1$ αφήνει τον τύπο

$$() \wedge (y \vee \bar{z}).$$

Η ύπαρξη ενός ”κενού όρου” $()$ αποκλείει την ικανοποιησιμότητα.

Ευφυής Εκθετική Αναζήτηση

Υπαναχώρηση (Backtracking)



Αυτή η εναλλακτική αναπαράσταση είναι χρήσιμη για την λήψη των δύο αποφάσεων που επανειλημμένα ανακύπτουν:

1. ποιο υποπρόβλημα θα επεκτείνουμε στην συνέχεια
2. ποια μεταβλητή διακλάδωσης θα χρησιμοποιήσουμε

Οι σωστές επιλογές σε αυτά τα ερωτήματα εξαρτώνται συνήθως από το πρόβλημα. Υπάρχουν βέβαια μερικές γενικές προτροπές.

Το όφελος της υπαναχώρησης προέρχεται από την ικανότητα να απαλείφει τμήματα και οι απαλειφές συμβαίνουν όταν συναντάμε κενό όρο:

- είναι λογικό να διαλέγουμε το υποπρόβλημα το οποίο περιέχει τον μικρότερο όρο
- και να επιλέγουμε μια μεταβλητή αυτού του όρου.

Σε περίπτωση που υπάρχει δυνατότητα επιλογής ισοδύναμων υποπροβλημάτων, μια εύλογη πολιτική είναι να επιλέξουμε αυτό που βρίσκεται χαμηλότερα στο δέντρο, ελπίζοντας πως είναι πιο κοντά σε μια ικανοποιούσα ανάθεση.

Ουσιαστικά για την υλοποίηση μιας γενικής μεθόδου backtracking χρειαζόμαστε μια συνάρτηση test η οποία έχει την εξής λειτουργία.

Εξετάζει ένα υποπρόβλημα και δηλώνει γρήγορα ένα από τρία αποτελέσματα:

1. αποτυχία: το υποπρόβλημα δεν έχει λύση
2. επιτυχία: βρέθηκε μια λύση του υποπροβλήματος
3. αβεβαιότητα

Backtracking

Έναρξη με πρόβλημα P_0
Έστω $S = \{P_0\}$ το σύνολο των ενεργών υποπροβλημάτων
while *S δεν είναι κενό* **do**
    choose – επιλογή ενός υποπροβλήματος $P \in S$ και αφαίρεση του από το S
    expand – επέκταση υποπροβλήματος σε μικρότερα $P_1, P_2, \ldots, P_k$
    for *κάθε* P_i **do**
        if *test*(P_i) *επιτυχές* **then**
            τέλος και ανακοίνωση λύσης
        end
        if *test*(P_i) *αποτύχει* **then**
            απόρριψη της P_i
        end
        προσθήκη του P_i στο S
    end
end

Γενικεύοντας την μέθοδο του backtracking σε προβλήματα βελτιστοποίησης (ελαχιστοποίηση ή μεγιστοποίηση) καταλήγουμε στην μέθοδο **branch-and-bound**.

Υποθέστε πως έχουμε ένα πρόβλημα ελαχιστοποίησης, π.χ το πρόβλημα της εύρεσης μιας ελάχιστης κάλυψης κορυφών σε ένα γράφημα.

Γενικεύοντας την μέθοδο του backtracking σε προβλήματα βελτιστοποίησης (ελαχιστοποίηση ή μεγιστοποίηση) καταλήγουμε στην μέθοδο **branch-and-bound**.

Υποθέστε πως έχουμε ένα πρόβλημα ελαχιστοποίησης, π.χ το πρόβλημα της εύρεσης μιας ελάχιστης κάλυψης κορυφών σε ένα γράφημα.

Όπως και προηγουμένως θα ασχοληθούμε με μερικές λύσεις, κάθε μια από τις οποίες αντιπροσωπεύει ένα υποπρόβλημα, και συγκεκριμένα ποιος είναι ο καλύτερος τρόπος (ή ποιο είναι το κόστος του καλύτερου τρόπου) για την ολοκλήρωση αυτής της λύσης.

Χρειαζόμαστε επίσης μια βάση για την απαλοιφή των μερικών λύσεων.

Για να απορρίψουμε ένα υποπρόβλημα, πρέπει να είμαστε σίγουροι ότι το κόστος του υπερβαίνει το κόστος κάποιας άλλης λύσης την οποία έχουμε ήδη συναντήσει.

Αλλά το ακριβές κόστος του μας είναι άγνωστο και γενικά δεν μπορεί να υπολογιστεί αποδοτικά. Έτσι χρησιμοποιούμε ένα γρήγορο **κάτω φράγμα** για το κόστος αυτό.

branch and bound (minimization)

Έναρξη με πρόβλημα P_0
Έστω $S = \{P_0\}$ το σύνολο των ενεργών υποπροβλημάτων
bestsofar = ∞
while *S δεν είναι κενό* **do**
    choose – επιλογή ενός υποπροβλήματος $P \in S$ και αφαίρεση του από το S
    expand – επέκταση υποπροβλήματος σε μικρότερα $P_1, P_2, \ldots, P_k$
    for *κάθε* P_i **do**
        if P_i *πλήρης λύση* **then**
            ενημέρωση bestsofar
        else
            if *lowerbound*(P_i) < *bestsofar* **then**
                προσθήκη του P_i στο S
            end
        end
    end
end

Έστω το πρόβλημα της κάλυψης κορυφών (vertex cover) σε ένα γράφημα $G(V, E)$. Μια μερική λύση είναι ένα σύνολο κόμβων $S \subseteq V$ που καλύπτει ένα μέρος του γραφήματος.

Σε κάθε βήμα του αλγορίθμου διακλάδωσης και οριοθέτησης, επεκτείνουμε μια μερική λύση S κατά έναν κόμβο $u \in V \setminus S$. Υπάρχουν $|V \setminus S|$ τρόποι να γίνει η επέκταση και άρα τόσα υποπροβλήματα της μορφής $S \cup \{u\}$.

Πως μπορούμε να φράξουμε από κάτω το κόστος της ολοκλήρωσης μιας μερικής λυσης $S \subseteq V$;

Κάτω φράγμα

Το μέγεθος ενός ταιριάσματος (matching) στο υπογράφημα που αποτελείται μόνο από τους κόμβους $V \setminus E$ (και τις μεταξύ ακμές τους).

Ένα ταίριασμα είναι ένα σύνολο ακμών M ώστε κάθε κόμβος του γραφήματος να ανήκει το πολύ σε μια ακμή. Μια οποιαδήποτε κάλυψη κορυφών πρέπει να καλύπτει τις ακμές του M και άρα πρέπει να περιέχει τουλάχιστον $|M|$ κόμβους.

Τοπική Αναζήτηση

Η τεχνική της τοπικής αναζήτησης (local search) εμπνέεται από την εξέλιξη, η οποία μέσω της αυξητικής της διεργασίας εισάγει μεταλλάξεις, τις δοκιμάζει, και τις διατηρεί αν λειτουργούν καλά.

Η τοπική αναζήτηση μπορεί να εφαρμοστεί σε οποιοδήποτε εγχείρημα βελτιστοποίησης.

Local Search

Έστω s οποιαδήποτε αρχική λύση
while *υπάρχει κάποια λύση* s' *στην γειτονιά της* s *με κόστος*$(s') < $ *κόστος*(s) **do**
  αντικατέστησε την s με s'
end

Τοπική Αναζήτηση

Σε ένα τυπικό πρόβλημα βελτιστοποίησης έχουμε ένα μεγάλο (κατά κανόνα εκθετικού μεγέθους) σύνολο δυνατών λύσεων C και μια συνάρτηση κόστους $c(\cdot)$ που μετρά την ποιότητα κάθε λύσης. Σκοπός είναι να βρούμε μια λύση η οποία ελαχιστοποιεί το κόστος.

Τώρα προσθέτουμε και την έννοια της *σχέσης γειτνίασης* (neighbor relation) των λύσεων, για να ανατυπώσουμε την ιδέα ότι μια λύση S' μπορεί να ληφθεί με μικρή τροποποίηση μιας άλλης λύσης S.

Μπορούμε να θεωρήσουμε πως η σχέση γειτνίασης ορίζει ένα (γενικά μη κατευθυνόμενο) γράφημα για το σύνολο των δυνατών λύσεων, με ακμές που συνδέουν γειτονικά ζευγάρια λύσεων. Ο αλγόριθμος τοπικής αναζήτησης διασχίζει αυτό το γράφημα, προσπαθώντας να κινηθεί προς μια καλή λύση.

Το πρόβλημα Δεδομένου ενός γραφήματος $G(V, E)$, λέμε ότι ένα σύνολο κόμβων $S \subseteq V$ αποτελεί **κάλυψη κορυφών** αν κάθε ακμή $e \in E$ έχει τουλάχιστον ένα άκρο στο S.

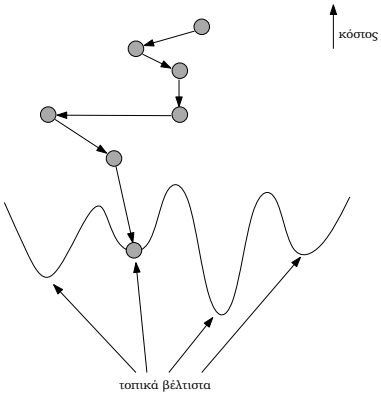
Σχέση Γειτνίασης Δεδομένου μιας κάλυψης κορυφών S μπορούμε να ορίσουμε μια απλή σχέση γειτνίασης ως εξής: μια κάλυψη κορυφών S' γειτνιάζει με την S εαν μπορεί να προκύψει από την S είτε με προσθήκη είτε με διαγραφή ενός κόμβου.

Τοπική Αναζήτηση για Κάλυψη Κορυφών σε Γράφημα $G(V, E)$

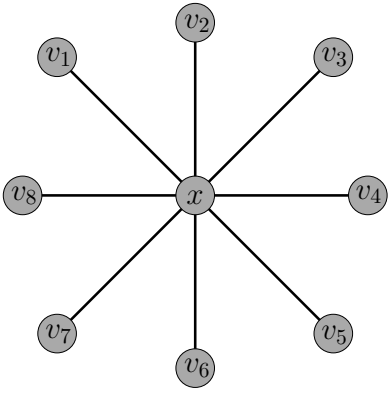
Έστω s η λύση με όλους τους κόμβους, $s = V$
while *υπάρχει εφικτή λύση* s' *με* $|s'| = |s| - 1$ **do**
  αντικατέστησε την s με s'
end

Το πρόβλημα μιας τέτοιας μεθόδου είναι πως μπορεί να παγιδευτεί σε τοπικά βέλτιστα.

Το πρόβλημα μιας τέτοιας μεθόδου είναι πως μπορεί να παγιδευτεί σε τοπικά βέλτιστα.



Έστω το παρακάτω στιγμιότυπο και η αρχική λύση $s = \{v_1, v_2, v_3, v_4, v_5, v_6, v_7, v_8, x\}$.

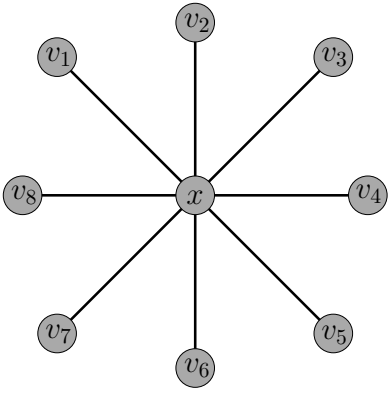


Η γειτονιά της s είναι οι λύσεις

- $s_1 = \{v_2, v_3, v_4, v_5, v_6, v_7, v_8, x\}$
- $s_2 = \{v_1, v_3, v_4, v_5, v_6, v_7, v_8, x\}$
- $s_3 = \{v_1, v_2, v_4, v_5, v_6, v_7, v_8, x\}$
- $s_4 = \{v_1, v_2, v_3, v_5, v_6, v_7, v_8, x\}$
- $s_5 = \{v_1, v_2, v_3, v_4, v_6, v_7, v_8, x\}$
- $s_6 = \{v_1, v_2, v_3, v_4, v_5, v_7, v_8, x\}$
- $s_7 = \{v_1, v_2, v_3, v_4, v_5, v_6, v_8, x\}$
- $s_8 = \{v_1, v_2, v_3, v_4, v_5, v_6, v_7, x\}$
- $s_x = \{v_1, v_2, v_3, v_4, v_5, v_6, v_7, v_8\}$

Όλες οι λύσεις είναι καλύτερες σε κόστος και άρα μπορεί να επιλεγεί οποιαδήποτε.

Έστω πως επιλέγεται η λύση $s_x = \{v_1, v_2, v_3, v_4, v_5, v_6, v_7, v_8\}$.



Η γειτονιά της s_x είναι μόνο η λύση

- $s = \{v_1, v_2, v_3, v_4, v_5, v_6, v_7, v_8, x\}$

Αλλά η λύση s έχει μεγαλύτερο κόστος από την s_x.

Ο αλγόριθμος έχει παγιδευτεί σε ένα τοπικό βέλτιστο.

Η τυχαιοκρατία (randomization) μπορεί να αποτελέσει ανεκτίμητο σύμμαχο στην τοπική αναζήτηση.

Χρησιμοποιείται συνήθως σε δύο σημεία:

1. για την επιλογή μια τυχαίας αρχικής λύσης
2. για την επιλογή μιας τοπικής κίνησης όταν υπάρχουν αρκετές διαθέσιμες

Η τυχαιοκρατία (randomization) μπορεί να αποτελέσει ανεκτίμητο σύμμαχο στην τοπική αναζήτηση.

Χρησιμοποιείται συνήθως σε δύο σημεία:

1. για την επιλογή μια τυχαίας αρχικής λύσης
2. για την επιλογή μιας τοπικής κίνησης όταν υπάρχουν αρκετές διαθέσιμες

Όταν υπάρχουν πολλά τοπικά βέλτιστα, η τυχαιοκρατία είναι ένας τρόπος για να διασφαλίσουμε ότι υπάρχει τουλάχιστον κάποια πιθανότητα να φθάσουμε στο σωστό. Επαναλαμβάνουμε την τοπική αναζήτηση αρκετές φορές, με διαφορετικό τυχαίο φύτρο (seed), και επιστρέφουμε την καλύτερη λύση.

Συνήθως καθώς αυξάνει το μέγεθος του προβλήματος, μεγαλώνει ο λόγος των κακών προς τα καλά τοπικά βέλτιστα, μέχρι του σημείου μερικές φορές να γίνει εκθετικά μεγάλος.

Σε αυτές τις περιπτώσεις η απλή επανάληψη της τοπικής αναζήτησης λίγες φορές δεν είναι αποτελεσματική λύση.

Μια διαφορετική πορεία προσέγγισης είναι να επιτρέπουμε περιστασιακά κινήσεις οι οποίες

- αυξάνουν το κόστος, με την ελπίδα πως θα απομακρύνουν την αναζήτηση από τα αδιέξοδα.

Η μέθοδος της προσομοιωμένης ανόπτησης ορίζει εκ νέου την τοπική αναζήτηση εισάγοντας την έννοια μιας θερμοκρασίας T.

Η μέθοδος της προσομοιωμένης ανόπτησης ορίζει εκ νέου την τοπική αναζήτηση εισάγοντας την έννοια μιας θερμοκρασίας T.

Local Search

Έστω s οποιαδήποτε αρχική λύση
repeat
  τυχαία επιλογή μιας λύσης s' στην γειτονιά της s
  if $\Delta = cost(s') - cost(s)$ *είναι αρνητικό* **then**
    αντικατέστησε την s με την s'
  else
    αντικατέστησε την s με την s' με πιθανότητα $\frac{1}{e^{\Delta/T}}$
  end
until *true*

- εαν $T = 0$ είναι η προηγούμενη τοπική αναζήτηση
- εαν T μεγάλο, περιστασιακά γίνονται αποδεκτές κινήσεις οι οποίες αυξάνουν το κόστος

Ποια είναι όμως η σωστή τιμή του T;

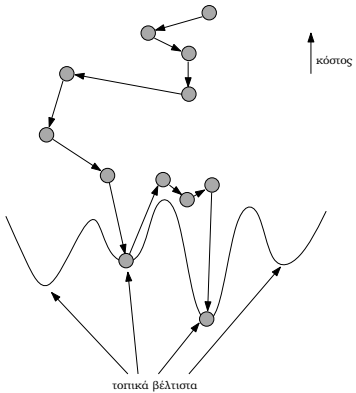
Το τέχνασμα είναι να ξεκινήσουμε με T μεγάλο και μετά να το μειώσουμε σταδιακά στο μηδέν.

- Αρχικά η τοπική αναζήτηση μπορεί να περιπλανιέται αρκετά ελεύθερα, με μικρή μόνο προτίμηση για λύσεις χαμηλού κόστους,
- καθώς περνά ο χρόνος αυτή η προτίμηση γίνεται ισχυρότερη και το σύστημα παραμένει κυρίως στην περιοχή χαμηλότερου κόστους του χώρου αναζήτησης, με περιστασιακές εξορμήσεις έξω από αυτή για να ξεφύγει από τα τοπικά βέλτιστα.

Τελικά, μόλις πέσει ακόμη παραπάνω η θερμοκρασία, το σύστημα συγκλίνει προς μια λύση.

Προσομοιωμένη Ανόπτηση

Simulated Annealing



Κόστος προσομειωμένης ανόπτησης:

- λόγω της μεταβολής της θερμοκρασίας και της αρχικής ελευθερίας κινήσεων, χρειάζονται πολύ περισσότερες τοπικές κινήσεις μέχρι να επιτευχθεί σύγκλιση,
- αποτελεί σχεδόν τέχνη, η επιλογή ενός καλού χρονοδιαγράμματος σύμφωνα με το οποίο θα μειώνεται η θερμοκρασία.
  Αυτό ονομάζεται *χρονοδιάγραμμα ανόπτησης* (annealing schedule).

Σε πολλες όμως περιπτώσεις, όπου η ποιότητα των λύσεων βελτιώνεται σημαντικά, αξίζει τον κόπο.