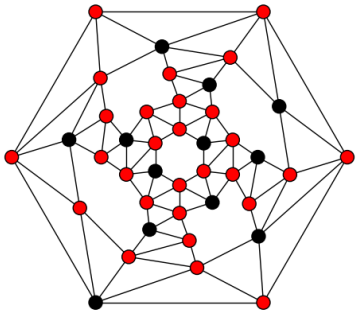
Πανεπιστήμιο Πατρών

Τμήμα Ηλεκτρολόγων Μηχανικών και Τεχνολογιάς Υπολογιστών

Τελική εργασία

Vertex Cover Problem



Δημήτριος Δήμου

8 Ιουλίου 2016

Περιεχόμενα

Κατάλογος σχημάτων

Κεφάλαιο 1

Set cover problems

Το set cover problem είναι ένα κλασικό πρόβλημα στον τομέα της συνδυαστικής βελτιστοποίησης και της θεωρίας υπολογιστών, η μελέτη του οποίου έχει οδηγήσει στην ανάπτυξη θεμελιωδών τεχνικών στο πεδίο των προσεγγιστικών αλγορίθμων. Λόγω της γενικής του διατύπωσης βρίσκει εφαρμογές σε μια ευρεία γκάμα προβλημάτων.

1.1 Διατύπωση

Το πρόβλημα μπορεί να διατυπωθεί με διάφορους τρόπους, είτε ως πρόβλημα βελτιστοποίησης όπου ζητείται ο ελάχιστος αριθμός υποσυνόλων ή, αν έχει ανατεθεί συνάρτηση κόστους στα υποσύνολα, ζητείται το σύνολο με το ελάχιστο κόστος, είτε ως πρόβλημα απόφασης.

Απλή διατύπωση

Δεδομένου ενός σύμπαντος U αποτελούμενο από n στοιχεία κι ενός συνόλου από υποσύνολα του U, $S = \{S_1, ..., S_k\}$ τέτοια ώστε η ένωσή τους να είναι το σύνολο U, βρες το ελάχιστο υποσύνολο του S που καλύπτει όλα τα στοιχεία του U.

Με συνάρτηση κόστους

Δεδομένου ενός σύμπαντος U αποτελούμενο από n στοιχεία, μιας συλλογής από υποσύνολα του U, $S = S_1, ..., S_k$, και μίας συνάρτησης κόστους $c : S \to \boldsymbol{Q}^+$, βρες το υποσύνολο του S με το ελάχιστο κόστος που καλύπτει όλα τα στοιχεία του U.

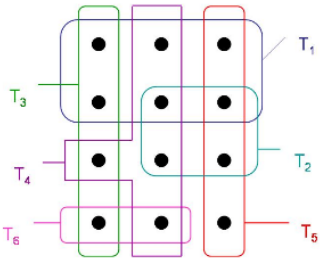
Πρόβλημα απόφασης

Δεδομένου ενός σύμπαντος U αποτελούμενο από n στοιχεία, μιας συλλογής από υποσύνολα του U, $S = S_1, ..., S_k$, και ενός ακεραίου k, αποφάσισε αν υπάρχει υποσύνολο του S με το πολύ k στοιχεία που καλύπτει όλα τα στοιχεία του U.

Παράδειγμα

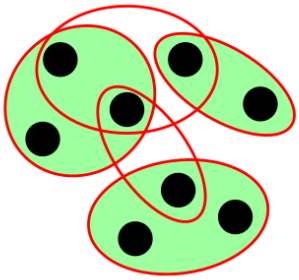
Έστω το σύμπαν $U = \{1, 2, 3, 4, 5\}$ και η συλλογή από υποσύνολα του $S = \{\{1, 2, 3\}, \{2, 4\}, \{3, 4\}, \{4, 5\}\}$. Η ένωση των στοιχείων του S καλύπτει το U. Η ελάχιστη συλλογή υποσυνόλων του S που καλύπτει το U είναι τα : $\{\{1, 2, 3\}, \{4, 5\}\}$.

Σχήμα 1.1: Set cover example



Το ελάχιστο set cover είναι το σύνολο $S' = \{T_3, T_4, T_5\}$

Σχήμα 1.2: Minimum set cover example



1.2 NP-πληρότητα

Το set cover decision problem είναι ένα από τα 21 NP-πλήρη προβλήματα του Karp που αποδείχθηκε ότι είναι NP-πλήρης το 1972. Αυτό σημαίνει ότι ανήκει στην κλάση NP δηλαδή δεδομένου ενός σύμπαντος U, μιας συλλογής S από υποσύνολα, ενός ακεραίου k και μίας λύσης S' η λύση αυτή μπορεί να επαληθευτεί σε πολυωνυμικό χρόνο όσον αφορά το μέγεθος των στοιχείων της εισόδου. Επίσης ανήκει και στην κλάση NP-hard. Αυτό οδήγησε στην ανάπτυξη προσεγγιστικών αλγορίθμων για την επίλυση του προβλήματος αυτού.

1.3 Λύσεις

1.3.1 Πρόβλημα ακέραιου προγραμματισμού

Το minimum set cover problem μπορεί να διατυπωθεί ως το ακόλουθο πρόβλημα ακέραιου προγραμματισμού

$$min\{\sum_{S \in \mathcal{S}} x_S\}$$

subject to

$$\sum_{S:e \in \mathcal{S}} x_S \geq 1, \quad \forall e \in \mathcal{U}$$

$$x_S \in \{0, 1\}$$

Όπου οι μεταβλητές x_S αντιπροσωπεύουν την ύπαρξή τους στο set cover σύνολο, το άθροισμά τους είναι ο αριθμός των συνόλων που χρησιμοποιούμε για να καλύψουμε το σύμπαν και είναι η συνάρτηση που θέλουμε να ελαχιστοποιήσουμε, ενώ ο περιορισμός δηλώνει ότι για κάθε στοιχείο $e \in U$ θέλουμε τουλάχιστον ένα σύνολο S που να το περιέχει. Επειδή το πρόβλημα ακέραιου προγραμματισμού είναι NP-hard χαλαρώνουμε τους περιορισμούς του προβλήματος για το x_S και το ανάγουμε σε πρόβλημα γραμμικού προγραμματισμού το οποίο λύνεται σε πολυωνυμικό χρόνο. Έτσι καταλήγουμε στο πρόβλημα:

$$min\{\sum_{S \in \mathcal{S}} x_S\}$$

subject to

$$\sum_{S:e \in \mathcal{S}} x_S \geq 1, \quad \forall e \in \mathcal{U}$$

$$x_S \in [0, 1]$$

Επειδή το integrality gap αυτού του προβλήματος είναι το πολύ $\log n$ η χαλάρωση του δίνει factor-$\log n$ προσεγγιστικό αλγόριθμο. Αν κάθε στοιχείο εμφανίζεται το πολύ σε $\mathcal{F}$ τότε μπορεί να βρεθεί λύση σε πολυωνυμικό χρόνο η οποία προσεγγίζει το βέλτιστο με παράγοντα $\mathcal{F}$ χρησιμοποιώντας το πρόβλημα γραμμικού προγραμματισμού.

1.3.2 Άπληστος αλγόριθμος

Υπάρχει και ένας άπληστος αλγόριθμος για προσέγγιση σε πολυωνυμικό χρόνο ο οποίος διαλέγει τα σύνολα με έναν μόνο κανόνα: σε κάθε στάδιο διαλέγει το σύνολο που περιέχει τον μεγαλύτερο αριθμό από ακάλυπτα στοιχεία.

Αλγόριθμος:

1. $C \leftarrow \emptyset$
2. While $C \neq \mathcal{U}$ do
   - (α΄) Find the set whose cost effectiveness is smallest, say S_i.
     Let $a = \frac{c(S_i)}{|S_i - C|}$.
     Pick S_i and $\forall e \in S_i - C$, set $price(e) = a$.
   - (β΄) $C \leftarrow S_i \cup C$
3. Output C

1.4 Hitting set

Το πρόβλημα set cover είναι ισοδύναμο με το πρόβλημα hitting set. Αν σε έναν διμερή γράφο το ένα σύνολο κόμβων $\mathcal{U}$ αντιπροσωπεύει τα υποσύνολα $\mathcal{S}$ του σύμπαντος, το άλλο σύνολο κόμβων $\mathcal{V}$ αντιπροσωπεύει τα στοιχεία του σύμπαντος και οι ακμές αντιπροσωπεύουν την συμπερίληψη ενός στοιχείου σε ένα σύνολο τότε βρίσκουμε τον ελάχιστο αριθμό κόμβων του συνόλου $\mathcal{U}$ που καλύπτει όλους τους κόμβους του συνόλου $\mathcal{V}$.

1.5 Εφαρμογές

Ένα παράδειγμα εφαρμογής του set cover problem σε πραγματικά προβλήματα είναι στην αποδοτική ανίχνευση κακόβουλων προγραμμάτων. Παραδείγματος χάρη αν θεωρήσουμε ότι το σύμπαν μας αποτελείτε από τους γνωστούς ιούς, και έχουμε μία συλλογή από κομμάτια κώδικα που εμφανίζονται σε κακόβουλο λογισμικό, τότε βρίσκοντας το set cover του συνόλου αρκεί να ψάξουμε μόνο για αυτά τα κομμάτια κώδικα για να πιστοποιήσουμε την ύπαρξη κακόβουλου λογισμικού. Ένα άλλο παράδειγμα, έστω ότι θέλουμε να αγοράσουμε ένα συγκεκριμένο αριθμό από προϊόντα και οι προμηθευτές τα προσφέρουν σε διάφορες προσφορές με διαφορετικούς συνδυασμούς υλικών (π.χ. Προμηθευτής 1: 2 τόνους ατσάλι + 500 πλακάκια για x €, Προμηθευτής 1: 3.5 τόνους ατσάλι + 200 πλακάκια για y €), τότε μπορούμε να χρησιμοποιήσουμε το set cover αυτών των συνόλων για να αγοράσουμε τα επιθυμητά υλικά ελαχιστοποιώντας παράλληλα το κόστος.

Κεφάλαιο 2

Vertex cover problem

Το vertex cover ενός γράφου είναι ένα σύνολο κόμβων τέτοιο ώστε κάθε ακμή του γράφου είναι προσκείμενη σε τουλάχιστον ένα κόμβο του συνόλου αυτού. Το πρόβλημα εύρεσης του ελάχιστου vertex cover είναι κλασικό πρόβλημα στον τομέα της συνδυαστικής βελτιστοποίησης και της θεωρίας υπολογιστών και κλασικό παράδειγμα NP-hard προβλήματος βελτιστοποίησης.

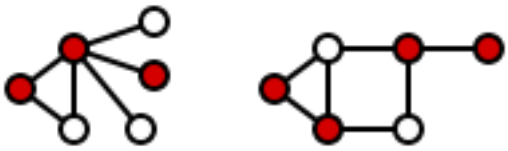
2.1 Διατύπωση

Το vertex cover V' ενός μη κατευθυντικού γράφου $G = (V, E)$ είναι ένα υποσύνολο του V τέτοιο ώστε:

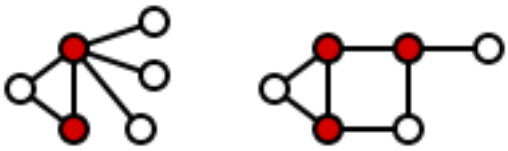
$$\forall uv \in E \Rightarrow u \in V' \vee v \in V'$$

Ένα τέτοιο σύνολο λέμε ό τι καλύπτει τις ακμές του G. Το ελάχιστο vertex cover ενός γράφου G είναι το σύνολο V' με τον μικρότερο αριθμό στοιχείων.

Σχήμα 2.1: Vertex cover



Σχήμα 2.2: Minimum vertex cover



2.2 NP-πληρότητα

Στη γενική περίπτωση το πρόβλημα vertex cover είναι NP-πλήρης οπότε είναι απίθανο να βρεθεί ακριβής αλγοριθμική λύση σε πολυωνυμικό χρόνο εκτός και αν P = NP. Η NP-πληρότητα μπορεί να αποδειχθεί με υπαγωγή από το 3-SAT πρόβλημα ή από το Clique πρόβλημα. Σε ειδικές περιπτώσεις γράφων όμως μπορούν να βρεθούν πολυωνυμικοί αλγόριθμοι. Με εξαντλητική αναζήτηση το πρόβλημα μπορεί να λυθεί σε $2^k n^{O(1)}$ χρόνο, όπου k το μέγεθος του συνόλου και n ο αριθμός των κόμβων, το οποίο κάνει το πρόβλημα fixed-parameter tractable.

Οπότε αν ενδιαφερόμαστε για μικρά k μπορούμε να επιλέξουμε αυτή τη μέθοδο και να έχουμε λύση σε πολυωνυμικό χρόνο.

2.3 Λύσεις

2.3.1 Πρόβλημα ακέραιου προγραμματισμού

Το minimum vertex cover problem μπορεί να διατυπωθεί ως το ακόλουθο πρόβλημα ακέραιου προγραμματισμού

$$min\{\sum_{u\in V} c(v)x_v\}$$

subject to

$$x_u + x_v \geq 1, \quad \forall (u,v) \in E$$

$$x_v \in \{0,1\} \quad \forall v \in V$$

όπου $c : V \to \boldsymbol{R}^+$ μια συνάρτηση κόστους για τους κόμβους του γράφου. Ο περιορισμός

$$x_v \in \{0,1\} \quad \forall v \in V$$

σημαίνει ότι ένας κόμβος v είτε ανήκει στο σύνολο V' είτε όχι, ενώ ο περιορισμός

$$x_u + x_v \geq 1, \quad \forall (u,v) \in E$$

σημαίνει ότι για κάθε ακμή τουλάχιστον ένας κόμβος της πρέπει να ανήκει στο σύνολο V' και η συνάρτηση που θέλουμε να ελαχιστοποιήσουμε είναι το άθροισμά των βαρών των κόμβων που βρίσκονται στο σύνολο V' δηλαδή τα v εκείνα για τα οποία το x_v είναι 1.

Χαλάρωση

Επειδή το πρόβλημα ακέραιου προγραμματισμού είναι NP-hard χαλαρώνουμε τους περιορισμούς του προβλήματος για το x_v και το ανάγουμε σε πρόβλημα γραμμικού προγραμματισμού το οποίο λύνεται σε πολυωνυμικό χρόνο. Έτσι καταλήγουμε στο πρόβλημα:

$$min\{\sum_{u\in V} c(v)x_v\}$$

subject to

$$x_u + x_v \geq 1, \quad \forall (u,v) \in E$$

$$x_v \in [0,1], \quad \forall v \in V$$

Integrality gap

Το integrality gap ενός προβλήματος γραμμικού προγραμματισμού που έχει προκύψει από τη χαλάρωση ενός προβλήματος ακέραιου προγραμματισμού ορίζεται το:

$$\sup_I \frac{OPT(I)}{OPT_f(I)}$$

όπου $OPT(I)$ είναι η βέλτιστη λύση του προβλήματος ακέραιου προγραμματισμού και $OPT_f(I)$ είναι η βέλτιστη λύση του προβλήματος γραμμικού προγραμματισμού.

Το integrality gap του παραπάνω προβλήματος είναι 2 οπότε η χαλάρωση του δίνει έναν factor-2 προσεγγιστικό αλγόριθμο. Επίσης η γραμμική χαλάρωση του προβλήματος είναι half-integral, δηλάδη υπάρχει βέλτιστη λύση στην οποία $x_v \in \{0, \frac{1}{2}, 1\}$

Half-integrality

Το παραπάνω πρόβλημα ακέραιου προγραμματισμού έχει αλλή μια πολύ ενδιαφέρουσα ιδιότητα: κάθε λύση ακραίου σημείου είναι half-integral δηλαδή $x_v \in \{0, \frac{1}{2}, 1\} \quad \forall x_v \in V'$[Vaz03].

Απόδειξη:
Έστω το σύνολο των κόμβων για το οποίο η λύση $\boldsymbol{x}$ δεν είναι half-integral, χωρίζουμε αυτό το σύνολο :

$$V_+ = \left\{ v \middle| \frac{1}{2} < x_v < 1 \right\}, \quad V_- = \left\{ v \middle| 0 < x_v < \frac{1}{2} \right\}$$

Για κάποιο $\varepsilon > 0$ ορίζουμε τις ακόλουθες λύσεις:

$$y_v = \begin{cases} x_v + \varepsilon, & x_v \in V_+ \\ x_v - \varepsilon, & x_v \in V_- \\ x_v, & \text{otherwise} \end{cases}$$

και

$$z_v = \begin{cases} x_v - \varepsilon, & x_v \in V_+ \\ x_v + \varepsilon, & x_v \in V_- \\ x_v, & \text{otherwise} \end{cases}$$

Έχουμε ότι $V_+ \cup V_- \neq \emptyset$ οπότε το $\boldsymbol{x}$ είναι διάφορο του $\boldsymbol{y}$ και $\boldsymbol{z}$, και για αρκετά μικρό ε έχουμε $0 \geq y, z \geq 1$. Επίσης ισχύει $x = \frac{1}{2}(y + z)$. Οπότε μένει να δείξουμε ότι οι $\boldsymbol{y}$, $\boldsymbol{z}$ είναι εφικτές λύσεις του προβλήματος.

Περίπτωση 1:

$$x_i + x_j > 1 \implies \begin{cases} (x_i + x_j) - (y_i + y_j) \leq 2\varepsilon \\ (x_i + x_j) - (z_i + z_j) \leq 2\varepsilon \end{cases}$$

Και για αρκετά μικρό ε μπορούμε να έχουμε

$$\begin{cases} y_i + y_j \geq 1 \\ z_i + z_j \geq 1 \end{cases}$$

Περίπτωση 2: $x_i + x_j = 1 \implies$

(α′) $x_i = 0, x_j = 1$
(β′) $x_i = 1, x_j = 0$
(γ′) $x_i = \frac{1}{2}, x_j = \frac{1}{2}$
(δ′) $x_i \in V_-, x_j \in V_+$ or $x_i \in V_+, x_j \in V_- \implies (x_i - \varepsilon) + (x_j + \varepsilon) = 1$

Άρα αφού η λύση $\boldsymbol{x}$ είναι κυρτός συνδυασμός δύο εφικτών λύσεων δεν είναι λύση ακραίου σημείου. Οπότε κάθε λύση ακραίου σημείου του χαλαρωμένου προβλήματος γραμμικού προγραμματισμού είναι half-integral. Τέλος μια προσεγγιστική λύση με παράγοντα 2 μπορεί να βρεθεί λύνοντας πρώτα το χαλαρωμένο πρόβλημα, του οποίου η λύση ξέρουμε ότι είναι half-integral, και έπειτα κατασκευάζουμε μια λύση $\boldsymbol{y}$ με τον ακόλουθο τρόπο:

$$\begin{cases} y_i = 1, \text{if} \quad x_i \in \{\frac{1}{2}, 1\} \\ y_i = 0, \text{otherwise} \end{cases}$$

2.3.2 Προσεγγιστικοί αλγόριθμοι

Έχουν αναπτυχθεί πολλές παραλλαγές προσεγγιστικών αλγορίθμων που λύνουν το συγκεκριμένο πρόβλημα. Ο πιο απλός αλγόριθμος είναι factor-2 προσεγγιστικός και αναπτύχθηκε ανεξάρτητα από τους Fanica Gavril και τον Μιχάλη Γιαννακάκη.
Η γενική ιδέα είναι η εξής: σε κάθε επανάληψη διαλέγει μια ακμή και εισάγει και τα δύο άκρα (u, v) της στο vertex cover V', και αφαιρεί από το σύνολο των ακμών κάθε ακμή που είναι προσκείμενη είτε στον κόμβο u είτε στον v μέχρι να μείνει το κενό σύνολο.
Αλγόριθμος:

1. $V' \leftarrow \emptyset$
2. $E' \leftarrow E$
3. While $E' \neq \emptyset$ do
   - α′) let (u, v) be an arbitrary edge of E'
   - β′) $V' \leftarrow V' \cup \{u, v\}$
   - γ′) remove from E' every edge incident on either u or v
4. Output V'

Ο αλγόριθμος αυτός τρέχει σε χρόνο $O(|V|+|E|)$[Cor+09]. Όσον αφορά τον παράγοντα προσέγγισης του αλγορίθμου φαίνεται εύκολα ότι για το σύνολο των ακμών που επιλέγονται στο βήμα α') ισχύει

$$|V^*| \geq |A|$$

αφού το σύνολο A δεν περιέχει προσκείμενες ακμές και επειδή το σύνολο V' που επιστρέφει ο αλγόριθμος περιέχει και τις δυο κορυφές των ακμών που επιλέγει έχουμε

$$|V'| = 2|A|$$

οπότε

$$|V'| \leq 2|V^*|$$

Έχουν αναπτυχθεί και άλλοι προσεγγιστικοί αλγόριθμοι με καλύτερο παράγοντα προσέγγισης, όπως $2 - \Theta\left(\frac{1}{\sqrt{\log|V|}}\right)$[Kar09] αλλά δεν έχει βρεθεί καλύτερος αλγόριθμος σταθερού προσεγγιστικού παράγοντα. Το minimum vertex cover πρόβλημα είναι $APX-$πλήρης δηλαδή δεν μπορεί να προσεγγιστεί αυθαίρετα καλά αν δεν ισχύει $P = NP$. Οι Dinur και Safra απέδειξαν ότι το πρόβλημα δε μπορεί να προσεγγιστεί με παράγοντα μικρότερο του 1.3606 για έναν αρκετά μεγάλο γράφο αν δεν ισχύει $P = NP$, επίσης αν ισχύει η εικασία unique games τότε το πρόβλημα δεν μπορεί να προσεγγιστεί με σταθερό πράγοντα μικρότερο του 2.

Κεφάλαιο 3

Ειδικές περιπτώσεις

Το πρόβλημα vertex cover παρόλο που είναι NP-πλήρης στη γενική περίπτωση σε κάποιες ειδικές περιπτώσεις γράφων είναι δυνατό να λυθεί σε πολυωνυμικό χρόνο. Παρακάτω παραθέτονται κάποιες από αυτές τις ειδικές περιπτώσεις και ορισμένες λύσεις και πορίσματα.

3.1 Bigraphs

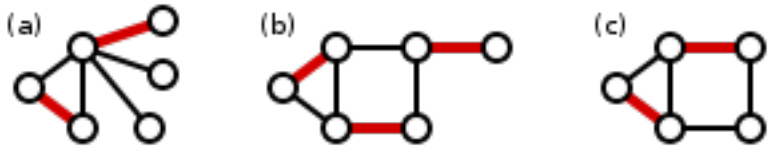
3.1.1 Εισαγωγικές έννοιες

1. Διμερής γράφοι:
   Οι διμερείς γράφοι είναι ειδικές περιπτώσεις γράφων που οι κόμβοι τους μπορούν να χωριστούν σε δύο ανεξάρτητα σύνολα $\mathcal{U}$ και $\mathcal{V}$ έτσι ώστε κάθε ακμή να ενώνει ένα κόμβο του $\mathcal{U}$ με έναν κόμβο του $\mathcal{V}$.

2. Matching:
   Δεδομένου ενός γράφου $G = (V, E)$ ονομάζουμε matching στον G ένα σύνολο μη γειτονικών ακμών, δηλαδή ακμές που δεν είναι προσκείμενες στον ίδιο κόμβο. Μέγιστο matching ονομάζουμε το σύνολο με τον μέγιστο αριθμό ακμών.

Σχήμα 3.1: Matching example

Σχήμα 3.2: Maximum matching example



3.1.2 Θεώρημα Konig

Σε αυτό το είδος γράφων βρίσκει εφαρμογή το θεώρημα του Konig το οποίο αναφέρει πως για κάθε διμερή γράφο $G = (V, E)$ ισχύει $\nu(G) = \tau(G)$ όπου
$\nu(G) :=$ maximum size of a matching in G,
$\tau(G) :=$ minimum size of a vertex cover in G.

3.1.3 Απόδειξη

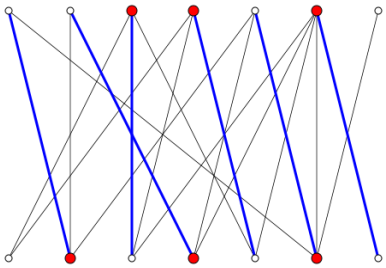
Έστω ένας διμερής γράφος $G = (L, R, E)$ και ένα μέγιστο ταίριασμα M. Τότε επειδή κανένας κόμβος ενός vertex cover δεν μπορεί να καλύπτει περισσότερες από δύο ακμές του συνόλου M (διαφορετικά δεν θα ήταν ταίριασμα), ένα vertex cover μεγέθους $|M|$ θα είναι το ελάχιστο vertex cover.
Για να δημιουργήσουμε ένα τέτοιο vertex cover, έστω U το σύνολο των μη ταιριασμένων κόμβων του L, και Z το σύνολο των ακμών που είτε είναι στο U είτε συνδέονται με αυτό μέσω εναλλακτικών μονοπατιών (alternating paths). Τότε το σύνολο $V' = (L \setminus Z) \cup (R \cap Z)$ είναι ένα vertex cover.

3.1.4 Συμπεράσματα

Οπότε χρησιμοποιώντας τον αλγόριθμο Hopcroft-Karp, ο οποίος βρίσκει ένα μέγιστο ταίριασμα σε ένα διμερή γράφο σε χρόνο $O(|E|\sqrt{|V|})$, μπορούμε έπειτα να υπολογίσουμε το σύνολο V' αποδοτικά. Επίσης για όλους τους γράφους ισχύει

$$\max_{\text{matching M}} |M| \leq \min_{\text{vertex cover U}} |U| \leq 2 \cdot \left(\max_{\text{mathing M}} |M| \right)$$

Σχήμα 3.3: Maximum matching - minimum vertex cover example



3.2 Tree graphs

Για δένδρα υπάρχει ένας άπληστος αλγόριθμος που βρίσκει το ελάχιστο vertex cover σε πολυωνυμικό χρόνο.

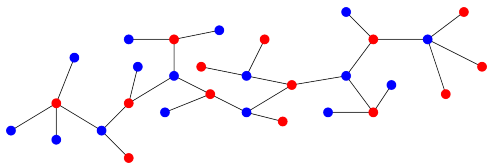
3.2.1 Εισαγωγικές έννοιες

Ένα δένδρο είναι ένας μη κατευθυντικός γράφος $G = (V, E)$ που είναι συνεκτικός και δεν έχει κύκλους.

3.2.2 Αλγόριθμος

Η βασική ιδέα του αλγορίθμου είναι η εξής: χρησιμοποιώντας την αναζήτηση πρώτα σε βάθος βρίσκουμε όλα τα φύλλα του δένδρου και έπειτα για κάθε φύλλο επιλέγουμε τον πατέρα του, έπειτα επιλέγουμε κάθε εσωτερικό κόμβο που τα παιδιά του δεν έχουν επιλεχθεί μέχρι να μην υπάρχουν άλλοι κόμβοι να επιλεχθούν.

Σχήμα 3.4: Vertex cover in tree example



3.3 Hypergraphs

3.3.1 Εισαγωγικές έννοιες

Υπεργράφος είναι μια γενίκευση του γράφου όπου μια ακμή μπορεί να συνδέσει περισσότερους από έναν κόμβους. Πιο αυστηρά ένας υπεργράφος H είναι ένα ζευγάρι $H = (X, E)$ όπου X είναι ένα σύνολο του οποίου τα στοιχεία είναι οι κόμβοι και E είναι ένα σύνολο αποτελούμενο από μη κενά υποσύνολα του X και ονομάζονται υπερακμές.

Το πρόβλημα εύρεσης του vertex cover ενός hypergraph ονομάζεται transversal, και είναι ισοδύναμο με το hitting set.

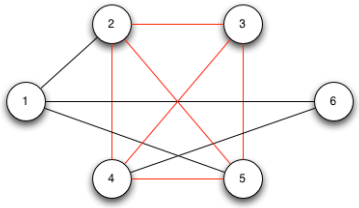
3.4 Δυικά προβλήματα

Το πρόβλημα vertex cover συνδέται και με άλλα συνδυαστικά προβλήματα σε γράφους μέσα από κάποιες δυικές σχέσεις. Κάποια από αυτά τα προβλήματα αναφέρονται παρακάτω.

3.4.1 Clique

Ένα clique ενός μη κατευθυντικού γράφου $G = (V, E)$ είναι ένα υποσύνολο των ακμών $C \subseteq V$ τέτοιο ώστε όλοι οι κόμβοι του να είναι γειτονικοί ανά δύο. Δεδομένου ενός μη κατευθυντικού γράφου $G = (V, E)$ ορίζουμε το συμπλήρωμα του ως $\overline{G} = (V, \overline{E})$ όπου $\overline{E} = \{(u, v) : u, v \in V, u \neq v, (u, v) \notin E\}$, τότε αν υπάρχει ένα clique C στον $\overline{G}$ το $V \setminus C$ είναι ένα vertex cover του G.

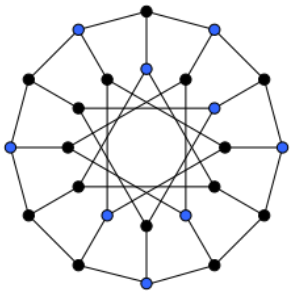
Σχήμα 3.5: Clique example



3.4.2 Independent set

Το independent set ενός γράφου είναι ένα σύνολο των κόμβων του οι οποίοι δεν είναι γειτονικοί ανά δύο. Το πρόβλημα εύρεσης του μέγιστου independent σετ είναι ένα κλασικό NP-hard πρόβλημα βελτιστοποίησης. Το πρόβλημα αυτό σχετίζεται με το πρόβλημα vertex cover καθώς για κάθε independent set ενός γράφου το συμπλήρωμα του είναι ένα vertex cover για αυτόν τον γράφο.

Σχήμα 3.6: Independent set example



Κεφάλαιο 4

Εφαρμογές

4.1 Παραδείγματα εφαρμογών

Το πρόβλημα εύρεσης του vertex cover ενός γράφου βρίσκει εφαρμογές σε διαφόρων ειδών προβλήματα τα οποία μπορεί εκ πρώτης όψεος να μην έχουν κάποιο κοινό γνώρισμα αλλά μπορούν να αναχθούν σε προβλήματα εύρεσης ενός ελαχίστου υποσύνολου το οποίο να "καλύπτει" ένα άλλο σύνολο.

Κάποια απλά παραδείγματα εφορμογής είναι τα εξής:

Έστω ότι θέλουμε να τοποθετήσουμε τροχονόμους στο οδικό δίκτυο μια πόλης, το πρόβλημα που προκύπτει είναι πώς μπορούμε να τοποθετήσουμε τους τροχονόμους με βέλτιστο τρόπο ώστε να χρησιμοποιήσουμε τον ελάχιστο αριθμό τροχονόμων που να "καλύπτουν" όλους τους δρόμους της πόλης. Αυτό το πρόβλημα μπορεί να μοντελοποιηθεί ως ένα minimum vertex cover πρόβλημα όπου οι ακμές είναι οι δρόμοι της πόλης και οι τροχονόμοι οι κόμβοι.

Άλλο παρόμοιο παράδειγμα: έστω ότι θέλουμε να τοποθετήσουμε κάμερες σε ένα κτήριο, πως μπορούμε να τοποθετήσουμε τις κάμερες με βέλτιστο τρόπο ώστε να έχουμε τον ελάχιστο αριθμό καμερών που να καλύπτουν όλους τους χώρους του κτηρίου; Και αυτό είναι ένα πρόβλημα που μπορεί να μοντελοποιηθεί ως minimum vertex cover πρόβλημα.

Ένα άλλο πρακτικό παράδειγμα είναι στην δυναμική ανίχνευση race conditions. Αν έχουμε ένα νήμα το οποίο γράφει σε μία θέσης μνήμης και έπειτα ένα άλλο νήμα προσπαθήσει να γράψει στην ίδια θέση μνήμης σημαίνει ότι κρατάει ένα lock για συγκεκριμένη θέση. Έτσι μπορούμε να ορίσουμε δύο σύνολα ένα το οποίο περιέχει τα νήματα και ένα που περιέχει τα locks για τις θέσεις μνήμης και ακμές που να αντιπροσωπεύουν την ιδιοκτησία κάποιου lock από ένα νήμα. Τότε το minimum hitting set αντιπροσωπεύει τον ελάχιστο σύνολο απο locks που είναι race-free. Το οποίο χρησιμοποιείται στη συνέχεια για την εξάλειψη περιττών εγγραφών.[OC03]

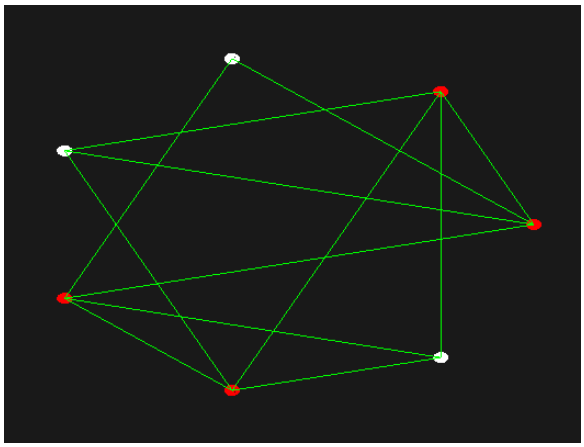
Τέλος ένα παράδειγμα από τον τομέα της υπολογιστικής βιοχημείας όπου σε πολλά προβλήματα χρειάζεται η εξάλειψη συγκρούσεων μεταξύ αλληλουχιών ενός δείγματος. Το πρόβλημα μοντελοποείται ως γράφος όπου οι κόμβοι αντιπροσωπεύουν τις αλληλουχίες του δείγματος και οι ακμές τις μεταξύ τους συγκρούσεις. Ο στόχος είναι να αφαιρεθούν όσον το δυνατόν λιγότεροι κόμβοι ώστε να μην υπάρχουν καθόλου συγκρούσεις στον γράφο.[Lan+02]

4.2 Υλοποίηση

```
std::vector<Node> Graph::vertexCover() {
  std::vector<Node> vertex_cover;
  std::vector<Edge> new_edge;

  new_edge = edges;

  while (new_edge.size() > 0) {
    Edge e = new_edge[0];
    vertex_cover.push_back(e.u);
    vertex_cover.push_back(e.v);
    remove_edge(new_edge, e.u);
    remove_edge(new_edge, e.v);

    std::cout << "Edge size " << new_edge.size() << "\n";
    std::cout << "Verteex c size " << vertex_cover.size() << "\n";
  }
  return vertex_cover;
}

void remove_edge(std::vector<Edge> &edges, Node u) {
  for (auto it = edges.begin(); it != edges.end();) {
    if ((*it).u == u | (*it).v == u) {
      edges.erase(it);
    } else {
      ++it;
    }
  }
}
```

Η παραπάνω συνάρτηση υλοποιεί τον προσεγγιστικό αλγόριθμο που παρουσιάστηκε στη παράγραφο 2.3.2. Η συνάρτηση καλείται σε ένα γράφο που αποτελείται από ένα σύνολο ακμών (edges) και ένα σύνολο κόμβων (vertices) και επιστρέφει ένα σύνολο κόμβων (vertex_cover) που αντιπροσωπεύει το vertex cover του γράφου. Η συνάρτηση remove_edge χρησιμοποιείται για να διαγράφει από ένα σύνολο ακμών όλες τις ακμές που είναι προσκύμενες σε ένα συγκεκριμένο κόμβο. Παρακάτω φαίνονται τα αποτελέσματα του αλγορίθμου, με κόκκινο συμβολίζονται οι κόμβοι που ανήκουν στο vertex cover του γράφου ενώ με άσπρο αυτοί που δεν ανήκουν.

Σχήμα 4.1: Vertex cover implementation example 1, 7 vertices



Με χρήση της βιβλιοθήκης Cbc που αποτελεί μέρος του COIN-OR project, και υλοποιεί αλγορίθμους για επίλυση προβλημάτων ακέραιου προγραμματισμού, μοντελοποιούμε το vertex cover problem σε πρόβλημα ακεραιού προγραμματισμού. Φορτώνουμε τον πίνακα γειτνίασης του παραπάνω γράφου.

Σχήμα 4.2: Adjacency matrix

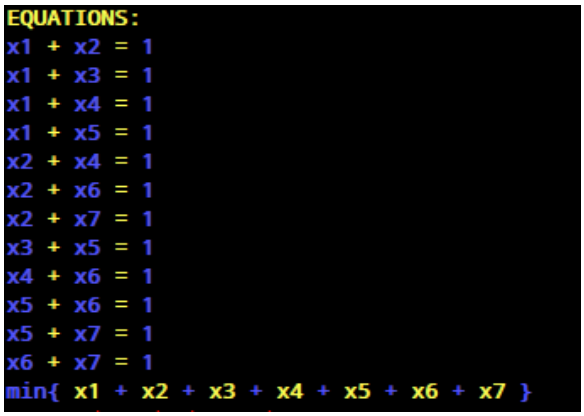
```
dimitris@DD:~/Documents/Libraries/Cbc-2.9.8/Cbc/examples$ ./MIPVertexCover adjacency7.adj
Started reading
Number of vertices 7
Finished reading adjacency file
Started making edges vector
Finished making edges vector
Nodes 7
Edges 12
ADJACENCY MATRIX:
      0     1     2     3     4     5     6
0     0     1     1     1     1     0     0
1     1     0     0     1     0     1     1
2     1     0     0     0     1     0     0
3     1     1     0     0     0     1     0
4     1     0     1     0     0     1     1
5     0     1     0     1     1     0     1
6     0     1     0     0     1     1     0
```

Και έπειτα τον χρησιμοποιούμε για να βρούμε τους περιορισμούς του προβλήματος.

Σχήμα 4.3: System Equations

```
SYSTEM EQUATION MATRIX:
E/N     x1    x2    x3    x4    x5    x6    x7
0      |1     1     0     0     0     0     0|                          | 1 |
1      |1     0     1     0     0     0     0|          | x1 |          | 1 |
2      |1     0     0     1     0     0     0|          | x2 |          | 1 |
3      |1     0     0     0     1     0     0|          | x3 |          | 1 |
4      |0     1     0     1     0     0     0|          | x4 |          | 1 |
5      |0     1     0     0     0     1     0|          | x5 |          | 1 |
6      |0     1     0     0     0     0     1|    *     | x6 |    =     | 1 |
7      |0     0     1     0     1     0     0|          | x7 |          | 1 |
8      |0     0     0     1     0     1     0|                          | 1 |
9      |0     0     0     0     1     1     0|                          | 1 |
10     |0     0     0     0     1     0     1|                          | 1 |
11     |0     0     0     0     0     1     1|                          | 1 |
```

Σχήμα 4.4: Solution



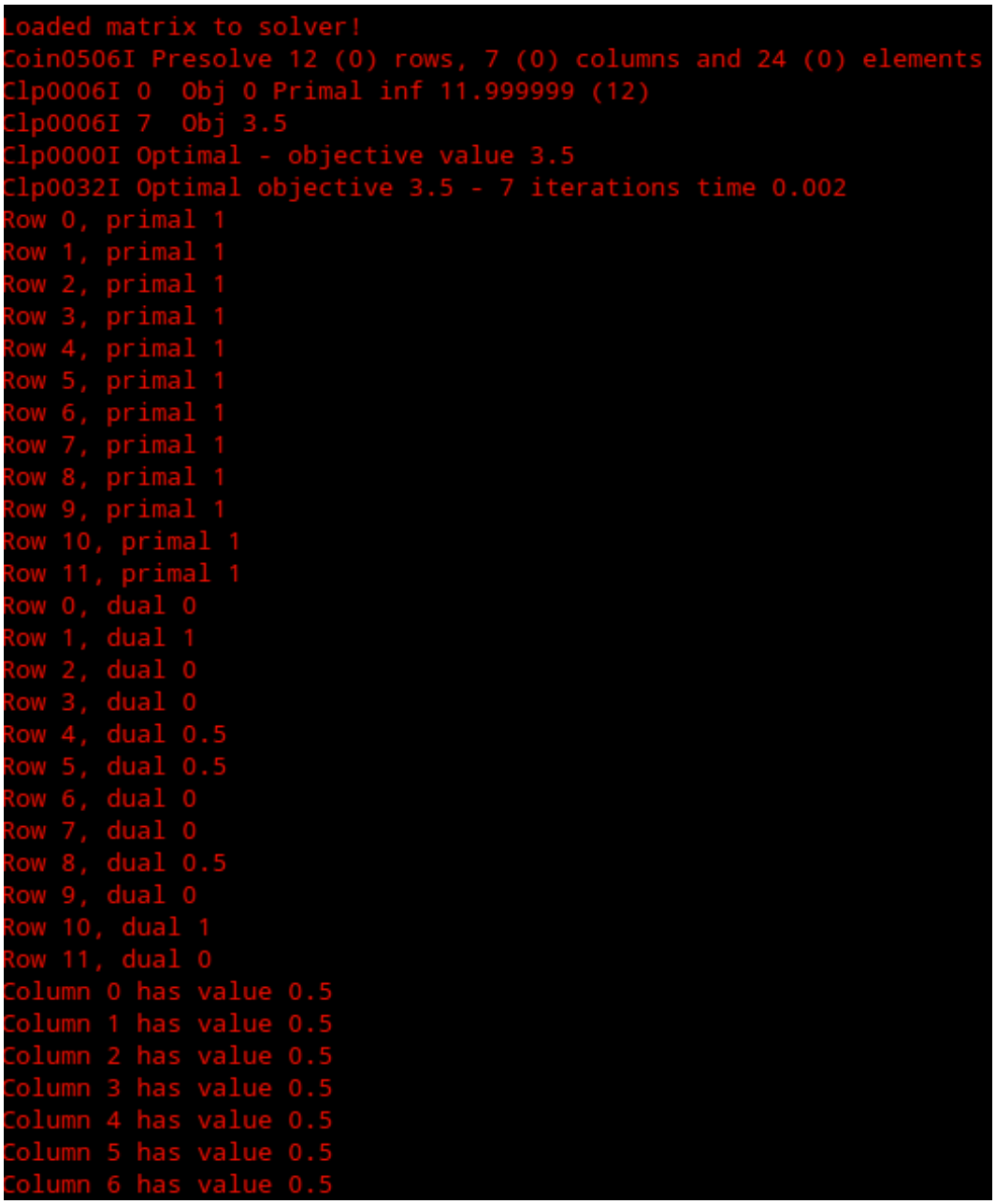
Και τέλος η λύση του προβλήματος δείχνει ότι επιλέγοντε 4 κόμβοι (αυτοί για τους οποίους η τιμή είναι 1) για το vertex cover όπως βρήκε και ο άπληστος αλγόριθμος.

Σχήμα 4.5: Vertex cover implementation example 1, 7 vertices

```
Started packed matrix!
numberRows 12
numberColumns 7
numberElements 24
Started solver!
Loaded matrix to solver!
Started initial solve!
Coin0506I Presolve 12 (0) rows, 7 (0) columns and 24 (0) elements
Clp0006I 0  Obj 0 Primal inf 11.999999 (12)
Clp0006I 7  Obj 3.5
Clp0000I Optimal - objective value 3.5
Clp0032I Optimal objective 3.5 - 7 iterations time 0.002
Clp0000I Optimal - objective value 3.5
Clp0000I Optimal - objective value 3.5
Cbc0046I Root node pass 1, 12 rows, 0 total tight cuts  -  objective 3.5
Clp0000I Optimal - objective value 5
Cbc0016I Integer solution of 5 found by strong branching after 0 iterations and 0 nodes (0.00 seconds)
Clp0006I 0  Obj 3.5 Primal inf 0.4999999 (1)
Clp0006I 1  Obj 4
Clp0000I Optimal - objective value 4
Clp0000I Optimal - objective value 4
Cbc0016I Integer solution of 4 found by strong branching after 1 iterations and 0 nodes (0.00 seconds)
Node 0 depth 0 unsatisfied 7 sum 3.5e+13 obj 4 guess 7.5 branching on -1
Cbc0001I Search completed - best objective 4, took 1 iterations and 0 nodes (0.00 seconds)
Cbc0032I Strong branching done 4 times (8 iterations), fathomed 1 nodes and fixed 0 variables
Cbc0035I Maximum depth 0, 0 variables fixed on reduced cost
Clp0000I Optimal - objective value 4
Clp0006I 0  Obj 3.5 Primal inf 3.4999993 (7)
Clp0006I 2  Obj 4
Clp0000I Optimal - objective value 4
i 0 has value 1
i 1 has value 1
i 4 has value 1
i 5 has value 1
```

Ενώ η χαλάρωση του προβλήματος και η λύση του με γραμμικό προγραμματισμό δίνει σε όλες τις μεταβλητές την τιμή 0.5, οπότε σύμφωνα με τα προηγούμενα θα πρέπει να επιλέξουμε όλους τους κόμβους στο vertex cover.

Σχήμα 4.6: Vertex cover implementation example 1, 7 vertices



Βιβλιογραφία

[Cor+09] Thomas H. Cormen κ.ά. *Introduction to Algorithms*. 3rd. MIT Press, 2009.

[Kar09] George Karakostas. ``A better approximation ratio for the vertex cover problem''. Στο: *ACM Transactions on Algorithms* (2009).

[Lan+02] G. Lancia κ.ά. ``SNPs Problems, Complexity and Algorithms''. Στο: *Lecture Notes in Computer Science* (2002).

[OC03] Robert O'Callahan και Jong-Deok Choi. ``Hybrid dynamic data race detection''. Στο: *Proceedings of the ninth ACM SIGPLAN symposium on Principles and practice of parallel programming*. 2003, σσ. 167–178.

[Vaz03] Vijay V. Vazirani. *Approximation Algorithms*. Springer, 2003.