1.1 Υπολογιστική Πολυπλοκότητα, Αναγωγές και NP-πληρότητα

Όπως είναι γνωστό, δεν επιδέχονται όλα τα προβλήματα αλγόριθμους που να τα λύνουν αποδοτικά, δηλαδή σε πολυωνυμικό χρόνο. Παρότι υπάρχουν πολλά προβλήματα για τα οποία γνωρίζουμε γρήγορους αλγόριθμους επίλυσής τους (όπως για παράδειγμα η εύρεση ενός ελάχιστου δέντρου ζεύξης, ενός συντομότερου μονοπατιού, ή της εκκεντρότητας σε ένα γράφημα) υπάρχουν επίσης πολλά προβλήματα για τα οποία δεν γνωρίζουμε κανέναν αποδοτικό αλγόριθμο που να τα επιλύει (όπως για παράδειγμα ο 3-χρωματισμός, η εύρεση κύκλου Hamilton και η εύρεση ανεξάρτητου συνόλου σε ένα γράφημα). Για τον λόγο αυτόν έχει αναπτυχθεί μια θεωρία, η Υπολογιστική Πολυπλοκότητα, η οποία μελετάει υπολογιστικά προβλήματα από την σκοπιά του χώρου και του χρόνου που χρειάζονται από έναν αλγόριθμο για να λυθούν. Τα διάφορα προβλήματα έχουν ταξινομηθεί σε διάφορες κλάσεις οι οποίες αντιπροσωπεύουν προβλήματα που έχουν κάποιο κοινό χαρακτηριστικό. Οι πιο συνηθισμένες κλάσεις είναι η $\mathcal{P}$ και η $\mathcal{NP}$. Ας ορίσουμε τι είναι οι κλάσεις $\mathcal{P}$ και $\mathcal{NP}$:

Ορισμός 1.1. Θα λέμε ότι ένα πρόβλημα ανήκει στην κλάση $\mathcal{P}$ όταν υπάρχει αλγόριθμος που λύνει το πρόβλημα αυτό σε χρόνο $O(n^c)$, όπου n το μέγεθος της εισόδου του προς επίλυση προβλήματος (η είσοδος αυτή ονομάζεται στιγμιότυπο) και $c > 0$.

Ορισμός 1.2. Θα λέμε ότι ένα πρόβλημα ανήκει στην κλάση $\mathcal{NP}$ όταν υπάρχει αλγόριθμος που επαληθεύει μια λύση για το πρόβλημα αυτό σε χρόνο $O(n^c)$.

Όσον αφορά τον ορισμό της κλάσης $\mathcal{NP}$, αυτός αφορά προβλήματα απόφασης και γίνεται και με Μηχανές Turing, αλλά δεν θα μπούμε σε τεχνικές λεπτομέρειες στις σημειώσεις αυτές. Για λεπτομέρειες στα ελληνικά δείτε τα [2, 4], ενώ για λεπτομέρειες στα αγγλικά δείτε το [3].

Ένα πρόβλημα απόφασης είναι ένα πρόβλημα με 2 δυνατές απαντήσεις: ναι ή οχι. Για παράδειγμα, το πρόβλημα ΚΥΚΛΟΣ HAMILTON ως πρόβλημα απόφασης είναι το εξής: Έχει ένα δοθέν γράφημα G κύκλο Hamilton; Το γράφημα G ονομάζεται στιγμιότυπο του προβλήματος.

Όσον αφορά τα στιγμιότυπα, αυτά χωρίζονται σε ναι-στιγμιότυπα και όχι-στιγμιότυπα. Για παράδειγμα, στο πρόβλημα ΚΥΚΛΟΣ HAMILTON, ένας κύκλος C_n είναι πάντα ένα ναι-στιγμιότυπο, ενώ ένα δέντρο είναι πάντα ένα όχι-στιγμιότυπο.

Στον ορισμό των κλάσεων $\mathcal{P}, \mathcal{NP}$ και γενικότερα των κλάσεων πολυπλοκότητας, αυτές αναφέρονται σε γλώσσες και όχι σε προβλήματα, οπότε ενώ θα αναφερόμαστε σε "προβλήματα", στην πραγματικότητα αναφερόμαστε σε "γλώσσες". Ξανά όμως δεν θα μπούμε σε τεχνικές λεπτομέρειες για να αποφευχθεί η χρήση ορολογίας από τις Μηχανές Turing.

Υπάρχει η πεποίθηση ότι $\mathcal{P} \neq \mathcal{NP}$ αλλά δεν έχει αποδειχθεί ακόμα, οπότε παραμένει μια εικασία. Ο λόγος που πιστεύουμε ότι $\mathcal{P} \neq \mathcal{NP}$ είναι ότι μετά από τόσα χρόνια έρευνας κανείς δεν έχει καταφέρει να βρει αποδοτικό αλγόριθμο για κανένα πρόβλημα της κλάσης $\mathcal{NP}$.

Έχοντας κάνει μια (πολύ) γρήγορη εισαγωγή στα βασικά της Υπολογιστικής Πολυπλοκότητας, ήρθε η ώρα να ορίσουμε τι σημαίνει ότι ένα πρόβλημα είναι $\mathcal{NP}$-δύσκολο και $\mathcal{NP}$-πλήρες.

Ορισμός 1.3. Θα λέμε ότι ένα πρόβλημα είναι $\mathcal{NP}$-πλήρες αν ανήκει στην κλάση $\mathcal{NP}$ και είναι και $\mathcal{NP}$-δύσκολο.

Τι είναι όμως $\mathcal{NP}$-δύσκολο;

Ορισμός 1.4. Θα λέμε ότι ένα πρόβλημα είναι $\mathcal{NP}$-δύσκολο αν οποιοδήποτε πρόβλημα που ανήκει στην κλάση $\mathcal{NP}$ μπορεί να αναχθεί σε αυτό.

Πώς μπορεί να αναχθεί;

Ορισμός 1.5. Έστω Α και Β δύο προβλήματα, και έστω f μία συνάρτηση αναγωγής. Θα λέμε ότι το Α ανάγεται στο Β, και θα συμβολίζουμε $A \leq_P B$, αν για κάθε στιγμιότυπο w ισχύει ότι: w ναι-στιγμιότυπο του Α $\iff$ $f(w)$ ναι-στιγμιότυπο του B.

Προσοχή ότι στον παραπάνω ορισμό, η αναγωγή γίνεται από ένα ήδη γνωστό $\mathcal{NP}$ πρόβλημα στο δικό μας, και όχι αντίστροφα. Το σύμβολο $\leq_P$ σημαίνει ότι η αναγωγή γίνεται σε πολυωνυμικό χρόνο. Προφανώς αναγωγές οι οποίες είναι εκθετικές σε χρόνο δεν μας προσφέρουν κάτι, αφού το ζητούμενο είναι να φτιάξουμε γρήγορους

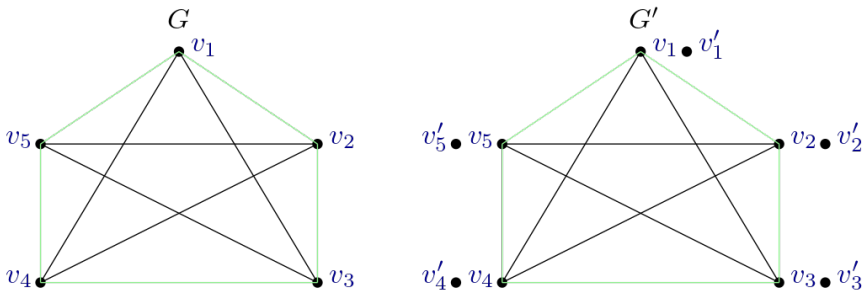
αλγόριθμους. Συνεπώς, από τον παραπάνω ορισμό, οι λύσεις του Α θα είναι λύσεις και του Β, ενώ αντίστοιχα οι μη λύσεις του Α δεν θα είναι λύσεις ούτε για το Β. Τα $\mathcal{NP}$-δύσκολα προβλήματα χαρακτηρίζονται και ως τα πιο αντιπροσωπευτικά προβλήματα για την κλάση $\mathcal{NP}$. Το 1972, ο Karp παρουσίασε στην εργασία του "Reducibility Among Combinatorial Problems, [5]" έναν κατάλογο από 21 προβλήματα συνδυαστικής, θεωρίας γραφημάτων, λογικής κ.ά. τα οποία είναι $\mathcal{NP}$-πλήρη. Ένα γνωστό $\mathcal{NP}$-πλήρες πρόβλημα είναι το παρακάτω:

Ορισμός 1.6. Στο πρόβλημα ΚΥΚΛΟΣ HAMILTON, το ζητούμενο είναι αν υπάρχει κύκλος σε ένα δοθέν γράφημα που να περνάει ακριβώς μία φορά από κάθε κορυφή του.

Αντίστοιχα, στο πρόβλημα ΜΙΣΟΣ ΚΥΚΛΟΣ HAMILTON το ζητούμενο είναι ένας κύκλος που περνάει από τις μισές κορυφές σε ένα δοθέν γράφημα.

Παράδειγμα 1.7. Με δεδομένο ότι το πρόβλημα ΚΥΚΛΟΣ HAMILTON είναι $\mathcal{NP}$-πλήρες, θα δειχθεί ότι το πρόβλημα ΜΙΣΟΣ ΚΥΚΛΟΣ HAMILTON είναι $\mathcal{NP}$-πλήρες.

Απόδειξη. Αρχικά, θα δείξουμε ότι το πρόβλημα ΜΙΣΟΣ ΚΥΚΛΟΣ HAMILTON ανήκει στην κλάση $\mathcal{NP}$. Χρειαζόμαστε ένα πιστοποιητικό ώστε να ελέγξουμε αν ένας δοθέν κύκλος που αποτελείται από $n/2$ κορυφές του G υπάρχει πράγματι στο G. Αυτό γίνεται σε χρόνο γραμμικό ως προς το G, δηλαδή $O(n+m)$. Αν μας δοθεί μια υποψήφια λύση $v_1, v_2, v_3, \ldots, v_{n/2}, v_1$ το μόνο που χρειάζεται να κάνουμε είναι να ελέγξουμε στο γράφημα G ότι η κορυφή v_1 συνδέεται με την v_2, η v_2 με την v_3 κ.ο.κ., και εν τέλει η $v_{n/2}$ με την v_1. Επίσης πρέπει να ελέγξουμε ότι στη λύση που μας δόθηκε καμία κορυφή (πλην της πρώτης) δεν υπάρχει πάνω από μία φορά. Αυτό που μένει είναι να κάνουμε αναγωγή το πρόβλημα ΚΥΚΛΟΣ HAMILTON στο πρόβλημα ΜΙΣΟΣ ΚΥΚΛΟΣ HAMILTON.
(Ευθύ, από ολόκληρο σε μισό κύκλο) Έστω γράφημα G με n κορυφές, το οποίο έχει κύκλο Hamilton. Αν τώρα, δημιουργήσουμε ένα γράφημα G' το οποίο είναι ίδιο με το G αλλά έχει ακόμα n απομονωμένες κορυφές, τότε στο νέο γράφημα G' θα έχουμε έναν κύκλο που περνάει από τις μισές κορυφές του.
(Αντίστροφο, από μισό σε ολόκληρο κύκλο) Αντίστροφα, αν από το G' διαγράψουμε τις απομονωμένες κορυφές θα έχουμε ένα γράφημα $G'' = G$ το οποίο έχει κύκλο που περνάει από τις μισές κορυφές του. □



Σχήμα 1.1: Παράδειγμα αναγωγής. Ο κύκλος $(v_1, v_2, v_3, v_4, v_5, v_1)$ είναι κύκλος Hamilton.

Δείξαμε λοιπόν ότι ΚΥΚΛΟΣ HAMILTON $\leq_P$ ΜΙΣΟΣ ΚΥΚΛΟΣ HAMILTON. Συνεπώς ξέρουμε ότι το πρόβλημα ΜΙΣΟΣ ΚΥΚΛΟΣ HAMILTON είναι, υπολογιστικά, τουλάχιστον τόσο δύσκολο όσο και το ΚΥΚΛΟΣ HAMILTON, οπότε δεν ελπίζουμε σε γρήγορο αλγόριθμο. Τι κάνουμε όμως για να λύσουμε τέτοιου είδους προβλήματα αποδοτικά; Θα πρέπει να αρκεστούμε σε μια προσεγγιστική λύση ή να ελπίζουμε το στιγμιότυπο για το προς επίλυση πρόβλημα να έχει χαρακτηριστικά που να μας επιτρέπουν να το λύσουμε σε πολυωνυμικό χρόνο.

Στο πρόβλημα ΚΑΛΥΜΜΑ ΑΠΟ ΚΟΡΥΦΕΣ, το οποίο αναφέρεται παρακάτω, ενώ γενικά είναι $\mathcal{NP}$-πλήρες, αυτό δεν ισχύει αν περιοριστούμε σε γραφήματα που είναι δέντρα.

Γενικότερα, στα διάφορα προβλήματα που καλείται να επιλύσει η Πληροφορική, υπάρχουν πάντα τρεις παράμετροι: **Απόδοση, χρόνος, αξιοπιστία**. Δυστυχώς μόνο δύο από τις παραπάνω παραμέτρους μπορούν να ισχύουν ταυτόχρονα, καθώς δεν γίνεται να έχουμε αλγόριθμους που να βρίσκουν βέλτιστη λύση, σε πολυωνυμικό χρόνο και για όλα τα στιγμιότυπα. Το παραπάνω θα πάψει να ισχύει, θεωρητικά τουλάχιστον, αν δειχθεί ότι $\mathcal{P} = \mathcal{NP}$ ([6]).

1.2 Παραμετρική Πολυπλοκότητα και η κλάση FPT

Καθώς η Θεωρία Υπολογισμού μας περιορίζει με το να κατατάσσει τα διάφορα προβλήματα σε εύκολα και δύσκολα, έχουν αναπτυχθεί διάφορες άλλες θεωρίες πολυπλοκότητας[1] (Υπολογιστική Πολυπλοκότητα, Λεπτομερής Πολυπλοκότητα [16, 17], Δυναμική Πολυπλοκότητα [15], Περιγραφική Πολυπλοκότητα [13], Κβαντική Πολυπλοκότητα [14] και άλλες) προκειμένου να ρίξουν φως στο σκοτάδι. Ωστόσο, το αντικείμενό τους, με εξαίρεση την Υπολογιστική Πολυπλοκότητα, είναι αρκετά τεχνικό. Στο παρόν κεφάλαιο θα γίνει μια εισαγωγή στην Παραμετρική Πολυπλοκότητα. Η θεωρία αυτή μελετάει τα προβλήματα ως προς την δυσκολία επίλυσης τους δεδομένης μιας ή περισσότερων παραμέτρων. Για παράδειγμα, ένα $\mathcal{NP}$-πλήρες πρόβλημα που αφορά γραφήματα μπορεί να λύνεται πιο εύκολα αν παραμετροποιηθεί από τον μέγιστο βαθμό του αντί για κάποιο άλλο δομικό του στοιχείο. Μια γνωστή κλάση της Παραμετρικής Πολυπλοκότητας είναι η εξής:

Ορισμός 1.8. Ένα πρόβλημα L με παράμετρο έναν θετικό ακέραιο k, καλείται παραμετρικά βατό (fixed parameter tractable, FPT) αν υπάρχει αλγόριθμος A, μια συνάρτηση f, και μια σταθερά $c > 0$ τέτοια ώστε δοθέντος ενός στιγμιοτύπου I για το L, ο αλγόριθμος A να αποφασίζει επιτυχώς αν $I \in L$ σε χρόνο τάξης $f(k)|I|^c$. Η κλάση πολυπλοκότητας που περιέχει όλους αυτούς τους αλγορίθμους ονομάζεται FPT.

Με άλλα λόγια, στην κλάση FPT ανήκουν όσοι αλγόριθμοι έχουν χρόνο εκτέλεσης $O(f(k)n^c)$, για κάποια σταθερά c. Προκειμένου η επίλυση των διάφορων προβλημάτων να γίνει λίγο γρηγορότερη, καθώς οι συναρτήσεις $f(k)$ στους FPT αλγόριθμους είναι εκθετικές, χρησιμοποιείται μια μέθοδος η οποία λέγεται πυρηνοποίηση. Πυρηνοποίηση είναι η διαδικασία η οποία μετασχηματίζει σε πολυωνυμικό χρόνο την είσοδο ενός προβλήματος σε μια απλούστερη είσοδο. Με τον τρόπο αυτό, το μετασχηματισμένο πρόβλημα θα μπορεί να λυθεί πιο γρήγορα από το αρχικό.

Δεν μπορούμε να φτιάξουμε πάντα FPT αλγόριθμους για τα διάφορα υπολογιστικά προβλήματα. Για παράδειγμα το πρόβλημα της k-Κλίκας (εύρεση κλίκας μεγέθους τουλάχιστον k) με παράμετρο το k, δεν έχει FPT αλγόριθμο. Έχει όμως αν για παράμετρο βάλουμε τον μέγιστο βαθμό Δ.

1.3 Πυρηνοποίηση

Ας ορίσουμε ένα ακόμα γνωστό $\mathcal{NP}$-πλήρες πρόβλημα.

Ορισμός 1.9. Στο πρόβλημα Σύνολο Αναδρασης Κορυφων, το ζητούμενο είναι, δεδομένου ενός (πολυ)γραφήματος[2] G και ενός ακεραίου k να βρούμε k το πολύ κορυφές η αφαίρεση των οποίων καθιστά το G άκυκλο.

Σε ένα πολυγράφημα επιτρέπονται πολλαπλές ακμές μεταξύ δύο κορυφών. Ας δούμε μερικούς κανόνες οι οποίοι μας βοηθάνε να απλοποιήσουμε το γράφημα G χωρίς να επηρεάσουμε τους κύκλους.

1. Διαγράφουμε όσες κορυφές έχουν βαθμό 0 ή 1. Προφανώς αυτές οι κορυφές δεν συνεισφέρουν σε κανέναν κύκλο.

2. Αν δύο κορυφές έχουν πολλαπλές ακμές μεταξύ τους, διαγράφουμε τόσες, ώστε να απομείνουν μόνο δύο. Ο λόγος που αφήνουμε δύο ακμές είναι ότι ενώ έχουμε κύκλο, δεν ξέρουμε ποια κορυφή να διαγράψουμε, καθώς αυτό εξαρτάται και από το υπόλοιπο γράφημα, όπως στο σχήμα 1.2.

3. Αν έχουμε κορυφή βαθμού ακριβώς 2, κάνουμε σύνθλιψη αυτής της κορυφής. Δηλαδή αν έχουμε τρεις κορυφές u, v, w και η v συνδέεται μόνο με την u και την w, διαγράφουμε την v και ενώνουμε την u με την w. Αν οι u, w είχαν ήδη ακμή μεταξύ τους, τώρα θα έχουν διπλή ακμή.

4. Αν μια κορυφή έχει βρόχο, τη διαγράφουμε και μειώνουμε την παράμετρο k κατά 1. Αυτό γίνεται γιατί κάθε κορυφή με βρόχο έχει κύκλο, οπότε πρέπει να διαγραφεί από το G.

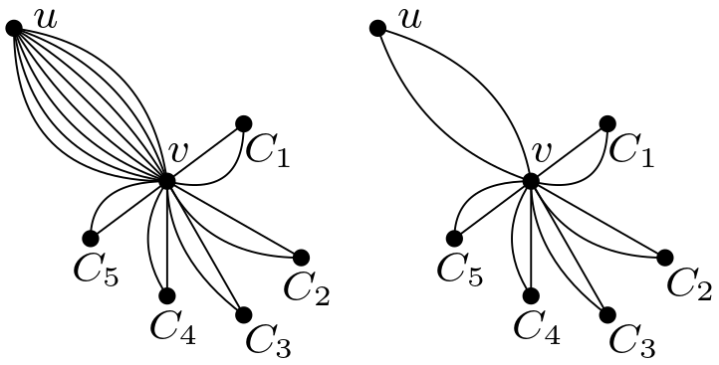
[1] Καθαρά για λόγους πληρότητας αναφέρεται και η υπολογισιμότητα/θεωρία αναδρομής, η οποία μελετάει τα προβλήματα που μπορούν να λυθούν από τα διάφορα θεωρητικά υπολογιστικά μοντέλα, π.χ. Μηχανές Turing με μαντεία [18].

[2] Επιτρέπονται οι βρόχοι και οι πολλαπλές ακμές μεταξύ δύο κορυφών.

5. Αν προκύψει ότι $k < 0$ τότε δεν έχουμε λύση. Σε αυτήν την περίπτωση θα πρέπει είτε να αλλάξουμε παράμετρο, είτε να μεγαλώσουμε το k.

6. Έστω Δ ο μέγιστος βαθμός κορυφής στο G. (Αποδεικνύεται ότι) αν $|V(G)| \geq (\Delta + 1)k$ ή $|E(G)| \geq 2\Delta k$ τότε και πάλι δεν έχουμε λύση.

Αν εφαρμόζοντας τους παραπάνω κανόνες μέχρις ότου να μην εφαρμόζεται κανένας τότε γνωρίζουμε ότι για το γράφημα G ισχύει ότι $|V(G)| < (\Delta+1)k$ και $|E(G)| < 2\Delta k$. Σε αυτήν την περίπτωση θα λέμε ότι έχουμε πυρήνα μεγέθους $O(\Delta k)$. Επιπλέον είναι προφανές ότι οι παραπάνω κανόνες εφαρμόζονται όλοι σε πολυωνυμικό χρόνο (γιατί;).



Σχήμα 1.2: Παράδειγμα κανόνα αναγωγής 2. Μετά και την αφαίρεση των πολλαπλών ακμών, πρέπει να αποφασίσουμε ποια από τις κορυφές u, v θα συμπεριληφθεί στη λύση. Η κορυφή v τέμνεται με άλλους 5 κύκλους, οπότε αυτή θα επιλεχθεί.

1.4 Επιπλέον τεχνικές

Εκτός από την τεχνική της πυρηνοποίησης, υπάρχουν και άλλες τεχνικές που μας βοηθάνε να λύσουμε τα διάφορα προβλήματα. Παρακάτω θα αναφερθούν κάποιες από αυτές και θα δοθεί η βασική τους ιδέα.

1.4.1 Φραγμένα δέντρα αναζήτησης (Bounded search trees)

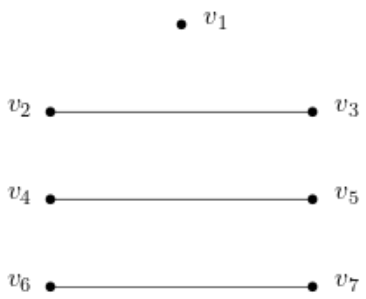
Είναι σύνηθες όταν έχουμε έναν αλγόριθμο επίλυσης ενός προβλήματος να μετράμε τον αριθμό των στοιχείων του δέντρου αναδρομής του ώστε να βρούμε την πολυπλοκότητα του αλγόριθμου αυτού. Η μέθοδος αυτή χρησιμοποιείται και στην Παραμετρική Πολυπλοκότητα. Ένα ακόμα γνωστό $\mathcal{NP}$-πλήρες πρόβλημα είναι το Καλυμμα Απο Κορυφες.

Ορισμός 1.10. Στο πρόβλημα Καλυμμα Απο Κορυφες, το ζητούμενο είναι, δεδομένου ενός γραφήματος G να βρούμε ένα $S \subseteq V(G)$ τέτοιο ώστε κάθε ακμή του G να συνδέεται με τουλάχιστον μία κορυφή του S.

Στην παραμετροποιημένη εκδοχή του προβλήματος, στην είσοδο έχουμε και έναν ακέραιο k και σκοπός είναι να ισχύει ότι $|S| \leq k$. Ένας απλός αναδρομικός αλγόριθμος είναι ο εξής: Για κάθε κορυφή $v \in V(G)$, είτε τοποθετούμε στο S την v και μειώνουμε την παράμετρο κατά 1, είτε όλους τους γείτονές της και μειώνουμε την παράμετρο κατά $|N(v)|$, όπου $N(v)$ είναι το σύνολο των γειτόνων της v.

Στην περίπτωση που έχουμε να κάνουμε με κορυφές βαθμού το πολύ 1, η λύση είναι τετριμμένη, όπως στο σχήμα 1.3. Συνεπώς θεωρούμε ότι όλες οι κορυφές έχουν βαθμό τουλάχιστον 2. Η κορυφή v_1 δεν ανήκει στη λύση αφού δεν έχει καμία προσπίπτουσα ακμή, ενώ για τις άλλες κορυφές του γραφήματος, χρειαζόμαστε ακριβώς μία από κάθε ακμή.

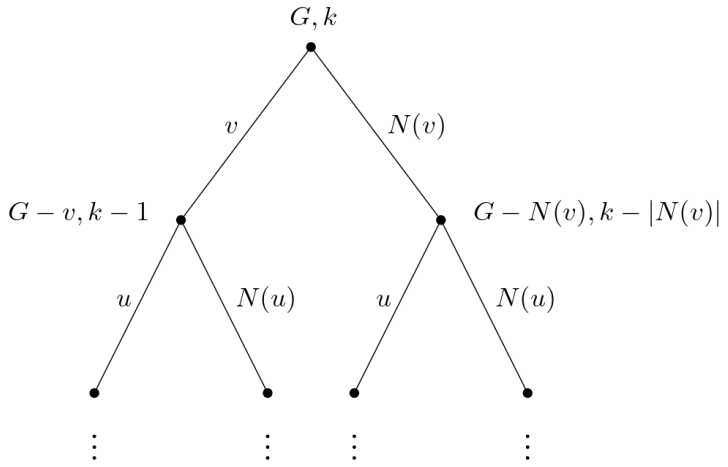
Ο χρόνος εκτέλεσης του παραπάνω αλγόριθμου είναι άνω φραγμένος από τον αριθμό των κορυφών στο δέντρο αναδρομής του επί τον χρόνο που αφιερώνει ο αλγόριθμος σε κάθε κορυφή. Το δέντρο αναζήτησης έχει την εξής μορφή: Ξεκινάει με μία ρίζα, και κάνει διακλάδωση. Στην μία περίπτωση ξανατρέχει ο ίδιος αλγόριθμος με



Σχήμα 1.3: Τετριμμένες περιπτώσεις για ΚΑΛΥΜΜΑ ΑΠΟ ΚΟΡΥΦΕΣ. Ένα κάλυμμα είναι το $\{v_2, v_4, v_6\}$, ενώ ένα δεύτερο είναι το $\{v_3, v_4, v_7\}$.

παράμετρο $k-1$ αντί k, ενώ στην δεύτερη ξανατρέχει ο ίδιος αλγόριθμος με παράμετρο το πολύ $k-2$ αντί k, βλ. σχήμα 1.4. Συνεπώς έχουμε μια αναδρομή της μορφής $T(k) \leq T(k-1)+T(k-2)$, για $k \geq 2$, αλλιώς $T(k) = 1$. Όσον αφορά την πρώτη περίπτωση, διαλέξτε την αγαπημένη σας μέθοδο από τα Διακριτά Μαθηματικά για να λύσετε την παραπάνω ανισότητα. Προκύπτει ότι $T(k) \leq 1.6181^k$, και άρα ο αλγόριθμος έχει χρόνο εκτέλεσης $O(1.6181^k n^c)$, είναι δηλαδή FPT.

Στην πραγματικότητα, αφού για γραφήματα με μέγιστο βαθμό 2 το πρόβλημα ΚΑΛΥΜΜΑ ΑΠΟ ΚΟΡΥΦΕΣ λύνεται σε πολυωνυμικό χρόνο, η αναδρομική σχέση είναι $T(k) \leq T(k-1) + T(k-3)$ αφού μας ενδιαφέρουν κορυφές βαθμού τουλάχιστον 3, η οποία μας δίνει $T(k) \leq 1.4656^k$.



Σχήμα 1.4: Παράδειγμα δέντρου αναδρομής για ΚΑΛΥΜΜΑ ΑΠΟ ΚΟΡΥΦΕΣ.

1.4.2 Επαναληπτική συμπίεση (Iterative compression)

Με την μέθοδο της επαναληπτικής συμπίεσης προσπαθούμε να κατασκευάσουμε έναν αλγόριθμο ο οποίος, επαναληπτικά, μειώνει το μέγεθος του στιγμιοτύπου του προς επίλυση προβλήματος.

Επιστρέφουμε ξανά στο ΚΑΛΥΜΜΑ ΑΠΟ ΚΟΡΥΦΕΣ. Έστω G το γράφημα για το οποίο ψάχνουμε ένα κάλυμμα απο κορυφές μεγέθους το πολύ k και έστω $v_1, v_2, \ldots, v_n$ μια διάταξη των κορυφών του. Ένας αλγόριθμος επαναληπτικής συμπίεσης, είναι ο εξής:

Έστω G_i το γράφημα που αποτελείται από τις πρώτες i κορυφές του G, για $0 \leq i \leq n$, και έστω X_i ένα κάλυμμα από κορυφές για το G_i. Παρατηρούμε ότι μέχρι και $i = k$ μπορούμε να λάβουμε υπόψιν όλες τις κορυφές του εκάστοτε G_i για το κάλυμμά μας. Έστω τώρα ότι για το G_{i+1}, έχουμε ότι $Z_{i+1} = X_i \cup v_{i+1}$ ένα υποψήφιο κάλυμμα. Αν $|Z_{i+1}| \leq k$ (αυτό για να ισχύσει πρέπει στα προηγούμενα βήματα του αλγόριθμου να κάνουμε μια πιο έξυπνη επιλογή του καλύμματος από κορυφές από αυτό που αναφέρθηκε) θέτουμε $X_{i+1} = Z_{i+1}$ και συνεχίζουμε για το επόμενο i. Αν $|Z_{i+1}| > k$, τότε ελέγχουμε για όλα τα $2^{|Z_{i+1}|} \leq 2^{k+1}$ υποσύνολα, αν κάποιο από αυτά είναι κάλυμμα από κορυφές μεγέθους το πολύ k. Αν δεν βρούμε κανένα τέτοιο υποσύνολο του Z_{i+1} τότε δεν έχουμε λύση. Αν βρούμε, συνεχίζουμε με το επόμενο i.

Ο παραπάνω αλγόριθμος δίνει την γενική ιδέα για το πως μπορούμε να μειώσουμε το μέγεθος μιας δοθείσας λύσης, αν αυτό είναι εφικτό.

Σε αρκετές περιπτώσεις η μέθοδος της επαναληπτικής συμπίεσης κάνει χρήση προσεγγιστικών αλγορίθμων για να δουλέψει. Το να κατασκευάσει κάποιος έναν FPT αλγόριθμο για ένα πρόβλημα μπορεί να είναι δύσκολο ή/και χρονοβόρο, ενώ το να συμπιεστεί μια ήδη υπάρχουσα λύση είναι αισθητά ευκολότερο. Στο παράδειγμά μας, το κάλυμμα από κορυφές έχει έναν 2-προσεγγιστικό αλγόριθμο (έναν αλγόριθμο ο οποίος επιστρέφει ένα κάλυμμα κορυφών με μέγεθος το πολύ διπλάσιο του βέλτιστου). Μπορούμε λοιπόν να χρησιμοποιήσουμε την λύση του 2-προσεγγιστικού αλγόριθμου ώστε με επαναληπτική συμπίεση να βρούμε, αν υπάρχει, ένα κάλυμμα από κορυφές μεγέθους του πολύ k. Για περισσότερα σε προσεγγιστικούς αλγορίθμους, μια καλή πηγή είναι το βιβλίο [6], διαθέσιμο στον σύνδεσμο αυτόν.

1.4.3 Δυναμικός Προγραμματισμός (Dynamic Programming)

Όπως και στους κλασικούς αλγόριθμους, έτσι και στους παραμετρικούς χρησιμοποιείται ο δυναμικός προγραμματισμός. Ο δυναμικός προγραμματισμός αντί να κοιτάει 'κοντόφθαλμα' για να βρει μια βέλτιστη λύση (όπως κάνουν οι άπληστοι αλγόριθμοι δηλαδή) σχεδιάζει αναδρομικά μια βέλτιστη λύση. Ο λόγος που αντιπαρατίθεται η έννοια του δυναμικού προγραμματισμού με την έννοια του άπληστου αλγόριθμου είναι ότι σε πολλά προβλήματα ένας προγραμματιστής καλείται να σκεφτεί αν το προς λύση πρόβλημα μπορεί να λυθεί με άπληστο αλγόριθμο ή με δυναμικό προγραμματισμό. Η απάντηση δεν είναι πάντα προφανής.

1.4.4 Πιθανοτικές μέθοδοι (Randomized methods)

Όπως πάντα, υπάρχουν αλγόριθμοι που χρησιμοποιούν εργαλεία από τις πιθανότητες προκειμένου να λύσουν ένα πρόβλημα. Υπάρχουν δύο είδη πιθανοτικών αλγορίθμων, οι Monte Carlo και οι Las Vegas. Οι Monte Carlo αλγόριθμοι είναι γρήγοροι αλλά μπορεί να δώσουν λάθος απάντηση (με μικρή πιθανότητα), ενώ οι Las Vegas δίνουν πάντα σωστή απάντηση αλλά δεν έχουμε κάποιο φράγμα για τον χρόνο εκτέλεσής τους.

Ειδικά πλέον που ο όρος μεγάλα δεδομένα (big data) τείνει να γίνει καθημερινότητα, η χρήση πιθανοτικών μεθόδων γίνεται επιτακτική (κυρίως για να διαβάζουμε ένα μέρος των δεδομένων αντί για όλα, αλλά να έχουμε επαρκή στοιχεία για την κατανομή τους και άλλα παρόμοια χαρακτηριστικά) στην κατασκευή αλγορίθμων. Μία κατηγορία τέτοιων αλγορίθμων είναι οι υπογραμμικοί αλγόριθμοι. Στον σύνδεσμο αυτόν περισσότερες λεπτομέρειες.

Παρακάτω θα δούμε έναν αλγόριθμο για το πρόβλημα Πολυχρωμο Μονοπατι.

Ορισμός 1.11. Ονομάζουμε k-χρωματισμό μια απεικόνιση $\chi : V \to [k]$, μια απεικόνιση που αντιστοιχίζει σε κάθε κορυφή του G ένα από τα k χρώματα. Η χ θέλουμε να είναι επί, να χρησιμοποιεί δηλαδή και τα k χρώματα για τον χρωματισμό των κορυφών του G. Δείτε και το κεφάλαιο 12 εδώ, (ή [10], αν ο σύνδεσμος δεν δουλεύει).

Ορισμός 1.12. Ένας k-χρωματισμός λέγεται έγκυρος όταν δεν υπάρχει ζεύγος γειτονικών κορυφών u, v στο G που η u και η v έχουν το ίδιο χρώμα, ενώ επιπλέον υπάρχει μια τουλάχιστον κορυφή στο G με καθένα από τα k χρώματα.

Ορισμός 1.13. Στο πρόβλημα Πολυχρωμο Μονοπατι το ζητούμενο είναι, δοθέντος ενός γραφήματος $G = (V, E)$, ενός ακεραίου k, και ενός k-χρωματισμού του G η εύρεση ενός απλού μονοπατιού μεγέθους k στο οποίο οι κορυφές του έχουν ανά δύο διαφορετικό χρώμα.

Λήμμα 1.14. Έστω G ένα k-χρωματισμένο γράφημα όπου θεωρούμε ότι όλοι οι $k^{|V|}$ χρωματισμοί είναι ισοπίθανοι και έστω $X \subseteq V(G)$ με $|X| = k$. Η πιθανότητα να είναι τα στοιχεία του X ανά δύο χρωματισμένα διαφορετικά είναι τουλάχιστον e^{-k}.

Απόδειξη. Το G έχει n κορυφές, και έχουμε k χρώματα στη διάθεσή μας, συνεπώς όλοι οι χρωματισμοί είναι k^n σε πλήθος. Ωστόσο οι έγκυροι είναι μόνο $k!k^{n-k}$, ώστε να χρωματίσουμε k κορυφές με όλα τα χρώματα (υπάρχουν $k!$ μεταθέσεις για το πως θα χρωματιστεί κάθε κορυφή του X), και μετά τις υπόλοιπες $n - k$ με οποιοδήποτε

χρώμα. Παρατηρούμε ότι $\frac{k!k^{n-k}}{k^n} = \frac{k!}{k^k} \geq e^{-k}$, αφού $e^k = \sum_{n=0}^{\infty} \frac{k^n}{n!}$ και άρα $e^k \geq \frac{k^k}{k!}$ (επί τη ευκαιρία, να γιατί χρειαζόμαστε τις σειρές Taylor). □

Έχοντας πει τα βασικά, ας δούμε τον αλγόριθμο. Για να βρούμε ένα μονοπάτι μεγέθους k, μπορούμε να χρησιμοποιήσουμε δυναμικό προγραμματισμό. Πιο συγκεκριμένα, θα πρέπει να ισχύει ότι:
$\text{Path}(S, u) = \bigvee_{uv \in E(G)} \text{Path}(S \setminus \chi(u), v)$, όπου $\chi(u)$ το χρώμα της κορυφής u και $\text{Path}(S, u) = \text{True}$ αν και μόνο αν υπάρχει πολύχρωμο μονοπάτι με άκρο την κορυφή u.

Θα επαναλάβουμε την παραπάνω πρόταση, αυτή τη φορά στα ελληνικά. Ξεκινώντας με το σύνολο των k χρωμάτων και ένα μονοπάτι που αποτελείται από μία κορυφή, έστω v, προφανώς ισχύει ότι $\text{Path}(\chi(v), v) = \text{True}$. Για να συνεχιστεί η αναδρομή, θα πρέπει να υπάρχει μια τουλάχιστον γειτονική κορυφή στην v, έστω η u, η οποία να έχει διαφορετικό χρώμα από την v. Συνεπώς το $\bigvee_{uv \in E(G)}$ είναι γιατί θέλουμε το μονοπάτι να συνεχιστεί σε μία γειτονική κορυφή της v, ενώ το $\text{Path}(S \setminus \chi(u), v)$ είναι γιατί θέλουμε η u να χρησιμοποιεί ένα χρώμα που δεν υπάρχει ήδη στο S.

Ο λόγος που ο παραπάνω αλγόριθμος είναι δυναμικός προγραμματισμός (και όχι άπληστος) είναι γιατί αναδρομικά ψάχνουμε μια βέλτιστη λύση (οπότε θα πρέπει τουλάχιστον ένα από τα εμφωλευμένα $\vee$ να είναι αληθές σε κάθε αναδρομή) αντί να ψάχνουμε αν ισχύει μια συγκεκριμένη ιδιότητα στους γείτονες της u.

Παράδειγμα 1.15. Θα εφαρμόσουμε δυναμικό προγραμματισμό στο γράφημα του σχήματος 1.5. Το γράφημα αυτό έχει χρωματικό αριθμό 6, συνεπώς ένα μονοπάτι για να είναι πολύχρωμο θα πρέπει να έχει 6 κορυφές με διαφορετικό χρώμα.

Απόδειξη. Το μονοπάτι αυτό είναι προφανώς το $P = (v_1, v_2, v_3, v_5, v_6, v_8)$, αλλά πάμε να το δούμε λίγο πιο φορμαλιστικά παρακάτω. Ξεκινώντας από την κορυφή v_1 και θέτοντας $S = \{\chi(v_1)\}$ ισχύει ότι $\text{Path}(S, v_1) = \text{True}$.
Για την πρώτη αναδρομή, έχουμε ότι:

$$\text{Path}(S, u) = \bigvee_{uv_1 \in E(G)} \text{Path}(S \setminus \chi(u), v_1), \text{ όπου } S = \{\text{black, red}\}, u = v_2.$$

Για την επόμενη αναδρομή:

$$\text{Path}(S, u) = \bigvee_{uv_2 \in E(G)} \text{Path}(S \setminus \chi(u), v_2), \text{ όπου } S = \{\text{black, red, green}\}, u = v_3.$$

Μετά όμως, επειδή η v_3 έχει δύο γείτονες, η αναδρομή θα έχει δύο $\vee$:

$$\text{Path}(S, u) = \bigvee_{uv_3 \in E(G)} \text{Path}(S \setminus \chi(u), v_3), \text{ όπου } S = \{\text{black, red, green}\}, u = v_4,$$

και,

$$\text{Path}(S, u) = \bigvee_{uv_3 \in E(G)} \text{Path}(S \setminus \chi(u), v_3), \text{ όπου } S = \{\text{black, red, green, blue}\}, u = v_5.$$

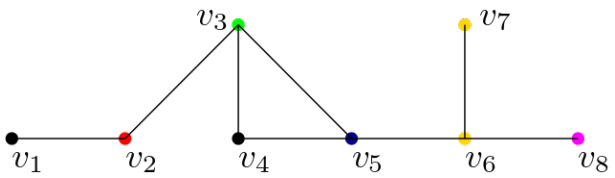
Η πρώτη αναδρομή δεν ισχύει γιατί έχουμε ήδη χρησιμοποιήσει το μαύρο, ισχύει όμως η δεύτερη. Τελικά θα φτάσουμε στην v_8 και θα ισχύει ότι $|S| = 6$, οπότε το μονοπάτι $P = (v_1, v_2, v_3, v_5, v_6, v_8)$ είναι πράγματι πολύχρωμο. □

Εφόσον το μονοπάτι που θέλουμε έχει μήκος k, υπάρχουν το πολύ 2^k υποσύνολα κορυφών για να ελέγξουμε αν ενάγουν πολύχρωμο μονοπάτι σε κάθε αναδρομή, οπότε ο παραπάνω αλγόριθμος έχει χρόνο εκτέλεσης $2^k n^c$, για κάποια σταθερά c.

Θεώρημα 1.16. Υπάρχει αλγόριθμος ο οποίος σε χρόνο $(2e)^k n^c$, για κάποια σταθερά c, επιστρέφει ένα πολύχρωμο μονοπάτι με k κορυφές ή αναφέρει ότι δεν υπάρχει λύση.

Απόδειξη. Ο παραπάνω αλγόριθμος τρέχει σε χρόνο $2^k n^c$, ενώ η πιθανότητα να χρωματίσουμε τυχαία ένα γράφημα με τις κορυφές του να έχουν ανά δυό διαφορετικό χρώμα είναι e^{-k}, όπως δείξαμε στο λήμμα 1.14. Συνεπώς, επαναλαμβάνοντας e^k φορές τον παραπάνω αλγόριθμο, προκύπτει το ζητούμενο. □

Η μέθοδος που εφαρμόστηκε παραπάνω, στην οποία εφαρμόσαμε πολλές φορές τον πιθανοτικό αλγόριθμο, ώστε ακόμα και αν κάποιες φορές αποτύχει, να εξασφαλίσουμε ότι αν υπάρχει λύση θα τη βρει με μεγάλη πιθανότητα σε κάποια από τις πολλαπλές εκτελέσεις, είναι συνηθισμένη τακτική και στην πράξη φαίνεται να δουλεύει καλά.



Σχήμα 1.5: Παράδειγμα πολύχρωμου γραφήματος με 6 χρώματα.

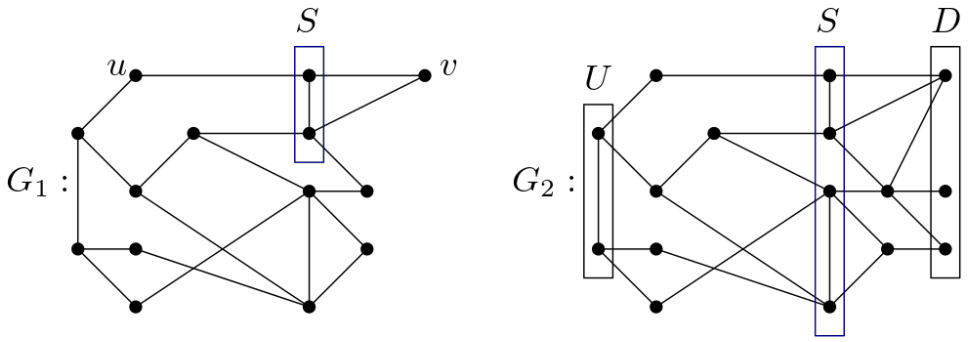
1.4.5 Σημαντικοί διαχωριστές (Important separators)

Η έννοια του διαχωριστή και οι γενικεύσεις του είναι πολύ σημαντικές, ειδικά σε αλγόριθμους για κατευθυνόμενα γραφήματα, προκειμένου να κατασκευαστούν όσο γίνεται γρηγορότεροι παραμετρικοί αλγόριθμοι.

Ορισμός 1.17. Έστω u, v δύο κορυφές ενός συνεκτικού γραφήματος G. Ονομάζουμε διαχωριστή ένα σύνολο κορυφών του G η αφαίρεση του οποίου διαγράφει όλα τα μονοπάτια μεταξύ των u, v. Αν ένας διαχωριστής είναι και ελάχιστος, έχει δηλαδή το ελάχιστο δυνατό πλήθος κορυφών, τότε ονομάζεται ελάχιστος διαχωριστής.

Η έννοια του διαχωριστή μπορεί να γενικευτεί ώστε αντί για κορυφές u, v να έχουμε δύο σύνολα $U, D \subseteq V(G)$ τα οποία είναι διακεκριμμένα, οπότε το ζητούμενο είναι αφαιρώντας τις κορυφές του διαχωριστή να μην υπάρχει κορυφή του U που να συνδέεται με κορυφή του D. Επιπλέον η έννοια του διαχωριστή μπορεί να γενικευτεί ακόμα περισσότερο για οικογένειες συνόλων.

Στο σχήμα 1.6 το σύνολο S είναι διαχωριστής των κορυφών u, v στο γράφημα G_1. Αντίστοιχα, για το γράφημα G_2, το σύνολο S είναι διαχωριστής των κορυφών που ανήκουν στα σύνολα U, D (είναι όμως ελάχιστοι;).



Σχήμα 1.6: Παραδείγματα διαχωριστών.

Συχνά, όπως και στο πρόβλημα Συνολο Αναδρασης Κορυφων, το ζητούμενο είναι δεδομένου ενός γραφήματος G να το κάνουμε άκυκλο. Στην περίπτωση που το G είναι κατευθυνόμενο, μέχρι σήμερα οι FPT αλγόριθμοι που έχουν αναπτυχθεί κάνουν χρήση διαχωριστών προκειμένου να κάνουν το G άκυκλο.

Ένας ακόμα περιορισμός που μπορούμε να λάβουμε υπόψιν όταν έχουμε να διαχωρίσουμε δύο σύνολα U, D είναι να μεγιστοποιήσουμε τον αριθμό των κορυφών με τις οποίες συνδέονται οι κορυφές του U στο γράφημα $G \setminus S$, όπου S ένας ελάχιστος διαχωριστής. Όταν ισχύει αυτός ο περιορισμός, τότε το S λέγεται ότι είναι Pareto-αποδοτικός διαχωριστής.

Η ονομασία προέρχεται από τον κανόνα του Pareto, ο οποίος αναφέρει ότι όταν αναλύουμε προβλήματα (π.χ. σε μια επιχείρηση) και τις συνέπειες αυτών, το 80% των προβλημάτων προέρχεται από το 20% των αιτιών που αφορούν τα προβλήματα αυτά. Μια άλλη ονομασία του κανόνα αυτού είναι ”κανόνας 80/20”, ωστόσο αυτό που είχε παρατηρήσει ο Pareto ήταν ότι το 80% του πλούτου της Ιταλίας το κατέχει το 20% του πληθυσμού της.

Περισσότερα για Pareto-αποδοτικότητα, αν σας αρέσει η θεωρία παιγνίων, στο βιβλίο [11].

1.4.6 Ακέραιος γραμμικός προγραμματισμός (Integer linear programming)

Ο γραμμικός προγραμματισμός δεν θα μπορούσε να λείπει από τις παρούσες σημειώσεις, μιας και παίζει σημαντικό ρόλο στην κατασκευή αλγορίθμων. Επανερχόμαστε ξανά στο πρόβλημα Καλυμμα Απο Κορυφες, και θα δείξουμε ότι με γ.π. μπορούμε να πάρουμε έναν πυρήνα μεγέθους $2k$.

Ο ορισμός του προβλήματος Κάλυμμα Απο Κορυφες λέει ότι θέλουμε τον ελάχιστο αριθμό κορυφών ενός γραφήματος G, ώστε τουλάχιστον ένα από τα άκρα κάθε ακμής του G να περιέχεται στο κάλυμμα. Σε γλώσσα γ.π. αυτό εκφράζεται ως εξής:

$$\min \quad \sum_{v \in V} x_v \qquad \text{υπό τους περιορισμούς:} \qquad \begin{array}{l} x_v + x_u \geq 1, \text{ για κάθε } uv \in E, \\ x_v \in \{0,1\}, \text{ για κάθε } v \in V. \end{array}$$

Για το ακέραιο πρόγραμμα, αν $x_v = 0$ σημαίνει ότι η v δεν ανήκει στο κάλυμμα, ενώ αν $x_v = 1$ η κορυφή ανήκει στο κάλυμμα. Ωστόσο το παραπάνω είναι ακέραιο πρόγραμμα άρα είναι $\mathcal{NP}$-δύσκολο, οπότε θα λύσουμε το αντίστοιχο χαλαρωμένο γ.π., στο οποίο ο περιορισμός $x_v \in \{0,1\}$ αλλάζει σε $0 \leq x_v \leq 1$. Θεωρούμε τα σύνολα:

- $V_0 = \left\{ v \in V : x_v < \frac{1}{2} \right\}$.
- $V_{1/2} = \left\{ v \in V : x_v = \frac{1}{2} \right\}$.
- $V_1 = \left\{ v \in V : x_v > \frac{1}{2} \right\}$.

Τα παραπάνω σύνολα είναι μια διαμέριση των κορυφών του G η οποία προκύπτει από τα βάρη τους, όπως έχουν προκύψει από την λύση του παραπάνω γ.π.. Οι κορυφές του συνόλου V_0 δεν μας ενδιαφέρουν καθώς μπορούμε να αποδείξουμε ότι για ένα βέλτιστο κάλυμμα S ισχύει ότι $V_1 \subseteq S \subseteq V_1 \cup V_{1/2}$ (παράγραφος 6.1, [8], ή/και παράγραφος 2.5, [7]).

Η βασική ιδέα της απόδειξης για το μέγεθος του πυρήνα χωρίζεται σε δύο μέρη. Αφενός, αφού δεν μας ενδιαφέρουν οι κορυφές που ανήκουν στο σύνολο V_0, προκύπτει άμεσα ότι το γράφημα G με παράμετρο[3] k έχει λύση για το Κάλυμμα Απο Κορυφες $\iff$ το γράφημα $G[V_{1/2}]$ με παράμετρο $k - |V_1|$ (αφού διαγράψαμε από το G όλες τις κορυφές του V_1) έχει λύση. Αφετέρου, για το συνολικό βάρος των κορυφών του $V_{1/2}$ ισχύει ότι $|V_{1/2}| = \sum_{v \in V_{1/2}} 2x_v \leq \sum_{v \in V} 2x_v = 2\sum_{v \in V} x_v \leq 2k$, από όπου προκύπτει το μέγεθος του πυρήνα. Με $G[V_{1/2}]$ συμβολίζουμε το γράφημα που προκύπτει από το G αν κρατήσουμε μόνο τις κορυφές του $V_{1/2}$ και τις μεταξύ τους ακμές[4]. Ο λόγος για το 2 στο άθροισμα είναι ότι οι κορυφές του $V_{1/2}$ έχουν τιμή $1/2$ ενώ για το ακέραιο γ.π. θέλουμε κάθε κορυφή του S να έχει τιμή 1.

Συχνά τα προβλήματα της θεωρίας γραφημάτων αντιμετωπίζονται με γ.π., καθώς η μελέτη που έχει γίνει στα γ.π. είναι αρκετά εκτενής. Τα γ.π. επίσης χρησιμοποιούνται εκτενώς στην κατασκευή προσεγγιστικών αλγορίθμων ([6]) και στην συνδυαστική βελτιστοποίηση γενικότερα ([12]). Μια ενδιαφέρουσα θεωρία γύρω από τον γ.π. λέγεται πολυεδρική συνδυαστική ([9]).

1.4.7 Φραγμένο δεντροπλάτος (Bounded treewidth)

Πολλά $\mathcal{NP}$-δύσκολα προβλήματα της θεωρίας γραφημάτων λύνονται σε πολυωνυμικό χρόνο όταν περιοριζόμαστε σε δέντρα. Για τον λόγο αυτόν, αναπτύχθηκε η έννοια του δεντροπλάτους, η οποία αφορά το κατά πόσο ένα γράφημα απέχει από το να είναι δέντρο. Το να έχει ένα γράφημα φραγμένο δεντροπλάτος μας βοηθάει στην κατασκευή (γρηγορότερων) αλγορίθμων σε σχέση με αυτούς που ήδη υπάρχουν. Προτού δοθεί ο ορισμός του δεντροπλάτους, θα δοθεί ο ορισμός της δεντροαποσύνθεσης (tree decomposition):

Ορισμός 1.18. Έστω $G = (V, E)$ ένα γράφημα και έστω T ένα δέντρο με κορυφές $X_1, \ldots, X_n$ για τα οποία ισχύουν τα εξής:

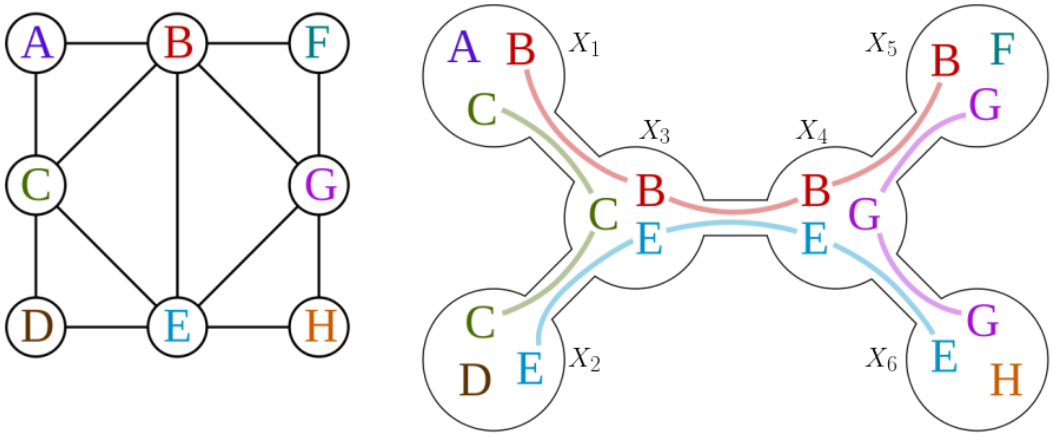
1. $\bigcup_{i=1}^{n} X_i = V$, δηλαδή κάθε κορυφή του G περιέχεται σε τουλάχιστον μία κορυφή X_i του T. Οι κορυφές X_i ονομάζονται και τσάντες (bags).
2. Αν $v \in X_i$ και $v \in X_j$ τότε $v \in X_k$ για κάθε κορυφή X_k του T η οποία βρίσκεται μεταξύ των X_i, X_j.
3. Για κάθε ακμή uv του G υπάρχει μια κορυφή X_i του T που περιέχει τις u, v.

[3] Θυμίζουμε ότι στην παραμετροποιημένη εκδοχή του Κάλυμμα Απο Κορυφες ψάχνουμε μια λύση μεγέθους το πολύ k.

[4] Το γράφημα αυτό λέγεται εναγώμενο (induced).

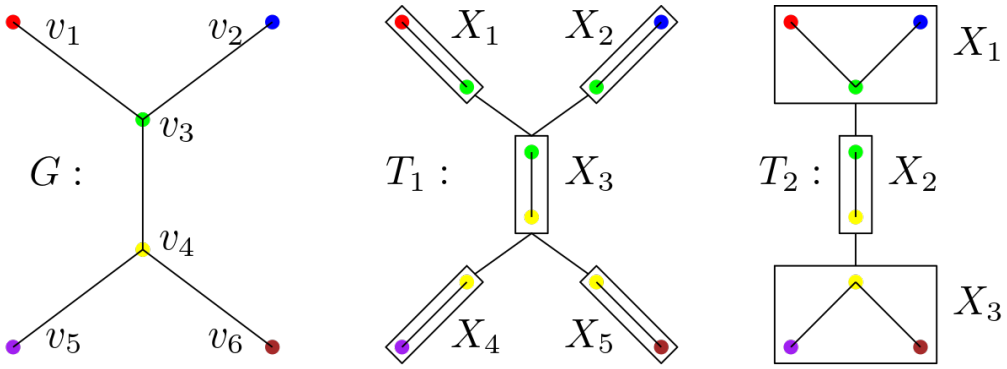
Το πλάτος της δεντροαποσύνθεσης ορίζεται ως $\text{width}(T) = \max |X_i| - 1$ (το -1 είναι ώστε τα δέντρα να έχουν δεντροπλάτος ίσο με 1). Το δεντροπλάτος του G συμβολίζεται με $\text{tw}(G)$ και ισούται με το ελάχιστο μέγεθος από όλες τις δυνατές δεντροαποσυνθέσεις του.

Όπως ήδη μαρτυρήσαμε στον ορισμό, δεν υπάρχει μοναδική δεντροαποσύνθεση για κάποιο γράφημα G, ωστόσο μας ενδιαφέρουν οι δεντροαποσυνθέσεις με όσο το δυνατόν μικρότερο μέγεθος. Μια τετριμμένη δεντροαποσύνθεση ενός γραφήματος G είναι να τοποθετηθεί όλο το G σε μια τσάντα. Ένα πιο κατατοπιστικό παράδειγμα είναι το εξής: Στο γράφημα του σχήματος 1.7 παρατηρούμε ότι πράγματι $\bigcup_{i=1}^{6} X_i = V$, ότι για κάθε $uv \in G$



Σχήμα 1.7: Παράδειγμα δεντροαποσύνθεσης. Πηγή.

υπάρχει μια τουλάχιστον τσάντα που περιέχει τις u, v (παρατηρήστε ότι η ακμή BE βρίσκεται σε δύο τσάντες, τις X_3, X_4). Επίσης η κορυφή B η οποία βρίσκεται στις τσάντες X_1, X_5 βρίσκεται και στις ενδιάμεσες τσάντες, δηλαδή τις X_3, X_4. Το παραπάνω γράφημα έχει δεντροπλάτος ίσο με 2, αφού δεν υπάρχει μικρότερη δεντροαποσύνθεση (αν προσπαθήσουμε να φτιάξουμε μια μικρότερη δεντροαποσύνθεση, θα προκύψουν κύκλοι στο T προκειμένου να ικανοποιηθεί ο ορισμός, τότε όμως το T θα πάψει να είναι δέντρο).



Σχήμα 1.8: Άλλο ένα παράδειγμα δεντροαποσύνθεσης.

Στο γράφημα του σχήματος 1.8 έχουμε δύο δεντροαποσυνθέσεις, η μία μεγέθους 1 (αριστερά) και η άλλη μεγέθους 2 (δεξιά).

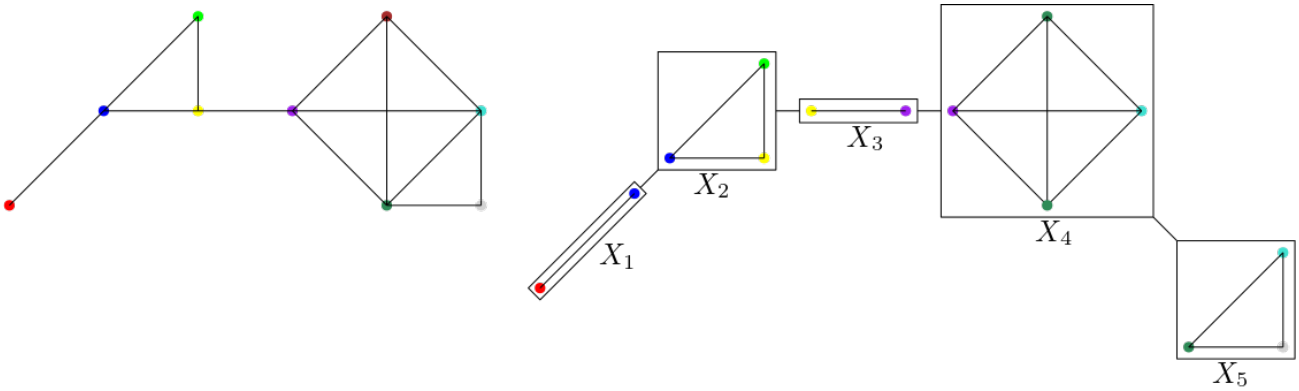
Παρόμοια έννοια είναι και η αποσύνθεση μονοπατιού όπου αντί να προκύψει ένα δέντρο T από την αποσύνθεση του G, το ζητούμενο είναι να προκύψει ένα μονοπάτι P, όπως φαίνεται στο σχήμα 1.9. Επιπλέον, είναι $\mathcal{NP}$-πλήρες το πρόβλημα του αν ένα γράφημα G έχει δεντροαποσύνθεση μεγέθους το πολύ k.

Ένας επιπλέον λόγος που η έννοια του δεντροπλάτους είναι χρήσιμη είναι πως υπάρχουν προβλήματα που λύνονται αποδοτικά με δυναμικό προγραμματισμό για γραφήματα με φραγμένο δεντροπλάτος.

Λεπτομέρειες για δεντροπλάτος υπάρχουν στα βιβλία [7, 1]. Οι εφαρμογές που βασίζονται στο δεντροπλάτος ενός γραφήματος είναι αρκετά τεχνικές, οπότε απλά θα δοθεί ένα πόρισμα που δείχνει την χρησιμότητα του δεντροπλάτους.

Πόρισμα 1.19 ([7])**.** Έστω G ένα γράφημα και μια δεντροαποσύνθεση του G μεγέθους το πολύ k. Υπάρχει αλγόριθμος που βρίσκει ένα ελάχιστο κάλυμμα από κορυφές στο G σε χρόνο $2^k \cdot n \cdot k^{O(1)}$.

Υπενθυμίζεται ότι το Κάλυμμα από Κορυφές και το Ανεξάρτητο Σύνολο σε ένα γράφημα G είναι ισοδύναμα προβλήματα (γιατί;) και άρα ο παραπάνω αλγόριθμος λύνει και το Ανεξάρτητο Σύνολο.



Σχήμα 1.9: Παράδειγμα αποσύνθεσης μονοπατιού μεγέθους 3.

Βιβλιογραφία

[1] Reinhard Diestel. **Graph Theory**, 5η έκδοση, 2016. Εκδόσεις Springer.

[2] Michael Sipser. **Εισαγωγή στη θεωρία υπολογισμού**, 3η έκδοση, 2020. Πανεπιστημιακές Εκδόσεις Κρήτης.

[3] Christos H. Papadimitriou. **Computational Complexity**, 1993. Εκδόσεις Pearson.

[4] Harry R. Lewis, Χρίστος Χ. Παπαδημητρίου. **Στοιχεία Θεωρίας Υπολογισμού**, 2η έκδοση, 2005. Εκδόσεις Κριτική.

[5] Richard Karp. **Reducibility Among Combinatorial Problems**, 1972. Wikipedia.

[6] David P. Williamson, David B. Shmoys. **The Design of Approximation Algorithms**, 2011. Εκδόσεις Cambridge University Press. Διαθέσιμο δωρεάν εδώ.

[7] Marek Cygan, Fedor V. Fomin, Łukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michał Pilipczuk, Saket Saurabh. **Parameterized Algorithms**, 2015. Εκδόσεις Springer. Διαθέσιμο δωρεάν εδώ.

[8] Fedor V. Fomin, Daniel Lokshtanov, Saket Saurabh, Meirav Zehavi. **Kernelization: Theory of Parameterized Preprocessing**, 2019. Εκδόσεις Cambridge University Press. Διαθέσιμο δωρεάν εδώ.

[9] A. Schrijver. **Combinatorial Optimization - Polyhedra and Efficiency Volume A-C**, 2002. Εκδόσεις Springer-Verlag.

[10] Σημειώσεις μαθήματος **Εφαρμογές Θεωρίας Γραφημάτων**, τμήμα Πληροφορικής, Πανεπιστήμιο Πειραιώς. Σύνδεσμος μαθήματος.

[11] Anna R. Karlin, Yuval Peres. **Game Theory, Alive**, 2016. Διαθέσιμο δωρεάν εδώ.

[12] Christos H. Papadimitriou, Kenneth Steiglitz. **Combinatorial Optimization: Algorithms and Complexity**, 1982. Εκδόσεις Dover Publications.

[13] Neil Immerman. **Descriptive Complexity**, 1999. Εκδόσεις Springer.

[14] Boaz Barak, Sanjeev Arora. **Computational Complexity: A Modern Approach**, 2007. Εκδόσεις Cambridge University Press. Web Draft.

[15] Thomas Zeume. **Small Dynamic Complexity Classes: An Investigation Into Dynamic Descriptive Complexity**, 2017. Εκδόσεις Springer.

[16] Karl Bringmann. **Fine-Grained Complexity Theory**, 2019. Διαθέσιμο στον σύνδεσμο αυτόν.

[17] Virginia Vassilevska Williams. **On some fine-grained questions in algorithms and complexity**, 2018. Διαθέσιμο στον σύνδεσμο αυτόν.

[18] Σημειώσεις μαθήματος **Θεωρία Αναδρομής**, ΔΠΜΣ Αλγόριθμοι, Λογική και Διακριτά Μαθηματικά, ΕΚΠΑ-ΕΜΠ. Σύνδεσμος μαθήματος.