Εθνικό Μετσόβιο Πολυτεχνείο
Τμήμα Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών
Τομέας Τεχνολογίας Πληροφορικής και Υπολογιστών

Διπλωματική εργασία:

ο Ισομορφισμός Γράφων και Κλάσεις Πολυπλοκότητας

Τζάννες Αλέξανδρος

Τριμελής επιτροπή Καθηγητών:

Επιβλέπων: Στάθης Ζάχος

Μέλη: Φώτω Αφράτη
Νίκος Παπασπύρου

Αθήνα, 2004

Ο Ισομορφισμός Γράφων και Κλάσεις Πολυπλοκότητας

Ο Ισομορφισμός Γράφων και Κλάσεις Πολυπλοκότητας

Τζάννες Αλέξανδρος

στους γονείς μου

Περίληψη. Ο Ισομορφισμός Γράφων (**GI**) είναι ένα από τα ελάχιστα προβλήματα που αν και ανήκουν στην κλάση NP δεν είναι γνωστό αν ανήκουν στο P ή αν είναι NP-Complete (ή αν ισχύει κάτι άλλο γι'αυτό). Πολλές παραλλαγές του προβλήματος έχουν μελετηθεί και έχουν βρεθεί να ανήκουν στο P με αποκορύφωμα την περίπτωση του ισομορφισμού επίπεδων γράφων για το οποίο μάλιστα βρέθηκε αλγόριθμος που εκτελείται σε γραμμικό χρόνο. Υπάρχουν πολλά στοιχεία τα οποία συνηγορούν στο ότι το **GI** δεν είναι NP-Complete, αλλά παρόλα αυτά παραμένει αρκετά "μακρυά" από το P καθώς δεν είναι καν γνωστό αν ανήκει στο co-NP. Ένα πρόβλημα το οποίο μέχρι πριν μερικά χρόνια (2002) είχε το ίδιο status με το **GI** είναι το primality, το πρόβλημα απόφασης αν ένας ακέραιος είναι πρώτος. Για το primality όμως η εικασία ήταν εδώ και πολλά χρόνια ότι ανήκει στο P καθώς είχε αποδειχθεί ότι ήταν στο $NP \cap co - NP$ και μάλιστα στο ZPP. Μέσα από αυτή τη διπλωματική εργασία, στόχος μου είναι να προσεγγίσω τη θεωρία πολυπλοκότητας μέσω ενός ανοιχτού προβλήματος (του **GI**) έτσι ώστε φανούν και κάποιες μεθοδολογίες αντιμετώπισης ανοιχτών προβλημάτων.

Λέξεις κλειδιά: ισομορφισμός, αυτομορφισμός, γράφος, δομική πολυπλοκότητα, πιθανοτικός, πολυωνυμική ιεραρχία, τελεστές, ποσοδείκτες, διαδραστικές αποδείξεις, μηδενική γνώση, Arthur-Merlin, παίγνια, lowness.

Abstract. The graph isomorphism (**GI**) is one of the few natural problems that are in NP but it is not known wether it is in P or if it is NP-Complete (or if something else holds). Many variations of the problem have been studied and found to be in P. The most impressive result in that direction is the linear time algorithm for the planar graph isomorphism. There are many results that indicate that **GI** is not NP-Complete, however the problem still lies quite "far away" from P since it is not know if it belongs to co-NP. A problem that had the same status until a few years ago (2002) was *primality*, the problem of deciding if an integer is prime. For *primality* however it was conjectured to belong to P since many years ago and was proven to belong to $NP \cap co - NP$ and even ZPP. Through this dissertation, my goal is to attack the complexity theory by means of an open problem (**GI**) in a way that will reveal some methodologies employed when trying to categorize a problem into a complexity class.

Keywords: graph, isomorphism, automorphism, structural, complexity, probabilistic, polynomial hierarchy, quantifiers, operators, interactive proofs, zero-knowledge, Arthur-Merlin, games, lowness.

Ευχαριστίες. Νιώθω την ανάγκη να ευχαριστήσω τα άτομα που με στήριξαν στην προσπάθεια της συγγραφής της παρούσας εργασίας. Αρχικά θέλω να ευχαριστήσω τον κ. Στάθη Ζάχο και τον κ. Άρη Παγουρτζή για τη βοήθεια και την καθοδήγηση κατά τη διάρκεια της εργασίας. Την Κατερίνα Ποτίκα που με βοήθησε σημαντικά με τη διόρθωση λαθών στο τελικό κείμενο. Τον Πάνο Χείλαρη που με βοήθησε με τα πρώτα μου βήματα στο LaTeX. Τον κ. Νίκο Παπασπύρου και την κα. Φώτω Αφράτη, μέλη της τριμελούς επιτροπής, για την άριστη συνεργασία που είχαμε γενικά κατά τη διάρκεια των σπουδών μου.

Τέλος θέλω να ευχαριστήσω όλους τους συγγενείς και τους φίλους μου που με στήριξαν και με στηρίζουν όλα τα χρόνια. Θέλω να αφιερώσω την εργασία αυτή στους γονείς μου για την αγάπη τους, τη στήριξη και για την απέραντη εμπιστοσύνη που μου δείχνουν καθημερινά.

Περιεχόμενα

Κατάλογος σχημάτων

Κατάλογος αλγορίθμων

Μέρος πρώτο

Ισομορφισμός Γράφων

Κεφάλαιο 1

Βασικές Έννοιες

1.1 Θεωρία Γραφημάτων

Αρχικά θα πρέπει να ορίσουμε τα αντικείμενα τα οποία θα μας απασχολήσουν κατά κύριο λόγο στη συνέχεια, τους γράφους. Πολλές φορές αναφέρονται στην ελληνική βιβλιογραφία και ως γραφήματα. Και οι δύο αυτές λέξεις θέλουν να αποδώσουν τον όρο 'graph'. Θα δείξουμε μια προτίμηση στον ελληνικό όρο 'γράφος' καθότι πιο κοντά στον ξένο.

Ορισμός 1.1. *Ένας γράφος $G = G(V, E)$ είναι ένα διατεταγμένο ζεύγος δύο ξένων μεταξύ τους συνόλων $V \neq \emptyset$ και E, όπου E είναι ένα υποσύνολο του συνόλου των μη διατεταγμένων ζευγών του V.*

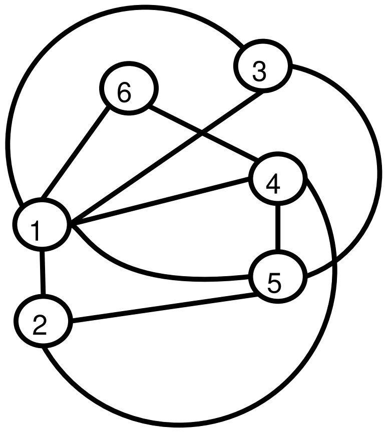
Το σύνολο V είναι το σύνολο των κορυφών (vertices) του γράφου, ενώ το E είναι το σύνολο των ακμών, ή πλευρών του γράφου (edges). Ο αριθμός των κορυφών ενός γράφου $|V| = |V(G)|$ ονομάζεται *τάξη* του γράφου G, ενώ ο αριθμός των πλευρών ενός γράφου $|E| = |E(G)|$ ονομάζεται *μέγεθος* του γράφου G.

Συμβολίζουμε με *(uv)* την πλευρά που ενώνει τις κορυφές $u \in V$ και $v \in V$. Επειδή οι πλευρές ορίστηκαν ως μη διατεταγμένα ζεύγη κορυφών, η πλευρά (uv) μπορεί να γραφεί και ως (vu). Μια πλευρά λέμε ότι προσπίπτει ή διέρχεται από τις κορυφές τις οποίες ενώνει. Αυτές λέγονται και άκρα της πλευράς.

Γειτονικές λέγονται δυο κορυφές όταν υπάρχει πλευρά που τις ενώνει. Γειτονικές λέγονται δυο πλευρές όταν προσπίπτουν σε κοινή κορυφή.

Με την παραπάνω ορολογία για τα σύνολα V και E οδηγούμαστε με φυσιολογικό τρόπο σε μια παρουσίαση των γράφων με σχηματικό τρόπο (βλ. Σχήμα 1.1).

Ορισμός 1.2. *Ο γράφος $G' = (V', E')$ είναι* **υπογράφος** *(subgraph) του γράφου $G(V, E)$ αν $V' \subset V$ και $E' \subset E$. Για την ειδική περίπτωση όπου $V' = V$*



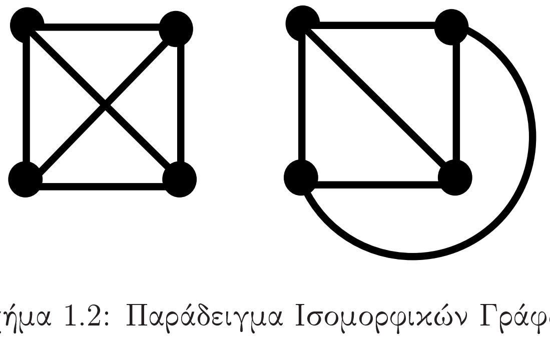
Σχήμα 1.1: Παράδειγμα γράφου

ο G' ονομάζεται **παράγων υπογράφος** *του G (spanning subgraph). Τέλος, αν ο G' περιέχει όλες τις πλευρές του G που προσπίπτουν στις κορυφές του V', τότε λέμε ότι ο G' είναι υπογράφος* **παραγόμενος** *από το V' (subgraph of G induced by V') και τον συμβολίζουμε ως $G[V']$.*

Συμβολίζουμε με $G(n)$ έναν αυθαίρετο γράφο τάξης n και με $G(n, m)$ έναν αυθαίρετο γράφο τάξης n και μεγέθους m.

Παρατήρησή 1.3. *Όταν σε έναν γράφο περιέχονται βρόγχοι ή πολλαπλές πλευρές, θα τον ονομάζουμε* **πολυγράφο***(multigraph).*

Στους γράφους, όπως φάνηκε και στο σχήμα 1.1, μπορούμε να δίνουμε ονόματα στις κορυφές και τις πλευρές. Η διαίσθησή μας, μας λέει πως τα ονόματα που δίνουμε δεν αλλάζουν τις ιδιότητες τους γράφους. Με το ίδιο σκεπτικό διαπιστώνουμε ότι ούτε και ο τρόπος που σχεδιάζουμε ένα γράφο έχει σημασία. Για παράδειγμα οι δύο γράφοι του σχήματος 1.2 δεν διαφέρουν. Λέμε ότι είναι *ισομορφικοί.*



Σχήμα 1.2: Παράδειγμα Ισομορφικών Γράφων

Ορισμός 1.4. *Ισομορφικοί λέγονται δύο γράφοι* $G_1(V_1, E_1)$ *και* $G_2(V_2, E_2)$ *για τους οποίους υπάρχει μια "1-1" και "επί" συνάρτηση (απεικόνιση)* $\varphi : V_1 \to V_2$ *έτσι ώστε :*

$$(vu) \in E_1 \iff (\varphi(v)\varphi(u)) \in E_2$$

Με άλλα λόγια, η συνάρτηση φ *πρέπει να διατηρεί τις γειτνιάσεις, αλλά και τις μη γειτνιάσεις κορυφών. Το γεγονός ότι δύο γράφοι είναι ισομορφικοί το συμβολίζουμε με* $G_1 \equiv G_2$.

Το πρόβλημα του ισομορφισμού γράφων έχει πληθώρα πρακτικών εφαρμογών και από θεωρητικής απόψεως παρουσιάζει μεγάλο ενδιαφέρον γιατί αν και εικάζεται ότι δεν είναι NP-Complete, δεν έχει βρεθεί ακόμα αλγόριθμος πολυωνυμικού χρόνου που να το λύνει στη γενική του περίπτωση.

1.2 Θεωρία Πολυπλοκότητας

Η θεωρία πολυπλοκότητας είναι μια σχετικά νέα περιοχή έρευνας η οποία αναπτύχθηκε μεταξύ άλλων, όταν διαπιστώθηκε ότι υπήρχαν προβλήματα τα οποία ήταν αρκετά πιο δύσκολα από άλλα και κανείς δεν μπορούσε να βρει κάποιον αλγόριθμο ο οποίος να τα λύνει αποδοτικά για όλες τις περιπτώσεις τους. Φυσικά μπαίνουν ερωτήματα όπως το τι θεωρείται αποδοτικό και το πότε θεωρείται δύσκολο ένα πρόβλημα. Παρότι σε μεγάλο βαθμό τα πράγματα έχουν κατασταλάξει για το πώς ορίζονται οι δύο παραπάνω έννοιες, πάντα μπορεί κανείς να αμφισβητήσει την καθολικότητα των ορισμών.

Ορισμός 1.5. *(Συμβολισμοί που βοηθούν στη θεωρία)*
Ορίζουμε αρχικά τα Ο, Ω και Θ.

- $f(n) = \mathcal{O}(g(n)) \iff \exists c \in \mathbb{N}, \exists n_0 \in \mathbb{N}, \forall n \geq n_0, f(n) \leq c \cdot g(n)$
  Διαισθητικά μπορούμε να πούμε ότι η f *αυξάνεται όπως η* g *ή και πιο αργά.*
- $f(n) = \Omega(g(n)) \iff \exists c \in \mathbb{N}, \exists n_0 \in \mathbb{N}, \forall n \geq n_0, f(n) \geq c \cdot g(n)$
  Αυτή είναι η αντίθετη περίπτωση όπου η f *αυξάνεται όπως η* g *ή και πιο γρήγορα.*
- $f(n) = \Theta(g(n)) \iff \exists c_1, c_2 \in \mathbb{N}, \exists n_0 \in \mathbb{N}, \forall n \geq n_0, c_1 \cdot g(n) \leq f(n) \leq c_2 \cdot g(n)$
  Το Θ *εκφράζει την περίπτωση όπου η* f *αυξάνει με ακριβώς όπως η* g *(πιθανώς με διαφορά μιας πολλαπλασιαστικής σταθεράς).*

Εναλλακτικά μπορούμε να δώσουμε τους παραπάνω ορισμούς υπό την μορφή των συνόλων $\mathcal{O}(g)$, $\Omega(g)$ *και* $\Theta(g)$.

- $\mathcal{O}(g) = \{f|\exists c > 0, \exists n_0 : \forall n > n_0, f(n) \leq c \cdot g(n)\}$
- $\Omega(g) = \{f|\exists c > 0, \exists n_0 : \forall n > n_0, f(n) \geq c \cdot g(n)\}$
- $\Theta(g) = \{f|\exists c_1 > 0, \exists c_2 > 0, \exists n_0 : \forall n > n_0, c_1 \cdot g(n) \leq f(n) \leq c_2 \cdot f(n)\}$
  ή εναλλακτικά : $\Theta(g) = \mathcal{O}(g) \bigcap \Omega(g)$

Τέλος, ορίζουμε με παρόμοιο τρόπο τα ο *και* ω

- $o(g) = \{f|\forall c > 0, \exists n_0 : \forall n > n_0, f(n) \leq c \cdot g(n)\}$
- $\omega(g) = \{f|\forall c > 0, \exists n_0 : \forall n > n_0, f(n) \geq c \cdot g(n)\}$

Οι παραπάνω ορισμοί μας ενδιαφέρουν για να εκφράσουμε το κόστος αλγορίθμων σε συνάρτηση με το μέγεθος της εισόδου τους (παράμετρος n). Είναι διαισθητικά προφανές ότι ένας αλγόριθμος πρέπει να κάνει περισσότερους υπολογισμούς όσο πιο μεγάλο είναι στιγμιότυπο (instance) του προβλήματος που πρόκειται να λύσει. Για παράδειγμα ή ταξινόμηση ενός πίνακα που έχει 10 στοιχεία γίνεται πιο γρήγορα από ότι αν είχε 100 στοιχεία. Επομένως έχουμε στο μυαλό μας αύξουσες συναρτήσεις όταν μιλάμε για συναρτήσεις που εκφράζουν πολυπλοκότητα.

Παρότι οι παραπάνω ορισμοί είναι ευρέως αποδεκτοί, υπάρχουν ενστάσεις για την ύπαρξη του n_0 πάνω από το οποίο ισχύουν οι ανισότητες, δηλαδή για το ότι οι ορισμοί αυτοί προσδιορίζουν την ασυμπτωτική συμπεριφορά των συναρτήσεων (που εκφράζουν την πολυπλοκότητα). Συχνά μας ενδιαφέρει η συμπεριφορά της συνάρτησης πολυπλοκότητας για μικρές εισόδους για τον απλό λόγο ότι για μεγάλες εισόδους το πρόβλημα είναι έτσι κι αλλιώς υπολογιστικά απρόσιτο. Ένα άλλο σημείο στους ορισμούς το οποίο προκαλεί συζητήσεις είναι το ότι αγνοούνται οι πολλαπλασιαστικές σταθερές. Έστω για παράδειγμα η συνάρτηση $f_1(n) = 10^{10^{10}}n$ ή οποία είναι $f_1 = \mathcal{O}(n)$ και η συνάρτηση $f_2(n) = n^{\log n}$ η οποία είναι $f_2 = \mathcal{O}(n^{\log n})$. Παρότι η f_1 είναι γραμμική ως προς την είσοδο και η f_2 είναι υπο-εκθετική (subexponential), είναι φανερό ότι για n_0 αρκετά μεγάλο ισχύει ότι $\forall n < n_0 : f_1(n) > f_2(n)$. Ενώ λοιπόν, από θεωρητικής απόψεως ένας αλγόριθμος με γραμμική πολυπλοκότητα είναι ιδιαίτερα θεμιτός, στην πράξη ενδεχομένως ένας αλγόριθμος με πολύ μεγαλύτερη πολυπλοκότητα να δουλεύει καλύτερα. Επίσης, αν θεωρήσουμε την $f_3 = n$ τότε είναι και αυτή $f(n) = \mathcal{O}(n)$ όπως και η f_1, αλλά στην πράξη η f_1 "κοστίζει" πολύ περισσότερο.

Παρόλα αυτά οι ορισμοί αυτοί είναι αρκετά καλοί για ρεαλιστικά προβλήματα γιατί εκεί συνήθως δεν εμφανίζονται υπερβολικά μεγάλες σταθερές και το n_0 των ορισμών μπορεί να επιλεγεί αρκετά μικρό.

Ένα θέμα που προκύπτει είναι αυτό της κωδικοποίησης της εισόδου. Μιλήσαμε για μέγεθος της εισόδου, το οποίο μάλιστα θεωρούμε ως παράμετρο στη συνάρτηση η οποία μας δίνει την πολυπλοκότητα ενός αλγορίθμου για τη συγκεκριμένη είσοδο. Το μέγεθος αυτό της εισόδου εξαρτάται από τον τρόπο με τον οποίο παριστάνουμε το στιγμιότυπο, από τον τρόπο δηλαδή που το κωδικοποιούμε. Ένας ακέραιος μπορεί να κωδικοποιηθεί σε συστήματα με διαφορετικές βάσεις, για παράδειγμα ο αριθμός 7 γράφεται 111_2 στο δυαδικό, 13_4 με βάση 4, κλπ. Γενικά χρειάζονται $\lceil \log_i n \rceil, i \geq 2$ για να αναπαραστήσουμε έναν ακέραιο n σε σύστημα με βάση $i \geq 2$. Από τον τύπο αλλαγής βάσης για τους λογαρίθμους $[\log_i n = \frac{\log_{i'} n}{\log_{i'} i}]$ παρατηρούμε ότι ο παρονομαστής είναι μια σταθερά, δεδομένων των i και i'. Με άλλα λόγια για την κωδικοποίηση ενός ακεραίου n χρειάζονται $\mathcal{O}(\log_2 n)$ σύμβολα. Επειδή λοιπόν οι υπολογιστές χρησιμοποιούν το δυαδικό σύστημα, και αφού δείξαμε ότι όλες οι κωδικοποιήσεις με βάση τουλάχιστον 2 έχουν την ίδια τάξη μεγέθους, θα χρησιμοποιούμε ως βάση το 2 για τους λογάριθμους.

Από την άλλη πλευρά, αν διαλέξουμε την αναπαράσταση με βάση 1 (unary representation) για τους ακέραιους, θα χρειαστούμε $n + 1$ σύμβολα (μονάδες) για την αναπαράσταση του ακεραίου n $(= \mathcal{O}(n))$. Προφανώς αυτή η κωδικοποίηση έχει μεγαλύτερη τάξη μεγέθους από τις κωδικοποιήσεις σε άλλες βάσεις. Συγκεκριμένα χρειάζονται εκθετικά περισσότερα ψηφία για την κωδικοποίηση σε βάση 1, από ότι για τις άλλες βάσεις.

Έτσι λοιπόν, το μέγεθος της εισόδου εξαρτάται από την κωδικοποίηση, ωστόσο οι διάφορες κωδικοποιήσεις της (εκτός αυτής με βάση το 1) διαφέρουν μεταξύ τους μόνο κατά μια πολλαπλασιαστική σταθερά, η οποία μπορεί να ενσωματωθεί στις σταθερές των ορισμών 1.5. Γενικά λοιπόν θα αποφεύγουμε την κωδικοποίηση με βάση 1 και όπου δεν αναφέρεται ρητά ποια κωδικοποίηση χρησιμοποιούμε, θα εννοείται ότι χρησιμοποιούμε κωδικοποίηση με βάση 2.

Τέλος αναφέρουμε ότι, όπως θα δούμε παρακάτω, πολλές φορές μπορούμε να χαλαρώσουμε ακόμη περισσότερο τους περιορισμούς για το ποια είναι μια θεμιτή κωδικοποίηση. Δηλαδή θα επιτρέπουμε η μια κωδικοποίηση να έχει πολυωνυμικά μεγαλύτερο μέγεθος από την άλλη. Έστω, για παράδειγμα ότι μια κωδικοποίηση για ένα πρόβλημα έχει μέγεθος n, τότε μια κωδικοποίηση που έχει μέγεθος $p(n)$, όπου p ένα πολυώνυμο, είναι αποδεκτή.

Στη συνέχεια θα μιλήσουμε εν συντομία για *τυπικές γλώσσες* γιατί θα μας βοηθήσουν να ορίσουμε κλάσεις πολυπλοκότητας και βοηθούν στη διαισθητική κατανόηση εννοιών.

Δεδομένου ενός αλφαβήτου Σ, το οποίο θεωρούμε χωρίς βλάβη της γενικότητας ότι είναι το $\Sigma = 0, 1$, συμβολίζουμε με Σ^* το σύνολο των πεπερασμένων ακολουθιών (*λέξεις*) $\alpha_1\alpha_2 \ldots a_n$ με $\alpha_i \in \Sigma$. Με $|w|$ συμβολίζουμε το μήκος μιας λέξης $w \in \Sigma^*$. *Τυπική γλώσσα* ονομάζεται ένα υποσύνολο Α του $\Sigma^*(A \subseteq \Sigma^*)$.

Επειδή υπάρχει αυτή η έννοια του υποσυνόλου, πολύ συχνά θα λέμε ότι το *σύνολο A* ανήκει σε κάποια κλάση και αυτό θα είναι το ίδιο με το να λέμε ότι *η γλώσσα A* ανήκει σε αυτή την κλάση.

Ορισμός 1.6. *P είναι η κλάση των γλωσσών που έχουν αλγόριθμο που αποφασίζει αν ένα string ανήκει στη γλώσσα ή όχι σε αριθμό βημάτων που φράσσεται από κάποιο πολυώνυμο ως προς την είσοδο. Λέμε ότι ο αλγόριθμος είναι πολυωνυμικού χρόνου.*

Ανάλογα, FP είναι η κλάση των συναρτήσεων που μπορούν να υπολογιστούν με αλγόριθμο πολυωνυμικού χρόνου.

Ακολούθως ορίζουμε την κλάση NP η οποία διαισθητικά είναι η κλάση των προβλημάτων απόφασης για τα οποία δεδομένης μιας λύσης, αυτή μπορεί να επαληθευτεί ή να απορριφθεί σε πολυωνυμικό χρόνο.

Ορισμός 1.7. *Μια γλώσσα L ανήκει στην κλάση NP αν και μόνο αν υπάρχει γλώσσα A στο P και ένα πολυώνυμο p έτσι ώστε $\forall x \in \Sigma^*, x \in L \iff \exists y, |y| \leq p(|x|) \wedge (x, y) \in A$*

Από τους παραπάνω ορισμούς προκύπτει ότι $P \subseteq NP$. Το αν το P είναι αυστηρώς μικρότερο από το NP είναι ίσως το μεγαλύτερο ανοιχτό πρόβλημα στη θεωρητική πληροφορική. Εικάζεται πάντως ότι $P \neq NP$.

Για τον ορισμό της πληρότητας που θα δώσουμε στη συνέχεια θα χρειαστούμε την έννοια της αναγωγής. Παραδείγματα ειδών αναγωγής μπορούμε να δούμε στους ορισμούς 2.1 και 2.6. Εδώ δίνουμε έναν γενικό ορισμό για το τι είναι αναγωγή.

Ορισμός 1.8. *Έστω σύνολα A και B. Λέμε ότι το A ανάγεται στο B κατά $\mathcal{R}$ με περιορισμό χρόνου $\mathcal{T}$ και γράφουμε $A \leq_{\mathcal{R}}^{\mathcal{T}} B$ αν, δεδομένου ενός αλγορίθμου που λύνει το B σε σταθερό χρόνο ($\mathcal{O}$), υπάρχει αλγόριθμος που ακολουθεί τους κανόνες που υπαγορεύει η $\mathcal{R}$ έτσι ώστε για κάθε x να αποφασίζει αν $x \in A$ σε χρόνο που φράσσεται από το $\mathcal{T}$.*

Στη συνέχεια δίνουμε την έννοια της πληρότητας για μια κλάση πολυπλοκότητας. Διαισθητικά ένα πρόβλημα το οποίο είναι πλήρες σε μια κλάση είναι από τα πιο δύσκολα προβλήματα της κλάσης.

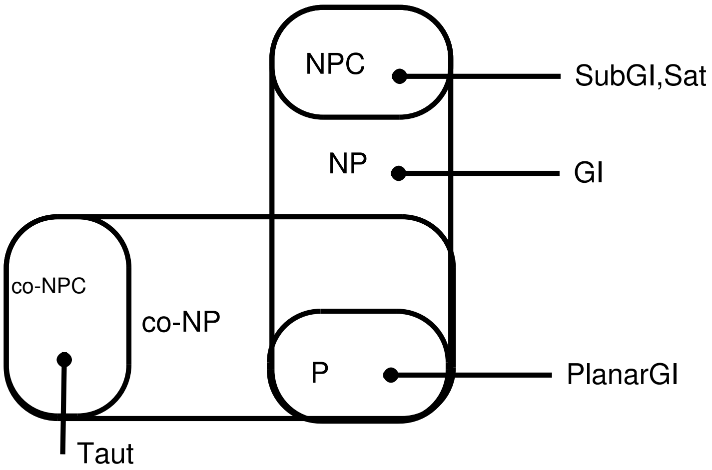
Ορισμός 1.9. *Έστω $\mathcal{C}$ μια κλάση πολυπλοκότητας και r ένα είδος αναγωγής. Ένα σύνολο $A \subseteq \Sigma^*$ είναι πλήρες για την κλάση $\mathcal{C}$ ως προς την αναγωγή r αν $A \in \mathcal{C}$ και κάθε σύνολο $B \in \mathcal{C}$ ανάγεται κατά r στο A.*

Ένα ενδιαφέρον πρόβλημα που σχετίζεται με τον ισομορφισμό γράφων και είναι NP-Complete [Kar72] είναι το πρόβλημα ισομορφισμού με υπογράφο.

Ορισμός 1.10. SubGI: *Δεδομένων δύο γράφων G και H το πρόβλημα είναι αν υπάρχει υπογράφος του H που να είναι ισομορφικός με τον G.*

Μέχρι σήμερα, σχεδόν όλα τα "φυσικά" προβλήματα που ανήκουν στο NP έχουν κατηγοριοποιηθεί είτε ως P είτε ως NP-Complete. Είναι ακόμη γνωστό ότι αν $P \neq NP$, υπάρχουν προβλήματα ενδιάμεσης πολυπλοκότητας τα οποία δεν ανήκουν ούτε στο P, ούτε είναι NP-Complete. Καλοί υποψήφιοι αυτής της κατηγορίας φαίνεται να είναι τα **GI** και **GA** (ο ισομορφισμός γράφων και ο αυτομορφισμός γράφου, προβλήματα που ορίζουμε στην επόμενη ενότητα, βλ. ορισμούς 1.12 και 1.13) μιας και μέχρι στιγμής δεν έχουν καταχωρηθεί ούτε στο P ούτε ως NP-Complete. Βέβαια το ίδιο ίσχυε και πριν μερικά χρόνια για το πρόβλημα αν κάποιος ακέραιος είναι πρώτος (*primality*) όμως η κατάσταση για το πρόβλημα αυτό ήταν πολύ διαφορετική, μιας και ανήκει στο $NP \cap co-NP$ και μάλιστα στην κλάση πολυπλοκότητας ZPP. Αντίθετα τα **GI** και **GA**, που ανήκουν προφανώς στο NP, δεν ξέρουμε αν ανήκουν στο co-NP.

Η κλάση των γλωσσών που το συμπλήρωμά τους ανήκει στο NP ονομάζεται co-NP. Όπως και για την κλάση NP υπάρχουν τα NP-Complete, έτσι και για την co-NP υπάρχουν τα (co-NP)-Complete. Ένα τέτοιο είναι το **Taut**, το σύνολο των ταυτολογιών στην προτασιακή λογική. Όπως πιστεύεται ότι τα NP-Complete προβλήματα δεν ανήκουν στο P, έτσι πιστεύεται ότι δεν ανήκουν και στο co-NP (αντίστοιχα δεν πιστεύεται ότι τα (co-NP)-Complete προβλήματα είναι στο NP). Ισχύει ότι αν κάποιο NP-Complete πρόβλημα ανήκει στο co-NP τότε $NP = co-NP$, όμως εικάζεται ότι κάτι τέτοιο δεν ισχύει. Τέλος ισχύει ότι $P \subseteq NP \cap co-NP$ και μάλιστα εικάζεται ότι το P είναι γνήσιο υποσύνολο του $NP \cap co-NP$.



Σχήμα 1.3: Διάγραμμα Κλάσεων

Τέλος ορίζουμε συνοπτικά την έννοια της μηχανής Turing (Turing Machine) γιατί είναι ένα ευρέως διαδεδομένο υπολογιστικό μοντέλο το ποίο συχνά χρησιμοποιείται και για ορισμούς κλάσεων, σε διάφορες παραλλαγές του.

Ορισμός 1.11. *[Pap94] Μια μηχανή Turing είναι μια τετράδα $M = (K, \Sigma, \delta, s))$. Το K είναι ένα πεπερασμένο σύνολο καταστάσεων και $s \in K$ είναι η αρχική κατάσταση. Το Σ είναι ένα πεπερασμένο σύνολο συμβόλων, το αλφάβητο της μηχανής. Υποθέτουμε ότι τα K και Σ είναι ξένα μεταξύ τους σύνολα. Το Σ περιέχει πάντα τα ειδικά σύμβολα $\sqcup$ και $\triangleright$: το σύμβολο του κενού και το πρώτο σύμβολο. Τέλος η δ είναι μια συνάρτηση από το $K \times \Sigma$ στο $(K \cup \{h, \text{“yes”}, \text{“no”}\}) \times \Sigma \times \{\leftarrow, \rightarrow, -\}$. Υποθέτουμε ότι h (η κατάσταση τερματισμού = halting state), “yes” (η κατάσταση αποδοχής), “no” (η κατάσταση απόρριψης) και οι κατευθύνσεις της κεφαλής της μηχανής ($\leftarrow$, $\rightarrow$ και $-$) δεν περιέχονται στο $K \cup \Sigma$.*

Η συνάρτηση δ είναι το “πρόγραμμα” της μηχανής. Θεωρούμε ότι η μηχανή δέχεται σαν είσοδο μια ακολουθία συμβόλων από το αλφάβητό της Σ, τρέχει το πρόγραμμα δ για να δώσει τελικά μια απάντηση “yes” ή “no” ή να τερματίσει χωρίς να δώσει απάντηση (h). Η μηχανή Turing όπως ορίστηκε είναι γνωστή και ως ντετερμινιστική μηχανή Turing (deterministic Turing Machine). Ονομάζεται ντετερμινιστική γιατί σε κάθε στιγμή η παρούσα κατάσταση και το σύμβολο που διαβάζει η κεφαλή της μηχανής προσδιορίζουν μονοσήμαντα ποια θα είναι η συμπεριφορά της μηχανής (η δ είναι συνάρτηση). Αν επεκτείνουμε τη δ στη σχέση $\Delta \subset (K \times \Sigma) \times [(K \cup \{h, \text{"}yes\text{"}, \text{"}no\text{"}, \}) \times \Sigma \times \{\leftarrow, \rightarrow, -\}]$ τότε έχουμε μια μη-ντετερμινιστική μηχανή Turing. Ο μη ντετερμινισμός προκύπτει από το γεγονός ότι από μια κατάσταση και για ένα σύμβολο από το αλφάβητο η μηχανή μπορεί δυνητικά να διαλέξει μεταξύ περισσότερων ενεργειών.

Με βάση τους ορισμούς που δώσαμε για τις μηχανές Turing μπορούμε να ορίσουμε τις κλάσεις P και NP ως τις κλάσεις των προβλημάτων απόφασης που λύνονται σε πολυωνυμικό χρόνο από μια ντετερμινιστική και από μια μη-ντετερμινιστική μηχανή Turing αντίστοιχα.

1.3 Ισομορφισμός Γράφων (GI)

Για το πρόβλημα απόφασης του ισομορφισμού γράφων έχουμε τον ακόλουθο ορισμό

Ορισμός 1.12. Ισομορφισμός Γράφων (GI): *Δεδομένων δύο γράφων G_1 και G_2, είναι αυτοί οι γράφοι ισομορφικοί;*

Το σύνολο των ισομορφισμών (συναρτήσεων απεικόνισης) δύο γράφων G_1 και G_2 συμβολίζεται με $Iso(G_1, G_2)$.

Είναι προφανές ότι το GI ανήκει στο NP μιας και δεδομένου μιας συνάρτησης απεικόνισης είναι εύκολο να ελέγξει κανείς αν πρόκειται για ισομορφισμό.

Το ανοιχτό πρόβλημα είναι αν ανήκει στο P ή αν είναι NP-Complete(ή αν ανήκει σε κάποια ενδιάμεση κλάση). Έχει αποδειχθεί [Lad75] ότι αν $P \neq NP$, υπάρχουν προβλήματα που δεν είναι ούτε στο P ούτε είναι NP-Complete. Το GI είναι ένας καλός υποψήφιος για μια τέτοια ενδιάμεση κλάση, μιας και ούτε πολυωνυμικός αλγόριθμος έχει βρεθεί, ούτε φαίνεται να είναι αρκετά δύσκολο για να είναι NP-Complete.

Ένα παραπλήσιο πρόβλημα που θα μας απασχολήσει παράλληλα με το GI είναι ο αυτομορφισμός γράφων (GA). Έστω S_n το σύνολο των μεταθέσεων n στοιχείων. Ένας αυτομορφισμός για το γράφο G με n κόμβους είναι μια μετάθεση $\varphi \in S_n$ τέτοια ώστε για κάθε ζεύγος κορυφών i, j του γράφου να ισχύει $(i, j) \in E \iff (\varphi(i), \varphi(j)) \in E$. (Με άλλα λόγια $\varphi(G) = G$). Προφανώς κάθε γράφος έχει έναν τετριμμένο αυτομορφισμό που ορίζεται από την ταυτοτική μετάθεση και καλείται *ταυτοτικός αυτομορφισμός*(id).

Ορισμός 1.13. Αυτομορφισμός Γράφου (GA): *Δεδομένου ενός γράφου G, αποφάσισε αν υπάρχει μη-τετριμμένος αυτομορφισμός αυτού.*

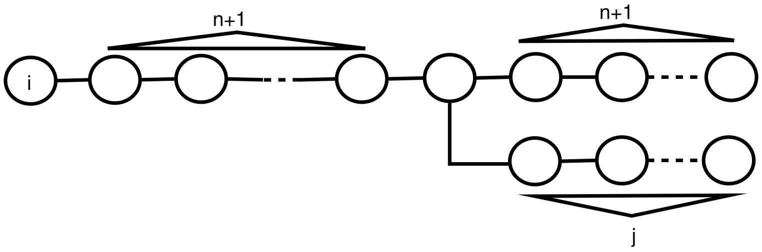
Το σύνολο των αυτομορφισμών ενός γράφου G είναι μια υποομάδα του S_n που τη συμβολίζουμε με $Aut(G)$. Είναι προφανές από τους παραπάνω ορισμούς ότι $Aut(G) \equiv Iso(G, G)$.

Ορίζουμε $\#\mathbf{GI}(G_1, G_2) = |Iso(G_1, G_2)|$ και $\#\mathbf{GA}(G) = |Aut(G)|$.

Έχουμε:

$$\#\mathbf{GI}(G_1, G_2) = \begin{cases} 0, & \text{if} G_1 \not\equiv G_2 \\ \#\mathbf{GA}(G_1) = \#\mathbf{GA}(G_2), & \text{if} G_1 \equiv G_2 \end{cases}$$

Πολλές φορές θέλουμε να βάλουμε μια ετικέτα j σε μια κορυφή i ενός γράφου G έτσι ώστε να την διαφοροποιήσουμε από τις άλλες κορυφές. Αυτό μπορούμε να το κάνουμε προσδένοντας στην κορυφή i ένα ειδικό γράφημα το οποίο να εξαρτάται και να προσδιορίζει το j. Έστω για παράδειγμα ότι ο γράφος G έχει n κορυφές. Η ετικέτα j μπορεί να είναι ένας γράφος $2n + j + 3$ κορυφών όπως φαίνεται στο σχήμα 1.4.



Σχήμα 1.4: Παράδειγμα ετικέτας

Σκοπός της αυτής είναι κάθε αυτομορφισμός του γράφου με ετικέτα να απεικονίζει το i στον εαυτό του και να μην δημιουργούνται επιπλέον αυτομορφισμοί που δεν υπήρχαν στο αρχικό γράφο.

Συμβολίζουμε με $G_{[i_1,\ldots,i_k]}$ το γράφο που προκύπτει από τον G στον οποίο ο κόμβος i_m έχει λάβει ετικέτα m. Παρατηρούμε ότι στους γράφους $G_{[i]}$ και $H_{[j]}$, όπου οι G και H έχουν n κορυφές, η κορυφή i του G λαμβάνει την ίδια ετικέτα (την 1) με την κορυφή j του H. Επομένως, αν οι γράφοι $G_{[i]}$ και $H_{[j]}$ είναι ισομορφικοί, το ίδιο θα ισχύει και για τους G και H και μάλιστα υπάρχει ισομορφισμός που απεικονίζει την κορυφή i του G στην κορυφή j του H.

Ένας ακόμα μετασχηματισμός γράφων που θα χρησιμοποιήσουμε είναι η *ένωση γράφων*. Δεδομένων δύο γράφων $G_1 = (V_1, E_1)$ και $G_2 = (V_2, E_2)$ όπου οι κορυφές V_1, V_2 αριθμούνται εξ' αρχής έτσι ώστε $V_1 \cap V_2 = \emptyset$. Συμβολίζουμε με $G_1 \cup G_2 = (V_1 \cup V_2, E_1 \cup E_2)$ τον γράφο που προκύπτει. Παρατηρούμε ότι αν και οι δύο γράφοι G_1 και G_2 είναι συνεκτικοί (αποτελούνται από μια συνεκτική συνιστώσα) τότε ένας αυτομορφισμός που εναλλάσσει τις κορυφές V_1 και V_2 μπορεί να θεωρηθεί ως ο συνδυασμός ενός ισομορφισμού από τον G_1 στον G_2, και ενός ισομορφισμού από τον G_2 στον G_1. Επομένως

$$\#\mathbf{GA}(G_1 \cup G_2) = \#\mathbf{GA}(G_1) \cdot \#\mathbf{GA}(G_2) + (\#\mathbf{GI}(G_1, G_2))^2$$

ή αλλιώς

$$\#\mathbf{GA}(G_1 \cup G_2) = \begin{cases} \#\mathbf{GA}(G_1) \cdot \#\mathbf{GA}(G_2), & \text{if } G_1 \not\equiv G_2 \\ 2 \cdot \#\mathbf{GA}(G_1) \cdot \#\mathbf{GA}(G_2), & \text{if } G_1 \equiv G_2 \end{cases}$$

1.4 Ορισμοί Κλάσεων

Ορισμός 1.14. NC:Nick's Class *Πήρε το όνομά της από τον Nick Pippenger και είναι η ένωση των κλάσεων NC_i για $i \geq 0$, δηλαδή*

$$NC = \bigcup_{i \geq 0} NC_i$$

.

Η κλάση αυτή είναι κυρίως γνωστή ως *παράλληλη κλάση* και ορίζεται με αντίστοιχο τρόπο. Εδώ θα δώσουμε έναν ορισμό που μιλά για κυκλώματα γιατί πολλές φορές θα θέλουμε να έχουμε μια τέτοια διαίσθηση και σε κάποιες αποδείξεις θα χρησιμοποιήσουμε την έννοια του κυκλώματος. Δεν έχουμε όμως ορίσει την κλάση NC_i που χρησιμοποιήσαμε παραπάνω.

Ορισμός 1.15. *NC_i είναι η κλάση των προβλημάτων απόφασης που λύνονται από ομογενή οικογένεια λογικών κυκλωμάτων με πολυωνυμικό μέγεθος, βάθος $\mathcal{O}(\log^i(n))$, και fan-in 2 (πλήθος εισόδων ανά στοιχείο).*

Με αντίστοιχο τρόπο ορίζονται οι κλάσεις AC και AC_i με τη διαφορά ότι δεν έχουμε περιορισμό στον αριθμό των εισόδων ανά στοιχείο (unbounded fan-in).

Ορισμός 1.16. *AC_i είναι η κλάση των προβλημάτων απόφασης που λύνονται από ομογενή οικογένεια λογικών κυκλωμάτων με πολυωνυμικό μέγεθος, βάθος $\mathcal{O}(\log^i(n))$, και απεριόριστο fan-in (πλήθος εισόδων ανά στοιχείο).*

Ορισμός 1.17. *AC είναι η ένωση των κλάσεων AC_i για $i \geq 0$, δηλαδή*

$$AC = \bigcup_{i \geq 0} AC_i$$

Από τον ορισμό είναι προφανές ότι θα ισχύει $NC_i \subseteq AC_i$. Το ενδιαφέρον είναι ότι παρότι η AC_i είναι πράγματι πιο μεγάλη από την NC_i δεν είναι πολύ μεγαλύτερη γιατί ισχύει ότι $AC_i \subseteq NC_{i+1}$. Από αυτό το αποτέλεσμα προκύπτει και ότι $NC = AC$. Και οι δύο κλάσεις περιέχουν την κλάση NL που θα ορίσουμε στη συνέχεια. Επίσης, και οι δύο κλάσεις έχουν παρόμοια συμπεριφορά σε τυχαία μαντεία.Συγκεκριμένα, για τυχαίο μαντείο A η NC_i^A περιέχεται γνήσια στην NC_{i+1}^A και η NC^A περιέχεται γνήσια στην P^A με πιθανότητα 1. Ομοίως, για τυχαίο μαντείο A η AC_i^A περιέχεται γνήσια στην AC_{i+1}^A και η AC^A περιέχεται γνήσια στην P^A με πιθανότητα 1.

Τέλος η NL περιέχεται και στην NC_2 που συνήθως συμβολίζουμε με NC^2 ενώ η NC_1 (που θα συμβολίζουμε με NC^1) περιέχεται στην L που ορίζουμε παρακάτω.

Ορισμός 1.18. L: Logarithmic Space
Είναι η κλάση των προβλημάτων απόφασης που λύνονται από ντετερμινιστική μηχανή Turing με περιορισμό η μνήμη που χρησιμοποιεί να είναι λογαριθμική ($\mathcal{O}(\log n)$) σε σχέση με την είσοδο. Η είσοδος δεν προσμετράται στο χώρο της μνήμης.

Ορισμός 1.19. NL: Nondeterministic Logarithmic-Space
Έχει την ίδια σχέση με την L, που έχει η NP με την P. Είναι δηλαδή η κλάση των προβλημάτων απόφασης που λύνονται από μη-ντετερμινιστική μηχανή Turing με λογαριθμική μνήμη σε σχέση με την είσοδο.

Προφανώς ισχύει $L \subseteq NL$.

Ορισμός 1.20. SL: Symmetric Logarithmic-Space
Είναι η κλάση των προβλημάτων που λύνονται από μια μη-ντετερμινιστική μηχανή Turing με λογαριθμική μνήμη (όπως για την L), έτσι ώστε:

1. *Αν η απάντηση είναι 'ναι', ένα ή περισσότερα υπολογιστικά μονοπάτια κάνουν την είσοδο αποδεκτή.*

2. *Αν η απάντηση είναι 'όχι', όλα τα μονοπάτια κάνουν την είσοδο μη-αποδεκτή.*

3. *Αν η μηχανή μπορεί να κάνει μη-ντετερμινιστική μετάβαση από μια κατάσταση A σε μια κατάσταση B, τότε μπορεί να κάνει επίσης τη μετάβαση από την κατάσταση B στην κατάσταση A. (Αυτή η προυπόθεση διαφοροποιεί την SL από την NL και την κάνει 'πιό συμμετρική'.)*

Από τον ορισμό προκύπτει με προφανή τρόπο ότι $L \subseteq SL \subseteq NL$.
Σε συνδυασμό με τα προηγούμενα αποτελέσματα έχουμε :

$$\mathbf{AC}^0 \subseteq \mathbf{NC}^1 \subseteq \mathbf{L} \subseteq \mathbf{SL} \subseteq \mathbf{NL} \subseteq \mathbf{NC}^2$$

Κεφάλαιο 2

Προβλήματα Απόφασης, Αναζήτησης και Αρίθμησης

2.1 Ορισμοί

Μέχρι στιγμής έχουμε εισάγει το πρόβλημα απόφασης **GI** στον ορισμό 1.12. Η απάντηση που λαμβάνουμε μπορεί να είναι 'ναι' ή 'όχι'. Ανάγεται ουσιαστικά σε ερώτηση αν ένα στιγμιότυπο ανήκει στο σύνολο των στιγμιότυπων με απάντηση 'ναι' σε κάποια ερώτηση. Πολλές φορές αυτό δεν είναι ικανοποιητικό γιατί δεν μας αρκεί να γνωρίζουμε ότι υπάρχει ισομορφισμός μεταξύ δύο γράφων, θέλουμε να μας δοθεί και ένας ισομορφισμός. Αυτή είναι η περίπτωση του προβλήματος *αναζήτησης* (search problem) ισομορφισμού γράφων.

Μια άλλη περίπτωση είναι τα προβλήματα *απαρίθμησης* (counting problems), στα οποία ζητούμενο είναι το πόσες λύσεις έχει το πρόβλημα που μας απασχολεί. Για παράδειγμα, για τον ισομορφισμό γράφων το πρόβλημα απαρίθμησης απαντά στο ερώτημα πόσοι διαφορετικοί ισομορφισμοί υπάρχουν μεταξύ δύο γράφων. Έχουμε ήδη εισάγει τον συμβολισμό $\#\mathbf{GI}(G_1, G_2)$ για τον αριθμό αυτό. Είναι προφανές από τη διατύπωση ότι το αποτέλεσμα θα είναι ένας ακέραιος μεγαλύτερος ή ίσος με το μηδέν. Επίσης προφανές είναι το γεγονός ότι το πρόβλημα απόφασης δεν είναι πιο 'δύσκολο' από το πρόβλημα απαρίθμησης αφού μπορεί να προσομοιωθεί με μια εκτέλεση του προβλήματος απαρίθμησης και αν το αποτέλεσμα είναι μηδέν η απάντηση είναι 'όχι', αλλιώς, αν το αποτέλεσμα είναι μεγαλύτερο του μηδενός, η απάντηση είναι 'ναι'. Το αντίθετο δεν είναι προφανές, αν δηλαδή το πρόβλημα απαρίθμησης είναι το πολύ το ίδιο δύσκολο με το πρόβλημα απόφασης, στην οποία περίπτωση τα δύο προβλήματα είναι της ιδίας δυσκολίας. Με αυτό το ζήτημα θα ασχοληθούμε αργότερα.

Το πρόβλημα απόφασης είναι επίσης το πολύ το ίδιο δύσκολο με το πρόβλημα αναζήτησης αφού μπορεί εύκολα να προσομοιωθεί με μια εκτέλεση του αλγόριθμου για το πρόβλημα αναζήτησης. Αν ο αλγόριθμος απαντήσει ότι δεν υπάρχει ισομορφισμός, το πρόβλημα απόφασης απαντά με 'όχι', ενώ αν δώσει έναν ισομορφισμό, το πρόβλημα απόφασης απαντά με 'ναι'. Και σε αυτή την περίπτωση δεν είναι προφανές το αντίστροφο, αν δηλαδή το πρόβλημα αναζήτησης είναι το πολύ το ίδιο δύσκολο με το πρόβλημα απόφασης.

Αυτό που κάναμε και στις δύο παραπάνω περιπτώσεις ήταν να ανάγουμε το πρόβλημα απόφασης στα προβλήματα αναζήτησης και απαρίθμησης για τον ισομορφισμό γράφων. Διαισθητικά ένα πρόβλημα A λέμε ότι ανάγεται σε ένα πρόβλημα B αν ένας αλγόριθμος για το πρόβλημα B μπορεί να χρησιμοποιηθεί, με περιορισμούς που θα οριστούν στη συνέχεια, για τη δημιουργία ενός αλγορίθμου για το πρόβλημα A.

2.2 Αναγωγές

Θα μας απασχολήσουν αρχικά δύο ειδών αναγωγές, η αναγωγή κατά Karp (γνωστή και ως *many-one reduction*) και η αναγωγή κατά Cook (γνωστή και ως αναγωγή Turing).

Ορισμός 2.1. *Έστω ότι $A, B \subseteq \Sigma^*$. Λέμε ότι η γλώσσα A ανάγεται κατά Karp στην B ($A \leq_m^p B$), αν υπάρχει μια συνάρτηση $f : \Sigma^* \to \Sigma^*$ υπολογιστή σε πολυωνυμικό χρόνο, τέτοια ώστε για κάθε λέξη $x \in \Sigma^*$ να ισχύει $x \in A \iff f(x) \in B$.*

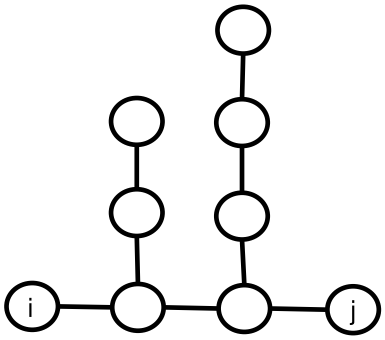
Αν ισχύουν ταυτόχρονα $A \leq_m^p B$ και $B \leq_m^p A$ λέμε ότι οι A και B είναι ισοδύναμες κατά Karp και γράφουμε $A \equiv_m^p B$.

Για να δώσουμε ένα παράδειγμα αναγωγής, θα ορίσουμε μια εκδοχή του ισομορφισμού για κατευθυνόμενους γράφους (**DirGI**). Από τη μία είναι προφανές ότι $\mathbf{GI} \leq_m^p \mathbf{DirGI}$: κάθε μη κατευθυνόμενη πλευρά αντικαθίσταται από ένα ζευγάρι πλευρών που έχουν αντίθετες κατευθύνσεις.

Η άλλη κατεύθυνση δεν είναι τόσο προφανής. Θα αποδείξουμε λοιπόν ότι $\mathbf{DirGI} \leq_m^p \mathbf{GI}$ και επομένως, λόγω του παραπάνω αποτελέσματος, $\mathbf{DirGI} \equiv_m^p \mathbf{GI}$. Αυτό το αποτέλεσμα δικαιολογεί τον γεγονός ότι εστιάζουμε την προσοχή μας σε μη κατευθυνόμενους γράφους.

Απόδειξη.

Δεδομένων δύο κατευθυνόμενων γράφων G και H κατασκευάζουμε δύο μη κατευθυνόμενους γράφους G' και H' ξεκινώντας από το ίδιο σύνολο κορυφών. Για κάθε κατευθυνόμενη πλευρά του αρχικού γράφου, έστω από τον κόμβο i στον κόμβο j, εισάγουμε 7 νέες κορυφές και τις συνδέουμε με τις κορυφές i και j με μη κατευθυνόμενες πλευρές, όπως φαίνεται στο σχήμα 2.1.



Σχήμα 2.1: Αντικατάσταση κατευθυνόμενης πλευράς

Επειδή οποιοσδήποτε ισομορφισμός μεταξύ των G' και H' απεικονίζει τους υπογράφους 7 κόμβων που κατασκευάσαμε με την αναγωγή σε υπογράφους του ιδίου είδους, εύκολα επιβεβαιώνει κανείς ότι ισχύει $G \equiv H \iff G' \equiv H'$. Συνεπώς η κατασκευή (συνάρτηση) $(G, H) \longmapsto (G', H')$ είναι μια αναγωγή Karp από το **DirGI** στο **GI**. □

Στη συνέχεια θα ασχοληθούμε με ένα άλλο, πάλι πιο γενικό πρόβλημα ισομορφισμού, το οποίο θα δείξουμε ότι ανάγεται στο **GI**. Πρόκειται για την περίπτωση που επιτρέπουμε σε κορυφές και πλευρές να έχουν κάποιο χρώμα, ή ετικέτα, ή βάρος.

Ορισμός 2.2. *Ο ισομορφισμός χρωματισμένων γράφων (***ColGI***) είναι το πρόβλημα απόφασης που αποφαίνεται αν δεδομένων δύο γράφων G_1 και G_2 υπάρχει ισομορφισμός που να διατηρεί επιπλέον και τα χρώματα των κόμβων και των πλευρών. Συμβολικά, έστω $v, u \in V$ και $(u, v) \in E$ για κάποιο χρωματισμένο γράφο $G(V, E)$ τότε με $\mathcal{C}(v)$ συμβολίζουμε το χρώμα της κορυφής v και με $\mathcal{C}(u, v)$ το χρώμα της πλευράς (u, v). Ο ισομορφισμός είναι μια ένα προς ένα και επί συνάρτηση $\varphi : V_1 \to V_2$ έτσι ώστε :*

$$(v, u) \in E_1 \iff (\varphi(v), \varphi(u)) \in E_2$$

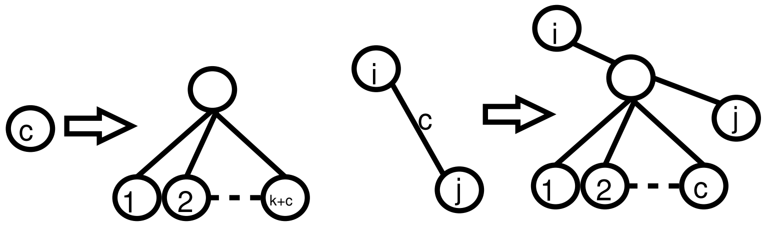
και

$$\mathcal{C}(v) = \mathcal{C}(\varphi(v)) \wedge \mathcal{C}(u) = \mathcal{C}(\varphi(u)) \wedge \mathcal{C}(v, u) = \mathcal{C}((\varphi(v), \varphi(u)))$$

Είναι προφανές ότι ο ισομορφισμός χρωματισμένων γράφων είναι μια πιο γενική περίπτωση του ισομορφισμού γράφων και επομένως **GI** $\leq_m^p$ **ColGI** (απλά θεωρούμε ότι ο γράφος έχει κοινό χρώμα σε όλες τις κορυφές και τις πλευρές και ζητάμε να λύσουμε ένα **ColGI** πρόβλημα). Το αντίθετο δεν είναι προφανές, αλλά ισχύει από ότι μας λέει το θεώρημα 2.3 παρακάτω.

Θεώρημα 2.3. *Το* **ColGI** *ανάγεται κατά Karp στο* **GI** *(***ColGI** $\leq_m^p$ **GI***).*

Απόδειξη. Συμβολίζουμε με $\mathcal{C}_V(G)$ το σύνολο των χρωμάτων των κορυφών του γράφου G και με $\mathcal{C}_E(G)$ το σύνολο των χρωμάτων των πλευρών του G Έστω ότι έχουμε δύο χρωματισμένους γράφους G_1 και G_2. Θεωρούμε διαφορετικά τα χρώματα για τις πλευρές και τις κορυφές και υποθέτουμε ότι οι δύο γράφοι έχουν τα ίδια σύνολα χρωμάτων για κορυφές και πλευρές αντίστοιχα, δηλαδή $\mathcal{C}_V(G_1) \equiv \mathcal{C}_V(G_2)$ και $\mathcal{C}_E(G_1) \equiv \mathcal{C}_E(G_2)$. Αριθμούμε αρχικά τα χρώματα των πλευρών και έστω k το πλήθος τους. Σημειώνουμε ότι το k δεν μπορεί να είναι πάνω από $\binom{n}{2} = \frac{n\cdot(n-1)}{2}$. Στη συνέχεια αριθμούμε τα χρώματα των κορυφών και έστω m το πλήθος τους. Κατασκευάζουμε έναν γράφο χωρίς χρώματα από τον χρωματισμένο γράφο ως εξής:



Σχήμα 2.2: Μετατροπές κορυφών και πλευρών

- Κάθε χρωματισμένη κορυφή με αριθμό χρώματος c την αντικαθιστούμε με μια κορυφή με k+c παιδιά (βλ. σχήμα 2.2 αριστερά).
- Κάθε χρωματισμένη πλευρά με αριθμό χρώματος c την αντικαθιστούμε με μια κορυφή με c παιδιά η οποία ενώνεται με τις κορυφές που ένωνε η πλευρά που αντικαταστήσαμε (βλ. σχήμα 2.2 δεξιά).

Ο γράφος που προκύπτει έχει την ιδιότητα ότι όποια κορυφή έχει βαθμό 1, αποτελεί μέρος κωδικοποίησης χρώματος του αρχικού γράφου. Η κατασκευή είναι πολυωνυμική και οι δύο γράφοι που κατασκευάζουμε με τον παραπάνω τρόπο είναι ισομορφικοί αν και μόνο αν οι δύο αρχικοί χρωματισμένοι γράφοι είναι ισομορφικοί. □

Πόρισμα 2.4. *Τα* **GI** *και* **ColGI** *είναι ισοδύναμα υπό αναγωγή Karp. Συμβολικά:*

$$\mathbf{GI} \equiv_m^p \mathbf{ColGI}$$

Σαν τελευταία αναγωγή Karp θα δώσουμε αυτή από τον ισομορφισμό μεταξύ πολυγράφων (multigraphs) σε ισομορφισμό μεταξύ χρωματισμένων γράφων. Υπενθυμίζουμε ότι πολυγράφος είναι ένας γράφος που επιτρέπεται να έχει περισσότερες από μία πλευρές μεταξύ δύο κόμβων όπως επιτρέπεται να περιέχει βρόγχους, δηλαδή πλευρές που τα δύο άκρα τους προσπίπτουν στην ίδια κορυφή.

Η αναγωγή από τον ισομορφισμό πολυγράφων στον ισομορφισμό χρωματισμένων γράφων είναι πολύ απλή. Στις κορυφές που δεν έχουν βρόγχους βάζουμε ετικέτα (χρώμα) '0' ενώ στις κορυφές με k βρόγχους δίνουμε ετικέτα 'k' και αφαιρούμε τους βρόγχους. Αντίστοιχα αν μεταξύ δύο κόμβων υπάρχουν k πλευρές, τις αντικαθιστούμε με μια πλευρά με ετικέτα 'k'. Με αυτή την κατασκευή δυο πολυγράφοι είναι ισομορφικοί αν και μόνο αν οι αντίστοιχοι χρωματισμένοι γράφοι είναι ισομορφικοί.

Μια παρατήρηση που πρέπει να κάνουμε σε αυτό το σημείο είναι ότι ενώ στους απλούς γράφους μπορούμε να θεωρούμε σαν μέγεθος τον αριθμό των κορυφών n μιας και το πλήθος των πλευρών δεν θα ξεπερνά τις $\binom{n}{2} = \mathcal{O}(n^2)$, στους πολυγράφους δεν ισχύει κάτι τέτοιο και εν γένει το πλήθος των πλευρών δεν είναι φραγμένο. Έτσι για τους πολυγράφους θεωρούμε ότι το μέγεθος της αναπαράστασης είναι $\mathcal{O}(|V| + |E|)$. Με αυτή τη θεώρηση διαπιστώνουμε ότι η κατασκευή του χρωματισμένου γράφου από το πολυγράφο γίνεται σε πολυωνυμικό χρόνο.

Ορισμός 2.5. *Δίνουμε κάποιους συνοπτικούς ορισμούς προβλημάτων που προκύπτουν ως συνδυασμοί προβλημάτων που έχουμε συναντήσει.*

- **DirColGI**: *Το πρόβλημα απόφασης ισομορφισμού μεταξύ δύο κατευθυνόμενων χρωματισμένων γράφων.*
- **MultGI**: *Το πρόβλημα απόφασης ισομορφισμού μεταξύ δύο πολυγράφων.*
- **DirMultGI**: *Το πρόβλημα απόφασης ισομορφισμού μεταξύ δύο κατευθυνόμενων πολυγράφων.*
- **ColMultGI**: *Το πρόβλημα απόφασης ισομορφισμού μεταξύ δύο χρωματισμένων πολυγράφων.*
- **DirColMultGI**: *Το πρόβλημα απόφασης ισομορφισμού μεταξύ δύο κατευθυνόμενων χρωματισμένων γράφων.*
- **DirGA**: *Το πρόβλημα απόφασης ύπαρξης μη-τετριμμένου αυτομορφισμού σε κατευθυνόμενο γράφο.*
- **ColGA**: *Το πρόβλημα απόφασης ύπαρξης μη-τετριμμένου αυτομορφισμού σε χρωματισμένο γράφο.*
- **DirColGA**: *Το πρόβλημα απόφασης ύπαρξης μη-τετριμμένου αυτομορφισμού σε κατευθυνόμενο χρωματισμένο γράφο.*

- **MultGA**: *Το πρόβλημα απόφασης ύπαρξης μη-τετριμμένου αυτομορφισμού σε πολυγράφο.*
- **DirMultGA**: *Το πρόβλημα απόφασης ύπαρξης μη-τετριμμένου αυτομορφισμού σε κατευθυνόμενο πολυγράφο.*
- **ColMultGA**: *Το πρόβλημα απόφασης ύπαρξης μη-τετριμμένου αυτομορφισμού σε χρωματισμένο πολυγράφο.*
- **DirColMultGA**: *Το πρόβλημα απόφασης ύπαρξης μη-τετριμμένου αυτομορφισμού σε κατευθυνόμενο χρωματισμένο πολυγράφο.*

Ισχύουν τα ακόλουθα δύο αποτελέσματα. Οι κατασκευές των αναγωγών είναι συνδυασμός των αναγωγών που έχουμε δώσει για κατευθυνόμενους και χρωματισμένους γράφους.

$\mathbf{GI} \equiv_m^p \mathbf{DirGI} \equiv_m^p \mathbf{ColGI} \equiv_m^p \mathbf{DirColGI}$ $\mathbf{GI} \equiv_m^p \mathbf{MultGI} \equiv_m^p \mathbf{DirMultGI} \equiv_m^p \mathbf{ColMultGI} \equiv_m^p \mathbf{DirColMultGI}$
$\mathbf{GA} \equiv_m^p \mathbf{DirGA} \equiv_m^p \mathbf{ColGA} \equiv_m^p \mathbf{DirColGA}$ $\mathbf{GA} \equiv_m^p \mathbf{MultGA} \equiv_m^p \mathbf{DirMultGA} \equiv_m^p \mathbf{ColMultGA} \equiv_m^p \mathbf{DirColMultGA}$

Διαπιστώνουμε ότι προβλήματα που φαίνονται αρχικά αρκετά πιο δύσκολα από το **GI** ανάγονται και μάλιστα κατά Karp σε αυτό. Έχει αποδειχθεί [Mil79] ότι πρόβλημα ισομορφισμού για οποιοδήποτε είδος πεπερασμένης αλγεβρικής δομής ανάγεται κατά Karp στο **GI**.

Στη συνέχεια δίνουμε τον ορισμό για μια πιο ισχυρή μορφή αναγωγής, την αναγωγή κατά Cook (γνωστή και ως Turing αναγωγή).

Ορισμός 2.6. *Έστω $A, B \subseteq \Sigma^*$ (το σύνολο των λέξεων επί ενός αλφαβήτου Σ), λέμε ότι το A* ανάγεται κατά Cook*(ή Turing) (σε πολυωνυμικό χρόνο) στο B και το συμβολίζουμε με $A \leq_T^p B$, αν υπάρχει αλγόριθμος για το A που χρειάζεται πολυωνυμικό χρόνο και μπορεί να έχει εντολές της μορφής*
if $y \in B$ **then** ... **else** ... **end**
Σχετικά με την πολυπλοκότητα ενός τέτοιου αλγορίθμου υποθέτουμε ότι το κόστος για την αποτίμηση της συνθήκης $y \in B$ είναι ένα μόνο βήμα.

Αν ισχύουν ταυτόχρονα $A \leq_T^p B$ και $B \leq_T^p A$ λέμε ότι οι A και B είναι ισοδύναμες κατά Cook (Turing) και γράφουμε $A \equiv_m^p B$.

Από τον ορισμό προκύπτει ότι $A \leq_m^p B \Rightarrow A \leq_T^p B$.

Πολλές φορές αναφερόμαστε στο πρόβλημα (γλώσσα) B ως μαντείο (oracle) μιας και υποθέτουμε ότι έχουμε μια εξωτερική 'δύναμη' (μαντείο) που μας απαντά σε ερωτήσεις $y \in B$ σε σταθερό χρόνο $\mathcal{O}(1)$.

Στη συνέχεια δείχνουμε ότι το πρόβλημα αυτομορφισμού γράφων (**GA**) ανάγεται κατά Cook στο πρόβλημα ισομορφισμού γράφων (**GI**). Υπενθυμίζουμε ότι με $G_{[i]}$ συμβολίζουμε ένα αντίγραφό του γράφου G με μια ετικέτα προσκολλημένη στον κόμβο i. Ο αλγόριθμος 2.1 υπολογίζει το **GA** σε πολυωνυμικό χρόνο, κάνοντας χρήση μαντείου για το **GI**.

Αλγόριθμος 2.1 GA $\leq_T^p$ GI

```
input graph G with n nodes;
for i := 1 to n − 1 do
  for j := i + 1 to n do
    if (G[i], G[j]) ∈ GI then
      accept
    end if
  end for
end for
reject
```

Προφανώς ο G έχει μη τετριμμένο αυτομορφισμό αν και μόνο αν έχει αυτομορφισμό που να απεικονίζει κάποιον κόμβο i σε κάποιον κόμβο j με $(i < j)$ ή ισοδύναμα, αν και μόνο αν οι γράφοι $G_{[i]}$ και $G_{[j]}$ είναι ισομορφικοί, μιας και από την κατασκευή της ετικέτας θα πρέπει να απεικονιστεί ο κόμβος i του $G_{[i]}$ στον κόμβο j του $G_{[j]}$. Επομένως δείξαμε ότι **GA** $\leq_T^p$ **GI**.

Αργότερα θα δείξουμε ότι η παραπάνω αναγωγή μπορεί να μετασχηματιστεί σε αναγωγή κατά Karp.

Στην πραγματικότητα η παραπάνω αναγωγή είναι ειδικής μορφής μιας και ο αλγόριθμος δίνει θετικό αποτέλεσμα αν και μόνο αν *κάποια* από τις ερωτήσεις στο μαντείο απαντήσει θετικά. Σε αυτή την ειδική περίπτωση μπορούμε να δώσουμε την έννοια της *διαζευκτικής αναγωγής*, που μπορούμε να τη θεωρήσουμε ως μια γενίκευση της αναγωγής Karp (many-one). Έτσι προκύπτει ο παρακάτω ορισμός

Ορισμός 2.7. *Ένα σύνολο A* ανάγεται διαζευκτικά *στο σύνολο B αν υπάρχει μια συνάρτηση f υπολογιστή σε πολυωνυμικό χρόνο ($f \in FP$) η οποία υπολογίζει για κάθε είσοδο x μια λίστα $(x_1, x_2, \ldots, x_m)$ από ερωτήσεις έτσι ώστε :*

$$x \in A \Longleftrightarrow \exists i, i = 1, \ldots, m : x_i \in B$$

Ανάλογα μπορεί κανείς να ορίσει την έννοια της *συζευκτικής αναγωγής*

Ορισμός 2.8. *Ένα σύνολο A* ανάγεται συζευκτικά *στο σύνολο B αν υπάρχει μια συνάρτηση f υπολογιστή σε πολυωνυμικό χρόνο ($f \in FP$) η οποία* *υπολογίζει για κάθε είσοδο x μια λίστα $(x_1, x_2, \ldots, x_m)$ από ερωτήσεις έτσι ώστε :*

$$x \in A \Longleftrightarrow \forall i, i = 1, \ldots, m : x_i \in B$$

Είναι εμφανές ότι οι κλάσεις P και NP (και γενικά οι κλάσεις της μορφής $P(\mathcal{C})$ και $NP(\mathcal{C})$) είναι κλειστές ως προς διαζευκτικές και συζευκτικές αναγωγές. Αυτό μπορεί να γίνει κατανοητό αντικαθιστώντας τις ερωτήσεις στο μαντείο κατευθείαν από μια αντίστοιχη λειτουργία μηχανής).

Στη συνέχεια ορίζουμε ένα πρόβλημα απόφασης που μοιάζει πολύ με αυτό του ισομορφισμού γράφων [Lub81]. Έστω G, H δυο γράφοι και $\pi_1, \ldots, \pi_k$ διαφορετικοί ισομορφισμοί μεταξύ των G και H. Για κάθε $k \geq 0$ ορίζουμε $\mathbf{GI_k} = \{(G, H, \pi_1, \ldots, \pi_k) |$ υπάρχει ισομορφισμός $\varphi \neq \pi_i, i = 1, \ldots, k\}$

Προφανώς $\mathbf{GI_0} \equiv \mathbf{GI}$. Επίσης ισχύει $\mathbf{GI_1} \equiv_m^p \mathbf{GA}$. Για την κατεύθυνση $\mathbf{GI_1} \leq_m^p \mathbf{GA}$ χρησιμοποιούμε την αναγωγή που απεικονίζει τους δύο δοσμένους γράφους G και H και τον ισομορφισμό π στον G.(Αφού οι G και H είναι ισομορφικοί θα ισχύει $\#\mathbf{GI}(G, H) = \#\mathbf{GA}(G) = \#\mathbf{GA}(H) \geq 1$. Το πρόβλημα απόφασης $\mathbf{GA}$ απαντά στο ερώτημα $\#\mathbf{GA} > 1$ οπότε αν έχουμε $\mathbf{GA}(G) = True$ τότε υπάρχει και άλλος ισομορφισμός μεταξύ των G και H.) Για την κατεύθυνση $\mathbf{GA} \leq_m^p \mathbf{GI_1}$ χρησιμοποιούμε την αναγωγή $G \longmapsto (G, G, id)$.

Με την ίδια μέθοδο που χρησιμοποιήσαμε για την απόδειξη του $\mathbf{GA} \leq_T^p \mathbf{GI}$ προκύπτει ότι $\mathbf{GI_k} \leq_T^p \mathbf{GI}$ για κάθε k. Αποδεικνύεται επίσης ότι $\mathbf{GI_k} \leq_m^p \mathbf{GI}$. Τέλος, δεν είναι γνωστό αν όλα τα προβλήματα της ακολουθίας $\mathbf{GI_0}, \mathbf{GI_1}, \ldots,$ ανάγονται στο $\mathbf{GA}$.

Στη συνέχεια (αλγόριθμος 2.2) δίνουμε ένα παράδειγμα αναγωγής μεταξύ συναρτήσεων.

Αφ'ενός ο αλγόριθμος για τον έλεγχο αν ένας γράφος είναι συνεκτικός είναι πολυωνυμικός. Αφ' ετέρου η πράξη $G = \overline{G}$ κοστίζει $\mathcal{O}(|E|) = \mathcal{O}(|V|^2)$, δηλαδή πολυωνυμικό χρόνο ως προς την είσοδο. Τέλος, από το πρώτο κεφάλαιο ξέρουμε ότι για συνεκτικούς γράφους ισχύει:

$$\#\mathbf{GA}(G_1 \cup G_2) = \begin{cases} \#\mathbf{GA}(G_1) \cdot \#\mathbf{GA}(G_2), & \text{if } G_1 \not\equiv G_2 \\ 2 \cdot \#\mathbf{GA}(G_1) \cdot \#\mathbf{GA}(G_2), & \text{if } G_1 \equiv G_2 \end{cases}$$

Η παραπάνω ιδιότητα αποδεικνύει την ορθότητα του αλγορίθμου. Παρατηρήστε ότι ο αλγόριθμος καλεί την υπορουτίνα $\#\mathbf{GA}$ ακριβώς τρεις φορές. Επίσης το μέγεθος της τιμής που επιστρέφει η υπορουτίνα $\#\mathbf{GA}$ είναι φραγμένο πολυωνυμικά, μιας και ένας γράφος με n κόμβους μπορεί να έχει το πολύ $n!$ αυτομορφισμούς($\mathcal{O}(\log n!) = \mathcal{O}(p(n))$). Συνεπώς ο αλγόριθμος τρέχει σε πολυωνυμικό χρόνο.

Αλγόριθμος 2.2 **#GI** $\in FP($**#GA**$)$

```
input(G,H)
if (G connected AND H not connected) OR (H connected and G not connected) then
  output 0;
else
  if G not connected then
    G = \overline{G};
    H = \overline{H};
  end if
  x := #GA(G);
  y := #GA(H);
  z := #GA(G ∪ H);
  if z = x · y then
    output 0
  else
    output x
  end if
end if
```

2.3 Αναγωγή του προβλήματος εύρεσης ισομορφισμού στο πρόβλημα απόφασης (GI)

Έστω ότι έχουμε μια θετική απάντηση στην ερώτηση αν δύο γράφοι είναι ισομορφικοί. Το επόμενο βήμα είναι να βρεθεί ένας ισομορφισμός. Το ερώτημα είναι τώρα αν η εύρεση κάποιου ισομορφισμού είναι "πιο δύσκολη" από την απλή διαπίστωση για το αν δύο γράφοι είναι ισομορφικοί ή όχι. Αντιστρέφοντας το ερώτημα, με μαντείο για τον ισομορφισμό γράφων, υπάρχει πολυωνυμικός αλγόριθμος που να κατασκευάζει έναν ισομορφισμό; Αν υπάρχει τέτοιος αλγόριθμος λέμε ότι το πρόβλημα έχει αυτοϋπολογίσιμες λύσεις.

Ορισμός 2.9. *Ένα πρόβλημα $A \in NP$ με την αναπαράσταση που έχουμε δώσει*

$$A = \{x | \exists y, |y| \leq p(|x|) \wedge (x, y) \in\} B$$

όπου $B \in P$ και p κάποιο πολυώνυμο, έχει αυτοϋπολογίσιμες *(*self-computable*) λύσεις αν υπάρχει συνάρτηση $f \in FP(A)$ τέτοια ώστε $\forall x \in A, (x, f(x)) \in B$.*

Πολλά φυσικά προβλήματα ικανοποιούν αυτόν τον ορισμό και υπάρχουν πολλοί διαφορετικοί τρόποι για να κατασκευάσουμε τη συνάρτηση για την οποία κάνει λόγο ο ορισμός 2.9. Ένας από αυτούς είναι να κάνουμε χρήση μιας *αυτο-αναγόμενης* (*self-reducibility*) δομής που ενδεχομένως να έχει το πρόβλημα. Δεν θα δώσουμε αυστηρό ορισμό αυτής της έννοιας, ωστόσο, διαισθητικά, ένα πρόβλημα έχει αυτο-αναγόμενη δομή αν μπορεί να αναχθεί σε μικρότερα στιγμιότυπα του ιδίου προβλήματος. Η αυτο-αναγωγή σχετίζεται στενά με την αναδρομή στον προγραμματισμό.

Ένα απλό παράδειγμα μας δίνει το πρόβλημα **Sat**. Έστω ένας Boolean τύπος F με μεταβλητές $x_1, \ldots, x_n$ και έστω οι F_{false} και F_{true} προκύπτουν από τον F θέτοντας την τιμή της μεταβλητής x_1 *false* και *true* αντίστοιχα. Τότε ο τύπος F είναι ικανοποιήσιμος αν και μόνο αν ο F_{false} ή ο F_{true} είναι ικανοποιήσιμος. Επίσης οι F_{false} και F_{true} έχουν λιγότερες μεταβλητές από τον F, αποτελούν μικρότερα στιγμιότυπα του προβλήματος, επομένως το πρόβλημα έχει αυτο-αναγόμενη δομή.

Για τον ισομορφισμό γράφων τα πράγματα είναι λίγο πιο δύσκολα και θα χρειαστεί να ορίζουμε κάποιες βοηθητικές έννοιες. Δεδομένου ενός συνόλου $A \in NP$, του συνόλου $B \in P$ το οποίο επαληθεύει τη λύση που δίνει το A και ενός πολυωνύμου p, ορίζουμε

Prefix $= \{(x, z) | z$ είναι πρόθεμα μιας συμβολοακολουθίας y έτσι ώστε $|y| \leq p(|x|) \wedge (x, y) \in B\}$

Λήμμα 2.10. *Αν* **PrefixA** $\leq_T^p$ **A** *τότε το A έχει αυτοϋπολογίσιμες λύσεις.*

Αλγόριθμος 2.3 Η συνάρτηση f της απόδειξης του λήμματος 2.10

```
input x;
if (x, ε) ∉ PrefixA then
  reject
else
  y := ε; /* η κενή συμβολοακολουθία */
  for i := 1 to p(|x|) do
    if (x, y0) ∈ PrefixA then
      y := y0;
    else if (x, y1) ∈ PrefixA then
      y := y1;
    end if
  end for
end if
output y; /*Μια λύση*/
```

Απόδειξη. Αρχικά (αλγόριθμος 2.3) δίνουμε μια συνάρτηση αναζήτησης f που βρίσκει μια λύση για το πρόβλημα A χρησιμοποιώντας το **Prefix** ως μαντείο,

Αλγόριθμος 2.4 Προσαρμογή της f του λήμματος 2.10 για το **PrefixGI**

```
input(G,H);
if (G, H, ∅) ∉ PrefixGI then
  output "non-isomorphic"
else
  φ := ∅; /*αρχικά η φ δεν είναι ορισμένη*/
  for i := 0 to n do
    for j := 0 to n do
      if (G, H, φ ∪ (i, j)) ∈ PrefixGI then
        k := j
      end if
    end for
    φ := φ ∪ (i, k);
  end for
  output φ; /*Ένας ισομορφισμός*/
end if
```

δηλαδή είναι $f \in FP(\textbf{PrefixA})$. Με $y0$ και $y1$ συμβολίζουμε την παράθεση (concatenation) του string y με τους αριθμούς 0 και 1 αντίστοιχα. Στη συνέχεια από τη μεταβατικότητα της αναγωγής Cook και αφού $\textbf{PrefixA} \leq_T^p A$ προκύπτει ότι $f \in FP(A)$. □

Στη συνέχεια, για να χρησιμοποιήσουμε το παραπάνω λήμμα, ορίζουμε το Prefix πρόβλημα για τον ισομορφισμό γράφων.

$\textbf{PrefixGI} = \{(G, H, \varphi)\}$ όπου G, H γράφοι n κορυφών, $\varphi : \{1, \ldots, i\} \rightarrow \{1, \ldots, n\}$ για κάποιο $i \leq n$ και η φ μπορεί να επεκταθεί σε ισομορφισμό.

Ο αλγόριθμος 2.4 είναι η προσαρμογή του αλγορίθμου 2.3 του λήμματος 2.10 για τον ισομορφισμό γράφων. Με το συμβολισμό $\varphi \cup (i, j)$ συμβολίζουμε την επέκταση της φ που απεικονίζει τον κόμβο i στον κόμβο j.

Θεώρημα 2.11. *[Sch76] Το* **PrefixGI** *ανάγεται κατά Karp στο* **GI** *(*$\textbf{PrefixGI} \leq_m^p \textbf{GI}$*).*

Απόδειξη. Δεδομένων των (G, H, φ) όπου οι γράφοι G και H έχουν n κορυφές και η φ είναι μια συνάρτηση απεικόνισης από το $\{1, \ldots, i\}$ στο $\{1, \ldots, n\}$, κατασκευάζουμε δυο νέους γράφους $G' = G_{[1,\ldots,i]}$ και $H' = H_{[\varphi(1),\ldots,\varphi(i)]}$. Η ορθότητα της αναγωγής προκύπτει από την παρατήρηση ότι κάθε πιθανός ισομορφισμός μεταξύ των G' και H' πρέπει να είναι μια επέκταση της φ. □

Από το παραπάνω θεώρημα, το λήμμα 2.10, τον αλγόριθμο για το **PrefixGI** και το γεγονός ότι η Karp αναγωγή συνεπάγεται και Cook αναγωγή, προκύπτει το ακόλουθο.

Πόρισμα 2.12. *Το* **GI** *έχει αυτό-υπολογίσιμες (self-computable) λύσεις.*

Το επόμενο που θα θέλαμε είναι να κάνουμε το ίδιο για το **GA**, να δείξουμε δηλαδή ότι έχει αυτό-υπολογίσιμες λύσεις. Ορίζουμε κατά ανάλογο τρόπο το **PrefixGA**.

PrefixGA $= \{(G, \varphi)\}$ όπου ο γράφος G έχει n κορυφές, $\varphi : \{1, \ldots, i\} \to \{1, \ldots, n\}$ για κάποιο $i \leq n$ και η φ μπορεί να επεκταθεί σε έναν μη-τετριμμένο αυτομορφισμό του G.

Όμως το **PrefixGA** είναι τουλάχιστον το ίδιο δύσκολο με το **GI**, δηλαδή **GI** $\leq_T^p$ **PrefixGA** και επειδή μέχρι στιγμής δεν υπάρχει αναγωγή από το **GI** στο **GA**, η παραπάνω μέθοδος δεν είναι πιθανό να δουλέψει. Η αναγωγή **GI** $\leq_T^p$ **PrefixGA** δίδεται από τον αλγόριθμο 2.5.

Αλγόριθμος 2.5 GI $\leq_T^p$ PrefixGA

input (G,H); /*Έστω ότι οι γράφοι G και H έχουν n κόμβους. Έστω επίσης ότι οι κόμβοι του H έχουν αριθμηθεί από το n+1 ως το 2n. Τέλος υποθέτουμε ότι οι γράφοι είναι συνεκτικοί, αλλιώς χρησιμοποιούμε τα συμπληρώματά τους.*/
for $i := n + 1$ **to** $2 \cdot n$ **do**
  if $(G \cup H, (1, i)) \in$ **PrefixGA then**
    accept
  end if
end for
reject

Στη συνέχεια θα δείξουμε με άλλο τρόπο ότι το **GA** έχει αυτο-υπολογίσιμες λύσεις.

Θεώρημα 2.13. *Το πρόβλημα αυτομορφισμού γράφων* **GA** *έχει αυτο-υπολογίσιμες (self-computable) λύσεις.*

Απόδειξη. Δείχνουμε απ' ευθείας ότι δεδομένου ενός γράφου G, ένας μη-τετριμμένος αυτομορφισμός του G μπορεί να κατασκευαστεί σε πολυωνυμικό χρόνο (εφόσον υπάρχει), με τη βοήθεια μαντείου για το **GA**.

Το πρώτο μέρος του αλγορίθμου 2.6 τοποθετεί όσο το δυνατόν περισσότερες ταυτοτικές απεικονίσεις (i,i) στον αυτομορφισμό φ, εφόσον μπορεί να

Αλγόριθμος 2.6 GA – construction $\in FP(\mathbf{GA})$

```
input G; /*Έστω ότι ο G έχει n κορυφές*/
if G ∈ GA then
  output "the graph is rigid"
else
  S := ∅; φ := ∅;
  for i := 1 to n do
    if G[i] ∈ GA then
      φ := φ ∪ (i, i);
      G := G[i]
    else
      S := S ∪ {i}
    end if
  end for
end if
/*Σε αυτό το σημείο το S περιέχει τουλάχιστον 2 στοιχεία, η φ μπορεί να
επεκταθεί σε μη-τετριμμένο αυτομορφισμό και κάθε τέτοιος αυτομορφισμός
δεν απεικονίζει στοιχεία του S στον εαυτό τους.*/
T := S; H := G;
for i ∈ S do
  for j ∈ T − {i} do
    if G[i] ∪ H[j] ∈ GA then
      k := j;
    end if
  end for
  φ := φ ∪ (i, k);
  T := T − k
  G := G[i]; H := H[k];
end for
```

επεκταθεί σε μη-τετριμμένο αυτομορφισμό. Το δεύτερο μέρος του αλγορίθμου δουλεύει με την βεβαίωση ότι η φ όπως έχει κατασκευαστεί ως εκείνο το σημείο, μπορεί να επεκταθεί σε μη-τετριμμένο αυτομορφισμό και επιπλέον οι κόμβοι που δεν έχουν μπει ακόμα στην φ (σύνολο S) δεν μπορούν να απεικονισθούν στον εαυτό τους. Αυτό που κάνει είναι να ελέγχει όλους τους δυνατούς συνδυασμούς κόμβων από το S. Η βεβαίωση που αναφέραμε χρειάζεται για την ορθότητα του ελέγχου “$G_{[i]} \cup H_{[j]} \in \mathbf{GA}$” γιατί αφού $\#\mathbf{GA}(G_{[i]}) = 1$ και $\#\mathbf{GA}(H_{[j]}) = 1$ αν η ένωση των γράφων έχει μη τετριμμένο αυτομορφισμό τότε οι δύο γράφοι είναι ισομορφικοί. Αυτό σημαίνει ότι υπάρχει μη-τετριμμένος αυτομορφισμός στον G που να απεικονίζει τον κόμβο i στον κόμβο j. □

Αναφέρουμε χωρίς απόδειξη ότι $\#\mathbf{GI} \in FP(\mathbf{GI})$. Η απόδειξη μπορεί να βρεθεί στο [KST93].

2.4 Μοναδικότητα Λύσης - Uniqueness of Solution

Πολλές φορές δεν μας ενδιαφέρει αν ένα πρόβλημα έχει λύση αλλά κάτι πιο εξειδικευμένο, αν έχει μοναδική λύση. Για την περίπτωση του ισομορφισμού γράφων το πρόβλημα ύπαρξης μοναδικού ισομορφισμού μεταξύ δύο γράφων (**UniqueGI**), είναι το πρόβλημα απόφασης που απαντά με 'ναι' αν δύο γράφοι έχουν έναν μοναδικό ισομορφισμό και με 'όχι' αν οι δύο γράφοι δεν έχουν ισομορφισμό ή έχουν περισσότερους από έναν ισομορφισμούς.

Θεώρημα 2.14. *Το πρόβλημα μοναδικού ισομορφισμού* **UniqueGI** *είναι ισοδύναμο για Cook αναγωγή με το* **GA**.

Απόδειξη.
Αρχικά θα δείξουμε ότι το συμπλήρωμα του **GA**, το $\overline{\mathbf{GA}}$, ανάγεται κατά Karp στο **UniqueGI**. Δεδομένου του γράφου G για τον οποίο θέλουμε να ξέρουμε αν έχει μη-τετριμμένους αυτομορφισμούς δίνουμε στο μαντείο για το **UniqueGI** το ζεύγος (G,G). Προφανώς υπάρχει ένας ισομορφισμός, ο ταυτοτικός, που αντιστοιχεί στον τετριμμένο αυτομορφισμό. Επομένως, αν ο G έχει μη-τετριμμένους αυτομορφισμούς, το ζεύγος (G,G) θα έχει περισσότερους από έναν ισομορφισμούς και το μαντείο θα απαντήσει αρνητικά. Αν το μαντείο απαντήσει θετικά, αυτό σημαίνει ότι ο G δεν έχει μη-τετριμμένους αυτομορφισμούς. Συνεπώς δείξαμε ότι $\overline{\mathbf{GA}} \leq_m^p$ **UniqueGI**.

Για την άλλη κατεύθυνση θεωρούμε το ακόλουθο πρόγραμμα που κάνει τρεις ανεξάρτητες ερωτήσεις στο μαντείο για το **GA**.

Αλγόριθμος 2.7 UniqueGI $\leq_T^p$ GA

```
input(G,H);
if G ∉ GA and H ∉ GA and (G ∪ H) ∈ GA then
  accept
else
  reject
end if
end.
```

Υπάρχει μοναδικός ισομορφισμός μεταξύ των G και H αν και μόνο αν και οι δύο γράφοι δεν έχουν μη-τετριμμένο αυτομορφισμό αλλά η ένωση των δύο γράφων έχει έναν μη-τετριμμένο αυτομορφισμό ο οποίος θα απεικονίζει τους κόμβους του G σε αυτούς του H και αντίστροφα. □

Μιας και δεν φαίνεται το **GI** να ανάγεται στο **GA**, το **UniqueGI** φαίνεται να είναι πιο εύκολο από το **GI**. Το γεγονός αυτό διαφοροποιεί το **GI** από τα NP-Complete προβλήματα για τα οποία συνήθως ισχύει ότι το πρόβλημα μοναδικής λύσης δεν είναι πιο εύκολο από το αρχικό. Ένα άλλο σημείο στο οποίο διαφέρει το **GI** από τα NP-Complete προβλήματα, είναι για τα NP-Complete προβλήματα οι εκδοχές απαρίθμησης των λύσεων (#) είναι πιο δύσκολες από τα αντίστοιχα προβλήματα απόφασης, ενώ για το **GI** ισχύει ότι το **#GI** είναι το ίδιο δύσκολο.

2.5 Προβλήματα Με Υπόσχεση - Promise Problems

Η παραπάνω κατηγορία προβλημάτων απόφασης (μοναδικής λύσης) μας παραπέμπει στο διαχωρισμό των προβλημάτων που έχουν λύση σε δύο κατηγορίες: αυτά που έχουν μοναδική λύση και σε αυτά που έχουν περισσότερες λύσης. Μια δικαιολογημένη ερώτηση είναι αν υπάρχει αλγόριθμος πιο αποδοτικός που να δίνει σωστή απάντηση για ένα υποσύνολο στιγμιοτύπων. Προβλήματα των οποίων τα στιγμιότυπα θέλουμε να πληρούν κάποιες προδιαγραφές ονομάζονται προβλήματα με υπόσχεση (promise problems). Πιο αυστηρά:

Ορισμός 2.15. *Ένα* Πρόβλημα με Υπόσχεση *(Promise Problem) είναι ένα ζεύγος συνόλων (Q,R). Ένα σύνολο L καλείται* λύση *του προβλήματος με υπόσχεση (Q,R) αν*

$$\forall x (x \in Q \Rightarrow (x \in L \Leftrightarrow x \in R)).$$

Για ευκολία μπορούμε να φανταστούμε το πρόβλημα με υπόσχεση σαν ένα πρόγραμμα με την εξής δομή:

input x
promise Q(x)
property R(x)

όπου τα Q και R είναι κατηγορήματα. Ένας αλγόριθμος που λύνει το πρόβλημα με υπόσχεση πρέπει να αποφασίσει αν R(x) υπό την υπόθεση ότι η είσοδος (το x) ικανοποιεί το κατηγόρημα Q. Αν δεν το ικανοποιεί ο αλγόριθμος μπορεί να απαντήσει με ‘ναι’ ή ‘όχι’ είτε το x ικανοποιεί το R, είτε όχι.

Στη συνέχεια θεωρούμε ένα πρόβλημα με υπόσχεση που έχει σχέση με το **GI**.

input H, G_1, G_2
promise $H \equiv G_1 \quad \textbf{xor} \quad H \equiv G_2$
property $H \equiv G_1$

Έστω A μια λύση του παραπάνω προβλήματος με υπόσχεση. Αποδεικνύεται ότι $\mathbf{GI} \leq_T^p A$.

Έστω **1GA** το σύνολο των γράφων με το πολύ έναν μη-τετριμμένο αυτομορφισμό. Παρατηρούμε ότι $\mathbf{1GA} \cap \mathbf{GA} = \mathbf{UniqueGA}$. Θεωρούμε το πρόβλημα με υπόσχεση (**1GA**,**GA**). Για το πρόβλημα αυτό δίνουμε χωρίς απόδειξη τα ακόλουθα αποτελέσματα.

$$\forall L \text{ solution of } (\mathbf{1GA}, \mathbf{GA}) : \mathbf{GA} \leq_T^p L$$

Από το οποίο προκύπτει σαν πόρισμα ότι $\mathbf{GA} \leq_T^p \mathbf{UniqueGA}$

Επίσης έχουμε ότι $\mathbf{UniqueGA} \leq_T^p \mathbf{GA}$ επομένως σε συνδυασμό με το παραπάνω προκύπτει ότι $\mathbf{UniqueGA} \equiv_T^p \mathbf{GA}$.

2.6 Ανάγοντας Πολλαπλές Ερωτήσεις σε Μία

Είδαμε στην αναγωγή του **GA** στο **GI** ότι κάναμε περισσότερες από μια ερωτήσεις στο μαντείο για το **GI** και επομένως η αναγωγή ήταν Cook και όχι Karp. Από την άλλη παρατηρούμε ότι μόλις το μαντείο δώσει θετική απάντηση σε μια από τις ερωτήσεις έχουμε απάντηση για το **GA** και επιπλέον οι ερωτήσεις δεν εξαρτώνται από το τι έχει απαντήσει μέχρι εκείνη τη στιγμή το μαντείο, αλλά αποκλειστικά από την είσοδο (αναγωγή διαζευκτικού πίνακα αληθείας).

Όπως θα δούμε σε αυτή την ενότητα, όλες οι ερωτήσεις $(G_{[i]}, G_{[j]}), i < j$ μπορούν να συνδυαστούν σε μια ερώτηση (G', G'') με τέτοιο τρόπο ώστε τα G' και G'' να είναι ισομορφικά αν και μόνο αν *τουλάχιστον* για ένα από τα ζεύγη $(G_{[i]}, G_{[j]}), i < j$ οι $G_{[i]}$ και $G_{[j]}$ είναι ισομορφικοί.

Ακολουθούν οι ορισμοί των εννοιών or-function και and-function.

Ορισμός 2.16. *Μια or-function για ένα σύνολο A είναι μια συνάρτηση f που απεικονίζει ακολουθίες από strings σε strings έτσι ώστε για κάθε ακολουθία $x_1, \ldots, x_n$ ισχύει:*

$$f(x_1, \ldots, x_n) \in A \Leftrightarrow \exists i, 1 \leq i \leq n, x_i \in A$$

Ορισμός 2.17. *Μια and-function για ένα σύνολο A είναι μια συνάρτηση f που απεικονίζει ακολουθίες από strings σε strings έτσι ώστε για κάθε ακολουθία $x_1, \ldots, x_n$ ισχύει:*

$$f(x_1, \ldots, x_n) \in A \Leftrightarrow \forall i, 1 \leq i \leq n, x_i \in A$$

Το ακόλουθο θεώρημα μας λέει ότι το **GA** έχει or-function υπολογίσιμη σε πολυωνυμικό χρόνο, αλλά δεν είναι γνωστό αν έχει και αντίστοιχη and-function.

Θεώρημα 2.18. *Το* **GA** *έχει or-function υπολογίσιμη σε πολυωνυμικό χρόνο.*

Δίνουμε χωρίς απόδειξη τα ακόλουθα δύο θεωρήματα.

Θεώρημα 2.19. *Το* **GI** *έχει and-function υπολογίσιμη σε πολυωνυμικό χρόνο.*

Θεώρημα 2.20. *Το* **GI** *έχει or-function υπολογίσιμη σε πολυωνυμικό χρόνο.*

Πόρισμα 2.21. *Το* **GA** *ανάγεται στο* **GI** *κατά Karp.*

Απόδειξη. Από το παραπάνω θεώρημα το **GI** έχει or-function υπολογίσιμη σε πολυωνυμικό χρόνο, άρα, και με βάση την παραπάνω συζήτηση, υπάρχει συνάρτηση g τέτοια ώστε:

$$g((G_{[1]}, G_{[2]}), \ldots, (G_{[n-1]}, G_{[n]})) \in \mathbf{GI} \Leftrightarrow G \in \mathbf{GA}$$

□

Κεφάλαιο 3

Παραλλαγές του Ισομορφισμού γράφων που είναι εύκολες

Όπως έχουμε εξηγήσει μέχρι στιγμής το πρόβλημα ισομορφισμού γράφων δεν είναι γνωστό αν ανήκει στο P, αν είναι NP-Complete ή ενδεχομένως αν ανήκει σε κάποια ενδιάμεση κλάση. Στο κεφάλαιο αυτό θα ασχοληθούμε με περιπτώσεις του ισομορφισμού γράφων που είναι αποδεδειγμένα εύκολες, με υποπεριπτώσεις δηλαδή που ανήκουν στο P.

3.1 Ισομορφισμός σε Επίπεδους Γράφους (PGI)

Ιστορικά, ένα από τα πρώτα αποτελέσματα για τον PGI αφορούσε επίπεδους γράφους με μέγιστο βαθμό 3 (triconnected planar graphs) μιας και ένας τέτοιος γράφος έχει μοναδική εμφύτευση (unique embedding) σε σφαίρα [Whi33]. Ο Weinberg [Wei66] εκμεταλλεύτηκε αυτό το γεγονός για να φτιάξει έναν αλγόριθμο που έλεγχε αν υπάρχει ισομορφισμός μεταξύ τέτοιων γράφων σε $O(|V|^2)$ χρόνο, όπου V είναι το σύνολο των κορυφών των δύο γραφημάτων. Το αποτέλεσμα επεκτάθηκε γενικά σε επίπεδους γράφους και βελτιώθηκε σε $O(|V|log|V|)$ από τους Hopcroft και Tarjan[HT72]. Τέλος, το 1974, οι Hopcroft και Wong [HW74] έδωσαν γραμμικό αλγόριθμο για τον ισομορφισμό επίπεδων γράφων. Ο αλγόριθμος είναι αρκετά περίπλοκος και, όπως λένε και οι συγγραφείς του, έχει κυρίως θεωρητικό ενδιαφέρον γιατί, λόγω της μεγάλης σταθεράς χρόνου, δεν θεωρείται πρακτικός.

3.2 Χρωματισμένοι Γράφοι

Όταν οι γράφοι με τους οποίους ασχολούμαστε έχουν χρώματα στις κορυφές θα μιλάμε για χρωματισμένους γράφους. Προφανώς στη γενική περίπτωση το πρόβλημα ισομορφισμού (αυτομορφισμού) χρωματισμένων γράφων δεν είναι πιο εύκολο από το αντίστοιχο πρόβλημα χωρίς χρώματα. Από ότι έχουμε αναφέρει μάλιστα τα δύο προβλήματα είναι ισοδύναμα υπό Karp αναγωγή 2.4. Σε αυτή την ενότητα θα μας απασχολήσουν ειδικές περιπτώσεις αυτών των προβλημάτων.

Θεώρημα 3.1. *[JKMT03] Το πρόβλημα αυτομορφισμού γράφου για χρωματισμένους γράφους με το πολύ 2 κόμβους με το ίδιο χρώμα (color classes of size at most 2) είναι ισοδύναμο κατά many-one* $\mathbf{AC}^0$ *με το* $\overline{UCC}$*, πρόβλημα απόφασης για το αν ένας γράφος έχει πάνω από μία συνεκτικές συνιστώσες. Συμβολικά:*

$$\mathbf{2-GA} \equiv_m^{\mathbf{AC}^0} \overline{\mathbf{UCC}}.$$

Θεώρημα 3.2. *[JKMT03]Το πρόβλημα ισομορφιμού γράφων για χρωματισμένους γράφους με το πολύ 3 κόμβους με το ίδιο χρώμα (color classes of size at most 3) ανήκει στην κλάση* **SL***(Symmetric Logarithmic). Συμβολικά:*

$$\mathbf{3-GI} \in \mathbf{SL}$$

.

Θεώρημα 3.3. *[JKMT03] Το πρόβλημα αυτομορφισμού γράφου για χρωματισμένους γράφους με το πολύ 3 κόμβους με το ίδιο χρώμα (color classes of size at most 3) ανήκει στην κλάση* **SL***(Symmetric Logarithmic). Συμβολικά:*

$$\mathbf{3-GA} \in \mathbf{SL}.$$

Θεώρημα 3.4. *[JKMT03] Το πρόβλημα ισομορφιμού γράφων για χρωματισμένους γράφους με το πολύ 2 κόμβους με το ίδιο χρώμα (color classes of size at most 2) είναι δύσκολο για την κλάση* **SL***(Symmetric Logarithmic) για αναγωγή* $\leq_m^{\mathbf{AC}^0}$*. Συμβολικά:*

$$\mathbf{2-GI}\ \mathit{is}\ \mathbf{SL}-\mathit{hard}.$$

Από τα θεωρήματα 3.2 και 3.4 προκύπτει το εξής:

Πόρισμα 3.5. *Τα* **2-GI** *και* **3-GI** *είναι* **SL***-Complete για αναγωγή* $\leq_m^{\mathbf{AC}^0}$.

3.3 Ισομορφισμός Δένδρων

Τα δένδρα είναι μια ειδική περίπτωση επίπεδων γραφημάτων και επομένως ο ισομορφισμός δένδρων ανήκει στο P και μάλιστα υπάρχει γραμμικός αλγόριθμος από ότι έχουμε ήδη αναφέρει [HT72]. Στη συνέχεια προτείνουμε έναν απλό αλγόριθμο για τον ισομορφισμό δένδρων [KST93]

1. Σε καθένα από τα δένδρα βάζουμε ετικέτα σε όλα τα φύλλα (κορυφές βαθμού 1) το '1'. Αν ο αριθμός των φύλλων των G_1 και G_2 είναι διαφορετικός απαντάμε "μη-ισομορφικά δένδρα".
2. Σε κάθε δένδρο, καθόρισε τα σύνολα κορυφών χωρίς ετικέτα S_1 και S_2 έτσι ώστε κάθε γείτονας κορυφής $v \in S_i$, εκτός από έναν το πολύ, να έχει ετικέτα. Δώσε στην κορυφή v ως προσωρινή ετικέτα την λίστα $(l_1, \ldots, l_k)$ των ετικετών των γειτόνων της που έχουν ετικέτα, ταξινομημένη σε μη-φθίνουσα σειρά. Σύγκρινε τις προσωρινές ετικέτες των κορυφών των S_1 και S_2. Αν οι ετικέτες δεν συμφωνούν (ως πολυσύνολα) δώσε απάντηση "μη-ισομορφικά δένδρα".
3. Αντικατάστησε τις προσωρινές ετικέτες (που είναι λίστες αριθμών) με νέες ετικέτες που είναι αριθμοί $k, k+1, k+2, \ldots$ έτσι ώστε οι κορυφές με την ίδια προσωρινή ετικέτα να έχουν την ίδια νέα ετικέτα και οι ετικέτες $k, k+1, k+2, \ldots$ να μην έχουν ξαναχρησιμοποιηθεί
4. Αν δεν έχουν όλες οι κορυφές ετικέτα, γύρισε στο βήμα 2
5. Δώσε ως απάντηση "ισομορφικά δένδρα"

Ο αλγόριθμος αυτός αντιμετωπίζει τις κορυφές ανά επίπεδα, ξεκινώντας από τα φύλλα. Είναι εύκολο να διαπιστώσει κανείς ότι σε κάθε ενδιάμεσο βήμα, ίσες ετικέτες αντιστοιχούν σε ισομορφικά υποδένδρα. Με άλλα λόγια, οι ετικέτες που δίνουμε σε κάθε κορυφή, κωδικοποιούν με μοναδικό τρόπο το υποδένδρο στο οποίο είναι ρίζα.

Στη συνέχεια θα δώσουμε κάποια αποτελέσματα πληρότητας του προβλήματος ισομορφισμού δένδρων για κλάσεις μικρότερες του P. Αυτό όμως σημαίνει ότι δεν μας αρκεί πλέον οι διαφορετικές αναπαραστάσεις να έχουν πολυωνυμική σχέση μεταξύ τους. Μάλιστα θα δούμε ότι έχουμε διαφορετικά αποτελέσματα πληρότητας ανάλογα ποια από τις δύο ακόλουθες αναπαραστάσεις θα χρησιμοποιούμε.

Ο πρώτος τρόπος αναπαράστασης ονομάζεται *αναπαράσταση συμβολοακολουθίας* (string representation) και ορίζεται αναδρομικά ως εξής. Το δένδρο με έναν κόμβο αναπαρίσταται από την συμβολοακολουθία (). Το δένδρο T που αποτελείται από μια ρίζα και υποδένδρα $T_1, \ldots, T_k$ όπου α_i η αναπαράσταση για το υποδένδρο T_i, έχει αναπαράσταση $(\alpha_1 \ldots \alpha_k)$. Προφανώς ένα δένδρο μπορεί να έχει περισσότερες αναπαραστάσεις σε συμβολοακολουθία, αναλόγως ποιόν κόμβο θα διαλέξουμε ως ρίζα. Μπορούμε να χρησιμοποιήσουμε και αυτόν τον τρόπο για την αναπαράσταση δένδρων με χρώματα στις κορυφές. Σε αυτή την περίπτωση χρησιμοποιούμε "χρωματιστές παρενθέσεις" δηλαδή αν C είναι το σύνολο των χρωμάτων, μετά από κάθε παρένθεση που ανοίγει χρησιμοποιούμε $\log(|C|)$ bits μετά το άνοιγμα της παρένθεσης για να κωδικοποιήσουμε το χρώμα.

Ο δεύτερος τρόπος αναπαράστασης ονομάζεται *αναπαράσταση με δείκτες* (pointer representation). Σε αυτή την περίπτωση θεωρούμε ότι τα δένδρα δίνονται από μια λίστα από ζευγάρια κόμβων που αναπαριστούν κατευθυνόμενες πλευρές. Όπως και στην προηγούμενη περίπτωση, αν ασχολούμαστε με χρωματισμένα δένδρα με την αναπαράσταση ενός κόμβου χρησιμοποιούμε $\log(|C|)$ bits για να κωδικοποιήσουμε το χρώμα του.

Στη συνέχεια θα χρησιμοποιούμε τους συμβολισμούς **TI** και **CTI** για τα προβλήματα ισομορφισμού δένδρων (Tree Isomorphism) και ισομορφισμού χρωματισμένων δένδρων (Colored Tree Isomorphism) αντίστοιχα. Επίσης θα χρησιμοποιούμε τους συμβολισμούς **TA** και **CTA** για τα προβλήματα αυτομορφισμού σε δένδρα (Tree Automorphism) και του αυτομορφισμού σε χρωματισμένα δένδρα (Colored Tree Automorphism).

Επειδή στα ακόλουθα αποτελέσματα θα χρειαστούμε την έννοια της many-one $\mathbf{AC}^0$ αναγωγής, την ορίζουμε ακολούθως.

Ορισμός 3.6. *Έστω δύο γλώσσες $A, B \subseteq \Sigma^*$. Λέμε ότι η γλώσσα A ανάγεται many-one $\mathbf{AC}^0$ στην B αν υπάρχει μια συνάρτηση f που να υπολογίζεται από μια* **DLOGTIME***-ομοιόμορφη (uniform) οικογένεια από $\mathbf{AC}^0$ κυκλώματα που έχουν την ιδιότητα ότι $\forall x \in \Sigma^*, x \in A \iff f(x) \in B$. Σε αυτή την περίπτωση γράφουμε $A \leq_m^{\mathbf{AC}^0} B$.*

Είναι εμφανές ότι $\mathbf{TI} \leq_m^{\mathbf{AC}^0} \mathbf{CTI}$ θεωρώντας ότι όλοι οι κόμβοι έχουν το ίδιο χρώμα. Το αντίθετο δεν είναι προφανές αλλά ισχύει επίσης.

Λήμμα 3.7. *Χρησιμοποιώντας είτε την αναπαράσταση συμβολοακολουθίας είτε την αναπαράσταση με δείκτες ισχύει ότι $\mathbf{CTI} \leq_m^{\mathbf{AC}^0} \mathbf{TI}$*

Η απόδειξη για κάθε μία από τις περιπτώσεις αναπαράστασης μπορεί να βρεθεί στο [JKMT03]. Από το λήμμα προκύπτει ότι τα **TI** και **CTI** είναι $\mathbf{AC}^0$ ισοδύναμα.

Μια παραλλαγή αυτού του λήμματος δείχνει ότι και τα **TA** και **CTA** είναι $\mathbf{AC}^0$ ισοδύναμα.

Στη συνέχεια δίνουμε κάποια αποτελέσματα πληρότητας από την ίδια εργασία για τα **TI** και **CTI**.

Θεώρημα 3.8. *Στην αναπαράσταση με συμβολοακολουθίες (string representation), τα* **TI** *και* **CTI** *είναι $\mathbf{NC}^1 - Complete$ υπό $\leq_m^{\mathbf{AC}^0}$ αναγωγή.*

Θεώρημα 3.9. *Στην αναπαράσταση με δείκτες (pointer representation), τα* **TI**, **CTI**, **TA** *και* **CTA** *είναι $\mathbf{L}^1 - Complete$ υπό $\leq_m^{\mathbf{AC}^0}$ αναγωγή.*

Παρατηρούμε ότι για την αναπαράσταση με δείκτες τα **TA** και **CTA** είναι **L**-Complete όπως και τα **TI** και **CTI**, ενώ για την αναπαράσταση με συμβολοακολουθίες δεν είναι γνωστό αν είναι $\mathbf{NC}^1$-Complete. Πρόκειται για μια ακόμα ένδειξη ότι το πρόβλημα του αυτομορφισμού είναι πιο εύκολο από το πρόβλημα του ισομορφισμού.

3.4 Προσεγγιστικές Μέθοδοι

Τα παρακάτω βρέθηκαν από το "A Compendium of NP optimization problems" (http://www.nada.kth.se/ viggo/problemlist/compendium.html)Πρόκειται για προβλήματα τα οποία έχουν να κάνουν με ισομορφισμό σε γράφους και τα οποία έχουν μελετηθεί από τη σκοπιά των προσεγγιστικών αλγορίθμων.

Ορισμός 3.10. MCS : MAXIMUM COMMON SUBGRAPH - *Μέγιστος Κοινός Υπογράφος*
Δεδομένων δύο γράφων $G_1(V_1, E_1)$ *και* $G_2(V_2, E_2)$ *ζητάμε να βρούμε* $E_1' \subseteq E_1$ *και* $E_2' \subseteq E_2$ *έτσι ώστε οι γράφοι* $G_1'(V_1, E_1')$ *και* $G_2'(V_2, E_2')$ *να είναι ισομορφικοί και να έχουμε το μέγιστο δυνατό αριθμό πλευρών* ($|E'|$).

Το πρόβλημα είναι NP-Complete και APX-hard

Ορισμός 3.11. MCIG : MAXIMUM COMMON INDUCED SUBGRAPH - *Μέγιστος Κοινός Παραγόμενος Υπογράφος*
Δεδομένων δύο γράφων $G_1(V_1, E_1)$ *και* $G_2(V_2, E_2)$ *ζητάμε να βρούμε* $V_1' \subseteq V_1$ *και* $V_2' \subseteq V_2$ *έτσι ώστε οι παραγόμενοι υπογράφοι των* G_1 *και* G_2 *να είναι ισομορφικοί και να έχουμε το μέγιστο δυνατό αριθμό κορυφών* ($|V'|$).

Δεν προσεγγίζεται με λόγο προσέγγισης $|V|^{\varepsilon}$ για $\varepsilon > 0$

Ορισμός 3.12. MAXIMUM COMMON EMBEDDED SUB-TREE- *Μέγιστο κοινό εμφυτευμένο υπο-δένδρο*
Δεδομένων δύο δένδρων T_1 *και* T_2*, ζητάμε ένα μέγιστο κοινό εμφυτευμένο υποδένδρο, δηλαδή ένα δένδρο με ετικέτες* T' *που να μπορεί να εμφυτευθεί και στα δύο δεδομένα δένδρα* T_1 *και* T_2*. Μια εμφύτευση από το* T' *στο* T *είναι μια injective συνάρτηση από τους κόμβους του* T' *στους κόμβους του* T *που διατηρεί ετικέτες και την ιδιότητα του προγόνου (ancestorship). Μιας και δεν χρειάζεται να διατηρηθεί η ιδιότητα της πατρότητας (fathership), το* T' *δεν χρειάζεται να είναι ένα κοινό υποδένδρο*

Το πρόβλημα προσεγγίζεται με παράγοντα $\log^2 n$, ως προς τον αριθμό των κόμβων των δένδρων [HT96]. Από την άλλη είναι APX-hard [ZJ94].

Ορισμός 3.13. MGT : MINIMUM GRAPH TRANSFORMATION - *Ελάχιστη Μετατροπή Γράφου*
Δεδομένων δύο γράφων $G_1(V_1, E_1)$ *και* $G_2(V_2, E_2)$ *ζητάμε την ελάχιστη μετατροπή έτσι ώστε ο* G_1 *να γίνει ισομορφικός με τον* G_2*. Κατά τη μετατροπή ένα σύνολο πλευρών* E' *αφαιρείται από τον* G_1 *και προστίθεται στον* G_2*. Προφανώς όταν ζητάμε την ελάχιστη μετατροπή θέλουμε να ελαχιστοποιήσουμε το* $|E'|$.

Μέρος δεύτερο

Θεωρία Πολυπλοκότητας

Κεφάλαιο 4

Ποσοδείκτες, Παίγνια και Διαδραστικές Αποδείξεις

Στο κεφάλαιο αυτό θα μας απασχολήσουν θέματα πολυπλοκότητας. Θα ορίσουμε κλάσεις με χρήση ποσοδεικτών και με χρήση τελεστών, θα ορίσουμε την πολυωνυμική ιεραρχία (PH) και θα δούμε πότε καταρρέει, θα μιλήσουμε για διαδραστικές αποδείξεις και μηδενική γνώση (Interactive Proofs and Zero Knowledge) και θα ασχοληθούμε με πιθανοτικές κλάσεις πολυπλοκότητας (probabilistic complexity classes). Τέλος θα ορίσουμε ιεραρχίες μέσα στην κλάση NP και θα δείξουμε ότι το **GI** δεν είναι NP-Complete εκτός κι αν καταρρέει η πολυωνυμική ιεραρχία, κάτι το οποίο δεν θεωρούμε πιθανό.

4.1 Ποσοδείκτες(Quantifiers)

Όταν ορίσαμε τις κλάσεις P και NP δώσαμε και τους ορισμούς μέσω μηχανών Turing. Έχουμε ήδη δει ότι μπορούν να οριστούν με περισσότερους από έναν τρόπους. Ένας τρόπος στον οποίο θα σταθούμε είναι αυτός των ποσοδεικτών.

Το υπολογιστικό μοντέλο που θα έχουμε στο μυαλό μας είναι αυτό ενός πλήρους δυαδικού δένδρου (full complete binary tree). Με $M(x,y)$ θα συμβολίζουμε το αποτέλεσμα του υπολογισμού πάνω στο δένδρο αυτό από τη ρίζα (x) ως το φύλλο y. Το αποτέλεσμα μπορεί να είναι “yes”, “no” ή και “no-answer”. Στην ουσία μπορούμε να περιοριστούμε μόνο στο “yes” και “no-answer”, αλλά δεν θα σταθούμε σε αυτό το σημείο.

Χρησιμοποιώντας τους παραπάνω συμβολισμούς ορίζουμε ξανά τις κλάσεις NP, co-NP και P.

Ορισμός 4.1. $L \in NP : \left\{ \begin{array}{l} x \in L \Rightarrow \exists y \quad M(x,y) = yes \\ x \notin L \Rightarrow \forall y \quad M(x,y) = no \end{array} \right.$

Θα γράφουμε $NP = (\exists/\forall)$.

Ορισμός 4.2. $L \in co-NP : \begin{cases} x \in L \Rightarrow \forall y \quad M(x,y) = yes \\ x \notin L \Rightarrow \exists y \quad M(x,y) = no \end{cases}$

Θα γράφουμε $co-NP = (\forall/\exists)$.

Ορισμός 4.3. $L \in P : \begin{cases} x \in L \Rightarrow \forall y \quad M(x,y) = yes \\ x \notin L \Rightarrow \forall y \quad M(x,y) = no \end{cases}$

Θα γράφουμε $P = (\forall/\forall)$.

4.2 Η Πολυωνυμική Ιεραρχία

Η πολυωνυμική ιεραρχία (PH), με την οποία θα ασχοληθούμε στη συνέχεια, χτίζεται πάνω στις κλάσεις P και NP οι οποίες αποτελούν τη μηδενική και την πρώτη της βαθμίδα αντίστοιχα. Δεν είναι γνωστό αν η πολυωνυμική ιεραρχία σχηματίζει μια άπειρη γνήσια ιεραρχία, ή αν, όπως λέμε, καταρρέει σε κάποιο επίπεδό της. Η εικασία είναι ότι η πολυωνυμική ιεραρχία δεν καταρρέεει, τουλάχιστον στα πρώτα επίπεδά της. Πολλά αποτελέσματα που θα δώσουμε ισχύουν με την προϋπόθεση ότι η ιεραρχία δεν καταρρέει σε κάποιο επίπεδό της. Τέτοια αποτελέσματα έχουν μεγάλη αξία καθώς, όπως είπαμε, δεν πιστεύουμε ότι η PH καταρρέει.

Μια έννοια που θα μας βοηθήσει να αντιληφθούμε κάποιες έννοιες στη συνέχεια είναι τα παιχνίδια δύο παικτών. Λέμε ότι ο παίκτης 1 έχει στρατηγική που κερδίζει αν *υπάρχει* μια κίνηση τέτοια ώστε, *για κάθε* κίνηση του αντιπάλου να *υπάρχει* μια κίνηση τέτοια ώστε, ... , να κερδίζει. Αυτό μπορούμε να το δούμε συμβολικά σαν μια εναλλαγή ποσοδεικτών (alteration of quantifiers).

Ας θυμηθούμε τον ορισμό που είχαμε δώσει για το NP (βλ. Ορισμό 1.7) Ένα σύνολο A είναι στην κλάση NP αν και μόνο αν υπάρχει σύνολο $B \in P$, κάποιο πολυώνυμο p και για κάθε x, $x \in A \Leftrightarrow \exists y, |y| \leq p(|x|)$ and $(x,y) \in B$

Μπορούμε να δούμε την κλάση NP σαν την κλάση των παιγνίων μιας κίνησης με παράμετρο την είσοδο x έτσι ώστε ο παίκτης 1 να έχει στρατηγική που κερδίζει ακριβώς για τα $x \in A$. Συχνά θα λέμε τον παίκτη 1 *Prover* και τον παίκτη 2 *Verifier*. Οι λόγοι θα γίνουν κατανοητοί στη συνέχεια.

Έτσι αν $x \in A$ ο Prover έχει μια κίνηση y την οποία μεταδίδει στον Verifier ο οποίος την επαληθεύει σε πολυωνυμικό χρόνο. Αντίθετα, αν $x \notin A$ ό,τι και να στείλει ο Prover στον Verifier, αυτός δεν θα επαληθεύσει.

Μπορούμε τώρα να γενικεύσουμε, ορίζοντας την πολυωνυμική ιεραρχία

Ορισμός 4.4. *Ένα σύνολο A είναι στην κλάση Σ_k^p ($k \geq 0$) αν και μόνο αν υπάρχει σύνολο $B \in P$ και πολυώνυμο p τέτοια ώστε για κάθε x να ισχύει:*

$$x \in A \Leftrightarrow \exists y_1 \forall y_2 \dots Q y_k \quad (x, y_1, y_2, \dots, y_k) \in B$$

όπου $Q = \forall$ αν το k είναι άρτιος και $Q = \exists$ αν το k είναι περιττός. Επιπλέον οι ποσοδείκτες αφορούν strings μεγέθους το πολύ $p(|x|)$.

Οι κλάσεις Π_k^p ορίζονται αντίστοιχα, εναλλάσσοντας στον παραπάνω ορισμό τους ποσοδείκτες $\forall$ και $\exists$ και έχοντας όπου Q το Q' που είναι το δύικο του Q($Q' = \exists$ αν το k είναι άρτιος και $Q' = \forall$ αν το k είναι περιττός).

Με το συμβολισμό που χρησιμοποιήσαμε προηγουμένως έχουμε ότι:

$$\Sigma_k^p = (\exists\forall\ldots Q/\forall\exists\ldots Q'), k \geq 1$$
$$\Pi_k^p = (\forall\exists\ldots Q'/\exists\forall\ldots Q), k \geq 1$$

Με τους παραπάνω ορισμούς η πολυωνυμική ιεραρχία ορίζεται ως:

$$PH = \bigcup_{k\geq 0} \Sigma_k^p$$

Από τους παραπάνω ορισμούς προκύπτουν τα εξής:

- $P = \Sigma_0^p = \Pi_0^p$
- $NP = \Sigma_1^p$
- $co - NP = \Pi_1^p$
- $\Sigma_k^p = co - \Pi_k^p$
- $\Pi_k^p = co - \Sigma_k^p$

Με βάση τα παραπάνω μπορούμε να ορίσουμε τον τελεστή $\mathcal{N}$ ως εξής:

Ορισμός 4.5. *Για την κλάση $\mathcal{C}$ η κλάση $\mathcal{N} \cdot \mathcal{C}$ είναι η κλάση των γλωσσών L για τις οποίες υπάρχει σύνολο $A \in \mathcal{C}$ και πολυώνυμο p τέτοιο ώστε:*

$$\begin{cases} x \in L \Rightarrow \exists y : (x, y) \in A \\ x \notin L \Rightarrow \forall y : (x, y) \notin A \end{cases}$$

Αντίστοιχα ορίζεται ο τελεστής $co - \mathcal{N}$.

Ορισμός 4.6. *Για την κλάση $\mathcal{C}$ η κλάση $co - \mathcal{N} \cdot \mathcal{C}$ είναι η κλάση των γλωσσών L για τις οποίες υπάρχει σύνολο $A \in \mathcal{C}$ και πολυώνυμο p τέτοιο ώστε:*

$$\begin{cases} x \in L \Rightarrow \forall y : (x, y) \in A \\ x \notin L \Rightarrow \exists y : (x, y) \notin A \end{cases}$$

Πρόταση 4.7. *Από τους παραπάνω ορισμούς προκύπτουν τα ακόλουθα συμπεράσματα, τα οποία θα είναι πολύ χρήσιμα στην απόδειξη προτάσεων*

1. $\Sigma_k^p = \overbrace{\mathcal{N} \cdot (co-\mathcal{N}) \cdot \ldots \cdot Q}^{k\ times} \cdot P$

2. $\Pi_k^p = \overbrace{(co-\mathcal{N}) \cdot \mathcal{N} \cdot \ldots \cdot Q'}^{k\ times} \cdot P$

3. $\mathcal{N} \cdot \Sigma_k^p = \Sigma_k^p$

4. $(co-\mathcal{N}) \cdot \Pi_k^p = \Pi_k^p$

5. $\Pi_{k+1}^p = (co-\mathcal{N}) \cdot \Sigma_k^p$

6. $\Sigma_{k+1}^p = \mathcal{N} \cdot \Pi_k^p$

$$\text{Όπου } Q = \begin{cases} co-\mathcal{N}, & \text{αν } Q \text{ άρτιος} \\ \mathcal{N}, & \text{αν } Q \text{ περιττός} \end{cases} \quad \text{και } Q' = \begin{cases} \mathcal{N}, & \text{αν } Q \text{ άρτιος} \\ co-\mathcal{N}, & \text{αν } Q \text{ περιττός} \end{cases}$$

Επιπλέον, είναι προφανές από τον ορισμό της πολυωνυμικής ιεραρχίας ότι $\Sigma_k^p \cup \Pi_k^p \subseteq \Sigma_{k+1}^p \cap \Pi_{k+1}^p$, for all $k \geq 0$.

Όπως έχουμε πει, εικάζουμε ότι η πολυωνυμική ιεραρχία δεν καταρρέει, δεν ισχύει δηλαδή ότι $\Sigma_k^p = \Sigma_{k+1}^p$ για κάποιο k. Από τον επαγωγικό ορισμό της ιεραρχίας ισχύει αμέσως ότι

- Αν για κάποιο k $\Sigma_k^p \neq \Sigma_{k+1}^p \Rightarrow \Sigma_0^p \neq \Sigma_1^p \neq \ldots \neq \Sigma_k^p \neq \Sigma_{k+1}^p$
- Αν για κάποιο k $\Sigma_k^p = \Sigma_{k+1}^p \Rightarrow \Sigma_k^p = \Sigma_{k+1}^p = \Sigma_{k+2}^p = \ldots$

Πρόταση 4.8. *Για κάθε $i \geq 1$, οι ακόλουθες προτάσεις είναι ισοδύναμες μεταξύ τους:*

1. $\Sigma_i^p = \Sigma_{i+1}^p$
2. $\Pi_i^p = \Pi_{i+1}^p$
3. $\Sigma_i^p = \Pi_i^p$
4. $\Sigma_i^p = \Pi_{i+1}^p$
5. $\Pi_i^p = \Sigma_{i+1}^p$
6. $PH = \Sigma_i^p$
7. *Η πολυωνυμική ιεραρχία καταρρέει στο Σ_i^p.*

Τέλος δίνουμε χωρίς απόδειξη ότι η πολυωνυμική ιεραρχία είναι υποσύνολο της PSCPACE ($PH \subseteq PSPACE$), την κλάση των γλωσσών που μπορούν να αναγνωριστούν από μηχανή Turing με πολυωνυμικά φραγμένη μνήμη (polynomially space-bounded Turing machine). Επιπλέον τα σύνολα Σ_k^p και Π_k^p έχουν *πλήρη* σύνολα υπό Karp (many-one) αναγωγή.

Αν θυμηθούμε τη συζήτηση περί μαντείων και έχοντας υπόψη και τον ορισμό 1.7 για την κλάση NP, το επόμενο θεώρημα γενικεύει τον ορισμό αυτό κάνοντας χρήση μαντείων:

Θεώρημα 4.9. $\Sigma_k^p = \overbrace{NP(\ldots NP(NP)\ldots)}^{k\ times}$ for all $k \geq 1$.

4.3 Διαδραστικές Αποδείξεις

Σε αυτή την ενότητα θα ασχοληθούμε με παιχνίδια πλήρους γνώσης δύο παικτών υπό την σκοπιά του Prover-Verifier. Θα εισάγουμε ποιοτικά την έννοια της *διαδραστικής απόδειξης* και θα δούμε τα αποτελέσματα που είναι γνωστά για το **GI**.

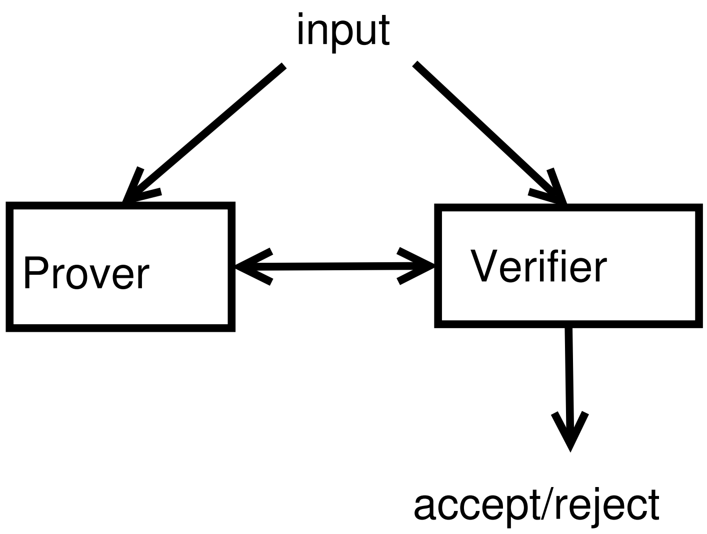
Ένα *σύστημα διαδραστικών αποδείξεων* αποτελείται από δύο αλγορίθμους που θα ονομάζουμε *Prover* και *Verifier*. Και οι δύο έχουν πρόσβαση στην είσοδο και μπορούν να ανταλλάσσουν μηνύματα δια μέσου καναλιού επικοινωνίας. Οι Prover και Verifier κάνουν υπολογισμούς ο καθένας με τη σειρά του. Ένας γύρος αποτελείται από υπολογισμούς από τον Prover ή τον Verifier ο οποίος στη συνέχεια τοποθετεί το αποτέλεσμα των υπολογισμών του στο κανάλι επικοινωνίας. Ο υπολογισμός τερματίζεται όταν ο Verifier απαντήσει “accept” ή “reject” (βλ. σχήμα 4.1).

Ο αλγόριθμος του Prover και του Verifier μπορεί να χρησιμοποιήσει οποιαδήποτε ντετερμινιστική λειτουργία που μπορεί να θεωρηθεί σαν εντολή γλώσσας μηχανής. Επιπλέον επιτρέπονται εντολές της μορφής:

guess randomly $x \in \{1, \ldots, i\}$

Επιτρέπεται δηλαδή η τυχαία επιλογή ενός στοιχείου από πεπερασμένο σύνολο ($S = \{1, \ldots, i\}$) και η ανάθεση της τιμής του σε μια μεταβλητή (x), θεωρώντας ομοιόμορφη κατανομή πάνω στο S. Θεωρούμε ότι το i (το πλήθος των στοιχείων του S) είναι πολυωνυμικά φραγμένο από μέγεθος n της εισόδου. Το αποτέλεσμα ενός τέτοιου αλγορίθμου είναι μια δυαδική τυχαία μεταβλητή με τιμές στο $\{accept, reject\}$ [1]. Συμβολίζουμε με $P \leftrightarrow V(x) = accept$ το

[1] Μια τρίτη τιμή, η “don’t know” μπορεί να προστεθεί ή να αντικαταστήσει κάποια από τις παραπάνω τιμές χωρίς να επηρεαστεί ουσιαστικά ο ορισμός, αλλά δεν θα επεκταθούμε επ’ αυτού.



Σχήμα 4.1: Σύστημα Διαδραστικών Αποδείξεων

συμβάν ένας συγκεκριμένος αλγόριθμος P του Prover και ένας συγκεκριμένος αλγόριθμος V του Verifier να δίνουν έξοδο "accept" για είσοδο x.

Σε πολλές περιπτώσεις θα θέλουμε να διαλέξουμε τυχαία μια μετάθεση π από το σύνολο των μεταθέσεων ενός συνόλου με n στοιχεία (S_n) υπό ομοιόμορφη κατανομή πιθανότητας. Θα γράφουμε :

guess randomly $\pi \in S_n$

Επειδή όμως το πλήθος των στοιχείων του S_n είναι $n!$ που (κατά πάσα πιθανότητα) δεν είναι πολυωνυμικά φραγμένο από την είσοδο, θα χρειαστούμε το ακόλουθο πρόγραμμα:

Αλγόριθμος 4.1 Προσομοίωση της εντολής **guess randomly** $\pi \in S_n$

for $i := 1$ **to** n **do**
  $\pi(i) := i$;
end for
for $j := n$ **downto** 2 **do**
  guess randomly $k \in \{1, \ldots, j\}$;
  $h := \pi(j); \pi(j) := \pi(k); \pi(k) := h$; /*swap(π(j),π(k))*/
end for

Ο πρώτος βρόγχος δημιουργεί την ταυτοτική μετάθεση και στη συνέχεια ο δεύτερος σχηματίζει μια τυχαία μετάθεση. Από την πρώτη επανάληψη προκύπτει ότι $\forall j \in \{1, \ldots, n\} : \pi(n) = j$ με πιθανότητα 1/n. Στη συνέχεια, με δεδομένο ότι $\pi(n) = j$, $\forall j' \in \{1, \ldots, n\} - \{j\} : \pi(n-1) = j'$ με πιθανότητα 1/(n-1), κοκ. Επομένως κάθε μετάθεση έχει την ίδια πιθανότητα $\frac{1}{n} \cdot \frac{1}{n-2} \cdot \ldots \cdot \frac{1}{1} = \frac{1}{n!}$.

4.3.1 Η Κλάση IP

Ορισμός 4.10. *Μια γλώσσα A γίνεται αποδεκτή από ένα σύστημα διαδραστικών αποδείξεων ($A \in IP$) αν υπάρχει πολυωνυμικός αλγόριθμος πιστοποίησης (verifier) V έτσι ώστε τα ακόλουθα να ισχύουν για κάθε είσοδο x:*

$$(1) \quad x \in A \Rightarrow \exists P Prob[P \leftrightarrow V(x) = accept] \geq 3/4$$

$$(2) \quad x \notin A \Rightarrow \forall P Prob[P \leftrightarrow V(x) = accept] \leq 1/4$$

Οι σταθερές 3/4 και 1/4 παρακάτω δεν είναι καθοριστικές για την κλάση. Όπως θα δούμε θα μπορούσαμε να είχαμε χρησιμοποιήσει τις σταθερές 1/2+ε και 1/2-ε αντίστοιχα, για κάθε $\varepsilon > 0$. Αυτό είναι συνέπεια της ιδιότητας *ενίσχυσης πιθανότητας*(probability amplification) που θα δείξουμε ότι έχει η κλάση.

Όταν κάποιος καθορίζει ένα σύστημα διαδραστικών αποδείξεων συνήθως ορίζει ένα πρωτόκολλο, δηλαδή πώς λαμβάνουν χώρα οι γύροι της επεξεργασίας για την περίπτωση $x \in A$. Καθορίζει επίσης και τον 'ιδανικό' Prover για την απαίτηση (1) του ορισμού. Για να επιβεβαιώσει κανείς την απαίτηση (2) του ορισμού, πρέπει να αναλύσει την πιθανοτική συμπεριφορά του πρωτοκόλλου για κάθε τυχαίο Prover. Παρατηρήστε ότι ο ορισμός δεν απαιτεί πολυωνυμικό χρόνο για τον Prover, αλλά έχει αποδειχθεί ότι ένας Prover με πολυωνυμικά φραγμένο χώρο αρκεί, επομένως $IP \subseteq PSPACE$.

Είναι προφανές από τον ορισμό της κλάσης IP ότι $NP \subseteq IP$: O Prover λύνει το πρόβλημα και δίνει τη λύση στον Verifier. Αυτός σε πολυωνυμικό χρόνο, και μάλιστα χωρίς τη χρήση της τυχαίας επιλογής τιμής για μεταβλητή, αποφασίζει χωρίς σφάλμα αν αυτό που του έδωσε ο Prover είναι λύση η όχι.

Το **GI** είναι στο NP και επομένως είναι και στο IP. Αυτό είναι προφανές. Αυτό που είναι εντυπωσιακό, εκ πρώτης όψης τουλάχιστον, είναι ότι το $\overline{\mathbf{GI}}$ έχει διαδραστική απόδειξη, παρότι δεν είναι γνωστό αν $\overline{\mathbf{GI}} \in NP$.

Πρόταση 4.11. $\overline{\mathbf{GI}} \in IP$.

Απόδειξη. Δίνουμε το παρακάτω πρωτόκολλο: Με είσοδο (G_1, G_2), γράφοι με n κορυφές, ο Verifier ξεκινά με τον ακόλουθο αλγόριθμο:

guess randomly $i \in \{1, 2\}$;
guess randomly $\pi \in S_n$;
$H := \pi(G_i)$;

Στη συνέχεια τοποθετεί το H στο κανάλι επικοινωνίας και ο Prover υπολογίζει κάποιο $j \in \{1, 2\}$ και το στέλνει στον Verifier (σκοπός είναι ο Prover να βρει από ποιόν γράφο προέκυψε ο H).Τέλος ο Verifier αποδέχεται αν και μόνο αν j=i. Αναλύουμε το πρωτόκολλο:

- Αν $G_1 \not\equiv G_2$ δηλαδή $(G_1, G_2) \in \overline{\mathbf{GI}}$)
  Τότε υπάρχει ένας αλγόριθμος για τον Prover ο οποίος κάνει τον Verifier να αποδέχεται. Απλά υπολογίζει το j ως :

$$j = \begin{cases} 1, & \text{if } H \equiv G_1 \\ 2, & \text{if } H \equiv G_2 \end{cases}$$

- Αν $G_1 \equiv G_2$ δηλαδή $(G_1, G_2) \notin \overline{\mathbf{GI}}$)
  Τότε όποιον αλγόριθμο κι αν χρησιμοποιήσουμε σαν Prover, η πιθανότητα ο Verifier να αποδέχεται το αποτέλεσμα του Prover είναι 1/2, αφού οι δύο γράφοι είναι ισομορφικοί. Όμως αυτό δεν είναι αρκετό, αφού πρέπει για κάθε Prover η πιθανότητα ο Verifier να αποδεχτει ενώ η είσοδος δεν ανήκει στο πρόβλημα, να είναι μικρότερη ή ίση του 1/4 και για αυτόν που έχουμε είναι τουλάχιστον 1/2. Αυτό που κάνουμε για να ρίξουμε την πιθανότητα στο 1/4 είναι ο Verifier να διαλέγει τυχαία i_1, i_2 και π_1, π_2 και ο Prover πρέπει να στείλει j_1, j_2 τέτοια ώστε $i_1 = j_2$ και $i_2 = j_2$. Τώρα η πιθανότητα εσφαλμένης αποδοχής από τον Verifier είναι το πολύ 1/4.

□

Παρατηρούμε ότι στο παραπάνω πρωτόκολλο είναι απαραίτητο οι τυχαίες επιλογές του Verifier για τα i_1 και i_2 να κρατηθούν μυστικές από τον Prover. Λέμε ότι ο Verifier έχει *ιδιωτικά νομίσματα*(private coins). Αποδεικνύεται ότι κάθε IP πρωτόκολλο μπορεί να επαναδιατυπωθεί ως ένα πρωτόκολλο δημοσίου νομίσματος (public coin protocol)[GS89].

Για καιρό η σχέση της κλάσης IP με την πολυωνυμική ιεραρχία δεν είχε διαλευκανθεί. Η απάντηση έγινε γνωστή το 1990 από τον A. Shamir. Την παραθέτουμε χωρίς απόδειξη.

Θεώρημα 4.12. *[Sha90]IP = PSPACE.*

Αυτό το αποτέλεσμα φαίνεται αρχικά να αναιρεί τη σημασία της πρότασης 4.11 αφού $\overline{\mathbf{GI}} \in co-NP$ και $co-NP \in PSPACE$. Πρέπει να θυμηθούμε όμως ότι ο ορισμός της IP επιτρέπει πολυωνυμικό αριθμό γύρων υπολογισμού, ενώ στο πρωτόκολλο που ορίσαμε για το $\overline{\mathbf{GI}}$ χρησιμοποιήσαμε σταθερό αριθμό από γύρους. Έστω IP(k) η κλάση διαδραστικών αποδείξεων με k γύρους (όπου k σταθερά). Στο [Bab80] (σε συνδυασμό με το [GS89]), αποδεικνύεται ότι IP(k)=IP(2) για κάθε $k \geq 2$. Επομένως από την πρόταση 4.11 προκύπτει ένα πιο ισχυρό αποτέλεσμα:

Πόρισμα 4.13. *[GMR89]* $\overline{\mathbf{GI}} \in IP(2)$.

4.3.2 Μηδενική Γνώση

Η έννοια των διαδραστικών αποδείξεων έχει γίνει γνωστή και δημοφιλής εξ'αιτίας της ιδιότητας μηδενικής γνώσης που μπορεί να έχει ένα σύστημα διαδραστικών αποδείξεων, υπό κάποιες προϋποθέσεις. Για το **GI** για παράδειγμα, το προφανές πρωτόκολλο είναι ο Prover να πάρει τους 2 γράφους της εισόδου και να υπολογίσει έναν ισομορφισμό φ αν υπάρχει τέτοιος και τον στέλνει στον Verifier. Αυτός με τη σειρά του επαληθεύει αν αυτό που του έστειλε ο Prover είναι ισομορφισμός. Στο πρωτόκολλο αυτό όμως ο Prover δίνει στον Verifier πληροφορία, δεν είναι πρωτόκολλο μηδενικής γνώσης. Στη συνέχεια δίνουμε ένα πρωτόκολλο για το **GI** το οποίο είναι μηδενικής γνώσης:

Με είσοδο (G_1, G_2), γράφοι με n κορυφές, ο Prover ξεκινά μαντεύοντας τυχαία $i \in \{1, 2\}$ και $\pi \in S_n$ και στη συνέχεια στέλνει το $H := \pi(G_i)$ στον Verifier. Ο Verifier διαλέγει τυχαία μια τιμή $j \in \{1, 2\}$ και τη στέλνει στον Prover. Στη συνέχεια ο Prover δίνει στον Verifier μια μετάθεση φ την οποία ο Verifier ελέγχει για την ιδιότητα $\varphi(G_j) = H$. Αυτός ο έλεγχος καθορίζει αν ο Verifier αποδέχεται ή όχι.

Με άλλα λόγια ο Prover αρχικά στέλνει στον Verifier ένα τυχαίο στοιχείο από το σύνολο

$$\{H|(H \equiv G_1) \text{ or } (H \equiv G_2)\},$$

στη συνέχεια ο Verifier ζητά από τον Prover να του δώσει έναν ισομορφισμό μεταξύ του H και του G_j όπου είναι ελεύθερος να επιλέξει το j.

Αν $G_1 \equiv G_2$ ο Prover μπορεί πάντα να ικανοποιήσει την απαίτηση του Verifier. Σε αντίθετη περίπτωση έχει 50% πιθανότητα. Επειδή θέλουμε να έχουμε μικρότερη πιθανότητα χρησιμοποιούμε την ίδια ιδέα που χρησιμοποιήσαμε στην απόδειξη της πρότασης 4.11, δηλαδή επαναλαμβάνουμε το πρωτόκολλο δύο φορές, οπότε η πιθανότητα ο Prover να οδηγήσει τον Verifier να αποδεχθεί ενώ $G_1 \not\equiv G_2$ γίνεται το πολύ 1/4.

Με αυτό το πρωτόκολλο ο Prover δεν δίνει καμία πληροφορία στον Verifier πέρα από το αν οι δύο γράφοι είναι ισομορφικοί ή όχι. Δεν βοηθά δηλαδή τον Verifier να βρει έναν ισομορφισμό.

4.4 Πιθανοτικές Κλάσεις

Πριν μπορέσουμε να αποδείξουμε περαιτέρω αποτελέσματα για τον ισομορφισμό γράφων θα χρειαστεί να εισάγουμε κάποιες βασικές έννοιες περί πιθανοτικών κλάσεων πολυπλοκότητας (probabilistic complexity classes). Θα επικεντρωθούμε στις κλάσεις BPP, RP, MA και AM ([Zac82], [Zac86b], [Zac86a], [ZF87]). Από την προηγούμενη ενότητα έχουμε εισάγει την έννοια του πιθανοτικού αλγόριθμου που είναι ένας ντετερμινιστικός αλγόριθμος που έχει επιπλέον τη δυνατότητα να διαλέγει τυχαία ένα στοιχείο από ένα πεπερασμένο σύνολο με ομοιόμορφη κατανομή πιθανότητας.

Αν το αποτέλεσμα ενός τέτοιου αλγορίθμου μπορεί να είναι 'accept' ή 'reject', τότε για κάθε είσοδο x ο αλγόριθμος καθορίζει μια δυαδική τυχαία μεταβλητή. Θα γράφουμε $M(x) = accept$ και θα εννοούμε ότι ο αλγόριθμος M για είσοδο x αποδέχεται. Θα μας απασχολήσουν τέτοιοι αλγόριθμοι που τρέχουν σε πολυωνυμικό χρόνο.

Ορισμός 4.14. *Η κλάση BPP(bounded-error probabilistic polynomial time) αποτελείται από όλα τα σύνολα A για τα οποία υπάρχει πιθανοτικός αλγόριθμος πολυωνυμικού χρόνου M(polynomial-time, probabilistic algorithm) έτσι ώστε για όλες τις εισόδους x,*

$$\begin{array}{lcl} x \in A & \Rightarrow & Prob[M(x) = accept] \geq 3/4 \\ x \notin A & \Rightarrow & Prob[M(x) = accept] \leq 1/4 \end{array}$$

Αν φανταστούμε το σύνολο των πιθανών υπολογισμών ενός πιθανοτικού αλγόριθμου σαν ένα δένδρο, κάθε υπολογισμός ξεκινά από τη ρίζα του δένδρου και ακολουθεί ένα μονοπάτι ως κάποιο φύλλο. Αν ο υπολογισμός αποδέχεται την είσοδο τότε λέμε ότι το φύλλο είναι τύπου 'accept'. Σε αντίθετη περίπτωση είναι τύπου 'reject'. Ο παραπάνω ορισμός λέει ποιοτικά ότι αν κάποια είσοδος x ανήκει στο πρόβλημα A, τότε τα περισσότερα φύλα του δένδρου θα είναι τύπου 'accept'. Αντιθέτως, αν δεν ανήκει, τα περισσότερα φύλλα θα είναι τύπου 'reject'. Παρακάτω θα συμβολίζουμε με $M(x, y)$ το αποτέλεσμα ($\in \{`accept', `reject'\}$) του αλγορίθμου M για είσοδο x και για υπολογισμό που οδηγεί στο φύλο y.

Όπως και στον ορισμό της κλάσης IP, η επιλογή των σταθερών 3/4 και 1/4 είναι αυθαίρετη. Στην πραγματικότητα αρκεί να έχουν τη μορφή 1/2+ε και 1/2-ε αντίστοιχα, για κάθε σταθερό $\varepsilon > 0$.

Θα ορίσουμε εδώ έναν νέο ποσοδείκτη ο οποίος ονομάζεται overwhelming majority quantifier. Τον συμβολίζουμε ως $\exists^+$ και η σημασία μιας έκφρασης της μορφής $\exists^+ yP(y)$ είναι ότι η πιθανότητα $Prob[P(y)] \geq 1/2 + \varepsilon$ για κάθε σταθερό ε, $0 < \varepsilon < 1/2$. Με αυτό το νέο ποσοδείκτη η κλάση BPP ορίζεται εκ' νέου ως εξής:

Ορισμός 4.15. *Η κλάση BPP(bounded-error probabilistic polynomial time) αποτελείται από όλα τα σύνολα A για τα οποία υπάρχει πιθανοτικός αλγόριθμος πολυωνυμικού χρόνου M(polynomial-time, probabilistic algorithm) έτσι ώστε για όλες τις εισόδους x,*

$$\begin{aligned} x \in A &\Rightarrow \exists^{+} y : M(x,y) = \text{`}accept\text{'} \\ x \notin A &\Rightarrow \exists^{+} y : M(x,y) = \text{`}reject\text{'} \end{aligned}$$

Σύμφωνα με το συμβολισμό που εισάγαμε στην αρχή του κεφαλαίου το BPP γράφεται και ως $BPP = (\exists^{+}/\exists^{+})$. Στο [Zac86a] αποδεικνύεται ότι $BPP = (\exists^{+}\forall/\forall\exists^{+})$. Πρόκειται για έναν αποφασιστικό (decisive) ορισμό του BPP. Ο ορισμός $BPP = (\exists^{+}/\exists^{+})$ δεν είναι αποφασιστικός γιατί $\exists y : P(x,y)$ και $\exists y : \neg P(x,y)$ δεν είναι αντιφατικά.

Βλέπουμε ότι έτσι όπως ορίστηκε η κλάση BPP ο πιθανοτικός αλγόριθμος επιτρέπεται να κάνει λάθος και όταν μια είσοδος ανήκει στο πρόβλημα, και όταν μια είσοδος δεν ανήκει στο πρόβλημα. Στην πράξη θα δούμε ότι συχνά οι αλγόριθμοι που σχεδιάζουμε κάνουν λάθος μόνο στη μία από τις δύο περιπτώσεις. Για να λάβουμε υπόψη αυτό το φαινόμενο ορίζουμε παρακάτω την κλάση RP που επιτρέπεται να σφάλει μόνο αν η είσοδος ανήκει στο πρόβλημα.

Ορισμός 4.16. *Η κλάση RP αποτελείται από όλα τα σύνολα A για τα οποία υπάρχει πιθανοτικός αλγόριθμος πολυωνυμικού χρόνου M(polynomial-time, probabilistic algorithm) έτσι ώστε για όλες τις εισόδους x,*

$$\begin{aligned} x \in A &\Rightarrow \exists^{+} y : M(x,y) = \text{`}accept\text{'} \\ x \notin A &\Rightarrow \forall y : M(x,y) = \text{`}reject\text{'} \end{aligned}$$

Συμβολικά $RP = (\exists^{+}/\forall)$.

Για την κλάση co-RP επομένως θα επιτρέπεται να γίνεται σφάλμα στην περίπτωση που η είσοδος δεν ανήκει στο πρόβλημα ($co - RP = (\forall/\exists^{+})$). Συμβολίζουμε με $ZPP = RP \cap co - RP$ την τομή των RP και co-RP. Ισχύουν οι σχέσεις:

$$P \subseteq ZPP \subseteq RP \subseteq NP$$

$$RP \cup co - RP \subseteq BPP$$

4.4.1 Ενίσχυση Πιθανότητας

Δεδομένου ενός πιθανοτικού αλγορίθμου M και ενός περιττού αριθμού t θεωρούμε τον αλγόριθμο 4.4.1 που ονομάζεται και πλειοψηφικό ψήφισμα.

Κάθε επανάληψη του βρόγχου **for** γίνεται με ανεξάρτητες τυχαίες επιλογές. Ο χρόνος εκτέλεσης του αλγορίθμου είναι $\mathcal{O}(t \cdot T_M(n))$, όπου T_M είναι ο χρόνος εκτέλεσης του M. Διαλέγοντας το t ως πολυωνυμική συνάρτηση του

Αλγόριθμος 4.2 Iter(M,t)

```
input x;
n := 0;
for i := 1 to t do
  run M on input x and let the result be r ∈ {accept, reject}
  if r = accept then
    n := n + 1;
  end if
end for
if n > t/2 then
  accept
else
  reject
end if
```

μήκους της εισόδου ($|x|$), η απόδοση του αλγορίθμου Iter(M,t) σε σχέση με την πιθανότητα λάθους βελτιώνεται θεαματικά. Το κόστος είναι μεγαλύτερος χρόνος εκτέλεσης, η αύξηση γίνεται όμως μόνο κατά έναν πολυωνυμικό παράγοντα.

Ας συμβολίσουμε με $Correct(M, x, A)$ την περίπτωση ο αλγόριθμος M για μια είσοδο x να απαντά σωστά για το αν $x \in A$. Από τον αρχικό ορισμό για το BPP έχουμε ότι $Prob[Correct(M, x, A)] \geq 3/4$. Θα αναλύσουμε την πιθανότητα του $Correct(Iter(M, t), x, A)$ για τυχαίο t.

Λήμμα 4.17. *Έστω E ένα ενδεχόμενο που λαμβάνει χώρα με πιθανότητα $1/2 + \varepsilon$, $0 < \varepsilon \leq 1/2$. Τότε το E συμβαίνει μέσα σε t ανεξάρτητες δοκιμές (t περιττός) τουλάχιστον $t/2$ φορές, με πιθανότητα τουλάχιστον*

$$1 - \frac{1}{2} \cdot (1 - 4 \cdot \varepsilon^2)^{t/2}.$$

Απόδειξη. Έστω q_i ή πιθανότητα το E να συμβαίνει ακριβώς i φορές σε t ανεξάρτητες δοκιμές. Από τη δυωνυμική κατανομή,

$$\begin{aligned} q_i &= \binom{t}{i} \cdot (1/2 + \varepsilon)^i \cdot (1/2 - \varepsilon)^{t-i} \\ &\leq \binom{t}{i} \cdot (1/2 + \varepsilon)^i \cdot (1/2 - \varepsilon)^{t-i} \cdot \left(\tfrac{1/2+\varepsilon}{1/2-\varepsilon}\right)^{t/2-i}, \text{ for } i \leq t/2 \\ &= \binom{t}{i} \cdot (1/4 - \varepsilon^2)^{t/2} \end{aligned}$$

Τώρα μπορούμε να υπολογίσουμε την πιθανότητα p να συμβεί το E περισσότερες από $t/2$ φορές, ως εξής:

$$\begin{aligned} p &= \Sigma_{i=\lceil t/2\rceil}^{n} q_i \\ &= 1 - \Sigma_{i=0}^{\lfloor t/2\rfloor} q_i \\ &\geq 1 - \Sigma_{i=0}^{\lfloor t/2\rfloor} \binom{t}{i} \cdot (1/4 - \varepsilon^2)^{t/2} \\ &= 1 - 2^{t-1} \cdot (1/4 - \varepsilon^2)^{t/2} \\ &= 1 - \tfrac{1}{2} \cdot (1 - 4\varepsilon^2)^{t/2} \end{aligned}$$

□

Λήμμα 4.18. *Δεδομένου ενός συνόλου $A \in BPP$ και ενός πιθανοτικού αλγορίθμου M για A με $Prob[Correct(M, x, A)] \geq 3/4$, έχουμε για κάθε φθίνουσα συνάρτηση q,*

$$Prob[Correct(Iter(M, c \cdot q(n)), x, A)] \geq 1 - 2^{-q(n)}$$

όπου $c = \frac{2}{\log 4/3} \leq 4.82$.

Απόδειξη. Προκύπτει από έναν στοιχειώδη υπολογισμό κάνοντας χρήση του λήμματος 4.17. Λύνουμε την ανισότητα

$$\frac{1}{2} \cdot (1 - 4 \cdot \varepsilon^2)^{t/2} \leq 2^{-q(n)}$$

για t (με $\varepsilon = 1/4$) και λαμβάνουμε σαν αποτέλεσμα $t \geq \frac{2}{\log 4/3} \cdot q(n)$. □

Θεώρημα 4.19. *(Probability Amplification Theorem)*
(Θεώρημα Ενίσχυσης Πιθανότητας) Έστω $A \in BPP$. Τότε για κάθε πολυώνυμο q υπάρχει ένας πιθανοτικός αλγόριθμος πολυωνυμικού χρόνου M τέτοιος ώστε για κάθε x, $Prob[Correct(M, x, A)] \geq 1 - 2^{-q(|x|)}$.

Απόδειξη. Άμεση από το λήμμα 4.18. □

Η απόδειξη του θεωρήματος δείχνει ότι έναν πιθανοτικό αλγόριθμο M με $Prob[Correct(M, x, A)] \geq 1/2 + \varepsilon$, όπου ε δεν είναι απαραίτητα σταθερά, αλλά μπορεί να είναι και συνάρτηση του μήκους της εισόδου —x—. Η ενίσχυση ώστε $Prob[Correct(M, x, A)] \geq 1 - 2^{-q(|x|)}$ είναι δυνατή χρησιμοποιώντας $t = \mathcal{O}(\varepsilon^2 \cdot q(|x|))$ επαναληπτικά βήματα. Αν είναι $\varepsilon = 1/p(|x|)$ για κάποιο πολυώνυμο p, τότε ο αριθμός των επαναλήψεων t παραμένει πολυωνυμικός.

Συνέπεια του θεωρήματος είναι οι γλώσσες που είναι στο BPP να θεωρούμε ότι μπορούν να υπολογιστούν αποτελεσματικά, μιας και με το θεώρημα της ενίσχυσης πιθανότητας μπορούμε να κάνουμε σε πολυωνυμικό χρόνο την πιθανότητα σφάλματος αμελητέα.

Υπό κάποιες συνθήκες η κατάσταση του BPP που δίνει πιθανότητα λάθους και για τις δύο περιπτώσεις $x \in A$ και $x \notin A$ μπορεί να μετατραπεί σε μια κλάση που σφάλει για τη μία από τις δύο περιπτώσεις, όπως για παράδειγμα η RP. Δίνουμε το παρακάτω θεώρημα χωρίς απόδειξη.

Θεώρημα 4.20. *Έστω $A \in NP \cap BPP$. Αν το A έχει αυτόυπολογίσιμες λύσεις (self-computable solutions) τότε $A \in RP$.*

Πόρισμα 4.21. *Αν* **GI** $\in BPP$ *τότε* **GI** $\in RP$. *Επίσης αν* **GA** $\in BPP$ *τότε* **GA** $\in RP$.

Απόδειξη. Τα **GI** και **GA** έχουν self-computable λύσεις από το πόρισμα 2.12 και το θεώρημα 2.13. ☐

Η σχέση μεταξύ των κλάσεων NP και BPP δεν είναι σαφής. Κατά πάσα πιθανότητα δεν είναι συγκρίσιμες. Ο εγκλεισμός του NP στο BPP έχει συνέπειες που δεν είναι πιθανό να ισχύουν (βλ. επίσης πόρισμα 4.32.)

Πόρισμα 4.22. *[Zac82] [Ko82] Αν $NP \subseteq BPP$ τότε $NP = RP$.*

Απόδειξη. Η υπόθεση ότι $NP \subseteq BPP$ συνεπάγεται ότι **Sat** $\in BPP$. Όμως το **Sat** έχει αυτο-υπολογίσιμες λύσεις. Επομένως, από το θεώρημα 4.20 έχουμε ότι **Sat** $\in RP$, που συνεπάγεται ότι $NP = RP$. ☐

Πόρισμα 4.23. *[Zac82] [Ko82] $BPP(BPP) = BPP$.*

4.4.2 Ο Τελεστής BP

Κάθε φορά που ένας πιθανοτικός αλγόριθμος εκτελεί μια εντολή **guess randomly**, διαλέγει ένα στοιχείο από ένα πεπερασμένο σύνολο S (πολυωνυμικού μεγέθους). Μπορούμε να προσομοιώσουμε αυτή τη συμπεριφορά με έναν ντετερμινιστικό αλγόριθμο ο οποίος έχει πρόσβαση σε μια ξεχωριστή είσοδο r που αποτελείται από ανεξάρτητα τυχαία bits με ομοιόμορφη κατανομή πιθανότητας. Αυστηρά, για κάθε πιθανοτικό αλγόριθμο πολυωνυμικού χρόνου M (που αποφασίζει αν ένα σύνολο A ανήκει στο BPP), υπάρχει πολυωνυμικός ντετερμινιστικό αλγόριθμος δύο εισόδων M', τέτοιος ώστε

$$Prob[(M' \text{ on input } x \text{ and } r \text{ accepts}) \Leftrightarrow x \in A] \geq 3/4$$

όπου η πιθανότητα υπολογίζεται επί όλων των πιθανών εισόδων r. Παρατηρούμε ότι για είσοδο x αρκεί να περιορίσουμε το μήκος του r σε κάποιο πολυώνυμο του $|x|$.

Ένας ντετερμινιστικός αλγόριθμος πολυωνυμικού χρόνου όπως ο M' είναι το υπολογιστικό μοντέλο στο οποίο στηρίζεται η κλάση P. Επομένως, η κλάση BPP μπορεί να θεωρηθεί σαν μια πιθανοτική επέκταση της κλάσης P. Στη συνέχεια θα ασχοληθούμε με πιθανοτικές επεκτάσεις αυθαίρετων κλάσεων και ειδικά της κλάσης NP. Συνεπώς θα θεωρήσουμε τη μετάβαση από το P στο BPP ως το αποτέλεσμα της εφαρμογής ενός τελεστή (παρόμοιο με τους τελεστές $\mathcal{N}$ και $co-\mathcal{N}$ που εισάγαμε νωρίτερα) στην κλάση P.

Ορισμός 4.24. *Έστω $\mathcal{C}$ μια κλάση συνόλων. Τότε $\mathcal{BP}\cdot\mathcal{C}$ είναι η κλάση των συνόλων A για τα οποία υπάρχει ένα σύνολο $B \in \mathcal{C}$ και ένα πολυώνυμο p έτσι ώστε για κάθε x,*

$$\begin{cases} x \in A \Rightarrow \exists^+ y : B(x,y) \\ x \notin A \Rightarrow \exists^+ y : \neg B(x,y) \end{cases}$$

όπου οι ποσοδείκτες είναι πολυωνυμικά φραγμένοι από το p.

Πρόταση 4.25. *Για κάθε κλάση* **C** *και $\mathcal{D}$,*

1. *$\mathcal{C} \subseteq \mathcal{D} \Rightarrow BP\cdot\mathcal{C} \subseteq BP\cdot\mathcal{D}$.*
2. *$co - BP\cdot\mathcal{C} = BP\cdot co - \mathcal{C}$ (δηλαδή αν η $\mathcal{C}$ είναι κλειστή ως προς συμπλήρωμα, τότε το ίδιο ισχύει και για την $BP\cdot\mathcal{C}$).*
3. *$\mathcal{C} \subseteq BP\cdot\mathcal{C}$ εφόσον η $\mathcal{C}$ έχει την ακόλουθη ιδιότητα: $A \in \mathcal{C} \Rightarrow A \times \Sigma^* \in \mathcal{C}$.(Αυτή η ιδιότητα ικανοποιείται αν η κλάση $\mathcal{C}$ είναι κλειστή για Karp αναγωγή.)*
4. *$BP\cdot\mathcal{C} \subseteq BPP(\mathcal{C})$.*

Το θεώρημα ενίσχυσης πιθανότητας παίζει σημαντικό ρόλο στις αποδείξεις ιδιοτήτων για την κλάση BPP. Συνεπώς είναι ενδιαφέρον να δει κανείς για ποιες κλάσεις $\mathcal{C}$ η κλάση $BP\cdot\mathcal{C}$ έχουν αυτή την ιδιότητα. Θα χρειαστούμε μια νέα αναγωγή, την αναγωγή πλειοψηφίας.

Ορισμός 4.26. *Ένα σύνολο A ανάγεται πλειοψηφικά στο σύνολο B ($A \leq^p_{maj} B$) αν υπάρχει μια συνάρτηση f υπολογίσιμη σε πολυωνυμικό χρόνο η οποία απεικονίζει strings σε ακολουθίες από strings έτσι ώστε για κάθε x, αν $f(x) = (y_1, \ldots, y_k)$ τότε*

$$x \in A \Leftrightarrow y_i \in B \text{ for more than k/2 many i's}$$

Πρόταση 4.27. *Αν η κλάση πολυπλοκότητας $\mathcal{C}$ είναι κλειστή ως προς αναγωγή πλειοφηφίας (δηλαδή $(A \leq^p_{maj} B \wedge B \in \mathcal{C}) \Rightarrow A \in \mathcal{C}$), τότε η κλάση $BP\cdot\mathcal{C}$ έχει την ιδιότητα της ενίσχυσης πιθανότητας, δηλαδή για κάθε σύνολο $A \in BP\cdot\mathcal{C}$ και για κάθε πολυώνυμο p υπάρχει ένα σύνολο $B \in \mathcal{C}$ τέτοιο ώστε για κάθε x,*

$$\begin{cases} x \in A \;\; \Rightarrow \;\; Prob[(x,r) \in B] \geq 1 - 2^{-p(|x|)}, \\ x \notin A \;\; \Rightarrow \;\; Prob[(x,r) \in B] \leq 2^{-p(|x|)} \end{cases}$$

όπου το string r επιλέγεται τυχαία με ομοιόμορφη κατανομή επί των strings κατάλληλου πολυωνυμικού μήκους.

Εύκολα μπορεί να διαπιστώσει κανείς ότι οι κλάσεις P, NP, Σ^p_k, Π^p_k και όλοι οι συνδυασμοί αυτών είναι κλειστές για την αναγωγή πλειοψηφίας.

Πρόταση 4.28. *Ιδιότητες του τελεστή BP:*

1. $BP \cdot BP \cdot \mathcal{C} = BP \cdot \mathcal{C}$*(βλ. επίσης πόρισμα 4.23)*
2. $\mathcal{N} \cdot BP \cdot \mathcal{C} \subseteq BP \cdot \mathcal{N} \cdot \mathcal{C}$
3. $(co - \mathcal{N}) \cdot BP \cdot \mathcal{C} \subseteq BP \cdot (co - \mathcal{N}) \cdot \mathcal{C}$
4. $BP \cdot \mathcal{C} \subseteq \mathcal{N} \cdot (co - \mathcal{N}) \cdot \mathcal{C} \cap (co - \mathcal{N}) \cdot \mathcal{N} \cdot \mathcal{C}$

Πόρισμα 4.29. $BPP \subseteq \Sigma_2^p \cap \Pi_2^p$.

Απόδειξη. Για το τέταρτο σημείο της πρότασης 4.28 θέτουμε $\mathcal{C} = P$. □

Πόρισμα 4.30. $BP \cdot \Sigma_k^p \subseteq \Pi_{k+1}^p$.

Απόδειξη. Για το τέταρτο σημείο της πρότασης 4.28 θέτουμε $\mathcal{C} = \Sigma_k^p$ και έχουμε:
$BP \cdot \Sigma_k^p \subseteq \Sigma_{k+2} \cap \Pi_{k+1}^p \Rightarrow BP \cdot \Sigma_k^p \subseteq \Pi_{k+1}^p$ □

Πόρισμα 4.31. $\Sigma_2^p(BP \cdot \Sigma_k^p) = \Sigma_{k+2}^p$.

Απόδειξη. $\Sigma_2^p(BP \cdot \Sigma_k^p) = \mathcal{N} \cdot (co - \mathcal{N}) \cdot P(BP \cdot \Sigma_k^p) \subseteq \mathcal{N} \cdot (co - \mathcal{N}) BPP(BP \cdot \Sigma_k^p) = \mathcal{N} \cdot (co - \mathcal{N}) \cdot BP \cdot P(\Sigma_k^p)$(βλ. πόρισμα 4.23)$\subseteq \mathcal{N} \cdot BP \cdot (co - \mathcal{N}) \cdot P(\Sigma_k^p)$ (βλ. πρόταση 4.28)$\subseteq \mathcal{N} \cdot \mathcal{N} \cdot (co - \mathcal{N}) \cdot (co - \mathcal{N}) \cdot P(\Sigma_k^p)$(βλ. πρόταση 4.28$= \Sigma_{k+2}^p$. □

Είδαμε στο πόρισμα 4.22 ότι αν $NP \subseteq BPP$ τότε $NP = RP$. Στο ακόλουθο πόρισμα δείχνουμε επιπλέον ότι αν $NP \subseteq BPP$, τότε $PH \subseteq BPP$, και συνεπώς, από το πόρισμα 4.29, η πολυωνυμική ιεραρχία καταρρέει στο επίπεδο $\Sigma_2^p = \Pi_2^p = BPP$.

Πόρισμα 4.32. *[Ko82] Αν $NP \subseteq BPP$, τότε $PH \subseteq BPP$.*

Το αποτέλεσμα αυτό γενικεύεται στο ακόλουθο θεώρημα:

Θεώρημα 4.33. *Αν $\Pi_k^p \subseteq BP \cdot \Sigma_k^p$, τότε $PH = BP \cdot \Sigma_k^p = \Sigma_{k+1}^p$*

4.4.3 Παιχνίδια Arthur-Merlin

Στην παρούσα ενότητα θα μας απασχολήσει η κλάση $BP \cdot NP$. Αν την φανταστούμε με τη μορφή παιγνίου Prover-Verifier μπορεί να ερμηνευτεί ως εξής:δεδομένου ενός συνόλου $A \in BP \cdot NP$, έχουμε ένα σύνολο $B \in P$ τέτοιο ώστε:

$$\begin{cases} x \in A \Rightarrow Prob[\exists y (x, r, y) \in B] \geq 3/4, \\ x \notin A \Rightarrow Prob[\exists y (x, r, y) \in B] \leq 1/4, \end{cases}$$

όπου η πιθανότητα λαμβάνεται ομοιόμορφα από όλα τα strings r κάποιου πολυωνυμικού μήκους ως προς το μήκος $|x|$. Αυτό σημαίνει ότι για να αποδειχθεί ότι ένα string x ανήκει στο A, ο Verifier ξεκινά διαλέγοντας ένα τυχαίο string r (πολυωνυμικού μήκους) και το στέλνει στον Prover. Αυτός με τη σειρά του βρίσκει ένα string y (πολυωνυμικού μήκους) και το στέλνει πίσω στον Verifier ο οποίος τελικά ελέγχει αν $(x, r, y) \in B$.

Παρατηρούμε ότι πρόκειται για μια ειδική περίπτωση του IP(2). Ο περιορισμός που έχουμε εδώ είναι ότι ο Verifier δεν επιτρέπεται να κάνει *ιδιωτικούς* υπολογισμούς με το r και να το αποκαλύπτει έμμεσα. Αντιθέτως πρέπει να το στείλει αμέσως μόλις το υπολογίσει. Δηλαδή οι 'ρίψεις κερμάτων' του Verifier είναι δημόσιες. Παρατηρήστε ότι η φαινομενικά πιο ισχυρή περίπτωση του να έχει ο Verifier 'ιδιωτικές ρίψεις' μοιάζει να είναι ζωτικής σημασίας για τα πρωτόκολλα των $\overline{\mathbf{GI}}$ και $\overline{\mathbf{GA}}$ που έχουμε δει.

Αυτό το είδος διαδραστικών αποδείξεων με μόνο δημόσιες ρίψεις νομισμάτων έχει προταθεί από τον Babai [Bab80] [BM88] για την κατηγοριοποίηση προβλημάτων της θεωρίας ομάδων. Συμβολίζει την κλάση $BP \cdot NP$ με AM, όπου το A συμβολίζει τον βασιλιά Arthur και το M τον μάγο Merlin. Αυτά είναι απλά άλλα ονόματα για τους Verifier και Prover αντίστοιχα. Ανάλογα με το ποιος παίκτης ξεκινά και με το πόσους γύρους έχει το παιχνίδι ορίζονται κλάσεις όπως οι M, A, MA, AM, MAM, AMA, MAMA, AMAM, ... Είναι σαφές ότι η κλάση M είναι η ίδια με την NP και η A είναι η ίδια με την BPP. Επιπλέον, η παραπάνω συζήτηση δείχνει ότι $AM = BP \cdot NP$. Τέλος, η κλάση MA είναι μια άλλη ανεξάρτητη ενδιαφέρουσα κλάση με την οποία θα ασχοληθούμε αργότερα.

Ο Babai δείχνει ότι αυτή η "Arthur-Merlin" ιεραρχία καταρρέει στην κλάση AM [Bab80][BM88] οπότε μόνο οι κλάσεις M=NP, A=BPP, MA και AM (και οι co-NP, co-MA και co-AM) ενδεχομένως να είναι διαφορετικές μεταξύ τους. Η απόδειξη αυτή μπορεί να βρεθεί και σε πιο σύντομη και κατανοητή μορφή στο [Zac88]. 'Οπως στην περίπτωση της IP, το παιχνίδι μπορεί να επεκταθεί σε πολυωνυμικό αριθμό γύρων. Συμβολίζουμε αυτή την κλάση με $AM(poly)$.

Οι Goldwasser και Sipser έδειξαν ότι οι ιδιωτικές ρίψεις μπορούν να προσομοιωθούν με δημόσιες ρίψεις και συνεπώς $IP = AM(poly)$ και $IP(2) = AM$. Από το πόρισμα 4.30 έχουμε $AM \subseteq \Pi_2^p$ και επομένως $IP(2) \subseteq \Pi_2^p$. Μια ακόμα συνέπεια του αποτελέσματος των Goldwasser και Sipser είναι ότι το $\overline{\mathbf{GI}}$ ανήκει στην AM.

Θεώρημα 4.34. $\overline{\mathbf{GI}} \in AM$.

Πόρισμα 4.35. *[BHZ87] [Sch88] Αν το* **GI** *είναι NP-Complete τότε η πολυωνυμική ιεραρχία καταρρέει στο δεύτερο επίπεδο* $\Sigma_2^p = \Pi_2^p = AM$.

Απόδειξη. Αν το **GI** είναι NP-Complete, τότε το $\overline{\mathbf{GI}}$ είναι co-NP-Complete.

Από το θεώρημα 4.34 $\overline{\mathbf{GI}} \in AM$ και επομένως $\Pi_1^p = co - NP \subseteq AM = BP \cdot NP = BP \cdot \Sigma_1^p$. Από το θεώρημα 4.33 προκύπτει το αποδεικτέο. □

4.5 Lowness and Collapses

Μερικές φορές όταν δεν μπορούμε να συγκρίνουμε απ'ευθείας δυο κλάσεις πολυπλοκότητας μπορούμε να τις συγκρίνουμε με έμμεσο τρόπο χρησιμοποιώντας μαντεία.

Ορισμός 4.36. *Ένα σύνολο A λέμε ότι είναι χαμηλό(low) για μια κλάση $\mathcal{C}$ αν $\mathcal{C}(A) = \mathcal{C}$. Αυτό σημαίνει ότι το μαντείο A μπορεί να αντικατασταθεί από υπολογισμούς μέσα στην κλάση $\mathcal{C}$.*

Αν η πολυπλοκότητα ενός προβλήματος δεν έχει διαλευκανθεί ακόμα, όπως για το **GI** για το οποίο δεν ξέρουμε αν είναι στο P ή αν είναι NP-Complete, η χαμηλότητα (lowness) μπορεί να δώσει κάποια πληροφορία.

Η χαμηλότητα προκύπτει συνήθως με δύο τρόπους. Ο πρώτος αν το σύνολο A (ή η κλάση στην οποία ανήκει) έχει χαμηλή πολυπλοκότητα. Ο δεύτερος αν το A έχει χαμηλή περιεκτικότητα πληροφορίας, όπως στην περίπτωση συνόλων επί αλφαβήτων ενός γράμματος.

Για την πολυωνυμική ιεραρχία για $k \geq 1$ και για κάθε σύνολο $A \in NP$ ισχύει η ακόλουθη σχέση:

$$\Sigma_k^p \overset{(1)}{\subseteq} \Sigma_k^p(A) \overset{(2)}{\subseteq} \Sigma_{k+1}^p$$

Αν ισχύει ισότητα στην (1) για κάποιο k, τότε το A είναι *χαμηλό*(low) για την Σ_k^p. Αντίθετα αν ισχύει η ισότητα στο (2) μπορούμε να πούμε ότι το A είναι *ψηλό*(high). Τέλος μπορεί να μην ισχύει καμία από τις ισότητες (1) και (2), οπότε καλούμε το σύνολο A *ενδιάμεσο*(intermediate). Παρατηρήστε ότι όλα τα σύνολα A στο P κάνουν την ισότητα (1) να ισχύει για κάθε k, ενώ όλα τα σύνολα A που είναι NP-Complete κάνουν την ισότητα (2) να ισχύει.

Ορισμός 4.37. *Ένα σύνολο $A \in NP$ είναι $low_k, k \geq 0$, (συμβολικά $A \in L_k$) αν $\Sigma_k^p(A) = \Sigma_k^p$. Ένα σύνολο A είναι $high_k, k \geq 0$, (συμβολικά $A \in H_k$) αν $\Sigma_k^p(A) = \Sigma_{k+1}^p$. Η κλάση $LH = \bigcup_{k \geq 0} L_k$ ονομάζεται* low hierarchy*(χαμηλή ιεραρχία) στο NP και η κλάση $HH = \bigcup_{k \geq 0} H_k$ ονομάζεται* high hierarchy*(υψηλή ιεραρχία) στο NP.*

Είναι εμφανές ότι έχουμε τις σχέσεις $L_0 \subseteq L_1 \subseteq \ldots$ και $H_0 \subseteq H_1 \subseteq \ldots$. Για χαμηλές βαθμίδες των ιεραρχιών έχουμε ακριβείς χαρακτηρισμούς των κλάσεων αυτών.

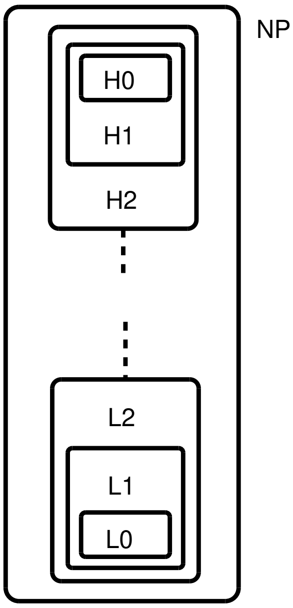
Πρόταση 4.38. *[Sch83] $L_0 = P$ και $L_1 = NP \cap co - NP$.*

Η σχέση μεταξύ των A και B που δίνεται από τη σχέση $A \in NP(B) \cap co-NP(B)$ μπορεί να θεωρηθεί σαν μια ασθενής μορφή αναγωγής. Συμβολικά $A \leq^p_{sn} B$ (strong nondeterministic Turing reducibility). Παρόμοια με την τελευταία πρόταση έχουμε:

Πρόταση 4.39. *[Sch83]*
$H_0 = \{A|A \text{ is NP} - \text{Complete w.r.t. } \leq^p_T - \text{reduction }\}$,
$H_1 = \{A|A \text{ is NP} - \text{Completew.r.t. } \leq^p_{sn} - \text{reduction }\}$.

Τη high hierarchy μπορεί να τη δει κανείς σαν μια γενίκευση της έννοιας NP-πληρότητας. Υπάρχουν παραδείγματα προβλημάτων που είναι στην H_1 αλλά δεν είναι γνωστό αν είναι και στην H_0. Αυτά τα προβλήματα είναι γνωστά και ως *γ-πλήρη*.

Το σχήμα 4.2 δίνει σχηματικά τις ιεραρχίες LH και HH μέσα στο NP.



Σχήμα 4.2: Ιεραρχίες **LH** και **HH**

Δεν είναι γνωστό αν οι LH και HH είναι γνήσιες ιεραρχίες, κάτι το οποίο θα σήμαινε ότι $P \neq NP$, αλλά οι LH και HH είναι ξένες (disjoint) αν και μόνο αν η PH δεν καταρρέει, όπως δηλώνει η παρακάτω πρόταση.

Πρόταση 4.40. *[Sch83]Αν $PH = \Sigma^p_k$ τότε $L_k = H_k = NP$.*
Αν $PH \neq \Sigma^p_k$ για κάποιο k, τότε $L_k \cap H_k = \emptyset$ και επιπλέον $NP - (L_k \cup H_k) \neq \emptyset$. Επιπλέον αν $PH \neq \Sigma^p_k$ για κάθε k, τότε $LH \cap HH = \emptyset$ και $NP - (LH \cup HH) \neq \emptyset$.

Από τα μέχρι στιγμής αποτελέσματα $[BPP(BPP) = BPP, NP(NP \cap co - NP) = NP, \Sigma_2^p(BPP) = \Sigma_2^p]$ και την παραπάνω πρόταση προκύπτει ότι κανένα NP-Complete σύνολο δεν μπορεί να ανήκει στο BPP ή στην τομή $\Delta NP = NP \cap co - NP$ εκτός κι αν η PH καταρρέει στο Σ_2^p.

Στη συνέχεια δείχνουμε ένα πολύ ισχυρό αποτέλεσμα (χωρίς απόδειξη) το οποίο χρησιμεύει ως τελευταίο βήμα για την απόδειξη ότι το **GI** είναι στην low hierarchy (LH).

Θεώρημα 4.41. $AM(AM \cap co - AM) = AM$.

Απόδειξη. Στο [Zac88] αποδεικνύεται ότι $AM = (\exists^+\exists/\exists^+\forall) = (\forall\exists/\exists^+\forall)$. Επομένως $AM(AM \cap co - AM) = (\forall\exists/\exists^+\forall)[(\forall\exists/\exists^+\forall) \cap (\exists^+\forall/\forall\exists) \subseteq (\forall\exists/\exists^+\forall)(\forall\exists/\exists^+\forall) = (\forall\exists\forall\exists/\exists^+\forall\exists^+\forall) \subseteq (\forall\forall\exists\exists/\exists^+\exists^+\forall\forall) = (\forall\exists/\exists^+\forall) = AM$
Άρα $AM(AM \cap co - AM) \subseteq AM)$. Προφανώς ισχύει $AM \subseteq AM(AM \cap co - AM)$ επομένως $AM(AM \cap co - AM) = AM$. □

Πόρισμα 4.42. *Το* **GI** *είναι χαμηλό για την* $AM(\cdot)$, *δηλαδή* $AM(\mathbf{GI}) = AM$.

Απόδειξη. Από το θεώρημα 4.34, το **GI** είναι στην τομή $NP \cap co - AM \subseteq AM \cap co - AM$. □

Πόρισμα 4.43. *Το* **GI** *είναι χαμηλό για την* $\Sigma_2^p(\cdot)$, *δηλαδή* $\Sigma_2^p(\mathbf{GI}) = \Sigma_2^p$, *δηλαδή* $\mathbf{GI} \in L_2$.

Απόδειξη. Ισχύει ότι $co - \mathcal{N} \cdot AM = \Pi_2^p$:
($co - \mathcal{N} \cdot AM \subseteq \Pi_2^p$):co-$\mathcal{N} \cdot AM = co - \mathcal{N} \cdot BP \cdot \mathcal{N} \cdot P \subseteq co - \mathcal{N} \cdot co - \mathcal{N} \cdot \mathcal{N} \cdot \mathcal{N} \cdot P = co - \mathcal{N} \cdot \mathcal{N} \cdot P = \Pi_2^p$
($\Pi_2^p \subseteq co - \mathcal{N} \cdot AM$): $\Pi_2^p = co - \mathcal{N} \cdot NP \subseteq co - \mathcal{N} \cdot AM$.
Αυτό το αποτέλεσμα δουλεύει και με μαντεία, δηλαδή $co - \mathcal{N} \cdot AM(\cdot) = \Pi_2^p(\cdot)$. Εφαρμόζοντας τώρα τον $co - \mathcal{N}$-τελεστή στα δύο μέλη της ισότητας $AM(\mathbf{GI} = AM$, λαμβάνουμε $\Pi_2^p(\mathbf{GI}) = \Pi_2^p$ και παίρνοντας το συμπλήρωμα έχουμε $\Sigma_2^p(\mathbf{GI}) = \Sigma_2^p$. □

Πόρισμα 4.44. *Το* **GI** *δεν ανήκει σε καμία κλάση* H_k *της ιεραρχίας HH, εκτός και αν η πολυωνυμική ιεραρχία PH καταρρέει στο* $\Sigma_{max(k,2)}^p$.

Επομένως ούτε καν ασθενείς έννοιες NP-πληρότητας δεν ισχύουν για το **GI**, αρκεί να μην καταρρέει η πολυωνυμική ιεραρχία.

Μέρος τρίτο

Παραρτήματα

Αναφορές

[Bab80] L. Babai. Trading group theory for randomness. *17th ACM Symp. Theory of Computing*, 9:421–429, 1980.

[BHZ87] R.B. Boppana, J. Hastad, and S. Zachos. Does co-np have short interactive proofs? *Information Processing Letters*, 25:27–32, 1987.

[BM88] L. Babai and S. Moran. Arthur-merlin games: a randomized proof system and a hierarchy of complexity classes. *Journal of Computer and System Sciences*, 36:254–276, 1988.

[GMR89] S. Goldwasser, S. Micali, and C. Rackoff. The knowledge complexity of interactive proof systems. *SIAM Journal of Computing*, 18:186–208, 1989.

[GS89] S. Goldwasser and M. Sipser. *Randomness and Computation*, volume 5 of *Advances in Computing Research*, chapter Private coins versus publis coins in interactive proof systems, pages 73–90. JAI Press, 1989.

[HT72] J. E. Hopcroft and R. E. Tarjan. Isomorphism of planar graphs (working paper). *Complexity of Computer Computations*, pages 147–150, 1972.

[HT96] M. M. Halldórsson and K. Tanaka. Approximation and special cases of common subtrees and editing distance. In *Proc. 7th Ann. Int. Symp. on Algorithms and Computation*, volume 1178 of *Lecture Notes in Comput. Sci.*, pages 75–84. Springer Verlang, 1996.

[HW74] J. E. Hopcroft and J. K. Wong. A linear time algorithm for isomorphism of planar graphs. In *ACM Symposium on Theory of Computing*, pages 172–184, 1974.

[JKMT03] Birgit Jenner, Johannes Köbler, Pierre McKenzie, and Jacobo Torán. Completeness results for graph isomorphism. *Journal of Computer and System Sciences*, 66:549–566, 2003.

[Kar72] R.M. Karp. Reducibility among combinatorial problems. *Complexity of Computer Computations*, pages 85–103, 1972.

[Ko82] K. Ko. Some observations on the probabilistic algorithms and np-hard problems. *Information Processing Letters*, 14:39–43, 1982.

[KST93] Johannes Kobler, Uwe Schoning, and Jacobo Toran. *The Graph Isomorphism Problem: Its Structural Complexity*. Birkhauser Boston, April 1993.

[Lad75] R. E. Ladner. On the structure of polynomial-time reducibilities. *Journal of the Assoc. Comput.Mach.*, 22:155–171, 1975.

[Lub81] A. Lubiw. Some NP-complete problems similar to graph isomorphism. *SIAM Journal of Computing*, 10:11–21, 1981.

[Mil79] G.L. Miller. Graph isomorphism, general remarks. *Journal of Computer and System Sciences*, 18:128–142, 1979.

[Pap94] Christos H. Papadimitriou. *Computational Complexity*. Addison Wesley Publishing Company, January 1994.

[Sch76] C.P. Schnorr. Optimal algorithms for self-transformable problems. In S. Michaelson and R. Milner, editors, *Automata, Languages, and Programming*, pages 322–337. Edinburgh University Press, 1976.

[Sch83] U. Schöning. A low and a high hierarchy within np. *Journal of Computer and Systems Sciences*, 27:14–28, 1983.

[Sch88] U. Schöning. Graph isomorphism is in the low hierarchy. *Journal of Computer and System Sciences*, pages 312–323, 1988.

[Sha90] A. Shamir. IP = PSPACE. *IEEE Symp. Foundations of Computer Science*, pages 11–15, 1990.

[Wei66] L. Weinberg. A simple and effective algorithm for determining isomorphism of planar triply connected graphs. *IEEE Trans. on Circuit Theory*, 13:142–148, 1966.

[Whi33] H. Whitney. A set of topological invariants for graphs. *American Journal of Mathematics*, 55:231–235, 1933.

[Zac82] S. Zachos. Rubustness of probabilistic computational complexity classes under definitional perturbations. *Inrormation and Control*, 54, 1982.

[Zac86a] S. Zachos. A decisive characterisation of BPP. *Information and Control*, 69:125–135, 1986.

[Zac86b] S. Zachos. Probabilistic quantifiers, adversaries, and complexity classes: an overview. In Springer-Verlang, editor, *1st Structure in Complexity Theory*, volume 223 of *Lecture Notes in Computer Science*, pages 383–400, 1986.

[Zac88] S. Zachos. Probabilistic quantifiers and games. *Journal of Computer and System Sciences*, 36(3):433–451, 1988.

[ZF87] S. Zachos and M. Fürer. Probabilistic quantifiers vs. distrustful adversaries. In Springer-Verlang, editor, *Foundations of Software Technology and Theoretical Computer Sciences*, volume 287 of *Lecture Notes in Computer Science*, pages 443–455, 1987.

[ZJ94] K. Zhang and T. Jiang. Some MAX SNP-hard results concerning unordered labeled trees. *Inform. Process. Lett.*, 49:249–254, 1994.