NP-ΠΛΗΡΟΤΗΤΑ ΚΑΙ ΠΡΟΣΕΓΓΙΣΤΙΚΟΙ ΑΛΓΟΡΙΘΜΟΙ

ΚΩΝ/ΝΟΣ ΚΑΡΑΜΠΟΥΛΑΣ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ
υπεύθυνος
Γρηγόριος Καραγιώργος
ΛΑΜΙΑ 2008

Περιεχόμενα

ABSTRACT

This book is an introduction to the theory of computational complexity. Computational complexity is the area of computer science that contemplates the reasons why some problems are so hard to solve by computers. The subjects that are covered in this book are, the complexity theory, Turing machines and the complexity classes, emphasizing to the theory of NP-Completeness and to the approximation algorithms.

For a quick look at the table of contents, Chapter 1 introduces computational complexity, problems and algorithms.Chapter 2 treats Turing machines providing proof techniques necessary for thoroughly understanding and applying the theory. Provides, also, definitions of time and space complexity. Finally, the last part of this chapter covers an introduction to the complexity classes and the relations between them.

Chapter 3 provides an overview of tree important complexity classes of problems, **P**, **NP** and **NP-complete**, emphasizing at the NP-complete. We describe them informally here and we shall define them more formally on chapter 3. The class **P** (polynomial time) consists of those problems that are solvable in polynomial time. More specifically, they are problems that can be solved in time $O(n^k)$ for some constant k, where n is the size of the input to the problem. The class **NP** (nondeterministic polynomial time) consists of those problems that are ''verifiable'' in polynomial time. What we mean here is that if we were somehow given a ''certificate'' of a solution, then we could verify that the certificate is correct in time polynomial in the size of the input to the problem. Any problem in P is also in NP, since if a problem is in P then we can solve it in polynomial time without even being given a certificate. The open question is whether or not P is a proper subset of NP. Perhaps the most compelling reason why theoretical computer scientists believe that $P \neq NP$ is the existence of the class of NP-complete problems. This class has the surprising property that if any NP-complete problem can be solved in polynomial time, then every problem in NP has a polynomial time solution, that is,$P = NP$. Despite, years of study, no polynomial time algorithm has ever been discovered for any NP-complete problem. A problem is characterized NP-Hard if all problems of NP class can be reduced in polynomial time in this. So if a NP-hard problem is and NP, then called **NP-complete**.

This chapter, also, contains a large number of theoretical topics in NP-completeness, such as reductions and the concept of completeness, immediately exemplified by Cook's theorem and Boolean Logic. In the next chapter 4, we report certain important problems that belong in the NP-complete class and are proved their NP-completeness.

Chapter 5 concentrates on the search for efficient approximation algorithms for NP-complete problems, an area whose development has seen considerable interplay with the theory of NP-completeness. In practice, an algorithm that returns near-optimal solutions is called an **approximation algorithm**. This chapter presents polynomial-time approximation algorithms for several NP-complete problems.

In the last capture metricconverterProductID6, in6, in conclusions, discusses various opinions that were exported for the complexity classes, the NP-completeness, and the approximation algorithms.

ΠΡΟΛΟΓΟΣ

Η εργασία αυτή είναι μία εισαγωγή στη θεωρία υπολογισμού. Τα θέματα που καλύπτονται εδώ είναι, εν συντομία, η θεωρία πολυπλοκότητας, η υπολογισιμότητα από μηχανές Turing ,και η υπολογιστική πολυπλοκότητα. Ο κύριος σκοπός αυτής της εργασίας είναι να παρουσιάσουμε εκτενέστερα την πολύ σημαντική κλάση, την NP-πλήρη κλάση, μαζί με πολλά από τα προβλήματά της, όπως και προσεγγιστικούς αλγορίθμους για την αντιμετώπισή τους.

Συγκεκριμένα, το κεφάλαιο 1, περιέχει μία εισαγωγή που αφορά την θεωρία πολυπλοκότητας, τα υπολογιστικά προβλήματα και την πολυπλοκότητα των αλγορίθμων. Αντικείμενο της Θεωρίας Πολυπλοκότητας είναι η μελέτη των υπολογιστικών προβλημάτων, και η κατάταξη τους σε «κλάσεις πολυπλοκότητας» ανάλογα τους υπολογιστικούς πόρους που αυτά απαιτούν για να επιλυθούν αλγοριθμικά. Κύρια εφαρμογή για την ανάπτυξη της θεωρίας πολυπλοκότητας είναι η ύπαρξη προβλημάτων που ενώ είναι επιλύσιμα, δεν είναι γνωστός κάποιος αποδοτικός αλγόριθμος για την επίλυση τους.

Το κεφάλαιο 2 περιγράφει το βασικό υπολογιστικό μοντέλο που χρησιμοποιείται, τη μηχανή Turing, και διάφορες παραλλαγές του. Δίνονται, επίσης, ορισμοί της πολυπλοκότητας χρόνου και χώρου αλγορίθμων και προβλημάτων. Επίσης στο τέλος του κεφαλαίου αυτού γίνεται μία εισαγωγή στις κλάσεις πολυπλοκότητας και στις γενικές σχέσεις μεταξύ αυτών.

Το κεφάλαιο 3 ασχολείται με τις τρεις σημαντικές κλάσεις, **P**, **NP** και **NP-πλήρη**. Στην αρχή γίνεται αναφορά στην κλάση **P** (polynomial time), η οποία είναι η πρώτη κλάση πολυπλοκότητας που συναντάμε. Η κλάση αυτή, όπως θα δούμε σε αυτό το κεφάλαιο, εμπεριέχει όλα τα προβλήματα που λύνονται σε πολυωνυμικό χρόνο. Δηλαδή σε αυτήν ανήκουν τα προβλήματα για τα οποία υπάρχει πολυωνυμικός αλγόριθμος που τα επιλύει. Πιο συγκεκριμένα, ανήκουν τα προβλήματα που λύνονται σε χρόνο $O(n^k)$ όπου k σταθερός ακέραιος και όπου n είναι το μέγεθος της εισόδου του προβλήματος, δηλαδή $R = ()$.

Στη συνέχεια εξετάζεται η κλάση NP (nondeterministic polynomial time).Η κλάση αυτή περιέχει το σύνολο των γλωσσών που μπορούν να αποφασιστούν από μία μη ντετερμινιστική μηχανή TOURING M , με πολυπλοκότητα χρόνου $O(n^k)$ όπου κ θετικός ακέραιος, δηλαδή $NP = \bigcup NTIME\ (n^k)$. Δηλαδή, η κλάση NP εμπεριέχει όλα τα προβλήματα που επαληθεύουν την λύση τους σε πολυωνυμικό χρόνο. Κάθε πρόβλημα στην κλάση P ανήκει επίσης και στη NP, δηλαδή ισχύει $P \subseteq NP$. Εδώ εισέρχεται και το ερώτημα $P = NP$ που είναι ίσως το πιο γνωστό άλυτο πρόβλημα στα μαθηματικά.

Τέλος σε αυτό το κεφάλαιο γίνεται εκτενή αναφορά στην κλάση **NP-πλήρη** (NP- complete).Έχουμε την τάση να ταυτίζουμε τα NP-δύσκολα προβλήματα με τη NP-πλήρη κλάση. Συγκεκριμένα, ένα πρόβλημα χαρακτηρίζεται NP-Hard αν όλα τα προβλήματα της κλάσης NP μπορούν να αναχθούν σε πολυωνυμικό χρόνο σε αυτό. Αν μάλιστα το NP-Hard πρόβλημα είναι και NP τότε το ονομάζουμε NP-ΠΛΗΡΕΣ (NP-complete) . Σε αυτό το σημείο αναφέρονται επίσης πολύ κρίσιμες έννοιες όπως αυτή της αναγωγής και της πληρότητας.

Τέλος υπάρχει μία ανάλυση της λογικής Bool, του θεωρήματος του Cook και Levin και του προβλήματος της ικανοποιησιμότητας.

Στο επόμενο κεφάλαιο 4, μελετάμε κάποια σημαντικά προβλήματα που ανήκουν στην NP-πλήρη κλάση και αποδεικνύεται η NP-πληρότητά τους. Στο κεφάλαιο 5 λύνονται κάποιες ασκήσεις που έχουν να κάνουν με προβλήματα της NP-πλήρους κλάσης.

Όταν αντιμετωπίζουμε ένα NP-πλήρες πρόβλημα βελτιστοποίησης, μπορεί να θέλουμε να εξετάσουμε αλγορίθμους που δεν παράγουν βέλτιστες λύσεις, αλλά λύσεις που είναι εγγυημένα κοντά στο βέλτιστο. Έτσι, στο επόμενο κεφάλαιο 6, θα παραθέσουμε κάποιους προσεγγιστικούς αλγόριθμους για τα προβλήματα αυτής της NP-πλήρους κλάσης. Προσεγγιστικός αλγόριθμος είναι ο αλγόριθμος που επιστρέφει λύσεις κοντά στις βέλτιστες λύσεις, μιας και δεν είναι δυνατή η λήψη μιας βέλτιστης λύσης για αυτά τα προβλήματα.

Στο τελευταίο κεφάλαιο παρατίθενται διάφορα συμπεράσματα που εξήχθησαν από την εργασία, τόσο για τις κλάσεις πολυπλοκότητας και την NP-πληρότητα, όσο και για τους προσεγγιστικούς αλγορίθμους.

Τέλος παρατίθεται παράρτημα, όπου στο πρώτο μέρος υλοποιούμε ένα προσεγγιστικό αλγόριθμο για το πρόβλημα της ακριβής επικάλυψης(Vertex-Cover), και στο δεύτερο μέρος παρατίθεται μία λίστα από πληθώρα προβλημάτων που ανήκουν στην NP-πλήρη κλάση.

Πολλά ευχαριστώ οφείλω στον καθηγητή, κύριο Γρηγόρη Καραγιώργο, που ήταν ο υπεύθυνος αυτής της πτυχιακής εργασίας.

Κων/νος Καραμπούλας
Αύγουστος 2008

Κεφάλαιο 1

ΘΕΩΡΙΑ ΠΟΛΥΠΛΟΚΟΤΗΤΑΣ

Σκοπός αυτού του εισαγωγικού κεφαλαίου είναι να εισάγουμε την έννοια της θεωρίας πολυπλοκότητας και των υπολογιστικών προβλημάτων [**17**], όπως και να κάνουμε μία αναφορά στην πολυπλοκότητα των αλγορίθμων.

1.1 ΘΕΩΡΙΑ ΠΟΛΥΠΛΟΚΟΤΗΤΑΣ, ΥΠΟΛΟΓΙΣΤΙΚΑ ΠΡΟΒΛΗΜΑΤΑ

Αντικείμενο της Θεωρίας Πολυπλοκότητας είναι η μελέτη των υπολογιστικών προβλημάτων, και η κατάταξη τους σε «κλάσεις πολυπλοκότητας» ανάλογα με τους υπολογιστικούς πόρους που αυτά απαιτούν για να επιλυθούν αλγοριθμικά. Κύρια εφαρμογή για την ανάπτυξη της θεωρίας πολυπλοκότητας είναι η ύπαρξη προβλημάτων που ενώ είναι επιλύσιμα δεν είναι γνωστός κάποιος αποδοτικός αλγόριθμος για την επίλυση τους.

Ένα υπολογιστικό πρόβλημα έχει τα παρακάτω χαρακτηριστικά:

• Για κάθε ενδεχόμενη είσοδο υπάρχει ένα μη κενό σύνολο από αποδεκτές εξόδους.

• Οι είσοδοι και οι έξοδοι μπορούν να περιγραφούν ως συμβολοακολουθίες από κάποιο πεπερασμένο αλφάβητο.

Ένα υπολογιστικό πρόβλημα μπορεί να περιγραφεί από διμερή σχέση R επί του Σ^*, όπου Σ είναι ένα πεπερασμένο αλφάβητο έτσι ώστε:

$(x, y) \in R$ αν το y είναι αποδεκτή έξοδος για την είσοδο x.

Οι κύριες κατηγορίες υπολογιστικών προβλημάτων είναι οι εξής:

• Πρόβλημα αναζήτησης: Είναι η γενικότερη μορφή υπολογιστικού προβλήματος. Για κάθε είσοδο ενδέχεται να υπάρχουν περισσότερες από μία αποδεκτές έξοδοι και ζητείται να βρούμε

μία από αυτές.

• Πρόβλημα υπολογισμού συνάρτησης: Για κάθε είσοδο υπάρχει μία ακριβώς αποδεκτή έξοδος.

• Πρόβλημα απόφασης: Για κάθε είσοδο υπάρχει μία ακριβώς αποδεκτή έξοδος που είναι είτε ΝΑΙ είτε ΟΧΙ. Μπορεί να περιγραφεί με ένα υποσύνολο του $Sigma^*$, το οποίο περιέχει τις εισόδους για τις οποίες η έξοδος είναι ΝΑΙ.

• Προβλήματα βελτιστοποίησης: Για κάθε είσοδο προσπαθούμε να βρούμε την καλύτερη λύση.

Στη συνέχεια θα θεωρήσουμε ως αποδοτικό αλγόριθμο έναν αλγόριθμο με πολυωνυμικό χρόνο εκτέλεσης. Αυτό σημαίνει ότι υπάρχει ένα πολυώνυμο p τέτοιο ώστε για κάθε είσοδο x ο αλγόριθμος να τερματίζει μετά από το πολύ $p(x)$ βήματα (ο χρόνος κάθε βήματος είναι ανεξάρτητος από την είσοδο).

Ο διαχωρισμός αυτός:

• είναι ανεξάρτητος από το συγκεκριμένο υπολογιστικό μοντέλο, καθώς όλα τα ρεαλιστικά υπολογιστικά μοντέλα που έχουν προταθεί εξομοιώνουν το ένα το άλλο με πολυωνυμική απώλεια χρόνου.

• είναι ανεξάρτητος από την αναπαράσταση της εισόδου, εφόσον αυτή δεν περιέχει άχρηστη πληροφορία.

• δείχνει ορθός, καθώς οι περισσότεροι πολυωνυμικοί αλγόριθμοι είναι πράγματι αποδοτικοί και το αντίστροφο.

• βοηθάει στην ανάπτυξη μιας θεωρίας, η οποία είναι χρήσιμη για την μελέτη των υπολογιστικών προβλημάτων.

Θα μπορούσαμε να πούμε ότι όλα τα υπολογιστικά προβλήματα μπορούν να ταξινομηθούν σε δύο κατηγορίες: εκείνα που μπορούνε να επιλυθούνε με αλγορίθμους και εκείνα που δεν μπορούνε. Με την μεγάλη πρόοδο της τεχνολογίας των υπολογιστών τις τελευταίες δεκαετίες, θα περίμενε κανείς ότι όλα τα προβλήματα της πρώτης κατηγορίας μπορούν σήμερα να επιλυθούν με ικανοποιητικό τρόπο. Δυστυχώς όμως η πρακτική του υπολογισμού μας αποκαλύπτει ότι πολλά προβλήματα, αν και εκ κατασκευής επιλύσιμα, δεν μπορούν να επιλυθούν υπό καμία πρακτική έννοια από υπολογιστές εξαιτίας των υπερβολικών χρονικών απαιτήσεών τους.

Για την κατανόηση των παραπάνω ας χρησιμοποιήσουμε ένα παράδειγμα όπου είμαστε υπεύθυνοι να προγραμματίσουμε τις επισκέψεις ενός περιοδεύοντος πωλητή σε 10 περιφερειακά γραφεία. Μας δίνεται ένας χάρτης με τις 10 πόλεις και τις αποστάσεις τους σε μίλια και μας ζητείται να σχεδιάσουμε το δρομολόγιο που ελαχιστοποιεί την συνολική απόσταση που θα διανυθεί. Θεωρητικά το πρόβλημα είναι επιλύσιμο. Όμως αν υπάρχουν οι προς επίσκεψη πόλεις, τότε το πλήθος των πιθανών δρομολογίων που είναι πεπερασμένο θα ήταν $(n-1)!$,δηλαδή στο παράδειγμά μας θα είχαμε να εξετάσουμε 9!=362,880 δρομολόγια! Φυσικά μπορεί

εύκολα να σχεδιαστεί ένας αλγόριθμος ο οποίος εξετάζει συστηματικά όλα τα δρομολόγια με σκοπό να βρει το συντομότερο. Παρ'ολα αυτά υπάρχουν πολλές διαδρομές προς εξέταση και έτσι ακόμα και με ένα ρυθμό υπερβολικά γρήγορο,ο απαιτούμενος χρόνος εξέτασης θα ήταν πάρα πολύ μεγάλοσ(ίσως πολλά δισεκατομμύρια φορές ο χρόνος ζωής του σύμπαντος!). Προφανώς, το γεγονός ότι ένα πρόβλημα είναι επιλύσιμο θεωρητικά δεν σημαίνει απευθείας ότι μπορεί ρεαλιστικά να λυθεί στην πράξη. Φαίνεται ότι για να εκφράσουμε την έννοια του «πρακτικά εφικτού αλγορίθμου», θα πρέπει να περιορίσουμε τις υπολογιστικές μηχανές μας να τρέχουν μόνο για έναν αριθμό βημάτων ο οποίος φράσσεται από ένα πολυώνυμο ως προς το μέγεθος της εισόδου. Σε αυτή την περίπτωση λοιπόν θα μπορούσαμε να χρησιμοποιήσουμε την πρότυπη μηχανή Turing, η οποία τερματίζει μετά από πολυωνυμικό αριθμό βημάτων.

1.2 ΠΟΛΥΠΛΟΚΟΤΗΤΑ ΑΛΓΟΡΙΘΜΩΝ

Αυτή η ενότητα έχει να κάνει με την αξιολόγηση των αλγορίθμων ([**14**],[**13**]). Όταν λέμε αξιολόγηση ενός αλγορίθμου εννοούμε την εύρεση ιδιοτήτων του που να μας επιτρέψουν να το χαρακτηρίσουμε σαν καλό ή κακό αλγόριθμο. Στην προηγούμενη μας ενότητα είπαμε ότι θεωρούμε ως αποδοτικό έναν αλγόριθμο με πολυωνυμικό χρόνο εκτέλεσης. Όμως τι είναι αποδοτικός αλγόριθμος; Με βάση ποιο μέτρο μπορούμε να χαρακτηρίσουμε καλό ή κακό έναν αλγόριθμο; Ποιες είναι οι ιδιότητες που μας ενδιαφέρουν σε έναν αλγόριθμο; Σε αυτή την ερώτηση η απάντηση είναι πως η σημαντικότερη παράμετρος που ενδιαφέρει σε έναν υπολογισμό είναι η ταχύτητα. Ακόμη και στην εποχή μας όπου η εξέλιξη των υπολογιστών έχει καταφέρει να κατασκευάσει μηχανές με εξαιρετικές ταχύτητες, τα καινούργια προβλήματα που καλούνται να λύσουν οι υπολογιστές κάνει την ταχύτητα της χρησιμοποιημένης μεθόδου, ίσως το πιο σημαντικό χαρακτηριστικό της. Βέβαια αυτό το χαρακτηριστικό δεν είναι και το μοναδικό. Η μνήμη είναι μία άλλη παράμετρος που ενδιαφέρει έναν υπολογισμό. Όπως και ο χρόνος έτσι και ο χώρος (μνήμη) αποτελεί υπολογιστικό πόρο σε ανεπάρκεια και κατά συνέπεια μας ενδιαφέρουν οι επιδόσεις του αλγορίθμου ως προς αυτόν. Λέμε ότι ο αλγόριθμος μπορεί να έχει λύση *χειρότερης περίπτωσης* (που είναι ο μέγιστος χρόνος ή χώρος που απαιτείται από έναν αλγόριθμο για να επιλύσει πρόβλημα μεγέθους n) , ή *μέσης περίπτωσης* (σε περίπτωση που είναι γνωστό το πιθανοτικό μοντέλο που διέπει τα δεδομένα, τότε είναι δυνατή η μέση ανάλυση του αλγορίθμου), ή *καλύτερης περίπτωσης* (ελάχιστος χρόνος ή χώρος που απαιτείται από έναν αλγόριθμο για να επιλύσει πρόβλημα μεγέθους n) . Λέμε για παράδειγμα ότι η συνάρτηση χρόνου του αλγορίθμου βυββλεσορτ είναι $O(n^2)$ [**18**]. Τι εννοούμε με αυτό θα το δούμε αμέσως τώρα. Παρακάτω δίνουμε κάποιους μαθηματικούς συμβολισμούς οι οποίοι χρησιμοποιούνται για την περιγραφή των επιδόσεων των αλγορίθμων και της πολυπλοκότητας των προβλημάτων.

Έστω,λοιπόν, ότι έχουμε την συνάρτηση g από το σύνολο των θετικών ακεραίων στο σύνολο των θετικών ακεραίων.

• Ορίζουμε ως συμβολισμό $O(g(n))$ (ασυμπτωτικό πάνω όριο) το σύνολο των συναρτήσεων $O(g(n)) = \{f(n) :$ υπάρχουν θετικές σταθερές c και n_0 έτσι ώστε $0 \leq f(n) \leq cg(n)$ για κάθε $n \geq n_0\}$.

• Ορίζουμε ως συμβολισμό $\Theta(g(n))$ (ασυμπτωτικό ακριβές όριο) το σύνολο των συναρτήσεων $\Theta(g(n)) = \{f(n) :$ υπάρχουν θετικές σταθερές c_1, c_2 και n_0 έτσι ώστε $0 \leq c_1 g(n) \leq f(n) \leq c_2 g(n)$ για κάθε $n \geq n_0\}$.

• Ορίζουμε ως συμβολισμό $\Omega(g(n))$ (ασυμπτωτικό κάτω όριο) το σύνολο των συναρτήσεων $\Omega(g(n)) = \{f(n) :$ υπάρχουν θετικές σταθερές c και n_0 έτσι ώστε $0 \leq cg(n) \leq f(n)$ για κάθε $n \geq n_0\}$.

• Ορίζουμε ως συμβολισμό $o(g(n))$ το σύνολο των συναρτήσεων $o(g(n)) = \{f(n) : f(n) \leq g(n)$ για πεπερασμένο αριθμό n και $\lim_{n\to\infty} \frac{f(n)}{g(n)} = 0\}$.

Μια συνάρτηση $f(n)$ ανήκει στο σύνολο των συναρτήσεων $\Theta(g(n))$ εάν υπάρχουν θετικοί ακέραιοι c_1 και c_2 έτσι ώστε η $f(n)$ να φράσσεται κάτω και άνω από τις συναρτήσεις $c_1 g(n)$ και $c_2 g(n)$ αντίστοιχα, για ικανοποιητικά μεγάλες τιμές του n. Αν και το $\Theta(g(n))$ είναι σύνολο, θα γράφουμε $<< f(n) = \Theta(g(n)) >>$ για να δηλώσουμε ότι η συνάρτηση f είναι μέλος του συνόλου $\Theta(g(n))$ και θα λέμε ότι η g είναι ένα *ασυμπτωτικά άνω και κάτω φράγμα της* f. Με άλλα λόγια οι συναρτήσεις f και g έχουν ακριβώς τον ίδιο ρυθμό αύξησης.

Ο συμβολισμός $O(g(n))$ δίνει ένα *ασυμπτωτικά άνω φράγμα της* f. Θα γράφουμε $f(n) = O(g(n))$ και θα λέμε ότι η f είναι τάξης g. Με άλλα λόγια, $f(n) = O(g(n))$ σημαίνει ότι ο ρυθμός αύξησης της f είναι μικρότερος ή ίδιος με τον ρυθμό αύξησης της g. Θα γράφουμε $f(n) = \Omega(g(n))$ όταν συμβαίνει το αντίθετο, δηλαδή όταν $g(n) = O(f(n))$. Τέλος αν $f(n) = o(g(n))$ η $f(n)$ έχει ρυθμό αύξησης μεγαλύτερο της $g(n)$.

ΣΥΜΠΕΡΑΣΜΑΤΑ:

Υπάρχουν υπολογιστικά προβλήματα που παρόλο το ότι είναι επιλύσιμα δεν είναι γνωστός κάποιος αποδοτικός αλγόριθμος για την επίλυση τους. Έτσι κατατάσσουμε αυτά τα προβλήματα

σε κλάσεις πολυπλοκότητας. Αυτή η κατάταξη γίνεται με βάση τους υπολογιστικούς πόρους που αυτά απαιτούν για να επιλυθούν αλγοριθμικά. Είδαμε στην προηγούμενη ενότητα ότι κάθε πρόβλημα και αλγόριθμος σπαταλά υπολογιστικούς πόρους για την επίλυση του, που εστιάζεται στον χρόνο που χρειάζεται για να επιλυθεί αλλά και στον αναγκαίο για τους υπολογισμούς χώρο (μνήμη) που σπαταλά ο αλγόριθμος. Έτσι ανάλογα με τους υπολογιστικούς πόρους που απαιτούν κατατάσσουμε τα προβλήματα αυτά σε κλάσεις πολυπλοκότητας. Στο κεφάλαιο 3 θα δούμε τις τρεις σημαντικότερες κλάσεις P, NP και NP-πλήρη, δίνοντας μεγαλύτερη έμφαση στην τελευταία.

Κεφάλαιο 2

ΥΠΟΛΟΓΙΣΤΙΚΑ ΜΟΝΤΕΛΑ ΚΑΙ ΚΛΑΣΕΙΣ ΠΟΛΥΠΛΟΚΟΤΗΤΑΣ

Στη θεωρία πολυπλοκότητας το μαθηματικό μοντέλο που έχει υιοθετηθεί για την μελέτη της υπολογιστικής πολυπλοκότητας των προβλημάτων είναι η μηχανή Turing **[19]**, μία μηχανή που προτάθηκε από τον Alan Turing σαν ένα μοντέλο υπολογισμού, ιδιαίτερα απλό και κατάλληλο για την ανάπτυξη της θεωρίας μας. Σε αυτό το κεφάλαιο λοιπόν θα δούμε την λειτουργία της μηχανής Turing και των παραλλαγών της.

2.1 ΕΙΣΑΓΩΓΙΚΑ

Στη θεωρία πολυπλοκότητας μας ενδιαφέρει η κατάταξη των προβλημάτων σε κλάσεις ανάλογα με τους υπολογιστικούς πόρους που αυτά απαιτούν για να επιλυθούν. Ειδικότερα:

- Ορίζουμε κλάσεις πολυπλοκότητας
- Βρίσκουμε σχέσεις ανάμεσα στις κλάσεις
- Βρίσκουμε αντιπροσωπευτικά προβλήματα κάθε κλάσης

Μία κλάση πολυπλοκότητας καθορίζεται συνήθως από τα παρακάτω πέντε στοιχεία:

- Κατηγορία υπολογιστικών προβλημάτων τα οποία περιέχει (απόφασης, αναζήτησης, βελτιστοποίησης....)
- Υπολογιστικό μοντέλο (Μηχανή Turing , RAM ....)
- Τύπο υπολογισμού (ντετερμινιστικός, μη ντετερμινιστικός, πιθανοτικός....)

• Υπολογιστικό αγαθό που αποτελεί το μέτρο πολυπλοκότητας (χρόνο, χώρο, τυχαία bit , επεξεργαστές, μηνύματα....)

• Όριο για το υπολογιστικό αγαθό (συνάρτηση του μεγέθους της εισόδου)

2.2 ΜΗΧΑΝΕΣ TURING

Για την μελέτη της υπολογιστικής πολυπλοκότητας των υπολογιστικών προβλημάτων χρησιμοποιείται η μηχανή Turing , μία μηχανή που χρησιμοποιείται ως ένα μοντέλο υπολογισμού που είναι ισοδύναμο με κάθε σύγχρονο υπολογιστικό μέσο.

Για την μελέτη και κατανόηση, λοιπόν, των κλάσεων πολυπλοκότητας θα πρέπει να ορίσουμε και να πούμε κάποια λόγια για την μηχανή Turing **[1]**, **[2]**,**[4]**,**[5]**,**[11]**.

Μία μηχανή Turing αποτελείται από μία μονάδα ελέγχου πεπερασμένων καταστάσεων και μία ταινία. Η επικοινωνία μεταξύ τους επιτυγχάνεται με μία κεφαλή που μπορεί να χρησιμοποιηθεί για ανάγνωση ή εγγραφή στην ταινία, δηλαδή η κεφαλή διαβάζει σύμβολα από την ταινία και χρησιμοποιείται επίσης για να μεταβάλει τα σύμβολα στην ταινία. Η μονάδα ελέγχου λειτουργεί σε διακριτά βήματα και σε κάθε βήμα εκτελεί δύο λειτουργίες με τρόπο που εξαρτάται από την τρέχουσα κατάστασή της και το σύμβολο της ταινίας που μόλις διαβάστηκε από την κεφαλή ανάγνωσης/εγγραφής:

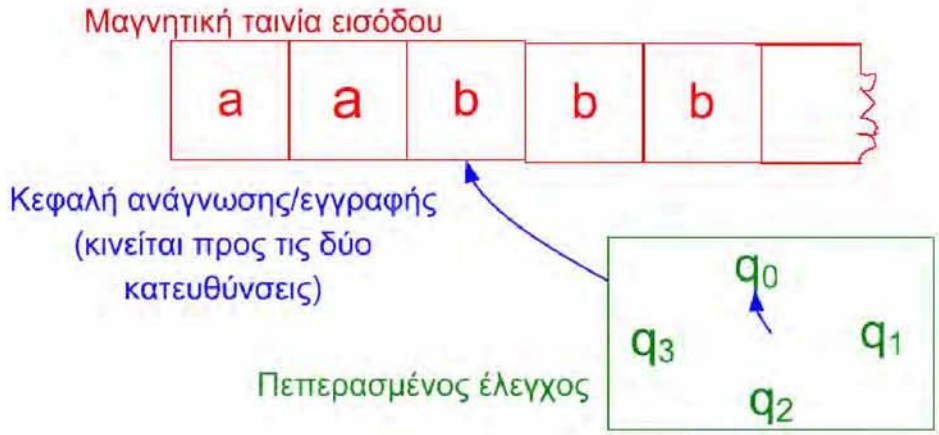
Τοποθετεί τη μονάδα ελέγχου σε μία νέα κατάσταση.

Είτε:

(α) Γράφει ένα σύμβολο στο τετράγωνο της ταινίας που μόλις σαρώθηκε, αντικαθιστώντας το ήδη υπάρχον εκεί,

είτε

(β) Μετακινεί την κεφαλή ανάγνωσης/εγγραφής κατά ένα τετράγωνο αριστερά ή δεξιά.



Σχήμα 2.2

Η ταινία αρχίζει από το αριστερό της άκρο, αλλά εκτείνεται απεριόριστα προς τα δεξιά. Για να εμποδίσουμε την μηχανή να μετακινήσει την κεφαλή της πέρα από το αριστερό άκρο της ταινίας, υποθέτουμε ότι το αριστερότερο τετράγωνο της ταινίας σημειώνεται πάντα με ένα ειδικό σύμβολο το οποίο δηλώνεται ως $\triangleright$.Υποθέτουμε ακόμα ότι όλες οι μηχανές Turing είναι σχεδιασμένες έτσι ώστε, όποτε η κεφαλή διαβάζει ένα $\triangleright$, να μετακινείται αμέσως προς τα δεξιά. Επίσης στην μηχανή Turing χρησιμοποιούνται τα διακεκριμένα σύμβολα $\leftarrow$ και $\rightarrow$ για τον συμβολισμό της κίνησης της κεφαλής προς τα αριστερά ή προς τα δεξιά αντίστοιχα.

Η τροφοδότηση μιας μηχανής Turing με είσοδο γίνεται με την τοποθέτηση της συμβολοσειράς εισόδου στα τετράγωνα στο αριστερό άκρο της ταινίας, αμέσως δεξιά από το σύμβολο $\triangleright$. Το υπόλοιπο της ταινίας περιέχει αρχικά κενά σύμβολα τα οποία δηλώνονται ως $\sqcup$. Η μηχανή επιτρέπεται να μετατρέπει την είσοδό της με οποιοδήποτε τρόπο θεωρεί κατάλληλο, καθώς επίσης και να γράφει στο απεριόριστο κενό τμήμα της ταινίας στα δεξιά. Εφόσον η μηχανή μπορεί να μετακινήσει την κεφαλή της μόνο κατά ένα τετράγωνο την φορά, μετά από κάθε πεπερασμένο υπολογισμό θα έχει επισκεφθεί πεπερασμένο πλήθος τετραγώνων.

Υπάρχουν οι ντετερμινιστικές μηχανές Turing και οι μη ντετερμινιστικές μηχανές Turing αλλά και άλλες μορφές τέτοιων υπολογιστικών μηχανών.

2.2.1 ΝΤΕΤΕΡΜΙΝΙΣΤΙΚΗ ΜΗΧΑΝΗ TURING

Μετά από αυτή την σύντομη εισαγωγή μας για το πώς λειτουργεί μία μηχανή Turing μπορούμε να παρουσιάσουμε τον αυστηρό ορισμό μιας ντετερμινιστικής μηχανής Turing [**1**]:

Παρακάτω ορίζουμε μία ντετερμινιστική μηχανή Turing :

Ορισμός 2.2.1(a)

Μία ντετερμινιστική μηχανή Turing είναι μία πεντάδα $(K, \Sigma, \delta, s, H)$, όπου

• Κ, είναι ένα πεπερασμένο σύνολο καταστάσεων.

• Σ, είναι ένα αλφάβητο, το οποίο περιέχει το κενό σύμβολο $\sqcup$ και το σύμβολο αριστερού άκρου $\triangleright$,αλλά δεν περιέχει τα σύμβολα $\leftarrow$και $\rightarrow$.

• $s \in K$ είναι η **αρχική κατάσταση**.

• $H \subseteq K$ είναι το σύνολο των **καταστάσεων τερματισμού**.

• δ, η συνάρτηση μετάβασης ,είναι μία συνάρτηση από το $(K-H)\times\Sigma$ στο $K\times(\Sigma\cup\{\leftarrow,\rightarrow\})$ τέτοια ώστε ,

(α) για κάθε $q \in K - H$, αν $\delta(q,\triangleright) = (p,b)$, τότε $b = \rightarrow$.

(β) για κάθε $q \in K - H$ και $a \in \Sigma$, αν $\delta(q,a) = (p,b)$,τότε $b \neq \triangleright$.

Αν $q \in K$-H, $a \in \Sigma$ και δ $(q,a)=(p,b)$ τότε η μηχανή M ,όταν βρίσκεται στην κατάσταση q και διαβάζει το σύμβολο a,θα εισέλθει στην κατάσταση p, και (1) αν b είναι σύμβολο του Σ, γράφει b στη θέση του a που μόλις διαβάστηκε ή (2) αν b είναι $\leftarrow$ ή $\rightarrow$,μετακινεί την κεφαλή στην κατεύθυνση b. Εφόσον η δ είναι συνάρτηση, η λειτουργία της μηχανής είναι ντετερμινιστική και θα σταματήσει μόνο όταν η μηχανή εισέλθει σε κατάσταση τερματισμού. Σύμφωνα με την απαίτηση (α) που είδαμε παραπάνω η μηχανή όταν συναντήσει το σύμβολο του αριστερού άκρου της ταινίας $\triangleright$,πρέπει να μετακινηθεί προς τα δεξιά. Έτσι το αριστερότερο σύμβολο $\triangleright$ δεν διαγράφεται ποτέ, και η μηχανή δεν ξεπερνάει πότε το αριστερό άκρο της ταινίας της. Σύμφωνα με το (β), η μηχανή δεν γράφει ποτέ ένα $\triangleright$,και συνεπώς το $\triangleright$είναι αλάνθαστη ένδειξη του αριστερού άκρου της ταινίας. Δηλαδή, μπορούμε να φανταστούμε το $\triangleright$ως ένα «προστατευτικό τείχος» που εμποδίζει την κεφαλή της μηχανής να ξεπεράσει άθελά της το αριστερό άκρο, το οποίο δεν παρεμβαίνει στον υπολογισμό της μηχανής με κανένα άλλο τρόπο. Τέλος η δ δεν ορίζεται για καταστάσεις του H και όταν η μηχανή οδηγηθεί σε κατάσταση τερματισμού ,η λειτουργία του σταματάει. Παρακάτω θα εκθέσουμε ένα παράδειγμα για την κατανόηση της λειτουργίας μίας ντετερμινιστικής μηχανήςTuring :

Παράδειγμα 2.2.1 (a) : Θεωρήστε τη μηχανή Turing M = $(K,\Sigma,\delta,s,\{h\})$, όπου

$$K = \{q_o, q_1, h\},$$

$$\Sigma = \{a, \sqcup, \triangleright\},$$

$$\text{s}= q_o,$$

και η δ δίνεται από τον παρακάτω πίνακα:

	q	σ	δ(q,σ)
1	q_0	a	(q_1, ⊔)
2	q_0	⊔	(h, ⊔)
3	q_0	▷	(q_0, →)
4	q_1	a	(q_0, a)
5	q_1	⊔	(q_0, →)
6	q_1	▷	(q_1, →)

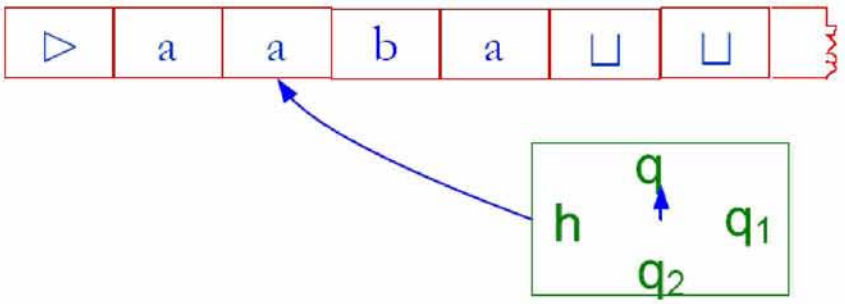
Πίνακας 2.2.1

Όταν η M ξεκινάει στην αρχική της κατάσταση q_o, σαρώνει με την κεφαλή της προς τα δεξιά, μεταβάλλοντας όλα τα a σε $\sqcup$ καθώς προχωράει, ώσπου να βρει ένα τετράγωνο της ταινίας που είδη περιέχει $\sqcup$ και τότε σταματάει. Πιο συγκεκριμένα, έστω ότι η M ξεκινάει με την κεφαλή της να διαβάζει το πρώτο από τα τέσσερα a, το τελευταίο από τα οποία ακολουθείται από $\sqcup$. Τότε η M θα μετακινηθεί εμπρός και πίσω ανάμεσα στις καταστάσεις q_o και q_1 τέσσερις φορές, μετατρέποντας διαδοχικά τα a σε $\sqcup$ και μετακινώντας την κεφαλή προς τα δεξιά. Έτσι σε αυτό το σημείο η M θα βρεθεί στην κατάσταση q_o με επόμενο σύμβολο το $\sqcup$ και, σύμφωνα με την δεύτερη γραμμή του πίνακα, θα τερματίσει.
Παρατηρείται ότι η τέταρτη γραμμή του πίνακα, δηλαδή η τιμή της (q_1,a), είναι αδιάφορη καθώς η M δεν πρόκειται ποτέ να βρεθεί στην κατάσταση q_1 με επόμενο σύμβολο ένα a αν έχει ξεκινήσει από την κατάσταση q_o. Παρ΄ όλα αυτά, πρέπει να υπάρχει κάποια τιμή που αντιστοιχεί στην δ (q_1,a), αφού η δ απαιτείται να είναι συνάρτηση με πεδίο ορισμού $K - H \times \Sigma$.
Περιληπτικά:
q_0: από την θέση $\triangleright$ πάει δεξιά και μένει στη q_0
q_0: διαβάζει το πρώτο a
-το αντικαθιστά με κενό $\sqcup$ και πάει στην κατάσταση q_1 **(1)**
-διαβάζει το $\sqcup$ και πάει δεξιά και πάει στην κατάσταση q_0 **(5)**
-επαναλαμβάνει το ίδιο ώσπου να αντικαταστήσει όλα τα a σε $\sqcup$
-είναι στην κατάσταση q_0 και διαβάζει $\sqcup$. Τότε πάει στην τελική κατάσταση h και τερματίζει. **(2)**
σημείωση: Η μεταβίβαση **(4)** δεν χρησιμοποιείται. Πρέπει να υπάρχει επειδή η δ είναι συνάρτηση.

Ορισμός 2.2.1 (b)
Μία συνολική κατάσταση μιας μηχανής Turing M= $(K,\Sigma,\delta,s,\{h\})$ είναι μέλος του $K \times \Sigma^* \times (\Sigma^*(\Sigma - \{\sqcup\}) \cup \{\varepsilon\})$ **[1]**.

Δηλαδή, υποθέτουμε ότι όλες οι συνολικές καταστάσεις αρχίζουν με το σύμβολο αριστερού άκρου και δεν τελειώνουν ποτέ με κενό, εκτός αν το κενό είναι το σύμβολο που μόλις διαβάστηκε.

Παράδειγμα 2.2.1 (b)
Θα δούμε αν η $((q, \triangleright aa, ba)$ είναι συνολική κατάσταση ή όχι.
- το αυτόματο είναι στην κατάσταση q.
-από αριστερά ως την κεφαλή είναι η λέξη $\triangleright aa$ και η κεφαλή διαβάζει το δεύτερο a.
-δεξιά από την κεφαλή είναι η λέξη ba.
-η τρίτη παράμετρος δεν τελειώνει με $\sqcup$.
Άρα η $((q, \triangleright aa, ba)$ είναι συνολική κατάσταση.



Σχήμα 2.2

Παράδειγμα 2.2.1 (c)

Σε αυτό το παράδειγμα δείχνουμε τέσσερις συναρτήσεις μετάβασης από τις οποίες οι δύο είναι συνολικές καταστάσεις (ΟΚ) , ενώ οι άλλες δύο όχι (ΟΧΙ ΟΚ).

$(q, \triangleright, aa\sqcup, \sqcup ba) : OK$

$(q, \triangleright aa\sqcup, ba\sqcup)$: ΟΧΙ OK!

$(q, \triangleright aa\sqcup, e) : OK$

(q, aa, ba) : ΟΧΙ OK!

Επίσης πρέπει να πούμε ότι η $\vdash^*$ είναι η ανακλαστική μεταβατική κλειστότητα της $\vdash$. Λέμε ότι η συνολική κατάσταση C_1 **παράγει** την συνολική κατάσταση C_2 αν $C_1 \vdash_M^* C_2$. Ένας **υπολογισμός** είναι μια ακολουθία από συνολικές καταστάσεις $C_0, C_1, ..., C_n$, για κάθε $n \geq 0$ τέτοιο ώστε

$$C_0 \vdash_M C_1 \vdash_M C_2 \vdash_M .... \vdash_M C_n.$$

Λέμε ότι ο υπολογισμός είναι **μήκους** n ή ότι έχει n **βήματα**, και γράφουμε $C_0 \vdash_M^n C_n$. [1]

Ορισμός 2.2.1 (c)

Για μια $M = (K, \Sigma, \delta, s, H)$ τέτοια ώστε $H = \{y, n\}$(αποτελείται από δύο διακεκριμένες καταστάσεις τερματισμού ''yes'' και ''no'')

• κάθε συνολική κατάσταση με κατάσταση τερματισμού y ονομάζεται **συνολική κατάσταση αποδοχής**

• η M **αποδέχεται** την w αν παράγει μια συνολική κατάσταση αποδοχής.

• κάθε συνολική κατάσταση με κατάσταση τερματισμού n ονομάζεται **συνολική κατάσταση απόρριψης**

• η M **απορρίπτει** την w αν παράγει μία συνολική κατάσταση τερματισμού

Αν $\Sigma_0 \subseteq \Sigma - \{\sqcup, \triangleright\}$ το αλφάβητο εισόδου της Mλέμε ότι:

Η M **αποφασίζει** μια γλώσσα $L \subseteq \Sigma_0^*$ αν για κάθε λέξη $w \in \Sigma_0^*$ είναι αληθές ότι:

• αν $w \in L$ η M δέχεται την w και

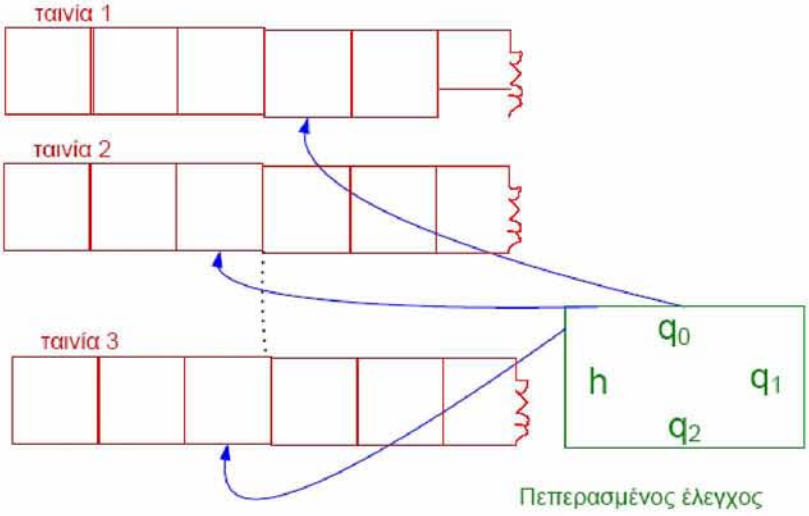
• αν $w \notin L$ η M απορρίπτει την w.

Μια γλώσσα λέγεται **αναδρομική** αν υπάρχει μία μηχανή που την αποφασίζει. [1]

2.2.2 ΜΗΧΑΝΗ k-ΤΑΙΝΙΩΝ ΚΑΙ ΜΗΧΑΝΗ ΤΥΧΑΙΑΣ ΠΡΟΣΠΕΛΑΣΗΣ

Στη συνέχεια θα κάνουμε μία μικρή αναφορά για τη μηχανή k ταινιών αλλά και τη μηχανή τυχαίας προσπέλασης.

Η μηχανή k ταινιών είναι μία μηχανή η οποία σε κάθε βήμα, κάθε κεφαλή διαβάζει και γράφει ή μετακινείται, όπως φαίνεται και στο παρακάτω σχήμα:



Σχήμα 2.2.2

Αυτό που θέλουμε να αναφέρουμε για τη μηχανή αυτή είναι η εξής πρόταση:

ΠΡΟΤΑΣΗ 2.2.2 (a):Οποιαδήποτε συνάρτηση υπολογίζεται ή οποιαδήποτε γλώσσα αποφασίζεται ή ημιαποφασίζεται από μια μηχανή k ταινιών, υπολογίζεται, αποφασίζεται ή ημιαποφασίζεται, αντίστοιχα από μια μηχανή Turing . Η κεφαλή στις μηχανές που είδαμε μέχρι τώρα μπορεί σε ένα βήμα να διαβάσει ένα τετράγωνο αριστερά ή δεξιά της τρέχουσας θέσης.

Στις μηχανές τυχαίας προσπέλασης η κεφαλή μπορεί να διαβάσει οποιοδήποτε τετράγωνο

της ταινίας σε ένα βήμα με την χρήση καταχωρητών οι οποίοι διαχειρίζονται τις διευθύνσεις των τετράγωνων της ταινίας. Αυτό που θέλουμε να αναφέρουμε για τη μηχανή αυτή είναι η εξής πρόταση:

ΠΡΟΤΑΣΗ 2.2.2 (b).Οποιαδήποτε συνάρτηση υπολογίζεται ή οποιαδήποτε γλώσσα αποφασίζεται ή ημιαποφασίζεται από μια μηχανή Turing τυχαίας προσπέλασης , υπολογίζεται, αποφασίζεται ή ημιαποφασίζεται, αντίστοιχα από μια μηχανή Turing .

2.2.3 ΜΗ ΝΤΕΤΕΡΜΙΝΙΣΤΙΚΗ ΜΗΧΑΝΗ TURING

Παρακάτω ορίζουμε μία μη ντετερμινιστική μηχανή Turing :

Ορισμός 2.2.3(a)
Μία μη ντετερμινιστική μηχανή Turing είναι μία πεντάδα (K,Σ,Δ,s,H), όπου K,Σ,s και H είναι όπως και στις πρότυπες μηχανές Turing και Δ είναι ένα υποσύνολο του $(K-H)\times\Sigma\times(K\times(\Sigma\cup\{\leftarrow,\rightarrow\}))$, αντί για μία συνάρτηση $(K-H)\times\Sigma$ στο $(K\times(\Sigma\cup\{\leftarrow,\rightarrow\}))$.
Πρέπει επίσης να ορίσουμε τι σημαίνει για μία μη ντετερμινιστική μηχανή Turing να υπολογίζει κάτι. Εφόσον μία μη ντετερμινιστική μηχανή θα επέστρεφε δύο διαφορετικές εξόδους ή τελικές καταστάσεις με την ίδια είσοδο, πρέπει να είμαστε προσεκτικοί ως προς το ποιο θεωρούμε ότι είναι το τελικό αποτέλεσμα του υπολογισμού μιας τέτοιας μηχανής. Για το λόγο αυτό εξετάζουμε μη ντετερμινιστικές μηχανές Turing που ημιαποφασίζουν γλώσσες. **[1]**

Ποτέ όμως λέμε ότι μία ντετερμινιστική μηχανή Turing M **δέχεται** μία είσοδο ω και πότε λέμε ότι η M **ημιαποφασίζει μία γλώσσα** L· Αυτό απαντάται με τον παρακάτω ορισμό.

Ορισμός 2.2.3(b) :
Έστω $M=(K,\Sigma,\Delta,s,H)$ μία μη ντετερμινιστική μηχανή Turing . Λέμε ότι η M **δέχεται** μία είσοδο $\omega\in(\Sigma-\{\triangleright,\sqcup\})^*$ αν $(s,\triangleright\underline{\sqcup}\omega)\vdash_M^*(h,u\underline{a}v)$ για κάποια $h\in H$ και $\alpha\in\Sigma, u,v\in\Sigma^*$.

Παρατηρούμε ότι μία μη ντετερμινιστική μηχανή δέχεται μία είσοδο ακόμη και αν έχει πολλούς ατέρμονους υπολογισμούς με την είσοδο αυτή-αρκεί όμως να υπάρχει τουλάχιστον ένας υπολογισμός που τερματίζει. Λέμε ότι η M **ημιαποφασίζει μία γλώσσα** $L\subseteq(\Sigma-\{\triangleright,\sqcup\})$ αν ισχύει για κάθε
$\omega\in(\Sigma-\{\triangleright,\sqcup\})$ το εξής: $\omega\in L$ αν και μόνο αν η M δέχεται την ω. **[1]**

Τέλος παρακάτω ορίζουμε τι σημαίνει για μία μη ντετερμινιστική μηχανή Turing να **αποφασίζει** μία γλώσσα ή να **υπολογίζει** μία συνάρτηση.

Ορισμός 2.2.3(c) :
Έστω M $= (K, \Sigma, \Delta, s, \{y, n\})$ μία μη ντετερμινιστική μηχανή Turing . Λέμε ότι η M **αποφασίζει** μία γλώσσα $L \subseteq (\Sigma - \{\triangleright, \sqcup\})$ αν οι ακόλουθες δύο συνθήκες ισχύουν για κάθε $\omega \in (\Sigma - \{\triangleright, \sqcup\})$:
(α) Υπάρχει φυσικός αριθμός N, ο οποίος εξαρτάται από την M και την ω, τέτοιος ώστε να μην υπάρχει συνολική κατάσταση C η οποία να ικανοποιεί την σχέση $(s, \triangleright\underline{\sqcup}\omega)\vdash_M^N C$.
(β) $\omega \in L$ αν και μόνο αν $(s, \triangleright\underline{\sqcup}\omega)\vdash_M^* (y, u\underline{a}v)$ για κάποια $u, v \in \Sigma^*, a \in \Sigma$.

Τέλος λέμε ότι η M **υπολογίζει** μία συνάρτηση $f : (\Sigma - \{\triangleright, \sqcup\})^* \mapsto (\Sigma - \{\triangleright, \sqcup\})^*$
αν οι ακόλουθες δύο συνθήκες ισχύουν για κάθε $\omega \in (\Sigma - \{\triangleright, \sqcup\})$:

(α) Υπάρχει N, ο οποίος εξαρτάται από την M και την ω ,τέτοιος ώστε να μην υπάρχει συνολική κατάσταση C η οποία να ικανοποιεί τη σχέση $(s, \triangleright\underline{\sqcup}\omega)\vdash_M^N C$.
(β) $(s, \triangleright\underline{\sqcup}\omega)\vdash_M^* (h, u\underline{a}v)$ αν και μόνο αν $ua = \triangleright\sqcup$ και $v = f(\omega)$.

ΠΡΟΤΑΣΗ 2.2.3: Οποιαδήποτε συνάρτηση υπολογίζεται ή οποιαδήποτε γλώσσα αποφασίζεται ή ημιαποφασίζεται από μια μη ντετερμινιστική μηχανή Turing,υπολογίζεται, αποφασίζεται ή ημιαποφασίζεται, αντίστοιχα από μια μηχανή Turing.

Διαβάζοντας τους παραπάνω ορισμούς παρατηρούμε αρχικά ότι, για να μπορεί μία μη ντετερμινιστική μηχανή να αποφασίζει μία γλώσσα ή να υπολογίζει μία συνάρτηση, απαιτούμε όλοι οι υπολογισμοί της να τερματίζουν και αυτό το επιτυγχάνουμε αν υποθέσουμε ότι δεν υπάρχει υπολογισμός ο οποίος να συνεχίζει μετά από N βήματα, όπου N είναι ένα «άνω φράγμα» το οποίο εξαρτάται από την μηχανή και την είσοδο. Επιπλέον, για να αποφασίζει η M μία γλώσσα, απαιτούμε μόνο τουλάχιστον ένας από τους δυνατούς υπολογισμούς της να καταλήγει σε αποδοχή της εισόδου.Κάποιοι, οι περισσότεροι ή και όλοι οι υπόλοιποι υπολογισμοί θα μπορούσαν να καταλήγουν σε απόρριψη της εισόδου(η μηχανή δέχεται την είσοδο βασιζόμενη στην ισχύ του μοναδικού αυτού υπολογισμού που οδηγεί σε αποδοχή).Τέλος για να μπορεί μία μη ντετερμινιστική μηχανή Turing να υπολογίζει μία συνάρτηση απαιτούμε όλοι οι δυνατοί υπολογισμοί να συμφωνούν ως προς το αποτέλεσμα. Αν όχι, δεν θα μπορούσαμε να αποφασίσουμε ποια είναι η σωστή τιμή της συνάρτησης.

2.2.4 ΥΠΟΛΟΓΙΣΜΟΙ ΧΩΡΟΥ ΚΑΙ ΧΡΟΝΟΥ

Για να μελετήσουμε τα υπολογιστικά προβλήματα και να τα κατηγοριοποιήσουμε ανάλογα με την δυσκολία επίλυσής τους θα ορίσουμε την έννοια του χρόνου (time) που διαρκεί ένας υπολογισμός αλλά και του χώρου (space) που απαιτεί μία μηχανή Turing **[4],[21]**.

ΟΡΙΣΜΟΣ 2.2.4(a) : Ο χρόνος που απαιτεί μία ντετερμινιστική μηχανή Turing M με είσοδο την συμβολοσειρά x είναι το πλήθος των βημάτων (κινήσεων) που κάνει η M ξεκινώντας από την αρχική διαμόρφωση για τον υπολογισμό του $M(\xi)$.Αν η M αποκλίνει τότε ο χρόνος είναι ίσος με άπειρο.

Στην περίπτωση μίας μη ντετερμινιστικής μηχανής Turing ο ορισμός του χρόνου είναι διαφορετικός.

ΟΡΙΣΜΟΣ 2.2.4(b): Ο χρόνος που απαιτεί μία μη ντετερμινιστική μηχανή Turing M με είσοδο την συμβολοσειρά x είναι το μικρότερο πλήθος βημάτων που μπορεί να κάνει η M ξεκινώντας από την αρχική διαμόρφωση για να φτάσει σε κατάσταση αποδοχής. Σε κάθε άλλη περίπτωση, ο χρόνος της M είναι, συμβατικά, ίσος με 1.

Ορίζουμε τώρα την πολυπλοκότητα χρόνου μιας μηχανής Turing:

ΟΡΙΣΜΟΣ 2.2.4(c) : Η πολυπλοκότητα χρόνου μιας μηχανής Turing είναι μία συνάρτηση $T_M : \mathrm{Z}_+ \rightarrow \mathrm{Z}_+$ που ορίζεται ως εξής:
$T_M(n) = \max\{m :\in (\Sigma\backslash\{\sqcup\})^*, |x| = n$ έτσι ώστε ο χρόνος που απαιτεί η M με είσοδο x είναι m.
Ο χρόνος που απαιτεί μία μηχανή Turing με είσοδο μήκους n είναι ο μέγιστος χρόνος πάνω σε όλες τις εισόδους μήκους n. Συνήθως υποθέτουμε ότι $T_M(n) \geq n$, αφού n βήματα πρέπει να κάνει η μηχανή μόνο για να διαβάσει όλα τα σύμβολα της εισόδου μήκους n.

Όσον αφορά την πολυπλοκότητα της ντετερμινιστική μηχανής k-ταινιών έχουμε:

ΠΡΟΤΑΣΗ 2.2.4: Για κάθε ντετερμινιστική μηχανή Turing k-ταινιών πολυπλοκότητας χρόνου $f(n)$ υπάρχει ισοδύναμη ντετερμινιστική μηχανή Turing k+2 ταινιών με είσοδο και έξοδο πολυπλοκότητας χρόνου $Of(n)$.

ΘΕΩΡΗΜΑ 2.2.4 (a): Για κάθε ντετερμινιστική μηχανή Turing k-ταινιών πολυπλοκότητας χρόνου $f(n)$ υπάρχει ισοδύναμη ντετερμινιστική μηχανή Turing μιας ταινίας πολυπλοκότητας χρόνου $O(f(n)^2)$.

ΘΕΩΡΗΜΑ 2.2.4(b): Για κάθε μη ντετερμινιστική μηχανή Turing πολυπλοκότητας χρόνου $f(n)$ υπάρχει ισοδύναμη ντετερμινιστική μηχανή Turing πολυπλοκότητας χρόνου $O(2^{f(n)})$.

2.3 ΚΛΑΣΕΙΣ ΠΟΛΥΠΛΟΚΟΤΗΤΑΣ

Ας περάσουμε τώρα στην έννοια της κλάσης πολυπλοκότητας[**4**],[**21**]. Μια κλάση πολυπλοκότητας δεν είναι τίποτα άλλο παρά ένα σύνολο γλωσσών με κοινές ιδιότητες. Υπάρχουν διάφοροι παράμετροι που προσδιορίζουν μία κλάση πολυπλοκότητας. Καταρχήν το μοντέλο του υπολογισμού. Συγκεκριμένα, εμείς θα χρησιμοποιήσουμε τη μηχανή Turing k-ταινιών .Μία κλάση πολυπλοκότητας χαρακτηρίζεται επίσης από τον τρόπο του υπολογισμού δηλαδή αν η μηχανή μας είναι ντετερμινιστική ή μη. Μία ακόμα παράμετρος είναι ο υπολογιστικός πόρος που θέλουμε να μετρήσουμε και τέλος να καθορίσουμε ένα φράγμα για τον υπολογισμό, δηλαδή μια συνάρτηση $f : \mathrm{Z}_+ \to \mathrm{Z}_+$.

Με βάση τα παραπάνω, μία κλάση πολυπλοκότητας ορίζεται ως εξής[**4**]:

ΟΡΙΣΜΟΣ 2.3 : Μία κλάση πολυπλοκότητας είναι το σύνολο των γλωσσών που αποφασίζονται από μία μηχανή Turing k-ταινιών που λειτουργεί με τον κατάλληλο τρόπο (ντετερμινιστικά ή μη) έτσι ώστε ,για οποιαδήποτε είσοδο x, η μηχανή να χρειάζεται το πολύ $f(x)$ μονάδες υπολογιστικού πόρου(χρόνου ή χώρου), όπου $f : \mathrm{Z}_+ \to \mathrm{Z}_+$ είναι η συνάρτηση πολυπλοκότητας της μηχανής.

Ορίζουμε στην συνέχεια τέσσερις γενικές κλάσεις πολυπλοκότητας, λαμβάνοντας υπόψιν κάθε φορά τον τρόπο του υπολογισμού και τον υπολογιστικό πόρο που θέλουμε να μετρήσουμε [**10**]:

Ορίζουμε σαν TIME$f(n)$ το σύνολο των γλωσσών που αποφασίζονται από κάποια ντετερμινιστική μηχανή Turing με πολυπλοκότητα χρόνου $f(n)$.
Ορίζουμε σαν NTIME$f(n)$ το σύνολο των γλωσσών που αποφασίζονται από κάποια μη ντετερμινιστική μηχανή Turing με πολυπλοκότητα χρόνου$f(n)$.
Ορίζουμε σαν SPACE$f(n)$ το σύνολο των γλωσσών που αποφασίζονται από κάποια ντετερμινιστική μηχανή Turing με πολυπλοκότητα χώρου $f(n)$.
Ορίζουμε σαν NSPACE$f(n)$ το σύνολο των γλωσσών L που αποφασίζονται από κάποια μη ντετερμινιστική μηχανή Turing με πολυπλοκότητα χώρου $f(n)$.

Εμείς στην εργασία μας θα ασχοληθούμε με τις κλάσεις πολυπλοκότητας χρόνου. Ας δούμε κάποιες απλές σχέσεις που συνδέουν τις κλάσεις πολυπλοκότητας που μόλις ορίσαμε. Ισχύει:
TIME$f(n) \subseteq NTIMEf(n)$
SPACE$f(n) \subseteq NSPACEf(n)$
TIME$f(n) \subseteq TIME2^{f(n)}$
TIME$(f(n) \subseteq SPACEf(n)$

Τέλος, κλείνοντας αυτό το κεφάλαιο θα αναφέρουμε ένα πολύ σημαντικό θεώρημα, το θεώρημα του Savitch **[4],[21]**:Έστω $f(n) \geq \log n$ μία κατασκευάσιμη συνάρτηση. Τότε ισχύει: $NSPACE(f(n)) \subseteq SPACE(f(n)^2)$.

ΣΥΜΠΕΡΑΣΜΑΤΑ:
Το μαθηματικό μοντέλο που μελετήσαμε σε αυτό το κεφάλαιο είναι η μηχανή Turing, το μοντέλο που χρησιμοποιείται από την Θεωρία Πολυπλοκότητας για την μελέτη της υπολογιστικής πολυπλοκότητας των προβλημάτων. Η μηχανή Turing είναι ένα εντυπωσιακά απλό μοντέλο, ιδιαίτερα κατάλληλο για την ανάπτυξη της θεωρίας μας που αποδεικνύεται όμως ότι παρέχει όλη την δυνατή υπολογιστική ισχύ που μπορεί να έχει ο μηχανιστικός υπολογισμός **[20]**, και μάλιστα με επιδόσεις αρκετά κοντά σε αυτές των ρεαλιστικών υπολογιστών και αυτά τα δύο χαρακτηριστικά (δηλαδή το ότι είναι αρκετά κοντά στους πραγματικούς (ρεαλιστικούς) υπολογιστές και ότι το μοντέλο αυτό είναι απλό διευκολύνοντας την ανάπτυξη της θεωρίας και την κατανόηση της) κάνουν την μηχανή Turing ως το ευρέως χρησιμοποιημένο υπολογιστικό μοντέλο.

Στο κεφάλαιο αυτό, είδαμε επίσης, πολλές παραλλαγές της μηχανής Turing .Η προσπάθεια για την εύρεση ενός όλο και καλύτερου υπολογιστικού μοντέλου οδήγησε τους ερευνητές σε διάφορες επεκτάσεις της μηχανής Turing. Βέβαια όλες οι προκύπτουσες μηχανές μέχρι τώρα έχουν αποδειχθεί ισοδύναμες με την βασική μηχανή Turing, αλλά παρόλο που δεν έχει βρεθεί επέκταση που να υπολογίζει καινούργια προβλήματα υπάρχουν μηχανές που υπολογίζουν πιο αποτελεσματικά τα υπάρχοντα προβλήματα και αυτό είναι σημαντικό. Άλλες πάλι επεκτάσεις έχουν το πλεονέκτημα ότι είναι πιο βολικές και μας επιτρέπουν να σχεδιάζουμε μηχανές Turing πολύ ευκολότερα από ότι με το βασικό σχήμα.

Εμείς δώσαμε έμφαση σε αυτό το κεφάλαιο στην λειτουργία της ντετερμινιστικής και μη ντετερμινιστικής μηχανής Turing, όπως και στη μηχανή $k-$ταινιών και τυχαίας προσπέλασης. Η διαφορά που εντοπίσαμε όσον αφορά την μη ντετερμινιστική από την ντετερμινιστική μηχανή είναι ότι στην μη ντετερμινιστική δίνουμε κατά κάποιο τρόπο στην μηχανή την ελευθερία να βρίσκεται ταυτόχρονα σε περισσότερα από ένα σημεία του υπολογισμού. Επίσης η μηχανή $k-$ταινιών παρέχει πολύ μεγαλύτερη ευχέρεια από την ντετερμινιστική στον σχεδιασμό του προγράμματος. Ο λόγος είναι ότι κάθε μη τετριμμένο πρόγραμμα χρησιμοποιεί ένα αριθμό από βοηθητικές μεταβλητές και με τις $k-$ταινίες υπάρχει η δυνατότητα να χρησιμοποιηθεί μία ταινία για κάθε μεταβλητή του προγράμματος, γεγονός που διευκολύνει τον σχεδιασμό. Τέλος η μηχανή Τυχαίας Προσπέλασης είναι ένα τυπικό μοντέλο που μοιάζει πολύ με τους σύγχρονούς υπολογιστές και ο χρόνος που απαιτεί η προσομοίωσή του από τη μηχανή Turing είναι ένα μικρό πολυώνυμο του χρόνου που απαιτεί η μηχανή Turing και αυτό το καθιστά σημαντικό.

Κεφάλαιο 3

ΟΙ ΚΛΑΣΕΙΣ P, NP ΚΑΙ NP-ΠΛΗΡΗΣ

Σε αυτό το κεφάλαιο εξετάζονται οι τρεις σημαντικές κλάσεις η P, η NP και η NP-πλήρης κλάση και οι σχέσεις που υπάρχουν μεταξύ τους. Στην αρχή γίνεται μία αναφορά στην κλάση P(polynomial time), η οποία είναι η πρώτη κλάση πολυπλοκότητας που συναντάμε, στην συνέχεια αναφέρεται η NP και τέλος γίνεται εκτενή αναφορά στην κλάση NP-πλήρη.

Σε αυτό το σημείο αναφέρονται επίσης πολύ κρίσιμες έννοιες όπως αυτή της αναγωγής και της πληρότητας. Τέλος υπάρχει μία ανάλυση της λογικής Bool, του θεωρήματος του Cook και Levin και του προβλήματος της ικανοποιησιμότητας.

3.1 Η ΚΛΑΣΗ P (POLYNOMIAL TIME)

Μια **πολυωνυμικά φραγμένη** μηχανή Turing ορίζεται ως εξής:

Ορισμός 3.1 :
Μία μηχανή Turing $M = (K, \Sigma, \delta, \varsigma, \mathrm{H})$ ονομάζεται πολυωνυμικά φραγμένη αν υπάρχει πολυώνυμο $p(n)$ τέτοιο ώστε να ισχύει το εξής: Για κάθε είσοδο x, δεν υπάρχει συνολική κατάσταση C τέτοια ώστε $(s, \triangleright\sqcup x)\vdash_M^{p(|x|)+1} C$.
Με άλλα λόγια η μηχανή τερματίζει πάντα μετά από $p(n)$ το πολύ βήματα, όπου n είναι το μήκος της εισόδου.
Μία γλώσσα ονομάζεται **πολυωνυμικά αποφασίσιμη** αν υπάρχει μία πολυωνυμικά φραγμένη μηχανή Turing που την αποφασίζει. Η κλάση όλων των πολυωνυμικά φραγμένων γλωσσών συμβολίζεται ως P. [1]

Η κλάση P εμπεριέχει όλα τα προβλήματα που λύνονται σε πολυωνυμικό χρόνο. Δηλαδή

σε αυτή την κλάση ανήκουν τα προβλήματα για τα οποία υπάρχει πολυωνυμικός αλγόριθμος που τα επιλύει. Πιο συγκεκριμένα, ανήκουν τα προβλήματα που λύνονται σε χρόνο $O(n^k)$ όπου k σταθερός ακέραιος και όπου n είναι το μέγεθος της εισόδου του προβλήματος, δηλαδή,

$$\text{P}=\bigcup DTIME\ (). \qquad [4]$$

Η κλάση P περιέχει τις γλώσσες που μπορούν να αποφασιστούν αποδοτικά, δηλαδή σε χρόνο ο οποίος φράσσεται από κάποιο πολυώνυμο στο μέγεθος της εισόδου. Οι πολυωνυμικά φραγμένες μηχανές Turing και η κλάση P εκφράζουν ικανοποιητικά τις διαισθητικές έννοιες, αντίστοιχα, των πρακτικά εφικτών αλγορίθμων και των ρεαλιστικά επιλύσιμων προβλημάτων.

Υπάρχουν πολλά προβλήματα που ανήκουν στην κλάση P. Για παράδειγμα, όλες οι κανονικές γλώσσες και όλες οι γλώσσες χωρίς συμφραζόμενα, ανήκουν εκεί. Επίσης γνωρίζουμε ότι η αντανακλαστική μεταβατική κλειστότητα μιας σχέσης μπορεί να υπολογιστεί σε πολυωνυμικό χρόνο. Ένα άλλο τέτοιο πρόβλημα είναι επίσης το 2-SAT.

Ένα από τα πιο γνωστά προβλήματα της κλάσης P είναι ο κύκλος EULER το οποίο είναι το εξής:

Κύκλος EULER :Δεδομένου ενός γραφήματος G, υπάρχει κλειστό μονοπάτι στο G το οποίο χρησιμοποιεί κάθε ακμή ακριβώς μία φορά.

Δεν είναι δύσκολο να δει κανείς ότι το πρόβλημα κύκλος EULER ανήκει στο P-με αυτό εννοούμε φυσικά ότι η αντίστοιχη γλώσσα $L = \{k(G)$:όπου το G είναι γράφημα EULER ανήκει στο P.

3.2 Η ΚΛΑΣΗ NP (nondeterministic polynomial time)

Μία άλλη σημαντική κλάση είναι η κλάση NP [1], το σύνολο όλων των γλωσσών που μπορούν να αποφασιστούν από μία μη ντετερμινιστική μηχανή Touring M, με πολυπλοκότητα χρόνου $O(n^k)$,όπου k θετικός ακέραιος, δηλαδή,

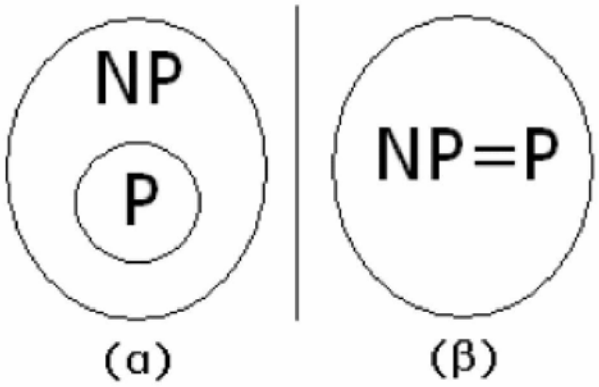
$$NP = \bigcup NTIME(n^k). \qquad [4]$$

Η κλάση NP περιέχει τις γλώσσες που μπορούν να επαληθευτούν αποδοτικά, δηλαδή υπάρχει γι΄ αυτές επαληθευτής (verifier) που λειτουργεί σε χρόνο που φράσσεται από κάποιο πολυώνυμο στο μέγεθος της εισόδου. Αυτό σημαίνει ότι αν δινόταν κατά κάποιο τρόπο ένα «πιστοποιητικό» της λύσης του προβλήματος, τότε θα μπορούσαμε να επαληθεύσουμε το γεγονός ότι το πιστοποιητικό είναι σωστό σε πολυωνυμικό χρόνο δηλαδή σε χρόνο που φράσσεται από κάποιο πολυώνυμο στο μέγεθος της εισόδου του προβλήματος. Ένα πιστοποιητικό

πρέπει να είναι πολυωνυμικά σύντομο, δηλαδή να έχει μήκος το οποίο να είναι το πολύ ένα πολυώνυμο ως προς το μήκος της εισόδου. Όλα τα προβλήματα στο NP έχουν πιστοποιητικά, αλλά μόνο προβλήματα στο NP έχουν πιστοποιητικά **[3]**.

Με άλλα λόγια η κλάση NP είναι η κλάση που εμπεριέχει όλα τα προβλήματα που επαληθεύουν την λύση τους σε πολυωνυμικό χρόνο. Τέτοια προβλήματα είναι το πρόβλημα του περιοδεύοντος πωλητή, κύκλος HAMILTON, διαμέριση κ.α.

Προφανώς κάθε πρόβλημα στη κλάση P ανήκει επίσης και στην NP, δηλαδή ισχύει $P \subseteq NP$ (αυτό ισχύει διότι οι ντετερμινιστικές μηχανές είναι απλά μη ντετερμινιστικές, στις οποίες οι σχέσεις μετάβασης συμβαίνει να είναι συνάρτηση). Ένα από τα σημαντικότερα προβλήματα της επιστήμης των υπολογιστών είναι το ερώτημα για το αν οι κλάσεις προβλημάτων NP και P ταυτίζονται ή αν το σύνολο NP είναι υπερσύνολο του P. Με άλλα λόγια μας ενδιαφέρει να δούμε αν μπορούν να εφευρεθούν ή όχι, αλγόριθμοι πολυωνυμικής πολυπλοκότηταs(συναρτήσει του μεγέθους του στιγμιότυπου) οι οποίοι να επιλύουν τα προβλήματα της κλάσης NP. Το σχήμα 3.2(α) παρακάτω, παρουσιάζει τις δύο αυτές πιθανές πραγματικότητες. Αυτό το ερώτημα μετά από πολλά χρόνια έρευνας δεν έχει ακόμα απαντηθεί αυστηρά αν και υπάρχουν ισχυρές ενδείξεις ότι τα δύο σύνολα δεν είναι ίδια και ότι υπάρχουν προβλήματα που ανήκουν στην κλάση NP αλλά όχι στην P.

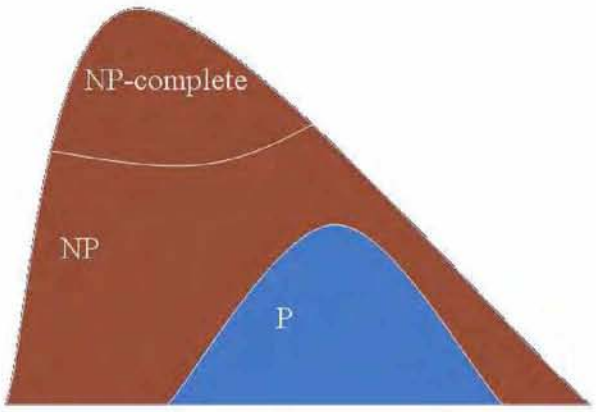


Σχήμα 3.2 (a) - (α) και (β)

Σχήμα 3.2 (a) : Τα δύο ενδεχόμενα όσον αφορά τη σχέση των κλάσεων NP και P. Το (α) φέρεται και ως το πιθανότερο.

ΕΙΚΑΣΙΑ: P=? NP

Αυτό είναι ουσιαστικά το $P = NP$ πρόβλημα που αποτελεί το πιο σημαντικό ανοικτό πρόβλημα στην θεωρητική επιστήμη των υπολογιστών σήμερα. Στο http://www.claymath.org προσφέρονται 1 εκ.δολάρια για την λύση του!!!!



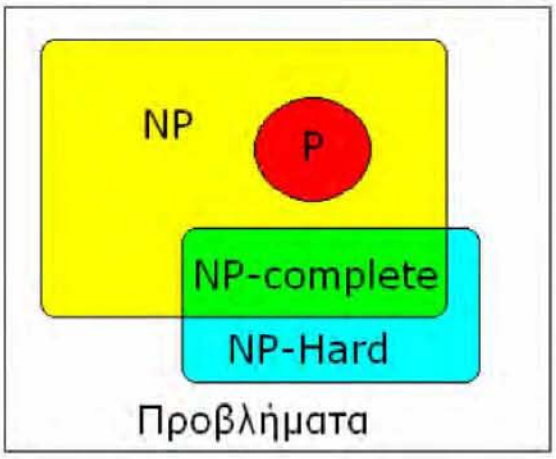
Σχήμα 3.2 (b)

Μερικά από τα προβλήματα που ανήκουν στο NP είναι «δύσκολα» προβλήματα για τα οποία ξέρουμε μόνο εκθετικούς αλγορίθμους για την επίλυσή τους και όχι πολυωνυμικούς. Ίσως υπάρχει πολυωνυμική λύση αλλά δεν έχει βρεθεί ακόμα.

3.3 Η NP-ΠΛΗΡΗΣ ΚΛΑΣΗ

Μία ενδιαφέρουσα κλάση προβλημάτων γύρω από τη σχέση των συνόλων NP και P είναι η NP- Hard. Ένα πρόβλημα χαρακτηρίζεται NP-Hard αν όλα τα προβλήματα της κλάσης NP μπορούν να αναχθούν σε πολυωνυμικό χρόνο σε αυτό. Αν μάλιστα το NP-Hard πρόβλημα είναι και NP τότε το ονομάζουμε NP-ΠΛΗΡΕΣ(NP-complete) **[4],[6].**

Η κλάση NP-πλήρης λοιπόν αποτελεί το υποσύνολο των πιο δύσκολων προβλημάτων του NP . Αν ένα από αυτά τα προβλήματα ανήκει στο P, τότε $P = NP$.Επίσης αν ένα NP-πλήρες πρόβλημα μπορεί να λυθεί σε πολυωνυμικό χρόνο, τότε κάθε πρόβλημα στην κλάση NP-πλήρη έχει αλγόριθμο που το επιλύει σε πολυωνυμικό χρόνο. Η σημασία λοιπόν, της κλάσης NP-Hard έγκειται στο γεγονός ότι αν ανακαλυφθεί ένας αλγόριθμος πολυωνυμικής πολυπλοκότητας $O(n^k)$ που να επιλύει ένα NP-Hard πρόβλημα Α τότε όλα τα NP προβλήματα Β θα επιλύονται σε πολυωνυμικό χρόνο. Θα τα αναγάγουμε σε πολυωνυμικό χρόνο $O(n^m)$ στο πρόβλημα Α και κατόπιν θα λύνουμε το Α σε χρόνο $O(n^k)$. Έτσι ένα NP πρόβλημα Β θα επιλύεται σε χρόνο $O(n^m+n^k)$. Με $O(n^m+n^k) = O(n^m)$ αν $m \geq k$ και $O(n^m+n^k) = O(n^k)$ αν $m < k$.



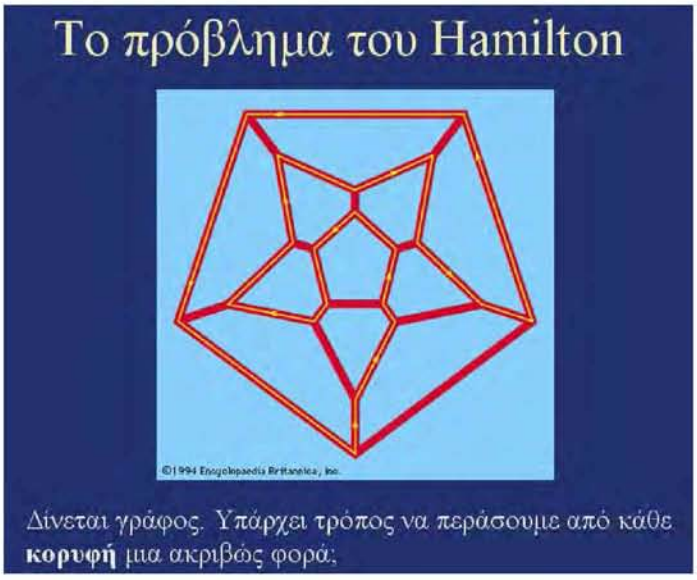
Σχήμα 3.3 (a)

Σχήμα 3.3 (a) : Τα πιθανά ενδεχόμενα όσον αφορά τη σχέση των κλάσεων P, NP, NP-Hard και NP-Complete .

Οι πιθανές σχέσεις των συνόλων P, NP, NP-Hard και NP-Complete σκιαγραφούνται στο παραπάνω σχήμα 3.3. Οι περιοχές που υπάρχουν σίγουρα είναι η κόκκινη (τα προβλήματα που ανήκουν στην κλάση P αλλά και στην κλάση NP) , η πράσινη (τα προβλήματα που ανήκουν στην κλάση NP και στην κλάση NP-Hard , τα NP-Complete δηλαδή) και η γαλάζια (δηλαδή τα προβλήματα που είναι NP-Hard αλλά όχι NP).

Σημείωση: Ένα παράδειγμα προβλήματος της 'γαλάζιας περιοχής' είναι το πρόβλημα της εύρεσης του κατά πόσο ένα πολυώνυμο με περισσότερες από μία μεταβλητές έχει ακέραια ρίζα. Το πρόβλημα αυτό δεν είναι καν αποκρίσιμο και επομένως ούτε NP. Από την άλλη το πρόβλημα αυτό μπορεί να αποδειχθεί ότι είναι NP-Hard .Να θυμίσουμε εδώ ότι τα NP προβλήματα αποφασίζονται από μία μη- ντετερμινιστική μηχανή Turing . Επομένως κάθε NP πρόβλημα είναι αποκρίσιμο.

Υπάρχουν πολλά τέτοια προβλήματα που ανήκουν σε αυτή τη κλάση. Το πρόβλημα του κύκλου Hamilton και το πρόβλημα του περιοδεύοντος πωλητή (TSP) , το πρόβλημα της ικανοποιησιμότητας και παραλλαγές του (SAT,3SAT,MAXSAT) ,το πρόβλημα της ακριβούς επικάλυψης (VERTEX-COVER),το πρόβλημα της κλίκας (CLIQUE) , είναι κάποια από τα πολλά προβλήματα που ανήκουν σε αυτή τη κλάση. Όμως για να μπορέσουμε να μελετήσουμε εκτενέστερα αυτά τα δύσκολα, από άποψη πολυπλοκότητας προβλήματα, πρέπει πρώτα να αναφέρουμε κάποια πράγματα όσον αφορά τη NP-πλήρη κλάση.



Σχήμα 3.3 (b)

Σχήμα 3.3 (b) **:** Το πρόβλημα του Hamilton .

3.3.1 ΑΝΑΓΩΓΕΣ

Τα NP-πλήρη προβλήματα έχουν την εξής σημαντική ιδιότητα: Όλα τα προβλήματα στο NP μπορούν να αναχθούν σε αυτά μέσω αναγωγών πολυωνυμικού χρόνου(με τον ίδιο τρόπο που όλες οι αναδρομικά απαριθμήσιμες γλώσσες ανάγονται στο πρόβλημα του τερματισμού).

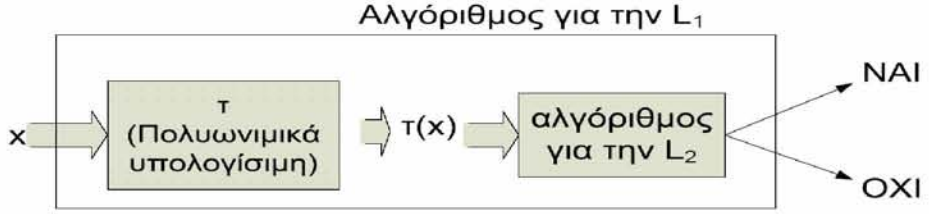
Ορισμός 3.3.1 (a)

Μία συνάρτηση $f : \Sigma^* \to \Sigma^*$ ονομάζεται υπολογίσιμη σε πολυωνυμικό χρόνο αν υπάρχει μία πολυωνυμικά φραγμένη μηχανή TURING M που την υπολογίζει. Έστω τώρα οι γλώσσες $L_1, L_2 \subseteq \Sigma^*$.Μία συνάρτηση υπολογίσιμη σε πολυωνυμικό χρόνο $\tau : \Sigma^* \to \Sigma^*$ ονομάζεται πολυωνυμική αναγωγή από την L_1 στην L_2 αν για κάθε $x \in \Sigma^*$ ισχύει το εξής: $x \in L_1$ αν και μόνο αν $\tau(x) \in L_2$. **[1]**

Οι πολυωνυμικές αναγωγές είναι σημαντικές διότι αποκαλύπτουν ενδιαφέρουσες σχέσεις μεταξύ υπολογιστικών προβλημάτων. Μία πολυωνυμική αναγωγή συσχετίζει δύο γλώσσες, όχι δύο προβλήματα. Ωστόσο, γνωρίζουμε ότι οι γλώσσες μπορούν να χρησιμοποιηθούν για την κωδικοποίηση σημαντικών υπολογιστικών προβλημάτων, όλων των ειδών. Επομένως, μπορούμε να πούμε ότι η τ είναι πολυωνυμική αναγωγή από το πρόβλημα L_1 στο πρόβλημα L_2 αν είναι πολυωνυμική αναγωγή μεταξύ των αντίστοιχων γλωσσών. Δηλαδή, η τ

μετασχηματίζει σε πολυωνυμικό χρόνο στιγμιότυπα του προβλήματος L_1 σε στιγμιότυπα του προβλήματος L_2 με τέτοιο τρόπο ώστε η x να είναι ένα «ναι» στιγμιότυπο του προβλήματος L_1 αν και μόνο αν η $\tau(x)$ είναι ένα «ναι» στιγμιότυπο του προβλήματος L_2.

Όταν έχουμε μία πολυωνυμική αναγωγή τ από το πρόβλημα L_1 στο πρόβλημα L_2, είναι δυνατόν να προσαρμόσουμε κάθε αλγόριθμο πολυωνυμικού χρόνου για το L_2 ώστε να κατασκευάσουμε έναν για τοL_1. Αν έχουμε έναν πολυωνυμικό αλγόριθμο για το L_2, η μέθοδος αυτή για την επίλυση του L_1 είναι επίσης πολυωνυμική, καθώς και το βήμα της αναγωγής και ο αλγόριθμος για την επίλυση του L_2, μπορούν να πραγματοποιηθούν σε πολυωνυμικό χρόνο. Με άλλα λόγια η ύπαρξη μιας πολυωνυμικής αναγωγής από το L_1 στο L_2 αποτελεί απόδειξη ότι το L_2 είναι τουλάχιστον όσο δύσκολο είναι τοL_1. Αν το L_2 είναι επιλύσιμο αποδοτικά τότε το ίδιο πρέπει να είναι και το L_1, και επίσης αν το L_1 απαιτεί εκθετικό χρόνο, τότε το ίδιο ισχύει και για το L_2.[1],[22].



Σχήμα 3.3.1

ΘΕΩΡΗΜΑ 3.3.1(a) :

Αν τ_1 είναι μία πολυωνυμική αναγωγή από την L_1 στην L_2 και τ_2 είναι μία πολυωνυμική αναγωγή από την L_2 στην L_3,τότε η σύνθεσή τους $\tau_1 \circ \tau_2$ είναι μία πολυωνυμική αναγωγή από την L_1 στην L_3. [1]

Ορισμός 3.3.1(b) :

Μία γλώσσα $L \subseteq \Sigma^*$ ονομάζεται NP-πλήρης αν
α)$L \in NP$ και
β)για κάθε γλώσσα $L' \in NP$,υπάρχει μία πολυωνυμική αναγωγή από την L' στην L. [1]

ΘΕΩΡΗΜΑ 3.3.2(b) :

Έστω L μία NP- πλήρης γλώσσα. Τότε $P = NP$ αν και μόνο αν$L \in P$. [1]

3.3.1.1 ΔΥΟ ΤΡΟΠΟΙ ΧΡΗΣΙΜΟΠΟΙΗΣΗΣ ΤΩΝ ΑΝΑΓΩΓΩΝ

Μέχρι στιγμής ο στόχος μιας αναγωγής από το πρόβλημα Α στο πρόβλημα Β ήταν απλός και ξεκάθαρος: Εμείς ξέρουμε τον τρόπο με τον οποίο μπορούμε να λύσουμε το πρόβλημα Β αποτελεσματικά και θέλουμε να χρησιμοποιήσουμε αυτή την γνώση για να λύσουμε ένα πρόβλημα Α. Ωστόσο την αναγωγή ενός προβλήματος σε ένα άλλο μπορούμε να την χρησιμοποιήσουμε και για ένα πιο συγκεκριμένο σκοπό: Εμείς ξέρουμε ότι το Α ανήκει στην κατηγορία των δύσκολων προβλημάτων και θέλουμε να χρησιμοποιήσουμε την αναγωγή για να αποδείξουμε ότι το πρόβλημα Β ανήκει και αυτό στην κατηγορία των δύσκολων προβλημάτων.

Αν θέλουμε, λοιπόν, να δείξουμε μία τέτοια αναγωγή από το πρόβλημα Α στο Β, τη συμβολίζουμε με $A \rightarrow B$.

Επίσης στις αναγωγές ισχύει η εξής ιδιότητα:
Αν $A \rightarrow B$ και $B \rightarrow \Gamma$ τότε $A \rightarrow \Gamma$([**9**]).
νοινδεντ

3.3.2 ΤΟ ΘΕΩΡΗΜΑ ΤΩΝ COOK ΚΑΙ LEVIN

Το θεώρημα αυτό αποδείχθηκε από τον S.Cook [**23**] και ανεξάρτητα από τον L. Levin[**24**].Ήταν η αρχή στο να δειχθεί ότι πολλά δύσκολα προβλήματα που απασχολούσαν διάφορες περιοχές της Επιστήμης των Υπολογιστών, των Μαθηματικών, της Επιχειρησιακής Έρευνας κλπ. ήταν στην πραγματικότητα NP-πλήρη. Η σημασία του αποτελέσματος αυτού έγινε αντιληπτή στην κλασσική εργασία τουR. Karp [**25**],όπου πολλά γνωστά προβλήματα δείχθηκαν NP-πλήρη. Από τότε εκατοντάδες προβλημάτων έχουν δειχθεί NP-πλήρη. Για μία εξαιρετική ανάπτυξη του θέματος ο αναγνώστης παραπέμπεται στο [**7**].

3.3.2.1 ΔΙΑΤΥΠΩΣΗ ΚΑΙ ΣΗΜΑΣΙΑ ΤΟΥ ΘΕΩΡΗΜΑΤΟΣ ΤΩΝ COOK ΚΑΙ LEVIN

Το θεώρημα των COOK και LEVIN είναι πολύ σημαντικό και η αξία του μπορεί να προσδιοριστεί στα δύο παρακάτω σημεία:

• Αρκεί να δείξουμε ότι ένα πρόβλημα της κλάσης NP- Hard ανήκει στην κλάση P για να αποδείξουμε την ισότητα $P = NP$.

• Αν αποδείξουμε ότι ένα πρόβλημα Α είναι NP-Hard τότε μπορούμε να αποδείξουμε ότι και ένα άλλο πρόβλημα Β είναι NP-Hard κάνοντας πολυωνυμική αναγωγή από το Α στο Β ($A \leq_p B$).Στη προηγούμενη μας παράγραφο καθορίσαμε την έννοια της αναγωγής ενός προβλήματος σε ένα άλλο. Όταν ένα πρόβλημα Α το αναγάγουμε σε ένα πρόβλημα Β τότε η λύση του Β μπορεί να χρησιμοποιηθεί για την λύση του Α. Αν η αναγωγή αυτή υπάρχει τότε κάθε πρόβλημα Γ της κλάσης NP θα ανάγεται στο Β ακολουθώντας την αναγωγή $\Gamma \leq_p A \leq_p B$. Δεν έχουμε εδραιώσει ότι υπάρχουν NP- πλήρη προβλήματα, αλλά πράγματι υπάρχουν πολλά τέτοια προβλήματα. Από την στιγμή που έχουμε αποδείξει το πρώτο NP-πλήρες πρόβλημα, μπορούμε να αποδείξουμε για περισσότερα προβλήματα ότι είναι NP-πλήρη, αν αναγάγουμε σε αυτά ένα πρόβλημα που είναι ήδη γνωστό ότι είναι NP-πλήρες και χρησιμοποιήσουμε την μεταβατικότητα των πολυωνυμικών αναγωγών. Η πρώτη όμως απόδειξη NP-πληρότητας θα πρέπει να είναι μία εφαρμογή του ορισμού: Πρέπει να αποδείξουμε ότι όλα τα προβλήματα στο NP ανάγονται στο εν λόγω πρόβλημα. Ιστορικά το πρώτο πρόβλημα που αποδείχθηκε ότι είναι NP- πλήρες από τον Stephen A. Cook το 1971 ήταν η ΙΚΑΝΟΠΟΙΗΣΙΜΟΤΗΤΑ (SAT).

Το θεώρημα τουCook και Levin**[6],[5]** διατυπώνεται με δύο τρόπους:
Το πρόβλημα SAT είναι NP-πλήρες
και
Αν $SAT \in P$ τότε $P = NP$

3.3.2.2 Η ΛΟΓΙΚΗ BOOL

Για να αποδείξουμε ότι το SAT είναι NP-πλήρες πρόβλημα πρώτα πρέπει να περιγράψουμε κάποια τεχνικά χαρακτηριστικά που έχουν να κάνουν με την λογική BOOL και το πως αυτή λειτουργεί για να την χρησιμοποιήσουμε στην απόδειξή μας για το SAT.

Στη λογική BOOL χρησιμοποιούμε μεταβλητές Bool $x_1, x_2, ...$Κάθε μεταβλητή συμβολίζει μία πρόταση που μπορεί εκ φύσεως να είναι αληθής (true) ή ψευδής (false) ανεξάρτητα από την τιμή αληθείας των υπολοίπων. Στη συνέχεια χρησιμοποιούμε συνδέσμους Bool για να συνδυάσουμε μεταβλητές Bool και να σχηματίσουμε πιο περίπλοκους τύπους Bool. Για την απόδειξη του προβλήματός μας χρειάζεται να εστιάσουμε σε τύπους Bool ενός συγκεκριμένου είδους ,το οποίο ορίζουμε παρακάτω.

Ορισμός 3.3.2.2(a):
Έστω $X = \{x_1, x_2, ..., x_n\}$ ένα πεπερασμένο σύνολο από μεταβλητές Bool, και έστω $\overline{X} = \{\overline{x_1}, \overline{x_2}, ..., \overline{x_n}\}$ όπου τα $\overline{x_1}, \overline{x_2}, ..., \overline{x_n}$ είναι νέα σύμβολα για τις αρνήσεις των $x_1, x_2, ..., x_n$. Ονομάζουμε τα μέλη του $X \cup \overline{X}$ (**literals**)**στοιχεία** (οι μεταβλητές είναι **θετικά στοιχεία** ,ενώ οι αρνήσεις των μεταβλητών είναι **αρνητικά στοιχεία**). Μία **συνθήκη**

C(**clause**) είναι ένα μη κενό σύνολο στοιχείων $C \subseteq X \cup \overline{X}$. Τέλος ένας τύπος Bool σε κανονική διαζευκτική μορφή είναι ένα σύνολο από συνθήκες ορισμένες στο X.

Με άλλα λόγια μία λογική έκφραση ονομάζεται **literal** αν είναι προτασιακή μεταβλητή ή άρνηση προτασιακής μεταβλητής. Επίσης μία λογική έκφραση ονομάζεται **clause** αν αποτελεί την διάζευξη ενός πεπερασμένου πλήθους από literals.

Μία λογική έκφραση που βρίσκεται σε *συζευκτική κανονική μορφή* (conjunctive normal form-CNF) αποτελεί την σύζευξη ενός πεπερασμένου πλήθους από clauses. Ένα clause μπορεί να αποτελείται από ένα μόνο literal, ενώ αντίστοιχα ένα clause αποτελεί μόνο του λογική έκφραση σεCNF. **[1]**

π.χ. έστω $X = \{x_1, x_2, ..., x_n\}$ ένα πεπερασμένο σύνολο από μεταβλητές Bool , και έστω $\overline{x_1}, \overline{x_2}, ..., \overline{x_n}$ οι αρνήσεις τους.Η $C_1 = (x_1, \overline{x_2}, x_3)$ είναι μία συνθήκη.Για να διαχωρίσουμε τα διάφορα στοιχεία στη συνθήκη χρησιμοποιούμε το διαχωριστικό $\vee$(**or**, στα ελληνικά **ή**) οπότε η παραπάνω συνθήκη μπορεί να γραφτεί ως $C = (x_1 \vee \overline{x_2} \vee x_3)$, και ένας τύπος Bool σε κανονική διαζευκτική μορφή:

$$F = \{(x_1 \vee \overline{x_2} \vee x_3), (\overline{x_1}), (x_2 \vee \overline{x_2})\}.$$

Αποτελείται από τρεις συνθήκες ,μία από τις οποίες είναι η C παραπάνω.
Παραπάνω ορίσαμε μόνο το συντακτικό, ή τη φαινομενική δομή ,του τύπου Bool. Στη συνέχεια ορίζουμε τη σημασιολογία, ή τη σημασία ενός τέτοιου τύπου.

Ορισμός 3.3.2.2(b):

Έστω F ένας τύπος Bool σε κανονική διαζευκτική μορφή ορισμένος ως προς τις μεταβλητές του $X = \{x_1, x_2, ..., x_n\}$.Μία απόδοση τιμών αληθείας για τον F είναι μία απεικόνιση από το X στο σύνολο $\{\top, \bot\}$, όπου $\top$ και $\bot$ είναι δύο νέα σύμβολα τα οποία συμβολίζουν το αληθές (true) και το ψευδές (false), αντίστοιχα. Λέμε ότι μία απόδοση τιμών αληθείας T ικανοποιεί τον F αν ισχύει το εξής: Για κάθε συνθήκη $C \in F$ υπάρχει τουλάχιστον μία μεταβλητή x_i τέτοια ώστε είτε (α) $\mathrm{T}(x_i)=\top$ και $x_i \in C$, είτε (β) $\mathrm{T}(x_i) = \bot$ και $\overline{x_i} \in C$. Δηλαδή μία συνθήκη ικανοποιείται αν περιέχει τουλάχιστον ένα αληθές στοιχείο, όπου το x_i θεωρείται αληθές αν και μόνο αν $\mathrm{T}(x_i) = \top$,και το $\overline{x_i}$ θεωρείται αληθές αν και μόνο αν $\mathrm{T}(x_i) = \bot$.
Τέλος ο F ονομάζεται ικανοποιήσιμος αν υπάρχει απόδοση τιμών αληθείας που τον ικανοποιεί. **[1]**

π.χ. Ο τύπος Bool $F(1)$ παραπάνω ικανοποιείται από την απόδοση τιμών αληθείας T, όπου $\mathrm{T}(x_1) = \bot, \mathrm{T}(x_2) = \top$,και $\mathrm{T}(x_3) = \top$.Η απόδοση τιμών αληθείας αυτή ικανοποιεί την συνθήκη $C = (x_1 \vee \overline{x_2} \vee x_3)$ επειδή $\mathrm{T}(x_3) = \top$ και $x_3 \in C_1$. Η T ικανοποιεί την

συνθήκη $C_2 = (\overline{x_1})$ επειδή $\overline{x_1} \in C_2$ και $\mathrm{T}(x_i) = \bot$. Τέλος η T ικανοποιεί την τρίτη συνθήκη $C_3 = (x_2 \vee \overline{x_2})$ (αυτό δεν προκαλεί μεγάλη έκπληξη αφού οποιαδήποτε απόδοση τιμών αληθείας θα ικανοποιούσε την C_3).Υπάρχουν πολλές αποδόσεις τιμών αληθείας που αποτυγχάνουν να ικανοποιήσουν τονF:Για παράδειγμα, οποιαδήποτε απόδοση τιμών αληθείας $\mathrm{T}'(x_1)$ με $\mathrm{T}(x_1) = \bot$ δεν θα μπορούσε να ικανοποιήσει την C_2, και επομένως αποτυγχάνει να ικανοποιήσει τον F. Παρ' όλα αυτά, ο F είναι ικανοποιήσιμος, διότι υπάρχει τουλάχιστον μία απόδοση τιμών αληθείας που τον ικανοποιεί. Έτσι προκύπτει το εξής σημαντικό πρόβλημα:

ΙΚΑΝΟΠΟΙΗΣΙΜΟΤΗΤΑ(SATISFABILITY): Δεδομένου ενός τύπου Bool F σε κανονική διαζευκτική μορφή, είναι ικανοποιήσιμος;

3.3.2.3 ΤΟ ΠΡΟΒΛΗΜΑ ΤΗΣ ΙΚΑΝΟΠΟΙΗΣΙΜΟΤΗΤΑΣ (SAT)

Το πρόβλημα της ικανοποιησιμότητας(SAT) **[4],[9]** είναι ένα πρόβλημα με μεγάλη πρακτική σημασία με εφαρμογές που κυμαίνονται από τον σχεδιασμό υπολογιστών έως την ανάλυση της εικόνας και την τεχνολογία του λογισμικού. Επίσης ανήκει στην κατηγορία των δύσκολων προβλημάτων. Μία περίπτωση του προβλήματος είναι η παρακάτω:

$(x \vee y \vee z)\ (x \vee \overline{y})\ (y \vee \overline{z})\ (z \vee \overline{x})\ (\overline{x} \vee \overline{y}\ \vee \overline{z})$:

Όπως είδαμε και προηγουμένως ο παραπάνω είναι ένας τύπος Bool σε κανονική διαζευκτική μορφή. Αυτός ο τύπος Bool είναι ικανοποιήσιμος αν υπάρχει απόδοση τιμών αληθείας που τον ικανοποιεί.

Το πρόβλημα SAT είναι το ακόλουθο: Δεδομένου ενός τύπου Bool σε κανονική διαζευκτική μορφή ,πρέπει να βρούμε αν υπάρχει απόδοση τιμών αληθείας που τον ικανοποιεί ή να αναφέρουμε ότι δεν υπάρχει. Δηλαδή δίνεται μία λογική έκφραση ϕ σε συζευκτική κανονική μορφή και ζητείται να αποφασίσουμε αν η ϕ είναι ικανοποιήσιμη(δηλαδή αν υπάρχει ανάθεση αληθοτιμών T στις μεταβλητές της ϕ τέτοια ώστε $T| = \phi$).

Το πρόβλημα SAT είναι ένα τυπικό πρόβλημα αναζήτησης .Μας δίνεται μία περίπτωση A (όπου είναι κάποια δεδομένα εισόδου, στην περίπτωσή μας είναι ένας τύποςBoolσε κανονική διαζευκτική μορφή) και μας ζητείται να βρούμε μία λύση B (στην περίπτωσή μας είναι μία ανάθεση που ικανοποιεί κάθε στοιχείο).Αν δεν υπάρχει λύση θα πρέπει να το αναφέρουμε.

Πιο συγκεκριμένα σε ένα πρόβλημα αναζήτησης κάθε προτεινόμενη λύση B σε μία περίπτωση A θα πρέπει να μπορεί να ελέγχεται γρήγορα για την ακρίβεια. Αυτό σημαίνει ότι για ένα αντικείμενο, η λύση B θα πρέπει να είναι συνοπτική(για να διαβαστεί γρήγορα) με μήκος το πολυωνυμικό όριο της περίπτωσης A. Αυτό είναι ξεκάθαρο ότι ισχύει στην περίπτωση του SAT όπου κάθε λύση B είναι μία ανάθεση στις μεταβλητές .Για να τυποποιήσουμε την έννοια του γρήγορου ελέγχου, θα πούμε ότι υπάρχει ένας αλγόριθμος πολυωνυμικού χρόνου που παίρνει σαν είσοδο A και B και αποφασίζει αν το B είναι λύση της A ή όχι. Για το SAT αυτό

είναι εύκολο αφού αυτό ακριβώς περιλαμβάνει ο έλεγχος, εάν η ανάθεση από το B πράγματι ικανοποιεί κάθε στοιχείο του A.

Επομένως:

Ένα πρόβλημα αναζήτησης διευκρινίζεται από έναν αλγόριθμο C που δέχεται δύο εισόδους, μία περίπτωση A και μία προτεινόμενη λύση B, και τρέχει σε πολυωνυμικό χρόνο |A|. Λέμε ότι η B είναι μία λύση του A αν και μόνο αν $C(\mathrm{A}, \mathrm{B}) = \mathrm{true}$.

Λαμβάνοντας υπόψη τη σημασία του προβλήματος αναζήτησης SAT , οι ερευνητές κατά τη διάρκεια των προηγούμενων 50 ετών έχουν προσπαθήσει να βρουν ικανοποιητικούς τρόπους να λύσουν το πρόβλημα, αλλά χωρίς επιτυχία. Οι γρηγορότεροι αλγόριθμοι που έχουμε είναι εκθετικοί.

Ακόμα, αξίζει να αναφέρουμε ότι υπάρχουν δύο φυσικές παραλλαγές του SAT για τις οποίες έχουμε καλούς αλγορίθμους. Αν όλες οι προτάσεις περιέχουν το πολύ ένα θετικό στοιχείο, τότε για την φόρμουλά Boolean (Horn formula) μία ικανοποιητική απόδοση αληθείας μπορεί να βρεθεί με τον άπληστο αλγόριθμο (greedy algorithm).Αν όλες οι προτάσεις έχουν μόνο δύο στοιχεία τότε το SAT μπορεί να λυθεί σε γραμμικό χρόνο βρίσκοντας τα δυνατότερα συνδεδεμένα συστατικά από ένα ειδικό γράφημα.

Μια ειδική περίπτωση του SAT αποτελούμενο από προτάσεις δύο στοιχείων είναι το 2SAT. Επίσης ένα άλλο πρόβλημα, που ανήκει στην κατηγορία των δύσκολων προβλημάτων, είναι το 3SAT. Το 3SAT αποτελείται από προτάσεις που περιέχουν τρία στοιχεία.

Σημείωση:

Το πρόβλημα SAT αντιστοιχεί στην γλώσσα L_{SAT} (που περιλαμβάνει όλες τις συμβολοακολουθίες που περιγράφουν ικανοποιήσιμες λογικές εκφράσεις).

Στη συνέχεια θα αναφερόμαστε στο πρόβλημα SAT αντί για τη γλώσσα L_{SAT}

Π.χ. λέμε ότι "το SAT ανήκει στην κλάση NP" , αντί να πούμε ότι η "γλώσσα L_{SAT} ανήκει στην κλάση NP" .

Πριν αποδείξουμε ότι το πρόβλημα SAT είναι NP πλήρες, θα αποδείξουμε ότι το πρόβλημα SAT ανήκει στην κλάση NP .

ΘΕΩΡΗΜΑ 3.3.2.3: Το SAT ανήκει στην κλάση NP

ΑΠΟΔΕΙΞΗ:

Δίνουμε μία γενική περιγραφή μίας μη ντετερμινιστικής μηχανής Turing με δύο ταινίες, που επιλύει το SAT σε πολυωνυμικό χρόνο.

Η είσοδος x της μηχανής είναι μία ακολουθία από το Σ^*_{SAT} , όπου $\Sigma_{SAT} = \{\upsilon, 0, 1, \neg, \vee, \wedge, (,)\}$.

Η μηχανή αρχικά, μετακινεί το δρομέα της πρώτης ταινίας προς τα δεξιά, ελέγχοντας αν η είσοδος παριστάνει μία λογική έκφραση σε CNF.

Ταυτόχρονα γράφει μη ντετερμινιστικά στη δεύτερη ταινία μία συμβολοσειρά από το $\{\upsilon, 0, 1, \sqcup\}$. Όταν η μηχανή συναντήσει το τέλος της εισόδου επαναφέρει τους δρομείς

στην αρχή κάθε ταινίας. Η παραπάνω διαδικασία απαιτεί χρόνο $O(|x|)$. Το περιεχόμενο της δεύτερης ταινίας ερμηνεύεται από την μηχανή ως αναπαράσταση μίας ανάθεσης αληθοτιμών: μία μεταβλητή είναι αληθής αν η αναπαράσταση της περιέχεται ως υπό-συμβολοακολουθία στην δεύτερη ταινία. Σημειώνεται ότι επειδή το μήκος του περιεχομένου της δεύτερης ταινίας είναι ίσο με το μήκος της εισόδου, είναι αρκετό για να αναπαρασταθεί οποιαδήποτε ανάθεση αληθοτιμών (μπορούν να γραφτούν οι αναπαραστάσεις όλων των μεταβλητών που περιέχονται στην είσοδο και συνεπώς και οποιοδήποτε υποσύνολο τους). Στη συνέχεια η μηχανή ελέγχει αν η λογική έκφραση στη πρώτη ταινία ικανοποιείται από την ανάθεση αληθοτιμών της δεύτερης ταινίας. Τα literals σε κάθε clause εξετάζονται ένα προς ένα, μέχρι να βρεθεί κάποιο που αληθεύει. Ο έλεγχος για το αν ένα literal αληθεύει γίνεται με αναζήτηση της μεταβλητής που αυτό περιέχει στην δεύτερη ταινία. Αν βρεθεί ένα literal που αληθεύει, τότε ολόκληρο το clause αληθεύει και η μηχανή συνεχίζει με το επόμενο clause.Αν ωστόσο όλα τα literals ενός clause είναι ψευδή, τότε το clause και συνεπώς και ολόκληρη η λογική έκφραση είναι ψευδής για την ανάθεση αληθοτιμών που είναι γραμμένη στη δεύτερη ταινία και το συγκεκριμένο υπολογιστικό μονοπάτι της μηχανής τερματίζει στην κατάσταση OXI (no). Αν η μηχανή φτάσει στο τέλος της εισόδου έχοντας ικανοποιήσει όλα τα clauses τότε η λογική έκφραση στην είσοδο είναι ικανοποιήσιμη και η μηχανή τερματίζει στην κατάσταση NAI (yes).

Ο χρόνος που απαιτείται για να διαπιστωθεί αν ένα literal αληθεύει είναι $O(|x|)$, καθώς στη χειρότερη περίπτωση πρέπει να εξεταστεί ολόκληρο το περιεχόμενο της δεύτερης ταινίας.

Ο συνολικός χρόνος για να αποφασιστεί αν η ανάθεση αληθοτιμών ικανοποιεί τη λογική έκφραση είναι $O(|x|^2)$.

• Αν η είσοδος είναι ικανοποιήσιμη, τότε η μηχανή σε κάποια ακολουθία μη ντετερμινιστικών επιλογών, θα γράψει στη δεύτερη ταινία μία ανάθεση αληθοτιμών που την ικανοποιεί.

• Αν αντίθετα η είσοδος είναι μη ικανοποιήσιμη τότε ότι και να γράψει η μηχανή στη δεύτερη ταινία η μηχανή θα διαπιστώσει ότι δεν αποτελεί ανάθεση αληθοτιμών που ικανοποιεί την είσοδο.

Συνεπώς το SAT ανήκει στην κλάση NP **[6]**,**[7]**.

3.3.2.3.1 ΑΠΟΔΕΙΞΗ ΤΟΥ ΘΕΩΡΗΜΑΤΟΣ COOK KAI LEVIN

ΘΕΩΡΗΜΑ COOK (3.3.2.3.1): Το πρόβλημαSAT είναι NP-πλήρες.

ΑΠΟΔΕΙΞΗ:

Επειδή SAT $\in$ NP, αρκεί να δείξουμε ότι το SAT είναι NP-Hard.Έστω μία γλώσσα $L \in NP$. Θα δείξουμε ότι $L \leq_p SAT$. Επειδή $L \in NP$, υπάρχει μία μη ντετερμινιστική μηχανή

Turing $M = (K, \Sigma, \Delta, S)$ με μία ταινία η οποία αποφασίζει την L σε χρόνο $p(|x|) = |x|^k$. Έστω $x \in \Sigma - \{\triangleright, \sqcup\}$, με $|x| = n$. Θα περιγράψουμε μία λογική έκφραση $R(x)$ σε CNF η οποία θα είναι ικανοποιήσιμη αν και μόνο αν το $x \in L$. Το σύνολο των προτασιακών μεταβλητών που περιέχονται στην $R(x)$ είναι $V \cup U \cup W$ όπου:

$$V = \{v_{i,q} | 0 \leq i \leq p(n), q \in K\}$$

$$U = \{u_{i,j} | 0 \leq i, j \leq p(n)\}$$

$$W = \{w_{i,j,\sigma} | 0 \leq i, j \leq p(n), \sigma \in \Sigma\}$$

Οι μεταβλητές συμβολίζουν τις παρακάτω προτάσεις:

$v_{i,q}$: μετά από i βήματα η μηχανή βρίσκεται στην κατάσταση q.
$u_{\mathrm{i,j}}$: μετά από i βήματα ο δρομέας βρίσκεται στην θέση j της ταινίας.
$\mathrm{w_{i,j,\sigma}}$: μετά από i βήματα το σύμβολο στην θέση j της ταινίας είναι το σ.
Παρατηρούμε ότι μετά από $p(n)$ βήματα, η δεξιότερη θέση που μπορεί να βρεθεί ο δρομέας είναι η $p(n)$ (δηλαδή η μηχανή δεν μπορεί να χρησιμοποιεί περισσότερο χώρο από ότι χρόνο). Σε περίπτωση που κάποιες θέσεις της ταινίας παραμένουν απροσπέλαστες θεωρούμε ότι περιέχουν χαρακτήρες $\sqcup$. Αν κάποια χρονική στιγμή $i \leq p(|x|)$ η μηχανή φτάσει σε τελική κατάσταση, θεωρούμε ότι τις επόμενες χρονικές στιγμές παραμένει στην ίδια διαμόρφωση. Για αυτό το σκοπό απαιτείται η χρήση μίας διευρυμένης σχέσης μετάβασης:

$$\Delta' = \Delta \cup \{((q, \sigma), (q, \sigma, -) | q \in \{yes, no, h\}, \sigma \in \Sigma\}$$

Η $R(x)$ θέλουμε να συμβολίζει την πρόταση:

˘ Υπάρχει ένα ορθό υπολογιστικό μονοπάτι της M με είσοδο x, που τερματίζει στην κατάσταση yes'' .

Η παραπάνω πρόταση ισοδυναμεί με τη σύζευξη των παρακάτω τεσσάρων προτάσεων:

•**A :** Κάθε χρονική στιγμή η μηχανή βρίσκεται σε μία ακριβώς κατάσταση, ο δρομέας σε μία ακριβώς θέση και κάθε θέση της ταινίας περιέχει ένα ακριβώς σύμβολο(δηλαδή η μηχανή βρίσκεται σε μία διαμόρφωση).

•**B :** Κάθε χρονική στιγμή η επόμενη διαμόρφωση στο υπολογιστικό μονοπάτι προκύπτει με βάση την σχέση μετάβασης Δ'.

•**C :** Η λειτουργία της μηχανής ξεκινάει από την αρχική της διαμόρφωση.

•**D :** Ο υπολογισμός τερματίζει στην κατάσταση yes.

Στη συνέχεια περιγράφουμε την αναπαράσταση των προτάσεων A,B,C,D ,χρησιμοποιώντας τις προτασιακές μεταβλητές του $V \cup U \cup W$. Η πρόταση A μπορεί να αναπαρασταθεί ως

σύζευξη ενός πλήθους από clauses, τα οποία χωρίζονται σε έξι σύνολα, κάθε ένα από τα οποία εκφράζει μία επί μέρους απαίτηση και όλα μαζί ισοδυναμούν με την Α.
Κάθε χρονική στιγμή η μηχανή βρίσκεται σε μία τουλάχιστον κατάσταση :

$$\{\bigvee_{q\in K} \upsilon_{i,q} | 0 \leq i \leq p(n)\}.$$

Κάθε χρονική στιγμή η μηχανή βρίσκεται το πολύ σε μία κατάσταση:

$$\{\neg\upsilon_{i,q} \vee \neg\upsilon_{i,r} | 0 \leq i \leq p(n), q, r \in K, q \neq r\}.$$

Κάθε χρονική στιγμή ο δρομέας βρίσκεται σε μία τουλάχιστον θέση:

$$\{\bigvee_{j\in K}^{p(n)} u_{i,j} | 0 \leq i \leq p(n)\}.$$

Κάθε χρονική στιγμή ο δρομέας βρίσκεται το πολύ σε μία θέση:

$$\{\neg u_{i,j} \vee \neg u_{i,k} | 0 \leq i \leq p(n), 0 \leq j \leq k \leq p(n)\}.$$

Κάθε χρονική στιγμή σε κάθε θέση της ταινίας υπάρχει τουλάχιστον ένα σύμβολο:

$$\{\bigvee_{\sigma\in\Sigma} w_{i,j,\sigma} : 0 \leq i, j \leq p(n)\}.$$

Κάθε χρονική στιγμή σε κάθε θέση της ταινίας υπάρχει το πολύ ένα σύμβολο:

$$\{\neg \mathrm{w}_{i,j,\sigma} \vee \neg \mathrm{w}_{i,j,\tau} : 0 \leq i, j \leq p(n), \sigma, \tau \in \Sigma, \sigma \neq \tau\}.$$

Η πρόταση Β μπορεί να αναπαρασταθεί ως σύζευξη ενός πλήθους από λογικές εκφράσεις, που μπορεί να χωριστούν σε δύο σύνολα. Οποιαδήποτε χρονική στιγμή η επόμενη κατάσταση, η επόμενη θέση του δρομέα και το σύμβολο που γράφεται στην ταινία, προκύπτουν από την τρέχουσα κατάσταση και το σύμβολο που διαβάζει ο δρομέας, με βάση κάποια από τις επιλογές που επιτρέπει η σχέση μετάβασης.

$$\{(\upsilon_{i,q} \wedge u_{i,j} \wedge w_{i,j,\sigma}) \rightarrow \bigvee_{((q,\sigma),(q',\sigma',\xi'))\in\Delta'} (\upsilon_{i+1,q'} \wedge u_{i+1,j+\gamma(\xi)} \wedge w_{i+1,\sigma'})$$

$0 \leq i, j \prec p(n), q \in K, \sigma \in \Sigma\}$, (όπου $\gamma(\leftarrow) = -1, \gamma(-) = 0, \gamma(\rightarrow) = 1$).
Σε κάθε βήμα το περιεχόμενο της ταινίας μπορεί να αλλάξει μόνο στη θέση που βρίσκεται ο δρομέας.

$$\{u_{i,j} \vee \neg w_{i,j,\sigma} \vee w_{i+1,\sigma} : 0 \leq i \prec p(n), 0 \leq j \leq p(n), \sigma \in \Sigma\}.$$

Για αναπαράσταση της B σε CNF αρκεί κάθε λογική έκφραση στο πρώτο σύνολο να μετατραπεί σε μία ισοδύναμη έκφραση σε CNF.Με αντίστοιχο τρόπο(όπως γίνεται η μετατροπή όταν έχουμε δύο επιλογές) γίνεται η μετατροπή αν το πλήθος των επιλογών είναι $d \geq 2$,οπότε προκύπτουν 3^d clauses, καθένα από τα οποία έχει $d+3$ literals. Η πρόταση C παριστάνεται από τη σύζευξη τετριμμένων clauses, καθένα από τα οποία αποτελείται από ένα μόνο literal:

$$C = v_{0,s} \wedge u_{0,0} \wedge w_{0,0} \rhd \bigwedge_{j=1}^{n} w_{0,j,x[j]} \wedge \bigwedge_{j=n+1}^{p(n)} w_{0,j\sqcup}$$

Τέλος $D = v_{p(n),yes}$.

Από την ανακατασκευή των A,B,C,D προκύπτει ότι η λογική έκφραση $R(x) = A \wedge B \wedge C \wedge D$ είναι ικανοποιήσιμη αν και μόνο αν το $x \in L$. Παρατηρούμε επίσης ότι το μέγεθος της $R(x)$ είναι πολυωνυμικό ως προς το n (το $R(x)$ περιέχει πολυωνυμικό αριθμό από clauses, καθένα από τα οποία περιέχει πολυωνυμικό αριθμό από literals). Η ανακατασκευή της $R(x)$ μπορεί να γίνει από μία ντετερμινιστική μηχανή Turing, σε πολυωνυμικό χρόνο.[**7**]

3.3.3 ΠΡΟΒΛΗΜΑΤΑ ΒΕΛΤΙΣΤΟΠΟΙΗΣΗΣ ΚΑΙ ΑΠΟΦΑΣΗΣ

Πολλά ενδιαφέροντα προβλήματα είναι προβλήματα βελτιστοποίησης **[1],[3]**, για τα οποία κάθε εφικτή λύση έχει μία συνδεδεμένη τιμή, και εμείς προσπαθούμε να βρούμε μία εφικτή λύση με την καλύτερη δυνατή τιμή. Για παράδειγμα, στο πρόβλημα το οποίο ονομάζεται SHORTEST-PATH μας δίνεται ένας μη κατευθυνόμενος γράφος G και κορυφές u και v, και εμείς πρέπει να βρούμε συντομότερο μονοπάτι από την u στην v,δηλαδή το μονοπάτι από το u στο v που περνάει από τις λιγότερες ακμές. Τα ΝΠ-πλήρη προβλήματα απευθύνονται άμεσα όχι σε προβλήματα βελτιστοποίησης ,αλλά σε προβλήματα απόφασης, στα οποία η απάντηση είναι απλά ένα ''ναι'' ή ''οχι'' .

Αν και το γεγονός ότι, αποδεικνύοντας πως ένα πρόβλημα είναι NP-πλήρες, μας περιορίζει στη σφαίρα των προβλημάτων απόφασης, ωστόσο υπάρχει μία βολική σχέση ανάμεσα στα προβλήματα βελτιστοποίησης και τα προβλήματα απόφασης. Όταν μας δίνεται ένα πρόβλημα βελτιστοποίησης τότε ένα τέτοιο πρόβλημα μπορούμε να το συσχετίσουμε με ένα συγγενικό πρόβλημα απόφασης θέτοντας ένα όριο στην τιμή για να βελτιστοποιηθεί. Για παράδειγμα, ας αναφερθούμε στο πρόβλημα που είδαμε και παραπάνω ,το SHORTEST-PATH.Ένα συγγενικό πρόβλημα απόφασης για το πρόβλημα αυτό, μπορεί να είναι το πρόβλημα PATH (μονοπάτι),εάν δοσμένου ενός μη κατευθυνόμενου γράφου G ,με κορυφές u και v και ενός ακεραίου αριθμού k, υπάρχει ένα μονοπάτι από το u στο v αποτελούμενο από το πολύ k ακμές.

Αυτή τη σχέση πού υπάρχει ανάμεσα σε ένα πρόβλημα βελτιστοποίησης και στο συγγενικό του πρόβλημα απόφασης μπορούμε να την χρησιμοποιήσουμε όταν θέλουμε να αποδείξουμε ότι αυτό το πρόβλημα βελτιστοποίησης είναι ΄΄δύσκολο΄΄ πρόβλημα. Αυτό συμβαίνει γιατί τα προβλήματα απόφασης είναι στην πλειοψηφία τους ΄΄έύκολα΄΄ ή το λιγότερο ΄΄οχι δύσκολα προβλήματα΄΄. Συγκεκριμένα ένα παράδειγμα είναι ότι μπορούμε να λύσουμε το πρόβλημα PATH (που είδαμε παραπάνω),λύνοντας το SHORTEST PATH και μετά να συγκρίνουμε τον αριθμό των κορυφών που βρίσκονται στο συντομότερο μονοπάτι με την τιμή της παραμέτρου k του προβλήματος απόφασης .Με άλλα λόγια, εάν ένα πρόβλημα βελτιστοποίησης είναι εύκολο, τότε το συγγενικό του πρόβλημα απόφασης είναι το ίδιο εύκολο. Μπορούμε να δηλώσουμε λοιπόν ότι, εάν μπορούμε να παρέχουμε τα αποδεικτικά στοιχεία ότι ένα πρόβλημα απόφασης είναι ΄΄δύσκολο΄΄, τότε μπορούμε να αποδείξουμε επίσης ότι και το συγγενικό πρόβλημα βελτιστοποίησης είναι ΄΄δύσκολο΄΄.

ΣΥΜΠΕΡΑΣΜΑΤΑ:

Στην αρχή αυτού του κεφαλαίου εισάγαμε τις τρεις κλάσεις την P, NP και NP-πλήρη. Το σημαντικό είναι να καταλάβουμε την διαφορά αυτών των κλάσεων η οποία εστιάζεται στην υπολογιστική πολυπλοκότητα που έχει κάθε πρόβλημα και στους υπολογιστικούς πόρους που απαιτεί για την επίλυσή του γι΄ αυτό και κάθε κλάση εμπεριέχει διαφορετικά προβλήματα. Η κλάση P εμπεριέχει όλα τα προβλήματα που λύνονται σε πολυωνυμικό χρόνο δηλαδή σε αυτή την κλάση ανήκουν τα προβλήματα για τα οποία υπάρχει πολυωνυμικός αλγόριθμος που τα επιλύει ενώ η κλάση NP εμπεριέχει όλα τα προβλήματα που επαληθεύουν την λύση τους σε πολυωνυμικό χρόνο. Αυτό που πρέπει να κατανοήσουμε είναι το ποια είναι η σχέση μεταξύ αυτών των δύο κλάσεων.

Αυτό που συμβαίνει είναι ότι κάθε πρόβλημα στη κλάση P ανήκει επίσης και στην NP όμως το μεγαλύτερο αναπάντητο ερώτημα είναι εάν ισχύει η σχέση Π=ΝΠ, δηλαδή εάν οι κλάσεις προβλημάτων NP και P ταυτίζονται. Αυτό το ερώτημα μετά από πολλά χρόνια έρευνας δεν έχει ακόμα απαντηθεί αυστηρά αν και υπάρχουν ισχυρές ενδείξεις ότι τα δύο σύνολα δεν είναι ίδια και ότι υπάρχουν προβλήματα που ανήκουν στην κλάση NP αλλά όχι

στην P.

Όσον αφορά τώρα τη σχέση μεταξύ των κλάσεων NP και NP-πλήρη ισχύει ότι τα προβλήματα που ανήκουν στο NP είναι «δύσκολα» προβλήματα για τα οποία ξέρουμε μόνο εκθετικούς αλγορίθμους για την επίλυσή τους και όχι πολυωνυμικούς (μιας και πολυωνυμικοί αλγόριθμοι δεν έχουν βρεθεί ακόμα). Έτσι ένα NP-δύσκολο πρόβλημα που ανήκει στο NP είναι NP-πλήρες. Αν μπορούσαμε να αποδείξουμε ότι ένα από τα δύσκολα προβλήματα που ανήκουν στην NP-πλήρη ανήκουν και στην P τότε θα λύναμε το μεγαλύτερο πρόβλημα στην επιστήμη των μαθηματικών, όπως προαναφέραμε, αφού θα δείχναμε ότι $P = NP$.

Στη συνέχεια του κεφαλαίου μιλήσαμε για την πολύ σημαντική έννοια της αναγωγής που σημαίνει ότι μπορούμε να αποδείξουμε ότι κάθε πρόβλημα ανήκει στην κλάση NP-πλήρη αφού όμως αναγάγουμε σε αυτό ένα ήδη γνωστό πρόβλημα που έχει αποδειχθεί ότι ανήκει στην NP-πλήρη κλάση. Έτσι, για παράδειγμα κάνοντας χρήση της γνώσης μας για το πρώτο πρόβλημα A που έχει αποδειχθεί ότι ανήκει στην κλάση NP-πλήρη μπορούμε να αποδείξουμε την NP-πληρότητα ενός δεύτερου προβλήματος B με την αναγωγή του A στο B. Βέβαια είναι κατανοητό ότι για να αρχίσουμε να αποδεικνύουμε ότι υπάρχουν προβλήματα που ανήκουν στην NP-πλήρη κλάση πρέπει να αποδείξουμε ένα πρόβλημα σίγουρα ότι είναι NP-πλήρες ώστε να μπορούμε να το αναγάγουμε σε άλλα προβλήματα για να αποδείξουμε την NP-πληρότητά τους. Το πρώτο λοιπόν πρόβλημα που αποδείχθηκε ότι είναι NP-πλήρες και σε αυτό το κεφάλαιο περιγράφεται η απόδειξή του, μέσω του θεωρήματος του Cook και Levin, είναι το πρόβλημα της ικανοποιησιμότητας. Τέλος αναφέρονται τα προβλήματα βελτιστοποίησης, δηλαδή τα προβλήματα που ζητείται να βρεθεί η βέλτιστη λύση, και τα προβλήματα απόφασης, δηλαδή τα προβλήματα στα οποία η λύση στο ερώτημα είναι μία απάντηση ˝ναι˝ ή ˝οχι˝.

Κεφάλαιο 4

ΠΡΟΒΛΗΜΑΤΑ ΤΗΣ NP-ΠΛΗΡΟΤΗΤΑΣ

Σε αυτό το κεφάλαιο παρατίθενται μερικά σημαντικά προβλήματα της NP-πλήρους κλάσης μαζί με τις αποδείξεις τους ότι ανήκουν σε αυτήν την κλάση.

4.1 ΠΑΡΑΛΛΑΓΕΣ ΤΟΥ SAT

4.1.1 ΤΟ ΠΡΟΒΛΗΜΑ 3SAT

Το πρόβλημα 3SAT [1] : δίνεται μία λογική έκφραση σε διαζευκτική κανονική μορφή, στην οποία κάθε clause έχει ακριβώς 3 literals, και ζητείται να αποφασίσουμε αν η λογική αυτή έκφραση είναι ικανοποιήσιμη.

ΘΕΩΡΗΜΑ 4.1.1: Το πρόβλημα 3-SAT είναι NP-πλήρες

Απόδειξη:

Το πρόβλημα αυτό ανήκει φυσικά στο NP, καθώς είναι ειδική περίπτωση ενός προβλήματος στο NP(του SAT).

Για να αποδείξουμε την πληρότητα θα χρησιμοποιήσουμε τις αναγωγές και θα αναγάγουμε το πρόβλημα SAT στο 3-SAT ,δηλαδή αρκεί να δείξουμε ότι $SAT \leq_p 3SAT$. Αυτό είναι ένα αρκετά κοινό είδος αναγωγής, κατά την οποία ένα πρόβλημα ανάγεται σε μία δική του ειδική περίπτωση. Σε αναγωγές του τύπου αυτού δείχνουμε πως ,αν ξεκινήσουμε από οποιοδήποτε στιγμιότυπο του γενικού προβλήματος, μπορούμε να ξεφορτωθούμε τα χαρακτηριστικά

που εμποδίζουν το στιγμιότυπο αυτό να ανήκει στην ειδική κατηγορία. Στην συγκεκριμένη περίπτωση πρέπει να δείξουμε πως, αν ξεκινήσουμε από οποιοδήποτε σύνολο συνθηκών F, μπορούμε να καταλήξουμε σε πολυωνυμικό χρόνο σε ένα ισοδύναμο σύνολο συνθηκών $\tau(F)$ με τρία το πολύ στοιχεία σε κάθε συνθήκη.

Η αναγωγή είναι απλή. Για κάθε συνθήκη του F με $k > 3$ στοιχεία, έστω

$$C = (\lambda_1 \vee \lambda_2 \vee .... \vee \lambda_k),$$

κάνουμε το εξής: Υποθέτουμε ότι $y_1, ..., y_{k-3}$ είναι νέες μεταβλητές Bool που δεν εμφανίζονται πουθενά αλλού στον τύπο Bool $\tau(F)$, και αντικαθιστούμε τη συνθήκη C με το ακόλουθο σύνολο από συνθήκες :

$$(\lambda_1 \vee \lambda_2 \vee y_1), (\overline{y_1} \vee \lambda_3 \vee y_2), (\overline{y_2} \vee \lambda_4 \vee y_3), ..., (\overline{y_{k-4}} \vee \lambda_{k-2} \vee y_{k-3}), (\overline{y_{k-3}} \vee \lambda_{_{k-1}} \vee \lambda_k)$$

Διασπούμε έτσι όλες τις «μεγάλες» συνθήκες του F, χρησιμοποιώντας διαφορετικό σύνολο μεταβλητών y_i για την καθεμία. Αφήνουμε τις «μικρές» συνθήκες ως έχουν .Ο τύπος Bool που προκύπτει είναι ο $\tau(F)$. Είναι εύκολο να δει κανείς ότι ο τ μπορεί να κατασκευαστεί σε πολυωνυμικό χρόνο. Ισχυριζόμαστε ότι ο $\tau(F)$ είναι ικανοποιήσιμος αν και μόνο αν ο F ήταν ικανοποιήσιμος. Η διαίσθηση λέει το εξής : Φανταστείτε ότι η μεταβλητή $\overline{y_i}$ λέει, «τουλάχιστον ένα από τα στοιχεία $\lambda_{i+2}, ..., \lambda_k$ είναι αληθές», και ότι η συνθήκη $(y_i \vee \lambda_{i+2} \vee y_{i+1})$ λέει, «αν η y_i είναι αληθής ,τότε το λ_{i+2} είναι αληθές είτε η y_{i+1} είναι αληθής».

Αυστηρά έστω ότι η απόδοση τιμών αλήθειας T ικανοποιεί τον $\tau(F)$. Θα δείξουμε ότι η T ικανοποιεί επίσης κάθε συνθήκη του F. Αυτό είναι τετριμμένο για τις μικρές συνθήκες και αν η T απεικονίζει όλα τα k στοιχεία μιας μεγάλης αρχικής συνθήκης σε$\perp$, τότε οι μεταβλητές y_i δεν θα ήταν ικανές από μόνες τους να ικανοποιήσουν όλες τις συνθήκες που προκύπτουν: Η πρώτη συνθήκη θα υποχρέωνε την y_1 να είναι $\top$, η δεύτερη την y_2 να είναι $\top$ και, τέλος, η προτελευταία συνθήκη θα γινόταν η αιτία ώστε η y_{k-3} να πρέπει να είναι $\top$, και καταλήγουμε σε αντίφαση στην τελευταία συνθήκη.

Αντίστροφα, αν υπάρχει απόδοση τιμών αληθείας T που ικανοποιεί τον F, τότε η T μπορεί να επεκταθεί σε μία απόδοση τιμών αληθείας που ικανοποιεί τον $\tau(F)$, ως εξής: Για κάθε μεγάλη συνθήκη $C = (\lambda_1 \vee \lambda_2 \vee .... \vee \lambda_k)$ του F ,έστω j ο μικρότερος δείκτης για τον οποίο $T(\lambda_j) = \top$ (εφόσον έχουμε υποθέσει ότι η T ικανοποιεί τον F,ένα τέτοιο j στοιχείο υπάρχει. Τότε θέτουμε τις τιμές αληθείας των νέων μεταβλητών $y_1, ..., y_{k-3}$ να είναι $T'(y_i) = \top$ αν $i \leq j-2$ και $T'(y_i) = \perp$ διαφορετικά. Είναι εύκολο να δει κανείς ότι η T ικανοποιεί τώρα τον $\tau(F)$, και η απόδειξη της ισοδυναμίας ολοκληρώθηκε.

4.1.2 ΤΟ ΠΡΟΒΛΗΜΑ MAX SAT

Παρακάτω φαίνεται ένα άλλο πρόβλημα το οποίο είναι μία παραλλαγή βελτιστοποίησης της ικανοποιησιμότητας.

ΤΟ ΠΡΟΒΛΗΜΑ MAXSAT [1] : Δεδομένων ενός συνόλου από συνθήκες F και ενός ακεραίου K, υπάρχει απόδοση τιμών αληθείας που ικανοποιεί τουλάχιστον K από τις συνθήκες.

ΘΕΩΡΗΜΑ 4.1.2: Το πρόβλημα MAX SAT είναι NP-πλήρες.

Απόδειξη:

Το ότι ανήκει στο NP είναι προφανές. Θα αναγάγουμε την ΙΚΑΝΟΠΟΙΗΣΙΜΟΤΗΤΑ στο MAX SAT

Η αναγωγή αυτή ,αν και είναι εξαιρετικά απλή ,είναι ενός είδους το οποίο είναι πολύ συνηθισμένο και πολύ χρήσιμο για την απόδειξη αποτελεσμάτων NP-πληρότητας. Θα πρέπει απλώς να παρατηρήσουμε ότι το MAX SAT είναι μία γενίκευση του προβλήματος της ΙΚΑΝΟΠΟΙΗΣΙΜΟΤΗΤΑΣ (SAT) ,δηλαδή κάθε στιγμιότυπο του SAT είναι ένα ειδικό είδος στιγμιότυπου του MAX SAT. Και το εξής ισχύει: Ένα στιγμιότυπο του SAT μπορεί να θεωρηθεί ως ένα στιγμιότυπο του MAX SAT, στο οποίο η παράμετρος K συμβαίνει να είναι ίση με το πλήθος των συνθηκών.

Αυστηρά, η αναγωγή έχει ως εξής: Δεδομένου ενός στιγμιότυπου F της ΙΚΑΝΟΠΟΙΗΣΙΜΟΤΗΤΑΣ με m συνθήκες, κατασκευάζουμε ένα ισοδύναμο στιγμιότυπο (F, m) του MAX SAT αν προσθέσουμε στο F την εύκολη υπολογίσιμη παράμετρο K=m. Προφανώς, υπάρχει απόδοση τιμών αληθείας που ικανοποιεί τουλάχιστον $K = m$ συνθήκες του F (από τις οποίες υπάρχουν ακριβώς m) αν και μόνο αν υπάρχει μία που ικανοποιεί όλες τις συνθήκες του F.

4.1.3 ΤΟ ΠΡΟΒΛΗΜΑ MAX2SAT

Πρόβλημα απόφασης MAX2SAT [6] : δίνεται μία λογική έκφραση ϕ σε συζευκτική κανονική μορφή, στην οποία κάθε clause έχει το πολύ 2 literals και ένας αριθμός K, και ζητείται να αποφασίσουμε αν υπάρχει ανάθεση αληθοτιμών που να ικανοποιεί τουλάχιστον K clauses της ϕ.

ΘΕΩΡΗΜΑ 4.1.3: Το πρόβλημα MAX2SAT είναι NP-πλήρες.

Απόδειξη:

Είναι εύκολο να δειχθεί ότι MAX2SAT $\in$ NP. Θα αποδείξουμε ότι $3SAT \leq_p MAX2SAT$. Έστω ένα στιγμιότυπο του 3SAT δηλαδή μία λογική έκφραση ϕ σε συζευκτική κανονική μορφή, στην οποία κάθε clause έχει ακριβώς 3 literals. Κατασκευάζω την πρόταση ψ που προκύπτει από την ϕ αν αντικαταστήσω κάθε clause $C_i = a_i \vee b_i \vee c_i$ με την παρακάτω πρόταση ψ_i:

$$\psi_i = a_i \wedge b_i \wedge c_i \wedge w_i$$

$$\wedge(\neg a_i \vee \neg b_i) \wedge (\neg b_i \vee \neg c_i) \wedge (\neg c_i \vee \neg a_i)$$

$$\wedge(a_i \vee \neg w_i) \wedge (b_i \vee \neg w_i) \wedge (c_i \vee \neg w_i)$$

Όπου w_i μία νέα μεταβλητή που εμφανίζεται μόνο στην ψ_i.

Τέλος θέτω $R(\phi) = (\psi, 7m)$, όπου m το πλήθος των clauses στην ϕ. Η παραπάνω κατασκευή γίνεται σε πολυωνυμικό χρόνο. Η ερώτηση που προκύπτει είναι πόσα clauses της ψ_i μπορεί να ικανοποιήσει μία ανάθεση αληθοτιμών T·

Αν a_i, b_i, c_i είναι όλα true τότε ικανοποιούνται το πολύ 7 clauses της ψ_i (θέτοντας $T(w_i) = true$). Αν δύο εκ των a_i, b_i, c_i είναι true τότε ικανοποιούνται 7 clauses της ψ_i (ανεξάρτητα από το $T(w_i)$). Αν ένα εκ των a_i, b_i, c_i είναι true τότε ικανοποιούνται το πολύ 7 clauses ψ_i (θέτοντας $T(w_i) = false$). Αν a_i, b_i, c_i είναι όλα false τότε ικανοποιούνται το πολύ 6 clauses της ψ_i (θέτοντας $T(w_i) = false$).

Θα δείξω ότι η ϕ είναι ικανοποιήσιμη αν υπάρχει ανάθεση αληθοτιμών που να ικανοποιεί τουλάχιστον $7m$ clauses της ψ.

Έστω ανάθεση αληθοτιμών T ικανοποιεί την ϕ. Για κάθε clause $C_i = a_i \vee b_i \vee c_i$ η T ικανοποιεί ένα εκ των a_i, b_i, c_i. Συνεπώς, μπορώ να επεκτείνω την T σε T' ορίζοντας το $T'(w_i)$ έτσι ώστε η T' να ικανοποιεί 7 clauses της ψ_i. Η T' ικανοποιεί συνολικά $7m$ clauses της ϕ. Αντίστροφα, έστω ανάθεση αληθοτιμών T' που ικανοποιεί $7m$ clauses της ψ. Συνεπώς η T' ικανοποιεί ακριβώς 7 clauses από κάθε πρόταση ψ_i. Άρα για κάθε clause $C_i = a_i \vee b_i \vee c_i$ η T' ικανοποιεί ένα εκ των a_i, b_i, c_i.

Ο περιορισμός T της T' στις μεταβλητές της ϕ ικανοποιεί την ϕ.

4.1.4 ΤΟ ΠΡΟΒΛΗΜΑ NAESAT

Πρόβλημα απόφασης NAESAT **[6]** : δίνεται μία λογική έκφραση ϕ σε συζευκτική κανονική μορφή, στην οποία κάθε clause έχει ακριβώς 3 literals, και ζητείται να αποφασίσουμε αν η ϕ είναι ικανοποιήσιμη από συμπληρωματικές αναθέσεις αληθοτιμών.

Αν μία λογική έκφραση ϕ βρίσκεται σε CNF, τότε είναι ικανοποιήσιμη από δύο συμπληρωματικές αναθέσεις αληθοτιμών αν και μόνο αν υπάρχει ανάθεση αληθοτιμών για την οποία σε κάθε clause της ϕ τα literals δεν έχουν όλα την ίδια αληθοτιμή. Αυτό δικαιολογεί τα αρχικά 'NAE' (not all equal) στην ονομασία του παραπάνω προβλήματος.

ΘΕΩΡΗΜΑ 4.1.4: Το πρόβλημα NAESAT είναι NP-πλήρες.

Απόδειξη:

Το NAESAT ανήκει στην κλάση NP. Θα δείξουμε ότι $3SAT \leq_p NAESAT$ (θα αναγάγουμε δηλαδή το 3SAT στο NAESAT). Έστω ένα στιγμιότυπο του 3SAT $\phi = \overset{m}{_{i=1}} C_i$ όπου $C_i = a_i \vee b_i \vee c_i$. Θα ανακατασκευάσουμε μία λογική έκφραση $R(\phi) = \overset{m}{_{i=1}} \psi_i$ η οποία θα είναι ικανοποιήσιμη από συμπληρωματικές αναθέσεις αληθοτιμών αν η ϕ είναι ικανοποιήσιμη. Η κατασκευή θα γίνει με διαδοχικούς μετασχηματισμούς της ϕ. Για κάθε i ορίζουμε

$$\psi_i' = ((a_i \vee b_i) \leftrightarrow x_i) \wedge ((x_i \vee c_i) \leftrightarrow y_i) \wedge (y_i \vee z_i) \wedge (y_i \vee \neg z_i)$$

όπου x_i, y_i, z_i νέες μεταβλητές, που εμφανίζονται μόνο στο ψ_i. Αποδεικνύεται εύκολα ότι η λογική έκφραση $R'(\phi) = \overset{m}{_{i=1}} \psi_i'$ είναι ικανοποιήσιμη αν η ϕ είναι ικανοποιήσιμη. Χρησιμοποιώντας απλές ιδιότητες της προτασιακής λογικής μπορούμε να μετασχηματίσουμε την $R'(\phi)$ σε CNF:

$$\psi_i' \equiv ((a_i \vee b_i) \rightarrow x_i) \wedge (x_i \rightarrow (a_i \vee b_i)) \wedge$$

$$((x_i \vee c_i) \rightarrow y_i)) \wedge (y_i \rightarrow (x_i \vee c_i) \wedge (y_i \vee z_i) \wedge (y_i \vee \neg z_i)$$

$$\equiv (\neg(a_i \vee b_i) \vee x_i) \wedge (\neg x_i \vee (a_i \vee b_i)) \wedge$$

$$(\neg(x_i \vee c_i) \vee y_i) \wedge (\neg y_i \vee (x_i \vee c_i)) \wedge (y_i \vee z_i) \wedge (y_i \vee \neg z_i)$$

$$\equiv (\neg(a_i \vee b_i) \vee x_i) \wedge (\neg x_i \vee (a_i \vee b_i)) \wedge$$

$$(\neg(x_i \vee c_i) \vee y_i) \wedge (\neg y_i \vee (x_i \vee c_i)) \wedge (y_i \vee z_i) \wedge (y_i \vee \neg z_i)$$

$$\equiv (\neg(a_i \vee x_i) \wedge (\neg b_i \vee x_i) \wedge (\neg x_i \vee a_i \vee b_i) \wedge$$

$$(\neg x_i \vee y_i) \wedge (\neg c_i \vee y_i) \wedge (\neg y_i \vee x_i \vee c_i) \wedge (y_i \vee z_i) \wedge (y_i \vee \neg z_i)$$

$$= \psi_i''$$

Η λογική έκφραση $R''(\phi) = \overset{m}{_{i=1}} \psi_i'' \equiv \overset{m}{_{i=1}} \psi_i' = R'(\phi)$ είναι ικανοποιήσιμη αν η ϕ είναι ικανοποιήσιμη. Η λογική έκφραση ψ_i προκύπτει αν σε κάθε clause της ψ_i'' που αποτελείται από δύο μόνο literals προσθέτουμε τη μεταβλητή w, η οποία είναι κοινή για όλα τα i:

$$\psi_i = (\neg a_i \vee x_i \vee w) \wedge (\neg b_i \vee x_i \vee w) \wedge (\neg x_i \vee a_i \vee b_i) \wedge$$

$$((\neg x_i \vee y_i \vee w) \wedge (\neg c_i \vee y_i \vee w) \wedge (\neg y_i \vee x_i \vee c_i) \wedge$$

$$(y_i \vee z_i \vee w) \wedge (y_i \vee \neg z_i \vee w)$$

Η παραπάνω κατασκευή της $R(\phi)$ μπορεί να γίνει σε πολυωνυμικό χρόνο ως προς το μέγεθος της ϕ. Αν η ϕ είναι ικανοποιήσιμη, τότε το ίδιο ισχύει και για την $R''(\phi)$.

Έστω T μία ανάθεση αληθοτιμών η οποία ικανοποιεί την $R''(\phi)$. Παρατηρούμε ότι η T δεν μπορεί να ικανοποιεί περισσότερα από δύο literals σε οποιαδήποτε clause της $R''(\phi)$. Για παράδειγμα αν ίσχυε $T(x_i) = false, T(a_i) = true$ και $T(b_i) = true$, τότε το clause $(\neg a_i \vee x_i)$ θα ήταν ψευδές και η T δεν θα ικανοποιούσε την $R''(\phi)$. Αν επεκτείνουμε την T, θέτοντας $T(w) = false$, τότε με την ανάθεση αληθοτιμών T σε κάθε clause της $R(\phi)$ περιέχεται τουλάχιστον ένα αληθές και ένα ψευδές literal. Συνεπώς $T| = \phi$ και $\bar{T}| = \phi$, οπότε η $R(\phi)$ είναι ικανοποιήσιμη από συμπληρωματικές αναθέσεις αληθοτιμών. Αντίστροφα αν υπάρχει ανάθεση αληθοτιμών T' τέτοια ώστε $T'| = \phi$ και $\bar{T}'| = \phi$, τότε η ανάθεση,

$$T = \begin{cases} \mathrm{T}', \alpha\nu \mathrm{T}'(w) = false \\ \overline{\mathrm{T}}' \alpha\lambda\lambda\iota\omega\varsigma \end{cases},$$

ικανοποιεί την $R''(\phi)$, άρα η ϕ είναι ικανοποιήσιμη.

4.2 ΠΡΟΒΛΗΜΑΤΑ ΑΠΟ ΘΕΩΡΙΑ ΓΡΑΦΗΜΑΤΩΝ

4.2.1 ΤΟ ΠΡΟΒΛΗΜΑ ΑΝΕΞΑΡΤΗΤΟ ΣΥΝΟΛΟ (INDEPENDENT SET)

ΑΝΕΞΑΡΤΗΤΟ ΣΥΝΟΛΟ (INDEPENDENT SET [**9**] : Δεδομένων ενός μη κατευθυνόμενου γραφήματος $G = (V, E)$ και ενός ακεραίου $g \geq 2$,υπάρχει υποσύνολο C του V τέτοιο ώστε , για κάθε $u_i, u_j \in C$, να μην υπάρχει ακμή μεταξύ των u_i, u_j.

Στο πρόβλημα ΑΝΕΞΑΡΤΗΤΟ ΣΥΝΟΛΟ μας δίνεται μια γραφική παράσταση και ένας ακέραιος αριθμός g, και ο στόχος είναι να βρούμε g κορυφές που είναι ανεξάρτητες, δηλαδή να μην υπάρχει ακμή μεταξύ δύο κορυφών. Υπάρχουν πολλά άλλα προβλήματα αναζήτησης με γραφικές παραστάσεις όπως το πρόβλημα επικάλυψη με κόμβους.

ΘΕΩΡΗΜΑ 4.2.1: **Το πρόβλημα ΑΝΕΞΑΡΤΗΤΟ ΣΥΝΟΛΟ είναι NP-πλήρες**

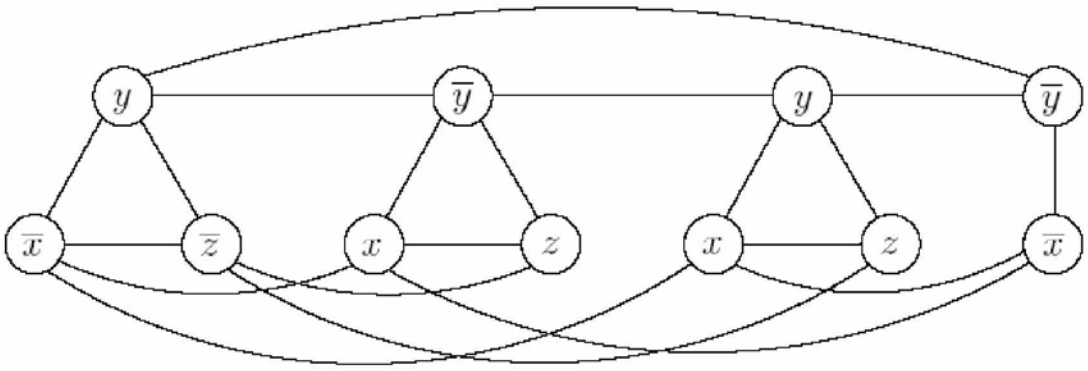
Απόδειξη:

Για να αποδείξουμε την NP πληρότητα , θα αναγάγουμε το πρόβλημα 3SAT στο πρόβλημα ΑΝΕΞΑΡΤΗΤΟ ΣΥΝΟΛΟ. Στο 3SAT η είσοδος είναι ένα σύνολο από προτάσεις, κάθε μία από τις οποίες έχει τρία ή λιγότερα στοιχεία, για παράδειγμα ,

$$(\overline{x} \vee y \vee \overline{z})(x \vee \overline{y} \vee z)(x \vee y \vee z)(\overline{x} \vee \overline{y}),$$

και ο στόχος είναι να βρούμε μία απόδοση τιμών αληθείας που ικανοποιεί την φόρμουλα.

Στο ΑΝΕΞΑΡΤΗΤΟ ΣΥΝΟΛΟ η είσοδος είναι μία γραφική παράσταση και ένα νούμερο g και το πρόβλημα είναι να βρούμε ένα σύνολο από g κορυφές που είναι ανεξάρτητες. Θα πρέπει λοιπόν με κάποιο τρόπο να ταιριάξουμε τους τύπους Boolean με τις γραφικές παραστάσεις.



Σχήμα 4.2.1

Σχήμα 4.2.1:σχήμα της πρότασης $(\overline{x} \vee y \vee \overline{z})(x \vee \overline{y} \vee z)(x \vee y \vee z)(\overline{x} \vee \overline{y})$

Σκεφτείτε. Για να διαμορφώσουμε μια ικανοποιητική απόδοση τιμών αληθείας πρέπει να επιλέξουμε το ένα στοιχείο από κάθε μια πρόταση και να του δώσουμε την τιμή true . Αλλά οι επιλογές μας πρέπει να είναι συνεπείς: εάν επιλέξουμε x σε μία πρόταση, δεν μπορούμε να επιλέξουμε $\overline{x}$ σε άλλη. Οποιαδήποτε συνεπής επιλογή των literals, ένα από κάθε πρόταση, διευκρινίζει μια ανάθεση αλήθειας (οι μεταβλητές για τις οποίες κανένα στοιχείο δεν έχει επιλεχτεί μπορούν να πάρουν οποιαδήποτε τιμή).

Έτσι ας παρουσιάσουμε μία πρόταση, όπως $(x \vee \overline{y} \vee z)$ με ένα τρίγωνο, που έχει ετικέτες κορυφών $x, \overline{y}, z$.Επαναλαμβάνουμε αυτή την κατασκευή για όλες τις προτάσεις. Στην γραφική παράσταση που προκύπτει, ένα ανεξάρτητο σύνολο πρέπει να επιλέξει το πολύ ένα στοιχείο από κάθε ομάδα (πρόταση). Για να οδηγηθούμε ακριβώς στην μια επιλογή από κάθε πρόταση, παίρνουμε τον στόχο g να είναι ο αριθμός των προτάσεων στο παράδειγμά μας, $g = 4$.

Αυτό που πρέπει να κάνουμε τώρα είναι να βρούμε ένα τρόπο που να μας αποτρέψει από την επιλογή αντίθετων στοιχείων (δηλαδή και τα δύο x και $\overline{x}$) στις διαφορετικές προτάσεις. Αλλά αυτό είναι εύκολο: βάζουμε μια ακμή μεταξύ δύο οποιωνδήποτε κορυφών που αντιστοιχούν σε αντίθετα στοιχεία. Η γραφική παράσταση που προκύπτει για το παράδειγμά μας παρουσιάζεται στο παραπάνω σχήμα 4.2.1.

Ανακεφαλαιώνοντας την κατασκευή.Δίνοντας μια περίπτωση I του 3SAT , δημιουργούμε μια περίπτωση $(G; g)$ όπως παρακάτω:

• η γραφική παράσταση G έχει ένα τρίγωνο για κάθε πρόταση (ή ακριβώς μια άκρη, εάν η πρόταση έχει δύο στοιχεία), με κορυφές που ονομάζονται από τα στοιχεία των προτάσεων, και έχουν τις πρόσθετες ακμές τους μεταξύ δύο οποιωνδήποτε κορυφών που αναπαριστούν αντίθετα στοιχεία.

• ο στόχος g τίθεται ως ο αριθμός των προτάσεων.

Σαφώς, αυτή η κατασκευή γίνεται σε πολυωνυμικό χρόνο. Εντούτοις, υπενθυμίζουμε ότι

για μία αναγωγή δεν χρειαζόμαστε μόνο έναν ικανοποιητικό τρόπο για να χαρτογραφηθούν οι περιπτώσεις του πρώτου προβλήματος σε περιπτώσεις του δεύτερου, αλλά και ένα τρόπο να ανακατασκευαστεί μια λύση στην πρώτη περίπτωση από οποιαδήποτε λύση της δεύτερης. Όπως πάντα, υπάρχουν δύο πράγματα να δείξουμε:

• Έχοντας ένα ανεξάρτητο σύνολο ε Σ από g κορυφές στο γράφημα G , είναι δυνατόν να ανακτηθεί ικανοποιητικά μια ικανοποιήσιμη απόδοση τιμών αληθείας στο I.

• Εάν η γραφική παράσταση G δεν έχει κανένα ανεξάρτητο σύνολο μεγέθους g , τότε η φόρμουλα Boolean I είναι μη ικανοποιήσιμη.

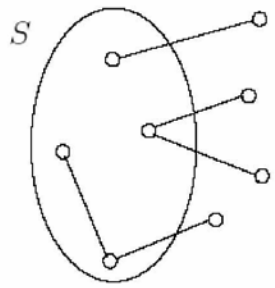
Είναι ευκολότερο να αποδείξουμε το αντίθετο ,δηλαδή , το ότι εάν η I έχει μία ικανοποιήσιμη ανάθεση τότε το G έχει ένα ανεξάρτητο σύνολο μεγέθους g. Για κάθε πρόταση επιλέγουμε οποιοδήποτε στοιχείο με τιμή μικρότερη από αυτή που η ικανοποιήσιμη ανάθεση είναι true (πρέπει να υπάρξει τουλάχιστον ένα τέτοιο στοιχείο) και προσθέτει την ακρίβεια στο S.

4.2.2 ΤΟ ΠΡΟΒΛΗΜΑ ΑΚΡΙΒΗΣ ΕΠΙΚΑΛΥΨΗ (VERTEX COVER)

ΑΚΡΙΒΗΣ ΕΠΙΚΑΛΥΨΗ (VERTEX COVER) **[1][9]**:Στην ακριβή επικάλυψη η είσοδος μας είναι μία γραφική παράσταση και ένας προϋπάρχων υπολογισμός b και η ιδέα είναι να βρούμε b κορυφές που καλύπτουν κάθε ακμή. Δηλαδή πρέπει να χρησιμοποιήσουμε όσο το δυνατόν λιγότερες κορυφές ώστε να καλυφθούν όλες οι ακμές μας. Δηλαδή μία ακριβής επικάλυψη ενός μη κατευθυνόμενου γράφου $G = (V, E)$ είναι ένα υποσύνολο $V' \subseteq V$ έτσι ώστε αν (u, v) είναι μία ακμή του G, τότε ή $u \in V'$ ή $v \in V'$ (ή και τα δύο). Το μέγεθος μιας ακριβούς επικάλυψης είναι ο αριθμός των κορυφών σε αυτή.

ΘΕΩΡΗΜΑ 4.2.2: Το πρόβλημα ΑΚΡΙΒΗΣ ΕΠΙΚΑΛΥΨΗ είναι NP-πλήρες.

Απόδειξη:



Σχήμα 4.2.2

Σχήμα 4.2.2: S είναι μία ακριβής επικάλυψη αν και μόνο αν $V - S$ είναι ένα ανεξάρτητο σύνολο

Μερικές αναγωγές στηρίζονται στην ευστροφία να ταιριάξουν δύο πολύ διαφορετικά προβλήματα. Άλλες καταγράφουν απλά το γεγονός ότι ένα πρόβλημα είναι μια λεπτή παραλλαγή του άλλου. Για να αναγάγουμε το πρόβλημα του ανεξάρτητου συνόλου στο πρόβλημα ακριβούς επικάλυψης πρέπει ακριβώς να παρατηρήσουμε ότι ένα σύνολο κόμβων S είναι μια ακριβής επικάλυψη της γραφικής παράστασης $G = (V, E)$ (δηλαδή τα S καλύπτουν κάθε ακμή στο E) εάν και μόνο εάν οι υπόλοιποι κόμβοι, $V - S$, είναι ένα ανεξάρτητο σύνολο του G (σχήμα 4.2.2).

Επομένως, για να λυθεί μια περίπτωση(G, g) του ανεξάρτητου συνόλου , απλά ψάχνουμε για μία ακριβή επικάλυψη του G με $|V|-g$ κόμβους. Εάν μια τέτοια ακριβής επικάλυψη υπάρχει, τότε παίρνουμε όλους τους κόμβους που δεν είναι μέσα σε αυτή. Εάν καμία τέτοια ακριβής επικάλυψη δεν υπάρχει, τότε το G ενδεχομένως να μην μπορεί να έχει ένα ανεξάρτητο σύνολο μεγέθους g.

Επίσης μπορούμε να αναγάγουμε το πρόβλημα SAT στο πρόβλημα ΑΚΡΙΒΗΣ ΕΠΙΚΑΛΥΨΗ.

Δοσμένου ενός στιγμιότυπου (U, F) του προβλήματος, η υποοικογένεια που αναζητάμε είναι ένα έγκυρο πιστοποιητικό. Το πιστοποιητικό είναι πολυωνυμικά σύντομο ως προς το μέγεθος του στιγμιοτύπου, και μπορεί να ελεγχθεί σε πολυωνυμικό χρόνο εάν πράγματι όλα τα στοιχεία του U εμφανίζονται ακριβώς μία φορά στα σύνολα της C. Τώρα θα γίνει η αναγωγή από το SAT στο πρόβλημα ΑΚΡΙΒΗΣ ΕΠΙΚΑΛΥΨΗ. Δηλαδή, μας δίνεται ένας τύπος Bool F με συνθήκες $\{C_1, ..., C_l\}$ ως προς τις μεταβλητές Bool $x_1, ..., x_n$, και πρέπει να δείξουμε πως κατασκευάζεται σε πολυωνυμικό χρόνο ένα ισοδύναμο στιγμιότυπο $\tau(F)$ του προβλήματος ΑΚΡΙΒΗΣ ΕΠΙΚΑΛΥΨΗ. Θα συμβολίζουμε τα στοιχεία της συνθήκης C_j ως $\lambda_{jk}, k = 1, ..., m_j$ όπου m_j είναι το πλήθος των στοιχείων της C_j.

Το σύμπαν του $\tau(F)$ είναι το σύνολο:

$$U = \{x_i : 1 \leq i \leq n\} \cup \{C_j : j = 1, ..., l\} \cup \{p_{jk} : 1 \leq j \leq l, k = 1, ..., m_j\}.$$

Δηλαδή, υπάρχει ένα στοιχείο για κάθε μεταβλητήBool, ένα για κάθε συνθήκη, και επίσης ένα στοιχείο για κάθε θέση σε κάθε συνθήκη.

Ας δούμε τώρα τα σύνολα της F. Κατ΄ αρχάς, για κάθε στοιχείο p_{jk}, έχουμε στην F ένα σύνολο $\{p_{jk}\}$. Δηλαδή, τα p_{jk} είναι πολύ εύκολο να καλυφθούν. Η δυσκολία έγκειται στο να καλύψουμε τα στοιχεία που αντιστοιχούν στις μεταβλητέςBoolκαι στις συνθήκες. Κάθε μεταβλητή x_i ανήκει σε δύο σύνολα της F, συγκεκριμένα το

$$T_{i,\mathrm{T}} = \{x_i\} \cup \{p_{jk} : \lambda_{jk} = \overline{x_i}\},$$

το οποίο περιέχει επίσης όλες τις αρνητικές εμφανίσεις των x_i , και

$$T_{i,\perp} = \{x_i\} \cup \{p_{jk} : \lambda_{jk} = x_i\},$$

με τις θετικές εμφανίσεις (παρατηρήστε την αντιστροφή στα πρόσημα).

Τέλος, κάθε συνθήκη C_i ανήκει σε m_j σύνολα, ένα για κάθε στοιχείο σε αυτή, συγκεκριμένα $\{C_j, p_{jk}\}, k = 1, ..., m_j$.Αυτά είναι όλα τα σύνολα της F, και έτσι ολοκληρώνεται η περιγραφή του $\tau(F)$.

Ας απεικονίσουμε την αναγωγή για το δεδομένο τύποBool$F$, με συνθήκες $C_1 = (x_1 \vee \overline{x_2})$, $C_2 = (\overline{x_1} \vee x_2 \vee x_3)$, $C_3 = (x_2)$, $C_4 = (\overline{x_2} \vee \overline{x_3})$.

Το σύμπαν του $\tau(F)$ είναι :

$$U = \{x_1, x_2, x_3, C_1, C_2, C_3, C_4, p_{11}, p_{12}, p_{21}, p_{22}, p_{23}, p_{31}, p_{41}, p_{42}\},$$

και η οικογένεια των συνόλων της F αποτελείται από τα εξής σύνολα:

$$\{p_{11}\}, \{p_{12}\}, \{p_{21}\}, \{p_{22}\}, \{p_{23}\}, \{p_{31}\}, \{p_{41}\}, \{p_{42}\},$$

$$T_{1,\perp} = \{x_1, p_{11}\},$$

$$T_{1,\top} = \{x_1, p_{21}\},$$

$$T_{2,\perp} = \{x_2, p_{22}, p_{31}\},$$

$$T_{2,\top} = \{x_2, p_{12}, p_{41}\},$$

$$T_{3,\perp} = \{x_3, p_{23}\},$$

$$T_{3,\top} = \{x_2, p_{42}\},$$

$$\{C_1, p_{11}\}, \{C_1, p_{12}\}, \{C_2, p_{21}\}, \{C_2, p_{22}\}, \{C_2, p_{23}\}, \{C_3, p_{31}\}, \{C_4, p_{41}\}, \{C_4, p_{42}\}$$

Ισχυριζόμαστε ότι το $\tau(F)$ έχει μία ακριβή επικάλυψη αν και μόνο αν ο F είναι ικανοποιήσιμος. Έστω ότι υπάρχει μία ακριβής επικάλυψη C. Εφόσον κάθε x_i πρέπει να καλυφθεί, ακριβώς ένα από τα δύο σύνολα $T_{i,\top}$ και $T_{i,\perp}$ πού περιέχουν τα x_i πρέπει να ανήκει στη C. Φανταστείτε ότι $T_{i,\top} \in C$ σημαίνει $T(x_i) = \top$, και ότι $T_{i,\perp} \in C$ σημαίνει $T(x_i) = \perp$, έτσι ορίζεται μία απόδοση τιμών αληθείας T. Ισχυριζόμαστε ότι η T ικανοποιεί τον F. Για την απόδειξη, θεωρήστε μία συνθήκη C_j. Το στοιχείο του U που αντιστοιχεί στη συνθήκη

αυτή πρέπει να καλύπτεται από ένα σύνολο $\{C_j, p_{jk}\}$ για $1 \leq k \leq m_j$. Αυτό σημαίνει ότι το στοιχείο p_{jk} δεν περιέχεται σε κανένα άλλο σύνολο στην ακριβή επικάλυψη C, συγκεκριμένα, δεν ανήκει στο σύνολο της C που περιέχει τη μεταβλητή που εμφανίζεται (σε άρνηση ή όχι) στο $k-$οστό στοιχείο της C_j. Αυτό όμως σημαίνει ότι η T κάνει το $k-$ οστό στοιχείο της $C_j\top$, και επομένως η C_j ικανοποιείται από την T. Άρα ο F είναι ικανοποιήσιμος.

Αντιστρόφως, υποθέτουμε ότι υπάρχει απόδοση τιμών αληθείας T που ικανοποιεί τον F. Κατασκευάζουμε τότε την εξής ακριβή επικάλυψη C: Για κάθε x_i η C περιέχει το σύνολο $T_{i,\top}$ αν $T(x_i) = \top$, και περιέχει το σύνολο $T_{i,\bot}$ αν $T(x_i) = \bot$. Επίσης, για κάθε συνθήκη C_j, η C περιέχει ένα σύνολο $\{C_j, p_{jk}\}$ τέτοιο ώστε το $k-$οστό στοιχείο της C_j γίνεται $\top$ από την T, και επομένως το p_{jk} δεν περιέχεται σε κανένα σύνολο που έχει επιλεγεί για την C μέχρι στιγμής – γνωρίζουμε ότι ένα τέτοιο k υπάρχει λόγω της υπόθεσης μας ότι η T ικανοποιεί τον F. Τέλος, εφόσον καλύψαμε όλα τα x_i και C_j, η C περιλαμβάνει αρκετά μονομελή σύνολα για να καλύψει οποιαδήποτε εναπομείναντα στοιχεία p_{jk} που δεν έχουν καλυφθεί από τα υπόλοιπα σύνολα. Στην παραπάνω απεικόνιση, η ακριβής επικάλυψη που αντιστοιχεί στην απόδοση τιμών αληθείας που ικανοποιεί τον τύπο $T(x_1) = \top$, $T(x_2) = \top$, $T(x_3) = \bot$ περιέχει τα εξής σύνολα:

$\mathrm{T}_{1,\top}, \mathrm{T}_{2,\top}, \mathrm{T}_{3,\bot}, \{C_1, p_{11}\}, \{C_2, p_{22}\}, \{C_3, p_{31}\}, \{C_4, p_{42}\}$, συν τα μονομελή $\{p_{12}\}, \{p_{21}\}, \{p_{23}\}, \{p_{41}\}$. Καταλήγουμε στο συμπέρασμα ότι το $\tau(F)$ έχει ακριβή επικάλυψη, και η απόδειξη ολοκληρώθηκε.

4.2.3 ΤΟ ΠΡΟΒΛΗΜΑ ΤΗΣ ΚΛΙΚΑΣ (CLIQUE PROBLEM)

ΤΟ ΠΡΟΒΛΗΜΑ ΤΗΣ ΚΛΙΚΑΣ **[6],[4]**: Δοσμένου ενός γράφου και ενός στόχου g, βρείτε ένα σύνολο από g κορυφές έτσι ώστε όλες οι πιθανές ακμές ανάμεσά τους να είναι παρούσες. Δηλαδή το πρόβλημα ΚΛΙΚΑ απαιτεί να είναι παρούσες όλες οι ακμές μεταξύ οποιωνδήποτε δύο κόμβων στο σύνολο.

ΘΕΩΡΗΜΑ 4.2.3: Το πρόβλημα της ΚΛΙΚΑΣ είναι NP-πλήρες.

Απόδειξη:

Ο πολυωνυμικός χρόνος αναγωγής f που χρησιμοποιούμε από το 3SAT στο CLIQUE μετασχηματίζει τις φόρμουλες σε γραφήματα. Στους κατασκευασμένους γράφους, κλίκες με ένα συγκεκριμένο μέγεθος αντιστοιχούν στις ικανοποιήσιμες τιμές της φόρμουλας. Ας υποθέσουμε ότι η φ είναι μία φόρμουλα με k προτάσεις όπως

$$\phi = (a_1 \vee b_1 \vee c_1) \wedge (a_2 \vee b_2 \vee c_3) \wedge .... \wedge (a_k \vee b_k \vee c_k).$$

Η αναγωγή f δημιουργεί την σειρά (G, k) , όπου G είναι ένας μη κατευθυνόμενος γράφος. Οι κόμβοι στο G είναι οργανωμένοι σε k ομάδες από τρεις κόμβους σε κάθε ομάδα και καλείτε τριάδα (τριπλες), $t_1, ..., t_k$. Κάθε τριάδα αντιστοιχεί σε μία από τις προτάσεις στη φόρμουλα ϕ και κάθε κόμβος σε μία τριάδα αντιστοιχεί σε ένα στοιχείο στην αντίστοιχη πρόταση.

Οι ακμές του G συνδέουν όλους τους κόμβους εκτός από τους δύο τύπους ζευγαριών των κόμβων στο G. Καμία ακμή δεν παρευρίσκεται μεταξύ δύο κόμβων στην ίδια τριάδα αλλά και καμία ακμή δεν παρευρίσκεται μεταξύ δύο κόμβων με αντίθετες ετικέτες όπως x_2 και $\overline{x_2}'$.

Τώρα εμείς δείχνουμε γιατί αυτό λειτουργεί. Θα δείξουμε ότι η ϕ είναι ικανοποιήσιμη αν η G έχει μία k-κλίκα.

Υποτίθεται ότι η ϕ έχει μία ικανοποιήσιμη τιμή. Σε αυτή την ικανοποιήσιμη τιμή , τουλάχιστον ένα στοιχείο είναι αληθές σε κάθε πρόταση. Σε κάθε τριάδα του G, επιλέγουμε ένα κόμβο που αντιστοιχίζεται σε ένα αληθές στοιχείο στην ικανοποιήσιμη τιμή. Εάν περισσότερα από ένα στοιχεία είναι αληθή σε μία ειδική πρόταση , επιλέγουμε ένα από τα αληθή στοιχεία αυθαίρετα. Έτσι οι κόμβοι μόλις επιλέχθηκαν από μία k-κλίκα. Ο αριθμός των κόμβων που επιλέχθηκαν είναι k, επειδή διαλέξαμε έναν από κάθε k τριάδα. Κάθε ζευγάρι από τους επιλεγμένους κόμβους ενώνεται από μία ακμή, επειδή κανένα ζευγάρι δεν αποτελεί μία από τις εξαιρέσεις που περιγράψαμε προηγουμένως. Μπορεί να μην είναι από την ίδια τριάδα επειδή εμείς επιλέξαμε μόνο έναν κόμβο από κάθε τριάδα. Δεν μπορούν να έχουν αντίθετες ετικέτες επειδή τα συνταιριαγμένα στοιχεία είναι και τα δύο αληθή στην ικανοποιήσιμη τιμή.

Υποτίθεται ότι ο G έχει μία k-κλίκα. Δύο από τους κόμβους της κλίκας δεν βρίσκονται στην ίδια τριάδα επειδή κόμβοι στην ίδια τριάδα δεν ενώνονται με ακμές. Γι΄ αυτό κάθε μία από τις k τριάδες περιέχει ακριβώς ένα κόμβο από την k-κλίκα. Προσδιορίζουμε την πραγματική αξία των μεταβλητών της ϕ έτσι ώστε για κάθε μαρκαρισμένο στοιχείο ένας κόμβος κλίκας να γίνει αληθής. Το να γίνει αυτό είναι πολύ πιθανό επειδή δύο κόμβοι που μαρκαρίζονται σε μία αντίθετη θέση δεν ενώνονται με μία ακμή και γι΄ αυτό το λόγω και οι δύο δεν μπορούν να είναι στην κλίκα. Αυτή η τιμή στις μεταβλητές ικανοποιεί την ϕ επειδή κάθε τριάδα περιέχει ένα κόμβο κλίκας και γι΄ αυτό κάθε πρόταση περιέχει ένα στοιχείο που είναι αληθές. Γι΄ αυτό η ϕ είναι ικανοποιήσιμη.

4.2.4 ΤΟ ΠΡΟΒΛΗΜΑ ΚΥΚΛΟΣ HAMILTON

Το πρόβλημα του ΚΥΚΛΟΥ HAMILTON , το οποίο μελετήθηκε από έναν άλλο διάσημο μαθηματικό του δεκάτου ενάτου αιώνα, τον William Rowan Hamilton,αλλά και πολλούς άλλους μαθηματικούς μετά από αυτόν, διατυπώνεται ως εξής:
ΚΥΚΛΟΣ HAMILTON [1] : Δεδομένου ενός γραφήματος G, υπάρχει κύκλος που περνάει από κάθε κόμβο του G ακριβώς μία φορά;

Ένας τέτοιος κύκλος ονομάζεται Hamilton, και ένα γράφημα που έχει έναν τέτοιο κύκλο ονομάζεται γράφημα Hamilton . Μετά από ενάμιση αιώνα εξονυχιστικής έρευνας από πολλούς μαθηματικούς, κανένας δεν κατάφερε να ανακαλύψει έναν πολυωνυμικό αλγόριθμο για το πρόβλημα ΚΥΚΛΟΣ HAMILTON. Φυσικά ο παρακάτω αλγόριθμος λύνει το πρόβλημα:

Εξέτασε όλες τις πιθανές μεταβάσεις κόμβων

για κάθε μία έλεγξε εάν είναι κύκλος Hamilton

Δυστυχώς, δεν καταφέρνει να είναι πολυωνυμικός, όπως επίσης και όλοι οι απλοί τρόποι βελτίωσης και επιτάχυνσης του.

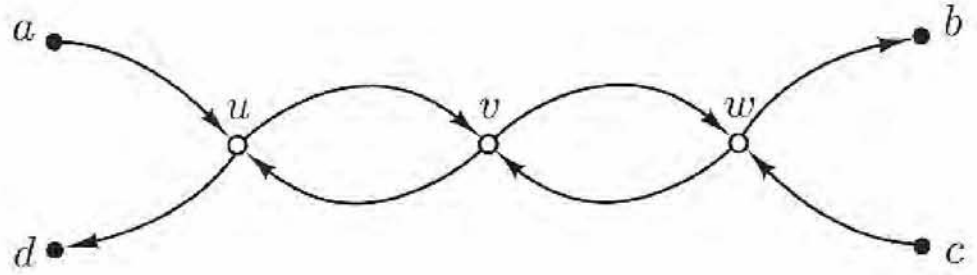
ΘΕΩΡΗΜΑ 4.2.4: Το πρόβλημα ΚΥΚΛΟΣ HAMILTON είναι NP-πλήρες.

Απόδειξη:

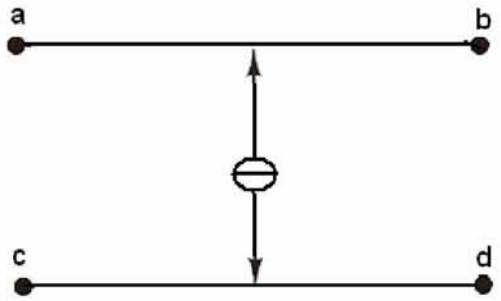
Ήδη γνωρίζουμε ότι είναι NP. Θα δείξουμε τώρα πώς ανάγεται η ΑΚΡΙΒΗΣ ΕΠΙΚΑΛΥΨΗ στον ΚΥΚΛΟ HAMILTON. Θα περιγράψουμε έναν αλγόριθμο πολυωνυμικού χρόνου ο οποίος, δεδομένου ενός στιγμιότυπου (U, F) του προβλήματος ΑΚΡΙΒΗΣ ΕΠΙΚΑΛΥΨΗ, παράγει ένα κατευθυνόμενο γράφημα $\mathrm{G} = \tau(\mathrm{U}, \mathrm{F})$ τέτοιο ώστε το G να έχει κύκλο HAMILTON αν και μόνο αν το (U, F) έχει ακριβή επικάλυψη.

Η κατασκευή βασίζεται σε ένα συγκεκριμένο απλό γράφημα που έχει ενδιαφέρουσες ιδιότητες ταιριαστές με το πρόβλημα ΚΥΚΛΟΣ HAMILTON -στην αργκό της NP-πληρότητας ένα τέτοιο γράφημα ονομάζεται **γκάτζετ**.

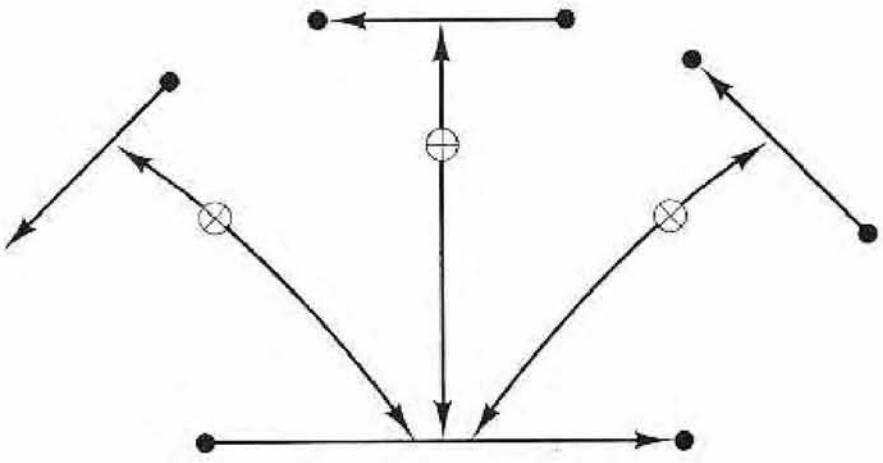
Στο σχήμα 4.2.4 φαίνεται ,ένα γκάτζετ. Μπορούμε να φανταστούμε ότι είναι ένα τμήμα ενός μεγαλύτερου γραφήματος G, το οποίο συνδέεται με το υπόλοιπο γράφημα μέσω των τεσσάρων κόμβων που αναπαριστώνται με συμπαγείς μικρούς κύκλους. Με άλλα λόγια , υπάρχουν και άλλοι κόμβοι και ακμές στο συνολικό γράφημα , αλλά δεν υπάρχουν άλλες ακμές γειτονικές στους τρεις μεσαίους κόμβους εκτός από αυτές που φαίνονται. Επιπλέον έστω ότι το G έχει ένα κύκλο HAMILTON, έναν κύκλο που διασχίζει κάθε κόμβο του G ακριβώς μία φορά. Το ερώτημα είναι, μέσω ποιών ακμών πρόκειται ο κύκλος HAMILTON να διασχίσει τους τρεις μεσαίους κόμβους; Είναι εύκολο να δει κανείς ότι υπάρχουν πραγματικά μόνο δύο δυνατότητες: Είτε οι ακμές (a, u),(u, v), (v, w), (w, b) είτε οι (c, w), (w, v), (v, u), (u, d) αποτελούν τμήμα του κύκλου HAMILTON.Όλες οι υπόλοιπες εναλλακτικές λύσεις αφήνουν εκτός του κύκλου HAMILTON κάποιους από τους τρεις κόμβους , και επομένως ο κύκλος αυτός δεν θα είναι καθόλου HAMILTON.



(a)



(b)



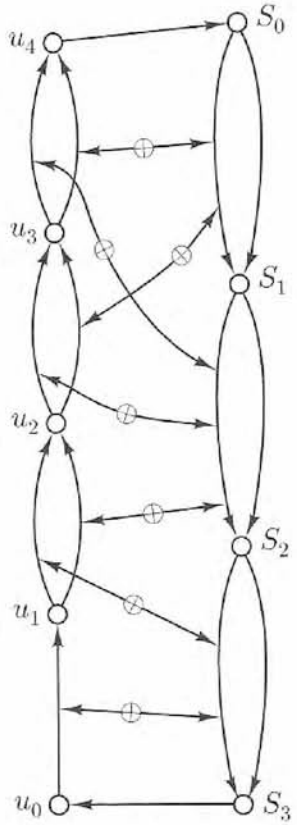
(c)

Σχήμα 4.2.4 (a)-(b)-(c)

Για να το θέσουμε διαφορετικά , ο απλός αυτός μηχανισμός μπορεί να θεωρηθεί ως δύο ακμές (a, b) και (c, d) με τον εξής επιπλέον περιορισμό: Σε οποιονδήποτε κύκλοHAMILTONτου συνολικού γραφήματος G ,διαχωρίζεται είτε η (a, b) είτε η (c, d), αλλά όχι και οι δύο. Η κατάσταση αυτή μπορεί να απεικονιστεί όπως το σχήμα 4.2.4(β), όπου οι δύο αρχικά μη σχετιζόμενες ακμές συνδέονται με ένα σημάδι `άποκλειστικού ή ˇ, το οποίο σημαίνει ότι ακριβώς μία από αυτές μπορεί να συμμετέχει σε οποιοδήποτε κύκλο HAMILTON. Όποτε εμφανίζεται η κατασκευή αυτή στην απεικόνιση ενός γραφήματος ,γνωρίζουμε ότι στην πραγματικότητα το γράφημα περιέχει ολόκληρο το υπογράφημα που φαίνεται στο σχήμα 4.2.4 (c). Το αποτέλεσμα είναι το ίδιο: Είτε διασχίζονται όλες οι ακμές στη μία πλευρά είτε διασχίζεται η ακμή στην άλλη , αλλά δεν μπορούν να συμβαίνουν και τα δύο.

Ο μηχανισμός αυτός έχει κεντρική σημασία στην κατασκευή μας του γραφήματος $\mathrm{G} = \tau(\mathrm{U}, \mathrm{F})$ το οποίο αντιστοιχεί στο στιγμιότυπο του προβλήματος ΑΚΡΙΒΗΣ ΕΠΙΚΑΛΥΨΗ με $U = \{u_1, ..., u_n\}$ και $F = \{S_1, ..., S_m\}$. Περιγράφουμε το γράφημα G στη συνέχεια. Υπάρχουν κόμβοι $u_o, u_1, ..., u_n$ και $S_0, S_1, ..., S_m$, δηλαδή ένας για κάθε στοιχείο του σύμπαντος U , και ένας για κάθε σύνολο στο δεδομένο στιγμιότυπο , συν δύο επιπλέον κόμβοι. Για $i = 1, ..., m$ υπάρχουν δύο ακμές (S_{i-1}, S_i). Φυσικά, στα γραφήματα δεν έχει νόημα να υπάρχουν δύο διαφορετικές ακμές οι οποίες να συνδέουν το ίδιο ζεύγος ακμών. Ο μόνος λόγος που το επιτρέπουνε στην παρούσα περίπτωση είναι ότι οι ακμές θα ενωθούν αργότερα με υπογραφήματα `άποκλειστικού ή ˇ , και επομένως δεν θα υπάρχουν παράλληλες ακμές στο τέλος της κατασκευής. Μία από τις δύο ακμές (S_{i-1}, S_i) ονομάζεται μεγάλη ακμή , και η άλλη μικρή ακμή .Για $i = 1, ..., n$, υπάρχουν μεταξύ των κόμβων u_{j-1} και u_j τόσες ακμές όσα είναι τα σύνολα της F που περιέχουν το στοιχείο u_j. Έτσι, μπορούμε να φανταστούμε ότι κάθε αντίγραφο της ακμής (u_{j-1}, u_j) αντιστοιχεί σε μία εμφάνιση του u_j σε ένα σύνολο. Τέλος, προσθέτουμε ακμές (u_n, S_0) και (u_n, S_0) και συνεπώς ˇκλείνουμε τον κύκλο ˇ.

Παρατηρήστε ότι η κατασκευή μέχρι στιγμής εξαρτάται μόνον από το μέγεθος του σύμπαντος U και το πλήθος και τα μεγέθη των σύνολον της ,και δεν εξαρτάται από το ποια ακριβώς σύνολα περιέχουν το κάθε στοιχείο. Η λεπτή αυτή δομή του στιγμιότυπου θα επηρεάσει την κατασκευή μας ως εξής: Καθώς κάθε αντίγραφο της ακμής (u_{j-1}, u_j) αντιστοιχεί σε μία εμφάνιση του u_j σε κάποιο σύνολο $S_i \in F$ έτσι ώστε $u_j \in S_i$, συνδέουμε με ένα υπογράφημα `άποκλειστικού ή ˇ το αντίγραφο αυτό της ακμής (u_{j-1}, u_j) με τη μεγάλη ακμή (S_{i-1}, S_i). Και έτσι ολοκληρώνεται η κατασκευή του γραφήματος $\tau(U, F)$.



Σχήμα 4.2.4 (d)

Ισχυριζόμαστε ότι το γράφημα $\tau(U, F)$ έχει κύκλο HAMILTON αν και μόνο αν το $\tau(U, F)$ έχει ακριβή επικάλυψη. Αρχικά, υποθέτουμε ότι ο κύκλος HAMILTON υπάρχει. Θα πρέπει να διασχίσει τους κόμβους που αντιστοιχούν στα σύνολα με σειρά $S_0, S_1, ..., S_m$, μετά να διασχίσει την ακμή (S_m, u_0), μετά τους κόμβους $u_o, u_1, ..., u_n$ και τέλος να ολοκληρώσει τη διαδρομή του κατά μήκος της ακμής (u_n, S_0) (σχήμα 4.2.4(d)).Το ερώτημα είναι, για κάθε σύνολο S_j θα διασχίσει τη μικρή ή την μεγάλη ακμή (S_{j-1}, S_j); Έστω C το σύνολο όλων των συνόλων T_j τέτοιο ώστε η μικρή ακμή (S_{j-1}, S_j) να διασχίζεται από τον κύκλο HAMILTON. Θα δείξουμε ότι η C είναι μία νόμιμη επικάλυψη συνόλου ,δηλαδή περιέχει όλα τα στοιχεία χωρίς επικαλύψεις (μεταξύ των υποσυνόλων). Αυτό όμως δεν είναι δύσκολο να το δει κάποιος: Σύμφωνα με την ιδιότητα του υπογραφήματος ˇ *αποκλειστικού ή* ˇ, τα στοιχεία των συνόλων τη ςC είναι ακριβώς τα αντίγραφα των ακμών της μορφής (u_{i-1}, u_i) που διασχίζονται από τον κύκλο HAMILTON και ο κύκλος HAMILTON διασχίζει ακριβώς μία τέτοια ακμή για κάθε στοιχείο $u_j \in U$. Προκύπτει ότι κάθε $u_j \in U$ περιέχεται σε ακριβώς ένα σύνολο της C , και άρα η C είναι ακριβής επικάλυψη.

Αντίστροφα, έστω ότι υπάρχει μία ακριβής επικάλυψη C.Τότε ένας κύκλος HAMILTON στο γράφημα $\tau(U, F)$ μπορεί να κατασκευαστεί ως εξής: Διασχίζουμε τα μικρά αντίγραφα όλων των ακμών (S_{j-1}, S_j) όπου $S_j \in C$, και τις μεγάλες ακμές για όλα τα υπόλοιπα σύνολα. Στη συνέχεια για κάθε στοιχείο u_i διασχίζουμε το αντίγραφο της ακμής (u_{i-1}, u_i) που

αντιστοιχεί στο μοναδικό σύνολο της C που περιέχει το u_i. Συμπληρώνουμε τον κύκλο HAMILTON με τις ακμές (u_n, S_0) και (u_n, S_0).

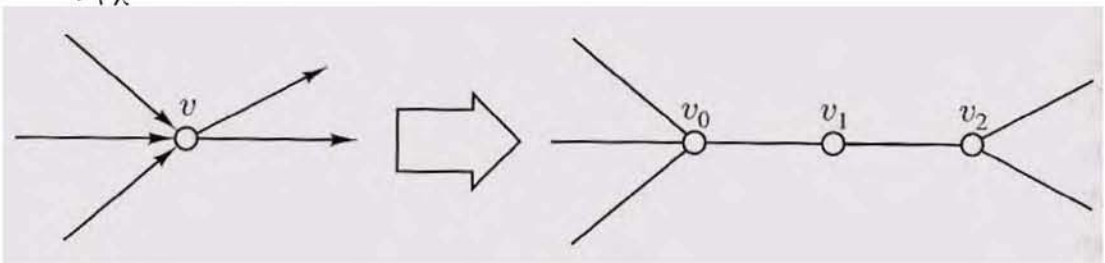
Από τη στιγμή που αποδεικνύεται ότι ένα πρόβλημα είναι NP-πλήρες , η έρευνα συχνά επικεντρώνεται στην επίλυση των ενδιαφερουσών αλλά βατών ειδικών περιπτώσεων του προβλήματος. Οι αποδείξεις NP-πληρότητας συχνά παράγουν στιγμιότυπα του προβλήματος στόχου, τα οποία είναι πολύπλοκα και `άφύσικα΄ .Το ερώτημα συχνά παραμένει, εάν τα στιγμιότυπα που έχουν ενδιαφέρον στην πράξη, και είναι πολύ λιγότερο πολύπλοκα, συνεχίζουν να μην είναι επιλύσιμα από κάποιον αποδοτικό αλγόριθμο. Εναλλακτικά, μπορεί συχνά να αποδειχθεί ότι ακόμα και σημαντικά περιορισμένες παραλλαγές του προβλήματος παραμένουν NP-πλήρη προβλήματα. Έτσι για να εισάγουμε μία ενδιαφέρουσα περίπτωση του ΚΥΚΛΟΥ HAMILTON, ορίζουμε το ΜΗ ΚΑΤΕΥΘΥΝΟΜΕΝΟΣ ΚΥΚΛΟΣ HAMILTON να είναι το πρόβλημα ΚΥΚΛΟΣ HAMILTON περιορισμένο σε γραφήματα τα οποία είναι μη κατευθυνόμενα, δηλαδή, συμμετρικά δίχως αυτομεταβάσεις.

4.2.5 ΤΟ ΠΡΟΒΛΗΜΑ ΜΗ ΚΑΤΕΥΘΥΝΟΜΕΝΟΣ ΚΥΚΛΟΣ HAMILTON

ΘΕΩΡΗΜΑ 4.2.5: Το πρόβλημα ΜΗ ΚΑΤΕΥΘΥΝΟΜΕΝΟΣ ΚΥΚΛΟΣ HAMILTON είναι NP-πλήρες.

Απόδειξη:

Θα αναγάγουμε το συνηθισμένο πρόβλημα ΚΥΚΛΟΣ HAMILTON σε αυτό. Δεδομένου ενός γραφήματος $G \subseteq V \times V$, θα κατασκευάσουμε ένα συμμετρικό γράφημα $G' \subseteq V' \times V'$,δίχως αυτομεταβάσεις, τέτοιο ώστε το G να έχει κύκλο HAMILTON αν και μόνο αν το G' έχει. Η κατασκευή, απεικονίζεται στο σχήμα 4.2.5, είναι η εξής: Κατ΄ αρχάς, $V' = \{v_0, v_1, v_2 : v \in V\}$, δηλαδή, το G' έχει τρεις κόμβους v_0,v_1 και v_2 για κάθε κόμβο v του G. Από αυτούς ο v_0 είναι ανεπίσημα, ο κόμβος εισόδου, στον οποίο θα κατευθυνθούν οι ακμές που εισέρχονται στον v, και v_2 είναι ο κόμβος εξόδου, στον οποίο θα πηγάζουν οι ακμές που εξέρχονται από τον v.



Σχήμα 4.2.5

Έτσι, οι ακμές του G' είναι οι εξής:

$$\{(u_2,u_0),(u_0,u_2):(u,v)\in G\}\cup\{v_0,v_1),(v_1,v_0),(v_1,v_2),(v_2,v_1):v\in V\}.$$

Δηλαδή, οι κόμβοι v_0,v_1,v_2 συνδέονται με ένα μονοπάτι με τη σειρά αυτή, και υπάρχει μη κατευθυνόμενη ακμή μεταξύ των v_2 και v_0 όποτε $(u,v)\in G$. Αυτό ολοκληρώνει την κατασκευή του G'.

Πρέπει τώρα να αποδείξουμε ότι το G' έχει έναν κύκλο HAMILTON αν και μόνο αν το G έχει έναν. Ας υποθέσουμε ότι ένας κύκλος HAMILTON του G' φθάνει στον κόμβο v_0 από μία ακμή της μορφής (u_2,v_0). Αν ο κύκλος HAMILTON αφήνει τον v_0 μέσω μιας ακμής εκτός της (v_0,v_1), τότε δεν μπορεί να ˇ μαζέψει ˇ τον κόμβο v_1 με κανέναν άλλο τρόπο, και έτσι ήταν λάθος η υπόθεση ότι είναι κύκλος HAMILTON .Επομένως η ακμή (v_0,v_1) πρέπει να ανήκει στον κύκλο, και η (v_1,v_2) επίσης. Στη συνέχεια, ο κύκλος πρέπει να συνεχίσει μέσω μίας από τις ακμές (v_2,w_0), όπου $(v,w)\in G$, από εκεί στους w_1,w_2, σε κάποιον z_0 όπου $(w,z)\in G$, κ.ο.κ. Συνεπώς, οι ακμές της μορφής (u_0,v_2) στον κύκλο HAMILTON του G' αποτελούν στην πραγματικότητα έναν κύκλο HAMILTON του G. Αντίστροφα, κάθε κύκλος HAMILTON $(v^1,v^2,....,v^{|V|})$ του G μπορεί να μετατραπεί σε κύκλο HAMILTON του G' ως εξής: $(v_0^1,v_1^1,v_2^1,v_0^2,v_1^2,v_2^2,....,v_0^{|V|},v_1^{|V|},v_2^{|V|})$. Καταλήγουμε στο συμπέρασμα ότι το G έχει κύκλο HAMILTON αν και μόνο αν το G' έχει κύκλο HAMILTON , και η απόδειξη ολοκληρώθηκε[**1**].

Το επόμενο αποτέλεσμά μας αφορά το διάσημο πρόβλημα του ΠΕΡΙΟΔΕΥΟΝΤΟΣ ΠΩΛΗΤΗ.

4.2.6 ΤΟ ΠΡΟΒΛΗΜΑ ΤΟΥ ΠΕΡΙΟΔΕΥΟΝΤΟΣ ΠΩΛΗΤΗ (TRAVELING SALESMAN PROBLEM – ή TSP)

Το πρόβλημα του ΠΕΡΙΟΔΕΥΟΝΤΟΣ ΠΩΛΗΤΗ[**1**], το οποίο εισάγαμε άτυπα και στην εισαγωγή μας, είναι ένα άλλο διατυπωμένο πρόβλημα για το οποίο, παρά τις έντονες ερευνητικές προσπάθειες αρκετών δεκαετιών, κανένας πολυωνυμικός αλγόριθμος δεν είναι γνωστός. Μας δίνεται ένα σύνολο $\{c_1,c_2,....,c_n\}$ από πόλεις, και ένας $n\times n$ πίνακας από μη αρνητικούς ακεραίους d όπου ο d_{ij} συμβολίζει την απόσταση ανάμεσα στην πόλη c_i και στην πόλη c_j. Υποθέτουμε ότι $d_{ii}=0$ και $d_{ij}=d_{ji}$ για όλα τα i και j. Μας ζητείται να βρούμε τη συντομότερη διαδρομή των πόλεων, δηλαδή μία αμφιμονοσήμαντη αντιστοιχία π από το σύνολο $\{1,2,...,n\}$ στον εαυτό του (όπου $\pi(i)$ είναι ,διαισθητικά, η πόλη την οποία επισκεπτόμαστε $i-$οστή στη διαδρομή), τέτοια ώστε η ποσότητα

$$c(\pi) = d_{\pi(1)\pi(2)} + d_{\pi(2)\pi(3)} + ... + d_{\pi(n-1)\pi(n)} + d_{\pi(n)\pi(1)}$$

να είναι όσο το δυνατόν μικρότερη.

ΠΡΟΒΛΗΜΑ ΠΕΡΙΟΔΕΥΟΝΤΟΣ ΠΩΛΗΤΗ : Δεδομένων ενός ακεραίου $n \geq 2$, ενός $n \times n$ πίνακα αποστάσεων d_{ij}, και ενός ακεραίου $B \geq 0$ - διαισθητικά, ο προϋπολογισμός (budget) του περιοδεύοντος πωλητή- βρείτε μία μετάθεση του $\{1, 2, ..., n\}$ τέτοια ώστε $c(\pi) \leq B$.

Θεώρημα 4.2.6 : Το πρόβλημα του ΠΕΡΙΟΔΕΥΟΝΤΟΣ ΠΩΛΗΤΗ (TRAVELING SALESMAN PROBLEM – ή TSP) είναι NP-πλήρες.

Απόδειξη:

Ήδη γνωρίζουμε ότι το πρόβλημα ανήκει στο NP. Για να δείξουμε την πληρότητα, θα αναγάγουμε το πρόβλημα ΜΗ ΚΑΤΕΥΘΥΝΟΜΕΝΟΣ ΚΥΚΛΟΣ HAMILTON στο πρόβλημα του ΠΕΡΙΟΔΕΥΟΝΤΟΣ ΠΩΛΗΤΗ. Δεδομένου ενός συμμετρικού γραφήματος G, όπου χωρίς να εξειδικεύουμε $V = \{v_1, ..., v_{|V|}\}$, κατασκευάζουμε το ακόλουθο στιγμιότυπο του προβλήματος του ΠΕΡΙΟΔΕΥΟΝΤΟΣ ΠΩΛΗΤΗ: το n, το πλήθος των πόλεων, ισούται με $|V|$, και η απόσταση d_{ij} μεταξύ οποιονδήποτε δύο πόλεων i και j είναι

$d_{ij} = 0$ αν $i = j$

$d_{ij} = 1$ αν $(v_i, v_j) \in G$

$d_{ij} = 2$ διαφορετικά

Εφόσον το G είναι ένα συμμετρικό γράφημα δίχως αυτομεταβάσεις, αυτή η συνάρτηση απόστασης είναι και η ίδια συμμετρική, δηλαδή, $d_{ij} = d_{ji}$ για όλες τις πόλεις i και j , όπως απαιτείται. Τέλος, ο προϋπολογισμός είναι $B = n$.

Προφανώς, κάθε˝περιοδεία˝ των πόλεων έχει κόστος ίσο με το πλήθος n συν το πλήθος των μεταξύ των πόλεων αποστάσεων που διανύθηκαν και δεν είναι ακμές του G. Συνεπώς, μία διαδρομή κόστους B ή μικρότερου υπάρχει αν και μόνο αν το πλήθος ˝ μη ακμών˝ που έχουν χρησιμοποιηθεί είναι μηδέν, δηλαδή αν και μόνο αν η διαδρομή είναι ένας κύκλος HAMILTON του G.

4.2.7 ΤΟ ΠΡΟΒΛΗΜΑ ΤΟΥ ΣΑΚΙΔΙΟΥ (KNAPSACK)

ΤΟ ΠΡΟΒΛΗΜΑ ΤΟΥ ΣΑΚΙΔΙΟΥ [1]: Δεδομένων ενός συνόλου $S = \{a_1, ..., a_n\}$ από μη αρνητικούς ακεραίους, και ενός ακεραίου K, όλοι σε δυαδική μορφή, υπάρχει υποσύνολο $P \subseteq S$ τέτοιο ώστε $\sum_{a_i \in P} a_i = K$;

Θεώρημα 4.2.7: Το πρόβλημα του ΣΑΚΙΔΙΟΥ είναι NP-πλήρες.

Απόδειξη:

Το γεγονός ότι το πρόβλημα του ΣΑΚΙΔΙΟΥ ανήκει στο NP είναι προφανές: Δεδομένων ενός στιγμιότυπου του προβλήματος του ΣΑΚΙΔΙΟΥ $a_1, ..., a_n$, και του K, ένα υποσύνολο P του $\{1, ..., n\}$, τέτοιο ώστε $\sum_{i \in P} a_i = K$, μπορεί να χρησιμεύσει ως πιστοποιητικό ότι η απάντηση στο δεδομένο στιγμιότυπο είναι "ναι". Είναι πολυωνυμικά σύντομο και μπορεί να ελεγχθεί σε πολυωνυμικό χρόνο με δυαδική πρόσθεση.

Θα αναγάγουμε τώρα το πρόβλημα ΑΚΡΙΒΗΣ ΕΠΙΚΑΛΥΨΗ στο πρόβλημα του ΣΑΚΙΔΙΟΥ. Μας δίνεται ένα σύμπαν $U = \{u_1, ..., u_n\}$ και μία οικογένεια $F = \{S_1, ..., S_m\}$υποσυνόλων του U. Θα κατασκευάσουμε ένα στιγμιότυπο $\mathsf{T}(U, F)$ του προβλήματος του ΣΑΚΙΔΙΟΥ, δηλαδή μη αρνητικούς ακέραιους $a_1, ..., a_k$, και έναν άλλο K τέτοιον ώστε να υπάρχει υποσύνολο $P \subseteq \{1, ..., k\}$ με $\sum_{i \in P} a_i = K$ αν και μόνο αν υπάρχει ένα σύνολο από σύνολα $C \subseteq F$ που είναι ξένα μεταξύ τους και συλλογικά καλύπτουν όλο το U.

Η κατασκευή αυτή είναι ιδιαίτερα απλή, καθώς βασίζεται σε μία μη αναμενόμενη σχέση ανάμεσα στην *ένωση συνόλων* και στην *πρόσθεση ακεραίων*. Υποσύνολα ενός συνόλου n στοιχείων, όπως εκείνα της F, μπορούν να αναπαρασταθούν ως συμβολοσειρές του $\{0, 1\}^n$ (βλέπε σχήμα 4.2.7).

Τέτοιες συμβολοσειρές μπορούν με τη σειρά τους να αναπαρασταθούν ως ακέραιοι μεταξύ του μηδενός και του $2^n - 1$, σε δυαδική μορφή. Αν πάρουμε τώρα την ένωση τέτοιων συνόλων, υπό την προϋπόθεση ότι είναι ξένα, είναι το ίδιο με το να προσθέσουμε τους αντίστοιχους ακεραίους. Καθώς στην ΑΚΡΙΒΗ ΕΠΙΚΑΛΥΨΗ ρωτάμε εάν από την ένωση των ξένων ομάδων προκύπτει το συνολικό U, αυτό μοιάζει να είναι το ίδιο με το ερώτημα εάν υπάρχουν ακέραιοι ανάμεσα στους δοθέντες οι οποίοι να αθροίζονται σε $K = 1 + 2 + 4 + ... + 2^{n-1}$, ο δυαδικός αριθμός με n άσους. Και αυτό μοιάζει πολύ με στιγμιότυπο του προβλήματος του ΣΑΚΙΔΙΟΥ.

$$S_1 = \{u_3, u_4\} \qquad S_1 = 0011 \qquad 0011$$

$$S_2 = \{u_2, u_3, u_4\} \qquad S_2 = 0111 \qquad 0111$$

$$S_3 = \{u_1, u_2\} \qquad S_3 = 1100 \qquad \frac{+1100}{1111}$$

Σχήμα 4.2.7
Από σύνολα(πρώτη στήλη) σε διανύσματα διφίων (δεύτερη στήλη) σε πρόσθεση ακέραιων με βάση m(τρίτη στήλη).

Η απλή αυτή αναγωγή έχει ένα πρόβλημα. Η στενή αντιστοιχία ανάμεσα στην ένωση συνόλων και στην πρόσθεση ακεραίων μπορεί να έχει κρατούμενο. Θεωρήστε, για παράδειγμα, το άθροισμα 11+13+15+24 = 63 - σε δυαδική μορφή 001011+001101+001111+ 011000 = 111111. Αν το μεταφράσουμε πίσω σε μορφή υποσυνόλων του $\{u_1, ..., u_6\}$, τα σύνολα $\{u_3, u_5, u_6\}$, $\{u_3, u_4, u_6\}$, $\{u_3, u_4, u_5, u_6\}$ και $\{u_2, u_3\}$ δεν είναι ούτε ξένα ούτε καλύπτουν ολόκληρο το U. Με άλλα λόγια, το κρατούμενο καθιστά τη μετάφραση ανάμεσα στην ένωση και στην πρόσθεση προβληματική.

Το πρόβλημα αυτό μπορεί να διορθωθεί ως εξής: Αντί να θεωρήσουμε τις συμβολοσειρές του $\{0,1\}^n$ ως ακεραίους στο m- αδικό σύστημα , όπου m ακεραίους $a_1, ..., a_m$, όπου $a_i = \sum_{u_j \in S_i} m^{j-1}$, ρωτάμε εάν υπάρχει ένα υποσύνολο που αθροίζεται σε $K = a_i = \sum^{n}_{u_j \in S_i} m^{j-1}$.Με τον τρόπο αυτό, το κρατούμενο δεν δημιουργεί πρόβλημα, καθώς από την πρόσθεση λιγότερων από m ψηφίων στο m- αδικό σύστημα, με καθένα από τα ψηφία να είναι είτε 0 είτε 1, δεν μπορεί ποτέ να προκύψει κρατούμενο. Είναι τώρα προφανές ότι το στιγμιότυπο του προβλήματος ΑΚΡΙΒΗΣ ΕΠΙΚΑΛΥΨΗ έχει λύση.

4.2.8 ΤΟ ΠΡΟΒΛΗΜΑ ΕΠΙΚΑΛΥΨΗ ΜΕ ΚΟΜΒΟΥΣ (NODE COVER)

ΕΠΙΚΑΛΥΨΗ ΜΕ ΚΟΜΒΟΥΣ (NODE COVER) [1] : Δεδομένου ενός μη κατευθυνόμενου γραφήματος $G = (V, E)$ και ενός ακεραίου $B \geq 2$, υπάρχει υποσύνολο C του V με $|C|$ τέτοιο ώστε το C να καλύπτει όλες τις ακμές του G.

Θεώρημα 4.2.8:Το πρόβλημα ΕΠΙΚΑΛΥΨΗ ΜΕ ΚΟΜΒΟΥΣ είναι NP-πλήρες.

Απόδειξη:

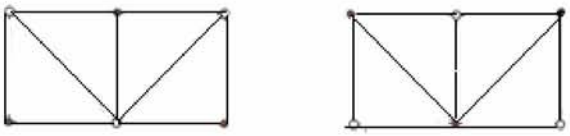
Για να αποδείξουμε ότι το πρόβλημα ΕΠΙΚΑΛΥΨΗ ΜΕ ΚΟΜΒΟΥΣ είναι NP πλήρες θα αναγάγουμε σε αυτό το πρόβλημα του ΑΝΕΞΑΡΤΗΤΟΥ ΣΥΝΟΛΟΥ.

Πριν παραθέσουμε όμως αυτή την απόδειξη μπορούμε να αναφέρουμε μία άλλη ενδιαφέρουσα αναγωγή αυτή του ΑΝΕΞΑΡΤΗΤΟΥ ΣΥΝΟΛΟΥ στο πρόβλημα της ΚΛΙΚΑΣ. Είδαμε προηγουμένως πως ανάγεται το 3SAT στο πρόβλημα της ΚΛΙΚΑΣ. Ωστόσο το πρόβλημα της ΚΛΙΚΑΣ, το οποίο απαιτεί να είναι παρούσες όλες οι ακμές μεταξύ οποιωνδήποτε δύο κόμβων στο σύνολο, είναι υπό κάποια έννοια το ακριβώς αντίθετο του ΑΝΕΞΑΡΤΗΤΟΥ ΣΥΝΟΛΟΥ. Η αναγωγή καθιστά την έννοια αυτή σαφή. Δεδομένου ενός στιγμιότυπου (G, K) του ΑΝΕΞΑΡΤΗΤΟΥ ΣΥΝΟΛΟΥ, όπου $G \subseteq V \times V$ είναι ένα μη κατευθυνόμενο

γράφημα και ο στόχος $K \geq 2$, δημιουργούμε ένα ισοδύναμο στιγμιότυπο (G', K') της ΚΛΙΚΑΣ αν πάρουμε απλώς $G' = V \times V - \{(i,i) : i \in V\} - G$, και διατηρήσουμε τον ίδιο στόχο, $K' = K$. Αυτό δουλεύει διότι, όπως είναι αρκετά εύκολο να ελέγξει κανείς, το μέγιστο ανεξάρτητο σύνολο του G είναι ακριβώς η μέγιστη κλίκα στο συμπλήρωμα του G, το γράφημα το οποίο περιέχει όλες τις ακμές που δεν είναι αυτομεταβάσεις και δεν ανήκουν στο G.

Μετά από αυτή τη μικρή αναφορά είμαστε έτοιμοι να παραθέσουμε την αναγωγή του ΑΝΕΞΑΡΤΗΤΟΥ ΣΥΝΟΛΟΥ στο πρόβλημα ΕΠΙΚΑΛΥΨΗ ΜΕ ΚΟΜΒΟΥΣ για να αποδείξουμε ότι το τελευταίο ανήκει στην NP πληρότητα.

Η ΕΠΙΚΑΛΥΨΗ ΜΕ ΚΟΜΒΟΥΣ είναι και αυτή το ακριβώς αντίθετο του ΑΝΕΞΑΡΤΗΤΟΥ ΣΥΝΟΛΟΥ υπό μία διαφορετική έννοια: εφόσον οι κόμβοι σε μία επικάλυψη με κόμβους $N \subseteq V$ ˇ χτυπάνε ˇ μεταξύ τους όλες τις ακμές, το σύνολο $V - N$ πρέπει να μην έχει καθόλου ακμές μεταξύ των στοιχείων του, και είναι επομένως ένα ανεξάρτητο σύνολο (βλέπε σχήμα 4.2.8). Άρα, το $N \subseteq V$ είναι μία επικάλυψη με κόμβους του G αν και μόνο αν το $V - N$ είναι ανεξάρτητο σύνολο του G. Συνεπώς, το μέγιστο ανεξάρτητο σύνολο του G έχει μέγεθος K ή μεγαλύτερο αν και μόνο αν η ελάχιστη επικάλυψη με κόμβους του G έχει μέγεθος $|V| - K$ ή μικρότερο. Η αναγωγή από το ΑΝΕΞΑΡΤΗΤΟ ΣΥΝΟΛΟ στην ΕΠΙΚΑΛΥΨΗ ΜΕ ΚΟΜΒΟΥΣ αφήνει το γράφημα ως έχει, και απλά αντικαθιστά το K με $|V| - K$.



Σχήμα 4.2.8

4.2.9 ΤΟ ΠΡΟΒΛΗΜΑ ΧΡΩΜΑΤΙΣΜΟΣ ΓΡΑΦΩΝ (3-COLORING

ΧΡΩΜΑΤΙΣΜΟΣ ΓΡΑΦΩΝ **[6]**: Ονομάζουμε *χρωματισμό* ενός γραφήματος $G(V,E)$ μία συνάρτηση $c : V \to C$, τέτοια ώστε $\forall\{u,v\} \in E(c(u) \neq c(v))$, όπου το C είναι ένα οποιοδήποτε σύνολο (π.χ. οι φυσικοί αριθμοί) που το ονομάζουμε σύνολο χρωμάτων. Ονομάζουμε χρωματικό αριθμό $\chi(G)$ ενός γραφήματος $G(V,E)$ το ελάχιστο πλήθος χρωμάτων που απαιτείται για να χρωματίσουμε το G.

Η γενική μορφή του προβλήματος είναι το πρόβλημα απόφασης $k - COLORING$ στο οποίο μας δίνεται ένα γράφημα G και ένας ακέραιος k και ζητιέται να αποφασίσουμε αν $\chi(G) \leq k$. Συγκεκριμένα, στο πρόβλημα απόφασης $3 - COLORING$ μας δίνεται ένα γράφημα G και ζητιέται να αποφασίσουμε αν $\chi(G) \leq 3$.

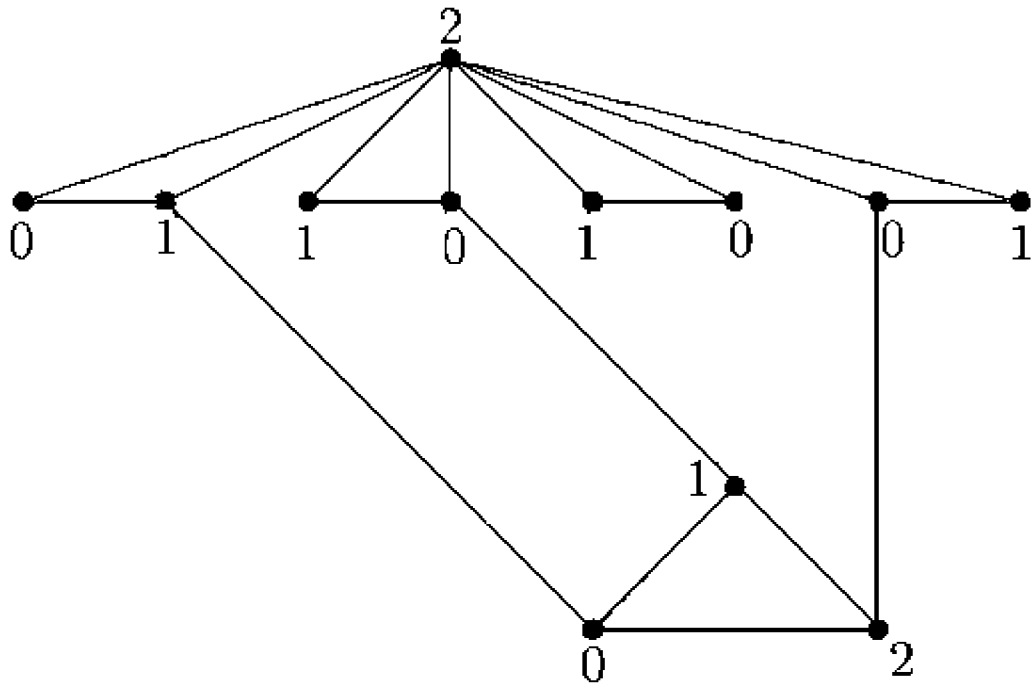
ΘΕΩΡΗΜΑ 4.2.9: Το ΠΡΟΒΛΗΜΑ 3-COLORING είναι NP-πλήρες.

Απόδειξη:

Είναι εύκολο να αποδείξουμε ότι το $3-COLORING$ ανήκει στην κλάση NP. Θα περιγράψουμε μία αναγωγή του προβλήματος NAESAT στο $3-COLORING$ για να αποδείξουμε ότι το $3-COLORING$ ανήκει στην κλάση NP-πλήρη.

Η αναγωγή αυτή είναι απλή. Μας δίνεται ένα σύνολο από προτάσεις $C_1, ..., C_m$ με 3 στοιχεία η κάθε πρόταση, περιλαμβάνοντας τις μεταβλητές $x_1, ..., x_n$, και μας ζητείται να βρούμε αν υπάρχει αληθής ανάθεση τιμών στις μεταβλητές τέτοια ώστε καμία πρόταση να μην έχει όλα τα στοιχεία της **true** (αληθή) ή όλα τα στοιχεία της **false** (ψευδή).

Θα κατασκευάσουμε ένα γράφημα G, και θα συμφωνήσουμε ότι αυτό το γράφημα θα μπορεί να χρωματιστεί με χρώματα $\{0, 1, 2\}$ αν και μόνο αν όλες οι προτάσεις μπορούν να πάρουν διαφορετικές τιμές. Οι τριάδες παίζουν σημαντικό ρόλο και σε αυτή την απόδειξη: Μία τριάδα μας αναγκάζει να χρησιμοποιήσουμε και τα τρία χρώματα σε κάθε κόμβο. Κατά συνέπεια, το γράφημά μας έχει για κάθε μεταβλητή x_i μία τριάδα, $[a, x_i, \neg x_i]$; όλες αυτές οι τριάδες μοιράζονται έναν κόμβο a (αυτός είναι ο κόμβος στην κορυφή που είναι χρωματισμένος με 2, στο παρακάτω σχήμα 4.2.9).



Σχήμα 4.2.9

Σχήμα 4.2.9: η αναγωγή στο $3-COLORING$

Κάθε πρόταση C_i αναπαρίστανται από μία τριάδα $[C_{i1}, C_{i2}, C_{i3}]$ (στο κάτω μέρος στο σχήμα 4.2.9). Τελικά, υπάρχει μία ακμή που συνδέει την C_{ij} με τον κόμβο που αναπαριστά

το j-οστό στοιχείο της C_i. Αυτό είναι που ολοκληρώνει και την κατασκευή το γραφήματος G.

Ισχυριζόμαστε, λοιπόν, ότι το G μπορεί να χρωματιστεί με τα χρώματα $\{0, 1, 2\}$ αν και μόνο αν το στιγμιότυπο που μας δίνεται από το NAESAT είναι ικανοποιήσιμο. Στην απόδειξη, υποτίθεται ότι είναι πράγματι 3-χρωματίσιμο. Μπορούμε να υποθέσουμε, ότι αλλάζοντας το όνομα του χρώματος αν χρειάζεται, αυτός ο κόμβος a παίρνει το χρώμα 2, και έτσι για κάθε i ένας από τους κόμβους x_i και $\neg x_i$ είναι χρωματισμένος με χρώμα 1 και ο άλλος με 0. Εάν ο x_i πάρει το χρώμα 1 τότε η μεταβλητή λέμε ότι είναι **true,** αλλιώς λέμε ότι είναι **false .** .Πως όμως μπορούν να χρωματιστούν τα τρίγωνα των προτάσεων· Εάν όλα τα στοιχεία σε μία πρόταση είναι τρυε, τότε το αντίστοιχο τρίγωνο δεν μπορεί να χρωματιστεί, και αφού το χρώμα 1 δεν μπορεί να χρησιμοποιηθεί, και επομένως ολόκληρο το γράφημα δεν είναι 3-χρωματίσιμο. Ομοίως εάν όλα τα στοιχεία είναι **false** το γράφημα δεν είναι πάλι 3-χρωματίσιμο. Αυτό ολοκληρώνει και την απόδειξη από τη μία κατεύθυνση.

Για την άλλη κατεύθυνση, υποτίθεται ότι μία ικανοποιήσιμη ανάθεση αληθοτιμών υπάρχει. Θα χρωματίσουμε τον κόμβο a με χρώμα 2, Για κάθε πρόταση, μπορούμε να χρωματίσουμε ένα τρίγωνο με προτάσεις ως εξής: Επιλέγουμε δύο στοιχεία σε αυτό με αντίθετες αληθοτιμές (αληθοτιμές υπάρχουν αφού η πρόταση είναι ικανοποιήσιμη) και χρωματίζουμε τις αντίστοιχες, προς αυτές, κορυφές με τα διαθέσιμα χρώματα μεταξύ {0,1} (0 εάν το στοιχείο είναι τρυε και 1 εάν το στοιχείο είναι φαλσε) – μετά χρωματίζουμε τον τρίτο κόμβο με 2.

4.3 ΑΛΛΑ ΠΡΟΒΛΗΜΑΤΑ

4.3.1 ΤΟ ΠΡΟΒΛΗΜΑ ΤΟΥ ΤΑΙΡΙΑΣΜΑΤΟΣ

Ένα από τα γνωστότερα προβλήματα σε γραφήματα είναι το πρόβλημα του ΤΑΙΡΙΑΣΜΑΤΟΣ (matching problem) **[7],[12],[26]**. Το πρόβλημα αυτό απαντάται με αρκετές διαφορετικές μορφές η απλούστερη από τις οποίες είναι το ΔΙΜΕΡΕΣ ΤΑΙΡΙΑΣΜΑ(bipartite matching): Δίνεται ένα διμερές γράφημα(ένα γράφημα στο οποίο το σύνολο των κόμβων του V διαμερίζεται σε δύο σύνολα V_1 και V_2 έτσι ώστε κάθε ακμή του να έχει το ένα άκρο της στο V_1 και το άλλο στο V_2) και ζητείται αν υπάρχει *πλήρες ταίριασμα* ,δηλαδή ένα υποσύνολο των ακμών του, που καλύπτει όλους τους κόμβους και επιπλέον κανένα ζευγάρι ακμών δεν έχει κοινό άκρο. Το πρόβλημα αυτό λύνεται σε πολυωνυμικό χρόνο. Όταν όμως επιχειρήσουμε να δούμε το ανάλογο πρόβλημα στην περίπτωση *τριμερούς γραφήματος* τότε έχουμε ένα NP-πλήρες πρόβλημα το **ΤΡΙΣΔΙΑΣΤΑΤΟ ΤΑΙΡΙΑΣΜΑ (THREE DIMENSIONAL MATCHING ή 3DM)**:

3DM: Δίνονται τρία ξένα σύνολα A, K και Σ με |A| = |K| = |Σ| και υποσύνολο E του Καρτεσιανού γινομένου A×K×Σ. Η ερώτηση ,λοιπόν, είναι αν υπάρχει M ⊆ E με |M| = |A| με όλες τις τριάδες στο M ξένες ανά δύο μεταξύ τους και στις τρεις συντεταγμένες.

ΘΕΩΡΗΜΑ 4.3.1(a) : Το ΤΡΙΣΔΙΑΣΤΑΤΟ ΤΑΙΡΙΑΣΜΑ είναι NP-πλήρες.

Απόδειξη:

Το πρόβλημα** αποδεικνύεται εύκολα ότι ανήκει στο NP. Εμείς θα αποδείξουμε την NP-πληρότητά του. Μαντεύουμε κάποιο ταίριασμα M και επιβεβαιώνουμε σε λογαριθμικό χώρο ότι κάθε τριάδα είναι ξένη με όλες τις άλλες και στις τρεις συντεταγμένες και ότι κάθε στοιχείο των συνόλων A, K και Σ βρίσκεται σε κάποια τριάδα.

Για να δείξουμε την πληρότητα θα ανάγουμε το 3SAT στο 3DM. Έστω ένα στιγμιότυπο του 3SAT ϕ με n μεταβλητές $x_1, x_2, ..., x_n$ και m προτάσεις $C_1, C_2, ..., C_m$. Θα κατασκευάσουμε ένα στιγμιότυπο του 3DM το οποίο θα έχει πλήρες ταίριασμα αν η ϕ είναι ικανοποιήσιμη. Για κάθε μεταβλητή x_i του 3SAT που εμφανίζεται k_i φορές με άρνηση ή χωρίς άρνηση (όποιο είναι το μεγαλύτερο) εισάγουμε τις τριάδες (και βέβαια τα στοιχεία των συνόλων A, K και Σ, που τις αποτελούν):

$$T_i^+ = \{(\overline{a_{ij}}, \kappa_{ij}, \sigma_{ij}), 1 \leq j \leq k_i\}$$

$$T_i^- = \{(a_{ij}, \kappa_{ij+1}, \sigma_{ij}), 1 \leq j \leq k_i\} \cup \{(\overline{a_{ik_i}}, \kappa_{i1}, \sigma_{ik_i})\}$$

Οι μεταβλητές κ_{ij} και σ_{ij} στις παραπάνω τριάδες δεν θα υπάρξουν σε άλλες τριάδες. Κατά συνέπεια και επειδή και αυτές πρέπει να καλυφθούν, θα πρέπει από τα παραπάνω σύνολα να επιλεγούν είτε οι τριάδες με τις $a_{ij}, 1 \leq j \leq k$, είτε οι τριάδες με τις $\overline{a_{ij}}, 1 \leq j \leq k$. Το αποτέλεσμα είναι όλες οι μεταβλητές $\overline{a_{ij}}$ να συμπεριφέρονται σαν αντίγραφα της x_i, ενώ οι a_{ij} σαν αντίγραφα της $\overline{x_i}$. Με άλλα λόγια κάθε ταίριασμα M θα πρέπει να επιλέξει μεταξύ των τριάδων του συνόλου T_i^+ και του T_i^- ή αλλιώς να δώσει στην x_i την τιμή αλήθεια ή ψέμα αντίστοιχα.

Στην συνέχεια για κάθε πρόταση C_j εισάγουμε ένα «κορίτσι» s_j και ένα «σπίτι» t_j και τις τρεις τριάδες:

$\Pi_j = \{(a_{ij}, s_j, t_j)$ όπου $x_i \in C_j\} \cup \{(\overline{a_{ij}}, s_j, t_j)$ όπου $\overline{x_i} \in C_j\}$

Όπως και προηγουμένως, οι s_j και t_j δεν συμμετέχουν αλλού ενώ κάθε μεταβλητή a_{ij} είναι ένα από τα «αντίγραφα» της x_i ανάλογα με το πρόσημο της τελευταίας. Το αποτέλεσμα είναι καταρχήν ότι μόνο μία από τις Π_j μπορεί να συμμετέχει στο ταίριασμα M. Για να γίνει αυτό η τριάδα που θα επιλεγεί θα περιέχει την μεταβλητή a_i που δεν θα επιλεγεί από τις τριάδες T_i^+ ή T_i^-. Αυτό σημαίνει ότι ένα άτομο από τα τρία της C_j καθορίζεται σαν αληθές από το M ή αλλιώς ότι η C_j ικανοποιείται.

Η τελευταία κατηγορία τριάδων έχει σκοπό να ταιριάσει τις επιπλέον μεταβλητές a_{ij} (τα «αγόρια») σε σχέση με τις μεταβλητές των άλλων δύο τύπων τα οποία εισήχθηκαν με τις παραπάνω τριάδες. Για τον σκοπό αυτό εισάγουμε για $1 \leq i \leq n$ και $1 \leq j \leq m$ τις τριάδες:

$$G_{ij} = \{(a_{ij}, g_l, h_l), (\overline{a_{ij}}, g_l, h_l) | 1 \leq l \leq k_i\}.$$

Έτσι αν για παράδειγμα επιλεγούν οι μεταβλητές $\overline{a_{ij}}, 1 \leq l \leq k_i$ στις τριάδες T_i^+ (που σημαίνει ότι η x_i είναι αληθής), οι μεταβλητές a_{ij} των τριάδων T_i^- θα ταιριάσουν κάθε μία με ένα ζευγάρι g_l, h_l από το σύνολο G_{ij}.

Είναι εύκολο να δούμε (σύμφωνα με ότι λέχθηκε για το ρόλο της τριάδας) ότι υπάρχει τρισδιάστατο ταίριασμα αν και μόνο αν το στιγμιότυπο του 3SAT είναι ικανοποιήσιμο.[**7**]

**Ένας ενδιαφέρον τρόπος να ιδωθεί το πρόβλημα είναι τα σύνολα A, K και Σ να παριστάνουν σύνολα αγοριών, κοριτσιών και σπιτιών αντίστοιχα και μία τριάδα $(a, \kappa, \sigma) \in$ E να είναι μία δυνατότητα το αγόρι α να είναι ευτυχισμένο με το κορίτσι κ στο σπίτι σ. Η ερώτηση λοιπόν είναι ισοδύναμη με το αν μπορούν όλα τα αγόρια και όλα τα κορίτσια να είναι ευτυχισμένα σε κάποιο σπίτι σύμφωνα με τις προτιμήσεις τους!

Μια γενίκευση του 3DM που παράλληλα είναι και πιο απλή στη διατύπωση είναι το **ΚΑΛΥΜΜΑ ΑΠΟ ΤΡΙΜΕΡΗ ΣΥΝΟΛΑ (EXACT COVER BY 3-SETS ή πιο σύντομα X3C)[6].** .Η απλούστερη διατύπωση του το κάνει να είναι πολλές φορές ευκολότερη αφετηρία για απόδειξη NP-πληρότητας από ότι το 3DM.

X3C: Δίνεται σύνολο X με $3m$ στοιχεία και σύνολο C τριμερών υποσυνόλων του X. Η ερώτηση ,λοιπόν, είναι αν υπάρχει *ακριβές κάλυμμα* του X στο C, δηλαδή υποσύνολο $C' \subseteq C$ έτσι ώστε κάθε στοιχείο του X να βρίσκεται σε ακριβώς ένα στοιχείο του C'.

ΘΕΩΡΗΜΑ 4.3.1 (b): Το ΠΡΟΒΛΗΜΑ ΚΑΛΥΜΜΑ ΑΠΟ ΤΡΙΜΕΡΗ ΣΥΝΟΛΑ είναι NP-πλήρες.

Απόδειξη:

Εύκολα διαπιστώνουμε ότι το X3C είναι γενίκευση του 3DM όπου το $X = A \cup K \cup \Sigma$ και το σύνολο των τριμελών συνόλων C είναι το σύνολο των τριάδων E στις οποίες αγνοούμε την σειρά. Συνεπώς η NP-πληρότητά του ΚΑΛΥΜΜΑΤΟΣ ΑΠΟ ΤΡΙΜΕΡΗ ΣΥΝΟΛΑ προκύπτει αμέσως από το ΤΡΙΣΔΙΑΣΤΑΤΟ ΤΑΙΡΙΑΣΜΑ.

Τελειώνουμε αυτή την ενότητα με ένα κεντρικό πρόβλημα της Θεωρίας βελτιστοποίησης που είναι η εύρεση του ελαχίστου (ή μεγίστου) μιας γραμμικής συνάρτησης σε ένα $n-$διάστατο χώρο που ορίζεται από ένα αριθμό γραμμικών περιορισμών. Τυπικά ζητάμε το διάνυσμα $x \in R^n$ έτσι ώστε:

$$\min c^T x$$

Όπου τα c, x και b είναι διανύσματα- στήλες με διάσταση n και το A ένας πίνακας $m \times n$. Είναι γνωστό ότι το παρακάτω πρόβλημα βελτιστοποίησης είναι πολυωνυμικά ισοδύναμο με ένα πρόβλημα ύπαρξης μιας λύσης σε τυποποιημένη μορφή:

$$Ax = b$$

$$x \geq 0$$

Το πρόβλημα αυτό είναι γνωστό σαν Γραμμικός Προγραμματισμός και λύνεται σε πολυωνυμικό χρόνο στο μέγεθος της εισόδου (που είναι το σύνολο των ψηφίων του πίνακα A και του διανύσματος b).

Όταν όμως ζητήσαμε μία *ακέραια* λύση για το πρόβλημα αυτό τότε η πολυπλοκότητα του προβλήματος (που τώρα λέγεται **ΑΚΕΡΑΙΟΣ ΓΡΑΜΜΙΚΟΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ ή ILP**) **[27],[28]**, αλλάζει και γίνεται NP-πλήρες.

ΘΕΩΡΗΜΑ 4.3.1 (c) : ο ΑΚΕΡΑΙΟΣ ΓΡΑΜΜΙΚΟΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ (ILP) είναι NP-πλήρες πρόβλημα.

Απόδειξη:

Το πρόβλημα αυτό είναι από τα λίγα προβλήματα που υπάρχουν στο οποίο το δύσκολο μέρος της απόδειξης είναι ότι ανήκει στο NP. Πραγματικά η μέθοδος του «μάντεψε και επαλήθευσε» εδώ δεν μπορεί να χρησιμοποιηθεί άμεσα. Ο λόγος είναι ότι δεν υπάρχει κανένας προφανής περιορισμός για το μέγεθος μιας λύσης x (τον αριθμό των ψηφίων της).Έτσι μια λύση x δεν είναι κατ΄ ανάγκη και «καλό πιστοποιητικό». Για να δειχθεί η ύπαρξη καλού πιστοποιητικού πρέπει να χρησιμοποιηθεί ένα θεώρημα για τον ILP το οποίο λέει ότι αν ένα στιγμιότυπο του ILP έχει λύση, τότε *έχει και λύση με μέγεθος πολυωνυμικά φραγμένο στο μέγεθος της εισόδου*. Το θεώρημα αυτό εξασφαλίζει ότι ο ΑΚΕΡΑΙΟΣ ΓΡΑΜΜΙΚΟΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ είναι στο NP.

Η απόδειξη της πληρότητας είναι πολύ εύκολη επειδή το ILP είναι ιδιαίτερα εκφραστικό πρόβλημα και περίπου κάθε NP-πλήρες πρόβλημα ανάγεται σε αυτό. Για παράδειγμα το 3SAT ανάγεται στο ILP εισάγοντας μια ακέραια μεταβλητή y_i για κάθε λογική x_i με τους περιορισμούς $0 \leq y_i \leq 1$. Είναι προφανές ότι αυτοί επιβάλουν η y_i να παίρνει μόνο τις τιμές 1 και 0. Στην συνέχεια και για κάθε πρόταση C εισάγουμε την ανισότητα $\sum_{x_i \in C} y_i + \sum_{\neg x_i \in C}(1 - y_i) \geq 1$. Αυτή επιβάλει τουλάχιστον ένα άτομο της C να είναι αληθές και άρα η C να ικανοποιείται.

4.3.2 ΤΟ ΠΡΟΒΛΗΜΑ ZOE

Στο πρόβλημα ZOE[**9**] μας δίνεται μια $m \times n$ μήτρα A με 0-1 καταχωρήσεις, και πρέπει να βρούμε ένα 0-1 διάνυσμα $x = (x_1, ..., x_n)$ τέτοιο ώστε οι m εξισώσεις

$$Ax = 1$$

να ικανοποιούνται, όπου με 1 θα δείχνουμε το διάνυσμα στήλης από όλα τα $1's$. Πως μπορούμε να εκφράσουμε το πρόβλημα 3D MATCHING σε αυτό το πλαίσιο;

ΘΕΩΡΗΜΑ 4.3.2:Το ΠΡΟΒΛΗΜΑ ZOE είναι NP-πλήρες.

Απόδειξη:

Για την απόδειξη της NP-πληρότητας αυτού του προβλήματος θα αναγάγουμε το πρόβλημα 3D MATCHING, που είδαμε νωρίτερα, στο ZOE.

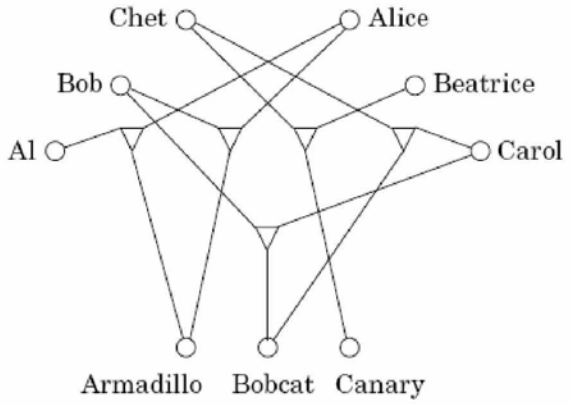
Το ZOE είναι ένα πολύ χρήσιμο πρόβλημα ακριβώς επειδή παρέχει ένα σχήμα με το οποίο πολλά συνδυαστικά προβλήματα μπορούν να εκφραστούν. Σε μια τέτοια διατύπωση σκεφτόμαστε τις 0-1 μεταβλητές ως περιγραφές μιας λύσης, και γράφουμε τις εξισώσεις εκφράζοντας τους περιορισμούς του προβλήματος.

Για παράδειγμα, εδώ φαίνεται πως μπορούμε να εκφράσουμε ένα στιγμιότυπο του 3D MATCHING (m αγόρια, m κορίτσια, m κατοικίδια και n αγόρια-κορίτσια-κατοικίδια τριάδα [boy-girl-pet triples])** στο ZOE. Έχουμε 0-1 μεταβλητές $(x_1, ..., x_n)$, μία τριάδα, όπου x_i=1 σημαίνει ότι η $i-$στη τριάδα επιλέχθηκε από το ταίριασμα, και $x_i = 0$ σημαίνει ότι δεν επιλέχθηκε.

Τώρα το μόνο που έχουμε να κάνουμε είναι να γράψουμε τις εξισώσεις δηλώνοντας ότι η λύση που περιγράφεται από το $x_i's$ είναι ένα νόμιμο ταίριασμα. Για κάθε αγόρι (ή κορίτσι, ή κατοικίδιο) υποθέτουμε ότι οι τριάδες που περιέχουν το αγόρι (ή το κορίτσι, ή το κατοικίδιο) είναι εκείνες που αριθμούνται $j_1, j_2, ..., j_k$ και η κατάλληλη εξίσωση είναι

$$x_{j1} + x_{j2} + ..|.. + x_{jk} = 1$$

που σημαίνει ότι ακριβώς μία από αυτές τις τριάδες πρέπει να περιλαμβάνονται στο ταίριασμα. Για παράδειγμα, αυτή είναι η A μήτρα για ένα στιγμιότυπο του προβλήματος 3D MATCHING που είδαμε νωρίτερα.



$$\mathbf{A} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 \end{pmatrix}$$

Σχήμα 4.3.2

Οι πέντε στήλες της A αντιστοιχίζονται στις πέντε τριάδες, ενώ οι εννέα γραμμές είναι για τους Al, Bob, Chet, Alice, Beatrice, Carol, Armadillo, Bobcat, και Canary, αντίστοιχα.

**Παραπάνω αναφέραμε την μορφή αυτού του προβλήματος με τη χρήση του των αγοριών, των κοριτσιών και των σπιτιών. Αυτή η τεχνική θα χρησιμοποιηθεί στην συνεχεία για την απόδειξη προβλημάτων, όπως και χρησιμοποιείται ευρέως για την απόδειξη παρόμοιων προβλημάτων, με διάφορες παραλλαγές φυσικά(για παράδειγμα, σε αυτό το παράδειγμα δεν χρησιμοποιούμε σπίτια αλλά κατοικίδια-πετς) όπως και διαφορετικούς συμβολισμούς.

4.3.3 ΤΟ ΠΡΟΒΛΗΜΑ ΑΘΡΟΙΣΜΑ ΥΠΟΣΥΝΟΛΩΝ (SUBSET SUM)

ΑΘΡΟΙΣΜΑ ΥΠΟΣΥΝΟΛΩΝ [**9**]: Δίνεται σύνολο S, n θετικών ακεραίων και επιπλέον ακέραιος B. Η ερώτηση που προκύπτει είναι εάν υπάρχει υποσύνολο $S' \subseteq S$ έτσι ώστε το άθροισμα των ακεραίων στο S' να είναι ακριβώς B.

ΘΕΩΡΗΜΑ 4.3.3: Το ΠΡΟΒΛΗΜΑ ΑΘΡΟΙΣΜΑ ΥΠΟΣΥΝΟΛΩΝ είναι NP-πλήρες.

Απόδειξη:

Αυτή είναι μία αναγωγή ανάμεσα σε δύο ειδικές περιπτώσεις του ILP προβλήματος: μία με πολλές εξισώσεις αλλά μόνο 0-1 συντελεστές, και η άλλη με μία μοναδική εξίσωση αλλά με αυθαίρετο αριθμό ακέραιων συντελεστών. Η αναγωγή βασίζεται σε μία απλή ιδέα: 0-1 διανύσματα μπορούν να κωδικοποιήσουν αριθμούς.

Για παράδειγμα, δίνεται το στιγμιότυπο του προβλήματος ZOE :

$$\mathbf{A} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix},$$

Εμείς ψάχνουμε για ένα σύνολο από στήλες από A το οποίο, προστιθέμενο μαζί, να αποτελεί ολόκληρο το $1's$ διάνυσμα. Αλλά αν σκεφτούμε, όμως, τις στήλες σαν δυαδικούς ακέραιους αριθμούς (που διαβάζονται από πάνω προς τα κάτω), τότε ψάχνουμε για ένα υποσύνολο των ακεραίων 18, 5, 4, 8 που προσθέτει μέχρι τον ακέραιο δυαδικό αριθμό $11111_2 = 31$. Και αυτό είναι ένα στιγμιότυπο του προβλήματος ΑΘΡΟΙΣΜΑ ΥΠΟΣΥΝΟΛΩΝ. Έτσι η αναγωγή ολοκληρώθηκε.

Εκτός από μία λεπτομέρεια, αυτή τη λεπτομέρεια που χαλάει συνήθως τη στενή επαφή μεταξύ των 0-1 διανυσμάτων και δυαδικών ακεραίων: *carry*. Εξαιτίας του carry, 5-βιτ δυαδικών ακεραίων μπορούν να προσθέσουν μέχρι το 31(για παράδειγμα, $5 + 6 + 20 = 31$ ή, στο δυαδικό σύστημα, $00101_2 + 00110_2 + 10100_2 = 11111_2$) ακόμα και όταν το άθροισμα από τα αντίστοιχα διανύσματα δεν είναι $(1, 1, 1, 1, 1)$. Αλλά αυτό είναι εύκολο να διορθωθεί: αυτό που κάνουμε είναι να σκεφτούμε τα διανύσματα των στηλών όχι σαν ακέραιους με βάση 2, αλλά σαν ακέραιους με βάση $n + 1$ -ένα παραπάνω από τον αριθμό των στηλών. Με αυτό τον τρόπο, δεδομένου ότι οι περισσότεροι ακέραιοι έχουν προστεθεί, και όλα τα ψηφία τους είναι 0 και1, δεν είναι κανένα carry, και η αναγωγή μας λειτουργεί.

4.3.4 ΤΟ ΠΡΟΒΛΗΜΑ BIN-PACKING

BIN-PACKING[**6**] : Δίνονται n θετικοί αριθμοί $\omega_1, \omega_2, ..., \omega_n$ και δύο αριθμοί C και K και ζητείται να αποφασίσουμε αν οι αριθμοί μπορούν να χωριστούν σε K σύνολα, καθένα από τα οποία περιέχει αριθμούς με άθροισμα το πολύ C.

ω_i: βάρος του αντικειμένου i

C: χωρητικότητα δοχείου

K: διαθέσιμο πλήθος δοχείων

ΘΕΩΡΗΜΑ 4.4.4: Το πρόβλημα BIN-PACKING είναι NP-πλήρες

ΑΠΟΔΕΙΞΗ:

Έστω τα σύνολα $B = \{a_1, a_2, ..., a_n\}$, $K = \{\kappa_1, \kappa_2, ..., \kappa_n\}$, $\Sigma = \{\sigma_1, \sigma_2, ..., \sigma_n\}$ και ένα σύνολο $E = \{t_1, t_2, ..., t_m\} \subseteq B \times K \times \Sigma$. Σχηματίζουμε ένα στιγμιότυπο του BIN-PACKING το οποίο περιέχει $4m$ αντικείμενα, ένα για κάθε τριάδα t_i και ένα για κάθε εμφάνιση στοιχείου a_j, κ_k ή σ_l σε τριάδα. Είναι πολύ βολικό να παραστήσουμε τα βάρη των αντικειμένων στο A- δικό αριθμητικό σύστημα, όπου $A = \max\{2n + 2, 41\}$. Συμβολίζουμε με $(x_4, x_3, x_2, x_1, x_0)$ τον αριθμό $x_4A^4 + x_3A^3 + x_2A^2 + x_1A + x_0$.

• Τα αντικείμενα που αντιστοιχούν στο a_j έχουν όλα το ίδιο βάρος $(9, 0, 0, 2n + 1 - j, 1)$, εκτός από ένα βαρύτερο αντικείμενο $o(a_j)$ με βάρος $(10, 0, 0, 2n + 1 - j, 1)$.

• Τα αντικείμενα που αντιστοιχούν στο κ_k έχουν όλα το ίδιο βάρος $(9, 0, 0, 2n + 1 - \kappa, 0, 2)$, εκτός από ένα βαρύτερο αντικείμενο $o(\kappa_k)$ με βάρος $(10, 0, 0, 2n + 1 - \kappa, 0, 2)$.

• Τα αντικείμενα που αντιστοιχούν στο σ_l έχουν όλα το ίδιο βάρος $(12, 2n + 1 - l, 0, 0, 4)$, εκτός από ένα βαρύτερο αντικείμενο $o(\sigma_l)$ με βάρος $(10, 2n + 1 - l, 0, 0, 4)$.

• Τέλος το μοναδικό αντικείμενο που αντιστοιχεί στην τριάδα $t = (a_j, \kappa_k, \sigma_l)$ έχει βάρος $(10, l, k, i, 8)$.

Επιλέγουμε $C = (40, 2n + 1, 2n + 1, 2n + 1, 15), K = m$. Παρατηρούμε ότι το άθροισμα των βαρών όλων των αντικειμένων είναι mC δηλαδή όσο η συνολική χωρητικότητα των δοχείων. Έστω ότι το E έχει ταίριασμα M.

Τοποθετούμε τα αντικείμενα στα δοχεία με τον παρακάτω τρόπο:

• Για κάθε τριάδα $t = (a_j, \kappa_k, \sigma_l)$ που ανήκει στο M τοποθετούμε σε ένα δοχείο το αντικείμενο που αντιστοιχεί στην t μαζί με τα $o(a_j)$,$o(\kappa_k)$,$o(\sigma_l)$.

• Για κάθε τριάδα $t = (a_j, \kappa_k, \sigma_l)$ που δεν ανήκει στο M τοποθετούμε σε ένα δοχείο το αντικείμενο που αντιστοιχεί στην t μαζί με τρία ακόμη αντικείμενα που αντιστοιχούν στα a_j, κ_k, σ_l και δεν έχουν τοποθετηθεί ήδη σε δοχείο.

Παρατηρούμε ότι κάθε αντικείμενο $o(x), x \in B \cup K \cup \Sigma$, χρησιμοποιείται ακριβώς μία φορά, καθώς το M είναι ταίριασμα. Το άθροισμα των βαρών για τα αντικείμενα ενός δοχείου είναι C σε κάθε περίπτωση. Συνεπώς αν το E έχει ταίριασμα, τότε τα $4m$ αντικείμενα μπορούν να τοποθετηθούν σε $4m$ δοχεία χωρητικότητας C.

Αντίστροφα έστω ότι τα $4m$ αντικείμενα ,μπορούν να τοποθετηθούν σε $4m$ δοχεία χωρητικότητας C. Επειδή η συνολική χωρητικότητα των δοχείων ισούται με το συνολικό βάρος των αντικειμένων θα πρέπει κάθε δοχείο να είναι εντελώς γεμάτο. Συνεπώς κάθε δοχείο θα πρέπει να περιέχει ακριβώς 4 αντικείμενα, καθώς 3 αντικείμενα δεν μπορούν να γεμίσουν ένα δοχείο, ενώ 5 αντικείμενα δεν χωράνε.
Η άθροιση των τεσσάρων αριθμών στο A- δικό σύστημα φαίνεται παρακάτω:

X_4	X_3	X_2	X_1	X_0
Y_4	Y_3	Y_2	Y_1	Y_0
Z_4	Z_3	Z_2	Z_1	Z_0
W_4	W_3	W_2	W_1	W_0
+40	$(2n+1)$	$(2n+1)$	$(2n+1)$	15

Οι τιμές 1,2,4,8 θα πρέπει να εμφανίζονται ακριβώς μία φορά στα ψηφία X_0,Y_0,Z_0,W_0 ώστε να σχηματιστεί το 15. Συνεπώς θα πρέπει τα τέσσερα αντικείμενα ενός δοχείου να είναι διαφορετικού τύπου δηλαδή να αντιστοιχούν στα $t = (a_{j'}, \kappa_{k'}, \sigma_{l'})$,$a_j, \kappa_k, \sigma_l$.

Άρα η άθροιση των τεσσάρων αριθμών στο A- δικό σύστημα γράφεται:

X_4	0	0	$(2n+1-j)$	1
Y_4	0	$(2n+1-k)$	0	2
Z_4	$(2n+1-l)$	0	0	4
W_4	l'	k'	j'	8
+40	$(2n+1)$	$(2n+1)$	$(2n+1)$	15

Για να είναι ορθό το αποτέλεσμα στην παραπάνω πρόσθεση θα πρέπει $j = j'$, $k = k'$ και $l = l'$. Συνεπώς κάθε δοχείο περιέχει ένα αντικείμενο που αντιστοιχεί σε μία τριάδα, μαζί με αντικείμενα που αντιστοιχούν στα μέλη της τριάδας. Επιπλέον το δοχείο που περιέχει το αντικείμενο που αντιστοιχεί στην $t = (a_j, \kappa_k, \sigma_l)$, περιέχει κανένα ή τρία αντικείμενα από τα $o(a_j)$,$o(\kappa_k)$,$o(\sigma_l)$ (αλλιώς το συνολικό βάρος δεν μπορεί να είναι C). Ένα ταίριασμα M του E αποτελείται από όλες τις τριάδες $t = (a_j, \kappa_k, \sigma_l)$, για τις οποίες τα $o(a_j)o(\kappa_k)$,$o(\sigma_l)$ βρίσκονται στο ίδιο δοχείο.
Άρα TRIPARTITE $-$ MATCHING $\leq$ BIN $-$ PACKING

ΣΥΜΠΕΡΑΣΜΑΤΑ: Σε αυτό το κεφάλαιο αναφέρονται πολλά προβλήματα που ανήκουν στην NP-πλήρη κλάση και περιγράφεται η απόδειξή τους. Είναι σημαντικό να κατανοήσουμε ότι κάθε πρόβλημα από αυτά που εκθέτονται παραπάνω αποδεικνύεται ότι ανήκει στην NP-πλήρη κλάση με την αναγωγή σ' αυτό κάποιου προβλήματος που έχουμε αποδείξει πιο πρώτα ότι ανήκει στην κλάση αυτή. Με αυτό τον τρόπο παραθέτουμε μία πληθώρα από τέτοια προβλήματα της κλάσης αυτής.

Κεφάλαιο 5

ΑΣΚΗΣΕΙΣ ΣΤΗ NP- ΠΛΗΡΗ ΚΛΑΣΗ

5.1 ΑΣΚΗΣΗ ΓΙΑ ΤΗΝ P=NP

Η εκφώνηση της παρακάτω άσκησης είναι από το βιβλίο του Sipser και είναι η άσκηση 7.17 την οποία και λύνουμε παρακάτω.

Εκφώνηση: Δείξτε ότι αν ίσχυε $P = NP$ (δηλαδή αυτές οι δυο κλάσεις ταυτίζονταν) τότε κάθε γλώσσα $A \in P$ εκτός της $A = 0$ και της $A = \Sigma^*$ θα ήταν NP-Complete.

Λύση: Αν μια γλώσσα $A \in P$ και ισχύει $P = NP$ τότε προφανώς ισχύει ότι $A \in NP$. Αρκεί τώρα να δείξουμε ότι το A είναι NP-HARD.

Θα δείξουμε ότι κάθε $B \in NP$ και κατά συνέπεια κάθε $B \in P$ ανάγεται σε πολυωνυμικό χρόνο στο $A(B \leq_p A)$. Αφού το $B \in P$ σημαίνει πως υπάρχει αποφασιστής M_B που αποφασίζει το B σε πολυωνυμικό χρόνο.

Η αναγωγή του B στο A έχει ως εξής:
f: ΄΄με είσοδο w:
1. Τρέξε την M_B με είσοδο w:
(a) Αν η M_B δεχτεί, επέστρεψε ένα w_1 τέτοιο ώστε $w_1 \in A$.
(b) Αν η M_B απορρίψει, επέστρεψε ένα w_2 τέτοιο ώστε $w_2 \in A$.΄΄
Έτσι λοιπόν έχουμε:
$\forall w \in B \Leftrightarrow f(w) = w_1 \in A$
$\forall w \in B \Leftrightarrow f(w) = w_2 \notin A$
Να σημειώσουμε ότι η γλώσσα A πρέπει να έχει οπωσδήποτε τόσο ένα $w_1 \in A$ όσο και ένα $w_2 \notin A$. Γι΄ αυτό και αποκλείονται οι γλώσσες $A = 0$ και $A = \Sigma^*$. Να σημειώσουμε πάλι ότι τις συμβολοσειρές w_1 και w_2 δεν χρειάζεται να τις εντοπίσει ο αλγόριθμός μας. Αρκεί να υπάρχουν αυτές.Αυτό το κάνουμε γιατί σκοπός μας δεν είναι να σχεδιάσουμε στην

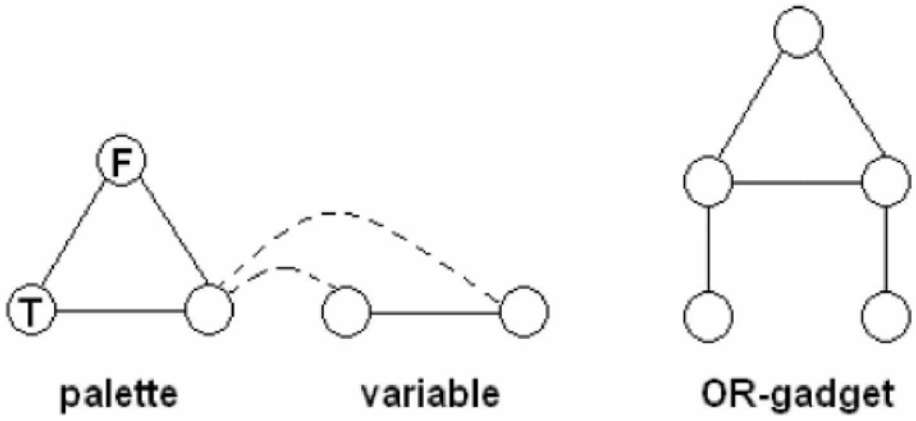
παραμικρή λεπτομέρεια την μηχανή Turing f που κάνει την αναγωγή, αλλά να αποδείξουμε ότι αυτή υπάρχει. Το να εντοπίσουμε μια συμβολοσειρά που αποδέχεται μια γλώσσα και μια που απορρίπτει δεν είναι εύκολο ακόμα και αν η γλώσσα αυτή αποφασίζεται σε πολυωνυμικό χρόνο. Φανταστείτε ότι η γλώσσα αυτή περιέχει μια μόνο συμβολοσειρά την w. Το να την εντοπίσουμε σημαίνει να δοκιμάσουμε πιθανόν όλες τις (άπειρες) συμβολοσειρές του Σ^*. Πράγμα που θέλει πιθανότατα μη πολυωνυμικό χρόνο.

5.2 ΑΣΚΗΣΗ ΓΙΑ ΤΟ ΠΡΟΒΛΗΜΑ 3-COLOR

Η εκφώνηση της παρακάτω άσκησης είναι από το βιβλίο τουSipserκαι είναι η άσκηση 7.27 την οποία και λύνουμε παρακάτω.

Εκφώνηση: Ένας χρωματισμός ενός γράφου (coloring) είναι μια ανάθεση χρωμάτων στους κόμβους του με τρόπο τέτοιο ώστε να μην υπάρχουν δυο γειτονικοί κόμβοι με το ίδιο χρώμα. Έστω $3-COLOR = \{\langle G\rangle$ ο G είναι γράφος του οποίου οι κόμβοι μπορούν να χρωματιστούν με 3 χρώματα με τρόπο τέτοιο ώστε να μην υπάρχουν δυο γειτονικοί κόμβοι με το ίδιο χρώμα$\}$. Να δείξετε ότι το πρόβλημα 3-COLOR είναι NP-Complete .

Υπόδειξη: Χρησιμοποιήστε τους τρεις υπογράφους του σχήματος 5.2 (a).



Σχήμα 5.2(a)

Σχήμα 5.2(a): Τα προτεινόμενα gadgets για την αναγωγή στην άσκηση 7.27.

Λύση:

Θα χρησιμοποιήσουμε τις κατασκευές που μας υποδεικνύει η εκφώνηση για να αναγάγουμε το 3-SAT στο 3-COLOR . Άρα από έναν ικανοποιήσιμο $3-(cnf)$ λογικό τύπο ϕ θα παίρνουμε 3-χρωματίσιμο γράφο. Κατασκευάζουμε ένα τρίγωνο που θα έχει τα ΄΄χρώματα΄΄: T(true), F (false) και A (auxilliary).Αυτός ο υπογράφος είναι η προτεινόμενη ΄΄παλέτα΄΄ (palette) που

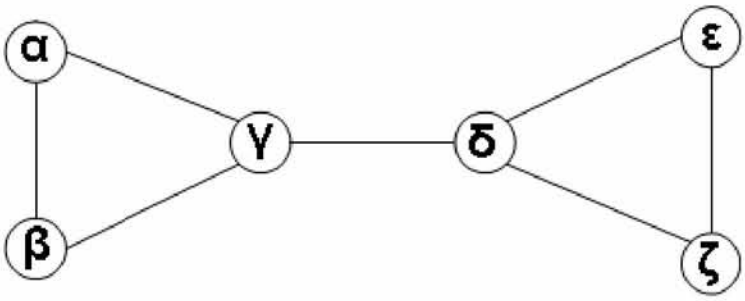
εμφανίζεται στο σχήμα 5.2 (a).

Για κάθε μεταβλητή x_i της ϕ φτιάχνουμε τον εξής γράφο:

Δυο κόμβοι ο ένας με ετικέτα x_i και ο άλλος με ετικέτα $\overline{x_i}$ οι οποίοι ενώνονται με πλευρά. Κάθε τέτοια κατασκευή ενώνεται με τον κόμβο A της παλέτας υποχρεώνοντας τη μια να "βαφτεί" F και την άλλη T.

Ακόμα για κάθε clause της ϕ εισάγουμε την κατασκευή του σχήματος 5.2 (b).

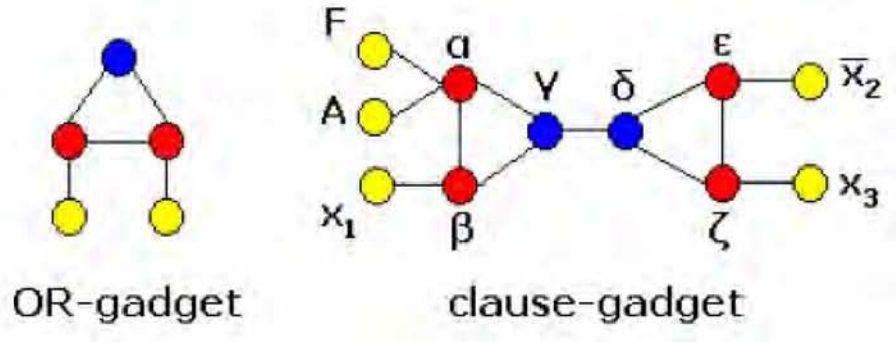
Τον κόμβο (α) τον συνδέουμε με τους κόμβους F και A της παλέτας ώστε να παίρνει πάντα το χρώμα T. Προσέξτε ότι το clause gadget το κατασκευάζουμε χρησιμοποιώντας δυο OR- gadgets. Αυτά τα OR -gadgets συνδέονται μέσω των κόμβων "κεφαλών" τους. Η σχέση του clause gadget με τα OR -gadgets απεικονίζεται στο σχήμα 5.2. (c) όσων αφορά την clause $(x_1 \vee \overline{x_2} \vee x_3)$.



Σχήμα 5.1(b)

Σχήμα 5.2(b): Το clause gadget που χρησιμοποιούμε.

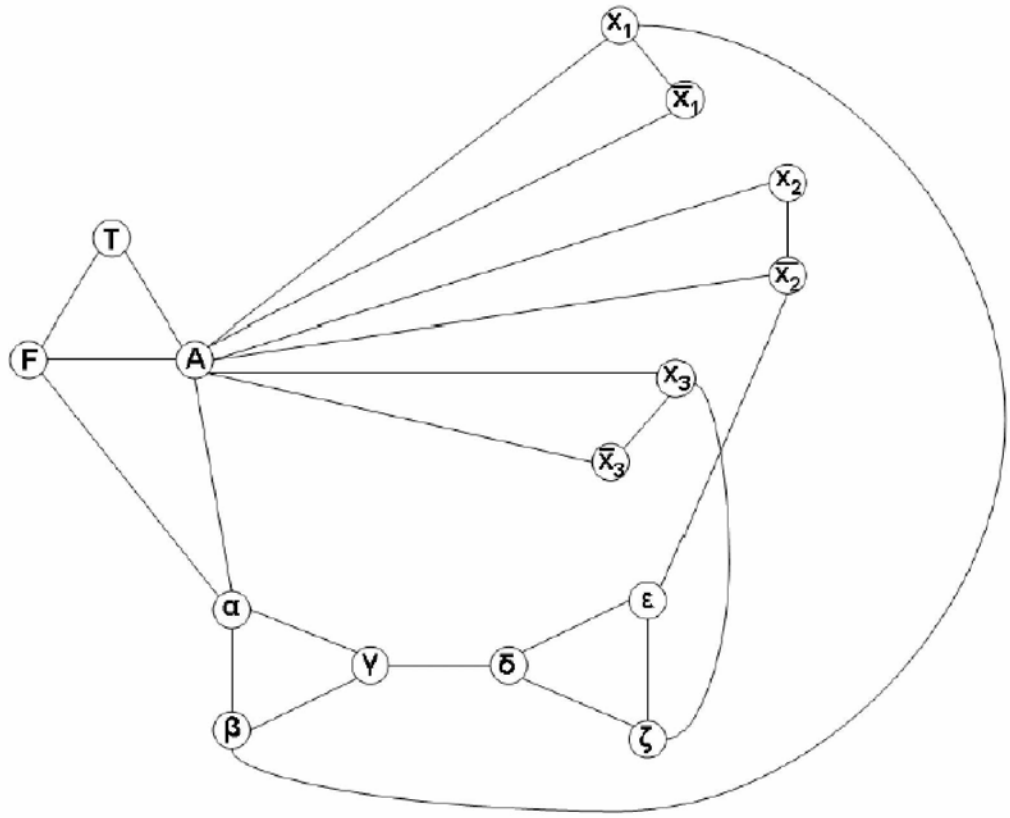
Ακόμα αν η clause είναι η $(x_1 \vee \overline{x_2} \vee x_3)$, συνδέουμε τον (β) κόμβο με τον κόμβο με ετικέτα x_1, τον κόμβο (ε) με αυτόν που έχει ετικέτα $\overline{x_2}$ και τον (ζ) με αυτόν που έχει ετικέτα x_3. Οι κόμβοι (γ) και (δ) συνδέονται όπως φαίνεται στο σχήμα 5.2(b).



Σχήμα 5.2(c)

Σχήμα 5.2(c): Η σχέση OR-gadget και clause γαδγετ για την clause $(x_1 \vee \overline{x_2} \vee x_3)$.

Αν $\phi = (x_1 \vee \overline{x_2} \vee x_3)$ τότε ο τελικός γράφος είναι αυτός που εμφανίζεται στο σχήμα 5.2(d):



Σχήμα 5.2(d)

Σχήμα 5.2(d): Ο τελικός γράφος για $\phi = (x_1 \vee \overline{x_2} \vee x_3)$.

Κάνοντας ανάλυση περιπτώσεων αποδεικνύεται ότι η ϕ είναι ικανοποιήσιμη αν και μόνο αν υπάρχει τρόπος να χρωματίσουμε τον γράφο με τρία χρώματα χωρίς δυο γειτονικοί κόμβοι να έχουν το ίδιο χρώμα. Άρα αποδείξαμε ότι η αναγωγή είναι επιτυχής και είναι εύκολο να διαπιστώσετε ότι ο χρόνος εκτέλεσης της αναγωγής είναι γραμμικά ανάλογος του μεγέθους της εισόδου, δηλαδή O(n) και επομένως πολυωνυμικός. Επομένως το 3-COLOR είναι NP-HARD.

Αυτό που απομένει είναι να αποδείξουμε ότι το $3 - COLOR \in NP$. Δοθέντος ενός γράφου και ενός 3-χρωματισμού για αυτόν, μπορεί να βρεθεί πολυωνυμικός αλγόριθμος επιβεβαίωσής του. Π.χ.:

"Έλεγξε αν της κάθε πλευράς τα άκρα έχουν διαφορετικό χρώμα. Αν βρεθεί πλευρά με άκρα του ίδιου χρώματος ΑΠΕΡΡΙΨΕ την είσοδο (δηλαδή το ζεύγος του γράφου και του 3-χρωματισμού του). Αν τέτοια πλευρά δεν βρεθεί ΑΠΟΔΕΞΟΥ την είσοδο."

Προφανώς η πολυπλοκότητα του αλγορίθμου αυτού είναι το πολύ $O(n^2)$ όσος δηλαδή και ο μέγιστος αριθμός πλευρών που ελέγχονται. Άρα αφού το 3-COLOR είναι NP- HARD και NP τότε θα είναι NP- Complete.

5.3 ΑΣΚΗΣΗ ΓΙΑ ΤΟ ΠΡΟΒΛΗΜΑ HALF-CLIQUE

Θα αποδείξουμε ότι το πρόβλημα $HALF - CLIQUE$ είναι NP-πλήρες πρόβλημα μέσω της υλοποίησης της παρακάτω άσκησης.

Η εκφώνηση της παρακάτω άσκησης είναι από το βιβλίο του Sipser και είναι η άσκηση 7.22 την οποία και λύνουμε παρακάτω.

Εκφώνηση: Έστω το πρόβλημα $HALF - CLIQUE = \{\langle G\rangle$ το G είναι ένας μη κατευθυνόμενος γράφος ο οποίος διαθέτει έναν πλήρη υπογράφο με τουλάχιστον $m/2$ κόμβους (όπου m το πλήθος των κόμβων του $G)\}$. Δείξτε ότι το πρόβλημα $HALF-CLIQUE$ είναι NP-Complete.

Λύση: Θα δείξουμε πρώτα ότι το $HALF - CLIQUE \in NP$.

Δεδομένης μιας λύσης δηλαδή ενός συνόλου με τουλάχιστον $m/2$ κόμβους του G μπορούμε να επιβεβαιώσουμε σε πολυωνυμικό χρόνο αν οι τουλάχιστον αυτοί $m/2$ κόμβοι αποτελούν μια κλίκα. Για να το κάνουμε αυτό για κάθε έναν ζευγάρι από αυτούς τους κόμβους που είναι το πολύ m, θα πρέπει να τσεκάρουμε αν συνδέονται από κάποια πλευρά. Τα ζευγάρια αυτά είναι το πολύ $\frac{m(m-1)}{2} = O(m^2)$.Άρα στη χειρότερη περίπτωση θα χρειαστούμε πολυωνυμικό χρόνο για να επιβεβαιώσουμε τη λύση.

Κατόπιν θα δείξουμε ότι το $HALF - CLIQUE$ είναι NP-HARD.Για να το δείξουμε αυτό θα αναγάγουμε σε πολυωνυμικό χρόνο το γνωστό NP -HARD πρόβλημα $CLIQUE = \{\langle G, k\rangle \mid$. Το G είναι μη κατευθυνόμενος γράφος ο οποίος διαθέτει μια k-κλίκα (δηλαδή έναν πλήρη υπογράφο με k κόμβους.

Ας δούμε αυτήν την αναγωγή όπου από τον αρχικό γράφο G παίρνουμε ένα νέο τον G': Αρχικά ελέγχουμε την τιμή του k.

• Αν $k = m/2$. Τότε $G = G'$. Προφανώς ο G έχει μιαk-κλίκα αν και μόνο αν οG' έχει μια $\frac{m'}{2}-$κλίκα (m' είναι το πλήθος των κόμβων του G'. Εδώ $m = m'$).

• Αν $k < m/2$. Ο G' προκύπτει από τον G ως εξής:Στον G προσθέτουμε $k' = m - 2k$ νέους κόμβους και τους συνδέουμε με κάθε κόμβο του G. Έτσι ο νέος γράφος G' θα έχει $m' = k' + m = 2m - 2k$. Επομένως ο Gέχει μια k-κλίκα αν και μόνο αν ο G' έχει μια $\frac{m'}{2}-$ κλίκα. Αυτό γιατί κάθε ένας από τους νεοεισερχόμενους κόμβους συνδέονται με όλους τους παλιούς και κατά συνέπεια και με όλους της κλίκας συμβάλλοντας έτσι στην αύξηση του μεγέθους της κλίκας κατά k'.

• Αν $k > m/2$. Ο G' προκύπτει από τον G ως εξής: Στον Gπροσθέτουμε αυτήν την φορά $k' = 2k - m$ τους οποίους όμως αφήνουμε ασύνδετους. Έτσι εδώ έχουμε $m' = k' + m = 2k$. Επομένως ο G έχει μια k-κλίκα αν και μόνο αν ο G' έχει μια $\frac{m'}{2}-$ κλίκα. Εδώ η παλιά κλίκα του G δεν μεγαλώνει αφού οι νέοι κόμβοι μένουν ασύνδετοι.

Επομένως το πρόβλημα είναι τόσο NP όσο και NP- HARD και έτσι είναι και NP - Complete.

5.4 ΑΣΚΗΣΗ ΓΙΑ ΤΟ ΠΡΟΒΛΗΜΑ MAX-CUT

Θα αποδείξουμε ότι το πρόβλημα MAX-CUT είναι NP-πλήρες πρόβλημα μέσω της υλοποίησης της παρακάτω άσκησης.

Η εκφώνηση της παρακάτω άσκησης είναι από το βιβλίο του Sipser και είναι η άσκηση 7.25 την οποία και λύνουμε παρακάτω.

Εκφώνηση: Μια **τομή** (cut) σε έναν μη κατευθυνόμενο γράφο είναι ένας επιμερισμός του συνόλου των κόμβων σε δυο ξένα μεταξύ τους υποσύνολα τα S και T. Το μέγεθος μιας τομής είναι ο αριθμός των πλευρών που έχουν το ένα τους άκρο στο ένα σύνολο (S) και το άλλο τους άκρο στο άλλο (T). Έστω το πρόβλημα $MAX-CUT = \{\langle G, k\rangle\}$ ο G είναι ένας μη κατευθυνόμενος γράφος και ο οποίος έχει μια τομή μεγέθους τουλάχιστον $k\}$. Δείξτε ότι το πρόβλημα MAX-CUT είναι NP- Complete.

Μπορείτε να χρησιμοποιήσετε το αποτέλεσμα της παραπάνω άσκησης 5.3.

Υπόδειξη:

Δείξτε ότι $\neq -SAT \leq_p MAX-CUT$. Η "κατασκευή" της μεταβλητής (variable gadget) για τη μεταβλητή x είναι μια συλλογή από $3c$ κόμβους με ετικέτα $''x''$ και άλλους $3c$ κόμβους με ετικέτα $''\overline{x}''$, με c να είναι το πλήθος των clauses. Όλοι οι κόμβοι που έχουν ετικέτα $''x''$ συνδέονται με όλους τους κόμβους που έχουν ετικέτα $''\overline{x}''$. Η "κατασκευή" της clause (clause gadget) είναι ένα τρίγωνο τριών πλευρών που συνδέουν τρεις κόμβους που έχουν ετικέτα τα literals που εμφανίζονται στην clause. Μην χρησιμοποιήσετε τον ίδιο κόμβο για περισσότερα από ένα clause gadgets . Αποδείξτε ότι αυτή η αναγωγή δουλεύει.

Λύση: Εκτελούμε την αναγωγή σύμφωνα με την υπόδειξη της εκφώνησης:

Από έναν $3-cnf$ λογικό τύπο ϕ που έχει $\neq -$ ανάθεση αλήθειας προκύπτει ένας γράφος G που έχει τομή μεγέθους τουλάχιστον $n(3k)^2 + 2k$. Όπου n ο αριθμός των μεταβλητών της ϕ, k ο αριθμός των clauses της ϕ. Κατόπιν:

1. $\forall$ μεταβλητή x_i της ϕ, προσθέτουμε $3k$ κόμβους για την x_i και άλλους $3k$ για την $\overline{x_i}$.

2. Συνδέουμε όλους τους κόμβους με ετικέτα x_i με όλους τους κόμβους που έχουν ετικέτα $\overline{x_i}$. Έτσι στο γράφο δημιουργούνται για κάθε μεταβλητή $x_i, 3k \times 3k = (3k)^2$ πλευρές. Άρα συνολικά για ν μεταβλητές έχουμε $n(3k)^2$ πλευρές.

3. Ακόμα συνδέουμε για κάθε clause της ϕ τα αντίστοιχα literals μεταξύ τους αν συνυπάρχουν στην clause. Επιλέγουμε για κάθε clause και κάθε literal a έναν αντιπρόσωπο από τους $(3k)$ ώστε αυτός να συνδεθεί με τον αντιπρόσωπο του literal b.

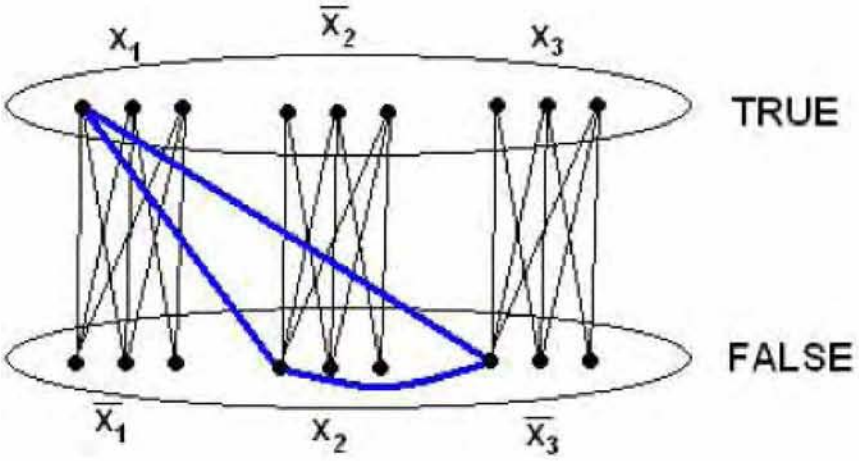
Αν η $\phi \in -SAT$ δηλαδή υπάρχει $\neq -$ ανάθεση αλήθειας, τότε χωρίζοντας τους κόμβους του γράφου σε αυτούς που έχουν αληθή τιμή και σε αυτούς που έχουν ψευδή τιμή, παίρνουμε δυο διαμερίσεις των κόμβων του G με μέγεθος τομής (cut) τουλάχιστον $n(3k)^2 + 2k$. Το $n(3k)^2$ οφείλεται στο γεγονός ότι οι $3k$ κόμβοι με ετικέταx_i θα μπουν σε διαφορετική διαμέριση από τους $3k$ με ετικέτα $\overline{x_i}$ υποχρεωτικά. Έτσι για τις μεταβλητές x_i έχουμε συνεισφορά $n(3k)^2$ πλευρών.

Οι $2k$ πλευρές προκύπτουν ως εξής: Για κάθε μια από τις k clauses αν έχουν άνιση ανάθεση πρέπει υποχρεωτικά δυο literals να βρεθούν στην ίδια διαμέριση και ένα literal στο άλλο. Αυτό σημαίνει ότι για κάθε clause δυο πλευρές θα ενώνουν κόμβους σε διαφορετικές διαμερίσεις. Έτσι θα έχουμε μια συνολική συνεισφορά $2k$ πλευρών στο μέγεθος της τομής από τις clauses.

Ας δώσουμε ένα απλό παράδειγμα όπου ο λογικός τύπος έχει μόνο μια clause : Έστω $\phi = (x_1 \vee \overline{x_2} \vee x_3)$. Μια $\neq -$ ανάθεση είναι η: $x_1 = 1, x_2 = 1, x_3 = 0$ αφού $x_1 \neq x_2$.

Ο παραγόμενος γράφος G εμφανίζεται στο σχήμα 5.4. Μπορούμε να παρατηρήσουμε σε αυτό ότι μεταξύ του συνόλου TRUE και του συνόλου FALSE υπάρχουν τουλάχιστον $n(3k)^2 + 2k$ πλευρές.

Το αντίστροφο αποδεικνύεται με τον ίδιο τρόπο. Σε γράφο που έχει διαμέριση με τομή τουλάχιστον $n(3k)^2 + 2k$ μέγεθος, αν αναθέσουμε τις τιμές ''TRUE'' στη μια διαμέριση και ''FALSE'' στην άλλη η αντίστοιχη ανάθεση στα literals του ϕ είναι μια $\neq -$ ανάθεση αλήθειας.



Σχήμα 5.4

Σχήμα5.4: Ο γράφος G όπως προκύπτει από την εφαρμογή της αναγωγής στον $\phi = (x_1 \vee \overline{x_2} \vee x_3)$. Οι μπλε πλευρές αντιστοιχούν στο clause gadget.

Κεφάλαιο 6

ΠΡΟΣΕΓΓΙΣΤΙΚΟΙ ΑΛΓΟΡΙΘΜΟΙ

Σε αυτό το κεφάλαιο παρατίθενται και αναλύονται προσεγγιστικοί αλγόριθμοι για κάποια προβλήματα της NP-πλήρους κλάσης.

6.1 ΠΡΟΣΕΓΓΙΣΤΙΚΟΣ ΛΟΓΟΣ ΚΑΙ ΣΦΑΛΜΑ ΠΡΟΣΕΓΓΙΣΗΣ

Πολλά προβλήματα πρακτικής σημασίας είναι NP-πλήρη αλλά πάρα το γεγονός ότι είναι πολύ σημαντικά δεν είναι δυνατή η λήψη μιας βέλτιστης λύσης. Εάν ένα πρόβλημα είναι NP-πλήρες είναι απίθανο να βρούμε έναν αλγόριθμο πολυωνυμικού χρόνου βέλτιστης λύσης. Υπάρχουν, όμως, τουλάχιστον τρεις προσεγγίσεις εύρεσης μιας τιμής που είναι κοντά στη βέλτιστη. Πρώτον, εάν οι πραγματικές εισαγωγές είναι μικρές, ένας αλγόριθμος με εκθετικό χρόνο εκτέλεσης μπορεί να είναι τέλεια ικανοποιητικός. Δεύτερον, έχουμε την δυνατότητα να μπορούμε να απομονώσουμε τις σημαντικές ειδικές περιπτώσεις που είναι επιλύσιμες σε πολυωνυμικό χρόνο. Τρίτον είναι επίσης δυνατόν να βρεθούν οι κοντινότερεσ-βέλτιστες λύσεις σε πολυωνυμικό χρόνο. Στην πράξη, οι κοντινότερες - βέλτιστες λύσεις είναι συχνά αρκετά καλές. Ένας αλγόριθμος που επιστρέφει τέτοιες λύσεις κοντά στις βέλτιστες λύσεις καλείται **προσεγγιστικός αλγόριθμος [1],[6][8][13].**

Όταν αντιμετωπίζουμε ένα NP-πλήρες πρόβλημα βελτιστοποίησης, μπορεί να θέλουμε να εξετάσουμε αλγορίθμους που δεν παράγουν βέλτιστες λύσεις, αλλά λύσεις που είναι εγγυημένα κοντά στο βέλτιστο. Δηλαδή σε ένα πρόβλημα βελτιστοποίησης :

-για κάθε δυνατή είσοδο x υπάρχει ένα σύνολο εφικτών λύσεων $F(x)$.

- κάθε εφικτή λύση $s \in F(x)$ έχει κόστος (ή αξία) $c(s)$.

• η επίλυση του προβλήματος συνίσταται στην εύρεση μίας εφικτής λύσης με βέλτιστο κόστος (μέγιστο ή ελάχιστο, ανάλογα με το πρόβλημα).

Σε κάθε πρόβλημα βελτιστοποίησης μπορούμε να αντιστοιχίσουμε ένα πρόβλημα απόφασης με την παρακάτω γενική μορφή:

• Είσοδος: ένα στιγμιότυπο x του προβλήματος βελτιστοποίησης και ένας αριθμός k.

• Ερώτηση: υπάρχει εφικτή λύση $s \in F(x)$ τέτοια ώστε $c(s) \leq (\geq) k$ ·

Πολλά από τα προβλήματα απόφασης μπορεί να αντιστοιχούν σε προβλήματα βελτιστοποίησης.

Αν για κάποιο πρόβλημα βελτιστοποίησης, λοιπόν, το αντίστοιχο πρόβλημα απόφασης είναι NP-πλήρες ,τότε, εκτός αν P=NP, δεν μπορούμε να σχεδιάσουμε αποδοτικό αλγόριθμο (πολυωνυμικό) για το πρόβλημα. Σε αυτή την περίπτωση ,λοιπόν, όπως αναφέραμε και προηγουμένως, στρεφόμαστε στον σχεδιασμό *προσεγγιστικού αλγορίθμου*, ο οποίος επιστρέφει μία εφικτή λύση, της οποίας το κόστος είναι κοντά σε αυτό της βέλτιστης.

Έστω, λοιπόν, ότι επιθυμούμε να βρούμε τέτοιες λύσεις (κοντά στις βέλτιστες) για ένα πρόβλημα βελτιστοποίησης, μεγιστοποίησης ή ελαχιστοποίησης. Δηλαδή έστω ένας αλγόριθμος A, ο οποίος για κάθε στιγμιότυπο x του προβλήματος επιστρέφει μία εφικτή λύση $A(x) \in F(x)$, με κόστος $APP(x) = c(A(x))$. Για κάθε στιγμιότυπο x του προβλήματος αυτού, υπάρχει μία βέλτιστη λύση με τιμή $opt(x)$ η οποία τιμή ας υποθέσουμε ότι είναι πάντα θετικός ακέραιος.

• Αν το πρόβλημα είναι πρόβλημα μεγιστοποίησης, τότε λέμε ότι ο A λύνει το πρόβλημα με παράγοντα προσέγγισης a, για $0 \prec a \prec 1$, αν
$\forall x \;\; APP(x) \geq a \cdot OPT(x)$.

• Αν το πρόβλημα είναι πρόβλημα ελαχιστοποίησης, τότε λέμε ότι ο A λύνει το πρόβλημα με παράγοντα προσέγγισης b, για $b \succ 1$, αν
$\forall x \;\; APP(x) \leq b \cdot OPT(x)$.

Το a μπορεί να είναι σταθερά ή φθίνουσα συνάρτηση του $|x|$. Αντίστοιχα το b μπορεί να είναι σταθερά ή αύξουσα συνάρτηση του $|x|$.

Ας υποθέσουμε τώρα ότι έχουμε ένα πολυωνυμικό αλγόριθμο A, ο οποίος όταν του δίνεται ένα στιγμιότυπο x του προβλήματος βελτιστοποίησης, επιστρέφει κάποια λύση με τιμή $A(x)$. Εφόσον το πρόβλημα είναι NP-πλήρες και ο A είναι πολυωνυμικός, δεν μπορούμε ρεαλιστικά να ελπίζουμε ότι η $A(x)$ αποτελεί πάντα τη βέλτιστη τιμή. Έστω, όμως, ότι γνωρίζουμε ότι η ακόλουθη ανισότητα ισχύει πάντα:

$|opt(x)\text{- } A\ (x)|/\ opt(x) \leq \varepsilon$,

όπου ε είναι κάποιος θετικός πραγματικός αριθμός, ελπίζουμε πολύ μικρός, που φράσσει από πάνω την χειρότερη περίπτωση σχετικού σφάλματος του αλγόριθμου A και ονομάζεται **σφάλμα προσέγγισης**. Αν ο αλγόριθμος A ικανοποιεί την ανισότητα αυτή για όλα τα στιγμιότυπα του προβλήματος x, τότε ονομάζεται ένας **ε- προσεγγιστικός αλγόριθμος**.

Αφού έχει αποδειχθεί ότι ένα πρόβλημα βελτιστοποίησης είναι NP-πλήρες, το εξής ερώτημα καθίσταται ιδιαίτερα σημαντικό: Υπάρχουν ε-προσεγγιστικοί αλγόριθμοι για το πρόβλημα

αυτό· Και αν ναι, πόσο μικρό μπορεί να γίνει το ε· Ας παρατηρήσουμε αρχικά ότι τέτοια ερωτήματα έχουν νόημα μόνο εάν υποθέσουμε ότι $R \neq NR$,διότι αν $R = NR$, τότε το πρόβλημα μπορεί να λυθεί ακριβώς, με $\varepsilon = 0$.

Η σχέση ανάμεσα στον παράγοντα προσέγγισης και στο σφάλμα προσέγγισης ενός αλγορίθμου φαίνεται στον παρακάτω πίνακα:

Πρόβλημα	Παράγοντα Προς.	Σφάλμα Προς.
Μεγιστοποίησης	$a = 1 - \varepsilon$	$\varepsilon = 1 - a$
Ελαχιστοποίησης	$b = \frac{1}{1-\varepsilon}$	$\varepsilon = 1 - \frac{1}{b}$

Πίνακας 6(α)

Όλα τα NP-πλήρη προβλήματα βελτιστοποίησης μπορούν συνεπώς να υποδιαιρεθούν σε τρεις μεγάλες κατηγορίες:

(α) Προβλήματα τα οποία είναι **πλήρως προσεγγίσιμα**, υπό την έννοια ότι υπάρχει για αυτά ένας ε-προσεγγιστικός αλγόριθμος πολυωνυμικού χρόνου για κάθε $\varepsilon > 0$,οσοδήποτε μικρό. Ένα NP-πλήρες πρόβλημα που ανήκει σε αυτή τη κατηγορία είναι ο χρονοπρογραμματισμός δύο μηχανών.

(β) Προβλήματα τα οποία είναι **μερικώς προσεγγίσιμα**, υπό την έννοια ότι υπάρχουν ε-προσεγγιστικοί αλγόριθμοι πολυωνυμικού χρόνου για αυτά, για κάποια περιοχή τιμών του ε, αλλά η περιοχή αυτή δεν φτάνει μέχρι το μηδέν, όπως στην περίπτωση των πλήρως προσεγγίσιμων προβλημάτων. Κάποια από τα NP-πλήρη προβλήματα βελτιστοποίησης που ανήκουν σε αυτή τη κατηγορία είναι το πρόβλημα της επικάλυψης με κόμβους (NODE-COVER) και το MAX SAT.

(γ) Προβλήματα τα οποία είναι **μη προσεγγίσιμα**, δηλαδή δεν υπάρχει ε-προσεγγιστικός αλγόριθμος για αυτά, με οσοδήποτε μεγάλο ε. Σε αυτή τη κατηγορία ανήκουν, δυστυχώς, πολλά από τα NP-πλήρη προβλήματα όπως, το πρόβλημα του περιοδεύοντος πωλητή, η κλίκα κ.α(πρέπει να υπάρξει τουλάχιστον ένα τέτοιο στοιχείο).

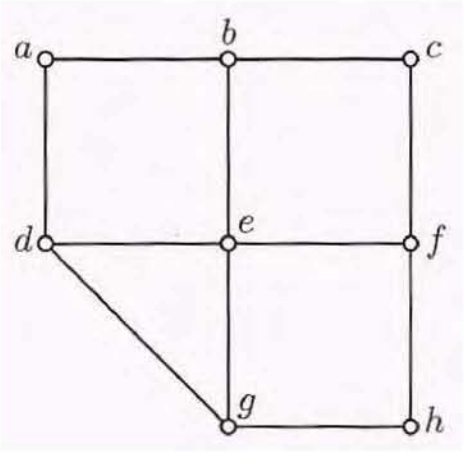
6.2 ΤΟ ΠΡΟΒΛΗΜΑ ΕΠΙΚΑΛΥΨΗ ΜΕ ΚΟΜΒΟΥΣ

Ας περιγράψουμε έναν 2-προσεγγιστικό αλγόριθμο για την ΕΠΙΚΑΛΥΨΗ ΜΕ ΚΟΜΒΟΥΣ[**1**], δηλαδή έναν αλγόριθμο , ο οποίος, για οποιοδήποτε γράφημα, επιστρέφει μία επικάλυψη με

κόμβους η οποία είναι το πολύ δύο φορές μεγαλύτερη από τη βέλτιστη. Ο αλγόριθμος είναι πολύ απλός:

$C := 0$ **While** υπάρχει εναπομένουσα ακμή $[u, v]$ στο G **do**
Πρόσθεσε τους u και v στο C, και διέγραψε τους από το G

Για παράδειγμα, στο γράφημα του παρακάτω σχήματος 6.2, ο αλγόριθμος θα μπορούσε να ξεκινήσει επιλέγοντας την ακμή $[a, b]$ και εισάγοντας και τα δύο άκρα της στο C, τότε και οι δύο κόμβοι (και οι γειτονικές τους ακμές φυσικά) διαγράφονται από το G. Στη συνέχεια μπορεί να επιλεγεί η $[e, f]$, και τέλος η $[g, h]$. Το σύνολο C που προκύπτει είναι μία επικάλυψη με κόμβους, καθώς κάθε ακμή του G πρέπει να αγγίζει έναν από τους κόμβους τησ(είτε επειδή την επέλεξε ο αλγόριθμος είτε επειδή τη διέγραψε). Στο συγκεκριμένο παράδειγμα , $C = \{a, b, e, f, g, h\}$, έχει έξι κόμβους, η οποία είναι το πολύ δύο φορές η βέλτιστη τιμή, στην περίπτωση αυτή, τέσσερα.



Σχήμα 6.2

Για να αποδείξουμε την εγγύηση «το πολύ δύο φορές», θεωρούμε την επικάλυψη C που επιστρέφει ο αλγόριθμος, και έστω $\widehat{C}$ η βέλτιστη επικάλυψη με κόμβους. Το $|C|$ είναι ακριβώς δύο φορές το πλήθος των ακμών που επιλέγει ο αλγόριθμος. Οι ακμές αυτές, όμως, λόγω ακριβώς του τρόπου με τον οποίο επιλέγονται, δεν έχουν κοινές κορυφές, και για καθεμία από αυτές τουλάχιστον ένα από τα άκρα της πρέπει να είναι στην $\widehat{C}$, καθώς η C είναι μία επικάλυψη με κόμβους. Προκύπτει ότι το πλήθος των ακμών που επιλέγονται από τον αλγόριθμο δεν είναι μεγαλύτερο από το βέλτιστο, και επομένως $|C| \leq 2\left|\widehat{C}\right|$, και αυτός είναι πράγματι ένας 2-προσεγγιστικός αλγόριθμος.

6.3 ΤΟ ΠΡΟΒΛΗΜΑ ΑΚΡΙΒΗΣ ΕΠΙΚΑΛΥΨΗ

Το πρόβλημα ΑΚΡΙΒΗΣ ΕΠΙΚΑΛΥΨΗ(VERTEX COVER) αποδείχθηκε ότι είναι NP-πλήρες προηγουμένως.

Μία ακριβής επικάλυψη ενός μη κατευθυνόμενου γράφου $G = (V, E)$ είναι ένα υποσύνολο $V' \subseteq V$ έτσι ώστε αν (u, v) είναι μία ακμή του G, τότε ή $u \in V'$ ή $v \in V'$ (ή και τα δύο). Το μέγεθος μιας ακριβούς επικάλυψης είναι ο αριθμός των κορυφών σε αυτή.

Το πρόβλημα ΑΚΡΙΒΗΣ ΕΠΙΚΑΛΥΨΗ[**3**] είναι να βρεθεί μία ακριβής επικάλυψη με ελάχιστο μέγεθος σε ένα δοσμένο μη κατευθυνόμενο γράφο. Καλούμε σαν μία ακριβή επικάλυψη μία βέλτιστη ακριβή επικάλυψη. Αυτό το πρόβλημα είναι μία έκδοση βελτιστοποίησης ενός NP-πλήρους προβλήματος απόφασης.

Αν και πρέπει να είναι δύσκολο να βρεθεί μία βέλτιστη ακριβής επικάλυψη σε ένα γράφο G, δεν είναι τόσο δύσκολο να βρεθεί μία ακριβής επικάλυψη κοντά στο βέλτιστο. Ο ακόλουθος προσεγγιστικός αλγόριθμος παίρνει σαν είσοδο ένα μη κατευθυνόμενο γράφο και επιστρέφει μία ακριβή επικάλυψη, το μέγεθος της οποίας είναι εγγυημένα όχι πάνω από το διπλάσιο μέγεθος μίας βέλτιστης ακριβούς επικάλυψης.

APPROX-VERTEX-COVER

1 $C \leftarrow 0$

2 $E' \leftarrow E[G]$

3 while $E' = 0$

4 **do** let (u, v) be an arbitrary edge of E'

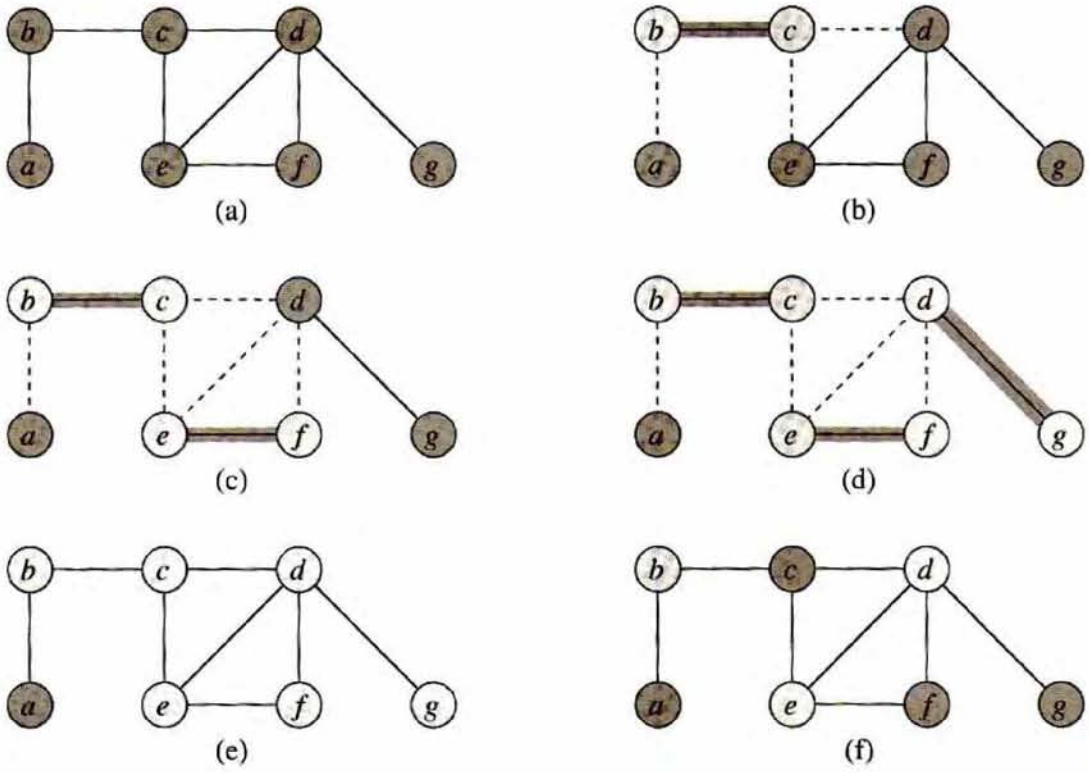
5 $C \leftarrow C \cup \{u, v\}$

6 remove from E' every edge incident on either u or v

7 return C

[3]

Το παρακάτω σχήμα 6.3 επεξηγεί την λειτουργία του APPROX-VERTEX-COVER . Η μεταβλητή C περιέχει την ακριβή επικάλυψη που κατασκευάστηκε. Στη γραμμή 1 αρχικοποιούμε το C στο κενό σύνολο. Στη γραμμή 2 τα σύνολα E' είναι μία αντιγραφή των ακμών του συνόλου $E[G]$ του γράφου. Ο βρόχος στις γραμμές 3 έως 6 επιλέγει επαναλαμβανόμενα μία ακμή (u, v) από το E', της προσθέτει σημεία τέλους u και v στο C , και διαγράφει όλες τις ακμές στο E' που είναι επικαλυμμένες από την u ή την v. Ο χρόνος εκτέλεσης αυτού του αλγορίθμου είναι $O(V + E)$, χρησιμοποιώντας γειτονικές λίστες που αντιπροσωπεύουν το E'.



Σχήμα 6.3

σχόλια σχήματος 6.3 : Η λειτουργία του προσεγγιστικού αλγόριθμου του προβλήματος ΑΚΡΙΒΗΣ ΕΠΙΚΑΛΥΨΗ(APPROX-VERTEX-COVER)
(a) Ο γράφος G ως είσοδος, ο οποίος έχει 7 κορυφές και 8 ακμές
(b) Η ακμή (b, c) είναι η πρώτη ακμή που επιλέχθηκε από τον APPROX-VERTEX-COVER.Οι κορυφές b και c, προστέθηκαν στο σύνολο C περιέχοντας την ακριβή επικάλυψη που δημιουργείται. Οι ακμές (a, b) , (c, e) και (c, d) μετακινήθηκαν από τότε που οι αντίστοιχες κορυφές είναι επικαλυμμένες από άλλες ακμές.
(c) Οι ακμές (e, f) επιλέχθηκαν: οι κορυφές e και f προστέθηκαν στο C.
(d) Οι ακμές (d, g) επιλέχθηκαν: οι κορυφές d και g προστέθηκαν στο C.
(e) Το σύνολο C, το οποίο είναι η ακριβής επικάλυψη παραγόμενη από τον APPROX-VERTEX-COVER, περιέχει τις 6 κορυφές b, c, d, e, f, g
(f) Η βέλτιστη ακριβής επικάλυψη για αυτό το πρόβλημα περιέχει μόνο τρεις κορυφές b, d και e.

ΘΕΩΡΗΜΑ 6.3:
Ο APPROX-VERTEX-COVER είναι ένας 2-προσεγγιστικός αλγόριθμος πολυωνυμικού χρόνου.[3]

Απόδειξη:

Έχουμε ήδη αποδείξει ότι ο APPROX-VERTEX-COVER τρέχει σε πολυωνυμικό χρόνο. Το σύνολο C των κορυφών που έχει επιστραφεί από τον APPROX-VERTEX-COVER είναι μία ακριβής επικάλυψη, από τότε που ο βρόχος του αλγορίθμου εκτελείται μέχρι κάθε ακμή στο $E[G]$ να έχει επικαλυφθεί από κάποιες κορυφές στο C.

Για να δούμε ότι ο APPROX-VERTEX-COVER επιστρέφει μία ακριβή επικάλυψη που είναι το πολύ διπλάσια από το μέγεθος μιας βέλτιστης επικάλυψης, ας πούμε ότι το A είναι το σύνολο των ακμών που είχαν επικαλυφθεί στη γραμμή 4 του APPROX-VERTEX-COVER. Προκειμένου να επικαλύψει τις ακμές στο A, κάθε κορυφή επικαλύπτει, συγκεκριμένα, μία βέλτιστη επικάλυψη C^*- πρέπει να περιλαμβάνεται το λιγότερο ένα σημείο τέλους από κάθε ακμή του A. Δύο ακμές στο A δεν μοιράζονται ένα σημείο τέλουσ- από τη στιγμή πού μία ακμή θα επιλεχθεί στη γραμμή 4 ,όλες οι άλλες ακμές που είναι συναφείς στα σημεία τέλους της θα διαγραφούν από το E' στη γραμμή 6. Κατά συνέπεια, δύο ακμές στο A δεν μπορούν να επικαλυφθούν από την ίδια κορυφή από το C^*, και έτσι έχουμε το κατώτατο όριο

$$|C^*| \geq |A| \tag{6.1}$$

στο μέγεθος μιας βέλτιστης ακριβούς επικάλυψης. Κάθε εκτέλεση της γραμμής 4 επιλέγει μία ακμή για την οποία κανένα από τα σημεία τέλους της δεν είναι ήδη στο C, παράγοντας ένα ανώτατο όριο στο μέγεθος που η ακριβής επικάλυψη επιστρέφεται:

$$|C^*| = 2\,|A| \tag{6.2}$$

Συνδυάζοντας τις ανισότητες (6.1) και (6.2) παίρνουμε:

$$|C| = 2\,|A|$$

$$\leq 2\,|C^*|\,,$$

και μετά από αυτό αποδεικνύεται το θεώρημα.

Ας κάνουμε μία απεικόνιση αυτής της απόδειξης. Κατ΄ αρχάς μπορεί να αναρωτιέστε πως είναι δυνατόν να αποδεικνύεται ότι το μέγεθος της ακριβούς επικάλυψης που επιστρέφεται από τον APPROX-VERTEX-COVER είναι το πολύ διπλάσιο από το μέγεθος μιας βέλτιστης ακριβούς επικάλυψης, αφού εμείς δεν γνωρίζουμε ποιο είναι το μέγεθος της βέλτιστης ακριβούς επικάλυψης. Η απάντηση είναι ότι χρησιμοποιούμε ένα κατώτατο όριο στη βέλτιστη ακριβή επικάλυψη. Το σύνολο A των ακμών που επιλέχθηκε στη γραμμή 4 του APPROX-VERTEX-COVER είναι στην πραγματικότητα ένα μέγιστο ταίριασμα (maximal matching) στο γράφο G (ένα μέγιστο ταίριασμα είναι ένα ταίριασμα το οποίο δεν είναι κατάλληλο υποσύνολο κανενός άλλου ταιριάσματος). Το μέγεθος ενός μέγιστου ταιριάσματος είναι, όπως συμφωνήσαμε προηγουμένως, ένα κατώτατο όριο στο μέγεθος μιας βέλτιστης ακριβούς επικάλυψης. Ο αλγόριθμος επιστρέφει μία ακριβή επικάλυψη της οποίας το μέγεθος είναι το

πολύ διπλάσιο από το μέγεθος του μέγιστου ταιριάσματος. Συσχετίζοντας το μέγεθος της λύσης που επιστρέφεται στο κατώτατο όριο, παίρνουμε τον προσεγγιστικό μας λόγο.

6.4 ΤΟ ΠΡΟΒΛΗΜΑ TSP

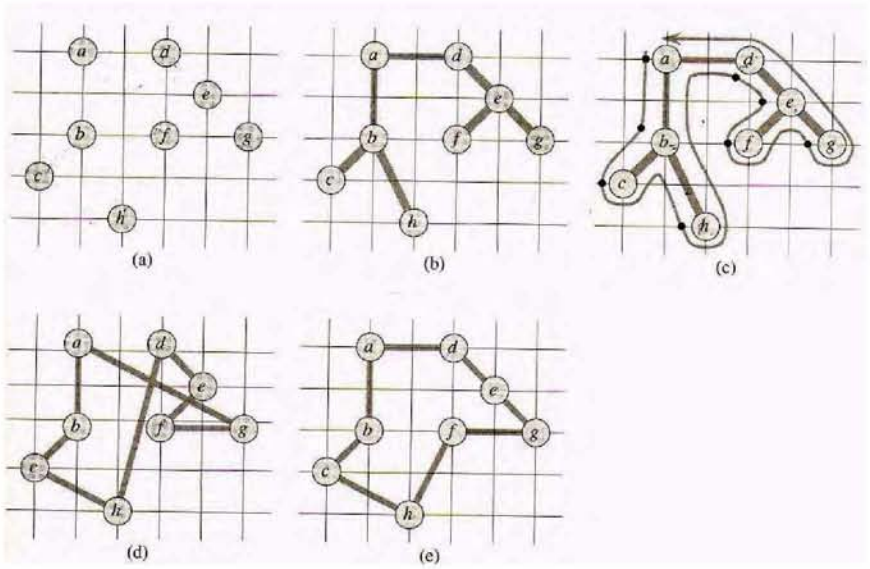
Είναι το πρόβλημα TSP **[3]** με τον επιπλέον περιορισμό ότι ισχύει η τριγωνική ανισότητα, δηλαδή για παράδειγμα για τους κόμβους v, u, w ισχύει η τριγωνική ανισότητα, δηλαδή $c(u,v)(u,w) + c(w,v)$.

Ισχύοντας η μεθοδολογία της προηγούμενης ενότητας, εμείς θα υπολογίσουμε πρώτα μία δομή- ένα ελάχιστο επικαλύπτον δέντρο- το βάρος του οποίου είναι ένα κατώτατο όριο από το μήκος μιας βέλτιστης διαδρομής ενός πλανόδιου πωλητή. Στη συνέχεια, θα χρησιμοποιήσουμε το ελάχιστο επικαλύπτον δέντρο για να δημιουργήσουμε μία διαδρομή, το κόστος της οποίας δεν είναι πάνω από το διπλάσιο από αυτό του βάρους του ελάχιστου επικαλύπτοντος δέντρου, εφόσον η συνάρτηση κόστους ικανοποιεί την τριγωνική ανισότητα. Ο ακόλουθος αλγόριθμος καλείται minimum-spanning-tree algorithm MST-Prim.

Προσεγγιστικός αλγόριθμος TSP διαδρομής (APPROX-TSP-TOUR(G, c))

1. select a vertex $r \in V[G]$ to be a '' root '' vertex
2. compute a minimum spanning tree T for G from root r using MST-PRIM (G, c, r)
3. let L be the first list of vertices visited in a preorder tree walk of T
4. **return** the hamiltonian cycle H that visits the vertices in the order L

[3]



Σχήμα 6.4

σχόλια σχήματος 6.4:
(a) Δίνεται ένα σύνολο από σημεία, που είναι οι κορυφές που έχουμε και βρίσκονται σε ένα πλέγμα ακέραιων αριθμών. Για παράδειγμα η f βρίσκεται μία μονάδα αριστερά και δύο μονάδες πάνω από την h. Η συνηθισμένη ευκλείδεια απόσταση χρησιμοποιείται σαν την συνάρτηση κόστους μεταξύ δύο σημείων.
(b) Ένα ελάχιστο επικαλύπτον δέντρο T από αυτά τα σημεία, όπως υπολογίζεται από το MST-PRIM.Η κορυφή a ορίζεται ως η ρίζα (αφετηρία) του δέντρου. Οι κορυφές τυχαίνει να έχουν όλες ετικέτες με γράμματα με τέτοιο τρόπο με τον οποίο προστίθενται στο κύριο δέντρο από τον MST-PRIM με αλφαβητική σειρά.
(c) Μία διάσχιση του T, ξεκινώντας από το a. Μία πλήρης διάσχιση ολόκληρου του δέντρου επισκέπτεται τις κορυφές με την σειρά:
$a, b, c, b, h, b, a, d, e, f, e, g, e, d, a$. Η preorder διαδρομή του T λοιπόν είναι η

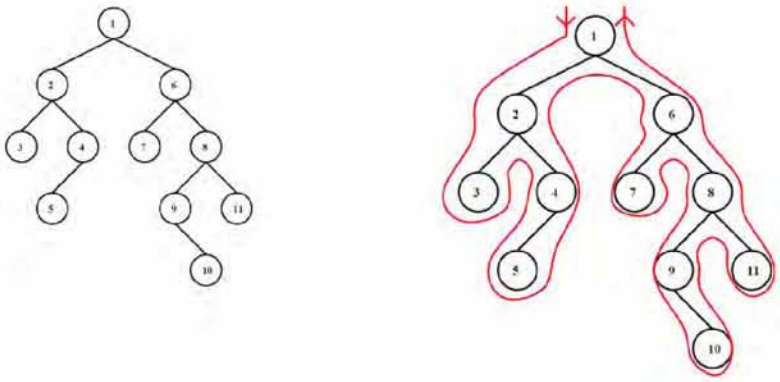
$$a, b, c, h, d, e, f, g$$

(d) Μία διαδρομή από τις κορυφές αποκτήθηκε διασχίζοντας τις κορυφές με τη διάταξη που προκύπτει από την πρεορδερ διάσχιση. Αυτή είναι η διαδρομή H που επιστρέφεται από τον APPROX-TSP-TOUR.Το συνολικό της κόστος είναι προσεγγιστικά 19.074.
(e) Μία βέλτιστη διαδρομή H^* για το δοσμένο σύνολο από τις κορυφές. Το συνολικό της κόστος είναι προσεγγιστικά 14.715.

Πριν προχωρήσουμε θα κάνουμε μία αναφορά στους τρόπους διάσχισης των δέντρων για να κατανοήσουμε τον τρόπο με τον οποίο μπορούμε να διασχίσουμε τα δέντρα ώστε να καταλάβουμε την λειτουργία του αλγορίθμου.

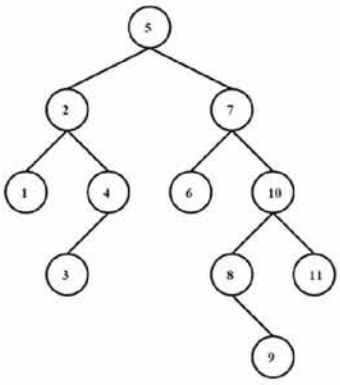
Οι βασικές διασχίσεις δέντρων είναι οι εξής:

1)η προδιατεταγμένη διάσχιση (preorder traversal),η οποία γίνεται ως εξής: επίσκεψη σε έναν κόμβο του δένδρου, επίσκεψη σε όλους τους κόμβους του αριστερού υποδένδρου, επίσκεψη σε όλους τους κόμβους του δεξιού υποδένδρου. Π.χ. κινούμαστε όπως δείχνει το σχήμα:



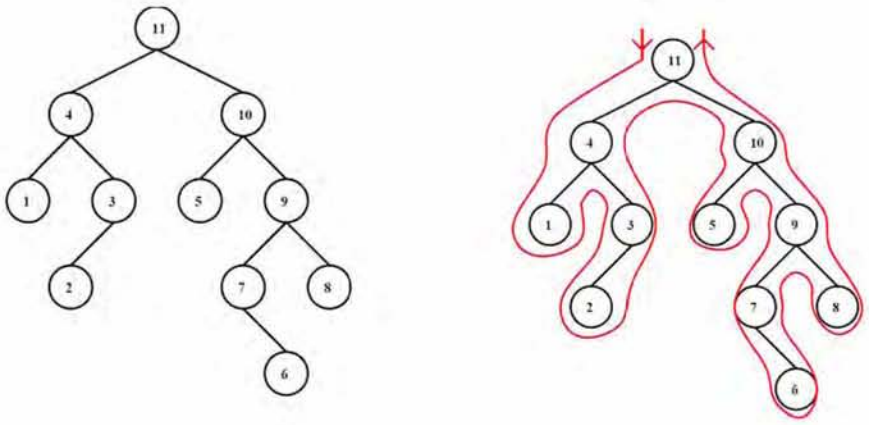
Σχήμα 6.3(b)

2)η ενδοδιατεταγμένη διάσχιση (inorder traversal) ,η οποία γίνεται ως εξής: επίσκεψη πρώτα στους κόμβους του αριστερού υποδένδρου, μετά στον κόμβο και τέλος στους κόμβους του δεξιού υποδένδρου. Π.χ. κινούμαστε όπως δείχνει το σχήμα:



Σχήμα 6.3(c)

3)η μεταδιατεταγμένη διάσχιση (postorder traversal),η οποία γίνεται ως εξής: επίσκεψη στους κόμβους του αριστερού υποδένδρου, μετά στους κόμβους του δεξιού υποδένδρου και τέλος στον κόμβο μέσω του οποίου συνδέονται. Π.χ. κινούμαστε όπως δείχνει το σχήμα:



Σχήμα 6.3(d)

Εμείς στο παράδειγμα μας χρησιμοποιούμε την πρεορδερ διάσχιση, δηλαδή, επισκεπτόμαστε κάθε κόμβο, πριν επισκεφτούμε (αριθμήσουμε) σε προδιάταξη το αριστερό και το δεξί υποδέντρο του.

Μετά από αυτή τη μικρή αναφορά ας ξαναγυρίσουμε στο θέμα μας και στο πρώτο σχήμα μας το οποίο επεξηγεί την λειτουργία του APPROX-TSP-TOUR . Στο σχήμα (a) φαίνεται το δοσμένο σύνολο των κορυφών και στο σχήμα (b) το ελάχιστο επικαλύπτον δέντρο T με ρίζα την κορυφή a από το MST-PRIM . Στο σχήμα (c) φαίνεται πως διασχίζονται οι κορυφές με την preorder διάσχιση του T και στο σχήμα (d) εμφανίζεται η αντίστοιχη διαδρομή, που είναι η διαδρομή που επιστρέφεται από τον APPROX-TSP-TOUR. Στο σχήμα (e) εμφανίζεται μία βέλτιστη διαδρομή, η οποία είναι περίπου 23% συντομότερη.

ΘΕΩΡΗΜΑ 6.4:
Ο APPROX-TSP-TOUR είναι ένας 2-προσεγγιστικός αλγόριθμος πολυωνυμικού χρόνου για το πρόβλημα TSP με τον περιορισμό της τριγωνικής ανισότητας.[3]

ΑΠΟΔΕΙΞΗ:

Έχουμε ήδη αποδείξει ότι ο APPROX-TSP-TOUR τρέχει σε πολυωνυμικό χρόνο. Σε αυτό το σημείο θα πούμε ότι το H^* είναι η βέλτιστη λύση και H είναι η λύση την οποία βρίσκει ο αλγόριθμος. Επίσης θα πούμε W την πλήρη διάσχιση του δέντρου, ενώ το ελάχιστο επικαλύπτον δέντρο είναι το T με κόστη c(H*), c(H), c(T), c(W) .

Αναλυτικότερα: Είπαμε ότι H^* είναι η βέλτιστη διαδρομή για το δοσμένο σύνολο από κορυφές. Επειδή πήραμε ένα επικαλύπτον δέντρο διαγράφοντας κάθε ακμή από τη διαδρομή, το βάρος του ελάχιστου επικαλύπτοντος δέντρου T είναι το κατώτατο όριο στο κόστος από μία βέλτιστη διαδρομή , το οποίο είναι

$$c(T) \leq c(H^*), \tag{6.3}$$

αφού ένας κύκλος Hamilton γίνεται σπαννινγ τρεε με την αφαίρεση μίας ακμής.
Ορίζουμε ως πλήρη διάσχιση του δέντρου W, τη διάσχιση που βρήκαμε προηγουμένως (σχήμα (c)), όπου διασχίσαμε το δέντρο με ρίζα το a,και είναι η

$$a, b, c, b, h, b, a, d, e, f, e, g, e, d, a$$

Αυτή λοιπόν τη διαδρομή είναι που λέμε W.
Δεδομένου ότι η πλήρης αυτή διαπέραση, διασχίζει κάθε ακμή ακριβώς δύο φορές, έχουμε

$$c(W) = 2c(T) \tag{6.4}$$

Εξισώνοντας την (6.3) και (6.4) προκύπτει ότι

$$c(W) \leq 2c(H^*) \tag{6.5}$$

και λοιπόν το κόστος του W είναι διπλάσιο από το κόστος της βέλτιστης διαδρομής.
Δυστυχώς, το W δεν είναι γενικά μία διαδρομή, επειδή επισκέφθηκε κάποιες κορυφές περισσότερες από μία φορές. Από την τριγωνική ανισότητα παρ΄ όλα αυτά μπορούμε να διαγράψουμε μία επίσκεψη σε οποιαδήποτε κορυφή από το W και το κόστος δεν θα αυξηθεί. Αφού λοιπόν ισχύει αυτή η λειτουργία, μπορούμε να μετακινήσουμε από το W όλες εκτός από τις πρώτες επισκέψεις σε κάθε κορυφή. Έτσι στο παράδειγμά μας αυτό αφήνει τη διάταξη

$$a, b, c, h, d, e, f, g,$$

Αυτή η διάταξη είναι η ίδια από αυτή που πήραμε από την πρεορδερ διάσχιση. Αυτός είναι ένας hamiltonian (χαμιλτόνιος) κύκλος, επειδή κάθε κορυφή διαπεράστηκε ακριβώς μία φορά, και στην πραγματικότητα είναι ο κύκλος που υπολογίστηκε από το APPROX-TSP-TOUR. Από τότε που το H προέκυψε διαγράφοντας κορυφές από την πλήρη διάσχιση W, έχουμε

$$c(H) \leq c(W) \tag{6.6}$$

Συνδυάζοντας τις ανισότητες (6.5) και (6.6) δείχνουμε ότι

$$c(H) \leq 2c(H^*),$$

το οποίο ολοκληρώνει και την απόδειξή μας.

6.5 ΤΟ ΠΡΟΒΛΗΜΑ MAX-CUT

Στο πρόβλημα βελτιστοποίησης MAX-CUT(OPT) δίνεται ένα γράφημα $G = (V, E)$ και ζητείται να βρεθεί ένα σύνολο κορυφών S υποσύνολο του V, τέτοιο ώστε να υπάρχουν όσο το δυνατόν περισσότερες ακμές ανάμεσα στα δύο σύνολα S και $V - S$ (δηλαδή το πλήθος των ακμών που ενώνουν κορυφές του S με τις κορυφές του $V - S$ να είναι μέγιστο).

Το πρόβλημα απόφασης που αντιστοιχεί στο MAX-CUT(OPT) είναι το MAX-CUT που είναι NP-πλήρες.

Έστω ο παρακάτω αλγόριθμος[**6**] για το MAX-CUT(OPT) :

• $S = 0$

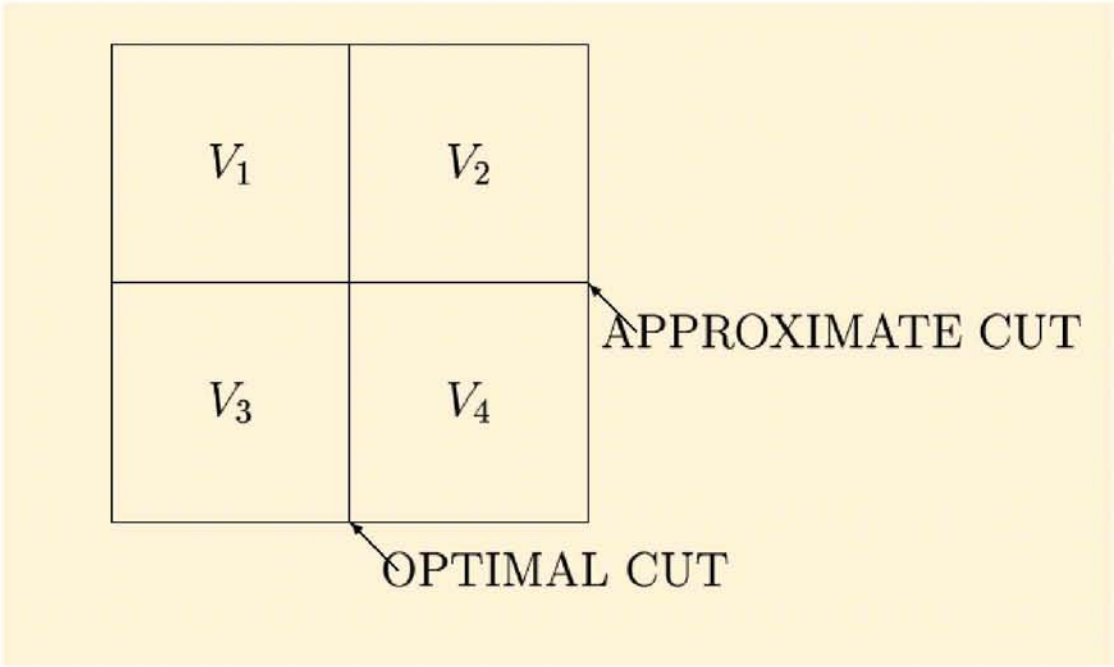
• Όσο υπάρχει κορυφή $v \in V$, η μετακίνηση της οποίας από ή προς το S αυξάνει το μέγεθος της τομής, μετακίνησε την v.

Ο αλγόριθμος ξεκινάει από μία αυθαίρετη τομή π.χ. $S = 0$. Σε κάθε βήμα προσπαθεί να κάνει μία τοπική βελτίωση: αν υπάρχει μία κορυφή της οποίας η μετακίνηση από το σύνολο στο οποίο ανήκει στο άλλο σύνολο αυξάνει το μέγεθος της τομής, τότε κάνει αυτή τη μετακίνηση. Αν δεν υπάρχει καμία τέτοια κορυφή τότε τερματίζει, επιστρέφοντας το σύνολο S. Ο αλγόριθμος τερματίζει το πολύ μετά από $|E|$ μετακινήσεις κορυφών. Κάθε μετακίνηση κορυφής απαιτεί πολυωνυμικό χρόνο.

ΘΕΩΡΗΜΑ 6.5: Ο παραπάνω αλγόριθμος για το MAX-CUT έχει παράγοντα προσέγγισης $\frac{1}{2}$.

Απόδειξη:

Έστω S' το σύνολο κορυφών που ορίζει μία βέλτιστη τομή. Οι κορυφές του V με βάση τις δύο τομέσ(τη βέλτιστη και την προσεγγιστική) χωρίζονται σε τέσσερα σύνολα V_1, V_2, V_3, V_4 έτσι ώστε : $S = V_1 \cup V_2$ και $S' = V_1 \cup V_3$.



Σχήμα 6.5

Συμβολίζουμε με $k_i(v)$ το πλήθος των ακμών που ενώνουν την κορυφή v με τις κορυφές του V_i και με $n_{i,j}$ το πλήθος των ακμών που ενώνουν κορυφές του V_i με κορυφές του V_j. Έστω τυχαία κορυφή $v \in V_1$. Ισχύει:

$$k_1 + k_2(v) \leq k_3(v) + k_4(v)$$

αλλιώς ο αλγόριθμος θα βελτίωνε την τομή μετακινώντας τη v από το S στο $V - S$. Άρα $k_2(v) \leq k_3(v) + k_4(v)$. Αθροίζοντας για όλες τις κορυφές του V_1 παίρνουμε:

$$n_{1,2} \leq n_{1,3} + n_{1,4}$$

Εργαζόμενοι με τον ίδιο τρόπο για τα σύνολα V_2, V_3, V_4 παίρνουμε:

$$n_{1,2} \leq n_{2,3} + n_{2,4}$$

$$n_{3,4} \leq n_{1,3} + n_{2,3}$$

$$n_{3,4} \leq n_{1,4} + n_{2,4}$$

Αθροίζοντας τις τέσσερις παραπάνω σχέσεις και διαιρώντας τα δύο μέλη της ανισότητας που προκύπτει με το 2 παίρνουμε:

$$n_{1,2} + n_{3,4} \leq n_{1,3} + n_{1,4} + n_{2,3} + n_{2,4}$$

Προφανώς

$$n_{1,4} + n_{2,3} \leq n_{1,3} + n_{1,4} + n_{2,3} + n_{2,4}.$$

Αθροίζοντας τις δύο τελευταίες σχέσεις κατά μέλη και παρατηρώντας ότι

$$OPT(G) = n_{1,2} + n_{3,4} + n_{1,4} + n_{2,3}$$

και

$$APP(G) = n_{1,3} + n_{1,4} + n_{2,3} + n_{2,4}$$

Προκύπτει ότι

$$OPT(G) \leq 2APP(G) \Leftrightarrow APP(G) \geq \frac{1}{2}OPT(G)$$

6.6 ΤΟ ΠΡΟΒΛΗΜΑ ΤΟΥ ΣΑΚΙΔΙΟΥ

Στο πρόβλημα του ΣΑΚΙΔΙΟΥ, το οποίο έχουμε ορίσει και στο προηγούμενο κεφάλαιο, ζητείται να γεμίσουμε το σακίδιο επιπλέον αντικείμενα με τη μέγιστη δυνατή αξία και συνολικό βάρος το πολύ W.

ΘΕΩΡΗΜΑ 6.6: Ο προσεγγιστικός λόγος για το πρόβλημα του ΣΑΚΙΔΙΟΥ είναι 0. Αυτό σημαίνει ότι για οποιοδήποτε $\varepsilon > 0$ υπάρχει ένας $\varepsilon-$προσεγγιστικός αλγόριθμος πολυωνυμικού χρόνου για αυτό το πρόβλημα.[6]

Απόδειξη:

Υποτίθεται ότι μας δίνεται ένα στιγμιότυπο του προβλήματος I του προβλήματος του σακιδίου. Δηλαδή, μας δίνονται n αντικείμενα με βάρη $w_1, w_2, ..., w_n$ και τιμές $p_1, p_2, ..., p_n$ και η χωρητικότητα W του σακιδίου και ζητείται να βρεθεί ένα υποσύνολο $S \subseteq \{1, 2, ..., n\}$ τέτοιο ώστε να μεγιστοποιεί το $\sum_{i \in S} p_i$ με τον περιορισμό $\sum_{i \in S} w_i \leq W$.

Το πρόβλημα απόφασης του ΣΑΚΙΔΙΟΥ είναι NP-πλήρες.

Παρακάτω περιγράφουμε έναν αλγόριθμο για το πρόβλημα του ΣΑΚΙΔΙΟΥ. Έστω n αντικείμενα με βάρη $w_1, w_2, ..., w_n$ και τιμές $p_1, p_2, ..., p_n$, W η χωρητικότητα του σακιδίου

και $K = \sum_{i \in S} p_i$ και $L = \max_{i=1}^{n} p_i$. Συμβολίζουμε $C(i, Y), \ \ 0 \leq i \leq n, 0 \leq Y \leq K$ την ελάχιστη χωρητικότητα που απαιτείται να έχει το σακίδιο, ώστε επιλέγοντας κάποια από τα i πρώτα αντικείμενα να πάρουμε συνολική αξία ακριβώς Y.

Το $C(i, Y)$ υπολογίζεται με βάση την παρακάτω αναδρομική σχέση:

$$C(0,0) = 0$$
$$C(0,Y) = \infty \text{ αν } Y \geq 1$$
$$C(i+1,Y) = \begin{cases} C(i,Y) \alpha \nu p_{i+1} > Y \\ \min(C(i,Y), C(i,Y-p_{i+1}) + w_{i+1}) \alpha \lambda \lambda \iota \varsigma \end{cases}$$

Στη συνέχεια δείχνουμε την ορθότητα της παραπάνω αναδρομικής σχέσης.

Για $i = 0$ έχουμε κενό σύνολο αντικειμένων και μπορούμε να επιτύχουμε συνολική αξία 0, με διαθέσιμο βάρος 0, και δεν μπορούμε να πετύχουμε θετική συνολική αξία, όσο μεγάλο βάρος και αν διαθέτουμε. Έστω ότι θέλουμε να βρούμε το βέλτιστο κόστος $C(i+1, Y)$, γνωρίζοντας τις τιμές των $C(i, Z)$ για όλα τα Z. Έχουμε δύο επιλογές για το αντικείμενο $i+1$:

• Αν δεν το επιλέξουμε τότε θα πρέπει να επιλέξουμε κάποια από τα i πρώτα αντικείμενα με το ελάχιστο δυνατό βάρος και αξία ακριβώς Y. Το κόστος σε αυτή την περίπτωση είναι $C(i, Y)$.

•Αντίθετα αν το επιλέξουμε, τότε το βάρος του συνυπολογίζεται στο κόστος. Επιπλέον θα πρέπει να επιλέξουμε κάποια από τα i πρώτα αντικείμενα με το ελάχιστο δυνατό βάρος, των οποίων η συνολική αξία θα πρέπει να είναι ακριβώς $Y - p_{i+1}$. Το κόστος σε αυτή την περίπτωση είναι $C(i, Y - p_{i+1}) + w_{i+1}$.

Από τις δύο επιλογές μας κρατάμε αυτή που δίνει το μικρότερο κόστος.

Το μέγιστο Y για το οποίο $C(n, Y) \leq W$ είναι η μέγιστη δυνατή αξία αντικειμένων που μπορούμε να βάλουμε στο σακίδιο. Ο υπολογισμός των $C(i, Y)$ μπορεί να γίνει από κάτω προς τα πάνω σε χρόνο $O(n^2 \cdot L)$. Οι επιλογές που κάνουμε κάθε φορά αποθηκεύονται σε ένα πίνακα S και χρησιμοποιούνται για το σχηματισμό της βέλτιστης λύσης.

Έστω I ένα στιγμιότυπο του προβλήματος όπως είπαμε στην αρχή της απόδειξής μας. Μπορούμε να θεωρήσουμε χωρίς βλάβη της γενικότητας ότι για κάθε αντικείμενο $w_i \leq W$, συνεπώς $OPT(I)$. Κατασκευάζουμε ένα νέο στιγμιότυπο I' που προκύπτει από το I αν μετατρέψουμε τα b λιγότερο σημαντικά ψηφία κάθε τιμής σε 0, δηλαδή θέτοντας για κάθε αντικείμενο $p'_i = 2^b \left\lfloor \frac{p_i}{2^b} \right\rfloor$. Τα βάρη των αντικειμένων και η χωρητικότητα του σακιδίου παραμένουν τα ίδια. Έστω S και S' τα σύνολα των αντικειμένων που αποτελούν βέλτιστη λύση για τα στιγμιότυπα I και I'. Μπορούμε να θεωρήσουμε ότι η κατασκευή του I' και η εκτέλεση του αλγορίθμου για το I', συνιστούν έναν προσεγγιστικό αλγόριθμο, ο οποίος επιστρέφει το S' ως μία προσεγγιστική λύση για το στιγμιότυπο I, με κόστος $APP(I)$.

Ισχύει:

$APP(I) = \sum_{i \in S} p_i \geq \sum_{i \in S'} p'_i \geq \sum_{i \in S} p'_i \geq \sum_{i \in S}(p_i - 2^b) \geq (\sum_{i \in S} p_i) - n \cdot 2^b = OPT(I) - n \cdot 2^b$

όπου η δεύτερη ανισότητα ισχύει επειδή το S' αποτελεί βέλτιστη λύση για το I'. Η παραπάνω σχέση συνεπάγεται ότι:

$APP(I) \geq OPT(I) \cdot (1 - \frac{n \cdot 2^b}{OPT(I)} \geq OPT(I) \cdot (1 - \frac{n \cdot 2^b}{L})$.

Έστω ότι θέλουμε ο παραπάνω προσεγγιστικός αλγόριθμος να έχει παράγοντα a. Αρκεί $(1 - \frac{n \cdot 2^b}{L}) \geq a \Leftrightarrow 2^b \leq \frac{(1-a) \cdot L}{n} \Leftrightarrow b \leq \log \frac{(1-a) \cdot L}{n}$.

Αν επιλέξουμε $b = \left\lfloor \log \frac{(1-a) \cdot L}{n} \right\rfloor$ τότε ο αλγόριθμος έχει παράγοντα προσέγγισης a. Επιπλέον κατά την εκτέλεση του αλγορίθμου για το στιγμιότυπο I', τα b λιγότερο σημαντικά ψηφία μπορούν να αγνοηθούν, συνεπώς ο χρόνος εκτέλεσης είναι: $O(\frac{n^2 \cdot L)}{2^b}) = O(\frac{n^3}{(1-a)})$, λαμβάνοντας υπόψη την επιλογή για την τιμή του b. Η κατασκευή του I' γίνεται σε χρόνο $O(|I|)$. Συνεπώς υπάρχει ένα σχήμα προσέγγισης πολυωνυμικού χρόνου για το διακριτό πρόβλημα σακιδίου.

Συνοψίζοντας, λέμε ότι υπάρχει ένας $\varepsilon-$προσεγγιστικός αλγόριθμος πολυωνυμικού χρόνου για κάθε $\varepsilon > 0$ και συνεπώς το κατώτατο όριο του λόγου είναι 0.

6.6.1 ΑΛΓΟΡΙΘΜΟΣ FPTAS ΓΙΑ ΤΟ ΠΡΟΒΛΗΜΑ ΤΟΥ ΣΑΚΙΔΙΟΥ

Παρακάτω παρατίθεται ένας ψευδοπολυωνυμικός αλγόριθμος για το KNAPSACK : **[8]**

Έστω P το μέγιστο κέρδος αντικειμένου. Τότε το συνολικό κέρδος δεν μπορεί να υπερβαίνει το $n \cdot P$. Για $i \in \{1, 2, ..., n\}$ και $p \in \{1, 2, ..., nP\}$, έστω $S_{i,p}$ ένα υποσύνολο του $\{a_1, a_2, ..., a_i\}$ με συνολικό κέρδος π και το ελάχιστο μέγεθος.

Έστω $A(i, p)$ το μέγεθος του $S_{i,p}$. Θέτουμε: $A(i, p) = \infty$ αν δεν υπάρχει $S_{i,p}$. Το $A(1, p)$ είναι γνωστό για κάθε $p \in \{1, 2, ..., nP\}$.

Ο ακόλουθος αναδρομικός τύπος υπολογίζει τα $A(i, p)$, γεμίζοντας μία προς μία τις γραμμές ενός $n \times nP$ πίνακα:

$$A(i+1, p) = \begin{cases} \min\{A(i, p), s(\alpha_{i+1}) + A(i, p - p(\alpha_{i+1}))\}, & \alpha\nu \ \ p(\alpha_{i+1}) < p \\ A(i, p), & \alpha\lambda\lambda\iota\varsigma \end{cases}$$

Για να γεμίσει τις γραμμές του πίνακα ο αλγόριθμος αυτός έχει πολυπλοκότητα $O(n^2 P)$ και επομένως είναι ψευδοπολυωνυμικός αφού εξαρτάται όχι μόνο από το μέγεθος της εισόδου (n αντικείμενα) αλλά και από το μέγιστο κέρδος ενός αντικειμένου P.

Ορισμός 6.6.1:

Έστω P ένα NP -Hard πρόβλημα με αντικειμενική συνάρτηση f_π (η συνάρτηση που θέλουμε να μεγιστοποιήσουμε ή να ελαχιστοποιήσουμε).

• Λέμε ότι ένας αλγόριθμος A είναι approximation scheme (προσεγγιστικό σχήμα) για το P, αν σε είσοδο (I, e), όπου I είναι ένα στιγμιότυπο του P και $e > 0$ είναι η παράμετρος σφάλματος, αποδίδει λύση s ώστε:

α) $f_\pi(I, s) \leq (1 + \varepsilon)OPT$, αν το P είναι πρόβλημα ελαχιστοποίησης,

β) $f_\pi(I, s) \geq (1 + \varepsilon)OPT$, αν το P είναι πρόβλημα μεγιστοποίησης,

όπου OPT το κόστος της βέλτιστης λύσης.

• Θα λέμε ότι ο A είναι polynomial time approximation scheme (PTAS- προσεγγιστικό σχήμα πολυωνυμικού χρόνου) αν για κάθε $\varepsilon > 0$ ο χρόνος εκτέλεσης του φράσσεται από ένα πολυώνυμο ως προς το |I|.

• Αν ο χρόνος εκτέλεσης του A φράσσεται από ένα πολυώνυμο ως προς το |I|και το 1/ε τότε ο A θα λέγεται fully polynomial time approximation scheme (FPTAS- προσεγγιστικό σχήμα πλήρους πολυωνυμικού χρόνου).

Η ιδέα πίσω από έναν αλγόριθμο που είναι PTAS ή FPTAS είναι ότι ανταλλάσουμε υπολογιστική ακρίβεια με πολυπλοκότητα χρόνου (όσο μικρότερο το σφάλμα ε τόσο μεγαλύτερη η παράμετρος 1/ε και επομένως η χρονική πολυπλοκότητα του FPTAS).

Αλγόριθμος 6.6.1 **FPTAS** για το KNAPSACK

1. Για $\varepsilon > 0$ θέτουμε $K = \frac{\varepsilon \mathrm{P}}{n}$.
2. Για κάθε αντικείμενο a_i θέτουμε $p^{'}(\alpha_i) = \left\lfloor \frac{p(\alpha_i)}{K} \right\rfloor$.
3. Με τα νέα κέρδη χρησιμοποίησε τον ψευδοπολυωνυμικό αλγόριθμο για να βρεις το βέλτιστο κέρδος $S^{'}$.
4. Έξοδος το $S^{'}$.

Θεώρημα 6.6.1(a) : profit$(S^{'}) \geq (1 - e)OPT$, όπου ORT είναι το βέλτιστο κέρδος για το δεδομένο στιγμιότυπο του KNAPSACK.

Απόδειξη:

Έστω S το σύνολο των αντικειμένων και O το υποσύνολο αυτού που δίνει τη βέλτιστη λύση για το KNAPSACK. Για το βήμα 2 του αλγόριθμου χρειάζεται να εκτελέσουμε την ευκλείδεια διαίρεση $0 \leq p(a) - p'(a) \cdot K < K$, όπου $0 \leq u < K$. Άρα $0 \leq p(a) - p'(a) \cdot K < K$. Αθροίζοντας για όλα τα αντικείμενα του O θα ισχύει ότι $p(O) - Kp'(O) \leq n \cdot K$. Το βήμα 3 θα δώσει ένα σύνολο τουλάχιστον εξίσου καλό όπως το O σε ότι αφορά τα νέα κέρδη. Άρα:

$$P(S) \geq K \cdot p'(O) \geq p(O) - n \cdot K = OPT - \varepsilon P \geq (1 - \varepsilon) \cdot OPT,$$

όπου η τελευταία ανίσωση έπεται από το γεγονός ότι OPT.

Θεώρημα 6.6.1(b) : Ο αλγόριθμος 6.6.1 είναι FPTAS για το KNAPSACK

.

Απόδειξη:

Ήδη έχουμε δείξει ότι είναι ε-κοντά στην βέλτιστη λύση, αρκεί να δείξουμε ότι είναι πολυωνυμικός στο μέγεθος της εισόδου n και στο $1/\varepsilon$.

Πράγματι, αφού $P' = \lfloor \frac{P}{K} \rfloor = \lfloor \frac{n}{\varepsilon} \rfloor$ έπεται ότι $O(n^2 P') = O(n^2 \lceil \frac{n}{\varepsilon} \rceil) = O(n^3 \frac{1}{\varepsilon})$.

ΣΥΜΠΕΡΑΣΜΑΤΑ:

Σε αυτό το κεφάλαιο λοιπόν εκθέσαμε κάποιους προσεγγιστικούς αλγορίθμους για κάποια από τα προβλήματα της NP-πλήρους κλάσης. Αυτό που πρέπει να κατανοήσουμε όσον αφορά τους προσεγγιστικούς αλγορίθμους είναι το πώς λειτουργούν. Ένας προσεγγιστικός αλγόριθμος χρησιμοποιείται για να παράγει μία λύση αράγει την βέλτιστη λύση. Έτσι η λύση που επιστρέφει ο προσεγγιστικός αλγόριθμος είναι μία λύση κοντά στο βέλτιστο που όμως είναι ικανοποιητική. Επίσης υπάρχει το σφάλμα προσέγγισης ε, που είναι αυτή η διαφορά της λύσης που επιστρέφει ο προσεγγιστικός αλγόριθμος από την βέλτιστη λύση που θα έπρεπε να επιστρέφει. Έτσι λέμε ότι ένα αλγόριθμος, για παράδειγμα, είναι 2-προσεγγιστικός αλγόριθμος (ή προσεγγιστικός αλγόριθμος με λόγο προσέγγισης 2), όταν ο λόγος της βέλτιστης λύσης από την λύση που επιστρέφει είναι 2.

Κεφάλαιο 7

ΕΠΙΛΟΓΟΣ

Σκοπός της εργασίας μας ήταν να κάνουμε μία εισαγωγή στην θεωρία υπολογισμού και την NP-πληρότητα. Τα θέματα που καλύπτονται εδώ είναι, εν συντομία, η θεωρία πολυπλοκότητας, η υπολογισιμότητα από μηχανές Turing ,και η υπολογιστική πολυπλοκότητα. Συγκεκριμένα, στο κεφάλαιο 1 δίνεται μία εισαγωγή όσον αφορά την θεωρία πολυπλοκότητας, τα υπολογιστικά προβλήματα και την πολυπλοκότητα των αλγορίθμων. Στο κεφάλαιο 2 περιγράφεται το βασικό υπολογιστικό μοντέλο που χρησιμοποιείται, η μηχανή Turing , και διάφορες παραλλαγές του, γίνεται αναφορά στην πολυπλοκότητα του χρόνου και χώρου των αλγορίθμων και προβλημάτων και τέλος γίνεται μία εισαγωγή στις κλάσεις πολυπλοκότητας. Οι τρεις σημαντικές κλάσεις **P**, **NP** και **NP-πλήρης,** εξετάζονται στο κεφάλαιο 3. Αρχικά αναφέρονται οι κλάσεις P και NP και στη συνέχεια γίνεται εκτενής αναφορά στην κλάση NP-πλήρη, ενώ στο επόμενο κεφάλαιο 4, μελετάμε κάποια σημαντικά προβλήματα που ανήκουν στην NP-πλήρη κλάση και αποδεικνύεται η NP-πληρότητά τους. Στο κεφάλαιο 5 λύνονται κάποιες ασκήσεις που έχουν να κάνουν με προβλήματα της NP-πλήρους κλάσης. Στο κεφάλαιο 6, παραθέτουμε κάποιους προσεγγιστικούς αλγόριθμους για τα προβλήματα της NP-πληρότητας. Τέλος παρατίθεται παράρτημα, όπου στο πρώτο μέρος υλοποιούμε ένα προσεγγιστικό αλγόριθμο για το πρόβλημα της ακριβής επικάλυψης (Vertex-Cover), και στο δεύτερο μέρος παρατίθεται μία λίστα από πληθώρα προβλημάτων που ανήκουν στην NP-πλήρη κλάση.

Σκοπός του πρώτου κεφαλαίου, όπως αναφέρθηκε προηγουμένως, ήταν να κάνουμε μία εισαγωγή όσον αφορά την θεωρία πολυπλοκότητας, τα υπολογιστικά προβλήματα και την πολυπλοκότητα των αλγορίθμων. Το γενικό συμπέρασμα, που είναι πολύ σημαντικό, είναι ότι η θεωρητική πληροφορική ταξινομεί τα προβλήματα σε υπολογιστικά ῾έύκολα῾ και ῾ύπολογιστικά῾ δύσκολα, και ανάλογα, τα κατατάσσει σε κλάσεις πολυπλοκότητας. Σε κλάσεις πολυπλοκότητας, λοιπόν, κατατάσσουμε υπολογιστικά προβλήματα τα οποία ενώ είναι επιλύσιμα δεν είναι γνωστός κάποιος αποδοτικός αλγόριθμος για την επίλυση τους. Όπως είπαμε, υπάρχουν προβλήματα που χαρακτηρίζονται υπολογιστικά ῾έύκολα῾ και υπολογιστικά ῾δύσκολα῾, επομένως η κατάταξη στις κλάσεις πολυπλοκότητας γίνεται με βάση τους υπολο-

γιστικούς πόρους που αυτά απαιτούν για να επιλυθούν αλγοριθμικά. Με αυτό εννοούμε πως κάθε πρόβλημα και αλγόριθμος σπαταλά υπολογιστικούς πόρους για την επίλυση του, που εστιάζεται στον χρόνο που χρειάζεται για να επιλυθεί αλλά και στον αναγκαίο για τους υπολογισμούς χώρο (μνήμη) που σπαταλά ο αλγόριθμος. Έτσι ανάλογα με τους υπολογιστικούς πόρους που απαιτούν κατατάσσουμε τα προβλήματα αυτά σε κλάσεις πολυπλοκότητας. Οι σημαντικότερες κλάσεις είναι η P, NP και NP-πλήρη, με τις οποίες και ασχολούμαστε σε αυτή την εργασία, δίνοντας μεγαλύτερη έμφαση βέβαια στην κλάση NP-πλήρη.

Πριν όμως δούμε αυτές τις κλάσεις πολυπλοκότητας μελετήσαμε τη μηχανή Turing.Η μηχανή Turing είναι ένα εντυπωσιακά απλό μοντέλο, που χρησιμοποιείται από την Θεωρία Πολυπλοκότητας για την μελέτη της υπολογιστικής πολυπλοκότητας των προβλημάτων, ιδιαίτερα κατάλληλο για την ανάπτυξη της θεωρίας μας που αποδεικνύεται ότι παρέχει όλη την δυνατή υπολογιστική ισχύ που μπορεί να έχει ο μηχανιστικός υπολογισμός και μάλιστα με επιδόσεις αρκετά κοντά σε αυτές των ρεαλιστικών υπολογιστών. Με άλλα λόγια η μηχανή Turing διαθέτει δύο σημαντικά χαρακτηριστικά. Πρώτον είναι αρκετά κοντά στους πραγματικούς (ρεαλιστικούς) υπολογιστές ώστε τα αποτελέσματα που θα παίρνουμε από την μαθηματική ανάλυση στο μοντέλο μας να είναι μεταφέρσιμα στις πραγματικές μηχανές και, δεύτερον, είναι ιδιαίτερα απλό μοντέλο και κάνει την ανάπτυξη της θεωρίας μας απλή. Αυτοί είναι οι λόγοι που κάνουν την μηχανή Turing ως το ευρέως χρησιμοποιημένο υπολογιστικό μοντέλο, και ο λόγος που την συναντήσαμε πολλές φορές στη συνέχεια τόσο για τον ορισμό των κλάσεων πολυπλοκότητας, όσο και για την απόδειξη της NP-πληρότητας πολλών σημαντικών προβλημάτων.

Στην εργασία μας, είδαμε επίσης πολλές παραλλαγές της μηχανής Turing.Η προσπάθεια για την εύρεση του «απόλυτου» υπολογιστικού μοντέλου οδήγησε τους ερευνητές σε διάφορες επεκτάσεις της μηχανής Turing . Όλες αυτές οι προσπάθειες αποδεικνύουν ότι δεν προσφέρουν τίποτα ως προς το ποιά προβλήματα μπορεί να επιλύσει το βελτιωμένο μοντέλο αφού οι προκύπτουσες μηχανές αποδεικνύονται ισοδύναμες με την βασική μηχανή Turing . Παρ΄ όλα αυτά η δημιουργία αυτών των επεκτάσεων της μηχανής Turing έχει αξία αφού μπορεί να μην έχει βρεθεί επέκταση που να υπολογίζει καινούργια προβλήματα, δεν ισχύει όμως το ίδιο και ως προς το πόσο αποτελεσματικά υπολογίζει τα υπάρχοντα προβλήματα.Άλλες πάλι επεκτάσεις έχουν το πλεονέκτημα ότι είναι πιο βολικές και μας επιτρέπουν να σχεδιάζουμε μηχανές Turing πολύ ευκολότερα από ότι με το βασικό σχήμα.

Εμείς δώσαμε έμφαση στην λειτουργία της ντετερμινιστικής και μη ντετερμινιστικής μηχανής Turing , όπως και στη μηχανή $k-$ταινιών και τυχαίας προσπέλασης. Η διαφορά που εντοπίσαμε όσον αφορά την μη ντετερμινιστική από την ντετερμινιστική μηχανή είναι ότι στην μη ντετερμινιστική δίνουμε κατά κάποιο τρόπο στην μηχανή την ελευθερία να βρίσκεται ταυτόχρονα σε περισσότερα από ένα σημεία του υπολογισμού. Επίσης η μηχανή $k-$ταινιών παρέχει πολύ μεγαλύτερη ευχέρεια από την ντετερμινιστική στον σχεδιασμό του προγράμματος. Ο λόγος είναι ότι κάθε μη τετριμμένο πρόγραμμα χρησιμοποιεί ένα αριθμό από βοηθητικές μεταβλητές και με τις $k-$ταινίες υπάρχει η δυνατότητα να χρησιμοποιηθεί μία ταινία για κάθε

μεταβλητή του προγράμματος, γεγονός που διευκολύνει τον σχεδιασμό. Τέλος η μηχανή τυχαίας προσπέλασης δεν είναι μία ακόμη παραλλαγή της μηχανής Turing αλλά ένα τυπικό μοντέλο που μοιάζει πολύ με τους σύγχρονους υπολογιστές. Η αξία του βρίσκεται στον χρόνο που απαιτεί η προσομοίωσή του από τη μηχανή Turing, ο οποίος είναι ένα μικρό πολυώνυμο του χρόνου που απαιτεί η μηχανή Turing.

Μετά από την αναφορά μας στις μηχανές Turing, είδαμε τις τρεις κλάσεις P, NP και NP-πλήρη με αναφορά σε πολύ κρίσιμες έννοιες όπως αυτή της αναγωγής και της πληρότητας, της λογικής Bool , του θεωρήματος του Cook και Levin και του προβλήματος της ικανοποιησιμότητας. Συμπερασματικά, για να κατανοήσουμε τις τρεις αυτές κλάσεις, είναι σημαντικό να καταλάβουμε τις βασικές σχέσεις και διαφορές που υπάρχουν μεταξύ τους και κυρίως ότι οι διαφορές αυτές εστιάζονται στη διαφορετική υπολογιστική πολυπλοκότητα του κάθε προβλήματος και στους υπολογιστικούς πόρους που απαιτεί για την επίλυσή του. Έτσι, όπως προαναφέραμε, η κλάση P εμπεριέχει όλα τα προβλήματα που λύνονται σε πολυωνυμικό χρόνο, δηλαδή σε αυτή την κλάση ανήκουν τα προβλήματα για τα οποία υπάρχει πολυωνυμικός αλγόριθμος που τα επιλύει, ενώ η κλάση NP εμπεριέχει όλα τα προβλήματα που επαληθεύουν την λύση τους σε πολυωνυμικό χρόνο. Αυτό που πρέπει να κατανοήσουμε είναι το ποια είναι η σχέση μεταξύ αυτών των δύο κλάσεων.

Αυτό που συμβαίνει είναι ότι κάθε πρόβλημα στην κλάση P ανήκει επίσης και στην NP, δηλαδή ισχύει $P \subseteq NP$, δηλαδή το σύνολο NP είναι υπερσύνολο του P (αυτό ισχύει διότι οι ντετερμινιστικές μηχανές είναι απλά μη ντετερμινιστικές, στις οποίες οι σχέσεις μετάβασης συμβαίνει να είναι συνάρτηση). Ωστόσο το μεγαλύτερο αναπάντητο ερώτημα είναι εάν ισχύει η σχέση $P = NP$, δηλαδή εάν οι κλάσεις προβλημάτων NP και P ταυτίζονται. Αυτό το ερώτημα μετά από πολλά χρόνια έρευνας δεν έχει ακόμα απαντηθεί αυστηρά, αν και υπάρχουν ισχυρές ενδείξεις ότι τα δύο σύνολα δεν είναι ίδια και ότι υπάρχουν προβλήματα που ανήκουν στην κλάση NP αλλά όχι στην P. Όμως ποια είναι η σχέση της κλάσης NP με την NP-πλήρη· Η απάντηση σε αυτό το ερώτημα είναι η έξης: Μερικά από τα προβλήματα που ανήκουν στο NP είναι «δύσκολα» προβλήματα για τα οποία ξέρουμε μόνο εκθετικούς αλγορίθμους για την επίλυσή τους και όχι πολυωνυμικούς (μιας και πολυωνυμικοί αλγόριθμοι δεν έχουν βρεθεί ακόμα). Έτσι ένα NP-δύσκολο πρόβλημα που ανήκει στο NP είναι NP-πλήρες. Αν μπορούσαμε να αποδείξουμε ότι ένα από τα δύσκολα προβλήματα που ανήκουν στην NP-πλήρη ανήκουν και στην P τότε θα λύναμε το μεγαλύτερο πρόβλημα στην επιστήμη των μαθηματικών, όπως προαναφέραμε, αφού θα δείχναμε ότι $P = NP$.

Επίσης πολύ σημαντική για την NP-πληρότητα είναι η έννοια της αναγωγής. Κι αυτό γιατί μπορούμε να αποδείξουμε ότι ένα πρόβλημα ανήκει στην κλάση NP-πλήρη με το να αναγάγουμε ένα άλλο πρόβλημα που έχουμε αποδείξει ότι ανήκει σε αυτή την κλάση στο δεύτερο πρόβλημα που θέλουμε να αποδείξουμε. Έτσι, κάνοντας χρήση της γνώσης μας για το πρώτο πρόβλημα που είναι NP-πλήρες μπορούμε να αποδείξουμε την NP-πληρότητα του δεύτερου προβλήματος. Βέβαια, είναι κατανοητό ότι για να αρχίσουμε να αποδεικνύουμε την NP πληρότητα κάποιων προβλημάτων πρέπει να αποδείξουμε ένα πρόβλημα σίγουρα ότι είναι

NP-πλήρες ώστε να μπορούμε να το αναγάγουμε σε άλλα προβλήματα για να αποδείξουμε την NP-πληρότητά τους. Το πρώτο λοιπόν πρόβλημα που αποδείχθηκε ότι είναι NP-πλήρες μέσω του θεωρήματος του Cook και Levin , είναι το πρόβλημα της ικανοποιησιμότητας. Στην εργασία μας χρησιμοποιούμε αναγωγές για να αποδείξουμε την NP-πληρότητα πολλών γνωστών προβλημάτων.

Τέλος ένα πολύ σημαντικό κομμάτι της πτυχιακής είναι οι προσεγγιστικοί αλγόριθμοι των NP-πλήρων προβλημάτων. Όσον αφορά τους προσεγγιστικούς αλγορίθμους είναι βασικό να κατανοήσουμε το πώς λειτουργούν. Ένας προσεγγιστικός αλγόριθμος χρησιμοποιείται για να παράγει μία λύση κοντά στο βέλτιστο μιας και για τα προβλήματα της NP-πλήρους κλάσης δεν υπάρχει πολυωνυμικός αλγόριθμος που να παράγει την βέλτιστη λύση. Έτσι η λύση που επιστρέφει ο προσεγγιστικός αλγόριθμος είναι μία λύση κοντά στο βέλτιστο που όμως είναι ικανοποιητική. Επίσης υπάρχει το σφάλμα προσέγγισης ε, που είναι η διαφορά της λύσης που επιστρέφει ο προσεγγιστικός αλγόριθμος από την βέλτιστη λύση που θα έπρεπε να επιστρέφει. Έτσι λέμε ότι ένα αλγόριθμος, για παράδειγμα, είναι 2-προσεγγιστικός αλγόριθμος (ή προσεγγιστικός αλγόριθμος με λόγο προσέγγισης 2), όταν ο λόγος της βέλτιστης λύσης από την λύση που επιστρέφει είναι 2.

ΠΑΡΑΡΤΗΜΑ

1. ΥΛΟΠΟΙΗΣΗ ΠΡΟΣΕΓΓΙΣΤΙΚΟΥ ΑΛΓΟΡΙΘΜΟΥ ΓΙΑ ΤΟ ΠΡΟΒΛΗΜΑ VERTEX-COVER

Σε αυτό το σημείο υλοποιούμε τον προσεγγιστικό αλγόριθμο που αναφέραμε στο κεφάλαιο 6(σελίδα 90) για το πρόβλημα Vertex-Cover. Ο αλγόριθμος που υλοποιήσαμε βρίσκει μία ακριβής επικάλυψη για τον δοσμένο μη κατευθυνόμενο γράφο του σχήματος 6.3(σελίδα 91). Όπως αναφέρουμε και στην αντίστοιχη ενότητα ο γράφος του σχήματος 6.3 (a) είναι ο αρχικός γράφος ενώ ο γράφος του σχήματος 6.3 (b) είναι ο γράφος που παράγεται από τον αλγόριθμο. Για την υλοποίηση του αλγορίθμου αναπαριστούμε τον γράφο με ένα πίνακα γειτνίασης με τιμές 1 και 0(1 εάν υπάρχει ακμή που ενώνει τους δύο αντίστοιχούς κόμβους, 0 ένα δεν υπάρχει που ενώνει τους δύο αντίστοιχούς κόμβους).

Στον αλγόριθμο εργαζόμαστε ως εξής. Ο αρχικός μας πίνακας είναι ο πίνακας ADJ. Τα στοιχεία του πίνακα ADJ τα αντιγράφουμε σε ένα πίνακα E στον οποίο και εργαζόμαστε. Ο πίνακας C είναι ο τελικός μονοδιάστατος πίνακας που παράγει ο αλγόριθμοσ(δηλαδή περιέχει την ακριβή επικάλυψη που κατασκευάστηκε).Ο βρόχος κάθε φορά που επιλέγει μία ακμή από τον E, προσθέτει τις αντίστοιχες κορυφές στο C και στην συνέχεια αφαιρεί από τον E όλες τις άλλες ακμές που προσπίπτουν σε αυτές τις δύο κορυφές που πρόσθεσε στον C.(για περισσότερες λεπτομέρειες για την ακριβή επικάλυψη αλλά και για το πώς λειτουργεί ο αντίστοιχος προσεγγιστικός αλγόριθμος ανατρέξτε στο κεφάλαιο 6, ενότητα 6.3 σελίδα 90).

Ακολουθεί ο αλγόριθμος του προγράμματος με ορισμένα σχόλια.
Είναι γραμμένος σε γλώσσα C++.
Σημείωση 1: παρατηρήθηκε ότι ο πίνακας ADJ είναι τριγωνικός. Έτσι δεν χρησιμοποιούνται στον αλγόριθμο τα στοιχεία ADJ[i][j] όπου $i >= j$.
Σημείωση 2: στον κύριο βρόχο, σε κάθε επανάληψη εντοπίζεται η «τελευταία» ακμή στον πίνακα E. Με απλή εντολή η επιλογή μπορεί για παράδειγμα να γίνεται με τυχαίο τρόπο κ.τ.λ.

```
#include <stdlib.h>
#include <stdio.h>
#define N 7

int main()
{
//Δήλωση μεταβλητών
 int i, j, x, y ; int ADJ[N][N], E[N][N], C[N];
// Αρχικοποίησε τα στοιχεία του πίνακα ADJ (δισδιάστατος πίνακας με 0 ή 1 για κάθε
ακμή, όπου αν το (i, j) είναι 1 αυτό σημαίνει ότι ...
//οι κορυφές i και j συνδέονται με ακμή.
ADJ[0][0] = 0; ADJ[0][1] = 1; ADJ[0][2] = 0; ADJ[0][3] = 0; ADJ[0][4] = 0; ADJ[0][5] =
0; ADJ[0][6] = 0;
ADJ[1][0] = 1; ADJ[1][1] = 0; ADJ[1][2] = 1; ADJ[1][3] = 0; ADJ[1][4] = 0; ADJ[1][5] =
0; ADJ[1][6] = 0;
ADJ[2][0] = 0; ADJ[2][1] = 1; ADJ[2][2] = 0; ADJ[2][3] = 1; ADJ[2][4] = 1; ADJ[2][5] =
0; ADJ[2][6] = 0;
ADJ[3][0] = 0; ADJ[3][1] = 0; ADJ[3][2] = 1; ADJ[3][3] = 0; ADJ[3][4] = 1; ADJ[3][5] =
1; ADJ[3][6] = 1;
ADJ[4][0] = 0; ADJ[4][1] = 0; ADJ[4][2] = 1; ADJ[4][3] = 1; ADJ[4][4] = 0; ADJ[4][5] =
1; ADJ[4][6] = 0;
ADJ[5][0] = 0; ADJ[5][1] = 0; ADJ[5][2] = 0; ADJ[5][3] = 1; ADJ[5][4] = 1; ADJ[5][5] =
0; ADJ[5][6] = 0;
ADJ[6][0] = 0; ADJ[6][1] = 0; ADJ[6][2] = 0; ADJ[6][3] = 1; ADJ[6][4] = 0; ADJ[6][5] =
0; ADJ[6][6] = 0;
// Αρχικοποίησε τον C (μονοδιάστατος πίνακας N με 0 ή 1 για κάθε κορυφή)
 for(i=0; i<=N-1; i++) C[i] = 0;
// Αντέγραψε στοιχεία του ADJ στον πίνακα E (δισδιάστατοι NxN με 0 ή 1 για κάθε ακμή
του γράφου)
 for(i=0; i<=N-1; i++) for(j=0; j<i; j++) E[i][j] = ADJ[i][j];
//ΚΥΡΙΟΣ ΒΡΟΧΟΣ ΠΡΟΓΡΑΜΜΑΤΟΣ
 do{
//Βρές μια ακμή (x, y) η οποία δεν έχει αφαιρεθεί από τον πίνακα E
 x=-1; y=-1;
 for(i=0; i<=N-1; i++)
 for(j=0; j<i; j++)
 if(E[i][j]==1) {x=i; y=j;}
//Αν βρέθηκε ακμή ...
 if((x!=-1)&&(y!=-1)) {
```

```
C[x]=1; C[y]=1; //...πρόσθεσε στον C τις κορυφές x και y
for(j=0;j<x;j++) E[x][j]=0;//...αφαίρεσε από τον E τις ακμές από τη x
for(i=0;i<=N-1;i++) E[i][x]=0;//...αφαίρεσε από τον E τις ακμές προς τη x
for(j=0;j<y;j++) E[y][j]=0;//...αφαίρεσε από τον E τις ακμές από τη y
for(i=0;i<=N-1;i++) E[i][y]=0;//...αφαίρεσε από τον E τις ακμές προς τη y
}
 }while((x!=-1)&&(y!=-1)); //αν δεν βρέθηκε ακμή σταμάτα
//Εκτύπωσε το σύνολο C
 printf("EKTYPOSI DEDOMENON SYNOLOU C (AKRIVIS
EPIKALYPSI)\n");
 for(i=0; i<=N-1; i++)
 printf("C[%d] = %d \n", i, C[i] );

}
```

Έτσι το αποτέλεσμα που βγάζει ο αλγόριθμος αν τον τρέξουμε είναι το σύνολο δεδομένων(ακριβής επικάλυψη) C:
C[0]=0
C[1]=1
C[2]=1
C[3]=1
C[4]=1
C[5]=1
C[6]=1

2. ΜΙΑ ΛΙΣΤΑ ΑΠΟ NP-ΠΛΗΡΗ ΠΡΟΒΛΗΜΑΤΑ

Παρακάτω αναφέρονται πολλά σημαντικά προβλήματα τα οποία ανήκουν στην NP-πλήρη κλάση[**7**],[**29**]

1. ΤΟ ΠΡΟΒΛΗΜΑ **ΙΣΟΜΟΡΦΙΣΜΟΣ ΥΠΟΓΡΑΦΗΜΑΤΟΣ (SUBGRAPH ISOMORPHISM)** ΕΙΝΑΙ ΝΠ-ΠΛΗΡΕΣ:

Στιγμιότυπο: Μη κατευθυντικά γραφήματα $G = (V_1, E_1)$ και $H = (V_2, E_2)$.
Ερώτηση: Περιέχει το G υπογράφημα ισομορφικό του H ·
Θα λέμε ότι ένα γράφημα G είναι ισομορφικό του H εάν υπάρχουν $V \subseteq V_1$ και $E \subseteq E_1$ τέτοια ώστε $V = |V_2|$ και $E = |E_2|$, και υπάρχει μία ένα- προς -ένα απεικόνιση $f : V_2 \to V$ τέτοια ώστε $(u, v) \in E_2$ αν και μόνο αν $(f(u), f(v)) \in E$.

2.ΤΟ ΠΡΟΒΛΗΜΑ **ΣΠΟΝΔΥΛΩΤΟ ΔΕΝΔΡΟ ΠΕΡΙΟΡΙΣΜΕΝΟΥ ΒΑΘΜΟΥ (DEGREE CONSTRAINED SPANNING TREE)** ΕΙΝΑΙ NP-ΠΛΗΡΕΣ:
Στιγμιότυπο: Μη κατευθυντικό γράφημα $G = (V, E)$ και θετικός ακέραιος $k \leq |V|$.
Ερώτηση: Υπάρχει σπονδυλωτό δένδρο (δένδρο το οποίο περιλαμβάνει όλους τους κόμβους του G) για το G στο οποίο κανένας κόμβος δεν έχει βαθμό μεγαλύτερο από k ·

3.ΤΟ ΠΡΟΒΛΗΜΑ **ΣΥΝΟΛΟ ΚΡΟΥΣΗΣ (HITTING SET)** ΕΙΝΑΙ NP-ΠΛΗΡΕΣ:
Στιγμιότυπο: Σύνολο C υποσυνόλων ενός συνόλου S και θετικός ακέραιος k.
Ερώτηση: Υπάρχει υποσύνολο $S' \subseteq S$ με $|S'| \leq k$ τέτοιο ώστε το S' να περιέχει το λιγότερο ένα στοιχείο από κάθε υποσύνολο στο C ·

4. ΤΟ ΠΡΟΒΛΗΜΑ ΤΗΣ **ΙΚΑΝΟΠΟΙΗΣΙΜΟΤΗΤΑΣ(SAT)** ΕΙΝΑΙ NP-ΠΛΗΡΕΣ:
Στιγμιότυπο: Σύνολο U από μεταβλητές και συλλογή C από clauses .
Ερώτηση: Υπάρχει ανάθεση αληθοτιμών για τις μεταβλητές της C ·

5.ΤΟ ΠΡΟΒΛΗΜΑ ΤΟΥ **3SAT** ΕΙΝΑΙ NP-ΠΛΗΡΕΣ:
Στιγμιότυπο: Σύνολο U από μεταβλητές και συλλογή C από clauses τέτοιο ώστε κάθε clause $c \in C$ να έχει $|c| = 3$.
Ερώτηση: Υπάρχει ανάθεση αληθοτιμών για τις μεταβλητές της C ·

6. ΤΟ ΠΡΟΒΛΗΜΑ **NOT-ALL-EQUAL** ΕΙΝΑΙ NP-ΠΛΗΡΕΣ:
Στιγμιότυπο: Στιγμιότυπο του 3SAT .
Ερώτηση: Υπάρχει αποτίμηση στις λογικές μεταβλητές έτσι ώστε κάθε πρόταση να έχει το λιγότερο ένα αληθές και το λιγότερο ένα ψευδές στοιχείο ·

7. ΤΟ ΠΡΟΒΛΗΜΑ **ΜΕΓΙΣΤΗ 2- ΙΚΑΝΟΠΟΙΗΣΙΜΟΤΗΤΑ (MAX2SAT)** ΕΙΝΑΙ NP-ΠΛΗΡΕΣ:
Στιγμιότυπο: Λογική έκφραση σε CNF όπου κάθε πρόταση έχει δύο στοιχεία (όπως στο 2SAT) και ακέραιος k.
Ερώτηση: Υπάρχει αποτίμηση στις μεταβλητές που να ικανοποιεί το λιγότερο k προτάσεις ·

8. ΤΟ ΠΡΟΒΛΗΜΑ **ΤΟΜΗ ΣΥΝΟΛΟΥ (SET SPLITTING)** ΕΙΝΑΙ NP-ΠΛΗΡΕΣ:
Στιγμιότυπο: Σύνολο C πεπερασμένων υποσυνόλων ενός συνόλου S.

Ερώτηση: Υπάρχει διαμέριση του S σε δύο σύνολα S_1 και S_2 έτσι ώστε κανένα από τα σύνολα του C να μην ανήκει πλήρως είτε στο S_1 είτε στο S_2 ·

9. ΤΟ ΠΡΟΒΛΗΜΑ **ΑΚΡΙΒΗΣ ΕΠΙΚΑΛΥΨΗ (VERTEX COVER)** ΕΙΝΑΙ NP-ΠΛΗΡΕΣ:
Στιγμιότυπο: Γράφημα $G = (V, E)$ και θετικός ακέραιος $K \leq |V|$.
Ερώτηση: Υπάρχει ένα υποσύνολο $V' \subseteq V$ έτσι ώστε αν (u, v) είναι μία ακμή του G, τότε ή $u \in V'$ ή $v \in V'$ ·

10. ΤΟ ΠΡΟΒΛΗΜΑ **ΚΥΡΙΑΡΧΟ ΣΥΝΟΛΟ (DOMINATING SET)** ΕΙΝΑΙ NP-ΠΛΗΡΕΣ:
Στιγμιότυπο: Μη κατευθυντικό γράφημα $G = (V, E)$ και θετικός ακέραιος $k \leq |V|$.
Ερώτηση: Έχει το G κυρίαρχο σύνολο μεγέθους k ή μικρότερο ·
Κυρίαρχο σύνολο ενός γραφήματος $G = (V, E)$ είναι ένα υποσύνολο κόμβων $V' \subseteq V$ τέτοιο ώστε για κάθε $u \in V \backslash V'$ υπάρχει $v \in V'$ για το οποίο $(u, v) \in E$.

11. ΤΟ ΠΡΟΒΛΗΜΑ **K- ΧΡΩΜΑΤΙΚΟΤΗΤΑ (K- COLORABILITY)** ΕΙΝΑΙ NP-ΠΛΗΡΕΣ:

Στιγμιότυπο: Μη κατευθυντικό γράφημα $G = (V, E)$ και θετικός ακέραιος $k \leq |V|$.
Ερώτηση: Είναι το G $k-$χρωματικό ·

12. ΤΟ ΠΡΟΒΛΗΜΑ **FEEDBACK NODE SET** ΕΙΝΑΙ NP-ΠΛΗΡΕΣ:
Στιγμιότυπο: Μη κατευθυντικό γράφημα $G = (V, E)$ και θετικός ακέραιος $k \leq |V|$.
Ερώτηση: Υπάρχει υποσύνολο κόμβων $V' \subseteq V$ με $|V'| \leq k$ το οποίο περιέχει τουλάχιστον έναν κόμβο από κάθε κετευθυντικό κύκλο του G ·

13. ΤΟ ΠΡΟΒΛΗΜΑ ΤΗΣ **ΚΛΙΚΑΣ (CLIQUE PROBLEM)** ΕΙΝΑΙ NP-ΠΛΗΡΕΣ:
Στιγμιότυπο: Γράφημα $G = (V, E)$ και θετικός ακέραιος $K \leq |V|$.
Ερώτηση: Περιέχει το G μία κλίκα μεγέθους K ή παραπάνω, για παράδειγμα ένα υποσύνολο $V' \subseteq V$ με $|V'| \geq K$ τέτοιο ώστε κάθε δύο κορυφές στο V' να συνδέονται με μία ακμή στο E·

14. ΤΟ ΠΡΟΒΛΗΜΑ **ΑΝΕΞΑΡΤΗΤΟ ΣΥΝΟΛΟ (INDEPENDENT SET)** ΕΙΝΑΙ NP-ΠΛΗΡΕΣ:
Στιγμιότυπο: Γράφημα $G = (V, E)$ και θετικός ακέραιος $K \leq |V|$.
Ερώτηση: Περιέχει το G ένα ανεξάρτητο σύνολο μεγέθους K ή παραπάνω, για παράδειγμα ένα υποσύνολο $V' \subseteq V$ με $|V'| \geq K$ τέτοιο ώστε δύο κορυφές στο V' να μην συνδέονται από μία ακμή στο E ·

15. ΤΟ ΠΡΟΒΛΗΜΑ **ΜΟΝΟΠΑΤΙ HAMILTON (HAMILTON PATH)** ΕΙΝΑΙ ΝΡ-ΠΛΗΡΕΣ:
Στιγμιότυπο: Μη κατευθυντικό γράφημα $G = (V, E)$ και δύο κόμβοι $u, v \in V$.
Ερώτηση: Έχει το G μονοπάτι Hamilton από τον u στον v ·

16. ΤΟ ΠΡΟΒΛΗΜΑ ΤΟΥ **ΠΕΡΙΟΔΕΥΟΝΤΟΣ ΠΩΛΗΤΗ (TRAVELING SALESMAN PROBLEM – ή TSP)** ΕΙΝΑΙ ΝΡ-ΠΛΗΡΕΣ:
Στιγμιότυπο: Σύνολο C από m πόλεις, απόσταση $d(c_i, c_j) \in Z^+$ για κάθε ζεύγος πόλης $c_i, c_j \in C$, θετικός ακέραιος B.
Ερώτηση: Υπάρχει μία διαδρομή από το C με μήκος B ή λιγότερο, για παράδειγμα υπάρχει μία μετάθεση $\langle c_{\pi(;;)}, c_{\pi(;;)}, ..., c_{\pi(m)} \rangle$ του C τέτοια ώστε $\left[\sum_{i=1}^{m-1} d(c_{\pi(i)}, c_{\pi(i+1)})\right] + d(c_{\pi(m)}, c_{\pi(;;)}) \leq B$ ·

17. ΤΟ ΠΡΟΒΛΗΜΑ **ΜΕΓΙΣΤΗ ΤΟΜΗ (MAX CUT)** ΕΙΝΑΙ ΝΡ-ΠΛΗΡΕΣ:
Στιγμιότυπο: Μη κατευθυντικά γραφήματα $G = (V, E)$, βάρος $w(e)$ για κάθε $e \in E$ και θετικός ακέραιος k.
Ερώτηση: Υπάρχει διαμέριση του σε ξένα μεταξύ τους σύνολα V_1 και V_2 έτσι ώστε το άθροισμα των βαρών των ακμών του E που έχουν ένα άκρο τους στο V_1 και το άλλο τους άκρο στο V_2 να είναι τουλάχιστον k ·

18. ΤΟ ΠΡΟΒΛΗΜΑ **ΤΡΙΣΔΙΑΣΤΑΤΟ ΤΑΙΡΙΑΣΜΑ (THREE DIMENSIONAL MATCHING ή 3DM)** ΕΙΝΑΙ ΝΡ-ΠΛΗΡΕΣ:
Στιγμιότυπο: Τρία ξένα σύνολα A, K και Σ με $|\mathrm{A}| = |\mathrm{K}| = |\Sigma|$ και υποσύνολο Ε του Καρτεσιανού γινομένου $\mathrm{A} \times \mathrm{K} \times \Sigma$.
Ερώτηση: Υπάρχει $\mathrm{M} \subseteq \mathrm{E}$ με $|\mathrm{M}| = |\mathrm{A}|$ με όλες τις τριάδες στο Μ ξένες ανά δύο μεταξύ τους και στις τρεις συντεταγμένες ·

19. ΤΟ ΠΡΟΒΛΗΜΑ **SET PACKING** ΕΙΝΑΙ ΝΡ-ΠΛΗΡΕΣ:
Στιγμιότυπο: Σύνολο C πεπερασμένων συνόλων και θετικός ακέραιος k.
Ερώτηση: Υπάρχει στο C το λιγότερο k σύνολα ξένα ανά δύο μεταξύ τους ·
(υπόδειξη: με αναγωγή στο X3C*)*

20. ΤΟ ΠΡΟΒΛΗΜΑ **ΔΙΑΜΕΡΙΣΜΟΣ (PARTITION)** ΕΙΝΑΙ ΝΡ-ΠΛΗΡΕΣ:
Στιγμιότυπο: Πεπερασμένο σύνολο A και μέγεθος $s(a) \in Z^+$.
Ερώτηση: Υπάρχει υποσύνολο $A' \subseteq A$ τέτοιο ώστε $\sum_{a \in A'} s(a) = \sum_{a \in A \setminus A} s(a)$ ·

21. ΤΟ ΠΡΟΒΛΗΜΑ **3-ΔΙΑΜΕΡΙΣΜΟΣ (3-PARTITION)** ΕΙΝΑΙ ΝΡ-ΠΛΗΡΕΣ:
Στιγμιότυπο: σύνολο A με $3m$ στοιχεία, θετικός ακέραιος B και μέγεθος $s(a)$ για κάθε $a \in A$ έτσι ώστε $B/4 \leq s(a) \leq B/2$ και $\sum_{a \in A} s(a) = mB$.

Ερώτηση: Υπάρχει διαμερισμός του A σε m τριμελή σύνολα έτσι ώστε το άθροισμα των βαρών σε κάθε σύνολο να είναι B ·
(*υπόδειξη:* με αναγωγή στο 3DM)

22.ΤΟ ΠΡΟΒΛΗΜΑ **ΑΘΡΟΙΣΜΑ ΥΠΟΣΥΝΟΛΩΝ (SUBSET SUM)** ΕΙΝΑΙ ΝΡ- ΠΛΗΡΕΣ:
Στιγμιότυπο: Πεπερασμένο σύνολο A και μέγεθος $s(a) \in Z^+$ και θετικός ακέραιος B.
Ερώτηση: Υπάρχει υποσύνολο $A' \subseteq A$ τέτοιο ώστε το άθροισμα του μεγέθους των στοιχείων στο A' να είναι ακριβώς B ·

23. ΤΟ ΠΡΟΒΛΗΜΑ **ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ ΠΟΛΛΩΝ ΕΠΕΞΕΡΓΑΣΤΩΝ (MULTIPROCESSOR SCHEDULING)** ΕΙΝΑΙ ΝΡ-ΠΛΗΡΕΣ:
Στιγμιότυπο: Σύνολο T εργασιών, αριθμός $m \in Z^+$ επεξεργαστών, διάρκεια $l(t) \in Z^+$ για κάθε εργασία και προθεσμία $D \in Z^+$.
Ερώτηση: Υπάρχει προγραμματισμός των m επεξεργαστών έτσι ώστε όλες οι εργασίες να τελειώνουν πριν την προθεσμία D ·
Προγραμματισμός εργασιών είναι μία συνάρτηση $\sigma : T \to Z_0^+$, έτσι ώστε για κάθε $u \geq 0$, ο αριθμός των εργασιών $t \in T$ για τις οποίες $\sigma(t) \leq u \leq \sigma(t) + l(t)$ είναι μικρότερος ή ίσος του m και έτσι ώστε για κάθε $t \in T, \sigma(t) + l(t) \leq D$.

24. ΤΟ ΠΡΟΒΛΗΜΑ **BIN-PACKING** ΕΙΝΑΙ ΝΡ-ΠΛΗΡΕΣ:
Στιγμιότυπο: n θετικοί αριθμοί $\omega_1, \omega_2, ..., \omega_n$ και δύο αριθμοί C και K.
Ερώτηση: μπορούν οι αριθμοί αυτοί να χωριστούν σε K σύνολα, καθένα από τα οποία περιέχει αριθμούς με άθροισμα το πολύ C ·

25. ΤΟ ΠΡΟΒΛΗΜΑ ΤΟΥ **ΣΑΚΙΔΙΟΥ (KNAPSACK)** ΕΙΝΑΙ ΝΡ-ΠΛΗΡΕΣ:
Στιγμιότυπο: Σύνολο $S = \{a_1, ..., a_n\}$ από μη αρνητικούς ακεραίους, και ακέραιος K, όλοι σε δυαδική μορφή.
Ερώτηση: Υπάρχει υποσύνολο $P \subseteq S$ τέτοιο ώστε $\sum_{a_i \in P} a_i = K$ ·

Βιβλιογραφία

[1] H. R. Lewis , X. X. Παπαδημητρίου, Στοιχεία Θεωρίας Υπολογισμού, Εκδόσεις Κριτική, Επιστημονική Βιβλιοθήκη, 2005,p 421-325, 329-367, 369-470.

[2] H. R. Lewis , C. H. Papadimitriou, Elements of the Theory of Computation, Second Edition, Prentice-Hall, 1998

[3] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, Clifford Stein, Introduction to algorithms, The MIT Press; Second Edition, Cambridge Massachusetts, London,England, 2001, p 966-1043.

[4] Michael Sipser, Introduction to the Theory of Computation, Second Edition, International Edition, Thomson Course Technology, Copyright ©2006, p 31-81,167-215, 251-332.

[5] Michael Sipser, Εισαγωγή στη Θεωρία Υπολογισμού, Πανεπιστημιακές Εκδόσεις Κρήτης, 2007

[6] Christos H. Papadimitriou, Computational Complexity, Addison-Wesley Publishing Company, Inc., University of California-San Diego, 1994, p. 19-56, 73-86, 139-218.

[7] Michael R. Garey and David S. Johnson, Computers and Intractability : *A Guide to the Theory of NP-Completeness.* W.H. Freeman and Company, StateplaceNew York, 1979, p. 17-76, 121-186, 187-285.

[8] V ijay. V. Vazirani. Approximation Algorithms. Springer, 1999, p.1-10, 68-73

[9] C.H. Papadimitriou, V ijay. V. Vazirani, S. Dasguta, Algorithms, Copyright ©2006

[10]Martin Davis, Ron Sigal, Elaine Weyuker, Computability, Complexity and Languages, Academic Press, 1994

[11] J. E. Hopcroft, R. Motwani, J. D. Ullman, Introduction to Automata Theory, Languages, and Computation, Addison-Wesley, 2nd Ed. (2000)

[12] C. Papadimitriou and K. Steiglitz. Combinatorial Optimization: Algorithms and Complexity. Prentice Hall, CityEnglewood Cliffs, placeStateNew Jersey, 1982.

[13] Cormen, Leiserson, Rivest, and Stein. Introduction to Algorithms, The MIT Press; 2nd edition, 2001

[14] Φ. Αφράτη, Γ. Παπαγεωργίου, Αλγόριθμοι: Μέθοδοι σχεδίασης και ανάλυση πολυπλοκότητας, Εκδόσεις Συμμετρία, Αθήνα (1993).

[15] D.S. Hochbaum, Approximation Algorithms for NP-Hard Problems, PWS Publishing Company, 1997

[16] Bernand M. Moret. The Theory of Computation, Addison-Wesley Longman, Inc., 1998

[17] A. V. Aho, J. placeE. Hopcroft, J. D. Ullman. The Design and Analysis of Computer Algorithms, Addison- Wesley, 1974

[18] D. Knuth. The Art of Computer Programming Volume 3: Sorting and Searching. Addition-Wesley Publishing Company, Inc., 1973.

[19] A. M Turing .On Computable Numbers, with an Application to the Entscheidiungsproblem, Proc. London Mathematical Society, 2, 42 pp. 230-265 and no.43, pp 544-546

[20] S.C. Kleene, Introduction to Metamathematics, D. Von Nostrand, 1952

[21] W.J. Savitch. Relationships between nondeterministic and deterministic tape complexities. J. Computer and systems Sciences 4:2, 1970.

[22] R. E. Lander, N. A. Lynch, A. L. Selman .A comparison of polynomial time reductions. Theoretical Computer Science, 1975

[23] S. A. Cook. The complexity of theorem- proving procedures. Proc of the 3rd IEEE Symposium on the Foundations of Computer Science, 151-158, 1971

[24] L. A. Levin. *Universal sorting problems.* Problems of Information Transmission, 9, 265-266, 1973

[25] R. M. Karp. *Reducibility among combinatorial problems.* Complexity of Computer Computation, J. W. Thatcher and R. E. Miller eds. 85-103, Plenum Press, StateplaceNew York 1972

[26] L. Lovasz, M. D. Plummer. Matching Theory. Annals of Discrete Mathematics, Vol 29, North-Holland, 1986

[27] C. H. Papadimitriou. *On the complexity of integer programming.* J. ACM, 28, 2, 1981

[28] A. Schrijer. Theory of Linear and Integer Programming. Wiley, StateplaceNew York, 1986

[29] D.S. Johnson. The NP-Completeness column *A on- going guide* . J. of Algorithms 4, 1981