

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΑΤΡΩΝ – ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ
ΤΜΗΜΑ ΜΑΘΗΜΑΤΙΚΩΝ

Πανεπιστημιακές παραδόσεις στο μάθημα

ΥΠΟΛΟΓΙΣΤΙΚΗ ΠΟΛΥΠΛΟΚΟΤΗΤΑ

(Δεύτερη Έκδοση)

Δημήτριος Ι. Καββαδίας
Επίκουρος Καθηγητής

Πάτρα, Νοέμβριος 2004

Πρόλογος

Οι σημειώσεις αυτές περιλαμβάνουν το μεγαλύτερο μέρος της ύλης του προπτυχιακού μαθήματος «Υπολογιστική Πολυπλοκότητα» που διδάσκεται στο 4ο έτος του Τμήματος Μαθηματικών του Πανεπιστημίου Πατρών. Η διάρθρωση τους είναι η ακόλουθη:

Στο πρώτο κεφάλαιο δίδεται μία εισαγωγή στις έννοιες της πολυπλοκότητας αλγορίθμων και προβλημάτων για διάφορα μέτρα πολυπλοκότητας. Το Κεφάλαιο 2 περιγράφει σε βάθος το βασικό υπολογιστικό μοντέλο που χρησιμοποιείται, τη μηχανή Turing, και τις διάφορες παραλλαγές του. Δίδονται ορισμοί της πολυπλοκότητας χρόνου και χώρου αλγορίθμων και προβλημάτων. Το επόμενο κεφάλαιο είναι αφιερωμένο στις γενικές σχέσεις μεταξύ κλάσεων πολυπλοκότητας. Σε αυτό περιλαμβάνονται όλες οι γνωστές σχέσεις όταν δεν υπάρχουν περιορισμοί στην συνάρτηση πολυπλοκότητας. Το Κεφάλαιο 4 αναφέρει τις πολύ κρίσιμες έννοιες της αναγωγής και της πληρότητας παρουσιάζοντας πλήρη προβλήματα για τις κλάσεις NL, P και NP. Τέλος το Κεφάλαιο 5 είναι αφιερωμένο στην NP-πληρότητα. Παρουσιάζεται πληθώρα προβλημάτων και αποδεικνύεται η πληρότητα τους για την κλάση NP.

Είμαι ευγνώμων στον κ. Ηλία Κ. Σταυρόπουλο, Διδάκτορα του Τμήματος Μηχανικών Ηλεκτρονικών Υπολογιστών & Πληροφορικής του Πανεπιστημίου Πατρών, για την ουσιαστική του βοήθεια. Σε αυτόν οφείλεται το μεγαλύτερο μέρος του Κεφαλαίου 5 και η επιμέλεια όλων των σχημάτων και της παρουσίασης.

Δημήτρης Ι. Καββαδίας
Πάτρα, Νοέμβριος 2004

στον Γιαννάκη

Περιεχόμενα

Πρόλογος	i
Περιεχόμενα	v
1 Εισαγωγή	1
1.1 Αλγόριθμοι και Πολυπλοκότητα	2
1.2 Πολυπλοκότητα Προβλημάτων	8
2 Η Μηχανή Turing	11
2.1 Περιγραφή – Ορισμός	12
2.2 Παράσταση και μέγεθος προβλημάτων	19
2.3 Παραλλαγές της Μηχανής Turing	22
2.3.1 Η μη ντετερμινιστική μηχανή Turing	23
2.3.2 Η καθολική μηχανή Turing	26
2.4 Υπολογισμοί με περιορισμούς	30
2.4.1 Θεωρήματα επιτάχυνσης και συμπίεσης της μνήμης	32
2.4.2 Μείωση στον αριθμό των ταινιών	33
2.5 Η Μηχανή Τυχαίας Προσπέλασης, RAM	35
2.5.1 Περιγραφή-Ορισμός	35
2.5.2 Ισοδυναμία RAM-Μηχανής Turing	38
2.6 Κλάσεις πολυπλοκότητας	41
2.7 Ασκήσεις	45
3 Ιδιότητες των κλάσεων πολυπλοκότητας	47
3.1 Η συνάρτηση πολυπλοκότητας	47
3.2 Θεωρήματα Ιεραρχίας	49
3.3 Θεωρήματα Χάσματος	51
3.4 Σχέσεις μεταξύ κλάσεων πολυπλοκότητας	53
3.4.1 Μη ντετερμινιστικός χρόνος	53
3.4.2 Μη ντετερμινιστικός χώρος	54
3.5 Ασκήσεις	62

4	Πληρότητα	63
4.1	Αναγωγές	64
4.2	Πληρότητα	66
4.3	Μερικά πλήρη προβλήματα	68
5	NP-πλήρη προβλήματα	79
5.1	Παραλλαγές της ΙΚΑΝΟΠΟΙΗΣΙΜΟΤΗΤΑΣ	79
5.2	Προβλήματα από τη Θεωρία Γραφημάτων	86
5.3	Άλλα προβλήματα	96
5.4	Ασθενής NP-πληρότητα	100
5.5	Αποδεικνύοντας NP-πληρότητα	105
5.6	Ασκήσεις	106
	Βιβλιογραφία	111

Κεφάλαιο 1

Εισαγωγή

Αντικείμενο της Υπολογιστικής Πολυπλοκότητας είναι τα υπολογιστικά προβλήματα καθώς και οι ενδογενείς ιδιότητες τους που τα κάνουν να επιλύονται ή να μην επιλύονται αποτελεσματικά. Σκοπός αυτού του κεφαλαίου είναι να εξηγήσει το αντικείμενο της Υπολογιστικής Πολυπλοκότητας δίνοντας έμφαση στην διάκριση μεταξύ της Θεωρίας Πολυπλοκότητας και της Θεωρίας Αλγορίθμων.

Ένα υπολογιστικό πρόβλημα είναι μια ερώτηση η οποία περιγράφεται από ένα σύνολο παραμέτρων ή αλλιώς ελεύθερων μεταβλητών καθώς και από την σχέση που πρέπει να έχει η λύση του με τις μεταβλητές εισόδου. Είναι σημαντική η διάκριση μεταξύ του προβλήματος και ενός στιγμιότυπου του: Το στιγμιότυπο προκύπτει από μια απόδοση συγκεκριμένων τιμών στις παραμέτρους του προβλήματος.

Σαν παράδειγμα ας θεωρήσουμε το πολύ χρήσιμο στην Πληροφορική και εξαιρετικά μελετημένο πρόβλημα της ταξινόμησης (sorting). Παράμετρος αυτού του προβλήματος είναι ένα διάνυσμα n πραγματικών αριθμών $I = (a_1, a_2, \dots, a_n)$. Λύση στο πρόβλημα της Ταξινόμησης είναι το «ταξινομημένο» διάνυσμα $I' = (a'_1, a'_2, \dots, a'_n)$ με συντεταγμένες τους ίδιους αριθμούς αλλά σε αύξουσα σειρά: $a'_i \leq a'_{i+1}, i = 1, \dots, n - 1$.

Ένα στιγμιότυπο του προβλήματος της Ταξινόμησης είναι π.χ. για $n = 4$ το διάνυσμα $I = (7, 3, 12, -2)$. Σε αυτή την περίπτωση η λύση είναι το διάνυσμα $I' = (-2, 3, 7, 12)$.

Η επίλυση ενός υπολογιστικού προβλήματος απαιτεί την σχεδίαση ενός αλγορίθμου ή αλλιώς μιας σαφούς βήμα προς βήμα διαδικασίας που να οδηγεί στην λύση σε πεπερασμένο αριθμό βημάτων.

Παρόλο που η ανάπτυξη τεχνικών σχεδιασμού και μελέτη των αλγορίθμων δεν είναι κύριο αντικείμενο του βιβλίου αυτού, οι αλγόριθμοι είναι η άλλη πλευρά (θετική;) της μελέτης των υπολογιστικών προβλημάτων και πολλές φορές θα χρειαστεί να περιγράψουμε αλγορίθμους για κάποιο πρόβλημα. Για

λόγους σαφήνειας και απλότητας θα απαιτούμε οι αλγόριθμοι μας να συντάσσονται από επιμέρους βήματα που είναι συνήθη σε απλά προγράμματα υπολογιστών. Όπου η απαίτηση συντομίας το επιβάλλει, θα ομαδοποιούμε επίσης συνθετότερες διεργασίες. Σε μια τέτοια περίπτωση όμως θα είναι κατανοητό ότι η ανάλυση σε πραγματικά «προγραμματιστικά» βήματα μπορεί να γίνει χωρίς πρόβλημα.

Για την περιγραφή των αλγορίθμων μας θα χρησιμοποιήσουμε μια ψευδο-γλώσσα προγραμματισμού με βασική δομή δανεισμένη από την Pascal ώστε να συνδυάζουμε την σαφήνεια και περιεκτικότητα με την συντομία. Ο λόγος που δεν χρησιμοποιούμε μια πραγματική γλώσσα προγραμματισμού είναι ότι θέλουμε να αποφύγουμε τις συντακτικές ιδιαιτερότητες μιας συγκεκριμένης γλώσσας και να επικεντρωθούμε στον καθαυτό αλγόριθμο. Σαν συνέπεια, δεν απαιτείται από τον αναγνώστη εξοικείωση με καμιά γλώσσα προγραμματισμού, παρά μόνο με τις βασικές αρχές του δομημένου προγραμματισμού για να κατανοήσει ένα αλγόριθμο. Εξάλλου όπως ήδη αναφέρθηκε, πολλές φορές σχετικά πολύπλοκες διεργασίες θα περιγράφονται στην φυσική γλώσσα ώστε να αναδεικνύεται η ουσία του αλγορίθμου.

1.1 Αλγόριθμοι και Πολυπλοκότητα

Ας δώσουμε ένα παράδειγμα περιγράφοντας ένα πολύ κοινό αλγόριθμο για το πρόβλημα της Ταξινόμησης, τον λεγόμενο αλγόριθμο της Φυσαλίδας (bubble-sort).

```
procedure bubble_sort;  
1: input  $I$ ; {ένα  $n$ -διάστατο διάνυσμα αριθμών}  
2: for  $i := 1$  to  $n - 1$  do  
3:   for  $j := 1$  to  $n - i$  do  
4:     if  $I(j) > I(j + 1)$  then  
5:       ανταλλάξε τα  $I(j + 1)$  και  $I(j)$ ;  
6:     end if  
7:   end for  
8: end for  
9: return  $I$ ;
```

Αλγόριθμος 1: Ο αλγόριθμος της Φυσαλίδας.

Η λειτουργία αυτού του αλγορίθμου είναι πολύ απλή. Στην ουσία κάνει ότι και ένας άνθρωπος που θέλει να ταξινομήσει ένα σύνολο από n ανακατεμένα χαρτιά σύμφωνα με την αύξουσα σειρά σελίδων: Στην αρχή βρίσκει, με διαδοχικές συγκρίσεις με όλα τα υπόλοιπα, το χαρτί με το μεγαλύτερο

νούμερο και το τοποθετεί τελευταίο. Στην συνέχεια βρίσκει το μεγαλύτερο από τα υπόλοιπα $n - 1$ χαρτιά και το τοποθετεί προτελευταίο κ.ο.κ. μέχρι το μικρότερο. Δηλαδή το μεγαλύτερο με τις διαδοχικές αλλαγές θέσεων ανεβαίνει στην τελευταία θέση σαν φυσαλίδα. Στον Αλγόριθμο 1 αυτή η λειτουργία φαίνεται στον εσωτερικό βρόχο ανακύκλωσης (βήματα 3–7): σαρώνουμε όλα τα διαδοχικά ζευγάρια από το πρώτο ως το τελευταίο στοιχείο μετακινώντας το μεγαλύτερο στην θέση με τον μεγαλύτερο δείκτη. Προφανώς μετά το τέλος της σάρωσης στην τελευταία θέση θα βρεθεί το μεγαλύτερο από όλα τα στοιχεία. Το διάστημα που κάθε φορά σαρώνουμε είναι από το στοιχείο 1 έως το $n - i$ (στην αρχή $i = 1$ και το διάστημα είναι από το 1 έως το $n - 1$, μετά $i = 2$ και το διάστημα γίνεται αντίστοιχα 1 έως $n - 2$ κλπ). Η ορθότητα του αλγορίθμου είναι προφανής και συνεπώς η τελευταία γραμμή πράγματι θα επιστρέψει το διάνυσμα I ταξινομημένο.¹

Σαν πρώτο λοιπόν βήμα στην κατανόηση του αντικειμένου της Θεωρίας Πολυπλοκότητας ας προσπαθήσουμε να μελετήσουμε την πολυπλοκότητα του αλγορίθμου της Φυσαλίδας και όχι ακόμη του προβλήματος της Ταξινόμησης.

Κάπως γενικά μιλώντας, αυτό σημαίνει την αξιολόγηση του συγκεκριμένου αλγορίθμου, δηλαδή την εύρεση των ιδιοτήτων του που να μας επιτρέψουν να τον χαρακτηρίσουμε σαν καλό ή κακό. Καλό ή κακό όμως με βάση ποιο μέτρο; Και ακόμη γενικότερα, ποιές είναι τελικά οι ιδιότητες πως μας ενδιαφέρουν σε ένα αλγόριθμο; Σ' αυτή την ερώτηση ο συνηθισμένος χρήστης προγραμμάτων ηλεκτρονικού υπολογιστή (άρα και αλγορίθμων) απαντά: Η ταχύτητα. Πραγματικά η ταχύτητα (ο χρόνος) σε ένα αλγόριθμο είναι από τα πλέον επιθυμητά χαρακτηριστικά. Ακόμη και στην εποχή μας όπου η εξέλιξη των υπολογιστών έχει καταφέρει να κατασκευάσει μηχανές με εξαιρετικές επιδόσεις, τα καινούργια προβλήματα που καλούνται να λύσουν οι υπολογιστές κάνει την ταχύτητα της χρησιμοποιούμενης μεθόδου ίσως το πιο σημαντικό χαρακτηριστικό της. Σε καμμία όμως περίπτωση δεν είναι και το μοναδικό: Η μνήμη είναι μία άλλη παράμετρος που ενδιαφέρει σε ένα υπολογισμό. Όπως και ο χρόνος, έτσι και ο χώρος (η μνήμη) αποτελεί υπολογιστικό πόρο σε ανεπάρκεια και κατά συνέπεια ενδιαφέρουν οι επιδόσεις του αλγορίθμου μας ως προς αυτόν.

Γενικά πάντως το μέτρο της πολυπλοκότητας που μας ενδιαφέρει μπορεί να είναι αρκετά γενικό ή αρκετά ειδικό ανάλογα με τις ιδιαιτερότητες του προβλήματος και/ή της υπολογιστικής μηχανής που χρησιμοποιούμε. Για παράδειγμα σε ένα παράλληλο υπολογιστή (έναν υπολογιστή όπου πολλές μονάδες επεξεργασίας συνεργάζονται για την επίλυση του ίδιου προβλήματος) ένα στοιχείο που ενδιαφέρει είναι ο αριθμός των μηνυμάτων που ανταλλάσσονται

¹Ένας αλγόριθμος είναι ορθός αν είναι μερικώς ορθός (δηλαδή για τις εισόδους που τερματίζει παράγει το αναμενόμενο αποτέλεσμα) και τερματίζει για κάθε είσοδο.

μεταξύ των επεξεργαστών. Η πρακτική αιτία για ένα τέτοιο μέτρο πολυπλοκότητας είναι βασικά τεχνολογική: η επικοινωνία μεταξύ των επεξεργαστών είναι πολύ πιο χρονοβόρα από την καθαυτή επεξεργασία σε ένα υπολογιστή. Συνεπώς ο αριθμός των ανταλλασσόμενων μηνυμάτων κατά την εκτέλεση ενός αλγορίθμου αποτελεί καλή ένδειξη της ταχύτητας που θα πετύχει.

Ας προσπαθήσουμε τώρα να εκτιμήσουμε τις επιδόσεις, ως προς τον χρόνο, του Αλγόριθμου 1. Πιο συγκεκριμένα, ας προσπαθήσουμε να εκτιμήσουμε τον αριθμό των συγκρίσεων που κάνει όταν ταξινομεί n αριθμούς. Όπως εξηγήσαμε παραπάνω, κατά την εκτέλεση του εξωτερικού βρόχου ανακύκλωσης (βήματα 2–8) για πρώτη φορά θα γίνουν $n - 1$ συγκρίσεις, τη δεύτερη εκτέλεση $n - 2$, την τρίτη $n - 3$ κλπ, ενώ την τελευταία μία σύγκριση. Συνεπώς ο συνολικός αριθμός συγκρίσεων είναι $n(n - 1)/2$. Εδώ μπορεί κάποιος να αναρωτηθεί κατά πόσο ο αριθμός των συγκρίσεων είναι πραγματική ένδειξη για τον χρόνο που θα χρειαστεί ο αλγόριθμος μας να ταξινομήσει n αριθμούς. Πραγματικά, όπως μπορεί να παρατηρήσει ο αναγνώστης, τα συνολικά βήματα του αλγορίθμου εξαρτώνται και από το αποτέλεσμα των συγκρίσεων: στο τελευταίο βήμα η αντιμετάθεση των $I(j + 1)$ και $I(j)$ θα γίνει μόνο όταν το πρώτο είναι μικρότερο του δεύτερου και συνεπώς στην περίπτωση αυτή επιπλέον χρόνος θα απαιτηθεί για να γίνει η επιπλέον λειτουργία. Μάλιστα ο χρόνος αυτός μπορεί να είναι σημαντικός αν για παράδειγμα υποθέσουμε ότι ο αλγόριθμος ταξινομεί εγγραφές στην δευτερεύουσα μνήμη του υπολογιστή (σκληρό δίσκο) δηλαδή μία λειτουργία κατά τεκμήριο αργή.

Υπάρχουν εντούτοις ισχυρά επιχειρήματα υπέρ της επιλογής των συγκρίσεων σαν μέτρου πολυπλοκότητας. Κατά πρώτον οι συγκρίσεις αποτελούν άνω φράγμα του αριθμού των αντιμεταθέσεων. Πράγματι, οι αντιμεταθέσεις μπορεί να είναι το πολύ όσες και οι συγκρίσεις δηλ. το πολύ $n(n - 1)/2$. (Παρεμπιπτόντως, αυτό μπορεί να συμβεί και μάλιστα στην περίπτωση που το διάνυσμα που δίδεται για ταξινόμηση έχει την αντίστροφη διάταξη, δηλαδή είναι ταξινομημένο σε φθίνουσα σειρά.) Συνεπώς ο αριθμός των συγκρίσεων χαρακτηρίζει την πολυπλοκότητα χειρότερης περίπτωσης (worst case complexity) του αλγορίθμου της Φυσαλίδας. Ακόμη όμως και στο τυχαίο τρέξιμο του αλγορίθμου (όπου γενικά περιμένουμε οι συγκρίσεις να είναι λιγότερες από τις αντιμεταθέσεις), ο αριθμός των συγκρίσεων αποτελεί καλή ένδειξη της συνολικής δουλειάς που κάνει ο αλγόριθμος μας.

Ο τρόπος αυτός εκτίμησης της πολυπλοκότητας ενός αλγορίθμου είναι γενικότερος. Τις περισσότερες φορές η ακριβής εκτίμηση του αριθμού των βημάτων μπορεί να είναι πολύ δύσκολη ή/και να εξαρτάται από την συγκεκριμένη είσοδο του αλγορίθμου (δηλαδή, από το συγκεκριμένο στιγμιότυπο του προβλήματος). Επίσης μπορεί και πάλι να μην αντανακλά απολύτως στο χρόνο που θα κάνει ένας υπολογιστής για να τρέξει, μια και οι πράξεις σε μια πραγματική μηχανή δεν διαρκούν όλες τον ίδιο χρόνο. Έτσι, στον αλγόριθμο της

Φυσαλίδας μια σχολαστική απάντηση θα έπρεπε να είναι της μορφής $f(n)$ συγκρίσεις και $g(n)$ αντιμεταθέσεις, μιας και είναι λογικό να υποθέσει κανείς ότι οι συγκρίσεις και οι αντιμεταθέσεις δεν διαρκούν τον ίδιο χρόνο στον υπολογιστή. Και πάλι όμως δεν θα είμασταν εντελώς ακριβείς: Θα έπρεπε να ληφθούν υπόψη τα βήματα που απαιτούνται για την εκτέλεση των δύο βρόχων ανακύκλωσης, κλπ. Είναι όμως προφανές ότι με τον τρόπο αυτό η απάντηση μας για το ποιά είναι η πολυπλοκότητα του αλγορίθμου μας αρχίζει να εκφεύγει από τον αλγόριθμο μας αυτό καθαυτό και μπαίνει σε λεπτομέρειες είτε προγραμματιστικές είτε κατασκευαστικές του υπολογιστή μας. Για όλους αυτούς τους λόγους συνήθως επιλέγουμε κάποιο βήμα του αλγορίθμου (που πολλές φορές μπορεί να είναι και συνθετότερη λειτουργία) το οποίο κυριαρχεί στο σύνολο των βημάτων και συνεπώς αποτελεί καλή ένδειξη της συνολικής δουλειάς του αλγορίθμου. Τα υπόλοιπα βήματα που ενδεχομένως να κάνει ο αλγόριθμος μας συνδέονται συνήθως γραμμικά (είναι δηλαδή ανάλογα) με αυτό το κυρίαρχο βήμα.

Η εξάρτηση του χρόνου που κάνει ένας αλγόριθμος για να απαντήσει από την συγκεκριμένη είσοδο, όπως επισημάνθηκε παραπάνω, μπορεί να είναι ιδιαίτερα έντονη σε σημείο που να εγείρει κριτική για την επιλογή μας να αναφέρουμε σαν πολυπλοκότητα ενός αλγορίθμου την πολυπλοκότητα χειρότερης περίπτωσης. Πράγματι θα ήταν δυνατό η κακή συμπεριφορά ενός αλγορίθμου να παρουσιάζεται μόνο σε ένα μικρό ποσοστό των στιγμιοτύπων του προβλήματος, έτσι ώστε ο αλγόριθμος να συμπεριφέρεται ικανοποιητικά στην μέση ή αναμενόμενη περίπτωση. Για ένα τέτοιο πρόβλημα και αλγόριθμο η αναμενόμενη πολυπλοκότητα του θα ήταν ιδιαίτερα χρήσιμη. Παρόλο που τέτοιοι συνδυασμοί προβλημάτων, αλγορίθμων και πεδίων εφαρμογών πραγματικά υπάρχουν, η μαθηματική μελέτη της αναμενόμενης πολυπλοκότητας απαιτεί την γνώση της κατανομής πιθανότητας στις εισόδους (στα στιγμιότυπα), μια πληροφορία που σπανίως είναι διαθέσιμη. Εξάλλου η πολυπλοκότητα χειρότερης περίπτωσης έχει μέσα της και ένα στοιχείο ασφάλειας: το χειρότερο που μπορεί να συμβεί είναι αυτό. Αντίθετα η αναμενόμενη πολυπλοκότητα είναι χρήσιμη (όταν μπορεί να υπολογιστεί) όταν ο αλγόριθμος χρησιμοποιείται επανειλημμένα στο ίδιο σύνολο στιγμιοτύπων. Στην πλειοψηφία λοιπόν των περιπτώσεων θα ασχοληθούμε με πολυπλοκότητα χειρότερης περίπτωσης.

Η συνάρτηση $n(n-1)/2$ είναι η πολυπλοκότητα χρόνου (time complexity) του αλγορίθμου της Φυσαλίδας. Όπως παρατηρεί ο αναγνώστης, πρόκειται για μια συνάρτηση μιας ακεραίας παραμέτρου (στο παράδειγμα μας του n δηλαδή της διάστασης του διανύσματος που θέλουμε να ταξινομήσουμε). Η παράμετρος αυτή ονομάζεται μέγεθος του προβλήματος και αποτελεί μέτρο του μεγέθους των δεδομένων εισόδου. Πραγματικά, ανεξάρτητα από την φύση του προβλήματος που εξετάζουμε και του αλγορίθμου που επιλέξαμε για την επίλυση του, περιμένουμε περισσότερος χρόνος να απαιτηθεί για με-

γάλα στιγμιότυπα από ότι για μικρότερα και συνεπώς η πολυπλοκότητα χρόνου πρέπει κατ' ανάγκη να είναι μια συνάρτηση του μεγέθους του προβλήματος. Αντίστοιχες συναρτήσεις ζητούνται για οποιοδήποτε μέτρο πολυπλοκότητας μας ενδιαφέρει. Η παράμετρος που προσδιορίζει το μέγεθος του προβλήματος εξαρτάται από την φύση του προβλήματος που εξετάζουμε. Στο πρόβλημα της ταξινόμησης επιλέξαμε το πλήθος των αριθμών σαν μέγεθος του προβλήματος. Η επιλογή αυτή είναι μάλλον δικαιολογημένη αν σκεφτεί κανείς ότι κάθε «λογική» αλγοριθμική διαδικασία ταξινόμησης (και για «λογικού» μεγέθους αριθμούς), διαχειρίζεται κάθε αριθμό σαν μία μονάδα. Τα περισσότερα προβλήματα χαρακτηρίζονται από κάποια φυσική παράμετρο που μπορεί να αποτελέσει μέγεθος του προβλήματος και συνεπώς τις περισσότερες φορές δεν θα υπάρχει δυσκολία στην επιλογή μας. Αξιοσημείωτες εξαιρέσεις στον κανόνα αυτό αποτελούν μερικά αριθμητικά προβλήματα στα οποία πρέπει να είμαστε περισσότερο προσεκτικοί. Σε κάθε περίπτωση όμως θα κάνουμε σαφές τι επιλέγουμε για μέγεθος του προβλήματος που εξετάζουμε. Περισσότερα για το θέμα αυτό αναφέρονται στο Κεφάλαιο 2.

Ας επανέλθουμε τώρα στην συνάρτηση $n(n-1)/2$ που όπως είπαμε είναι η συνάρτηση πολυπλοκότητας του αλγορίθμου της Φυσαλίδας και ας υπολογίσουμε τον αριθμό των συγκρίσεων που θα κάνει ο αλγόριθμος μας για τρία διαφορετικού μεγέθους προβλήματα όπως φαίνεται στον Πίνακα 1.1:

n	$n(n-1)/2$
10	45
100	4950
10000	49995000

Πίνακας 1.1: Πλήθος συγκρίσεων του αλγορίθμου της Φυσαλίδας για τρία διαφορετικού μεγέθους προβλήματα Ταξινόμησης

Μια παρατήρηση του πίνακα αυτού φανερώνει ότι ο αριθμός των συγκρίσεων που υπολογίζουμε όσο το n αυξάνει, εξαρτάται περισσότερο από τον τετραγωνικό προσθετέο $n^2/2$ παρά από τον γραμμικό $-n/2$. Πραγματικά αν είχαμε σαν συνάρτηση πολυπλοκότητας την $n^2/2$, η διαφορά από την πραγματική συνάρτηση γίνεται αμελητέα για μεγάλα n . Στη Θεωρία Αλγορίθμων (και ακόμη περισσότερο στη Θεωρία Πολυπλοκότητας) συνήθως αμελούμε τους μικρότερους προσθετέους και περιοριζόμαστε σε αυτόν που κυριαρχεί για μεγάλα στιγμιότυπα του προβλήματος, αυτόν που δείχνει τον ρυθμό αύξησης της συνάρτησης όταν αυξάνει το n . Στο ίδιο πλαίσιο μπορούμε να προχωρήσουμε αυτές τις σχέσεις και να παρατηρήσουμε ότι και ο συντελεστής $1/2$ του n^2 μπορεί να παραληφθεί σε μια πρώτη εκτίμηση της πολυπλοκότητας ενός αλγο-

ρίθμου (που όμως είναι και πολύ συνηθισμένη στην θεωρητική μελέτη των αλγορίθμων). Αυτό που περισσότερο ενδιαφέρει από την συνάρτηση πολυπλοκότητας ενός αλγορίθμου είναι ο ρυθμός αύξησης του μέτρου πολυπλοκότητας που μας ενδιαφέρει, παρά η απόλυτη τιμή του. Η γνώση του ρυθμού αύξησης μας αποκαλύπτει πλήρως το πως θα συμπεριφερθεί ο αλγόριθμος μας για μεγάλα στιγμιότυπα, ενώ η ακριβής τιμή δεν είναι πάντα αξιοποιήσιμη. Για παράδειγμα, αν ο χρόνος είναι το μέτρο πολυπλοκότητας, η απόλυτη τιμή (σε δευτερόλεπτα) εξαρτάται και από την ταχύτητα της μηχανής μας κάτι που ως γνωστό η τεχνολογική πρόοδος αλλάζει ραγδαία. Για τον λόγο αυτό συνηθίζεται να αναφέρουμε σαν συνάρτηση πολυπλοκότητας μόνο τον κυρίαρχο όρο. Με άλλα λόγια μας ενδιαφέρει η ασυμπτωτική συμπεριφορά του αλγορίθμου για μεγάλες τιμές του n . Έτσι λέμε ότι ο αλγόριθμος της Φυσαλίδας είναι τετραγωνικός. Πάντως πρέπει να αναφερθεί ότι ενώ η πρακτική αυτή είναι ο κανόνας, δεν λείπουν και οι εξαιρέσεις κυρίως σε πρακτικά προβλήματα και αλγορίθμους με «μικρές» συναρτήσεις πολυπλοκότητας. Η ταξινόμηση είναι ένα τέτοιο πρόβλημα και ο αναγνώστης μπορεί να βρεί ιδιαίτερα προσεκτικές αναλύσεις πολλών διαφορετικών αλγορίθμων για το πρόβλημα της ταξινόμησης στο κλασικό βιβλίο του Donald Knuth [15]. Στο επόμενο κεφάλαιο, όπου θα δωθεί και το αυστηρό μαθηματικό μοντέλο του υπολογιστή μας, η μηχανή Turing, θα δωθεί και θεωρητική εξήγηση για αυτή την επιλογή. Επί του παρόντος δίδουμε μερικούς αυστηρούς μαθηματικούς συμβολισμούς με την βοήθεια των οποίων θα περιγράψουμε τις επιδόσεις των αλγορίθμων μας, αλλά και την πολυπλοκότητα των προβλημάτων μας.

Ορισμός 1.1 Έστω συνάρτηση g από το σύνολο των θετικών ακεραίων στο σύνολο των θετικών ακεραίων.

1. Ορίζουμε ως $O(g(n))$ το σύνολο των συναρτήσεων

$$O(g(n)) = \{f(n) : \text{υπάρχουν θετικές σταθερές } c \text{ και } n_0 \text{ έτσι ώστε} \\ 0 \leq f(n) \leq cg(n) \text{ για κάθε } n \geq n_0\}.$$

2. Ορίζουμε ως $\Theta(g(n))$ το σύνολο των συναρτήσεων

$$\Theta(g(n)) = \{f(n) : \text{υπάρχουν θετικές σταθερές } c_1, c_2 \text{ και } n_0 \text{ έτσι ώστε} \\ 0 \leq c_1g(n) \leq f(n) \leq c_2g(n) \text{ για κάθε } n \geq n_0\}.$$

3. Ορίζουμε ως $\Omega(g(n))$ το σύνολο των συναρτήσεων

$$\Omega(g(n)) = \{f(n) : \text{υπάρχουν θετικές σταθερές } c \text{ και } n_0 \text{ έτσι ώστε} \\ 0 \leq cg(n) \leq f(n) \text{ για κάθε } n \geq n_0\}.$$

4. Ορίζουμε ως $o(g(n))$ το σύνολο των συναρτήσεων

$$o(g(n)) = \{f(n) : f(n) \leq g(n) \text{ για πεπερασμένο αριθμό } n \text{ και} \\ \lim_{n \rightarrow \infty} \frac{f(n)}{g(n)} = 0\}$$

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 $c_1g(n)$ και $c_2g(n)$ 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* για την συνάρτηση 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 η 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 αυστηρά μικρότερο από το ρυθμό αύξησης της $g(n)$.

Χρησιμοποιούμε τους παραπάνω συμβολισμούς για να περιγράψουμε την επίδοση των αλγορίθμων μας αλλά και την πολυπλοκότητα των προβλημάτων μας. Για παράδειγμα, η συνάρτηση χρόνου του αλγορίθμου της Φυσαλίδας είναι τάξης $O(n^2)$.

1.2 Πολυπλοκότητα Προβλημάτων

Ο αλγόριθμος της Φυσαλίδας που περιγράψαμε για το πρόβλημα της ταξινόμησης δεν είναι ο μοναδικός αλγόριθμος για το πρόβλημα αυτό. Υπάρχουν αρκετοί αλγόριθμοι [15] και πολλοί από αυτούς, όπως για παράδειγμα ο *αλγόριθμος του Σωρού* (heapsort), συμπεριφέρονται καλύτερα από τον της Φυσαλίδας. Ο αλγόριθμος του Σωρού για παράδειγμα έχει πολυπλοκότητα χρόνου $O(n \log n)$ δηλαδή σαφώς καλύτερη από το $O(n^2)$ του αλγορίθμου της Φυσαλίδας. Το εύλογο ερώτημα που προκύπτει είναι αν ο χρόνος $O(n \log n)$ μπορεί να βελτιωθεί ακόμη περισσότερο: αν δηλαδή υπάρχει αλγόριθμος που μπορεί να ταξινομήσει n αριθμούς σε χρόνο μικρότερο από $O(n \log n)$. Πρέπει να τονιστεί ότι το ερώτημα αυτό είναι διαφορετικής φύσης από το πρόβλημα μελέτης της πολυπλοκότητας κάποιου συγκεκριμένου αλγορίθμου (όπως κάναμε στην

περίπτωση του αλγορίθμου της Φυσαλίδας): τώρα ζητάμε να μάθουμε την πολυπλοκότητα του καλύτερου δυνατού αλγόριθμου για την ταξινόμηση μεταξύ όλων των αλγορίθμων (γνωστών και «μη γνωστών») που επιλύουν το συγκεκριμένο πρόβλημα. Η πολυπλοκότητα αυτή είναι και η πολυπλοκότητα χρόνου του προβλήματος της ταξινόμησης. Αντίστοιχα ορίζεται και η πολυπλοκότητα προβλημάτων για άλλα μέτρα που μπορεί να ενδιαφέρουν. Για το πρόβλημα της ταξινόμησης ισχύει το παρακάτω Θεώρημα, η απόδειξη του οποίου μπορεί να βρεθεί για παράδειγμα στο [1]:

Θεώρημα 1.1 *Κάθε αλγόριθμος που ταξινομεί μέσω συγκρίσεων χρειάζεται $\Omega(n \log n)$ συγκρίσεις για να ταξινομήσει n στοιχεία.*

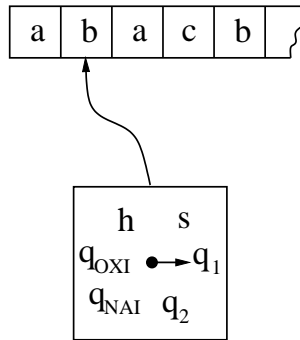
Αυτό που πραγματικά λέει το Θεώρημα 1.1 είναι ένα κάτω φράγμα στην δυνατότητα μας να ταξινομήσουμε n αριθμούς. Σε συνδυασμό όμως με ότι αναφέρθηκε για την ύπαρξη αλγορίθμου που ασυμπτωτικά καταφέρνει να ταξινομήσει στον ίδιο χρόνο $O(n \log n)$, καταλήγουμε ότι η πολυπλοκότητα του προβλήματος της ταξινόμησης είναι $\Theta(n \log n)$. Είναι όμως αναγκαίο να τονιστούν ορισμένα πράγματα ως προς την σημασία του παραπάνω θεωρήματος και τα όρια του: το Θεώρημα 1.1 δείχνει ότι σε κάποιο συγκεκριμένο πλαίσιο (κάποιες υποθέσεις) δεν είναι δυνατό να ταξινομηθούν n αριθμοί σε χρόνο καλύτερο από $O(n \log n)$. Το πλαίσιο αυτό καθορίζει ότι εργαζόμαστε σε ένα βασικό σύνολο στο οποίο υπάρχει μία μερική διάταξη και στο οποίο η μοναδική δυνατότητα που έχουμε είναι να αποφανθούμε μεταξύ δύο στοιχείων ποιό προηγείται. Στο πλαίσιο αυτό, το οποίο είναι αρκετά ρεαλιστικό, πράγματι ισχύει το παραπάνω αποτέλεσμα. Υπάρχουν όμως περιπτώσεις όπου υπάρχουν περισσότερες δυνατότητες για το αλγόριθμο μας. Για παράδειγμα αν υποθέσουμε ότι οι αριθμοί είναι «μικροί» (μέσα σε κάποιο διάστημα, έστω 1 έως m) τότε μπορούμε να τα καταφέρουμε ακόμη καλύτερα και να ταξινομήσουμε n αριθμούς σε γραμμικό χρόνο $O(n + m)$. Ο αλγόριθμος αυτός λέγεται αλγόριθμος του Κάδου (bucketsort).

Το Θεώρημα 1.1 περιγράφει την πολυπλοκότητα κάποιου προβλήματος. Πρόκειται για ένα ιδιαίτερα ακριβές αποτέλεσμα, το οποίο εξήχθη με μεθόδους απόλυτα προσαρμοσμένες στο συγκεκριμένο πρόβλημα. Δυστυχώς τέτοιου είδους ακριβή αποτελέσματα είναι πολύ δύσκολο να εξαχθούν και κατά συνέπεια είναι πολύ λίγα στην Θεωρία Πολυπλοκότητας. Αναγκαζόμαστε λοιπόν να χρησιμοποιήσουμε άλλες μεθόδους, περισσότερο γενικές, που δίνουν κατά βάση ένα χαρακτηρισμό του προβλήματος παρά ένα ακριβές κάτω φράγμα. Αυτές οι μέθοδοι είναι το αντικείμενο των επομένων κεφαλαίων.

Κεφάλαιο 2

Η Μηχανή Turing

Στο προηγούμενο κεφάλαιο περιγράψαμε με την βοήθεια μιας ψευδογλώσσας έναν αλγόριθμο για το πρόβλημα της ταξινόμησης. Το «τρέξιμο» αυτού του αλγορίθμου, αφέθηκε να εννοηθεί ότι θα γίνει σε ένα υπολογιστή ο οποίος είναι σε θέση να εκτελέσει προγράμματα μιας γλώσσας που ομοιάζει με την Pascal. Είναι γνωστό ότι όλοι οι υπολογιστές μπορούν να εκτελέσουν παρόμοιες εντολές. Ιδού όμως μια ενδιαφέρουσα προοπτική. Θα ήταν άραγε δυνατό να υπάρχει κάποια υπολογιστική μηχανή με εξαιρετικές ιδιότητες για την οποία θα μπορούσε να γραφεί ένας αλγόριθμος με πολύ καλύτερες επιδόσεις από τους γνωστούς; Με άλλα λόγια, ποιά είναι τα όρια που μπορεί να φτάσει ο (μηχανιστικός) υπολογισμός; Πέρα όμως από αυτό, είναι ανάγκη εφόσον στόχος μας είναι η αυστηρά μαθηματική μελέτη των υπολογιστικών προβλημάτων και των αλγορίθμων να καταλήξουμε σε κάποιο αυστηρό μοντέλο υπολογιστικής μηχανής στο οποίο θα γίνονται οι μετρήσεις των αναγκαίων υπολογιστικών πόρων με εννιαίο τρόπο. Το μοντέλο αυτό θα πρέπει να συνδυάζει τα παρακάτω χαρακτηριστικά: Πρώτον, θα πρέπει να είναι αρκετά κοντά στους πραγματικούς (ρεαλιστικούς) υπολογιστές ώστε τα αποτελέσματα που θα παίρνουμε από την μαθηματική ανάλυση στο μοντέλο μας να είναι μεταφέριμα στις πραγματικές μηχανές. Και δεύτερον, θα πρέπει να είναι ιδιαίτερα απλό ώστε η ανάπτυξη της θεωρίας μας να είναι απλή. Κατά παράδοξο τρόπο η απάντηση και στα δύο ερωτήματα που τέθηκαν, είναι κοινή. Το μαθηματικό μοντέλο το οποίο έχει υιοθετηθεί στη Θεωρία Πολυπλοκότητας είναι η *μηχανή Turing* η οποία 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 και μάλιστα με επιδόσεις αρκετά κοντά σε αυτές των ρεαλιστικών υπολογιστών.



Σχήμα 2.1: Σχηματική αναπαράσταση της μηχανής Turing.

2.1 Περιγραφή – Ορισμός

Η μηχανή Turing προτάθηκε από τον Alan Turing [30] σαν ένα μοντέλο υπολογισμού και είναι μια μηχανή που επιδρά σε *σύμβολα*. Τα σύμβολα που μπορεί να διαχειριστεί μία μηχανή Turing είναι στοιχεία ενός πεπερασμένου συνόλου συμβόλων που ονομάζεται *αλφάβητο*. Μία ακολουθία από σύμβολα του αλφαβήτου μας αποτελεί μια *λέξη* ή *συμβολοσειρά*. Τέλος, ένα σύνολο συμβολοσειρών αποτελεί μια *γλώσσα*. Τα σύμβολα της μηχανής μπορούμε να θεωρήσουμε ότι είναι γραμμένα σε μια «ταινία» που έχει άπειρο μήκος και προς τις δύο κατευθύνσεις (μοιάζει δηλαδή με την γεωμετρική ευθεία). Η ταινία χωρίζεται σε *κύτταρα*, σε κάθε ένα από τα οποία μπορεί να εγγραφεί ένα σύμβολο. Η μηχανή μπορεί να διαβάζει ή να γράφει ένα σύμβολο σε κάθε κίνηση της μέσω μιας *κεφαλής* ανάγνωσης/εγγραφής. Μια κίνηση της μηχανής Turing είναι συνδυασμός δύο πραγμάτων: του συμβόλου που βρίσκεται την χρονική στιγμή πριν από την κίνηση στο κύτταρο κάτω από την κεφαλή και της τρέχουσας *κατάστασης* στην οποία βρίσκεται η μηχανή μας. Μια κατάσταση μπορεί να ειπωθεί σαν ένα *στοιχείο μνήμης* και κάθε μηχανή Turing έχει ένα πεπερασμένο σύνολο από καταστάσεις. Μια σχηματική παράσταση της μηχανής Turing φαίνεται στο Σχήμα 2.1. Παρατηρούμε την ταινία της μηχανής καθώς και την κεφαλή εγγραφής/ανάγνωσης. Στο τετράγωνο πλαίσιο θεωρούμε ότι βρίσκονται οι καταστάσεις καθώς και ο μηχανισμός που αποφασίζει τις κινήσεις της μηχανής μας, όπως λέχθηκε σαν συνδυασμός της τρέχουσας κατάστασης και του συμβόλου που βρίσκεται κάτω από την κεφαλή. Αυτός όλος ο μηχανισμός συνιστά τον *πεπερασμένο έλεγχο* της μηχανής. Τι είναι όμως μια κίνηση της μηχανής Turing; Πολύ απλά μια κίνηση της μηχανής Turing συνίσταται στην αλλαγή του συμβόλου που βρίσκεται κάτω από την κεφαλή με κάποιο άλλο, στην εν συνεχεία κίνηση της μία θέση αριστερά ή δεξιά και στην αλλαγή της κατάστασης της.

Τα παραπάνω πρέπει φυσικά να ειπωθούν σαν μια «μηχανοποιημένη» περι-

γραφή της μηχανής Turing η οποία είναι ένα μαθηματικό αντικείμενο, χωρίς ταινίες, κεφαλές κλπ. Βοηθούν ωστόσο στην κατανόηση του μοντέλου μας. Προτού δώσουμε τον μαθηματικό ορισμό της μηχανής Turing, είναι χρήσιμο να συμπληρώσουμε την περιγραφή με ορισμένες ακόμη λεπτομέρειες. Πρέπει να λεχθεί ότι σε αυτές τις λεπτομέρειες υπάρχουν διαφορές στην βιβλιογραφία στον τρόπο που ορίζονται. Οι διαφορές αυτές είναι επουσιώδεις και με κανένα τρόπο δεν επηρεάζουν την συμπεριφορά του μοντέλου μας για τον σκοπό που το ορίσαμε. Έτσι πρέπει να κάνουμε μια σύμβαση για το πως δίνουμε την είσοδο στην μηχανή και πως αρχίζει ο υπολογισμός. Θεωρούμε λοιπόν ότι υπάρχει ένα ειδικό σύμβολο το \triangleright το οποίο υπάρχει σε κάποιο κύτταρο και αμέσως δεξιά από αυτό τοποθετείται η συμβολοσειρά εισόδου (η είσοδος στην μηχανή μας). Όλα τα κύτταρα που δεν περιέχουν κάποιο άλλο σύμβολο, θεωρείται ότι περιέχουν το κενό, \flat . Η κεφαλή της μηχανής πριν την αρχή του υπολογισμού θεωρείται ότι βρίσκεται πάνω από το \triangleright . Η μηχανή πριν από την αρχή του υπολογισμού βρίσκεται σε μια ειδική κατάσταση που την συμβολίζουμε με το s και ονομάζεται *αρχική κατάσταση*. Η μηχανή σταματά όταν βρεθεί σε μια από τις καταστάσεις $h, q_{\text{NAI}}, q_{\text{OXI}}$. Η πρώτη ονομάζεται κατάσταση τερματισμού, ενώ οι άλλες δύο, πέρα από το τέρμα του υπολογισμού, σηματοδοτούν η μεν πρώτη (κατάσταση q_{NAI}) την αποδοχή της εισόδου, η δε δεύτερη (κατάσταση q_{OXI}) την απόρριψη της εισόδου.

Ορισμός 2.1 Μια μηχανή Turing M είναι μία διατεταγμένη τετράδα $M = (K, \Sigma, \delta, s)$. Το K είναι το πεπερασμένο σύνολο καταστάσεων. Το Σ είναι το πεπερασμένο αλφάβητο της μηχανής, δηλαδή το σύνολο των συμβόλων που μπορούν να εμφανιστούν στην ταινία της. Το Σ πάντα περιέχει τα δύο σύμβολα \triangleright και \flat . Το $s \in K$ είναι η αρχική κατάσταση. Τέλος το $\delta : K \times \Sigma \rightarrow (K \cup \{h, q_{\text{NAI}}, q_{\text{OXI}}\}) \times \Sigma \times \{\leftarrow, \rightarrow, -\}$ είναι η συνάρτηση μετάβασης της μηχανής.

Στον ορισμό μας υποθέτουμε ότι οι τρεις καταστάσεις τερματισμού δεν είναι στοιχεία του K . Παρόμοια, τα σύμβολα του συνόλου $\{\leftarrow, \rightarrow, -\}$ δεν είναι στοιχεία του Σ . Η συνάρτηση δ προσδιορίζει τις κινήσεις που μπορεί να κάνει η μηχανή. Όπως ήδη λέχθηκε, μια κίνηση εξαρτάται από δύο πράγματα: την τρέχουσα κατάσταση και το σύμβολο κάτω από την κεφαλή. Ο συνδυασμός αυτών των δύο παραμέτρων μονοσήμαντα προσδιορίζει την κίνηση της μηχανής, δηλαδή την επόμενη κατάσταση, το σύμβολο που θα αντικαταστήσει αυτό που είναι κάτω από την κεφαλή και την κίνηση της κεφαλής (αριστερά, δεξιά ή στάσιμη). Αυτό ακριβώς προσδιορίζει η συνάρτηση δ . Για κάθε $q \in K$ και $\sigma \in \Sigma$ καθορίζει την τριάδα $\delta(q, \sigma) = (q', \sigma', D)$, δηλαδή την επόμενη κατάσταση, το επόμενο σύμβολο και την κίνηση D της κεφαλής αντίστοιχα. Το D είναι στοιχείο του συνόλου $\{\leftarrow, \rightarrow, -\}$. Μία επιπλέον σύμβαση είναι ότι η

συνάρτηση δ είναι τέτοια που ποτέ το σύμβολο \triangleright δεν σβήνεται και επιπλέον δεν επιτρέπει στην κεφαλή να κινηθεί αριστερά του. Η δ μπορεί να ειπωθεί σαν το «πρόγραμμα» της μηχανής. Η μόνη διαφορά από τα γνωστά προγράμματα (πέρα ίσως από το περιορισμένο ρεπερτόριο εντολών) είναι ότι είναι αμετάβλητο για μια δεδομένη μηχανή Turing, σε αντίθεση με τους υπολογιστές που παρέχουν την ευχέρεια να αλλάζουμε την συμπεριφορά τους αλλάζοντας το πρόγραμμα που τρέχουν. Σύντομα όμως θα διαπιστώσουμε ότι και αυτό το «μειονέκτημα» των μηχανών Turing μπορεί να διορθωθεί.

Η μηχανή αρχίζει τον υπολογισμό της έχοντας τοποθετημένη την κεφαλή της στο σύμβολο \triangleright , ενώ δεξιά της βρίσκεται η συμβολοσειρά εισόδου $x \in (\Sigma \setminus \{b\})^*$. (Εδώ παρατηρούμε ότι το κενό σύμβολο δεν μπορεί να είναι στοιχείο της εισόδου. Με αυτόν τον τρόπο το κενό σύμβολο σηματοδοτεί το τέλος της συμβολοσειράς εισόδου.) Στη συνέχεια η μηχανή ακολουθώντας τις μεταβάσεις που προβλέπει η συνάρτηση δ κάνει διαδοχικές κινήσεις αλλάζοντας καταστάσεις, αλλάζοντας σύμβολα και κινώντας την κεφαλή μέχρι (πιθανώς) να συναντήσει μία από τις τρεις καταστάσεις h, q_{NAI} ή q_{OXI} οπότε και σταματά. Σε αυτές τις περιπτώσεις η έξοδος (απάντηση) της μηχανής με είσοδο x συμβολίζεται με $M(x)$ και είναι $M(x) = \text{«NAI»}$ αν η τελευταία κατάσταση είναι η q_{NAI} , $M(x) = \text{«OXI»}$ αν η τελευταία κατάσταση είναι η q_{OXI} και τέλος αν η τελευταία κατάσταση είναι η h , η έξοδος είναι ότι περιέχεται στην ταινία της M , έστω η συμβολοσειρά y , από το σύμβολο \triangleright μέχρι το σύμβολο μετά το οποίο αρχίζει η άπειρη σειρά των κενών. Εφόσον η μηχανή έχει κάνει πεπερασμένο αριθμό κινήσεων, αναγκαστικά πρέπει να υπάρχει κάποιο τέτοιο σύμβολο. Σε τέτοια περίπτωση γράφουμε $M(x) = y$. Είναι πιθανό όμως η μηχανή μας να μην φθάσει ποτέ σε μία από τις τρεις καταστάσεις τερματισμού οπότε λέμε ότι *αποκλίνει* στο x . Σε τέτοια περίπτωση γράφουμε $M(x) = \uparrow$.

Μία πολύ χρήσιμη έννοια στην παρακολούθηση της λειτουργίας μιας μηχανής Turing είναι η έννοια της *διαμόρφωσης* (configuration) ή *στιγμιαίας περιγραφής* (instantaneous description). Διαισθητικά η διαμόρφωση είναι το σύνολο των πληροφοριών που μονοσήμαντα προσδιορίζουν την εξέλιξη του υπολογισμού μετά από κάποιο βήμα. Ισοδύναμα μπορούμε να δούμε την διαμόρφωση σαν μία «φωτογραφία» της μηχανής μας μία χρονική στιγμή. Δεν είναι δύσκολο να αντιληφθούμε ότι οι παράμετροι που επηρεάζουν τον υπολογισμό είναι η κατάσταση, το περιεχόμενο της ταινίας και η θέση της κεφαλής. Για ευκολία τις δύο τελευταίες πληροφορίες τις περιγράφουμε με δύο συμβολοσειρές u και v . Η u είναι η συμβολοσειρά από το σύμβολο \triangleright μέχρι και το σύμβολο κάτω από την κεφαλή. Η v αρχίζει από το σύμβολο δεξιά της κεφαλής μέχρις εκεί που αρχίζει η άπειρη σειρά των κενών. Επομένως μια διαμόρφωση C είναι ένα στοιχείο του Καρτεσιανού γινομένου $K \times (\Sigma \setminus \{b\})^* \times (\Sigma \setminus \{b\})^*$. Στα παραδείγματα που ακολουθούν περιγράφεται η εξέλιξη του υπολογισμού μιας μηχανής με παράθεση των διαδοχικών διαμορφώσεων.

Ξεκινάμε με ένα παράδειγμα μιας μηχανής Turing που μετά από πεπερασμένο αριθμό βημάτων τερματίζει στην κατάσταση q_{NAI} ή q_{OXI} .

Παράδειγμα 2.1 Έστω η μηχανή Turing $M = (K, \Sigma, \delta, s)$ όπου $\Sigma = \{\triangleright, b, a, \flat\}$, $K = \{s\}$, και δ η συνάρτηση μετάβασης που ορίζεται σύμφωνα με τον παρακάτω πίνακα:

q	σ	$\delta(q, \sigma)$
s	\triangleright	$(s, \triangleright, \rightarrow)$
s	a	$(q_{\text{NAI}}, a, -)$
s	b	(s, b, \rightarrow)
s	\flat	$(q_{\text{OXI}}, \flat, -)$

Ας δούμε τις μεταβάσεις της μηχανής M με είσοδο, για παράδειγμα την συμβολοσειρά $bbaab$, καταγράφοντας τις διαδοχικές διαμορφώσεις της:

$$\begin{aligned}
 (s, \triangleright, bbaab) &\mapsto (s, \triangleright b, baab) \\
 &\mapsto (s, \triangleright bb, aab) \\
 &\mapsto (s, \triangleright bba, ab) \\
 &\mapsto (q_{\text{NAI}}, \triangleright bba, ab)
 \end{aligned}$$

Η μηχανή M λοιπόν, μετά από ένα αριθμό βημάτων, τερματίζει στην κατάσταση q_{NAI} . Ας δούμε την λειτουργία της μηχανής για μια ακόμα είσοδο, π.χ. για την συμβολοσειρά bbb :

$$\begin{aligned}
 (s, \triangleright, bbb) &\mapsto (s, \triangleright b, bb) \\
 &\mapsto (s, \triangleright bb, b) \\
 &\mapsto (s, \triangleright bbb, \flat) \\
 &\mapsto (s, \triangleright bbbb, \flat) \\
 &\mapsto (q_{\text{OXI}}, \triangleright bbbb, \flat)
 \end{aligned}$$

Με είσοδο την συμβολοσειρά bbb η μηχανή M τερματίζει στην κατάσταση q_{OXI} . Είναι εύκολο να δει κανείς ότι η μηχανή M αποδέχεται όλες τις συμβολοσειρές που περιέχουν τον χαρακτήρα a και απορρίπτει αυτές που περιέχουν μόνο b .

Στη συνέχεια δίνουμε ένα παράδειγμα μιας μηχανής Turing η οποία με είσοδο μια συμβολοσειρά x δεν τερματίζει στην κατάσταση q_{NAI} ή q_{OXI} αλλά δίνει σαν έξοδο μια συμβολοσειρά y .

Παράδειγμα 2.2 Έστω η μηχανή Turing $M = (K, \Sigma, \delta, s)$ όπου $\Sigma = \{\triangleright, b, a, \text{b}\}$, $K = \{s, p\}$, και δ η συνάρτηση μετάβασης που ορίζεται σύμφωνα με τον παρακάτω πίνακα:

q	σ	$\delta(q, \sigma)$
s	\triangleright	$(s, \triangleright, \rightarrow)$
s	a	$(p, b, -)$
s	b	$(p, a, -)$
s	b	$(h, \text{b}, -)$
p	a	(s, a, \rightarrow)
p	b	(s, b, \rightarrow)
p	b	$(s, \text{b}, -)$

Ας δούμε τις μεταβάσεις της μηχανής M με είσοδο, για παράδειγμα την συμβολοσειρά $aabab$:

$$\begin{aligned}
 (s, \triangleright, aabab) &\mapsto (s, \triangleright a, abab) \\
 &\mapsto (p, \triangleright b, abab) \\
 &\mapsto (s, \triangleright ba, bab) \\
 &\mapsto (p, \triangleright bb, bab) \\
 &\mapsto (s, \triangleright bbb, ab) \\
 &\mapsto (p, \triangleright bba, ab) \\
 &\mapsto (s, \triangleright bbba, b) \\
 &\mapsto (p, \triangleright bbab, b) \\
 &\mapsto (s, \triangleright bbabb, \text{b}) \\
 &\mapsto (p, \triangleright bbaba, \text{b}) \\
 &\mapsto (s, \triangleright bbabab, \text{b}) \\
 &\mapsto (h, \triangleright bbabab, \text{b})
 \end{aligned}$$

Βλέπουμε λοιπόν ότι με είσοδο την συμβολοσειρά $aabab$, η μηχανή M τερματίζει μετά από πεπερασμένο πλήθος βημάτων και δίνει σαν έξοδο την συμβολοσειρά $bbaba$, δηλαδή την συμβολοσειρά εισόδου αντεστραμένη. Επαναλαμβάνοντας για ένα-δυο ακόμα παραδείγματα, εύκολα φτάνουμε στο συμπέρασμα ότι η λειτουργία της συγκεκριμένης μηχανής είναι να αντιστρέφει την είσοδο της.

Παρατήρηση 2.1 Μια πολύ χρήσιμη τεχνική για τον σχεδιασμό μηχανών Turing, είναι να χρησιμοποιούμε τον πεπερασμένο έλεγχο (τις καταστάσεις)

για την αποθήκευση σταθερού μεγέθους πληροφοριών. Έτσι αν έχουμε σχεδιάσει μερικώς μια μηχανή που εκτελεί κάποια λειτουργία και θέλουμε επιπλέον να απομνημονεύσει πληροφορία ενός δυαδικού ψηφίου (1 bit), μπορούμε να κάνουμε δύο αντίγραφα των καταστάσεων με τις ίδιες μεταβάσεις, στο ένα από τα οποία η τιμή του δυαδικού ψηφίου είναι 1 και στο άλλο είναι 0. Φυσικά αν πρέπει να απομνημονευτεί πληροφορία δύο δυαδικών ψηφίων, 4 αντίγραφα των καταστάσεων χρειάζονται, κλπ. Η μεγάλη αύξηση του αριθμού των καταστάσεων δεν ενοχλεί στο βαθμό που ο αριθμός τους παραμένει ανεξάρτητος της εισόδου.¹ Θα κάνουμε ευρεία χρήση αυτής της τεχνικής στα θεωρήματα που αποδεικνύονται στο επόμενο κεφάλαιο.

Ο αναγνώστης μπορεί να αναρωτηθεί ποιά η ανάγκη να έχουμε δύο τρόπους «φυσιολογικού» τερματισμού ενός υπολογισμού, δηλαδή μέσω αποδοχής (ή απόρριψης) και μέσω υπολογισμού μιας συμβολοσειράς εξόδου. Ο λόγος είναι να έχει το μοντέλο μας την δυνατότητα να χρησιμοποιείται είτε σαν αναγνωριστής γλωσσών, όπως στο Παράδειγμα 2.1, δηλαδή να αποφαινεται αν μια δοθείσα συμβολοσειρά x ανήκει ή όχι σε μια γλώσσα L , είτε σαν υπολογιστής συναρτήσεων, όπως στο Παράδειγμα 2.2. Η πρώτη χρήση του μοντέλου μας είναι ιδιαίτερα κατάλληλη για τα λεγόμενα προβλήματα απόφασης (decision problems) δηλαδή προβλήματα στα οποία η απάντηση είναι «ΝΑΙ» ή «ΟΧΙ». Η δεύτερη χρήση είναι κατάλληλη και για προβλήματα με συνθετότερη απάντηση. Φυσικά θα μπορούσαμε να χρησιμοποιούσαμε μόνο τον δεύτερο τρόπο και να βάζαμε την μηχανή μας να δώσει μια ρητή απάντηση (δηλαδή ένα «ΝΑΙ» ή ένα «ΟΧΙ») στην ταινία της και να αποφεύγαμε την χρήση των καταστάσεων q_{NAI} και q_{OXI} . Επειδή όμως τα προβλήματα απόφασης είναι εξαιρετικά μεγάλης σημασίας για την Θεωρία Πολυπλοκότητας, ο παρών ορισμός της μηχανής Turing αποδεικνύεται ιδιαίτερα βολικός. Πιο κάτω δίδονται ακριβείς ορισμοί αυτών των εννοιών.

Ορισμός 2.2 Έστω $L \subseteq (\Sigma \setminus \{b\})^*$ μία γλώσσα. Αν μία μηχανή Turing M είναι τέτοια ώστε για κάθε συμβολοσειρά $x \in L$, $M(x) = q_{\text{NAI}}$ και για κάθε συμβολοσειρά $x \notin L$, $M(x) = q_{\text{OXI}}$, τότε λέμε ότι η M αποφασίζει (decides) την L . Αν μία μηχανή Turing M είναι τέτοια ώστε για κάθε συμβολοσειρά $x \in L$, $M(x) = q_{\text{NAI}}$ και για κάθε συμβολοσειρά $x \notin L$, $M(x) = \uparrow$, τότε λέμε ότι η M αποδέχεται (accepts) την L .

Ορισμός 2.3 Αν για μία δοθείσα γλώσσα υπάρχει μηχανή Turing που να την αποφασίζει, τότε λέμε ότι η γλώσσα είναι αναδρομική (recursive). Παρόμοια,

¹ Στην πραγματικότητα μόνο πολλαπλασιαστική και όχι εκθετική αύξηση στον αριθμό των καταστάσεων είναι απαραίτητη, όμως η σχεδίαση είναι πολύ δυσκολότερη και η λειτουργία της μηχανής πολύ λιγότερο κατανοητή. Οι τεχνικές που χρησιμοποιούνται αναπτύχθηκαν σε ένα τελείως διαφορετικό χώρο, αυτόν της ψηφιακής σχεδίασης.

αν για μία γλώσσα υπάρχει μηχανή Turing που να την αποδέχεται, τότε λέμε ότι η γλώσσα είναι αναδρομικά αριθμήσιμη (*recursive enumerable*).

Είναι σημαντικό να επισημανθεί η διαφορά στο βαθμό αλγοριθμικής προσέγγισης που υπάρχει μεταξύ μιας αναδρομικής και μιας αναδρομικά αριθμήσιμης γλώσσας. Στην πρώτη περίπτωση έχουμε στην διάθεση μας μια υπολογιστική μηχανή που μπορεί σε πεπερασμένο χρόνο να αποφασίσει αν μια οποιαδήποτε συμβολοσειρά στο αλφάβητο μας ανήκει ή όχι στην γλώσσα μας. Στην δεύτερη περίπτωση υπάρχει μία *ασυμμετρία* μεταξύ των συμβολοσειρών που ανήκουν και αυτών που δεν ανήκουν στην γλώσσα. Σαφή γνώση μπορούμε να έχουμε μόνο για τις συμβολοσειρές που ανήκουν στη γλώσσα διότι στην αντίθετη περίπτωση ποτέ δεν είμαστε σε θέση να αποφανθούμε ότι περιμέναμε αρκετά και άρα να συμπεράνουμε ότι η μηχανή μας δεν πρόκειται να αποδεχθεί την συμβολοσειρά εισόδου. Έτσι *αλγόριθμο* για το πρόβλημα του κατά πόσον μια συμβολοσειρά x ανήκει σε μια γλώσσα L , θεωρούμε ότι έχουμε μόνο στην πρώτη περίπτωση, εκεί δηλαδή όπου και οι δύο πιθανές απαντήσεις («ΝΑΙ» και «ΟΧΙ») μπορούν να ληφθούν σε πεπερασμένο χρόνο. Αντίθετα στην δεύτερη περίπτωση δεν θεωρούμε ότι το πρόβλημα επιλύεται αλγοριθμικά.

Φυσικά αυτή η θεώρηση απαιτεί την ταύτιση της μηχανής Turing με την έννοια του αλγορίθμου κάτι που ίσως σε αυτό το στάδιο να φαίνεται περίεργο. Πραγματικά όμως, όσο πρωτόγονη και αν φαίνεται η μηχανή Turing μπορεί ναδειχθεί μέσω προσομοίωσης ότι υπολογίζει τα ίδια ακριβώς προβλήματα με οποιοδήποτε άλλο μοντέλο έχει προταθεί στην ιστορία των Μαθηματικών και της Πληροφορικής για να συλλάβει την έννοια του μηχανιστικού υπολογισμού [14]. Αυτά τα αποτελέσματα οδήγησαν στην λεγόμενη Θέση του Church σύμφωνα με την οποία αυτά τα ισοδύναμα υπολογιστικά μοντέλα (μεταξύ αυτών όπως ελέγχθη και η μηχανή Turing) ταυτίζονται με την έννοια του αλγορίθμου. Συνεπώς ένα πρόβλημα θα λέμε ότι επιλύεται αλγοριθμικά (έχει αλγόριθμο) αν γι' αυτό υπάρχει μία μηχανή Turing που να το επιλύει. Πρέπει να πούμε ότι επί του παρόντος τα παραπάνω ισχύουν ως προς το ποιά προβλήματα επιλύει η μηχανή Turing και όχι ως προς το πόσο αποδοτικά (γρήγορα ή οτιδήποτε άλλο μπορεί να μας ενδιαφέρει). Όπως θα δούμε όμως, ακόμα και ως προς την ταχύτητα η μηχανή Turing δεν είναι πολύ μακριά (με κάποια θεωρητική έννοια που θα ορίσουμε) ακόμη και από τους σύγχρονους υπολογιστές.

Ας επανέλθουμε τώρα στον άλλο τρόπο τερματισμού μιας μηχανής Turing μέσω της κατάστασης h . Σ' αυτή την περίπτωση η μηχανή μπορεί να χρησιμοποιήσει για τον υπολογισμό συναρτήσεων (από συμβολοσειρές σε συμβολοσειρές).

Ορισμός 2.4 Έστω $f(x) : (\Sigma \setminus \{b\})^* \longrightarrow \Sigma^*$ μία συνάρτηση από συμβολοσειρές (στο αλφάβητο Σ αλλά χωρίς το κενό σύμβολο) σε συμβολοσειρές. Θα λέμε ότι μια μηχανή Turing M υπολογίζει την f αν $M(x) = f(x)$ για

κάθε συμβολοσειρά $x \in (\Sigma - \{b\})^*$. Αν υπάρχει τέτοια μηχανή για την f , τότε θα λέμε ότι η συνάρτηση f είναι αναδρομική (*recursive function*).

Φυσικά επειδή οι αριθμοί αλλά και άλλα μαθηματικά αντικείμενα μπορούν να παρασταθούν με σύμβολα, είναι προφανές ότι μια μηχανή Turing μπορεί να υπολογίζει αριθμητικές συναρτήσεις αλλά και γενικότερα συναρτήσεις από οποιοδήποτε σύνολο.

2.2 Παράσταση και μέγεθος προβλημάτων

Όπως ήδη λέχθηκε ένα πρόβλημα απόφασης είναι ένα πρόβλημα στο οποίο οι πιθανές απαντήσεις είναι δύο: ναι ή όχι. Ορίζουμε τώρα ένα τέτοιο πρόβλημα απόφασης που θα μας απασχολήσει ιδιαίτερα στην συνέχεια.

Ορισμός 2.5 Μια μεταβλητή *Boole* είναι μία μεταβλητή η οποία μπορεί να λάβει μόνο δύο τιμές: αληθής (T) ή ψευδής (F). Εισάγουμε επίσης τους τρεις λογικούς τελεστές \vee (ή ή διάζευξη), \wedge (και ή σύζευξη) και \neg (όχι ή άρνηση). Έστω V ένα αριθμήσιμο σύνολο συμβόλων λογικών μεταβλητών. Μία έκφραση *Boole* είναι μια συμβολοσειρά η οποία ορίζεται στο αλφάβητο $V \cup \{(\,,\,), \vee, \wedge, \neg\}$ σύμφωνα με τους παρακάτω κανόνες: (1) κάθε στοιχείο του V είναι έκφραση *Boole* και (2) αν x και y είναι εκφράσεις *Boole* τότε είναι επίσης και οι $(x \vee y)$, $(x \wedge y)$ και $\neg x$. Τέλος μία έκφραση *Boole* n μεταβλητών $x_1, x_2, \dots, x_n \in V$, ορίζει μία συνάρτηση n μεταβλητών $f(x_1, \dots, x_n) : \{T, F\}^n \longrightarrow \{T, F\}$ σύμφωνα με τους προφανείς κανόνες, όπου δηλαδή το «ή» δύο λογικών μεταβλητών είναι αληθές αν τουλάχιστον μία από αυτές είναι αληθής, το «και» αν και οι δύο είναι αληθείς και το «όχι» μιας μεταβλητής είναι αληθές αν και μόνο αν η μεταβλητή είναι ψευδής.

Ορισμός 2.6 Το πρόβλημα της ικανοποιησιμότητας (*satisfiability problem* ή απλά SAT) είναι το πρόβλημα απόφασης στο οποίο δίδεται μία συνάρτηση *Boole* και ζητείται να προσδιοριστεί αν υπάρχει ανάθεση λογικών τιμών στις μεταβλητές της (αποτίμηση) έτσι ώστε η τιμή της συνάρτησης να γίνει αληθής (T).

Θα ασχοληθούμε πολύ στην συνέχεια με τις δυσκολίες επίλυσης που παρουσιάζει το SAT. Επί του παρόντος ας εξετάσουμε το τεχνικό πρόβλημα της παράστασης του SAT σαν είσοδο μιας μηχανής Turing. Κατ' αρχήν ο ορισμός των στιγμιοτύπων του σαν συμβολοσειρών μας διευκολύνει κατά προφανή τρόπο στην παράσταση τους σαν είσοδο στην ταινία. Μια δυσκολία είναι η εύρεση ενός εννιαίου τρόπου παράστασης αυθαίρετα μεγάλων στιγμιοτύπων μιας και η μηχανή Turing έχει εξ' ορισμού πεπερασμένο αλφάβητο και άρα

κάποια στιγμή θα εξαντληθούν τα σύμβολα παράστασης των μεταβλητών. Είναι όμως εύκολο να δωθεί λύση σ' αυτό το πρόβλημα εισάγοντας «δείκτες» στον τρόπο παράστασης. Αυτό συνεπάγεται τον εμπλουτισμό του αλφαβήτου της μηχανής με τα δέκα ψηφία 0..9 (αν χρησιμοποιούμε το δεκαδικό σύστημα παράστασης). Για παράδειγμα η συνάρτηση Boole $(x \vee \neg y) \wedge (\neg x \vee z \vee \neg y)$ μπορεί να παρασταθεί από την συμβολοσειρά $(x1\vee\neg x2)\wedge(\neg x1\vee x2\vee\neg x3)$.

Πάνω στις ίδιες ιδέες είναι εύκολο να δούμε ότι κάθε πρόβλημα μπορεί να παρασταθεί σαν μια συμβολοσειρά πάνω σε ένα κατάλληλο αλφάβητο και συνεπώς να γίνει «νόμιμη» είσοδος για μια μηχανή Turing. Σαν παράδειγμα αναφέρουμε τα προβλήματα στα οποία μέρος της εισόδου είναι κάποιο γράφημα (πολλά τέτοια θα μας απασχολήσουν στην συνέχεια) και για τα οποία απαιτείται κάποιος τρόπος για την παράσταση του γραφήματος. Ως γνωστό ένα γράφημα $G = (V, E)$ με n κόμβους (το σύνολο V), είναι μία διμελής σχέση E (οι ακμές) πάνω στο V . Ένας φυσικός τρόπος παράστασης είναι ασφαλώς με απλή παράθεση όλων των ακμών του γραφήματος την μία κατόπιν της άλλης. Ένας άλλος τρόπος είναι μέσω του πίνακα γειτνίασης του γραφήματος (ένας πίνακας $n \times n$, όπου το στοιχείο i, j έχει τιμή 1 αν οι κόμβοι v_i και v_j συνδέονται μεταξύ τους με ακμή και 0 διαφορετικά. Σε αυτό το τρόπο παράστασης πρέπει να συμφωνηθεί κάποια σειρά, για παράδειγμα κατά στήλες. Και σε αυτό το πρόβλημα παράστασης ισχύει ότι αναφέρθηκε για το SAT ως προς τους δείκτες των κόμβων.

Ειδικά για τα προβλήματα απόφασης (όπως το SAT παραπάνω), οι δύο πιθανές απαντήσεις που μπορούν να λάβουν («NAI» ή «OXI») θυμίζει αμέσως το σχήμα αποδοχής-απόρριψης μιας συμβολοσειράς από μια μηχανή Turing. Στην πραγματικότητα αυτό δεν είναι τυχαίο: η μηχανή Turing σχεδιάστηκε για να διευκολύνει την αντιμετώπιση τέτοιων προβλημάτων. Μπορούμε μάλιστα να παρατηρήσουμε ότι τα στιγμιότυπα στα οποία η απάντηση είναι «NAI» ενός προβλήματος απόφασης συνιστούν μια καλά ορισμένη γλώσσα. Για παράδειγμα όλα τα ικανοποιήσιμα στιγμιότυπα του SAT συνιστούν μια γλώσσα (ας την συμβολίσουμε με L_{SAT}) στο αλφάβητο $\Sigma = V \cup \{ (,), \vee, \wedge, \neg \}$. Συνεπώς το πρόβλημα της εύρεσης αν μια συνάρτηση Boole f , είναι ικανοποιήσιμη είναι ισοδύναμο με το πρόβλημα του αν η συμβολοσειρά x_f ανήκει ή όχι στην γλώσσα L_{SAT} , όπου x_f είναι η συμβολοσειρά που κωδικοποιεί την f , ίσως με τον τρόπο που περιγράψαμε παραπάνω. Ισχύει όμως και το αντίστροφο: μία οποιαδήποτε γλώσσα L σε ένα αλφάβητο Σ , ορίζει ένα πρόβλημα απόφασης δηλαδή το πρόβλημα στο οποίο δίδεται μια συμβολοσειρά $x \in \Sigma$ και ζητείται αν το $x \in L$ ή όχι. Αυτό είναι το πρόβλημα απόφασης το σχετιζόμενο με την L . Εξαιτίας αυτής της άμεσης σχέσης προβλημάτων απόφασης και γλωσσών πολλές φορές θα μιλάμε για την γλώσσα του προβλήματος π.χ. την γλώσσα του SAT.

Η παράσταση των στιγμιοτύπων ενός προβλήματος μέσω συμβολοσειρών, λύνει και το πρόβλημα του αυστηρού ορισμού του μεγέθους του στιγμιοτύπου. Έτσι έχουμε:

Ορισμός 2.7 *Ορίζουμε ως μέγεθος ενός στιγμιοτύπου ενός υπολογιστικού προβλήματος το μήκος της συμβολοσειράς (δηλαδή τον αριθμό των χαρακτήρων της) η οποία το παριστάνει σε κάποια σύμβαση παράστασης.*

Ο αναγνώστης μπορεί να παρατηρήσει ότι ο παραπάνω ορισμός ενώ προσδιορίζει με ακρίβεια το μέγεθος ενός προβλήματος όταν έχει συμφωνηθεί ο τρόπος παράστασης, αφήνει ανοικτό το ποιός είναι αυτός ο τρόπος παράστασης. Όπως φάνηκε στο παράδειγμα της παράστασης ενός γραφήματος, περισσότεροι από ένας τρόποι είναι δυνατοί και συνεπώς το μέγεθος ενός στιγμιοτύπου μπορεί να μεταβάλλεται ανάλογα με τον τρόπο που επιλέγουμε να το παραστήσουμε. Η απάντηση σ' αυτό είναι ότι πράγματι έτσι συμβαίνει, εντούτοις κάθε τρόπος παράστασης είναι αποδεκτός αρκεί να μην δίνει συμβολοσειρές πολύ μεγαλύτερες από τις ελάχιστες δυνατές. Αυτή η τελευταία απαίτηση σημαίνει συμβολοσειρές μήκους $\ell(I)$, όπου I το στιγμιότυπο, πολυωνυμικά σχετιζόμενες με τις ελάχιστες δυνατές $\ell_{\min}(I)$. Δηλαδή θα πρέπει να υπάρχει κάποιο σταθερό πολυώνυμο $p(n)$, έτσι ώστε $\ell(I) \leq p(\ell_{\min}(I))$ για κάθε στιγμιότυπο I του προβλήματος. Αυτή η προτίμηση σε πολυωνυμικά σχετιζόμενες συναρτήσεις είναι διάχυτη σε όλη την Θεωρία Πολυπλοκότητας. Θεωρούμε ότι τέτοιες συναρτήσεις είναι «κοντά» μεταξύ τους (ως προς τον ρυθμό αύξησης που παρουσιάζουν) και άρα μήκη παράστασης πολυωνυμικά σχετιζόμενα με τα ελάχιστα είναι αποδεκτά. Στην πράξη η απαίτηση αυτή εξασφαλίζεται πολύ εύκολα από κάθε «λογικό» τρόπο παράστασης των στιγμιοτύπων. Το μόνο που απαιτείται είναι η παράσταση των αριθμών όπως συνήθως σε δεκαδική (ή δυαδική) μορφή. Εναλλακτικός τρόπος θα ήταν το μοναδιαίο σύστημα όπου για παράδειγμα ο αριθμός 5 παριστάνεται σαν 11111. Σε αυτή την περίπτωση όμως, το μήκος της παράστασης είναι εκθετικά μεγαλύτερο από το ελάχιστο δυνατό και άρα δεν είναι αποδεκτό. Πάντως λογαριθμικός αριθμός ψηφίων είναι απαραίτητος για την παράσταση αριθμών και συνεπώς αν δεχθούμε ότι οι αριθμοί μας μπορεί να γίνουν αυθαίρετα μεγάλοι τότε αυτά τα ψηφία πρέπει να λαμβάνονται υπόψη στην εκτίμηση του μεγέθους των αριθμητικών προβλημάτων. Αντίθετα, αν λόγω της φύσης του προβλήματος (ή έστω σιωπηρά) δεχθούμε ότι οι αριθμοί μας έχουν κάποιο σταθερό μέγεθος (π.χ. μπορούν να χωρέσουν σε μία λέξη του υπολογιστή μας) τότε μπορούμε να τους χειριστούμε σαν μια μονάδα. Έτσι, στην Εισαγωγή εξετάσαμε το πρόβλημα της ταξινόμησης και δεχθήκαμε σαν μέγεθος το πλήθος n των αριθμών που ταξινομούμε και όχι το άθροισμα των ψηφίων τους. Αυτό έγινε γιατί δεχθήκαμε ότι κάθε αριθμός ήταν αρκετά μικρός ώστε να χωράει σε μία λέξη του υπολογιστή. Σε περίπτωση που

οι αριθμοί ήταν πολύ μεγάλοι (π.χ. με μέγεθος που είναι συνάρτηση του n) τότε το μέγεθος θα έπρεπε να είναι ο αριθμός των ψηφίων τους.

2.3 Παραλλαγές της Μηχανής Turing

Η προσπάθεια για την εύρεση του «απόλυτου» υπολογιστικού μοντέλου οδήγησε τους ερευνητές σε διάφορες επεκτάσεις της μηχανής Turing. Όλες όμως αυτές οι προσπάθειες αποδεικνύονται ότι δεν προσφέρουν τίποτα ως προς το ποιά προβλήματα μπορεί να επιλύσει το βελτιωμένο μοντέλο: οι προκύπτουσες μηχανές αποδεικνύονται ισοδύναμες με την βασική μηχανή Turing. Δεν είναι όμως χωρίς αξία αυτές οι προσπάθειες. Μπορεί να μην έχει βρεθεί επέκταση που να υπολογίζει καινούργια προβλήματα, δεν ισχύει όμως το ίδιο και ως προς το πόσο αποτελεσματικά (γρήγορα κλπ) τα υπολογίζει (μέσα πάντως σε κάποια περιθώρια όχι ιδιαίτερα μεγάλα). Άλλες πάλι επεκτάσεις είναι απλώς βολικές και μας επιτρέπουν να σχεδιάζουμε μηχανές Turing πολύ ευκολότερα από ότι με το βασικό σχήμα. Κάποιες τέτοιες επεκτάσεις του βασικού μοντέλου της μηχανής Turing δίνουμε στη συνέχεια.

Ταινία άπειρη και προς τις δύο κατευθύνσεις. Δεν υπάρχει κανένας ιδιαίτερος λόγος για την ύπαρξη κάποιου ειδικού συμβόλου όπως το \triangleright σαν αρχή της ταινίας. Μια φυσική παραλλαγή επομένως είναι να θεωρήσουμε ότι η κεφαλή μπορεί να κινείται και προς τις δύο κατευθύνσεις χωρίς περιορισμούς. Πρέπει βέβαια να συμφωνηθεί πως θα δίνεται η είσοδος και που θα βρίσκεται η κεφαλή στην αρχή του υπολογισμού. Έτσι θεωρούμε ότι η αρχή της συμβολοσειράς εισόδου τοποθετείται αυθαίρετα σε κάποιο σημείο της ταινίας με την κεφαλή πάνω από το πρώτο σύμβολο. Όλα τα άλλα κύτταρα της ταινίας περιέχουν το κενό b . Για την παραλλαγή αυτή μπορεί να δειχθεί μέσω προσομοίωσης ότι υπολογίζει ακριβώς ότι και η βασική μηχανή Turing [19].

Μηχανή k -ταινιών. Μια πιο χρήσιμη παραλλαγή είναι να θεωρήσουμε ότι η μηχανή μας έχει k (ένας σταθερός αριθμός) ταινίες (και k κεφαλές, μία για κάθε ταινία) αντί της μοναδικής του βασικού σχήματος. Η μηχανή τώρα αποφασίζει την επόμενη κίνηση της σαν συνάρτηση της κατάστασης στην οποία βρίσκεται και των k συμβόλων που βρίσκονται κάτω από τις κεφαλές. Τυπικά έχουμε:

Ορισμός 2.8 Μια μηχανή Turing k -ταινιών είναι μια τετράδα $M = (K, \Sigma, \delta, s)$ όπου K είναι το πεπερασμένο σύνολο καταστάσεων, Σ το πεπερασμένο αλφάβητο που περιλαμβάνει τα σύμβολα που μπορούν να εμφανιστούν στις ταινίες και $s \in K$ η αρχική κατάσταση. Η $\delta : (K \times \Sigma^k) \longrightarrow (K \cup \{h, q_{\text{NAI}}, q_{\text{OXI}}\}) \times (\Sigma \times \{\leftarrow, \rightarrow, -\})^k$ είναι η συνάρτηση μετάβασης. Όλες οι άλλες λεπτομέρειες είναι ίδιες με της μηχανής Turing μιας ταινίας.

Έτσι αν η μηχανή k -ταινιών βρίσκεται στην κατάσταση q και η πρώτη κεφαλή διαβάσει το σύμβολο σ_1 , η δεύτερη το σ_2 κλπ τότε το $\delta(q, \sigma_1, \dots, \sigma_k) = (q', \sigma'_1, D_1, \dots, \sigma'_k, D_k)$ με τα D_i στοιχεία του συνόλου $\{\leftarrow, \rightarrow, -\}$, προσδιορίζει ότι η επόμενη κατάσταση θα είναι η q' , τα καινούργια σύμβολα που θα γραφτούν στις k ταινίες θα είναι τα $\sigma'_1, \sigma'_2, \dots, \sigma'_k$, αντίστοιχα, οι δε k κεφαλές θα κινηθούν όπως υποδηλώνουν τα D_1, D_2, \dots, D_k , αντίστοιχα. Όπως και στην περίπτωση της μιας ταινίας ιδιαίτερα χρήσιμη αποδεικνύεται η έννοια της διαμόρφωσης που τώρα βέβαια περιλαμβάνει τις συμβολοσειρές και των k ταινιών. Έτσι η διαμόρφωση $(q, u_1, v_1, u_2, v_2, \dots, u_k, v_k)$ υποδηλώνει ότι η μηχανή βρίσκεται στην κατάσταση q και στην πρώτη ταινία αριστερά της κεφαλής βρίσκεται η συμβολοσειρά u_1 και δεξιά η v_1 , στην δεύτερη ταινία αντίστοιχα οι u_2 και v_2 κ.ο.κ.

Η μηχανή Turing k -ταινιών παρέχει πολύ μεγαλύτερη ευχέρεια από την συμβατική στον σχεδιασμό ενός 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 ευκολύνοντας τον σχεδιασμό. Ο αναγνώστης ίσως να διαισθάνεται ότι πολλές μεταβλητές μπορούν να «συμπιεστούν» σε μία και άρα να αντιλαμβάνεται ότι μία μοναδική ταινία είναι αρκετή (έστω και αν δυσκολεύει τον προγραμματισμό). Πράγματι αυτό συμβαίνει αλλά αναβάλλουμε την απόδειξη για την παράγραφο 2.4.2 ώστε να μπορέσουμε παράλληλα να εξετάσουμε ποιά είναι η απώλεια σε ταχύτητα κατά την προσομοίωση.

Μηχανή πολλαπλών κεφαλών. Επιτρέπουμε πολλές κεφαλές να υπάρχουν στην ίδια ταινία και να κινούνται ανεξάρτητα. Η ίδια περίπου προσομοίωση με την προηγούμενη περίπτωση αποδεικνύει ότι και αυτή η παραλλαγή είναι ισοδύναμη με την συμβατική μηχανή Turing.

Δισδιάστατη ταινία. Στην παραλλαγή αυτή η ταινία της μηχανής είναι το επίπεδο το οποίο χωρίζεται με ένα πλέγμα σε κύτταρα που αποθηκεύουν τα σύμβολα. Και πάλι μπορεί ναδειχθεί η ισοδυναμία με το βασικό μοντέλο.

2.3.1 Η μη ντετερμινιστική μηχανή Turing

Προχωρούμε τώρα στον ορισμό μιας παραλλαγής της μηχανής Turing η οποία είναι σε διαφορετική κατεύθυνση από τις προηγούμενες. Η παραλλαγή αυτή ονομάζεται *μη ντετερμινιστική μηχανή Turing*. Η διαφορά από την συμβατική (ντετερμινιστική) μηχανή είναι ότι τώρα δίνουμε κατά κάποιο τρόπο στην μηχανή την ελευθερία να βρίσκεται ταυτόχρονα σε περισσότερα από ένα σημεία του υπολογισμού. Σπεύδουμε να σημειώσουμε ότι η μη ντετερμινιστική μηχανή Turing είναι ένα θεωρητικό μη ρεαλιστικό μοντέλο που η αξία του βρίσκεται στην επιπλέον δυνατότητα ανάλυσης της πολυπλοκότητας προβλημάτων

που μας δίνει.

Ορισμός 2.9 Μια μη ντετερμινιστική μηχανή Turing είναι μία διατεταγμένη τετράδα $M = (K, \Sigma, \Delta, s)$. Το K είναι το πεπερασμένο σύνολο καταστάσεων της μηχανής, το Σ το αλφάβητο και η $s \in K$ είναι η αρχική κατάσταση. Το Δ είναι ένα υποσύνολο του $(K \times \Sigma) \times ((K \cup \{h, q_{\text{NAI}}, q_{\text{OXI}}\}) \times \Sigma \times \{\leftarrow, \rightarrow, -\})$ και ονομάζεται σχέση μετάβασης.

Η μη ντετερμινιστική μηχανή ομοιάζει με την ντετερμινιστική σε όλα εκτός από την ύπαρξη τώρα της σχέσης μετάβασης Δ στην θέση της συνάρτησης μετάβασης δ της ντετερμινιστικής μηχανής. Η μη ντετερμινιστική μηχανή δεν έχει μία και μόνη δυνατότητα κίνησης όπως η ντετερμινιστική, αλλά έχει δυνατότητα επιλογής μεταξύ ενός αριθμού κινήσεων και γι' αυτό μιλάμε για σχέση μετάβασης. Αυτό μπορούμε να το δούμε διαφορετικά μέσω της διαμόρφωσης. Όπως και στην ντετερμινιστική μηχανή, η διαμόρφωση μιας μη ντετερμινιστικής είναι μία τριάδα (q, u, v) όπου q η κατάσταση της μηχανής, και u, v τα τμήματα της ταινίας αριστερά και δεξιά της κεφαλής. Η διαφορά έγκειται στο ότι τώρα μια διαμόρφωση (q, u, v) δίνει δυνατότητα επιλογής μεταξύ ενός αριθμού επόμενων διαμορφώσεων σύμφωνα με την σχέση μετάβασης Δ . Έτσι λέμε ότι η διαμόρφωση (q, u, v) δίνει σε ένα βήμα την διαμόρφωση (q', u', v') και το συμβολίζουμε $(q, u, v) \rightarrow (q', u', v')$ αν υπάρχει στοιχείο της σχέσης μετάβασης Δ , $((q, \sigma), (q', \sigma', D))$ τέτοιο ώστε

1. αν $D = \leftarrow$, το u' είναι το u χωρίς το τελευταίο του σύμβολο (το σ) ενώ το v' είναι το v με το σύμβολο σ' στην αρχή του (δηλαδή $v' = \sigma'v$),
2. αν $D = \rightarrow$, το u' είναι το u με το σ' να έχει αντικαταστήσει το σύμβολο σ (που ήταν το τελευταίο σύμβολο του u) και με τελευταίο του σύμβολο το πρώτο σύμβολο του v ενώ το v' είναι το v χωρίς το πρώτο του σύμβολο,
3. αν $D = -$, το u' είναι το u με το τελευταίο σύμβολο του να έχει αντικατασταθεί με το σ' ενώ το v' ισούται με το v .

Μία ανάλογη δυνατότητα σε ένα πιο οικείο μοντέλο θα ήταν να φανταστούμε μία γλώσσα προγραμματισμού που προσφέρει και μία εντολή του τύπου **goto** 10, 20, 30 όπου οι αριθμοί είναι ετικέτες κάποιων εντολών. Η εντολή αυτή θα επέτρεπε στο πρόγραμμα μας, μετά την εκτέλεση της, να μεταφερθεί ο έλεγχος σε οποιαδήποτε από τις τρεις εντολές. Είναι εύκολο να παρατηρήσουμε ότι ενώ την πρώτη φορά που εκτελείται αυτή η εντολή οι δυνατότητες είναι 3, την δεύτερη φορά γίνονται 9, μετά 27, κ.ο.κ., δηλαδή αυξάνουν εκθετικά.

Αυτό που έχει περισσότερο ενδιαφέρον στην μη ντετερμινιστική μηχανή Turing είναι ο τρόπος με τον οποίο ορίζουμε την αποδοχή μιας συμβολοσειράς εισόδου x . Η μη ντετερμινιστική μηχανή είναι ιδιαίτερα «γενναιόδωρη»

στον τρόπο αποδοχής: Έτσι λέμε ότι η x γίνεται αποδεκτή από την μη ντετερμινιστική μηχανή M αν ξεκινώντας από την αρχική διαμόρφωση, υπάρχει για την M μία ακολουθία επιλογών επομένων κινήσεων (και άρα και διαμορφώσεων) που να οδηγεί στην κατάσταση αποδοχής q_{NAI} . Αντίθετα η x δεν γίνεται αποδεκτή αν δεν υπάρχει καμία ακολουθία κινήσεων που να οδηγεί στην q_{NAI} . Υπάρχει δηλαδή μία ασυμμετρία μεταξύ αποδοχής και απόρριψης σε μια μη ντετερμινιστική μηχανή.

Ορισμός 2.10 Έστω $L \subseteq (\Sigma \setminus \{b\})^*$ μία γλώσσα. Μία μη ντετερμινιστική μηχανή Turing M αποδέχεται την L αν για κάθε συμβολοσειρά $x \in L$, υπάρχει μία τουλάχιστον ακολουθία διαμορφώσεων που οδηγεί την M στην κατάσταση q_{NAI} . Αντίθετα, για κάθε συμβολοσειρά $x \notin L$, καμία ακολουθία διαμορφώσεων δεν οδηγεί στην q_{NAI} .

Για να αντιληφθούμε την υπολογιστική δύναμη που έχουμε δώσει στις μη ντετερμινιστικές μηχανές, ας προσπαθήσουμε να δείξουμε πως μπορούμε να επιλύσουμε το SAT με μία τέτοια μηχανή. Και για να έχουμε περισσότερο κατανοητό αλγόριθμο ας τον περιγράψουμε με την βοήθεια της «πολλαπλής» εντολής **goto** που αναφέραμε παραπάνω.

```

procedure nondeterministic_SAT;
1: input  $f(x_1, \dots, x_n)$ ; {Μια συνάρτηση Boole  $n$  μεταβλητών}
2: for  $i := 1$  to  $n$  do
3:   goto 10, 20;
4:   10:  $x_i := \text{true}$ ;
5:     goto 30;
6:   20:  $x_i := \text{false}$ ;
7:   30: continue; {Συνέχισε στην επόμενη εντολή }
8: end for
9:  $\text{value} := f(x_1, x_2, \dots, x_n)$ ;
10: if  $\text{value} = \text{true}$  then
11:   return 'Ικανοποίηση'
12: end if

```

Αλγόριθμος 2: Ένας μη ντετερμινιστικός αλγόριθμος για το SAT.

Στο παραπάνω «πρόγραμμα» (Αλγόριθμος 2) η χρήση του μη ντετερμινιστικού **goto** γίνεται για αποδοθούν τιμές στις n λογικές μεταβλητές στις εντολές 10 και 20. Στην συνέχεια γίνεται έλεγχος αν οι αποδοθείσες τιμές επαληθεύουν τη συνάρτηση. Αν μία τουλάχιστον αποτίμηση την επαληθεύει τότε γι' αυτή την αποτίμηση η τιμή της value θα γίνει αλήθεια, και το πρόγραμμα θα επιστρέψει «Ικανοποίηση». Η απάντηση αυτή θα δοθεί μόνο όταν υπάρχει μία τουλάχιστον αποτίμηση που επαληθεύει την f .

Πρέπει να λεχθεί ότι η μη ντετερμινιστική μετάβαση δεν είναι μία τυχαία επιλογή (μέσω κάποιου πειράματος) μίας από τις δυνατότητες μετάβασης. Μία τέτοια πιθανοτική επιλογή γίνεται στην λεγόμενη πιθανοτική μηχανή Turing, η οποία είναι διαφορετικό μοντέλο από την μη ντετερμινιστική. Είναι ορθότερο να δούμε την μη ντετερμινιστική μετάβαση, σαν ταυτόχρονη μετάβαση και στις δύο (ή όσες είναι) επιλογές. Μπορούμε να φανταστούμε ότι το πρόγραμμα μας με την εκτέλεση του `goto 10, 20` διαιρείται σε δύο αντίγραφα ένα από τα οποία εκτελεί άλμα στην εντολή 10 και ένα που εκτελεί άλμα στην εντολή 20. Αν ένα τουλάχιστον από τα (ίσως εκθετικά πολλά) αντίγραφα που θα δημιουργηθούν αποδεχθεί, τότε αποδέχεται και το συνολικό πρόγραμμα. Ισοδύναμα, μπορούμε να πούμε ότι κάθε αντίγραφο του προγράμματος σε κάποιο σημείο έχει δημιουργήσει μία συμβολοσειρά που μπορεί να αποτελεί λύση του προβλήματος (δηλαδή απόδειξη ή πιστοποιητικό για το ότι η απάντηση στο πρόβλημα είναι «NAI») την οποία όμως στην συνέχεια πρέπει να ελέγξει ότι πράγματι αποτελεί λύση. Στην περίπτωση του SAT το «πιστοποιητικό του NAI» είναι μια αποτίμηση που επαληθεύει την δοθείσα έκφραση Boole. Άρα κατά μια έννοια η μη ντετερμινιστική μηχανή έχει την ικανότητα να μαντεύει ένα πιστοποιητικό για το «NAI», αν υπάρχει, το οποίο στη συνέχεια πρέπει να επαληθεύσει. Αυτή η λύση μέσω «μάντεψε και επαλήθευσε» είναι χαρακτηριστικό του μη ντετερμινιστικού υπολογισμού.

Λίγο πιο προσεκτικοί πρέπει να είμαστε όταν ορίζουμε τι σημαίνει ότι μια μη ντετερμινιστική μηχανή Turing υπολογίζει μια συνάρτηση f .

Ορισμός 2.11 Έστω $f : \Sigma^* \longrightarrow \Sigma^*$ μία συνάρτηση. Μία μη ντετερμινιστική μηχανή Turing M υπολογίζει την f αν για κάθε $x \in \Sigma^*$ υπάρχει μία τουλάχιστον ακολουθία μεταβάσεων που οδηγούν την μηχανή στην κατάσταση h . Σε αυτή την περίπτωση στη ταινία της πρέπει να υπάρχει το $f(x)$. Αν η μηχανή δεν καταλήξει στην h τότε πρέπει να μην τερματίσει ποτέ.

Ο λόγος για τον ορισμό αυτό είναι ότι αλλιώς μία μη ντετερμινιστική μηχανή θα μπορούσε να υπολογίσει κάθε συνάρτηση.

Ανάλογη με την ντετερμινιστική είναι και η k -ταινιών μη ντετερμινιστική μηχανή Turing. Τώρα η σχέση μετάβασης Δ είναι ένα υποσύνολο του Καρτεσιανού γινομένου $(K \times \Sigma) \times ((K \cup \{h, q_{NAI}, q_{OXI}\}) \times (\Sigma \times \{\leftarrow, \rightarrow, -\})^k)$. Αποδοχή έχουμε και πάλι όταν υπάρχει μία τουλάχιστον ακολουθία επιλογών κινήσεων που να οδηγεί στην κατάσταση q_{NAI} .

2.3.2 Η καθολική μηχανή Turing

Παρατηρήσαμε ήδη ότι η μηχανή Turing διαφέρει ως προς τους γνωστούς μας υπολογιστές στο ότι έχει «σταθερή» συμπεριφορά, δεν μπορεί να προγραμματιστεί. Στην παράγραφο αυτή θα δείξουμε πως και το μειονέκτημα τους αυτό

μπορεί να ξεπεραστεί. Θα αρχίσουμε περιγράφοντας μια μέθοδο με την οποία παριστάνουμε μηχανές Turing σαν συμβολοσειρές ενός αλφαβήτου, έστω του $\{0, 1\}$.

Έστω λοιπόν $M = (K, \Sigma, \delta, s)$ μια οποιαδήποτε μηχανή Turing. Τα πρώτα στοιχεία της που θα κωδικοποιήσουμε είναι τα σύμβολα του αλφαβήτου της. Η κωδικοποίηση θα γίνει χρησιμοποιώντας μόνο το 0. Διατάσσουμε με κάποιο αυθαίρετο τρόπο το $\Sigma = \{\sigma_1, \sigma_2, \dots, \sigma_{|\Sigma|}\}$ και κωδικοποιούμε το σύμβολο σ_i με την συμβολοσειρά 0^i . Τα τρία σύμβολα μετάβασης \leftarrow, \rightarrow και $-$ που δεν ανήκουν στο Σ κωδικοποιούνται με τις συμβολοσειρές 0, 00 και 000 αντίστοιχα. Παρόμοια κωδικοποιούμε και τις καταστάσεις, αφού διατάζουμε το σύνολο K , φροντίζοντας η πρώτη κατάσταση στη διάταξη να είναι η αρχική κατάσταση s , ενώ με τις επόμενες συμβολοσειρές κωδικοποιούμε τις τελικές καταστάσεις h , q_{NAI} και q_{OXI} . Ο παρακάτω πίνακας συνοψίζει τις συμβάσεις αυτές:

σύμβολο	κωδικοποίηση	κατάσταση	κωδικοποίηση
σ_i	0^i	q_i	0^i
\leftarrow	0	h	$0^{ K +1}$
\rightarrow	00	q_{NAI}	$0^{ K +2}$
$-$	000	q_{OXI}	$0^{ K +3}$

Πίνακας 2.1: Συμβάσεις για την κωδικοποίηση των παραμέτρων μιας μηχανής Turing.

Βασίζόμενοι στις παραπάνω συμβάσεις κωδικοποιούμε τις μεταβάσεις της M : Η μετάβαση $\delta(q_i, \sigma_j) = (q_k, \sigma_l, D_m)$ (όπου $m = 1, 2, 3$ αν η μετάβαση προσδιορίζεται από τα σύμβολα \leftarrow, \rightarrow και $-$ αντίστοιχα), κωδικοποιείται σαν:

$$10^i 10^j 10^k 10^l 10^m 1$$

Παρατηρούμε ότι το τι παριστάνει ακριβώς κάθε ακολουθία μηδενικών προσδιορίζεται από την θέση της στην παραπάνω συμβολοσειρά.

Το τελευταίο στάδιο στην κωδικοποίηση της M είναι η παράθεση των συμβολοσειρών κωδικοποίησης όλων των μεταβάσεων που περιγράφει η δ . Με τον τρόπο αυτό η περιγραφή μιας συγκεκριμένης μετάβασης οριοθετείται από δύο διαδοχικά 11 και άρα είναι εύκολο να επισημανθεί.

Πρέπει να τονιστεί ότι η σύμβαση κωδικοποίησης που επιλέξαμε είναι αυθαίρετη και οποιαδήποτε άλλη θα ήταν αποδεκτή αρκεί να κωδικοποιούσε μονοσήμαντα μια μηχανή Turing. Μια αποδεκτή κωδικοποίηση πρέπει να μας επιτρέπει δοθείσης μιας συμβολοσειράς στο $\{0, 1\}^*$, να μπορούμε να αποφανθούμε (1) αν είναι «νόμιμη», αν δηλαδή κωδικοποιεί κάποια μηχανή και (2) ποιά

είναι αυτή η μηχανή. Είναι προφανές ότι ο προταθείς τρόπος το εξασφαλίζει αυτό.

Παράδειγμα 2.3 Τα παραπάνω φαίνονται καθαρά στο παρακάτω παράδειγμα που περιγράφει την κωδικοποίηση της μηχανής Turing του Παραδείγματος 2.1, η συνάρτηση μετάβασης του οποίου ορίζεται σύμφωνα με τον ακόλουθο πίνακα:

q	σ	$\delta(q, \sigma)$
s	\triangleright	$(s, \triangleright, \rightarrow)$
s	a	$(q_{\text{NAI}}, a, -)$
s	b	(s, b, \rightarrow)
s	\flat	$(q_{\text{OXI}}, \flat, -)$

Σύμφωνα με τις συμβάσεις που έχουμε κάνει, τα σύμβολα και οι καταστάσεις κωδικοποιούνται κατά τα παραπάνω σύμφωνα με τους πίνακες που ακολουθούν.

σύμβολο	κωδικοποίηση	κατάσταση	κωδικοποίηση
\triangleright	0	s	0
\flat	00	h	00
a	000	q_{NAI}	000
b	0000	q_{OXI}	0000

Είναι εύκολο να δούμε ότι η πρώτη μετάβαση της μηχανής κωδικοποιείται μέσω της συμβολοσειράς 101010101001, η δεύτερη μέσω της 1010001000100010001, η τρίτη μέσω της 101000010100001001 και η τέταρτη μετάβαση μέσω της συμβολοσειράς 101001000010010001. Η κωδικοποίηση όλης της μηχανής τώρα είναι η παράθεση των τεσσάρων παραπάνω συμβολοσειρών.

Η κωδικοποίηση μιας μηχανής Turing M σαν μια συμβολοσειρά στο αλφάβητο $\{0, 1\}$ μας επιτρέπει τώρα να δώσουμε την μηχανή σαν *είσοδο* σε άλλες μηχανές. Ο στόχος μας είναι να περιγράψουμε μια μηχανή Turing, την οποία θα ονομάσουμε *καθολική* (universal) και η οποία θα μπορεί να παίρνει σαν είσοδο μια κωδικοποιημένη μηχανή Turing M καθώς και την είσοδο της w (ίσως και αυτή κωδικοποιημένη ώστε το αλφάβητο της U να είναι το $\{0, 1\}$) και να μιμείται τον υπολογισμό της M με είσοδο το w . Μια τέτοια μηχανή είναι στην ουσία «προγραμματιζόμενη» με την έννοια ότι η συμπεριφορά της αλλάζει ανάλογα με την κωδικοποιημένη μηχανή M που έχει στην είσοδο της.

Η μηχανή U θα έχει τρεις ταινίες. Στην πρώτη που είναι η ταινία εισόδου δίδεται η M κωδικοποιημένη και στην συνέχεια το w . Η δεύτερη ταινία θα

κρατά το περιεχόμενο της ταινίας της M σε όλη την διάρκεια του υπολογισμού. Στην τρίτη ταινία θα γράφεται σε ποιά κατάσταση της M βρίσκεται η προσομοίωση. Η λειτουργία της U είναι η ακόλουθη: στην αρχή αντιγράφει το τμήμα της εισόδου της που κωδικοποιεί το w στην δεύτερη ταινία και μετά τοποθετεί την κεφαλή πάνω από το πρώτο σύμβολο του w . (Στην περίπτωση που είναι κωδικοποιημένο, πάνω από το πρώτο σύμβολο του κώδικα του συμβόλου.) Στην τρίτη ταινία της γράφεται ο κώδικας της αρχικής κατάστασης της M (το 0). Στην συνέχεια κάθε κίνηση της M προσομοιώνεται σε δύο φάσεις. Κατά την πρώτη φάση η κεφαλή της πρώτης ταινίας κινείται σε όλο το μήκος της στο οποίο είναι κωδικοποιημένη η M και ελέγχονται όλες οι μεταβάσεις, ώστε να βρεθεί μετάβαση με κατάσταση αυτή που είναι γραμμένη στη τρίτη ταινία και σύμβολο αυτό που υπάρχει κάτω από την δεύτερη κεφαλή. Έχοντας αναγνωρίσει την κατάσταση και το σύμβολο κάτω από την κεφαλή της M , η U προχωρεί και βλέπει ποιά είναι η κίνηση που κάνει η M με αυτές τις παραμέτρους. Τα στοιχεία αυτά βρίσκονται στον κώδικα της M στην πρώτη ταινία. Κατά την δεύτερη φάση, η U ενημερώνει την τρίτη ταινία της με την επόμενη κατάσταση της M και την δεύτερη ταινία της με τον κωδικό του καινούργιου συμβόλου. Αυτή η τελευταία κίνηση απαιτεί ίσως την μερική ολισθήση της δεύτερης ταινίας δεξιά, ώστε να ελευθερωθεί χώρος για το καινούργιο σύμβολο, (αν αυτό κωδικοποιείται με περισσότερα 0) ή αριστερά (αν κωδικοποιείται με λιγότερα). Οι ολισθήσεις αυτές δεν ενοχλούν στον βαθμό που δεν ενδιαφέρει ο χρόνος μιας σάρωσης ίσως και ολόκληρης της δεύτερης ταινίας για κάθε μία κίνηση της M . Σε αντίθετη περίπτωση μπορεί να αποφευχθεί, εγγράφοντας κάθε κώδικα συμβόλου s_i του w σε ακριβώς $|\Sigma|$ κύτταρα, καταλαμβάνοντας τα $|\Sigma| - i$ κύτταρα με κενά. Έτσι κάθε αλλαγή συμβόλου της δεύτερης ταινίας απαιτεί κίνηση μέσα σε ένα διάστημα $|\Sigma|$ (ένας σταθερός αριθμός) το πολύ κυττάρων. Τέλος, η κεφαλή της δεύτερης ταινίας τοποθετείται πάνω από το σύμβολο που προβλέπει η μετάβαση. Είναι εύκολο να δούμε ότι η U μιμείται πλήρως την λειτουργία της M με είσοδο το w και επομένως ότι η έξοδος της είναι η ίδια με την έξοδο της M με είσοδο το w .

Παρατήρηση 2.2 Τελειώνουμε αυτή την παράγραφο με την παρατήρηση ότι είναι εύκολο να επεκταθεί η ιδέα της κωδικοποίησης και σε μηχανές Turing πολλών ταινιών (άσκηση) και κατά συνέπεια να κατασκευαστεί καθολική μηχανή που να προσομοιώνει μηχανές πολλών ταινιών. Η δυσκολία είναι τώρα ότι η μηχανή U θα πρέπει να κρατά το περιεχόμενο πολλών ταινιών της M . Η αύξηση του αριθμού των ταινιών της U σε $k + 2$ (όπου k ο αριθμός των ταινιών της M) δεν είναι λύση διότι τότε η U θα έχει μεταβλητό αριθμό ταινιών ανάλογα με την μηχανή που προσομοιώνει και άρα δεν θα είναι πραγματικά «καθολική». Ευτυχώς, όπως αναφέρθηκε παραπάνω, η M μπορεί να μετασχηματιστεί ώστε να έχει σταθερό αριθμό ταινιών (μία ή ακόμη και δύο για λόγους

ταχύτητας στην προσομοίωση όπως θα δούμε παρακάτω) και άρα η U να έχει σταθερό αριθμό ταινιών.

2.4 Υπολογισμοί με περιορισμούς

Η μελέτη της πολυπλοκότητας των διαφόρων προβλημάτων με την βοήθεια της μηχανής Turing, απαιτεί τον ορισμό του χρόνου που διαρκεί ένας υπολογισμός καθώς και του χώρου που απαιτεί και ίσως και άλλων υπολογιστικών πόρων που μας ενδιαφέρουν.

Ορισμός 2.12 *Ο χρόνος που απαιτεί μια ντετερμινιστική μηχανή M με είσοδο το x είναι ο αριθμός των βημάτων (κινήσεων) που θα κάνει η μηχανή ξεκινώντας από την αρχική διαμόρφωση για τον υπολογισμό του $M(x)$. Αν η M αποκλίνει στο x τότε ο χρόνος είναι ∞ .*

Περισσότερο προσεκτικοί πρέπει να είμαστε όταν ορίζουμε τον χρόνο που κάνει μια μη ντετερμινιστική μηχανή. Όπως λέχθηκε μια μη ντετερμινιστική μηχανή μπορεί να αποδέχεται την συμβολοσειρά εισόδου για κάποια ακολουθία επιλογών, αλλά και να κάνει μερικές επιλογές οι οποίες να την οδηγούν σε απόρριψη της ή ακόμη και να μην σταματήσει καθόλου.

Ορισμός 2.13 *Ο χρόνος που απαιτεί μια μη ντετερμινιστική μηχανή M με είσοδο το x είναι ο μικρότερος αριθμός βημάτων που μπορεί να κάνει η μηχανή ξεκινώντας από την αρχική διαμόρφωση για να φτάσει στην κατάσταση q_{NAI} . Συμβατικά, σε κάθε άλλη περίπτωση ο χρόνος της μηχανής με είσοδο το x ισούται με τη μονάδα.*

Έχοντας ορίσει τι σημαίνει χρόνος υπολογισμού μιας συμβολοσειράς, μπορούμε να ορίσουμε την πολυπλοκότητα χρόνου μιας μηχανής Turing.

Ορισμός 2.14 *Η πολυπλοκότητα χρόνου $T_M(n)$ μιας μηχανής Turing M (ντετερμινιστικής ή μη) είναι μια συνάρτηση από τους φυσικούς στους φυσικούς που ορίζεται όπως παρακάτω:*

$$T_M(n) = \max \{ m : \exists x \in (\Sigma \setminus \{b\})^*, |x| = n \text{ έτσι ώστε} \\ \text{ο χρόνος που απαιτεί η } M \text{ με είσοδο το } x \text{ είναι } m \}.$$

Ο χρόνος επομένως που απαιτεί μια μηχανή με είσοδο μήκους n είναι ο μέγιστος πάνω σε όλες τις εισόδους μήκους n . Συνήθως υποθέτουμε ότι $T_M(n) \geq n$ γιατί μια μηχανή χρειάζεται χρόνο τουλάχιστον n σε είσοδο μεγέθους n , ώστε να διαβάσει όλα τα σύμβολα της εισόδου. Αυτό δεν είναι

αναγκαίο, είναι όμως μια σύμβαση που κάνουμε ώστε να αποφεύγουμε να εξετάζουμε τετριμμένες μηχανές.

Ενδιαφέρον έχει επίσης να δούμε τι σημαίνει πολυπλοκότητα χρόνου μιας μη ντετερμινιστικής μηχανής. Ο Ορισμός 2.13 σε συνδυασμό με τον 2.14 μας λέει ότι η τιμή της πολυπλοκότητας χρόνου στο n μιας μη ντετερμινιστικής μηχανής είναι ο μέγιστος χρόνος που απαιτεί η μηχανή πάνω σε όλες τις συμβολοσειρές μήκους n που η μηχανή αποδέχεται. Δηλαδή στην μη ντετερμινιστική μηχανή αδιαφορούμε για τις συμβολοσειρές όπου η μηχανή δεν αποδέχεται (δηλαδή όλοι οι υπολογισμοί σε αυτές καταλήγουν σε ΟΧΙ ή έχουν άπειρο μήκος).

Ο υπολογισμός μιας μηχανής Turing με περιορισμό στον χώρο (μνήμη) ορίζεται ανάλογα, με μια μόνο μικρή παραλλαγή. Κατ' αρχή είναι φυσικό να επιλέξουμε σαν μοντέλο υπολογισμού την μηχανή Turing k -ταινιών. Σε μια πρώτη προσέγγιση είναι επίσης φυσικό να επιχειρήσουμε να ορίσουμε σαν χώρο που απαιτεί μια μηχανή Turing με είσοδο το x , το άθροισμα των κυττάρων που χρησιμοποιεί η μηχανή και στις k -ταινίες της. Μια προσεκτικότερη ματιά όμως εύκολα φανερώνει ότι με αυτό τον τρόπο χάνεται η ικανότητα του μοντέλου μας να ξεχωρίσει ανάμεσα σε προβλήματα που έχουν σημαντικές διαφορές στις απαιτήσεις τους σε μνήμη.

Ας υποθέσουμε ότι έχουμε ένα πρόβλημα στο οποίο η απάντηση είναι ιδιαίτερα μεγάλη (έχει μεγάλο μήκος). Κατά συνέπεια η ανάγκη καταγραφής αυτής της απάντησης σε μία ταινία εξόδου, θα χρέωνε τον υπολογισμό μας με μεγάλη μνήμη την οποία δεν χρησιμοποίησε για την παραγωγή της εξόδου αλλά για την έξοδο αυτή καθ' αυτή. Συνεπώς κάποιος καλύτερος από πλευράς μνήμης αλγόριθμος δεν θα ήταν δυνατό να χαρακτηριστεί αν προστίθετο στην πολυπλοκότητα χώρου και το μήκος της εξόδου. Τα ίδια ισχύουν και για το μήκος της εισόδου: αν μετρηθεί στην συνολική μνήμη μπορεί να οδηγήσει σε λάθος αποτέλεσμα. Διαισθητικά, αυτό που θέλουμε να μετρήσουμε είναι πόσο «πρόχειρο» χρησιμοποιεί ο αλγόριθμος μας (η μηχανή μας) για τον υπολογισμό του. Πρέπει να επισημάνουμε ότι εδώ δεν ισχύει ότι στην περίπτωση της πολυπλοκότητας χρόνου όπου, όπως αναφέρθηκε, μικρότεροι ρυθμοί αύξησης από της γραμμικής συνάρτησης δεν είναι παραδεκτοί: χρειάζεται τουλάχιστον $O(n)$ χρόνος για κάθε μηχανή που δεν κάνει τετριμμένη δουλειά.

Έτσι λοιπόν ορίζουμε τη μηχανή k -ταινιών με είσοδο και έξοδο σαν μια μηχανή στην οποία η πρώτη από τις ταινίες της (η ταινία εισόδου) είναι μόνο ανάγνωσης και η τελευταία (η ταινία εξόδου) μόνο εγγραφής. Ο χώρος που απαιτεί μια μηχανή Turing με είσοδο μια συμβολοσειρά x , είναι το άθροισμα των κυττάρων που χρησιμοποιεί η μηχανή στις υπόλοιπες $k - 2$ ταινίες. Στην περίπτωση της μη ντετερμινιστικής μηχανής είναι το μικρότερο τέτοιο άθροισμα για να φτάσει η μηχανή στην κατάσταση q_{NAI} . Συμβατικά, σε κάθε άλλη

περίπτωση ο χώρος της μηχανής με είσοδο το x ισούται με τη μονάδα. Έχουμε λοιπόν:

Ορισμός 2.15 Η πολυπλοκότητα χώρου $S_M(n)$ μιας μηχανής Turing M (ντετερμινιστικής ή μη) είναι μια συνάρτηση από τους φυσικούς στους φυσικούς που ορίζεται όπως παρακάτω:

$$S_M(n) = \max \{ m : \exists x \in (\Sigma \setminus \{b\})^*, |x| = n \text{ έτσι ώστε} \\ \text{ο χώρος που απαιτεί η } M \text{ με είσοδο το } x \text{ είναι } m \}.$$

2.4.1 Θεωρήματα επιτάχυνσης και συμπίεσης της μνήμης

Στην Εισαγωγή επιχειρηματολογήσαμε πάνω στην παράσταση της πολυπλοκότητας αλγορίθμων και προβλημάτων, χωρίς πολλαπλασιαστικές και προσθετικές σταθερές. Παρόλο που σημαντικός λόγος γι' αυτό είναι η αδυναμία μας να υπολογίζουμε αυτές τις σταθερές, αναφέραμε και σαν αιτία την ραγδαία βελτίωση των επιδόσεων των υπολογιστών (χρόνος, μνήμη) κάτι που κάνει τις μεταβολές στις σταθερές ενός αλγορίθμου να συγχέονται με την βελτίωση των υπολογιστών. Στην παράγραφο αυτή δίνουμε δύο θεωρητικά αποτελέσματα που δικαιολογούν την επιλογή μας αυτή.

Θεώρημα 2.1 Αν μία γλώσσα L αποφασίζεται από μία μηχανή Turing σε χρόνο $f(n)$, τότε για οποιοδήποτε $\epsilon > 0$, η L αποφασίζεται σε χρόνο $\epsilon f(n) + n + 1$.

Απόδειξη: Έστω M η μηχανή πολυπλοκότητας χρόνου $f(n)$ που αποφασίζει την L . Θα περιγράψουμε μια μηχανή M' η οποία εργάζεται σε χρόνο $\epsilon f(n) + n + 1$. Η M' θα έχει ένα πολύ πλουσιότερο αλφάβητο και πολύ περισσότερες καταστάσεις από την M . Κάθε σύμβολο της M' θα παριστάνει ένα διάνυσμα d συμβόλων της M (το d είναι μία παράμετρος στην οποία θα δώσουμε τιμή αργότερα). Για τον σκοπό αυτό το αλφάβητο της M' θα περιέχει, εκτός του Σ και ακόμη $|\Sigma|^d$ σύμβολα, ένα για κάθε d -διαστατο διάνυσμα συμβόλων του Σ . Η M' αρχίζει την λειτουργία της «συμπιέζοντας» d σύμβολα της M σε ένα: σαρώνει την ταινία εισόδου της και για κάθε d σύμβολα εισόδου που διαβάζει γράφει ένα σύμβολο από το Σ^d σε μία δεύτερη ταινία. Αυτό το πετυχαίνει χρησιμοποιώντας την τεχνική της παρατήρησης της Σελίδας 14. Αποθηκεύει στις καταστάσεις της τα d σύμβολα καθώς η πρώτη της κεφαλή τα διαβάζει και συνεπώς για τον σκοπό αυτό αρκούν $|\Sigma|^d$ καταστάσεις ακόμα. Ο χρόνος αυτής της διαδικασίας είναι $n + 1$ βήματα μια και όλοι οι υπολογισμοί γίνονται στον πεπερασμένο έλεγχο της M' . Στην συνέχεια χρησιμοποιεί την δεύτερη ταινία σαν ταινία εισόδου και την πρώτη σαν κανονική ταινία εργασίας. Από εδώ

και πέρα όλες οι ταινίες της M εμφανίζονται «συμπιεσμένες» σε αντίστοιχες ταινίες της M' . Για την προσομοίωση η M' εργάζεται σε φάσεις. Σε κάθε φάση κινεί τις κεφαλές της μία θέση αριστερά, μετά δύο θέσεις δεξιά και πάλι μια θέση αριστερά. Με τον τρόπο αυτό η M' αποθηκεύει στις καταστάσεις της το περιεχόμενο $3d$ κυττάρων από κάθε ταινία της M . Στις καταστάσεις της αποθηκεύει και τη θέση των κεφαλών της M μέσα στο διάστημα d κυττάρων κάτω από τις κεφαλές της. Στην συνέχεια d κινήσεις της M προσομοιώνονται σε δύο μόνο κινήσεις της M' , γιατί σε d κινήσεις, η M το πολύ να επισκεφθεί ένα από τα γειτονικά διαστήματα d συμβόλων της M' . Κατά συνέπεια μία φάση της M' διαρκεί το πολύ 6 βήματα. Ο υπολογισμός των μεταβάσεων της M δεν απαιτεί καθόλου επιπλέον χρόνο γιατί γίνεται στον πεπερασμένο έλεγχο της M' . Συνεπώς ο χρόνος λειτουργίας της M' θα είναι $6\lceil \frac{f(n)}{d} \rceil + n + 1$. Επιλέγοντας $d = 6/\epsilon$ καταλήγουμε στο ζητούμενο. \square

Το Θεώρημα 2.1 (Θεώρημα γραμμικής επιτάχυνσης) μας επιτρέπει να αφαιρούμε από όρια χρόνου τις πολλαπλασιαστικές σταθερές καθώς και προσθετέους με μικρότερο ρυθμό αύξησης. Αντίστοιχο θεώρημα ισχύει και για όρια χώρου:

Θεώρημα 2.2 *Αν μία γλώσσα L αποφασίζεται από μία μηχανή Turing σε χώρο $f(n)$, τότε για οποιοδήποτε $\epsilon > 0$, η L αποφασίζεται σε χώρο $\epsilon f(n)$.*

Απόδειξη: Η απόδειξη είναι ανάλογη του Θεωρήματος 2.1. Σημειώνουμε ότι η προσθετική σταθερά $n + 1$ δεν είναι πλέον απαραίτητη στο χώρο της M' μιας και ο χώρος της ταινίας εισόδου δεν υπολογίζεται σε μια μηχανή με είσοδο και έξοδο. \square

Τέλος ακριβώς αντίστοιχα θεωρήματα μπορούν να δοθούν για επιτάχυνση και συμπίεση της μνήμης μιας μη ντετερμινιστικής μηχανής Turing.

2.4.2 Μείωση στον αριθμό των ταινιών

Μπορούμε τώρα να επανέλθουμε στο ζήτημα της προσομοίωσης μιας μηχανής k -ταινιών από μια μηχανή με μοναδική ταινία και να δείξουμε το αποτέλεσμα που υποσχεθήκαμε στην ενότητα 2.3.

Θεώρημα 2.3 *Αν μια γλώσσα L αποφασίζεται από μια μηχανή Turing k -ταινιών σε χρόνο $f(n)$, τότε αποφασίζεται και από μια μηχανή μιας ταινίας σε χρόνο $f(n)^2$.*

Απόδειξη: Έστω M μια μηχανή Turing k -ταινιών με πολυπλοκότητα χρόνου $f(n)$. Η προσομοίωση της από μια συμβατική μηχανή M' , απαιτεί την αποτύπωση των k -ταινιών και των αντίστοιχων κεφαλών στην μοναδική ταινία της

M' . Αυτό το επιτυγχάνουμε ως εξής: Θεωρούμε ότι η ταινία της M' χωρίζεται σε $2k$ «αυλάκια». Τα k περιττά αυλάκια θα είναι πανομοιότυπα με τις k ταινίες της M . Έτσι η πρώτη ταινία της M καταγράφεται στο πρώτο αυλάκι της M' , η δεύτερη ταινία στο τρίτο αυλάκι κ.ο.κ. Τα αντίστοιχα άρτια αυλάκια περιέχουν σε όλα τα κύτταρα τους κενά εκτός από ένα και μόνο κύτταρο στο οποίο υπάρχει ένα διαφορετικό σύμβολο, έστω το 1. Ο σκοπός αυτού του συμβόλου είναι να επισημαίνει την θέση της αντίστοιχης κεφαλής. Έτσι στο δεύτερο αυλάκι υπάρχει 1 στην θέση της πρώτης κεφαλής, στο τέταρτο αυλάκι στην θέση της δεύτερης κεφαλής κλπ. Η εικόνα συνεπώς της μοναδικής ταινίας της μηχανής μοιάζει με το Σχήμα 2.2. Η M' καταχωρεί τα $2k$ κύτταρα μιας θέσης της ταινίας της σε ένα μόνο σύμβολο. Για να το πετύχει αυτό το αλφάβητο της M' , έστω Σ' είναι το $\Sigma \times \{b, 1\} \times \Sigma \times \{b, 1\}, \dots, \Sigma \times \{b, 1\}$ όπου υπάρχουν k εμφανίσεις του Σ . Δηλαδή μια «στήλη» $2k$ συμβόλων της ταινίας συμβολίζεται με ένα και μόνο σύμβολο του Σ' . Η προσομείωση μιας κίνησης της M γίνεται με μια σάρωση όλης της ταινίας της M' από την τρέχουσα θέση μέχρι την θέση της τελευταίας (δεξιότερης) κεφαλής της M και πάλι πίσω. Με την προς τα δεξιά σάρωση η M' επισημαίνει τα σύμβολα με τα 1 (τις θέσεις των k κεφαλών της M) και αποθηκεύει στην κατάσταση της τα σύμβολα κάτω από τις k κεφαλές. Έχοντας ήδη αποθηκευμένη και την κατάσταση της M μπορεί να υπολογίσει την επόμενη κίνηση της. Κατά τη σάρωση προς τα αριστερά και μέχρι την θέση της αριστερότερης κεφαλής αλλάζει τα σύμβολα στις θέσεις που αντιστοιχούν στις κεφαλές της M' και ενδεχομένως ένα από τα αμέσως γειτονικά τους ώστε να αποδίδεται το γεγονός ότι κάποια από τις κεφαλές μετακινήθηκε εκεί. Συνεπώς αν η αριστερότερη και η δεξιότερη κεφαλή της M βρίσκονται σε απόσταση m μεταξύ τους, μια κίνηση της M απαιτεί $2m + 2$ κινήσεις της M' . Εφόσον όμως η M τρέχει σε χρόνο $f(n)$, η αριστερότερη και η δεξιότερη κεφαλή της θα βρεθούν σε απόσταση το πολύ $f(n)$ σε οποιοδήποτε σημείο του υπολογισμού της. Συνεπώς ο χρόνος της μηχανής M' είναι $f(n)(2f(n) + 2) \leq 3f(n)^2$. Ο χρόνος αυτός μπορεί να γίνει ακριβώς $f(n)^2$ αν παρατηρήσουμε ότι σε ένα προκαταρκτικό στάδιο μπορούμε να μετατρέψουμε την M με βάση το Θεώρημα 2.1 ώστε να τρέχει σε χρόνο $f(n)/\sqrt{3}$. Τότε η M' θα τρέχει σε χρόνο $f(n)^2$. \square

Πόρισμα 2.1 Αν μια γλώσσα L αποφασίζεται από μια μη ντετερμινιστική μηχανή Turing k -ταινιών σε χρόνο $f(n)$, τότε αποφασίζεται και από μια μηχανή μιας ταινίας σε χρόνο $f(n)^2$.

Απόδειξη: Παρόμοια με την απόδειξη του παραπάνω θεωρήματος. \square

Με περισσότερο προσεκτική προσομείωση μπορούμε να δείξουμε ότι δύο ταινίες προσομοιώνουν πολύ αποτελεσματικότερα μια μηχανή με $k > 2$ ταινίες:

a	b	c	d	a	b	c	a
			1				
a	a	c	d	b	a	d	d
1							
a	a	a	c	b	b	d	c
						1	

Σχήμα 2.2: Προσομοίωση μιας μηχανής Turing k ταινιών από μηχανή μιας ταινίας.

Θεώρημα 2.4 Αν μια γλώσσα L αποφασίζεται από μια μη ντετερμινιστική μηχανή Turing k -ταινιών, $k > 2$, σε χρόνο $f(n)$, τότε γίνεται αποδεκτή και από μια μηχανή δύο ταινιών σε χρόνο $f(n) \log f(n)$.

Απόδειξη: Η απόδειξη δόθηκε στο [9]. Περιγράφεται επίσης στο βιβλίο των Hopcroft και Ullman [10]. \square

2.5 Η Μηχανή Τυχαίας Προσπέλασης, RAM

Παρουσιάζουμε τώρα ένα άλλο υπολογιστικό μοντέλο, την *RAM*, (Random Access Machine) ή *Μηχανή Τυχαίας Προσπέλασης*. Η RAM δεν είναι μία ακόμη παραλλαγή της μηχανής Turing αλλά ένα τυπικό μοντέλο που μοιάζει πολύ με τους σύγχρονους υπολογιστές. Η αξία του μοντέλου δεν βρίσκεται τόσο στο αποτέλεσμα ισοδυναμίας με την μηχανή Turing που θα δείξουμε, (ο αναγνώστης θα πρέπει ήδη να έχει πειστεί ότι η μηχανή Turing υπολογίζει ότι και ένας σύγχρονος υπολογιστής) αλλά κυρίως στον χρόνο που απαιτεί η προσομοίωση του από την μηχανή Turing. Ο χρόνος αυτός είναι ένα μικρό πολυώνυμο του χρόνου που απαιτεί η μηχανή Turing.

2.5.1 Περιγραφή-Ορισμός

Όπως η μηχανή Turing χρησιμοποιεί την ταινία της σαν μνήμη, η RAM έχει ένα απεριόριστο αριθμό από καταχωρητές r_{-1}, r_0, r_1, \dots σαν στοιχεία μνήμης. Κάθε καταχωρητής μπορεί να αποθηκεύσει ένα ακέραιο αριθμό αυθαίρετα μεγάλο. Οι δύο πρώτοι καταχωρητές έχουν ιδιαίτερο ρόλο και όνομα. Ο καταχωρητής r_{-1} λέγεται *απαριθμητής προγράμματος* και συμβολίζεται με PC.

Ο καταχωρητής r_0 λέγεται *συσσωρευτής* και συμβολίζεται με AC. Ένα πρόγραμμα της RAM $\Pi = (\pi_1, \pi_2, \dots, \pi_k)$ είναι μία πεπερασμένη ακολουθία λειτουργιών RAM, όπου κάθε λειτουργία μπορεί να είναι μία από τις λειτουργίες που περιλαμβάνονται στον Πίνακα 2.2. Όλες οι λειτουργίες της RAM, πλην της τελευταίας η οποία σταματά την εκτέλεση του προγράμματος, παίρνουν ένα όρισμα. Το όρισμα x μπορεί να έχει τρεις μορφές: « k », « $= k$ » και « $*k$ », όπου k κάποιος ακέραιος αριθμός. Όταν το x είναι « k », τότε όρισμα είναι το περιεχόμενο του καταχωρητή με δείκτη k δηλαδή του r_k . Όταν το x είναι « $*k$ », τότε υπονοείται το περιεχόμενο του καταχωρητή του οποίου ο δείκτης είναι ο r_k δηλαδή του r_{r_k} . Τέλος όταν το x είναι « $= k$ » τότε όρισμα είναι ο ίδιος ο ακέραιος k . Ειδικά για την λειτουργία STORE (μεταφέρει το περιεχόμενο του συσσωρευτή στον καταχωρητή που υποδεικνύει το όρισμα), δεν έχει έννοια ένα όρισμα « $= k$ » και συνεπώς δεν υπάρχει. Όλες οι εντολές που έχουν το όρισμα x , παράλληλα με την κύρια λειτουργία τους, αυξάνουν και τον αριθμητή προγράμματος κατά ένα. Αυτό αντανακλά στην ακολουθιακή εκτέλεση των προγραμμάτων στους υπολογιστές. Οι εντολές JUMP και JNEG (άλμα χωρίς και με συνθήκη αντίστοιχα) επηρεάζουν μόνο το περιεχόμενο του αριθμητή προγράμματος. Στην RAM τα δεδομένα θεωρούμε ότι δίδονται στους καταχωρητές αρχίζοντας από τον r_1 και καταλαμβάνοντας διαδοχικά όσους καταχωρητές απαιτεί το στιγμιότυπο του προβλήματος. Η αρχική τιμή του PC είναι 1. Όλοι οι άλλοι καταχωρητές έχουν αρχική τιμή 0.

Λειτουργία		Περιγραφή	
LOAD	x	AC:= x	και PC:=PC+1
STORE	x	x :=AC	και PC:=PC+1
ADD	x	AC:=AC+ x	και PC:=PC+1
SUB	x	AC:=AC- x	και PC:=PC+1
JUMP	k	PC:= k	
JNEG	k	if AC<0 then PC:= k	
HALT			

Πίνακας 2.2: Οι λειτουργίες της RAM

Οι λειτουργίες της RAM είναι λοιπόν ένα μικρό υποσύνολο των εντολών μηχανής ενός υπολογιστή. Το κύριο χαρακτηριστικό της RAM σε σχέση με την μηχανή Turing είναι η δυνατότητα *άμεσης προσπέλασης* οποιουδήποτε από τους καταχωρητές με αναφορά του δείκτη του. Αυτό το χαρακτηριστικό είναι και αυτό που δίνει το όνομα στο μοντέλο. Έχουμε μάλιστα επιτρέψει και την δυνατότητα *έμμεσης διευθυνσιοδότησης* που είναι και από τα ισχυρότερα σημεία ενός σύγχρονου υπολογιστή. Η δυνατότητα αυτή επιβάλλει την ανάγκη

κάθε καταχωρητής να μπορεί να αποθηκεύσει αυθαίρετα μεγάλους αριθμούς.

Παράδειγμα 2.4 Το πρόγραμμα RAM του Πίνακα 2.3 αποτελεί υλοποίηση του αλγορίθμου της Φυσαλίδας του Παραδείγματος 1. Θεωρούμε ότι οι n αριθμοί δίδονται στους καταχωρητές r_4 μέχρι και r_{n+3} . Ο εξωτερικός βρόγχος έχει μεταβλητή ελέγχου το i η οποία φυλάσσεται στον r_1 και μεταβάλλεται από 4 έως $n+2$. Η μεταβλητή ελέγχου του εσωτερικού βρόγχου j , φυλάσσεται στο r_2 και μεταβάλλεται από 4 έως $n+3-i$. Χρησιμοποιείται επίσης ο καταχωρητής r_3 για φύλαξη του $j+1$. Παρατηρείστε την σχετική δυσκολία που υπάρχει (που είναι γνωστή σε κάθε προγραμματιστή σε γλώσσα assembly) για να γίνουν και απλές εντολές όπως η ανταλλαγή τιμών δύο μεταβλητών.

A/A	Λειτουργία	Λειτουργία	Σχόλια
1.	LOAD =4	AC:=4	
2.	STORE 1	$i := 4$	
3.	LOAD =n	AC:=n { Αρχή ελέγχου εξωτερικού βρόγχου }	
4.	ADD =2	AC:=n+2	
5.	SUB 1	AC:=n+2-i	
6.	JNEG 37	Αν $n+2 < i$ τότε πήγαινε στο τέλος	
7.	LOAD =4	AC:=4	
8.	STORE 2	$j := 4$	
9.	ADD =1	AC:=AC+1	
10.	STORE 3	$r_3 := j+1$	
11.	LOAD =n	AC:=n { Αρχή ελέγχου εσωτερικού βρόγχου }	
12.	ADD =3	AC:=n+3	
13.	SUB 1	AC:=n+3-i	
14.	SUB 2	AC:=n+3-i-j	
15.	JNEG 33	Αν $n+3-i < j$ τότε συνέχισε με τον εξωτερικό βρόγχο	
16.	LOAD *2	AC:=I[j]	
17.	SUB *3	AC:=I[j] - I[j+1]	
18.	JNEG 27	Αν $I[j] < I[j+1]$ τότε μην κάνεις ανταλλαγή	
19.	LOAD *2	AC:=I[j] { Αρχή της ανταλλαγής }	
20.	ADD *3	AC:=I[j]+I[j+1]	
21.	STORE *2	$r_{r_2} := I[j] + I[j+1]$	
22.	SUB *3	AC:=I[j]	
23.	STORE *3	$r_{r_3} := I[j]$	
24.	LOAD *2	AC:=I[j]+I[j+1]	
25.	SUB *3	AC:=I[j+1]	
26.	STORE *2	$r_{r_2} := I[j+1]$ { Τέλος της ανταλλαγής }	
27.	LOAD 2	AC:=j { Αρχή αύξησης του j κατά 1 }	
28.	ADD =1	AC:=j+1	
29.	STORE 2	$j:=j+1$	
30.	ADD =1	AC:=j+2	
31.	STORE 3	$r_3 := j+2$	
32.	JUMP 11		
33.	LOAD 1	AC:=i { Αρχή αύξησης του i κατά 1 }	
34.	ADD =1	AC:=i+1	
35.	STORE 1	$i:=i+1$	
36.	JUMP 3		
37.	HALT		

Πίνακας 2.3: Οι λειτουργίες της RAM του παραδείγματος 2.5.

2.5.2 Ισοδυναμία RAM–Μηχανής Turing

Στην συνέχεια αποδεικνύουμε ένα αποτέλεσμα που δεν πρέπει να αποτελεί έκπληξη: Η RAM μπορεί να προσομοιώσει την λειτουργία κάθε μηχανής Turing. Θα πρέπει όμως πριν να συμφωνηθούν μερικές λεπτομέρειες ως προς τον τρόπο εισόδου μιας συμβολοσειράς και την αποδοχή.

Έτσι, αν η μηχανή Turing M έχει αλφάβητο το $\Sigma = \{\sigma_1, \sigma_2, \dots, \sigma_{|\Sigma|}\}$ και η συμβολοσειρά εισόδου της είναι η w όπου $|w| = n$, τότε η είσοδος στην RAM δίνεται στους πρώτους n καταχωρητές (από τον r_1 έως τον r_n) οι οποίοι έχουν καταχωρημένο ένα ακέραιο στο διάστημα $1..|\Sigma|$, δηλαδή τον αύξοντα αριθμό του αντιστοίχου συμβόλου του w . Ο καταχωρητής r_{n+1} έχει το 0, ώστε να σηματοδοτείται το τέλος της συμβολοσειράς. Ως προς την αποδοχή, θα λέμε ότι η RAM αποδέχεται την είσοδο που δόθηκε με τον παραπάνω τρόπο αν μετά το πέρας του υπολογισμού (με τη λειτουργία HALT), στον συσσωρευτή υπάρχει το 1. Την απορρίπτει αν στον συσσωρευτή υπάρχει το 0. Τέλος ο αριθμός των βημάτων που κάνει ένας υπολογισμός της RAM, είναι ο συνολικός αριθμός των λειτουργιών RAM που θα εκτελεστούν μέχρι και τη λειτουργία HALT.

Θεώρημα 2.5 Έστω L μία γλώσσα που γίνεται αποδεκτή από μια μηχανή Turing με πολυπλοκότητα χρόνου $f(n)$. Τότε η L γίνεται αποδεκτή από μία RAM σε $O(f(n))$ βήματα.

Απόδειξη: Έστω $M = (K, \Sigma, s, \delta)$ η μηχανή Turing. Στον καταχωρητή 1 θα αποθηκεύεται η θέση της κεφαλής. Για τον λόγο αυτό όλη η είσοδος «ολισθαίνει» κατά μία θέση, δηλαδή αρχίζει από τον καταχωρητή 2 έως τον $n + 1$. Χρειάζεται το πολύ $O(n)$ για αυτή την προ-επεξεργασία.

Το πρόγραμμα Π της RAM διακρίνεται σε $|K|$ τμήματα. Κάθε τέτοιο τμήμα Q_i αντιστοιχεί σε μία συγκεκριμένη κατάσταση της μηχανής Turing. Οι λειτουργίες που είναι προγραμματισμένες σε αυτό το τμήμα αφορούν τις μεταβάσεις της μηχανής Turing όταν αυτή βρίσκεται στην κατάσταση q_i . Κατά συνέπεια κάθε τέτοιο τμήμα Q_i διακρίνεται σε το πολύ $|\Sigma|$ υποτμήματα όσα και τα σύμβολα σ_j για τα οποία ορίζεται η $\delta(q_i, \sigma_j)$. Σε κάθε τέτοιο υποτμήμα λοιπόν αλλάζει το περιεχόμενο του καταχωρητή που κρατά το τρέχον σύμβολο κάτω από την κεφαλή με την νέα του τιμή, και την τιμή του καταχωρητή r_2 , αν χρειάζεται, αυξάνοντας ή μειώνοντας την κατά ένα, ώστε να αντανακλά στην κίνηση της κεφαλής. Τέλος κάνει ένα άλμα στο υποτμήμα Q_j που αντιστοιχεί στην καινούργια κατάσταση q_j . Το πρόγραμμα του υποτμήματος που υλοποιεί την μετάβαση $\delta(q_i, \sigma_j) = (q_l, \sigma_m, D)$ φαίνεται στον Πίνακα 2.4. Το υποτμήμα αυτό αρχίζει στην λειτουργία υπ' αριθμό M_{ij} .

Ο αριθμός των βημάτων για την προσομοίωση μιας μετάβασης από μία

Αρ. Λειτ.	Λειτουργία	Σχόλια
M_{ij}	LOAD *1	Φόρτωσε το σύμβολο σ_k κάτω από την κεφαλή.
$M_{ij}+1$	SUB =j	AC:=k-j.
$M_{ij}+2$	JNEG $M_{ij}+12$	Αν $k - j < 0$ $\sigma_k \neq \sigma_j$, δοκίμασε το επόμενο.
$M_{ij}+3$	LOAD =j	AC:=j.
$M_{ij}+4$	SUB *1	AC:=j-k.
$M_{ij}+5$	JNEG $M_{ij}+12$	Αν $j - k < 0$ $\sigma_k \neq \sigma_j$, δοκίμασε το επόμενο.
$M_{ij}+6$	LOAD =m	Για να είμαστε εδώ $\sigma_k = \sigma_j$. AC:=m.
$M_{ij}+7$	STORE *1	Το σύμβολο σ_m αντικαθιστά το σ_k .
$M_{ij}+8$	LOAD 1	Φόρτωσε την θέση της κεφαλής.
$M_{ij}+9$	ADD =d	$d = -1, 1, 0$ αν $D = \leftarrow, \rightarrow, -$ αντίστοιχα.
$M_{ij}+10$	STORE 1	Κίνηση της κεφαλής.
$M_{ij}+11$	JUMP M_{l1}	Άλλα στο τμήμα της κατάστασης q_l (επόμενης).

Πίνακας 2.4: Προσομοίωση της μετάβασης $\delta(q_i, \sigma_j) = (q_l, \sigma_m, D)$

κατάσταση είναι συνεπώς όσος χρειάζεται για την εκτέλεση όλων των υπο-τμημάτων που είναι $|\Sigma|$ επί 12, συνολικά δηλαδή $O(f(n))$. \square

Για την αντίστροφη κατεύθυνση, δηλαδή την προσομοίωση μιας RAM από μια μηχανή Turing πρέπει να ληφθούν υπόψη ορισμένα στοιχεία. Κατά πρώτο η δυνατότητα της RAM να χειρίζεται αριθμούς αυθαίρετου μήκους αποτελεί πρόβλημα αν μετράμε σαν πολυπλοκότητα της RAM τον αριθμό των βημάτων της. Ο λόγος είναι ότι πολύ περίπλοκες εντολές μπορεί να προσομοιωθούν σαν πράξεις μεταξύ αριθμών αυθαίρετου μήκους. Για παράδειγμα πράξεις μεταξύ διανυσμάτων μπορεί να γίνουν σε ένα βήμα κάτι που δεν είναι ρεαλιστικό. Ένα καλύτερο συνεπώς μέτρο είναι να μετράμε «χρόνο» ενός RAM προγράμματος αντί για βήματα. Αυτό το μετράμε σαν ανάλογο (και συνεπώς για τους σκοπούς μας σαν ίσο) με τον αριθμό των δυαδικών ψηφίων των αριθμών που εμπλέκονται σε μία RAM λειτουργία. Αν για παράδειγμα στον καταχωρητή r_{17} υπάρχει ο αριθμός 100000, χρεώνουμε την λειτουργία LOAD 17 με 8 μονάδες χρόνου (2 για το όρισμα «17» και 6 για το 100000). Ονομάζουμε αυτό τον τρόπο μέτρησης χρόνου σαν το *λογαριθμικό κόστος* σε αντίθεση με το *μοναδιαίο* όπου κάθε λειτουργία στοιχίζει μία μονάδα χρόνου.

Θα πρέπει επίσης να ληφθεί υπόψη και η δυνατότητα της RAM να διευθυνσιοδοτεί αυθαίρετους καταχωρητές. Έτσι για παράδειγμα αν η «χρήσιμη» είσοδος στην RAM βρίσκεται από τον καταχωρητή 2^n και μετά, ενώ οι προηγούμενοι καταχωρητές δεν περιέχουν τίποτα, θα στοίχιζε μόνο $O(n)$ μονάδες χρόνου στην RAM να την προσπελάσει με το λογαριθμικό κόστος και μόνο μία μονάδα με το μοναδιαίο κόστος. Μια μηχανή Turing όμως θα έπαιρνε αναγκαστικά χρόνο εκθετικό. Συνεπώς θα πρέπει σαν μέγεθος εισόδου στην RAM να θεωρήσουμε τον δείκτη του τελευταίου καταχωρητή που περιέχει τμήμα της

εισόδου.

Θεώρημα 2.6 Έστω μία συνάρτηση $g : N^n \rightarrow N$, n ορισμάτων η οποία μπορεί να υπολογιστεί από μία RAM σε χρόνο $O(f(n))$. Τότε υπάρχει μία μηχανή Turing που υπολογίζει την g σε χρόνο $O(f(n)^3)$.

Απόδειξη: Μια μηχανή Turing M που προσομοιώνει την RAM δέχεται στην ταινία εισόδου της τα περιεχόμενα των n καταχωρητών εισόδου της RAM σε δυαδική μορφή χωρισμένα με κάποιο ειδικό σύμβολο έστω το $\#$. Η M έχει μία ταινία για τον απαριθμητή προγράμματος PC και μία για τον συσσωρευτή AC. Για να αντιμετωπίσουμε την πιθανή χρήση από την RAM καταχωρητών με αυθαίρετα μεγάλο δείκτη, όλοι οι καταχωρητές που έχουν προσπελαστεί έστω και μία φορά αποθηκεύονται μαζί με τον δείκτη τους στην τέταρτη ταινία της M σαν $\#i\#r_i$. Το πρώτο στάδιο λειτουργίας της M είναι λοιπόν να αντιγράψει στην τέταρτη ταινία τους n καταχωρητές εισόδου μαζί με τους δείκτες τους στην παραπάνω μορφή. Στην συνέχεια κάθε αλλαγή περιεχομένου ενός καταχωρητή προστίθεται στο τέλος της ταινίας αυτής (πάντα μαζί με τον δείκτη του καταχωρητή). Για να βρεί λοιπόν η M το περιεχόμενο ενός καταχωρητή σαρώνει την τέταρτη ταινία από το τέλος προς την αρχή μέχρι να βρεί την πρώτη εμφάνιση του δείκτη που ζητά (που συνοδεύεται από την πιο πρόσφατη τιμή του καταχωρητή) και στην συνέχεια την τιμή του καταχωρητή. Αν δεν βρεθεί ένας δείκτης τότε υποθέτει σαν τιμή του αντίστοιχου καταχωρητή το 0. Συνεπώς δύο σαρώσεις το πολύ απαιτούνται για κάθε προσπέλαση στην μνήμη της RAM (η δεύτερη όταν έχουμε έμμεση διευθυνσιοδότηση). Οι προσομοιώσεις των λειτουργιών της RAM μπορούν εύκολα να γίνουν από «υποπρογράμματα» της μηχανής M που κάθε ένα απαιτεί σταθερό χρόνο. Συνεπώς ο χρόνος της προσομοίωσης κυριαρχείται από την σάρωση της τέταρτης ταινίας. Πόσο μεγάλη μπορεί να γίνει αυτή; Προφανώς θα γραφούν το πολύ $f(n)$ ζευγάρια δεικτών-καταχωρητών. Με το λογαριθμικό κόστος λοιπόν, κάθε αποθηκευμένος αριθμός στην τέταρτη ταινία στοιχίζει τα ίδια βήματα και στην μηχανή Turing και στην RAM και συνεπώς μετρά σαν μία μονάδα. Άρα η σάρωση της ταινίας στην περίπτωση αυτή στοιχίζει στη μηχανή Turing χρόνο $O(f(n))$ (όσο και οι αποθηκευμένοι αριθμοί). Συνεπώς στην περίπτωση του λογαριθμικού κόστους ο συνολικός χρόνος της M είναι $O(f(n)^2)$. Για το μοναδιαίο κόστος, ας παρατηρήσουμε ότι κάθε λειτουργία της RAM μπορεί μόνο να αυξήσει το μήκος του αποθηκευμένου αριθμού κατά 1 ψηφίο (στην περίπτωση των ADD και SUB)² και συνεπώς στην τέταρτη ταινία της M θα γραφούν το πολύ $O(f(n)^2)$ ψηφία. Άρα αν χρησιμοποιούμε το μοναδιαίο κόστος τότε η

² Αυτό οφείλεται στο ότι το ρεπερτόριο της RAM που υιοθετήσαμε δεν περιέχει πράξεις όπως ο πολλαπλασιασμός που μπορεί να διπλασιάσουν τον αριθμό των ψηφίων του αποτελέσματος

σάρωση στοιχίζει στην μηχανή Turing όσα και τα εγγεγραμμένα ψηφία δηλαδή $O(f(n)^2)$. Στην περίπτωση λοιπόν αυτή ο συνολικός χρόνος είναι $O(f(n)^3)$. \square

Παρατήρηση 2.3 Τα αποτελέσματα της παραπάνω ενότητας σε σχέση με την μηχανή RAM, ένα μοντέλο πολύ κοντά στους πραγματικούς υπολογιστές, αλλά και οι άλλες παραλλαγές της μηχανής Turing που μελετήσαμε, έδωσαν εκτός από την ισοδυναμία όλων των μοντέλων ως προς το τι μπορούν να υπολογίσουν και επιπλέον πολυωνυμική συσχέτιση του χρόνου που γίνεται ο υπολογισμός. Αυτή η πολυωνυμική συσχέτιση παρακολουθεί και όλα τα μοντέλα υπολογισμού που προτάθηκαν κατά καιρούς αρκεί να είναι ρεαλιστικά. Με αυτό εννοούμε ότι δεν επιβάλουμε στον ορισμό τους ούτε αδικαιολόγητους περιορισμούς και αδυναμίες που θα τα έκαναν υπερβολικά αργά, ούτε και ιδιότητες που θα έκαναν την φυσική τους υλοποίηση αδύνατη. Κάθε τέτοιο «λογικό» μοντέλο αποδεικνύεται ότι είναι πολυωνυμικά ισοδύναμο με την μηχανή Turing. Μπορούμε λοιπόν να ομιλούμε και για την *Πολυωνυμική Θέση του Church* όπου εκτός από την υπολογιστική ικανότητα του μοντέλου περιλαμβάνεται και η ταχύτητα του.

2.6 Κλάσεις πολυπλοκότητας

Οι υπολογισμοί που γίνονται με περιορισμένο κάποιο υπολογιστικό πόρο (χρόνος, μνήμη κλπ) με την βοήθεια κάποιου υπολογιστικού μοντέλου (ντετερμινιστική ή μη ντετερμινιστική μηχανή Turing) μπορούν να χρησιμεύσουν στην κατηγοριοποίηση των υπολογιστικών προβλημάτων με φυσικό τρόπο:

Ορισμός 2.16 Ορίζουμε σαν $\text{TIME}(f(n))$ το σύνολο των γλωσσών L που αποφασίζονται από κάποια ντετερμινιστική μηχανή Turing με πολυπλοκότητα χρόνου $f(n)$.

Το $\text{TIME}(f(n))$ είναι μία κλάση πολυπλοκότητας. Είναι δηλαδή ένα σύνολο γλωσσών που αποφασίζονται από κάποια μηχανή μέσα σε κάποια περιθώρια καθοριζόμενα από την συνάρτηση $f(n)$, κάποιου υπολογιστικού πόρου (στην περίπτωση μας του χρόνου). Η συνάφεια όμως των προβλημάτων απόφασης με τις αντίστοιχες γλώσσες (όπως εξηγήθηκε παραπάνω) μας επιτρέπει να μιλάμε και για το σύνολο προβλημάτων απόφασης που τα στιγμιότυπα τους μεγέθους n , επιλύονται από κάποια ντετερμινιστική μηχανή Turing με πολυπλοκότητα $f(n)$, (ή για συντομία, σε χρόνο $f(n)$). Ανάλογοι είναι και οι παρακάτω ορισμοί:

Ορισμός 2.17 Ορίζουμε σαν $\text{NTIME}(f(n))$ το σύνολο των γλωσσών L που αποφασίζονται από κάποια μη ντετερμινιστική μηχανή Turing με πολυπλοκότητα χρόνου $f(n)$.

Ορισμός 2.18 Ορίζουμε σαν $\text{SPACE}(f(n))$ το σύνολο των γλωσσών L που αποφασίζονται από κάποια ντετερμινιστική μηχανή Turing με πολυπλοκότητα χώρου $f(n)$.

Ορισμός 2.19 Ορίζουμε σαν $\text{NSPACE}(f(n))$ το σύνολο των γλωσσών L που αποφασίζονται από κάποια μη ντετερμινιστική μηχανή Turing με πολυπλοκότητα χώρου $f(n)$.

Μερικές κλάσεις πολυπλοκότητας είναι αρκετά σημαντικές ώστε να έχουν το δικό τους όνομα και σύμβολο. Η πρώτη τέτοια κλάση που εισάγουμε είναι η κλάση P που ορίζεται σαν:

$$P = \bigcup_{k>0} \text{TIME}(n^k).$$

Η κλάση P επομένως είναι η κλάση των γλωσσών που αποφασίζονται από μια ντετερμινιστική μηχανή Turing σε χρόνο που είναι φραγμένος από κάποιο πολυώνυμο στο μήκος της εισόδου.

Ο ορισμός του P δεν είναι απόλυτα σύμφωνος με τον ορισμό 2.16. Ο λόγος είναι ότι η P είναι ένωση κλάσεων πολυπλοκότητας και ορίζεται με την βοήθεια απειρίας συναρτήσεων $f(n)$. Είναι συνεπώς πραγματική κλάση πολυπλοκότητας το P ; Με αυτό ερωτάμε στην ουσία αν υπάρχει μία συνάρτηση $p(n)$ τέτοια που το $\text{TIME}(p(n))$ να περιλαμβάνει μόνο τις πολυωνυμικά αναγνωρίσιμες γλώσσες και τίποτε άλλο. Το παρακάτω θεώρημα μας λέει ότι υπάρχει. Αντίστοιχα θεωρήματα ισχύουν και για τις κλάσεις που θα ορίσουμε με ανάλογο τρόπο παρακάτω.

Θεώρημα 2.7 Έστω $f_1(n), f_2(n), \dots$, ένα αναδρομικά αριθμήσιμο σύνολο αναδρομικών συναρτήσεων έτσι ώστε για κάθε i και n , $f_i(n) < f_{i+1}(n)$. Τότε υπάρχει αναδρομική συνάρτηση $f(n)$ τέτοια που:

$$\text{TIME}(f(n)) = \bigcup_{i \geq 1} \text{TIME}(f_i(n))$$

Απόδειξη: Η απόδειξη δόθηκε στο [21]. □

Η σημασία της κλάσης P είναι ότι θεωρείται ότι περιλαμβάνει τα προβλήματα απόφασης (σύμφωνα με την αντιστοιχία με τις γλώσσες) με ικανοποιητικό αλγόριθμο επίλυσης. Αυτό μπορεί να ακούγεται σε πρώτη προσέγγιση παράξενο μιας και εφόσον δεν έχουμε θέσει κανένα περιορισμό στον εκθέτη του πολυωνύμου μας (έχουμε εντούτοις επιβάλλει να είναι σταθερός αριθμός), θα πρέπει να αποδεχθούμε και χρόνους της τάξης $O(n^{100})$ σαν ικανοποιητικούς! Παρόλο που προβλήματα με τέτοια φράγματα μπορεί να κατασκευαστούν, τα

περισσότερα πρακτικά προβλήματα με πολυωνυμικούς αλγορίθμους έχουν πολυπλοκότητες με μικρούς εκθέτες 4–5 το πολύ. Εξάλλου σε σύγκριση με τους εκθετικούς αλγορίθμους, ακόμη και με πολυπλοκότητα $O(2^{\frac{n}{1000}})$, ένας πολυωνυμικός κάποια στιγμή (για πολύ μεγάλα n ίσως) θα είναι ταχύτερος. Βέβαια θα μπορούσε αυτό να συμβεί για τόσο μεγάλα n που να μην έχει καμία πρακτική αξία, είναι όμως γενική η αίσθηση ότι στην πράξη ένας πολυωνυμικός αλγόριθμος είναι καλύτερος από ένα εκθετικό. Επί πλέον, φαίνεται ότι η εύρεση ενός πολυωνυμικού αλγορίθμου απαιτεί την εκμετάλλευση κάποιων ιδιοτήτων του προβλήματος και όχι απλά έξυπνο σχεδιασμό. Για παράδειγμα οι διάφορες «εξωτικές» δομές δεδομένων δεν μπορούν να μετατρέψουν έναν εκθετικό αλγόριθμο σε πολυωνυμικό. Μπορούν να βελτιώσουν τον χρόνο αλλά για τόσο δραματική αλλαγή απαιτείται συνήθως άλλη προσέγγιση στο πρόβλημα. Αντίστοιχα επιχειρήματα προσφέρουν και τα υπολογιστικά μοντέλα και κυρίως η πολυωνυμική συσχέτιση που φαίνεται να έχουν μεταξύ τους όλα τα «λογικά» μοντέλα (βλ. και Παρατήρηση 2.3).

Πέρα όμως από αυτά η κλάση P είναι μία προσπάθεια να οριστεί μαθηματικά μια διαισθητική έννοια, όπως αυτή του «ικανοποιητικά επιλύσιμου» προβλήματος. Κάθε τέτοια προσπάθεια είναι καταδικασμένη να παρουσιάσει λάθη και μάλιστα και προς τις δύο κατευθύνσεις: και ικανοποιητικά επιλύσιμα προβλήματα μένουν εκτός P και μη ικανοποιητικά επιλύσιμα τοποθετούνται στο P. Παρόλα αυτά, αυτή η μαθηματική αποτύπωση μιας διαισθητικής έννοιας αφενός είναι κοντά στην αλήθεια και αφετέρου έχει προσφέρει πολλά στην ανάπτυξη μιας κομψής θεωρίας.

Η δεύτερη πολύ σημαντική κλάση που θα εισάγουμε είναι η κλάση NP:

$$NP = \bigcup_{k \geq 0} \text{NTIME}(n^k).$$

Το NP είναι η κλάση των γλωσσών που αποφασίζονται από μία μη ντετερμινιστική μηχανή Turing σε χρόνο πολυωνυμικό στο μήκος της εισόδου. Για να κατανοήσουμε ποιές γλώσσες (άρα και προβλήματα) ανήκουν στην κατηγορία αυτή, ας θυμηθούμε ότι η μη ντετερμινιστική μηχανή Turing υπολογίζει (εξ' ορισμού) με ένα τρόπο που είναι ισοδύναμος με πρόβλεψη της απάντησης και στην συνέχεια με επιβεβαίωση της πρόβλεψης (το «μάντεψε και επαλήθευσε»). Αυτό, σε συνδυασμό με όσα αναφέρθηκαν παραπάνω για ότι οι πολυωνυμικοί χρόνοι είναι ικανοποιητικοί, χαρακτηρίζει τα προβλήματα στο NP σαν αυτά που επιλύονται μέσω αυτής της διαδικασίας σε ικανοποιητικό χρόνο. Συνεπώς, πρόκειται για προβλήματα για τα οποία υπάρχει κάποια είδους πληροφορία η οποία να μπορεί να ελεγχθεί σε πολυωνυμικό χρόνο αν πράγματι αποτελεί λύση. Αυτή η πληροφορία λέγεται πιστοποιητικό (certificate) και κάθε πρόβλημα στο NP έχει ένα τέτοιο «καλό» πιστοποιητικό, δηλαδή εύκολα ελέγξιμο. Αντίστροφα, η ύπαρξη καλού πιστοποιητικού είναι απόδειξη ότι ένα πρόβλημα

ανήκει στο NP.

Παράδειγμα 2.5 Το πρόβλημα της ικανοποιησιμότητας (SAT), που ορίσαμε παραπάνω είναι ένα πρόβλημα στο NP. Ο μη ντετερμινιστικός αλγόριθμος που δώθηκε στο Παράδειγμα 2 τρέχει σε γραμμικό χρόνο $O(n)$ και συνεπώς αυτό κατατάσσει αμέσως το SAT στο NP. Γράφουμε $SAT \in NP$ (ίσως χρησιμοποιώντας χαλαρά τους συμβολισμούς, το ακριβές θα ήταν $L_{SAT} \in NP$). Στο πρόβλημα αυτό, το πιστοποιητικό είναι μια ανάθεση τιμών στις μεταβλητές, τέτοια που η συνάρτηση να γίνει αληθής. Κάθε ικανοποίησιμο («NAI») στιγμιότυπο του SAT έχει εξ' ορισμού μια τουλάχιστον τέτοια αποτίμηση. Η επιβεβαίωση ότι μια αποτίμηση πράγματι ικανοποιεί το στιγμιότυπο γίνεται σε γραμμικό χρόνο $O(n)$ και άρα το SAT έχει καλό πιστοποιητικό.

Ισχύει ασφαλώς ότι $P \subseteq NP$ επειδή μία ντετερμινιστική μηχανή μπορεί να ειδοωθεί σαν μια μη ντετερμινιστική όπου σε κάθε στιγμή οι δυνατότητες μετάβασης είναι μόνο μία. Η σημασία του NP βρίσκεται στο ότι σ' αυτό κατατάσσεται πληθώρα πρακτικών και χρήσιμων προβλημάτων για τα οποία δεν υπάρχει γνωστός ικανοποιητικός (πολυωνυμικός) αλγόριθμος αλλά και για τα οποία δεν έχειδειχθεί μέχρι σήμερα ότι δεν υπάρχει τέτοιος αλγόριθμος. Το SAT είναι ένα τέτοιο πολύ σημαντικό πρόβλημα και θα συναντήσουμε και πολλά άλλα στην συνέχεια. Ακριβώς επειδή έχουν γίνει τόσες ανεπιτυχείς προσπάθειες για εύρεση πολυωνυμικών αλγορίθμων για τα προβλήματα στο NP, θεωρείται απίθανο να ισχύει $P=NP$. Πάντως απόδειξη ακόμη δεν έχει δοθεί και το πρόβλημα αυτό θεωρείται το σημαντικότερο στην Θεωρητική Πληροφορική.

Η πρώτη κλάση πολυπλοκότητας χρόνου που, όπως θα δούμε, είναι γνήσιο υπερσύνολο του P (και κατά συνέπεια περιέχει προβλήματα αποδεδειγμένα δύσκολα) είναι η κλάση εκθετικού χρόνου EXPTIME:

$$EXPTIME = \bigcup_{k>0} TIME(2^{n^k}).$$

Αλλά και μερικές κλάσεις πολυπλοκότητας χώρου είναι ιδιαίτερα σημαντικές.

Έτσι η κλάση

$$L = SPACE(\log n)$$

και η ανάλογη μη ντετερμινιστική

$$NL = NSPACE(\log n)$$

είναι οι κλάσεις λογαριθμικού ντετερμινιστικού και μη ντετερμινιστικού χώρου αντίστοιχα.

Ιδιαίτερα σημαντική επίσης γιατί περιέχει πολλά προβλήματα από την Θεωρία Παιγνίων και άλλα είναι η κλάση πολυωνυμικού χώρου:

$$\text{PSPACE} = \bigcup_{k \geq 0} \text{SPACE}(n^k).$$

Πολλές φορές επίσης ενδιαφερόμαστε και για τα συμπληρώματα κάποιων κλάσεων πολυπλοκότητας, δηλαδή για τα σύνολα των γλωσσών που είναι συμπληρωματικές κάποιας γλώσσας στην κλάση πολυπλοκότητας. Τυπικά:

Ορισμός 2.20 Έστω L μία γλώσσα. Η συμπληρωματική της \bar{L} είναι η γλώσσα η οποία περιλαμβάνει όλες τις συμβολοσειρές στο Σ^* που δεν ανήκουν στην L . Έστω επίσης C μια κλάση πολυπλοκότητας. Η συμπληρωματική της κλάση $\text{co-}C$ (co από το complement, συμπλήρωμα) ορίζεται σαν:

$$\text{co-}C = \{L \mid L \in \Sigma^*, \text{ και } \bar{L} \in C\}$$

Η αξία των συμπληρωματικών κλάσεων πολυπλοκότητας βρίσκεται στο ότι σε αυτές κατατάσσονται τα προβλήματα απόφασης στα οποία παρουσιάζουν ενδιαφέρον τα στιγμιότυπα με απάντηση «ΟΧΙ». Θα δούμε τέτοια προβλήματα στα επόμενα κεφάλαια.

2.7 Ασκήσεις

1. Κατασκευάστε μηχανή Turing που να αναγνωρίζει την γλώσσα $L = \{w \mid w \in \{a, b\}^*, w = w^R\}$. Με τον συμβολισμό w^R παριστάνουμε την συμβολοσειρά w γραμμένη από το τέλος προς την αρχή.
2. Κατασκευάστε μηχανή Turing που να αποφασίζει την γλώσσα $L = \{ww \mid w \in \{a, b\}^*\}$.
3. Κατασκευάστε μηχανή Turing που να υπολογίζει την συνάρτηση $f : \Sigma^* \longrightarrow \Sigma^*, f(x) = x^R$.
4. Έστω ότι για δύο συναρτήσεις f και g από το Σ^* στο Σ^* υπάρχουν δύο μηχανές Turing M_f και M_g αντίστοιχα που τις υπολογίζουν. Δείξτε πως μπορείτε να κατασκευάσετε μηχανή Turing για την σύνθεση των συναρτήσεων $f \circ g$.

5. Δώστε μια 3-ταινιών μηχανή Turing η οποία με είσοδο δύο αριθμούς σε δυαδική μορφή στην πρώτη της ταινία (χωρισμένους με το σύμβολο #) υπολογίζει:
 - (α) το άθροισμα τους,
 - (β) το γινόμενο τους.
6. Ορίστε με ακρίβεια την μηχανή Turing πολλαπλών κεφαλών που περιγράφεται στην παράγραφο 2.3. Εξηγήστε πως μία συμβατική μηχανή μπορεί να προσομοιώσει την μηχανή πολλαπλών κεφαλών.
7. Το ίδιο με την παραπάνω άσκηση για την μηχανή Turing με διδιάστατη ταινία.
8. Το ίδιο με την παραπάνω άσκηση για την μηχανή Turing με διδιάστατη ταινία.
9. Δώστε μια μηχανή Turing που να απαιτεί χρόνο $2^{|x|}$ για κάθε είσοδο x .
10. Δώστε μια μηχανή Turing που να απαιτεί χώρο $2^{|x|}$ για κάθε είσοδο x .
11. Κατασκευάστε μηχανή Turing 2-ταινιών που να αποφασίζει την γλώσσα των «καρκινικών» συμβολοσειρών της Άσκησης 1 σε χρόνο $O(n)$.
12. Κατασκευάστε μηχανή Turing μιας ταινίας που να αποφασίζει τη γλώσσα της Άσκησης 1 σε χρόνο $O(n^2)$. Δείξτε ότι οποιαδήποτε μηχανή μιας ταινίας απαιτεί χρόνο $\Omega(n^2)$ γι' αυτή τη γλώσσα. (Το τελευταίο ερώτημα είναι δύσκολο. Η απόδειξη δόθηκε στο [8] και μπορεί να βρεθεί και στο [10]. Σε συνδυασμό με την προηγούμενη άσκηση δείχνει ότι μια μηχανή δύο ταινιών είναι ταχύτερη από μία μηχανή μιας ταινίας. Αυτό πρέπει να συγκριθεί με το αποτέλεσμα του Θεωρήματος 2.3.)
13. Δώστε ένα πρόγραμμα RAM που δοθέντος ενός φυσικού n , υπολογίζει το $\lceil \log n \rceil$.
14. Δώστε ένα πρόγραμμα RAM που να υπολογίζει το γινόμενο δύο ακεραίων αριθμών.

Κεφάλαιο 3

Ιδιότητες των κλάσεων πολυπλοκότητας

Στο κεφάλαιο αυτό εξετάζουμε διάφορες ιδιότητες που ισχύουν για τις κλάσεις πολυπλοκότητας. Οι παράμετροι που θα μας απασχολήσουν είναι το μέτρο πολυπλοκότητας (χρόνος ή χώρος), ο τρόπος αποδοχής (ντετερμινιστικός ή μη) και η συνάρτηση πολυπλοκότητας.

3.1 Η συνάρτηση πολυπλοκότητας

Οι απαιτήσεις που έχουμε από μια συνάρτηση ώστε να είναι κατάλληλη για συνάρτηση πολυπλοκότητας, είναι πολύ μικρές. Γενικά κάθε συνάρτηση $f : \mathbb{Z}^+ \rightarrow \mathbb{Z}^+$ είναι κατάλληλη και μέσα σε αυτές ασφαλώς οι συνηθισμένες συναρτήσεις. Υπάρχουν όμως και συναρτήσεις με ιδιαίτερα ασυνήθιστους τρόπους ορισμού, που δεν μπορούν να υπολογιστούν μέσα στα πλαίσια χρόνου και χώρου που οι ίδιες προσδιορίζουν! Τέτοιες συναρτήσεις θα περιγράψουμε στο Θεώρημα 3.3, ένα θεώρημα που μοιάζει να είναι εντελώς ενάντια στην διαίσθησή μας. Τα περισσότερα χρήσιμα αποτελέσματα εντούτοις προκύπτουν όταν περιορίσουμε λίγο τις επιλογές μας για την συνάρτηση πολυπλοκότητας σε περισσότερο «φυσικές» συναρτήσεις.

Ορισμός 3.1 Ονομάζουμε μία συνάρτηση $f : \mathbb{Z}^+ \rightarrow \mathbb{Z}^+$ καλά υπολογίσιμη αν υπάρχει μία μηχανή Turing η οποία με είσοδο τον ακέραιο n , (σε δυαδική μορφή) υπολογίζει τον ακέραιο $f(n)$ (επίσης σε δυαδική μορφή) σε χρόνο $O(f(n))$. Μια συνάρτηση $f : \mathbb{Z}^+ \rightarrow \mathbb{Z}^+$ ονομάζεται κατασκευάσιμη αν υπάρχει μηχανή Turing η οποία για κάθε είσοδο x με $|x| = n$, χρησιμοποιεί ακριβώς $f(n)$ χώρο. \square

Παρατήρηση 3.1 Αν μια συνάρτηση είναι καλά υπολογίσιμη τότε από τον ορισμό μπορούμε να βελτιώσουμε την μηχανή Turing ώστε να κάνει ακριβώς $f(n)$ βήματα για τον υπολογισμό της $f(n)$. Αυτό προκύπτει από το Θεώρημα 2.1 ότι δηλαδή ο χρόνος της μηχανής μπορεί να γίνει $\epsilon \cdot f(n)$ για $\epsilon < 1$. Στην συνέχεια η μηχανή (γνωρίζοντας το $f(n)$) μπορεί να περιμένει ώστε να αποδεχθεί ακριβώς την χρονική στιγμή $f(n)$.

Είναι εύκολο να δει κανείς ότι οι κλάσεις των καλά υπολογίσιμων και η κλάση των κατασκευάσιμων συναρτήσεων είναι πολύ πλούσιες. Περιλαμβάνουν όλες τις συνηθισμένες συναρτήσεις n^k , a^n , $\log n$ κλπ. Ταυτόχρονα είναι εύκολο να διαπιστωθεί ότι αν f και g είναι καλά υπολογίσιμες ή κατασκευάσιμες τότε το ίδιο συμβαίνει και για τις $f + g$, $f \cdot g$, f^g (άσκηση).

Ο λόγος για αυτούς τους ορισμούς είναι ότι αν χρησιμοποιούμε τέτοιες συναρτήσεις, μπορούμε να υποθέσουμε ότι οι αντίστοιχες μηχανές Turing λειτουργούν σαν περιοριστές χρόνου και χώρου σε άλλες μηχανές και έτσι να έχουν περισσότερο προβλέψιμη συμπεριφορά. Έτσι έχουμε το παρακάτω λήμμα:

Λήμμα 3.1 Έστω f μια καλά υπολογίσιμη συνάρτηση και έστω L μια γλώσσα που αποφασίζεται από μια μηχανή Turing M (ντετερμινιστική ή μη), με πολυπλοκότητα χρόνου $f(n)$. Τότε υπάρχει μηχανή M' που αποφασίζει την L κάνοντας ακριβώς $f(n)$ βήματα σε κάθε υπολογισμό.

Απόδειξη: Επειδή η $f(n)$ είναι καλά υπολογίσιμη υπάρχει μηχανή M_f η οποία την υπολογίζει σε ακριβώς $f(n)$ βήματα. Η M' συνεπώς εργάζεται κάνοντας εναλλάξ ένα βήμα της M και ένα της M_f . Αν η συμβολοσειρά εισόδου $x \in L$ τότε η M θα αποδεχθεί σε χρόνο ίσως μικρότερο από $f(n)$. Σε τέτοια περίπτωση η M' απλώς παρακολουθεί την M_f και αποδέχεται σε χρόνο ακριβώς $f(n)$. Αν αντίθετα μετά την παρέλευση χρόνου $f(n)$, δεν έχουμε αποδοχή από την M , τότε η M' απορρίπτει την είσοδο. \square

Λήμμα 3.2 Έστω f μια κατασκευάσιμη συνάρτηση και έστω L μια γλώσσα που αποφασίζεται από μια μηχανή Turing M (ντετερμινιστική ή μη) με πολυπλοκότητα χώρου $f(n)$. Τότε υπάρχει μηχανή M' που αποφασίζει την L χρησιμοποιώντας χώρο ακριβώς $f(n)$ σε κάθε υπολογισμό.

Απόδειξη: Επειδή η L είναι κατασκευάσιμη, υπάρχει μηχανή Turing M_f η οποία χρησιμοποιεί ακριβώς $f(n)$ κύτταρα σε κάθε υπολογισμό. Η M' αρχίζει την λειτουργία της προσομοιώνοντας την M_f . Η τελευταία θα χρησιμοποιήσει ακριβώς $f(n)$ κύτταρα τα οποία η M' σημειώνει. Στην συνέχεια προσομοιώνει την M . Αν η M ζητήσει περισσότερο χώρο από τον σημειωμένο τότε η M' απορρίπτει την είσοδο και σταματά. Είναι προφανές ότι η M' χρησιμοποιεί χώρο ακριβώς $f(n)$ και αποδέχεται ακριβώς όταν αποδέχεται η M . \square

3.2 Θεωρήματα Ιεραρχίας

Όταν οι χρησιμοποιούμενες συναρτήσεις πολυπλοκότητας ανήκουν στις κατηγορίες του Ορισμού 3.1 τότε ισχύουν «Ιεραρχίες» χρόνου και χώρου, με την έννοια ότι αν έχουμε στην διάθεση μας περισσότερους υπολογιστικούς πόρους (χρόνο ή χώρο), μπορούμε να αναγνωρίσουμε και περισσότερες γλώσσες.

Θεώρημα 3.1 (Ιεραρχίας χρόνου) *Αν η συνάρτηση $f(n)$ είναι καλά υπολογίσιμη τότε υπάρχει κάποια γλώσσα που ανήκει στο $\text{TIME}(f(n)g(n))$ αλλά όχι στο $\text{TIME}(f(n))$. Η $g(n)$ μπορεί να είναι οποιαδήποτε συνάρτηση αυξάνει ταχύτερα από την $n \log f(n)$, δηλαδή $g(n) = \Omega(n \log f(n))$.*

Απόδειξη: Η απόδειξη βασίζεται στην εύρεση μιας τέτοιας γλώσσας L . Η L περιλαμβάνει τις συμβολοσειρές w_M του αλφαβήτου $\{0, 1\}$ όπου ο συμβολισμός αυτός δηλώνει την συμβολοσειρά που κωδικοποιεί την μηχανή M , με τον τρόπο της Ενότητας 2.3.2 (συμπληρωμένο ώστε να κωδικοποιεί και μηχανές πολλών ταινιών). Οι κωδικοποιημένες με τις συμβολοσειρές της L μηχανές, θα πρέπει να απορρίπτουν την δική τους κωδικοποίηση σε χρόνο το πολύ $f(n)$. Τυπικά έχουμε:

$$L = \{w_M \mid \text{η μηχανή } M \text{ απορρίπτει την } w_M \text{ σε χρόνο το πολύ } f(|w_M|)\}$$

Θα δείξουμε κατ' αρχήν ότι $L \in \text{TIME}(nf(n) \log f(n))$. Για τον σκοπό αυτό θα περιγράψουμε μία μηχανή M_1 με πολυπλοκότητα χρόνου $nf(n) \log f(n)$ η οποία αποφασίζει την L . Η λειτουργία της M_1 είναι περίπου όπως η καθολική μηχανή U της Ενότητας 2.3.2 και πιο συγκεκριμένα όπως στη Παρατήρηση 2.2 μια και η M ίσως να έχει περισσότερες από μια ταινίες. Η M_1 κατ' αρχή αναλύει «συντακτικά» την είσοδο της και την απορρίπτει σε περίπτωση που δεν είναι νόμιμη κωδικοποίηση κάποιας μηχανής Turing. Η λειτουργία αυτή γίνεται σε γραμμικό χρόνο. Σε περίπτωση που είναι κωδικοποίηση κάποιας μηχανής Turing M k ταινιών, προσομοιώνει την λειτουργία της M με είσοδο την ίδια συμβολοσειρά w_M όπως η καθολική μηχανή U . Η μόνη διαφορά είναι ότι η M_1 αποδέχεται όταν η M απορρίπτει την είσοδο της και αντίστροφα. Για να ελέγξει αν η M σταματά σε $f(n)$ βήματα όταν η είσοδος της είναι η w_M , η M_1 χρησιμοποιεί μία μηχανή M^* η οποία υπολογίζει την $f(n)$ σε χρόνο $f(n)$. Η M^* χρησιμοποιείται από την M_1 σαν μετρητής των βημάτων της M και υπάρχει επειδή η $f(n)$ είναι καλά υπολογίσιμη. (Αυτός είναι ο λόγος που είναι απαραίτητη αυτή η υπόθεση.) Αν η M απαιτήσει περισσότερο χρόνο από $f(n)$, τότε η M_1 απορρίπτει. Ο χρόνος της προσομοίωσης προκύπτει αν θυμηθούμε τις λεπτομέρειες της καθολικής μηχανής. Κάθε κίνηση της M προσομοιώνεται σε δύο φάσεις και ένα επιπλέον σταθερό αριθμό κινήσεων. Κάθε φάση απαιτεί την σάρωση της κωδικοποιημένης περιγραφής της M . Όμως αυτή είναι η

συμβολοσειρά εισόδου της M και άρα κάθε φάση διαρκεί n βήματα. Υπάρχει μία επιπλέον απώλεια για την μετατροπή της M σε μηχανή δύο ταινιών. Με βάση το Θεώρημα 2.4.2 αυτή η μετατροπή γίνεται σε χρόνο $f(n) \log f(n)$, οπότε η μηχανή M_1 είναι 4 ταινιών και πολυπλοκότητας χρόνου $nf(n) \log f(n)$. Αν επιλέξουμε η M_1 να είναι 3 ταινιών τότε με βάση το Θεώρημα 2.3 θα έχει πολυπλοκότητα $nf(n)^2$.

Η απόδειξη ότι $L \notin \text{TIME}(f(n))$ χρησιμοποιεί την μέθοδο της «διαγωνιοποίησης». Έστω, αντίθετα από την υπόθεση, ότι $L \in \text{TIME}(f(n))$. Τότε θα πρέπει να υπάρχει μία μηχανή Turing M_2 με πολυπλοκότητα χρόνου $f(n)$ η οποία αποφασίζει την L . Σύμφωνα με το Λήμμα 3.1, μπορούμε να υποθέσουμε ότι η M_2 σταματά σε όλες τις εισόδους μήκους n , σε ακριβώς $f(n)$ βήματα. Ας προσπαθήσουμε τώρα να δούμε αν η συμβολοσειρά w_{M_2} που κωδικοποιεί την M_2 ανήκει στην L . Αν $w_{M_2} \in L$ (δηλαδή $M_2(w_{M_2}) = \text{ΝΑΙ}$, εφόσον η M_2 αποφασίζει την L) τότε επειδή πάντα σταματά σε $f(n)$ βήματα, η M_2 απορρίπτει την δική της κωδικοποίηση και $M_2(w_{M_2}) = \text{ΟΧΙ}$ (από τον ορισμό της L), άτοπο. Αν πάλι $w_{M_2} \notin L$ (δηλαδή $M_2(w_{M_2}) = \text{ΟΧΙ}$) τότε η $M_2(w_{M_2}) = \text{ΝΑΙ}$ (πάλι από τον ορισμό της L), πάλι άτοπο. Συνεπώς η υπόθεση ότι η M_2 υπάρχει είναι λάθος και άρα $L \notin \text{TIME}(f(n))$. \square

Αλλάζοντας ελαφρά την γλώσσα L , έτσι ώστε να υπάρχει ένα πρόθεμα από 1 κατάλληλου μήκους εμπρός από την w_M , ο συντελεστής n μπορεί να παραληφθεί [7], βλ. επίσης και ασκήσεις.

Πόρισμα 3.1 Το P περιέχεται γνήσια στο EXPTIME :

$$P \subset \text{EXPTIME}$$

Απόδειξη: $P = \bigcup_{k>0} \text{TIME}(n^k)$. Όμως $\text{TIME}(n^k) \subseteq \text{TIME}(2^n)$, και από το Θεώρημα της Ιεραρχίας $\text{TIME}(2^n) \subset \text{TIME}(n^2 2^n) \subseteq \text{TIME}(2^{3n}) \subseteq \text{EXPTIME}$. \square

Θεώρημα 3.2 (Ιεραρχίας χώρου) Αν η συνάρτηση $f(n)$ είναι κατασκευάσιμη τότε υπάρχει κάποια γλώσσα που ανήκει στο $\text{SPACE}(g(n)f(n))$ αλλά όχι στο $\text{SPACE}(f(n))$. Η $g(n)$ είναι οποιαδήποτε συνάρτηση που αυξάνει ταχύτερα από $\log n$ δηλαδή $g(n) = \Omega(\log n)$.

Απόδειξη: Η απόδειξη βασίζεται σε αυτή του Θεωρήματος 3.1. Η γλώσσα L είναι τώρα:

$$L = \{w_M \mid \text{η μηχανή } M \text{ απορρίπτει την } w_M \text{ σε χώρο το πολύ } f(|w_M|)\}$$

Η απόδειξη ότι $L \notin \text{SPACE}(f(n))$ γίνεται όπως και στο παραπάνω Θεώρημα με διαγωνιοποίηση. Το περιθώριο χώρου στο οποίο γίνεται ο υπολογισμός της L

βρίσκεται με εκτίμηση των αντίστοιχων απαιτήσεων σε χώρο της μηχανής M_1 . Ο παράγοντας $\log n$ προκύπτει από την ανάγκη κωδικοποίησης των επιπλέον συμβόλων της M και το πολύ $\log n$ σύμβολα απαιτούνται για κάθε σύμβολο της M . \square

Πόρισμα 3.2 Η κλάση L περιέχεται γνήσια στην $PSPACE$:

$$L \subset PSPACE$$

Απόδειξη: $L = SPACE(\log n) \subset SPACE(n^2) \subseteq PSPACE$. Ο πρώτος εγκλεισμός προκύπτει από το Θεώρημα 3.2. \square

3.3 Θεωρήματα Χάσματος

Στην προηγούμενη ενότητα δείξαμε ότι αν αυξήσουμε τους διαθέσιμους υπολογιστικούς πόρους, τότε μπορούμε να αναγνωρίσουμε επιπλέον γλώσσες. Οι επιπλέον υπολογιστικοί πόροι παριστάνονται με κατάλληλη αύξηση της συνάρτησης πολυπλοκότητας, ανάλογα αν ο πόρος είναι ο χρόνος ή ο χώρος. Ήταν όμως σημαντική προϋπόθεση στην απόδειξη των αντίστοιχων θεωρημάτων, οι συναρτήσεις πολυπλοκότητας να είναι είτε καλά υπολογίσιμες είτε κατασκευάσιμες ώστε να μπορεί να τεθεί ένα *a priori* όριο στους υπολογισμούς μας. Όταν εγκαταλείψουμε αυτή την υπόθεση τότε τα παραπάνω παύουν να ισχύουν και αντίθετα μπορεί να δειχθούν αποτελέσματα αρκετά περίεργα.

Θεώρημα 3.3 (Θεώρημα χάσματος [3, 29]) Για κάθε αναδρομική συνάρτηση $f(n) > n$, υπάρχει μία αναδρομική συνάρτηση $g(n)$ τέτοια ώστε κάθε γλώσσα L που υπολογίζεται σε χρόνο $f(g(n))$ να υπολογίζεται επίσης σε χρόνο $g(n)$. Τυπικά: $TIME(f(g(n))) = TIME(g(n))$.

Απόδειξη: Κατ' αρχή ας παρατηρήσουμε ότι αυτό που λέει το θεώρημα είναι ότι μεταξύ $f(g(n))$ και $g(n)$ υπάρχει ένα «χάσμα» μέσα στο οποίο δεν οριοθετείται η πολυπλοκότητα χρόνου καμμίας γλώσσας.

Έστω λοιπόν $f(n) > 0$ μία αναδρομική συνάρτηση. Βασιζόμενοι στην $f(n)$ θα ορίσουμε την $g(n)$, με στόχο ακριβώς αυτό που προδιαγράφεται από το θεώρημα. Ας θεωρήσουμε κάποια αυθαίρετη διάταξη M_1, M_2, \dots όλων των μηχανών Turing. Η κωδικοποίηση που κάναμε στην Παράγραφο 2.3.2 μας επιτρέπει να το κάνουμε αυτό εύκολα: η διάταξη μπορεί να είναι η λεξικογραφική διάταξη των συμβολοσειρών που κωδικοποιούν τις μηχανές Turing¹. Ας περιοριστούμε προς στιγμή στο πεπερασμένο σύνολο των n πρώτων μηχανών αυτής

¹Η λεξικογραφική διάταξη είναι αυτή όπου οι μικρότερες συμβολοσειρές προηγούνται των μεγαλύτερων και μεταξύ ισομήκων συμβολοσειρών η διάταξη είναι όπως του λεξικού, θεωρώντας αυθαίρετα ότι το 0 προηγείται του 1.

της διάταξης, M_1, M_2, \dots, M_n για κάποιο συγκεκριμένο n . Η g για αυτή την τιμή του n , ορίζεται έτσι ώστε η πολυπλοκότητα χρόνου όλων των n πρώτων μηχανών να μην είναι μεταξύ $f(g(n))$ και $g(n)$. Δηλαδή $T_{M_i}(n) > f(g(n))$ ή $T_{M_i}(n) < g(n)$, $i = 1, 2, \dots, n$. Εδώ με T_M συμβολίζουμε την πολυπλοκότητα χρόνου της μηχανής M . Αυτό σημαίνει ότι για κάθε i , $1 \leq i \leq n$, η M_i , είτε:

1. σε κάθε είσοδο μεγέθους n , τερματίζει σε χρόνο μικρότερο ή ίσο του $g(n)$, είτε
2. υπάρχει κάποια συμβολοσειρά μήκους n με είσοδο την οποία η M_i τερματίζει σε χρόνο μεγαλύτερο του $f(g(n))$, είτε
3. υπάρχει κάποια συμβολοσειρά μήκους n με είσοδο την οποία η M_i δεν τερματίζει.

Θέλουμε να δώσουμε στην $g(n)$ μία τιμή k έτσι ώστε να ισχύουν τα παραπάνω. Πως όμως μπορεί να υπολογιστεί αυτό το k δεδομένου ότι για μερικές συμβολοσειρές κάποιες από τις μηχανές μπορεί να μην τερματίζουν καθόλου (περίπτωση 3); Αυτό γίνεται δοκιμάζοντας διαδοχικά όλους τους φυσικούς αρχίζοντας από $k = 1$ μέχρι να βρούμε ένα k που να πληρεί τα παραπάνω. Για μία δεδομένη τιμή του k υπολογίζεται το $f(k)$ και στην συνέχεια προσομοιώνονται διαδοχικά όλες οι μηχανές M_1, M_2, \dots, M_n με εισόδους όλες τις δυνατές συμβολοσειρές μεγέθους n του αλφαβήτου τους. Οι προσομοιώσεις τερματίζονται σε $f(k)$ βήματα για κάθε είσοδο και επειδή ο αριθμός των μηχανών και ο αριθμός των πιθανών εισόδων είναι πεπερασμένοι, η όλη διαδικασία για δεδομένο k θα περατωθεί σε πεπερασμένο χρόνο. Αν ευρεθεί μία μηχανή που ο μέγιστος χρόνος που τερματίζει μεταξύ όλων των συμβολοσειρών είναι κάποιος αριθμός μεταξύ $k + 1$ και $f(k)$, τότε η τιμή αυτή του k απορρίπτεται και δοκιμάζουμε το $k + 1$. Επειδή το κάτω άκρο του διαστήματος των «απαγορευμένων» τιμών $(k, f(k)]$ συνεχώς αυξάνει, η διαδικασία αυτή οπωσδήποτε θα μας δώσει ένα τέτοιο k^* (το k θα ξεπεράσει την μέγιστη πολυπλοκότητα μεταξύ όλων των μηχανών από τις M_1, \dots, M_n με πεπερασμένη πολυπλοκότητα). Θέτουμε $g(n) = k^*$. Συνεπώς η $g(n)$, έστω και αν ορίστηκε με περιέργο τρόπο, είναι μια καλά ορισμένη, υπολογίσιμη συνάρτηση.

Ας υποθέσουμε τώρα ότι υπάρχει μία γλώσσα L που ανήκει στην κλάση $\text{TIME}(f(g(n)))$ αλλά όχι στην $\text{TIME}(g(n))$. Αυτό σημαίνει ότι η L αποφασίζεται από μια μηχανή Turing M_j με πολυπλοκότητα χρόνου $T_{M_j}(n)$ με $T_{M_j}(n) \leq f(g(n))$, όπου j κάποιος αμέριστος ανεξάρτητος του n . Όμως για όλα τα $n > j$ έχουμε, από τον τρόπο ορισμού της $g(n)$, ότι $g(n) \geq T_{M_j}(n)$, και άρα η L αποφασίζεται από μία μηχανή της οποίας η πολυπλοκότητα για όλα τα μήκη συμβολοσειρών εισόδου είναι μικρότερη ή ίση του $g(n)$, εκτός από ένα πεπερασμένο πλήθος εισόδων (αυτές που είναι μικρότερες ή ίσες του j)

για τις οποίες δεν γνωρίζουμε την συμπεριφορά της M_j . Αλλάζοντας λοιπόν το σύνολο καταστάσεων της M_j μπορούμε να την μετατρέψουμε σε μία M'_j η οποία σε κάθε συμβολοσειρά μήκους n που γίνεται αποδεκτή, απαιτεί χρόνο το πολύ $g(n)$. Η L συνεπώς ανήκει στο $\text{TIME}(g(n))$, άτοπο. \square

Ανάλογα με το 3.3 θεωρήματα ισχύουν και για τα υπόλοιπα τρία μέτρα πολυπλοκότητας που μας ενδιαφέρουν. Οι αποδείξεις είναι παρόμοιες με την παραπάνω με τις ανάλογες προσαρμογές στα φράγματα προσομοίωσης των μηχανών.

3.4 Σχέσεις μεταξύ κλάσεων πολυπλοκότητας

Εξετάζοντας τα δύο σημαντικότερα μέτρα πολυπλοκότητας, χρόνο και χώρο, μπορούμε να δείξουμε ορισμένες γενικές σχέσεις μεταξύ διαφόρων κλάσεων πολυπλοκότητας, δηλαδή σχέσεις που ισχύουν για κάθε συνάρτηση πολυπλοκότητας. Μοναδική απαίτηση, αν τεθεί, θα είναι η συνάρτηση να είναι καλά υπολογίσιμη ή κατασκευάσιμη. Τέτοιες γενικές σχέσεις δεν υπάρχουν ιδιαίτερα πολλές, μερικές δε είναι περίπου προφανείς. Οι περισσότερες δύσκολες από αυτές αφορούν τον μη ντετερμινιστικό χώρο.

3.4.1 Μη ντετερμινιστικός χρόνος

Ξεκινώντας από τις απλούστερες σχέσεις μπορούμε αμέσως να διατυπώσουμε το παρακάτω θεώρημα:

Θεώρημα 3.4 Για συνάρτηση πολυπλοκότητας $f(n)$ ισχύει:

$$\text{TIME}(f(n)) \subseteq \text{NTIME}(f(n)) \quad (3.1)$$

και αντίστοιχα:

$$\text{SPACE}(f(n)) \subseteq \text{NSPACE}(f(n)) \quad (3.2)$$

Απόδειξη: Κάθε ντετερμινιστική μηχανή μπορεί να ειπωθεί σαν μία μη ντετερμινιστική η οποία σε κάθε βήμα έχει μόνο μία επιλογή. Συνεπώς οποιαδήποτε γλώσσα γίνεται δεκτή σε ντετερμινιστικό χρόνο (αντίστοιχα χώρο) $f(n)$, γίνεται δεκτή και στον ίδιο μη ντετερμινιστικό χρόνο (αντίστοιχα χώρο). \square

Είναι εξαιρετικά σημαντικό το ερώτημα αν οι παραπάνω εγκλεισμοί είναι γνήσιοι ή όχι. Για συγκεκριμένες συναρτήσεις έχει δειχθεί αυτό (π.χ. γραμμικό χρόνο [25]), στη γενική περίπτωση όμως παραμένει ανοικτό πρόβλημα. Στη συνέχεια θα δώσουμε ένα φράγμα για τον μη ντετερμινιστικό υπολογισμό εξετάζοντας με τι απώλειες σε χρόνο θα μπορούσε μια συμβατική μηχανή να αποδεχθεί την γλώσσα που αποδέχεται μια μη ντετερμινιστική. Είχαμε ήδη

επισημάνει στο Κεφάλαιο 2 ότι οι πολλαπλές επιλογές της μη ντετερμινιστικής μηχανής αυξάνουν με εκθετικό τρόπο τα δυνατά υπολογιστικά μονοπάτια. Το παρακάτω θεώρημα επιβεβαιώνει αυτό ακριβώς:

Θεώρημα 3.5 Έστω $f(n)$ μια συνάρτηση πολυπλοκότητας. Τότε για κάποια σταθερά $c > 1$ ισχύει:

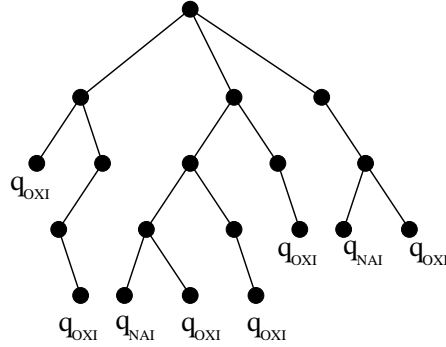
$$\text{NTIME}(f(n)) \subseteq \text{TIME}(c^{f(n)}) \quad (3.3)$$

Απόδειξη: Έστω L μία γλώσσα και M μία μη ντετερμινιστική μηχανή Turing με πολυπλοκότητα χρόνου $f(n)$ που αποδέχεται την L . Σύμφωνα με την παραπάνω παρατήρηση, μπορούμε να υποθέσουμε ότι κάθε υπολογισμός της M ανεξάρτητα από το αν οδηγεί σε αποδοχή ή όχι, διαρκεί ακριβώς $f(n)$ βήματα. Στο Σχήμα 3.1 παριστάνεται το ανάπτυγμα του μη ντετερμινιστικού υπολογισμού της M σε μία συμβολοσειρά $x \in L$. Κάθε κόμβος παριστάνει μία διαμόρφωση και τα παιδιά του τις δυνατές διαμορφώσεις στις οποίες μπορεί να μεταβεί. Η ρίζα του δένδρου είναι η αρχική διαμόρφωση. Η προσομοίωση του μη ντετερμινιστικού υπολογισμού από μια ντετερμινιστική μηχανή M' , μπορεί να γίνει με εξέταση όλων των δυνατών μεταβάσεων που μπορεί να κάνει η M σε κάθε βήμα. Στο σχήμα, αυτό ισοδυναμεί με επίσκεψη, με κάποια σειρά, όλων των κόμβων του δένδρου μέχρι να εξεταστεί αν υπάρχει φύλλο με κατάσταση q_{NAI} . Σε τέτοια περίπτωση η M' θα αποδεχθεί, αλλιώς, αν δεν υπάρχει φύλλο με κατάσταση q_{NAI} , θα απορρίψει την είσοδο. Πόσο χρόνο θα πάρει αυτή η διαδικασία; Αν υποθέσουμε ότι ο μέγιστος αριθμός μεταβάσεων της M σε κάθε σημείο του υπολογισμού της είναι c (αυτό στο σχήμα φαίνεται με τον μέγιστο βαθμό του δένδρου), τότε υπάρχουν το πολύ $c^{f(n)}$ κόμβοι στο δένδρο σε βάθος μέχρι $f(n)$. Η M' πρέπει να επισκεφθεί αυτούς το πολύ τους κόμβους μέχρι να ανακαλύψει αν υπάρχει κόμβος με κατάσταση q_{NAI} . \square

3.4.2 Μη ντετερμινιστικός χώρος

Για τις σχέσεις που αφορούν τον μη ντετερμινιστικό χώρο είναι ιδιαίτερα χρήσιμη η παράσταση του μη ντετερμινιστικού υπολογισμού σαν ένα κατευθυντικό γράφημα, όπως αποδείχθηκε χρήσιμο για τον μη ντετερμινιστικό χρόνο, το «ανάπτυγμα» του υπολογισμού σε ένα δένδρο.

Συγκεκριμένα ας θεωρήσουμε ένα κατευθυντικό γράφημα (ένα γράφημα στο οποίο κάθε ακμή έχει αρχή και τέλος) στο οποίο κάθε κόμβος είναι μία διαμόρφωση της μη ντετερμινιστικής μηχανής μας M . Θα συμβολίζουμε το γράφημα αυτό με G_M . Όλες οι πιθανές διαμορφώσεις της μηχανής παριστάνονται στο γράφημα αυτό. Αν η μηχανή έχει πολυπλοκότητα χώρου $f(n)$, τότε ο αριθμός των διαφορετικών διαμορφώσεων φράσσεται από την ποσότητα $c|K|f(n)|\Sigma|^{f(n)}$. Το K είναι το σύνολο των καταστάσεων και το Σ το



Σχήμα 3.1: Ανάπτυγμα μη ντετερμινιστικού υπολογισμού.

αλφάβητο της M . Η ποσότητα $|\Sigma|^{f(n)}$ μετρά όλες τις δυνατές συμβολοσειρές μεγέθους $f(n)$ (το φράγμα χώρου της μηχανής) στο αλφάβητο Σ . Ο συντελεστής $f(n)$ υπάρχει διότι οι κεφαλές της μηχανής μπορούν να βρίσκονται σε $f(n)$ διαφορετικές θέσεις. Τέλος το c είναι μια σταθερά εξαρτώμενη από τον αριθμό των ταινιών. Δύο κόμβοι I_1 και I_2 ενώνονται με την ακμή (I_1, I_2) αν η M μπορεί σε ένα βήμα να μεταβεί από την διαμόρφωση I_1 στην διαμόρφωση I_2 . Υπάρχει ακριβώς μία αρχική διαμόρφωση στο γράφημα αυτό, έστω η I_s , και ένας αριθμός από διαμορφώσεις αποδοχής στις οποίες η κατάσταση είναι η q_{NAI} . Το πρόβλημα λοιπόν της αποδοχής ή όχι μιας εισόδου από την μηχανή, ανάγεται στην ύπαρξη ενός μονοπατιού από την I_s σε μία τουλάχιστον από τις καταστάσεις αποδοχής. Το πρόβλημα αυτό είναι ένα γνωστό γραφοθεωρητικό πρόβλημα και ονομάζεται Πρόβλημα της ΠΡΟΣΠΕΛΑΣΙΜΟΤΗΤΑΣ (REACHABILITY problem) και τυπικά ένα στιγμιότυπο του αποτελείται από ένα κατευθυντικό γράφημα και δύο κόμβους s και t . Ζητείται αν υπάρχει μονοπάτι από τον s στον t . Το πρόβλημα λύνεται εύκολα σε χρόνο $O(n^2)$ [5] όπου n ο αριθμός των κόμβων του γραφήματος.

Η σχέση μεταξύ χώρου και χρόνου φαίνεται στο παρακάτω θεώρημα:

Θεώρημα 3.6 Έστω $f(n) \geq \log n$ μια κατασκευάσιμη συνάρτηση πολυπλοκότητας. Τότε για κάποια σταθερά $c > 1$ ισχύει:

$$\text{SPACE}(f(n)) \subseteq \text{TIME}(c^{f(n)}) \quad (3.4)$$

Απόδειξη: Έστω M_1 μια μηχανή Turing με πολυπλοκότητα χώρου $f(n)$. Ο αριθμός των διαφορετικών διαμορφώσεων της M_1 φράσσεται σύμφωνα με τα παραπάνω από την ποσότητα $g(n) = c_1 |K| f(n) |\Sigma|^{f(n)}$. Μια μηχανή M_2 μπορεί να προσομοιώσει την M_1 χρησιμοποιώντας επιπλέον μια μηχανή M' σαν απεριθμητή για τα βήματα της M_1 . Αν η προσομοίωση περατωθεί σε χρόνο μικρότερο από $g(n)$ τότε η M_2 αποδέχεται αν και μόνο αν η M_1 αποδεχθεί. Αν αντίθετα η προσομοίωση ξεπεράσει σε χρόνο το $g(n)$ τότε κάποια διαμόρφωση

έχει επαναληφθεί και συνεπώς η M_1 έχει πέσει σε loop και δεν πρόκειται να αποδεχθεί. Το αποτέλεσμα προκύπτει αν παρατηρήσουμε ότι $g(n) \leq c^{f(n)}$ για κάποια σταθερά c . \square

Θεώρημα 3.7 (Θεώρημα Savitch [26].) Έστω $f(n) \geq \log n$ μια κατασκευάσιμη συνάρτηση. Τότε ισχύει: $\text{NSPACE}(f(n)) \subseteq \text{SPACE}(f(n)^2)$.

Απόδειξη: Η απόδειξη βασίζεται σε μια έξυπνη προσομοίωση μιας μη ντετερμινιστικής μηχανής Turing M η οποία έχει πολυπλοκότητα χώρου $f(n)$ από μια ντετερμινιστική M_1 σε χώρο $f(n)^2$.

Ας υποθέσουμε ότι η M_1 έχει διαθέσιμο το γράφημα G_M των διαμορφώσεων της M όπως περιγράφηκε παραπάνω. Η υπόθεση αυτή δεν είναι ρεαλιστική αν θέλουμε η M_1 να χρησιμοποιήσει χώρο $f(n)^2$ μια και υπάρχουν $O(c^{f(n)})$ διαμορφώσεις, κάθε μία από τις οποίες απαιτεί χώρο $f(n)$. Αργότερα θα δούμε πως θα παρακάμψουμε αυτή τη δυσκολία. Ο αλγόριθμος μας θα βασιστεί σε μία λογική (boolean) υπορουτίνα $Check(I_1, I_2, i)$ με τρεις παραμέτρους. Οι δύο πρώτες είναι διαμορφώσεις της μηχανής M (κόμβοι στο γράφημα G_M) και η τρίτη ένας ακέραιος αριθμός. Ο στόχος είναι η $Check(I_1, I_2, i)$ να είναι αληθής (*true*) αν η διαμόρφωση I_1 μπορεί να οδηγήσει σε 2^i το πολύ βήματα στην διαμόρφωση I_2 . Στο γράφημα G_M αυτό είναι ισοδύναμο με το να υπάρχει μονοπάτι μήκους το πολύ 2^i από τον κόμβο I_1 στον I_2 . Η υλοποίηση της $Check$ γίνεται με βάση την εξής παρατήρηση: υπάρχει μονοπάτι μήκους το πολύ 2^i για $i \geq 1$ από τον I_1 στον I_2 αν υπάρχει μονοπάτι μήκους το πολύ 2^{i-1} από τον I_1 σε κάποιο ενδιάμεσο κόμβο I' και επίσης μονοπάτι μήκους το πολύ 2^{i-1} από τον I' στον I_2 . Αν $i = 0$ τότε η $Check(I_1, I_2, i)$ είναι αληθής αν υπάρχει ακμή από τον I_1 στον I_2 ή αλλιώς, η μηχανή M μπορεί να μεταβεί σε ένα βήμα από την διαμόρφωση I_1 στην I_2 . Έχοντας διαθέσιμη την $Check$, και λαμβανομένου υπόψη ότι η μέγιστη απόσταση μεταξύ δύο οποιονδήποτε κόμβων στον G_M είναι το πολύ $c^{f(n)}$, η M_1 πρέπει να ελέγξει αν υπάρχει μονοπάτι μήκους το πολύ $c^{f(n)}$ μεταξύ της αρχικής διαμόρφωσης και κάθε μιας από τις διαμορφώσεις αποδοχής I_{NAI} . Όλη προσομοίωση φαίνεται στον Αλγόριθμο 3.

Η ορθότητα της προσομοίωσης είναι προφανής. Η M_1 θα εξετάσει διαδοχικά όλες τις δυνατές διαμορφώσεις αποδοχής (δοκιμάζοντας όλες τις δυνατές συμβολοσειρές μήκους $f(n)$ και όλες τις δυνατές θέσεις της κεφαλής) και για κάθε μία από αυτές θα δοκιμάσει αν η $Check$ επιστρέφει *true*. Αν έστω και για μία επιστραφεί *true* τότε η M_1 θα αποδεχθεί, αλλιώς θα απορρίψει.

Ως προς τις απαιτήσεις χώρου, τα πράγματα είναι πιο σύνθετα. Κατ' αρχήν παρατηρούμε ότι μια διαμόρφωση απαιτεί χώρο $O(f(n))$ για να καταγραφεί. Αν συνεπώς μπορέσουμε να περιορίσουμε την $Check$ στον απαιτούμενο χώρο $f(n)^2$, τότε όλες οι διαμορφώσεις αποδοχής μπορούν να δοκιμαστούν στα πλαίσια του χώρου που έχουμε, με επαναχρησιμοποίηση του χώρου. Πόσο χώρο

Input H περιγραφή μιας μηχανής Turing M

```

procedure Check( $I_1, I_2, i$ ) {Απαντά «true» αν η διαμόρφωση  $I_1$  μπορεί
σε  $2^i$  το πολύ βήματα να οδηγήσει στην διαμόρφωση  $I_2$ }
begin { Check }
if  $i = 0$  and (η  $I_1$  μπορεί να οδηγήσει σε ένα βήμα στην  $I_2$ ) then
    return true;
end if
if  $i > 0$  then
    for κάθε διαμόρφωση  $I'$  στον  $G_M$  do
        if Check( $I_1, I', i - 1$ ) and Check( $I', I_2, i - 1$ ) then
            return true;
        end if
    end for
end if
end { Check }

begin { Main }
for κάθε διαμόρφωση αποδοχής  $I_{NAI}$  do
    if Check( $I_s, I_{NAI}, \lceil f(n) \log c \rceil$ ) then
        return true
    end if
end for
end { Main }

```

Αλγόριθμος 3: Αλγόριθμος προσομοίωσης.

όμως καταλαμβάνει η *Check*; Κατ' αρχή οι παράμετροι μιας κλήσης της *Check* απαιτούν χώρο $O(f(n))$. Επίσης όλες οι μεταβλητές στο σώμα της υπορουτίνας καθώς και οι παράμετροι των δύο αναδρομικών κλήσεων της, επίσης μπορούν να καταγραφούν σε χώρο $O(f(n))$. Όμως από την αναδρομική μορφή της *Check* υπάρχει απαίτηση για αποθήκευση όλων των κλήσεων της μέχρι το μέγιστο βάθος της αναδρομής. Επειδή σε κάθε αναδρομική κλήση η παράμετρος i μικραίνει κατά 1, και η αρχική τιμή της ήταν $\lceil f(n) \log c \rceil = O(f(n))$ συμπεραίνουμε ότι καθ' όλη την διάρκεια της προσομοίωσης μπορούν να υπάρξουν το πολύ $O(f(n))$ «ενεργές» κλήσεις της *Check*. Ο συνολικός απαιτούμενος χώρος είναι λοιπόν $O(f(n)^2)$ που μπορεί να γίνει ακριβώς $f(n)^2$ χρησιμοποιώντας το Θεώρημα 2.2. Απομένει επομένως το πρόβλημα του απαγορευτικού μεγέθους του γραφήματος G_M . Αν όμως παρατηρήσουμε προσεκτικά τον Αλγόριθμο 3 θα δούμε ότι δεν απαιτείται η καταγραφή του G_M . Το μόνο που απαιτείται είναι η δυνατότητα να αποφασίζει η M_1 για το αν δύο διαμορφώσεις της M

συνδέονται με ακμή. Κάτι τέτοιο προκύπτει από την συνάρτηση μετάβασης της M και απαιτεί σταθερό χώρο για να αποφασιστεί. \square

Παρατήρηση 3.2 Στην Ενότητα 2.6 ορίσαμε την κλάση πολυωνυμικού χώρου $PSPACE = \bigcup_{k>0} SPACE(n^k)$. Δεν ορίστηκε καμμία κλάση μη ντετερμινιστικού πολυωνυμικού χώρου (που μάλλον θα έπαιρνε το όνομα $NPSPACE$) και το Θεώρημα Savitch το εξηγεί αυτό: $NSPACE(n^k) \subseteq SPACE(n^{2k})$ για κάθε k και άρα $NPSPACE = PSPACE$.

Παρατήρηση 3.3 Το Θεώρημα του Savitch ποιοτικά λέγει ότι μια φραγμένη ως προς τον χώρο μη ντετερμινιστική μηχανή Turing δεν είναι πολύ ισχυρότερη από μια ντετερμινιστική. Η τελευταία απαιτεί το πολύ το τετράγωνο του χώρου της πρώτης. Διαισθητικά αυτό οφείλεται στο ότι οι επαναλήψεις των υπολογισμών που φαίνεται να είναι η μοναδική μέθοδος που μια ντετερμινιστική μηχανή προσομοιώνει μια μη ντετερμινιστική, μπορούν να ξαναχρησιμοποιήσουν τον ίδιο χώρο. Αντίθετα, μια μη ντετερμινιστική μηχανή φαίνεται να είναι σημαντικά ισχυρότερη ως προς τον χρόνο, οι επαναλήψεις κοστίζουν εκθετικά σε επιπλέον χρόνο (βλ. Θεώρημα 3.5).

Το επόμενο Θεώρημα επίσης αποτελεί μια ακόμη ισχυρότερη ένδειξη προς αυτή την κατεύθυνση. Ας θυμηθούμε (βλ. Ορισμό 2.20) ότι η συμπληρωματική μιας κλάσης πολυπλοκότητας C αποτελείται από τις συμπληρωματικές γλώσσες της C ή ισοδύναμα από τα υπολογιστικά προβλήματα στα οποία, με την ίδια διατύπωση, ενδιαφέρουν τα στιγμιότυπα με απάντηση «ΟΧΙ». Αν η C είναι μια μη ντετερμινιστική κλάση, και δεδομένου του τρόπου αποδοχής μιας μη ντετερμινιστικής μηχανής, η αναγνώριση τέτοιων στιγμιότυπων φαίνεται να απαιτεί τον εξαντλητικό έλεγχο όλων των δυνατών υπολογισμών ώστε να βεβαιωθούμε ότι δεν υπάρχει υπολογισμός με «ΝΑΙ». Είναι σημαντικό να δούμε ότι, ακόμη και αν έχουμε στην διάθεση μας για τον σκοπό αυτό μια μη ντετερμινιστική μηχανή, δεν φαίνεται να μπορούμε να αποφύγουμε την εξαντλητική αναζήτηση. Όσον αφορά λοιπόν μη ντετερμινιστικές κλάσεις χρόνου, φαίνεται ότι πράγματι υπάρχει διαφορά στην πολυπλοκότητα της κλάσης $NTIME(f(n))$ από την συμπληρωματική της $co-NTIME(f(n))$ ². Το ίδιο πιστευόταν για περίπου 25 χρόνια για τον μη ντετερμινιστικό χώρο. Το παρακάτω Θεώρημα 3.8 όμως δείχνει ότι το σύνολο των γλωσσών που αναγνωρίζονται σε κάποια όρια χώρου από μια μη ντετερμινιστική μηχανή, είναι κλειστό ως προς το συμπλήρωμα, δηλαδή αν στο σύνολο αυτό περιλαμβάνεται μία γλώσσα L τότε περιλαμβάνεται και η \bar{L} .

²Χρησιμοποιούμε συνεχώς τον όρο «φαίνεται» γιατί δεν υπάρχει επί του παρόντος απόδειξη για αυτό. Μόνο ισχυρές ενδείξεις

Για την απόδειξη θα χρησιμοποιήσουμε πάλι το γράφημα των διαμορφώσεων μιας μηχανής Turing. Αρχίζουμε με την απόδειξη ενός λήμματος για το πρόβλημα της REACHABILITY (μάλιστα μία ελαφρά ισχυρότερη μορφή του) που έχει ενδιαφέρον από μόνο του.

Λήμμα 3.3 Έστω ένα κατευθυντικό γράφημα G με n κόμβους, δύο συγκεκριμένοι κόμβοι s και t και ακέραιος k . Τότε το πρόβλημα του κατά πόσο υπάρχει μονοπάτι μήκους το πολύ k από τον s στον t μπορεί να αποφασιστεί σε μη ντετερμινιστικό χώρο $\log n$.

Απόδειξη: Μια μη ντετερμινιστική μηχανή κατασκευάζει ένα μονοπάτι μήκους το πολύ k από τον s καταχωρώντας μόνο το εκάστοτε τέλος του μονοπατιού με τον τρόπο που φαίνεται στον Αλγόριθμο 4. Στην αρχή το τέλος είναι ο κόμβος s . Στην συνέχεια και για $k - 1$ βήματα η μηχανή μαντεύει ένα κόμβο u από τους n και ελέγχει αν υπάρχει ακμή από το τρέχον τέλος προς τον u . Αν υπάρχει τότε το τρέχον τέλος γίνεται ο u . Αν μάλιστα $u = t$ τότε η μηχανή αποδέχεται.

```

procedure reachability( $G, s, t, k$ )
  {Γράφημα  $G$ , κόμβοι  $s$  και  $t$  και ακέραιος  $k$ }
1: current_end :=  $s$ ;
2: for  $i := 1$  to  $k - 1$  do
3:   μάντεψε ένα κόμβο  $u$ ;
4:   if υπάρχει ακμή (current_end,  $u$ ) then
5:     current_end :=  $u$ 
6:   end if
7:   if current_end =  $t$  then
8:     then return true
9:   end if
10: end for

```

Αλγόριθμος 4: Αλγόριθμος ΠΡΟΣΠΕΛΑΣΙΜΟΤΗΤΑΣ.

Ο χώρος που απαιτείται είναι ο χώρος που χρειάζεται για την καταγραφή των 5 μεταβλητών που χρησιμοποιούνται και είναι $O(\log n)$ εφόσον κάθε μία περιέχει ένα αριθμό από το 1 έως το n . \square

Το πραγματικά ενδιαφέρον όμως γραφοθεωρητικό πρόβλημα που θα δείξουμε στον δρόμο προς το τελικό μας θεώρημα, είναι το *συμπληρωματικό* της ΠΡΟΣΠΕΛΑΣΙΜΟΤΗΤΑΣ το πρόβλημα της ΜΗ ΠΡΟΣΠΕΛΑΣΙΜΟΤΗΤΑΣ (NON-REACHABILITY problem): δοθέντων γραφήματος G , κόμβων s και t και ακέραιου k , να αποφασιστεί αν δεν υπάρχει μονοπάτι μήκους k ή μικρότερο από τον s στον t . Κατ' αρχή πρέπει να παρατηρήσουμε ότι η υπορουτίνα *reachability*

που περιγράψαμε παραπάνω δεν επιλύει την ΜΗ ΠΡΟΣΠΕΛΑΣΙΜΟΤΗΤΑ. Πρόκειται για μη ντετερμινιστικό αλγόριθμο και κατά συνέπεια μόνο οι θετικές του απαντήσεις έχουν σημασία (δηλαδή αναγνωρίζει μόνο τα «ΝΑΙ» στιγμιότυπα).

Για την επίλυση του NON-REACHABILITY θα χρειαστεί να μετρήσουμε τους κόμβους που είναι στην γειτονιά ακτίνας k —την συμβολίζουμε με N_k — δηλαδή τους κόμβους σε απόσταση k ή μικρότερη από τον s . Δεν μπορούμε όμως να το κάνουμε αυτό απευθείας χρησιμοποιώντας την *reachability*. Και πάλι, μόνο τα θετικά στιγμιότυπα θα είχαν σημασία επειδή η *reachability* είναι μη ντετερμινιστική.

Για να υπολογίσουμε το μέγεθος της γειτονιάς ακτίνας k , $|N_k|$, υπολογίζουμε διαδοχικά τα μεγέθη των γειτονιών N_0, N_1, \dots, N_{k-1} . Προφανώς $|N_0| = 1$ (μόνο ο $s \in N_0$). Έστω ότι έχουμε το $|N_{k-1}|$. Για να υπολογίσουμε το $|N_k|$, σαρώνουμε διαδοχικά όλους τους κόμβους του G και για κάθε κόμβο v_i ελέγχουμε αν είτε $v_i \in N_{k-1}$ είτε υπάρχει ακμή από κάποιο κόμβο v_j του N_{k-1} προς τον v_i . Αυτό δεν μπορεί να γίνει απευθείας μια και το N_{k-1} δεν είναι διαθέσιμο ούτε βέβαια με την χρήση της *reachability* για τους λόγους που αναφέρθηκαν παραπάνω. Αντίθετα σαρώνουμε πάλι τους κόμβους του G και για κάθε κόμβο v_j ελέγχουμε αν $v_j \in N_{k-1}$. Ο έλεγχος αυτός γίνεται χρησιμοποιώντας την *reachability* και άρα μόνο τα θετικά στιγμιότυπα αναγνωρίζονται. Για τον λόγο αυτό κρατούμε ένα απαριθμητή m ο οποίος μετρά τους κόμβους v_j που ανήκουν στο N_{k-1} , ώστε αν στο τέλος βρεθούν λιγότεροι από το (γνωστό) $|N_{k-1}|$, να μπορέσουμε να απορρίψουμε αυτό τον συγκεκριμένο μη ντετερμινιστικό υπολογισμό. Αν λοιπόν $v_j \in N_{k-1}$ τότε εύκολα ελέγχουμε αν είτε $v_i = v_j$, (που σημαίνει $v_i \in N_{k-1} \subseteq N_k$, είτε αν υπάρχει ακμή (v_j, v_i) . Και στις δύο περιπτώσεις αυξάνουμε τον απαριθμητή για το μέγεθος της N_k . Ο μη ντετερμινιστικός αυτός αλγόριθμος, φαίνεται στον Αλγόριθμο 5.

Έχοντας λοιπόν διαθέσιμο το πλήθος των κόμβων της γειτονιάς ακτίνας k από τον s μπορούμε να λύσουμε τη ΜΗ ΠΡΟΣΠΕΛΑΣΙΜΟΤΗΤΑ. Στην πραγματικότητα ο Αλγόριθμος 5, το δίνει αυτό σαν παραπροϊόν.

Λήμμα 3.4 Έστω G ένα κατευθυντικό γράφημα με n κόμβους, s και t δύο διακεκριμένοι κόμβοι του G και k ένας ακέραιος. Τότε το πρόβλημα του κατά πόσο δεν υπάρχει μονοπάτι μήκους k ή μικρότερο από τον s στον t , λύνεται σε μη ντετερμινιστικό χώρο $\log n$.

Απόδειξη: Έστω $|N_k|$ το πλήθος των κόμβων του G που βρίσκονται σε απόσταση k ή μικρότερη από τον s . Έστω G' το γράφημα που προκύπτει από το G όταν αφαιρεθεί ο t και $|N'_k|$ το αντίστοιχο πλήθος κόμβων στο G' . Τότε δεν υπάρχει μονοπάτι στο G μήκους k ή μικρότερο από τον s στον t , ανν $|N_k| = |N'_k|$. Και οι δύο αυτές ποσότητες υπολογίζονται σε μη ντετερμινιστικό χώρο $\log n$ χρησιμοποιώντας τον Αλγόριθμο 5. \square

```

Input:  $|N_{k-1}|$  = το πλήθος των κόμβων σε απόσταση  $k-1$  ή μικρότερη από τον  $s$ .
Output:  $|N_k|$  = το πλήθος των κόμβων σε απόσταση  $k$  ή μικρότερη από τον  $s$ .

1:  $l := 0$ ;
2: for  $j := 1$  to  $n$  do
3:    $m := 0$ ;
4:   for  $j := 1$  to  $n$  do
5:     if reachability( $G, s, v_j, k-1$ ) then
6:        $m := m + 1$ ;
7:       if ( $v_i = v_j$ ) or (υπάρχει ακμή ( $v_j, v_i$ )) then
8:          $l := l + 1$ ;
9:       end if
10:    end if
11:  end for
12:  if  $m < |N_{k-1}|$  then
13:    exit {απορρίπτουμε αυτό τον υπολογισμό}
14:  end if
15: end for
16: return  $l$ ;

```

Αλγόριθμος 5: Αλγόριθμος μη ντετερμινιστικού $O(\log n)$ χώρου για το μέγεθος μιας k -γειτονιάς.

Θεώρημα 3.8 (Θεώρημα Immerman-Szelepcsényi [11, 28]). Για κάθε κατασκευάσιμη συνάρτηση $f(n) \geq \log n$ ισχύει ότι

$$NSPACE(f(n)) = co-NSPACE(f(n)).$$

Απόδειξη: Έστω L μια γλώσσα που αποφασίζεται από μια μη ντετερμινιστική μηχανή M_1 με πολυπλοκότητα χώρου $f(n)$. Μια μηχανή M_2 η οποία αποφασίζει την \bar{L} , πρέπει να αποδέχεται μία συμβολοσειρά ακριβώς όταν η M_1 απορρίπτει. Η M_2 χρησιμοποιεί το γράφημα των διαμορφώσεων της G_{M_1} με έμμεσο τρόπο όπως κάναμε στο Θεώρημα του Savitch. Στο γράφημα αυτό, το αν η συμβολοσειρά εισόδου απορρίπτεται είναι ισοδύναμο με το NON-REACHABILITY πρόβλημα όπου s είναι η αρχική διαμόρφωση I_s και t είναι μία διαμόρφωση αποδοχής. Η M_2 λοιπόν κατασκευάζει όλες τις δυνατές διαμορφώσεις αποδοχής της M_1 και ελέγχει σε μη ντετερμινιστικό χώρο $f(n)$ για μη προσπελασιμότητα από την I_s χρησιμοποιώντας τον αλγόριθμο του Λήμματος 3.4. \square

Κεφάλαιο 4

Πληρότητα

Η εύρεση της πολυπλοκότητας ενός προβλήματος είναι ένα δύσκολο εγχείρημα. Στην Εισαγωγή παρουσιάσαμε ένα θεώρημα για το πρόβλημα της ταξινόμησης n αριθμών. Όπως αναφέρθηκε εκεί τέτοια ακριβή αποτελέσματα είναι πολύ λίγα στην Θεωρία Πολυπλοκότητας και όσα υπάρχουν απαιτούν μεθόδους προσαρμοσμένες στις ιδιαιτερότητες του συγκεκριμένου προβλήματος. Η γενική μέθοδος που προσφέρει η Θεωρία Πολυπλοκότητας είναι η κατηγοριοποίηση του προβλήματος που εξετάζουμε σε μία κλάση πολυπλοκότητας της οποίας γνωρίζουμε τις ιδιότητες δηλαδή τον βαθμό δυσκολίας που παρουσιάζει η επίλυση των προβλημάτων σε αυτές. Έτσι για παράδειγμα η εύρεση ενός πολυωνυμικού αλγορίθμου για ένα πρόβλημα το κατατάσει στο P , δηλαδή στα προβλήματα που θεωρούμε ότι επιλύονται ικανοποιητικά. Αντίθετα ένα πρόβλημα στο $EXPTIME-P$ (το οποίο γνωρίζουμε από το θεώρημα της ιεραρχίας ότι είναι μη κενό) είναι ένα πρόβλημα για το οποίο μπορούμε βέβαια να επιχειρηματολογήσουμε ότι δεν επιλύεται ικανοποιητικά. Ένα τέτοιο αποτέλεσμα, ακόμη και αν δεν δίνει τον καλύτερο δυνατό αλγόριθμο για το πρόβλημα μας, εντούτοις δίνει τουλάχιστον ένα καλό κάτω φράγμα. Πως όμως μπορούμε να έχουμε αποτελέσματα σαν αυτά; Πως μπορούμε να δείξουμε ότι δεν υπάρχει ένας πολυωνυμικός αλγόριθμος για κάποιο πρόβλημα; Κατ' αρχή είναι προφανές ότι η απόδειξη ότι ένα πρόβλημα $L \in EXPTIME$ δεν είναι από μόνο του απόδειξη δυσκολίας μια και $P \subset EXPTIME$. Αυτό που στην πραγματικότητα θα θέλαμε, είναι να δείξουμε ότι το L είναι από τα δυσκολότερα προβλήματα στο $EXPTIME$. Γενικότερα μιλώντας (μια και «ενδιάμεσα» από το P και το $EXPTIME$ είναι ενδεχόμενο να υπάρχουν και άλλες ενδιαφέρουσες κλάσεις που γνήσια περιλαμβάνουν το P), θα θέλαμε να βρούμε ένα τρόπο απόδειξης ότι ένα πρόβλημα είναι από τα δυσκολότερα σε μια κλάση C .

4.1 Αναγωγές

Ο παραπάνω στόχος απαιτεί την εύρεση ενός τρόπου να πούμε ότι ένα πρόβλημα είναι τουλάχιστον τόσο δύσκολο όσο και ένα άλλο. Ο τρόπος αυτός είναι η αναγωγή ενός προβλήματος L_1 σε ένα άλλο L_2 . Με τον όρο αυτό εννοούμε ένα τρόπο να εξασφαλίσουμε μια μέθοδο επίλυσης του L_1 αν γνωρίζουμε ένα αλγόριθμο για το L_2 . Ο τρόπος αυτός απαιτεί την μετατροπή ενός στιγμιότυπου του L_1 σε ένα στιγμιότυπο του L_2 , έτσι ώστε να διατηρείται η απάντηση στα δύο στιγμιότυπα: το στιγμιότυπο του L_1 πρέπει να έχει απάντηση «ΝΑΙ» αν και μόνο αν το αντίστοιχο στιγμιότυπο του L_2 έχει απάντηση «ΝΑΙ». Αν βρεθεί μια τέτοια μετατροπή μπορούμε να βρούμε την απάντηση σε ένα στιγμιότυπο του L_1 , επιλύοντας την εικόνα του στο L_2 . Πρέπει όμως να θέσουμε και κάποιους περιορισμούς ως προς τον τρόπο μετατροπής μεταξύ των δύο προβλημάτων. Δεν είναι χρήσιμη μια μετατροπή η οποία είναι εξαιρετικά δύσκολο να υπολογιστεί μια και τότε η δυσκολία επίλυσης του L_1 κρύβεται πίσω από την δυσκολία της μετατροπής. Τις παραπάνω προϋποθέσεις εκπληρεί η αναγωγή λογαριθμικού χώρου.

Ορισμός 4.1 Έστω L_1 και L_2 δύο γλώσσες. Λέμε ότι η L_1 ανάγεται στην L_2 (γράφουμε $L_1 \leq L_2$), αν υπάρχει συνάρτηση $R : \Sigma^* \rightarrow \Sigma^*$ τέτοια που $x \in L_1$ αν και μόνο αν $R(x) \in L_2$. Η L_1 ανάγεται στην L_2 σε λογαριθμικό χώρο (γράφουμε $L_1 \leq_{LS} L_2$) αν επιπλέον η $R(x)$ υπολογίζεται από μια μηχανή Turing με είσοδο και έξοδο, σε χώρο $O(\log n)$.

Είναι εύκολο να δούμε ότι μία λογαριθμικού χώρου αναγωγή είναι και αναγωγή πολυωνυμικού χρόνου. Η συνάρτηση $R(x)$ δηλαδή μπορεί να υπολογιστεί σε πολυωνυμικό χρόνο. Η απόδειξη είναι ανάλογη με του Θεωρήματος 3.6. Συνεπώς αν $L_1 \leq_{LS} L_2$ και υπάρχει πολυωνυμικός αλγόριθμος για την L_2 τότε υπάρχει και πολυωνυμικός αλγόριθμος για την L_1 . Μπορούμε να αποφασίσουμε αν $x \in L_1$, υπολογίζοντας σε πολυωνυμικό χρόνο το $R(x)$ και αποφασίζοντας, επίσης σε πολυωνυμικό χρόνο, αν $R(x) \in L_2$. Ο λόγος που χρησιμοποιούμε την αναγωγή λογαριθμικού χώρου είναι για να υπάρχει ενιαία αναγωγή και για κλάσεις που φαίνονται «μικρότερες» από την P όπως η L και η NL. Όλες αυτές οι κλάσεις (καθώς και οι υπόλοιπες που ορίστηκαν στην Ενότητα 2.6) είναι κλειστές ως προς τις αναγωγές λογαριθμικού χώρου, δηλαδή αν $L_1 \leq_{LS} L_2$ και $L_2 \in C$, όπου C μία τέτοια κλάση, τότε και $L_1 \in C$.

Θεώρημα 4.1 Οι κλάσεις L, NL, P, NP, PSPACE και EXPTIME είναι κλειστές ως προς τις αναγωγές λογαριθμικού χώρου.

Απόδειξη: Είναι εύκολο να δούμε ότι με την σειρά αυτή κάθε κλάση περιέχεται στην επόμενη της. Το θεώρημα είναι εύκολο ναδειχθεί (όπως παραπάνω για την P) για όλες τις κλάσεις από την P και πέρα.

Υπάρχει ένα λεπτό σημείο για τις L και NL . Έστω $L_1 \in L$ (η περίπτωση της NL είναι παρόμοια) και έστω M_1 μια μηχανή Turing με πολυπλοκότητα χώρου $O(\log n)$ που την αποφασίζει. Έστω επίσης $L_2 \leq_{LS} L_1$ μέσω μιας συνάρτησης $R(x)$ η οποία υπολογίζεται σε λογαριθμικό χώρο από μια μηχανή Turing M_2 . Αν επιχειρήσουμε να αποφασίσουμε σε λογαριθμικό χώρο αν $x \in L_2$, εξετάζοντας κατά πόσο $R(x) \in L_2$ κατασκευάζοντας την «σύνθεση» M των δύο μηχανών M_2 και M_1 αντιμετωπίζουμε το ακόλουθο πρόβλημα: Το $R(x)$ μπορεί να είναι πολυωνυμικά μεγάλο στο $|x|$ μια και η μηχανή M_2 που θα το υπολογίσει μπορεί να εργαστεί για χρόνο $|x|^k$ όπου $k > 0$ μια σταθερά. Σε αυτή την περίπτωση η σύνθετη μηχανή, θα έχει την ταινία εξόδου της M_2 στην οποία είναι γραμμένο το $R(x)$ σαν ταινία εργασίας και άρα ο συνολικός χώρος των ταινιών εργασίας της δεν θα είναι λογαριθμικά φραγμένος. Κατασκευάζουμε λοιπόν την M ως ακολούθως. Η M_2 αρχίζει τον υπολογισμό με είσοδο το x και γράφει στην ταινία εξόδου της (η οποία είναι και ταυτόχρονα η ταινία εισόδου της M_1 και βέβαια ταινία εργασίας της M). Όταν ένα ψηφίο της $R(x)$ γραφεί στην έξοδο, διακόπτεται η λειτουργία της M_2 και αρχίζει η λειτουργία της M_1 . Η M_1 έχει σαν είσοδο το ψηφίο της $R(x)$. Αν ζητήσει ένα καινούργιο ψηφίο τότε διακόπτεται η λειτουργία της και επανερχόμαστε στην M_2 η οποία εργάζεται μέχρι να παράγει ένα δεύτερο ψηφίο το οποίο γράφει πάνω στο παλιό. Όσο η M_1 ζητά καινούργια ψηφία τότε η προσομοίωση συνεχίζεται εναλλάξ χρησιμοποιώντας μόνο ένα σύμβολο της ταινίας εξόδου της M_2 . Τι γίνεται όμως αν η M_1 ζητήσει ένα προηγούμενο ψηφίο; Για να αντιμετωπιστεί αυτό η M_2 κρατά σε μια βοηθητική ταινία τον αύξοντα αριθμό i του ψηφίου που γράφει στην έξοδο. Χρειάζονται $\log |x|^k = O(\log |x|)$ ψηφία για να καταγραφεί αυτός ο αριθμός. Όταν λοιπόν ζητηθεί το προηγούμενο ψηφίο $i - 1$, τότε η M_2 ξαναρχίζει από την αρχή την λειτουργία της μέχρι να παράγει το ψηφίο $i - 1$ στην έξοδο. Τότε επανερχόμαστε στην M_1 κ.ο.κ. Ο συνολικός χώρος είναι προφανές ότι είναι $O(\log n)$. \square

Η παραπάνω τεχνική χρησιμοποιείται και στην απόδειξη του παρακάτω πολύ χρήσιμου αποτελέσματος.

Θεώρημα 4.2 *Η σύνθεση δύο αναγωγών λογαριθμικού χώρου είναι αναγωγή λογαριθμικού χώρου.*

Απόδειξη: Έστω L_1 , L_2 και L_3 γλώσσες και έστω ότι $L_1 \leq_{LS} L_2$ και $L_2 \leq_{LS} L_3$. Τότε υπάρχουν συναρτήσεις R και R' υπολογίσιμες σε λογαριθμικό χώρο τέτοιες που $x \in L_1$ ανν $R(x) \in L_2$ και $y \in L_2$ ανν $R'(y) \in L_3$. Είναι προφανές συνεπώς ότι $x \in L_1$ ανν $R'(R(x)) \in L_3$. Η σύνθεση των δύο συναρτήσεων $R' \circ R$ μπορεί να υπολογιστεί επίσης σε λογαριθμικό χώρο χρησιμοποιώντας την παραπάνω τεχνική. \square

4.2 Πληρότητα

Όπως είπαμε και στην αρχή αυτού του κεφαλαίου η μελέτη της πολυπλοκότητας ενός προβλήματος απαιτεί αφενός την ένταξη ενός προβλήματος σε μία κλάση πολυπλοκότητας (αυτό βάζει στην ουσία ένα πάνω φράγμα στην πολυπλοκότητα του προβλήματος), αλλά και την απόδειξη ότι το πρόβλημα αυτό είναι από τα δυσκολότερα στην κλάση (αυτό είναι ένα κάτω φράγμα). Αυτή η τελευταία ιδιότητα μπορεί ναδειχθεί με την βοήθεια της αναγωγής (λογαριθμικού χώρου ή άλλου τύπου) που παρουσιάστηκε παραπάνω.

Ορισμός 4.2 Έστω C μία κλάση πολυπλοκότητας και έστω L μία γλώσσα. Η L είναι C -πλήρης (C -complete) αν:

1. Η L ανήκει στην C και
2. Κάθε γλώσσα στην κλάση C ανάγεται στην L .

Στον παραπάνω ορισμό οι αναγωγές που εμείς θα χρησιμοποιήσουμε είναι οι αναγωγές λογαριθμικού χώρου. Δεν είναι όμως και οι μοναδικές που χρησιμοποιούνται στην βιβλιογραφία. Υπάρχουν αρκετές διαφορετικού τύπου αναγωγές που κατηγοριοποιούν τα προβλήματα σε διαφορετικούς βαθμούς δυσκολίας [16, 2].

Ο παραπάνω ορισμός φαίνεται αρκετά απαιτητικός. Δεν είναι καν προφανές ότι μία κλάση πολυπλοκότητας έχει έστω και μία πλήρη γλώσσα. Και πραγματικά, υπάρχουν κλάσεις πολυπλοκότητας για τις οποίες είναι γνωστό ότι δεν υπάρχει πλήρης γλώσσα και άλλες για τις οποίες δεν γνωρίζουμε αν έχουν. Μπορούμε όμως εύκολα να δούμε ότι οι κλάσεις τις οποίες έχουμε αναφέρει στην Ενότητα 2.6 έχουν όλες πλήρη γλώσσα. Η γλώσσα αυτή για μία κλάση C που ορίζεται σαν το σύνολο των γλωσσών που γίνονται δεκτές από μια μηχανή Turing ντετερμινιστική (ή όχι) σε χρόνο (ή χώρο) $f(n)$ είναι η:

$$L_C = \{w_M \# x \mid \text{η μηχανή } M \text{ με τα συγκεκριμένα χαρακτηριστικά αποδέχεται το } x\};$$

όπου w_M είναι η συμβολοσειρά που κωδικοποιεί την M και το $\#$ ένα ειδικό σύμβολο για να χωρίζει τις δύο συμβολοσειρές. Όλες οι κλάσεις αυτές ορίζονται με βάση τα χαρακτηριστικά και την πολυπλοκότητα κάποιων μηχανών και για τον λόγο αυτό ονομάζονται *συντακτικές* κλάσεις. Οι κλάσεις P, NP, PSPACE κλπ. είναι όλες συντακτικές κλάσεις. Για μια συντακτική κλάση C λοιπόν, η γλώσσα L_C που ορίζεται όπως παραπάνω είναι C -πλήρης. Έστω L' μία οποιαδήποτε γλώσσα στο C και έστω M' η μηχανή η οποία την αποφασίζει. Προφανώς η M' έχει τα χαρακτηριστικά που απαιτεί η κλάση. Έστω x

μία συμβολοσειρά. Μπορούμε να αποφασίσουμε αν $x \in L'$ καταστρώνοντας το $y = w_M \# x$ και ελέγχοντας αν $y \in L_C$. Προφανώς $x \in L'$ αν $y \in L_C$.

Η πραγματική αξία της έννοιας της πληρότητας όμως δεν βρίσκεται σε τέτοιες τεχνητές πλήρεις γλώσσες. Πραγματικό κέρδος υπάρχει όταν κάποια πρακτική γλώσσα που αντιστοιχεί σε κάποιο φυσικό υπολογιστικό πρόβλημα αποδειχθεί πλήρης για κάποια κλάση. Πολλές φορές μάλιστα η αποδοχή μιας κλάσης σαν χρήσιμης για την κατηγοριοποίηση προβλημάτων γίνεται από την απόδειξη ότι γι' αυτή υπάρχει κάποιο πρακτικό πλήρες πρόβλημα. Θα δούμε παρακάτω τέτοια πρακτικά πλήρη προβλήματα κυρίως για την NP. Επί του παρόντος ας εξετάσουμε μερικές ιδιότητες των πλήρων προβλημάτων.

Λήμμα 4.1 Έστω C' κλάση πολυπλοκότητας κλειστή ως προς αναγωγές και έστω $C' \subseteq C$. Αν η C' -πλήρης γλώσσα L ανήκει στην C' , τότε $C' = C$.

Απόδειξη: Κάθε γλώσσα $L' \in C$, ανάγεται στην L . Επειδή η C' είναι κλειστή ως προς αναγωγές, $L' \in C'$. \square

Λήμμα 4.2 Αν δύο κλάσεις C και C' είναι κλειστές ως προς αναγωγές και η γλώσσα L είναι πλήρης και ως προς τις δύο, τότε $C = C'$.

Απόδειξη: Εφόσον η L είναι πλήρης ως προς την C τότε για οποιαδήποτε γλώσσα $L' \in C$, ισχύει $L' \leq_{LS} L \in C'$ και επειδή η C' είναι κλειστή ως προς αναγωγές $L' \in C'$ και $C \subseteq C'$. Με τον ίδιο τρόπο δείχνεται και ότι $C' \subseteq C$. \square

Τα παραπάνω δύο λήμματα δικαιολογούν τον χαρακτηρισμό των πλήρων προβλημάτων σαν των δυσκολότερων σε μία κλάση. Αν υποψιαζόμαστε ότι μια κλάση ταυτίζεται με μία μικρότερη της, αρκεί να δείξουμε ότι ένα πλήρες πρόβλημα της βρίσκεται στην μικρότερη κλάση.

Η εύρεση πλήρων προβλημάτων για μια κλάση μπορεί να διευκολυνθεί χρησιμοποιώντας την μεταβατικότητα των αναγωγών (Θεώρημα 4.2) αν ήδη γνωρίζουμε ένα πλήρες πρόβλημα για την κλάση.

Λήμμα 4.3 Έστω C μία κλάση και L ένα πλήρες πρόβλημα της C . Το πρόβλημα L' είναι C' -πλήρες αν:

1. Η L' ανήκει στην C και
2. Η L ανάγεται στην L' .

Απόδειξη: Προφανής από τον ορισμό της πληρότητας και το Θεώρημα 4.2. \square

Η απόδειξη ότι ένα πρόβλημα είναι πλήρες για μια κλάση συνήθως δείχνεται χρησιμοποιώντας το παραπάνω λήμμα. Χρειάζεται όμως για τον σκοπό αυτό η γνώση ενός τουλάχιστον πλήρους προβλήματος για την κλάση. Στην επόμενη ενότητα αποδεικνύουμε την πληρότητα τριών προβλημάτων ως προς ισάριθμες κλάσεις.

4.3 Μερικά πλήρη προβλήματα

Αρχίζουμε αποδεικνύοντας ένα κλασσικό θεώρημα της Θεωρίας Πολυπλοκότητας το οποίο λέγει ότι το πρόβλημα της ΙΚΑΝΟΠΟΙΗΣΙΜΟΤΗΤΑΣ, το γνωστό μας SAT είναι NP-πλήρες. Το θεώρημα αυτό αποδείχθηκε από τον S. Cook [4] και ανεξάρτητα από τον L. Levin [18]. Ήταν η αρχή στο ναδειχθεί 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 του R. Karp [13] όπου πολλά γνωστά προβλήματα δείχθηκαν NP-πλήρη. Από τότε εκατοντάδες προβλημάτων έχουνδειχθεί NP-πλήρη. Για μία εξαιρετική ανάπτυξη του θέματος ο αναγνώστης παραπέμπεται στο [6].

Θεώρημα 4.3 Η ΙΚΑΝΟΠΟΙΗΣΙΜΟΤΗΤΑ (SAT) είναι NP-πλήρες πρόβλημα.

Απόδειξη: Έχουμε ήδη δείξει στο Παράδειγμα 2.5 ότι το SAT ανήκει στο NP. Απομένει επομένως ναδειχθεί ότι αν L είναι μια οποιαδήποτε γλώσσα στο NP, η L ανάγεται σε λογαριθμικό χώρο στην γλώσσα L_{SAT} .

Εφόσον η $L \in \text{NP}$, υπάρχει μία μη ντετερμινιστική πολυωνυμική μηχανή Turing $M = (K, \Sigma, s, \Delta)$ η οποία αποφασίζει την L . Έστω $p(n)$ η πολυπλοκότητα χρόνου της M , όπου $p(n)$ είναι ένα πολώνυμο. Μπορούμε να υποθέσουμε (Λήμμα 3.1) ότι η M εργάζεται για $p(n)$ ακριβώς βήματα σε κάθε υπολογισμό με είσοδο w όπου $|w| = n$ και άρα ο χώρος που χρησιμοποιεί η M κατά την διάρκεια ενός υπολογισμού είναι το πολύ $p(n)$ (σε κάθε βήμα η M το πολύ να δεσμεύει και ένα ακόμη κύτταρο από την ταινία της). Κάνουμε ακόμη την παραδοχή, επίσης για λόγους ευκολίας, ότι η μηχανή έχει μετατραπεί έτσι ώστε κάθε συνδυασμός κατάστασης και συμβόλου να ορίζει τουλάχιστον μία μετάβαση. Η μετατροπή αυτή είναι πολύ εύκολη, απλώς ορίζουμε σαν επόμενη κατάσταση ενός μη ορισθέντος συνδυασμού την $q_{\text{ΟΧΙ}}$. Θα μας διευκολύνει στην απόδειξη μας, αν φανταστούμε ότι έχουμε ένα πίνακα διαστάσεων $p(n) \times p(n)$, όπου η γραμμή i του πίνακα είναι το τμήμα της ταινίας με τα πρώτα $p(n)$ χρησιμοποιούμενα κύτταρα, την χρονική στιγμή i . Αν μάλιστα σε κάθε γραμμή του πίνακα σημειώσουμε και την θέση της κεφαλής και την κατάσταση που βρισκόταν η M εκείνη την χρονική στιγμή, τότε ο πίνακας αυτός περιγράφει όλη την ιστορία του υπολογισμού της M με είσοδο την συμβολοσειρά w (αυτή μπορούμε να την δούμε στην γραμμή 1 του πίνακα) μέχρι το πέρας του υπολογισμού. Στην ουσία κάθε γραμμή του πίνακα είναι η διαμόρφωση της μηχανής την αντίστοιχη χρονική στιγμή. Στην τελευταία γραμμή του πίνακα μπορούμε, ελέγχοντας αν η τελευταία κατάσταση ήταν η $q_{\text{ΝΑΙ}}$, να αποφανθούμε αν η μηχανή M αποδέχθηκε ή όχι την συμβολοσειρά w στον συγκεκριμένο υπολογισμό (η M είναι μη ντετερμινιστική).

Τα διάφορα στοιχεία του πίνακα αυτού προφανώς δεν είναι άσχετα μεταξύ τους. Για παράδειγμα αν την χρονική στιγμή t η κεφαλή βρίσκεται στο κύτταρο j , τότε κατά την χρονική στιγμή $t+1$, τα κύτταρα στις θέσεις k όπου $k \geq j+1$ και $k \leq j-1$ θα πρέπει να έχουν το ίδιο σύμβολο που είχαν την στιγμή t . Επίσης το σύμβολο που υπάρχει στην θέση j την στιγμή $t+1$ δεν είναι τυχαίο: προέρχεται από την εγγραφή που έκανε η M και άρα προκύπτει από την σχέση μετάβασης της M . Στόχος μας είναι, εισάγοντας ένα αριθμό από λογικές μεταβλητές, να περιγράψουμε τον πίνακα αυτό καταστρώνοντας μια συνάρτηση Boole η οποία θα είναι ικανοποιήσιμη αν και μόνο αν ο πίνακας που περιγράφουν οι μεταβλητές μας είναι «νόμιμος» (είναι σύμφωνος δηλαδή με τους κανόνες της μηχανής μας) και επιπλέον στην τελευταία σειρά του πίνακα υπάρχει η κατάσταση q_{NAI} .

Προχωρούμε τώρα στην προσεκτική διατύπωση των απαραίτητων περιορισμών ώστε ένας πίνακας σαν τον παραπάνω να είναι νόμιμος.

1. Σε κάθε χρονική στιγμή (δηλαδή γραμμή του πίνακα) και σε κάθε θέση στη ταινία, υπάρχει γραμμένο ακριβώς ένα σύμβολο.
2. Σε κάθε χρονική στιγμή η κεφαλή βρίσκεται σε ακριβώς ένα κύτταρο.
3. Σε κάθε χρονική στιγμή η μηχανή βρίσκεται σε ακριβώς μία κατάσταση
4. Δύο διαδοχικές γραμμές του πίνακα διαφέρουν κατά ένα το πολύ σύμβολο, αυτό που στην πρώτη γραμμή ήταν κάτω από την κεφαλή.
5. Οι καταστάσεις, οι θέσεις της κεφαλής και τα σύμβολα της ταινίας σε δύο διαδοχικές χρονικές στιγμές πρέπει να επιτρέπονται από τους κανόνες της M .
6. Στην αρχική γραμμή υπάρχει η συμβολοσειρά εισόδου και η κατάσταση είναι η s (αρχική).
7. Η κατάσταση την τελευταία χρονική στιγμή είναι η q_{NAI} .

Είναι εύκολο να δούμε ότι οι παραπάνω 7 περιορισμοί πράγματι εξασφαλίζουν την μορφή του πίνακα που επιθυμούμε. Οι περιορισμοί 1 – 4 εξασφαλίζουν την μορφή που πρέπει να έχει κάθε πίνακας με διαμορφώσεις οποιασδήποτε μηχανής Turing. Ο περιορισμός 5 επιβάλλει οι διαδοχικές γραμμές του πίνακα να είναι διαδοχικές διαμορφώσεις της συγκεκριμένης μηχανής M , ενώ ο περιορισμός 6 ότι η είσοδος της M είναι η w και αρχική κατάσταση η s . Τέλος ο 7 ότι η M αποδέχεται την w .

Εισάγουμε τέσσερεις κατηγορίες μεταβλητών ανάλογα με τον ρόλο που τις προορίζουμε να παίξουν. (Δίνουμε τις μεταβλητές με δείκτες ώστε να τονίσουμε τον ρόλο της κάθε μιας, στην προσπάθεια περιγραφής του πίνακα.)

Μεταβλητές $x_{tj\sigma}$. Η μεταβλητή $x_{tj\sigma}$ θα είναι αληθής ανν την χρονική στιγμή t , στην θέση j της ταινίας υπάρχει γραμμένο το σύμβολο σ . Εδώ $1 \leq t \leq p(n)$, $1 \leq j \leq p(n)$ και το σ μπορεί να είναι οποιοδήποτε σύμβολο του αλφαβήτου Σ της M . Συνεπώς αυτή η κατηγορία περιλαμβάνει $p(n)^2|\Sigma|$ μεταβλητές.

Μεταβλητές y_{tq} . Η μεταβλητή y_{tq} θα είναι αληθής ανν την χρονική στιγμή t , η μηχανή βρίσκεται στην κατάσταση q . Εδώ $1 \leq t \leq p(n)$ και το q μπορεί να είναι οποιαδήποτε κατάσταση της M . Συνεπώς αυτή η κατηγορία περιλαμβάνει $p(n)|K|$ μεταβλητές.

Μεταβλητές z_{tj} . Η μεταβλητή z_{tj} θα είναι αληθής ανν την χρονική στιγμή t η κεφαλή της μηχανής βρίσκεται στην θέση j . Εδώ $1 \leq t \leq p(n)$ και $1 \leq j \leq p(n)$. Η κατηγορία αυτή περιλαμβάνει $p(n)^2$ μεταβλητές.

Μεταβλητές c_{tk} . Η μεταβλητή c_{tk} είναι αληθής ανν την χρονική στιγμή t , η μηχανή κάνει την υπ' αριθμό k μη ντετερμινιστική επιλογή. Το k είναι ένας ακέραιος στο διάστημα 1 έως το μέγιστο αριθμό μη ντετερμινιστικών επιλογών που μπορεί να έχει η μηχανή σε οποιαδήποτε στιγμή. Επειδή αυτές είναι σταθερές (ανεξάρτητες του n), υπάρχουν $O(p(n))$ τέτοιες μεταβλητές.

Προχωρούμε τώρα στην διατύπωση λογικών εκφράσεων Boole, μία για κάθε ένα περιορισμό που όταν επαληθεύονται τότε ικανοποιείται και ο αντίστοιχος περιορισμός. Στις παρακάτω σχέσεις οι δείκτες που εμφανίζονται κάτω από τους λογικούς τελεστές \wedge και \vee κινούνται σε όλο το επιτρεπόμενο πεδίο τους εκτός αν σημειώνεται διαφορετικά. Δηλαδή για τα t και j έχουμε $1 \leq t, j \leq p(n)$ για τα $\sigma, \sigma' \in \Sigma$ και για τα $q, q' \in K$. Τα \mathbf{x} , \mathbf{y} , \mathbf{z} και \mathbf{c} είναι διανύσματα των μεταβλητών $x_{tj\sigma}, y_{tq}, z_{tj}$ και c_{tk} αντίστοιχα.

Περιορισμός 1

$$f_1(\mathbf{x}) = \bigwedge_{\substack{t,j \\ \sigma \neq \sigma'}} (\neg x_{tj\sigma} \vee \neg x_{tj\sigma'})$$

Είναι εύκολο να δούμε ότι αν η παραπάνω έκφραση ικανοποιείται, τότε ικανοποιείται και ο Περιορισμός 1. Οι επιμέρους προτάσεις λένουν ότι για οποιαδήποτε χρονική στιγμή t και οποιοδήποτε κύτταρο j δεν μπορεί να είναι οι λογικές μεταβλητές δύο διαφορετικών συμβόλων και οι δύο αληθείς. Το πολύ ένα σύμβολο μπορεί να περιέχεται σε κάθε κύτταρο οποιαδήποτε στιγμή. Το ότι ακριβώς ένα σύμβολο περιέχεται σε κάθε κύτταρο εξασφαλίζεται σε συνδυασμό με τον περιορισμό 5.

Περιορισμός 2

$$f_2(\mathbf{z}) = \bigwedge_{\substack{t \\ j \neq j'}} (\neg z_{tj} \vee \neg z_{tj'})$$

Με το ίδιο σκεπτικό όπως και παραπάνω η έκφραση f_2 επιβάλλει η κεφαλή να βρίσκεται σε ακριβώς ένα κύτταρο.

Περιορισμός 3

$$f_3(\mathbf{y}) = \bigwedge_{\substack{t \\ q \neq q'}} (\neg y_{tq} \vee \neg y_{tq'})$$

Ομοίως όπως παραπάνω.

Περιορισμός 4

$$f_4(\mathbf{x}, \mathbf{z}) = \bigwedge_{\substack{1 \leq t \leq p(n)-1 \\ j \neq j'}} (\neg x_{tj\sigma} \vee \neg z_{tj'} \vee x_{t+1j\sigma})$$

Στην παραπάνω σχέση κάθε παρένθεση επιβάλλει ότι αν $x_{tj\sigma} = z_{tj'} = \text{true}$, τότε $x_{t+1j\sigma} = \text{true}$. Με άλλα λόγια, σε κάθε χρονική στιγμή αν η κεφαλή βρίσκεται στην θέση j' , τότε την επόμενη χρονική στιγμή, σε οποιοδήποτε άλλο κύτταρο το σύμβολο που ήταν γραμμένο εκεί παραμένει το ίδιο. Σε συνδυασμό με τον περιορισμό 2 ο οποίος επιβάλλει η κεφαλή να είναι σε μία μόνο θέση έχουμε το ζητούμενο.

Περιορισμός 5

Έστω ότι για συγκεκριμένη κατάσταση q και σύμβολο σ , οι επιτρεπόμενες μεταβάσεις είναι οι $((q, \sigma), (q_k, \sigma_k, D_k)) \in \Delta$, όπου $1 \leq k \leq |((q, \sigma), (q_k, \sigma_k, D_k))|$. Στην παρακάτω σχέση το d_k είναι 1, -1 και 0, αν το D_k της μετάβασης είναι αντίστοιχα \rightarrow, \leftarrow και $-$.

$$f_5(\mathbf{x}, \mathbf{y}, \mathbf{z}, \mathbf{c}) = \bigwedge_{\substack{1 \leq t \leq p(n)-1 \\ q, \sigma, j}} \left[(\neg z_{tj} \vee \neg y_{tq} \vee \neg x_{tj\sigma} \vee \bigvee_k c_{tk}) \wedge \right. \\ \left. \bigwedge_{\substack{k \text{ έτσι ώστε} \\ ((q, \sigma), (q_k, \sigma_k, D_k)) \in \Delta}} (\neg c_{tk} \vee y_{t+1q_k}) \wedge (\neg c_{tk} \vee x_{t+1j\sigma_k}) \wedge (\neg c_{tk} \vee z_{t+1j+d_k}) \right].$$

Η παραπάνω σχέση επιβάλλει όταν $z_{tj} = y_{tq} = x_{tj\sigma} = \text{true}$, τότε τουλάχιστον μία από τις μεταβλητές c_{tk} πρέπει να είναι αληθής. Δηλαδή μία τουλάχιστον μη ντετερμινιστική μετάβαση πρέπει να γίνει στο βήμα t . Οι επόμενες τρεις προτάσεις ορίζονται με βάση τις μη ντετερμινιστικές μεταβάσεις $((q, \sigma), (q_k, \sigma_k, D_k))$ για συγκεκριμένα q και σ . Έτσι αν η μη ντετερμινιστική μετάβαση είναι η k , ($c_{tk} = \text{true}$), τότε θα πρέπει να είναι αληθείς και οι μεταβλητές y_{t+1q_k} , $x_{t+1j\sigma_k}$ και z_{t+1j+d_k} . Με άλλα λόγια την επόμενη χρονική στιγμή η επόμενη κατάσταση, το καινούργιο σύμβολο και η τυχούσα κίνηση της κεφαλής, είναι όπως καθορίζονται από την μη ντετερμινιστική μετάβαση k . Σημειώνουμε ότι επειδή από τους προηγούμενους περιορισμούς η μηχανή μπορεί να βρίσκεται το πολύ

σε μία κατάσταση και θέση της κεφαλής ενώ το πολύ ένα μόνο σύμβολο βρίσκεται σε κάθε κύτταρο, μόνο μία μη ντετερμινιστική μετάβαση είναι δυνατή κάθε φορά.

Περιορισμός 6

Συμβολίζουμε με w_j το j -οστό σύμβολο της συμβολοσειράς w .

$$f_6(\mathbf{x}, \mathbf{y}, \mathbf{z}) = x_{11} \triangleright \bigwedge_{j=2}^{n+1} x_{1j} w_j \bigwedge_{j=n+2}^{p(n)} x_{1j} b \wedge y_{1s} \wedge z_{11}$$

Η συνάρτηση αυτή επιβάλλει την χρονική στιγμή 1 το πρώτο σύμβολο της ταινίας να είναι το αρχικό σύμβολο \triangleright , στην συνέχεια και από την θέση 2 έως $n+1$ βρίσκεται η συμβολοσειρά εισόδου w , ενώ όλα τα υπόλοιπα σύμβολα έχουν το κενό b . Επίσης η κατάσταση την στιγμή 1 είναι η αρχική s και η κεφαλή βρίσκεται στην θέση 1.

Περιορισμός 7

$$f_7(\mathbf{y}) = y_{p(n)q_{\text{NAI}}}$$

Η συνολική έκφραση επομένως είναι η σύζευξη των παραπάνω επτά προτάσεων.

$$\varphi(\mathbf{x}, \mathbf{y}, \mathbf{z}, \mathbf{c}) = f_1(\mathbf{x}) \wedge f_2(\mathbf{z}) \wedge f_3(\mathbf{y}) \wedge f_4(\mathbf{x}, \mathbf{z}) \wedge f_5(\mathbf{x}, \mathbf{y}, \mathbf{z}, \mathbf{c}) \wedge f_6(\mathbf{x}, \mathbf{y}, \mathbf{z}) \wedge f_7(\mathbf{y})$$

Ας παρατηρήσουμε τώρα ότι η κατασκευή της φ , μπορεί να γίνει σε λογαριθμικό χώρο. Αυτό είναι εύκολο ναδειχθεί μια και όπως αναφέρθηκε παραπάνω ο συνολικός αριθμός των μεταβλητών που χρησιμοποιεί η φ είναι πολυωνυμικός στο n . Από το γεγονός αυτό και την μορφή των επιμέρους επτά προτάσεις, έπεται ότι το μήκος τους είναι επίσης πολυωνυμικό. Μπορούμε συνεπώς να κατασκευάσουμε την φ σε λογαριθμικό χώρο με ένα σταθερό αριθμό από βοηθητικές μεταβλητές που σε κάθε μία καταχωρείται ένας αριθμός στο διάστημα 1 έως $p(n)$. Χρειάζονται $\log n$ ψηφία γι' αυτό, συνεπώς ο χώρος είναι $O(\log n)$.

Απομένει τώρα ναδειχθεί ότι η μηχανή Turing M αποδέχεται την συμβολοσειρά w αν και μόνον αν η φ είναι ικανοποιήσιμη.

Μόνο αν. Έστω ότι η $\varphi(\mathbf{x}, \mathbf{y}, \mathbf{z}, \mathbf{c})$ είναι ικανοποιήσιμη. Αυτό σημαίνει ότι υπάρχει αποτίμηση στις μεταβλητές της που κάνει και τις επτά συναρτήσεις $f_i, i = 1, \dots, 7$ αληθείς. Όμως από τον τρόπο κατασκευής των συναρτήσεων, η ταυτόχρονη ικανοποίηση τους ικανοποιεί και τους 7 περιορισμούς που αναφέρθηκαν. Ειδικότερα, οι τέσσερις πρώτες συναρτήσεις επιβάλλουν σε κάθε χρονική στιγμή το πολύ μία τιμή στις μεταβλητές που παριστάνουν το περιεχόμενο κάθε κυττάρου, την κατάσταση της μηχανής και την θέση της κεφαλής. Όμως λόγω της f_5 και της παραδοχής που κάναμε ότι κάθε συνδυασμός κατάστασης και συμβόλου ορίζεται στην σχέση μετάβασης, συνάγεται ότι κάθε χρονική στιγμή υπάρχει τουλάχιστον ένα σύμβολο σε κάθε κύτταρο, η μηχανή

βρίσκεται σε τουλάχιστον μία κατάσταση και η κεφαλή σε μία θέση. Συνολικά δηλαδή πληρούνται και οι 7 περιορισμοί που τέθηκαν. Οι τιμές λοιπόν των μεταβλητών σε μια ικανοποιούσα αποτίμηση προσδιορίζουν μία ακολουθία διαμορφώσεων της M στην οποία πρώτη είναι η αρχική διαμόρφωση με είσοδο το w , η τελική είναι διαμόρφωση αποδοχής κάθε μία δε προκύπτει από την προηγούμενη σύμφωνα με τους κανόνες μετάβασης της M , πρόκειται δηλαδή για υπολογισμό αποδοχής της w .

Αν. Αντίστροφα, έστω ότι η M αποδέχεται την w . Υπάρχει κατά συνέπεια ακολουθία διαμορφώσεων της M που από την αρχική διαμόρφωση καταλήγει σε διαμόρφωση αποδοχής. Η αποτίμηση η οποία σε κάθε μεταβλητή δίδει τιμή σύμφωνα με τις διαμορφώσεις αυτές, προφανώς ικανοποιεί την φ . \square

Στο παραπάνω θεώρημα έχουμε δείξει και κάτι ισχυρότερο από την NP-πληρότητα του SAT. Αν παρατηρήσουμε την συντακτική μορφή του στιγμιότυπου του SAT που κατασκευάσαμε, θα δούμε ότι η συνάρτηση φ είναι συζεύξεις διαζεύξεων. Οι διαζεύξεις λέγονται και προτάσεις (clauses). Η φ συνεπώς επιβάλλει την ταυτόχρονη ικανοποίηση όλων των προτάσεων από τις οποίες απαρτίζεται. Όταν ένα στιγμιότυπο του SAT είναι στην μορφή αυτή λέμε ότι είναι σε κανονική συζευκτική μορφή (conjunctive normal form ή CNF). Μάλιστα τις περισσότερες φορές σαν SAT αναφέρεται στην βιβλιογραφία η κανονική συζευκτική μορφή. Στην ουσία δηλαδή δείξαμε:

Πόρισμα 4.1 Το SAT είναι NP-πλήρες ακόμα και σε κανονική συζευκτική μορφή.

Απόδειξη: Άμεσα από τη απόδειξη του Θεωρήματος 4.3. \square

Η βιβλιογραφία για το SAT περιλαμβάνει πολλές επιπλέον ειδικές μορφές του οι οποίες παραμένουν NP-πλήρεις. Θα εξετάσουμε αρκετές από αυτές στο επόμενο κεφάλαιο.

Παρόλο που το Θεώρημα 4.3 δείχνει ότι το SAT είναι στην γενικότητα του και στην κανονική συζευκτική μορφή NP-πλήρες, άλλες μορφές του είναι πολυωνυμικές. Μία μάλιστα από αυτές έχει ιδιαίτερη σημασία διότι είναι πλήρης αυτή τη φορά για την κλάση P.

Ορισμός 4.3 Ονομάζουμε μία λογική πρόταση Horn αν είναι διάζευξη λογικών μεταβλητών το πολύ μία από τις οποίες είναι χωρίς άρνηση. Μία λογική έκφραση είναι Horn αν είναι σύζευξη Horn προτάσεων. Τέλος HORNSAT ονομάζουμε το πρόβλημα απόφασης στο οποίο δίδεται μία Horn έκφραση και ζητείται αν είναι ικανοποιήσιμη.

Παράδειγμα 4.1 Η παρακάτω λογική έκφραση είναι Horn:

$$(\neg x_1 \vee \neg x_2 \vee \neg x_4 \vee x_5) \wedge (x_2) \wedge (\neg x_2 \vee \neg x_3 \vee \neg x_4) \wedge (\neg x_1 \vee \neg x_3 \vee x_5) \wedge (x_3)$$

Οι προτάσεις Horn πολλές φορές ονομάζονται και *συνεπαγωγές* (implications). Αυτό δικαιολογείται διότι μια πρόταση Horn στην ουσία λέγει ότι η ταυτόχρονη αλήθεια κάποιων μεταβλητών (αυτών που εμφανίζονται με άρνηση) συνεπάγεται και την αλήθεια της θετικής μεταβλητής. Για παράδειγμα η πρώτη πρόταση του παραδείγματος είναι ισοδύναμη με την λογική συνεπαγωγή: $x_1 \wedge x_2 \wedge x_4 \longrightarrow x_5$.

Θεώρημα 4.4 *Το HORNSAT είναι P-πλήρες.*

Απόδειξη: Δείχνουμε κατ' αρχή ότι το HORNSAT ανήκει στο P. Ας παρατηρήσουμε ότι οι μόνες προτάσεις που μπορεί να επιβάλουν σε κάποια μεταβλητή να είναι αληθείς είναι οι συνεπαγωγές, οι πλήρως αρνητικές προτάσεις ικανοποιούνται μόνο με ψευδείς μεταβλητές. Στην συνέχεια εκτελούμε συνεχώς το παρακάτω βήμα: συντηρούμε συνεχώς ένα σύνολο μεταβλητών T οι οποίες πρέπει να είναι κατ' ανάγκη αληθείς. Αρχίζουμε λοιπόν από τις προτάσεις που είναι μόνο μία θετική μεταβλητή όπως οι (x_2) και (x_3) στο παράδειγμα. Αυτές προφανώς πρέπει να γίνουν κατ' ανάγκη αληθείς. Αν δεν υπάρχουν, τότε η έκφραση είναι προφανώς ικανοποιήσιμη κάνοντας όλες τις μεταβλητές ψευδείς. Εξετάζουμε διαδοχικά τις συνεπαγωγές και αν οι αρνητικές τους μεταβλητές είναι αληθείς τότε προσθέτουμε και την θετική στο T . Στο παράδειγμα, η συνεπαγωγή $(x_2 \wedge x_3 \longrightarrow x_5)$ επιβάλει στην x_5 να γίνει αληθής. Επειδή κάθε φορά μία τουλάχιστον μεταβλητή προστίθεται στο T , αυτή η διαδικασία θα τελειώσει σε χρόνο $O(nm)$ όπου n ο αριθμός των μεταβλητών και m των προτάσεων. Από τον τρόπο κατασκευής του T όλες οι μεταβλητές σε αυτό πρέπει οπωσδήποτε να είναι αληθείς για να ικανοποιηθεί η έκφραση. Αν λοιπόν κάποια πλήρως αρνητική έκφραση δεν μπορεί να ικανοποιηθεί επειδή οι μεταβλητές της είναι όλες στο T , τότε και η έκφραση είναι μη ικανοποιήσιμη. Αλλιώς δίνουμε την τιμή ψευδής στις υπόλοιπες μεταβλητές και έχουμε την συνολική αποτίμηση. Ο συνολικός χρόνος είναι προφανώς $O(nm)$.

Για να δείξουμε ότι το πρόβλημα είναι πλήρες για το P πρέπει να ανάγουμε κάθε γλώσσα στο P στο HORNSAT. Έστω λοιπόν μια γλώσσα L η οποία αποφασίζεται από μια ντετερμινιστική πολυωνυμική μηχανή Turing M . Θα πρέπει δοθείσης της M και μιας συμβολοσειράς εισόδου w να κατασκευάσουμε μία Horn έκφραση φ έτσι ώστε η M να αποδέχεται το w αν η φ είναι ικανοποιήσιμη. Η κατασκευή είναι πανομοιότυπη με την κατασκευή που έγινε στο Θεώρημα 4.3. Για παράδειγμα οι 7 περιορισμοί που τέθηκαν για την ορθή περιγραφή του υπολογισμού της M , ασφαλώς ισχύουν και τώρα. Το ίδιο και οι επιμέρους συναρτήσεις που κατασκευάστηκαν για να τους επιβάλουν. Μία μόνο διαφορά υπάρχει: η μηχανή M τώρα είναι ντετερμινιστική και συνεπώς μόνο μία μετάβαση είναι δυνατή κάθε φορά. Αν παρατηρήσουμε όμως και τις 7 συναρτήσεις που κατασκευάσαμε στην απόδειξη του Θεωρήματος 4.3, είναι όλες σε μορφή Horn εκτός από την πρώτη πρόταση της συνάρτησης f_5 η οποία

ακριβώς χρησιμεύει για να περιγράψει τις πολλαπλές μεταβάσεις της μη ντετερμινιστικής μηχανής! Συνεπώς στην περίπτωση μας οι μη αρνητικές μεταβλητές είναι μόνο μία και άρα και η f_5 είναι Horn πρόταση. Η απόδειξη είναι πλήρης. \square

Προχωρούμε τώρα στην εύρεση πλήρους προβλήματος για την κλάση NL. Το πρόβλημα που θα παρουσιάσουμε είναι μία ακόμη παραλλαγή του SAT!

Ορισμός 4.4 Μια k CNF έκφραση είναι μια έκφραση Boole σε κανονική συζευκτική μορφή στην οποία κάθε πρόταση έχει το πολύ k μεταβλητές. Το πρόβλημα k SAT είναι το πρόβλημα στο οποίο ζητείται αν μία k CNF έκφραση είναι ικανοποιήσιμη.

Όπως θα δούμε στο επόμενο κεφάλαιο το k SAT είναι NP-πλήρες για οποιοδήποτε $k \geq 3$. Όταν $k = 2$ όμως το πρόβλημα γίνεται πολυωνυμικό και μάλιστα είναι πλήρες για την NL (η οποία γνωρίζουμε ότι περιέχεται στην P). Μια 2CNF πρόταση $(x \vee y)$ είναι ισοδύναμη με τις δύο συνεπαγωγές: $(\neg x \longrightarrow y)$ και $(\neg y \longrightarrow x)$. Υπάρχει δηλαδή και εδώ όπως στο HORNSAT η μετάδοση μιας λογικής τιμής σε μία μόνο μεταβλητή. Το γεγονός αυτό, η έλλειψη δηλαδή πολλαπλών επιλογών, είναι αυτό που διαισθητικά κάνει το 2SAT ευκολότερο. Οι συνεπαγωγές αυτές τώρα θυμίζουν τις ακμές ενός κατευθυντικού γραφήματος. Δοθείσης λοιπόν μίας 2CNF συνάρτησης $\varphi(x_1, x_2, \dots, x_n)$ κατασκευάζουμε το ακόλουθο κατευθυντικό γράφημα G_φ : Το G_φ έχει $2n$ κόμβους ένα για κάθε μεταβλητή της φ και ένα για την άρνηση της. Υπάρχει ακμή από τον κόμβο x_i στον κόμβο x_j ($i \neq j$) αν η 2CNF πρόταση $(\neg x_i \vee x_j)$ είναι πρόταση της φ . Ας παρατηρήσουμε μάλιστα ότι αν υπάρχει η $(x_i \longrightarrow x_j)$ τότε υπάρχει κατ' ανάγκη και η $(\neg x_j \longrightarrow \neg x_i)$.

Θεώρημα 4.5 Η φ είναι ικανοποιήσιμη αν και μόνο αν δεν υπάρχει μονοπάτι στον G_φ από ένα κόμβο x στον κόμβο $\neg x$.

Απόδειξη: Ας παρατηρήσουμε κατ' αρχή ότι αν μια ικανοποιούσα την φ αποτίμηση κάνει την x αληθή, τότε κάθε προσημασμένη μεταβλητή κατά μήκος ενός μονοπατιού με αρχή το x γίνεται αληθής. Αντίστροφα αν ένα μονοπάτι καταλήγει σε ψευδή μεταβλητή, τότε κάθε μεταβλητή σε κατεύθυνση αντίθετη της φοράς του μονοπατιού γίνεται επίσης ψευδής. Μπορούμε δηλαδή να πούμε ότι η αλήθεια μεταδίδεται με την φορά του μονοπατιού ενώ το ψέμα αντίστροφα. Αν επομένως η φ είναι ικανοποιήσιμη και το x είναι αληθές, τότε δεν υπάρχει μονοπάτι από τον x στον $\neg x$. Αν πάλι το x είναι ψευδές τότε δεν υπάρχει μονοπάτι από τον $\neg x$ στον x . Επειδή όμως από την κατασκευή του G_φ , ένα μονοπάτι έχει και το συμμετρικό του, έχουμε το ζητούμενο.

Αντίστροφα, αν δεν υπάρχει μονοπάτι από ένα κόμβο x στον $\neg x$ τότε δεν υπάρχει και από τον $\neg x$ στον x . Θα δείξουμε ότι η φ είναι ικανοποιήσιμη

κατασκευάζοντας μία λογική αποτίμηση που την ικανοποιεί. Ξεκινάμε από μία αυθαίρετη μεταβλητή x στην οποία δίνουμε τιμή αλήθεια και ταυτόχρονα βέβαια στην άρνηση της ψέμα. Από αυτές τις δύο τιμές, όλες οι μεταβλητές στις οποίες καταλήγουν μονοπάτια από το x γίνονται αληθείς και όλες οι οποίες είναι αρχή μονοπατιών προς το $\neg x$, ψευδείς. Δεν υπάρχει περίπτωση αντίφασης στις τιμές αυτών των μεταβλητών γιατί δεν υπάρχει στον G_φ μονοπάτι από μια μεταβλητή στην άρνηση της. Συνεχίζουμε αυτή την διαδικασία επιλέγοντας αυθαίρετα μεταβλητές χωρίς τιμή μέχρι όλες οι μεταβλητές να αποκτήσουν τιμή. Η αποτίμηση που κατασκευάστηκε έτσι ικανοποιεί την φ γιατί σε κάθε ακμή $(x_i \rightarrow x_j)$ (που αντιστοιχεί βέβαια στην πρόταση $(\neg x_i \vee x_j)$), είτε η x_i είναι ψευδής, είτε η x_j αληθής ή και τα δύο, συνεπώς η φ είναι ικανοποιησιμη.

□

Το παραπάνω θεώρημα στην ουσία αποτελεί και μία αναγωγή του 2SAT στο πρόβλημα της ΜΗ ΠΡΟΣΠΕΛΑΣΙΜΟΤΗΤΑΣ (NON-REACHABILITY), πρόβλημα που δείξαμε στο Λήμμα 3.4 ότι λύνεται σε μη ντετερμινιστικό λογαριθμικό χώρο. Δίνει επίσης και ένα ταχύ πολυωνυμικό αλγόριθμο για την επίλυση του 2SAT: Καταστρώνουμε το G_φ και στην συνέχεια με αναζήτηση κατά βάθος ελέγχουμε αν υπάρχει ή όχι μονοπάτι από ένα κόμβο στον αντίθετο του. Η διαδικασία αυτή μπορεί να γίνει σε γραμμικό χρόνο $O(n)$ χωρίζοντας το γράφημα στις ισχυρά συνεκτικές συνιστώσες του [5].

Η σχέση όμως του 2SAT με την ΜΗ ΠΡΟΣΠΕΛΑΣΙΜΟΤΗΤΑ είναι ανάλογη και προς την αντίθετη κατεύθυνση. Η ΜΗ ΠΡΟΣΠΕΛΑΣΙΜΟΤΗΤΑ ανάγεται στο 2SAT. Ισχύει δηλαδή το θεώρημα:

Θεώρημα 4.6 *Η ΜΗ ΠΡΟΣΠΕΛΑΣΙΜΟΤΗΤΑ και το 2SAT είναι προβλήματα αμοιβαία αναγώμενα το ένα στο άλλο με αναγωγές λογαριθμικού χώρου.*

Απόδειξη: Η μία κατεύθυνση είναι στην ουσία το Θεώρημα 4.5 με την επιπλέον παρατήρηση ότι η κατασκευή του G_φ γίνεται σε λογαριθμικό χώρο.

Για την αντίστροφη κατεύθυνση, έστω ένα στιγμιότυπο της ΜΗ ΠΡΟΣΠΕΛΑΣΙΜΟΤΗΤΑΣ, δηλαδή γράφημα G και δύο διακεκριμένοι κόμβοι s και t . Εισάγουμε μία λογική μεταβλητή για κάθε κόμβο του γραφήματος. Στην συνέχεια για κάθε ακμή (x, y) του G εισάγουμε την πρόταση $(\neg x \vee y)$. Τέλος εισάγουμε και τις προτάσεις (s) και $(\neg t)$. Είναι εύκολο να δούμε ότι η συνάρτηση φ που έχει τις προτάσεις που κατασκευάσαμε είναι ικανοποιήσιμη αν και μόνο αν δεν υπάρχει μονοπάτι στο γράφημα G από τον s στον t . □

Πόρισμα 4.2 *Το 2SAT είναι NL-πλήρες.*

Απόδειξη: Το Θεώρημα 4.5 μας επιτρέπει να ανάγουμε το 2SAT στο πρόβλημα της ΜΗ ΠΡΟΣΠΕΛΑΣΙΜΟΤΗΤΑΣ το οποίο γνωρίζουμε από το Λήμμα 3.3 ότι λύνεται σε μη ντετερμινιστικό χώρο $O(\log n)$. Άρα το 2SAT ανήκει στο NL.

Ως προς την πληρότητα είναι ευκολότερο να δείξουμε ότι η ΠΡΟΣΠΕΛΑΣΙΜΟΤΗΤΑ είναι NL-πλήρης. Από το Θεώρημα Immerman-Szelepcsényi (Θεώρημα 3.8) γνωρίζουμε ότι $NL = co-NL$ και συνεπώς αυτό θα συνεπάγεται ότι και η ΜΗ ΠΡΟΣΠΕΛΑΣΙΜΟΤΗΤΑ και (από το Θεώρημα 4.6) το 2SAT θα είναι NL-πλήρη. Ας θεωρήσουμε λοιπόν μία γλώσσα L στο NL η οποία αποφασίζεται από μία μη ντετερμινιστική μηχανή Turing M σε χώρο $O(\log n)$. Μπορούμε εύκολα να μετατρέψουμε το γράφημα των διαμορφώσεων της M έτσι ώστε να έχει μόνο μία διαμόρφωση αποδοχής (αυτό γίνεται αλλάζοντας όλες τις μεταβάσεις που οδηγούν στην q_{NAI} με μεταβάσεις σε μία ενδιάμεση κατάσταση q_0 και μία μετάβαση από την q_0 στην q_{NAI}). Συνεπώς η αποδοχή της συμβολοσειράς εισόδου είναι ισοδύναμη με την ύπαρξη μονοπατιού από την αρχική διαμόρφωση I_s στην μοναδική διαμόρφωση αποδοχής I_f , ένα πρόβλημα ΠΡΟΣΠΕΛΑΣΙΜΟΤΗΤΑΣ. Όλος ο χειρισμός του γραφήματος γίνεται έμμεσα όπως στο Θεώρημα Savitch, δηλαδή σε χώρο λογαριθμικό. \square

Κεφάλαιο 5

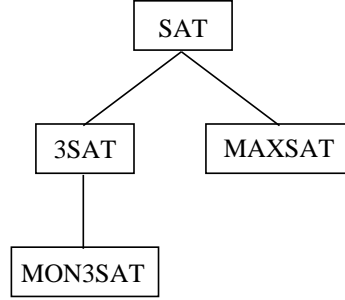
NP-πλήρη προβλήματα

Στο κεφάλαιο αυτό θα παρουσιάσουμε ένα αριθμό υπολογιστικών προβλημάτων τα οποία θα δείξουμε ότι είναι NP-πλήρη. Τα περισσότερα από αυτά ήταν γνωστά προβλήματα σε πολλές περιοχές των Διακριτών Μαθηματικών και της Επιχειρησιακής Έρευνας, πριν από την διατύπωση της θεωρίας της NP-πληρότητας. Από το κεφάλαιο αυτό θα δικαιολογηθεί ο ισχυρισμός που διατυπώσαμε στην Ενότητα 2.6 για την ύπαρξη στην κλάση NP πολλών χρήσιμων και πρακτικών προβλημάτων.

5.1 Παραλλαγές της ΙΚΑΝΟΠΟΙΗΣΙΜΟΤΗΤΑΣ

Έχουμε ήδη δει και ασχοληθεί με το πρόβλημα της ΙΚΑΝΟΠΟΙΗΣΙΜΟΤΗΤΑΣ SAT σε διάφορα σημεία των προηγούμενων κεφαλαίων. Η σημασία του για την Θεωρία Πολυπλοκότητας πρέπει να είναι πλέον φανερή. Στην ενότητα αυτή θα τονίσουμε ακόμη περισσότερο την σημασία αυτή, κυρίως αυτή τη φορά, σαν εργαλείο και αφετηρία για την απόδειξη νέων NP-πλήρων προβλημάτων. Θα ασχοληθούμε με ακόμη τρεις διαφορετικές εκδόσεις του προβλήματος της ικανοποιησιμότητας λογικών εκφράσεων, το 3SAT, το MON3SAT, και το MAXSAT. Όπως θα δούμε στη συνέχεια, τα προβλήματα αυτά κληρονομούν την δυσκολία του SAT, είναι και αυτά NP-πλήρη. Στο Σχήμα 5.1 παρουσιάζουμε τις αναγωγές που θα ακολουθήσουν. Θα είμαστε πού αναλυτικοί και τυπικοί στις αποδείξεις, τουλάχιστον στα πρώτα προβλήματα, ώστε να γίνει όσο το δυνατό πιο κατανοητή η μεθοδολογία τέτοιων αποδείξεων.

Το πρόβλημα της 3ΙΚΑΝΟΠΟΙΗΣΙΜΟΤΗΤΑΣ (3SATISFIABILITY, 3SAT) είναι μια ειδική περίπτωση του προβλήματος SAT όπου οι προτάσεις κάθε στιγμιότυπου έχουν ακριβώς 3 άτομα. Όπως θα δείξουμε στη συνέχεια, και αυτή η ειδική περίπτωση του SAT είναι πλήρης για την κλάση NP.



Σχήμα 5.1: Σχηματική αναπαράσταση αναγωγών σε προβλήματα Λογικής

Στιγμιότυπο: Λογική έκφραση ϕ σε Συζευκτική Κανονική Μορφή ορισμένη σε n μεταβλητές, με m προτάσεις, όπου κάθε πρόταση περιέχει ακριβώς 3 άτομα.

Ερώτηση: Υπάρχει αποτίμηση αληθείας που να ικανοποιεί την ϕ ;

Θεώρημα 5.1 *Το 3SAT είναι NP-πλήρες.*

Απόδειξη: Κατ' αρχήν, είναι εύκολο να δούμε ότι το πρόβλημα 3SAT ανήκει στην κλάση NP αφού δοθέντος ενός στιγμιότυπου αυτού μπορούμε να μαντέψουμε μια αποτίμηση αληθείας $v = v_1v_2 \cdots v_n$ και να επαληθεύσουμε σε πολωνυμικό χρόνο εάν η αποτίμηση αυτή ικανοποιεί την ϕ .

Για να δείξουμε ότι το 3SAT είναι πλήρες για την κλάση NP θα ανάγουμε ένα γνωστό NP-πλήρες πρόβλημα στο άγνωστο πρόβλημα μας. Από το προηγούμενο κεφάλαιο γνωρίζουμε ένα NP-πλήρες πρόβλημα, το SAT. Έστω λοιπόν $X = \{x_1, x_2, \dots, x_n\}$ το σύνολο των μεταβλητών και $C = \{c_1, c_2, \dots, c_m\}$ το σύνολο των προτάσεων που συνθέτουν ένα τυχαίο στιγμιότυπο f του SAT. Θα κατασκευάσουμε ένα στιγμιότυπο του 3SAT, δηλαδή ένα σύνολο μεταβλητών $X' = \{x'_1, x'_2, \dots, x'_{n'}\}$ και ένα σύνολο προτάσεων $C' = \{c'_1, c'_2, \dots, c'_{m'}\}$, με κάθε πρόταση να περιέχει ακριβώς 3 άτομα, που θα συνθέτουν την λογική έκφραση ϕ , έτσι ώστε, η ϕ είναι ικανοποιήσιμη αν και μόνο αν η f είναι ικανοποιήσιμη.

Η κατασκευή του στιγμιότυπου του 3SAT βασίζεται στην αντικατάσταση κάθε πρότασης c_i της f , $i = 1, \dots, m$, από ένα σύνολο C'_i από ισοδύναμες προτάσεις όπου κάθε μια θα περιέχει ακριβώς 3 άτομα. Οι προτάσεις του συνόλου C'_i ορίζονται στις μεταβλητές της ϕ αλλά και σε ένα σύνολο βοηθητικών μεταβλητών. Έτσι, το σύνολο των μεταβλητών που στο οποίο ορίζεται η λογική έκφραση ϕ και το σύνολο των προτάσεων αυτής είναι αντίστοιχα,

$$X' = X \cup \bigcup_{i=1}^m X'_i \quad \text{και} \quad C' = \bigcup_{i=1}^m C'_i.$$

Έστω ότι η τυχαία πρόταση c_i της f αποτελείται από άτομα του συνόλου των μεταβλητών $\{z_1, \dots, z_k\} \subseteq X$, όπου k θετικός ακέραιος, $k \leq n$. Το σύνολο των προτάσεων της ϕ που αντιστοιχούν στην c_i εξαρτάται από την τιμή του k :

$k = 1$: Σε αυτή την περίπτωση η c_i αποτελείται από ένα μόνο άτομο, έστω το z_1 . Για τον ορισμό του συνόλου C'_i των προτάσεων της ϕ χρησιμοποιούμε το σύνολο των βοηθητικών μεταβλητών $X'_i = \{y_i^1, y_i^2\}$ (δύο βοηθητικές μεταβλητές). Το σύνολο των προτάσεων C'_i είναι:

$$C'_i = \{(z_1 \vee y_i^1 \vee y_i^2), (z_1 \vee \neg y_i^1 \vee y_i^2), (z_1 \vee y_i^1 \vee \neg y_i^2), (z_1 \vee \neg y_i^1 \vee \neg y_i^2)\}.$$

Το C'_i ορίζεται στη μεταβλητή z_1 και στις δύο νέες βοηθητικές μεταβλητές του συνόλου X'_i .

$k = 2$: Σε αυτή την περίπτωση η c_i αποτελείται από δύο άτομα, έστω τα z_1, z_2 . Για τον ορισμό του C'_i χρησιμοποιούμε το σύνολο των βοηθητικών μεταβλητών $X'_i = \{y_i^1\}$ (μία βοηθητική μεταβλητή). Το σύνολο των προτάσεων C'_i είναι:

$$C'_i = \{(z_1 \vee z_2 \vee y_i^1), (z_1 \vee z_2 \vee \neg y_i^1)\}.$$

$k = 3$: Σε αυτή την περίπτωση η c_i αποτελείται από τρία άτομα και παραμένει ως έχει στην λογική έκφραση ϕ . Δηλαδή, $X'_i = \emptyset$ (καμμία βοηθητική μεταβλητή) και $C'_i = \{c_i\}$.

$k > 3$: Σε αυτή την περίπτωση χρησιμοποιούμε το σύνολο των $k - 3$ βοηθητικών μεταβλητών $X'_i = \{y_i^j : 1 \leq j \leq k - 3\}$. Το σύνολο των προτάσεων C'_i είναι:

$$C'_i = \{(z_1 \vee z_2 \vee y_i^1)\} \cup \{(\neg y_i^j \vee z_{j+2} \vee y_i^{j+1}) : 1 \leq j \leq k - 4\} \\ \cup \{(\neg y_i^{k-3} \vee z_{k-1} \vee z_k)\}.$$

Ορίσαμε έτσι σε κάθε περίπτωση το σύνολο των προτάσεων C'_i της ϕ που αντιστοιχούν σε κάθε πρόταση c_i της f . Ξεκινώντας λοιπόν από ένα τυχαίο στιγμιότυπο f του SAT κατασκευάσαμε ένα στιγμιότυπο ϕ του 3SAT.

Θα δείξουμε τώρα ότι η ϕ είναι ικανοποιήσιμη αν και μόνο αν η f είναι ικανοποιήσιμη. Γι' αυτό αρκεί να δείξουμε ότι το σύνολο των προτάσεων C'_i της ϕ είναι ικανοποιήσιμο αν και μόνο αν η πρόταση c_i της ϕ είναι ικανοποιήσιμη. Και αυτό διότι, από τον τρόπο που κατασκευάσαμε τα σύνολα των προτάσεων της ϕ , κάθε σύνολο προτάσεων C'_i ορίστηκε στο σύνολο των βοηθητικών μεταβλητών X'_i τα στοιχεία του οποίου είναι διαφορετικά από τα στοιχεία του συνόλου βοηθητικών μεταβλητών X'_j που αντιστοιχεί στο σύνολο των προτάσεων C'_j της ϕ .

Ας υποθέσουμε αρχικά ότι η πρόταση c_i είναι ικανοποιήσιμη και έστω $t : X \rightarrow \{0, 1\}$ μια αποτίμηση αληθείας που την ικανοποιεί. Θα δείξουμε ότι η t μπορεί να επεκταθεί σε μια αποτίμηση αληθείας $t' : X' \rightarrow \{0, 1\}$ που να ικανοποιεί το σύνολο C'_i . Εφόσον το σύνολο των μεταβλητών $X' \setminus X$ χωρίζεται στα σύνολα X'_i , και οι μεταβλητές σε κάθε σύνολο X'_i εμφανίζονται μόνο στο σύνολο των προτάσεων C'_i , αρκεί να δείξουμε ότι η αποτίμηση αληθείας t μπορεί να επεκταθεί κάθε φορά στα σύνολα X'_i και να επαληθεύσουμε ότι η αποτίμηση αυτή ικανοποιεί το αντίστοιχο σύνολο προτάσεων C'_i . Ας εξετάσουμε λοιπόν μια προς μια τις παραπάνω περιπτώσεις:

$k = 1$: Το σύνολο C'_i ικανοποιείται από την τιμή που αποδίδει η t στη μεταβλητή z_1 . Άρα οι μεταβλητές του X'_i μπορούν να πάρουν αυθαίρετες τιμές.

$k = 2$: Το σύνολο C'_i ικανοποιείται από τις τιμές που αποδίδει η t στις μεταβλητή z_1 και z_2 . Άρα οι μεταβλητές του X'_i μπορούν επίσης να πάρουν αυθαίρετες τιμές.

$k = 3$: Σε αυτή την περίπτωση η μοναδική πρόταση του C'_i ικανοποιείται ήδη από την t .

$k > 3$: Αφού η t ικανοποιεί την c_i , υπάρχει τουλάχιστον ένας θετικός ακέραιος l για το οποίο το άτομο z_l ικανοποιείται από την t . Αν $l = 1$ ή $l = 2$, τότε θέτουμε $t(y_i^j) = 0$. Αν $l = k - 1$ ή $l = k$, τότε θέτουμε $t(y_i^j) = 1$. Τέλος, σε κάθε άλλη περίπτωση θέτουμε $t(y_i^j) = 1$, όταν $1 \leq j \leq l - 2$, και $t(y_i^j) = 0$ όταν $l - 1 \leq j \leq k - 3$. Με αυτό τον τρόπο η αποτίμηση αληθείας t' ικανοποιεί κάθε πρόταση του συνόλου C'_i .

Σε κάθε περίπτωση λοιπόν, η αποτίμηση αληθείας t που ικανοποιεί την λογική έκφραση f μπορεί να επεκταθεί σε μια t' που ικανοποιεί την ϕ . Αντίστροφα τώρα, ας υποθέσουμε ότι η ϕ είναι ικανοποιήσιμη και έστω t' μια αποτίμηση αληθείας αυτής. Είναι εύκολο να δει κανείς, από τον τρόπο που κατασκευάσαμε την ϕ , ότι ο περιορισμός t της t' στις μεταβλητές του συνόλου X είναι μια αποτίμηση αληθείας που ικανοποιεί κάθε πρόταση της f . Άρα η λογική έκφραση ϕ είναι ικανοποιήσιμη αν και μόνο αν η f είναι ικανοποιήσιμη.

Για να ολοκληρώσουμε την απόδειξη πρέπει να δείξουμε ότι ο παραπάνω μετασχηματισμός μπορεί να γίνει σε λογαριθμικό χώρο στο μέγεθος της εισόδου, που στην περίπτωση μας είναι το μέγεθος του στιγμιοτύπου του προβλήματος SAT. Παρατηρούμε ότι το πλήθος των προτάσεων της ϕ (προτάσεις με 3 άτομα η κάθε μια) είναι της τάξης $O(nm)$. Άρα, το μέγεθος του στιγμιοτύπου του 3SAT φράσσεται από μια πολυωνυμική συνάρτηση του μεγέθους του τυχαίου στιγμιοτύπου του SAT. Για να γίνει επομένως ο μετασχηματισμός απαιτείται

σταθερός αριθμός από βοηθητικές μεταβλητές που καταχωρούν πολυωνυμικά μεγάλους αριθμούς, συνεπώς $O(\log(nm))$ χώρος είναι αρκετός. \square

Το 3SAT είναι το περισσότερο χρησιμοποιούμενο πρόβλημα για την απόδειξη ενός αποτελέσματος NP-πληρότητας. Το 3SAT μπορεί να χρησιμοποιηθεί όπου και το SAT και επιπλέον δίνει απλούστερες αποδείξεις. Επίσης, επειδή περιλαμβάνει στη διατύπωση του μια πρόταση της Λογικής είναι ευκολότερο να το «μειωθεί» το πρόβλημα που θέλουμε να δείξουμε NP-πλήρες από το αν χρησιμοποιούσαμε άλλο πρόβλημα σαν αφετηρία. Συνήθως, αν η φύση του προβλήματος δεν υποδεικνύει άμεσα πιο πρόβλημα πρέπει να επιλέξουμε για την αναγωγή μας, είναι καλή τακτική να αρχίζουμε από το 3SAT.

Το πρόβλημα της ικανοποιησιμότητας μιας λογικής έκφρασης ϕ ζητά αν υπάρχει ή όχι μια αποτίμηση αληθείας που να ικανοποιεί κάθε πρόταση της ϕ . Είναι λογικό να αναρωτηθεί κανείς αν υπάρχει μια αποτίμηση αληθείας που να μην ικανοποιεί κατ' ανάγκη όλες της προτάσεις της ϕ αλλά όσο το δυνατό περισσότερες. Πρόκειται για την έκδοση *βελτιστοποίησης* του SAT, το πρόβλημα της ΜΕΓΙΣΤΗΣ ΙΚΑΝΟΠΟΙΗΣΙΜΟΤΗΤΑΣ (MAXIMUM SATISFIABILITY ή MAXSAT για συντομία), η οποία μπορεί να πάρει τη μορφή προβλήματος απόφασης ορίζοντας ως στόχο έναν θετικό ακέραιο K :

Στιγμιότυπο: Λογική έκφραση ϕ σε ΚΣΜ ορισμένη σε n μεταβλητές, με m προτάσεις, και θετικός ακέραιος $k \leq m$.

Ερώτηση: Υπάρχει αποτίμηση αληθείας που να ικανοποιεί τουλάχιστον k προτάσεις της ϕ ;

Θεώρημα 5.2 *Το MAXSAT είναι NP-πλήρες.*

Απόδειξη: Είναι εύκολο να δούμε ότι το MAXSAT ανήκει στην κλάση NP αφού μπορούμε σε πολυωνυμικό χρόνο να μαντέψουμε μια αποτίμηση αληθείας και να επαληθεύσουμε ότι η αποτίμηση αυτή ικανοποιεί τουλάχιστον k προτάσεις την ϕ . Για να δείξουμε ότι το MAXSAT είναι πλήρες για την κλάση NP θα ανάγουμε το πρόβλημα SAT στο MAXSAT.

Η αναγωγή είναι πολύ απλή: αρκεί να δούμε ότι το MAXSAT είναι μια γενίκευση του SAT, δηλαδή ότι κάθε στιγμιότυπο του SAT είναι ειδική περίπτωση του MAXSAT όπου η παράμετρος k τυχαίνει να είναι ίση με το πλήθος των προτάσεων, m . Αυτό το είδος της αναγωγής είναι αρκετά συχνό και πολύ χρήσιμο στο να αποδεικνύουμε ότι κάποια προβλήματα είναι NP-πλήρη.

Τυπικά η αναγωγή κατασκευάζει από ένα στιγμιότυπο του SAT ένα στιγμιότυπο του MAXSAT όπου η μεν έκφραση ϕ είναι πανομοιότυπη με του SAT, ο δε ακέραιος K είναι όλες οι προτάσεις του SAT. Είναι προφανές ότι το δοθέν SAT είναι ικανοποιήσιμο αν υπάρχει αποτίμηση που ικανοποιεί K ή περισσότερες προτάσεις του MAXSAT. \square

Το πρόβλημα της ΜΟΝΟΤΟΝΗΣ ΞΙΚΑΝΟΠΟΙΗΣΙΜΟΤΗΤΑΣ (MON3SAT) αποτελεί ειδική περίπτωση του 3SAT όπου κάθε πρόταση περιέχει 3 άτομα είτε όλα με θετικό είτε όλα με αρνητικό πρόσημο. Θα δείξουμε ότι το πρόβλημα MON3SAT είναι πλήρες για την κλάση NP.

Στιγμιότυπο: Λογική έκφραση ϕ σε ΚΣΜ ορισμένη σε n μεταβλητές, με m προτάσεις, όπου κάθε πρόταση περιέχει ακριβώς 3 άτομα είτε με θετικό είτε με αρνητικό πρόσημο.

Ερώτηση: Υπάρχει αποτίμηση αληθείας που να ικανοποιεί την ϕ ;

Θεώρημα 5.3 *Το MON3SAT είναι NP-πλήρες.*

Απόδειξη: Κατ' αρχήν, είναι εύκολο να δούμε ότι το MON3SAT ανήκει στην κλάση NP αφού μπορούμε σε πολυωνυμικό χρόνο να μαντέψουμε μια αποτίμηση αληθείας και να επαληθεύσουμε εάν η αποτίμηση αυτή ικανοποιεί την ϕ .

Για να δείξουμε ότι το MON3SAT είναι πλήρες για την κλάση NP θα ανάγουμε το 3SAT (γνωστό NP-πλήρες πρόβλημα) στο MON3SAT. Έστω λοιπόν $X = \{x_1, x_2, \dots, x_n\}$ το σύνολο των μεταβλητών και $C = \{c_1, c_2, \dots, c_m\}$ το σύνολο των προτάσεων που συνθέτουν ένα τυχαίο στιγμιότυπο f του 3SAT (κάθε πρόταση $c_i, i = 1, \dots, m$, περιέχει ακριβώς 3 άτομα). Θα κατασκευάσουμε ένα στιγμιότυπο του MON3SAT, δηλαδή ένα σύνολο μεταβλητών $X' = \{x'_1, x'_2, \dots, x'_{n'}\}$ και ένα σύνολο προτάσεων $C' = \{c'_1, c'_2, \dots, c'_{m'}\}$, με κάθε πρόταση $c'_j, j = 1, \dots, m'$, να περιέχει ακριβώς 3 άτομα, είτε όλα με θετικό είτε όλα με αρνητικό πρόσημο, που θα συνθέτουν την λογική έκφραση ϕ , έτσι ώστε, η ϕ είναι ικανοποιήσιμη αν και μόνο αν η f είναι ικανοποιήσιμη.

Η κατασκευή του στιγμιότυπου του MON3SAT βασίζεται στην αντικατάσταση κάθε πρότασης c_i της f , $i = 1, \dots, m$, από ένα σύνολο C'_i από ισοδύναμες προτάσεις όπου κάθε μια θα περιέχει ακριβώς 3 άτομα είτε όλα θετικά είτε όλα αρνητικά. Οι προτάσεις του συνόλου C'_i ορίζονται στις μεταβλητές της f αλλά και σε ένα σύνολο βοηθητικών μεταβλητών. Έτσι, το σύνολο των μεταβλητών στο οποίο ορίζεται η λογική έκφραση ϕ και το σύνολο των προτάσεων αυτής είναι αντίστοιχα,

$$X' = X \cup \bigcup_{i=1}^m X'_i \quad \text{και} \quad C' = \bigcup_{i=1}^m C'_i.$$

Έστω ότι η τυχαία πρόταση c_i της f αποτελείται από άτομα του συνόλου των μεταβλητών $\{z_{i_1}, z_{i_2}, z_{i_3}\} \subseteq X$, τα οποία εμφανίζονται στην c_i με τυχαίο πρόσημο. Εξετάζουμε τις παρακάτω περιπτώσεις:

1. $c_i = (z_{i_1} \vee z_{i_2} \vee z_{i_3})$. Σε αυτή την περίπτωση το σύνολο των προτάσεων της ϕ που αντιστοιχούν στην c_i περιέχει απλά την πρόταση c_i , δηλαδή

$C'_i = \{c_i\}$ και $X'_i = \emptyset$, μιας και όλα τα άτομα της c_i έχουν θετικό πρόσημο.

2. $c_i = (\neg z_{i_1} \vee z_{i_2} \vee z_{i_3})$. Για τον ορισμό των προτάσεων C'_i της ϕ χρησιμοποιούμε 2 βοηθητικές μεταβλητές $X'_i = \{y_i^1, y_i^2\}$ και το σύνολο C'_i ορίζεται ως

$$C'_i = \{(\neg z_{i_1} \vee \neg y_i^1 \vee \neg y_i^2), (y_i^1 \vee z_{i_2} \vee z_{i_3}), (y_i^2 \vee z_{i_2} \vee z_{i_3})\}.$$

3. $c_i = (\neg z_{i_1} \vee \neg z_{i_2} \vee z_{i_3})$. Για τον ορισμό των προτάσεων C'_i της ϕ χρησιμοποιούμε 2 βοηθητικές μεταβλητές $X'_i = \{y_i^1, y_i^2\}$ και το σύνολο C'_i ορίζεται ως

$$C'_i = \{(\neg z_{i_1} \vee \neg z_{i_2} \vee \neg y_i^1), (\neg z_{i_1} \vee \neg z_{i_2} \vee \neg y_i^2), (y_i^1 \vee y_i^2 \vee z_{i_3})\}.$$

4. $c_i = (\neg z_{i_1} \vee \neg z_{i_2} \vee \neg z_{i_3})$. Όπως και στην πρώτη περίπτωση, $C'_i = \{c_i\}$ και $X'_i = \emptyset$.

Ορίσαμε έτσι σε κάθε περίπτωση το σύνολο των προτάσεων C'_i της ϕ που αντιστοιχούν σε κάθε πρόταση c_i της f . Ξεκινώντας λοιπόν από ένα τυχαίο στιγμιότυπο f του 3SAT κατασκευάσαμε ένα στιγμιότυπο ϕ του MON3SAT. Θα δείξουμε τώρα ότι η ϕ είναι ικανοποιήσιμη αν και μόνο αν η f είναι ικανοποιήσιμη. Γι' αυτό αρκεί να δείξουμε ότι το σύνολο των προτάσεων C'_i της ϕ είναι ικανοποιήσιμο αν και μόνο αν η πρόταση c_i της f είναι ικανοποιήσιμη.

Ας υποθέσουμε αρχικά ότι η πρόταση c_i είναι ικανοποιήσιμη και έστω $t : X \rightarrow \{0, 1\}$ μια αποτίμηση αλήθειας που την ικανοποιεί. Θα δείξουμε ότι η t μπορεί να επεκταθεί σε μια αποτίμηση αλήθειας $t' : X' \rightarrow \{0, 1\}$ που να ικανοποιεί το σύνολο C'_i . Εφόσον το σύνολο των μεταβλητών $X' \setminus X$ χωρίζεται στα σύνολα X'_i , και οι μεταβλητές σε κάθε σύνολο X'_i εμφανίζονται μόνο στο σύνολο των προτάσεων C'_i , αρκεί να δείξουμε ότι η αποτίμηση αλήθειας t μπορεί να επεκταθεί κάθε φορά στα σύνολα X'_i και να επαληθεύσουμε ότι η αποτίμηση αυτή ικανοποιεί το αντίστοιχο σύνολο προτάσεων C'_i . Ας εξετάσουμε λοιπόν μια προς μια τις παραπάνω περιπτώσεις:

- Έχουμε ότι $c_i = (z_{i_1} \vee z_{i_2} \vee z_{i_3})$, $C'_i = \{c_i\}$ και $X'_i = \emptyset$. Σε αυτή την περίπτωση, το σύνολο C'_i ικανοποιείται ήδη από την t .
- Έχουμε ότι $c_i = (\neg z_{i_1} \vee z_{i_2} \vee z_{i_3})$, $X'_i = \{y_i^1, y_i^2\}$ και $C'_i = \{(\neg z_{i_1} \vee \neg y_i^1 \vee \neg y_i^2), (y_i^1 \vee z_{i_2} \vee z_{i_3}), (y_i^2 \vee z_{i_2} \vee z_{i_3})\}$.

Η t αποδίδει τιμή αλήθεια σε τουλάχιστον ένα άτομο της πρότασης c_i . Συνεπώς, τουλάχιστον μία από τις προτάσεις του C'_i ικανοποιείται από τις τιμές που αποδίδει η t στις μεταβλητές z_{i_1} , z_{i_2} και z_{i_3} . Είναι εύκολο

να δούμε ότι υπάρχουν κατάλληλες τιμές των βοηθητικών μεταβλητών X'_i που να ικανοποιούν και τις υπόλοιπες προτάσεις του C'_i .

Για παράδειγμα, έστω ότι $t(z_{i_1}) = 1, t(z_{i_2}) = 1$ και $t(z_{i_3}) = 0$, δηλαδή ότι η πρόταση c_i επαληθεύεται μόνο από την τιμή της μεταβλητής z_{i_2} . Η μεταβλητή z_{i_2} εμφανίζεται με το ίδιο πρόσημο στην δεύτερη και τρίτη πρόταση του C'_i . Άρα αυτές επαληθεύονται αυτόματα, ανεξάρτητα από τις τιμές των βοηθητικών μεταβλητών y_i^1 και y_i^2 . Για να επαληθεύσουμε τώρα και την πρώτη πρόταση του C'_i αρκεί να θέσουμε $t'(y_i^1) = 0$ ή $t'(y_i^2) = 0$.

- Έχουμε ότι $c_i = (\neg z_{i_1} \vee \neg z_{i_2} \vee z_{i_3})$, $X'_i = \{y_i^1, y_i^2\}$ και $C'_i = \{(\neg z_{i_1} \vee \neg z_{i_2} \vee \neg y_i^1), (\neg z_{i_1} \vee \neg z_{i_2} \vee \neg y_i^2), (y_i^1 \vee y_i^2 \vee z_{i_3})\}$. Όπως και στην προηγούμενη περίπτωση, η t επαληθεύει τουλάχιστον μία από τις προτάσεις του C'_i και εύκολα μπορεί να επεκταθεί, δίνοντας κατάλληλες τιμές στις βοηθητικές μεταβλητές y_i^1 και y_i^2 , έτσι ώστε να επαληθεύει και τις υπόλοιπες προτάσεις του C'_i .
- Έχουμε ότι $c_i = (\neg z_{i_1} \vee \neg z_{i_2} \vee \neg z_{i_3})$, $C'_i = \{c_i\}$ και $X'_i = \emptyset$. Σε αυτή την περίπτωση, το σύνολο C'_i ικανοποιείται ήδη από την t .

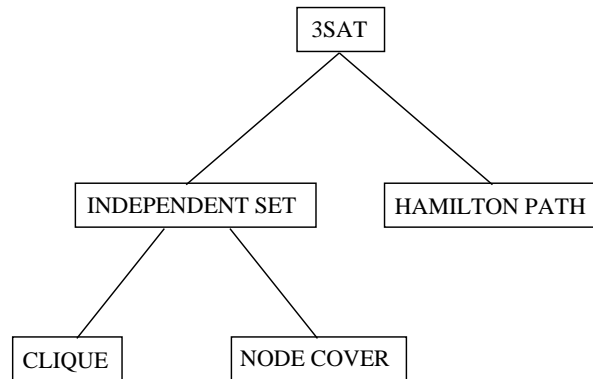
Σε κάθε περίπτωση λοιπόν, η αποτίμηση αληθείας t που ικανοποιεί την λογική έκφραση f μπορεί να επεκταθεί σε μια t' που ικανοποιεί την ϕ . Αντίστροφα τώρα, ας υποθέσουμε ότι η ϕ είναι ικανοποιήσιμη και έστω t' μια αποτίμηση αληθείας αυτής. Είναι εύκολο να δει κανείς, από τον τρόπο που κατασκευάσαμε την ϕ , ότι ο περιορισμός t της t' στις μεταβλητές του συνόλου X είναι μια αποτίμηση αληθείας που ικανοποιεί κάθε πρόταση της f . Άρα η λογική έκφραση f είναι ικανοποιήσιμη αν και μόνο αν η ϕ είναι ικανοποιήσιμη.

Για να ολοκληρώσουμε την απόδειξη αρκεί να δούμε ότι ο παραπάνω μετασχηματισμός μπορεί να γίνει σε λογαριθμικό χώρο στο μέγεθος της εισόδου. \square

5.2 Προβλήματα από τη Θεωρία Γραφημάτων

Στην ενότητα αυτή θα ασχοληθούμε με κλασικά προβλήματα που προέρχονται από την Θεωρία Γραφημάτων. Η σχηματική αναπαράσταση των αναγωγών που θα ακολουθήσουν φαίνεται στο Σχήμα 5.2.

Ορισμός 5.1 Ένα γράφημα $G = (V, E)$ είναι ένα πεπερασμένο σύνολο V από κόμβους και ένα πεπερασμένο σύνολο E από ακμές. Κάθε ακμή $e \in E$ είναι ένα ζεύγари κόμβων (i, j) του V , $i \neq j$. Εάν υπάρχει μια διάταξη



Σχήμα 5.2: Σχηματική αναπαράσταση αναγωγών σε προβλήματα Θεωρίας Γραφημάτων.

μεταξύ των κόμβων i και j , τότε θα λέμε ότι το γράφημα είναι κατευθυντικό. Σε διαφορετική περίπτωση θα λέμε ότι το γράφημα είναι μη κατευθυντικό.

Στη συνέχεια θα ασχοληθούμε με μη κατευθυντικά γραφήματα.

Ορισμός 5.2 Ένα ανεξάρτητο σύνολο (*independent set*) ενός γραφήματος $G = (V, E)$ είναι ένα σύνολο κορυφών $I \subseteq V$ του οποίου κανένα ζευγάρι κόμβων του δεν συνδέεται με ακμή, δηλαδή, για κάθε $i, j \in I$ με $i \neq j$, ισχύει ότι $(i, j) \notin E$. Ως μέγεθος του I θα καλούμε τον πληθάνισμο του I .

Το πρόβλημα του ΑΝΕΞΑΡΤΗΤΟΥ ΣΥΝΟΛΟΥ (INDEPENDENT SET) ορίζεται ως εξής:

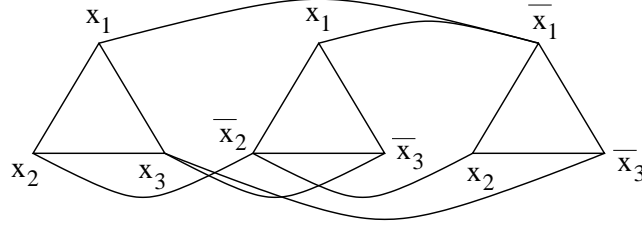
Στιγμιότυπο: Μη κατευθυντικό γράφημα $G = (V, E)$ και ακέραιος $K \geq 0$.

Ερώτηση: Έχει το G ανεξάρτητο σύνολο μεγέθους K ή μεγαλύτερο;

Θεώρημα 5.4 Το ΑΝΕΞΑΡΤΗΤΟ ΣΥΝΟΛΟ είναι NP-πλήρες.

Απόδειξη: Είναι φανερό ότι το INDEPENDENT SET ανήκει στην κλάση NP αφού σε πολυωνυμικό χρόνο μπορούμε να μαντέψουμε ένα υποσύνολο κόμβων του V με πληθάνισμο K και να επαληθεύσουμε εάν είναι ανεξάρτητο σύνολο. Θα αποδείξουμε ότι το INDEPENDENT SET είναι NP-πλήρες κάνοντας αναγωγή από το 3SAT. Έστω ένα τυχαίο στιγμιότυπο του 3SAT: Λογική έκφραση ϕ με m προτάσεις C_1, \dots, C_m , ορισμένη σε n μεταβλητές. Θα κατασκευάσουμε ένα στιγμιότυπο του INDEPENDENT SET, δηλαδή ένα γράφημα $G = (V, E)$ και έναν μη αρνητικό ακέραιο K , έτσι ώστε ο G έχει ανεξάρτητο σύνολο μεγέθους τουλάχιστον K αν και μόνο αν η ϕ είναι ικανοποιήσιμη. Η κατασκευή του G γίνεται ως εξής: Για κάθε πρόταση $C_i = (x_{i1} \vee x_{i2} \vee x_{i3})$ της ϕ φτιάχνουμε

$$(x_1 \vee x_2 \vee x_3) \wedge (x_1 \vee \neg x_2 \vee \neg x_3) \wedge (\neg x_1 \vee x_2 \vee \neg x_3)$$



Σχήμα 5.3: Αναγωγή του 3SAT στο INDEPENDENT SET.

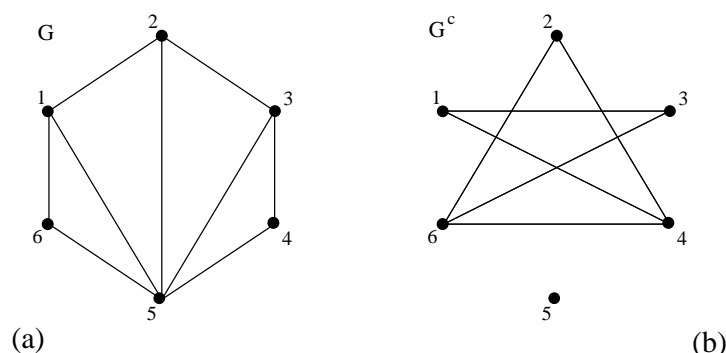
μια τριάδα κόμβων, ένας κόμβος για κάθε άτομο της πρότασης, τους οποίους συνδέουμε μεταξύ τους με ακμές. Με αυτό τον τρόπο έχουμε κατασκευάσει m τρίγωνα. Επίσης, συνδέουμε με ακμή δύο κόμβους που ανήκουν σε διαφορετικά τρίγωνα αν και μόνο αν τα άτομα που αντιστοιχούν σε αυτούς τους κόμβους έχουν αντίθετα πρόσημα. Ένα παράδειγμα αυτής της κατασκευής φαίνεται στο Σχήμα 5.3.

Πιο αυστηρά, η κατασκευή του στιγμιοτύπου του INDEPENDENT SET είναι η εξής: $G = (V, E)$ με $V = \{v_{ij} : i = 1, \dots, m, j = 1, 2, 3\}$ και $E = \{(v_{ij}, v_{ik}) : i = 1, \dots, m, j \neq k\} \cup \{(v_{il}, v_{ik}) : i \neq l, x_{il} = \neg x_{ik}\}$. Επίσης, θέτουμε $K = m$. Θα δείξουμε ότι το γράφημα G που κατασκευάσαμε έχει ανεξάρτητο σύνολο μεγέθους τουλάχιστον K αν και μόνο αν η λογική έκφραση ϕ είναι ικανοποιήσιμη.

Έστω ότι το γράφημα G έχει ανεξάρτητο σύνολο I μεγέθους τουλάχιστον K . Επειδή μόνο ένας κόμβος από κάθε τρίγωνο μπορεί να μετέχει στο I και υπάρχουν m τρίγωνα, το I έχει μέγεθος ακριβώς K , δηλαδή περιέχει ακριβώς ένα κόμβο από κάθε τρίγωνο και κατά συνέπεια ένα άτομο από κάθε πρόταση της ϕ . Επιπλέον, το I δεν περιέχει συμπληρωματικά άτομα, αφού οι αντίστοιχοι κόμβοι του G συνδέονται μεταξύ τους με ακμή. Ας θεωρήσουμε την αποτίμηση αλήθειας t που αποδίδει τιμή αλήθεια στα άτομα που αντιστοιχούν στους κόμβους του I και οποιαδήποτε τιμή στις υπόλοιπες μεταβλητές. Είναι φανερό ότι η t ικανοποιεί την ϕ .

Αντίστροφα, έστω ότι η ϕ είναι ικανοποιήσιμη και έστω t μια αποτίμηση αλήθειας που ικανοποιεί την ϕ . Αναγνωρίζουμε ένα άτομο από κάθε πρόταση που έχει τιμή αλήθεια και τοποθετούμε τον αντίστοιχο κόμβο του αντίστοιχου τριγώνου στο σύνολο I . Το I αποτελεί ανεξάρτητο σύνολο του G μεγέθους $K = m$. \square

Ένα εύκολο πόρισμα του παραπάνω θεωρήματος αφορά το λεγόμενο πρόβλημα της ΚΛΙΚΑΣ (CLIQUE). Ένα σύνολο κόμβων $C \subseteq V$ ενός γραφήματος



Σχήμα 5.4: **(a),(b)** Αναγωγή του INDEPENDENT SET στο CLIQUE. Το σύνολο $I = \{2, 4, 6\}$ αποτελεί ανεξάρτητο σύνολο του γραφήματος G και κλίκα του συμπληρωματικού γραφήματος G^c . **(a)** Αναγωγή του INDEPENDENT SET στο NODE COVER. Το σύνολο $I = \{2, 4, 6\}$ αποτελεί ανεξάρτητο σύνολο του γραφήματος ενώ το σύνολο $V \setminus I = \{1, 3, 5\}$ αποτελεί κάλυμμα κόμβων αυτού.

$G = (V, E)$ αποτελεί κλίκα (clique) του G εάν κάθε ζευγάρι κόμβων του C συνδέεται με ακμή. Το πρόβλημα CLIQUE είναι το ακόλουθο:

Στιγμιότυπο: Μη κατευθυντικό γράφημα $G = (V, E)$ και ακέραιος $K \geq 0$.

Ερώτηση: Έχει το G κλίκα μεγέθους K ή μεγαλύτερη;

Πόρισμα 5.1 Το πρόβλημα της ΚΛΙΚΑΣ είναι NP-πλήρες.

Είναι φανερό ότι αν το γράφημα G έχει ανεξάρτητο σύνολο I τότε το I αποτελεί κλίκα για το συμπληρωματικό γράφημα G^c , και αντίστροφα. Το συμπληρωματικό γράφημα $G^c = (V, E^c)$ ενός γραφήματος $G = (V, E)$ είναι το γράφημα που έχει ακμές όλες τις δυνατές ακμές που μπορούν να οριστούν από τους κόμβους του V πλην των ακμών του E , δηλαδή, $E^c = \{(u, v) : u \neq v, (u, v) \notin E\}$.

Ένα παράδειγμα ενός γραφήματος G και του συμπληρωματικού του, G^c , καθώς και η σχέση που έχει ένα ανεξάρτητο σύνολο του G με μια κλίκα του G^c φαίνεται στο Σχήμα 5.4. Αφού το πρόβλημα INDEPENDENT SET είναι NP-πλήρες, το πρόβλημα CLIQUE επίσης είναι NP-πλήρες. \square

Ορισμός 5.3 Ένα κάλυμμα κόμβων (node cover) ενός γραφήματος $G = (V, E)$ είναι ένα σύνολο κορυφών $C \subseteq V$ για το οποίο ισχύει ότι κάθε ακμή του G έχει τουλάχιστον έναν κόμβο της μέσα στο C .

Το πρόβλημα ΚΑΛΥΜΜΑ ΚΟΜΒΩΝ (NODE COVER) ορίζεται ως εξής:

Στιγμιότυπο: Μη κατευθυντικό γράφημα $G = (V, E)$ και ακέραιος $K \geq 0$.

Ερώτηση: Έχει το G κάλυμμα κόμβων μεγέθους K ή μικρότερο;

Θεώρημα 5.5 Το ΚΑΛΥΜΜΑ ΚΟΜΒΩΝ είναι NP-πλήρες.

Απόδειξη: Είναι φανερό ότι αν το γράφημα G έχει ανεξάρτητο σύνολο I μεγέθους τουλάχιστον K τότε το $C = V \setminus I$ αποτελεί κάλυμμα κορυφών του G μεγέθους το πολύ $|V| - K$. Ένα παράδειγμα φαίνεται στο Σχήμα 5.4 (a). Αφού το πρόβλημα INDEPENDENT SET είναι NP-πλήρες, το πρόβλημα NODE COVER επίσης είναι NP-πλήρες. \square

Προχωρούμε τώρα σε μία δύσκολη απόδειξη που χρησιμοποιεί την τεχνική του σχεδιασμού ειδικών κατασκευών (gadgets) που έχουν συγκεκριμένες ιδιότητες.

Ορισμός 5.4 Ένας κύκλος hamilton (hamilton cycle) ενός γραφήματος $G = (V, E)$ είναι ένας απλός κύκλος, μονοπάτι $\{v_0, \dots, v_k\}$ με $v_0 = v_k$, που διέρχεται από κάθε κόμβο του V . Το πρόβλημα του ΚΥΚΛΟΥ HAMILTON (HAMILTON CYCLE) είναι, δοθέντος ενός γραφήματος G , να αποφανθούμε εάν ο G έχει ή όχι κύκλο hamilton.

Στιγμιότυπο: Μη κατευθυντικό γράφημα $G = (V, E)$.

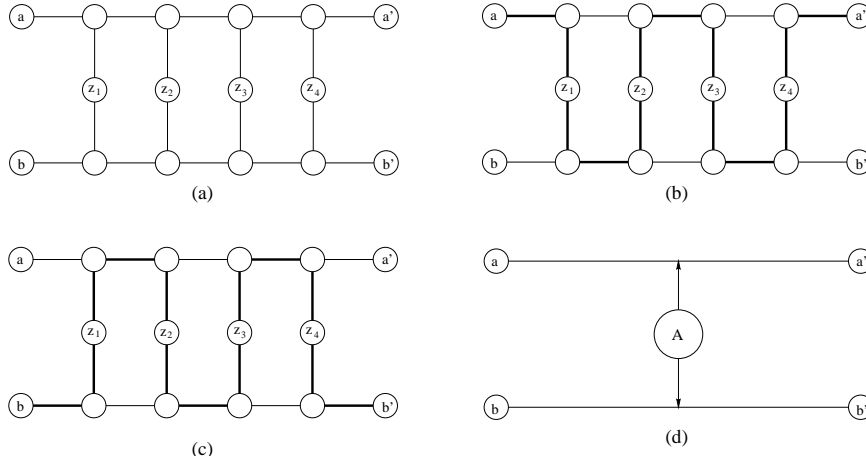
Ερώτηση: Έχει το G κύκλο hamilton;

Θεώρημα 5.6 Ο ΚΥΚΛΟΣ HAMILTON είναι NP-πλήρες πρόβλημα.

Απόδειξη: Το πρόβλημα προφανώς ανήκει στην κλάση NP αφού σε πολυνομικό χρόνο μπορούμε να επιλέξουμε μια ακολουθία $|V|$ το πλήθος κόμβων, στην οποία κάθε κόμβος εμφανίζεται ακριβώς μία φορά, και να ελέγχουμε ότι εάν αυτή η ακολουθία σχηματίζει κύκλο hamilton.

Για να δείξουμε ότι το HAMILTON CYCLE είναι και πλήρες για την κλάση NP, ανάγουμε το πρόβλημα 3SAT σε αυτό. Έστω λογική έκφραση ϕ με m προτάσεις C_1, \dots, C_k , ορισμένη στο σύνολο μεταβλητών $X = \{x_1, \dots, x_n\}$. Θα κατασκευάσουμε γράφημα $G = (V, E)$ το οποίο έχει κύκλο hamilton αν και μόνο αν η ϕ είναι ικανοποιήσιμη. Η κατασκευή αυτή βασίζεται σε διάφορα υπογράφημα που πληρούν συγκεκριμένες ιδιότητες και συνθέτουν το γράφημα G (σκαριφήματα, gadgets).

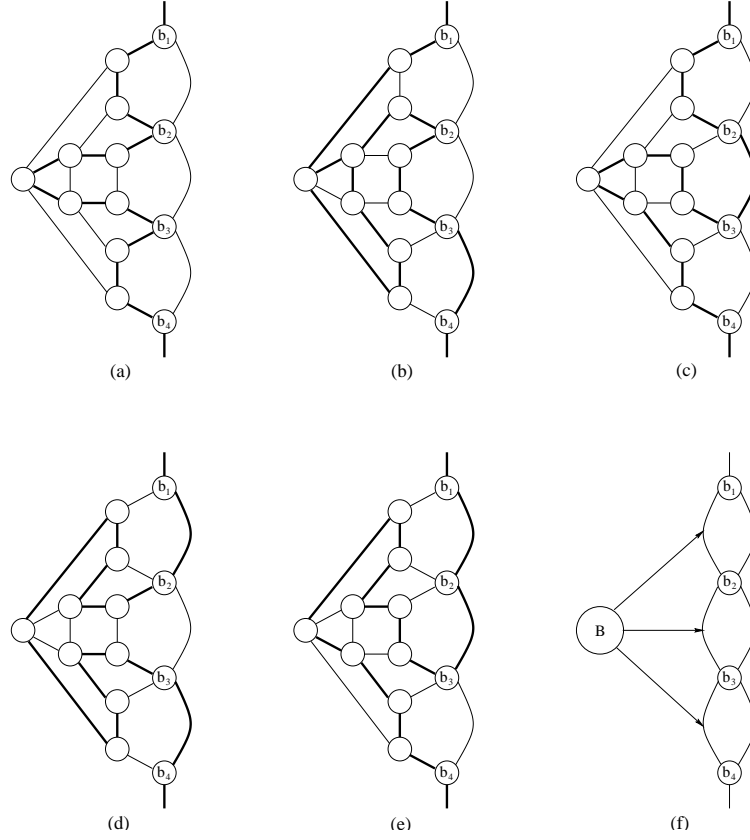
Το πρώτο σκαρίφημα παρουσιάζεται στο Σχήμα 5.5 (a). Το A θα είναι υπογράφημα του γραφήματος G το οποίο συνδέεται με το G μόνο με τους κόμβους a, a', b , και b' . Ας υποθέσουμε τώρα ότι το G έχει κύκλο hamilton. Τότε ο κύκλος hamilton θα διέρχεται υποχρεωτικά από τους κόμβους z_1, z_2, z_3 , και z_4 και μάλιστα με έναν από τους δυο τρόπους, όπως φαίνεται στα σχήματα 5.5 (b)–(c). Μπορούμε λοιπόν μεταχειριζόμαστε το υπογράφημα A σαν να είχαμε ένα ζευγάρι ακμών (a, a') και (b, b') με τον περιορισμό ο



Σχήμα 5.5: **(a)** Το σκαρίφημα A στην αναγωγή του 3SAT στο HAMILTON CYCLE. **(b)-(c)** Οι δύο μοναδικοί τρόποι με τους οποίους ο κύκλος Hamilton διαπερνά τις ακμές του A εάν το A είναι ένα υπογράφημα κάποιου γραφήματος G και συνδέεται με το G μόνο με τους κόμβους a, a', b , και b' . **(d)** Μια πιο συμπαγής μορφή του σκαριφήματος A .

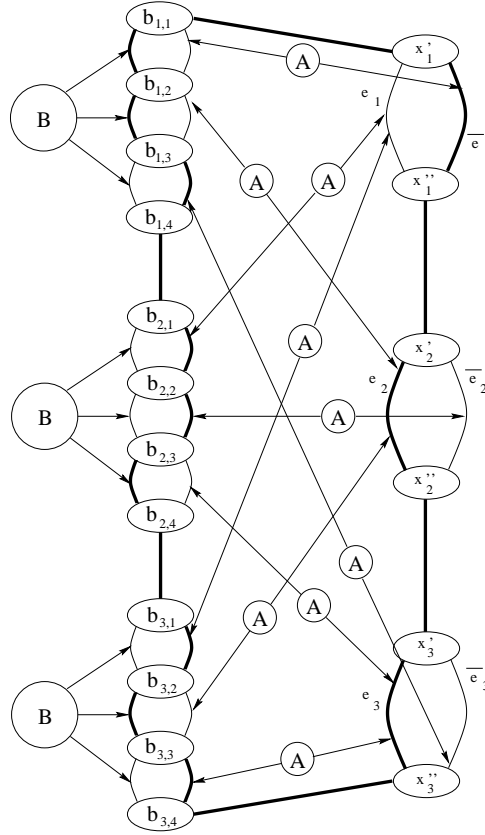
κύκλος hamilton να συμπεριλαμβάνει ακριβώς μία από τις ακμές αυτές. Θα παριστάνουμε λοιπόν το υπογράφημα A με μια πιο συμπαγή μορφή, όπως στο σχήμα 5.5 (d).

Το δεύτερο σκαρίφημα που θα χρησιμοποιήσουμε είναι το υπογράφημα B όπως φαίνεται στο Σχήμα 5.6. Υποθέτουμε ότι το B είναι ένα υπογράφημα κάποιου γραφήματος G το οποίο συνδέεται με το G μόνο με τους κόμβους b_1, b_2, b_3 , και b_4 . Έστω ότι το G έχει κύκλο hamilton. Τότε ο κύκλος hamilton δεν μπορεί να διέρχεται ταυτόχρονα από όλες τις ακμές (b_1, b_2) , (b_2, b_3) , και (b_3, b_4) διότι τότε δεν θα περνούσε από κανένα άλλο κόμβο του B . Κατά συνέπεια ο κύκλος hamilton θα διέρχεται από υποσύνολο των ακμών αυτών. Στα σχήματα 5.6 (a)–(e) βλέπουμε τους δυνατούς τρόπους που μπορεί να γίνει αυτό (υπάρχουν και άλλοι δύο ακόμα, οι συμμετρικοί των (b) και (e)). Υπάρχουν δηλαδή 7 συνολικά διαφορετικοί τρόποι με τους οποίους μπορεί ο κύκλος hamilton να περνά από τους παραπάνω κόμβους και 1 τρόπος με τον οποίο δεν μπορεί, γεγονός που μας παραπέμπει στους δυνατούς τρόπους που μπορεί να ικανοποιηθεί ή όχι μια πρόταση με τρία άτομα. Όπως θα δούμε στη συνέχεια, το υπογράφημα B θα παίξει το ρόλο μιας πρότασης. Μπορούμε να παριστάνουμε το υπογράφημα B με μια πιο συμπαγή μορφή, όπως φαίνεται στο σχήμα 5.6 (f), με την προϋπόθεση ένα τουλάχιστον από τα μονοπάτια που δείχνουν τα βέλη να συμπεριλαμβάνεται στον κύκλο hamilton.



Σχήμα 5.6: Το σκαρίφημα B στην αναγωγή του 3SAT στο HAMILTON CYCLE. (a)–(e) Οι δυνατοί τρόποι, και οι συμμετρικοί των (b) και (e), με τους οποίους ένας κύκλος hamilton μπορεί να περιέχει διέρχεται από τις ακμές (b_1, b_2) , (b_2, b_3) , και (b_3, b_4) . (f) Μια πιο συμπαγή μορφή του B . Τουλάχιστον ένα από τα μονοπάτια που δείχνουν τα βέλη πρέπει να συμπεριλαμβάνεται στον κύκλο hamilton.

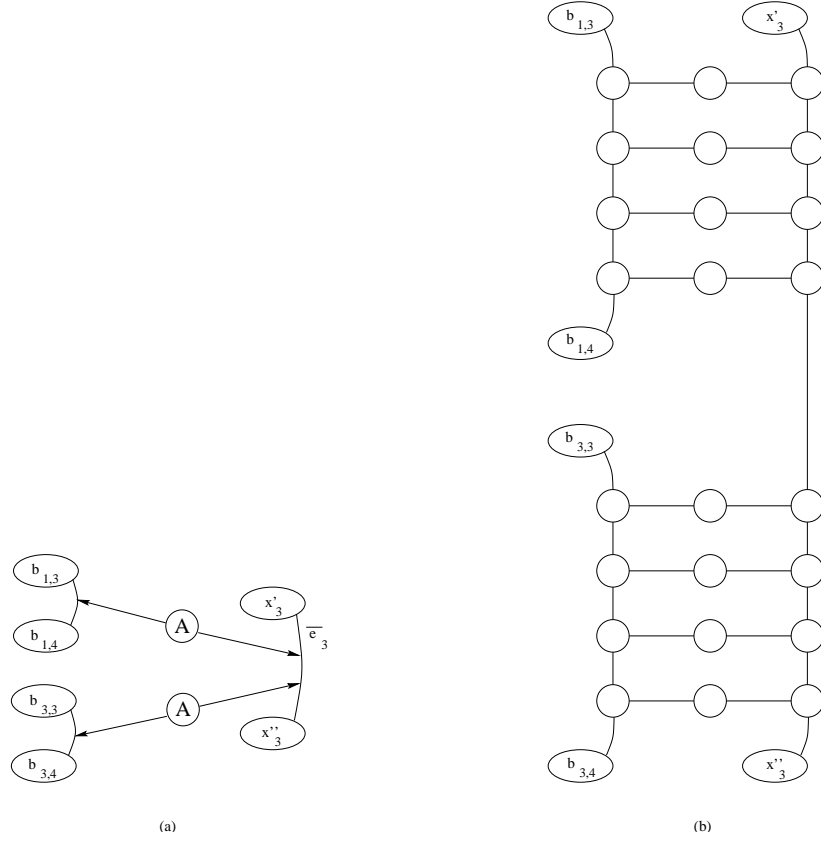
Το γράφημα G αποτελείται βασικά από αντίγραφα των υπογραφημάτων A και B (βλέπε Σχήμα 5.7 για ένα συγκεκριμένο παράδειγμα). Για κάθε πρόταση $C_i, i = 1, \dots, m$ της ϕ συμπεριλαμβάνουμε ένα αντίγραφο του B και ενώνουμε αυτά τα αντίγραφα μεταξύ τους ως εξής: Αν b_{ij} είναι ο κόμβος b_j στο i -οστό υπογράφημα B , συνδέουμε τον κόμβο $b_{i,4}$ με τον $b_{i+1,1}$, για κάθε $i = 1, 2, \dots, k-1$. Στη συνέχεια για κάθε μεταβλητή x_m της ϕ συμπεριλαμβάνουμε τους κόμβους x'_m και x''_m . Συνδέουμε τους κόμβους αυτούς μέσω δύο αντίγραφων της ακμής (x'_m, x''_m) , τις οποίες συμβολίζουμε ως e_m και \bar{e}_m . Η ιδέα είναι ότι εάν η ακμή e_m ανήκει στον κύκλο hamilton τότε η μεταβλητή x_m παίρνει τιμή 1 ενώ εάν η ακμή \bar{e}_m ανήκει στον κύκλο hamilton τότε η μεταβλητή x_m παίρνει τιμή 0. Κάθε τέτοιο ζευγάρι ακμών δημιουργούν ένα



Σχήμα 5.7: Το γράφημα G που αντιστοιχεί στη λογική έκφραση $\phi = (\neg x_1 \vee x_2 \vee \neg x_3) \wedge (x_1 \vee \neg x_2 \vee x_3) \wedge (x_1 \vee x_2 \vee \neg x_3)$. Μια αποτίμηση αληθείας t που ικανοποιεί την ϕ είναι η $t(x_1) = 0, t(x_2) = 1$ και $t(x_3) = 1$. Ο κύκλος hamilton που αντιστοιχεί στην t παρουσιάζεται με τις έντονες ακμές.

μικρό βρόχο. Συνδέουμε αυτούς τους βρόχους στη σειρά προσθέτοντας τις ακμές (x'_m, x''_{m+1}) για $m = 1, 2, \dots, n - 1$. Συνδέουμε ακόμα την αριστερή πλευρά (πρόταση) του γραφήματος με την δεξιά του πλευρά (μεταβλητή) μέσω των ακμών $(b_{1,1}, x'_1)$ και $(b_{k,4}, x''_n)$ που είναι οι πιο ακραίες ακμές του σχήματος 5.7.

Δεν έχουμε ολοκληρώσει ακόμα τη κατασκευή του γραφήματος G , αφού δεν έχουμε ακόμα συσχετίσει τις μεταβλητές με τις προτάσεις της ϕ . Αν το j -οστό άτομο της πρότασης C_i είναι το x_m τότε χρησιμοποιούμε ένα αντίγραφο του υπογραφήματος A για να συνδέσουμε την ακμή $(b_{i,j}, b_{i,j+1})$ με την ακμή e_m . Ενώ αν το j -οστό άτομο της πρότασης C_i είναι το $\neg x_m$ τότε χρησιμοποιούμε ένα αντίγραφο του A για να συνδέσουμε την ακμή $(b_{i,j}, b_{i,j+1})$ με την ακμή \bar{e}_m .



Σχήμα 5.8: Η κατασκευή που χρησιμοποιούμε στην περίπτωση όπου μια ακμή e_m ή \bar{e}_m επιρεάζεται από πολλά αντίγραφα του υπογραφήματος A .

Το ότι συνδέσαμε δυο ακμές του G μέσω των υπογραφημάτων A ουσιαστικά σημαίνει ότι αντικαταστήσαμε κάθε ακμή με τις πέντε ακμές του πάνω ή κάτω τμήματος του σχήματος 5.6 (a) και προσθέσαμε επίσης τις ακμές που διέρχονται από τους κόμβους z_1, z_2, z_3 και z_4 . Κάθε άτομο l_m είναι δυνατό να εμφανίζεται σε πολλές προτάσεις της ϕ και κατά συνέπεια μια ακμή e_m ή \bar{e}_m είναι δυνατό να επηρεάζεται από αρκετά αντίγραφα του A (π.χ. η ακμή \bar{e}_m στο σχήμα 5.7). Σε αυτή την περίπτωση συνδέουμε τα αντίγραφα του A στη σειρά, αντικαθιστώντας κατάλληλα την ακμή e_m ή την \bar{e}_m με μια σειρά από ακμές, όπως φαίνεται στο σχήμα 5.8.

Έχοντας ολοκληρώσει πλέον την κατασκευή του γραφήματος G θα αποδείξουμε ότι το γράφημα G έχει κύκλο hamilton αν και μόνο αν η λογική έκφραση ϕ είναι ικανοποιήσιμη. Αρχικά ας υποθέσουμε ότι το G έχει κύκλο hamilton και έστω h αυτός. Ο κύκλος h πρέπει να έχει ειδική μορφή:

- Περιλαμβάνει την ακμή $(b_{1,1}, x'_1)$ ενώνοντας το αριστερό τμήμα του γρα-

φήματος με το δεξί.

- Μετά περνά όλους τους κόμβους x' και x'' επιλέγοντας είτε την ακμή e_m είτε την \bar{e}_m , αλλά όχι και τις δύο.
- Στη συνέχεια διέρχεται από την ακμή $(b_{k,4}, x_n'')$ και επιστρέφει πάλι στην αριστερή πλευρά του γραφήματος.
- Τέλος, διαπερνά όλα τα αντίγραφα του υπογραφήματος B από κάτω προς τα πάνω.

Στην πραγματικότητα ο κύκλος h διαπερνά και ακμές των υπογραφημάτων A . Χρησιμοποιούμε αυτά τα υπογραφήματα για να δηλώσουμε ότι ο κύκλος h διαπερνά είτε την μία είτε την άλλη ακμή από αυτές που συνδέουν τα υπογραφήματα A .

Έχοντας λοιπόν έναν κύκλο hamilton h , ορίζουμε μια αποτίμηση αληθείας για την ϕ ως εξής: αν η ακμή e_m ανήκει στον h τότε $x_m = 1$ ενώ αν η ακμή \bar{e}_m ανήκει στον h τότε $x_m = 0$. Η αποτίμηση αυτή ικανοποιεί την ϕ . Πράγματι, αν θεωρήσουμε μια πρόταση C_i της ϕ και το αντίστοιχο υπογράφημα B του G . Κάθε ακμή $(b_{i,j}, b_{i,j+1})$ συνδέεται μέσω ενός υπογραφήματος A είτε με την ακμή e_m είτε με την \bar{e}_m , ανάλογα με τον αν το j -οστό άτομο της πρότασης είναι το x_m ή το $\neg x_m$. Η ακμή $(b_{i,j}, b_{i,j+1})$ ανήκει στον h αν και μόνο αν το αντίστοιχο άτομο έχει τιμή 0. Εφόσον κάθε μια από τις τρεις ακμές $(b_{i,1}, b_{i,2})$, $(b_{i,2}, b_{i,3})$ και $(b_{i,3}, b_{i,4})$ της πρότασης C_i ανήκουν σε ένα υπογράφημα B , δεν μπορούν και οι τρεις ταυτόχρονα να ανήκουν στον h . Μία λοιπόν από τις ακμές πρέπει να έχει ένα αντίστοιχο άτομο με τιμή 1 και έτσι η πρόταση C_i ικανοποιείται. Και αυτό ισχύει για κάθε πρόταση της ϕ . Άρα η ϕ είναι ικανοποιήσιμη.

Αντίστροφα τώρα, έστω ότι η ϕ είναι ικανοποιήσιμη από κάποια αποτίμηση αληθείας t . Μπορούμε να κατασκευάσουμε ένα κύκλο hamilton h για το γράφημα G χρησιμοποιώντας τους προηγούμενους κανόνες, ως εξής: ο h διαπερνά την ακμή e_m αν $x_m = 1$, την ακμή \bar{e}_m αν $x_m = 0$, και την ακμή $(b_{i,j}, b_{i,j+1})$ αν και μόνο αν η t αποδίδει τιμή 0 στο j -οστό άτομο της πρότασης C_i .

Όσο αφορά την κατασκευή του γραφήματος G , μπορεί να γίνει σε λογαριθμικό χώρο. Περιέχει ένα υπογράφημα B για κάθε πρόταση της ϕ , δηλαδή k το πλήθος υπογραφήματα B . Επίσης περιέχει ένα αντίγραφο του υπογραφήματος A για κάθε εμφάνιση κάθε ατόμου στην ϕ , άρα συνολικά υπάρχουν $3k$ αντίγραφα του υπογραφήματος A . Εφόσον τα υπογραφήματα A και B έχουν σταθερό μέγεθος, το γράφημα G έχει $O(k)$ κόμβους και ακμές και η κατασκευή του μπορεί να γίνει εύκολα σε λογαριθμικό χώρο. \square

Ο ΚΥΚΛΟΣ HAMILTON παραπέμπει αμέσως σε ένα διάσημο πρόβλημα, το πρόβλημα του ΠΕΡΙΟΔΕΥΟΝΤΟΣ ΠΩΛΗΤΟΥ (TRAVELING SALESMAN PROB-

LEM, TSP):

Στιγμιότυπο: Πίνακας D διαστάσεων $n \times n$ ακέραιων κοστών μεταξύ n πόλεων και ακέραιος B .

Ερώτηση: Υπάρχει μετάθεση των n πόλεων σ έτσι ώστε το κόστος της επίσκεψης και των n πόλεων με τη σειρά που υποδηλώνει η σ , από μία φορά την κάθε μια, με κατάληξη στην αρχική να είναι μικρότερο ή ίσο του B ; Δηλαδή υπάρχει σ ώστε:

$$\sum_{i=1}^{n-1} D_{\sigma(i)\sigma(i+1)} + D_{\sigma(n)\sigma(1)} \leq B$$

Το πρόβλημα αυτό βέβαια είναι γνωστότερο στην έκδοση βελτιστοποίησης στην οποία δίδεται ο πίνακας των κοστών και ζητείται η ελάχιστη κλειστή διαδρομή που επισκέπτεται και τις n πόλεις, κάθε μία από μία φορά.

Θεώρημα 5.7 *Το TSP είναι NP-πλήρες.*

Απόδειξη: Το θεώρημα αυτό είναι βασικά πόρισμα του ΚΥΚΛΟΥ HAMILTON. Δοθέντος ενός γραφήματος G κατασκευάζουμε πίνακα κοστών D θέτοντας $D_{ij} = 1$ αν η ακμή (i, j) υπάρχει στον G και $D_{ij} = 2$ αν δεν υπάρχει. Θέτουμε επίσης $B = n$. Είναι φανερό ότι αν στον G υπάρχει κύκλος hamilton, τότε η διαδρομή που υποδεικνύει ο κύκλος hamilton έχει κόστος n . Αντίστροφα, αν υπάρχει διαδρομή κόστους $B = n$ ή μικρότερη, τότε επειδή κάθε διαδρομή έχει n ακμές κάθε ακμή στην διαδρομή πρέπει να έχει κόστος 1, δηλαδή η διαδρομή είναι κύκλος hamilton στον G . \square

5.3 Άλλα προβλήματα

Ένα από τα γνωστότερα προβλήματα σε γραφήματα που όμως εντάσσεται περισσότερο στην περιοχή της Επιχειρησιακής Έρευνας είναι το πρόβλημα του ΤΑΙΡΙΑΣΜΑΤΟΣ (matching problem) [17, 24, 20]. Το πρόβλημα αυτό απαντάται με αρκετές διαφορετικές μορφές η απλούστερη από τις οποίες είναι το ΔΙΜΕΡΕΣ ΤΑΙΡΙΑΣΜΑ (bipartite matching): Δίδεται ένα διμερές γράφημα¹ 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 (και η γενικότερη έκδοση του με βάρη) λύνεται σε πολυωνυμικό χρόνο με την λεγόμενη *ουγγρική μέθοδο* [17]. Όταν

¹ένα 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

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):

Στιγμιότυπο: Δίδονται τρία ξένα σύνολα A , K και Σ με $|A| = |K| = |\Sigma|$ και υποσύνολο E του Καρτεσιανού γινομένου $A \times K \times \Sigma$.

Ερώτηση: Υπάρχει $M \subseteq E$ με $|M| = |A|$ με όλες τις τριάδες στο M ξένες ανά δύο μεταξύ τους και στις τρεις συντεταγμένες;

Θεώρημα 5.8 Το ΤΡΙΣΔΙΑΣΤΑΤΟ ΤΑΙΡΙΑΣΜΑ είναι NP-πλήρες.

Απόδειξη: Το πρόβλημα² αποδεικνύεται εύκολα ότι ανήκει στο NP. 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, \dots, x_n και m προτάσεις C_1, C_2, \dots, C_m . Θα κατασκευάσουμε ένα στιγμιότυπο του 3DM το οποίο θα έχει πλήρες ταίριασμα ανν 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):

$$\begin{aligned} T_i^+ &= \{(\bar{a}_{ij}, \kappa_{ij}, \sigma_{ij}), 1 \leq j \leq k_i\} \\ T_i^- &= \{(a_{ij}, \kappa_{ij+1}, \sigma_{ij}), 1 \leq j < k_i\} \cup \{(\bar{a}_{ik_i}, \kappa_{i1}, \sigma_{ik_i})\} \end{aligned}$$

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 \bar{a}_{ij} , $1 \leq j \leq k$. Το αποτέλεσμα είναι όλες οι μεταβλητές \bar{a}_{ij} να συμπεριφέρονται σαν αντίγραφα της x_i , ενώ οι a_{ij} σαν αντίγραφα της \bar{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 έξι τριάδες:

$$\Pi_j = \{(a_{ij}, s_j, t_j) \text{ όπου } x_i \in C_j\} \cup \{(\bar{a}_{ij}, s_j, t_j) \text{ όπου } \bar{x}_i \in C_j\}$$

²Ένας ενδιαφέρων τρόπος να ιδωθεί το πρόβλημα είναι τα σύνολα A , K και Σ να παριστάνουν σύνολα αγοριών, κοριτσιών και σπιτιών αντίστοιχα και μία τριάδα $(\alpha, \kappa, \sigma) \in E$ 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!

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 κατ' αρχή ότι μόνο μία από τις τριάδες 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 εισήχθησαν με τις παραπάνω τριάδες. Για τον σκοπό αυτό εισάγουμε για $1 \leq i \leq n$ και $1 \leq j \leq m$ τις τριάδες:

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

Έτσι αν για παράδειγμα επιλεγούν οι μεταβλητές \bar{a}_{ij} , $1 \leq j \leq k_i$ 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 τον ρόλο κάθε τριάδας) ότι υπάρχει τρισδιάστατο ταίριασμα αν και μόνο αν το στιγμιότυπο του 3SAT είναι ικανοποιήσιμο. \square

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

Στιγμιότυπο: Δίδεται σύνολο X με $3m$ στοιχεία και σύνολο C τριμελών υποσυνόλων του X .

Ερώτηση: 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' ;

Θεώρημα 5.9 Το ΚΑΛΥΜΜΑ ΑΠΟ ΤΡΙΜΕΛΗ ΣΥΝΟΛΑ (EXACT COVER BY 3-SETS) 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-πληρότητα του ΚΑΛΥΜΜΑΤΟΣ ΑΠΟ ΤΡΙΜΕΛΗ ΣΥΝΟΛΑ προκύπτει αμέσως από το ΤΡΙΣΔΙΑΣΤΑΤΟ ΤΑΙΡΙΑΣΜΑ. \square

Τελειώνουμε αυτή την ενότητα με ένα κεντρικό πρόβλημα της Θεωρίας βελτιστοποίησης που είναι η εύρεση του ελαχίστου (ή μεγίστου) μιας γραμμι-

κής συνάρτησης σε ένα n -διάστατο χώρο που ορίζεται από ένα αριθμό γραμμικών περιορισμών. Τυπικά ζητάμε το διάνυσμα $x \in \mathbb{R}^n$ έτσι ώστε:

$$\begin{aligned} \min c^T x \\ Ax \geq b \end{aligned}$$

Όπου τα c , x και b είναι διανύσματα-στήλες με διάσταση n και το A ένας πίνακας $m \times n$. Είναι γνωστό [27] ότι το παραπάνω πρόβλημα βελτιστοποίησης είναι πολυωνυμικά ισοδύναμο με ένα πρόβλημα ύπαρξης μιας λύσης σε τυποποιημένη μορφή:

$$\begin{aligned} Ax &= b \\ x &\geq 0 \end{aligned}$$

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) με τον λεγόμενο Ελλειψοειδή Αλγόριθμο για τον Γραμμικό Προγραμματισμό [27, 24].

Όταν όμως ζητήσουμε μια *ακέραια* λύση για το πρόβλημα αυτό τότε η πολυπλοκότητα του προβλήματος (που τώρα λέγεται ΑΚΕΡΑΙΟΣ ΓΡΑΜΜΙΚΟΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ), αλλάζει και γίνεται NP-πλήρες.

Θεώρημα 5.10 *Ο ΑΚΕΡΑΙΟΣ ΓΡΑΜΜΙΚΟΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ (ILP) είναι NP-πλήρες πρόβλημα.*

Απόδειξη: Το πρόβλημα αυτό είναι το πρώτο που συναντάμε (και από τα λίγα που υπάρχουν) στο οποίο το δύσκολο μέρος της απόδειξης είναι ότι ανήκει στο NP. Πραγματικά η προσφιλής μας 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 ναδειχθεί η ύπαρξη καλού πιστοποιητικού πρέπει να χρησιμοποιηθεί ένα θεώρημα για τον ILP [22] το οποίο λέγει 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 πληρότητας είναι πολύ εύκολη επειδή το ILP είναι ιδιαίτερα εκφραστικό πρόβλημα και περίπου κάθε NP-πλήρες πρόβλημα ανάγεται σ' αυτό. Για παράδειγμα το 3SAT ανάγεται στο ILP εισάγοντας μια ακέραιη μεταβλητή y_i για κάθε λογική μεταβλητή x_i με τους περιορισμούς $0 \leq y_i \leq 1$. Είναι προφανές ότι αυτοί επιβάλλουν 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 επιβάλλει τουλάχιστον ένα άτομο της C να είναι αληθές και άρα η C να ικανοποιείται. \square

5.4 Ασθενής NP-πληρότητα

Ας δούμε τώρα ένα πρόβλημα στο οποίο σημαντικό ρόλο στην διατύπωση του αλλά και στην δυσκολία του φαίνεται να παίζουν οι αριθμοί.

Στιγμιότυπο: Δίδεται σύνολο S n θετικών ακεραίων και επιπλέον ακέραιος B .

Ερώτηση: Υπάρχει υποσύνολο $S' \subseteq S$ έτσι ώστε το άθροισμα των ακεραίων στο S' να είναι ακριβώς B ;

Το πρόβλημα αυτό λέγεται ΑΘΡΟΙΣΜΑ ΥΠΟΣΥΝΟΛΟΥ (SUBSET SUM) και για αυτό ισχύει:

Θεώρημα 5.11 *Το ΑΘΡΟΙΣΜΑ ΥΠΟΣΥΝΟΛΟΥ) είναι NP-πλήρες.*

Απόδειξη: Είναι εύκολο να δούμε ότι το πρόβλημα είναι στο NP. Μαντεύουμε ένα υποσύνολο S' και επιβεβαιώνουμε ότι το άθροισμα των ακεραίων σε αυτό είναι ακριβώς B .

Η απόδειξη πληρότητας γίνεται με αναγωγή του X3C που περιγράψαμε παραπάνω. Ας θεωρήσουμε ένα στιγμιότυπο του X3C δηλαδή σύνολο C με $|C| = n$ από τριμελή υποσύνολα ενός συνόλου X με $|X| = 3m$. Θα κατασκευάσουμε n ακεραίους και ένα ακέραιο επιπλέον B έτσι ώστε να υπάρχει υποσύνολο των ακεραίων αθροιζόμενο στο B ανν το X3C έχει λύση.

Ένας χρήσιμος τρόπος να παραστήσουμε ένα από τα τριμελή σύνολα του C είναι με την βοήθεια ενός διανύσματος-δείκτη με διάσταση $3m$, όσα είναι τα στοιχεία του X . Το διάνυσμα αυτό έχει ακριβώς τρία 1 και όλα τα άλλα στοιχεία του 0. Αν τα 1 είναι στις θέσεις i , j και k , εννοούμε ότι τα στοιχεία x_i , x_j και x_k του X απαρτίζουν το συγκεκριμένο τριμελές σύνολο. Έτσι το στιγμιότυπο X3C μπορεί να παρασταθεί με ένα σύνολο $3m$ -διάστατων διανυσμάτων με συντεταγμένες 0-1. Στο Σχήμα 5.9 φαίνεται ένα παράδειγμα με $|X| = 12$ και $|C| = 8$. Τοποθετώντας τα διανύσματα σε ένα πίνακα έτσι ώστε οι αντίστοιχες συντεταγμένες να βρίσκονται στην ίδια στήλη, η λύση του X3C είναι ένα υποσύνολο των γραμμών στο οποίο σε κάθε στήλη υπάρχει ακριβώς ένα 1. Στο Σχήμα 5.9 μια λύση φαίνεται σημειωμένη με *.

Αυτή η παρατήρηση γεννά την ακόλουθη σκέψη: Αν θεωρήσουμε τις γραμμές σαν αριθμούς με $3m$ ψηφία, ζητάμε άραγε ένα υποσύνολο των αριθμών που να αθροίζονται στον αριθμό $11 \cdots 1$, με $3m$ μονάδες; Όχι ακριβώς. Αν μεν έχουμε μια λύση, τότε βέβαια το άθροισμα των αντίστοιχων αριθμών είναι ο αριθμός $11 \cdots 1$ με $3m$ μονάδες. Το αντίστροφο όμως δεν ισχύει. Αν οι αριθμοί μας είναι για παράδειγμα δυαδικοί, τότε σε μια στήλη είναι δυνατό να έχουμε μερικό άθροισμα 1 γιατί σε αυτή υπάρχουν τρία 1 εξαιτίας δε της ύπαρξης κρατουμένου το πλήρες άθροισμα μπορεί να γίνει $11 \cdots 1$ χωρίς να υπάρχει μόνο ένα 1 σε κάθε στήλη. Για παράδειγμα οι δυαδικοί αριθμοί 0111, 0110 και

0	0	1	0	0	0	1	0	1	0	0	0	*
1	0	0	0	0	0	0	1	0	1	0	0	
0	1	0	0	0	1	0	0	1	0	0	0	
1	0	0	1	1	0	0	0	0	0	0	0	*
0	0	0	0	0	0	1	0	1	0	1	0	
0	1	0	0	0	0	0	1	0	0	0	1	*
0	0	0	0	0	1	0	0	0	1	1	0	*
0	0	1	0	1	0	0	0	0	0	0	1	

Σχήμα 5.9: Παράσταση ενός στιγμιοτύπου του προβλήματος X3C μέσω δυαδικών διανυσμάτων.

0010 αθροίζονται στο 1111. Πως μπορούμε λοιπόν να αποφύγουμε την ύπαρξη κρατουμένου; Κατ' αρχή αν δούμε τους αριθμούς μας σαν δεκαδικούς τότε μέχρι εννιά αριθμοί μπορεί να προστεθούν χωρίς την δημιουργία κρατουμένου εφόσον σε κάθε στήλη θα υπάρχουν μέχρι εννιά 1. Για να μπορέσουμε κατά συνέπεια να αθροίσουμε μέχρι n αριθμούς χωρίς να δημιουργηθεί κρατούμενο, πρέπει οι αριθμοί μας να θεωρηθούν σαν γραμμένοι στο σύστημα με βάση $n+1$. Έτσι για παράδειγμα η πρώτη γραμμή του παραδείγματός μας πρέπει να παριστάνει τον αριθμό $(n+1)^3 + (n+1)^5 + (n+1)^9$. Γενικότερα αν τα σύνολα στο C δοθούν σαν σύνολα C_1, C_2, \dots, C_n , όπου κάθε C_i είναι σύνολο τριών ακεραίων στο διάστημα $[0, 3m-1]$, τότε οι αριθμοί του στιγμιοτύπου του SUBSET SUM που κατασκευάζουμε είναι οι $v_i = \sum_{j \in C_i} (n+1)^j, i = 1, 2, \dots, n$. Το ζητούμενο άθροισμα B είναι βέβαια $B = \sum_{j=0}^{3m-1} (n+1)^j$ δηλαδή ο αριθμός $11 \dots 1$ με $3m$ μονάδες στο σύστημα με βάση $n+1$. Είναι φανερό από τα παραπάνω ότι το σύνολο των n αριθμών που κατασκευάσαμε έχει υποσύνολο που αθροίζεται στον αριθμό B ανν το δοθέν στιγμιότυπο του X3C έχει απάντηση «ΝΑΙ». □

Το ενδιαφέρον στοιχείο στην παραπάνω απόδειξη είναι οι εκθετικά μεγάλοι αριθμοί που αναγκαστήκαμε να κατασκευάσουμε. Είναι άραγε απαραίτητοι τόσο μεγάλοι αριθμοί ή μήπως κάποια καλύτερη αναγωγή θα μπορούσε να γίνει στην οποία να κατασκευάζαμε στιγμιότυπο με αριθμούς πολυωνυμικά φραγμένους στο μέγεθος του στιγμιοτύπου; Κάτι τέτοιο φαίνεται πολύ απίθανο: Θα δείξουμε στην συνέχεια ότι το ΑΘΡΟΙΣΜΑ ΥΠΟΣΥΝΟΛΟΥ ανήκει σε μία κατηγορία NP-πλήρων προβλημάτων που μέρος τουλάχιστον της δυσκολίας τους οφείλεται στους μεγάλους αριθμούς. Για να το δείξουμε αυτό θα ακολουθήσουμε την αντίστροφη οδό: Θα παρουσιάσουμε ένα αλγόριθμο για το SUBSET SUM (μάλιστα, για ένα γενικότερο πρόβλημα) ο οποίος είναι πολυωνυμικός όταν οι αριθμοί του στιγμιοτύπου είναι πολυωνυμικοί.

Το πρόβλημα του ΣΑΚΙΔΙΟΥ (KNAPSACK problem) είναι το ακόλουθο:

Στιγμιότυπο: Δίδεται σύνολο S , n αντικείμενων όπου το i -οστό αντικείμενο έχει αξία v_i και βάρος w_i , με $v_i, w_i \in \mathbb{Z}^+$, καθώς και δύο επιπλέον θετικοί ακέραιοι B και W .

Ερώτηση: Υπάρχει υποσύνολο $S' \subseteq S$ των αντικειμένων έτσι ώστε το συνολικό τους βάρος να είναι μικρότερο ή ίσο του W , ενώ η συνολική τους αξία μεγαλύτερη ή ίση του B ;

Όπως και στην περίπτωση του ΠΕΡΙΟΔΕΥΟΝΤΟΣ ΠΩΛΗΤΟΥ, το πρόβλημα αυτό τίθεται φυσικότερα στην έκδοση βελτιστοποίησης: Ζητείται υποσύνολο των αντικειμένων με βάρος μικρότερο από το W (την αντοχή του σακιδίου) αλλά με την μεγαλύτερη δυνατή αξία.

Θεώρημα 5.12 Το πρόβλημα του ΣΑΚΙΔΙΟΥ είναι NP-πλήρες.

Απόδειξη: Το πρόβλημα αποδεικνύεται εύκολα στο NP. Για να αποδείξουμε την πληρότητα ας παρατηρήσουμε ότι η ειδική περίπτωση του στην οποία αριθμητικά το βάρος κάθε αντικειμένου είναι ίσο με την αξία του, $v_i = w_i, i = 1, 2, \dots, n$ και $B = W$, ζητά ένα υποσύνολο S' των αντικειμένων έτσι ώστε $\sum_{i \in S'} w_i \leq W$ και $\sum_{i \in S'} v_i \geq B$, ή αλλιώς $\sum_{i \in S'} v_i = B$. Πρόκειται δηλαδή για ένα στιγμιότυπο του ΑΘΡΟΙΣΜΑΤΟΣ ΥΠΟΣΥΝΟΛΟΥ που δείχθηκε NP-πλήρες στο Θεώρημα 5.11. \square

Φυσικά, μια και η απόδειξη NP-πληρότητας για το ΣΑΚΙΔΙΟ έγινε με αναγωγή του ΑΘΡΟΙΣΜΑΤΟΣ ΥΠΟΣΥΝΟΛΟΥ, η παρατήρηση που έγινε παραπάνω για τους μεγάλους αριθμούς ισχύει και εδώ.

Προχωρούμε τώρα στην παρουσίαση ενός αλγορίθμου για το πρόβλημα του ΣΑΚΙΔΙΟΥ (και μάλιστα στην γενικότερη έκδοση βελτιστοποίησης) με ενδιαφέρουσες ιδιότητες. Η βασική δομή που θα χρησιμοποιήσουμε είναι ένας διδιάστατος πίνακας $W \times n$, έστω V (βλ. Αλγόριθμο 6).

Ας δούμε τώρα πως εργάζεται ο αλγόριθμος αυτός. Η σημασία του στοιχείου $V[x, i]$ είναι η εξής: το $V[x, i]$ είναι η μεγαλύτερη αξία που μπορούμε να έχουμε από κάποιο υποσύνολο των i πρώτων αντικειμένων με βάρος (του υποσυνόλου) ακριβώς x . Προφανώς $V[x, 0] := 0$ για κάθε x και αυτό λέγει η πρώτη γραμμή του αλγορίθμου. Στην συνέχεια όλος ο αλγόριθμος βασίζεται στην ακόλουθη παρατήρηση: Η $V[x, i + 1]$ μπορεί να οφείλεται σε ένα υποσύνολο των $i + 1$ πρώτων αντικειμένων που είτε περιλαμβάνει το στοιχείο $i + 1$ είτε όχι. Αν δεν το περιλαμβάνει τότε βέβαια $V[x, i + 1] = V[x, i]$. Αν όμως το περιλαμβάνει τότε στο βάρος του υποσυνόλου περιλαμβάνεται και το βάρος του στοιχείου $i + 1$. Από τα υπόλοιπα i στοιχεία συνεπώς θέλουμε το $V[x - w_{i+1}, i]$ ενώ έχει δωθεί και η αξία του, v_{i+1} . Η $V[x, i + 1]$ συνεπώς είναι η μεγαλύτερη από τις δύο παραπάνω ποσότητες και αυτό λέγει η σχετική γραμμή του αλγορίθμου. Όταν όλες οι θέσεις του πίνακα συμπληρωθούν, το μόνο που απομένει είναι να βρούμε αυτή με τη μεγαλύτερη αξία.

```

Input:  $n$  ζευγάρια θετικών ακεραίων  $v_i$  και  $w_i$  και θετικοί ακέραιοι  $B$  και  $W$ .
procedure knapsack
for  $x := 0$  to  $W$  do
     $V[x, 0] := 0$ ;
end for
for  $i := 0$  to  $n - 1$  do
    for  $x := 0$  to  $W$  do
         $V[x, i + 1] = \max\{V[x, i], v_{i+1} + V[x - w_{i+1}, i]\}$ ;
        {Βρες το μεγαλύτερο στοιχείο του πίνακα  $V$ , έστω  $K$ }
    end for
end for
return  $K$ ;

```

Αλγόριθμος 6: Αλγόριθμος για το πρόβλημα του ΣΑΚΙΔΙΟΥ.

Ο αλγόριθμος αυτός ανήκει σε μία γενικότερη μέθοδο που ονομάζεται *δυναμικός προγραμματισμός* [24] με βασικό χαρακτηριστικό την εύρεση της καλύτερης λύσης σαν συνδυασμό των λύσεων πολλών επιμέρους προβλημάτων.

Ο χρόνος αυτού του αλγορίθμου είναι βέβαια $O(nW)$, μια πολυωνυμική συνάρτηση στα n και W κάτι που σε συνδυασμό με το Θεώρημα 5.12 φαίνεται να οδηγεί στο $P=NP$! Όμως το πρόβλημα είναι ότι το W είναι μία ποσότητα που δεν είναι φραγμένη πολυωνυμικά στο μέγεθος του προβλήματος. Αν η παράσταση των αριθμών είναι (όπως συνήθως συμβαίνει) σε δυαδική (ή δεκαδική) μορφή, τότε το μέγεθος των αριθμών είναι εκθετικά μεγαλύτερο του αριθμού των ψηφίων τους και άρα του μήκους του στιγμιότυπου του προβλήματος. Παρόλα αυτά ο Αλγόριθμος 6 αποδεικνύει το:

Θεώρημα 5.13 *Το πρόβλημα του ΣΑΚΙΔΙΟΥ είναι πολυωνυμικό για τα στιγμιότυπα στα οποία οι αριθμοί είναι μικροί, δηλαδή με μέγεθος πολυωνυμικά φραγμένο με το μέγεθος του προβλήματος.*

Απόδειξη: Άμεσα από τον Αλγόριθμο 6. □

Έστω I ένα στιγμιότυπο οποιουδήποτε προβλήματος. Συμβολίζουμε με $\max(I)$ τον μεγαλύτερο αριθμό που εμφανίζεται στο στιγμιότυπο I και με $\ell(I)$ το μήκος του I στην παράσταση που χρησιμοποιούμε.

Ορισμός 5.5 Ένα πρόβλημα λέγεται αριθμητικό αν δεν υπάρχει πολώνυμο p έτσι ώστε $\max(I) \leq p(\ell(I))$ για όλα τα στιγμιότυπα I του προβλήματος.

Ο ορισμός αυτός λέγει ότι αν η διατύπωση του προβλήματος είναι τέτοια που να εμφανίζονται σε αυτό αυθαίρετα μεγάλοι αριθμοί χωρίς ανάλογη επιβάρυνση

στο μέγεθος της συμβολοσειράς που το κωδικοποιεί, τότε χαρακτηρίζεται σαν «αριθμητικό». Το ΣΑΚΙΔΙΟ και ο ΠΕΡΙΟΔΕΥΩΝ ΠΩΛΗΤΗΣ είναι ασφαλώς τέτοια προβλήματα. Αντίθετα το SAT ή το CLIQUE στην διατύπωση τους μπορεί μεν να έχουν αυθαίρετα μεγάλους αριθμούς (τους δείκτες των μεταβλητών και των κόμβων του γραφήματος αντίστοιχα), όμως το μήκος της συμβολοσειράς που κωδικοποιεί τα στιγμιότυπα είναι ανάλογο του μεγαλύτερου αριθμού (μια και κάθε μεταβλητή στο SAT ή κόμβος στο CLIQUE εμφανίζεται τουλάχιστον μία φορά)³. Υπάρχει όμως μια σημαντική διαφορά μεταξύ του ΣΑΚΙΔΙΟΥ και του ΠΕΡΙΟΔΕΥΟΝΤΟΣ ΠΩΛΗΤΟΥ. Η απόδειξη NP-πληρότητας του ΠΕΡΙΟΔΕΥΟΝΤΟΣ ΠΩΛΗΤΟΥ στο Θεώρημα 5.7 χρησιμοποίησε μόνο μικρούς αριθμούς (1 και 2) σε αντίθεση με το πρόβλημα του ΣΑΚΙΔΙΟΥ. Ο παρακάτω ορισμός επισημαίνει αυτή τη διαφορά:

Ορισμός 5.6 Ένα αριθμητικό πρόβλημα λέγεται NP-πλήρες με την ισχυρή έννοια (ισχυρά NP-πλήρες) αν παραμένει NP-πλήρες και για τα στιγμιότυπα I όπου $\max(I) \leq p(\ell(I))$, δηλαδή για τα στιγμιότυπα όπου οι εμφανιζόμενοι αριθμοί είναι πολυωνυμικά φραγμένοι στο μέγεθος του στιγμιότυπου.

Το TSP είναι λοιπόν NP-πλήρες με την ισχυρή έννοια ενώ το KNAPSACK είναι ασθενώς NP-πλήρες.

Ορισμός 5.7 Ένας αλγόριθμος είναι ψευδοπολυωνυμικός αν η πολυπλοκότητα χρόνου του είναι $O(p(\max(I), \ell(I)))$ είναι δηλαδή πολυωνυμικός τόσο στο μέγεθος της εισόδου όσο και στο μέγεθος του μεγαλύτερου εμφανιζόμενου αριθμού.

Ο Αλγόριθμος 6 είναι ένας ψευδοπολυωνυμικός αλγόριθμος για το KNAPSACK. Ισχύει προφανώς:

Παρατήρηση 5.1 Ένα ισχυρά NP-πλήρες πρόβλημα δεν έχει ψευδοπολυωνυμικό αλγόριθμο εκτός αν $P=NP$.

Η απόδειξη είναι εύκολη γιατί από τον ορισμό αν υπήρχε ψευδοπολυωνυμικός αλγόριθμος θα είχε πολυπλοκότητα $O(p(\max(I), \ell(I)))$ για κάποιο πολώνυμο p ακόμα και για στιγμιότυπα I για τα οποία $\max(I) \leq g(\ell(I))$ όπου g ένα άλλο πολώνυμο. Όμως η σύνθεση πολωνύμων είναι επίσης πολώνυμο και συνεπώς έχουμε πολυωνυμικό αλγόριθμο για ένα NP-πλήρες πρόβλημα.

Οι ψευδοπολυωνυμικοί αλγόριθμοι είναι πολλές φορές και αποτελεσματικοί στην πράξη και σαν τέτοιοι αποτελούν, αν υπάρχουν, ιδιαίτερα επιθυμητή λύση για ένα NP-πλήρες πρόβλημα.

³Υποθέτουμε εδώ μια «λογική» παράσταση των στιγμιότυπων, όπου ένας δείκτης χρησιμοποιείται αν όλοι οι μικρότεροι του έχουν ήδη χρησιμοποιηθεί

5.5 Αποδεικνύοντας NP-πληρότητα

Στην ενότητα αυτή θα συνοψίσουμε τις τεχνικές που χρησιμοποιήσαμε στις αποδείξεις NP-πληρότητας που παραθέσαμε. Η ανάπτυξη είναι σύμφωνα με το [6]. Σπεύδουμε όμως να σημειώσουμε ότι δεν υπάρχουν μέθοδοι και «συνταγές» για τον σκοπό αυτό. Οι αποδείξεις NP-πληρότητας είναι μάλλον τέχνη και χρειάζονται διαίσθηση και δημιουργικότητα ώστε να περιγραφεί στην «γλώσσα» ενός προβλήματος ένα άλλο πρόβλημα που μπορεί να είναι εντελώς διαφορετικό.

Μια εμφανής κατηγοριοποίηση των αποδείξεων πάντως, κατά σειρά δυσκολίας είναι η ακόλουθη.

Ειδική περίπτωση. Πολλές φορές συμβαίνει το προς απόδειξη πρόβλημα να είναι μια γενίκευση ενός ήδη γνωστού NP-πλήρους προβλήματος (και άρα το τελευταίο να είναι ειδική περίπτωση του πρώτου). Σε τέτοια περίπτωση το πρόβλημα μας κληρονομεί την δυσκολία της ειδικής του περίπτωσης και είναι προφανώς NP-δύσκολο. Το μόνο που απομένει είναι η απόδειξη ότι ανήκει στο NP, κάτι που τις περισσότερες φορές είναι εύκολο. (Προσοχή όμως σε προβλήματα όπως το ILP, βλ. Θεώρημα 5.10). Χρησιμοποιήσαμε αυτή την τεχνική στις αποδείξεις για παράδειγμα του MAXSAT και του KNAPSACK.

Τοπική αντικατάσταση. Στις αποδείξεις αυτές αντικαθιστούμε στοιχεία του γνωστού NP-πλήρους προβλήματος με απλές κατασκευές που περιγράφουν την λειτουργία του στοιχείου τοπικά δηλαδή αποκομμένα σε κάποιο βαθμό από το υπόλοιπο πρόβλημα. Αυτή ήταν η τεχνική μας στην απόδειξη του 3SAT του MON3SAT και του TSP.

Σχεδιασμός στοιχείων. Είναι σαφώς ο γενικότερος και δυσκολότερος τρόπος απόδειξης. Απαιτεί τον σχεδιασμό περίπλοκων πολλές φορές κατασκευών που κάθε μία έχει εξειδικευμένο ρόλο στην απόδειξη και συνήθως συνδυάζεται η λειτουργία τους με τρόπο όχι τετριμμένο. Οι κατασκευές στο HAMILTON CYCLE το SUBSET SUM και το 3DM ήταν βέβαια αυτού του τύπου.

Ένα άλλο ζήτημα που αντιμετωπίζεται στην πράξη είναι η επιλογή του κατάλληλου NP-πλήρους προβλήματος για αναγωγή. Το βιβλίο των M. Garey και D. Johnson [6] περιέχει εκατοντάδες γνωστά NP-πλήρη προβλήματα τα οποία μπορεί να είναι καλή αφετηρία για απόδειξη. Ο κατάλογος αυτός από την εμφάνιση του [6], συμπληρώνεται σε τακτά χρονικά διαστήματα στο [12] από το 1981 και εφεξής. Πάντως τις περισσότερες φορές, είναι λάθος να αναζητείται το πρόβλημα που «μοιάζει» περισσότερο με το προς απόδειξη. Πολλά προβλήματα που «μοιάζουν» έχουν αποδείξεις εντελώς διαφορετικές. Για αυτό η καλύτερη επιλογή είναι η πρώτη τουλάχιστον προσπάθεια για απόδειξη να γίνει με αναγωγή του 3SAT.

Σε κάθε περίπτωση όμως είναι σημαντικό να κατανοηθεί που βρίσκεται η δυσκολία του προς απόδειξη προβλήματος, προτού επιχειρηθεί η αναγωγή. Η

γνώση της δυσκολίας αυτής επιτρέπει την εκμετάλλευση της εκφραστικής δύναμης του προβλήματος ώστε να μπορέσουμε να το εκμεταλλευτούμε και να περιγράψουμε το ήδη γνωστό NP-πλήρες πρόβλημα. Για παράδειγμα στην απόδειξη του HAMILTON CYCLE το γράφημα A περιγράφει την δυαδικότητα που έχουν οι λογικές μεταβλητές. Μόνο αφού κατανοήσουμε ότι το HAMILTON CYCLE μπορεί να εκφράσει αυτή τη δυαδικότητα μπορούμε να προχωρήσουμε στον σχεδιασμό στοιχείων σαν το γράφημα A . Για την κατανόηση αυτή είναι τις περισσότερες φορές απαραίτητο να προηγηθεί μια προσπάθεια εύρεσης πολυωνυμικού αλγορίθμου και να κατανοηθεί η αιτία της αποτυχίας της προσπάθειας ώστε να στραφούμε στην απόδειξη NP-πληρότητας. Τις περισσότερες φορές μάλιστα για μια απόδειξη απαιτούνται πολλές διαδοχικές προσπάθειες εύρεσης πολυωνυμικού αλγορίθμου, στην συνέχεια προσπάθεια αναγωγής και πάλι πίσω μέχρι να κατανοηθεί πλήρως η δυσκολία του προβλήματος. Η τακτική αυτή είναι απαραίτητη αν προσπαθούμε να βρούμε την πολυπλοκότητα ενός καινούργιου προβλήματος για το οποίο δεν έχουμε γνωρίζουμε ότι είναι NP-πλήρες.

5.6 Ασκήσεις

1. Αποδείξτε ότι το πρόβλημα ΙΣΟΜΟΡΦΙΣΜΟΣ ΥΠΟΓΡΑΦΗΜΑΤΟΣ (SUBGRAPH ISOMORPHISM) είναι NP-πλήρες:

Στιγμιότυπο: Μη κατευθυντικά γράφημα $G = (V_1, E_1)$ και $H = (V_2, E_2)$.

Ερώτηση: Περιέχει το G υπογράφημα ισομορφικό του H ;

Θα λέμε ότι το γράφημα G περιέχει υπογράφημα 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 μια ένα-προς-ένα απεικόνιση $f : V_2 \rightarrow V$ τέτοια ώστε $(u, v) \in E_2$ αν και μόνο αν $(f(u), f(v)) \in E$.

2. Αποδείξτε ότι το πρόβλημα ΣΠΟΝΔΥΛΩΤΟ ΔΕΝΔΡΟ ΠΕΡΙΟΡΙΣΜΕΝΟΥ ΒΑΘΜΟΥ (DEGREE CONSTRAINED SPANNING TREE) είναι NP-πλήρες:

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. Αποδείξτε ότι το πρόβλημα ΣΥΝΟΛΟ ΚΡΟΥΣΗΣ (HITTING SET) είναι NP-πλήρες:

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. Αποδείξτε ότι το πρόβλημα NOT-ALL-EQUAL-3SAT είναι NP-πλήρες:

Στιγμιότυπο: Στιγμιότυπο του 3SAT.

Ερώτηση: Υπάρχει αποτίμηση στις λογικές μεταβλητές έτσι ώστε κάθε πρόταση να έχει το λιγώτερο ένα αληθές και το λιγώτερο ένα ψευδές άτομο;

5. Αποδείξτε ότι το πρόβλημα ΜΕΓΙΣΤΗ 2-ΙΚΑΝΟΠΟΙΗΣΙΜΟΤΗΤΑ (MAX-2SAT) είναι NP-πλήρες:

Στιγμιότυπο: Λογική έκφραση σε CNF όπου κάθε πρόταση έχει δύο άτομα (όπως στο 2SAT) και ακέραιος k .

Ερώτηση: Υπάρχει αποτίμηση στις μεταβλητές που να ικανοποιεί το λιγώτερο k προτάσεις;

Υπόδειξη: Ανάγετε το 3SAT. Συγκρίνετε το αποτέλεσμα αυτό με το Θεώρημα 4.5 και τον πολυωνυμικό αλγόριθμο που προκύπτει για το 2SAT.

6. Αποδείξτε ότι το πρόβλημα ΤΟΜΗ ΣΥΝΟΛΟΥ (SET SPLITTING) είναι NP-πλήρες:

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 ;

7. Αποδείξτε ότι το πρόβλημα ΚΥΡΙΑΡΧΟ ΣΥΝΟΛΟ (DOMINATING SET) είναι NP-πλήρες:

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 $u \in V \setminus V'$ υπάρχει $v \in V'$ για τον οποίο $(u, v) \in E$.

8. Αποδείξτε ότι το πρόβλημα K-ΧΡΩΜΑΤΙΚΟΤΗΤΑ (K-COLORABILITY) είναι NP-πλήρες:

Στιγμιότυπο: Μη κατευθυντικό γράφημα $G = (V, E)$ και θετικός ακέραιος $k \leq |V|$.

Ερώτηση: Είναι το G k -χρωματικό;

Ένα γράφημα $G = (V, E)$ είναι k -χρωματικό αν υπάρχει απεικόνιση $f : V \rightarrow \{1, 2, \dots, k\}$ τέτοια ώστε $f(u) \neq f(v)$ για κάθε $(u, v) \in E$.

9. Αποδείξτε ότι το πρόβλημα FEEDBACK NODE SET είναι NP-πλήρες:

Στιγμιότυπο: Κατευθυντικό γράφημα $G = (V, E)$ και θετικός ακέραιος $k \leq |V|$.

Ερώτηση: Υπάρχει υποσύνολο κόμβων $V' \subseteq V$ με $|V'| \leq k$ το οποίο να περιέχει τουλάχιστον έναν κόμβο από κάθε κατευθυντικό κύκλο του G ;

10. Αποδείξτε ότι το πρόβλημα ΜΟΝΟΠΑΤΙ HAMILTON (HAMILTON PATH) είναι NP-πλήρες:

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 ;

Ένα μονοπάτι hamilton ενός γραφήματος $G = (V, E)$ από τον u στον v είναι ένα απλό μονοπάτι με άκρα τους κόμβους u και v που διέρχεται από κάθε κόμβο του G .

11. Αποδείξτε ότι το πρόβλημα ΜΕΓΙΣΤΗ ΤΟΜΗ (MAX CUT) είναι NP-πλήρες:

Στιγμιότυπο: Μη κατευθυντικό γράφημα $G = (V, E)$, βάρος (θετικός ρητός) $w(e)$ για κάθε $e \in E$ και θετικός ακέραιος k .

Ερώτηση: Υπάρχει διαμέριση του V 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 ;

12. Αποδείξτε ότι το πρόβλημα SET PACKING είναι NP-πλήρες:

Στιγμιότυπο: Σύνολο C πεπερασμένων συνόλων και θετικός ακέραιος k

Ερώτηση: Υπάρχουν στο C το λιγότερο k σύνολα ξένα ανά δύο μεταξύ τους.

Υπόδειξη: Ανάγετε το $X3C$.

13. Αποδείξτε ότι το πρόβλημα της ΔΙΑΜΕΡΙΣΗΣ (PARTITION) είναι NP-πλήρες:

Στιγμιότυπο: Πεπερασμένο σύνολο A και μέγεθος $s(a) \in \mathbb{Z}^+$.

Ερώτηση: Υπάρχει υποσύνολο $A' \subseteq A$ τέτοιο ώστε $\sum_{a \in A'} s(a) = \sum_{a \in A \setminus A'} s(a)$;

14. Αποδείξτε ότι το πρόβλημα 3-ΔΙΑΜΕΡΙΣΜΟΣ (3-PARTITION) είναι NP-πλήρες:

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: Ανάγεται το 3DM. Το πρόβλημα είναι ισχυρά NP-πλήρες.

15. Αποδείξτε ότι το πρόβλημα ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ ΠΟΛΛΩΝ ΕΠΕΞΕΡΓΑΣΤΩΝ (MULTIPROCESSOR SCHEDULING) είναι NP-πλήρες:

Στιγμιότυπο: Σύνολο T εργασιών, αριθμός $m \in \mathbb{Z}^+$ επεξεργαστών, διάρκεια $\ell(t) \in \mathbb{Z}^+$ για κάθε εργασία και προθεσμία $D \in \mathbb{Z}^+$.

Ερώτηση: Υπάρχει προγραμματισμός των m επεξεργαστών έτσι ώστε όλες οι εργασίες να τελειώσουν πριν την προθεσμία D ;

Προγραμματισμός εργασιών είναι μία συνάρτηση $\sigma : T \rightarrow \mathbb{Z}_0^+$, έτσι ώστε για κάθε $u \geq 0$, ο αριθμός των εργασιών $t \in T$ για τις οποίες $\sigma(t) \leq u \leq \sigma(t) + \ell(t)$ είναι μικρότερος ή ίσος του m και έτσι ώστε για κάθε $t \in T$, $\sigma(t) + \ell(t) \leq D$.

Βιβλιογραφία

- [1] A. V. Aho, J. E. Hopcroft and J. D. Ullman. *The Design and Analysis of Computer Algorithms*. Addison-Wesley, 1974.
- [2] J. L. Balcázar, J. Diaz and J. Gabarró. *Structural Complexity vols. I and II* Springer Verlag, Berlin 1988.
- [3] A. Borodin. *Computational complexity and the existence of complexity gaps*. J. ACM 19: 1, 158-174, 1972.
- [4] S. A. Cook. *The complexity of theorem-proving procedures*. Proc. of the 3rd IEEE Symposium on the Foundations of Computer Science, 151-158, 1971.
- [5] T. H. Cormen, C. E. Leiserson, and R. L. Rivest. *Introduction to Algorithms*. MIT Press, 1990.
- [6] M. R. Garey and D. S. Johnson. *Computers and Intractability – A Guide to the Theory of NP-Completeness*. W. H. Freeman, New York, 1979.
- [7] J. Hartmanis and R. E. Stearns. *On the computational complexity of algorithms*. Trans. American Mathematical Society 117, 285-306, 1965.
- [8] F. C. Hennie. *One-tape off-line Turing machine computations*. Information and Control 8: 6, 553-578, 1965.
- [9] F. C. Hennie and R. E. Stearns. *Two-tape simulation of multitape Turing machines* J. ACM 13:4, 533-546.
- [10] J. E. Hopcroft and J. D. Ullman. *Introduction to Automata Theory, Languages, and Computation* Addison-Wesley, 1979.
- [11] N. Immerman. *Nondeterministic space is closed under complementation* SIAM J. Computing 17, 35-38, 1988.

- [12] D.S. Johnson. *The NP-completeness column: An on-going guide* J. of Algorithms 4, 1981.
- [13] R. M. Karp. *Reducibility among combinatorial problems* Complexity of Computer Computations, J. W. Thatcher and R. E. Miller eds. 85-103, Plenum Press, New York, 1972.
- [14] S. C. Kleene. *Introduction to metamathematics* D. Van Nostrand, 1952.
- [15] D. Knuth. *The Art of Computer Programming, Volume 3: Sorting and Searching*. Addison-Wesley Publishing Company, Inc., 1973.
- [16] R. E. Ladner, N. A. Lynch and A. L. Selman. *A comparison of polynomial time reducibilities*. Theoretical Computer Science, 1, 103-124, 1975.
- [17] E. L. Lawler. *Combinatorial Optimization: Networks and Matroids*. Holt, Reinhart and Winston, New York 1976.
- [18] L. A. Levin. *Universal sorting problems*. Problems of Information Transmission, 9, 265-266, 1973.
- [19] H. R. Lewis and C. H. Papadimitriou. *Elements of the Theory of Computation*. Prentice-Hall, Inc., USA, 1981.
- [20] L. Lovász and M. D. Plummer. *Matching Theory*. Annals of Discrete Mathematics, Vol. 29, North-Holland 1986.
- [21] E. M. Creight and A. R. Meyer *Classes of computable functions defined by bounds of computation*. Proc. first ACM Symposium on the Theory of Computing, pp. 79-88, 1969.
- [22] C. H. Papadimitriou. *On the complexity of integer programming*. J. ACM, 28, 2, pp. 765-769, 1981.
- [23] C. H. Papadimitriou. *Computational Complexity*. Addison Wesley, Reading, Massachusetts, 1994.
- [24] C. H. Papadimitriou and K. Steiglitz. *Combinatorial Optimization: Algorithms and Complexity*. Prentice-Hall, Englewood Cliffs, New Jersey, 1982.
- [25] W. J. Paul, N. Pippenger, E. Szemerédi, and W. T. Trotter. *On determinism versus nondeterminism and related problems* Proc. 24th IEEE Symp. on the Foundations of Computer Science, pp. 429-438, 1983.

- [26] W. J. Savitch. *Relationships between nondeterministic and deterministic tape complexities* J. Computer and Systems Sciences 4:2, 177-192, 1970.
- [27] A. Schrijer. *Theory of Linear and Integer Programming* Wiley, New York, 1986.
- [28] R. Szelepcsényi. *The method of forcing for nondeterministic automata* Bull. of the EATCS, 33, 96-100, 1987.
- [29] B. A. Trakhtenbrot. *Turing computations with logarithmic delay* Algebra i Logika, 3:4, 33-48, 1964.
- [30] A. M. Turing. *On computable numbers, with an application to the Entscheidungsproblem.* Proc. London Mathematical Society, 2, 42 pp.230-265, and no. 43, pp 544-546, 1936.

ΒΙΒΛΙΟΓΡΑΦΙΑ
