Τα όρια του Υπολογισμού και η Πολυπλοκότητα: Οι θεμελιώσεις της Επιστήμης των Υπολογιστών

Paul G. Spirakis
Professor, Patras University, Greece
Director, Research & Academic Computer Technology Institute (RACTI)
spirakis@cti.gr

Αλγόριθμοι και πολυπλοκότητα

Προβλήματα, αλγόριθμοι και τα όρια της υπολογισιμότητας

Στις αρχές του αιώνα μας, ένας από τους μεγαλύτερους μαθηματικούς, ο David Hilbert, στα πλαίσια της περίφημης διάλεξής του στο Διεθνές Συνέδριο Μαθηματικών στο Παρίσι το 1900 αναφέρθηκε σε 23 ανοικτά προβλήματα τα οποία θεωρούσε ως τα πιο σημαντικά για εκείνη την εποχή. Το δέκατο από αυτά ρωτούσε εάν μπορούσε να βρεθεί μια *διαδικασία* που να απαντά στο εάν μία δοσμένη διοφαντική εξίσωση έχει λύση ή όχι. Αρκετά χρόνια μετά, το 1928, στο Διεθνές Συνέδριο Μαθηματικών στη Μπολόνια και ως συνέχεια της προηγούμενης διάλεξής του, έθετε το πρόβλημα του να βρεθεί διαδικασία που θα μπορούσε να ελέγχει εάν ένα μαθηματικό (λογικό) σύστημα, όπως είναι για παράδειγμα η Ευκλείδειος Γεωμετρία, είναι *συνεπές*, δηλαδή δεν εμπεριέχει *αντιφάσεις*. Είναι προφανές το πόσο χρήσιμη θα ήταν μια τέτοια διαδικασία, καθώς ένας μαθηματικός που προτείνει ένα πολύπλοκο μαθηματικό σύστημα, το οποίο λόγω του όγκου των αξιωμάτων ή της πληθώρας των αποδεικτικών μεθόδων που περιέχει ξεφεύγει από την δυνατότητα ελέγχου από τον άνθρωπο, απλά θα τροφοδοτούσε (με μια κατάλληλη αναπαράσταση) το μαθηματικό του σύστημα στη μηχανική διαδικασία και η διαδικασία θα απαντούσε στο ερώτημα του αν το σύστημα αυτό περικλείει αντιφάσεις. Ένα άλλο πρόβλημα που έθεσε ο Hilbert στη διάλεξή του το 1928 και που ήταν κατά κάποιον τρόπο γενίκευση του δέκατου προβλήματος της διάλεξης του 1900, ήταν δοθέντος ενός μαθηματικού συστήματος να διαπιστωθεί εάν μια δοσμένη μαθηματική πρόταση αποτελεί *θεώρημα* του συστήματος ή όχι, το περίφημο *Entscheidungsproblem*. Ο Hilbert ήλπιζε ότι για τα παραπάνω προβλήματα θα έπρεπε να υπήρχε κάποια διαδικασία που να τα απαντά μέσα σε πεπερασμένο αριθμό βημάτων (περατοκρατική διαδικασία ή *αλγόριθμος*, όπως θα δούμε πιο κάτω).

Το 1931, ο μαθητής του Hilbert και εξίσου μεγάλος μαθηματικός Kurt Gödel, απέδειξε στην περίφημη εργασία του [Go31] ότι, δυστυχώς δύο από τις πεποιθήσεις του Hilbert ήταν λανθασμένες. Πιο συγκεκριμένα, ο Gödel παρουσίασε μια μαθηματική πρόταση στα πλαίσια του αξιωματικού συστήματος των φυσικών αριθμών (η πρώτη προσπάθεια αξιωματοποίησης του συστήματος των φυσικών αριθμών έγινε το 1910 από τους Whitehead και Russell, στο τρίτομο έργο τους *Principia Mathematica* [WhRu10]) για την οποία δεν ήταν δυνατό να αποδειχθεί με χρήση των αποδεικτικών μεθόδων του συστήματος αυτού ούτε ότι είναι θεώρημα ούτε ότι δεν είναι. Αυτή η μεγάλη ανακάλυψη, είναι το περίφημο *θεώρημα μη πληρότητας του Gödel*. Επιπλέον, ο Gödel απέδειξε ότι η συνέπεια ενός μαθηματικού συστήματος δεν είναι δυνατόν να αποδειχθεί με κανόνες και μεθόδους του ίδιου του συστήματος.

Το 1936 ο Alan Turing γινόταν ο θεμελιωτής της επιστήμης των υπολογιστών, με την ιστορική εργασία του "On computable numbers, with an application to the *Entscheidungsproblem*" ([Tu36]). Στην εργασία αυτή ο Turing όρισε ένα υπολογιστικό μοντέλο, που ονομάστηκε μετά προς τιμή του *Μηχανή Turing* (Turing Machine) το οποίο δεν είναι τίποτε άλλο παρά μια αφαιρετική περιγραφή, ένα μοντέλο, της έννοιας

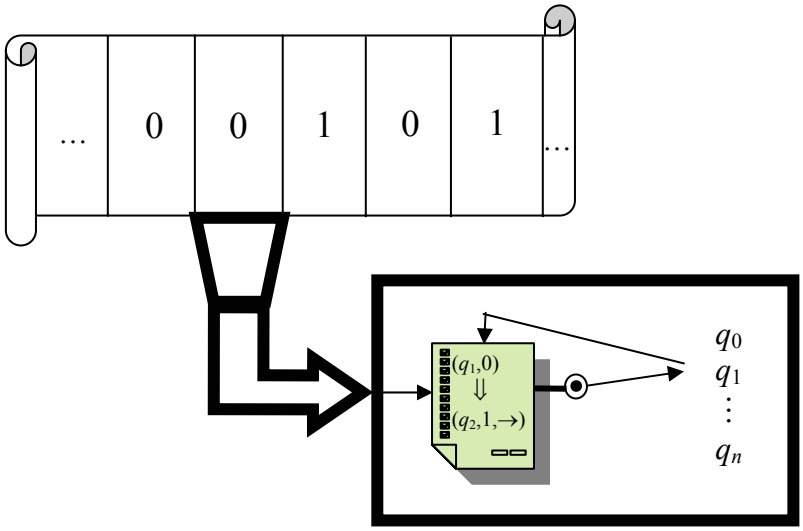
της *αλγοριθμικής υπολογισιμότητας* και του ηλεκτρονικού υπολογιστή όπως τον ξέρουμε σήμερα (δείτε Ενότητα 2). Επιπλέον, απαντώντας αρνητικά και στο τρίτο πρόβλημα του Hilbert, *Entscheidungsproblem*, απέδειξε ότι το πρόβλημα του εάν μια Μηχανή Turing τερματίζει για μια δοσμενη είσοδο, είναι *μη αλγοριθμικά αποφασίσιμο*. Με άλλα λόγια, δεν υπάρχει αλγόριθμος (Μηχανή Turing δηλαδή) που να δέχεται ως είσοδο την περιγραφή της μηχανής και της εισόδου της και να αποφασίζει αν κάποτε θα σταματήσει η μηχανή. (Δύο πολύ καλές αναφορές για όλα τα παραπάνω, είναι τα βιβλία των Davis [Da58] και van Heijenoort [vanHe67].)

Η συνεισφορά του Turing ήταν διπλή. Κατ' αρχήν συμπλήρωσε το αποτέλεσμα του Gödel. Παρεκκλίνοντας λίγο, στην πραγματικότητα, η απόδειξη του Turing ήταν μια απλούστερη έκδοση της απόδειξης μη πληρότητας του Gödel. Όμως, ο Turing είχε το πλεονέκτημα ότι εργαζόταν σε ένα μαθηματικό σύστημα που διευκόλυνε το έργο της επίδειξης μιας πρότασης για την οποία δεν υπάρχει αλγόριθμος που να αποφασίζει εάν είναι θεώρημα ή όχι. Η πρόταση, όπως είπαμε, είναι η εξής: *Η μηχανή Turing M τερματίζει κάποτε με είσοδο τη συμβολοσειρά x;* Ο Gödel, από την άλλη μεριά, είχε το μειονέκτημα ότι εργαζόταν σε ένα κάπως δύσκαμπτο φορμαλισμό, αυτόν των αναδρομικών συναρτήσεων, και έπρεπε να *εφεύρει*, κατά κάποιον τρόπο, μία μέθοδο γραφής μαθηματικών προτάσεων με χρήση απλά και μόνο αναδρομικών συναρτήσεων σε φυσικούς αριθμούς. Εάν προσέξει κανείς την απόδειξή του, θα διαπιστώσει ότι, πριν επιδείξει την περίφημη αυτοαναφερόμενη πρόταση που δεν μπορεί να αποδειχθεί εάν είναι θεώρημα ή όχι, χτίζει βήμα-βήμα μια μηχανή, που προσομοιάζει το μηχανισμό υπολογισμού της γνωστής συναρτησιακής γλώσσας προγραμματισμού Lisp. Αυτή είναι και η ομορφιά της απόδειξης του. Επανερχόμενοι όμως στην επόμενη συνεισφορά του Turing, η οποία είχε μεγάλη επίδραση στη μετέπειτα πορεία της επιστήμης των υπολογιστών, μέσα από τη χρησιμοποίηση της μηχανής που πρότεινε για την περιγραφή αλγόριθμων για διάφορα προβλήματα, υπήρξε η συνειδητοποίηση ότι πέρα από τη διαπίστωση ότι ένα πρόβλημα είναι επιλύσιμο (με το να επιδείξουμε έναν αλγόριθμο, ή μηχανή Turing, που να το επιλύει) μας ενδιαφέρει η επίλυση να γίνεται όσο το δυνατόν γρηγορότερα.

Ο πρώτος που διερεύνησε ζητήματα σχετικά με την πολυπλοκότητα της αλγοριθμικής επίλυσης προβλημάτων, ήταν ο ίδιος ο Gödel το 1956, πάλι σε σχέση με απόδειξη θεωρημάτων στα πλαίσια αξιωματικών συστημάτων. Με ένα γράμμα του προς τον John von Neumann, ο Gödel αναρωτιέται σχετικά με την πολυπλοκότητα εύρεσης αποδείξεων σε θεωρήματα που είναι διατυπωμένα με βάση τους κανόνες ενός τυπικού συστήματος (για μια αναφορά σε αυτό το γεγονός, δείτε την εργασία του Juris Hartmanis [Har89] καθώς και την τεχνική αναφορά [ChHa94]). Πιο συγκεκριμένα, αναφέρεται στη μηχανή Turing ως υπολογιστικό μοντέλο, και μετά ρωτά ποια είναι η συνάρτηση που φράσσει τον αριθμό των βημάτων που χρειάζονται για να βρεθούν αποδείξεις μήκους n. Στην πραγματικότητα ο Gödel ρωτούσε τον von Neumann, σύμφωνα με τη σημερινή τεχνική ορολογία, για την *ντετερμινιστική* (deterministic) υπολογιστική πολυπλοκότητα του προβλήματος της *απόδειξης θεωρημάτων* (theorem proving). Στο ίδιο γράμμα, ο Gödel επίσης ρωτάει σχετικά με την υπολογιστική πολυπλοκότητα του ελέγχου εάν ένας φυσικός αριθμός είναι πρώτος (primality testing), και μας εκπλήσσει κάπως το γεγονός ότι εκφράζει την πεποίθηση ότι η απόδειξη θεωρημάτων δεν πρέπει να είναι και τόσο δύσκολο πρόβλημα υπολογιστικά. Δυστυχώς, ο von Neumann ήδη έπασχε από καρκίνο και πέθανε ένα χρόνο αργότερα. Ποτέ δεν υπήρξε απάντηση στο γράμμα, και φαίνεται ότι ο Gödel δεν προσπάθησε να διερευνήσει περισσότερο το πολύ σημαντικό ερώτημα που έθεσε.

Γρήγορα Επιλύσιμα Προβλήματα: η κλάση P και η πρακτική υπολογισιμότητα

Πριν ορίσουμε τις βασικές έννοιες υπολογιστικής πολυπλοκότητας θα δώσουμε μία σύντομη περιγραφή της *μηχανής Turing* (δείτε Σχήμα 1). Μια μηχανή Turing αποτελείται από ένα πεπερασμένο μηχανισμό ο οποίος βρίσκεται πάντα σε κάποια κατάσταση ανάλογα με την πορεία των υπολογισμών, μια *απεριόριστου μήκους* ταινία διαιρεμένη σε κελιά, και μια κεφαλή ανάγνωσης των κελιών η οποία μπορεί να διαβάζει τα περιεχόμενα ενός κελιού κάθε φορά. Η ταινία έχει όριο προς τα αριστερά, ενώ εκτείνεται στο άπειρο προς τα δεξιά. Σε κάθε ένα από τα κελιά μπορεί να βρίσκεται ένα σύμβολο μέσα από ένα πεπερασμένο αριθμό συμβόλων που αποτελούν το αλφάβητο της ταινίας της μηχανής. Για να ξεκινήσει ο υπολογισμός, ένας αριθμός από τα πρώτα συνεχόμενα κελιά περιέχει στην αρχή την είσοδο της μηχανής η οποία αποτελείται από σύμβολα μέσα από το αλφάβητο εισόδου της μηχανής.



Σχήμα 1: Μία μηχανή Turing n καταστάσεων που ετοιμάζεται να μεταβεί στην κατάσταση q_2, γράφοντας στη θέση του 0 το 1 και μεταφέροντας την κεφαλή μία θέση δεξιά

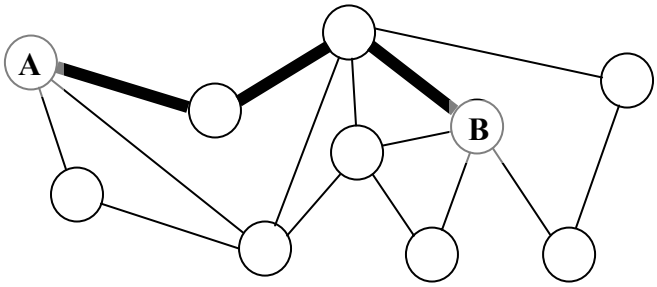
Σε κάθε βήμα, μια μηχανή Turing μπορεί να κάνει κάτι από τα ακόλουθα: (α) να αλλάξει εσωτερική κατάσταση (β) να αλλάξει το σύμβολο το οποίο βρίσκεται στο κελί που διαβάζει η κεφαλή ή (γ) να μετακινήσει την κεφαλή δεξιά ή αριστερά κατά ένα κελί. Όπως έχουμε ήδη πει, αυτό το τόσο απλό μοντέλο είναι γενικά αποδεκτό ότι μπορεί να υπολογίσει κάθε τι που μπορεί να υπολογιστεί *αλγοριθμικά* (η περίφημη *υπόθεση του Church*) και είναι ισοδύναμο ως προς το τι μπορεί να υπολογιστεί με τον ηλεκτρονικό υπολογιστή μας.

Ας επανέλθουμε όμως στο ζήτημα του πόσο χρόνο μπορεί να χρειαστεί μια μηχανή Turing για να επιλύσει ένα πρόβλημα. Όπως είδαμε, πρώτος Gödel έθεσε αυτό το ερώτημα, χωρίς όμως να το διερευνήσει περισσότερο. Μέσα στο διάστημα 1965-1967, ο J. Edmonds σε δύο εργασίες του, χρησιμοποίησε για πρώτη φορά την έννοια *καλός* (αποδοτικός) *αλγόριθμος* για εκείνους τους αλγόριθμους (μηχανές Turing) που για να επιλύσουν ένα στιγμιότυπο μεγέθους n, χρειάζονται *το πολύ* cn^k βήματα, για *κάποιους αριθμούς* c, k.

Γρήγορα έγινε αποδεκτό, με βάση διάφορα πειστικά επιχειρήματα που μπορεί κανείς να βρει αναλυτικά στα βιβλία [Papa94] και [HoUl79], ότι τα προβλήματα που θα πρέπει να

θεωρούνται *γρήγορα επιλύσιμα* είναι η κλάση όλων των προβλημάτων για τα οποία υπάρχει αλγόριθμος (μηχανή Turing) που τα επιλύει χρησιμοποιώντας πολυωνυμικό αριθμό βημάτων στο μέγεθος του στιγμιότυπου. Ένα από τα ισχυρότερα επιχειρήματα είναι το ότι για κάθε *ρεαλιστικό* μοντέλο υπολογισμού *M*, εάν ένας αλγόριθμος είναι πολυωνυμικού χρόνου για μια μηχανή Turing, τότε ο αλγόριθμος θα είναι πολυωνυμικού χρόνου ακόμα και αν περιγραφτεί στα πλαίσια των κανόνων του μοντέλου *M* και αντίστροφα. Με άλλα λόγια, η ιδιότητα της επιλυσιμότητας ενός προβλήματος γρήγορα (δηλαδή σε πολυωνυμικό αριθμό στοιχειωδών βημάτων) είναι μία *αμετάβλητη ιδιότητα* μέσα στην κλάση όλων των *ρεαλιστικών* μοντέλων της έννοιας της μηχανικής υπολογισιμότητας.

Ο Edmonds είχε σωστή διαίσθηση όσον αφορά το τι πρέπει να είναι καλός αλγόριθμος, και είχε εντοπίσει προβλήματα που επιδέχονται τέτοιους αλγορίθμους, Όμως για ένα συγκεκριμένο πρόβλημα, το *Πρόβλημα του Περιπλανώμενου Πωλητή* (Traveling Salesman Problem ή TSP) δεν είχε καταφέρει να βρει έναν *καλό* αλγόριθμο. Πριν όμως ορίσουμε το πρόβλημα αυτό θα εξετάσουμε κάποιο άλλο. Έστω ότι δίνεται ένας χάρτης μιας χώρας με συνδέσεις μεταξύ των πόλεων με δρόμους και δύο συγκεκριμένες πόλεις Α και Β (Σχήμα 2).



Σχήμα 2: Ένα οδικό δίκτυο πόλεων και μία σύνδεση μεταξύ των πόλεων Α και Β

Το πρόβλημα που τίθεται είναι εάν το δίκτυο επιτρέπει τη σύνδεσή τους, έστω και μέσα από άλλες πόλεις (ας καλέσουμε το πρόβλημα αυτό *PATH*). Αυτό είναι σημαντικό πρόβλημα καθώς εάν το προτεινόμενο δίκτυο δεν επιτρέπει τη σύνδεση κάποιων πόλεων τότε είναι λανθασμένο και θα πρέπει να επανασχεδιαστεί. Το πρόβλημα αυτό έχει μια απλή και *γρήγορη* μηχανική λύση που φαίνεται στο Σχήμα 3. Μια πρόχειρη ανάλυση μας δείχνει ότι ο αλγόριθμος τερματίζει μετά από *το πολύ* n^2 βήματα για *κάθε* χάρτη με *n* πόλεις.

(Έχοντας ξεκινήσει από την πόλη Α)
Μέχρι να φτάσεις στην πόλη Β κάνε *ένα* από τα εξής τρία βήματα:

- Εάν βρίσκεσαι στην πόλη Α και **δεν υπάρχει** δρόμος που να μην έχεις ακολουθήσει ξανά, **τερμάτισε** και **δήλωσε** ότι οι πόλεις Α και Β δεν συνδέονται μεταξύ τους.
- Από την πόλη που βρίσκεσαι, κοίταξε όλους του δρόμους που φεύγουν από αυτήν και **ακολούθησε** κάποιο δρόμο που δεν έχεις ακολουθήσει παλαιότερα.
- Εάν όμως έχεις ακολουθήσει κάποτε **όλους** τους δρόμους που φεύγουν από την τρέχουσα πόλη **γύρισε πίσω** σε μία πόλη που είχες επισκεφτεί πιο πριν ακολουθώντας προς τα πίσω το δρόμο από τον οποίο **πρωτοεπισκεύτηκες** την τρέχουσα πόλη.

Δήλωσε ότι οι πόλεις Α και Β συνδέονται μεταξύ τους.

Σχήμα 3: Ένας αλγόριθμος ελέγχου σύνδεσης μεταξύ των πόλεων Α και Β (πρόβλημα PATH)

Επίσης, βλέπουμε ότι η παραπάνω λύση αποτελείται από *στοιχειώδη* και εύκολα υλοποιήσιμα βήματα, μπορεί να εκτελεστεί εντελώς *μηχανικά* δοσμένου του χάρτη της χώρας με το οδικό δίκτυο, και *πάντοτε τερματίζει* δίνοντας τη *σωστή* απάντηση, δηλαδή είναι ένας *αλγόριθμος*. Στο Σχήμα 3 βλέπουμε μία πιθανή εύρεση διασύνδεσης μεταξύ των Α και Β.

Ας αλλάξουμε όμως λίγο το πρόβλημα έτσι ώστε να μην μας ενδιαφέρει απλά και μόνο η σύνδεση των δύο πόλεων μέσα από το οδικό δίκτυο αλλά να μας ενδιαφέρει επιπρόσθετα να συνδέονται με κάποια διαδρομή η οποία να περνά ακριβώς *μία* φορά από *όλες* τις υπόλοιπες πόλεις. Αυτό είναι και το πρόβλημα του *Πρόβλημα του Περιπλανώμενου Πωλητή* (Traveling Salesman Problem – TSP). Κάποιος αμέσως θα σκεφτόταν να εφαρμόσει μια παραλλαγή του αλγόριθμου στο Σχήμα 2 και σε αυτό το πρόβλημα. Επίσης, θα περίμενε κανείς και το καινούριο πρόβλημα να είναι επιλύσιμο σε περίπου τον ίδιο αριθμό βημάτων. Μοιάζουν τόσο, εξάλλου, τα δύο προβλήματα. Τα πράγματα είναι όμως πολύ διαφορετικά! Πραγματικά, μπορούμε να εφαρμόσουμε έναν αλγόριθμο όπως ο παραπάνω που απλά θα ξεκινά από την Α και θα ακολουθεί τους δρόμους μέχρι να βρει την Β, κρατώντας κάθε φορά μία σημείωση των πόλεων που έχει επισκεφτεί. Εάν από μία πόλη μπορεί να επισκεφτεί μόνο πόλεις που έχει ήδη επισκεφτεί, τότε ο αλγόριθμος μπορεί π.χ. να επιστρέψει σε μία προηγούμενη πόλη και να ακολουθήσει κάποιον άλλο δρόμο ελπίζοντας αυτή τη φορά να βρει τη ζητούμενη διαδρομή. Εάν μελετήσουμε τον αριθμό των βημάτων αυτού του αλγόριθμου, θα δούμε ότι μάλλον δεν είναι και τόσο μικρός! Ο αλγόριθμος αυτός μπορεί να εξερευνά διαδρομές που ξαφνικά διαπιστώνεται ότι δεν μπορεί να αποτελούν μέρος μίας καθολικής διαδρομής που περνά από όλες τις πόλεις. Και τότε ο αλγόριθμος θα πρέπει να ψάξει για άλλη διαδρομή. Μία προσεκτική ανάλυση του αλγόριθμου που προτείναμε, αποκαλύπτει ότι για έναν χάρτη με n πόλεις, ο αριθμός των βημάτων στη χειρότερη περίπτωση μπορεί να φθάσει, χονδρικά, μέχρι n^n.Ακόμη και με τους ταχύτερους υπολογιστές στον κόσμο σήμερα, για έναν χάρτη 1000 πόλεων, θα χρειαστούν πάρα πολλά χρόνια υπολογισμού για να βρεθεί το πολυπόθητο μονοπάτι! Στο Σχήμα 4 βλέπουμε ποια αποτελέσματα έχει η τεχνολογία όσον αφορά το πόσο μεγάλα στιγμιότυπα μπορούν να επιλυθούν σε μία ώρα (δείτε [GaJo79]).

Χρόνος τρεξίματος αλγόριθμου	Μέγιστο μέγεθος στιγμιότυπου με σημερινή τεχνολογία	100 φορές πιο γρήγορη τεχνολογία	1000 φορές πιο γρήγορη τεχνολογία
N	N_1	$100N_1$	$1000N_1$
n^2	N_2	$10N_2$	$31.6N_2$
n^3	N_3	$4.64N_3$	$10N_3$
n^5	N_4	$2.5N_4$	$3.98N_4$
2^n	N_5	$6.64+N_5$	$9.97+N_5$
3^n	N_6	$4.19+N_6$	$6.29+N_6$

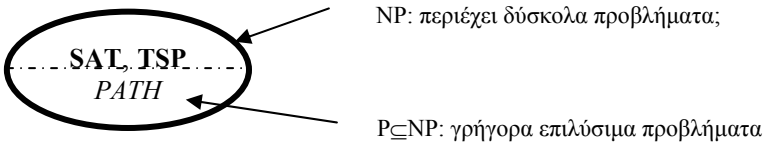
Σχήμα 4: Το μεγαλύτερο στιγμιότυπο που επιλύεται μέσα σε μία ώρα

Ίσως όμως, να σκεφτεί κανείς, να υπάρχει κάποιος καλύτερος αλγόριθμος. Αρκεί κανείς να σκεφτεί πιο προσεκτικά. Όμως ο Edmonds, όπως και πολλοί άλλοι μετέπειτα, δεν είχε κατορθώσει να βρει ένα γρήγορο αλγόριθμο γι' αυτό το πρόβλημα και αμέσως διατύπωσε την εικασία ότι μάλλον *δεν* πρέπει να υπάρχει πολυωνυμικός αλγόριθμος γι' αυτό το πρόβλημα. Όμως, μια άλλη σημαντική παρατήρηση, με δεδομένο ότι το πρόβλημα δεν φαινόταν να επιλύεται από πολυωνυμικό αλγόριθμο, ήταν ότι *εάν* μας

δοθεί μια διαδρομή που κάποιος ισχυρίζεται ότι είναι λύση ενός στιγμιότυπου του TSP, τότε είναι εύκολο υπολογιστικά να ελέγξουμε *γρήγορα* (δηλαδή σε πολυωνυμικό αριθμό βημάτων) ότι πράγματι αυτή η διαδρομή είναι η επιθυμητή. Η κλάση, λοιπόν, των προβλημάτων που έχουν αυτή την ιδιότητα ονομάστηκε NP (συνήθως ο ορισμός δίνεται με βάση τις *μη ντετερμινιστικές* μηχανές Turing αλλά, για απλούστευση, αποφύγαμε να χρησιμοποιήσουμε το φορμαλισμό αυτών των μηχανών). Προσέξτε, επίσης, ότι η κλάση P είναι υποσύνολο της κλάσης NP, δηλαδή $P \subseteq NP$, χωρίς μέχρι σήμερα να έχει αποδειχθεί (αποτελεί σημαντικό ανοικτό πρόβλημα της θεωρίας της πολυπλοκότητας) εάν η σχέση υποσυνόλου είναι *γνήσια* (δηλαδή εάν $P \neq NP$).

Οι ερευνητές, λοιπόν, είχαν στα χέρια τους προβλήματα για τα οποία είχαν ήδη βρει πολυωνυμικούς αλγορίθμους επίλυσης, και προβλήματα που αν και δεν διαφαινόταν εάν μπορούσαν να επιλυθούν από πολυωνυμικό αλγόριθμο ή όχι, παρόλα αυτά, εάν παρεχόταν μια υποψήφια λύση, υπήρχε πολυωνυμικός αλγόριθμος που να *ελέγχει* εάν πράγματι αυτή είναι λύση ή όχι. Η επόμενη σκέψη ήταν να βρεθούν εκείνα τα προβλήματα αυτής της κλάσης που πρέπει να είναι τα πιο δύσκολα να επιλυθούν δηλαδή αυτά που αντιπροσωπεύουν τη δυσκολία όλης της κλάσης με την έννοια ότι εάν μπορούσαμε να λύναμε πολυωνυμικά ένα από αυτά, θα λύναμε πάλι πολυωνυμικά όλα τα προβλήματα του NP. Έτσι γεννήθηκε η έννοια της *αναγωγής πολυωνυμικού χρόνου* από ένα πρόβλημα σε ένα άλλο. Σύμφωνα με αυτή την έννοια, για να δείξουμε ότι ένα πρόβλημα είναι τουλάχιστον τόσο δύσκολο όσο ένα άλλο όσον αφορά την επίλυση με πολυωνυμικό αλγόριθμο, αρκεί να βρούμε μια απεικόνιση στιγμιότυπων του πρώτου σε στιγμιότυπα του δευτέρου που γρήγορα, έτσι ώστε μια λύση στο δεύτερο πρόβλημα να αντιστοιχεί σε μία λύση του πρώτου. Έτσι, δείχνουμε ότι το δεύτερο πρόβλημα δεν μπορεί να είναι πιο εύκολο να λυθεί από το πρώτο. Τα προβλήματα που ανήκουν στο NP και έχουν αυτή την ιδιότητα καλούνται *πλήρη ως προς την κλάση NP* (NP-complete).

Το πρώτο πρόβλημα που βρέθηκε να έχει αυτή την ιδιότητα, είναι το πρόβλημα της ικανοποιησιμότητας λογικών προτάσεων (SATISFIABILITY ή SAT) και η σημαντική αυτή ανακάλυψη, γνωστή ως *Θεώρημα Cook-Levin*, έγινε ανεξάρτητα στις αρχές της δεκαετίας του ’70 στις πρωτοποριακές εργασίες των Stephen Cook [Cook71], Richard Karp [Karp72], και Leonid Levin [Levin73]. Την ανακάλυψη αυτή, ακολούθησαν πλήθος δημοσιεύσεων όπου μια πληθώρα άλλων προβλημάτων βρέθηκαν να είναι πλήρη ως προς την κλάση NP ([GaJo79]). Το θεμελιώδες ερώτημα,, που παραμένει βασανιστικά αναπάντητο, είναι το εάν ισχύει P=NP, με την επιστημονική κοινότητα να συγκλίνει στο ότι κάτι τέτοιο δεν ισχύει. Στο Σχήμα 5 βλέπουμε την πρωταρχική κατάταξη των προβλημάτων ανάλογα με την ευκολία επίλυσής τους.



Σχήμα 5: Η εικόνα των προβλημάτων σε σχέση με την πολυπλοκότητα επίλυσής τους

<u>Ερευνητικές κατευθύνσεις</u>

Μέσα από τις προσπάθειες που περιγράψαμε προηγουμένως γεννήθηκε, και σήμερα έχει πια ωριμάσει, η *Θεωρία Πολυπλοκότητας*. Η θεωρία αυτή εξετάζει τα διάφορα προβλήματα ως προς το πόσο γρήγορα μπορούν να επιλυθούν και μία από τις

μεγαλύτερες επιτυχίες της είναι η κατάταξή τους σε δύο μεγάλες κατηγορίες. Στην πρώτη ανήκουν τα προβλήματα για τα οποία έχει ήδη βρεθεί κάποιος γρήγορος αλγόριθμος που για στιγμιότυπα (π.χ. όπως ο χάρτης) μεγέθους n (ο αριθμός των πόλεων του χάρτη) απαιτούνται το πολύ n^2, n^3 ή, γενικά, n^k βήματα, για κάποιο *σταθερό* αριθμό k. Τα προβλήματα αυτά καλούνται *προβλήματα επιλύσιμα σε πολυωνυμικό χρόνο* (λόγω της μορφής, πολυώνυμο, που έχει η έκφραση για τον αριθμό βημάτων) και αποτελούν την περίφημη κλάση P. Η δεύτερη κατηγορία συμπεριλαμβάνει προβλήματα όπως το TSP που είναι από τα δυσκολότερα προβλήματα του NP και για τα οποία *μέχρι τώρα* δεν έχει βρεθεί γρήγορος αλγόριθμος, ενώ οι ταχύτεροι αλγόριθμοι που έχουν προταθεί απαιτούν αριθμό βημάτων της μορφής 2^n, 3^n ή ακόμα και n^n, δηλαδή *εκθετικά* πολλά βήματα. Τα προβλήματα αυτά αποτελούν την περίφημη κλάση των *NP πλήρων* προβλημάτων, των πιο δύσκολων στο NP. Αυτό που πρέπει να έχουμε κατά νου είναι ότι αν και μέχρι σήμερα κανείς δεν έχει αποδείξει ότι γι' αυτά τα προβλήματα πραγματικά *δεν* υπάρχουν γρήγοροι αλγόριθμοι, η Θεωρία της Πολυπλοκότητας, μέσα από μια μαθηματικά θεμελιωμένη μεθοδολογία και επιχειρηματολογία, δίνει πειστήρια για το ότι δεν μπορεί να υπάρχουν τέτοιοι αλγόριθμοι.

Όμως η έννοια της πολυπλοκότητας δεν περιορίζεται μόνο στη μελέτη της χειρότερης περίπτωσης για την επίλυση των στιγμιότυπων ενός προβλήματος. Σχετικά με άλλες θεωρήσεις της έννοιας αυτής, ο Andrei Kolmogorov, ο Ray Solomonoff, και ο Gregory Chaitin πρότειναν, ανεξάρτητα ([Kolm65,Sol64,Chaitin66]) την έννοια της *περιγραφικής πολυπλοκότητας* ή, όπως λέγεται, *Kolmogorov complexity*. Ας θεωρήσουμε τις συμβολοσειρές 000000000000000 και 010100001110001 των 15 bits. Υπό το πρίσμα της κλασσικής θεωρίας πιθανοτήτων, οι δύο αυτές συμβολοσειρές έχουν την ίδια πιθανότητα εμφάνισης σε ένα πείραμα ρίψης ενός δίκαιου νομίσματος: $(½)^{15}$. Όμως, η συμβολοσειρά 000000000000000 μπορεί εύκολα να περιγραφεί από την έκφραση ***20 μηδενικά*** ενώ η 010100001110001 δεν φαίνεται να επιτρέπει μια σύντομη περιγραφή της. Η πρώτη συμβολοσειρά έχει χαμηλή περιγραφική πολυπλοκότητα ενώ η δεύτερη υψηλή. Υπάρχει σειρά θεωρητικών αποτελεσμάτων (π.χ. [OSW94]) που δείχνει ότι στιγμιότυπα που κωδικοποιούνται με συμβολοσειρές υψηλής περιγραφικής πολυπλοκότητας, αποτελούν δύσκολα στιγμιότυπα των αντίστοιχων υπολογιστικών προβλημάτων.

Το 1986 ο Leonid Levin ήταν ο επινοητής της έννοιας της πολυπλοκότητας μέσης περίπτωσης (average case complexity) προβλημάτων στην πρωτοποριακή του εργασία [Levin86]. Σε αντιστοιχία με την NP-πληρότητα, ο Levin όρισε την *πληρότητα ως προς μέση περίπτωση* η οποία μέσα από κατάλληλες αναγωγές μεταξύ προβλημάτων, που όμως λαμβάνουν υπόψη την πιθανοτική κατανομή των στιγμιότυπων (ώστε να υπάρχει η έννοια της μέσης περίπτωσης), διακρίνει μεταξύ δύσκολων προβλημάτων που είναι εύκολα στη μέση περίπτωση και δύσκολων προβλημάτων που παραμένουν δύσκολα και στη μέση περίπτωση (δείτε την ανάλυση [LeVe] για την ανάλυση μέσης πολυπλοκότητας μιας έκδοσης του προβλήματος χρωματισμού γραφήματος). Επίσης, δείτε τις εργασίες [Aj96,AjDw97] όπου για την πρώτη ανάλυση προβλήματος το οποίο έχει την ιδιότητα η πολυπλοκότητα μέσης περίπτωσης να ισούται με την πολυπλοκότητα χειρότερης περίπτωσης.

Μια άλλη όψη της πολυπλοκότητας διαφάνηκε το 1991 όταν οι Cheeseman, Taylor και Kanefsky πραγματοποιήθηκε σειρά πειραμάτων με το πρόβλημα του χρωματισμού γραφημάτων με 3 χρώματα. Το πρόβλημα αυτό αναφέρεται σε μία δομή που αποτελείται από n κόμβους κάποιοι από τους οποίους συνδέονται μέσω m συνδέσμων. Ζητείται χρωματισμός των κόμβων με χρήση τριών, το πολύ,

διαφορετικών χρωμάτων έτσι ώστε κόμβοι που συνδέονται μεταξύ τους να έχουν διαφορετικό χρώμα (3-COLORING). Το πρόβλημα αυτό ανήκει στην πολύ σημαντική οικογένεια των NP-πλήρων προβλημάτων για τα οποία συζητήσαμε πιο πάνω. Το φαινόμενο που παρατηρήθηκε στα πειράματα ήταν το εξής: σε γραφήματα στα οποία ο λόγος $r = \frac{m}{n}$ βρισκόταν γύρω από μία συγκεκριμένη τιμή r_0 (που εκτιμήθηκε πειραματικά), οι αλγόριθμοι που χρησιμοποιήθηκαν για τον χρωματισμό αυτών των γραφημάτων δυσκολεύονταν στο να βρούν μία κατάλληλη ανάθεση χρωμάτων. Όμως για γραφήματα που ο λόγος απομακρυνόταν από την τιμή αυτή, οι αλγόριθμοι τερμάτιζαν γρήγορα. Επίσης, όταν ο λόγος απομακρυνόταν προς τα κάτω από την τιμή r_0, τότε σχεδόν όλα τα γραφήματα με n κόμβους και m συνδέσεις είχαν κάποιο χρωματισμό με τρία χρώματα ενώ για λόγους πάνω από την τιμή r_0, σχεδόν κανένα γράφημα δεν είχε χρωματισμό. Το φαινόμενο αυτό ονομάζεται *αλλαγή φάσης* ή *κατάστασης* καθώς τα γραφήματα μπορούν είτε να χρωματίζονται με τρία χρώματα (η μία φάση) είτε να μην χρωματίζονται (η άλλη φάση) και η μετάβαση από τη μία στην άλλη κατάσταση συμβαίνει απότομα, όταν ο λόγος των δύο παραμέτρων που καθορίζουν ένα γράφημα (οι συνδέσεις και οι κόμβοι) διασχίζει την τιμή r_0. Διαφαίνεται, ότι γύρω από την τιμή r_0 (κατώφλι) συγκεντρώνονται «δύσκολα» γραφήματα, που προσδίδουν στο πρόβλημα του χρωματισμού γραφημάτων με τρία χρώματα το χαρακτηριστικό του δύσκολα επιλύσιμου (NP-πλήρους) προβλήματος. Παρόμοια αποτελέσματα παρατηρήθηκαν και για το πρόβλημα της ικανοποιησιμότητας λογικών τύπων 3-SAT *m* προτάσεων πάνω σε *n* λογικές μεταβλητές όπου έχουμε τρεις εμφανίσεις μεταβλητών ή των συμπληρωμάτων τους σε κάθε πρόταση (δείτε την εργασία [Stam03] για μα επισκόπηση των φαινομένων αλλαγής φάσης).

Οι νέες εξελίξεις στην Τεχνολογία και οι νέες προκλήσεις

Σήμερα, το Διαδίκτυο και ο Παγκόσμιος Ιστός (www) αποτελούν στην πράξη μια παγκόσμια Μηχανή, με άφθονη εγκατεσπαρμένη πληροφορία και υπέροχες δυνατότητες άντλησής της και Επικοινωνίας.

Ταυτόχρονα, οι κινητές επικοινωνίες (mobile communications & computing) επεκτείνουν τον Ιστό προσθέτοντας δυναμικά, τοπικά δίκτυα ανταλλαγής Πληροφορίας.

Το όραμα της Ευρωπαϊκής Ένωσης για Διάχυτη Νοημοσύνη (Ambient Intelligence) είναι πλέον κοντά στην υλοποίηση του.

Τέλος, οι Κβαντικοί Υπολογιστές δείχνουν κάποιες θεωρητικές ελπίδες για άρση των φραγμάτων Πολυπλοκότητας.

Απέναντι στις προκλήσεις αυτές, υπάρχει ανάγκη επέκτασης των θεμελιώσεων. Η προσπάθεια ήδη γίνεται και αναμένονται ενδιαφέροντα συμπεράσματα.

Βιβλιογραφία

[Aj96] M. Ajtai. Generating hard instances of lattice problems. In Proc. 28th Symposium on Theory of Computing (STOC), pp. 99-108, 1996.

[AjDw97] M. Ajtai and C. Dwork. A public-key cryptosystem with worst-case/average-case equivalence. In Proc. 29th Symposium on Theory of Computing (STOC), pp. 284-293, 1997.

[Chaitin66] G.H. Chaitin. On the length of programs for computing finite binary sequences. Journal of the ACM 13, pp. 547-570, 1966.

[ChHa94] S. Chari and J. Hartmanis. On the Intellectual Terrain around NP. Technical Report, MPI-I-94-103, Max Planck Institut für Informatik, 1994.

[Cook71] S. Cook. The complexity of theorem-proving procedures. In Proc. 3rd Annual ACM Symposium on Theory of Computing, pp. 151-158, ACM Press, 1971.

[Da58] M. Davis. Computability and unsolvability. McGraw-Hill, 1958.

[GaJo79] M.R. Garey and D.S. Johnson. Computers and Intractability. Freeman, San Francisco, CA, 1979.

[Go31] K. Gödel. Über formal unentscheidbare Sätze der Principia Mathematica und verwandter Systeme, I (On formally undecidable theorems in Principia Mathematica and related systems). Monatshefte für Math. und Physik, 38:173-198, 1931. Για μία μετάφραση στα Αγγλικά. δείτε την αναφορά [vanHe67] πιο κάτω.

[Har89] J. Hartmanis. Gödel, von Neumann and the P=?NP problem. Bulletin of the EATCS, 38:101-107, 1989.

[vanHe67] J. van Heijenoort. From Frege to Gödel: A Source Book in Mathematical Logic. MA: Harvard University Press, 1967.

[HoUl79] J.E. Hopcroft and J.D. Ullman. Introduction to Automata Theory, Languages and Computation. Addison-Wesley, 1979.

[Karp72] R.M. Karp. Reducibility among combinatorial problems. In Miller and Thatcher (eds.) Complexity of Computer Computations, pp. 85-103, Plenum Press, 1972.

[Kolm65] A. Kolmogorov. Three approaches to the concept of The Amount of Information. Probl. of Inform. Transm., Vol. 1/1, 1965.

[Levin73] L. Levin. Universal'nyĭe perebornyĭe zadachi (universal search problems: in Russian). Problemy Peredachi Informatsii 9(3), pp. 265-266, 1973.

[Levin86] L.A. Levin. Average case complete problems. SIAM Journal on Computing 15, pp. 285-286, 1986.

[LeVe] L.A. Levin and R. Venkatesan. An Average Case NP-Complete Graph Problem. Submitted.

[OSW94] P. Orponen, K. Ko, U. Schöning, and O. Watanabe, Instance Complexity. Journal of the ACM 41, 96-121, 1994.

[Papa94] C.H. Papadimitriou. Computational Complexity. Addison - Wesley, 1994.

[Sol64] R.J. Solomonoff. A formal theory of inductive inference. Information and Control, Vol. 7/1, pp. 1-22, 1964.

[Stam03] Y. C. Stamatiou Threshold Phenomena: The Computer Scientist's Viewpoint, EATCS (European Association of Theoretical Computer Science) Bulletin 80, 199-234, June 2003.

[Tu36] A.M. Turing. On computable numbers, with an application to the Entsceidungsproblem. Proc. London Math. Society 2(43):230-265, 1936. Μία διόρθωση στην αρχική εργασία: 2(43): 544-546, 1937.

[WhRu10] A.N. Whitehead and B. Russell. Principia Mathematica. Τρεις τόμοι, Cambridge University Press, London, 1910, 1912, 1913.