Εισαγωγή στη θεωρία πολυπλοκότητας

M.A.

21/10/2020

Θεωρία υπολογιστικής πολυπλοκότητας

- Η θεωρία πολυπλοκότητας έχει σκοπό να υπολογίσει τη δυσκολία επίλυσης των προβλημάτων [AB09; GJ79]
- Κάνει μια χονδρική εκτίμηση του χρόνου $t(N)$ που χρειάζεται για τον υπολογισμό, ως συνάρτηση μιας βασικής παραμέτρου του προβλήματος.
- Τα (επιλύσιμα) προβλήματα κατατάσσονται σε δύο κατηγορίες «δυσκολίας», αυτά που είναι *πολυωνυμικής τάξης*, δηλαδή υπάρχει αλγόριθμος επίλυσής τους σε πολυωνυμικό χρόνο, και αυτά που είναι εκθετικής τάξης, δηλαδή υπάρχει μόνο αλγόριθμος που χρειάζεται εκθετικό χρόνο.

Αλγόριθμος πολυωνυμικού/εκθετικού χρόνου

- Μια συνάρτηση $f(n)$ λέμε ότι είναι $O(g(n))$ εφόσον υπάρχει σταθερά c τέτοια ώστε

$$|f(n)| \leq c|g(n)|$$

για κάθε $n \geq 0$.

- *Αλγόριθμος πολυωνυμικού χρόνου* λέγεται αυτός που έχει συνάρτηση χρονικής πολυπλοκότητας $O(p(n))$ για κάποια πολυωνυμική συνάρτηση $p(n)$ του μήκους της εισόδου n.
- Όποιος αλγόριθμος δεν μπορεί να φραχθεί από πολυωνυμική συνάρτηση λέγεται *αλγόριθμος εκθετικού χρόνου*.

«Καλοί» και «κακοί» αλγόριθμοι

- Ο Cobham σε ένα κλασσικό άρθρο με τίτλο *The intrinsic computational difficulty of functions* [Cob65] εξίσωσε τους «καλούς» αλγόριθμους με τους πολυωνυμικούς.
- Η άποψη αυτή προφανώς ελέγχεται για μικρού μεγέθους προβλήματα, όπου ενίοτε είναι προτιμότερο να χρησιμοποιήσει κανείς εκθετικό αλγόριθμο.
- Άλλη γνωστή «εξαίρεση» είναι ο αλγόριθμος επίλυσης προβλημάτων γραμμικού προγραμματισμού Simplex. Ενώ είναι εκθετικής πολυπλοκότητας συνήθως είναι πολύ αποδοτικός.

Χείριστος και μέσος χρόνος εκτέλεσης

- Η συμπεριφορά αυτή συνδέεται με το γεγονός ότι η παραπάνω ταξινόμηση κυρίως αναφέρεται στην ύπαρξη (ή μη) μιας τουλάχιστον περίστασης, όπου μπορεί να εμφανισθεί η ανάγκη για αυξημένο χρόνο εκτέλεσης.
- Στην περίπτωση τέτοιων προβλημάτων είναι χρήσιμο να ενδιαφερθεί κανείς για το *μέσο* χρόνο εκτέλεσης του αλγορίθμου.

Σύγκριση του χρόνου εκτέλεσης πολυωνυμικών και μη αλγορίθμων

- Στον επόμενο πίνακα δείχνει πόσο γρήγορα χειροτερεύει ο χρόνος εκτέλεσης ενός αλγορίθμου στις δυο αυτές κατηγορίες.
- Στον πίνακα υποτίθεται ότι κάθε πράξη διαρκεί 1μsec.
- Για $n = 3$ και οι δύο αλγόριθμοι χρειάζονται 27 μsec.

	$n = 3$	$n = 10$	$n = 20$
n^3	27×10^{-6}	0.001 sec	0.008 sec
3^n	27×10^{-6}	0.059 sec	58 min
	$n = 30$	$n = 40$	$n = 60$
n^3	0.027 sec	0.064 sec	0.216 sec
3^n	6.5 yr	3855 αιώνες	1.3×10^{13} αιώνες

Κέρδος από την αύξηση της ταχύτητας

- Το επόμενο παράδειγμα δείχνει ότι σε αλγόριθμους εκθετικού χρόνου το κέρδος από την αύξηση της ταχύτητας των μηχανών είναι μηδαμινό.
- Ο αλγόριθμος Α χρειάζεται σήμερα χρόνο εκτέλεσης n^3 ενώ ο αλγόριθμος Β χρειάζεται χρόνο 3^n.
- Αν η ταχύτητα των υπολογιστών γίνει 1000 φορές μεγαλύτερη, πόσο πιο μεγάλα προβλήματα μπορούμε να λύσουμε σε σύγκριση με τα σημερινά;

Κέρδος από την αύξηση της ταχύτητας

1. Στο παρόν ο πολυωνυμικός αλγόριθμος Α επιλύει πρόβλημα μεγέθους m σε χρόνο $t = m^3\delta$, όπου κάθε πράξη θέλει χρόνο δ.
   Ο Α στο μέλλον θα λύνει ένα πρόβλημα έστω μεγέθους n σε χρόνο $t = n^3 \times \frac{\delta}{1000}$, δηλαδή $m^3 r = n^3/1000$, άρα
   $$n = 10m$$

2. Στο παρόν ο εκθετικός αλγόριθμος Β επιλύει πρόβλημα μεγέθους m σε χρόνο $T = 3^m\delta$,
   ενώ στο μέλλον θα επιλύει ένα πρόβλημα μεγέθους n σε χρόνο $T = 3^n \frac{\delta}{1000}$, επομένως $3^m = \frac{3^n}{1000}$ και τελικά
   $$n = m + \frac{\ln 1000}{\ln 3} \approx m + 6.29$$

Με δυο λόγια το κέρδος στο μέγεθος του προβλήματος που μπορεί να λυθεί από μια αύξηση ταχύτητας τριών τάξεων μεγέθους είναι ασήμαντο.

Επίδραση της κωδικοποίησης

- Βασικό μέγεθος που μπαίνει στους παραπάνω υπολογισμούς είναι το μήκος της εισόδου για το πρόβλημα, δηλαδή το μήκος της περιγραφής των απαραίτητων δεδομένων (π.χ. το μήκος της σε bits).
- Το μήκος μπορεί να διαφέρει ανάλογα με την κωδικοποίηση.
- Αποδεικνύεται εύκολα ότι τα διαφορετικά συστήματα κωδικοποίησης δίνουν μεν διαφορετικά μήκη περιγραφής, αλλά οι μεταξύ τους μετατροπές έχουν πολυωνυμική πολυπλοκότητα.
- Κατά συνέπεια η ύπαρξη πολυωνυμικής λύσης για ένα σύστημα κωδικοποίησης δίνει πολυωνυμική λύση για όλα τα συστήματα κωδικοποίησης του ίδιου προβλήματος.

Σημείωση: Υπάρχουν προβλήματα, όπου το μήκος της λύσης είναι μη πολυωνυμικό. Όμως συνήθως η αναφορά σε μη πολυωνυμικό αλγόριθμο υπονοεί το χρόνο εκτέλεσης.

Πολυπλοκότητα προβλημάτων

- Η κατάταξη των αλγορίθμων μπορεί να μετατραπεί σε κατάταξη των *επιλύσιμων προβλημάτων* ως εξής:
  Ένα πρόβλημα είναι πολυωνυμικής πολυπλοκότητας εφόσον υπάρχει αντίστοιχος πολυωνυμικός αλγόριθμος.
- Δεν είναι επιλύσιμα όλα τα προβλήματα με την έννοια ότι για ορισμένα δεν υπάρχει αλγόριθμος (οποιασδήποτε πολυπλοκότητας) που να τα επιλύει.
- Για παράδειγμα, δεν υπάρχει αλγόριθμος που να μπορεί να αποφασίσει αν ένα αυθαίρετο πρόγραμμα με αυθαίρετη είσοδο μπορεί να τερματίσει [Tur36]. Τα προβλήματα αυτά χαρακτηρίζονται ως undecidable.
- Επίσης υπάρχουν προβλήματα που δεν επιλύονται σε πολυωνυμικό χρόνο ακόμη και από ένα μη αιτιοκρατικό υπολογιστή που μπορεί να χειριστεί απεριόριστο αριθμό από ανεξάρτητες υπολογιστικές ακολουθίες εν παραλλήλω. Τα προβλήματα αυτά λέγονται non-deterministically intractable.
- Ωστόσο τα περισσότερα «δύσκολα» προβλήματα μπορούν να λυθούν σε πολυωνυμικό χρόνο από μια μη αιτιοκρατική μηχανή.

Προβλήματα NP-πλήρη (NP-complete)

- Η δημιουργία της κατηγορίας την προβλημάτων που ονομάζονται NP-πλήρη κατ' αρχήν δεν επιλύει αυστηρά το πρόβλημα της δυσκολίας των προβλημάτων, που κατατάσσονται σ' αυτήν.
- Τα κατατάσσει σε μια κατηγορία, για την οποία υπάρχουν αποχρώσες ενδείξεις δυσκολίας, αλλά μέχρι τώρα κανείς δεν κατάφερε να αποδείξει ότι τα προβλήματα αυτής της κατηγορίας ανήκουν εν τέλει στα πολυωνυμικά ή στα εκθετικά.
- Ωστόσο ανήκουν όλα στην ίδια κατηγορία, με την έννοια ότι αν ένα δειχθεί πολυωνυμικό, όλα θα είναι πολυωνυμικά. Αν ένα δειχθεί εκθετικό, όλα θα είναι εκθετικά.

Προβλήματα απόφασης

- Η περιγραφή ενός προβλήματος Π συνήθως περιλαμβάνει διάφορα δομικά στοιχεία, όπως σύνολα, γράφους, συναρτήσεις κλπ.
- Κατόπιν τίθεται μια ερώτηση, που στην περίπτωση ενός προβλήματος *απόφασης* έχει δύο μόνο απαντήσεις, «ναι» ή «όχι».
- Όταν όλα τα δομικά στοιχεία του προβλήματος Π πάρουν συγκεκριμένες τιμές, έχουμε μια *περίσταση* του προβλήματος.
- Έστω D_Π το σύνολο των περιστάσεων. Δεδομένου ότι αυτή πλέον έχει μια συγκεκριμένη από τις δυο παραπάνω απαντήσεις, είτε ανήκει στο Y_Π, δηλαδή στις περιστάσεις που δέχονται καταφατική απάντηση, είτε στο $D_\Pi - Y_\Pi$.

Περιοδεύων Πωλητης (Traveling Salesman)

ΠΕΡΙΣΤΑΣΗ: Πεπερασμένο σύνολο

$$C = \{c_1, c_2, \ldots, c_m\}$$

«πόλεων», «απόσταση» $d(c_i, c_j) \in Z^+$ για κάθε ζεύγος πόλεων $(c_i, c_j) \in C^2$ και ένα φράγμα $B \in Z^+$.
ΕΡΩΤΗΣΗ: Υπάρχει ένας «γύρος» σε όλες τις πόλεις του C με συνολικό μήκος μικρότερο ή ίσο του B, δηλαδή διάταξη

$$< c_{\pi(1)}, c_{\pi(2)}, \ldots, c_{\pi(m)} >$$

του C τέτοια ώστε

$$\sum_{i=1}^{m-1} d(c_{\pi(i)}, c_{\pi(i+1)}) + d(c_{\pi(m)}, c_{\pi(1)}) \leq B \qquad ?$$

Προβλήματα απόφασης και προβλήματα βελτιστοποίησης

- Ένα πρόβλημα απόφασης μπορεί να μετατραπεί σε πρόβλημα βελτιστοποίησης και αντίστροφα.
- Το αντίστοιχο με το προηγούμενο πρόβλημα του περιοδεύοντος πωλητή πρόβλημα βελτιστοποίησης δεν θα περιλάμβανε την παράμετρο B (άνω φράγμα μήκους γύρου) και θα ζητούσε την ελαχιστοποίηση του μήκους του γύρου.
- Αντίστροφα, το πρόβλημα ελαχιστοποίησης μιας συνάρτησης $c(x_1, x_2, \ldots, x_m)$ μετασχηματίζεται σε πρόβλημα απόφασης αν προστεθεί η παράμετρος και τεθεί η ερώτηση κατά πόσον ισχύει ότι $c(x_1, x_2, \ldots, x_m) \leq B$.

Προβλήματα απόφασης και προβλήματα βελτιστοποίησης (II)

Συνήθως στις αποδείξεις ότι ένα πρόβλημα είναι NP-πλήρες αυτό διατυπώνεται ως πρόβλημα απόφασης και όχι ως πρόβλημα βελτιστοποίησης. Δύο λόγοι εξηγούν αυτή την επιλογή:

1. Το πρόβλημα βελτιστοποίησης είναι τουλάχιστον όσο δύσκολο είναι το πρόβλημα απόφασης. Πράγματι, αν π.χ. λύσουμε το πρόβλημα ελαχιστοποίησης της αντικειμενικής συνάρτησης $c(.)$ και υπολογίσουμε το ελάχιστό της b, για να λύσουμε το αντίστοιχο πρόβλημα απόφασης αρκεί να συγκρίνουμε το b με το B.
2. Τα προβλήματα απόφασης έχουν το φυσικό τους αντίστοιχο στη θεωρία υπολογισμού, δηλαδή στην έννοια της *γλώσσας*.

Γλώσσες και σχήματα κωδικοποίησης

- Για κάθε πεπερασμένο σύνολο Σ, το σύνολο Σ^* περιέχει όλες τις σειρές από στοιχεία του Σ. Π.χ. αν $\Sigma = \{0, 1\}$, στο Σ^* περιέχονται η κενή συμβολοσειρά ϵ, οι συμβολοσειρές 0,1,00,01,10,11, 000, 001 κ.ο.κ.
- Γλώσσα (επί του Σ) λέγεται οποιοδήποτε υποσύνολο του Σ^*. Π.χ. το σύνολο $\{0, 1, 00, 01, 10, 11\}$ είναι μια γλώσσα επί του $\{0, 1\}$.
- Το σχήμα κωδικοποίησης e, με το οποίο περιγράφεται κάθε περίσταση ενός προβλήματος Π, χρησιμοποιεί σύμβολα από ένα αλφάβητο Σ.
- Το αντίστοιχο σύνολο Σ^* διαιρείται με αυτόν τον τρόπο σε τρία υποσύνολα, που περιέχουν αντίστοιχα
  1. τις συμβολοσειρές που δεν περιγράφουν περιστάσεις του Π,
  2. τις συμβολοσειρές που περιγράφουν περιστάσεις του Π, δηλαδή $I \in D_\Pi$, όπου D_Π το σύνολο των περιστάσεων του (προβλήματος απόφασης) Π, αλλά επί πλέον παίρνουν θετική απάντηση ($I \in Y_\Pi \subseteq D_\Pi$) και
  3. τις συμβολοσειρές που περιγράφουν περιστάσεις I του Π με αρνητική απάντηση ($I \in D_\Pi - Y_\Pi$).

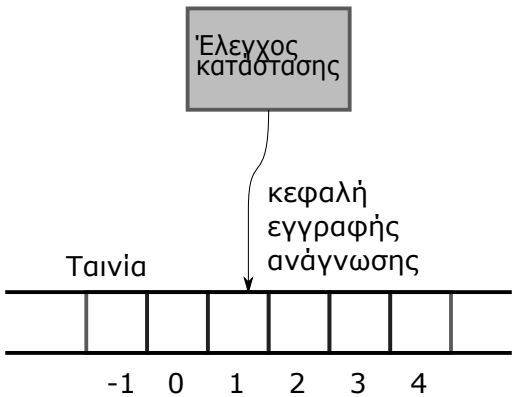
Η χρήση της έννοιας της γλώσσας στην απόδειξη ιδιοτήτων

- Θεωρούμε τα strings που κωδικοποιούν περιστάσεις του Π (με βάση το σχήμα κωδικοποίησης e, που χρησιμοποιεί το αλφάβητο Σ) που παίρνουν καταφατική απάντηση. Το σύνολο αυτών των strings είναι προφανώς υποσύνολο του Σ^*, επομένως είναι μια γλώσσα πάνω στο Σ. Δηλώνουμε αυτή τη γλώσσαμε το σύμβολο $L[\Pi, e]$.
- Αυτή η γλώσσα αποτελεί το μέσο για την απόδειξη ορισμένων ιδιοτήτων ανεξάρτητων από την συγκεκριμένη κωδικοποίηση, ενώ παραμένουμε σε «λογικά» σχήματα κωδικοποίησης.

Το υπολογιστικό μοντέλο και η γλώσσα

- Για να μελετηθεί η πολυπλοκότητα ενός αλγορίθμου ή ενός προβλήματος είναι απαραίτητο να χρησιμοποιηθεί ένα μοντέλο του τρόπου με τον οποίο γίνονται υπολογισμοί.
- Το υπολογιστικό μοντέλο, που μπορεί να χρησιμοποιηθεί για το σκοπό της μελέτης της πολυπλοκότητας προβλημάτων και αποδίδει κατά τα γνωστά τη λειτουργία ενός υπολογιστή, είναι η ντετερμινιστική μηχανή Turing με μια ταινία (Deterministic one tape Turing Machine, DTM).
- Ένα πρόγραμμα μιας DTM *αναγνωρίζει* μια γλώσσα εξ ορισμού όταν για κάθε συμβολοσειρά της γλώσσας αυτής τερματίζει σε μια δεδομένη κατάσταση.
- Η αντιστοιχία μεταξύ μιας γλώσσας και ενός προβλήματος είναι επομένως η εξής:
  Ένα πρόγραμμα λύνει το πρόβλημα απόφασης Π με την κωδικοποίηση e αν το M δέχεται τη γλώσσα $L[\Pi, e]$.

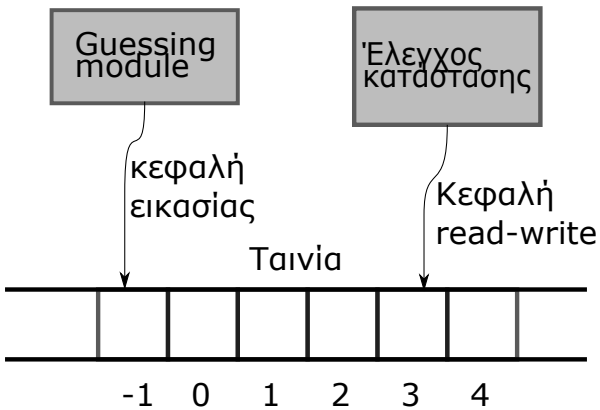
Η αιτιοκρατική μηχανή Turing



Ένα πρόγραμμα για μια αιτιοκρατική μηχανή Turing (DTM - Deterministic Turing Machine) καθορίζεται από τα εξής:

1. Ένα πεπερασμένο σύνολο Γ από σύμβολα ταινίας, ένα υποσύνολο συμβόλων «εισόδου» $\Sigma \subset \Gamma$ (που είναι τα σύμβολα που είναι γραμμένα στην ταινία αρχικά) και ένα «κενό» (blank) σύμβολο $b \in \Gamma - \Sigma$.
2. Ένα πεπερασμένο σύνολο καταστάσεων Q, που περιλαμβάνει μια αρχική κατάσταση q_0 και δύο καταστάσεις διακοπής (halt states) q_Y και q_N.
3. Μια συνάρτηση μετάβασης $\delta : (Q - \{q_Y, q_N\} \times \Gamma \to Q \times \Gamma \times \{-1, +1\}$. Δηλαδή λαμβάνοντας υπόψη το σύμβολο στην τρέχουσα θέση της κεφαλής και την τρέχουσα κατάσταση (α) επιλέγεται η επόμενη κατάσταση, (β) γράφεται ένα νέο σύμβολο (ή κενό) στην τρέχουσα θέση και μετά η κεφαλή κινείται μια θέση αριστερά ή δεξιά.

Η μη αιτιοκρατική μηχανή Turing



Ένα πρόγραμμα για μια μη αιτιοκρατική μηχανή Turing έχει ως εξής:

1. Δεδομένης μιας ακολουθίας (string) εισόδου $x \in \Sigma^*$, που είναι γραμμένη στις θέσεις από 1 ως $|x|$ της ταινίας, το «τμήμα εικασίας» (guessing module) κατευθύνει την κεφαλή εγγραφής να γράψει κάποιο σύμβολο από το Γ στην τρέχουσα θέση και να κατόπιν να κινηθεί μια θέση αριστερά ή να σταματήσει.
2. Στη συνέχεια παίρνει τον έλεγχο το αιτιοκρατικό μέρος και λειτουργεί ως DTM.

Μαντεύοντας και δοκιμάζοντας λύσεις

- Μερικά προβλήματα είναι δύσκολο να λυθούν, αλλά αν προταθεί μια λύση μπορεί εύκολα να διαπιστωθεί κατά πόσο η λύση αυτή είναι αποδεκτή.
- Στο παράδειγμα του περιοδεύοντος πωλητή οποιοδήποτε μονοπάτι περνάει από όλες τις πόλεις είναι υποψήφια λύση. Στη συνέχεια είναι εύκολο να υπολογισθεί το μήκος του και να διαπιστωθεί αν είναι μικρότερο από το φράγμα B.

Μη αιτιοκρατικός αλγόριθμος

- Ας υποτεθεί μη αιτιοκρατικός αλγόριθμος που αποτελείται
  (a) από μια φάση εικασίας της λύσης και
  (b) από μια φάση δοκιμής της λύσης.
- Κατά την πρώτη φάση η μηχανή διαβάζει τη δεδομένη περίσταση I του προβλήματος και παράγει στην έξοδό της μια δομή S (πιθανή λύση). Στο πρόβλημα του περιοδεύοντος πωλητή πρόκειται για ένα οποιοδήποτε γύρο S που περνάει από όλες τις πόλεις.
- Στη δεύτερη φάση, αμφότερα τα I, S εισάγονται στον αλγόριθμο, ο οποίος υπολογίζει κατά πόσο η απάντηση είναι θετική ή αρνητική.
- Η πολυπλοκότητα αυτού του αλγορίθμου δύο φάσεων ονομάζεται *μη αιτιοκρατική πολυωνυμική (non-deterministic polynomial)* αν για κάθε $I \in Y_\Pi$ υπάρχει ένα S τέτοιο ώστε η φάση δοκιμής να παράγει μια θετική απάντηση σε χρόνο πολυωνυμικό ως προς το μήκος της περίστασης.
- Αυτή η απαίτηση περιορίζει και το χρόνο εκτέλεσης και το μήκος της δομής.

Ο χώρος NP

- Αν ένας αιτιοκρατικός αλγόριθμος μπορεί να λύσει το πρόβλημα του περ. πωλητή, μπορεί επίσης να απαντήσει την ερώτηση «είναι αληθές ότι δεν υπάρχει γύρος μικρότερος από B;»
- Αν η απάντηση είναι θετική, προκύπτει αμέσως ότι η απάντηση στο αρχικό πρόβλημα είναι αρνητική.
- Όμως, ο μη αιτιοκρατικός αλγόριθμος για να λύσει το αντίστροφο πρόβλημα πρέπει να εξετάσει όλους τους δυνατούς γύρους, πράγμα αδύνατο εφόσον καθένας προκύπτει με κλήρωση.
- Επομένως υπάρχει μια ασυμμετρία στην περίπτωση του μη αιτιοκρατικού αλγόριθμου, αφού δεν μπορεί να επιλύσει το αντίστροφο πρόβλημα.
- Ένα πρόβλημα που είναι επιλύσιμο από ένα μη αιτιοκρατικό πολυωνυμικό αλγόριθμο ανήκει εξ ορισμού στην κλάση των *μη αιτιοκρατικών πολυωνυμικών* προβλημάτων (Non-deterministic Polynomial problems – NP).

Η σχέση μεταξύ P και NP

- Το ζήτημα της σχέσης μεταξύ των κλάσεων P και NP είναι ένα διάσημο ανοιχτό μαθηματικό πρόβλημα.
- Μέχρι τώρα γνωρίζουμε ότι $P \subseteq NP$. Το ερώτημα είναι κατά πόσο ισχύει $P = NP$.
- Πιθανώς η πλειοψηφία πιστεύει ότι $P \subset NP$.[1]
- Υπάρχει επίσης το εξής θεώρημα: Αν $\Pi \in NP$, υπάρχει πολυώνυμο p τέτοιο ώστε το Π να μπορεί να λυθεί από αιτιοκρατικό αλγόριθμο πολυπλοκότητας $O(2^{p(n)})$.

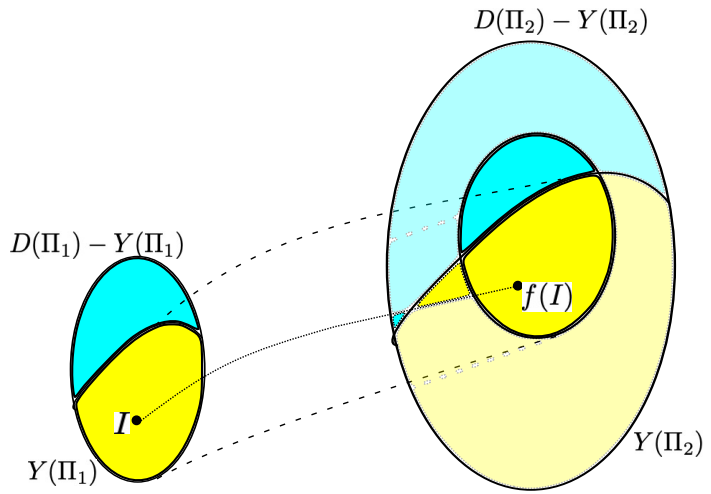
[1] http://www.newyorker.com/tech/elements/a-most-profound-math-problem

Πολυωνυμικοί μετασχηματισμοί

- Ο πολυωνυμικός μετασχηματισμός του προβλήματος απόφασης Π_1 στο πρόβλημα απόφασης Π_2 είναι μια συνάρτηση $f : D_{\Pi_1} \to D_{\Pi_2}$, που ικανοποιεί τους εξής περιορισμούς:
  - Η f είναι υπολογίσιμη με αλγόριθμο πολυωνυμικού χρόνου.
  - $\forall I \in D_{\Pi_1} : I \in Y_{\Pi_1} \Leftrightarrow f(I) \in Y_{\Pi_2}$

  Στην περίπτωση αυτή γράφουμε $\Pi_1 \propto \Pi_2$.
- Η παραπάνω σχέση είναι προφανώς μεταβατική, δηλαδή από τις $\Pi_1 \propto \Pi_2$ και $\Pi_2 \propto \Pi_3$ συνάγεται η $\Pi_1 \propto \Pi_3$.
- Στην περίπτωση που $\Pi_1 \propto \Pi_2$ και $\Pi_2 \propto \Pi_1$ τα δυο προβλήματα λέγονται *πολυωνυμικά ισοδύναμα*.

Η αναγωγή $\Pi_1 \propto \Pi_2$ με απλά λόγια



- Μετασχηματίζω το Π_1 σε μια ειδική μορφή του Π_2, δηλαδή κάθε περίσταση I του πρώτου σε μια περίσταση $f(I)$ του δεύτερου (ενώ το αντίστροφο μπορεί να μην ισχύει για κάθε περίσταση του δεύτερου).
- Αν ξέρω να λύσω το Π_2, τότε μπορώ να λύσω και το Π_1 (ενώ το αντίστροφο γενικά δεν ισχύει).
- Άρα το Π_2 είναι το λιγότερο όσο «δύσκολο» είναι το Π_1 ή ακόμη πιο δύσκολο.

NP-πλήρη προβλήματα

- 'Ενα πρόβλημα απόφασης Π λέγεται *NP-πλήρες* (NP-complete) αν $\Pi \in \text{NP}$ και για όλα τα άλλα προβλήματα απόφασης $\Pi' \in \text{NP}$ ισχύει $\Pi' \propto \Pi$ (δηλαδή μέσα στο NP δεν υπάρχει «πιο δύσκολο» πρόβλημα).
- Αν για ένα πρόβλημα $\Pi'' \in \text{NP}$ αποδειχθεί ότι $\Pi \propto \Pi''$, είναι κι αυτό NP-πλήρες. Αυτό έχει συμβεί ως τώρα για ένα μεγάλο αριθμό προβλημάτων.
- Αν ένα NP-πλήρες πρόβλημα αποδειχθεί πως είναι πολυωνυμικό, το ίδιο ισχύει αμέσως για όλα τα προβλήματα στο NP (περιλαμβανομένων και άλλων NP-πλήρων προβλημάτων, αφού και αυτά ανήκουν στο NP). Στην περίπτωση βέβαια αυτήν θα ισχύει $\text{P} - \text{NP} = \emptyset$.
- Αν ένα NP-πλήρες πρόβλημα αποδειχθεί εκθετικό, το ίδιο θα ισχύει για όλα τα NP-πλήρη προβλήματα, που είναι μεταξύ τους «εξ ίσου δύσκολα».

Εναλλακτικός τρόπος απόδειξης

Το να ακολουθήσει κανείς κατά γράμμα τον ορισμό του NP-πλήρους προβλήματος και να δείξει ότι στο NP δεν υπάρχει πιο δύσκολο πρόβλημα θα χρειαζόταν εξαιρετικά μεγάλο αριθμό από αναγωγές. Το επόμενο λήμμα δίνει μια λύση.

Λήμμα: Αν το Π_1 και το Π_2 ανήκουν αμφότερα στο NP κι αν το Π_1 είναι NP-πλήρες κι αν $\Pi_1 \propto \Pi_2$, το Π_2 είναι κι αυτό NP-πλήρες. $\square$

Η απόδειξη είναι εύκολη. Αρκεί να δειχθεί για οποιοδήποτε άλλο πρόβλημα $\Pi' \in$ NP ότι $\Pi' \propto \Pi_2$. Αφού το Π_1 είναι NP-πλήρες, ισχύει $\Pi' \propto \Pi_1$ και βάσει της υπόθεσης $\Pi_1 \propto \Pi_2$ και της μεταβατικής ιδιότητας ισχύει ότι $\Pi' \propto \Pi_2$. Κατά συνέπεια προκειμένου να δείξει κανείς ότι ένα πρόβλημα Π είναι NP-πλήρες μπορεί απλώς και μόνο να δείξει ότι

- $\Pi \in$ NP,
- για κάποιο γνωστό NP-πλήρες πρόβλημα Π' ισχύει $\Pi' \propto \Pi$.

Για να ξεκινήσει η γέννηση τέτοιων προβλημάτων χρειάζεται ένα πρώτο NP-πλήρες πρόβλημα, που προφανώς το παραπάνω λήμμα αδυνατεί να το δώσει. Το πρώτο αυτό πρόβλημα το παρέχει το περίφημο θεώρημα του Cook.

Το θέωρημα Cook-Levin, I

- Προκειμένου να διατυπωθεί το πρόβλημα χρειάζονται δυο λόγια για τον συμβολισμό.
  Αν $u_1, u_2, \ldots, u_n$ είναι λογικές μεταβλητές, ο συμβολισμός $\{u_1, u_2, \ldots, u_n\}$ υποδηλώνει την πρόταση που είναι αληθής αν μια τουλάχιστον από τις $u_1, u_2, \ldots, u_n$ είναι αληθής ("u_1 είτε u_2 είτε..."). Επίσης η $\overline{u}$ είναι αληθής όταν και μόνον αν η u είναι ψευδής.
- Μια συλλογή προτάσεων C θεωρείται ικανοποιήσιμη αν υπάρχει τουλάχιστον μια ανάθεση τιμών αλήθειας στις εμπλεκόμενες μεταβλητές, που καθιστά αληθείς όλες τις προτάσεις της συλλογής.
- Π.χ. η συλλογή προτάσεων

$$C = \{\{u_1, \overline{u}_2\}, \{\overline{u}_1, u_2\}\}$$

είναι ικανοποιήσιμη δεδομένου ότι αν αμφότερες οι u_1, u_2 είναι αληθείς, οι δυο προτάσεις $\{u_1, \overline{u}_2\}$ και $\{\overline{u}_1, u_2\}$ είναι επίσης αληθείς.

Το θέωρημα Cook-Levin, II

- Κατόπιν όλων αυτών το πρόβλημα για το ικανοποιήσιμο μιας συλλογής είναι το εξής:
  Πρόβλημα: **SATISFIABILITY** (SAT)
  ΠΕΡΙΣΤΑΣΗ: Σύνολο λογικών μεταβλητών U και συλλογή C προτάσεων επί του συνόλου U.
  ΕΡΩΤΗΣΗ: Υπάρχει ικανοποιούσα ανάθεση τιμών αληθείας για την C;
- *Θεώρημα*: Το πρόβλημα SATISFIABILITY είναι NP-πλήρες.
- Το παραπάνω πρόβλημα αποδεικνύεται εύκολα ότι ανήκει στο NP. Η πλήρης απόδειξη είναι εκτεταμένη και ο αναγνώστης παραπέμπεται στο βιβλίο των Garey και Johnson.

Ένα πολύ χονδρικό σκίτσο απόδειξης

- Το SAT είναι στο NP αφού ένας μη αιτιοκρατικός αλγόριθμος αρκεί μόνο να μαντέψει μια αντιστοίχιση των μεταβλητών στις τιμές TRUE/FALSE και να εξετάσει αν ικανοποιεί όλες τις προτάσεις.
- Το SAT περιγράφεται από μια γλώσσα $L_{\text{SAT}} = L(\text{SAT}, e)$ για κάποιο λογικό σχήμα κωδικοποίησης e. Πρέπει να δείξουμε ότι $\forall L \in \text{NP} : L \propto L_{\text{SAT}}$.
- Αφού κάθε γλώσσα L_M in NP μπορεί να περιγραφεί από μια NDTM μηχανή M πολυωνυμικού χρόνου που θα αναγνωρίζει την L_M, αρκεί να προδιαγράψει κανείς μια έκφραση Boole που «προσομοιώνει» τη μηχανή, δηλαδή είναι ικανοποιήσιμη αν και μόνον αν η μηχανή M αναγνωρίζει τη γλώσσα L_M.
- Ένα τελικό σημείο στην απόδειξη είναι ότι ο μετασχηματισμός αυτός είναι πολυωνυμικός.

Μέθοδος απόδειξης ότι ένα πρόβλημα Π είναι NP-complete

Ανακεφαλαιώνοντας, η συνήθης πρακτική για να αποδειχθεί ότι ένα πρόβλημα Π είναι NP-πλήρες περιλαμβάνει τα εξής τρία βήματα:

1. Να δειχθεί ότι το Π ανήκει στην κλάση NP.
2. Να επιλεγεί ένα γνωστό NP-πλήρες πρόβλημα Π' και να δειχθεί ότι υπάρχει μετασχηματισμός f από το το Π' στο Π.
3. Να δειχθεί ότι ο μετασχηματισμός f είναι πολυωνυμικός.

Πρόβλημα NP-δυσχερές (NP-hard)

- Το βήμα (1) από τα παραπάνω τρία βήματα απόδειξης ότι ένα πρόβλημα είναι NP-πλήρες μπορεί να μη μπορεί να γίνει για ορισμένα προβλήματα.
- Στην περίπτωση αυτή το πρόβλημα ναι μεν δεν είναι NP-πλήρες, αλλά και δεν μπορεί να λυθεί σε πολυωνυμικό χρόνο, εκτός αν P=NP.
- Ένα τέτοιο πρόβλημα καλείται *NP-δυσχερές (NP-hard)* και είναι τουλάχιστο τόσο δύσκολο όσο δύσκολο είναι ένα NP-πλήρες πρόβλημα.
- Συχνά η μελέτη ενός προβλήματος περιορίζεται στην αναγωγή ενός NP-πλήρους προβλήματος σε αυτό, οπότε απλώς αποδεικνύεται ότι είναι NP-δυσχερές.

Συλλογές προβλημάτων

- Σήμερα πλήθος προβλημάτων έχουν δειχθεί ότι είναι NP-πλήρη. Ευρείες συλλογές προβλημάτων μπορούν να βρεθούν στο βιβλίο των Garey και Johnson [GJ79] και στο βιβλίο της Dorit Hochbaum [Hoc96], καθώς και online.
- Πέραν της κατάταξης των συγκεκριμένων προβλημάτων, που εμφανίζονται στις συλλογές, ο σκοπός της δημοσίευσης τέτοιων συλλογών είναι η διευκόλυνση των ερευνητών νέων προβλημάτων στην επιλογή ενός βολικού προβλήματος Π', προκειμένου να κάνουν την αναγωγή από το Π' στο πρόβλημα Π, το οποίο υποψιάζονται ότι είναι NP-πλήρες.
- Και μόνη εξ άλλου η ανάγνωση μιας σειράς προβλημάτων που είναι NP-πλήρη, δίνει πολύτιμη πείρα στον ερευνητή ενός νέου προβλήματος που αναρωτιέται κατά πόσο το δικό του πρόβλημα είναι πολυωνυμικό ή μη.

Βιβλιογραφία

S. Arora and B. Barak. *Computational complexity: a modern approach*. Cambridge University Press, 2009.

A. Cobham. "The intrinsic computational difficulty of functions". In: *International Congress for Logic Methodology and Philosophy of Science*. Amsterdam: North Holland, 1965, pp. 24–30.

M. R. Garey and D. S. Johnson. *Computers and Intractability, a Guide to the Theory of NP-Completeness*. WH Freeman and Co, San Francisco, 1979.

D. S. Hochbaum. *Approximation algorithms for NP-hard problems*. PWS Publishing Co., 1996.

A. M. Turing. "On computable numbers, with an application to the Entscheidungsproblem". In: *J. of Math* 58.345-363 (1936), p. 5.