ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ

Αλγόριθμοι Τοπικού Υπολογισμού και Εφαρμογές τους

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

του

Ματίκα Γεώργιου

Επιβλέπων: Δημήτριος Φωτάκης
Επίκουρος Καθηγητής Ε.Μ.Π.

Αθήνα, Σεπτέμβριος 2016

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ

Αλγόριθμοι Τοπικού Υπολογισμού και Εφαρμογές τους

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

του

Ματίκα Γεώργιου

Επιβλέπων: Δημήτριος Φωτάκης
Επίκουρος Καθηγητής Ε.Μ.Π.

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 5η Σεπτεμβρίου 2016.

.....	.....	.....
Δημήτριος Φωτάκης	Άρης Παγουρτζής	Νικόλαος Παπασπύρου
Επίκουρος Καθηγητής Ε.Μ.Π.	Αναπληρωτής Καθηγητής Ε.Μ.Π.	Αναπληρωτής Καθηγητής Ε.Μ.Π.

Αθήνα, Σεπτέμβριος 2016

.....

Ματίκας Γεώργιος

Διπλωματούχος Ηλεκτρολόγος Μηχανικός και Μηχανικός Υπολογιστών Ε.Μ.Π.

Περίληψη

Τα τελευταία χρόνια παρατηρείται το φαινόμενο τα σύνολα δεδομένων που δίνονται ως είσοδοι σε αλγόριθμους προς επεξεργασία να είναι ασύλληπτα μεγάλα. Λογικό είναι, λοιπόν, ο χρόνος που θα κάνει ο αλγόριθμος να τα επεξεργαστεί να είναι εξίσου πολύς. Πολλές φορές, όμως, χρειαζόμαστε τα αποτελέσματα σε σύντομο χρονικό διάστημα ώστε να πάρουμε άμεσα κάποια απόφαση. Γι' αυτό και το ενδιαφέρον έχει στραφεί πλέον προς τους αλγόριθμους που τρέχουν σε υπογραμμικό χρόνο σε σχέση με το μέγεθος της εισόδου. Στην περίπτωση που δεν μας ενδιαφέρει ολόκληρη η λύση του προβλήματος, αλλά αρκούμαστε σε ένα μικρό μέρος αυτής κάθε φορά, μπορούμε να χρησιμοποιήσουμε έναν αλγόριθμο τοπικού υπολογισμού οι οποίοι παρουσιάζονται στην εργασία αυτή. Παρουσιάζονται δύο παράλληλοι αλγόριθμοι που τρέχουν σε υπογραμμικό χρόνο. Στη συνέχεια δίνονται αλγόριθμοι τοπικού υπολογισμού για αρκετά σημαντικά γραφοθεωρητικά προβλήματα που βασίζονται στους προηγούμενους, και συγκεκριμένα για τα προβλήματα του maximal independent set, του 2-χρωματισμού υπεργράφου και του maximal matching. Συζητούνται, επίσης, εφαρμογές αυτού του είδους αλγορίθμων σε διάφορες περιοχές της Επιστήμης Υπολογιστών, όπως ο Σχεδιασμός Μηχανισμών και το Probabilistic Inference.

Abstract

In recent years, we observe the phenomenon that the datasets being given as input to algorithms are growing rapidly fast in size. As a consequence, the time needed for an algorithm to manipulate those kinds of inputs has grown a lot too. Most frequently, though, we need the results in a short amount of time in order to make some kind of decision. That's why research nowadays is being focused on algorithms that run in sublinear time. When we don't really care about the whole solution of a problem but we only need to learn a small portion of it we can use a local computation algorithm, which we define in this thesis. We start by presenting two parallel algorithms that run in sublinear time on multiple machines concurrently. We continue by giving local computation algorithms for some very important graph-theoretic problems that are based on those previous parallel algorithms. More specifically, we focus on the problem of maximal independent set, of hypergraph 2-coloring and of maximal matching. We also discuss some applications of this new kind of algorithms on various areas of Computer Science, like Mechanism Design and Probabilistic Inference.

Περιεχόμενα

Κατάλογος Σχημάτων

Κεφάλαιο 1

Εισαγωγή

Μέχρι πρόσφατα, όταν αναφερόμασταν στον όρο αλγόριθμο, εννοούσαμε την κλασική έννοια του αλγορίθμου. Κατά την έννοια αυτή, ένας αλγόριθμος διαβάζει μια είσοδο, εκτελεί κάποιους υπολογισμούς και στο τέλος τυπώνει μια έξοδο. Στη σημερινή εποχή της πολύ εύκολης πρόσβασης στο Internet και των τεράστιων data centers τα σύνολα δεδομένων που αποτελούν είσοδοι διαφόρων αλγορίθμων έχουν αποκτήσει και αυτά ασύλληπτες διαστάσεις. Σύμφωνα με τα τελευταία στατιστικά δεδομένα, το eBay — το μεγαλύτερο site δημοπρασιών στον κόσμο — διαθέτει κατά μέσο όρο 700 εκατομμύρια προϊόντα προς πώληση σε περίπου 155 εκατομμύρια ενεργούς χρήστες. Άλλη μια εταιρία-μεγαθήριο, η Google, παρέχει χώρους διαφημίσεων σε εκατομμύρια ιστοσελίδες που το επιθυμούν, για τους οποίους ανταγωνίζονται εκατομμύρια διαφημιζόμενοι.

Λογικό είναι, λοιπόν, ένας αλγόριθμος που επεξεργάζεται τέτοιου είδους δεδομένα να αργεί υπερβολικά μόνο και μόνο για να τα διαβάσει, καθιστώντας τον μη λειτουργικό για ιδιαίτερα συχνή χρήση. Για να μας είναι ένας αλγόριθμος χρήσιμος θα πρέπει να παράγει το αποτέλεσμα που επιθυμούμε σε ένα εύγλωττο, σχετικά, χρονικό διάστημα. Θα πρέπει, επομένως, να τρέχει σε χρόνο υπογραμμικό — sublinear — (πχ πολυλογαριθμικό) σε σχέση με το μέγεθος της εισόδου. Οι αλγόριθμοι αυτοί αποτελούν μια ιδιαίτερη κατηγορία αλγορίθμων, τους Υπογραμμικούς Αλγόριθμους. Στο Κεφάλαιο 2 αναφερόμαστε εκτενέστερα στους αλγόριθμους αυτούς καθώς και στη χρησιμότητά τους.

Διάφορες τεχνικές χρησιμοποιούνται με σκοπό την εκτέλεση ενός αλγορίθμου σε χρόνο υπογραμμικό. Οι δύο πιο συνηθισμένες, παραδείγματα των οποίων θα δούμε έμπρακτα και στην εργασία αυτή, είναι η εκτέλεση ενός αλγορίθμου σε περισσότερα του ενός μηχανήματα ταυτόχρονα, κερδίζοντας έτσι χρόνο, καθώς και η προσέγγιση — και όχι εύρεση ακριβώς — της βέλτιστης λύσης, διαβάζο-

ντας μόνο τμήμα της εισόδου. Η χρήση των τεχνικών αυτών ορίζει δύο νέες κατηγορίες αλγορίθμων, τους Παράλληλους Αλγόριθμους (parallel/distributed algorithms) και τους Προσεγγιστικούς Αλγόριθμους (approximation algorithms) αντίστοιχα. Ιδιαίτερη αναφορά στις κατηγορίες αλγορίθμων αυτές θα κάνουμε επίσης στο Κεφάλαιο 2.

Στη διπλωματική εργασία αυτή παρουσιάζεται μια τρίτη τεχνική με στόχο την επίλυση του παραπάνω ζητήματος. Η τεχνική που παρουσιάζεται βασίζεται στην ακόλουθη παρατήρηση. Πολλές φορές δεν μας είναι απαραίτητη ολόκληρη η λύση ενός προβλήματος, αλλά μας ενδιαφέρει μόνο ένα συγκεκριμένο μέρος αυτής. Για παράδειγμα, το eBay δε χρειάζεται να ξέρει ανά πάσα χρονική στιγμή για όλα τα αντικείμενα σε ποιον θα πωληθούν και πόσο, αλλά σε κάθε χρονική στιγμή ενδιαφέρεται το πολύ για ένα συγκεκριμένο προϊόν (πχ όταν λήγει μια δημοπρασία). Επομένως, ο αλγόριθμος δε χρειάζεται να διαβάσει και να επεξεργαστεί ολόκληρο το dataset με τους ενεργούς χρήστες και τα διαθέσιμα προϊόντα για να βρει την πλήρη λύση και να μας απαντήσει, παρά μόνο να εξετάσει τους χρήστες οι οποίοι ενδιαφέρθηκαν για το προϊόν αυτό — ο αριθμός των οποίων λογικά είναι πολύ μικρότερος του συνολικού.

Οι αλγόριθμοι αυτοί ονομάζονται *Αλγόριθμοι Τοπικού Υπολογισμού* (Local Computation Algorithms) καθώς εκτελούν υπολογισμούς μόνο γύρω από το ζητούμενο τμήμα της λύσης και όχι σε ολόκληρο το δοθέν σύνολο δεδομένων. Δέχονται ερωτήματα σχετικά με ένα μέρος της λύσης του προβλήματος κάθε φορά και όχι για ολόκληρη τη λύση. Αν ένα πρόβλημα έχει πολλές δυνατές λύσεις, τότε όλες οι απαντήσεις του αλγορίθμου θα πρέπει να είναι συνεπείς με τουλάχιστον μία δεκτή λύση. Στόχος είναι οι συγκεκριμένοι αλγόριθμοι να τρέχουν σε χρόνο ανάλογο με το μέγεθος της απάντησης που αναμένουμε από αυτούς και όχι με το συνολικό μέγεθος της εισόδου.

Για να ορίσουμε τους Αλγόριθμους Τοπικού Υπολογισμού και να αναπτύξουμε τη νοοτροπία πίσω από την τεχνική αυτή επιλέξαμε δύο πολύ σημαντικά προβλήματα του χώρου της Επιστήμης Υπολογιστών και παρουσιάζουμε ΑΤΥ που σκοπό έχουν την επίλυσή τους. Τα προβλήματα αυτά είναι η εύρεση ενός Maximal Independent Set σε έναν δοσμένο γράφο και ο 2-Χρωματισμός ενός Υπεργράφου, τα οποία ορίζονται επακριβώς στο Κεφάλαιο 3. Τα συγκεκριμένα προβλήματα, ωστόσο, δεν αντιμετωπίζονται για πρώτη φορά εδώ αλλά υπάρχουν ήδη αλγόριθμοι οι οποίοι δίνουν ακριβείς λύσεις για το καθένα. Συγκεκριμένα, και για τα δύο προβλήματα παρουσιάζουμε στο Κεφάλαιο 4 από έναν παράλληλο αλγόριθμο — τον αλγόριθμο του Luby για το maximal independent set και τον αλγόριθμο του Alon για το 2-χρωματισμό υπεργράφου — πάνω στη λογική των οποίων βασίστηκαν και οι αλγόριθμοι τοπικού υπο-

λογισμού που θα παρουσιάσουμε. Η διαφορά έγκειται στο ότι οι τελευταίοι απαντούν σε ερωτήματα σχετικά με τη βέλτιστη λύση σε χρόνο σημαντικά μικρότερο από το χρόνο που χρειάζονται οι αντίστοιχοι παράλληλοι αλγόριθμοι για να δώσουν ολόκληρη τη βέλτιστη λύση.

Στο Κεφάλαιο 5 ορίζεται πλέον το μοντέλο των Αλγορίθμων Τοπικού Υπολογισμού και δίνονται επακριβώς οι αλγόριθμοι για τα δύο προβλήματα που αναφέραμε, αναλύοντας τις επιμέρους φάσεις του κάθε αλγορίθμου και τη χρονική τους πολυπλοκότητα. Οι αλγόριθμοι αυτοί, στην αρχική τους μορφή, τρέχουν σε υπογραμμικό χρόνο — όπως ακριβώς θα θέλαμε — αλλά χρησιμοποιούν αρκετή μνήμη για το σκοπό αυτό. Στο Κεφάλαιο 6 παρουσιάζουμε κάποιες τεχνικές οι οποίες βοηθούν ώστε να μειώσουμε την κατανάλωση μνήμης από τους αλγορίθμους αυτούς σε υπογραμμικά επίσης επίπεδα. Στη συνέχεια του Κεφαλαίου 6 δίνονται οι βελτιωμένοι πια ΑΤΥ. Τέλος, στα Κεφάλαια 7 και 8 δίνονται εφαρμογές των αλγορίθμων τοπικού υπολογισμού σε άλλες ερευνητικές περιοχές και συγκεκριμένα στις περιοχές του Σχεδιασμού Μηχανισμών και του Probabilistic Inference. Παρουσιάζουμε, λοιπόν, ΑΤΥ για συγκεκριμένα προβλήματα της εκάστοτε περιοχής. Μια νίξη για το τι αντιπροσωπεύει η κάθε περιοχή από αυτές δίνεται στο εισαγωγικό Κεφάλαιο 2.

Κεφάλαιο 2

Υπόβαθρο

Στην ενότητα αυτή θα αναφερθούμε στις σημαντικότερες κατηγορίες αλγορίθμων που συναντάμε στη διπλωματική εργασία αυτή καθώς και στις περιοχές του Σχεδιασμού Μηχανισμών και του Probabilistic Inference, όπου βρίσκουν εφαρμογή οι Αλγόριθμοι Τοπικού Υπολογισμού που θα αναπτύξουμε.

2.1 Υπογραμμικοί Αλγόριθμοι

Μέχρι πρόσφατα, η εύρεση ενός αλγορίθμου γραμμικού χρόνου κατά την προσπάθεια επίλυσης ενός προβλήματος θεωρούνταν ιδιαίτερα ικανοποιητικό επίτευγμα. Έτσι, οι έρευνες επικεντρώνονταν στην εύρεση αλγορίθμων που τρέχουν σε γραμμικό χρόνο σε σχέση με την είσοδο (πχ $O(n^2)$). Ωστόσο, όπως αναφέραμε και παραπάνω, στις μέρες μας τα σύνολα δεδομένων που δίδονται ως είσοδοι στους διάφορους αλγορίθμους έχουν αποκτήσει ασύλληπτες διαστάσεις. Οι παραπάνω αλγόριθμοι, λοιπόν, θα χρειαστούν υπερβολικά πολύ χρόνο για να επεξεργαστούν το σύνολο των δεδομένων εισόδου και πολλές φορές χρειαζόμαστε τα αποτελέσματα άμεσα ή διαθέτουμε περιορισμένο χρόνο για τη λήψη μιας απόφασης. Η μόνη λύση τότε είναι η χρήση αλγορίθμων που τρέχουν σε υπογραμμικό χρόνο [55, 1]. Τυπικά, ένας αλγόριθμος λέμε ότι τρέχει σε υπογραμμικό χρόνο όταν $T(n) = o(n)$. Τα τελευταία χρόνια έχει λάβει χώρα εκτεταμένη έρευνα σε αλγορίθμους υπογραμμικού χρόνου, ιδιαίτερα στην περιοχή των προβλημάτων βελτιστοποίησης και του property testing (μια εναλλακτική μορφή προσέγγισης για προβλήματα απόφασης).

Οι αλγόριθμοι που τρέχουν σε υπογραμμικό χρόνο είναι κυρίως είτε *παράλληλοι αλγόριθμοι* είτε *προσεγγιστικοί αλγόριθμοι* είτε αλγόριθμοι για *property*

testing. Συγκεκριμένα, οι παράλληλοι αλγόριθμοι είναι η μόνη κατηγορία αλγορίθμων από τις παραπάνω που μπορούν να τρέξουν σε υπογραμμικό χρόνο και παρόλα αυτά να δώσουν ακριβή αποτελέσματα, χωρίς να έχει γίνει κάποια υπόθεση για τη μορφή της εισόδου. Το πετυχαίνουν αυτό τρέχοντας ταυτόχρονα ανεξάρτητα τμήματα του κώδικά τους σε πολλούς διαφορετικούς επεξεργαστές ή μηχανήματα, μοιράζοντας έτσι το φόρτο εργασίας και εξοικονομώντας χρόνο. Από την άλλη, οι δύο τελευταίες κατηγορίες συνδέονται μεταξύ τους στενά επειδή και οι δύο βασίζονται γερά στην πιθανοτική εκτέλεσή τους και στη διαδικασία της τυχαίας δειγματοληψίας. Για να τρέξουν οι αλγόριθμοι αυτοί σε υπογραμμικό χρόνο ο μοναδικός τρόπος είναι να διαβάσουν μόνο τμήμα της εισόδου και να βγάλουν συμπεράσματα με βάση αυτό το μικρό δείγμα. Για τα περισσότερα φυσικά προβλήματα, αυτό σημαίνει ότι το αποτέλεσμα που θα παράξει ο εκάστοτε αλγόριθμος θα είναι προσεγγιστικό και θα ενέχει ένα μικρό βαθμό αβεβαιότητας. Θα αναφερθούμε πιο αναλυτικά στις δύο πρώτες κατηγορίες από αυτές παρακάτω.

Πέρα από τις παραπάνω κατηγορίες, συχνά ως υπογραμμικοί αλγόριθμοι θεωρούνται και οι *data streaming* αλγόριθμοι, μια ειδική μορφή άμεσων (online) αλγορίθμων στους οποίους επίσης θα αναφερθούμε στη συνέχεια. Στους αλγόριθμους αυτούς θεωρούμε ότι η είσοδος έρχεται σαν μια ροή δεδομένων (stream). Θεωρούμε, επίσης, ότι διαθέτουν υπογραμμικό μέγεθος μνήμης σε σχέση με το μήκος της ροής δεδομένων και καθώς έρχονται τα δεδομένα αποφασίζουν ποια από αυτά θα κρατήσουν στη μνήμη για μελλοντική επεξεργασία και ποια όχι. Περισσότερα για αλγορίθμους data streaming μπορούν να βρεθούν στα κείμενα [48, 10]. Υπάρχουν, ακόμη, αλγόριθμοι που τρέχουν σε υπογραμμικό χρόνο δοθέντος, όμως, κάποιων συγκεκριμένων υποθέσεων για την είσοδο (έχει πχ κάποια συγκεκριμένη μορφή ή δομή). Το πιο χαρακτηριστικό παράδειγμα τέτοιου αλγορίθμου είναι η *δυαδική αναζήτηση* (binary search) κατά την οποία αναζητείται η θέση ενός στοιχείου σε μια λίστα, δεδομένου όμως ότι η λίστα αυτή είναι ταξινομημένη. Ο αλγόριθμος δυαδικής αναζήτησης τρέχει σε λογαριθμικό χρόνο σε σχέση με το μέγεθος της εισόδου $O(\log n)$, καθώς δεν ελέγχει όλα τα στοιχεία ένα ένα εκμεταλλευόμενος τη σειρά με την οποία αυτά είναι δοσμένα. Τέλος, υπάρχουν κβαντικοί αλγόριθμοι που θεωρητικά τρέχουν σε υπογραμμικό χρόνο, όπως ο αλγόριθμος αναζήτησης μη ταξινομημένης λίστας του Grover [25]. Ο αλγόριθμος αναζήτησης του Grover τρέχει σε χρόνο $O(n^{1/2})$.

Παράδειγμα αλγορίθμου που τρέχει σε υπογραμμικό χρόνο αποτελεί ο αλγόριθμος των Chazelle, Rubinfeld και Trevisan για τον υπολογισμό του βάρους του Minimum Spanning Tree ενός γράφου. Όπως είναι λογικό, το αποτέλε-

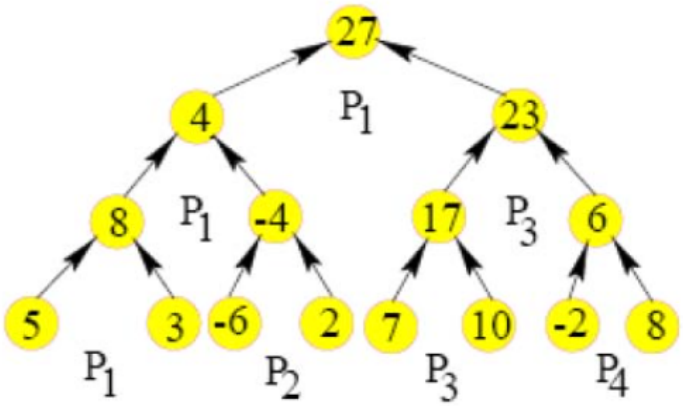
σμα του αλγόριθμου αυτού είναι προσεγγιστικό μιας και στο χρόνο αυτό δεν μπορεί να αναλύσει ολόκληρη την είσοδο που δέχεται. Δοθέντος ενός γράφου G με μέσο βαθμό κόμβων d και βάρη ακμών στο σύνολο $\{1, \ldots, w\}$ ο συγκεκριμένος αλγόριθμος δίνει το βάρος του ελάχιστου συνδετικού δέντρου με περιθώριο λάθους το πολύ ε, $0 < \varepsilon < 1/2$, σε χρόνο μόλις $O(dw\varepsilon^{-2} \log \frac{dw}{\varepsilon})$ [11].

2.2 Παράλληλοι Αλγόριθμοι

Μέχρι πριν μερικά χρόνια, η κυρίαρχη μόδα μεταξύ των εταιριών κατασκευής επεξεργαστών ήταν να αυξάνουν την ταχύτητα ρολογιού των επεξεργαστών τους ώστε να πετυχαίνουν αυξημένη απόδοση. Έτσι, εκτελούνταν περισσότερες εντολές ενός κλασικού σειριακού αλγόριθμου ανά δευτερόλεπτο και ο αλγόριθμος αυτός έβγαζε το αποτέλεσμά του γρηγορότερα. Ωστόσο, η ταχύτητα ρολογιού αυτή έφτασε σε ένα ανώτατο κατώφλι στην οποία μπορεί να φθάσει χωρίς να εμφανιστούν λειτουργικά προβλήματα που αποτρέπουν την περαιτέρω αύξησή της. Τα προβλήματα αυτά είναι εγγενή και οφείλονται στο ίδιο το υλικό κατασκευής των επεξεργαστών, δηλαδή στο πυρίτιο. Συγκεκριμένα, από κάποια ταχύτητα και πάνω η θερμοκρασία στην οποία φθάνει ο εκάστοτε επεξεργαστής είναι υπερβολικά μεγάλη για ένα φυσιολογικό σύστημα ψύξης και αρχίζουν να εμφανίζονται διαρροές ενέργειας. Η τωρινή τάση, λοιπόν, των εταιριών έγκειται στο να τοποθετούν περισσότερους του ενός πυρήνες ανά επεξεργαστή (ή περισσότερους του ενός επεξεργαστές σε μεγάλης κλίμακας συστήματα), οι οποίοι εκτελούν ταυτόχρονα διαφορετικές εντολές ο καθένας. Για να εκμεταλλευτούμε, όμως, τα οφέλη αυτών των πολυεπεξεργαστικών συστημάτων δεν αρκούν οι κλασικοί σειριακοί αλγόριθμοι. Θα πρέπει να σχεδιάσουμε αλγορίθμους οι οποίοι να ορίζουν πολλαπλές εντολές ανά βήμα οι οποίες και θα εκτελούνται παράλληλα σε διαφορετικούς επεξεργαστές η καθεμία, τους λεγόμενους δηλαδή *παράλληλους αλγόριθμους*.

Ένα πολύ απλό παράδειγμα προβλήματος που μπορεί να επιλυθεί χρησιμοποιώντας έναν παράλληλο αλγόριθμο είναι το άθροισμα ενός συνόλου αριθμών. Υπό την κλασική οπτική γωνία, για να αθροίσουμε n το πλήθος αριθμούς θα χρειαζόταν να εκτελέσουμε $n-1$ συνολικά προσθέσεις τη μία μετά την άλλη σε έναν επεξεργαστή. Αν είχαμε, όμως, στη διάθεσή μας περισσότερους του ενός επεξεργαστές τότε θα μπορούσαμε να τρέξουμε τον εξής παράλληλο αλγόριθμο: Χωρίζουμε τους αριθμούς σε ζευγάρια και τους προσθέτουμε παράλληλα ανά δύο σε διαφορετικό επεξεργαστή το κάθε ζευγάρι (έστω ότι έχουμε ιδα-

νικά $n/2$ επεξεργαστές). Έτσι σε μία μόλις μονάδα χρόνου καταλήγουμε σε $n/2$ αριθμούς που θα πρέπει τώρα να προσθέσουμε. Επαναλαμβάνοντας την ίδια διαδικασία θα καταλήξουμε σε $n/4$ αριθμούς προς άθροιση κοκ. έως ότου καταλήξουμε στο τελικό άθροισμα. Αφού σε κάθε βήμα το σύνολο των αριθμών προς άθροιση υποδιπλασιάζεται και κάθε βήμα παίρνει χρόνο $O(1)$ (δεδομένου ότι όλες οι προσθέσεις του βήματος αυτού γίνονται σε διαφορετικό επεξεργαστή η καθεμία) ο συνολικός χρόνος που θα χρειαστούμε είναι $\log n + 1 = O(\log n)$. Στο Σχήμα 2.1 φαίνεται μια εφαρμογή του αλγορίθμου για 8 συνολικά αριθμούς και 4 επεξεργαστές. Όπως βλέπουμε χρειάζεται χρόνος ίσος με $log8 + 1 = 4$ μονάδες αντί για 7 που θα χρειαζόμασταν για να κάνουμε τις προσθέσεις μία μία σειριακά.



Σχήμα 2.1: Παράδειγμα παράλληλης άθροισης αριθμών

Για την ανάλυση ενός οποιουδήποτε αλγορίθμου χρειαζόμαστε κάποιο μοντέλο υπολογισμού για να στηριχθούμε και να υπολογίσουμε τη χρονική και χωρική πολυπλοκότητά του. Όσον αφορά τους σειριακούς αλγόριθμους, το μοντέλου που χρησιμοποιείται κατά κόρον είναι η *μηχανή τυχαίας προσπέρασης* (random-access machine, RAM). Κατά το μοντέλο αυτό, η μηχανή αποτελείται από μια επεξεργαστική μονάδα και μια μονάδα μνήμης συνδεδεμένα μεταξύ τους. Κάθε λειτουργία της αριθμητικής και λογικής μονάδας και κάθε προσπέλαση της μνήμης παίρνει μια μονάδα χρόνου. Όσον αφορά τους παράλληλους αλγόριθμους, όμως, τα πράγματα είναι πιο πολύπλοκα μιας και δεν υπάρχει μία μόνο συγκεκριμένη διαρρύθμιση των διαφόρων συστατικών μερών. Έχουμε τρία βα-

σικά είδη διαρρύθμισης: μηχανές τοπικής μνήμης, μηχανές αρθρωτής μνήμης και μηχανές παράλληλης τυχαίας προσπέλασης (parallel random-access machines, PRAM). Στις μηχανές τοπικής μνήμης έχουμε n επεξεργαστές με τη δική τους τοπική μνήμη ο καθένας και όλοι μαζί είναι συνδεδεμένοι σε ένα κοινό δίκτυο διασύνδεσης. Στις μηχανές αρθρωτής μνήμης έχουμε n επεξεργαστές και m ξεχωριστές μονάδες μνήμης όλα μαζί συνδεδεμένα σε ένα κοινό δίκτυο διασύνδεσης. Στις μηχανές παράλληλης τυχαίας προσπέλασης έχουμε n επεξεργαστές όλους συνδεδεμένους με μία κοινή μνήμη. Η διαφορά μεταξύ των μοντέλων αυτών έγκειται στον τρόπο με τον οποίο προσπελάζεται η μνήμη. [57, 22, 23, 58] Σε οποιαδήποτε ανάλυση παράλληλου αλγορίθμου στην εργασία αυτή χρησιμοποιείται το PRAM μοντέλο υπολογισμού, όπου κάθε επεξεργαστής μπορεί παράλληλα με οποιονδήποτε άλλο να προσπελάσει τη μνήμη σε μία μόνο μονάδα χρόνου. Πιο συγκεκριμένα, χρησιμοποιείται το EREW (exclusive-read, exclusive-write) PRAM μοντέλο, όπου δεν επιτρέπεται η παράλληλη εγγραφή ή ανάγνωση στην ίδια διεύθυνση μνήμης από δύο ή περισσότερους επεξεργαστές.

Όσον αφορά την πολυπλοκότητα των παράλληλων αλγορίθμων, έχουμε την κλάση NC [35, 52] για τα προβλήματα που μπορούν να επιλυθούν αποδοτικά σε έναν παράλληλο υπολογιστή κατά αντιστοιχία με την κλάση P για το σειριακό μοντέλο. Ως κλάση NC ορίζεται το σύνολο των προβλημάτων απόφασης τα οποία λύνονται σε πολυλογαριθμικό χρόνο σε έναν παράλληλο υπολογιστή ο οποίος έχει στη διάθεσή του πολυωνυμικό αριθμό από επεξεργαστές. Τυπικά:

Ορισμός 1. Ένα πρόβλημα ανήκει στην κλάση NC αν υπάρχουν σταθερές c και k τέτοιες ώστε το πρόβλημα αυτό να λύνεται σε χρόνο $O(\log^c n)$ χρησιμοποιώντας $O(n^k)$ παράλληλους επεξεργαστές.

Η κλάση NC αποτελεί υποσύνολο της κλάσης P, καθώς όλοι οι πολυλογαριθμικοί παράλληλοι υπολογισμοί μπορούν να εξομοιωθούν από αντίστοιχους σειριακούς σε πολυωνυμικό χρόνο. Ωστόσο, παραμένει μέχρι και σήμερα ανοιχτό πρόβλημα το αν οι δύο αυτές κλάσεις ταυτίζονται $NC = P$ ή όχι. Ο παραπάνω ορισμός δεν επηρεάζεται από το πως το μοντέλο $PRAM$ διαχειρίζεται την παράλληλη εγγραφή/ανάγνωση μιας συγκεκριμένης θέσης μνήμης από δύο ή περισσότερους επεξεργαστές. Τέλος, πήρε το όνομά της (NC='Nick's Class') από τον Nick Pippenger, ο οποίος είχε κάνει εκτεταμένη έρευνα πάνω σε κυκλώματα πολυλογαριθμικού βάθους και πολυωνυμικού μεγέθους.

Ένα από τα σημαντικότερα προβλήματα που μπορούν να λυθούν σε παράλληλες μηχανές κερδίζοντας σημαντική βελτίωση στο χρόνο εκτέλεσης είναι ο πολλαπλασιασμός πινάκων. Η ιδέα πίσω από την παραλληλοποίηση του προβλήματος αυτού έγκειται στην τακτική 'διαίρει και βασίλευε', σπάζοντας τους

αρχικούς πίνακες σε μικρότερους υποπίνακες και εκτελώντας πολλαπλασιασμούς ανάμεσα σε αυτούς τους μικρότερους πλέον πίνακες. Επειδή οι επιμέρους αυτοί πολλαπλασιασμοί είναι ανεξάρτητοι μεταξύ τους μπορούν να εκτελεστούν ταυτόχρονα σε παράλληλους επεξεργαστές και μιας και το μέγεθός τους είναι μικρότερο των αρχικών θα έχουμε το επιθυμητό αποτέλεσματα πολύ πιο γρήγορα. Συγκεκριμένα, ο απλός σειριακός αλγόριθμος πολλαπλασιαμού δύο πινάκων $n \times n$ χρειάζεται χρόνο $O(n^3)$ ενώ ο αλγόριθμος του Strassen [59] τρέχει σε χρόνο $O(n^{2.81})$ και μπορεί να παραλληλοποιηθεί αρκετά εύκολα καθώς στηρίζεται στην παραπάνω λογική.

Στη διπλωματική εργασία αυτή θα συναντήσουμε άλλους δύο πολύ σημαντικούς παράλληλους αλγόριθμους. Αυτόν του Alon για 2-χρωματισμό υπεργράφου, ο οποίος τρέχει σε χρόνο $O(\log n)$ χρησιμοποιώντας $n^{O(1)}$ παράλληλους επεξεραστές, και αυτόν του Luby για την εύρεση ενός Maximal Independent Set σε γράφο, ο οποίος τρέχει σε χρόνο $O(\log^2 n)$ χρησιμοποιώντας $O(m)$ παράλληλους επεξεργαστές.

2.3 Προσεγγιστικοί Αλγόριθμοι

Τα περισσότερα προβλήματα βελτιστοποίησης ανήκουν στην κατηγορία των NP-δύσκολων προβλημάτων. Αυτό σημαίνει πως, δεχόμενοι την ευρέως αποδεκτή εικασία ότι P $\neq$ NP, δεν υπάρχει αλγόριθμος πολυωνυμικού χρόνου που να υπολογίζει τη βέλτιστη λύση τους και μάλιστα η αναζήτησή της και μόνο είναι απαγορευτικά χρονοβόρα. Προβλήματα βελτιστοποίησης είναι τα προβλήματα αυτά όπου αναζητούμε την καλύτερη δυνατή λύση ανάμεσα σε όλες τις αποδεκτές. Συνήθως, τα προβλήματα αυτά έχουν ένα αντίστοιχο πρόβλημα απόφασης από το οποίο προκύπτει άμεσα το πρόβλημα της βελτιστοποίησης. Η έννοια της 'καλύτερης' λύσης ορίζεται με βάση μια *αντικειμενική συνάρτηση* (objective function) η οποία δέχεται ως όρισμα μια οποιαδήποτε αποδεκτή λύση για την εκάστοτε είσοδο και της αποδίδει μια τιμή. Ανάλογα αν έχουμε πρόβλημα μεγιστοποίησης ή ελαχιστοποίησης, καλύτερη είναι μια λύση που έχει υψηλότερη ή χαμηλότερη αντίστοιχα τιμή αντικειμενικής συνάρτησης από μια άλλη. Βέλτιστη είναι η λύση ή λύσεις που έχουν την υψηλότερη ή χαμηλότερη αντίστοιχη τιμή αντικειμενικής συνάρτησης. Παράδειγμα προβλήματος απόφασης είναι το πρόβλημα Vertex-Cover όπου μας δίνεται ένας γράφος και ένας ακέραιος k και πρέπει να αποφανθούμε αν υπάρχει vertex cover[1] μεγέθους $< k$. Το αντίστοιχο

[1]Vertex cover ενός γράφου ονομάζεται ένα υποσύνολο κόμβων του έτσι ώστε κάθε ακμή του γράφου να έχει άκρη στο σύνολο αυτό.

πρόβλημα βελτιστοποίησης είναι η εύρεση του μικρότερου vertex cover (Min-Vertex-Cover) έχοντας ως αντικειμενική συνάρτηση τον αριθμό των κόμβων στο vertex cover. Σημαντικά αποτελέσματα στο συγκεκριμένο πρόβλημα έχουν δώσει οι Parnas και Ron [54], τα οποία θα μας φανούν χρήσιμα και στο σχεδιασμό των αλγορίθμων τοπικού υπολογισμού παρακάτω, καθώς και οι Onak, Ron, Rosen και Rubinfeld [31], οι οποίοι δώσαν έναν αλγόριθμο υπογραμμικού χρόνου που παρέχει λύσεις αρκετά κοντά στη βέλτιστη.

Ακριβώς σε αυτό το πρόβλημα εύρεσης της βέλτιστης λύσης σε προβλήματα βελτιστοποίησης έρχονται να βοηθήσουν οι *προσεγγιστικοί αλγόριθμοι* (approximation algorithms) [63, 61]. Συγκεκριμένα, χαλαρώνοντας την απαίτηση της εύρεσης της ακριβούς βέλτιστης λύσης, μπορούμε να υπολογίσουμε μια λύση η τιμή της αντικειμενικής συνάρτησης της οποίας να είναι πολύ κοντά σε αυτή της βέλτιστης λύσης ακόμα και σε υπογραμμικό χρόνο πολλές φορές. Αυτό ακριβώς κάνουν και οι προσεγγιστικοί αλγόριθμοι. Διαβάζουν μέρος της εισόδου και υπολογίζουν μια προσέγγιση της βέλτιστης λύσης μαζί με κάποιες εγγυήσεις για την απόσταση αυτής της προσεγγιστικής λύσης από τη βέλτιστη. Πολλές φορές, βέβαια, μπορεί να θέλουμε να χρησιμοποιήσουμε προσεγγιστικούς αλγορίθμους και για άλλα προβλήματα για τα οποία υπάρχουν ήδη πολυωνυμικοί αλγόριθμοι. Για παράδειγμα, μπορεί να θέλουν να πάρουμε πιο γρήγορα το αποτέλεσμα, να έχουμε κάποιον περιορισμό στη διαθέσιμη μνήμη ή ο προσεγγιστικός αλγόριθμος να κλιμακώνεται πιο εύκολα σε κάποιο κατανεμημένο σύστημα σε σχέση με τον αντίστοιχο πολυωνυμικό.

Θα ορίσουμε τώρα τι σημαίνει ο όρος α-προσεγγιστικός αλγόριθμος, τον οποίο θα χρησιμοποιήσουμε αρκετές φορές και στο υπόλοιπο της εργασίας.

Ορισμός 2. Έστω πρόβλημα βελτιστοποίησης Π και $F(S)$ η αντικειμενική συνάρτηση με όρισμα μια αποδεκτή λύση S. Συμβολίζουμε με $OPT(x)$ τη βέλτιστη λύση για είσοδο x. Τότε *α-προσεγγιστικός αλγόριθμος* για το πρόβλημα Π ονομάζεται ένας αλγόριθμος ALG που για οποιαδήποτε είσοδο x ισχύει:

$$F(OPT(x)) \leq F(ALG(x)) \leq \alpha \cdot F(OPT(x)).$$

Το α αποκαλείται *λόγος προσέγγισης* του αλγορίθμου. Για προβλήματα ελαχιστοποίησης έχουμε $\alpha > 1$ ενώ για προβλήματα μεγιστοποίησης $\alpha < 1$. Έτσι, ένας $\frac{1}{2}$-προσεγγιστικός αλγόριθμος για ένα πρόβλημα μεγιστοποίησης δίνει πάντα λύση η οποία έχει τιμή αντικειμενικής συνάρτησης τουλάχιστον μισή από αυτή της βέλτιστης λύσης.

Η επόμενη ερώτηση που έρχεται τώρα στο μυαλό είναι για ποια προβλήματα μπορούμε να πάρουμε καλούς προσεγγιστικούς αλγορίθμους και για ποια όχι.

Για το σκοπό αυτό ορίζουμε τις λεγόμενες κλάσεις προσεγγισιμότητας, τις κυριότερες από τις οποίες αναφέρουμε στη συνέχεια:

Ορισμός 3. Λέμε ότι ένα πρόβλημα βελτιστοποίησης ανήκει στην κλάση *Polynomial-time Approximation Scheme* (PTAS) αν για κάθε $\varepsilon > 0$ υπάρχει αλγόριθμος που τρέχει σε πολυωνυμικό χρόνο και δίνει μια λύση η οποία βρίσκεται κατά παράγοντα $(1 + \varepsilon)$ κοντά στη βέλτιστη για προβλήματα ελαχιστοποίησης και κατά παράγοντα $(1 - \varepsilon)$ κοντά για προβλήματα μεγιστοποίησης.

Υποσύνολο της παραπάνω κλάσης αποτελεί η πιο αυστηρή *Fully Polynomial-time Approximation Scheme* (FPTAS) κλάση, η οποία απαιτεί ο αλγόριθμος να είναι πολυωνυμικός τόσο ως προς το μέγεθος της εισόδου n όσο και ως προς τον όρο $1/\varepsilon$. Ένα γνωστό πρόβλημα που ανήκει στην κλάση FPTAS είναι το πρόβλημα του σακιδίου (knapsack problem).

Ορισμός 4. Λέμε ότι ένα πρόβλημα βελτιστοποίησης ανήκει στην κλάση *APX* (εκ του 'approximable') αν υπάρχει αλγόριθμος που τρέχει σε πολυωνυμικό χρόνο και δίνει μια λύση η οποία βρίσκεται κατά κάποιον σταθερό παράγοντα κοντά στη βέλτιστη.

Προφανώς η κλάση APX περιλαμβάνει εξ ορισμού την κλάση PTAS. Η διαφορά έγγειται στο ότι για να ανήκει ένα πρόβλημα στην κλάση PTAS θα πρέπει να υπάρχει ένας αλγόριθμος που να δίνει λύση για κάθε παράγοντα κοντά στη βέλτιστη λύση, ενώ στην κλάση APX αρκεί να υπάρχει αλγόριθμος για ένα μόνο συγκεκριμένο παράγοντα. Μάλιστα, εκτός και αν αποδειχθεί ότι $P = NP$, έχουμε ότι PTAS $\neq$ APX, δηλαδή η κλάση PTAS είναι αυστηρό υποσύνολο της APX. Παράδειγμα προβλήματος που ανήκει στη δεύτερη αλλά όχι στην πρώτη είναι το πρόβλημα bin packing.

Ένα από τα σημαντικότερα αποτελέσματα στην περιοχή είναι το αποτέλεσμα των Feige, Goldwasser, Lovasz, Safra και Szegedy [60] σχετικά με τη μη προσεγγισιμότητα του προβλήματος της εύρεσης ενός Ανεξάρτητου Συνόλου (Independent Set) σε έναν δοσμένο γράφο. Το συγκεκριμένο αποτέλεσμα άνοιξε το δρόμο για περαιτέρω έρευνα γύρω από τη μη προσεγγισιμότητα έκτοτε. Συγκεκριμένα για το πρόβλημα MAX-3SAT ο J. Hastad [28] έδειξε ότι το να πλησιάσεις κατά παράγοντα $7/8 + \varepsilon$ κοντά στη βέλτιστη λύση αποτελεί NP-hard πρόβλημα. Έχουμε ωστόσο και αποτελέσματα προς την αντίθετη πλευρά. Για παράδειγμα ο D. Johnson [30] έδειξε βέλτιστους λόγους προσέγγισης για αρκετά σημαντικά προβλήματα όπως της εύρεσης Independent Set ή Set Cover σε δοσμένο γράφο, το πρόβλημα MAX-SAT κ.ά. δοθέντως πάντα ότι $P \neq NP$.

2.4 Άμεσοι Αλγόριθμοι

Άμεσοι Αλγόριθμοι (Online Algorithms) [34, 16] είναι οι αλγόριθμοι που δε δέχονται την είσοδο στο σύνολό της αλλά τμηματικά, σαν μια ακολουθία από αιτήματα, και εκτελούν μια συγκεκριμένη ενέργεια σαν απόκριση σε κάθε αίτημα. Μάλιστα, τα αιτήματα αυτά που δέχεται ένας άμεσος αλγόριθμος πρέπει να εξυπηρετηθούν με τη σειρά με την οποία κατέφθασαν. Έστω, λοιπόν, $r = r(1) \dots r(n)$ η ακολουθία των αιτημάτων. Χαρακτηριστικό των αλγορίθμων αυτών είναι ότι τα μελλοντικά αιτήματα δεν είναι γνωστά και έτσι πρέπει να δράσουν όσο καλύτερα μπορούν με βάση το μέρος της εισόδου που έχουν δει μέχρι την παρούσα στιγμή χωρίς να ξέρουν τι επίκειται στο μέλλον. Δηλαδή, όταν εξυπηρετείται το αίτημα $r(t)$ ο αλγόριθμος δε γνωρίζει τα αιτήματα $r(t')$ με $t' > t$. Επίσης, κάθε φορά που ο αλγόριθμος εξυπηρετεί ένα αίτημα προκαλείται κάποιο κόστος. Προφανώς, στόχος στο σχεδιασμό άμεσων αλγορίθμων είναι να ελαχιστοποιηθεί το κόστος που θα προκληθεί συνολικά μετά την εξυπηρέτηση του συνόλου των αιτημάτων της ακολουθίας.

Οι άμεσοι αλγόριθμοι συγκρίνονται συχνά με τους *offline αλγόριθμους*. Οι offline αλγόριθμοι έχουν στη διάθεσή τους ολόκληρη την είσοδο εξαρχής. Επομένως, όταν τους ζητείται να εξυπηρετήσουν ένα αίτημα μπορούν να δράσουν βασισμένοι σε ολόκληρη την ακολουθία αιτημάτων που τους είναι γνωστή. Εξαιτίας αυτού, προφανώς, οι offline αλγόριθμοι αποδίδουν συχνά αρκετά καλύτερα από τους αντίστοιχούς τους άμεσους αλγορίθμους για ένα συγκεκριμένο πρόβλημα. Συγκρίνοντας, έτσι, έναν άμεσο αλγόριθμο με τον αντίστοιχό του offline αλγόριθμο μπορούμε να τον αξιολογήσουμε και να πάρουμε ένα μέτρο της απόδοσής τους στη χειρότερη περίπτωση. Η ανάλυση αυτή ονομάζεται *ανταγωνιστική ανάλυση* (competitive analysis). Συγκεκριμένα έχουμε:

Ορισμός 5. Έστω ALG ένας άμεσος αλγόριθμος που δέχεται μια ακολουθία αιτημάτων r και OPT ο βέλτιστος offline αλγόριθμος που εξυπηρετεί την ακολουθία αυτή με το ελάχιστο δυνατό κόστος. Συμβολίζουμε με $ALG(r)$ και $OPT(r)$ τα κόστη των δύο αλγορίθμων αφού εξυπηρετήσουν το σύνολο των αιτημάτων της ακολουθίας r. Τότε ο αλγόριθμος ALG ονομάζεται *k-ανταγωνιστικός* αν υπάρχει σταθερά c τέτοια ώστε για οποιαδήποτε ακολουθία r να ισχύει:

$$ALG(r) \leq k \cdot OPT(r) + c.$$

Το k ονομάζεται *δείκτης ανταγωνισιμότητας* (competitive ratio). Βέλτιστος άμεσος αλγόριθμος για ένα πρόβλημα είναι τότε αυτός με τον μικρότερο δείκτη ανταγωνισιμότητας.

Σε πολλά προβλήματα οι άμεσοι αλγόριθμοι αποδίδουν καλύτερα αν τους επιτραπεί να κάνουν κάποιες τυχαίες επιλογές κατά την εκτέλεσή τους. Στην περίπτωση που επιτρέπεται κάτι τέτοιο, ο δείκτης ανταγωνισιμότητας του αλγορίθμου αυτού ορίζεται σε σχέση με κάποιον αντίπαλο. Ο αντίπαλος κατασκευάζει μια ακολουθία αιτημάτων r και την παρέχει στον αλγόριθμο. Γνωρίζει πάντα κατά το σχεδιασμό της ακολουθίας αυτής την περιγραφή του αλγορίθμου. Υπάρχουν, ωστόσο, τρία διαφορετικά είδη αντιπάλων ανάλογα με το αν ο αντίπαλος γνωρίζει τις απαντήσεις και τις τυχαίες επιλογές του αλγορίθμου στα προηγούμενα αιτήματα προτού σχεδιάσει τα επόμενα που θα του παρέχει. Έτσι, σύμφωνα με τον Ben-David [14], έχουμε:

Oblivious αντίπαλος: Ένας oblivious αντίπαλος κατασκευάζει ολόκληρη την ακολουθία αιτημάτων r προτού παρέχει στον αλγόριθμο οποιοδήποτε αίτημα από αυτά.

Online προσαρμοστικός αντίπαλος: Ένας αντίπαλος αυτού του τύπου μπορεί να παρακολουθεί τις απαντήσεις του αλγορίθμου στα αιτήματα που του δίνει και να σχεδιάζει τα επόμενα αιτήματα με βάση τις απαντήσεις αυτές. Πρέπει, ωστόσο, να δίνει τα αιτήματα αυτά κατά *online* τρόπο, μη γνωρίζοντας δηλαδή τις τυχαίες επιλογές που έκανε ή θα κάνει ο αλγόριθμος.

Offline προσαρμοστικός αντίπαλος: Ένας αντίπαλος αυτού του τύπου μπορεί να παρακολουθεί τις απαντήσεις του αλγορίθμου στα αιτήματα που του δίνει και να σχεδιάζει τα επόμενα αιτήματα με βάση τις απαντήσεις αυτές. Παρέχει ακόμη τα αιτήματα αυτά κατά *offline* τρόπο, γνωρίζοντας δηλαδή τις τυχαίες επιλογές που κάνει ο αλγόριθμος για να απαντήσει στα ερωτήματα.

2.5 Σχεδιασμός Μηχανισμών

Θα ορίσουμε αρχικά το μοντέλο δημοπρασιών που χρησιμοποιούμε. Έστω ένας πωλητής που θέλει να πουλήσει ένα αντικείμενο και n ενδιαφερόμενοι αγοραστές. Κάθε αγοραστής i έχει μια αξία v_i για το αντικείμενο αυτό, δηλαδή το μέγιστο ποσό που προτίθεται να πληρώσει για να αποκτήσει το αντικείμενο. Η αξία αυτή κάθε αγοραστή θεωρούμε ότι είναι ιδιωτική του πληροφορία και κανείς άλλος δεν τη γνωρίζει. Προφανώς κάθε αγοραστής συμμετέχει στη δημοπρασία για να αποκτήσει το αντικείμενο όσο το δυνατόν φθηνότερα, ενώ αν η τιμή του υπερβεί το v_i τότε δεν το θέλει καθόλου. Το κέρδος (utility) u_i ενός αγοραστή i θεωρούμε ότι είναι ίσο με 0 όταν δεν αποκτάει το αντικείμενο και ίσο με $v_i - p$ όταν το αποκτάει και πληρώνει p. Το μοντέλο κέρδους αυτό ονομάζεται στη βιβλιογραφία *quasi-linear μοντέλο*.

Το μοντέλο δημοπρασιών που χρησιμοποιούμε ανήκει στην κατηγορία των *sealed-bid δημοπρασιών*. Στις δημοπρασίες αυτές κάθε αγοραστής i κάνει μια προσφορά για το αντικείμενο b_i την οποία την αναφέρει μόνο στον πωλητή και η οποία μπορεί να είναι και διαφορετική από την αξία του για το αντικείμενο αυτό. Έχουμε έτσι το διάνυσμα προσφορών $\mathbf{b} = (b_1, \ldots, b_n)$. Ο πωλητής στη συνέχεια αποφασίζει σε ποιον θα δώσει το αντικείμενο, δηλαδή τον *κανόνα κατανομής* $\mathbf{x}(\mathbf{b}) \subseteq \mathcal{R}^n$. Κάθε x_i είναι ίσο με 0 ή 1 ανάλογα αν ο αγοραστής i κέρδισε το αντικείμενο ή όχι και έχουμε συνολικά ένα μόνο 1. Ο πωλητής αποφασίζει επίσης και την τιμή στην οποία θα πωληθεί το αντικείμενο, δηλαδή τον *κανόνα πληρωμών* $\mathbf{p}(\mathbf{b}) \subseteq \mathcal{R}^n$. Όλα τα p_i είναι ίσα με 0 εκτός του αγοραστή εκείνου που κέρδισε εν τέλει το αντικείμενο. Οι πληρωμές είναι πάντα θετικές ώστε ο πωλητής να παίρνει χρήματα και όχι να δίνει. Αν είχαμε περισσότερα του ενός ίδια αντικείμενα τα $\mathbf{x}$ και $\mathbf{p}$ θα προσαρμόζονταν αναλόγως. Το κέρδος κάθε αγοραστή i είναι τότε $u_i(\mathbf{b}) = v_i \cdot \mathbf{x}_i(\mathbf{b}) - p_i(\mathbf{b})$

Υπάρχουν κάποιες ιδιότητες που επιδιώκουμε να ισχύουν σε κάθε δημοπρασία που σχεδιάζουμε, επειδή βοηθούν τόσο τους αγοραστές να συμμετέχουν όσο και τον ίδιο τον πωλητή να προβλέψει το αποτέλεσμα. Οι δύο πιο σημαντικές είναι οι ιδιότητες της *φιλαλήθειας* και της *individual rationality*:

Ορισμός 6. *Φιλαλήθεια.* Ένας μηχανισμός ονομάζεται φιλαλήθης όταν η κυρίαρχη στρατηγική κάθε αγοραστή i είναι να δηλώσει ως προσφορά b_i την πραγματική ιδιωτική του αξία v_i. Πιο συγκεκριμένα, για οποιαδήποτε προσφορά b'_i του αγοραστή i και για οποιεσδήποτε προσφορές $\mathbf{b}_{-i}$ των υπολοίπων αγοραστών θα ισχύει:

$$u_i(v_i, \mathbf{b}_{-i}) \geq u_i(b'_i, \mathbf{b}_{-i}).$$

Δηλαδή, δηλώνοντας την πραγματική του αξία μεγιστοποιείται το κέρδος του, οτιδήποτε και αν κάνουν οι υπόλοιποι αγοραστές.

Ορισμός 7. *Individual Rationality.* Η ιδιότητα αυτή εγγυάται ότι οποιοσδήποτε αγοραστής i δηλώσει την πραγματική του αξία v_i ως προσφορά b_i πρόκειται να έχει μη αρνητικό κέρδος u_i. Πιο συγκεκριμένα, για οποιεσδήποτε προσφορές $\mathbf{b}_{-i}$ των υπολοίπων αγοραστών θα ισχύει:

$$u_i(v_i, \mathbf{b}_{-i}) \geq 0.$$

Δηλαδή, δηλώνοντας την πραγματική του αξία δεν πρόκειται να βγει ποτέ χαμένος συμμετέχοντας στο μηχανισμό, οτιδήποτε και αν κάνουν οι υπόλοιποι αγοραστές. Η ιδιότητα αυτή πολλές φορές αναφέρεται στη βιβλιογραφία και ως voluntary participation (εθελοντική συμμετοχή).

Από την πλευρά του πωλητή, η ιδιότητα της φιλαλήθειας τον βοηθάει να επιχειρηματολογήσει σχετικά με την έκβαση της δημοπρασίας. Η μόνη υπόθεση που πρέπει να γίνει για το σκοπό αυτό είναι ότι ένας αγοραστής που διαθέτει μια κυρίαρχη στρατηγική θα τη χρησιμοποιήσει, μιας και χρησιμοποιώντας οποιαδήποτε άλλη στρατηγική μπορεί να βγει ζημιωμένος. Σε αυτή την υπόθεση στην ουσία στηρίζεται και ολόκληρη η Θεωρία Παιγνίων. Από την πλευρά του αγοραστή, η ιδιότητα αυτή του λύνει τα χέρια ως προς τον τρόπο με τον οποίο θα συμμετάσχει καθώς δε χρειάζεται να εξετάσει όλες τις δυνατές στρατηγικές αλλά αρκεί να δηλώσει ως προσφορά την πραγματική του αξία. Η ιδιότητα της individual rationality του εγγυάται ακόμη ότι δηλώνοντας την πραγματική του αξία δεν πρόκειται να βγει ζημιωμένος συμμετέχοντας στο μηχανισμό, οπότε δεν υπάρχει και λόγος να μη συμμετέχει.

Οι κυριότεροι στόχοι του σχεδιασμού δημοπρασιών συνήθως είναι η μεγιστοποίηση του κέρδους του πωλητή ή η μεγιστοποίηση του *κοινωνικού καλού* (social welfare/surplus). Το κοινωνικό καλό αντιπροσωπεύει κατά κάποιο τρόπο τη 'χαρά' της κοινωνίας για μια συγκεκριμένη κατανομή των αντικειμένων. Προσθέτοντας τα κέρδη όλων των συμμετεχόντων στο μηχανισμό, δηλαδή και του πωλητή και των αγοραστών, έχουμε το εξής άθροισμα:

$$\underbrace{\sum_{i=1}^{n} p_i}_{\text{πωλητής}} + \underbrace{\sum_{i=1}^{n} (v_i x_i - p_i)}_{\text{αγοραστές}} = \sum_{i=1}^{n} v_i x_i.$$

Στο παραπάνω άθροισμα θεωρήσαμε και τον πωλητή ως παίκτη του μηχανισμού με κέρδος τις πληρωμές που παίρνει για τα αντικείμενα που πουλάει. Βλέπουμε ότι οι πληρωμές ακυρώνονται μεταξύ τους και ως κοινωνικό καλό μένει μόνο το άθροισμα των αξιών των αγοραστών που κέρδισαν κάποιο αντικείμενο. Προκειμένου, επομένως, να μεγιστοποιηθεί το κοινωνικό καλό θα πρέπει τα αντικείμενα να δοθούν στους αγοραστές με τις μεγαλύτερες αξίες.

Ένας πολύ απλός μηχανισμός δημοπρασίας για την περίπτωση ενός αντικειμένου με πολλούς αγοραστές είναι η λεγόμενη *second-price δημοπρασία* ή *δημοπρασία Vickrey*. Στη δημοπρασία αυτή, που είναι δημοπρασία τύπου sealed-bid, το αντικείμενο δίνεται στον πλειοδότη (δηλαδή στον αγοραστή που έκανε τη μεγαλύτερη προσφορά). Η πληρωμή που ανατίθεται στον αγοραστή αυτόν να δώσει είναι ίση με τη δεύτερη μεγαλύτερη προσφορά αγοραστή. Η δημοπρασία αυτή αποδεικνύεται αρκετά εύκολα [62] ότι είναι φιλαλήθης και έχει την ιδιότητα της individual rationality. Επίσης, μεγιστοποιεί προφανώς και το κοινωνικό καλό δεδομένου ότι δίνει το αντικείμενο στον παίκτη με τη μεγαλύτερη προσφο-

ρά και, εφόσον ο μηχανισμός είναι φιλαλήθης, στον παίκτη με τη μεγαλύτερη αξία.

Ευτυχώς για όλους τους σχεδιαστές μηχανισμών, υπάρχει ένα θεώρημα το οποίο δεδομένου ενός κανόνα κατανομής των αντικειμένων, δίνει αρκετά εύκολα και με συγκεκριμένο τρόπο τον αντίστοιχο κανόνα πληρωμών ώστε ο μηχανισμός να είναι φιλαλήθης και να παρέχει το μέγιστο κέρδος. Αρκεί, λοιπόν, ο σχεδιαστής ενός μηχανισμού να καταλήξει στον κανόνα κατανομής που θέλει ανάλογα με το σκοπό του και χρησιμοποιώντας το θεώρημα αυτό θα πάρει στη συνέχεια άμεσα και αποδοτικά τον κανόνα πληρωμών. Δίνουμε πρώτα έναν απαραίτητο ορισμό σχετικά με τον κανόνα κατανομής:

Ορισμός 8. Ένας κανόνας κατανομής $\mathbf{x}$ ονομάζεται *μονότονος* αν για κάθε αγοραστή i και για οποιεσδήποτε προσφορές $\mathbf{b}_{-i}$ των υπολοίπων αγοραστών, η συνάρτηση κατανομής $x_i(b_i, \mathbf{b}_{-i})$ είναι αύξουσα ως προς το b_i.

Με άλλα λόγια, αυξάνοντας την προσφορά του ένας οποιοσδήποτε αγοραστής μπορεί μόνο να κερδίσει περισσότερα αντικείμενα και όχι να χάσει κάποιο από αυτά που ήδη έχει (αν έχει). Τότε, έχουμε το εξής πολύ σημαντικό θεώρημα στο Σχεδιασμό Μηχανισμών το οποίο ονομάζεται *Λήμμα του Myerson*:

Θεώρημα 1. *([49]). Αν ένας κανόνας κατανομής $\mathbf{x}$ είναι μονότονος, τότε υπάρχει μοναδικός κανόνας πληρωμών $\mathbf{p}$ έτσι ώστε ο μηχανισμός $(\mathbf{x}, \mathbf{p})$ να είναι φιλαλήθης, και μάλιστα ο κανόνας αυτός $\mathbf{p}$ δίνεται από συγκεκριμένο τύπο.*

Το παραπάνω αποτέλεσμα ισχύει, ωστόσο, μόνο στην περίπτωση που πωλείται ένα μόνο αντικείμενο ή πολλά ίδια και κάθε αγοραστής έχει μία αξία για το αντικείμενο αυτό. Αν θελήσουμε να σχεδιάσουμε μια δημοπρασία για περισσότερα του ενός αντικείμενα, τότε θα πρέπει να μεταβούμε σε ένα γενικότερο μοντέλο μηχανισμών. Συγκεκριμένα, έστω ότι έχουμε n αγοραστές και ένα σύνολο Ω από πιθανά αποτελέσματα όσον αφορά την κατανομή των αντικειμένων. Κάθε αγοραστής τώρα έχει ως ιδιωτική πληροφορία μια αξία $v_i(\omega)$ για κάθε ένα ενδεχόμενο $\omega \in \Omega$. Στο μοντέλο αυτό έχουμε το εξής θεμελιώδες θεώρημα στο Σχεδιασμό Μηχανισμών:

Θεώρημα 2. *([26, 19, 62]). Σε κάθε γενικό μοντέλο σχεδιασμού μηχανισμών υπάρχει ένας φιλαλήθης μηχανισμός ο οποίος μεγιστοποιεί το κοινωνικό καλό.*

Ο παραπάνω μηχανισμός ονομάζεται VCG από τα αρχικά των ονομάτων των εμπνευστών του (Vickrey-Clarke-Groves). Δεδομένου ότι μεγιστοποιεί το

κοινωνικό καλό ο κανόνας κατανομής θα πρέπει να είναι ο εξής:

$$\mathbf{x}(\mathbf{b}) = \text{argmax}_{\omega\in\Omega} \sum_{i=1}^{n} b_i(\omega).$$

Έτσι επιλέγεται το αποτέλεσμα που μεγιστοποιεί το άθροισμα των προσφορών και, δοθέντος ότι ο μηχανισμός είναι φιλαλήθης, μεγιστοποιεί και το άθροισμα των αξιών (δηλαδή το κοινωνικό καλό). Ως κανόνας πληρωμών ορίζεται ο εξής: ένας αγοραστής i πληρώνει την απώλεια κέρδους που προκαλεί στους υπόλοιπους $n-1$ αγοραστές από την παρουσία του. Δηλαδή:

$$p_i(\mathbf{b}) = \underbrace{\max_{\omega\in\Omega} \sum_{j\neq i} b_j(\omega)}_{\text{χωρίς τον i}} - \underbrace{\sum_{j\neq i} b_j(\omega^*)}_{\text{με τον i}},$$

όπου $\omega^* = \mathbf{x}(\mathbf{b})$ το αποτέλεσμα που επιλέχθηκε με τον παραπάνω κανόνα κατανομής.

Επειδή στην πράξη δεν μπορούμε να γνωρίζουμε τις πραγματικές αξίες των αγοραστών κατά τη φάση σχεδιασμού ενός μηχανισμού, ώστε να σχεδιάσουμε τον πιο κατάλληλο για την περίσταση, θεωρούμε ότι οι αξίες των αγοραστών ακολουθούν κάποιες συγκεκριμένες κατανομές τις οποίες γνωρίζουμε. Οι κατανομές αυτές μπορεί να έχουν προκύψει από στοιχεία προηγούμενων εκτελέσεων του μηχανισμού, δημογραφικά στοιχεία των αγοραστών, εταιρίες δημοσκοπήσεων καθώς και από συνδυασμό των προαναφερθέντων και άλλων πηγών. Τότε τα πράγματα είναι αρκετά πιο δύσκολα για το σχεδιασμό του κατάλληλου μηχανισμού. Τα τελευταία χρόνια η έρευνα έχει επικεντρωθεί γύρω από το συγκεκριμένο πρόβλημα εύρεσης ενός αποδοτικού μηχανισμού με το βέλτιστο κέρδος στην περίπτωση πολλών αντικειμένων, αν και το πρόβλημα παραμένει ακόμη ανοικτό.

Έστω, για παράδειγμα, η περίπτωση όπου ένας πωλητής προσφέρει πολλά διαφορετικά αντικείμενα σε έναν μόνο αγοραστή ο οποίος ενδιαφέρεται να αποκτήσει το πολύ ένα από αυτά (unit-demand buyer). Έχουμε τα εξής δύο πολύ σημαντικά αποτελέσματα:

Θεώρημα 3. *([18]). Όταν οι κατανομές των αξιών των αγοραστών για τα αντικείμενα είναι ανεξάρτητες μεταξύ τους η εύρεση ενός βέλτιστου ντετερμινιστικού μηχανισμού που να μεγιστοποιεί το κέρδος είναι υπολογιστικά δύσκολη διαδικασία.*

Θεώρημα 4. *([64]). Ακόμα και όταν οι κατανομές των αξιών των αγοραστών για τα αντικείμενα είναι ίδιες μεταξύ τους ή το μέγεθος του support*[2] *τους είναι το πολύ 3, η εύρεση ενός βέλτιστου ντετερμινιστικού μηχανισμού που να μεγιστοποιεί το κέρδος είναι ένα NP-πλήρες πρόβλημα.*

Ακόμη και αν επιτρέψουμε και πιθανοτικούς μηχανισμούς τα αποτελέσματα δεν αλλάζουν ιδιαίτερα:

Θεώρημα 5. *([17]). Όταν πουλάμε πολλά διαφορετικά αντικείμενα σε πολλούς αγοραστές πιθανότατα*[3] *δεν υπάρχει βέλτιστος μηχανισμός που να μεγιστοποιεί το κέρδος ο οποίος να υλοποιείται αποδοτικά.*

2.6 Probabilistic Inference

Στατιστικός Συμπερασμός είναι η διαδικασία της εξαγωγής συμπερασμάτων για μια κατάσταση με βάση κάποιες παρατηρήσεις που έχουμε αποκτήσει. Για παράδειγμα, έστω ότι σε ένα δωμάτιο υπάρχει ένας αριθμός από ανιχνευτές οι οποίοι συνεργάζονται προκειμένου να διαπιστώσουν την ύπαρξη φωτιάς ή μη, καθένας από τους οποίους έχει μια συγκεκριμένη πιθανότητα επιτυχούς ανίχνευσης φωτιάς. Τότε, η κλασική ερώτηση στατιστικού συμπερασμού είναι ο υπολογισμός της πιθανότητας ύπαρξης φωτιάς με βάση τις τιμές των ανιχνευτών. Υπάρχει, όμως, πιθανότητα κάποιος από τους ανιχνευτές να είναι χαλασμένος και να μη παράγει τα αναμενόμενα αποτελέσματα. Χρειάζεται, λοιπόν, να είμαστε σίγουροι ότι και ένας, για παράδειγμα, από τους ανιχνευτές να χαλάσει η εκτίμηση για φωτιά θα είναι η ίδια. Εισάγεται, έτσι, η ιδέα του *Εύρωστου Στατιστικού Συμπερασμού* (Robust Probabilistic Inference), όπου οι παρατηρήσεις στις οποίες βασιζόμαστε για να βγάλουμε συμπεράσματα μπορεί να είναι κακοβούλως τροποποιημένες.

Έστω, λοιπόν, ότι θέλουμε να προβλέψουμε την τιμή μιας δυαδικής μεταβλητής και έχουμε στη διάθεσή μας έναν αριθμό από παρατηρήσεις. Κάποιες, όμως, από αυτές τις παρατηρήσεις μπορεί να είναι όπως είπαμε και παραπάνω κακοβούλως τροποποιημένες. Έχουμε, έτσι, ένα παίγνιο ανάμεσα στον *προβλέπτη*, που προσπαθεί να προβλέψει σωστά την τιμή της μεταβλητής, και τον *αντίπαλο*, ο οποίος προσπαθεί να κάνει τον προβλέπτη να αποτύχει έχοντας

[2]Support μιας κατανομής ονομάζεται το σύνολο των σημείων για τα οποία η συνάρτηση κατανομής δεν είναι μηδενική.

[3]Εκτός και αν $ZPP \supseteq P^{\#P}$.

στη διάθεσή του ένα σύνολο από δυνατές τροποποιήσεις που μπορεί να εφαρμόσει στις παρατηρήσεις για να τις αλλάξει. Στόχος είναι να προβλέψουμε όσο το δυνατόν καλύτερα την τιμή της μεταβλητής. Το πρόβλημα αυτό μπορεί να περιγραφεί σαν ένα γραμμικό πρόβλημα συγκεκριμένου τύπου. Στην εργασία αυτή παρουσιάζεται ένας αλγόριθμος τοπικού υπολογισμού ο οποίος λύνει προσεγγιστικά τέτοιου είδους γραμμικά προβλήματα — άρα και το πρόβλημα υπολογισμού της τιμής της μεταβλητής από τις τροποποιημένες παρατηρήσεις — σε πολυλογαριθμικό χρόνο.

Κεφάλαιο 3

Ορισμοί

3.1 Συμβολισμοί

Έστω $n \geq 1$ ένας φυσικός αριθμός. Τότε με $[n]$ συμβολίζουμε το σύνολο $\{1, \ldots, n\}$. Όλοι οι λογάριθμοι είναι με βάση το 2.

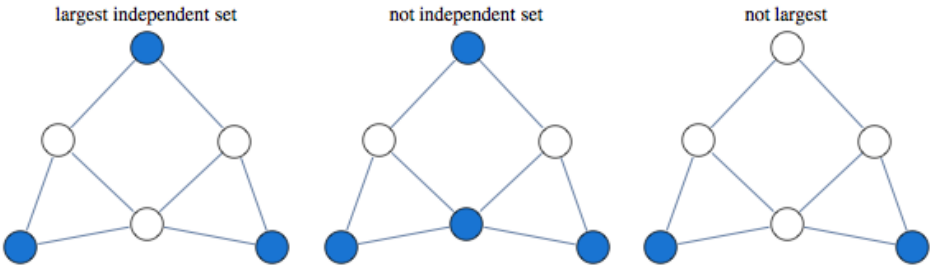
Έστω $G = (V, E)$ ένας γράφος. Όλοι οι γράφοι θεωρούνται μη κατευθυνόμενοι. Τότε, με $d_G(u, v)$ συμβολίζουμε την απόσταση μεταξύ των κόμβων u και v, δηλαδή το μήκος του συντομότερου μονοπατιού από τον έναν κόμβο στον άλλον. Με $N_G(u) = \{u \in V(G) : (u, v) \in E(G)\}$ συμβολίζουμε το σύνολο των γειτόνων του u. Επίσης, $N_G^+(U) = N_G(u) \cup \{u\}$. Με $d_G(u)$ συμβολίζουμε το βαθμό του κόμβου u. Όποτε δεν υπάρχει κίνδυνο σύγχυσης με τους συμβολισμούς παραλείπουμε το δείκτη G από τα $d_G(u, v)$, $N_G(u)$ και $d_G(u)$.

3.2 Maximal Independent Set

Ένα ανεξάρτητο σύνολο (independent set) ενός γράφου G είναι ένα σύνολο από μη γειτονικούς κόμβους του G. Ένα independent set (IS) ονομάζεται maximal independent set (MIS) αν δεν περιλαμβάνεται εξ ολοκλήρου σε κανένα άλλο IS. Το παρακάτω σχήμα βοηθάει στην κατανόηση των ιδεών:

Για την εύρεση ενός MIS υπάρχει άπληστος αλγόριθμος ο οποίος τρέχει σε γραμμικό χρόνο:

1. Αρίθμησε όλους τους κόμβους του G ως $1, 2, \ldots, n$
2. $S \leftarrow \emptyset$



Σχήμα 3.1: Παραδείγματα independent set

3. Από $i = 1$ έως n:

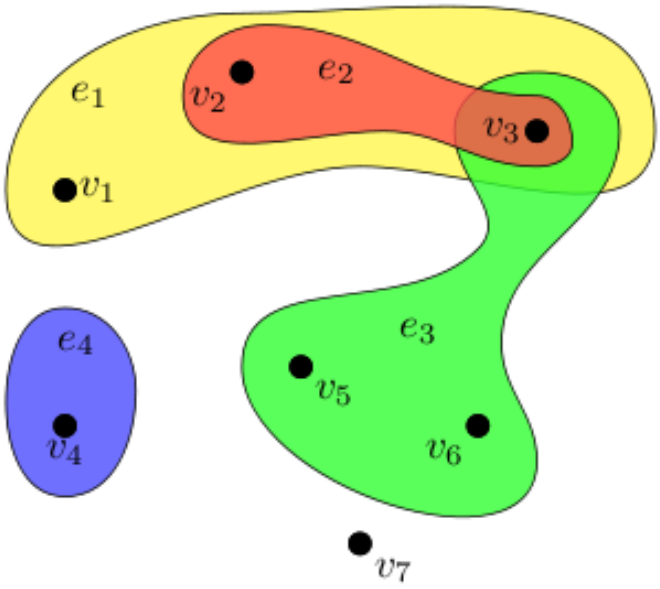
   (α′) Αν ο i δεν έχει γείτονες στο S, $S \leftarrow S \cup \{i\}$

Το MIS που προκύπτει από τον παραπάνω αλγόριθμο ονομάζεται το *λεξικογραφικά πρώτο maximal independent set* (lexicographically first maximal independent set ή LFMIS). Ο Cook [20] απέδειξε ότι η εύρεση του LFMIS είναι πρόβλημα NC^1-πλήρη για το P, ενισχύοντας έτσι τη γενικότερη διαίσθηση που υπήρχε ότι δεν υπάρχει αλγόριθμος NC που να παράγει το LFMIS. Ωστόσο, οι Karp και Wigderson [36] ανέπτυξαν έναν γρήγορο παράλληλο αλγόριθμο ο οποίος χρησιμοποιεί $O(n^2)$ επεξεργαστές και τρέχει σε χρόνο $O((\log n)^4)$. Έδειξαν έτσι ότι το πρόβλημα του MIS ανήκει στην κλάση NC^4. Ο αλγόριθμος του Luby που θα παρουσιάσουμε στην ενότητα αυτή αποδεικνύει ότι το πρόβλημα του MIS ανήκει στην κλάση NC^2.

Το πρόβλημα της εύρεσης ενός maximal independent set είναι ιδιαίτερα σημαντικό μιας και χρησιμοποιείται σε όλο και περισσότερους παράλληλους αλγόριθμους σε μορφή υπορουτίνας. Για παράδειγμα, ο Karloff [32] χρησιμοποίησε τον αλγόριθμο για MIS ως υπορουτίνα για το πρόβλημα του Odd Set Cover. Επιπλέον, πολλά προβλήματα έχει δειχθεί ότι ανήκουν και αυτά στην κλάση NC^2 δίνοντας αναγωγές από αυτά στο πρόβλημα εύρεσης ενός MIS. Συγκεκριμένα, οι Karp και Wigderson [36] έδωσαν NC^1 αναγωγές από τα προβλήματα Maximal Set Packing και Maximal Matching στο MIS και μια NC^2 αναγωγή από το πρόβλημα της 2-Ικανοποιησιμότητας στο MIS. Ο Luby [45] έδωσε, επιπλέον, μια NC^1 αναγωγή από το πρόβλημα Maximal Coloring στο MIS.

3.3 2-Χρωματισμός Υπεργράφου

Υπεργράφος H ονομάζεται ένα ζευγάρι $H = (V, E)$, όπου V είναι ένα σύνολο από αντικείμενα που ονομάζονται *κόμβοι* και E είναι ένα σύνολο από μη-κενά υποσύνολα του V που ονομάζονται *υπερακμές*. Ένα παράδειγμα ενός υπεργράφου φαίνεται στο παρακάτω σχήμα:



Σχήμα 3.2: Παράδειγμα υπεργράφου

Ένας υπεργράφος ονομάζεται *k-ομοιόμορφος* αν κάθε υπερακμή του περιέχει ακριβώς k κόμβους. Ένας *2-χρωματισμός* ενός υπεργράφου H είναι μια αντιστοίχιση c από κόμβους σε δύο χρώματα, πχ $c : V \to \{\text{red}, \text{blue}\}$, έτσι ώστε καμία υπερακμή του H να μην περιέχει μόνο ένα χρώμα. *Γράφος Εξαρτήσεων* G του υπεργράφου H ονομάζεται ένας γράφος ο οποίος έχει έναν κόμβο για κάθε μία υπερακμή του H και μια υπερακμή E_i ενώνεται με μια άλλη υπερακμή E_j αν και μόνο αν $E_i \cap E_j \neq \emptyset$. Με $\mathcal{E}(u)$ συμβολίζουμε το σύνολο των υπερακμών στις οποίες ανήκει ο u.

3.4 k-wise independent τυχαίες μεταβλητές

Έστω $X_1, \ldots, X_n$ τυχαίες μεταβλητές με τιμές στο σύνολο {0,1} οι οποίες ακολουθούν μια κατανομή $D : \{0,1\}^n \to [0,1]$ και $1 \leq k \leq n$ ένας ακέραιος. Η κατανομή D είναι *k-wise independent* αν για οποιαδήποτε $i_1, i_2, \ldots, i_k$ (όλα μοναδικά) και $t_1, \ldots, t_k \in \{0,1\}$ ισχύει

$$Pr_{X_1,\ldots,X_n \sim D}[X_{i_1} = t_1, \ldots, X_{i_k} = t_k] = Pr[X_{i_1} = t_1] \ldots Pr[X_{i_k} = t_k].$$

Μια τυχαία μεταβλητή ονομάζεται k-wise independent αν η συνάρτηση κατανομής της είναι k-wise independent.

Support μιας κατανομής D, $\text{supp}(D)$, ονομάζεται το σύνολο των σημείων για τα οποία η συνάρτηση κατανομής δεν είναι μηδενική, $\text{supp}(D) = \{x : D(x) > 0\}$. Μια διακριτή συνάρτηση κατανομής είναι *συμμετρική* αν $D(x) = 1/|\text{supp}(D)|$ για κάθε $x \in \text{supp}(D)$. Αν μια κατανομή $D : \{0,1\}^n \to [0,1]$ είναι συμμετρική με $|\text{supp}(D)| \leq 2^m$ για κάποιο $m \leq n$, τότε μπορούμε να δεικτοδοτήσουμε τα στοιχεία του support της D με $\{0,1\}^m$, όπου το m καλείται *seed length* της τυχαίας μεταβλητής που ακολουθεί κατανομή D.

Θεώρημα 6. *Για οποιαδήποτε $1 \leq k \leq n$, υπάρχει μια συμμετρική κατανομή $D : \{0,1\}^n \to [0,1]$ με μέγεθος support το πολύ $n^{\lfloor k/2 \rfloor}$, η οποία είναι k-wise independent. Με άλλα λόγια, υπάρχει μια k-wise independent τυχαία μεταβλητή $x = (x_1, \ldots, x_n)$ με seed length το πολύ $O(k \log n)$. Επίσης, για κάθε $1 \leq i \leq n$, το x_i μπορεί να υπολογισθεί σε χώρο $O(k \log n)$.*

3.5 Lovász Local Lemma

Υπάρχουν περιπτώσεις που το ζητούμενο ενός προβλήματος είναι η επαλήθευση της ύπαρξης μιας δομής σε ένα σύνολο στοιχείων. Με άλλα λόγια ζητείται να αποδείξουμε ότι τα στοιχεία του συνόλου παρουσιάζουν κάποιες συγκεκριμένες ιδιότητες. Για να το επιτύχουμε αυτό μπορούμε να εφαρμόσουμε την πιθανοτική μέθοδο. Με βάση αυτή, επιλέγουμε ένα στοιχείο του συνόλου στην τύχη και δείχνουμε ότι αυτό ικανοποιεί όλες τις ζητούμενες ιδιότητες με κάποια θετική πιθανότητα. Συγκεκριμένα, τις περισσότερες φορές, η πιθανότητα αυτή είναι αρκετά μεγάλη και τείνει στο 1 καθώς οι παράμετροι του προβλήματος τείνουν στο άπειρο. Μάλιστα, χρησιμοποιώντας την πιθανοτική μέθοδο, παίρνουμε συνήθως και έναν πιθανοτικό αλγόριθμο ο οποίος παράγει ένα σύνολο με την αντίστοιχη δομή. Υπάρχουν, ωστόσο, και περιπτώσεις όπου η ύπαρξη μιας συγκεκριμένης δομής σε ένα σύνολο αποδεικνύεται κάνοντας χρήση γεγονότων

τα οποία ισχύουν με κάποια θετική πιθανότητα η οποία, όμως, είναι εκθετικά μικρή. Αντιθέτως με πριν, τέτοιες αποδείξεις συνήθως δεν δίνουν αλγορίθμους για τα αντίστοιχα προβλήματα.

Τέτοιες περιπτώσεις είναι τα προβλήματα τα οποία λύνονται με τη βοήθεια του Lovász Local Lemma. Αναφέρουμε παρακάτω τη 'συμμετρική' εκδοχή του λήμματος.

Λήμμα 1. *Έστω $A_1, A_2, \ldots, A_k$ μια σειρά από γεγονότα τέτοια ώστε $Pr[A_i] \leq p$ για όλα τα $1 \leq i \leq k$ και κάθε γεγονός είναι ανεξάρτητο από όλα τα άλλα εκτός από το πολύ d από αυτά. Αν*

$$ep(d+1) \leq 1,$$

όπου $e = 2.718\ldots$ η βάση των φυσικών λογαρίθμων, τότε υπάρχει πιθανότητα να μη συμβεί κανένα από αυτά, δηλαδή:

$$Pr[\cap_{i=1}^{k} \bar{A}_i] > 0$$

Οι εφαρμογές του παραπάνω λήμματος είναι πάρα πολλές. Μερικές από αυτές μπορούν να βρεθούν στα [2, 6, 4, 3, 7, 21]. Μία από αυτές με την οποία θα ασχοληθούμε και περαιτέρω στην εργασία αυτή είναι ο χρωματισμός υπεργράφου.

Κεφάλαιο 4

Υπάρχοντες Αλγόριθμοι

4.1 Γενικά

Όπως αναφέρθηκε και προηγουμένως, υπάρχουν διάφορα είδη αλγορίθμων με στόχο την επίλυση ενός προβλήματος σε υπογραμμικό χρόνο, όπως οι παράλληλοι αλγόριθμοι και οι προσεγγιστικοί αλγόριθμοι. Κατανεμημένοι αλγόριθμοι έχουν προταθεί τα τελευταία χρόνια για αρκετά προβλήματα, όπως για χρωματισμό, maximal independent set, dominating set, κτλπ (βλέπε [38, 39, 40, 41, 42, 43, 12, 13]). Στη συνέχεια θα παρουσιάσουμε δύο τέτοιους κατανεμημένους αλγορίθμους: του Luby για maximal independent set και του Alon για χρωματισμό υπεργράφου. Στους αλγόριθμους αυτούς θα βασιστούμε για να παραγάγουμε αλγόριθμους τοπικού υπολογισμού για τα αντίστοιχα προβλήματα.

4.2 Ο αλγόριθμος του Luby

Στο Σχήμα 4.1 φαίνεται ο αλγόριθμος του Luby για το πρόβλημα της εύρεσης ενός maximal independent set ενός γράφου. Ο αλγόριθμος αυτός είναι ένας Monte Carlo αλγόριθμος, δηλαδή τρέχει σε ντετερμινιστικό χρόνο εκτέλεσης αλλά το αποτέλεσμα που δίνει ενδέχεται να είναι λάθος με κάποια μικρή πιθανότητα. Είσοδος του αλγορίθμου είναι ένας μη κατευθυνόμενος γράφος $G = (V, E)$. Έξοδος, φυσικά, είναι ένα MIS $I \subseteq V$. Ως $d(v)$ νοείται ο βαθμός του κόμβου v στο γράφο G'.

```
I ← ∅
G'=(V', E') ← G=(V, E)
Όσο V' ≠ ∅ επανέλαβε
        Παράλληλα, για κάθε v∈V', υπολόγισε το d(v)
        Παράλληλα, για κάθε v∈V'
                Αν d(v)=0 τότε πρόσθεσε το v στο I και σβήστο από το V'
        Υπολόγισε το n'=|V'|
        Βρες v∈V' τέτοιο ώστε d(v) είναι μέγιστο
        Αν d(v)>=n'/16 τότε βάλε το v στο I και όρισε G' το γράφο που προκύπτει
        από τους κόμβους V'-({v}∪N({v}))
        Αλλιώς:
                Υπολόγισε έναν πρώτο q τέτοιο ώστε n'<=q<=2n'
                Στην τύχη επέλεξε x και y τέτοια ώστε 0<=x,y<=q-1
                X← ∅
                Παράλληλα, για κάθε  v∈V',
                        Υπολόγισε n(v) = q / (2d(v))
                        Υπολόγισε I(v) = (i+vj) mod q
                        Αν I(v) <= n(v) τότε πρόσθεσε το v στο X
                I' ← X
                Παράλληλα, για κάθε  v∈X,  w∈X,
                        Αν (v, w) ∈ E' τότε
                                Αν d(v) <= d(w) τότε I' ← I' - {v}
                                Αλλιώς I' ← I' - {w}
                I ← I∪I'
                Y ← I'∪N(I')
                G' = (V', E') είναι ο υπογράφος που προκύπτει από το V' - Y
```

Σχήμα 4.1: Ο αλγόριθμος του Luby

Ανάλυση Έστω m_1 ο αριθμός των ακμών του G' πριν την εκτέλεση του βρόχου while, m_2 η τυχαία μεταβλητή του αριθμού των ακμών του G' μετά την εκτέλεση του βρόχου while και m_3 η τυχαία μεταβλητή του αριθμού των

ακμών που αφαιρέθηκαν από τον G' κατά την εκτέλεση του βρόχου while. Τότε, φυσικά, έχουμε $m_2 = m_1 - m_3$. Έστω E_v το γεγονός ο κόμβος v να επιλεγεί στο βήμα (*) του αλγορίθμου. Τα γεγονότα αυτά είναι αμοιβαίως ανεξάρτητα (mutually independent). Έστω, ακόμη, $p_v = Pr[E_v] = \frac{1}{2d(v)}$ και $sum_v = \sum_{w \in adj(v)} p_w$.

Λήμμα 2. $Pr[v \in N(I')] \geq \frac{1}{4} \min(sum_v, 1)$.

Απόδειξη. Αριθμούμε τους γειτονικούς κόμβους του v ως εξής $\{1, \ldots, d(v)\}$. Έστω

$$E'_1 = E_1$$
$$E'_i = (\bigcap_{j=1}^{i-1} E_j) \cap \neg E_i, \quad 2 \leq i \leq d(v)$$

και

$$A_i = \bigcap_{x \in adj(i), d(x) \geq d(i)} \neg E_x.$$

Τότε,

$$Pr[v \in N(I')] \geq \sum_{i=1}^{d(v)} Pr[A_i | E'_i] Pr[E'_i].$$

Αλλά,

$$Pr[A_i | E'_i] \geq Pr[A_i] \geq 1 - \sum_{x \in adj(i), d(x) \geq d(i)} p_x \geq \frac{1}{2}.$$

Αποδεικνύεται σε επόμενο λήμμα ότι

$$\sum_{i=1}^{d(v)} Pr[E'_i] = Pr[\cup_{i=1}^{d(v)} E_i] \geq \frac{1}{2} \min(sum_v, 1).$$

Οπότε έχουμε τελικά:

$$Pr[v \in N(I')] \geq \frac{1}{4} \min(sum_v, 1). \quad \square$$

Αποδεικνύουμε στη συνέχεια το λήμμα που χρησιμοποιήσαμε στην παραπάνω απόδειξη. Η απόδειξη γίνεται για την περίπτωση που τα γεγονότα $\{E_w\}$ είναι ανεξάρτητα ανά δύο (pairwise independent), υπόθεση ασθενέστερη από αυτήν που είχαμε κάνει αρχικά ότι τα γεγονότα αυτά είναι αμοιβαίως ανεξάρτητα. Άρα ο αλγόριθμος εξακολουθεί να λειτουργεί κανονικά και υπό ασθενέστερες συνθήκες.

Λήμμα 3. *Έστω γεγονότα $E_1, \ldots, E_n$, $1 \leq i \leq n$, τέτοια ώστε $Pr[E_i] = p_i$ και για $1 \leq i \neq j \leq n$, $Pr[E_i \cap E_j] = p_i \cdot p_j$. Θέτουμε $sum = \sum_i p_i$. Τότε $Pr[\cup_i E_i] \geq \frac{1}{2}\min(sum, 1)$.*

Απόδειξη. Έστω $E'_k = \bigcup_{i=1}^{k} E_i$ και $\alpha_k = \sum_{i=1}^{k} p_i$. Υποθέτουμε ότι τα γεγονότα $\{E_w\}$ βρίσκονται σε φθίνουσα σειρά πιθανότητας, δηλαδή $p_1 \geq p_2 \geq \cdots \geq p_n$, χωρίς άρση της γενικότητας. Τότε έχουμε $Pr[E'_n] \geq Pr[E'_k]$ για κάθε $1 \leq k \leq n$. Επειδή τα γεγονότα $\{E_w\}$ είναι ανεξάρτητα ανά δύο έχουμε ότι

$$Pr[E'_k] \geq \alpha_k - \sum_{1 \leq i < j \leq k} p_i \cdot p_j.$$

Το δεξί μέρος της ανισότητας αυτής, όμως, ελαχιστοποιείται όταν όλα τα p_i ισούνται με $\frac{\alpha_k}{k}$. Οπότε,

$$Pr[E'_k] \geq \alpha_k \left[1 - \frac{\alpha_k(k-1)}{2k}\right].$$

Παίρνουμε τώρα περιπτώσεις για το α_n. Αν $\alpha_n \leq 1$ τότε

$$Pr[E'_n] \geq \frac{\alpha_n}{2} = \frac{sum}{2}.$$

Αν $\alpha_n \geq 1$ τότε έστω $l = \min\{k : \alpha_k \geq 1\}$. Αν $l = 1$ τότε

$$Pr[E'_1] \geq 1.$$

Αν $l \geq 2$ τότε $\alpha_{l-1} < 1 \leq \alpha_l \leq \frac{l}{l-1}$. Επομένως,

$$Pr[E'_l] \geq \alpha_l \left[1 - \frac{\alpha_l(l-1)}{2l}\right] \geq \frac{1}{2}. \quad \square$$

Μπορούμε πλέον να αποδείξουμε το κύριο θεώρημα που θα μας βοηθήσει να υπολογίσουμε το χρόνο εκτέλεσης του αλγορίθμου.

Θεώρημα 7. $E[m_3] \geq \frac{1}{8}m_1$

Απόδειξη. Για οποιοδήποτε $W \subseteq V$, έστω ET το σύνολο των ακμών που ακουμπάν το W:

$$ET(W) = \{(v, w) \in E | v \in W \text{ ή } w \in W\}.$$

Χρησιμοποιώντας αυτόν το συμβολισμό οι ακμές που αφαιρέθηκαν από τον G' κατά την εκτέλεση του βρόχου while είναι οι ακμές στο $ET[I' \cup N(I')]$. Κάθε ακμή (v, w) που ανήκει στο παραπάνω σύνολο ανήκει σε αυτό είτε επειδή $v \in I' \cup N(I')$ είτε επειδή $w \in I' \cup N(I')$. Επομένως,

$$\begin{aligned} E[m_3] &\geq \frac{1}{2} \sum_{v \in V'} d(v) Pr[v \in I' \cup N(I')] \\ &\geq \frac{1}{2} \sum_{v \in V'} d(v) Pr[v \in N(I')]. \end{aligned}$$

Από το λήμμα 3, $Pr[v \in N(I')] \geq \frac{1}{4} \min(sum_v, 1)$. Οπότε, τελικά:

$$E[m_3] \geq \frac{1}{8} \left(\sum_{\substack{v \in V' \\ sum_v \leq 1}} d(v) sum_v + \sum_{\substack{v \in V' \\ sum_v > 1}} d(v) \right) \geq$$

$$\frac{1}{8} \left(\sum_{\substack{v \in V' \\ sum_v \leq 1}} \sum_{w \in adj(v)} \frac{d(v)}{2d(w)} + \sum_{\substack{v \in V' \\ sum_v > 1}} \sum_{w \in adj(v)} 1 \right) \geq$$

$$\frac{1}{8} \sum_{\substack{(v,w) \in E' \\ sum_v \leq 1 \\ \wedge sum_w \leq 1}} \frac{1}{2} \left(\frac{d(v)}{d(w)} + \frac{d(w)}{d(v)} \right) + \frac{1}{8} \sum_{\substack{(v,w) \in E' \\ sum_v \leq 1 \\ \wedge sum_w \leq 1}} \left(\frac{d(v)}{2d(w)} + 1 \right) + \frac{1}{8} \sum_{\substack{(v,w) \in E' \\ sum_v > 1 \\ \wedge sum_w \leq 1}} 2 \geq$$

$$\geq \frac{1}{8} |E'| = \frac{1}{8} m_1 \quad \square$$

Χρόνος εκτέλεσης Από το Θεώρημα 7 έχουμε $E[m_3] \geq \frac{1}{8} m_1$ και συνεπώς $E[m_2] \geq \frac{7}{8} m_1$. Επομένως, ο αναμενόμενος αριθμός εκτελέσεων του βρόχου while είναι $O(\log n)$ με αρκετά μεγάλη πιθανότητα. Κάθε επανάληψη του βρόχου μπορεί να εκτελεστεί σε χρόνο $O(\log n)$ χρησιμοποιώντας $O(m)$ επεξεργαστές, όπου ο αναμενόμενος αριθμός τυχαίων bits είναι $O(n)$. Άρα, τελικά, ο αναμενόμενος χρόνος εκτέλεσης του αλγορίθμου του Luby είναι $O(\log^2 n)$ χρησιμοποιώντας $O(m)$ επεξεργαστές, όπου αναμενόμενος αριθμός τυχαίων bits είναι $O(n \log n)$.

4.3 Ο αλγόριθμος του Alon

Στην ενότητα αυτή θα παρουσιάσουμε τον αλγόριθμο του Alon για την εύρεση ενός 2-χρωματισμού ενός υπεργράφου H. Ο αλγόριθμος αυτός είναι πιθανοτικός και τρέχει σε τρεις φάσεις. Στη Φάση 1 χρωματίζουμε τυχαία και ανεξάρτητα όλους τους κόμβους του H είτε με μπλε είτε με κόκκινο με ίση πιθανότητα. Αν οποιαδήποτε ακμή αποκτήσει τουλάχιστον αn κόμβους του ίδιου χρώματος τότε ονομάζουμε την ακμή αυτή 'κακή'. Έστω B το σύνολο όλων των 'κακών' ακμών. Μια τυχαία ακμή ανήκει στο σύνολο B με πιθανότητα το πολύ $2\sum_{i\leq \alpha n}\binom{n}{i}/2^n \leq 2\cdot 2^{(H(a)-1)n}$, όπου $H(x)=-x\log x-(1-x)\log(1-x)$ η δυαδική συνάρτηση εντροπίας.

Αποδεικνύουμε στη συνέχεια ένα λήμμα που θα μας βοηθήσει στην ανάλυση του αλγορίθμου. Στην απόδειξη αυτή κάνουμε χρήση μιας συγκεκριμένης δομής που ονομάζουμε 1,2-δέντρο. Ένα υποσύνολο κόμβων C του G (όπου G ο γράφος εξαρτήσεων) ονομάζεται 1,2-δέντρο αν τραβώντας μια ακμή μεταξύ κάθε $A_i, A_j \in C$ των οποίων η απόσταση στον G είναι 1 ή 2, ο προκύπτων γράφος είναι συνεκτικός.

Λήμμα 4. *Η πιθανότητα οποιοδήποτε 1,2-δέντρο C στον G του οποίου όλοι κόμβοι ανήκουν στο B να έχει μέγεθος το πολύ $d\log(2N)$ είναι τουλάχιστον 1/2.*

Απόδειξη. Θα φράξουμε το μέγεθος των ζητούμενων 1,2-δέντρων φράσσοντας τον αριθμό των 2,3-δέντρων μεγέθους u στον G. 2,3-δέντρο ονομάζεται ένα υποσύνολο κόμβων T του G αν τραβώντας μια ακμή μεταξύ κάθε $A_i, A_j \in T$ των οποίων η απόσταση στον G είναι 2 ή 3, ο προκύπτων γράφος είναι συνεκτικός και, επίσης, οι αποστάσεις των A_i, A_j στον G είναι τουλάχιστον 2. Έστω γράφος ενός συνόλου κόμβων του G όπου υπάρχει ακμή μεταξύ δύο κόμβων του αν και μόνο αν οι κόμβοι αυτοί στον G έχουν απόσταση 2 ή 3. Παρατηρούμε ότι οποιοδήποτε 2,3-δέντρο σε ένα σύνολο T από κόμβους του G περιέχει ένα δέντρο με κόμβους του T στο νέο γράφο. Ο νέος γράφος έχει μέγιστο βαθμό μικρότερο από $D = d^3$.

Ισχύει ([37]) ότι ένα άπειρο D-regular δέντρο με ρίζα περιέχει ακριβώς $\frac{1}{(D-1)u+1}\binom{Du}{u}$ τον αριθμό υποδέντρα μεγέθους u με ρίζα. Αμέσως συμπεραίνουμε ότι ο αριθμός των δέντρων μεγέθους u τα οποία περιέχουν έναν συγκεκριμένο κόμβο σε οποιονδήποτε γράφο με μέγιστο βαθμό D δεν υπερβαίνει τον παραπάνω αριθμό, ο οποίος είναι μικρότερος από $(eD)^u$. Για οποιοδήποτε 2,3-δέντρο T μεγέθους u ξέρουμε επίσης ότι $Pr[T \subseteq B] \leq$. Επομένως, ο αναμενόμενος αριθμός από 2,3-δέντρα $T \subseteq B$ μεγέθους u είναι το πολύ $N(eDp)^u$.

Αν τώρα υποθέσουμε ότι ισχύει $4ed^3 < 2^{n(1-H(\alpha))}$, τότε $eDp < 0.5$ και αν $u = \log(2N)$ ο παραπάνω όρος είναι το πολύ $1/2$.

Έχουμε δείξει δηλαδή έως τώρα ότι δεν υπάρχει 2,3-δέντρο με μέγεθος μεγαλύτερο από $\log(2N)$ του οποίου όλοι οι κόμβοι ανήκουν στο B με πιθανότητα τουλάχιστον $1/2$. Για ένα maximal 2,3-δέντρο T στο C ισχύει ότι κάθε $A_i \in C$ είναι γείτονας στο G ενός $A_j \in T$. Υπάρχουν, όμως, λιγότεροι από d γείτονες A_i ενός οποιουδήποτε A_j. Συνεπώς, $\log(2N) \geq |T| \geq |C|/d$ και τελικά:

$$|C| \leq d \log(2N). \quad \square$$

Επαναλαμβάνουμε τη Φάση 1 του αλγορίθμου μέχρις ότου δεν υπάρχει κανένα 1,2-δέντρο με μέγεθος μεγαλύτερο του $d\log(2N)$ του οποίου όλοι οι κόμβοι να ανήκουν στο B. Τότε λέμε ότι η Φάση 1 είναι *επιτυχημένη* και τερματίζουμε την επανάληψη. Η πιθανότητα να είναι επιτυχημένη μια εκτέλεση της Φάσης 1 είναι τουλάχιστον $1/2$ με βάση το παραπάνω λήμμα. Ο αναμενόμενος χρόνος εκτέλεσης της Φάσης 1, επομένως, είναι γραμμικός.

Στη Φάση 2 θα διορθώσουμε τις 'κακές' ακμές που προέκυψαν στη Φάση 1. Αυτό το πετυχαίνουμε επαναχρωματίζοντας τους κόμβους του H που ανήκουν στις 'κακές' ακμές, ώστε να μην υπάρχει καμία μονοχρωματική ακμή. Με τον επαναχρωματισμό υπάρχει το ενδεχόμενο κάποιες ακμές να γίνουν μονοχρωματικές. Δε χρειάζεται όμως να ανησυχούμε για όλες τις ακμές. Μόνο για αυτές που έχουν τουλάχιστον αn κόμβους που ανήκουν σε 'κακές' ακμές. Αν μια ακμή έχει λιγότερους από τόσους κόμβους σε 'κακές' ακμές, τότε λιγότεροι από αn κόμβοι θα επαναχρωματιστούν και η ακμή αυτή έχει τουλάχιστον αn κόμβους κάθε χρώματος πριν τον επαναχρωματισμό. Επομένως, μετά τον επαναχρωματισμό δε πρόκειται να γίνει μονοχρωματική. Τις ακμές που έχουν τουλάχιστον αn κόμβους που ανήκουν σε 'κακές' ακμές τις ονομάζουμε 'επικίνδυνες'. Επαναχρωματίζοντας όλους τους κόμβους στις 'κακές' ακμές τυχαία και ανεξάρτητα επαναχρωματίζουμε τουλάχιστον αn κόμβους σε κάθε 'επικίνδυνη' ακμή και άρα η πιθανότητα αυτή να γίνει μονοχρωματική είναι λιγότερο από $2^{-\alpha n}$. Υποθέτοντας ότι $2e(d+1) < 2^{\alpha n}$ και δεδομένου ότι κάθε ακμή τέμνει το πολύ άλλες d, από το Λήμμα 1 έχουμε ότι υπάρχει επαναχρωματισμός όπου καμία ακμή να είναι μονοχρωματική.

Μια πολύ σημαντική παρατήρηση για τη διαδικασία του επαναχρωματισμού είναι ότι ο επαναχρωματισμός των ακμών κάθε maximal 1,2-δέντρου C από 'κακές' ακμές μπορεί να γίνει ξεχωριστά. Και αυτό γιατί δεν υπάρχει 'επικίνδυνη' ακμή που να τέμνει δύο ή περισσότερα τέτοια maximal 1,2-δέντρα. Επομένως, μπορούμε να επαναχρωματίσουμε τις ακμές σε κάθε C προσέχοντας μόνο να μη

γίνει μονοχρωματική κάποια 'επικίνδυνη' ακμή που να τέμνει κάποια ακμή του C. Δεδομένου ότι ο αριθμός των κόμβων που πρέπει να επαναχρωματιστούν σε κάθε 1,2-δέντρο C είναι σχετικά μικρός ($O(\log N)$), μπορούμε να βρούμε τον κατάλληλο επαναχρωματισμό εφαρμόζοντας εξαντλητική αναζήτηση. Κάνοντάς το αυτό για ένα συγκεκριμένο C θα πάρει χρόνο $O(2^{O(\log N)}) = N^{O(1)}$ και συνολικά για όλα τα C θα πάρει πολυωνυμικό χρόνο. Άρα τελικά ο αλγόριθμος του Alon τρέχει σε πολυωνυμικό χρόνο. Ο αλγοριθμός αυτός μπορεί να παραλληλοποιηθεί, όπως έχουμε αναφέρει και πρωτύτερα, ώστε να τρέχει σε χρόνο $O(\log n)$ χρησιμοποιώντας $n^{O(1)}$ παράλληλους επεξεραστές.

Κεφάλαιο 5

Αλγόριθμοι Τοπικού Υπολογισμού

Οι αλγόριθμοι τοπικού υπολογισμού που θα παρουσιαστούν εδώ αποτελούν γενίκευση των *τοπικών αλγορίθμων* (local algorithms). Οι τοπικοί αλγόριθμοι, όταν εφαρμόζονται σε γραφοθεωρητικά προβλήματα, εξετάζουν μόνο ένα μικρό τμήμα της εισόδου το οποίο βρίσκεται γύρω από ένα συγκεκριμένο κόμβο ώστε να παράξουν τη ζητούμενη λύση. Χρησιμοποιώντας την αναγωγή των Parnas και Ron [54] μπορούμε να μετατρέψουμε κατανεμημένους αλγόριθμους με σταθερό αριθμό γύρων, όπως αυτούς του προηγούμενου κεφαλαίου, σε τοπικούς αλγορίθμους.

5.1 Μοντέλο

Έστω $n = |x|$ το μέγεθος της εισόδου x.

Ορισμός 9. Για είσοδο x, έστω $F(x) = \{y | y \text{ δεκτή λύση για είσοδο } x\}$. Το ζητούμενο *πρόβλημα* είναι η εύρεση ενός $y \in F(x)$.

Ορισμός 10. Ένας $(t(n), s(n), \delta(n))$-*αλγόριθμος τοπικού υπολογισμού* $\mathcal{A}$ είναι ένας (πιθανοτικός) αλγόριθμος ο οποίος δέχεται μια σειρά από ερωτήματα $q_1, \ldots, q_k$ και δίνει αντίστοιχα απαντήσεις $y_{q_1}, \ldots, y_{q_k}$, οι οποίες αποτελούν τμήματα μιας συγκεκριμένης δεκτής απάντησης $y \in F(x)$. Η πιθανότητα να επιτύχει σε όλα τα ερωτήματα είναι τουλάχιστον $1 - \delta(n)$. Έχει στη διάθεσή του μια γεννήτρια τυχαίων αριθμών και μπορεί να αποθηκεύει και να ανακτά πληροφορίες από προηγούμενους υπολογισμούς. Ο $\mathcal{A}$ τρέχει σε χρόνο το πολύ

$t(n)$, ο οποίος θέλουμε να είναι υπογραμμικός σε σχέση με το μέγεθος της εισόδου, και χρειάζεται χώρο το πολύ $s(n)$.

Τόσο η είσοδος x όσο και η έξοδος y θεωρούμε ότι είναι πολύ μεγάλες. Συγκεκριμένα η είσοδος δε χρειάζεται να είναι σε κάποια προκαθορισμένη μορφή. Επίσης, ο παραπάνω ορισμός των αλγορίθμων τοπικού υπολογισμού αποκλείει την πιθανότητα ο αλγόριθμος να απαντήσει υπολογίζοντας πρώτα ολόκληρη τη λύση y.

Ορισμός 11. Ένας ΑΤΥ $\mathcal{A}$ ονομάζεται *query order oblivious* (*query oblivious* για συντομία) αν οι απαντήσεις του δεν εξαρτώνται από τη σειρά των ερωτημάτων.

Ορισμός 12. Ένας ΑΤΥ $\mathcal{A}$ ονομάζεται *παραλληλοποιήσιμος* αν υποστηρίζει παράλληλα ερωτήματα.

5.2 Maximal Independent Set

Έστω γράφος G ο οποίος έχει n κόμβους με μέγιστο βαθμό d.

5.2.1 Επισκόπηση του Αλγόριθμου

Ο αλγόριθμος τοπικού υπολογισμού που περιγράφουμε εδώ, με βάση τη λογική για τους ΑΤΥ που αναπτύξαμε παραπάνω, δέχεται ερωτήματα για κόμβους του G και απαντάει στο αν ο εκάστοτε κόμβος ανήκει σε κάποιο maximal independent set. Ο αλγόριθμος τρέχει σε 2 φάσεις. Στην πρώτη φάση προσομοιώνει τον παράλληλο αλγόριθμο του Luby που περιγράψαμε στην Ενότητα 4.2: ο ζητούμενος κόμβος προσπαθεί να μπει στο IS με μια μικρή πιθανότητα και τα καταφέρνει μόνο αν κανένας από τους γείτονές του δεν προσπαθεί να κάνει το ίδιο. Επαναλαμβάνουμε τη Φάση 1 έως ότου σχεδόν όλοι οι κόμβοι είτε έχουν μπει στο IS είτε κάποιος γείτονάς τους είναι στο IS (οπότε αυτοί δεν πρόκειται να μπουν ποτέ). Αν ισχύει ένα από τα δυο για το ζητούμενο κόμβο τότε ο αλγόριθμος τελείωσε. Αλλιώς συνεχίζει με τη Φάση 2 όπου εκτελεί τον άπληστο αλγόριθμο (που αναφέραμε στην Ενότητα 3.3) στη συνεκτική συνιστώσα με τους εναπομείναντες κόμβους γύρω από τον ζητούμενο. Η συνιστώσα αυτή έχει, ύστερα από τη Φάση 1, σχετικά μικρό μέγεθος οπότε και η Φάση 2 θα ολοκληρωθεί πολύ γρήγορα.

Ο αλγόριθμος αντιστοιχεί σε κάθε κόμβο του G και μια κατάσταση. Ανά πάσα στιγμή κάθε κόμβος μπορεί να βρίσκεται σε μία εκ των εξής τριών καταστάσεων: (α) 'selected', δηλαδή ανήκει στο MIS, (β) 'deleted', τουλάχιστον ένας γείτονάς του ανήκει στο MIS, και (γ) '$\perp$'. Όλοι οι κόμβοι ξεκινούν στην κατάσταση '$\perp$'. Άπαξ και κάποιος κόμβος μεταβεί σε κατάσταση 'selected' ή 'deleted' μένει σε αυτή καθ' όλη την υπόλοιπη διάρκεια εκτέλεσης του αλγόριθμου. Θα περιγράψουμε τώρα αναλυτικά τις δύο αυτές φάσεις και θα ορίσουμε τις χρονικές πολυπλοκότητές τους.

5.2.2 Φάση 1

Στο Σχήμα 5.1 φαίνεται ο ψευδοκώδικας της Φάσης 1 του αλγορίθμου. Η συνάρτηση MIS(u, i) επιστέφει ως αποτέλεσμα την κατάσταση του κόμβου u στο τέλος του γύρου i. Στην ουσία, η MIS(u, i) τρέχει τον αλγόριθμο του Luby που έχουμε ήδη παρουσιάσει για έναν γύρο. Στο γύρο αυτό ο κόμβος προσπαθεί να 'επιλέξει' τον εαυτό του με πιθανότητα $1/2$d. Το ίδιο προσπαθούν να κάνουν και όλοι οι γείτονές του που είναι σε κατάσταση '$\perp$'. Ο κόμβος u μπαίνει τελικά στο IS μόνο αν επιλεγεί ο ίδιος αλλά κανένας από τους γείτονές του, αλλιώς συνεχίζουμε στον επόμενο γύρο. Αν ο u μπει εν τέλει στο IS ανανεώνουμε την κατάστασή του σε 'selected' και την κατάσταση όλων των γειτόνων του σε 'deleted'.

MIS(u,i): κατάσταση του κόμβου u στο γύρο i

1. Αν u = "selected" ή "deleted", επέστρεψε "selected" ή "deleted" αντίστοιχα
2. Για κάθε γείτονα v του u: Αν MIS(v,i-1) = "selected", τότε u ← "deleted"
3. Ο u επιλέγει τον εαυτό του με πιθανότητα 1/2d
   Αν επιλεγεί τότε
   i. Για κάθε γείτονα v του u: Αν v = "⊥", ο v επιλέγει τον εαυτό του με πιθανότητα 1/2d
   ii. Αν επιλεγεί κάποιος γείτονας, επέστρεψε "⊥"
   iii. Αλλιώς, u ← "selected"
   Αλλιώς, επέστρεψε "⊥"

Σχήμα 5.1: Φάση 1 του ΑΤΥ για MIS

Με βάση το [45], αν τρέξουμε τη Φάση 1 για $O(\log n)$ γύρους τότε δε θα υπάρχει κανένας κόμβος σε κατάσταση '$\perp$' μετά την ολοκλήρωση της φάσης

αυτής. Κανένας κόμβος σε κατάσταση '$\perp$' όμως συνεπάγεται ότι βρέθηκε ένα MIS και ο αλγόριθμος έχει ολοκληρωθεί. Το να επαναλάβουμε τη Φάση 1 για τόσους γύρους ωστόσο καθυστερεί πάρα πολύ. Κάνοντας την εξής παρατήρηση μπορούμε να μειώσουμε τον αριθμό των γύρων που θα τρέξουμε τη Φάση 1 αρκετά: καθώς αυξάνουμε σταδιακά τους γύρους από $O(1)$ έως $O(\log n)$ μειώνεται αντίστοιχα το μέγεθος των συνεκτικών συνιστωσών από εναπομείναντες κόμβους για τους οποίους δεν έχει αποφασίσει ο αλγόριθμος αν ανήκουν ή όχι στο MIS. Αποδεικνύεται στο [56] πως αν τρέξουμε τη Φάση 1 για $O(d \log d)$ γύρους τότε όλες οι συνεκτικές συνιστώσες από τους εναπομείναντες κόμβους σε κατάσταση '$\perp$' έχουν, σχεδόν σίγουρα, μέγεθος το πολύ $\text{poly}(d) \log n$. Θα μπορέσουμε, έτσι, στη Φάση 2 να εκτελέσουμε τον άπληστο αλγόριθμο στη συνεκτική συνιστώσα με τους εναπομείναντες κόμβους γύρω από τον ζητούμενο πολύ γρήγορα, αγνοώντας όλες τις υπόλοιπες συνιστώσες. Συγκεκριμένα αποδεικνύεται ότι:

Θεώρημα 8. *Αφού τρέξουμε τη Φάση 1, όλες οι συνεκτικές συνιστώσες από τους εναπομείναντες κόμβους σε κατάσταση '$\perp$' έχουν, με πιθανότητα τουλάχιστον $1 - 1/n$, μέγεθος το πολύ $O(\text{polylog}(d)n)$.*

Χρόνος εκτέλεσης Για να υπολογίσουμε το $\text{MIS}(u, i)$ θα πρέπει να καλέσουμε αναδρομικά το $\text{MIS}(v, j)$ για όλους τους προηγούμενους γύρους, $1 \leq j \leq i-1$, τόσο για τον u όσο και για όλους τους γείτονές του, $v \in N^+(u)$. Επειδή ο μέγιστος αριθμός γειτόνων ενός κόμβου είναι d και τρέχουμε τη Φάση 1 για $O(d \log d)$ γύρους, ο χρόνος εκτέλεσης της Φάσης 1 είναι $d^{O(r)} = d^{O(d \log d)}$ (το δέντρο της αναδρομής έχει ύψος $O(r)$ και branching factor d).

5.2.3 Φάση 2

Αν ο ζητούμενος κόμβος u μετά τη Φάση 1 είναι ακόμη σε κατάσταση '$\perp$' (δηλαδή ο αλγόριθμος δεν αποφάνθηκε αν ο κόμβος ανήκει ή όχι στο MIS μετά από $O(d \log d)$ γύρους της Φάσης 1) τότε πρέπει να τρέξουμε τη Φάση 2. Στη Φάση 2 κρατάμε μόνο τους κόμβους του G σε κατάσταση '$\perp$' και βρίσκουμε τις συνεκτικές συνιστώσες του νέου γράφου. Έστω $C(u)$ η συνεκτική συνιστώσα που περιέχει τον u. Θα επικεντρωθούμε αποκλειστικά σε αυτή τη συνεκτική συνιστώσα, μιας και το αν ανήκει ο u στο MIS εξαρτάται μόνο από το αν ανήκουν ή όχι σε αυτό οι γείτονές τους και μόνο. Ο u προφανώς δεν έχει γείτονες σε οποιαδήποτε άλλη συνεκτική συνιστώσα, οπότε μπορούμε να αγνοήσουμε όλες τις υπόλοιπες. Αν το μέγεθός της $C(u)$ είναι μεγαλύτερο από $c_2 \log n$ για κάποια

σταθερά c_2 που εξαρτάται μόνο από το d, τότε διακόπτουμε την εκτέλεση και επιστρέφουμε 'Fail' ώστε να μην παραβιάσουμε το χρονικό όριο εκτέλεσης που θέλουμε ιδανικά να επιτύχουμε. Αλλιώς, εκτελούμε τον άπληστο αλγόριθμο που αναφέραμε στην εισαγωγή του κεφαλαίου αυτού για να βρούμε το MIS της $C(u)$, η οποία είναι αρκετά μικρή ώστε ο άπληστος αλγόριθμος να τελειώσει σχετικά γρήγορα.

Χρόνος εκτέλεσης Για να ελέγξουμε ότι όλοι οι κόμβοι της $C(u)$ είναι σε κατάσταση '$\perp$' τρέχουμε τη Φάση 1 για καθέναν από αυτούς με χρόνο κάθε φορά $d^{O(d \log d)}$. Άρα ο χρόνος εκτέλεσης της Φάσης 2 είναι το πολύ $O(|C(u)|) \cdot d^{O(d \log d)} \leq O(d^{O(d \log d)} \cdot \log n)$. Δεδομένου ότι η ποσότητα αυτή είναι μεγαλύτερη από το χρόνο εκτέλεσης της Φάσης 1, τόσος είναι και ο συνολικός χρόνος εκτέλεσης του αλγορίθμου τοπικού υπολογισμού που παρουσιάσαμε για το MIS.

Συνολικός Χώρος Για τη Φάση 1 χρειάζεται να διατηρούμε στη μνήμη τις τυχαίες επιλογές κάθε κόμβου, ώστε οι απαντήσεις του αλγορίθμου μας να είναι συνεπείς ως προς κάποια δεκτή λύση. Στη Φάση 2 πρέπει να κρατάμε πληροφορίες για τους κόμβους της συνεκτικής συνιστώσας $C(u)$, οι οποίοι είναι το πολύ $O(\log n)$. Άρα ο αλγόριθμος χρειάζεται συνολικά χώρο $O(n)$.

Έχουμε, τελικά, το εξής αποτέλεσμα:

Θεώρημα 9. *Έστω γράφος G με n κόμβους και μέγιστο βαθμό d. Τότε υπάρχει ένας $(O(d^{O(d \log d)} \cdot \log n), O(n), 1/n)$-αλγόριθμος τοπικού υπολογισμού ο οποίος, δοθέντος ενός κόμβου u, αποφασίζει αν ο u ανήκει σε κάποιο maximal independent set. Επίσης, όλες του οι απαντήσεις είναι συνεπείς ως προς κάποιο συγκεκριμένο MIS.*

Χρησιμοποιώντας τον Αλγόριθμο Τοπικού Υπολογισμού που παρουσιάσαμε παραπάνω καταφέραμε, επομένως, να πάρουμε την επιθυμητή πληροφορία για ένα συγκεκριμένο κόμβο σε χρόνο μόλις $O(d^{O(d \log d)} \cdot \log n)$. Αντιθέτως, αν περιμέναμε να αποκτήσουμε με τον αλγόριθμο του Luby ολόκληρο το maximal independent set θα χρειαζόμασταν $O(\log^2 n)$ χρόνο και $O(m)$ επεξεργαστές, καθυστερώντας έτσι και ξοδεύοντας αρκετά περισσότερους πόρους.

5.3 2-χρωματισμός υπεργράφου

Έστω k-ομοιόμορφος υπεργράφος H ο οποίος έχει m κόμβους, N υπερακμές και κάθε υπερακμή τέμνει το πολύ d άλλες. Άρα $m \leq kN$ και αν θεωρήσουμε τα k και d σταθερές τότε έχουμε $m = O(N)$.

Επίσης, γνωρίζουμε ότι οι Erdös και Lovász [21] απέδειξαν χρησιμοποιώντας το Lovász Local Lemma το εξής:

Θεώρημα 10. *Αν $e(d+1) < 2^{n-1}$ τότε οποιοσδήποτε υπεργράφος H στον οποίο κάθε ακμή έχει τουλάχιστον $n \geq 2$ κόμβους και καμία ακμή δεν τέμνει περισσότερες από d άλλες ακμές είναι 2-χρωματίσιμος, όπου $e = 2.718$.*

5.3.1 Επισκόπηση Αλγορίθμου

Ο αλγόριθμος τοπικού υπολογισμού που περιγράφουμε εδώ, με βάση πάλι τη λογική για τους ΑΤΥ που αναπτύξαμε προηγουμένως, δέχεται ερωτήματα για κόμβους του H και απαντάει με το χρώμα του εκάστοτε κόμβου σε κάποιο 2-χρωματισμό του H. Ο αλγόριθμος αυτός προσομοιώνει τον αλγόριθμο του Alon που περιγράψαμε στην Ενότητα 4.3. Τρέχει, επομένως, σε 3 φάσεις. Στην πρώτη φάση χρωματίζουμε έναν έναν τους κόμβους του H σε μια τυχαία σειρά (πχ στη σειρά με την οποία δέχεται τα ερωτήματα ο αλγόριθμος). Αν κάποια στιγμή μια υπερακμή έχει k_1 κόμβους ενός χρώματος και κανέναν του άλλου, τότε σημειώνουμε αυτήν την ακμή ως 'dangerous' και τους αχρωμάτιστους κόμβους της ως 'troubled'. Τους κόμβους αυτούς δεν τους χρωματίζουμε στη Φάση 1. Αν ο ζητούμενος κόμβος γίνει 'troubled' τρέχουμε τη Φάση 2. Αρχικά διαγράφουμε όλες τις υπερακμές οι οποίες περιέχουν πλέον κόμβους και των δύο χρωμάτων, μιας και τους αχρωμάτιστους κόμβους που περιέχουν μπορούμε πλέον να τους χρωματίσουμε όπως θέλουμε. Στη συνέχεια, επαναλαμβάνουμε την ίδια διαδικασία στη συνεκτική συνιστώσα από εναπομείναντες ακμές γύρω από τον ζητούμενο κόμβο, αλλά τώρα μια υπερακμή γίνεται 'dangerous' αν έχει $k_1 + k_2$ κόμβους ενός χρώματος και κανέναν του άλλου. Αν ο ζητούμενος κόμβος δε χρωματιστεί ούτε τώρα, τρέχουμε τη Φάση 3. Κατά τη διάρκεια της Φάσης 1 το μέγεθος των συνεκτικών συνιστωσών από εναπομείναντες υπερακμές μειώνεται αρκετά. Κατά τη διάρκεια της Φάσης 2 μειώνεται ακόμα περισσότερο. Αυτό μας επιτρέπει στη Φάση 3 να εκτελέσουμε μια εξαντλητική αναζήτηση για χρωματισμό στη συνεκτική συνιστώσα από τις νέες εναπομείναντες ακμές γύρω από τον ζητούμενο κόμβο. Η συνιστώσα αυτή είναι πλέον αρκετά μικρή οπότε η αναζήτηση θα τελειώσουμε σχετικά γρήγορα. Η ύπαρξη

του ζητούμενου χρωματισμού είναι εγγυημένη από το Lovász Local Lemma σύμφωνα με το [5].

Κατά την εκτέλεση του αλγόριθμου ένας κόμβος μπορεί να είναι σε μία από τις ακόλουθες καταστάσεις: 'uncolored', 'red', 'blue', 'trouble-1' και 'trouble-2'. Μια υπερακμή μπορεί να είναι σε μία από τις εξής: '$\perp$', 'safe', 'unsafe-1', 'unsafe-2', 'dangerous-1' και 'dangerous-2'. Οι αριθμοί στις καταστάσεις έχουν να κάνουν με το σε ποια φάση μπήκαν στην αντίστοιχη κατάσταση. Αρχικά όλοι οι κόμβοι είναι σε κατάσταση 'uncolored' και όλες οι υπερακμές σε '$\perp$'. Θα περιγράψουμε τώρα αναλυτικά τις τρεις αυτές φάσεις και θα ορίσουμε τις χρονικές πολυπλοκότητές τους.

5.3.2 Φάση 1

Στο Σχήμα 5.2 φαίνεται ο ψευδοκώδικας της Φάσης 1 του αλγορίθμου. Αν ο κόμβος x είναι ήδη ζωγραφισμένος επιστρέφουμε απευθείας το χρώμα του. Αν είναι σε κατάσταση 'trouble-1' τότε τρέχουμε τη Φάση 2. Αν είναι σε κατάσταση 'trouble-2' τότε τρέχουμε τη Φάση 3. Αλλιώς, επιλέγουμε ένα χρώμα στην τύχη και του αναθέτουμε αυτό. Στη συνέχεια, ανανεώνουμε την κατάσταση όλων των υπερακμών που τον εμπεριέχουν. Συγκεκριμένα, αν μια υπερακμή περιλαμβάνει πλέον κόμβους και των δύο χρωμάτων αλλάζουμε την κατάστασή της σε 'safe'. Αν περιλαμβάνει k_1 κόμβους του ίδιου χρώματος και κανέναν του άλλου, τότε αλλάζουμε την κατάστασή της σε 'dangerous-1' και την κατάσταση όλων των αχρωμάτιστων κόμβων της σε 'trouble-1'. Τέλος, αν δεν έχει κανέναν αχρωμάτιστο κόμβο αλλά είναι ακόμα στην αρχική κατάσταση '$\perp$' (πράγμα που συμβαίνει αν όλοι οι κόμβοι της είναι είτε χρωματισμένοι με το ίδιο χρώμα είτε 'trouble-1'), τότε αλλάζουμε την κατάστασή της σε 'unsafe-1'.

Χρόνος εκτέλεσης Αν ο ζητούμενος κόμβος x είναι χρωματισμένος τότε ο χρόνος εκτέλεσης της Φάσης 1 είναι προφανώς $O(1)$. Αν ο κόμβος είναι αχρωμάτιστος τότε πρέπει να ανανεωθούν οι καταστάσεις από το πολύ $d+1$ υπερακμές, αφού κάθε κόμβος ανήκει σε το πολύ $d+1$ υπερακμές, και οι καταστάσεις από το πολύ $k(d+1)$ κόμβους, αφού κάθε υπερακμή περιλαμβάνει ακριβώς k κόμβους. Άρα ο χρόνος εκτέλεσης σε αυτή την περίπτωση είναι $O(kd) = O(1)$, αφού τα k και d θεωρούνται σταθερές. Τέλος, αν ο x είναι σε κατάσταση 'trouble-1' τότε ο χρόνος εκτέλεσης είναι ίσος με της Φάσης 2, ενώ αν είναι σε κατάσταση 'trouble-2' είναι ίσος με της Φάσης 3.

Phase1(x):

1. Αν x χρωματισμένος, επέστρεψε χρώμα(x)
2. Αν x είναι trouble-1, επέστρεψε **Phase2**(x)
3. Αν x είναι trouble-2, επέστρεψε **Phase3**(x)
4. Αλλιώς,
   a) Επέλεξε τυχαία ένα χρώμα c από {red, blue}
   b) Ανανέωσε όλες τις υπερακμές στο E(x)
   c) Επέστρεψε το χρώμα c

Σχήμα 5.2: Φάση 1 του ΑΤΥ για χρωματισμό υπεργράφου

5.3.3 Φάση 2

Στο Σχήμα 5.3 φαίνεται ο ψευδοκώδικας της Φάσης 2. Κατά τη Φάση 1 του αλγορίθμου μας μπορεί ο ζητούμενος κόμβος να έγινε 'trouble-1' αρκετά νωρίς χωρίς να προλάβουν να χρωματιστούν αρκετοί κόμβοι. Θα τρέξουμε, λοιπόν, ένα 'προπαρασκευαστικό' στάδιο στην αρχή της Φάσης 2 κατά το οποίο θα χρωματίσουμε όσους κόμβους από αυτούς που δεν είχαν χρωματιστεί μπορούμε ακόμη εξερευνώντας τον γράφο εξαρτήσεων G. Το στάδιο αυτό στην ουσία αποτελεί μέρος της πρώτης φάσης του αλγορίθμου του Alon. Στο τέλος του σταδίου αυτού θα έχουμε βρει τη συνεκτική συνιστώσα από 'surviving-1' υπερακμές γύρω από τον κόμβο x στον G, έστω $C_1(x)$. Στη συνέχεια, χρωματίζουμε τους κόμβους της $C_1(x)$ με παρόμοιο τρόπο με τη Φάση 1, το οποίο αποτελεί και το κύριο στάδιο της Φάσης αυτής. Στόχος του σταδίου αυτού είναι η συνεκτική συνιστώσα από 'surviving-2' τώρα υπερακμές γύρω από τον x στον G να είναι αρκετά μικρή στο τέλος της Φάσης 2, ώστε στην επόμενη φάση να εκτελέσουμε εξαντλητική αναζήτηση αρκετά γρήγορα. 'Surviving-1' υπερακμές καλούμε τις υπερακμές σε κατάσταση 'dangerous-1' ή 'unsafe-1', ενώ 'surviving-2' αυτές σε κατάσταση 'dangerous-2' ή 'unsafe-2'.

Ξεκινάμε, λοιπόν, εξερευνώντας τον γράφο εξαρτήσεων G. Διατηρούμε ένα σύνολο από υπερακμές $\mathcal{E}_1$, το οποίο θα εξελιχθεί στη συνεκτική συνιστώσα $C_1(x)$, και ένα σύνολο από κόμβους V_1. Το σύνολο V_1 περιέχει κάθε στιγμή

Phase2(x):

1. Βρες τη συνεκτική συνιστώσα C1(x) από surviving-1 υπερακμές γύρω από τον x στον G
2. Αν |C1(x)| > clogN, τότε τερμάτισε με FAIL
3. Επανέλαβε για O(logN/loglogN) φορές ή μέχρι να βρεθεί «καλός» χρωματισμός:
   a) Χρωμάτισε όλους τους κόμβους του C1(x) τυχαία: C1(x) → S1(x)
   b) Εξερεύνησε το $G|_{S1(x)}$
   c) Έλεγξε αν ο χρωματισμός είναι «καλός»
4. Επέστρεψε το χρώμα του x

Σχήμα 5.3: Φάση 2 του ΑΤΥ για χρωματισμό υπεργράφου

όλους τους αχρωμάτιστους κόμβους των υπερακμών στο $\mathcal{E}_1$. Ορίζουμε αρχικά $\mathcal{E}_1 = \mathcal{E}(x)$, καθώς το χρώμα του κόμβου x εξαρτάται άμεσα από τα χρώματα των κόμβων των υπερακμών οι οποίες τον περιέχουν. Στη συνέχεια, χρωματίζουμε τυχαία τους κόμβους του συνόλου V_1. Κάθε φορά που χρωματίζουμε έναν κόμβο ανανεώνουμε όλες τις υπερακμές στις οποίες ανήκει όπως ακριβώς στη Φάση 1. Αν κάποια υπερακμή γίνει 'safe' τότε την αφαιρούμε από το $\mathcal{E}_1$ και τους αχρωμάτιστους κόμβους της που ανήκουν μόνο σε αυτή από το V_1. Συνεχίζουμε μέχρι να χρωματίσουμε όλους τους κόμβους στο V_1. Μετά ελέγχουμε αν υπάρχει υπερακμή που είναι γειτονική στο G με κάποια του $\mathcal{E}_1$ και, αν δεν είναι 'safe', την προσθέτουμε στο $\mathcal{E}_1$ και αντίστοιχα τους αχρωμάτιστους κόμβους της στο V_1, καθώς τα χρώματα των κόμβων των υπερακμών οι οποίες περιέχουν τον x εξαρτώνται άμεσα από τα χρώματα των κόμβων των υπερακμών που τις τέμνουν. Επαναλαμβάνουμε τώρα την προαναφερθείσα διαδικασία χρωματισμού για τους νέους κόμβους του V_1. Η διαδικασία ολοκληρώνεται όταν δεν υπάρχουν πλέον αχρωμάτιστοι κόμβοι στο σύνολο V_1 (όλοι οι κόμβοι των υπερακμών του $\mathcal{E}_1$ είναι είτε χρωματισμένοι είτε σε κατάσταση 'trouble-1') ή αν το μέγεθος του $\mathcal{E}_1$ γίνει μεγαλύτερο από $c_1 \log N$, όπου c_1 μια σταθερά (ώστε να μην παραβιάσουμε το χρονικό όριο εκτέλεσης που θέλουμε ιδανικά να επιτύχουμε). Το σύνολο $\mathcal{E}_1$ είναι εν τέλει η συνεκτική συνιστώσα $C_1(x)$ και σύμφωνα με το ακόλουθο λήμμα, που αποδεικνύεται στο [5], έχει σχεδόν σίγουρα μέγεθος το πολύ $c_1 \log N$. Ολοκληρώνεται έτσι το 'προκατασκευαστικό' αυτό στάδιο της Φάσης 2.

Λήμμα 5. *([5]). Έστω $S \subseteq G$ το σύνολο των 'surviving-1' υπερακμών μετά τη Φάση 1. Τότε, με πιθανότητα τουλάχιστον $1 - \frac{1}{2N}$, όλες οι συνεκτικές*

συνιστώσες του $G|_S$ έχουν η κάθε μία μέγεθος το πολύ $c_1 \log N$.

Συνεχίζουμε τώρα με το κύριο μέρος της Φάσης 2. Δεδομένου ότι δεν υπάρχει ακμή που να συνδέει τη $C_1(x)$ με οποιαδήποτε άλλη συνεκτική συνιστώσα του G, δεν υπάρχει κόμβος ακμής της $C_1(x)$ που το χρώμα του να εξαρτάται από χρώμα κόμβου που δεν ανήκει σε ακμή της $C_1(x)$. Μπορούμε, επομένως, να χρωματίσουμε τους κόμβους των υπερακμών της ανεξάρτητα από όλους τους άλλους κόμβους. Ακολουθούμε διαδικασία παρόμοια με τη Φάση 1 χρωματίζοντας τυχαία τους κόμβους. Και πάλι μόλις χρωματίσουμε έναν κόμβο ανανεώνουμε την κατάσταση όλων των υπερακμών στις οποίες ανήκει. Συγκεκριμένα, αν μια υπερακμή περιλαμβάνει πλέον κόμβους και των δύο χρωμάτων αλλάζουμε την κατάστασή της σε 'safe'. Αν περιλαμβάνει $k_1 + k_2$ τώρα κόμβους του ίδιου χρώματος και κανέναν του άλλου, αλλάζουμε την κατάστασή της σε 'dangerous-2' και την κατάσταση όλων των αχρωμάτιστων κόμβων της σε 'trouble-2'. Τέλος, αν δεν έχει κανέναν αχρωμάτιστο κόμβο και όλοι οι κόμβοι της είναι είτε χρωματισμένοι με το ίδιο χρώμα είτε 'trouble-2', τότε αλλάζουμε την κατάστασή της σε 'unsafe-2'.

Στόχος της παραπάνω διαδικασίας είναι οι συνεκτικές συνιστώσες του G από 'surviving-2' υπερακμές της $C_1(x)$ να είναι αρκετά μικρές μετά το τέλος της, ώστε — αν χρειαστεί — στη Φάση 3 να εκτελέσουμε μια εξαντλητική αναζήτηση στη συνεκτική συνιστώσα γύρω από τον x σχετικά γρήγορα. Έστω $S_1(x)$ το σύνολο από 'surviving-2' υπερακμές της $C_1(x)$. Εφαρμόζοντας το Λήμμα 5 στον $G|_{S_1(x)}$ έχουμε ότι, σχεδόν σίγουρα, κάθε μία από τις συνεκτικές συνιστώσες έχει μέγεθος το πολύ $c_2 \log\log N$, όπου c_2 μια σταθερά. Αν ισχύει αυτό για το χρωματισμό που κάναμε στη $C_1(x)$, τότε λέμε ότι ο χρωματισμός είναι *καλός*. Αν όχι, επαναλαμβάνουμε την παραπάνω διαδικασία. Επαναλαμβάνοντας τη διαδικασία το πολύ $O\left(\frac{\log N}{\log\log N}\right)$ φορές θα πάρουμε, με πιθανότητα τουλάχιστον $1 - \frac{1}{2N^2}$, χρωματισμό τέτοιον ώστε κάθε μία από τις συνεκτικές συνιστώσες του $G|_{S_1(x)}$ να έχει μέγεθος το πολύ $c_2 \log\log N$, δηλαδή ικανοποιητικά μικρό.

Χρόνος εκτέλεσης Με βάση την παραπάνω ανάλυση της Φάσης 2, έχουμε ότι:

Πρόταση 1. *Ο χρόνος εκτέλεσης της Φάσης 2 είναι* $O(\text{polylog}N)$.

5.3.4 Φάση 3

Στη Φάση 3 βρίσκουμε τη συνεκτική συνιστώσα του $G|_{S_1(x)}$ με υπερακμές γύρω από τον κόμβο x, έστω $C_2(x)$, εξερευνώντας τον G με τρόπο παρόμοιο με αυτόν στη Φάση 2. Το Lovász Local Lemma μας εγγυάται την ύπαρξη ενός 2-χρωματισμού του υπεργράφου H σύμφωνα με την Ενότητα 5.3. Επομένως, μπορούμε να χρωματίσουμε τον x ψάχνοντας εξαντλητικά έναν 2-χρωματισμό της $C_2(x)$ ανάμεσα σε όλους τους δυνατούς χρωματισμούς της.

Χρόνος εκτέλεσης Αποδεικνύουμε την εξής πρόταση για το χρόνο εκτέλεσης της Φάσης 3:

Πρόταση 2. *Ο χρόνος εκτέλεσης της Φάσης 3 είναι* $O(\text{polylog}N)$.

Απόδειξη. Αντίστοιχα με τη Φάση 2, βρίσκουμε τη συνεκτική συνιστώσα $C_2(x)$ σε χρόνο $O(\log\log N)$. Επίσης, δεδομένου ότι η $C_2(x)$ έχει μέγεθος το πολύ $c_2 \log\log N$, η εξαντλητική αναζήτηση ενός 2-χρωματισμού ανάμεσα σε όλους τους δυνατούς χρωματισμούς της παίρνει χρόνο το πολύ $2^{O(|C_2(x)|)} = 2^{O(\log\log N)} = \text{polylog}N$. Άρα ο συνολικός χρόνος εκτέλεσης της Φάσης 3 ισούται με $O(\log\log N + \text{polylog}N) = O(\text{polylog}N)$.

□

Έχουμε, τελικά, το εξής αποτέλεσμα:

Θεώρημα 11. *Έστω υπεργράφος H και d, k τέτοια ώστε να υπάρχουν τρεις θετικοί ακέραιοι k_1, k_2 και k_3 τέτοιοι ώστε να ισχύουν τα εξής:*

$$\begin{aligned} k_1 + k_2 + k_3 &= k, \\ 16d(d-1)^3(d+1) &< 2^{k_1}, \\ 16d(d-1)^3(d+1) &< 2^{k_2}, \\ 2e(d+1) &< 2^{k_3}. \end{aligned}$$

Τότε υπάρχει ένας $(O(\text{polylog}N), O(N), 1/N)$-αλγόριθμος τοπικού υπολογισμού ο οποίος, δοθέντος ενός κόμβου u, επιστρέφει το χρώμα του u σε κάποιο 2-χρωματισμό του H. Επίσης, όλες του οι απαντήσεις είναι συνεπείς ως προς κάποιο συγκεκριμένο 2-χρωματισμό του H.

Χρησιμοποιώντας τον Αλγόριθμο Τοπικού Υπολογισμού που παρουσιάσαμε παραπάνω καταφέραμε, επομένως, να πάρουμε την επιθυμητή πληροφορία για το

χρώμα ενός συγκεκριμένου κόμβου σε χρόνο μόλις $O(\text{polylog}N)$. Αντιθέτως, αν περιμέναμε να αποκτήσουμε με τον αλγόριθμο του Alon ολόκληρο το 2-χρωματισμό του υπεργράφου θα χρειαζόμασταν είτε πολυωνυμικό χρόνο είτε αρκετά περισσότερους επεξεργαστές.

Κεφάλαιο 6

Βελτιώνοντας τη Χωρική Πολυπλοκότητα

Όλοι οι παραπάνω Αλγόριθμοι Τοπικού Υπολογισμού τρέχουν μεν σε χρόνο πολυλογαριθμικό, αλλά χρειάζονται γραμμικό χώρο στη μνήμη. Χρησιμοποιώντας τη θεωρία των δέντρων ερωτημάτων (query trees) και των ψευδοτυχαίων αριθμών μπορούμε να βελτιώσουμε τους παραπάνω αλγορίθμους και να τους κάνουμε να τρέχουν τόσο σε πολυλογαριθμικό χρόνο όσο και με πολυλογαριθμικό χώρο.

6.1 Δέντρα ερωτημάτων (Query trees)

Έστω ένας υπεργράφος H ο οποίος είναι k-ομοιόμορφος και κάθε υπερακμή του τέμνει το πολύ d άλλες. Μας ζητείται το χρώμα του κόμβου x. Μπορούμε να μοντελοποιήσουμε τη διαδικασία χρωματισμού του κόμβου x με ένα *query tree*. Ρίζα του query tree είναι ο κόμβος x και οι κόμβοι του πρώτου επιπέδου είναι οι κόμβοι από το χρώμα των οποίων εξαρτάται το χρώμα του x. Γενικά, το χρώμα των κόμβων ενός επιπέδου i εξαρτάται από τα χρώματα των κόμβων του επιπέδου $i+1$. Μας ενδιαφέρει να μάθουμε σε τι τιμές κυμαίνεται το μέγεθος του query tree αυτού.

Το χρώμα ενός κόμβου εξαρτάται από τα χρώματα το πολύ $D = k(d+1)$ άλλων κόμβων, αφού κάθε κόμβος ανήκει σε το πολύ $d+1$ υπερακμές και κάθε υπερακμή περιλαμβάνει ακριβώς k κόμβους (ο γράφος είναι k-ομοιόμορφος). Οι κόμβοι, δηλαδή, του παραπάνω query tree έχουν μέγιστο βαθμό D. Επομένως, για να φράξουμε το μέγεθος του δέντρου αυτού μπορούμε να χρησιμοποιήσου-

με στη θέση του ένα (άπειρο) D-regular δέντρο $\mathcal{T}$ με ρίζα το x. Προφανώς, κάνοντας αυτήν την αλλαγή το μέγεθος του query tree μπορεί μόνο να μεγαλώσει. Επίσης, ο αλγόριθμος του Alon στον οποίον στηρίχθηκε ο ΑΤΥ που περιγράψαμε στο προηγούμενο κεφάλαιο δουλεύει για οποιαδήποτε σειρά χρωματισμού, άρα δουλεύει για τυχαία σειρά. Μπορούμε τότε αντί για το query tree $\mathcal{T}$ να χρησιμοποιήσουμε ένα *random query tree*. Ένα random query tree με ρίζα το x κατασκευάζεται ως εξής:

Σε κάθε κόμβο w του $\mathcal{T}$ αναθέτουμε στην τύχη έναν πραγματικό αριθμό $r(w) \in [0,1]$. Τον τυχαίο αυτό αριθμό ονομάζουμε rank (βαθμό) του w. Αναπτύσσουμε ένα υποδέντρο T του $\mathcal{T}$ με ρίζα το x ως εξής: ένας κόμβος w ανήκει στο υποδέντρο T αν και μόνο αν ο πατέρας του ανήκει στο T, $\text{parent}(w) \in T$, και $r(w) < r(\text{parent}(w))$ (για ευκολία θεωρούμε όλα τα ranks διακριτούς αριθμούς). Αν $|T|$ η τυχαία μεταβλητή που δηλώνει το μέγεθος του random query tree T, έχουμε το εξής θεώρημα:

Θεώρημα 12. *Για οποιονδήποτε μέγιστο βαθμό κόμβου $D \geq 2$, υπάρχει σταθερά $C(D)$ η οποία εξαρτάται μόνο από το D τέτοια ώστε για αρκετά μεγάλο N να ισχύει:*

$$Pr[|T| > C(D)\log^{D+1} N < 1/N^2].$$

Η βασική ιδέα της απόδειξης του παραπάνω θεωρήματος έγκειται στο να χωρίσουμε τους κόμβους του random query tree σε επίπεδα ανάλογα με το rank τους. Αρχικά χωρίζουμε το διάστημα τιμών των ranks $[0,1]$ σε $D+1$ υποδιαστήματα: $I_i := \left(1 - \frac{i}{D+1}, 1 - \frac{i-1}{D+1}\right]$ για $i = 1, 2, \ldots, D$ και $I_{D+1} = \left[0, \frac{1}{D+1}\right]$. Οπότε τώρα σπάμε το query tree σε $D+1$ επίπεδα έτσι ώστε ένας κόμβος u ανήκει στο επίπεδο i αν $r(u) \in I_i$. Έστω ότι η ρίζα x του random query tree T ανήκει στο επίπεδο 1, $r(x) \in I_1$. Τότε οι κόμβοι του T που ανήκουν στο επίπεδο 1 σχηματίζουν ένα δέντρο $T_1 = T_1^{(1)}$ με ρίζα το x. Οι κόμβοι του T στο επίπεδο 2 σχηματίζουν ένα δάσος $\{T_2^{(1)}, \ldots, T_2^{(m_2)}\}$, οι κόμβοι στο επίπεδο 3 ένα δάσος $\{T_3^{(1)}, \ldots, T_3^{(m_3)}\}$, κοκ. Το ακόλουθο λήμμα, που αποδεικνύεται στο [8], φράζει το μέγεθος όλων αυτών των δέντρων:

Λήμμα 6. *Για οποιαδήποτε $1 \leq i \leq D+1$ και $1 \leq j \leq m_i$, με πιθανότητα τουλάχιστον $1 - 1/N^3$, ισχύει $|T_i^{(j)}| = O(\log N)$.*

Απόδειξη του Θεωρήματος 12. Το μέγεθος του δέντρου T_1 από το Λήμμα 6 είναι το πολύ $O(\log N)$ με πιθανότητα τουλάχιστον $1 - 1/N^3$. Η ρίζα οποιουδήποτε δέντρου στο επίπεδο 2 πρέπει να έχει πατέρα στο T_1. Επομένως, ο αριθμός των δέντρων του επιπέδου 2 είναι το πολύ D φορές ο αριθμός των

κόμβων του T_1, δηλαδή $m_2 = D \cdot O(\log N) = O(\log N)$. Εφαρμόζοντας, τώρα, το Λήμμα 6 σε κάθε δέντρο του επιπέδου 2 παίρνουμε ότι ο αριθμός των κόμβων του επιπέδου 2 είναι το πολύ $m_2 \cdot O(\log N) = O(\log^2 N)$. Αντίστοιχα, η ρίζα οποιουδήποτε δέντρου στο επίπεδο 3 πρέπει να έχει πατέρα σε κάποιο από τα επίπεδα 1 ή 2, άρα ο αριθμός των δέντρων του επιπέδου 3 είναι το πολύ $m_3 = D(O(\log N) + O(\log^2 N)) = O(\log^2 N)$ και ο αριθμός των κόμβων του είναι το πολύ $m_3 \cdot O(\log N) = O(\log^3 N)$. Με παρόμοιο τρόπο παίρνουμε ότι $m_i = O(\log^{i-1} N)$ και οι κόμβοι του επιπέδου i είναι το πολύ $O(\log^i N)$, για $i = 2, \ldots, D+1$. Τότε, ο συνολικός αριθμών κόμβων και των $D+1$ επιπέδων είναι το πολύ $O(\log N) + O(\log^2 N) + \cdots + O(\log^{D+1} N) = O(\log^{D+1} N)$, υποθέτοντας ότι το αποτέλεσμα του Λήμματος 6 ισχύει για όλα τα υποδέντρα όλων των επιπέδων. Αυτό συμβαίνει με πιθανότητα τουλάχιστον $1 - O(\log^{D+1} N)/N^3 > 1 - 1/N^2$, από την ανισότητα Boole για ένωση γεγονότων. □

6.2 Βαθύτερη ανάλυση

Στην ενότητα αυτή βελτιώνουμε το άνω φράγμα για το μέγεθος ενός query tree που θεσπίσαμε στην προηγούμενη ενότητα για την περίπτωση που οι κόμβοι έχουν μέγιστο βαθμό d και επεκτείνουμε το νέο φράγμα και στην περίπτωση που οι βαθμοί των κόμβων είναι κατανεμημένοι διωνυμικά, ανεξάρτητα μεταξύ τους, με μέση τιμή d. Έχουμε, λοιπόν, ότι:

Λήμμα 7. *Έστω G γράφος του οποίου οι κόμβοι έχουν μέγιστο βαθμό d ή βαθμούς κατανεμημένους ανεξάρτητα ο καθένας με βάση διωνυμική κατανομή: $deg(v) \sim B(n, d/n)$. Τότε υπάρχει σταθερά $C(d)$, η οποία εξαρτάται μόνο από το d, έτσι ώστε να ισχύει*

$$Pr[|T| > C(d) \log n] < 1/n^2,$$

όπου η πιθανότητα λαμβάνεται για όλες τις δυνατές μεταθέσεις π των κόμβων του G και T είναι ένα random query tree υπό τη μετάθεση π.

Για την περίπτωση κόμβων με μέγιστο βαθμό d ξαναγράφουμε το Λήμμα 6 ως εξής:

Πρόταση 3. *Έστω $L \geq d+1$ ένας ακέραιος και T ένα άπειρο d-regular query tree. Τότε, για οποιαδήποτε $1 \leq i \leq L$ και $1 \leq j \leq m_i$, ισχύει $Pr[|T_i^{(j)}| \geq n] \leq$*

$\sum_{i=n}^{\infty} 2^{-ci} \leq 2^{-\Omega(n)}$ *για όλα τα* $n \geq \beta$*, όπου* β *σταθερά. Συγκεκριμένα, υπάρχει σταθερά* c_0 *η οποία εξαρτάται μόνο από το* d *τέτοια ώστε για κάθε* $n \geq 1$*,*

$$Pr[|T_i^{(j)}| \geq n] \leq e^{-c_0 n}.$$

Για την περίπτωση κόμβων με βαθμούς κατανεμημένους διωνυμικά, ανεξάρτητα μεταξύ τους, με μέση τιμή d αποδεικνύεται η εξής πρόταση στο [47]:

Πρόταση 4. *Έστω* $L \geq d+1$ *ένας ακέραιος και* T *ένα query tree με βαθμούς κατανεμημένους διωνυμικά, ανεξάρτητα μεταξύ τους:* $deg(u) \sim B(n, d/n)$*. Τότε, για οποιαδήποτε* $1 \leq i \leq L$ *και* $1 \leq j \leq m_i$*, ισχύει* $Pr[|T_i^{(j)}| \geq n] \leq \sum_{i=n}^{\infty} 2^{-ci} \leq 2^{-\Omega(n)}$ *για όλα τα* $n \geq \beta$*, όπου* β *σταθερά.*

Το παρακάτω πόρισμα προκύπτει άμεσα από τις Προτάσεις 3 και 4:

Πόρισμα 1. *Έστω* $L \geq d+1$ *ένας ακέραιος και* T *ένα άπειρο d-regular query tree ή ένα query tree με βαθμούς κατανεμημένους διωνυμικά, ανεξάρτητα μεταξύ τους:* $deg(u) \sim B(n, d/n)$*. Τότε, για οποιαδήποτε* $1 \leq i \leq L$ *και* $1 \leq j \leq m_i$*, με πιθανότητα τουλάχιστον* $1 - 1/n^3$*, ισχύει* $|T_i^{(j)}| = O(\log n)$*.*

Η ακόλουθη πρόταση δείχνει ότι καθώς αυξάνουμε τα επίπεδα, το μέγεθος του δέντρου αυξάνεται το πολύ έναν σταθερό παράγοντα για κάθε επίπεδο.

Πρόταση 5. *Έστω* $L \geq d+1$ *ένας ακέραιος και* T *ένα άπειρο d-regular query tree ή ένα query tree με βαθμούς κατανεμημένους διωνυμικά, ανεξάρτητα μεταξύ τους:* $deg(u) \sim B(n, d/n)$*. Για οποιαδήποτε* $1 \leq i \leq L$ *και* $1 \leq j \leq m_i$*, υπάρχουν σταθερές* $\eta_1, \eta_2 > 0$ *τέτοιες ώστε αν* $\sum_{j=1}^{m_i} |T_i^{(j)}| \leq \eta_1 \log n$ *τότε* $Pr[\sum_{j=1}^{m_{i+1}} |T_{i+1}^{(j)}| \geq \eta_1 \eta_2 \log n] < 1/n^2$ *για όλα τα* $n > \beta$*, όπου* β *σταθερά.*

Απόδειξη. Έστω Z_k ο αριθμός των κόμβων του επιπέδου k και $Y_k = \sum_{i=1}^{k} Z_i$. Υποθέτουμε ότι κάθε κόμβος i σε επίπεδο $\leq k$ είναι ρίζα ενός δέντρου μεγέθους z_i στο επίπεδο $k+1$. Τότε έχουμε ότι $Z_{k+1} = \sum_{i=1}^{Y_k} z_i$.

Από την Πρόταση 3, υπάρχουν σταθερές c_0 και β τέτοιες ώστε για κάθε υποδέντρο $T_k^{(i)}$ στο επίπεδο k και κάθε $n > \beta$ έχουμε $Pr[|T_k^{(i)}| = n] \leq e^{-c_0 n}$. Επομένως, η πιθανότητα τα δέντρα του επιπέδου $k+1$ να έχουν μεγέθη ακριβώς $(z_1, \ldots, z_{Y_k})$ είναι το πολύ $\prod_{i=1}^{Y_k} e^{-c_0(z_i - \beta)} = e^{-c_0(Z_{k+1} - \beta Y_k)}$. Το ίδιο αποτέλεσμα έχουμε και με την Πρόταση 4.

Θα φράξουμε την πιθανότητα $Z_{k+1} = \eta Y_k$ για κάποια σταθερά $\eta > 0$ αρκετά μεγάλη. Δοθέντος ενός Y_k υπάρχουν το πολύ $\binom{Z_{k+1}+Y_k-1}{Y_k-1} < \binom{Z_{k+1}+Y_k}{Y_k}$ περιπτώσεις διανυσμάτων $(z_1, \ldots, z_{Y_k})$ που πραγματοποιούν το Z_{k+1}. Τότε:

$$\begin{aligned} Pr[|T_{k+1}| = Z_{k+1}] &< \binom{Z_{k+1}+Y_k}{Y_k} e^{-c_0(Z_{k+1}-\beta Y_k)} \\ &< \left(\frac{e(Z_{k+1}+Y_k)}{Y_k}\right)^{Y_k} e^{-c_0(Z_{k+1}-\beta Y_k)} \\ &= (e(1+\eta))^{Y_k} e^{-c_0(\eta-\beta)Y_k} \\ &= e^{Y_k(-c_0(\eta-b)+\ln(\eta+1)+1)} \\ &\leq e^{-c_0\eta Y_k/2}. \end{aligned}$$

Επομένως, υπάρχει σταθερά c' τέτοια ώστε $Pr[|T_{k+1}| \geq \eta Y_k] \leq e^{-c'\eta Y_k}$. Συγκεκριμένα, αν $\eta Y_k = \Omega(\log n)$, τότε $|T_{k+1}| \geq \eta Y_k$ με πιθανότητα το πολύ $1/n^3$. □

Μπορούμε τώρα να αποδείξουμε το Λήμμα 7.

Απόδειξη του Λήμματος 7. Από το Πόρισμα 1 έχουμε ότι το μέγεθος οποιουδήποτε υποδέντρου, και συγκεκριμένα του T_1, είναι το πολύ $O(\log n)$ με πιθανότητα τουλάχιστον $1 - 1/n^3$. Από την προηγούμενη πρόταση έχουμε ότι υπάρχει σταθερά η τέτοια ώστε αν ο αριθμός των κόμβων στο επίπεδο k είναι το πολύ $|T_k|$, τότε για τον αριθμό των κόμβων του επιπέδου $k+1$ έχουμε $|T_{k+1}| \leq \eta \sum_{i=1}^{k} |T_i| + O(\log n) \leq 2\eta|T_k| + O(\log n)$. Από την ανισότητα του Boole και για τα L επίπεδα, έχουμε ότι ο συνολικός αριθμός κόμβων του query tree είναι το πολύ $O((2\eta)^L \log n) = O(\log n)$, με πιθανότητα τουλάχιστον $1 - 1/n^2$. □

6.3 k-wise independent random orderings

Έστω $m \geq 1$ ένας ακέραιος και $\mathcal{R}$ ένα ολικώς διατεταγμένο σύνολο. Ένα *ordering* του $[m]$ είναι μια '1-1' συνάρτηση $r : [m] \to \mathcal{R}$. Έστω $\mathbf{r} = \{r_i\}_{i \in I}$ μια οικογένεια από orderings με δείκτες στο I. Τότε, *random ordering* $D_{\mathbf{r}}$ του $[m]$ είναι μια κατανομή πάνω στην οικογένεια από orderings $\mathbf{r}$. Λέμε ότι ένα random ordering $D_{\mathbf{r}}$ είναι *k-wise independent* για $2 \leq k \leq m$ αν, για οποιοδήποτε υποσύνολο $S \subset [m]$ μεγέθους το πολύ k, η κατανομή που προκύπτει αν περιορίσουμε το $D_{\mathbf{r}}$ στο S είναι ένα uniform permutation πάνω στο

S. Επίσης, λέμε ότι το $D_{\mathbf{r}}$ είναι *ϵ-almost k-wise independent* αν η στατιστική απόσταση του $D_{\mathbf{r}}$ από ένα k-wise independent random ordering είναι το πολύ ϵ. Το παρακάτω θεώρημα μας εγγυάται την αποδοτική κατασκευή $1/m^2$-almost k-wise independent random orderings.

Θεώρημα 13. *Έστω ακέραιοι $m \geq 2$ και $2 \leq k \leq m$. Τότε υπάρχει κατασκευή ενός $\frac{1}{m^2}$-almost k-wise independent random ordering πάνω στο $[m]$ του οποίου το seed length είναι $O(k \log^2 m)$.*

Απόδειξη. Θεωρούμε για ευκολία ότι το m είναι δύναμη του 2 και $s = 4 \log m$. Έστω s ανεξάρτητες k-wise independent τυχαίες μεταβλητές στο $\{0,1\}^m$, $Z_1, \ldots, Z_s$, τις οποίες γράφουμε ως εξής:

$$\begin{aligned} Z_1 &= z_{1,1}, \ldots, z_{1,m}, \\ Z_2 &= z_{2,1}, \ldots, z_{2,m}, \\ &\ldots\ldots \\ Z_s &= z_{s,1}, \ldots, z_{s,m}. \end{aligned}$$

Για ordering του $[m]$ χρησιμοποιούμε τη συνάρτηση $r : [m] \to \{0, 1, \ldots, 2^s - 1\}$ όπου $r(i)$ είναι ο αριθμός που αναπαριστά η στήλη i παραπάνω στο δυαδικό σύστημα, δηλαδή $r(i) := z_{1,i} z_{2,i} \ldots z_{s,i}$. Η συνάρτηση αυτή θα πρέπει γενικά να είναι '1-1'.

Έστω i και j δύο διακριτοί δείκτες, $1 \leq i < j \leq m$. Δεδομένου ότι τα $z_{l,1}, \ldots, z_{l,m}$ είναι k-wise independent, έχουμε ότι $Pr[z_{l,i} = z_{l,j}] = 1/2$. Επίσης, δεδομένου ότι οι $Z_1, \ldots, Z_s$ είναι ανεξάρτητες, έχουμε:

$$\begin{aligned} Pr[r(i) = r(j)] &= Pr[z_{l,i} = z_{l,j} \text{ για όλα τα } 1 \leq l \leq s] \\ &= \prod_{l=1}^{s} Pr[z_{l,i} = z_{l,j}] = (1/2)^s = 1/m^4. \end{aligned}$$

Από την ανισότητα του Boole για όλα τα $\binom{m}{2}$ ζευγάρια δεικτών παίρνουμε ότι οι αριθμοί $r(1), \ldots, r(m)$ είναι όλοι διακριτοί με πιθανότητα τουλάχιστον $1 - 1/m^2$. Άρα η συνάρτηση r είναι '1-1' με την ίδια πιθανότητα.

Δεδομένου ότι τα $Z_1, \ldots, Z_s$ είναι k-wise independent τυχαίες μεταβλητές στο $\{0,1\}^m$, τότε, για κάθε σύνολο k δεικτών $\{i_1, \ldots, i_k\}$, τα $r(i_1), \ldots, r(i_k)$ είναι ανεξάρτητα μεταξύ τους. Με την προϋπόθεση ότι είναι και διακριτά, το ordering που προκύπτει αν περιορίσουμε τη συνάρτηση r στο σύνολο $\{i_1, \ldots, i_k\}$ είναι ένα independent ordering. Άρα το ordering που παράγει η r είναι ένα $\frac{1}{m^2}$-almost k-wise independent random ordering, αφού η συνάρτηση r δεν είναι

'1-1' με πιθανότητα το πολύ $1/m^2$. Επιπλέον, με βάση το Θεώρημα 6, το seed length κάθε μιας από τις τυχαίες μεταβλητές $Z_1, \ldots, Z_s$ είναι $O(k \log m)$, άρα χρειαζόμαστε συνολικά χώρο $O(k \log^2 m)$ και για τις s. □

6.4 2-χρωματισμός υπεργράφου

Εφαρμόζουμε τώρα τις τεχνικές που περιγράψαμε στις Ενότητες 6.1 έως 6.3 στον ΑΤΥ για 2-χρωματισμό υπεργράφου που δώσαμε στο προηγούμενο κεφάλαιο.

6.4.1 Βελτιώσεις

Ξεκινάμε κατασκευάζοντας ένα random query tree με ρίζα τον ζητούμενο κόμβο x και κόμβους αυτούς από τους οποίους εξαρτάται το χρώμα του x και χρωματίζουμε το δέντρο αυτό από κάτω προς τα πάνω. Το random query tree T αυτό κατασκευάζεται όπως δείξαμε στην Υποενότητα 6.1. Από το Θεώρημα 12, με πιθανότητα τουλάχιστον $1 - 1/m^2$, ο συνολικός αριθμός κόμβων του T είναι το πολύ polylog$m$, δηλαδή polylog$N$. Επομένως, χρωματίζοντας το δέντρο T από κάτω προς τα πάνω θα χρειαστεί, σχεδόν σίγουρα, να χρωματίσουμε το πολύ polylog$N$ κόμβους πριν χρωματίσουμε τη ρίζα x. Οπότε και ο χρόνος εκτέλεσης της φάσης 1 του νέου ΑΤΥ είναι, με πιθανότητα τουλάχιστον $1 - 1/m^2$, το πολύ $O(\text{polylog}N)$.

Επιπλέον, κατά το σχηματισμό του random query tree T μόνο η σχετική διάταξη των κόμβων παίζει ρόλο. Συγκεκριμένα, ελέγχουμε μόνο αν $r(x) < r(y)$ ενώ οι ακριβείς τιμές των $r(x)$ και $r(y)$ δεν μας είναι χρήσιμες. Μπορούμε, επομένως, να αντικαταστήσουμε τη γεννήτρια τυχαίων αριθμών του αλγορίθμου με μια ισοδύναμη συνάρτηση random ordering $r \in S_m$ (όπου S_m το symmetric group των m στοιχείων). Επιπλέον, δεδομένου ότι το μέγεθος του query tree είναι σχεδόν σίγουρα πολυλογαριθμικό, το random ordering δε χρειάζεται να δουλεύει και για τους m κόμβους του υπεργράφου ταυτόχρονα αλλά αρκεί να είναι ένα polylog$N$-wise independent random ordering. Από το Θεώρημα 13 μπορούμε να κατασκευάσουμε ένα $\frac{1}{m^2}$-almost k-wise independent random ordering για τους κόμβους του υπεργράφου με $k = \text{polylog}N$ χρησιμοποιώντας χώρο $O(k \log^2 m) = O(\text{polylog}N)$.

Εξίσου με τα ranks, δε χρειάζεται να κρατάμε στη μνήμη τα χρώματα όλων των κόμβων του υπεργράφου, αλλά μόνο για τους κόμβους που χρειαζόμαστε. Η απόδειξη του βασικού λήμματος του αλγορίθμου του Alon (βλ. [9]) ισχύει

ακόμα και αν ο τυχαίος χρωματισμός είναι $c \log N$-wise independent, όπου c σταθερά. Μπορούμε, δηλαδή, να αντικαταστήσουμε τους πραγματικά τυχαίους αριθμούς στο $\{0,1\}^m$ που χρησιμοποιούμε για χρώματα με $c \log N$-wise independent τυχαίους αριθμούς στο $\{0,1\}^m$. Από το Θεώρημα 6 χρειαζόμαστε $O(\log^2 N)$ χώρο για να αποθηκεύσουμε τον νέο ψευδοτυχαίο χρωματισμό.

6.4.2 Ο νέος ΑΤΥ

Χρησιμοποιούμε το νέο φράγμα για το μέγεθος των query trees για να βελτιώσουμε τον ΑΤΥ για 2-χρωματισμό υπεργράφου. Στην ακόλουθη ανάλυση θεωρούμε ότι:

$$k \geq 3\lceil \log 16d(d-1)^3(d+1) \rceil + \lceil \log 2e(d+1) \rceil$$
$$k_i = k_{i-1} - \lceil \log 16d(d-1)^3(d+1) \rceil.$$

Η γενική διαδικασία

Σε κάθε φάση i ξεκινάμε με τα υποσύνολα E_i και V_i των E και V αντίστοιχα, όπου κάθε υπερακμή περιέχει τουλάχιστον k_i κόμβους. Το V_i κάθε στιγμή περιέχει όλους τους αχρωμάτιστους κόμβους των υπερακμών στο E_i. Η παρακάτω διαδικασία αποτελεί στην ουσία γενίκευση της Φάσης 1 του αλγορίθμου του Alon.

Αρχικά, αναθέτουμε στην τύχη χρώματα στους κόμβους του V_i. Κάθε φορά που χρωματίζουμε έναν κόμβο ανανεώνουμε την κατάσταση όλων των υπερακμών που τον εμπεριέχουν. Αν μια υπερακμή περιλαμβάνει k_{i+1} κόμβους του ίδιου χρώματος και κανέναν του άλλου, τότε αλλάζουμε την κατάστασή της σε '*dangerous*' και την κατάσταση όλων των αχρωμάτιστων κόμβων της σε '*saved*'. Συνεχίζουμε μέχρις ότου όλοι οι κόμβοι στο V_i να είναι είτε χρωματισμένοι είτε 'saved'.

Ως '*survived*' καλούμε όλες τις υπερακμές που δεν περιέχουν κόμβους και των δύο χρωμάτων. Έστω S_i το σύνολο των '*survived*' υπερακμών μετά τη Φάση i. Η πιθανότητα το $G|_{S_i}$ να περιέχει συνεκτική συνιστώσα μεγέθους d^3u είναι το πολύ $|V_i|2^{-u}$ με βάση το [5]. Ειδικότερα, αν επαναλάβουμε τον τυχαίο χρωματισμό των κόμβων t_i φορές, δεν θα υπάρχει συνεκτική συνιστώσα στο $G|_{S_i}$ μεγέθους μεγαλύτερου του d^3u με πιθανότητα

$$(|V_i|2^{-u})^{t_i}. \tag{6.1}$$

Αν κατά τη Φάση i ο κόμβος x χρωματιστεί τότε επιστρέφουμε το χρώμα αυτό ως αποτέλεσμα. Αλλιώς, εξερευνούμε τον $G|_{S_i}$ ώστε να βρούμε τη συνεκτική συνιστώσα $C_i(x)$ με υπερακμές γύρω από τον x. Δεδομένου ότι η $C_i(x)$ δε συνδέεται με καμία άλλη συνεκτική συνιστώσα του G, μπορούμε να χρωματίσουμε τους κόμβους των υπερακμών της ανεξάρτητα από όλους τους άλλους κόμβους. Τρέχουμε λοιπόν τη Φάση $i+1$ με είσοδο $E_{i+1} = C_i(x)$.

Φάση 1

Στο Σχήμα 6.1 φαίνεται ο ψευδοκώδικας της Φάσης 1 του αλγορίθμου. Στη Φάση 1 ξεκινάμε έχοντας ως είσοδο ολόκληρο τον υπεργράφο, δηλαδή $E_1 = E$ και $V_1 = V$. Κατασκευάζουμε ένα random query tree με ρίζα το x και χρωματίζουμε τους κόμβους του από κάτω προς τα πάνω, όπως στις προηγούμενες ενότητες. Αν ο x χρωματιστεί επιστρέφουμε το χρώμα του.

Φάση 1
Είσοδος: ένας κόμβος x
Έξοδος: ένα χρώμα {red ή blue}

1. Βρες ένα query tree T με ρίζα στο x χρησιμοποιώντας BFS, με βάση το ordering π
2. Χρωμάτισε τυχαία τους κόμβους του T σύμφωνα με τη σειρά που ορίζει το π
3. Αν ο x είναι χρωματισμένος, επέστρεψε το χρώμα του
4. Αλλιώς:
   a. Ξεκινώντας από το E(x) τρέξε BFS στο $G|_{S_1}$ ώστε να βρεις τη συνεκτική συνιστώσα $E_2=C_1(x)$ από "survived" υπερακμές γύρω από τον x
   b. Έστω V_2 το σύνολο από αχρωμάτιστους κόμβους στην E_2.
      Τρέξε **Φάση 2** (x, E_2, V_2)

Σχήμα 6.1: Φάση 1 του νέου ΑΤΥ για χρωματισμό υπεργράφου

Αν ο x γίνει 'saved' τότε εξερευνούμε τον $G|_{S_1}$ για να βρούμε τη συνεκτική συνιστώσα $C_1(x)$. Κάθε φορά που συναντάμε έναν νέο κόμβο κατασκευάζουμε το query tree του και χρωματίζουμε τους κόμβους του από κάτω προς τα πάνω, όπως ακριβώς κάναμε και για τον x. Για το μέγεθος της $C_1(x)$ έχουμε ότι $Pr[|C_1(x)| > 4d^3 \log n] < n2^{-4\log n} = n^{-3}$. Δεδομένου ότι σχεδόν σίγουρα υπάρχουν το πολύ $O(\log n)$ ακμές στη $C_1(x)$ (άρα και $O(\log n)$ κόμβοι, αφού ο υπεργράφος είναι k-ομοιόμορφος με k σταθερά) και κάθε query tree έχει μέγεθος το πολύ $O(\log n)$, θα χρωματίσουμε το πολύ $O(\log^2 n)$ κόμβους.

Αφού μας ενδιαφέρει να χειριζόμαστε το πολύ $O(\log^2 n)$ κόμβους κάθε στιγμή, μπορούμε να χρησιμοποιήσουμε έναν ψευδοτυχαίο χρωματισμό που είναι

$O(\log^2 n)$-wise independent και μια ψευδοτυχαία διάταξη που είναι n^{-3}-almost $O(\log^2 n)$-wise independent. Με βάση τα Θεωρήματα 6 και 13 τα ανωτέρω μπορούν να κατασκευαστούν σε χώρο και χρόνο $O(\log^4 n)$.

Φάση 2 και 3

Οι Φάσεις 2 και 3 είναι απλά επαναλήψεις τις γενικής διαδικασίας που περιγράψαμε πιο πάνω. Στόχος τους είναι να μειώσουν όσο το δυνατόν περισσότερο το μέγεθος της συνεκτικής συνιστώσας από εναπομείναντες ακμές γύρω από το ζητούμενο κόμβο x, ώστε τελικά να εφαρμόσουμε πάνω της εξαντλητική αναζήτηση για χρωματισμό. Στο Σχήμα 6.2 φαίνεται ο ψευδοκώδικας των φάσεων αυτών.

Φάση i (x, E_i, V_i) $i \in 2,3$
Είσοδος: ένας κόμβος $x \in V_i$ και υποσύνολα $E_i \subseteq E$ και $V_i \subseteq V$
Έξοδος: ένα χρώμα {red ή blue} ή FAIL

1. Επανέλαβε τα επόμενα logn φορές και σταμάτα αν βρεθεί καλός χρωματισμός
   a. Σειριακά προσπάθησε να χρωματίσεις κάθε κόμβο του V_i ομοιόμορφα στην τύχη
   b. Εξερεύνησε το γράφο εξαρτήσεων του $G|_{Si}$
   c. Έλεγξε αν ο χρωματισμός είναι καλός
2. Αν ο x είναι χρωματισμένος, επέστρεψε το χρώμα του
3. Αλλιώς:
   a. Υπολόγισε τη συνεκτική συνιστώσα $C_i(x)=E_{i+1}$ καθώς και το V_{i+1}
   b. Τρέξε **Φάση i+1** (x, E_{i+1}, V_{i+1})

Σχήμα 6.2: Φάσεις 2-3 του νέου ΑΤΥ για χρωματισμό υπεργράφου

Συγκεκριμένα, έχουμε με μεγάλη πιθανότητα ότι $|E_2| = |C_1(x)| \leq 4d^3 \log n$ και κάθε ακμή έχει k_2 αχρωμάτιστους κόμβους. Από τη σχέση 6.1, μετά από $t_2 = \log n$ επαναλήψεις του τυχαίου χρωματισμού, η συνεκτική συνιστώσα από 'survived' υπερακμές έχει μέγεθος $2d^3 \log \log n$ με πιθανότητα

$$((4d^3 k_2 \log n) 2^{-2 \log \log n})^{\log n} < n^{-3}.$$

Ομοίως, στη Φάση 3 έχουμε με μεγάλη πιθανότητα ότι $|E_3| < 2d^3 \log \log n$ και κάθε ακμή έχει k_3 αχρωμάτιστους κόμβους. Από τη σχέση 6.1, μετά από $t_3 = \log n$ επαναλήψεις του τυχαίου χρωματισμού, η συνεκτική συνιστώσα από 'survived' υπερακμές έχει μέγεθος $\frac{\log \log n}{k_4}$ με πιθανότητα

$$\left((2d^3 k_3 \log \log n) 2^{-\frac{\log \log n}{d^3 k_4}} \right)^{\log n} < n^{-3}.$$

Φάση 4

Στο Σχήμα 6.3 φαίνεται ο ψευδοκώδικας της Φάσης 4. Συγκεκριμένα, έχουμε με μεγάλη πιθανότητα ότι $|E_4| < \frac{\log\log n}{k_4}$ και κάθε ακμή έχει k_4 αχρωμάτιστους κόμβους. Έχουμε ρίξει πλέον αρκετά το μέγεθος της συνεκτικής συνιστώσας E_4 και το Lovász Local Lemma μας εγγυάται την ύπαρξη ενός 2-χρωματισμού του υπεργράφου. Επομένως, μπορούμε να χρωματίσουμε τον x ψάχνοντας εξαντλητικά ένα 2-χρωματισμό της E_4 ανάμεσα σε όλους τους δυνατούς χρωματισμούς της σε χρόνο $O(\log n)$.

Φάση 4 (x, E_4, V_4)
Είσοδος: ένας κόμβος $x \in V_4$ και υποσύνολα $E_4 \subseteq E$ και $V_4 \subseteq V$
Έξοδος: ένα χρώμα {red ή blue}

1. Δοκίμασε όλους τους δυνατούς χρωματισμούς της συνεκτικής συνιστώσας V_4 και χρωμάτισέ της χρησιμοποιώντας έναν δυνατό χρωματισμό
2. Επέστρεψε το χρώμα του x στο χρωματισμό αυτό

Σχήμα 6.3: Φάση 4 του νέου ΑΤΥ για χρωματισμό υπεργράφου

Έχουμε, τελικά, το εξής αποτέλεσμα για τον βελτιωμένο ΑΤΥ για 2-χρωματισμό υπεργράφου:

Θεώρημα 14. *Έστω k-ομοιόμορφος υπεργράφος H τέτοιος ώστε κάθε υπερακμή να τέμνει το πολύ d άλλες υπερακμές. Έστω $k \geq 16\log d + 19$. Τότε υπάρχει ένας $(O(\log^4 N), O(\log^4 N), 1/N)$-αλγόριθμος τοπικού υπολογισμού ο οποίος, δοθέντος ενός κόμβου u, επιστρέφει το χρώμα του u σε κάποιο 2-χρωματισμό του H. Επίσης, όλες του οι απαντήσεις είναι συνεπείς ως προς κάποιο συγκεκριμένο 2-χρωματισμό του H.*

6.5 Maximal Independent Set

Για τη Φάση 1, ο ΑΤΥ χρειάζεται να διατηρεί στη μνήμη τις τυχαίες επιλογές κάθε κόμβου, ώστε όλες οι απαντήσεις του να είναι συνεπείς ως προς κάποια δεκτή λύση. Δηλαδή, η απόφαση που θα πάρει για έναν κόμβο σχετικά με το αν είναι μέσα στο MIS ή όχι θα πρέπει να ισχύει για όλες τις επόμενες απαντήσεις που θα δώσει. Μια σημαντική παρατήρηση είναι ότι όλοι οι υπολογισμοί είναι 'τοπικοί' και, επομένως, μπορούμε να αντικαταστήσουμε τα πραγματικά τυχαία

bits με τυχαία bits περιορισμένης ανεξαρτησίας, μειώνοντας έτσι το χώρο που χρειάζεται ο αλγόριθμος.

Αρχικά θέτουμε ως νέο μέγιστο βαθμό του G το $\tilde{d} = 2^{\lceil \log d \rceil}$ ($d \leq \tilde{d} < 2d$). Την πιθανότητα $1/2\tilde{d}$ που χρησιμοποιεί ο αλγόριθμος την παράγουμε τώρα ρίχνοντας $\log \tilde{d} = \lceil \log d \rceil$ ανεξάρτητα δίκαια κέρματα. Έχουμε έτσι ότι κάθε ενδεχόμενο B_u εξαρτάται από το πολύ $d^{O(\log d)} \cdot \log \tilde{d} = d^{O(\log d)}$ τυχαία bits. Επιπλέον, η απόδειξη του Θεωρήματος 8 ισχύει ακόμα και αν τα ενδεχόμενα $\{B_u\}_{u \in H^3}$ είναι $c \log n$-wise independent, όπου c σταθερά. Αυτό συμβαίνει όταν τα τυχαία bits που χρησιμοποιεί ο ΑΤΥ είναι k-wise independent με $k = d^{O(\log d)} \cdot c \log n = d^{O(\log d)} \log n$. Δεδομένου ότι ο αριθμός των τυχαίων bits που χρησιμοποιεί η Φάση 1 είναι $m = d^{O(\log d)} \cdot n$, χρειαζόμαστε εν τέλει μια k-wise independent τυχαία μεταβλητή στο $\{0,1\}^m$. Από το Θεώρημα 6, για την τυχαία αυτή μεταβλητή χρειαζόμαστε χώρο $O(k \log m) = d^{O(\log d)} \log^2 n$ και κάθε τυχαίο bit μπορεί να υπολογισθεί σε χρόνο $O(k \log m) = d^{O(\log d)} \log^2 n$.

Έχουμε, τελικά, το εξής αποτέλεσμα για τον νέο ΑΤΥ για MIS:

Θεώρημα 15. *Έστω γράφος G με n κόμβους και μέγιστο βαθμό d. Τότε υπάρχει ένας $(O(d^{O(d \log d)} \cdot \log^3 n), O(d^{O(d \log d)} \cdot \log^2 n), 1/n)$-αλγόριθμος τοπικού υπολογισμού ο οποίος, δοθέντος ενός κόμβου u, αποφασίζει αν ο u ανήκει σε κάποιο maximal independent set. Επίσης, όλες του οι απαντήσεις είναι συνεπείς ως προς κάποιο συγκεκριμένο MIS.*

Κεφάλαιο 7

Εφαρμογές στο Σχεδιασμό Μηχανισμών

Στο κεφάλαιο αυτό θα παρουσιάσουμε τη λογική πίσω από έναν Αλγόριθμο Τοπικού Υπολογισμού για το πρόβλημα του Maximal Matching και θα χρησιμοποιήσουμε τον αλγόριθμο αυτό για να αποδείξουμε την ύπαρξη ΑΤΥ για ένα συγκεκριμένο είδος δημοπρασιών της περιοχής του Σχεδιασμού Μηχανισμών.

7.1 Μοντέλο

Ορισμός 13. Ένας μηχανισμός $\mathcal{M} = (\mathcal{A}, \mathcal{P})$ είναι $(t(n), s(n), \delta(n))$-τοπικός αν τόσο η συνάρτηση κατανομής $\mathcal{A}$ όσο και η συνάρτηση πληρωμών $\mathcal{P}$ υπολογίζονται από $(t(n), s(n), \delta(n))$-αλγορίθμους τοπικού υπολογισμού.

Ένας *φιλαλήθης τοπικός μηχανισμός* $\mathcal{M} = (\mathcal{A}, \mathcal{P})$ είναι ένας τοπικός μηχανισμός ο οποίος είναι παράλληλα φιλαλήθης. Δηλαδή, η επικρατούσα στρατηγική για κάθε πράκτορα είναι να δηλώσει την πραγματική του αξία, άσχετα με το αν ο μηχανισμός είναι τοπικός.

7.2 Συνδυαστικές δημοπρασίες

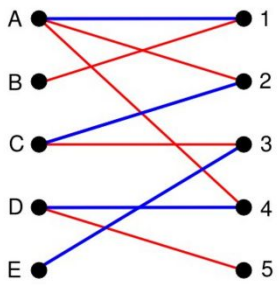
Συνδυαστικές δημοπρασίες (combinatorial auctions) είναι δημοπρασίες στις οποίες οι αγοραστές μπορούν να κάνουν bid σε σύνολα αντικειμένων. Έστω ότι m αντικείμενα βγαίνουν σε δημοπρασία μεταξύ n αγοραστών. Δίνουμε καθολικά φιλαλήθεις τοπικούς μηχανισμούς για τις εξής δύο περιπτώσεις:

1. Όλοι οι αγοραστές έχουν την ίδια αξία για κάθε αντικείμενο στο σύνολό τους και η ιδιωτική πληροφορία κάθε αγοραστή είναι το σύνολο των αντικειμένων για τα οποία ενδιαφέρεται.

2. Το σύνολα των αγοραστών είναι κοινή γνώση και η ιδιωτική πληροφορία κάθε αγοραστή είναι η αξία του για τα αντικείμενα που τον ενδιαφέρουν, η οποία θεωρούμε ότι είναι ίδια για όλα τα αντικείμενα αυτά.

7.3 Maximal Matching

Έστω γράφος $G = (V, E)$. Ένα *ταίριασμα* (matching) είναι ένα υποσύνολο ακμών του E τέτοιο ώστε κανένα ζευγάρι ακμών να μην έχει κοινό κόμβο. Ένα ταίριασμα ονομάζεται *maximal* αν καμιά άλλη ακμή δεν μπορεί να προστεθεί σε αυτό χωρίς να παραβιάσει την προαναφερθείσα ιδιότητα. Ένα παράδειγμα ενός ταιριάσματος φαίνεται στο παρακάτω σχήμα:



Σχήμα 7.1: Παράδειγμα ενός ταιριάσματος

Στην online εκδοχή του προβλήματος οι ακμές φτάνουν σε μια άγνωστη σειρά. Για την εύρεση ενός maximal matching μπορεί να χρησιμοποιηθεί ο ακόλουθος άπληστος αλγόριθμος. Έστω ότι φτάνει η ακμή e. Αν η e δεν είναι ήδη στο ταίριασμα, τότε ελέγχουμε όλες τις γειτονικές τις ακμές για να δούμε αν είναι κάποια από αυτές στο ταίριασμα. Αν καμιά τους δεν είναι, τότε προσθέτουμε την e στο ταίριασμα. Αντιθέτως, στη local computation εκδοχή

του προβλήματος δίνουμε στον ΑΤΥ μια ακμή $e \in E$ ως είσοδο και του ζητάμε να μας απαντήσει αν η ακμή αυτή ανήκει σε κάποιο maximal matching ή όχι.

Ο αλγόριθμος, λοιπόν, όταν του δίνεται ως είσοδος μια ακμή e φτιάχνει ένα DAG με ρίζα την e χρησιμοποιώντας BFS στις ακμές. Υποθέτουμε ότι ο G έχει μέγιστο βαθμό κόμβου d. Από το Λήμμα 6, το μέγεθος του DAG είναι το πολύ $O(\log n)$ (καθώς το query tree αποτελεί άνω φράγμα για το DAG). Μπορούμε, επομένως, να κατασκευάσουμε ένα $O(\log n)$-wise independent random ordering για τις ακμές με seed length $O(\log^3 n)$ (Θεώρημα 13), αφού ανά πάσα στιγμή θα χειριζόμαστε το πολύ $O(\log n)$ κόμβους. Για να αποφασίσει τελικά ο ΑΤΥ αν η e ανήκει σε κάποιο maximal matching τρέχει τον άπληστο αλγόριθμο στις ακμές του DAG με τη σειρά που ορίζει το παραπάνω ordering. Παίρνουμε έτσι το εξής αποτέλεσμα:

Θεώρημα 16. *Έστω γράφος $G = (V, E)$ με n κόμβους και μέγιστο βαθμό d. Τότε υπάρχει ένας $(O(d^{O(d \log d)} \cdot \log n), O(n), 1/n)$-αλγόριθμος τοπικού υπολογισμού ο οποίος, δοθέντος μιας ακμής e, αποφασίζει αν η e ανήκει σε κάποιο maximal matching. Επίσης, όλες του οι απαντήσεις είναι συνεπείς ως προς κάποιο συγκεκριμένο maximal matching.*

7.4 Unit-demand αγοραστές

Στην ενότητα αυτή θα ασχοληθούμε με την περίπτωση *unit-demand* αγοραστών: κάθε αγοραστής ενδιαφέρεται για το πολύ k αντικείμενα και για κάθε αντικείμενο ενδιαφέρεται το πολύ ένας πολυλογαριθμικός αριθμός από αγοραστές. Στην ενότητα 2.5 είδαμε ότι το να υπολογίσουμε τη βέλτιστη λύση στην περίπτωση αυτή είναι αρκετά δύσκολο. Εδώ θα χρησιμοποιήσουμε το μοντέλο των Αλγορίθμων Τοπικού Υπολογισμού για να πάρουμε μια καλή προσέγγιση της βέλτιστης λύσης αρκετά γρήγορα — συγκεκριμένα σε χρόνο μόλις $O(\log^4 n)$.

7.4.1 Unit-demand αγοραστές με uniform value

Έστω $\mathcal{I}$ ένα σύνολο από n unit-demand αγοραστές και $\mathcal{J}$ ένα σύνολο από m αντικείμενα. Κάθε αγοραστής ενδιαφέρεται για ένα σύνολο J_i από το πολύ k αντικείμενα. Θεωρούμε τα n, m και k σταθερές. Μπορούμε να αναπαραστήσουμε αυτή τη δημοπρασία ως έναν γράφο $G = (V, E)$ όπου $V = \mathcal{J} \cup \mathcal{I}$ και $E = \{(i, j) : i \in \mathcal{I}, j \in J_i\}$. Η αξία ενός αγοραστή είναι η ίδια για όλα τα αντικείμενα, έστω 1. Η αξία του αγοραστή i για το υποσύνολο S είναι $v_i(S) = 1$ αν

$S \cap J_i \neq \emptyset$ και 0 αλλιώς. Δηλαδή, οι αγοραστές δεν κάνουν διακρίσεις μεταξύ των αντικειμένων που θέλουν. Θεωρούμε ότι το utility ενός αγοραστή i είναι quasi-linear, δηλαδή όταν αποκτά τα αντικείμενα του συνόλου S και πληρώνει p τότε το utility του είναι $u_i(S, p) = v_i(S) - p$. Υποθέτουμε ότι υποσύνολα J_i επιλέγονται ομοιόμορφα στην τύχη.

Στόχος του τοπικού μηχανισμού που θα παρουσιάσουμε είναι να μεγιστοποιεί το κοινωνικό καλό. Για να το κάνει αυτό, θα πρέπει να ικανοποιήσει όσο το δυνατόν περισσότερους αγοραστές κατανέμοντάς τους ένα μόνο αντικείμενο από το σύνολό τους, εφόσον για αυτούς δεν έχει διαφορά αν θα αποκτήσουν ένα ή περισσότερα αντικείμενα (όλα τα σύνολα με τουλάχιστον ένα αντικείμενο που επιθυμούν έχουν την ίδια αξία). Αυτό στο γράφο G ανάγεται στο να βρεθεί ένα maximum matching μεταξύ αγοραστών και αντικειμένων. Αυτό το είδος δημοπρασίας ονομάζεται k-UDUV (unit-demand, uniform value).

Ωστόσο, όπως αποδεικνύεται ακολούθως, δεν είναι δυνατό να λυθεί τοπικά το πρόβλημα του maximum matching.

Θεώρημα 17. *([27]). Δεν υπάρχει ΑΤΥ για το πρόβλημα του maximum matching.*

Απόδειξη. Έστω σύνολο από ομομορφικούς γράφους $\mathcal{G} = \{G_i\}$, καθένας από τους οποίους έχει $2n$ κόμβους $\{v_1, \ldots, v_{2n}\}$. Σε κάθε γράφο G_i οι κόμβοι v_{-i}[1] σχηματίζουν έναν κύκλο (περιττού μήκους) και ο κόμβος v_i συνδέεται με τον κόμβο $v_{(i-1) \bmod 2n}$. Μας δίνεται ένας τυχαίος γράφος $G \in \mathcal{G}$ και η ακμή $e = (v_1, v_2)$ και θα θέλαμε να μάθουμε αν η e ανήκει στο maximum matching. Παρατηρούμε ότι η ακμή e ανήκει στα μισά ακριβώς maximum matchings.

Ας υποθέσουμε ότι ο γράφος είναι ο G_n ή ο G_{n+1}. Τότε, στο κατανεμημένο μοντέλο, η απόσταση μεταξύ της ακμής e και του διαχωριστικού σημείου των G_n και G_{n+1} είναι n ακμές. Δηλαδή χρειάζεται χρόνος τουλάχιστον $\Omega(n)$ για να διακρίνουμε σωστά μεταξύ των δύο γράφων.

Στο δικό μας μοντέλο τοπικού υπολογισμού γράφουμε τις ακμές σε μια τυχαία σειρά. Τότε, ο ΑΤΥ χρειάζεται να απαντήσει σε n κατά μέσο όρο ερωτήματα προτού καταφέρει να διακρίνει σωστά μεταξύ του G_n και του G_{n+1}. Επομένως, δεν μπορεί να υπάρξει ΑΤΥ που να απαντάει στο πρόβλημα του maximum matching. □

Πόρισμα 2. *Δεν υπάρχει ΑΤΥ για το πρόβλημα του maximum matching σε διμερείς γράφους.*

[1] Με v_{-i} συμβολίζουμε όλους τους κόμβους του γράφου εκτός του κόμβου i, δηλαδή $\{v_1, \ldots, v_{i-1}, v_{i+1}, \ldots, v_{2n}\}$.

Απόδειξη. Έστω σύνολο από ομομορφικούς γράφους $\mathcal{G} = \{G_i\}$, καθένας από τους οποίους έχει $2n$ κόμβους $\{v_1, \ldots, v_{2n}\}$. Σε κάθε γράφο G_i οι κόμβοι $\{v_1, \ldots, v_{i-1}, v_{i+2}, \ldots, v_{2n}\}$ σχηματίζουν έναν κύκλο (άρτιου μήκους). Ο κόμβος v_i συνδέεται με τον κόμβο $v_{(i-1) \bmod 2n}$ και ο $v_{(i+1) \bmod 2n}$ συνδέεται με τον $v_{(i+2) \bmod 2n}$. Παρατηρούμε ότι οι ακμές (v_{i-1}, v_i) και (v_{i+1}, v_{i+2}) είναι πάντα μέσα στο maximum matching, το μέγεθος του οποίο είναι n. Η απόδειξη είναι ακριβώς όπως στη γενική περίπτωση. □

Αφού δεν υπάρχει ΑΤΥ για τον υπολογισμό maximum matching θα αρκεστούμε στην εύρεση ενός *approximation* (προσέγγισης) του maximum matching. Έστω $\mathcal{A}_{UDUV}$ ο αλγόριθμος για maximal matching που παρουσιάσαμε στην ενότητα 7.3. Θα χρησιμοποιήσουμε τον αλγόριθμο αυτό για να πάρουμε ένα 1/2-approximation του maximum matching.

Έστω ο μηχανισμός $\mathcal{M}_{UDUV} = (\mathcal{A}_{UDUV}, \mathcal{P}_{UDUV})$ ο οποίος λαμβάνει ως είσοδο ένα υποσύνολο $J'_i \subset \mathcal{J}$ από κάθε αγοραστή. Ο αλγόριθμος κατανομής $\mathcal{A}_{UDUV}$ εξετάζει τα αντικείμενα ένα τη φορά με μια συγκεκριμένα σειρά. Η σειρά αυτή προκύπτει ως εξής: ο μηχανισμός αναθέτει σε κάθε αντικείμενο j ανεξάρτητα και ομοιόμορφα έναν πραγματικό αριθμό $r_j \sim_u [0, 1]$. Τα αντικείμενα εξετάζονται σε φθίνουσα σειρά του r_j. Όταν εξετάζεται ένα αντικείμενο j, αν υπάρχει κάποιος αγοραστής i ο οποίος ενδιαφέρεται για το αντικείμενο αυτό, δηλαδή $j \in J'_i$, και δεν έχει του έχει δοθεί ακόμη κάποιο αντικείμενο, τότε το αντικείμενο j δίνεται στον i. Αν υπάρχουν πάνω από ένας τέτοιοι αγοραστές τότε η ισοπαλία λύνεται με κάποιο συγκεκριμένο τρόπο, πχ λεξικογραφικά.

Επειδή οι αριθμοί r_j ανατίθενται ανεξάρτητα, οι αγοραστές δεν μπορούν να κάνουν τίποτα για να αλλάξουν τη σειρά με την οποία θα εξεταστούν τα αντικείμενα ώστε να αλλοιώσουν την έκβαση του μηχανισμού προς όφελός τους.

Δεδομένου ότι οι αξίες όλων των αντικειμένων είναι ίδιες, οποιοδήποτε σχήμα πληρωμών $\mathcal{P}_{UDUV}$ δουλεύει, αρκεί να ικανοποιεί την ιδιότητα της *individual rationality*[2]. Άρα, η πληρωμή μπορεί να είναι οποιαδήποτε τιμή στο διάστημα $[0, 1]$. Έστω, λοιπόν, $p = 1/2$ για όποιον αγοραστή αποκτά κάποιο αντικείμενο και $p = 0$ για όποιον δεν αποκτά.

Θεώρημα 18. *Στην περίπτωση της δημοπρασίας k-UDUV, ο μηχανισμός $\mathcal{M}_{UDUV} = (\mathcal{A}_{UDUV}, \mathcal{P}_{UDUV})$ είναι καθολικά φιλαλήθης και δίνει ένα $\frac{1}{2}$-approximation της βέλτιστης κατανομής.*

[2]Η ιδιότητα της *individual rationality* ή *voluntary participation* εγγυάται ότι κανένας παίκτης δεν πρόκειται να βγει ζημιωμένος από το να συμμετέχει στο παίγνιο. Δηλαδή το utility οποιουδήποτε παίκτη είναι σε κάθε έκβαση μη αρνητικό.

Απόδειξη. Για να δείξουμε ότι ο μηχανισμός $\mathcal{M}_{UDUV}$ είναι φιλαλήθης αρκεί να δείξουμε ότι κανένας αγοραστής δεν πρόκειται να επωφεληθεί με το να δηλώσει $J_i' \neq J_i$. Το πετυχαίνουμε αυτό σε δύο βήματα. Πρώτα θα δείξουμε ότι για οποιοδήποτε J_i', το να δηλώσεις $J_i \cap J_i'$ κυριαρχεί αδύναμα[3] του να δηλώσεις J_i'. Μετά θα δείξουμε ότι το να δηλώσεις J_i κυριαρχεί αδύναμα του να δηλώσεις οποιοδήποτε $J_i^* \subseteq J_i$.

Από τα σύνολα $J_i \cap J_i'$ και J_i' ο αγοραστής ενδιαφέρεται μόνο για τα αντικείμενα που ανήκουν στο $J_i \cap J_i'$ και καθόλου για αυτά στο $J_i' \setminus J_i$. Αν ένα αντικείμενο στο $J_i \cap J_i'$ δοθεί στον παίκτη i δηλώνοντας J_i', θα δοθεί επίσης σε αυτόν και αν δηλώσει $J_i \cap J_i'$. Άρα η αξία του παίκτη i δεν μπορεί να μειωθεί δηλώνοντας $J_i \cap J_i'$ αντί για J_i' και, συνεπώς, το $J_i \cap J_i'$ κυριαρχεί αδύναμα του J_i'.

Θα δείξουμε τώρα ότι το να δηλώσεις J_i κυριαρχεί αδύναμα του να δηλώσεις οποιοδήποτε $J_i^* \subseteq J_i$. Αν ο αγοραστής i δεν κερδίσει κανένα αντικείμενο δηλώνοντας J_i^* τότε προφανώς η παραπάνω πρόταση ισχύει. Έστω ότι ο i αποκτά το αντικείμενο j δηλώνοντας J_i^*. Τότε, δηλώνοντας J_i, αν δεν έχει αποκτήσει κανένα αντικείμενο από το σύνολο $J_i \setminus J_i^*$ πριν ο αλγόριθμος εξετάσει το αντικείμενο j, τότε θα αποκτήσει το j. Επομένως, αν αποκτήσει κάποιο αντικείμενο δηλώνοντας J_i^* θα το αποκτήσει επίσης κάποιο αντικείμενο και αν δηλώσει J_i. Θα έχει δηλαδή την ίδια αξία και *utility*. Συνεπώς, το J_i κυριαρχεί αδύναμα του $J_i^* \subseteq J_i$.

Έστω ένας αγοραστής που δεν αποκτά κάποιο αντικείμενο στην περίπτωση του $\mathcal{A}_{UDUV}$, ενώ αποκτά κάποιο στη βέλτιστη κατανομή. Το αντικείμενο αυτό κατανέμεται σε κάποιον διαφορετικό αγοραστή στο $\mathcal{A}_{UDUV}$. Άρα, ο αριθμός των αγοραστών που αποκτούν κάποιο αντικείμενο στη βέλτιστη κατανομή και όχι στο $\mathcal{A}_{UDUV}$ φράζεται από τον αριθμό των αγοραστών που αποκτούν αντικείμενο στο $\mathcal{A}_{UDUV}$. Συνεπώς, το $\mathcal{A}_{UDUV}$ αποτελεί $\frac{1}{2}$-approximation της βέλτιστης λύσης. □

Σύμφωνα με το Θεώρημα 18, η συνάρτηση κατανομής $\mathcal{A}_{UDUV}$ αποτελεί από έναν $(O(d^{O(d \log d)} \cdot \log n), O(n), 1/n)$-ΑΤΥ για το πρόβλημα του maximal matching. Επομένως, προκύπτει από την παραπάνω ανάλυση το ακόλουθο θεώρημα:

Θεώρημα 19. *Η δημοπρασία k-UDUV έχει έναν $(O(\log^4 n), O(\log^3 n), 1/n)$-μηχανισμό τοπικού υπολογισμού ο οποίος είναι καθολικά φιλαλήθης και παρέχει*

[3]Με τον όρο η στρατηγική Α *κυριαρχεί αδύναμα* της Β εννοούμε ότι με το να ακολουθήσεις την Α δεν πρόκειται να επωφεληθείς λιγότερο από το αν ακολουθούσες τη Β.

ένα $\frac{1}{2}$-approximation του βέλτιστου κοινωνικού καλού.

7.4.2 Unit-demand αγοραστές, uniform-buyer-value

Έστω $\mathcal{I}$ ένα σύνολο από n unit-demand αγοραστές και $\mathcal{J}$ ένα σύνολο από m αντικείμενα. Κάθε αγοραστής ενδιαφέρεται για ένα σύνολο J_i από το πολύ k αντικείμενα, το οποίο αποτελεί κοινή γνώση. Θεωρούμε τα n, m και k σταθερές. Η αξία ενός αγοραστή i, έστω t_i, είναι η ίδια για όλα τα αντικείμενα και γνωστή μόνο σε αυτόν. Μπορούμε να αναπαραστήσουμε αυτή τη δημοπρασία ως έναν γράφο με βάρη $G = (V, E)$ όπου $V = \mathcal{J} \cup \mathcal{I}$ και $E = \cup_i E_i$ με $E_i = \{(i, j) : j \in J_i\}$. Κάθε ακμή $e \in E_i$ έχει βάρος $w(e) = t_i$. Η αξία του i για το υποσύνολο S είναι $v_i(S) = t_i$ αν $S \cap J_i \neq \emptyset$ και 0 αλλιώς. Θεωρούμε ότι το utility ενός αγοραστή i είναι quasi-linear, δηλαδή όταν αποκτά τα αντικείμενα του συνόλου S και πληρώνει p τότε το utility του είναι $u_i(S, p) = v_i(S) - p$. Υποθέτουμε ότι οι αξίες ακολουθούν μια τυχαία κατανομή, η οποία είναι ίδια για όλους τους αγοραστές, και τα υποσύνολα J_i ακολουθούν ομοιόμορφη ή διωνυμική κατανομή.

Στόχος του τοπικού μηχανισμού που θα παρουσιάσουμε είναι να μεγιστοποιεί το κοινωνικό καλό. Για να το κάνει αυτό, θα πρέπει να ικανοποιήσει όσο το δυνατόν περισσότερους αγοραστές κατανέμοντάς τους ένα μόνο αντικείμενο από το σύνολό τους, εφόσον για αυτούς δεν έχει διαφορά αν θα αποκτήσουν ένα ή περισσότερα αντικείμενα (όλα τα σύνολα με τουλάχιστον ένα αντικείμενο που επιθυμούν έχουν την ίδια αξία t_i). Αυτό το είδος δημοπρασίας ονομάζεται k-UDUBV (unit-demand, uniform buyer value).

Ορίζουμε πρώτα τον αλγόριθμο κατανομής $\mathcal{A}_{UDUBV}$. Αρχικά, ο $\mathcal{A}_{UDUBV}$ ταξινομεί τους αγοραστές σε φθίνουσα σειρά ως προς την αξία τους. Εξετάζοντας τους αγοραστές με αυτή τη σειρά, σε έναν αγοραστή i ανατίθεται ένα αντικείμενο j αν υπάρχει κάποιο $j \in J_i$ που δεν έχει ανατεθεί ακόμα. Αν υπάρχουν περισσότερα του ενός τέτοια αντικείμενα η ισοπαλία λύνεται με κάποιο συγκεκριμένο τρόπο, πχ λεξικογραφικά. Αν δεν υπάρχει κανένα, τότε ο i δεν αποκτά κανένα αντικείμενο. Ο $\mathcal{A}_{UDUBV}$ συνεχίζει έως ότου κανένα αντικείμενο να μην μπορεί να ανατεθεί κάπου.

Πρόταση 6. *Ο αλγόριθμος κατανομής $\mathcal{A}_{UDUBV}$ παρέχει ένα $\frac{1}{2}$-approximation της βέλτιστης κατανομής, με βάση τις δηλωθέντες αξίες.*

Απόδειξη. Παριστάνουμε τη δημοπρασία με έναν διμερή γράφο $G = (U, W, E)$, όπου το σύνολο U αντιπροσωπεύει τους αγοραστές και το W τα αντικείμενα. Μια ακμή μεταξύ ενός αγοραστή i και ενός αντικειμένου j υπάρχει αν και μόνο

αν $j \in J_i$. Το βάρος της ακμής αυτής είναι ίσο με την αξία που δήλωσε ο αγοραστής αυτός, έστω b_i. Στην ουσία, η βέλτιστη κατανομή βρίσκει ένα maximum matching μεταξύ αγοραστών και αντικειμένων, ενώ ο $\mathcal{A}_{UDUBV}$ βρίσκει ένα maximal matching αφού εξετάζει τους αγοραστές στη σειρά με βάση το b_i του καθενός.

Αν μια ακμή $e = (i, j)$ βρίσκεται στο ταίριασμα του $\mathcal{A}_{UDUBV}$ αλλά όχι στο βέλτιστο ταίριασμα, τότε παίρνει τη θέση των πολύ δύο ακμών του βέλτιστου ταιριάσματος (μιας ακμής e' που αγγίζει τον i και μιας e'' που αγγίζει το j). Δεδομένου ότι ο $\mathcal{A}_{UDUBV}$ εξετάζει τις ακμές σε φθίνουσα σειρά ως προς το βάρος τους έχουμε ότι $w(e) \geq w(e')$ και $w(e) \geq w(e'')$. Επομένως, $2w(e) \geq w(e') + w(e'')$ και ο $\mathcal{A}_{UDUBV}$ παρέχει ένα $\frac{1}{2}$-approximation της βέλτιστης κατανομής. □

Στη συνέχεια ορίζουμε το σχήμα πληρωμών $\mathcal{P}_{UDUBV}$. Αν ένας αγοραστής i δεν αποκτήσει κάποιο αντικείμενο, τότε δεν πληρώνει τίποτα. Αν αποκτήσει ένα αντικείμενο, τότε πληρώνει p_i το οποίο υπολογίζεται ως εξής: Τρέχουμε αρχικά τον $\mathcal{A}_{UDUBV}$ χωρίς τον αγοραστή i. Ο i τελικά πληρώνει την μικρότερη τιμή από αυτές στις οποίες πουλήθηκαν τα αντικείμενα που τον ενδιαφέρουν. Για να δείξουμε ότι ο μηχανισμός είναι φιλαλήθης αρκεί να δείξουμε ότι το να δηλώσουν οι αγοραστές ως b_i τις πραγματικές τους αξίες t_i αποτελεί την επικρατούσα στρατηγική για αυτούς, δηλαδή $u_i(t_i, b_{-i}) \geq u_i(b_i, b_{-i})$ για οποιαδήποτε b_i, b_{-i}.

Πρόταση 7. *Στο μηχανισμό $\mathcal{M}_{UDUBV} = (\mathcal{A}_{UDUBV}, \mathcal{P}_{UDUBV})$, για οποιονδήποτε αγοραστή i και δηλωθείσα αξία b_i, το να δηλώσει t_i επικρατεί αδύναμα του να δηλώσει b_i.*

Απόδειξη. Θέτουμε b_{-i} τις δηλωθείσες αξίες όλων των άλλων αγοραστών εκτός του i.

Έστω ότι ο i σκέφτεται να δηλώσει $b_i > t_i$. Τότε αν ο i κέρδιζε ήδη ένα αντικείμενο δηλώνοντας t_i, τότε δηλώνοντας περισσότερα δεν πρόκειται να αλλάξει κάτι καθώς κάθε αγοραστής αποκτά το πολύ ένα αντικείμενο και οι πληρωμές είναι ανεξάρτητες της δηλωθείσας αξίας. Αν ο i δεν κέρδιζε κάποιο αντικείμενο πριν τότε θα ισχύει $p_i \geq b_i = t_i$, όπου p_i η πληρωμή που προκύπτει αν τρέξουμε τον μηχανισμό χωρίς αυτόν. Αν δηλώνοντας περισσότερα αποκτήσει τελικά κάποιο αντικείμενο, τότε θα πληρώσει τουλάχιστον t_i και το utility του θα είναι μη-θετικό.

Έστω ότι ο i σκέφτεται να δηλώσει $b_i < t_i$. Τότε αν ο i δεν κέρδιζε κάποιο αντικείμενο δηλώνοντας t_i δεν πρόκειται να κερδίσει ούτε τώρα, καθώς ο $\mathcal{A}_{UDUBV}$ εξετάζει πρώτα τους αγοραστές με μεγαλύτερα b_i. Αν ο i κέρδιζε

ήδη ένα αντικείμενο πριν, τότε δηλώνοντας λιγότερα είτε πληρώνει και πάλι p_i όσο συνεχίζει να κερδίζει κάποιο αντικείμενο είτε δεν κερδίζει τελικά κανένα αντικείμενο και έχει συνεπώς μηδενικό utility. □

Από τις προτάσεις 6 και 7 έχουμε το εξής:

Θεώρημα 20. *Ο μηχανισμός $\mathcal{M}_{UDUBV}$ είναι καθολικά φιλαλήθης και παρέχει ένα $\frac{1}{2}$-approximation του βέλτιστου κοινωνικού καλού.*

Σύμφωνα με το Θεώρημα 18, η συνάρτηση κατανομής $\mathcal{A}_{UDUV}$ αποτελεί από έναν $(O(\log^4 n), O(\log^3 n), 1/n)$-ΑΤΥ για το πρόβλημα του maximal matching. Ωστόσο, πρέπει να τρέξουμε τον $\mathcal{A}_{UDUV}$ μια φορά για να βρούμε την κατανομή και άλλες k φορές για να υπολογίσουμε τις πληρωμές. Από την παραπάνω ανάλυση προκύπτει το ακόλουθο θεώρημα:

Θεώρημα 21. *Η δημοπρασία k-UDUBV έχει έναν $(O(\log^4 n), O(\log^3 n), 1/n)$-μηχανισμό τοπικού υπολογισμού ο οποίος είναι καθολικά φιλαλήθης και παρέχει ένα $\frac{1}{2}$-approximation του βέλτιστου κοινωνικού καλού.*

Κεφάλαιο 8

Εφαρμογές στο Probabilistic Inference

Στο κεφάλαιο αυτό παρουσιάζουμε συνοπτικά το βασικό πρόβλημα που προσπαθεί να αντιμετωπίσει η περιοχή του Probabilistic Inference και ορίζουμε ένα συγκεκριμένο είδος γραμμικών προγραμμάτων τα οποία συναντώνται συχνά στη συγκεκριμένη περιοχή. Δείχνουμε, στη συνέχεια, την ύπαρξη ΑΤΥ για την προσεγγιστική επίλυση των προγραμμάτων αυτών και το πώς αυτή βρίσκει εφαρμογή στο αρχικό πρόβλημα της περιοχής.

8.1 Μοντέλο

Το πρόβλημά μας εδώ είναι να προβλέψουμε την τιμή μιας δυαδικής μεταβλητής $Z = \{0, 1\}$. Για το σκοπό αυτό έχουμε στη διάθεσή μας n σήματα $S_1, \ldots, S_n$ συσχετισμένα με τη Z τα οποία παίρνουν και αυτά τιμές στο $\{0, 1\}$. Η μεταβλητή Z και τα n σήματα ακολουθούν μια κοινή κατανομή D την οποία θεωρούμε γνωστή. Συγκεκριμένα, θεωρούμε ότι γνωρίζουμε την κατανομή της Z: $Pr[Z = 0] = p$ και $Pr[Z = 1] = 1 - p$.

Στο μοντέλο υπάρχει επίσης ένας αντίπαλος, ο οποίος έχει στη διάθεσή του ένα σύνολο από m ντετερμινιστικούς κανόνες τροποποίησης $\Psi = \{\rho_i : \{0, 1\}^n \times \{0, 1\} \to \{0, 1\}^n\}$. Για τους κανόνες αυτούς πρέπει να ισχύουν τα εξής: α) κάθε κανόνας ρ_i πρέπει να μπορεί να υπολογίζεται σε πολυωνυμικό χρόνο και β) ο αντίστροφος ρ_i^{-1} πρέπει επίσης να μπορεί να υπολογίζεται σε πολυωνυμικό χρόνο. Ο αντίπαλος μπορεί να εφαρμόσει οποιουσδήποτε κανόνες του Ψ στα αρχικά σήματα για να τα τροποποιήσει, βασισμένος ακόμη και στην

πραγματική τιμή της Z. Στόχος του αντιπάλου είναι να μας κάνει να προβλέψουμε ανεπιτυχώς την τιμή της Z.

Ο δικός μας στόχος είναι, δοθέντος των παρατηρημένων σημάτων $x \in \{0,1\}^n$ (δηλαδή των σημάτων μετά την τροποποίησή τους από τον αντίπαλο), να προβλέψουμε την τιμή της μεταβλητής $Z \in \{0,1\}$. Μια πολιτική π είναι μια στοχαστική απεικόνιση από τα παρατηρημένα σήματα $x \in \{0,1\}^n$ σε μια πρόβλεψη $\{0,1\}$. Θέλουμε με άλλα λόγια να βρούμε την πολιτική που προβλέπει καλύτερα τη μεταβλητή Z παρά τις κακόβουλες ενέργειες του αντιπάλου.

Έστω $q_\pi(x) = Pr[\pi(x) = 0]$ η *οριακή συνάρτηση κατανομής* της π. Το αναμενόμενο σφάλμα της πολιτικής π ως προς τον κανόνα τροποποίησης ρ_i είναι:

$$\begin{aligned} error(\pi, \rho_i) &= E_Z E_x[Pr[\pi(x) \neq Z]] \\ &= E_Z E_x[q_\pi(x) I(Z = 1) + (1 - q_\pi(x)) I(Z = 0)]. \end{aligned}$$

Το αναμενόμενο σφάλμα της πολιτικής π ως προς ένα σύνολο κανόνων τροποποίησης Ψ είναι:

$$error(\pi) = \max_{\rho_i \in \Psi} error(\pi, \rho_i).$$

Μια πολιτική π^* είναι βέλτιστη ως προς ένα σύνολο κανόνων τροποποίησης Ψ αν:

$$\pi^* = \arg \min_{\pi} \max_{\rho_i \in \Psi} error(\pi, \rho_i)$$

και το βέλτιστο σφάλμα τότε είναι:

$$error^* = error(\pi^*) = \min_{\pi} \max_{\rho_i \in \Psi} error(\pi, \rho_i).$$

Παρομοίως, μια πολιτική $\overline{\pi}$ είναι ϵ-βέλτιστη, για οποιοδήποτε $\epsilon > 0$, ως προς ένα σύνολο κανόνων τροποποίησης Ψ αν:

$$error(\overline{\pi}) \leq error^* + \epsilon = \min_{\pi} \max_{\rho_i \in \Psi} error(\pi, \rho_i) + \epsilon.$$

8.2 Η βέλτιστη πολιτική

Ορίζουμε τη βέλτιστη πολιτική π^* εξετάζοντας την οριακή συνάρτηση κατανομής της $q^* = q_{\pi^*}$. Τονίζουμε ότι μια οριακή συνάρτηση κατανομής ορίζει μοναδικά μια πολιτική. Θα γράψουμε ένα γραμμικό πρόγραμμα που θα ελαχιστοποιεί το σφάλμα ως προς το χειρότερο κανόνα τροποποίησης και το οποίο θα έχει ως λύση την q^*.

Έστω $A(x, \beta, i)$ η πιθανότητα να έχουμε $Z = \beta$ δεδομένου ότι ο αντίπαλος εφάρμοσε τον κανόνα $\rho_i \in \Psi$ προκειμένου να προκύψει το παρατηρημένο σήμα $x \in \{0,1\}^n$. Τότε:

$$\begin{aligned} A(x, \beta, i) &= Pr[Z = \beta \wedge x = \rho_i(S, Z)] \\ &= \sum_{y \in \rho_i^{-1}(x)} D(\beta; y). \end{aligned}$$

Η πιθανότητα ενός σφάλματος δεδομένης μιας πολιτική π με οριακές πιθανότητες q όταν ο αντίπαλος εφαρμόζει τον κανόνα ρ_i είναι:

$$error(q, \rho_i) = \sum_{x \in \{0,1\}^n} [A(x, 1, i)q(x) + A(x, 0, i)(1 - q(x))].$$

Στόχος μας είναι να βρούμε μια πολιτική π ή ισοδύναμα τις οριακές πιθανότητες q, οι οποίες θα ελαχιστοποιούν το σφάλμα δεδομένου ότι ο αντίπαλος εφάρμοσε τον κανόνα ρ_i. Αυτό είναι ισοδύναμο με το να ελαχιστοποιήσουμε, ως προς όλες τις συναρτήσεις $q : \{0,1\}^n \to [0,1]$, την τιμή $\max_{\rho_i}\{error(q, \rho_i)\}$. Το πετυχαίνουμε αυτό με το εξής γραμμικό πρόγραμμα:

$$\begin{aligned} &\min_{q,Er} Er \\ &\forall i \in [1, m] \qquad \sum_{x \in 0,1^n} error(q, \rho_i) \leq Er \\ &Er \geq 0 \\ &\forall x \in \{0,1\}^n \qquad q(x) \geq 0, q(x) \leq 1. \end{aligned}$$

Η μεταβλητή Er ελαχιστοποιεί το σφάλμα της χειρότερης περίπτωσης. Το πρώτο σετ από ανισότητες, μία για κάθε κανόνα ρ_i του αντιπάλου, φράζει το σφάλμα της χειρότερης περίπτωσης. Το τελευταίο σετ από ανισότητες, μία για κάθε $x \in \{0,1\}^n$, εγγυάται ότι η πρόβλεψη θα είναι μέσα στο $[0,1]$.

Παρατηρώντας ότι $\sum_x A(x, 0, i) = Pr[Z = 0] = p$ παίρνουμε το εξής ΓΠ, το οποίο ονομάζουμε *Primal-Πρόβλεψη*:

$$\begin{aligned} &\min_{q,Er} Er \\ &\forall i \in [1, m] \qquad p \leq Er + \sum_{x \in 0,1^n} q(x)(A(x, 0, i) - A(x, 1, i)) \\ &Er \geq 0 \\ &\forall x \in \{0,1\}^n \qquad q(x) \geq 0, q(x) \leq 1. \end{aligned}$$

Το παραπάνω ΓΠ έχει εκθετικό αριθμό από μεταβλητές (ένα $q(x)$ για κάθε $x \in \{0,1\}$) και εκθετικό αριθμό από ανισότητες (μια $q(x) \leq 1$ για κάθε $x \in \{0,1\}$).

Φτιάχνουμε τώρα το δυαδικό πρόγραμμα. Το ΓΠ αυτό έχει μεταβλητές y_i, οι οποίες προκύπτουν από τους περιορισμούς του σφάλματος για κάθε κανόνα ρ_i, και r_x, οι οποίες προκύπτουν από τους περιορισμούς $q(x) \leq 1$. Έχουμε, λοιπόν, το εξής δυαδικό ΓΠ:

$$\begin{aligned}
\max_{y,r_x} \quad & p\sum_{i=1}^{m} y_i - \sum_{x\in\{0,1\}^n} r_x \\
\forall x \quad & \sum_{i=1}^{m} y_i(A(x,0,i) - A(x,1,i)) \leq r_x \\
& \sum_{i=1}^{m} y_i \leq 1 \\
\forall i \quad & y_i \geq 0 \\
\forall x \quad & r_x \geq 0.
\end{aligned}$$

Φαίνεται εύκολα ότι η τιμή του δυαδικού ΓΠ επιτυγχάνεται για $\sum_i y_i = 1$. Σκεφτόμαστε το y_i ως την πιθανότητα ο αντίπαλος να χρησιμοποιήσει τον κανόνα ρ_i. Επίσης θέτουμε:

$$R_x(y) = \max\left\{0, \sum_{i=1}^{m} y_i(A(x,0,i) - A(x,1,i))\right\}.$$

Παίρνουμε έτσι το εξής ΓΠ, το οποίο ονομάζουμε *Dual-Πρόβλεψη*:

$$\begin{aligned}
\max_{y,r_x} \quad & p - \sum_{x\in\{0,1\}^n} R_x(y) \\
& \sum_{i=1}^{m} y_i = 1 \\
\forall i \quad & y_i \geq 0.
\end{aligned}$$

8.3 Αλγόριθμος Τοπικού Υπολογισμού

Στην ενότητα αυτή παρουσιάζουμε ένα συγκεκριμένο είδος γραμμικών προβλημάτων τα οποία ονομάζουμε *nice block angular (NBA)* και για τα οποία

αποδεικνύουμε ότι υπάρχει αλγόριθμος τοπικού υπολογισμού ο οποίος τα λύνει προσεγγιστικά.

8.3.1 Nice Block Angular Γραμμικά Προγράμματα

Ένα γραμμικό πρόβλημα λέγεται *block angular* αν ο πίνακας περιορισμών του μπορεί να διαχωριστεί σε ένα μικρό αριθμό από πυκνές γραμμές, τις οποίες ονομάζουμε περιορισμούς *πυρήνα*, και σε έναν μεγάλο αριθμό από μπλοκ ασύνδετα από γραμμές και στήλες, τα οποία ονομάζουμε *πέταλα*. Ο πίνακας περιορισμών δηλαδή έχει την εξής μορφή:

$$A = \begin{pmatrix} A_1^k & \cdots & A_r^k \\ A_1^p & & 0 \\ & \ddots & \\ 0 & & A_r^p \end{pmatrix}$$

Για παράδειγμα, το ΓΠ Primal-Πρόβλεψη έχει m περιορισμούς πυρήνα που είναι κοινοί για όλες τις μεταβλητές και 2^n πέταλα τα οποία περιέχουν μία μεταβλητή το καθένα (δηλαδή τα μπλοκ είναι 1×1).

Η παραπάνω block angular δομή συνεπάγεται επίσης ότι το δυαδικό ΓΠ έχει ένα μικρό πυρήνα από μεταβλητές που εμφανίζονται σε όλους τους περιορισμούς και ένα μεγάλο αριθμό από πέταλα με μεταβλητές που εμφανίζονται σε μπλοκ με λίγους περιορισμούς το καθένα. Με άλλα λόγια, δοθέντος μιας ανάθεσης των μεταβλητών πυρήνα, η βέλτιστη ανάθεση για τις μεταβλητές κάθε πετάλου μπορεί να υπολογιστεί ανεξάρτητα. Για παράδειγμα, στο ΓΠ Dual-Πρόβλεψη της Ενότητας 8.2, η τιμή των μεταβλητών των πετάλων r_x δίνεται ως συνάρτηση $R_x(y)$ των μεταβλητών πυρήνα y.

Θεωρούμε ότι οι δυαδικές μεταβλητές πυρήνα ανήκουν σε ένα φραγμένο πεδίο ορισμού και ότι οι συντελεστές του ΓΠ είναι φραγμένοι. Απαιτούμε, επίσης, οι δυαδικές μεταβλητές των πετάλων να ακολουθούν μια κατανομή η οποία να δειγματοληπτείται αποδοτικά, έτσι ώστε η βέλτιστη τιμή οποιασδήποτε μεταβλητής να μην είναι πολύ μεγαλύτερη από την πιθανότητά της.

Λέμε ότι ένα γραμμικό πρόγραμμα είναι *nice block angular (NBA)* αν ικανοποιεί όλες τις 'καλές' ιδιότητες που περιγράψαμε παραπάνω.

8.3.2 Κύριο αποτέλεσμα

Είναι γενικά πολύ εύκολο ένας ΑΤΥ να παράγει απαντήσεις οι οποίες να είναι τοπικά συνεπείς ως προς μια ϵ-βέλτιστη πολιτική, αφού κάθε τιμή $\hat{q}(x)$ έχει

εκθετικά μικρή συνεισφορά στο συνολικό σφάλμα (ακόμα και το να απαντά συνέχεια 0 είναι τοπικά συνεπές). Ωστόσο, δεν μπορούμε να έχουμε καμιά εγγύηση σχετικά με την πιθανότητα του σφάλματος μιας πολιτικής η οποία εφαρμόζει έναν ϵ-προσεγγιστικό ΑΤΥ ο οποίος δεν είναι query-oblivious. Μας ενδιαφέρει, λοιπόν, ένας ΑΤΥ ο οποίος να είναι query-oblivious , έτσι ώστε οι απαντήσεις που δίνει να είναι συνεπείς ως προς μια γενική λύση $\hat{q}$ ανεξάρτητα από το ερώτημα x.

Μπορούμε να διατυπώσουμε το παρακάτω γενικό θεώρημα το οποίο αποδεικνύεται στο [46]:

Θεώρημα 22. *Για οποιοδήποτε $\epsilon > 0$, το πρόβλημα εύρεσης μιας ϵ-additive-προσεγγιστικής λύσης σε ένα NBA γραμμικό πρόγραμμα L, δοθέντος ενός oracle για την κατανομή μ_L, έχει έναν query oblivious αλγόριθμο τοπικού υπολογισμού A ο οποίος επιτυγχάνει με πιθανότητα τουλάχιστον $1 - \delta$ και τρέχει σε χρόνο* $\mathrm{poly}(\log N, m, 1/\epsilon, \log(1/\delta))$.

Εφαρμόζοντας, εν τέλει, τον παραπάνω ΑΤΥ A στο γραμμικό πρόγραμμα της Ενότητας 8.2 μπορούμε, δοθέντος των παρατηρημένων σημάτων x, να ρωτάμε τον A για την τιμή του $q(x)$ σε μια ϵ-βέλτιστη λύση.

[1] R. Rubinfeld A. Czumaj, S. Muthukrishnan and C. Sohler. Sublinear algorithms. 2008.

[2] I. Algor and Noga Alon. The star arboricity of graphs. 1989.

[3] Noga Alon. The strong chromatic number of a graph.

[4] Noga Alon. The linear arboricity of graphs. 1988.

[5] Noga Alon. A parallel algorithmic version of the local lemma. 1991.

[6] Noga Alon, A. Bar-Noy, N. Linial, and D. Peleg. On the complexity of radio communication. 1989.

[7] Noga Alon and N. Linial. Cycles of length 0 modulo k in directed graphs. 1989.

[8] Noga Alon, Ronitt Rubinfeld, Shai Vardi, and Ning Xie. Space-efficient local computation algorithms. 2011.

[9] Noga Alon and J. Spencer. The probabilistic method. 2000.

[10] M. Datar R. Motwani B. Babcock, S. Babu and J. Widom. Testing random variables for independence and identity. *Proc. 42nd IEEE Conference on Foundations of Computer Science*, 2001.

[11] R. Rubinfeld B. Chazelle and L. Trevisan. Approximating the minimum spanning tree weight in sublinear time. 2001.

[12] L. Barenboim and M. Elkin. Distributed $(\Delta + 1)$-coloring in linear (in Δ) time. 2009.

[13] L. Barenboim and M. Elkin. Deterministic distributed vertex coloring in polylogarithmic time. 2010.

[14] S. Ben-David and A. Borodin. A new measure for the study of on-line algo- rithms. *Algorithmica*, 1994.

[15] G. Blelloch and Bruce Maggs. Parallel algorithms.

[16] Avrim Blum. On-line algorithms in machine learning. 1998.

[17] A. Deckelbaum C. Daskalakis and C. Tzamos. The complexity of optimal mechanism design. 2013.

[18] A. Deckelbaum C. Daskalakis and C. Tzamos. Optimal pricing is hard. 2015.

[19] E. H. Clarke. Multipart pricing of public goods. *Public Choice*, 1971.

[20] S. Cook. A taxonomy of problem with fast parallel algorithms. 1985.

[21] P. Erdös and L. Lovász. Problems and results on 3-chromatic hypergraphs and some related questions. 1975.

[22] L. M. Goldschlager. A unified approach to models of synchronous parallel machines. *Conference Record of the Tenth Annual ACM Symposium on Theory of Computing*, 1978.

[23] L. M. Goldschlager. A universal interconnection pattern for parallel computers. *Journal of the Association for Computing Machinery*, 1982.

[24] T. Griffiths and A. Yuille. Technical introduction: A primer on probabilistic inference.

[25] L. Grover. A fast quantum mechanical algorithm for database search. *Proceedings, STOC 1996*, 1996.

[26] T. Groves. Incentives in teams. *Econometrica*, 1973.

[27] Avinatan Hassidim, Yishay Mansour, and Shai Vardi. Local computation mechanism design. 2014.

[28] J. Hastad. Some optimal inapproximability results. 2002.

[29] Matthew Jackson. Mechanism theory. 2000.

[30] D. Johnson. Approximation algorithms for combinatorial problems.

[31] M. Rosen K. Onak, D. Ron and R. Rubinfeld. A near-optimal sublinear-time algorithm a near-optimal sublinear-time algorithm for approximating the minimum vertex cover size. 2011.

[32] H. Karloff. Randomized parallel algorithm for the odd-set cover problem.

[33] Howard Karloff. *Linear Programming.* Birkhäuser, 1991.

[34] R. M. Karp. On-line algorithms versus off-line algorithms: How much is it worth to know the future? 1992.

[35] R. M. Karp and V. Ramachandran. A survey of parallel algorithms for shared memory ma- chines. 1990.

[36] R. M. Karp and A. Wigderson. A fast parallel algorithm for the maximal independent set problem. 1984.

[37] D. Knuth. *The Art of Computer Programming.* Addison Wesley, 1969.

[38] F. Kuhn. Local multicoloring algorithms: Computing a nearly-optimal tdma schedule in constant time. 2009.

[39] F. Kuhn and T. Moscibroda. Distributed approximation of capacitated dominating sets. 2007.

[40] F. Kuhn, T. Moscibroda, T. Nieberg, and R. Wattenhofer. Fast deterministic distributed maximal inde- pendent set computation on growth-bounded graphs. 2005.

[41] F. Kuhn, T. Moscibroda, and R. Wattenhofer. What cannot be computed locally. 2004.

[42] F. Kuhn, T. Moscibroda, and R. Wattenhofer. The price of being near-sighted. 2006.

[43] F. Kuhn and R. Wattenhofer. On the complexity of distributed graph coloring. 2006.

[44] R. Kumar and Ronitt Rubinfeld. Algorithms column: Sublinear time algorithms. 2003.

[45] Michael Luby. A simple parallel algorithm for the maximal independent set problem. 1985.

[46] Yishay Mansour, Aviad Rubinstein, and Moshe Tennenholtz. Robust probabilistic inference. 2015.

[47] Yishay Mansour, Aviad Rubinstein, Shai Vardi, and Ning Xie. Converting online algorithms to local computation algorithms. 2012.

[48] S. Muthukrishnan. Data streams: algorithms and applications. *Proc. 14th Annual ACM–SIAM Symposium on Discrete Algorithms*, 2003.

[49] R. Myerson. Optimal auction design. 1981.

[50] H. N. Nguyen and K. Onak. Constant-time approximation algorithms via local improvements. 2008.

[51] K. Onak. New sublinear methods in the struggle against classical problems. 2010.

[52] Christos Papadimitriou. *Computational Complexity*. Addison Wesley, 1994.

[53] David Parkes. Iterative combinatorial auctions: Achieving economic and computational efficiency. 2001.

[54] M. Parnas and D. Ron. Approximating the minimum vertex cover in sublinear time and a connection to distributed algorithms. 2007.

[55] Ronitt Rubinfeld and Asaf Shapira. Sublinear time algorithms. 2011.

[56] Ronitt Rubinfeld, Gil Tamir, Shai Vardi, and Ning Xie. Fast local computation algorithms. 2011.

[57] s. Fortune and J. Wyllie. Parallelism in random access machines. *Conference Record of the Tenth Annual ACM Symposium on Theory of Computing*, 1978.

[58] W. J. Savitch and M. Stimson. Time bounded random access machines with parallel processing. *Journal of the Association for Computing Machinery*, 1979.

[59] V. Strassen. Gaussian elimination is not optimal. *Numerische Mathematik*, 1969.

[60] L. Lovàsz S. Safra U. Feige, S. Goldwasser and M. Szegedy. Interactive proofs and the hardness of approximating cliques. *Journal of the Association for Computing Machinery*, 1996.

[61] Vijay Vazirani. *Approximation Algorithms*. Springer.

[62] William Vickrey. Counterspeculation, auctions and competitive sealed tenders. *The Journal of Finance*, 1961.

[63] D. Williamson and D. Shmoys. *The Design of Approximation Algorithms*. Cambridge University Press, 2010.

[64] D. Paparas X. Sun M. Yannakakis Xi Chen, I. Diakonikolas. The complexity of optimal multidimensional pricing. 2013.