Εθνικο Μετσοβιο Πολυτεχνειο

Σχολη Ηλεκτρολογων Μηχανικων και Μηχανικων Υπολογιστων

Τομεας Τεχνολογιας Πληροφορικης και Υπολογιστων

Επεξεργασία Σύνθετων Ερωτημάτων σε Μεγάλης Κλίμακας Αποκεντρωμένα Συστήματα

Διδακτορικη Διατριβη

του

ΓΕΩΡΓΙΟΥ ΤΣΑΤΣΑΝΙΦΟΥ

Επιβλέπων : Τιμολέων Σελλής
Καθηγητής Ε.Μ.Π.

Εργαστηριο Συστηματων Βασεων Γνωσεων και Δεδομενων

Αθήνα, Οκτώβριος 2014

Εθνικο Μετσοβιο Πολυτεχνειο

Σχολη Ηλεκτρολογων Μηχανικων και Μηχανικων Υπολογιστων

Τομεας Τεχνολογιας Πληροφορικης και Υπολογιστων

Εργαστηριο Συστηματων Βασεων Γνωσεων και Δεδομενων

Επεξεργασία Σύνθετων Ερωτημάτων σε Μεγάλης Κλίμακας Αποκεντρωμένα Συστήματα

Διδακτορικη Διατριβη

του

ΓΕΩΡΓΙΟΥ ΤΣΑΤΣΑΝΙΦΟΥ

Τριμελής Επιτροπή : Τιμολέων Σελλής
Ιωάννης Βασιλείου
Νεκτάριος Κοζύρης

Ενεκρίθη από την Εξεταστική Επιτροπή την 24^{η} Οκτωβρίου 2014.

... Τ. Σελλής Καθηγητής ΕΜΠ	... Ι. Βασιλείου Καθηγητής ΕΜΠ	... Ν. Κοζύρης Καθηγητής ΕΜΠ
... Κ. Κοντογιάννης Καθηγητής ΕΜΠ	... Α. Σταφυλοπάτης Καθηγητής ΕΜΠ	... Δ. Φωτάκης Επικ. Καθηγητής ΕΜΠ

...
Δ. Τσουμάκος
Επικ. Καθηγητής
Τμήμα Πληροφορικής
Ιόνιο Πανεπιστήμιο

Περίληψη

Στη παρούσα διατριβή προτείνουμε έναν καινοτόμο τρόπο για να αποθηκεύεται με κατανεμημένο τρόπο και να ανακτάται αποδοτικά πολυδιάστατο περιεχόμενο σε μεγάλης κλίμακας αποκεντροποιημένα συστήματα, όπου πόροι και χρήστες μπορούν να εισέρχονται, να εξέρχονται ή να αποτυγχάνουν αυθαίρετα, χωρίς ποτέ όμως να διακυβεύεται η κανονική λειτουργία του συνολικού συστήματος. Μία σειρά σημαντικών εφαρμογών έχει χτιστεί πάνω στην δικτυακή υποδομή του κατανεμημένου ευρετηρίου. Συγκεκριμένα υλοποιήσαμε ένα κατανεμημένο αποθετήριο για RDF δεδομένα το οποίο υποστηρίζει ερωτήματα μοτίβου (triple pattern queries), συνδετικά (conjunctive) και διαζευκτικά ερωτήματα (disjunctive queries) και είναι σε θέση να απαντάει γρήγορα ερωτήματα μεταβατικής κλειστότητας (transitive closures). Στο ίδιο πλαίσιο προτείνουμε ένα RDF content-based publish-subscribe μοντέλο που επιτρέπει την εγγραφή συνδρομητών επιλεκτικά σε RDF περιεχόμενο. Επιπλέον, σχεδιάσαμε αλγορίθμους για την κατανεμημένη επεξεργασία top-k ερωτημάτων καθώς κι ερωτημάτων κορυφογραμμής (skyline computation). Ακόμα, παρουσιάζουμε κατανεμημένες μεθόδους αναζήτηση διαφοροποιημένου αποτελέσματος (search result diversification) για δομημένα δεδομένα. Τέλος, συνοδεύουμε την εργασία με μία εκτεταμένη πειραματική αποτίμηση των μεθόδων μας με τη χρήση πραγματικών και συνθετικών δεδομένων μεταβλητού μεγέθους, διαστάσεων, κατανομών κι άλλων παραμέτρων. Τα αποτελέσματα ήταν ενθαρρυντικά κι επιβεβαίωσαν την αποτελεσματικότητα κι αποδοτικότητα των μεθόδων μας τις οποίες συγκρίνουμε με άλλες σύγχρονες εδραιωμένες μεθόδους και τεχνικές. Το ευρύ έργο μας οδήγησε σε μία σειρά από άρθρα που έχουν ήδη δημοσιευτεί σε έγκριτα συνέδρια και περιοδικά.

Abstract

In this dissertation we propose a novel method for accessing and storing efficiently multi-dimensional content in a large-scale decentralized environment, where users join or leave arbitrarily as new resourses are made available to the participants and old ones are rendered obsolete, without ever compromising the functionality of the overlay. A series of important applications is built on top of our overlay. More specifically, we implemented a distibuted RDF/S repository that supports tripple pattern query, conjunctive, disjunctive queries, and is also able to compute transitive closures efficiently. Our paradigm is accompanied by a RDF content-based publish-subscibe model that allows selective subscriptions over RDF content of specific specifications. Furthermore, we developed algorithms for distributed processing of top-k queries and skyline queries. We also designed distributed algorithms for search result diversification over structured data. Finally, we present a thorough experimental evaluation with real and synthetic data of variable size, dimensionality, distribution and other parameters. The results were encouraging and validated the effectiveness and efficiency of our methods, which were compared against other consolidated schemes. Our wide work led to a series of articles that have been published in refereed journals and conferences.

Λέξεις Κλειδιά

δείκτες

κατανεμημένα συστήματα

επεκτασιμότητα-κλιμάκωση

δίκτυα ομοτίμων

ερωτήματα εύρους

ερωτήματα κοντινότερων γειτόνων

RDF ερωτήματα μοτίβου

ερωτήματα σύζευξης/διάξευξης

τεχνικές επισήμανσης

ερωτήματα μεταβατικής κλειστότητας

ερωτήματα κατάταξης

ερωτήματα προτίμησης

ερωτήματα κορυφογραμμής

αναζήτηση διαφοροποίησης αποτελέσματος

άπληστη ευρηστική αναζήτηση

αναζήτηση εναλλαγής / αναρρίχηση λόφων

Keywords

index structures

distributed systems

scalability

peer-to-peer networks

range queries

nearest neighbors queries

RDF triple pattern queries

conjunctive/disjunctive queries

labeling schemes

transitive closure computation

rank queries

top-k queries

skyline computation

search result diversification

greedy search

interchange / hill climbing

Περιεχόμενα

Κατάλογος πινάκων

Κατάλογος σχημάτων

List of Algorithms

Κεφάλαιο 1

Εισαγωγή

A journey of a thousand miles
begins with a single step.

Lao Tzu

Υπάρχει ένα συνεχές ενδιαφέρον για τη βελτίωση της ανάκτησης δεδομένων σε αποκεντρωμένα δίκτυα μεγάλης κλίμακας. Οι πιο αποτελεσματικές τεχνικές ακολουθούν το παράδειγμα των δομημένων δικτύων ομοτίμων (structured peer-to-peer networks) [82] και βρίσκουν εφαρμογές σε κατανεμημένα συστήματα αρχείων όπως το CEPH [122] και το OceanStore [80, 101], σε σχεσιακά query engines όπως το PIER [70, 71], στον εντοπισμό πόρων στο πλέγμα (grid) [72] ή στο νέφος (cloud) με το RT-CAN [119], σε υπηρεσίες αναζήτησης (search engines) όπως το yacy* [39], καθώς και σε torrent trackers για συστήματα διανομής περιεχομένου [98, 126]. Στα δομημένα δίκτυα, ένα πρωτόκολλο επικοινωνίας το οποίο είναι συμβατό με όλους τους κόμβους είναι υπεύθυνο για την διανομή περιεχομένου στους ομότιμους κόμβους (peers) που συμμετέχουν στο δίκτυο κι έτσι η αναζήτηση περιεχομένου επικεντρώνεται στον εντοπισμό των κόμβων που είναι υπεύθυνοι για αυτό.

Αν και πλήθος από δίκτυα ομοτίμων έχουν προταθεί κατά καιρούς, μόνο λίγα από αυτά είναι ικανά για την αποθήκευση και την ανάκτηση πολυδιάστατων δεδομένων. Οι σχετικές εργασίες που έχουν προταθεί μπορούν να χωριστούν στις εξής κύριες κατηγορίες. Η πρώτη περιλαμβάνει λύσεις που επεκτείνουν κάποια μονοδιάστατη δικτυακή δομή. Ο πιο άμεσος τρόπος είναι να επιλεγεί μία μόνο διάσταση προς δεικτοδότηση και να αγνοηθούν όλες οι υπόλοιπες, π.χ. με τη χρήση ενός μονοδιάστου δείκτη όπως το Chord [109]. Πρόκειται για μιά προσέγγιση με σαφή μειονεκτήματα όπως για παράδειγμα το ότι χάνεται πλέον η δυνατότητα να απαντηθούν ερωτήματα ως προς κάποια άλλη διάσταση εκτός αυτής στην οποία στηρίζεται το ευρετήριο χωρίς κάποιο επιπλέον κόστος (message overhead). Μια εναλλακτική λύση είναι να οριστούν ευρετήρια για κάθε διάσταση ξεχωριστά, όπως στα Mercury [30], MAAN [36]. Ωστόσο, αυτές οι προσεγγίσεις αναγκάζονται να επεξεργαστούν κάθεμια διάσταση ξεχωριστά με αποτέλεσμα ένα μεγάλο μέρος της πληροφορίας που ανακτούν να μην ικανοποιεί το ερώτημα του χρήστη (false hits), αυξάνοντας έτσι το κόστος επεξεργασίας (χρόνος, bandwidth) των ερωτημάτων και επιδρώντας έτσι αρνητικά στην απόδοση και την επεκτασιμότητα (scalability) των τελεστών αυτών. Πιο συγκεκριμένα οι προσεγγίσεις αυτές μετασχηματίζουν τον αρχικό χώρο κλειδιών (key-space) σε μία ενιαία τεχνητή διάσταση χρησιμοποιώντας μία καμπύλη πληρώσεως του χώρου (space-filling curve), όπως Hilbert [20] ή z-καμπύλες (z-curve) [90] στα SCRAP [61], ZNET [108], P-GridZ [31]. Οι τεχνικές αυτές δεν είναι τόσο αποτελεσματικές για πολλές διαστάσεις όπου η έννοια της τοπικότητας δεν μπορεί να διατηρηθεί. Για παράδειγμα, μια ορθογώνια περιοχή στον αρχικό πολυδιάστατο χώρο αντιστοιχεί σε πολλαπλά μη συνεχόμενα διαστήματα στον μετασχηματισμένο χώρο.

Η δεύτερη κατηγορία περιλαμβάνει δομημένα δίκτυα που έχουν σχεδιαστεί ειδικά για την αποθήκευση πολυδιάστατης πληροφορίας, όπως τα CAN [99] και MURK [61]. Σύμφωνα με την προσέγγιση αυτή κάθε ομότιμος κόμβος (peer) είναι υπεύθυνος για μια ορθογώνια περιοχή του χώρου και συνάπτει επικοινωνία με τους κόμβους που είναι υπεύθυνοι για πα-

ρακείμενες περιοχές του χώρου. Καθώς η δρομολόγηση επωφελείται από την ύπαρξη πολλών διαστάσεων (περισσότερες επιλογές για να προωθηθεί ένα ερώτημα και πιο μικρά μονοπάτια) το κόστος επεξεργασίας για τους περισσότερους τύπους ερωτημάτων είναι καλύτερο από γραμμικό ως προς το μέγεθος του δικτύου. Κύρια αδυναμία τους όμως είναι ότι δεν μπορούν να επωφεληθούν από μια ιεραρχική δομή ευρετηρίου. Ως εκ τούτου, ερωτήματα για απομακρυσμένο περιεχόμενο στο πολυδιάστατο χώρο αναπόφευκτα δρομολογούνται μέσω πάρα πολλών ενδιάμεσων κόμβων.

Η τελευταία κατηγορία περιλαμβάνει μεθόδους όπως το BATON [74], το VBI-tree [75] και το P2PR-tree [86], τα οποία λειτουργούν ως μία κατανεμημένη "απεικόνιση" ενός συμβατικού πολυδιάστατου ιεραρχικού δείκτη, όπως για παράδειγμα ένα δυαδικό R-δέντρο (R-tree) [64, 106, 25]. Η βασική ιδέα στο VBI-tree είναι ότι κάθε ομότιμος κόμβος αντιστοιχεί σε δύο κόμβους του δέντρου (έναν εσωτερικό κι ένα φύλλο) και εγκαθιδρά συνδέσεις με τον κόμβο στο αμέσως παραπάνω και παρακάτω επίπεδο. Επιπλέον υπάρχουν συνάψεις με επιλεγμένους κόμβους στο ίδιο επίπεδο του δέντρου που ανήκουν σε διαφορετικά υποδέντρα. Τα ερωτήματα επεξεργάζονται παρομοίως με συμβατικές προσεγγίσεις για τον σκληρό δίσκο, δηλαδή, ο δείκτης διασχίζεται ξεκινώντας από κάποιον κόμβο υψηλά στην ιεραρχία. Συνεπώς, αυτές οι μέθοδοι διατηρούν ιδιότητες όπως το λογαριθμικό κόστος αναζήτησης (λόγω της διαμέτρου του δικτύου), αλλά αντιμετωπίζουν σοβαρούς περιορισμούς. Ομότιμοι κόμβοι που αντιστοιχούν σε κόμβους ψηλά στο δέντρο μπορούν γρήγορα να υπερφορτωθούν λόγω της επεξεργασίας των ερωτημάτων που λαμβάνουν. Αν και αυτό είναι μια επιθυμητή ιδιότητα σε κεντρικά ευρετήρια προκειμένου να ελαχιστοποιηθεί ο αριθμός των I/O ενεργειών με την διατήρηση αυτών των κόμβων στην κύρια μνήμη, είναι ένας περιοριστικός παράγοντας σε δυναμικά κατανεμημένα συστήματα αφού οδηγεί σε συμφόρηση. Επιπλέον, το σύστημα γίνεται ιδιαίτερα ευαίσθητο σε σφάλματα αφού απαιτείται σημαντική προσπάθεια από το σύστημα για να ανακάμψει όταν ένας από τους ομότιμους κόμβους υψηλά στο δέντρο χάνεται απρόσμενα. Επιπλέον τα R-δέντρα (R-trees) είναι γνωστό ότι δεν είναι εξίσου αποτελεσματικά για πολλές διαστάσεις, π.χ. άνω του 8, ιδιότητα η οποία μεταφέρεται στους αποκεντρωμένους ομολόγους τους. Επιπλέον, λόγω του γεγονότος ότι είναι αναγκαστικά δυαδικό το δέντρο αυτό, δημιουργούνται αρκετές αλληλοεπικαλύψεις κι έτσι υπάρχουν πολλαπλά αντίγραφα ορισμένων δεδομένων. Ακόμα, ένα R-δέντρο σχεδιασμένο για τη δευτερεύουσα μνήμη (secondary memory) έχει πολύ διαφορετική μορφή και χαρακτηριστικά, δεν έχει βάθος αλλά αντίθετα είναι πολύ πλατύ. Ο λόγος έγκειται στο γεγονός ότι σε κάθε εσωτερικό κόμβο διαμερίζεται ο χώρος σε πολύ περισσότερα τμήματα και πιο αποτελεσματικά. Για παράδειγμα, για χωρικά δεδομένα δύο διαστάσεων μία μέση σελίδα (disk-page) των 4 KB χωράει πάνω από 200 minimum bounding rectangles (MBRs) που αντιστοιχούν στις περιοχές που δεικτοδοτούν τα ισάριθμα υποδέντρα. Άλλα μειονεκτήματα τα αποφεύγουν συγκεκριμένοι αλγόριθμοι από το πεδίο των βάσεων δεδομένων που στοχεύουν στο να "χτίζουν" αποδοτικά R-δέντρα στη δευτερεύουσα μνήμη.

Παρακινημένοι από αυτές τις παρατηρήσεις προτείνουμε τεχνικές επεξεργασίας ερωτημάτων που βασίζονται σε ένα καινοτόμο δομημένο δίκτυο ομοτίμων που ονομάζεται **M**ulti-attribute **I**ndexing for **D**istributed **A**rchitecture **S**ystems (MIDAS) [113]. Το MIDAS αποτελεί μια διαφορετική προσέγγιση από τις υπάρχουσες μεθόδους. Πρώτον, απεικονίζει μία ιεραρχική πολυδιάστατη δομή, το k-d δέντρο [27], με κατανεμημένο τρόπο. Αυτό έχει μια σειρά από πλεονεκτήματα. Όντας δυαδικό δέντρο επιτρέπει απλή και αποτελεσματική δρομολόγηση, με τρόπο που θυμίζει τον Plaxton αλγόριθμο [97] για μονοδιάστατες δομές. Σε αντίθεση με άλλες πολυδιάστατες δομές, π.χ. VBI-tree [75], οι ομότιμοι κόμβοι στο MIDAS αντιστοιχούν στα φύλλα του k-d δέντρου. Αυτό, λειτουργεί προς ώφελος των σημείων συμφόρησης (bottlenecks) και αυξάνει την επεκτασιμότητα (scalability) του συστήματος αφού κανένας κόμβος πλέον δεν ξεχωρίζει, π.χ. κόμβοι υπεύθυνοι για ανώτερα ιεραρχικά επίπεδα, κι έτσι δεν επιβαρύνονται με υπερβολικό φορτίο. Επιπλέον, το MIDAS είναι συμβατό με καθιερωμένες τεχνικές για εξισορρόπηση φόρτου εργασίας, δεδομένων και τεχνικές αντιγράφων (replication) που αποσκοπούν στην ανοχή σε σφάλματα (fault-tolerance).

Χρησιμοποιώντας την υποδομή που προσφέρει το MIDAS, οι πιο σημαντικοί και χαρακτηριστικοί τύποι πολυδιάστατων ερωτημάτων επεξεργάζονται αποτελεσματικά για αυθαίρετο αριθμό διαστάσεων. Ειδικότερα, δείχνουμε ότι για ένα δίκτυο n κόμβων, ερωτήματα

σημείων (point queries) κι εύρους (range queries) απαντώνται σε $O(\log n)$ βήματα. Πρόκειται για απόδοση καλύτερη ως προς άλλα d-διάστατα δομημένα συστήματα, όπως το CAN [99] που χρειάζεται $O(d\sqrt[d]{n})$ βήματα. Επιπλέον, προτείνουμε δύο εναλλακτικές μεθόδους για την επεξεργασία ερωτημάτων πλησιέστερων γειτόνων (nearest neighbors queries). Η πρώτη, που ονομάζεται eager processing, έχει χαμηλό latency με αναμενόμενη τιμή τα $O(\log n)$ βήματα, αλλά μπορεί να απασχολήσει έναν μεγάλο αριθμό από κόμβους κατά τη διάρκεια της επεξεργασίας. Η δεύτερη, που ονομάζεται iterative processing, έχει υψηλότερο latency (αναμενόμενη τιμή στα $O(\log^2 n)$ βήματα), αλλά είναι πιο συντηρητική κι επιβαρύνει πολύ λιγότερους κόμβους κι εξοικονομεί bandwidth κι άλλους πόρους του δικτύου. Μια εκτενής πειραματική μελέτη για πραγματικά χωρικά και συνθετικά δεδομένα διαφόρων διαστάσεων επιβεβαιώνει τα ανωτέρω.

Επιπλέον παρουσιάζουμε μια σειρά από σημαντικές εφαρμογές οι οποίες εκμεταλεύονται την υποδομή που προσφέρει το MIDAS και τις αποδοτικές μεθόδους αναζήτησης που παρέχει για πολυδιάστατο περιεχόμενο. Πρώτα, δείχνουμε το MIDAS-RDF [114], ένα κατανεμημένο αποθετήριο για RDF(S) δεδομένα, το οποίο υποστηρίζει ερωτήματα μοτίβου (triple pattern queries), συνδετικά (conjunctive) και διαζευκτικά ερωτήματα (disjunctive queries), καθώς κι έναν γρήγορο τρόπο για τον υπολογισμό ερωτημάτων μεταβατικής κλειστότητας (transitive closure computation). Ωστόσο, το πιο σημαντικό χαρακτηριστικό του MIDAS-RDF είναι ότι εφαρμόζει ένα επαγωγικό μοντέλο (inference model) που επιτρέπει την αποτελεσματική εξαγωγή και αποθήκευση νέας γνώσης που προέρχεται από την επεξεργασία αποθηκευμένης πληροφορίας με τη χρηση τεχνικών επισήμανσης (labeling scheme) μέσω προσαρμοσμένων μεθόδων συλλογιστικής για δυναμικά κατανεμημένα περιβάλλοντα (distributed reasoning). Συγκεκριμένα, η προσέγγισή μας υπερτερεί ως προς τους υπάρχοντες αλγόριθμους για κατανεμημένη συλλογιστική που βασίζονται σε μεταβατικές σχέσεις καθώς προσπελαύνουν σταδιακά έναν μεγάλο αριθμό από κόμβους που εξαρτάται από τη διάμετρο του RDF γράφου, αδυναμίες που δεν συμμερίζεται η δική μας μέθοδος. Ακόμα, προτείνουμε ένα RDF publish-subscribe μοντέλο που επιτρέπει στους ομότιμους κόμβους να εγγράφονται ως συνδρομητές επιλεκτικά σε RDF περιεχόμενο που τους ενδιαφέρει. Αντίθετα με τις περισσότερες κατανεμημένες λύσεις που περιορίζονται σε συνδρομές βάσει συγκεκριμένου θέματος που ορίζεται από κάποιες λέξεις κλειδιά, το MIDAS-RDF υποστηρίζει συνδρομές στη βάση δημοσιευμένου RDF περιεχομένου.

Ο όρος *ερωτήματα κατάταξης* (rank queries) αναφέρεται σε ερωτήματα τα οποία επιτάσσουν μία ολική διάταξη των πλειάδων και κατα συνέπεια την ανάκτηση των καλύτερων εκ των διατεταγμένων στοιχείων. Μελετούμε τους εξής κύριους τύπους τέτοιων ερωτημάτων:

- Τα *top-k* ερωτήματα επιβάλλουν μία διάταξη του πεδίου ορισμού (domain) μέσω μία μονότονης συνάρτησης. Το αποτέλεσμα περιλαμβάνει k πλειάδες οι οποίες αποτιμούνται υψηλότερα βάσει της συνάρτησης όταν συγκρίνονται με οποιαδήποτε άλλη αποθηκευμένη εγγραφή.

- Τα ερωτήματα *κορυφογραμμής* [34] επιβάλλουν μία μερική διάταξη του πεδίου ως ορίζεται από μία συνάρτηση συμψηφισμού Pareto aggregation που προσδιορίζεται σε κάθε χαρακτηριστικό ξεχωριστά (όσον αφορά τη μερική διάταξη, δύο εγγραφές μπορεί να μην δύναται να συγκριθούν μεταξύ τους). Η απάντηση σε ένα ερώτημα κορυφογραμμής είναι ένα σύνολο από στοιχεία που μεγιστοποιούν αυτή τη σχέση μερικής διάταξης. Τονίζουμε ότι για τη διάταξη των top-k ερωτημάτων υπάρχει ένα στοιχείο για το οποίο κανένα άλλο σημείο δεν επιτυγχάνει καλύτερο σκορ ενώ για τα ερωτήματα κορυφογραμμής μπορούν να βρεθούν άνω του ενός σημεία για τα οποία κανένα άλλο καλύτερο στοιχείο δεν υπάρχει.

- Τα ερωτήματα *διαφοροποίησης αποτελέσματος* (result diversification) [38] ενοποιούν δύο έννοιες που έρχονται σε αντίθεση. Η *σχετικότητα* ή *ομοιότητα* μίας πλειάδας ορίζεται από την απόστασή της από το ερώτημα. Από την άλλη, η *διαφοροποίηση* μίας πλειάδας ως προς το υπόλοιπο μέρος του αποτελέσματος ορίζεται από τον συνυπολογισμό της απόστασής της από τις εγγραφές αυτές. Η απάντηση σε ένα τέτοιο ερώτημα αποτελείται από ένα σύνολο k πλειάδων που επιτυγχάνει την καλύτερη τιμή σύμφωνα με μία αντικειμενική συνάρτηση (objective function) που συνδυάζει την σχετικότητα και

τη διαφοροποίηση του αποτελέσματος. Τονίζουμε ότι εδώ κατατάσσονται σύνολα από πλειάδες κι όχι οι ίδιες οι εγγραφές κι άρα μπορεί να δειχθεί ότι το πρόβλημα είναι NP-hard [62].

Ειδικά τα ερωτήματα διαφοροποίησης αποτελεσμάτων αποτελούν μία σύγχρονη ανοιχτή πρόκληση με την οποία απασχολείται τα τελευταία χρόνια η επιστημονική κοινότητα. Πιο συγκεκριμένα, στην εργασία αυτή παρουσιάζουμε μια διαφορετική οπτική του προβλήματος υπό το πρίσμα κριτηρίων που αφορούν το περιεχόμενο (content-based definitions). Εναλλακτικές προσεγγίσεις του προβλήματος βασίζονται είτε στην αρχή της κάλυψης (coverage-based definitions) σύμφωνα με την οποία κάθε καινούρια πλειάδα που προστίθεται στο αποτέλεσμα καλύπτει και μία διαφορετική κατηγορία κάποιας ταξονομίας, ή της νεωτερικότητας (novelty-based definitions), σύμφωνα με την οποία κάθε πλειάδα θα πρέπει να προσθέτει καινούρια πληροφορία στο αποτέλεσμα. Η προσέγγισή μας υιοθετεί μία **M**aximal **M**arginal **R**elevance (MMR) [38] στρατηγική κατάταξης των αποθηκευμένων αντικειμένων και στηρίζεται στις εξής ευριστικές (heuristics) τεχνικές αναζήτησης:

απληστείας (greedy) όπου ξεκινώντας από ένα άδειο αποτέλεσμα το "γεμίζουμε" σταδιακά προσθέτοντας σε κάθε επανάληψη το αντικείμενο (έγγραφο, πλειάδα) στην υψηλότερη κατάταξη,

εναλλαγής (interchange) όπου από ένα αρχικό αποτέλεσμα το βελτιώνουμε σε κάθε επανάληψη του αλγορίθμου αντικαθιστώντας στοιχεία του αποτελέσματος με άλλα καλύτερα έως ότου επέλθει μία στάσιμη κατάσταση όπου δεν επιδέχεται περαιτέρω βελτίωσης. Η τεχνική αυτή είναι επίσης γνωστή κι ως αναρρίχηση λόφων (hill climbing).

Βέβαια, έως τώρα έχουν εξευρευνηθεί κυρίως εκδοχές του προβλήματος που έχουν να κάνουν με συνδυαστική βελτιστοποίηση (combinatorial optimization), οι οποίες απαιτούν ανάγνωση ολόκληρης της εισόδου προκειμένου να υπολογίσουν το διαφοροποιημένο αποτέλεσμα. Συνεπώς, οι υπάρχουσες λύσεις είναι αρκετά ακριβές από άποψη I/O και χρόνου έπεξεργασίας. Από όσο μπορούμε να γνωρίζουμε αυτή είναι η πρώτη δουλειά που προσεγγίζει τα προβλήματα αυτά υπό το πρίσμα ενός δυναμικού κατανεμημένου περιβάλλοντος στο οποίο κόμβοι μπορούν να εισέρχονται ή να εξέρχονται αυθαίρετα χωρίς να επηρεάζεται κατά τα άλλα η κανονική λειτουργία του συνολικού συστήματος.

Το παρόν έχει δομηθεί ως εξής: στο κεφάλαιο 2 παρουσιάζουμε τη σχετική βιβλιογραφία οργανωμένη σε συγκεκριμένες θεματικές ενότητες. Στο κεφάλαιο 3 δείχνουμε την αρχιτεκτονική ενός καινοτόμου κατανεμημένου ευρετηρίου το οποίο εξειδικεύεται στην ανάκτηση κι αποθήκευση πολυδιάστατου περιεχομένου. Στο κεφάλαιο 4 μελετάμε αποδοτικούς αλγορίθμους επεξεργασίας ερωτημάτων (i) σημείων, (ii) εύρους, και (iii) κοντινότερων γειτόνων. Στο κεφάλαιο 5 προτείνουμε ένα κατανεμημένο αποθετήριο για RDF(S) δεδομένα το οποίο στηρίζεται στη δικτυακή υποδομή που παρουσιάσαμε στα προηγούμενα κεφάλαια. Στο κεφάλαιο 6 παρουσιάζουμε ένα πλαίσιο επεξεργασίας ερωτημάτων κατάταξης πάνω σε κατανεμημένα πολυδιάστατα δεδομένα. Πιο συγκεκριμένα, μελετάμε σε βάθος κι αναλύουμε μεθόδους και τεχνικές για την επεξεργασία (i) ερωτημάτων προτίμησης (top-k queries), (ii) κορυφογραμμής (skyline computation), καθώς και (iii) διαφοροποιημένου αποτελέσματος (search result diversification). Τέλος, στο κεφάλαιο 7 ανακεφαλαιώνουμε και συνοψίζουμε τη συνεισφορά μας.

Κεφάλαιο 2

Σχετικές Εργασίες

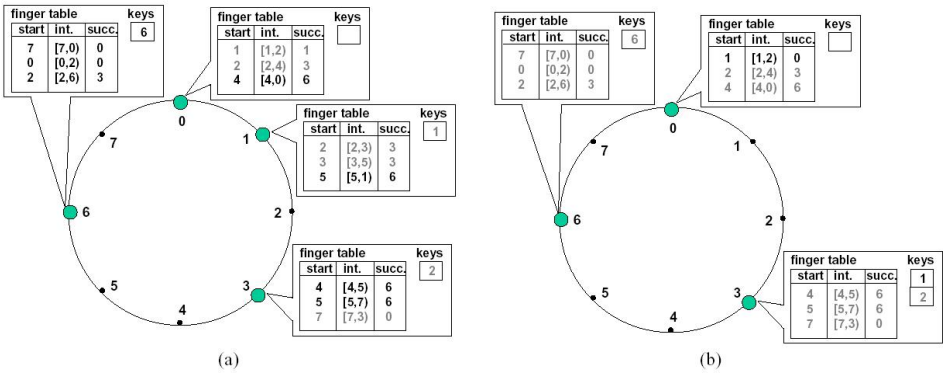
By three methods we may learn wisdom:
First, by reflection, which is noblest;
Second, by imitation, which is easiest;
and third by experience, which is the bitterest.

Confucius

Αυτό το κεφάλαιο εισάγει τον αναγνώστη σε βασικά προαπαιτούμενα και παραθέτει το βασικό υπόβραθρο που χρειάζεται για την κατανόηση της διατριβής. Επίσης παρουσιάζουμε τις πιο σημαντικές προόδους (state of the art) που έχουν γίνει έως σήμερα στο πεδίο των δομημένων δικτύων ομοτίμων (structured peer-to-peer networks), διάφορες τεχνολογίες του Σημασιολογικού Ιστού (Semantic Web), μεθόδους κατανεμημένης συλλογιστικής (distributed reasoning) καθώς και τους κύριους τύπους ερωτημάτων κατάταξης (rank queries), ήτοι top-k queries, ερωτήματα κορυφογραμμής (skyline queries), και διαφοροποιημένα αποτελέσματα (search result diversification).

2.1 Κατανεμημένοι Πίνακες Κατακερματισμού

Τα δομημένα δίκτυα ομοτίμων (structured peer-to-peer networks) [82] είναι μία σχετικά πρόσφατη τεχνολογία που μετράει μόλις μία δεκαετία σχεδόν. Τα συστήματα αυτά βασίζονται σε ένα πρωτόκολλο επικοινωνίας το οποίο είναι εκ των προτέρων γνωστό σε όλους τους κόμβους του δικτύου προκειμένου να διασφαλιστεί ότι οποιοσδήποτε κόμβος μπορεί να κατευθύνει αποτελεσματικά ένα ερώτημα προς τους κόμβους που είναι υπεύθυνοι για το επιθυμητό περιεχόμενο, ανεξάρτητα από το πόσο σπάνιο είναι ή που βρίσκεται αποθηκευμένο στο δίκτυο. Την πιο σημαντική κατηγορία δομημένων δικτύων ομοτίμων αποτελούν οι κατανεμημένοι πίνακες κατακερματισμού (Distributed Hash-Tables), ή DHTs εν συντομία. Τα DHTs είναι αποκεντροποιημένα, κατανεμημένα σύστηματα που παρέχουν υπηρεσίες αναζήτησης παρόμοιες με αυτές ενός πίνακα κατακερματισμού. Χρησιμοποιούν μια παραλλαγή του consistent hashing [79] προκειμένου να εκχωρηθεί η ευθύνη αποθήκευσης για κάθε ζευγάρι κλειδί-τιμή (key-value pair) σε ένα συγκεκριμένο κόμβο του δικτύου. Λόγω της δομής τους προσφέρουν συγκεκριμένες εγγυήσεις όσον αφορά την ανάκτηση περιεχομένου βάσει αναζήτησης για κάποιο κλειδί, π.χ. για αναζήτηση συγκεκριμένου κλειδιού απαιτείται το πολύ λογαριθμικός αριθμός βημάτων ως προς το μέγεθος του δικτύου. Τα DHTs αποτελούν μια αξιόπιστη υποδομή για τη δημιουργία σύνθετων υπηρεσιών, όπως κατανεμημένων συστημάτων αρχείων (filesystems), συστημάτων διανομής περιεχομένου (file sharing), συνεργατικού web-caching, multicast τεχνολογιών, κατανεμημένων υπηρεσιών DNS, torrent trackers, κλπ. Εν γένει, πρόκειται για μία τεχνολογία που απευθύνεται σε δικτυακές εφαρμογές των οποίων οι πόροι και οι χρήστες, οι οποίοι μπορούν να βρίσκονται γεωγραφικά διασκορπισμένοι εισέρχονται στο δίκτυο, απέρχονται και αποτυγχάνουν χωρίς όμως να διακυβεύεται η κανονική λειτουργία του συνολικού συστήματος.

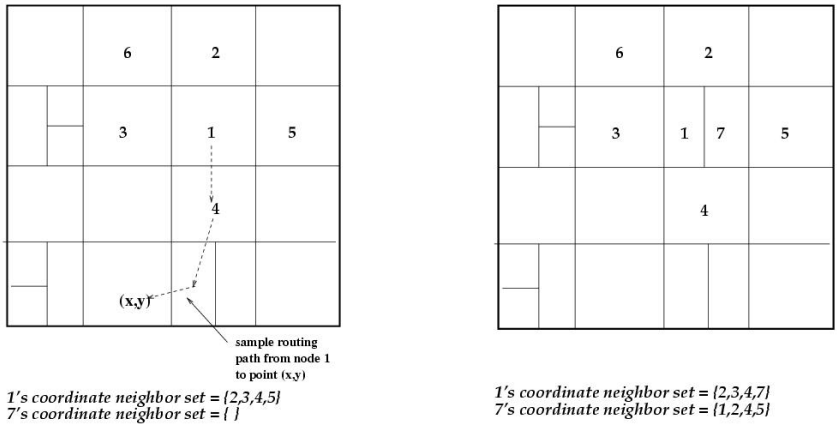


Σχήμα 2.1: Αναπαράσταση προσθήκης κι αποχώρησης κόμβου στο Chord.

Ίσως το πιο γνωστό παράδειγμα να είναι το Chord [109], το οποίο χρησιμοποιεί μια παραλλαγή του consistent hashing [79] προκειμένου να συνδέσει με μοναδικό τρόπο μονοδιάστατα αναγνωριστικά τους πόρους με τους κόμβους του δικτύου. Κατ' αρχήν το Chord σχηματίζει ένα δακτύλιο στον οποίο το κάθε κλειδί αντιστοιχίζεται στον πρώτο κόμβο του δικτύου του οποίου το αναγνωριστικό είναι ίσο ή έπεται του αναγνωριστικού του κόμβου. Κάθε κόμβος στο Chord διατηρεί επικαιροποιημένη πληροφορία για άλλους $\log n$ κόμβους, και διεκπεραιώνει αναζητήσεις σε $O(\log n)$ βήματα (hops), όπου n είναι το μέγεθος του δικτύου. Συγκεκριμένα ο κάθε κόμβος γνωρίζει ποιός κόμβος έχει το αμέσως προηγούμενο κι επόμενο αναγνωριστικό. Επιπλέον γνωρίζει τους κόμβους του δικτύου που είναι υπεύθυνοι για τα κλειδιά που απέχουν $2^x, x > 1$ από το δικό του αναγνωριστικό. Συνεπώς, είναι σε θέση να δρομολογήσει οποιοδήποτε ερώτημα για κάποιο κλειδί με το να το προωθήσει προς τον κόμβο ο οποίος βρίσκεται κοντινότερα αλλά δεν προηγείται από το κλειδί του ερωτήματος. Προκειμένου να εισέλθει ένας νέος κόμβος στο Chord ψάχνει να βρει τον κόμβο υπεύθυνο για κάποιο τυχαίο κλειδί. Στη συνέχεια αναλαμβάνει το άνω μισό του φορτίου δεδομένων (data-load), δηλαδή το αποθηκευμένο περιεχόμενο που αντιστοιχεί στα μισά κλειδιά με τις μεγαλύτερες τιμές. Ακόμα, κατασκευάζει το δικό του routing table με τους κόμβους του δικτύου που είναι υπεύθυνοι για τα κλειδιά που απέχουν $2^x, x > 1$ από το δικό του καινούριο αναγνωριστικό πάνω στον δακτύλιο, πολλοί εκ των οποίων είναι κοινοί με τον αρχικό κόμβο από τον οποίον προήλθε ο νέος. Αντίστοιχα, όταν ένας κόμβος θέλει να αποχωρήσει, απλά παραχωρεί το φορτίο του στον κόμβο του οποίου το αναγνωριστικό ακολουθεί. Στο σχήμα 2.1(a) δείχνουμε έναν δακτύλιο που αποτελείται από τους κόμβους 0, 1 και 3 ενώ ο κόμβος 6 έχει μόλις εισέλθει στο δίκτυο και κάνει γνωστή την παρουσία του κατασκευάζοντας το δικό του finger-table κι οπότε οι υπόλοιποι κόμβοι ενημερώνουν ανάλογα τα δικά τους μαζί με το διάστημα για το οποίο είναι υπεύθυνος ο καινούριος κόμβος. Αντίστοιχα, στο σχήμα 2.1(b) έχουμε την αποχώρηση του κόμβου 1 και κατά συνέπεια ο κόμβος επιβαρύνεται έπειτα με τα κλειδιά και τη ζώνη ευθύνης για τα οποία ήταν υπεύθυνος ο κόμβος που αποχώρησε. Μια παραλλαγή του Chord προτείνεται στο [51] στο M-Chord [88] για πολυδιάστατα κλειδιά χρησιμοποιώντας τεχνικές από το vp-tree [129] ή το iDistance τα οποία χρησιμοποιούν τις (μονοδιάστατες) αποστάσεις των αντικειμένων από συγκεκριμένα σημεία (vantage points) προκειμένου να επιτρέψουν την αποτελεσματική αναζήτηση πολυδιάστατου περιεχομένου. Παρόμοιες τεχνικές έχουν υλοποιηθεί σε κατανεμημένο περιβάλλον στα [131, 73].

Μια άλλη κατηγορία περιλαμβάνει δέντροειδείς δομές, όπως για παράδειγμα το P-Grid [14, 8, 15, 9, 13, 46, 10] το Kademlia [83], το TreeP [69], το Tapestry [135] και το Pastry [102]. Η αναζήτηση σε αυτά τα συστήματα γίνεται όπως στον αλγόριθμο του Plaxton [97]. Η βασική ιδέα βρίσκεται στο ότι για να βρεθεί ο υπεύθυνος κόμβος για ένα δεδομένο κλειδί θα πρέπει αναδρομικά ο κάθε λήπτης του ερωτήματος να το προωθήσει στον εκάστοτε γνωστό κόμβο του οποίου το αναγνωριστικό έχει το μεγαλύτερο κοινό πρόθεμα (largest common prefix) με το αιτούμενο μονοδιάστατο κλειδί του ερωτήματος. Το αίτημα προωθείται στον κόμβο αυτόν κι η διαδικασία επαναλαμβάνεται έως ότου το αίτημα να φθάσει εν τέλει στον ιδιοκτήτη του κλειδιού και του περιεχομένου που συσχετίζεται με αυτόν. Οι αναζητήσεις λαμβάνουν χώρα

σε $O(\log n)$ βήματα (hops) και κάθε κόμβος διατηρεί $O(\log n)$ πληροφορία για άλλους κόμβους στο δίκτυο. Όσον αφορά τα πρωτόλλα εισαγωγής κι αποχώρησης κόμβος, για το P-Grid ο νέος κόμβος επιλέγει ένα τυχαίο κλειδί κι εντοπίζει τον υπεύθυνο κόμβο. Στη συνέχεια υπάρχουν δύο ενδεχόμενα, είτε ο νέος κόμβος να αντιγράψει τα περιεχόμενα του παλαιού με σκοπό να αυξήσει τη διαθεσιμότητα () και το redundancy του δικτύου αλλά και να μειώσει την κίνηση που δέχεται ο παλαιός, είτε να γίνει υπεύθυνος για τα μισά κλειδιά με το να επεκτείνει το δέντρο σε βάθος κι έτσι τα αντίστοιχα αναγνωριστικά των δύο εμπλεκομένων κόμβων κατά ένα δυφίο (bit), όπου πάλι έτσι θα μειωθεί κατά ένα ποσοστό η κίνηση στον αρχικό κόμβο. Ανίστοιχα για την αποχώριση ενός κόμβου χωρίς αντίγραφα, ο αδερφός κόμβος (sibling) αναλαμβάνει την ζώνη αμφότερων κόμβων μαζί με τα αντίστοιχα κλειδιά ενώ το αναγνωριστικό του μειώνεται κατά ένα δυφίο (bit). Το ευρετήριο που προτείνεται στα επόμενα κεφάλαια ανήκει στην κατηγορία αυτή καθώς διαθέτει δεντροειδή δομή παρόμοια με αυτήν του k-d δέντρου (k-d tree) [28, 27, 59] με λογαριθμικό αριθμό γειτόνων για κάθε κόμβο αλλά διαφέρει στο ότι πρόκειται για ένα εγγενώς πολυδιάστατο ευρετήριο που είναι σε θέση να εκτελεί αναζητήσεις για πολυδιάστατα κλειδιά σε $O(\log n)$ βήματα (hops). Πιο συγκεκριμένα, το k-d δέντρο [28] ανήκει στις χωρικές και γεωμετρικές δομές δεδομένων [60, 33] και χρησιμοποιείται για πολυδιάστατα δεδομένα που αποθηκεύονται στην κύρια μνήμη. Πρόκειται για ένα δυαδικό δέντρο αναζήτησης (binary search tree) το οποίο αντιπροσωπεύει έναν αναδρομικό διαμερισμό του d-διάστατου μετρικού χώρου σε υποπεριοχές διαχωριζόμενες από $(d-1)$-διάστατες επιφάνειες οι οποίες είναι παράλληλες ως προς κάποιον άξονα και η κατεύθυνσή τους εναλάσσεται μεταξύ των d διαφορετικών επιλογών. Το προσαρμόστικό k-d δέντρο (adaptive k-d tree) [27] επιλέγει να χωρίζει τον χώρο με τέτοιο τρόπο ώστε να υπάρχει ίσος αριθμός κλειδιών σε κάθε πλευρά. Στη συγκεκριμένη παραλλαγή οι διαμερίσεις δεν χρειάζεται να αντιστοιχούν σε κάποιο συγκεκριμένο σημείο και οι διαστάσεις δεν χρειάζεται να εναλλάσονται κυκλικά. Για παράδειγμα, μπορεί να χρησιμοποιείται κάθε φορά η διάσταση που έχει τη μεγαλύτερη έκταση.



Σχήμα 2.2: Αναπαράσταση προσθήκης νέου κόμβου στο CAN.

Στη συνέχεια παρουσιάζουμε τα πιο σημαντικά δομημένα δίκτυα ομοτίμων τα οποία είναι σχεδιασμένα για να δεικτοδοτούν περιεχόμενο βάσει πολυδιάστατων κλειδιών. Στο CAN [99] κάθε κόμβος είναι υπεύθυνος για μια συγκεκριμένη παραλληλόγραμμη περιοχή του d-διάστατου χώρου. Κάθε κόμβος περιέχει πληροφορίες σχετικά με τους κόμβους των οποίων οι περιοχές συνορεύουν ή εφάπτονται κι οπότε η πληροφορία που πρέπει να διατηρεί ενήμερη ο κάθε κόμβος έχει μέγεθος $O(d)$. Οι αιτήσεις για d-διάστατα κλειδιά δρομολογούνται άπλησταπρος τον γειτονικό κόμβο του οποίου η ζώνη βρίσκεται κοντινότερα στο κλειδί. Συνεπώς, το κόστος της αναζήτησης φράσσεται από $O(d\sqrt[d]{n})$ βήματα. Κατ᾽ αναλογίαν σχεδιάστηκε το MURK [61] όπου ο χώρος κλειδιών σχηματίζει έναν d διάστατο τόρο (torus). Μια εφαρμογή του CAN με βάση το R-δέντρο (R-tree), ονόμαζεται RT-CAN [119], έχει βρει εφαρμογή σε περιβάλλοντα νέφους (cloud).

Για να επεκταθεί το CAN προκειμένου να φιλοξενήσει έναν ακόμα κόμβο θα πρέπει

Taxonomy	Chord	Kademlia	Tapestry/Pastry	CAN
Architecture	unidirectional and circular keyspace.	xor metric distance between points.	plaxton-style mesh network.	multidimensional keyspace.
Routing	find the node with min id succeeding the queried key.	match key and node id based routing.	match key and suffix/prefix in node id.	maps points in a coordinate space among all peers.
Parameters	n peers	n peers, $\log B$ bits	n peers, $\log B$ bits	n peers, d dims
Performance	$O(\log n)$	$O(\log_B n)+c$	$O(\log_B n)$	$O(\sqrt[d]{n})$
Routing State	$\log n$	$B\log_B n+B$	$2B\log_B n$	$2d$
Join/Leave	$\log^2 n$	$\log_B n+c$	$\log_B n$	$2d$

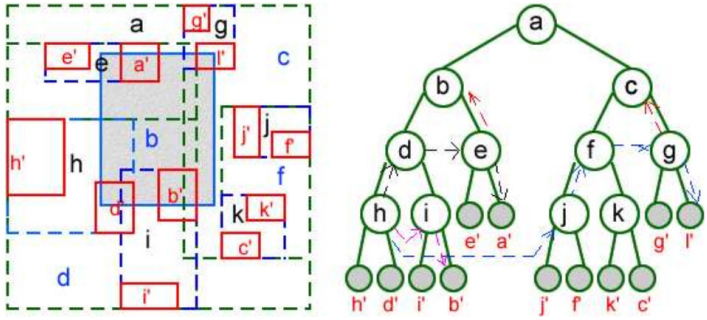
Πίνακας 2.1: Σύγκριση αρχιτεκτονικών διαφόρων δικτύων ομοτίμων.

ένας τυχαίος κόμβος να αναθέσει μία d-διάστατη παραλληλόγραμμη περιοχή στον καινούριο κόμβο. Συγκεκριμένα, η περιοχή αυτή αντιστοιχεί στο μισό της αρχικής, ενώ στο MURK [61] ο διαχωρισμός γίνεται με τέτοιο τρόπο ώστε να κατέχουν αμφότεροι κόμβοι τον ίδιο αριθμό κλειδιών. Έπειτα ενημερώνονται όλοι οι γειτονικοί κόμβοι του αρχικού για την αλλαγή στην τοπολογία του δικτύου κι όπου χρειάζεται αλλάζουν τα αντίστοιχα στοιχεία στα αντίστοιχα routing tables. Ομοίως, για να αποχωρήσει κάποιος κόμβος θα πρέπει να αναθέσει την περιοχή για την οποία είναι υπεύθυνος μαζί με όλα τα δεδομένα τα οποία ανήκουν σε αυτήν σε κάποιον από τους γειτονικούς του κόμβους με τον οποίον θα εφάπτεται σε όλες πλυν μίας διάστασης. Αν δεν υπάρχει τέτοιος κόμβος τότε παραχωρεί το περιεχόμενό του ως έχει και φιλοξενείται ως εικονικός κόμβος (virtual node) σε κάποιο κοντινό μηχάνημα χωρίς μεγάλο φορτίο (task-load). Στο Σχήμα 2.2 παρουσιάζουμε μία αναπαράσταση των πρωτοκόλλων εισαγωγής νέων κόμβων κι αποχώρησης. Αρχικά ο νέος κόμβος επιλέγει ένα τυχαίο σημείο του πολυδιάστατου χώρου κι εντοπίζει τον υπεύθυνο κόμβο 1 χρησιμοποιώντας την υποδομή του δικτύου. Έπειτα ο κόμβος αυτός χωρίζεται στα δύο κι ο καινούριος κόμβος 7 αναλαμβάνει τη μισή ζώνη υπευθυνότητας του αρχικού κόμβου 1 μαζί με όλα τα key-value ζεύγη που τη συνοδεύουν αλλά επιπλέον σχηματίζει το routing table ενώ ταυτόχρονα ενημερώνει όλους τους γείτονές του προκειμένου να ενημερώσουν κι αυτοί τα δικά τους routing tables ανάλογα.

Το G-Grid [91] αποτελεί ένα ακόμα εγγενώς πολυδιάστατο κατανεμημένο ευρετήριο, το οποίο διαχειρίζεται τον αρχικό χώρο κλειδιών χωρίς κάποιου είδους μετασχηματισμό και μπορεί έτσι να επωφελείται από την έννοια της τοπικότητας για να επεξεργάζεται αποδοτικά ερωτήματα εύρους (range queries). Όλοι οι κόμβοι του δέντρου αντιστοιχούν σε κάποιον πραγματικό κόμβο του δικτύου ενώ οι κόμβοι που βρίσκονται στα ανώτερα επίπεδα έχουν αναγνωριστικά που αποτελούν πρόθεμα για τα αναγνωριστικά των κόμβων στα κατώτερα επίπεδα κι είναι υπεύθυνοι για περιοχές που περικλείουν πλήρως τις περιοχές των κόμβων στα κατώτερα επίπεδα. Αντίστοιχα, κόμβοι που βρίσκονται στο ίδιο επίπεδο διαφέρουν κατά ένα τουλάχιστον bit και οι περιοχές για τις οποίες ευθύνονται δεν αλληλοεπικαλύπτονται.

Εναλλακτικές προσεγγίσεις όπως το SCRAP [61] και το ZNET [108], χρησιμοποιούν μία καμπύλη πληρώσεως του χώρου (space-filling curve) για να μετασχηματίσουν τον αρχικό πολυδιάστατο χώρο σε μια ενιαία διάσταση και στη συνέχεια να καθίσταται ικανή η χρήση ένός συμβατικού μονοδιάστατου δείκτη στον μετασχηματισμένο χώρο. Για παράδειγμα, στο P-GridZ [31], [32] χρησιμοποιείται η z-καμπύλη (z-curve) [90] πάνω από το P-Grid για την υποστήριξη πολυδιάστατων ερωτημάτων εύρους (range queries). Το πρόβλημα με αυτές τις μεθόδους είναι ότι η τοπικότητα του αρχικού χώρου δεν μπορεί να διατηρηθεί για μεγάλο αριθμό διαστάσεων. Ως εκ τούτου ένα απλό ερώτημα εύρους αποσυντίθεται σε πολλαπλά ερωτήματα στον μετασχηματισμένο χώρο, γεγονός που αυξάνει το κόστος επεξεργασίας. Επιπλέον πολλά από τα αντικείμενα που έχουν ανακτηθεί από τον μετασχηματισμένο χώρο δεν ικανοποιούν τελικά το ερώτημα (false hits) κι έτσι απορρίπτονται μετέπειτα, αφού όμως έχει επιβαρυνθεί το σύστημα για την ανάκτηση και τη μεταφορά τους.

Το MAAN [36] και το Mercury [30] μπορούν να υποστηρίξουν ένα φάσμα πολυδιάστατων ερωτημάτων προβάλλοντάς τα στη μία διάσταση του Chord. Δεν αποτελούν εγγενώς πολυδιάστατα ευρετήρια όπως οι δομές που αντιμετωπίζουν κάθε διάσταση ανεξάρτητα. Ως



Σχήμα 2.3: Αναπαράσταση επεξεργασίας ερωτήματος εύρους στο VBI-tree.

αποτέλεσμα, ένα ερώτημα εύρους διαβιβάζεται στην πρώτη τιμή που εμφανίζεται στην περιοχή αυτή και στη συνέχεια προωθείται κατά μήκος των γειτονικών κόμβων. Μία "ακριβή" διαδιασία αφού πολλοί μη σχετικοί με το ερώτημα κόμβοι απασχολούνται στα πλαίσια της επεξεργασίας του ερωτήματος αφού πολλοί μη σχετικοί με το ερώτημα κόμβοι μπορεί να προβάλλουν τον πολυδιάστατο χώρο τους εντός του διαστήματος στο οποίο προβάλλεται το ίδιο το ερώτημα. Ξέχωρα από ζητήματα σπατάλης πόρων, τίθεται και το θέμα της απόδοσης. Ειδικά αν αναλογιστούμε ότι οι σχετικοί κόμβοι προσεγγίζονται ένας προς έναν ξεκινώντας από το ένα άκρο του ερωτήματος εύρους μέχρι να βρεθεί ο κόμβος υπεύθυνος για το άλλο άκρο και στο μεσοδιάστημα όλοι οι σχετικοί κόμβοι να επιστρέψουν το δικό τους μέρος του αποτελέσματος στον αιτούντα κόμβο. Άρα δεν είναι δύσκολο να διαπιστώσουμε ότι ανάλογα με την επιλεκτικότητα (selectivity) του ερωτήματος υπάρχει το ενδεχόμενο να απαιτηθεί ιδιαίτερα πολύς χρόνος επεξεργασίας με κατα συνέπεια την ιδιαίτερα μειωμένη απόδοση.

Algorithm 1: *u*.VBI-Join (Node *v*): Εκτελείται στον κόμβο *u* για την εισαγωγή του νέου κόμβου *v*.

```
1  if u.leftRoutingTable.isFull() and u.rightRoutingTable.isFull()
2  and (not u.leftChild ≠ nil or not u.rightChild ≠ nil ) then
3      u.setChild (v) ;
4  else
5      if not u.leftRoutingTable.isFull() or not u.rightRoutingTable.isFull() then
6          u.parent.VBI-Join (v) ;
7      else
8          w = someNodesNotHavingEnoughChildrenIn (u.leftRoutingTable,
           u.rightRoutingTable) ;
9          if w ≠ nil then
10             w.VBI-Join (v) ;
11         else
12             y.VBI-Join (v) //// Forward request to an adjacent node y ;
```

Το VBI-tree [75] που αποτελεί επέκταση του BATON [74] ένα κατανεμημένο αποθετήριο για πολυδιάστατα δεδομένα. Η δομή του βασίζεται σε ιεραρχικές δομές δεδομένων όπως το R-tree [64, 106, 25] και το M-tree [42]. Παρέχει μια αφαιρετική δομή δέντρου με σκοπό την υποστήριξη ιεραρχικών δομών δεικτοδότησης, όπου το τμήμα του χώρου για το οποίο είναι υπεύθυνος ένας κόμβο επικαλύπτει πλήρως τις περιοχές που διαχειρίζονται οι κόμβοι στο επόμενο επίπεδο. Οι κόμβοι στο VBI-tree χωρίζονται σε δύο κατηγορίες: τα φύλλά όπου αποθηκεύονται τα δεδομένα και οι εσωτερικοί κόμβοι όπου χρησιμοποιούνται για τη δρομολόγηση (routing). Ακόμα, ο κάθε κόμβος διατηρεί πληροφορία για τον γονέα του, τα παιδιά

του στο δέντρο αλλά και για κόμβους που βρίσκονται στο ίδιο επίπεδο με αυτόν, για την ακρίβεια για έναν από κάθε υποδέντρο στο οποίο δεν ανήκει ο ίδιος. Το Σχήμα 2.3 απεικονίζει την επεξεργασία ενός ερωτήματος εύρους (range query) όπου ξεκινώντας από τον κόμβο h, το ερώτημα προωθείται ταυτόχρονα σε κόμβους γειτονικούς αλλά κι από διαφορετικά επίπεδα, που αντιπροσωπεύουν ζώνες του χώρου οι οποίες επικαλύπτονται με το ερώτημα. Παρ' όλα αυτά, μπορεί να προκληθεί συμφόρηση καθώς οι κόμβοι που αντιστοιχούν στην κορυφή του δέντρου δέχονται δυσανάλογα υψηλό φορτίο κι επιφορτίζονται με το να διανέμουν και να προωθούν μηνύματα από διάφορους κόμβους σε διάφορα επίπεδα. Επιπλέον, τα πρωτόλλα εισαγωγής κι αποχώρησης κόμβων είναι πιο πολύπλοκα από αυτά που εξετάσαμε προηγουμένως εξ αιτίας της επιπλέον προσπάθειας που καταβάλλεται προκειμένου να διατηρηθεί το δέντρο ισοζυγισμένο. Αυτό επιτυγχάνεται με την αναδρομική εξέταση των routing tables έως ότου διαπιστωθούν "κενά" στο δέντρο με τη χρήση του Αλγόριθμου 1 και τα οποία θα συμπληρωθούν ανάλογα με τους νέους κόμβους. Αντίστοιχα, για να αποχωρήσει με επιτυχία ένας κόμβος θα πρέπει να βρει έναν κατάλληλο αντικαταστάτη (surrogate node) ο οποίος θα πρέπει να ανήκει στα χαμηλά επίπεδα του δέντρου χωρίς όμως ο Αλγόριθμος 2 να λαμβάνει υπόψην το φορτίο των υποψήφιων κόμβων που καλούνται να επωμιστούν τα δεδομένα και τις αντίστοιχες αιτήσεις για το περιεχόμενο του κόμβου που αποχωρεί. Κατά συνέπεια τα πρωτόκολλα του VBI-tree είναι πιο ακριβά, για παράδειγμα η εισαγωγή ενός νέου κόμβου λαμβάνει χώρα σε $6\log n$ βήματα (hops) συμπεριλαμβανομένου και του κόστους ενημερώσεως των routing tables των σχετιζόμενων κόμβων, ενώ η γενική περίπτωση της αποχώρησης $8\log n$ βήματα, όπου το n συμβολίζει το μέγεθος του δικτύου. Το P2PR-tree [86] αποτελεί μία ακόμα παρόμοια εργασία. Σε άλλες προσεγγίσεις, όπως για παράδειγμα στο [110], προτείνεται η κατασκευή ενός κατανεμημένου ευρετηρίου κατ' εικόναν του quadtree [104].

Algorithm 2: u.VBI-FindReplacement (Node v): Εκτελείται στον κόμβο u για την εύρεση αντικαταστάτη του v.

```
1  if u.leftChild ≠ nil then
2      u.leftChild.VBI-FindReplacement (v) ;
3  else if u.rightChild ≠ nil then
4      u.rightChild.VBI-FindReplacement (v) ;
5  else
6      w = someNodesNotHavingEnoughChildrenIn (u.leftRoutingTable,
       u.rightRoutingTable) ;
7      if w ≠ nil then
8          w.VBI-FindReplacement (v) ;
9      else
10         u.replaceNode (v) ;
```

2.2 Αποθετήρια RDF Δεδομενων

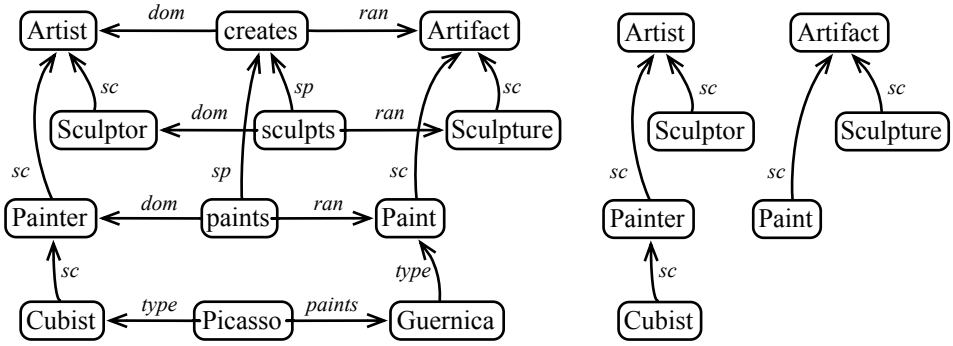
2.2.1 RDF/S Έννοιες

Ο Σημασιολογικός Ιστός (Semantic Web) [29] αποτελείται από μία ομάδα μεθόδων και τεχνολογιών που έχουν ως απώτερο σκοπό τους να αποδόσουν στους υπολογιστές τη δυνατότητα να κατανοήσουν το νόημα και τη σημασία της πληροφορίας σχετικά με τον παγκόσμιο ιστό (World Wide Web). Ο όρος επινοήθηκε από το διευθυντή του World Wide Web Consortium (W3C) [6] Tim Berners-Lee, ο οποίος ορίζει το σημασιολογικό ιστό ως τον "ιστό των δεδομένων που μπορούν να επεξεργαστούν οι υπολογιστές άμεσα ή έμμεσα". Οι τεχνολογίες αυτές περιλαμβάνουν το Resource Description Framework (RDF), που εκφράζονται από μια ποικιλία μορφών δεδομένων (π.χ., RDF/XML, N3, Turtle, N-triple) και σημασιολογίες όπως το RDF Schema (RDFS) και η Web Ontology Language (OWL), που παρέχουν μία

τυπική κι επίσημη περιγραφή των εννοιών, των όρων και των σχέσεων στα πλαίσια ενός συγκεκριμένου γνωστικού πεδίου.

Το μοντέλο δεδομένων του Resource Description Framework (RDF) [63] εκφράζει μία πληροφορία ως δήλωση για κάποιους πόρους. Κάθε έννοια που μπορεί να έχει ένα Universal Resource Identifier (URI) μπορεί να θεωρηθεί ως ένας πόρος. Το RDF αποτελείται από ένα σύνολο από κατηγορήματα που ονομάζονται RDF δηλώσεις και παρουσιάζονται ως τριπλέτες (triples). Μια τριπλέτα `(υποκείμενο, κατηγόρημα, αντικείμενο)` εκφράζει τη σχέση που συμβολίζεται με το `κατηγόρημα` μεταξύ των εννοιών που συμβολίζονται ως `υποκείμενο` και `αντικείμενο`. Οι έννοιες και οι σχέσεις αυτές μπορούν να οπτικοποιηθούν σε έναν επισημασμένο γράφο όπου το κατηγόρημα και το αντικείμενο είναι κόμβοι ή ακμές.

Κατ᾽ επέκταση, ένα σύνολο από RDF τριπλέτες ορίζει ένα γράφημα RDF. Το σύνολο των κόμβων είναι το σύνολο των υποκειμένων και των αντικειμένων των τριπλετών. Οπότε, μία (u, α, v) τριπλέτα ορίζει μια κατευθυνόμενη ακμή από το υποκείμενο u στο αντικείμενο v, επισημασμένη με την ετικέτα που αντιστοιχεί στην ιδιότητα α. Η εικόνα 2.4(α΄) δείχνει ένα παράδειγμα RDF γραφήματος. Η `(Picasso, paints, Guernica)` τριπλέτα παρουσιάζεται ως μία ακμή με την ετικέτα `paints` από τον κόμβο `Picasso` προς τον κόμβο `Guernica`.



(α΄) RDF graph (β΄) The *sc* subgraph

Σχήμα 2.4: Παράδειγμα RDF γράφου.

Το σύνολο όλων των τριπλετών σχετικά με μία ιδιότητα α ορίζουν α-υπογράφημα του συνολικού RDF-γραφήματος. Για παράδειγμα, το σχήμα 2.4(β΄) απεικονίζει το αντίστοιχο υπογράφημα για την ιδιότητα `sc`. Σε ένα υπογράφημα, ένας κόμβος *u υπάγει/συμπεριλαμβάνει* έναν άλλο v αν υπάρχει τουλάχιστον ένα μονοπάτι από τον κόμβο v προς τον u. Έτσι έχουμε στο σχήμα 2.4(β΄), την έννοια `Artist` να συμπεριλαμβάνει τις έννοιες `Scupltor`, `Painter` και `Cubist`.

Η προδιαγραφή του RDF περιλαμβάνει ένα μηχανισμό, που ονομάζεται RDF Schema (RDFS), ο οποίος παρέχει έναν τύπο συστήματος (type system[1]) για RDF μοντέλα. Παρέχει ευέλικτες δυνατότητες για την τυποποιημένη μοντελοποίηση της πληροφορίας. Καθίσταται δυνατή η πολλαπλή κληρονομικότητα μέσω των σχετικών ιδιοτήτων `rdfs:subClassOf` και `rdfs:subPropertyOf` (συντομογραφία `sc` και `sp` στο Σχήμα 2.4) κι η ταξινόμηση των πόρων `rdf:type` (συντομογραφία `type`). Αν και το RDFS δεν παρέχει μηχανισμούς για τον καθορισμό περιορισμών σε ιδιότητες, μπορεί κανείς όμως να δηλώσει τόσο το πεδίο ορισμού όσο και το πεδίο τιμών, `rdfs:domain` και `rdfs:range` αντίστοιχα (συντομογραφία `dom` και `ran` στο Σχήμα 2.4).

Η σημαντική (semantics) στο RDFS ορίζεται μέσα από μια σειρά από αξιωματικές τριπλέτες και *κανόνες συνεπαγωγής (entailment rules)* [5], που κατά συνέπεια ορίζουν το σύνολο των έγκυρων συμπερασμάτων. Αυτοί οι συμπερασματικοί κανόνες μπορούν να εξηγηθούν απλούστερα ως εξής: Κάθε συμπερασματικός κανόνας διαθέτει ένα σύνολο από μέρη τα οποία συλλογικά καθορίζουν το σώμα του κι αντιπροσωπεύουν "επεκταμένες" RDF δηλώσεις, όπου μεταβλητές μπορούν να καταλάβουν οποιαδήποτε από τις τρεις πιθανές θέσεις

[1] πρόκειται για μια συλλογή κανόνων που αναθέτουν μια ιδιότητα που ονομάζεται τύπος στις διάφορες δομές ενός προγράμματος, όπως μεταβλητές, εντολές ή συναρτήσεις.

στην τριπλέτα (είτε εκείνη του αντικειμένου, είτε του κατηγορήματος, είτε του αντικειμένου). Έκαστο μέρος από το σώμα του κανόνα αποτελεί μία RDF δήλωση. Η εφαρμογή των κανόνων συνεπαγωγής σε ένα RDF γράφημα οδηγεί στη δημιουργία νέων τριάδων που μπορούν με τη σειρά τους να θεωρηθούν ως μια συμπερασματική κλειστότητα (inferred closure) του αρχικού γράφου. Ο μόνος τρόπος για να προσδιοριστεί αν μια συγκεκριμένη δήλωση μπορεί να εξαχθεί από ένα RDF γράφημα είναι μέσω του ελέγχου για το αν είναι μέλος της συμπερασματικής του κλειστότητας.

Στα απλά συστήματα συμπερασμού, υπάρχουν δύο κύριες στρατηγικές συλλογισμού. Πρώτον, η *forward chaining* στρατηγική αφορά την εφαρμογή των κανόνων εξαγωγής συμπερασμάτων για γνωστά γεγονότα με σκοπό τη δημιουργία νέων δεδομένων. Οι κανόνες αυτοί μπορούν στη συνέχεια να εφαρμοστούν εκ νέου με συνδυασμό των αρχικών δεδομένων και έτσι συνάγεται ακόμα περισσότερη πληροφορία. Η διαδικασία αυτή είναι επαναληπτική και συνεχίζεται μέχρι να μην μπορούν να δημιουργηθούν νέα δεδομένα. Το βασικό πλεονέκτημα αυτής της προσέγγισης είναι ότι όταν όλα τα συμπεράσματα έχουν εξαχθεί, τότε η επεξεργασία οποιουδήποτε ερωτήματος διεκπεραιώνεται εξαιρετικά γρήγορα. Στα μειονεκτήματα συγκαταλέγεται το μεγαλύτερο κόστος εκκίνησης λόγω της απαραίτητης προεπεξεργασίας καθώς και ο αυξημένος χώρος αποθήκευσης.

Από την άλλη πλευρά, στο *backward chaining* ξεκινάμε από ένα γεγονός το οποίο πρέπει να αποδειχθεί. Συνήθως το σύστημα εξετάζει τη τράπεζα γνώσης (knowledge base) προκειμένου να διαπιστώσει εάν το γεγονός αυτό υπάρχει κι αν όχι τότε εξετάζει το σύνολο των κανόνων για να επιλέξει με ποιους θα μπορούσε να συνδυαστεί. Ως εκ τούτου, απαιτείται ένας επιπλέον έλεγχος για να διαπιστωθεί τι άλλα στοιχεία υπάρχουν που θα μπορούσαν να συνδυαστούν με κάποιους από αυτούς τους κανόνες. Κάθε γεγονός που συμπεραίνεται με αναδρομικό τρόπο ελέγχεται αν υποστηρίζεται από την τράπεζα γνώσης. Οπότε η αναζήτηση τερματίζεται όταν είτε όλα τα γεγονότα στα οποία κατέληξε ο αποδεικνύονται ή δεν μπορούν να βρεθούν υποψήφιες λύσεις. Στα πλεονεκτήματα αυτής της προσέγγισης συμπεριλαμβάνεται το ότι δεν υπάρχουν επιπλέον κόστη προεπεξεργασίας ή επιπλέον απαιτήσεων σε αποθηκευτικό χώρο. Το πιο σημαντικό μειονέκτημα είναι ότι η επεξεργασία γίνεται από την αρχή για κάθε φορά που εκτελείται ένα ερώτημα και για τα σύνθετα ερωτήματα η αναζήτηση μπορεί να είναι υπολογιστικά ακριβή κι αργή.

Η πιο διαδεδομένη γλώσσα για την αναζήτηση σε RDF δεδομένα είναι η SPARQL [4]. Πολύ παρόμοια στο συντακτικό με την SQL, μπορεί να χρησιμοποιηθεί για να εκφράσει ερωτήματα από διαφορετικές πηγές δεδομένων, για παράδειγμα για δεδομένα που αποθηκεύονται ως εγγενώς RDF ή η πρόσβαση σε αυτά γίνεται μέσω κάποιου middleware. Επίσης ορισμένα ερωτήματα περιλαμβάνουν συζεύξεις (conjunctions) και διαζεύξεις (disjunctions) γεγονός που αυξάνει την πολυπλοκότητα.

Συστήματα αποθήκευσης κι επεξεργασίας RDF δεδομένων όπως για παράδειγμα το Hexastore [123], το RDFSuite [21], το 3store [67], το DLDB [92, 93], το KAON [118], είναι κεντρικοποιημένα συστήματα, παρόμοια με τα συστήματα διαχείρισης βάσεων δεδομένων (DBMS), που επιτρέπουν την αποθήκευση, την αναζήτηση, τη διαχείριση και τον συμπερασμό (inference) για RDF γράφους. Οι αρχιτεκτονικές που επιτρέπουν την αποτελεσματική αποθήκευση RDF δεδομένων και επεξεργασία RDF ερωτημάτων περιλαμβάνουν τις τεχνικές *giant triple table*, *property table* [125, 124], *vertical partitioning* [7, 44], καθώς και την εφαρμογή συγκεκριμένων τυπών ευρετηρίων.

Πολύ πρόσφατα έχουν προταθεί κατανεμημένα συστήματα ανάλυσης RDF δεδομένων που βασίζονται σε σύγχρονες τεχνολογίες map-reduce [47, 48]. Η πιο διαδεδομένη υλοποίηση του είναι το hadoop [2] το οποίο αποτελείται από δύο επίπεδα: (i) το επίπεδο αποθήκευσης δεδομένων ή αλλιώς Hadoop Distributed File System (HDFS), και (ii) το επίπεδο επεξεργασίας δεδομένων, γνωστό κι ως το map-reduce framework. Συγκεκριμένα, το HDFS είναι ένα σύστημα αρχείων διαμορφωμένο σε blocks. Τα αρχεία απαρτίζονται από blocks κι αποθηκεύονται στους κόμβους δεδομένων του cluster. Ένας ξεχωριστός κόμβος, name-node, είναι υπεύθυνος για τη διαχείριση των μεταδεδομένων σχετικά με το μέγεθος και την αποθήκευση των blocks. Το δεύτερο επίπεδο ακολουθεί μία παραδοσιακή master-slave αρχιτεκτονική. Κάθε διεργασία αποτελείται από επιμέρους map διεργασίες και reduce διεργασίες που εκτελούνται στους slave κόμβους.

	Storage	Multidimensional	Balancing	Reasoning
Edutella [87]	unstructured	N/A	No	No
SONs [45]	hierarchical	N/A	No	No
RDFPeers [35]	MAAN	No	No	No
Battre et al. [24]	Pastry	No	Yes	FC
Kaoudi et al. [77]	Pastry	No	No	BC
MARVIN [89]	Chord	No	No	FC
Piazza [65]	unstructured	N/A	No	Yes
GridVine [12]	P-Grid	No	Yes	Yes
MIDAS-RDF	MIDAS	Yes	Yes	FC

Πίνακας 2.2: Χαρακτηριστικά διαφορετικών RDF αποθετηρίων.

Το σύστημα hadoopRDF [53] βασίζεται στο hadoop [2] προκειμένου να διαμοιράσει τα στοιχεία της RDF γνώσης δεδομένων στους slave κόμβους του hadoop cluster με την χρήση ενός Sesame εξυπηρετητή για την αποθήκευση και την ανάκτηση των RDF δεδομένων. Στο ίδιο πνεύμα, το H2RDF [95] δημιουργεί μοτίβα πλειάδων βάσει των RDF δεδομένων για να τα αποθηκεύσει σε HBase [3] πίνακες που αναπαριστούν δείκτες ως προς όλους τους συνδυασμούς των στοιχείων των RDF τριπλετών με τρόπο παρόμοιο με το Hexastore [123]. Η δουλειά αυτή επεκτείνεται στο H2RDF+ [94] που χρησιμοποιεί τεχνικές συμπίεσης των δεδομένων ενώ sort-merge αλγόριθμοι χρησιμοποιουνται για την επεξεργασία join λειτουργιών.

2.2.2 Κατανεμημένη Αποθήκευση RDF Δεδομένων και Μέθοδοι Συλλογιστικής

Ο Πίνακας 2.2 επισκοπεί τα χαρακτηριστικά των πιο γνωστών κατανεμημένων συστημάτων αποθήκευσης κι επεξεργασίας RDF καθώς και των μηχανισμών που προτείνουμε που περιλαμβάνονται στο MIDAS-RDF. Το Edutella [87] διατηρεί τη δομή πολλών ανεξάρτητων RDF βάσεων δεδομένων τις οποίες συνδέει μέσω ενός μη δομημένου δίκτυου επικάλυψης το οποίο αποτελείται από απλούς και προνομιούχους κόμβους (super-peers). Όλα τα δεδομένα παραμένουν στην αρχική τους θέση και τα ερωτήματα δρομολογούνται μέσω τεχνικών παρόμοιων με της πλημμυρίδας (flooding). Παρ' όλα αυτά, λόγω του ότι γίνεται χρήση αρκετά απλών και στοιχειωδών τεχνικών, δεν υπάρχει καμία εγγύηση ότι το περιεχόμενο που θα ερωτηθεί τελικά θα καταφέρει να ανακτηθεί ενώ επιπλέον δεν παρέχονται εγγυήσεις για την επεκτασιμότητα (scalability) και την απόδοση του συστήματος. Εν αντιθέσει, το MIDAS-RDF βασίζεται σε ένα κατανεμημένο ευρετήριο με δομημένη αρχιτεκτονική για πολυδιάστατα δεδομένα κι υποστηρίζει την αποτελεσματική επεξεργασία ερωτημάτων εύρους (range queries) παρέχοντας συγκεκριμένες εγγυήσεις ως προς την απόδοση.

Οι Crespo και Molina προτείνουν στο [45] μια ιεραρχία ταξινόμησης, που μοιάζει με οντολογία, και χρησιμεύσει ως βάση για το σχηματισμό του δικτύου. Στα Semantic Overlay Networks (SONs), οι κόμβοι που διαθέτουν σημασιολογικά παρόμοιο περιεχόμενο είναι τοπολογικά συγκεντρωμένοι σχηματίζοντας συνδέσμους μεταξύ τους ενώ ένας κόμβος δύναται να ανήκει σε περισσότερες από μία συστάδες ταυτόχρονα. Τα ερωτήματα υποβάλλονται σε επεξεργασία με τον προσδιορισμό των SONs που διαθέτουν το πιο σχετικό περιεχόμενο. Στη συνέχεια, το ερώτημα στέλνεται στους κόμβους αυτών των SONs που θα το προωθήσουν στα υπόλοιπα μέλη του μέσω multicast πρωτοκόλλων. Από την άλλη, στο MIDAS-RDF δεν απαιτείται κάποια μοντελοποίηση σημασιολογικής εγγύτητας ενώ επιπλέον παρέχονται πολύ αποτελεσματικοί μηχανισμοί δρομολόγησης.

Οι RDFPeers [35] είναι μία από τις πρώτες προσπάθειες RDF αποθετηρίων που στηρίζονται σε δομημένα δίκτυα ομοτίμων. Βασίζονται στο MAAN [36] και αποθηκεύουν την κάθε τριπλέτα στο κατανεμημένο ευρετήριο τρεις φορές, μία φορά με βάση το υποκείμενο, άλλη μία με βάση το κατηγόρημα και μια ακόμα με βάση το αντικείμενο. Υπάρχουν μερικά προφανή μειονεκτήματα σε αυτή την προσέγγισή. Πρώτον, υπάρχει ένας πλεονασμός (redundancy factor) για όλες τις τριπλέτες που ισούται με τρία κι επιφέρει ένα *σημαντικό αποθηκευτικό κόστος* ενώ ταυτόχρονα *δεν προσφέρει κανένα απολύτως όφελος* από πλευράς απόδοσης

ή διαθεσιμότητας (availability) αφού ακόμα κι οι τρεις πλεονασματικές τριπλέτες μπορούν να βρίσκονται στον ίδιο κόμβο. Επιπλέον, η σημασιολογία του RDF γράφου αγνοείται εντελώς κατά τη διάρκεια της δρομολόγησης και της επεξεργασίας των ερωτημάτων. Λόγω των χαρακτηριστικών της υποκείμενης υποδομής αποθήκευσης, το RDFPeers μπορεί στη χειρότερη περίπτωση, π.χ., για ερωτήματα με χαμηλή επιλεκτικότητα, να απαντηθούν σε γραμμικό αριθμό από βήματα (hops) σε σχέση με το μέγεθος του δικτύου. Επίσης, τα RDFPeers είναι επιρρεπή σε ανισορροπίες φορτίου, όταν οι κόμβοι που είναι υπεύθυνοι για δημοφιλείς λέξεις-κλειδιά γρήγορα επιβαρύνονται με μεγάλο αριθμό αιτήσεων. Αμφότερες ιδιαίτερα σημαντικές αδυναμίες τις οποίες θα δείξουμε πως μπορούν να αποφευχθούν σε ένα τέτοιο σύστημα με αποδοτικό και κομψό τρόπο. Σημειώνουμε ότι στο MIDAS-RDF δεν υπάρχουν διπλότυπες τριπλέτες και διατίθονται αποδοτικοί αλγόριθμοι δρομολόγησης με το μέγιστο αριθμό βημάτων να είναι λογαριθμικός ως προς το μέγεθος του δικτύου. Ακόμα, δεν πάσχει από ανισοκατανομές φόρτου επεξεργασίας που να οφείλονται σε δημοφιλείς λέξεις-κλειδιά, ενώ επιπλέον υποστηρίζει αποτελεσματικούς μηχανισμούς συμπερασμού (distributed reasoning).

Όσον αφορά τις μεθόδους κατανεμημένης RDFS συλλογιστικής ειδικότερα, οι Fang et al. [57] προτείνουν μια επαναληπτική forward chaining διαδικασία χωρίς να συμπεριλαμβάνουν τεχνικές αντιμετώπισης εξισορρόπησης φόρτου εργασίας (task-skew). Οι Kaoudi et al. [77] συγκρίνουν δύο γνωστές προσεγγίσεις για RDFS συλλογιστική, backward chaining και forward chaining για κατανεμημένους πίνακες κατακερματισμού, και καταλήγουν στο συμπέρασμα ότι το backward chaining είναι πιο αποτελεσματικό. Συγκριτικά με το έργο αυτό, το MIDAS-RDF διαθέτει μηχανισμούς για επαυξητικές ενημερώσεις (incremental updates), διαγραφές τριπλετών μαζί με όλες τις συναχθείσες τριπλέτες, και εμποδίζει τη συναγωγή διπλότυπων τριπλετών. Οι Battré et al. [24] παρουσιάζουν ένα πρότυπο για την εκτέλεση συλλογισμού πάνω σε τοπικά αποθηκευμένες τρπλέτες και εισάγουν μια πολιτική για την αντιμετώπιση ανισορροπιών φορτίων. Το σύστημα Marvin [89] αποτελεί μια παράλληλη και κατανεμημένη πλατφόρμα για την επεξεργασία μεγάλων RDF δεδομένων σε ένα μη-δομημένο δίκτυο ομοτίμων (unstructured peer-to-peer network) όπου οι συγγραφείς προτείνουν μια επαναληπτική διαδικασία για τον υπολογισμό της μεταβατικής κλειστότητας του συνόλου των δεδομένων. Οι Serafini και Tamilin προτείνουν στο [107] ένα σύστημα που βασίζεται σε χειροκίνητες αντιστοιχίσεις βάσει οντολογιών.

Μια εντελώς διαφορετική προσέγγισή παρουσιάζεται στο [105] όπου κατανέμονται οι κανόνες συλλογισμού αντί για τα δεδομένα έτσι ώστε ο κάθε κόμβος ξεχωριστά να είναι υπεύθυνος για την εκτέλεση ενός συγκεκριμένου μέρους της διαδικασίας συλλογισμού. Ωστόσο, το σύστημα αυτό δεν διαθέτει μηχανισμούς για την αντιμετώπιση ανισορροπιών φορτίου (skew) και συνεπώς δεν επεκτείνεται καλά (scalability). Το Piazza [65] βασίζεται σε αντιστοιχίσεις μεταξύ των επιμέρους τεχνικών προώθησης ερωτημάτων σε σημασιολογικά σχετικούς κόμβους παρά σε ένα κατανεμημένο δείκτη που να βασίζεται σε τεχνικές διαμερισμού του δικτύου. Τέλος, το GridVine [12] υλοποιεί ένα σημασιολογικό δίκτυο διαχωρίζοντας το λογικό επίπεδο (logical layer) από το φυσικό επίπεδο (physical layer) εφαρμόζοντας έτσι την αρχή της ανεξαρτησίας δεδομένων. Το λογικό επίπεδο περιλαμβάνει λειτουργίες για την υποστήριξη σημασιολογικής συμβατότητας και διαλειτουργικότητας, συμπεριλαμβανομένου και της αναζήτησης βάσει χαρακτηριστικών, κληρονομικότητας σχήματος (schema inheritance), διαχείρισης σχήματος (schema management) και αντιστοίχισης σχήματος (schema mapping). Υποστηρίζει επίσης μία τεχνική συμψηφισμού σχήματος, γνωστή κι ως semantic gossiping [11], με σκοπό την αποτελεσματική σημασιολογική διαλειτουργικότητα σε αποκεντρωμένα συστήματα.

2.2.3 Τεχνικές Επισήμανσης

Στη συνέχεια παρέχουμε μια σύντομη επισκόπηση των τεχνικών επισήμανσης (labeling schemes) για το Σημασιολογικό Ιστό (Semantic Web). Για μία λεπτομερή μελέτη για τις υπάρχουσες τεχνολογίες στον συγκεκριμένο τομέα μπορεί κανείς να ανατρέξει στα [41, 40]. Οι τεχνικές αυτές μπορούν να χωριστούν σε δύο βασικές κατηγορίες, σε τεχνικές που λειτουργούν επί τη βάση (i) διαστημάτων (interval-based labels), και (ii) προθεμάτων (prefix-based labels).

Οι Agrawal et al. [17] προτείνουν ένα σύστημα που εκχωρεί διαστήματα στους κόμβους ενός γραφήματος, με τέτοιο τρόπο ώστε η προτεραιότητα μεταξύ δύο κόμβων (γονέας-παιδί) να μπορεί να διαπιστώνεται από τον έλεγχο του συγκεκριμένου διαστήματος. Κατ' αρχάς, θα πρέπει να υπολογιστεί το δέντρο επικάλυψης (spanning tree) T του αρχικού γράφου. Έπειτα, ένας κόμβος u του δέντρου T επισημαίνεται με την ετικέτα $[\texttt{minpost}(u), \texttt{post}(u)]$, όπου $\texttt{post}(u)$ είναι η σειρά του u κατά τη μεταθεματική διάσχιση (postorder traversal) των κόμβων που αποτελούν το spanning tree T, ενώ το $\texttt{minpost}(u)$ αποτελεί το χαμηλότερο post αναγνωριστικό μεταξύ όλων των κόμβων που υπάγονται στον u. Οπότε, όλοι οι κόμβοι του γραφήματος εξετάζονται σε αντίστροφη τοπολογική σειρά και για κάθε ακμή (u,v) όλα τα διαστήματα που σχετίζονται με τον v προωθούνται στην ετικέτα του u. Συνεπώς ένας κόμβος v είναι πρόγονος του u εαν όλα τα διαστήματα του u συμπεριλαμβάνονται σε εκείνα των v. Σημειώστε ότι για να αποφευχθούν οι συχνές επαναταξινομήσεις κι επαναεπισημάνσεις των κόμβων αλλά και για να καθιστάται δυνατή η υποστήριξη σταδιακών ενημερώσεων, αφήνονται επί τούτου κενά στα διαστήματα που δημιουργούνται με διάφορους τρόπους, π.χ. με το να ανατίθονται μη συνεχόμενα αναγνωριστικά στους κόμβους.

Υπάρχουν κι άλλες τεχνικές διαστημάτων που παρουσιάζουν παρόμοιες ιδιότητες. Για παράδειγμα, σύμφωνα με τα [49, 50] ένας κόμβος u του δέντρου επικάλυψης (spanning tree) T επισημαίνεται με $[\texttt{pre}(u), \texttt{post}(u)]$, όπου $\texttt{pre}(u)$ είναι η σειρά του κόμβου u σύμφωνα με μία προθεματική διάσχιση (preorder traversal) του δέντρου T. Ομοίως, ο Τσακαλίδης στο [111] δείχνει ότι ένας κόμβος v είναι πρόγονος του u αν και μόνον αν $\texttt{pre}(v) \leq \texttt{pre}(u) \leq \texttt{post}(v)$, γεγονός που οδηγεί σε μια τεχνική επισήμανσης με ετικέτες μέγιστου μεγέθους $2\log n$. Ο Peleg [96] προτείνει έναν εναλλακτικό τρόπο επισήμανσης με ετικέτες μήκους $O(\log n)$ που δίνονται σε οποιοδήποτε κόμβο u και v ώστε να μπορεί κανείς εξετάζοντάς αυτες να προσδιορίσει με αποδοτικό τρόπο τον ελάχιστο κοινό πρόγονο των κόμβων u και v.

Επιπλέον για τους ιδίους σκοπούς έχουν προταθεί διάφορες τεχνικές [78, 22, 16] που βασίζονται στα προθέματα των ετικετών για να προσδιορίσουν αν κάποιος κόμβος υπάγεται σε κάποιον άλλον ή για να ανακτηθεί ο κοντινότερος γονέας δύο οποιονδήποτε κόμβων, εφόσον υπάρχει. Έστω ένα αλφάβητο $\Sigma = \{\sigma_1, \cdots, \sigma_M\}$. Τότε μπορούμε να ορίσουμε ετικέτες στους κόμβους με επαναληπτικό τρόπο ως εξής: για ένα κόμβο u με ετικέτα $\texttt{id}(u) \in \Sigma^*$, υποθέτουμε μία διάταξη για τους κόμβους που υπάγονται άμεσα στον u κι έστω ο κόμβος v το k-οστό παιδί του u, τότε έχουμε ότι $\texttt{ετικέτα}(v) = \texttt{ετικέτα}(u)\ \sigma_k$. Ένα από τα σημαντικότερα πλεονεκτήματα αυτής της προσέγγισης είναι η δυνατότητά της να χειριστεί στοιχειώδεις ενημερώσεις αποτελεσματικά. Όσο η σειρά μεταξύ των απογόνων δεν είναι σημαντική μπορεί κανείς πάντα να προσθέτει ένα ακόμα νέο παιδί χωρίς να χρειαστεί να ανατεθούν ξανά ετικέτες για οποιονδήποτε προ-υπάρχων κόμβο. Σημειώστε ότι για να καθίσταται δυνατή η διαχείριση της γενικής περίπτωσης γραφημάτων ένας κόμβος επιτρέπεται να έχει πολλαπλές ετικέτες και συγκεκριμένα μία για κάθε γονέα. Ο έλεγχος για το εαν ένας κόμβος v είναι ένας πρόγονος του u είναι ισοδύναμος με τον έλεγχο για το εαν η ετικέτα του κόμβου v αποτελεί πρόθεμα της ετικέτας του u. Επιπλέον, για δύο οποιουσδήποτε κόμβους u και w, ο πλησιέστερος κοινός πρόγονος τους είναι ο κόμβος ο οποίος είναι επισημασμένος με το μακρύτερο κοινό πρόθεμα των u, w, κάτι το οποίο μπορεί εύκολα να υπολογιστεί σε χρόνο $O(\min\{|\texttt{ετικέτα}(u)|, |\texttt{ετικέτα}(w)|\})$, όπου $|\texttt{ετικέτα}(u)|$ συμβολίζει το μήκος του αναγνωριστικού του κόμβου u.

2.3 Top-k Ερωτήματα

Η επεξεργασία των top-k ερωτημάτων περιλαμβάνει την εύρεση k πλειάδων οι οποίες είναι καλύτερες σύμφωνα με κάποια συνάρτηση κατάταξης (ranking function). Διακρίνουμε δύο παραλλαγές της κατανεμημένης εκδοχής του προβλήματος. Με την *κάθετη διαμέριση* (vertical partitioning) ένας κόμβος διατηρεί όλες τις τιμές ενός μόνο χαρακτηριστικού (attribute) ολόκληρης της σχέσης. Στη πρώτη προσέγγισή [55] για την αντιμετώπιση αυτού του προβλήματος εισήχθη ο περίφημος Threshold αλγόριθμος (TA) κι ο αλγόριθμος του Fagin (FA). Μεταγενέστερες εργασίες προσπαθούν να βρουν καλύτερες λύσεις: ο Three-Phase Uniform Threshold [37] (TPUT) αλγόριθμος βελτιώνει τους περιορισμούς του TA ουσιαστικά. Αργότερα, ο TPUT βελτιώθηκε από τον KLEE [84], ο οποίος υποστηρίζει ακόμα προσεγγιστική επεξεργασία

top-k ερωτημάτων. Ο συγκεκριμένος προτάθηκε με δύο παραλλαγές, η πρώτη απαιτεί τρεις φάσεις, ενώ η δεύτερη δουλεύει σε δύο κυκλικές διαδρομές.

Η δεύτερη προσέγγιση βασίζεται στην *οριζόντια κατανομή* (horizontal partitioning) των δεδομένων, κατηγορία στην οποία υπάγεται κι η δική μας δουλειά. Εδώ ο κάθε κόμβος του δικτύου διατηρεί μόνο ένα υποσύνολο όλων των πλειάδων για όλες όμως τις στήλες της αρχικής σχέσης. Επιπλέον, υπάρχει αρκετή σχετική βιβλιογραφία για αδόμητα δίκτυα ομοτίμων. Ένας αλγόριθμος "πλημμυρίδας" που ακολουθείται από μια φάση συγχώνευσης προτείνεται στο [19]. Σε μία διαφορετική προσέγγισή που προτείνεται στο [23], προνομιούχοι κόμβοι (super-peers) επιβαρύνονται με την επεξεργασία top-k ερωτημάτων, μια προσέγγιση που προκαλεί μεγάλες ανισοκατανομές φορτίου. Στο SPEERTO [117] κάθε κόμβος υπολογίζει το σύνολο εγγραφών που απαντούν στο τοπικό k-skyband[2], ως ένα βήμα προ-επεξεργασίας. Στη συνέχεια, κάθε προνομιούχος κόμβος συναθροίζει τα k-skyband σύνολα των κόμβων και τα συνδυάζει με κατάλληλο τρόπο για να απαντάνε στα εισερχόμενα ερωτήματα. Οι λύσεις που προτείνονται στα BRANCA [136] και ARTO [103] αποθηκεύουν σε προσωρινές μνήμες προηγούμενα τελικα και ενδιάμεσα αποτελέσματα για την αποφυγή επαναυπολογισμού μέρους των νέων ερωτημάτων. Η δική μας μέθοδος είναι η πρώτη που επεξεργάζεται οριζοντίως κατανεμημένα δεδομένα που βρίσκονται αποθηκευμένα σε δομημένα δίκτυα ομοτίμων.

2.4 Ερωτήματα Κορυφογραμμής

Τα ερωτήματα κορυφογραμμής όπως ορίζονται στο [34] ανακτούν τις πλειάδες για τις οποίες δεν υπάρχει καμία πλειάδα που να είναι καλύτερη σε οποιαδήποτε διάσταση. Μια εξαιρετική επισκόπηση για τον κατανεμημένο υπολογισμό ερωτημάτων κορυφογραμμής αποτελεί το [68] όπου οι μέθοδοι τόσο για δομημένα όσο και για αδόμητα δίκτυα μελετούνται λεπτομερώς. Πιο ειδικά για τα δομημένα δίκτυα, το DSL [127] στηρίζεται στο CAN [99] για την δεικτοδότηση πολυδιάστατων δεδομένων. Για την επεξεργασία ερωτημάτων δημιουργείται μία ιεραρχία στην οποία ο κόμβος που είναι υπεύθυνος για την περιοχή που περιέχει την κάτω αριστερή γωνία της περιοχής που αντιπροσωπεύει το ερώτημα είναι η ρίζα. Η ιεραρχία χτίζεται με τέτοιο τρόπο όπου οι κόμβοι των οποίων οι εγγραφές δεν μπορούν να είναι καλύτερες από κάποιου άλλου λαμβάνουν ταυτόχρονα το ερώτημα. Ένας ομότιμος κόμβος που λαμβάνει ένα ερώτημα υπολογίζει την δική του κορυφογραμμή από τις εγγραφές οι οποίες σχετίζονται με την περιοχή για την οποία είναι υπεύθυνος. Αφού δεχθεί τα τοπικά σύνολα κορυφογραμμών από όλους τους γειτονικούς κόμβους που προηγούνται στην ιεραρχία, τα συμψηφίζει υπολογίζοντας την κορυφογραμμή που ορίζεται με βάση τόσο τα τοπικά δεδομένα όσο και τα δεδομένα που έλαβε από τους γείτονές του. Στη συνέχεια, η κορυφογραμμή αυτή διαβιβάζεται στους κόμβους που είναι υπεύθυνοι για γειτονικές περιοχές με τέτοιο τρόπο που να φθάνει παράλληλα μόνο στους κόμβους οι οποίοι δεν μπορούν να κυριαρχήσουν ο ένας ως προς τον άλλο όσον αφορά τα δεδομένα τους. Εξάλλου κανένας από τους υπόλοιπους γείτονές δεν μπορεί να συμβάλλει στο αποτέλεσμα.

Το SSP [120] (Skyline Space Partitioning) αποτελεί μία μέθοδο που προορίζεται για κατανεμημένη επεξεργασία ερωτημάτων κορυφογραμμής στηριζόμενο στη δικτυακή υποδομή του BATON [74]. Ο πολυδιάστατος χώρος δεδομένων αντιστοιχίζεται σε μονοδιάστατα κλειδιά χρησιμοποιώντας μία z-καμπύλη (z-curve), λόγω του περιορισμένου αριθμού διαστάσεων που υποστηρίζει το BATON. Η επεξεργασία των ερωτημάτων ξεκινάει από τον κόμβο που είναι υπεύθυνος για την περιοχή που περιέχει την αρχή των αξόνων (for uncostrained skyline queries). Ο κόμβος αυτός υπολογίζει τα σημεία που έχουν αποθηκευτεί τοπικά σε αυτόν και ταυτόχρονα βρίσκονται εκ κατασκευής στην συνολική κορυφογραμμή. Στην συνέχεια, επιλέγει από αυτά το πιο κυρίαρχο σημείο και το χρησιμοποιεί για να περιορίσει το χώρο αναζήτησης και να αποκλείσει από τη διαδίκασία όσον τον δυνατόν περισσότερους μη σχετικούς με το ερώτημα κόμβους. Τέλος, το ερώτημα προωθείται στους σχετικούς κόμβους που επαναλαμβάνουν τη διαδικασία για να προσθέσουν στο αποτέλεσμα τις δικές τους πλειάδες έως ότου να μην υπάρχουν άλλοι κόμβοι που να μπορούν να συνεισφέρουν στην απάντηση, κι άρα ο αρχικός κόμβος θα έχει συγκεντρώσει όλες τις ενδιάμεσες κορυφογραμμές για να

[2] τύπος ερωτήματος που επιστρέφει τα σημεία που είναι "χειρότερα" από μέχρι $k-1$ άλλα.

τις συνδυάσει και να εξαιρέσει τα false positives προτού επιστρέψει την τελική απάντηση στον κόμβο που εξέδωσε αρχικά το ερώτημα. Το Skyframe [121] προτάθηκε για να εφαρμοστεί στο BATON [74] και στο CAN [99]. Στο Skyframe τα ερωτήματα προωθούνται προς ένα σύνολο κόμβων όπου ο καθένας είναι υπεύθυνος για μια περιοχή που εμπεριέχεται μία ελάχιστη τιμή σε τουλάχιστον μία διάσταση. Σε κάθε βήμα ο κόμβος που εξέδωσε το ερώτημα λαμβάνει τα τοπικά ενδιάμεσα αποτελέσματα και τα επεξεργάζεται ώστε να καθορίσει εάν υπάρχουν ακόμα κόμβοι που θα πρέπει να ερωτηθούν ώστε να ζητήσει από αυτούς τα σημεία που απαρτίζουν τις τοπικές κορυφογραμμές. Η διαδικασία αυτή συνεχίζεται έως ότου δεν υπάρχουν άλλοι κόμβοι να ερωτηθούν κι άρα ο αρχικός κόμβος υπολογίζει την τελική κορυφογραμμή από τα ενδιάμεσα τοπικά αποτελέσματα.

2.5 Διαφοροποιημένα Αποτελέσματα

Αυτή η ενότητα είναι οργανωμένη ως εξής: Το πρώτο μέρος της περιγράφει διάφορους ορισμούς για το πρόβλημα της διαφοροποίησης (search result diversification) που προτείνονται στη βιβλιογραφία, ενώ το δεύτερο συγκρίνει και ταξινομεί αλγορίθμους που προτείνονται για τον εντοπισμό διαφορετικών (diverse) αντικειμένων. Μια λεπτομερέστερη έρευνα σχετικά με το state of the art μπορεί να βρεθεί στο [52].

Οι μέχρι σήμερα προσεγγίσεις που ακολουθούν οι σχετικές εργασίες ορίζουν γενικά τρεις (μερικές φορές αλληλοεπικαλυπτόμενες) βασικές κατηγορίες διαφοροποίησης αποτελέσματος βάσει συγκεκριμένων χαρακτηριστικών: (i) *περιεχόμενο* (content-based definitions), για διαφοροποιημένα στοιχεία που προκύπτουν βάσει των τιμών που παίρνουν για κάθε διάσταση, (ii) *καινοτομία* (nobelty-based), για την προώθηση αντικειμένων που εμπεριέχουν νέα πληροφορία σε σχέση με εκείνα που προηγούνται, και (iii) *κάλυψη* (coverage-based), που σκοπεύει στο να συμπεριληφθούν τέτοια στοιχεία ώστε να καλύφθούν όσες περισσότερες κατηγορίες και θεματικές περιοχές είναι δυνατόν.

Κατ' αρχάς, οι μέθοδοι που λειτουργούν βάσει του περιεχομένου στοχεύουν στη μεγιστοποίηση της ελάχιστης απόστασης μεταξύ κάθε ζεύγους αντικειμένων στο αποτέλεσμα. Η πλέον σημαντική εργασία είναι πιθανότατα η **M**aximal **M**arginal **R**anking (MMR) [38] όπου οι Carbonell και Goldstein εισάγουν έναν συμβιβασμό (trade-off) μεταξύ διαφορετικότητας και συνάφειας των αποτελεσμάτων αναζήτησης μέσω του συνδυασμού των δύο συναρτήσεων, όπως φαίνεται στην Εξίσωση 2.1, με τον έναν παράγοντα να υπολογίζει την ομοιότητα μεταξύ των εγγράφων ενώ ο άλλος την ομοιότητα μεταξύ του κάθε εγγράφου και του ερωτήματος. Επιπλέον, μια παράμετρος λ ελέγχει το βαθμό του trade-off για να εξισορροπεί τους δύο παράγοντες.

$$\min_S f(S|\vec{q}) = \min_S(\max_{\vec{x}\in S} \lambda d(\vec{q},\vec{x}) - (1-\lambda) \min_{\vec{x},\vec{y}\in S} d(\vec{x},\vec{y})) \tag{2.1}$$

Υπό το πρίσμα αυτό η [62] είναι μια ακόμα επιφανής εργασία που ακολουθεί μια αξιωματική προσέγγιση κι αναπτύσει μια σειρά από αξιώματα τα οποία ένας τέτοιος μηχανισμός θα πρέπει να ακολουθεί. Προτείνει μία ποικιλία μεθοδολογιών όπως η διαφοροποίηση επί τη βάσει είτε της σημασιολογικής απόστασης των στοιχείων είτε της κατηγορικής απόστασης (categorical distance) βάσει ταξονομιών, καθώς επίσης αναλύονται τα πλεονεκτήματα και τα μειονεκτήματα της κάθε τεχνικής. Από την άλλη, οι συγγραφείς στο [115] επικεντρώνονται σε δομημένα δεδομένα για να προτείνουν μια κατάλληλη συνάρτηση ομοιότητας για μία δεδομένη διάταξη ή για τη σειρά των χαρακτηριστικών (attributes). Μια ενδιαφέρουσα προσέγγιση παρουσιάζεται στο [66], όπου επεκτείνεται το πρόβλημα των k πλησιέστερων γειτόνων (k nearest neighbors) με τον ορισμό ενός ελάχιστου ορίου απόστασης μεταξύ των στοιχείων του συνόλου του αποτελέσματος προκειμένου να διασφαλιστεί η ποικιλομορφία.

Σε μια πιο πρόσφατη εργασία [58], οι συγγραφείς υιοθετούν μια προσέγγιση βασισμένη σε Voronoi κελύφη (cells) προκειμένου να ανακτήσει τα αντικείμενα που μεγιστοποιούν μία συνάρτηση καινοτομίας (novelty function). Πιο συγκεκριμένα, τα στοιχεία που αποτελούν ήδη την απάντηση ορίζονται ως τα κέντρα των κελυφών. Στη συνέχεια, όλα τα αντικείμενα που τυχαίνει να τοποθετούνται επί των κορυφών και των ακμών του Voronoi διαγράμματος

εξετάζονται και τα καλύτερα από αυτά εισάγονται στην απάντηση. Μια παρόμοια προσέγγιση προτείνεται στο [116] που όμως απαιτεί να εξεταστούν όλα τα αντικείμενα προτού υπολογιστεί η συνάρτηση στόχος.

Η καινοτομία (novelty) μπορεί να θεωρηθεί ως ένα μέσο για την αποφυγή πλεονασμών και διπλοτύπων καθώς προσδίδει επιπλέον πληροφορία στο αποτέλεσμα ενώ η διαφορετικότητα (diversity) μπορεί να θεωρηθεί ως η ανάγκη για την επίλυση ασαφειών (disambiguation). Εμμένοντας στην αρχή αυτή, τα ερωτήματα και τα αντικείμενα αντιμετωπίζονται ως σύνολα από "ψήγματα πληροφορίας" (information nuggets) στο [43], ενώ η σχετικότητα είναι μία συνάρτηση των nuggets που ποσοτικοποιεί το κατα πόσο αυτά εμπεριέχονται σε πιθανές ερμηνείες του ερωτήματος και του αποτελέσματος (κι εμμέσως των προθέσεων του χρήστη). Στο [134] εξουσιοδοτούνται προσαρμοστικά συστήματα φιλτραρίσματος (adaptive filtering systems) προκειμένου να ξεχωρίσουν τα "καινοτόμα" αντικείμενα από τα πλεονασματικά. Τέτοια συστήματα ταυτοποιούν τα στοιχεία που είναι παρόμοια με αυτά που έχουν προηγουμένως επιλεχθεί για το ίδιο θέμα αλλά ταυτόχρονα είναι ανόμοια με αυτά επειδή εμπεριέχουν επιπρόσθετη πληροφορία.

Από την άλλη πλευρά, στο [128] υποδηλώνεται από την πλευρά του χρήστη η άποψη ότι η αναζήτηση για καινούρια πληροφορία δεν είναι ισοδύναμη με την αναζήτηση για ποικιλομορφία στο αποτέλεσμα καθώς οι προτιμήσεις των μεμονωμένων χρηστών κατευθύνονται προς την εξεύρεση περισσότερης πληροφορίας σχετικά με συγκεκριμένα επιμέρους θέματα ενδιαφέροντος, παρά σε μία τυφλή, μη-κατευθυνόμενη αναζήτηση για κάθε νέα πληροφορία. Στο [18] η προσέγγιση αυτή επεκτείνεται λαμβάνοντας υπόψη τη σχετική σημασία των διαφόρων στα nuggets (ως κατανομή σε κατηγορίες), καθώς και το γεγονός ότι διαφορετικά έγγραφα τα οποία εμπεριέχουν το ίδιο nugget μπορούν να ικανοποιήσουν το χρήστη σε διαφορετικό βαθμό. Παρά το γεγονός ότι οι περισσότερες εργασίες κάνουν στην καλύτερη περίπτωση μόνο μια έμμεση χρήση των θεμάτων των εγγράφων, στο [132] προτείνονται μοντέλα που χρησιμοποιούνται στην παραδοσιακή ανάκτηση αλλά επιδιώκεται να τροποποιηθεί η κατάταξη (ranking) έτσι ώστε να συμπεριληφθούν έγγραφα που να σχετίζονται με διάφορα επιμέρους υποθέματα ως προς κάποια ταξονομία (taxonomy).

Επιπλέον, το πρόβλημα της διαφοροποίησης έχει δειχθεί ότι είναι NP-hard εν γένει, δεδομένου ότι συσχετίζεται άμεσα με το πρόβλημα p-dispersion. Ο στόχος του προβλήματος είναι να επιλεγούν p από n δοθέντα σημεία έτσι ώστε η ελάχιστη απόσταση μεταξύ οποιουδήποτε ζεύγους σημείων να είναι μέγιστο. Αρκετά συχνά όμως η αντικειμενική συνάρτηση που μεγιστοποιείται είναι η μέση απόσταση μεταξύ δύο σημείων. Εξετάζονται στο [62] διαφορετικές συνθέσεις του προβλήματος, όπως η *min-max διαφοροποίηση* (min-max diversification), *max-sum διαφοροποίηση* (max-sum diversification) και μία *mono-objective* διατύπωση του προβλήματος. Επιπρόσθετα, ο συνδυασμός αυτών των κριτηρίων έχει μελετηθεί στο [133] ως πρόβλημα βελτιστοποίησης (optimization).

Λόγω της αυξημένης πολυπλοκότητας του προβλήματος, οι περισσότερες προσπάθειες βασίζονται σε προσεγγιστικές λύσεις (approximation schemes). Πολλοί ευριστικοί αλγόριθμοι έχουν προταθεί κι έχουν χρησιμοποιηθεί για την επίλυση παραλλαγών του προβλήματος. Οι ευριστικές αυτές μπορούν να ταξινομηθούν σε δύο κατηγορίες: (i) άπληστη (greedy heuristics) και (ii) εναλλαγής (interchange heuristics) ή αναρρίχηση λόφων (hill climbing). Η πλειοψηφία των προτεινόμενων αλγορίθμων υιοθετεί μια άπληστη προσέγγιση της οποίας η λειτουργία κατανοείται εύκολα διαισθητικά ενώ ταυτόχρονα είναι αποτελεσματική και γρήγορη. Μία απλή μεθοδολογία προτείνεται στο [18] σύμφωνα με την οποία δεδομένου ενός συνόλου με τα k πιο σχετικά έγγραφα σε ένα ερώτημα, αυτά αναδιατάσονται εκ νέου με τέτοιο τρόπο ώστε η συνάρτηση στόχος του μοντέλου να μεγιστοποιείται. Στο πλαίσιο των συστημάτων συστάσεων (recommender systems), το [137] παρουσιάζει μια άπληστη τεχνική, σύμφωνα με την οποία το αποτέλεσμα ξεκινά με την πιο σχετική σύσταση (recommendation). Στη συνέχεια, οι προτάσεις που δεν αποτελούν μέρος της απάντησης ταξινομούνται ως προς τη συνάφειά τους με το ερώτημα και την ελάχιστη απόσταση τους από οποιοδήποτε στοιχείο της απάντησης, με τη θέση της κάθε σύστασης να αποτελείται από έναν γραμμικό συνδυασμό της θέσης της στις δύο λίστες. Έτσι σε κάθε επανάληψη, προσθέτουμε στο αποτέλεσμα τη σύσταση με την καλύτερη θέση έως ότου η απάντηση να αποτελείται από k συστάσεις. Μια άλλη παρόμοια προσέγγιση που προτείνεται στο [130] ξεκινάει με το πιο σχετικό αντικείμενο

και προσθέτει το επόμενο πιο σχετικό αν και μόνον αν το στοιχείο αυτό είναι αρκετά μακριά από τα έως τώρα ανακτηθέντα στοιχεία της απάντησης (σε σύγκριση με ένα συγκεκριμένο όριο) έως ότου υπάρχουν k στοιχεία στην απάντηση. Ένας παρόμοιος αλγόριθμος προτείνεται [66] στο πλαίσιο των χωρικών βάσεων δεδομένων.

Από την άλλη πλευρά, οι ευριστικές τεχνικές *εναλλαγής* (interchange heuristics) αρχίζουν από ένα τυχαίο σύνολο και στη συνέχεια επαναληπτικά βελτιώνουν το αποτέλεσμα με την εναλλαγή ενός στοιχείου με ένα άλλο που δεν ήταν προηγουμένως μέρος της απάντησης. Σε κάθε επανάληψη, εξετάζονται για την ανταλλαγή τα σημεία που βελτιώνουν την ποικιλομορφία της απάντησης σε μεγαλύτερο βαθμό. Στο πλαίσιο των (ημι)δομημένων δεδομένων προτείνεται μία παρόμοια λύση στο [81]. Μια ακόμα ευριστική παρουσιάζεται στο [130], όπου αρχίζοντας με τα k σχετικότερα στοιχεία σε κάθε επανάληψη το στοιχείο που συμβάλλει λιγότερο στην ποικιλομορφία του αποτελέσματος υποκαθιστάται με το πλέον υποσχόμενο στοιχείο που δεν ήταν πριν μέρος της απάντησης. Αυτή η επαναληπτική διαδικασία τερματίζεται όταν δεν υπάρχουν στοιχεία με ποικιλότητα (diversity) μεγαλύτερη από ένα συγκεκριμένο όριο.

Κεφάλαιο 3

Η Αρχιτεκτονική του MIDAS

If you cannot make it good,
at least make it look good.

Abhi Sharma

Το παρόν κεφάλαιο παρουσιάζει τη δομή και περιγράφει τις βασικές λειτουργίες του MIDAS καθώς και τη πληροφορία που είναι αποθηκευμένη σε κάθε κόμβο. Ειδικότερα, η Ενότητα 3.1 περιγράφει τη δομή του κατανεμημένου ευρετηρίου και πως χρησιμοποιείται για το σχηματισμό του δικτύου. Η Ενότητα 3.2 περιγράφει αναλυτικά τις πληροφορίες που αποθηκεύονται τοπικά σε κάθε κόμβο του δικτύου MIDAS καθώς και πως γίνεται χρήση αυτής για την αποτελεσματική δρομολόγηση των μηνυμάτων. Οι Ενότητες 3.3, 3.4 και 3.4.1 αναλύουν τις λειτουργίες εισαγωγής κόμβων, αποχώρησης και απρόσμενης απώλειας κόμβων στο MIDAS.

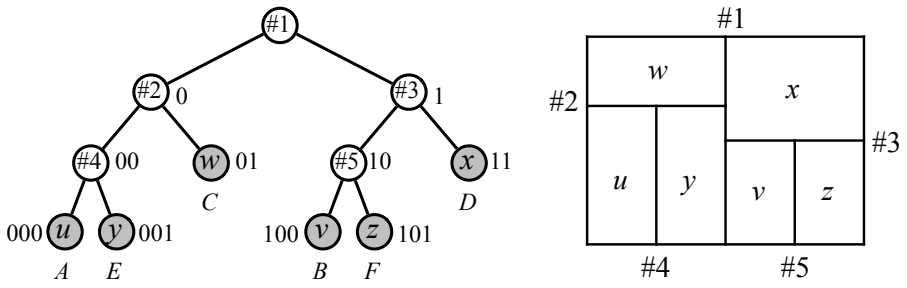
3.1 Η Δομή του MIDAS

Το κατανεμημένο ευρετήριο MIDAS ακολουθεί το παράδειγμα του προσαρμοστικού k-d δέντρου (adaptive k-d tree) [27] προκειμένου να δεικτοδοτήσει τον πολυδιάστατο χώρο. Ας αναλογιστούμε ένα μετρικό χώρο d διαστάσεων $I = [\vec{\ell}_I, \vec{h}_I]$, που ορίζεται από το κάτω όριο $\vec{\ell}_I$ και άνω όριο $\vec{h}_I$ και μπορεί να παρομοιαστεί με ένα διάστημα d διαστάσεων. Το k-d δέντρο είναι ένα δυαδικό δέντρο κατά βάσει, στο οποίο κάθε κόμβος $T[i]$ αντιστοιχεί σε έναν παραλληλόγραμμο πολλών διαστάσεων I_i που σχηματίζεται παράλληλα με τους άξονες. Η ρίζα $T[1]$ αντιστοιχεί σε ολόκληρο το χώρο, δηλαδή ισχύει ότι $I_1 = I$. Κάθε εσωτερικός κόμβος $T[i]$ έχει πάντα δύο παιδιά, τους κόμβους $T[2i]$ και $T[2i+1]$, των οποίων τα ορθογώνια προέρχονται από τη διάσπαση του υποχώρου I_i κατά μήκος κάποιας διάστασης d_i σε κάποια συγκεκριμένη τιμή s_i που περιλαμβάνεται στο συγκεκριμένο χώρο. Τα κριτήρια διάσπασης των χώρων (οι τιμές των s_i και d_i) επεξηγούνται στην Ενότητα 3.3 . Σημειώστε ότι d_i αντιπροσωπεύει τη διάσταση στην οποία διασπάται ο κόμβος $T[i]$ και όχι στην i-ιοστή διάσταση.

Ας εξετάσουμε για παράδειγμα τον κόμβο $T[i]$ και τους θυγατρικούς του κόμβους, $T[2i]$ και $T[2i+1]$ αλλά και τις αντίστοιχες πολυδιάστατες ζώνες που τους αντιστοιχούν όπου $I_{2i} = [\vec{\ell}_{2i}, \vec{h}_{2i}]$ και $I_{2i+1} = [\vec{\ell}_{2i+1}, \vec{h}_{2i+1}]$. Αν υποθέσουμε ότι το αριστερό παιδί είναι ο κόμβος $T[2i]$ και ότι του αποδίδεται το κάτω τμήμα του I_i, τότε ισχύει ότι (i) $\vec{\ell}_{2i}[d_j] = \vec{\ell}_{2i+1}[d_j]$ κι ότι $\vec{h}_{2i}[d_j] = \vec{h}_{2i+1}[d_j]$ για κάθε διάσταση $d_j \neq d_i$, κι επιπλέον (ii) $\vec{h}_{2i}[d_i] = \vec{\ell}_{2i+1}[d_i] = s_i$ για τη διάσταση d_i. Γράφουμε λοιπόν $I_{2i} \uplus^{d_i} I_{2i+1}$ για να υποδηλώσουμε ότι οι παραπάνω ιδιότητες ισχύουν για τις εκάστοτε διαμερίσεις.

Κάθε κόμβος λοιπόν του k-d δέντρου συνδέεται με ένα δυαδικό αναγνωριστικό το οποίο μπορεί να εξαχθεί από τη διαδρομή από τη ρίζα ως τον κόμβο κι ορίζεται αναδρομικά για κάθε κόμβο. Η ρίζα ταυτίζεται με το άδειο αναγνωριστικό $\varnothing$ ενώ αριστερά (κι αντίστοιχα δεξιά) το παιδί ενός εσωτερικού κόμβου ταυτίζεται με το αναγνωριστικό του πατρικού του

κόμβου επαυξημένο με το δυφίο (bit) 0 (ή με 1 αντίστοιχα). Στην Εικόνα 3.1(α΄) απεικονίζεται ένα k-d δέντρο με έντεκα κόμβους που προέρχονται από πέντε διαδοχικές διασπάσεις του αρχικού χώρου. Δίπλα σε κάθε κόμβο παρατίθεται το αντίστοιχο αναγνωριστικό. Λόγω των ιεραρχικών διασπάσεων, τα τετράγωνα των κόμβων σε οποιοδήποτε επίπεδο του k-d δέντρου αποτελούν μια μη-επικαλυπτόμενη διαμέριση του συνόλου του χώρου I. Στην Εικόνα 3.1(β΄) φαίνονται οι περιοχές που αντιστοιχούν στα φύλλα του Σχήματος 3.1(α΄) όπου οι διασπάσεις είναι αριθμημένες δίπλα στις αντίστοιχες τομές παράλληλα στους άξονες.



(α΄) Το k-d δέντρο (β΄) Παραλλ/μμες περιοχές φύλλων

Σχήμα 3.1: Παράδειγμα ενός δισδιάστατου k-d δέντρου έξι κόμβων.

Κατά σύμβαση θεωρούμε τις πλειάδες ως ζευγάρια κλειδιών-τιμών, όπου τα *κλειδία* είναι d διαστάσεων. Έτσι, ένα κλειδί αντιπροσωπεύεται με ένα διάνυσμα στον d-διάστατο χώρο I που ευρετηριάζει το k-d δέντρο. Ένα φύλλο ενός k-d δέντρου αποθηκεύει όλες τις πλειάδες των οποίων τα κλειδιά αντιστοιχούν στην παραλληλόγραμμη περιοχή που του έχει ανατεθεί. Η ιεραρχική δομή του k-d δέντρου επιτρέπει αποτελεσματικές μεθόδους για την επεξεργασία ερωτημάτων, όπως για ερωτήματα εύρους (range queries) και πλησιέστερου γείτονα (nearest neighbor queries).

3.2 Δρομολόγηση στο MIDAS

Ένας ομότιμος κόμβος (peer) στο MIDAS αντιστοιχεί σε ένα φύλλο του k-d δέντρου, κι αποθηκεύει τοπικά όλα τα ζεύγη κλειδιών-τιμών των οποίου τα κλειδιά εμπεριέχονται στο πολυδιάστατο διάστημα του συγκεκριμένου φύλλου που αντιστοιχεί στη *ζώνη* του κόμβου. Τονίζουμε ότι οι εσωτερικοί κόμβοι στο k-d δέντρο, ήτοι οι μη σκιασμένοι κόμβοι στην Εικόνα 3.1(α΄) δεν αντιστοιχούν σε ομότιμους κόμβους αλλά αποτελούν κατά σύμβαση κατευθυντήριες οδηγίες για την αποτελεσματική δρομολόγηση των μηνυμάτων.

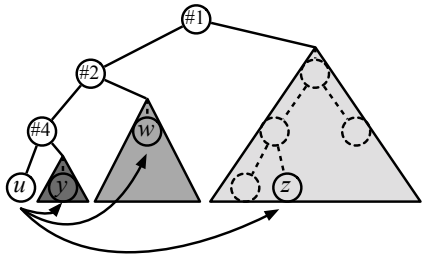
Λήμμα 1 *Για κάθε σημείο του χώρου I, υπάρχει ακριβώς ένας κόμβος στο MIDAS υπεύθυνος γι᾽ αυτό.*

Απόδειξη 1 *Κάθε ομότιμος κόμβος αντιστοιχεί σε ένα φύλλο του k-d δέντρου. Το λήμμα ισχύει επειδή τα φύλλα αποτελούν μία μη επικαλυπτόμενη διαμέριση του συνόλου του χώρου I.* ■

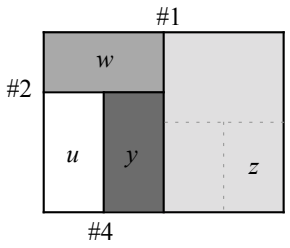
Ένας ομότιμος κόμβος u εμπεριέχει μόνο μερική πληροφορία σχετικά με την δομή του κατανεμημένου k-d δέντρου, η οποία όμως είναι επαρκής για να γίνει επεξεργασία πολύπλοκων ερωτημάτων που περιγράφονται εκτενώς στα επόμενα κεφάλαια. Ειδικότερα, ένας κόμβος u κρατάει τοπικά την εξής πληροφορία για το δίκτυο: (i) Τον πίνακα $u.id$ από δυφία (bitset) που αντιπροσωπεύει το αναγνωριστικό του φύλλου με $u.id[i]$ το i-στό πιο σημαντικό δυφίο. (ii) Η μεταβλητή $u.depth$ αντιστοιχεί στο βάθος του φύλλου στο k-d δέντρο, ή ισοδύναμα, στον αριθμό των δυφίων (bits) στο $u.id$. (iii) Στον πίνακα $u.sdim$ μήκους $u.depth$ στο στοιχείο $u.sdim[i]$ αντιστοιχεί η διάσταση που διασπάστηκε ο χώρος I_{i-1} για τον οποίον είναι υπεύθυνος ο γονέας του i-στού κόμβου στο μονοπάτι από τη ρίζα του δέντρου προς τον κόμβο u του δικτύου. (iv) Στον πίνακα $u.split$ μήκους $u.depth$ βρίσκεται στην i-οστή θέση η

τιμή στην οποία διασπάται η ζώνη του i-οστού κόμβου στο μονοπάτι από τη ρίζα προς τον κόμβο u στην διάσταση $u.split[i]$. (v) Ο πίνακας $u.link$ αποτελεί μια σειρά μήκους $u.depth$ που αντιστοιχεί στον πίνακα δρομολόγησης του κόμβου u, δηλαδή περιέχει την απαραίτητη πληροφορία (διεύθυνση IP, port number) ώστε να μπορεί να προωθεί μηνύματα στον αντίστοιχο κόμβο. (vi) Η λίστα $u.backlink$ εμπεριέχει όλους τους κόμβους οι οποίοι έχουν τον κόμβο u στη $link$ δομή τους κι άρα ο κόμβος u δέχεται κίνηση κι εξυπηρετεί όλους τους κόμβους της λίστας αυτής.

Στη συνέχεια εξηγούμε το περιεχόμενο του πίνακα $u.link$ που αποτελεί τον πίνακα δρομολόγησης του κόμβου u. Εξετάζοντας όλα τα δυνατά προθέματα του αναγνωριστικού του κόμβου u παρατηρούμε ότι υπάρχουν ακριβώς $u.depth$ από αυτά. Ειδικότερα, το κάθε πρόθεμα αντιστοιχεί σε ένα υποδέντρο του k-d δέντρου που εμπεριέχει το φύλλο του δέντρου που αντιστοιχεί στον ομότιμο κόμβο u (το φύλλο που έχει ως αναγνωριστικό το $u.id$) και προσδιορίζει μοναδικά έναν κόμβο. Στο παράδειγμα του Σχήματος 3.1(α΄), το αναγνωριστικό 000 έχει τρία δυνατά προθέματα: 0, 00 και 000, που αντιστοιχούν στα υποδέντρα με ρίζες τους εσωτερικούς κόμβους του k-d δέντρου με τα αντίστοιχα αναγνωριστικά. Αν για κάθε ένα από αυτά τα προθέματα αντιστρέψουμε το λιγότερο σημαντικό δυφίο τότε έχουμε το αναγνωριστικό που αντιστοιχεί στη ρίζα του συμμετρικού υποδέντρου για το οποίο δεν υπάρχει μεγαλύτερο υποδέντρο που να μην εμπεριέχει τον κόμβο u. Στο Σχήμα 3.2(α΄) δείχνουμε τα μέγιστα συμμετρικά υποδέντρα για τον κόμβο με $u.id = 000$, τα οποία μάλιστα έχουν τις ρίζες τους στους κόμβους 1, 01 και 001, τα οποία επισημαίνονται ως σκιασμένα τρίγωνα.



(α΄) Συμμετρικά υποδέντρα του κόμβου u ως προς το μονοπάτι του προς τη ρίζα του k-d δέντρου.



(β΄) Περιοχές που αντιπροσωπεύουν οι σύνδεσμοι του κόμβου u

Σχήμα 3.2: Σχηματική αναπαράσταση του πίνακα δρομολόγησής του κόμβου u.

Για κάθε ένα δυνατό πρόθεμα και το αντίστοιχο μέγιστο συμμετρικό υποδέντρο, ο κόμβος u εγκαθιδρύει μία σύνδεση με κάποιον κόμβο που να ανήκει σε αυτό. Δεδομένου ότι ένα υποδέντρο μπορεί να εμπεριέχει πολλά φύλλα κι άρα αντιστοιχούν πολλοί ομότιμοι κόμβοι σε αυτό, ο κόμβος u χρειάζεται να γνωρίζει μόνο έναν από αυτούς. Για παράδειγμα, στο σχήμα 3.2(α΄) δείχνουμε για κάθε συμμετρικό υποδέντρο του κόμβου u τον κόμβο με τον οποίον συνδέεται. Παρατηρούμε βεβαίως ότι κάθε κόμβος έχει μόνο μερική γνώση σχετικά με τη δομή του k-d δέντρου. Έτσι στην Εικόνα 3.2(β΄) απεικονίζεται αυτή η τοπική γνώση για τον κόμβο u ο οποίος διαθέτει πληροφορία μόνο για τις διασπάσεις (#1, #2 και #4) που βρίσκονται κατά μήκος της διαδρομής του προς τη ρίζα του δέντρου. Τα σκιασμένα παραλληλόγραμμα αντιστοιχούν στα υποδέντρα με την ίδια απόχρωση στο Σχήμα 3.2(α΄). Οπότε ο κόμβος u γνωρίζει ακριβώς έναν από τους κόμβους που εμπίπτουν σε κάθε ένα παραλληλόγραμμο που σχηματίζεται. Παρατηρούμε ωστόσο ότι αυτά τα ορθογώνια καλύπτουν το σύνολο του χώρου I. Αυτό είναι αναγκαίο προκειμένου να διασφαλιστεί ότι ο κόμβος u είναι σε θέση να εντοπίσει οποιονδήποτε άλλον κόμβο με τον τρόπο που περιγράφεται στην Ενότητα 4.1, και να επεξεργαστεί σύνθετα ερωτήματα όπως περιγράφουμε στα επόμενα κεφάλαια.

Ο Πίνακας $u.link$ αποτελεί τον πίνακα δρομολόγησης του κόμβου u. Η εγγραφή $u.link[i]$ περιέχει τη διεύθυνση ενός από τους κόμβους που βρίσκεται στο μέγιστο συμμετρικό υποδέντρο που προέρχεται από το μήκους i πρόθεμα του αναγνωριστικού $u.id$. Άρα ο κόμβος u θα εγκαθιδρύσει συνδέσεις με τρεις ακριβώς ομότιμους κόμβους κι οπότε $u.link = \{z, w, y\}$. Ο Πίνακας 3.1 απεικονίζει τον πίνακα $link$ για κάθε κόμβο . Ο συμβολισμός $u(000)$ υποδεικνύει ότι ο κόμβος u αντιστοιχεί στο φύλλο του k-d δέντρου με αναγνωριστικό 000. Επιπλέον, ο

συμβολισμός 10:*v* (100) υποδεικνύει ότι ο κόμβος *v* με το αναγνωριστικό 100 βρίσκεται στο υποδέντρο που αντιστοιχεί στο συμμετρικό πρόθεμα 10. Η πρώτη σειρά του Πίνακα 3.1 δείχνει ότι ο κόμβος *u* διαθέτει τρεις συνδέσμους για τους κόμβους *z*, *w* και *y*. Αυτοί προκύπτουν από τα μέγιστα συμμετρικά υποδέντρα με ρίζες στους κόμβους του k-d δέντρου με αναγνωριστικά 1, 01 και 001, αντίστοιχα.

Peer	*link* **entries**		
u(000)	1: *z*(101)	01: *w*(01)	001: *y*(001)
y(001)	1: *z*(101)	01: *w*(01)	000: *u*(000)
w(01)	1: *v*(100)	00: *u*(000)	
v(100)	0: *w*(01)	11: *x*(11)	101: *z*(101)
z(101)	0: *y*(001)	11: *x*(11)	100: *v*(100)
x(11)	0: *u*(000)	10: *v*(100)	

Πίνακας 3.1: Αναπαράσταση πινάκων δρομολόγησης.

3.3 Εισαγωγή Νέων Κόμβων

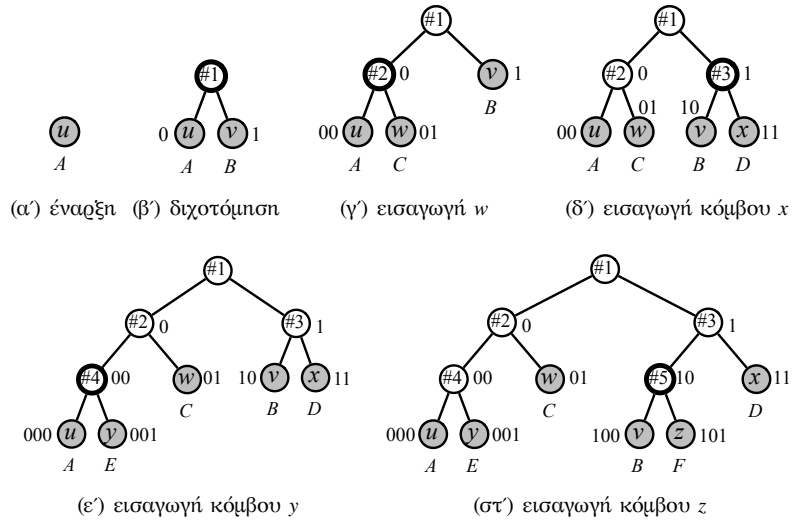
Όταν ένα νέος κόμβος επιθυμεί να συμμετάσχει στο δίκτυο του MIDAS επιλέγει αρχικά ένα τυχαίο σημείο $\vec{p}$ στο συνολικό χώρο κλειδιών *I* κι εντοπίζει τον κόμβο *u* που υπάρχει ήδη στο δίκτυο κι είναι υπεύθυνος για το συγκεκριμένο κλειδί (αναλυτική συζήτηση της μεθόδου στην Ενότητα 4.1). Τώρα υπάρχουν δύο πιθανές περιπτώσεις ανάλογα με την κατάσταση του κόμβου *u*.

Σύμφωνα με την πρώτη περίπτωση, εάν στον host που είναι υπεύθυνος για τον κόμβο *u* δεν φιλοξενούνται άλλοι κόμβοι, τότε το φύλλο του k-d δέντρου με αναγνωριστικό *u.id* χωρίζεται και δημιουργούνται δύο νέα φύλλα καθώς επεκτείνεται το δέντρο κατά βάθος. Η διάσταση *sdim* στην οποία λαμβάνει χώρα το κόψιμο του χώρου μπορεί να είναι αυτή κατά οποία η κατανομή των δεδομένων στον κόμβο *u* παρουσιάζει το μεγαλύτερο εύρος ή κάθε φορά να επιλέγεται σε σειρά διαφορετική διάσταση ή απλά τυχαία. Επιπλέον, η τιμή *split* στην οποία χωρίζεται ο χώρος του κόμβου *u* στη διάσταση *sdim* επιλέγεται με τέτοιον τρόπο ώστε να είναι αμφότεροι κόμβοι υπεύθυνοι για ίσο αριθμό κλειδιών. Το μεγαλύτερο μισό του κόμβου *u* ανατίθεται στον καινούριο κόμβο *w* που αντιστοιχεί στο δεξί νέο φύλλο, ενώ ο κόμβος *u* είναι υπεύθυνος πλεόν για το χαμηλότερο μισό που αντιστοιχεί στο αριστερό νέο φύλλο.

Προκειμένου να διασφαλιστεί η σωστή λειτουργία, ακολουθείται η εξής διαδιασία: (1) Ο κόμβος *u* στέλνει στον *w* τις πλειάδες που εμπίπτουν στη δικαιοδοσία του. (2) Ο κόμβος *u*: (i) προσθέτει το δυφίο 0 στο αναγνωριστικό του *u.id*, (ii) επεκτείνει τον πίνακα δρομολόγησής *u.link* κατά μία εγγραφή προσθέτοντας τον κόμβο *w* στο τέλος, (iii) προσθέτει τη διάσταση κατά την οποία διαχωρίστηκε ο χώρος στον πίνακα *u.sdim* και την αντίστοιχη τιμή στον πίνακα *u.split*. (3) Ταυτόχρονα, ο κόμβος *w*: (i) αντιγράφει την κατάσταση του αρχικού κόμβου *u*, (ii) επεκτείνει το αναγνωριστικό του πατρικού κόμβου προσθέτοντας το ψηφίο 1 διαμορφώνοντας έτσι κατάλληλα το αναγνωριστικό του *w.id*, (iii) τοποθετεί έναν σύνδεσμο προς τον κόμβο *u* στον πίνακα δρομολόγησής του *w.link* (iv) εξυπηρετεί τους κόμβους στον πίνακα *w.backlink* που αντιστοιχούν στους μισούς κόμβους από αυτούς που εξυπηρετούσε αρχικά ο κόμβος *u*, (v) ενημερώνει τους συνδέσμους του πίνακα *w.backlink* ανάλογα ώστε να ενημερώσουν τους δικούς τους πίνακες δρομολόγησής ανάλογα ώστε να συμπεριλαμβάνουν τον καινούριο κόμβο στην θέση που κατείχε προηγουμένως ο κόμβος *u*.

Η δεύτερη περίπτωση εφαρμόζεται όταν ο υπολογιστής ο οποίος φιλοξενεί τον κόμβο *u* είναι υπεύθυνος για περισσότερους κόμβος. Στην περίπτωση αυτή ανατίθεται ο καινούριος κόμβος αναλαμβάνει πλήρως τις λειτουργίες του κόμβου *u*, ενημερώνοντας έπειτα τους συνδέσμους του κόμβου *u* για την αλλαγή αυτή.

Στη συνέχεια παρουσιάζουμε ένα παράδειγμα του πως σχηματίζεται το δίκτυο του Σχήματος 3.3. Θεωρούμε αρχικά ότι υπάρχει ένας μόνο κόμβος *u*, του οποίου η ζώνη αντιστοιχεί σε ολόκληρο τον χώρο κλειδιών, όπως δείχνουμε στο Σχήμα 3.3(α΄). Η εισαγωγή ενός νέου



Σχήμα 3.3: Σχηματισμός δικτύου με τη σταδιακή εισαγωγή νέων κόμβων.

κόμβου στο δίκτυο θα προκαλέσει την επέκταση του δέντρου έστω κατά μήκος της πρώτης διάστασης (κόψιμο #1 στο Σχήμα 3.3). Στο Σχήμα 3.3(β′) δείχνουμε το αποτέλεσμα μίας τέτοιας εισαγωγής όπου ο κόμβος *u* γίνεται πλέον υπεύθυνος για το φύλλο με αναγνωριστικό 0 ενώ ο νέος κόμβος *v* αποκτά το αναγνωριστικό 1. Ακολουθώντας την ίδια ακριβώς διαδικασία εξαπλώνεται το δίκτυο με την εισαγωγή των νέων κόμβων *w*,*x*,*y* και *z*. Πιο αναλυτικά ας υποθέσουμε ότι ένας επιπλέον νέος κόμβος θέλει να εισαχθεί στο δίκτυο. Ο κόμβος αυτός επιλέγει ένα τυχαίο σημείο το οποίο έστω ότι εμπίπτει στη ζώνη ευθύνης του κόμβου *u*. Οπότε, ο κόμβος αυτός που έχει αναγνωριστικό 0 χωρίζεται έστω κατά μήκος της δεύτερης διάστασης (κόψιμο #2). Έτσι ο κόμβος *u* θα είναι πλέον υπεύθυνος για τον νέο κόμβο που αντιστοιχεί στο αριστερό φύλλο που προκύπτει από την επέκταση του δέντρου στο Σχήμα 3.3(γ′) και θα έχει αναγνωριστικό 00 ενώ το δεξί φύλλο με αναγνωριστικό 01 ανατίθεται στον νεο-εισερχόμενο κόμβο *w*. Η μετέπειτα εισαγωγή ενός ακόμα κόμβου που εμπίπτει στη ζώνη ευθύνης του κόμβου *v* προκαλεί τον χωρισμό του κόμβου που έχει το αναγνωριστικό 1 κατά μήκος της διάστασης *v.sdim*[2] με το κόψιμο #3 όπως δείχνουμε στο Σχήμα 3.3(δ′). Έτσι ο κόμβος *v* έχει πλέον το αναγνωριστικό 10 κι ο νέος κόμβος *x* θα έχει το αναγνωριστικό 11. Τώρα η εισαγωγή ενός ακόμα νέου κόμβου *y* θα προκαλέσει το διαμελισμό του κόμβου *u* κατά τη διάσταση *u.sdim*[3] και θα είναι έπειτα υπεύθυνος για το αριστερό φύλλο που προέρχεται από την επέκταση του δέντρου. Συνεπώς, ο κόμβος *u* θα ταυτοποιείται πλέον από το αναγνωριστικό 000 ενώ ο νέος κόμβος *y* από το αναγνωριστικό 001 κι είναι υπεύθυνος για τον κόμβο που αντιστοιχεί στο δεξί φύλλο όπως δείχνουμε στο Σχήμα 3.3(ε′) με το κόψιμο #4 του χώρου. Τέλος, η εισαγωγή του κόμβου *z* προκαλεί το διαμελισμό του χώρου του φύλλου με αναγνωριστικό 10 κατά μήκος της διάστασης *v.sdim*[3] με το κόψιμο #5. Ο νέος κόμβος *z* θα έχει αναγνωριστικό 101 ενώ το νέο αναγνωστικό του κόμβου *v* θα είναι το 100. Στο Σχήμα 3.3(στ′) δείχνουμε τη μορφή του k-d δέντρου μετά και την τελευταία αυτή εισαγωγή.

Το ακόλουθο λήμμα δείχνει ότι οι εισαγωγές κόμβων σύμφωνα με το πρωτόκολλο του MIDAS είναι ασφαλείς και δεν ανατρέπουν την ισχύ του Λήμματος 1 σε οποιαδήποτε μετέπειτα φάση του δικτύου.

Λήμμα 2 *Μετά από κάθε εισαγωγή κόμβου, η σταθερά (invariant) του MIDAS παραμένει.*

Απόδειξη 2 *Σύμφωνα με την πρώτη περίπτωση, για την εισαγωγή ενός νέου κόμβου, έστω w, χρειάζεται να χωριστεί στο MIDAS ένας άλλος κόμβος, έστω u, και να αναθέσει τμήμα των λειτουργιών του που έχει αναλάβει στον νέο κόμβο στα πλαίσια της συμμετοχής τους στο δίκτυο. Έστω ότι ο κόμβος u′ ο κόμβος που προέρχεται από τον αρχικό u. Τότε ισχύει ότι ο κόμβος του k-d δέντρου u είναι γονέας αμφότερων u′ και w ενώ το αναγνωριστικό του u.id αποτελεί πρόθεμα των αναγνωριστικών των δύο θυγατρικών κόμβων u′.id και w.id. Επίσης*

σημειώνουμε ότι εκ κατασκευής ισχύει ότι $I_u = I_{u'} \uplus^{d_u} I_w$. *Οπότε, κάθε σημείο που ενέπιπτε στον αρχικό χώρο κλειδιών του* u *θα ανήκει τώρα είτε στον κόμβο* u' *είτε στον* w *αλλά όχι και στους δύο. Όλα τα υπόλοιπα σημεία παραμένουν ως έχουν αφού δεν τους αφορά η συγκεκριμένη αλλαγή στην τοπολογία του δικτύου.*

Για τη δεύτερη περίπτωση παρατηρούμε ότι η εισαγωγή ενός κόμβου δεν προκαλεί καμία παντελώς αλλαγή στο κατανεμημένο k-d δέντρο. Συνεπώς, οι σταθερές (invariant) του MIDAS διατηρούνται μετά από κάθε εισαγωγή κόμβου στο δίκτυο. ■

Η πιθανοτική φύση του μηχανισμού εισαγωγής κόμβων του MIDAS επιτυγχάνει έναν σημαντικό στόχο. Διασφαλίζει ότι το αναμενόμενο βάθος του k-d δέντρου, ήτοι το μήκος της μέγιστης διαδρομής από τη ρίζα προς τα φύλλα, είναι λογαριθμικό ως προς τον συνολικό αριθμό κόμβων στο δίκτυο. Το ακόλουθο λήμμα υποστηρίζει τον ισχυρισμό αυτό.

Λήμμα 3 *Το αναμενόμενο βάθος του κατανεμημένου k-d δέντρου που αντικατοπτρίζει το MIDAS μετά την εξαρχής κατασκευής με την εισαγωγή* n *κόμβων ισούται με* $O(\log n)$ *κι έχει σταθερή διακύμανση.*

Απόδειξη 3 *Ας θεωρήσουμε ένα δίκτυο από* n *κόμβους στο MIDAS. Αφού κάθε εσωτερικός κόμβος του k-d δέντρου έχει ακριβώς δύο θυγατρικούς κόμβους, υποδηλώνοντας έτσι κάποια διαμέριση του αρχικού χώρου στους κόμβους, συνεπάγεται ότι συνολικά υπάρχουν* $n-1$ *εσωτερικοί κόμβοι. Το k-d δέντρο που προέρχεται από την αφαίρεση των φύλλων είναι ένα παράδειγμα ενός* τυχαίου relaxed k-d tree *(random relaxed k-d tree), όπως αυτό ορίζεται στο [54], το οποίο αποτελεί προέκταση του* τυχαίου k-d δέντρου *(random k-d tree) ως αυτό ορίζεται στο [26]. Αυτό συμβαίνει διότι οι τιμές κι οι διαστάσεις επιλέγονται τυχαία κι ανεξάρτητα μεταξύ τους για τον εκάστοτε διαμερισμό.*

Έχει δειχθεί στα [54] και [26] ότι η κατασκευή ενός k-d δέντρου με n *τυχαίες εισαγωγές είναι ισοδύναμη με τη δημιουργία της ίδιας δεντρικής δομής από* n *τυχαίες εισαγωγές σε ένα δυαδικό δέντρο αναζήτησης (binary search tree). Γνωρίζουμε ήδη για τα* τυχαία δυαδικά δέντρα αναζήτησης *(*random binary search trees*), ότι το αναμενόμενο μήκος μονοπατιού από τη ρίζα προς τα φύλλα είναι λογαριθμικό μέγεθος ως προς το συνολικό αριθμό κόμβων στο δίκτυο. Η ανάλυση στο [100] δείχνει ότι το* μέγιστο μήκος μονοπατιού *ή αλλιώς το βάθος του δέντρου, έχει αναμενόμενη τιμή* $O(\log n)$ *και διακύμανση* $O(1)$. *Η ανάλυση αυτή ισχύει αυτούσια για τη δομή του MIDAS που αντικατοπτρίζει ένα κατανεμημένο k-d δέντρο που απαρτίζεται από* n *κόμβους που εισάγονται σε τυχαία σημεία στο δίκτυο.*■

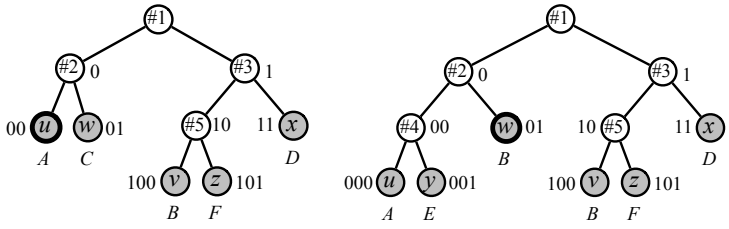
Το προηγούμενο λήμμα είναι απαραίτητο για τον ορισμό ασυμπτωτικών φραγμάτων ως προς την απόδοση της επεξεργασίας ερωτημάτων στο MIDAS. Πρώτον, υπονοεί ότι η πληροφορία που διατηρείται σε κάθε κόμβο έχει λογαριθμικό μέγεθος ως προς το μέγεθος του δικτύου. Επιπλέον, στο Κεφάλαιο 4 υπολογίζουμε τα άνω φράγματα του κόστους επεξεργασίας διαφόρων τελεστών κι ερωτημάτων.

3.4 Αποχώρηση Κόμβων

Όταν ένας κόμβος επιθυμεί να αποχωρήσει από το δίκτυο υπόκειται σε δύο περιπτώσεις ανάλογα με τη θέση του στο κατανεμημένο k-d δέντρο.

Σύμφωνα με την πρώτη περίπτωση, έχουμε το παράδειγμα του Σχήματος 3.3 με τον κόμβο y ο οποίος πρόκειται να αποχωρήσει και του οποίο ο κόμβος με τον οποίον έχουν κοινό πατέρα στο κατανεμημένο k-d δέντρο είναι εξίσου φύλλο κι άρα αντιστοιχεί σε έναν ομότιμο κόμβο του δικτύου ονόματι u. Υπενθυμίζουμε ότι ο κόμβος y συνδέεται με τον u καθώς διατηρεί για αυτόν την τελευταία θέση στον πίνακα δρομολόγησης $y.link$. Οπότε όταν ο κόμβος y αποχωρεί η δομή του δικτύου θα πρέπει να προσαρμοστεί ανάλογα αφαιρώντας τα φύλλα του δέντρου y και u ούτως ώστε ο πατέρας τους να γίνει φύλλο του k-d δέντρου. Έτσι, ο κόμβος u θα αναλάβει επιπλέον τις λειτουργίες του "αδερφού" κόμβου y που αποχωρεί κι η τοπολογία του δικτύου θα τροποποιηθεί με τέτοιο τρόπο ώστε να συσχετιστεί με τον πατρικό κόμβο αμφότερων φύλλων. Στο Σχήμα 3.4(α′) δείχνουμε τη δομή του δέντρου μετέπειτα της διαγραφής του κόμβου. Σημειωτέον ότι ο κόμβος u θα είναι πλέον υπεύθυνος για τα κλειδιά

τα οποία αντιστοιχούν στην περιοχή που αποτελεί την ένωση της ζώνης του κόμβου που αποχώρησε με τη δική του ενώ το αναγνωριστικό του γενικεύεται/μειώνεται κατά ένα δυφίο.



(α′) αποχώρηση κόμβου y (β′) αποχώρηση κόμβου u

Σχήμα 3.4: Δύο παραδείγματα αποχώρησης κόμβων.

Ακολουθείται η εξής διαδικασία στο MIDAS προκειμένου να διασφαλίσουμε ότι όλες οι απαραίτητες αλλαγές στη περίπτωση αυτή εφαρμόζονται κατάλληλα στο δίκτυο: (1) Ο κόμβος y αποστέλει στον "αδερφό" του κόμβο u την περιοχή για την οποία ήταν υπεύθυνος μαζί με όλα τα ζεύγη κλειδιών-τιμών που εμπίπτουν στην συγκεκριμένη ζώνη. (2) Ο κόμβος u: (i) γενικεύει/μειώνει το αναγνωριστικό του κατά ένα δυφίο, (ii) μειώνει το βάθος του συγκεκριμένου υποδέντρου κατά ενα, και (iii) αφαιρεί την τελευταία εγγραφή από τους πίνακες $u.sdim$, $u.split$, $u.link$. (3) Ο κόμβος y ενημερώνει τους κόμβους που εξυπηρετεί για το γεγονός ώστε να αντικαταστήσουν τη πληροφορία που διατηρούσαν για τον y στους πίνακες δρομολόγησής τους με τα στοιχεία του u. (4) Ο κόμβος u εξυπηρετεί μαζί με τους δικούς του και τους κόμβους που εξυπηρετούσε ο κόμβος y πριν την αποχώρησή του.

Για τη δεύτερη περίπτωση, ας εξετάσουμε στο Σχήμα 3.4(β′) την αποχώρηση του κόμβου w του οποίου ο "αδερφός" κόμβος δεν αποτελεί φύλλο του κατανεμημένου k-d δέντρου αλλά εσωτερικό κόμβο κι άρα δεν μπορεί να ενσωματωθεί απλά σε κάποιον άλλο κόμβο όπως έγινε στην προηγούμενη περίπτωση. Κατά συνέπεια ο κόμβος w θα πρέπει να φιλοξενηθεί μαζί με κάποιον άλλον κόμβο ως έχει. Έτσι ο κόμβος w ζητάει από τον κόμβο που γνωρίζει με το ελάχιστο φορτίο να αναλάβει πλήρως τις λειτουργίες του και να ενημερώσει κατάλληλα τους κόμβους που εξυπηρετούνται από τον w ώστε να επικαιροποιήσουν τους πίνακες δρομολόγησής τους ανάλογα. Στο δικό μας παράδειγμα στο Σχήμα 3.4(β′) ο κόμβος w καταλήγει να φιλοξενείται στο ίδιο host μαζί με τον κόμβο v.

Το ακόλουθο λήμμα δείχνει ότι οι αποχωρήσεις κόμβων στο MIDAS είναι ασφαλής.

Λήμμα 4 *Μετά από την αποχώρηση ενός κόμβου, η σταθερά (invariant) του MIDAS εξακολουθεί να ισχύει.*

Απόδειξη 4 *Σύμφωνα με την πρώτη περίπτωση, η αποχώρηση θα προκαλέσει την αφαίρεση δύο φύλλων του κατανεμημένου k-d δέντρου. Ας θεωρήσουμε w τον κόμβο που αποχωρεί και u τον "αδερφό" του κόμβο στο δέντρο. Ακόμα, ας θεωρήσουμε τον κόμβο u' που αντιστοιχεί στον u όπως αυτός προκύπτει μετά τις αλλαγές που πυροδοτεί η αποχώρηση του w. Ειδικότερα ο u' αντιστοιχεί στον εσωτερικό κόμβο που αποτελεί τον πατρικό κόμβο για τους u και w κι άρα οι χώροι τους αποτελούν διαμέριση του χώρου του u' ο οποίος διασπάστηκε στη διάσταση $d_{u'}$ όταν αυτοί δημιουργήθηκαν με τον τρόπο που περιγράφθηκε στην προηγούμενη ενότητα, κι άρα $I_{u'} = I_u \uplus^{d_{u'}} I_w$. Οπότε τα σημεία που είχαν ανατεθεί στους κόμβους u και w τώρα βρίσκονται στη δικαιοδοσία του κόμβου u'. Πρόκειται για τον κόμβο u επεκταμένο κατάλληλα ώστε να συμπεριλάβει τον w. Καμία αλλαγή δεν προβλέπεται για τους υπόλοιπους κόμβους του δικτύου.*

Για τη δεύτερη περίπτωση καμία αλλαγή δε λαμβάνει χώρα όσον αφορά τη δομή του k-d δέντρου.Οπότε για αμφότερα σενάρια η σταθερά του MIDAS διατηρείται μετά την αποχώρηση ενός κόμβου. ■

3.4.1 Απρόσμενες Απώλειες Κόμβων

Σε ένα δυναμικό περιβάλλον είναι αναμενόμενο από ένα ποσοστό κόμβων να αποχωρούν από το δίκτυο απρόσμενα λόγω τεχνικών προβλημάτων ή άλλων λόγων χωρίς να έχουν προηγουμένως προλάβει να εκτελέσουν το πρωτόκολλο αποχώρησης που περιγράψαμε (churn failures). Προκειμένου να διασφαλιστεί η απρόσκοπτη λειτουργία του δικτύου έχει σχεδιαστεί ένας μηχανισμός για την αντιμετώπιση αυτών των περιπτώσεων. Ας θεωρήσουμε ότι ένας κόμβος αποχωρεί απρόσμενα από το δίκτυο, τότε η διαδικασία που ακολουθείται στο MIDAS έχει ως άξονες: (i) την ανάληψη από κάποιον άλλον κόμβο της περιοχής για την οποία ήταν υπεύθυνος ο κόμβος που αποχώρησε απρόσμενα, (ii) την ανάκτηση των δεδομένων που χάθηκαν με την απρόσμενη αποχώρηση.

Όσον αφορά το πρώτο σκέλος, όλοι οι κόμβοι οι οποίοι ήταν συνδεδεμένοι με τον κόμβο που αποχώρησε απρόσμενα γίνονται γνώστες της συγκεκριμένης απώλειας με διάφορους τρόπους, π.χ. ελέγχοντας περιοδικά αν οι κόμβοι των οποίων τα στοιχεία διατηρούν στον πίνακα δρομολόγησης εξακολουθούν να τους εξυπηρετούν. Κάθε ένας κόμβος που εξυπηρετούνταν από έναν κόμβο του οποίου η λειτουργία σταματάει γνωρίζει την περιοχή για την οποία ήταν υπεύθυνος αλλά μόνο ένας από αυτούς θα αναλάβει την περιοχή αυτή. Για άλλα κατανεμημένα ευρετήρια που έχουν προταθεί κατά καιρούς αυτό εγείρει θέματα συμφωνίας μεταξύ των διαφόρων μερών. Για παράδειγμα στο CAN ο γείτονας του κόμβου που αποχώρησε υπεύθυνος για τη μικρότερη περιοχή θα αναλάβει την περιοχή που απωλέσθει. Επίσης έχουν προταθεί κι άλλες μέθοδοι που κατ᾽ ουσίαν απαιτούν την εκτέλεση κάποιου κατανεμημένου αλγόριθμου με σκοπό την επιλογή του πιο κατάλληλου κόμβου για την ανάθεση της αντίστοιχης περιοχής, π.χ. ο κόμβος με το μικρότερο φορτίο. Όμως κάτι τέτοιο δεν είναι απαραίτητο στη συγκεκριμένη περίπτωση για το MIDAS. Συγκεκριμένα αν υπάρχει κόμβος με κοινό πατέρα με τον κόμβο που χάθηκε που να αντιστοιχεί σε φύλλο του κατανεμημένου k-d δέντρου, τότε ο κόμβος αυτός αυτόματα αναλαμβάνει την περιοχή του "αδερφού" κόμβου του και το βάθος στο συγκεκριμένο υποδέντρο μειώνεται κατά ένα. Διαφορετικά, ο κόμβος με το μικρότερο αναγνωριστικό ή ο κόμβος με το μικρότερο φορτίο εκ των διαθέσιμων συνδέσμων του απωλεσθέντα κόμβου αναλαμβάνει τις λειτουργίες του στον κόμβο που αντικατέστησε τον απωλεσθέντα κόμβο.

Σχετικά με το δεύτερο σκέλος, θεωρούμε ότι οι κόμβοι οι οποίοι εισάγουν τα δεδομένα, επαναλαμβάνουν περιοδικά τη λειτουργία αυτή. Έτσι για κάθε πλειάδα υπάρχει μία χρονική παράμετρος time-to-live (TTL) σύμφωνα με την οποία επανα-εισάγονται τα δεδομένα. Οπότε οι χαμένες πλειάδες οι οποίες βρίσκονταν στον κόμβο που χάθηκε εν τέλει θα επανατοποθετηθούν.

3.5 Συζήτηση

Στο κεφάλαιο αυτό είδαμε την αρχιτεκτονική ενός νέου δομημένου δικτύου ομοτίμων (structured peer-to-peer network) το οποίο αντικατοπτρίζει ένα κατανεμημένο k-d δέντρο (virtual distributed k-d tree) του οποίου το αναμενόμενο βάθος φράσσεται στο $O(\log n)$. Επακολούθως, οι πίνακες δρομολόγησης (routing tables) τους οποίους διατηρεί ο κάθε ένας κόμβος ενήμερους με τις αλλαγές που λαμβάνουν χώρα στο δίκτυο έχουν μέγεθος το οποίο δεν υπερβαίνει το βάθος του δέντρου στο οποίο αντιστοιχεί το overlay. Αντίστοιχο αντίκτυπο έχει η δομή του MIDAS στην απόκριση των αλγορίθμων δρομολόγησης.

Παρόμοια δενδροειδή μορφή σχηματίζουν συστήματα όπως το P-Grid τα οποία στοχεύουν στην αποθήκευση ζευγών κλειδιών-τιμών βάσει μονοδιάστασταστατων αναγνωριστικών τα οποία αποτελούνται από 160 ψηφία. Πάνω στη δομή αυτή τα πρωτόκολλα εισαγωγής και αποχώρησης κόμβων καθορίζουν το εάν ένας εσωτερικός κόμβος θα αντιγράφεται εξολοκλήρου στα επόμενα επίπεδα του δέντρου ή θα μοιράζεται το φορτίο του στους δύο θυγατρικούς κόμβους. Αντίστοιχη διεπαφή και δομή έχει και το Chord του οποίο όμως σε αφαιρετικό επίπεδο η δομή του έχει σχεδιαστεί με τέτοιο τρόπο ώστε να σχηματίζει έναν δακτύλιο.

Ανάλογου με το MIDAS σκοπού συστήματα που στοχεύουν στην κατασκευή ευρετηρίων επί τη βάσει πολυδιάστατων κλειδιών όπως το CAN, σχηματιζουν μία δομή που παραπέμπει σε κάναβο (mesh). Πιο συγκεκριμένα, έκαστος κόμβος είναι υποχρεωμένος να γνωρίζει όλους

τους κόμβους οι οποίοι είναι υπεύθυνοι για τις συνεχής (contiguous) ζώνες που έπονται ή προηγούνται στον πολυδιάστατο χώρο κλειδιών του υπό εξέταση κόμβου. Αυτό συνεπάγεται ότι οι πίνακες δρομολόγησης αυξάνουν το μέγεθός τους γραμμικά ώς προς το μέγεθος του δικτύου καθώς και τη διαστασημότητα του χώρου κλειδιών όταν για το MIDAS το μέγεθος των finger tables παρουσιάζει λογαριθμική συμπεριφορά ως προς το μέγεθος του δικτύου, γεγονός που το αναλεικνύει ως μία πιο ελκυστική λύση για πολλών ειδών προβλήματα.

Κεφάλαιο 4

Επεξεργασία Ερωτημάτων

I have an answer.
Do you know the question?

Στο κεφάλαιο αυτό προτείνουμε μεθόδους επεξεργασίας σύνθετων ερωτημάτων για πολυδιάστατα δεδομένα στο MIDAS. Στην Ενότητα 4.1 περιγράφουμε μεθόδους για ερωτήματα σημείων, στην Ενότητα 4.2 για ερωτήματα εύρους και στην Ενότητα 4.3 για ερωτήματα κοντινότερων γειτόνων.

4.1 Ερωτήματα Σημείων

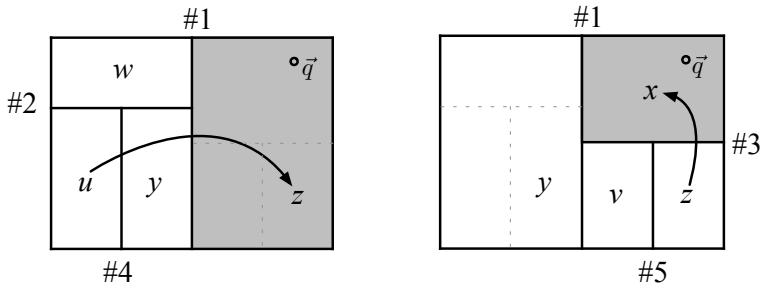
Η δομή του MIDAS που αντικατοπτρίζει ένα κατανεμημένο k-d δέντρο επιτρέπει την ανάπτυξη ιεραρχικών μεθόδων για αποτελεσματική δρομολόγηση. Δείχνουμε πως ένας κόμβος μπορεί να επεξεργαστεί ερωτήματα σημείων, δηλαδή τον εντοπισμό του κόμβου υπεύθυνου για ένα συγκεκριμένο σημείο του χώρου I, σε λογαριθμικό μέγιστο αριθμό βημάτων σε σχέση με το συνολικό αριθμό κόμβων στο δίκτυο.

Ο Αλγόριθμος 3 δίνει τις λεπτομέρειες για το πως ακριβώς γίνεται η επεξεργασία τέτοιων ερωτημάτων στο MIDAS. Ας υποθέσουμε ότι ο κόμβος u λαμβάνει ένα τέτοιο ερώτημα για το σημείο $\vec{q}$. Εάν η ζώνη του κόμβου αυτού εμπεριέχει το σημείο $\vec{q}$ (εφόσον αυτό υπάρχει), τότε ο υπεύθυνος κόμβος για το σημείο επιστρέφει στον εκδότη w του ερωτήματος το αντίστοιχο ζεύγος `(κλειδιού-`$\vec{q}$`, τιμής)` (γραμμές 1–3). Διαφορετικά, ο κόμβος u θα πρέπει να βρει τον πιο σχετικό κόμβο για να προωθήσει το ερώτημα. Ο πιο σχετικός κόμβος είναι αυτός ο οποίος έγκειται στο συμμετρικό υποδέντρο του κόμβου λήπτη του ερωτήματος το οποίο συμπεριλαμβάνει το σημείο $\vec{q}$ του ερωτήματος στην περιοχή που αντιπροσωπεύει. Οπότε ο κόμβος u εξετάζει την τοπική/μερική εικόνα που διαθέτει για τη δομή του δικτύου και το k-d δέντρο που αυτή αντικατοπτρίζει (όπως σχηματίζεται από τους πίνακες *sdim* και *split*) κι έτσι επιλέγεται το συμμετρικό υποδέντρο στο οποίο ανήκει το πολυδιάστατο κλειδί $\vec{q}$ (γραμμές 5–11). Έπειτα, το ερωτημα προωθείται στον σύνδεσμο που αντιστοιχεί στο συγκεκριμένο υποδέντρο (γραμμή 7) προκειμένου να συνεχιστεί η διαδικασία αυτή αναδρομικά έως ότου βρεθεί ο κάτοχος του συγκεκριμένου κλειδιού.

Στο Σχήμα 4.1(α΄) δείχνουμε το σημείο $\vec{q}$ του ερωτήματος το οποίο εκδίδεται από τον κόμβο u. Επίσης εικονίζεται η μερική εικόνα του δικτύου που έχει ο κόμβος u. Εφόσον όμως η απάντηση του ερωτήματος δεν εμπίπτει στην αρμοδιότητα του κόμβου u, θα πρέπει να το προωθήσει κατάλληλα στον κόμβο z που είναι πιο σχετικός για να επεξεργάζεται τα ερωτήματα τα οποία αφορούν τη σκιαγραφημένη περιοχή (1ο βήμα). Στη συνέχεια ο κόμβος z χρησιμοποιώντας τη δική του μερική γνώση που έχει για τη δομή του δικτύου, θα προωθήσει το ερώτημα στον πιο σχετικό κόμβο που γνωρίζει (2ο βήμα) που για την περίπτωσή μας είναι ο κόμβος x. Τέλος, ο κόμβος x είναι υπεύθυνος για το κλειδί του ερωτήματος θα επιστρέψει τη σχετική πλειάδα στον κόμβο u που εξέδωσε το ερώτημα αρχικά.

Algorithm 3: u.Point $(\vec{q}, w)$: Ο κόμβος u επεξεργάζεται το ερώτημα $\vec{q}$ που εξέδωσε ο w.

```
1 if u.IsLocal(q⃗) then
2 |   u.Send_to (w, u.Get_val (q⃗)) ;
3 |   return;
4 for j ← 0 to u.depth do
5 |   d ← u.sdim[j] ;
6 |   if (u.id[j] = 0 and q⃗[d] ≥ u.split[j]) or (u.id[j] = 1 and q⃗[d] < u.split[j]) then
7 |   |   u.link[j].Point (q⃗, w) ;
8 |   |   return;
```



(α′) επεξεργασία και προώθηση από τον κόμβο u

(β′) επεξεργασία και προώθηση από τον κόμβο z

Σχήμα 4.1: Παράδειγμα επεξεργασίας ερωτήματος σημείου $\vec{q}$ για δύο διαστάσεις.

Λήμμα 5 *Ο αναμενόμενος αριθμός βημάτων για ερωτήματα σημείου φράσσεται από το* $O(\log n)$ *βήματα, όπου* n *το μέγεθος του δικτύου.*

Απόδειξη 5 *Αρχικά δείχνουμε ότι ο απαιτούμενος αριθμός βημάτων ισούται με το βάθος του k-d δέντρου στη χειρότερη περίπτωση.*

Ας θεωρήσουμε τον κόμβο v *που λαμβάνει ένα ερώτημα για το σημείο* $\vec{q}$ *κι ας υποθέσουμε ότι το βάθος του ισούται με* k *στον πίνακα δρομολόγησης link του κόμβου* u*. Εξαιτίας του αλγόριθμου δρομολόγησης (γραμμή 7), το σημείο* $\vec{q}$ *ανήκει στο υποδέντρο βάθους* k *που εμπεριέχει τον κόμβο* v*. Οπότε το σημείο* $\vec{q}$ *του ερωτήματος δεν θα μπορούσε να απαντηθεί στα συμμετρικά υποδέντρα του παραλήπτη κόμβου* v *με βάθος μικρότερο του* k *αλλά μόνο σε αυτά μεγαλύτερου βάθους.*

Άρα ο απαιτούμενος αριθμός επαναπροωθήσεων είναι στη χειρότερη περίπτωση ίσος με το βάθος του δέντρου. Κι από το Λήμμα 3 ότι το αναμενόμενο βάθος είναι $O(\log n)$*.* ■

4.2 Ερωτήματα Εύρους

Ένα ερώτημα εύρους ορίζεται από ένα πολυδιάστατο διάστημα $[\vec{\ell}, \vec{h}]$, όπου $\vec{\ell}$ το κάτω φράγμα και $\vec{h}$ το άνω φράγμα κι αιτείται όλων των ζευγών `(κλειδιού-`$\vec{q}$`, τιμής)` που ανήκουν στο συγκεκριμένο διάστημα. Η συμβατική προσέγγιση επί του θέματος απαιτεί τον εντοπισμό του κόμβου υπεύθυνου για ένα από τα δύο φράγματα και στη συνέχεια τη σειριακή προσπέλαση όλων των κόμβων που είναι υπεύθυνοι για ζώνες εντός του ορισμένου διαστήματος. Εν αντιθέσει, στο MIDAS αξιοποιείται η δομή του κατανεμημένου k-d δέντρου προκειμένου να ταυτοποιηθούν παράλληλα οι σχετικοί με το ερώτημα κόμβοι με παρόμοιο τρόπο με τον shower αλγόριθμο στο [46] για μονοδιάστατα δεδομένα.

Ο Αλγόριθμος 4 δίνει τις λεπτομέρειες της δράσης που αναλαμβάνει ο ομότιμος κόμβος u με την παραλαβή ενός ερωτήματος εύρους για την περιοχή $Q = [\vec{\ell}, \vec{h}]$ που εκδόθηκε από τον κόμβο w. Πρώτα ο κόμβος u εντοπίζει τις τοπικά αποθηκευμένες πλειάδες εντός του διαστήματος

που ορίζεται με το ερώτημα Q και τις αποστέλει στον εκδότη του ερωτήματος w (γραμμές 1–3). Στη συνέχεια ο κόμβος u εξετάζει τα συμμετρικά υποδέντρα όπως αυτά προκύπτουν από την μερική εικόνα που έχει για το δίκτυο μέσω των πινάκων *sdim* και *split* (γραμμές 4–15) και κάθε φορά που ο υπό εξέταση σύνδεσμος αφορά περιοχές που επικαλύπτουν το πολυδιάστατο διάστημα του ερωτήματος (γραμμές 6 και 10) προωθείται στον ανάλογο σύνδεσμο το υπο-ερώτημα που αφορά την τομή της περιοχής που αντιπροσωπεύει ο σύνδεσμος με το ερώτημα Q (γραμμές 7–9 και 11–13).

Algorithm 4: u.Range $(\vec{\ell},\vec{h},w)$: Ο κόμβος u επεξεργάζεται το ερώτημα εύρους $Q=[\vec{\ell},\vec{h}]$ που εκδίδει ο w.

```
1  if u.Overlaps (ℓ⃗, h⃗) then
2      u.Send_to (w, u.Get_vals (ℓ⃗, h⃗));
3  for j ← 0 to u.depth do
4      d ← u.sdim[j] ;
5      if u.id[j] = 0 and u.split[j] < h⃗[d] then
6          ℓ⃗′ ← ℓ⃗ ;
7          ℓ⃗′[d] ← u.split[j] ;
8          u.link[j].Range (ℓ⃗′, h⃗, w) ;
9      else if u.id[j] = 1 and u.split[j] > ℓ⃗[d] then
10         h⃗′ ← h⃗ ;
11         h⃗′[d] ← u.split[j] ;
12         u.link[j].Range (ℓ⃗, h⃗′, w) ;
```

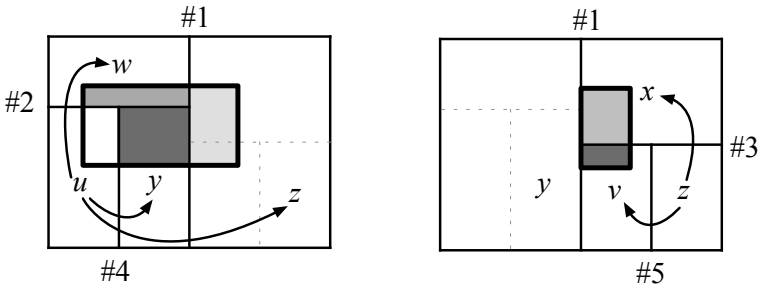
Στο Σχήμα 4.2 παρουσιάζουμε ένα παράδειγμα ερωτήματος το οποίο εκδίδεται από τον κόμβο u. Αρχικά, ο κόμβος u εκτελεί τον Αλγόριθμο 4 για το διάστημα που δείχνουμε με έντονη σκούρα γραμμή στο Σχήμα 4.2(α΄). Ο κόμβος u ανακτά τις πλειάδες εντός της δικής του ζώνης που ανήκουν στο διάστημα. Επιπλέον, για κάθε ένα από τα συμμετρικά υποδέντρα προωθεί το τμήμα το αρχικού ερωτήματος που αφορά τη συγκεκριμένη περιοχή με μηνύματα να δείχνονται ως βέλη στο σχήμα. Για παράδειγμα, ο κόμβος z που αντιστοιχεί στον σύνδεσμο του u για το υποδέντρο που είναι συμμετρικό ως προς τη ρίζα, λαμβάνει το ερώτημα που αντιστοιχεί στην ελαφρώς γραμμοσκιασμένη περιοχή.

Οι κόμβοι w, y και z λαμβάνουν τα ανάλογα υπο-ερωτήματα από τον κόμβο u για διαστήματα που πέφτουν εντός της περιοχής που αντιπροσωπεύουν ως σύνδεσμοι του u για να τα προωθήσουν περαιτέρω. Στο Σχήμα 4.2(β΄) δείχνουμε την επεξεργασία του ερωτήματος που λαμβάνει χώρα στον ομότιμο κόμβο z. Παρατηρούμε ότι το συγκεκριμένο διάστημα δεν επικαλύπτει την ζώνη του κόμβου z κι άρα δεν διαθέτει δεδομένα που να αφορούν το ερώτημα. Τότε ο κόμβος z προωθεί τα ανάλογα υπο-ερωτήματα στους συνδέσμους που αφορούν το ερώτημα που έλαβε. Συνεπώς, ο κόμβος z αποστέλει τα ανάλογα υπο-ερωτήματα στους κόμβους v και x που αντιστοιχούν στις γραμμοσκιασμένες περιοχές του Σχήματος 4.2(β΄) (2η επαναπροώθηση). Τέλος, οι κόμβοι v και x επεξεργάζονται τοπικά τα ερωτήματα χωρίς να τα προωθήσουν περαιτέρω.

Όπως εξηγήσαμε στο παράδειγμα του Σχήματος 4.2, το συγκεκριμένο ερώτημα χρειάστηκε μόλις δύο βήματα/επαναπροωθήσεις έως ότου να απαντηθεί πλήρως. Στο πρώτο βήμα, το ερώτημα του κόμβου u φτάνει στους y, w, και z, και στο δεύτερο τους κόμβους v και x. Το ακόλουθο λήμμα αποδεικνύει ότι ο αναμενόμενος αριθμός από επαναπροωθήσεις είναι λογαριθμικός ως προς το μέγεθος του δικτύου.

Λήμμα 6 *Ο αναμενόμενος αριθμός από επαναπροωθήσεις για ένα ερώτημα εύρους φράσσεται από το* $O(\log n)$.

Απόδειξη 6 *Δείχνουμε ότι ο μεγιστος απαιτούμενος αριθμός από επαναπροωθήσεις ισούται με το βάθος του κατανεμημένου k-d δέντρου.*



(α′) επεξεργασία και προώθηση από τον κόμβο u

(β′) επεξεργασία και προώθηση από τον κόμβο z

Σχήμα 4.2: Παράδειγμα επεξεργασίας ερωτήματος εύρους για δύο διαστάσεις.

Ας θεωρήσουμε τον ομότιμο κόμβο v που λαμβάνει από τον κόμβο u ένα ερώτημα εύρους για το διάστημα $Q = [\vec{\ell}, \vec{h}]$ κι ότι ο κόμβος v είναι για τον u ο σύνδεσμος για το συμμετρικό υποδέντρο βάθους k. Εκ κατασκευής το ληφθέν υπο-ερώτημα μπορεί να απαντηθεί πλήρως στους κόμβους που αντιστοιχούν στο συγκεκριμένο υποδέντρο κι άρα η περιοχή που αφορά δεν επικαλύπτει κανέναν κόμβο μικρότερου βάθους. Συνεπώς, ο v οφείλει να προωθήσει το ερώτημα στους συνδέσμους του οι οποίοι αντιστοιχούν στα υποδέντρα τα οποία έχουν βάθος μεγαλύτερο του k.

Οπότε ο συνολικός αριθμός δυνατόν επαναπροωθήσεων φράσσεται από το βάθος του κατανεμημένου k-d δέντρου. Το γεγονός ότι το αναμενόμενο βάθος του δέντρου είναι $O(\log n)$ (από το Λήμμα 3) ολοκληρώνει την απόδειξη. ■

4.3 Ερωτήματα Κοντινότερων Γειτόνων

Δεδομένου ενός σημείου $\vec{c}$ και μίας παραμέτρου k, ένα ερώτημα κοντινότερων γειτόνων αιτείται των k κοντινότερων πλειάδων στο $\vec{c}$ σύμφωνα με μία μετρική απόστασης όπως η Ευκλείδια. Στην ενότητα αυτή προτείνουμε μεθόδους για την επεξεργασία στο MIDAS τέτοιου είδους ερωτημάτων για οποιαδήποτε μετρική απόσταση που μπορεί να οριστεί σε έναν d-διάστατο Ευκλείδιο χώρο I.

Αυτού του είδους τα ερωτήματα προφανώς συνιστούν μεγαλύτερη πρόκληση από τους τύπους ερωτημάτων που συναντήσαμε στις προηγούμενες ενότητες, όπως για παράδειγμα τα ερωτήματα εύρους, κυρίως λόγω του γεγονότος ότι δεν είναι γνωστό εκ των προτέρων το εύρος της αναζήτησης ή αλλιώς η μέγιστη απόσταση γύρω από το κέντρο του ερωτήματος στην οποία υπάρχουν έως k πλειάδες. Αυτό σημαίνει ότι η έκταση της αναζήτησης δεν είναι προκαθορισμένη όπως γίνεται με τα ερωτήματα εύρους. Για την επεξεργασία τέτοιου είδους ερωτημάτων προτείνουμε στην ενότητα αυτή δύο διαφορετικές τεχνικές. Η πρώτη, ονόματι eager processing, είναι πιο γρήγορη αφού απαιτεί $O(\log n)$ βήματα, ενώ η δεύτερη, ονόματι iterative processing, που αποσκοπεί στην καλύτερη αξιοποίηση των πόρων του δικτύου, απαιτεί $O(\log^2 n)$ βήματα.

4.3.1 Eager Processing

Στόχος αυτής της προσέγγισης είναι η εκτίμηση με κατανεμημένο τρόπο της απόστασης των k κοντινότερων στοιχείων από το κέντρο $\vec{c}$ του ερωτήματος. Η συγκεκριμένη μέθοδος προσπελαύνει παράλληλα τους κόμβους που βρίσκονται κάθε φορά εντός της εκτίμησης που έχει γίνει για το εύρος της αναζήτησης. Κάθε ερώτημα συνοδεύεται από μία εγγύηση (M, R), που σημαίνει ότι M πλειάδες εντός απόστασης R από το $\vec{c}$ έχουν ήδη ανακτηθεί. Η μέθοδος τερματίζεται όταν το M φτάνει το k *και* όλες οι πλειάδες εντός ακτίνας R από το $\vec{c}$ έχουν ανακτηθεί. Όταν αυτό συμβαίνει τότε το R θα έχει πάρει την τιμή της απόστασης της k-ιοστής κοντινότερης πλειάδας από το $\vec{c}$ και το τελικό αποτέλεσμα θα έχει ανακτηθεί.

Ας θεωρήσουμε τον ομότιμο κόμβο w που εκδίδει ένα ερώτημα k κοντινότερων γειτόνων. Αρχικά θα πρέπει να εντοπίσει τον κόμβο, έστω z, ο οποίος είναι υπεύθυνος για το $\vec{c}$ με μία διαδικασία παρόμοια με αυτήν που ακολουθείται για τα ερωτήματα σημείων. Θα αναφερόμαστε από εδώ και πέρα στον κόμβο z ως *συντονιστή*. Όλοι οι κόμβοι που λαμβάνουν ένα τέτοιο ερώτημα, συμπεριλαμβανομένου και του συντονιστή, εκτελούν τον Αλγόριθμο 5. Το ερώτημα συμπεριλαμβάνει τις εξής παραμέτρους: (i) το κέντρο $\vec{c}$ του ερωτήματος, (ii) το αναμενόμενο μέγεθος του αποτελέσματος k, (iii) την εγγύηση (M,R), (iv) το φράγμα βάθους D, (v) καθώς και τις διευθύνσεις του εκδότη του ερωτήματος w αλλά και του συντονιστή κόμβου z. Πιο αναλυτικά, ο ρόλος της παραμέτρου D είναι να περιοριστεί η προώθηση του αιτήματος εντός του υποδέντρου βάθους D, εξασφαλίζοντας έτσι ότι κανένας κόμβος δε θα λάβει το αίτημα περισσότερες της μίας φορές. Ο κόμβος συντονιστής του αιτήματος είναι αυτός που αρχικοποιεί τις παραμέτρους με τις τιμές $M = 0$, $R = 0$, και $D = 0$. Στηριζόμενος στην τοπική/μερική γνώση που διαθέτει ένας κόμβος εκτελεί τα εξής βήματα: (1) ανακτά και μεταδίδει το δικό του μέρος του αποτελέσματος, (2) ενημερώνει την εγγύηση (M',R'), όπου $M' \geq \min\{M,K\}$ ενώ το R' είναι μία εκτίμηση της πραγματικής απόστασης των k πλειάδων από το $\vec{c}$ κι ενημερώνεται σταδιακά καθώς εκτελείται η επεξεργασία του ερωτήματος, και (3) προωθεί νέα αιτήματα για περαιτέρω επεξεργασία προς τους κόμβους που βρίσκονται εντός R' απόστασης από το $\vec{c}$.

Algorithm 5: u.NN $(\vec{c},k,M,R,D,w,z)$: Ο κόμβος u επεξεργάζεται ένα k-NN ερώτημα με κέντρο το $\vec{c}$ που εκδίδει ο κόμβος w και συντονίζει ο κόμβος z. M πλειάδες εντός απόστασης R έχουν ήδη ανακτηθεί. Το αίτημα μπορεί να προωθηθεί εάν χρειάζεται μόνο προς συνδέσμους του u που αντιστοιχούν στο υποδέντρο στο οποίο ανήκει βάθους D.

```
1  insert in S up to K closest to c⃗ tuples within distance R;
2  if k − |S| = 0 then
3  |  //case I;
4  |  R ← r(S);
5  |  M ← k;
6  else if M ≥ k − |S| then
7  |  //case II;
8  |  M ← M + |S|;
9  else if M < k − |S| then
10 |  //case III;
11 |  insert in S up to k−|S|−M closest to c⃗ tuples;
12 |  R ← max{R, r(S)};
13 |  M ← M + |S|;
14 u.Send_to (w, S) ;
15 u.Send_to (z, |S|, r(S)) //only for iterative processing;
16 for j ← D + 1 to u.depth do
17 |  if M < k or u.link[j].Overlaps (c⃗,R) then
18 |  |  u.link[j].NN (c⃗,k,M,R, j,w,z) ;
```

Ας θεωρήσουμε τον κόμβο u που λαμβάνει ένα τέτοιο ερώτημα κι εκτελεί τον Αλγόριθμο 5. Πρώτα, ο κόμβος u ανακτά τις τοπικά αποθηκευμένες πλειάδες εντός ακτίνας R από το κέντρο του ερωτήματος $\vec{c}$ και τις προσθέτει στο ενδιάμεσο αποτέλεσμα που δεν έχει πάρει ακόμα την τελική του μορφή (γραμμή 1). Φυσικά στην περίπτωσή όπου υπάρχουν άνω των k πλειάδων εντός της ορισμένης ακτίνας τότε μόνο οι κοντινότερες k ενσωματώνονται στο αποτέλεσμα. Στη συνέχεια, ο κόμβος λήπτης του αιτήματος ανανεώνει την εγγύηση (M,R) ενημερώνοντάς την ανάλογα ώστε να ληφθεί υπόψην κι η δική του συνεισφορά. Είναι πιθανόν να χρειαστεί να ανακτηθούν επιπλέον πλειάδες ανάλογα με το μέγεθος του ενδιάμεσου αποτελέσματος. Ειδικότερα υπάρχουν τρία ενδεχόμενα.

Σύμφωνα με την πρώτη περίπτωση το αποτέλεσμα ήδη εμπεριέχει k πλειάδες (γραμμές 2–5). Επιπλέον, ορίζουμε $r(S)$ την απόσταση από το κέντρο του ερωτήματος $\vec{c}$ της μακρύτερης

πλειάδας στο τρέχον αποτέλεσμα. Ο κόμβος u θα πρέπει να επαναδιατυπώσει την εγγύηση (M,R) που είχε λάβει ως $(k,r(S))$ αφού στο ενημερωμένο αποτέλεσμα υπάρχουν πλέον οι k κοντινότερες πλειάδες εντός $r(S)$ απόστασης (γραμμές 4, 5) κι έτσι να μειώσει την αρχική εκτίμηση για το εύρος της αναζήτησης. Στη δεύτερη περίπτωση (γραμμές 6–8) το αποτέλεσμα επαυξημένο με τις τοπικά αποθηκευμένες εγγραφές είναι πλέον μεγαλύτερο από k, δηλαδή $M \geq k - |S|$. Όλα τα στοιχεία είναι εντός ακτίνας R από το κέντρο. Οπότε, η παράμετρος M παιρνει την τιμή $M + |S|$ (γραμμή 8).

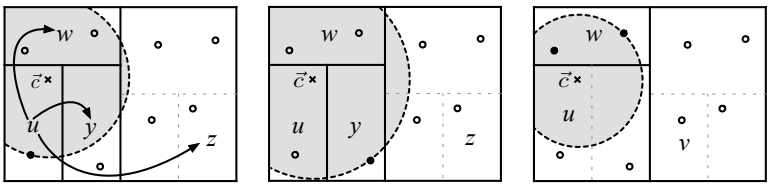
Για την τρίτη περίπτωση (γραμμές 9-13), το μέγεθος του αποτελέσματος εξακολουθεί να είναι μικρότερο του k παρά την προσαύξηση του αποτελέσματος με τις τοπικά αποθηκευμένες πλειάδες εντός R ακτίνας, δηλαδή $M < k - |S|$. Στην περίπτωση αυτή ο κόμβος θα πρέπει να εμπλουτίσει το αποτέλεσμα με πλειάδες σε μεγαλύτερη απόσταση και να ενημερώσει την εγγύηση κατάλληλα για την επέκταση του χώρου αναζήτησης. Οπότε οι $k - |S| - M$ κοντινότερες πλειάδες σε ακτίνα μεγαλύτερη του R ανακτόνται (γραμμή 11) κι έτσι η εγγύηση παίρνει τις ανάλογες τιμές στις γραμμές 12, 13. Σε αυτό το σημείο λοιπόν η εγγύηση (M,R) του αποτελέσματος συμπεριλαμβάνει τη συνεισφορά του ομότιμου κόμβου u στην επεξεργασία του ερωτήματος.

Συνοψίζοντας, η τιμή της παραμέτρου M που ποσοτικοποιεί την τοπική συνεισφορά σε εγγραφές στο αποτέλεσμα είναι ίση με k στην πρώτη περίπτωση όπου το μέγεθος του αποτελέσματος παρέμεινε το ίδιο αν και πιθανόν να χρειάστηκε να αντικατασταθούν κάποιες εγγραφές του με κοντινότερες στο $\vec{c}$ ενώ ταυτόχρονα η εκτίμηση του εύρους αναζήτησης R μειώνεται κατάλληλα ώστε να αναλογεί στο ενημερωμένο αποτέλεσμα, όπως γίνεται και για την δεύτερη περίπτωση, αν και το μικρότερο αποτέλεσμα θα πρέπει να αυξηθεί με τα τουλάχιστον $k - |S|$ τοπικά αποθηκευμένα κοντινότερα σημεία. Η ίδια εκτίμηση για την τρίτη περίπτωση αυξάνεται ώστε να καλύπτει το μακρύτερο σημείο που πλέον περιλαμβάνεται στο αποτέλεσμα, το οποίο αναγκαστικά ξεπερνά την αρχική εκτίμηση ώστε να μπορέσει το αποτέλεσμα να φτάσει τις k πλειάδες.

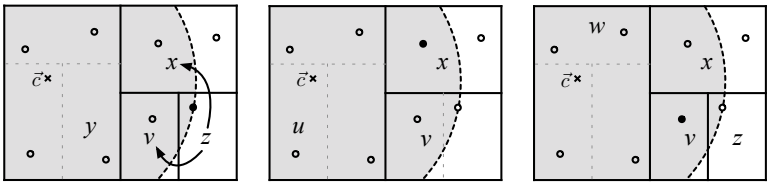
Επακολούθως, όλες οι πλειάδες του αποτελέσματος S αποστέλονται στον κόμβο w που εξέδωσε αρχικά το ερώτημα (γραμμή 14). Η γραμμή 15 εκτελείται μόνον από τον iterative processing αλγόριθμο που θα εξηγήσουμε στην αμέσως επόμενη ενότητα. Το τελευταίο τμήμα της μεθόδου (γραμμές 16–18) αποστέλει το ερώτημα ενημερωμένο κατάλληλα για περαιτέρω επεξεργασία προς τους σχετικούς συνδέσμους που αντιπροσωπεύουν τα συμμετρικά υποδέντρα βάθους μεγαλύτερου του D (γραμμή 16). Το ερώτημα αποστέλεται σε έναν από τους συνδέσμους του κόμβου u, εάν περισσότερες πλειάδες απαιτούνται, ήτοι $M < k$, ή όταν ο σύνδεσμος εμπεριέχει πλειάδες εντός της έως τώρα εκτίμησης R του εύρους αναζήτησης από το κέντρο του ερωτήματος $\vec{c}$ (γραμμή 17). Κάθε φορά που επαναπροωθείται το μήνυμα, ανατίθεται στην αντίστοιχη παράμετρο βάθους D η τιμή που αντιστοιχεί στο βάθος του αντίστοιχου υποδέντρου j. Με τον τρόπο αυτόν εξασφαλίζουμε ότι η περαιτέρω επεξεργασία που οφείλεται στο συγκεκριμένο αντίγραφο του ερωτήματος δεν θα επεκταθεί πέρα από τους κόμβους που ανήκουν στο υποδέντρο που αντιπροσωπεύεται από τον σύνδεσμο στη j-οστή θέση του πίνακα δρομολόγησης. Με την τεχνική αυτή κανένας κόμβος δεν καταλήγει να λαμβάνει πολλαπλά αντίτυπα του ίδιου ερωτήματος.

Το Σχήμα 4.3 δείχνουμε ένα παράδειγμα εκτέλεσης του eager processing αλγορίθμου για ένα 2-NN ερώτημα. Οι μικροί κύκλοι αναπαριστούν τα αποθηκευμένα δεδομένα ενώ ο σταυρός αναπαριστά το κέντρο $\vec{c}$ του ερωτήματος. Έστω ότι ο κόμβος u εκδίδει το ερώτημα. Δεδομένου ότι το $\vec{c}$ ανήκει στον u, ο κόμβος u θα εκτελέσει τον Αλγόριθμο 5 με αρχικές παραμέτρους $M = R = 0$. Το αποτέλεσμα S παραμένει άδειο μετά την εκτέλεση της γραμμής 1. Εφόσον το παράδειγμα εμπίπτει στην τρίτη περίπτωση καθώς μόνο μία πλειάδα υπάρχει εντός του κόμβου u στο Σχήμα 4.3(α′), θα έχουμε $M = 1$ κι η εκτίμηση R του εύρους αναζήτησης τίθεται ίση με την ακτίνα του σκιαγραφημένου κύκλου του Σχήματος 4.3(α′). Επακόλουθα, αφού $M < k$, ο κόμβος u προωθεί το ερώτημα στους κόμβους που γνωρίζει y, w και z, όπως δείχνουν και τα βέλη στο Σχήμα 4.3(α′), με την παράμετρο D να παίρνει την τιμή 3, 2 και 1, αντίστοιχα για κάθε μήνυμα.

Στη συνέχεια εξετάζουμε την επεξεργασία του ερωτήματος από τον κόμβο y μετά την πρώτη προώθηση του ερωτήματος από τον κόμβο u. Παρατηρούμε ότι ο κόμβος y δεν έχει πλειάδες εντός της προσωρινής εκτίμησης του εύρους αναζήτησης R, καθώς υποδεικνύει ο



(α′) στον κόμβο u (βήμα 0) (β′) στον κόμβο y (βήμα 1) (γ′) στον κόμβο w (βήμα 1)



(δ′) στον κόμβο z (βήμα 1) (ε′) στον κόμβο x (βήμα 2) (στ′) στο κόμβο v (βήμα 2)

Σχήμα 4.3: Επεξεργασία ερωτημάτων 2-NN με τη μέθοδο eager processing.

δίσκος στο Σχήμα 4.3(α′). Οπότε, εφαρμόζουμε την τρίτη περίπτωσή: ο κόμβος y ανακτά την πιο κοντινή πλειάδα, όπως φαίνεται στο Σχήμα 4.3(β′)), αυξάνει την παράμετρο M κατά 1 και θέτει την εκτίμηση R σύμφωνα με την ακτίνα του επεκταμένου δίσκου στην Εικόνα 4.3(β′). Ο κόμβος y δε θα επαναπροωθήσει το ερώτημα περαιτέρω καθώς η παράμετρος D ισούται με το βάθος του στο κατανεμημένο k-d δέντρο.

Παράλληλα λαμβάνει χώρα κι η επεξεργασία του ερωτήματος από πλευράς του κόμβου w. Υπάρχουν όμως δύο πλειάδες εντός του κόμβου w οι οποίες πέφτουν εντός της προηγουμένως ορισμένης ακτίνας R. Οπότε εφαρμόζουμε την πρώτη περίπτωση καθώς $|S| = 2 = K$. Η παράμετρος M γίνεται ίση με δύο καθώς η εκτίμηση R μειώνεται στην απόσταση της δεύτερης πλειάδας από το $\vec{c}$ στο Σχήμα 4.3(γ′). Και πάλι ο κόμβος y δε θα επαναπροωθήσει το ερώτημα περαιτέρω καθώς η παράμετρος D ισούται με 2, δηλαδή το βάθος του στο κατανεμημένο k-d δέντρο.

Στο Σχήμα 4.3(δ′) δείχνουμε την επεξεργασία του ερωτήματος στον κόμβο z, ο οποίος είναι επίσης μόλις 1 επαναπροώθηση μακριά από τον εκδότη του ερωτήματος. Εφαρμόζει την τρίτη περίπτωσή της μεθόδου επεξεργασίας, καθώς δεν επικαλύπτει το αρχικό εύρος ζώνης της αναζήτησης. Συνεπώς, ο κόμβος z ανακτά την πιο κοντινή πλειάδα στο $\vec{c}$ που έχει αποθηκευμένη τοπικά και θέτει $M = 2$ καθώς επίσης αυξάνει την αρχική εκτίμηση του εύρους αναζήτησης R στην ακτίνα του σκιαγραφημένου δίσκου του Σχήματος 4.3(δ′). Καθώς έχουμε ότι $D = 1$ και $z.depth = 3$, ο κόμβος z αποστέλει ένα αίτημα με την ενημερωμένη εγγύηση (M,R) στους συνδέσμους του που αντιστοιχούν σε μεγαλύτερα βάθη, δηλαδή τους κόμβους x και v για βάθη 2 και 3, αντίστοιχα.

Τώρα, η επεξεργασία του ερωτήματος από τους ομότιμους κόμβους x και v λαμβάνει χώρα παράλληλα κατά τη δεύτερη επαναπροώθηση του ερωτήματος, όπως δείχνουμε στα Σχήματα 4.3(ε′) και 4.3(στ′). Η δεύτερη περίπτωση του Αλγορίθμου 5 ισχύει για αμφότερους κόμβους καθώς $M = 2$. Επειδή υπάρχουν εγγραφές εντός της εκτίμησης R του εύρους αναζήτησης όπως αυτή ορίστηκε στο Σχήμα 4.3(δ′), αυξάνεται η παράμετρος M κατά 1 ενώ το R παραμένει ως έχει στα Σχήματα 4.3(ε′) και 4.3(στ′). Οι κόμβοι x και v δεν προωθούν περαιτέρω το ερώτημα καθώς η παράμετρος D ισούται με το βάθος τους.

Εν κατακλείδι, ο κόμβος που αρχικά εξέδωσε το ερώτημα θα έχει λάβει εφτά εγγραφές συνολικά, όπως δείχνουμε στο Σχήμα 4.3. Ανάμεσά τους θα επιλέξει τις δύο κοντινότερες στο κέντρο $\vec{c}$ του ερωτήματος που τυχαίνει να είναι εκείνες που ανακτήθηκαν από τον κόμβο w στο πρώτο μόλις βήμα.

Στα επόμενα δύο λήμματα υπολογίζουμε την πολυπλοκότητα του eager processing αλγορίθμου κι αποδεικνύουμε την ορθότητά του.

Λήμμα 7 *Ο αναμενόμενος αριθμός από επαναπροωθήσεις για τον eager processing αλγόριθμο για την ανάκτηση των κοντινότερων γειτόνων είναι* $O(\log n)$.

Απόδειξη 7 *Αρχικά, ο κόμβος εκδότης εντοπίζει τον κόμβο-συντονιστή που είναι υπεύθυνος για το κεντρο $\vec{c}$ του ερωτήματος κοντινότερων γειτόνων σε $O(\log n)$ βήματα. Στη συνέχεια, ο κόμβος-συντονιστής αρχίζει την εκτέλεση του Αλγορίθμου 5. Με την παραλαβή ενός αιτήματος με παράμετρο D, ένας κόμβος θα προωθήσει το ερώτημα μόνο προς τους κόμβους που έχουν βάθος μεγαλύτερο από D κι άρα στο αντίστοιχο συμμετρικό υποδέντρο του κόμβου που του το προώθησε βάθους D. Για κάθε επαναπροώθηση αυξάνεται η παράμετρος αυτή κατά τόσο όσο είναι το βάθος του υποδέντρου στο οποίο προωθείται το ερώτημα (γραμμές 14–18, Αλγόριθμος 5). Οπότε, σε κάθε επαναπροώθηση οι αιτήσεις κατευθύνονται σε ολοένα μικρότερα υποδέντρα. Στη χειρότερη περίπτωση, ο αριθμός από επαναπροωθήσεις είναι ίσος με το βάθος του k-d δέντρου. Το γεγονός ότι το αναμενόμενο βάθος του δέντρου είναι $O(\log n)$ από το Λήμμα 3) ολοκληρώνει την απόδειξη.* ■

Λήμμα 8 *Ο eager processing αλγόριθμος ανακτά τις k κοντινότερες πλειάδες από το κέντρο $\vec{c}$ του ερωτήματος.*

Απόδειξη 8 *Θα χρησιμοποιήσουμε την τεχνική της δια της ατόπου απαγωγής για να δείξουμε ότι η απάντηση που λαμβάνει ο εκδότης του ερωτήματος αποτελεί υπερσύνολο των k κοντινότερων γειτόνων. Έστω ότι υπάρχει μία εγγραφή $\vec{\tau}$ η οποία αν και συγκαταλέγεται στις k κοντινότερες δεν επιστρέφεται στον εκδότη. Διακρίνουμε δύο ενδεχόμενα ανάλογα με το εάν κάποιο αντίγραφο του ερωτήματος θα προσπελάσει ή όχι τον κόμβο u στον οποίον είναι αποθηκευμένη η συγκεκριμένη εγγραφή.*

Για το πρώτο ενδεχόμενο, έστω ότι ο κόμβος u προσπελαύνεται κατά την εκτέλεση του Αλγορίθμου 5 αλλά δεν συμπεριλαμβάνεται στο αποτέλεσμα S. Για να συμβεί αυτό θα πρέπει το $\vec{\tau}$ να μη βρίσκεται εντός της εκτίμησης R για το εύρος της αναζήτησης. Για την περίπτωση I (γραμμές 2–5) σημαίνει ότι ο κόμβος u έχει k εγγραφές οι οποίες είναι εγγύτερα στο $\vec{c}$ από ότι είναι το $\vec{\tau}$ — κι άρα έχουμε αντίφαση αφού αυτό συνεπάγεται ότι το $\vec{\tau}$ δεν ανήκει στους κοντινότερους γείτονες. Για την περίπτωση II (γραμμές 6–8) θα έχουμε ότι υπάρχουν περισσότερες των k εγγραφών ($M + |S| \geq K$) εντός της δεδομένης εκτίμησης R κι άρα θα πρέπει να μειωθεί αναλόγως. Ομοίως, η περίπτωση III (γραμμές 9–13) ερμηνεύεται ότι υπάρχουν k πλησιέστερες εγγραφές από το $\vec{\tau}$, που έρχεται σε αντίφαση, αφού χωρίς να συμπεριλαμβάνεται το στοιχείο $\vec{\tau}$ του κόμβου u έχουμε ότι $M + |S| = K$. Για τις περιπτώσεις II και III αυτο μπορεί να συμβεί μόνο στο ενδεχόμενο εσφαλμένης εγγύησης (M,R), για παράδειγμα να μην υπάρχουν τουλάχιστον M εγγραφές εντός R ακτίνας από το $\vec{c}$.

Οπότε αρκεί να δείξουμε ότι δεδομένου της λήψης μία έγκυρης εγγύησης και της όποιας τοπικής επεξεργασίας του ερωτήματος, ο Αλγόριθμος 5 δημιουργεί μία εξίσου έγκυρη εγγύηση. Αυτό ισχύει εκ κατασκευής για την περίπτωση I. Για περιπτώσεις II και III επίσης ισχύει επειδή το M αυξάνεται ανάλογα με το πόσα στοιχεία προστίθονται στο αποτέλεσμα κι έτσι η εκτίμηση R του εύρους αναζήτησης ενημερώνεται σωστά στην απόσταση της μακρύτερης πλειάδας του προσαυξημένου αποτελέσματος. Σε κάθε περίπτωση όμως το νέο ζεύγος (M,R) παραμένει έγκυρο, που σημαίνει ότι το πρώτο ενδεχόμενο είναι αδύνατον αφού όλα του τα σκέλη έρχονται σε αντίθεση.

Για το δεύτερο ενδεχόμενο, έχουμε δύο περιπτώσεις στις οποίες ο κόμβος u δεν προσπελαύνεται. Είτε η συνθήκη στη γραμμή 17 δεν ισχύει για το υποδέντρο που εμπεριέχει τον κόμβο u, είτε το συγκεκριμένο υποδέντρο δεν εξετάζεται. Η συνθήκη δεν ισχύει εάν $M \geq k$ και το υποδέντρο στο οποίο υφίσταται ο κόμβος u δεν βρίσκεται εντός ακτίνας R, που συνεπάγεται ότι ούτε το $\vec{\tau}$ είναι κι άρα δεν ανήκει στους κοντινότερους γείτονες. Η άλλη περίπτωση, σύμφωνα με την οποία αγνοούνται κατα τις επαναπροωθήσει τα εκάστοτε υποδέντρα τα οποία εμπεριέχουν τον κόμβο u δεν δύναται να συμβεί δεδομένου ότι ο εκδότης του ερωτήματος θέτει στην παράμετρο D αρχικά την τιμή 0, κι όλοι οι κόμβοι εξετάζουν τα συμμετρικά τους υποδέντρα για μεγαλύτερο βάθος διαδοχικά. ■

4.3.2 Iterative processing

Ομοίως με τον eager processing αλγόριθμο, η μέθοδος που παρουσιάζουμε στην ενότητα αυτή, ονόματι iterative processing, υπολογίζει ένα άνω φράγμα για την απόσταση της k-ιοστής πλειάδας του αποτελέσματος από το κέντρο $\vec{c}$ του ερωτήματος. Αυτή τη φορά όμως ο

κόμβος-συντονιστής της επεξεργασίας του ερωτήματος, που παρεμπιπτόντως είναι ο κόμβος που είναι υπεύθυνος για το κέντρο $\vec{c}$ του ερωτήματος, ενορχηστρώνει τη διαδικασία με τέτοιο τρόπο ώστε ο συνολικός αριθμός μηνυμάτων να ελαχιστοποιηθεί. Η κεντρική ιδέα της μεθόδου βασίζεται στο να κατευθυνθεί η αναζήτηση προς τους κόμβους οι οποίοι βρίσκονται εγγύτερα στο $\vec{c}$ κι έπειτα να επεκταθεί στους γειτονικούς κόμβους τόσο όσο χρειάζεται προκειμένου να γεμίσει το αποτέλεσμα με τουλάχιστον k πλειάδες. Ειδικότερα, σε κάθε γύρο ο κόμβος-συντονιστής περιορίζει την προώθηση των αιτημάτων διαδοχικά σε ολοένα και μεγαλύτερα υποδέντρα τα οποία όμως πάντοτε εμπεριέχουν και το κέντρο $\vec{c}$ του ερωτήματος. Η διαδικασία αυτή τερματίζεται όταν σχηματιστεί η απάντηση των k κοντινότερων γειτόνων ή ολόκληρη η κατανεμημένη δομή έχει προσπελαστεί.

Αρχικά, ο εκδότης του ερωτήματος w εντοπίζει τον κόμβο z που είναι υπεύθυνος για το κέντρο $\vec{c}$ του ερωτήματος. Στη συνέχεια, ο κόμβος z θα εκτελέσει τον Αλγόριθμο 6 ως συντονιστής της διαδικασίας προωθώντας το ερώτημα στους σχετικούς κόμβους που θα εκτελέσουν τον Αλγόριθμο 5 της προηγούμενης ενότητας, συγκεντρώνοντας κι επεξεργάζοντας τα ενδιάμεσα αποτελέσματα τους κι επαναπροωθώντας το ερώτημα ανάλογα στους εναπομείναντες σχετικούς κόμβους. Η iterative processing μέθοδος διαφοροποιείται της προηγούμενης ως προς το ότι ο κάθε κόμβος που λαμβάνει μέρος στην κατανεμημένη επεξεργασία του ερωτήματος επιβάλλεται να αποστείλει την τοπική εγγύηση $(|S|, r(S))$ για το ενδιάμεσο αποτέλεσμα στο συντονιστή κόμβο (γραμμή 15, Αλγόριθμος 5). Δεδομένου των λαμβανομένων ζευγών (M,R) μεγέθους αποτελέσματος - εκτίμησης εύρους αναζήτησης, ο συντονιστής κόμβος είναι σε θέση να διαπιστώσει για το εάν είναι απαραίτητος ένας ακόμα γύρος επεξεργασίας.

Στη συνέχεια, θα μελετήσουμε το ρόλο που παίζει ο συντονιστής κόμβος στην κατανεμημένη επεξεργασία του ερωτήματος. Εν αρχή, ο κόμβος z ανακτά τις έως k κοντινότερες πλειάδες στο κέντρο του ερωτήματος που είναι αποθηκευμένες τοπικά (Αλγόριθμος 6, γραμμή 1). Έπειτα υπολογίζει την τοπική εγγύηση για το τοπικό αυτό αποτέλεσμα και την εισάγει στο σύνολο G (γραμμή 3) που εμπεριέχει τις επί μέρους εγγυήσεις των κόμβων που συμμετέχουν. Επακολούθως, η επεξεργασία του ερωτήματος διεκπεραιώνεται σε διαδοχικούς γύρους (γραμμές 4–16) όπου ξεκινώντας από το βάθος του κόμβου z, η τιμή της παραμέτρου D μειώνεται κατά ένα σε κάθε γύρο έως ότου έχει σχηματιστεί πλήρως το αποτέλεσμα.

Algorithm 6: z.NN_manage $(\vec{c}, K, w)$: Ο κόμβος z συντονίζει ένα K-NN ερώτημα με κέντρο το $\vec{c}$ που εκδίδει ο κόμβος w.

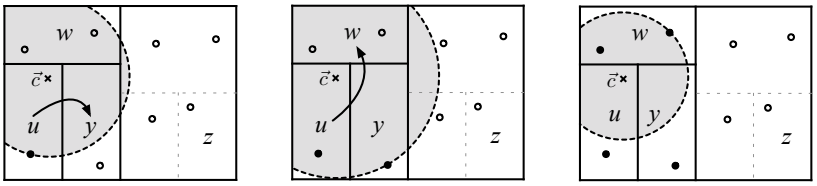
```
1  insert in S up to K closest to c⃗ tuples;
2  z.Send_to (w, S);
3  insert in G local guarantee (|S|, r(S));
4  for D ← z.depth down to 1 do
5      sort entries in G according to distance;
6      M ← 0;
7      for each (M_i, R_i) ∈ G do
8          M ← M + M_i;
9          R ← R_i;
10         if M ≥ K then break;
11     if M < K or z.link[D].Overlaps (c⃗, R) then
12         z.link[D].NN (c⃗, K, M, R, D, w, z);
13         repeat
14             z.Receive (M_i, R_i);
15             insert in G local guarantee (M_i, R_i);
16         until messages from D hops away are received;
```

Στην αρχή κάθε γύρου κατασκευάζεται μία ολική εγγύηση (M,R) που συνοψίζει όλες τις τοπικές εγγυήσεις που ελήφθησαν στον προηγούμενο γύρο επεξεργασίας (γραμμές 4–10). Ειδικότερα, οι τοπικές εγγυήσεις στο G ταξινομούνται σύμφωνα με την εκτίμηση R του εύρους αναζήτησης (γραμμή 4) κι ο συντονιστής κόμβος τις εξετάζει σειριακά (γραμμές 7–10).

Η τοπική εγγύηση (M_i, R_i) ενημερώνει την ολική (M, R) στις γραμμές 8, 9. Εάν η παράμετρος M πάρει τιμές μεγαλύτερες του k, τότε το R είναι τουλάχιστον τόσο όσο η πραγματική απόσταση της k-ιοστής κοντινότερης πλειάδας του αποτελέσματος. Εν τοιαύτη περιπτώσει, το ζεύγος (M, R) είναι η πιο ισχυρή ολική εγγύηση που δύναται να εξαχθεί από το G. Οπότε δεν υπάρχει ανάγκη να εξεταστούν οι λοιπές τοπικές εγγυήσεις (γραμμές 10).

Μετά τη διαδικασία αυτή, το ζεύγος (M, R) συνοψίζει την εικόνα όλων των τοπικών αποτελεσμάτων που εξήχθησαν. Ο συντονιστής κόμβος προωθεί το ερώτημα στο σύνδεσμό του για βάθος D αν και μόνον αν απαιτούνται περισσότερα στοιχεία για το αποτέλεσμα ή ο σύνδεσμος αυτός αντιπροσωπεύει περιοχή στην οποία εμπεριέχονται πλειάδες εντός ακτίνας R από το κέντρο του ερωτήματος (γραμμές 11, 12). Το ερώτημα αυτό δεν ξεφεύγει από το συμμετρικό υποδέντρο του συντονιστή για βάθος D. Τότε ο συντονιστής κόμβος λαμβάνει τοπικές εγγυήσεις από όλους τους κόμβους που χρειάστηκε να προσπελάσει στο γύρο αυτό και περιμένει έως ότου να ληφθούν τα μηνύματα από τους κόμβους που απέχουν ως D βήματα (γραμμές 13–16). Οι τοπικές εγγυήσεις που ελήφθησαν εισάγονται στο G κι άρα χρησιμοποιούνται για τον υπολογισμό της ολικής εγγύησης στον επόμενο γύρο.

Στο Σχήμα 4.4 παρουσιάζουμε το ρόλο του συντονιστή κόμβου για τη μέθοδο iterative processing για ένα 2-NN ερώτημα. Αρχικά, ο συντονιστής κόμβος u ανακτά την τοπικά αποθηκευμένη πλειάδα ενώ με R_u συμβολίζουμε την απόστασή της από το κέντρο $\vec{c}$. Οπότε, στον πρώτο γύρο έχουμε ότι $D = u.depth = 3$ ενώ η τοπική εγγύηση γίνεται $(1, R_u)$ κι αναπαριστούμε την εκτίμηση του εύρους αναζήτησης με τον δίσκο στο Σχήμα 4.4(α΄). Αφού $1 < k$, ο συντονιστής αποστέλει ένα αίτημα στο σύνδεσμό του y για βάθος 3. Στη συνέχεια ο κόμβος y εκτελεί τον Αλγόριθμο 5 κι ανακτά την τοπικά αποθηκευμένη πλειάδα που βρίσκεται σε απόσταση R_y στο Σχήμα 4.3(β΄). Επίσης αποστέλει στον συντονιστή κόμβο τη σχετική τοπική εγγύηση $(1, R_y)$ προκειμένου να καθορίσει τα επόμενα βήματα της αναζήτησης.



(α΄) γύρος 1 (βάθος 3) (β΄) γύρος 2 (βάθος 2) (γ΄) γύρος 3 (βάθος 1)

Σχήμα 4.4: Επεξεργασία ερωτημάτων 2-NN με τη μέθοδο iterative processing για τον συντονιστή κόμβο u.

Στην αρχή του δεύτερου γύρου με $D = 2$, ο συντονιστής κόμβος διαθέτει δύο τοπικές εγγυήσεις: τη δική του $(1, R_u)$, κι εκείνη του y $(1, R_y)$. Μετά την εκτέλεση των γραμμών 5–10, κατασκευάζει την ολική εγγύηση $(2, R_y)$ που δείχνουμε με έναν δίσκο στο Σχήμα 4.4(β΄). Ο συντονιστής τότε αποστέλει ένα αίτημα στο σύνδεσμο w για βάθος 2 επειδή το αντίστοιχο συμμετρικό υποδέντρο επικαλύπτει το δίσκο. Ο κόμβος w ανακτά δύο πλειάδες κι αποστέλει την τοπική εγγύηση $(2, R_w)$, με R_w να υποδηλώνει την απόσταση από το $\vec{c}$ της μακρύτερη εγγραφής εκ των δύο στο Σχήμα 4.3(γ΄).

Στον τρίτο γύρο με $D = 1$, ο συντονιστής υπολογίζει την ολική εγγύηση $(2, R_w)$ όπως φαίνεται στο Σχήμα 4.4(γ΄), παίρνοντας υπόψην μόνο την τοπική εγγύηση του w αφού $R_w < R_u < R_y$. Ο χώρος αναζήτησης που σχηματίζεται δεν επικαλύπτεται από την περιοχή που αντιστοιχεί στο συμμετρικό υποδέντρο για βάθος 1, κι άρα δεν χρειάζεται να προωθηθεί το ερώτημα περαιτέρω.

Εν κατακλείδι, ο εκδότης του ερωτήματος θα έχει παραλάβει τις τέσσερις πλειάδες που σημειώνουμε με σκούρους κύκλους στο Σχήμα 4.4 που συγκριτικά είναι ακριβέστερο αποτέλεσμα από αυτό της προηγούμενης μεθόδου αφού εμπεριέχει λιγότερα false positives.

Τα δύο επόμενα λήμματα υπολογίζουν την πολυπλοκότητα της μεθόδου κι αποδεικνύουν την ορθότητά της.

Λήμμα 9 *Ο αναμενόμενος αριθμός από επαναπροωθήσεις για την iterative processing μέθοδο είναι* $O(\log^2 n)$.

Απόδειξη 9 *Αρχικά, ο εκδότης του ερωτήματος εντοπίζει σε $O(\log n)$ βήματα τον κόμβο που θα παίξει το ρόλο του συντονιστή της επεξεργασίας εκτελώντας τον Αλγόριθμο 6. Ο μέγιστος αριθμός από γύρους που απαιτείται είναι ίσος με το βάθος του συντονιστή κόμβου στο κατανεμημένο k-d δέντρο. Όμως σε κάθε γύρο η επεξεργασία περιορίζεται στο υποδέντρο του k-d δέντρου που αντιστοιχεί στην περιοχή που βρίσκεται πιο κοντά στο κέντρο του ερωτήματος. Κάθε ένας κόμβος που προσπελαύνεται εκτελεί τον Αλγόριθμο 5. Όπως και στην απόδειξη του Λήμματος 7, ο κάθε γύρος ολοκληρώνεται σε αριθμό βημάτων ίσο ή μικρότερο από το βάθος του αντίστοιχου υποδέντρου. Από το Λήμμα 3 έχουμε ότι το αναμενόμενο βάθος ολόκληρου του δέντρου είναι $O(\log n)$ κι άρα ο συνολικός αριθμός επαναπροωθήσεων είναι $O(\log^2 n)$.* ■

Λήμμα 10 *Η μέθοδος iterative processing ανακτά τους k κοντινότερους γείτονες από το κέντρο $\vec{c}$ του ερωτήματος.*

Απόδειξη 10 *Η απόδειξη είναι παρόμοια με αυτή του Λήμματος 8 με δύο όμως διαφορές:*

Πρώτον, χρειάζεται να δείξουμε ότι οι ολικές εγγυήσεις που κατασκευάζει ο συντονιστής κόμβος είναι έγκυρες. Εφόσον έχουμε από το Λήμμα 8 ότι όλες οι ληφθείσες εγγυήσεις είναι έγκυρες, τότε οι γραμμές 5–10 υπολογίζουν την ολική εγγύηση από το σύνολο των τοπικών.

Δεύτερον, με u συμβολίζουμε τον κόμβο που εμπεριέχει τον κοντινότερο γείτονα, αλλά υποθέτοντας ότι ο κόμβος u δεν προσπελαύνεται, αυτό μπορεί να συμβεί σε δύο περιπτώσεις: είτε το υποδέντρο του συντονιστή κόμβου που εμπεριέχει τον κόμβο u δεν εξετάζεται, είτε η συνθήκη της γραμμής 11 του Αλγορίθμου 6 δεν ισχύει για το υπο εξέταση υποδέντρο. Το πρώτο ενδεχόμενο δεν μπορεί να συμβεί καθώς ο συντονιστής κόμβος εξετάζει όλα τα υποδέντρα διαδοχικά από κάτω προς τα πάνω. Το δεύτερο δεν μπορεί να συμβεί για έγκυρη εγγύηση κι άρα έχουμε αντίφαση. ■

4.4 Πειραματική Αποτίμηση

Προκειμένου να αξιολογήσουμε τις μεθόδους μας και να επιβεβαιώσουμε τα θεωρητικά αποτελέσματα από τις προηγούμενες ενότητες, προσομοιάζουμε ένα δυναμικό περιβάλλον κι αποτιμούμε την απόδοση της επεξεργασίας κάθε τύπου ερωτημάτων.

4.4.1 Μεθοδολογία

Συγκρίνουμε το MIDAS με τρεις γνωστές τεχνικές από τη βιβλιογραφία. Το MAAN [36] ανήκει στην κατηγορία των μονοδιάστατων δομών που επεκτάθηκαν σε πολυδιάστατες. Από την άλλη, το CAN [99] είναι από τη φύση του πολυδιάστατο. Τέλος, το VBI-tree [75] προσαρμόζει δομές που έχουν προταθεί για αποθήκευση πολυδιάστατων δεδομένων στο σκληρό δίσκο.

4.4.2 Τοπολογία Δικτύου

Προσομοιώνουμε ένα δυναμικό δίκτυο το οποίο υφίσταται αιθαίρετες εισαγωγές κι αποχωρήσεις ομότιμων κόμβων σε δύο διακριτές φάσεις. Στην πρώτη φάση, νέοι κόμβοι εισέρχονται στο δίκτυο συνεχόμενα έως ότου αυτό να απαρτίζεται από 100000 κόμβους. Στη δεύτερη φάση κόμβοι αποχωρούν από το δίκτυο συνεχόμενα έως ότου το δίκτυο φτάσει και πάλι τους 1000 κόμβους. Όταν μεταβάλλουμε το μέγεθος του δικτύου, δείχνουμε στα γραφήματα τα αποτελέσματα για την πρώτη φάση ενώ τα αποτελέσματα της δεύτερης που είναι ευθέως ανάλογα και παραλείπονται.

4.4.3 Δεδομένα κι Ερωτήματα

Χρησιμοποιούμε πραγματικά και συνθετικά δεδομένα. Τα μεν προέρχονται από το R-tree Portal[1] και σημειώνεται ως *NE*. Αποτελείται από χωρικά σημεία που αντιπροσωπεύουν

[1] http://www.rtreeportal.org

125000 ταχυδρομικούς κωδικούς σε τρεις μητροπολιτικές περιοχές, την Νέα Υόρκη, τη Φιλαδέλφεια και τη Βοστόνη. Τα συνθετικά δεδομένα, που σημειώνουμε ως *Synthetic*, είναι περίπου ίσου μεγέθους και κάθε μία διάσταση κατασκευάζεται ανεξάρτητα από τις υπόλοιπες με συνάρτηση κατανομής πιθανότητας (probability density function) που δίνεται από τον τύπο $f_{X=x} = 2x$ για $0 \leq x \leq 1$ και $f_{X=x} = 0$ διαφορετικά.

Για τα ερωτήματα σημείων επιλέγουμε ομοιόμορφα κι ανεξάρτητα ένα τυχαίο σημείο του συνολικού χώρου. Για τα ερωτήματα εύρους ομοίως επιλέγουμε ένα τυχαίο σημείο ως την αρχή της ορθογώνιας περιοχής καθώς το μέγεθος καθορίζεται με τέτοιο τρόπο ώστε η αναζήτηση να επιστρέψει συγκεκριμένο αριθμό πλειάδων, μέγεθος που ορίζουμε ως επιλεκτικότητα (selectivity). Αναλόγως επιλέγουμε και τα κέντρα των ερωτημάτων κοντινότερων γειτόνων.

4.4.4 Παράμετροι

Η πειραματική μας αποτίμηση εξετάζει τρεις παραμέτρους. Το μέγεθος του δικτύου μεταβάλλεται από τους 1000 ομότιμους κόμβους ως τους 100000. Η διαστασιμότητα των συνθετικών δεδομένων μεταβάλλεται από τις 2 ως τις 13 διαστάσεις. Τέλος, η επιλεκτικότητα (selectivity) των ερωτημάτων εύρους και κοντινότερων γειτόνων μεταβάλλεται από τις 25 ως και τις 150 πλειάδες. Τα πεδία των παραμέτρων μαζί με τις προκαθορισμένες τους τιμές συνοψίζονται στον Πίνακα 4.1. Όταν μεταβάλλουμε μία παράμετρο όλες οι υπόλοιπες τίθονται στις προκαθορισμένες τους τιμές. Τέλος τα γραφήματα αποτελούν τις μέσες τιμές του κάθε μεγέθους για 50000 ερωτήματα που εκδόθηκαν σε 15 διαφορετικές τοπολογίες.

Παράμετρος	**Προκαθορισμένη τιμή**	**Εύρος**
Μέγεθος δικτύου	10K	1K, 4K, 7K, 10K, 40K, 70K, 100K
Διαστασιμότητα δεδομένων	2	2, 3, 5, 7, 11, 13
Επιλεκτικότητα δρωτήματος	50	25, 50, 75, 100, 125, 150

Πίνακας 4.1: Παράμετροι πειραματικής αποτίμησης.

4.4.5 Μετρικές

Για όλες τους τύπους ευρετηρίων μετράμε μεγέθη που σχετίζονται με τη λειτουργία του δικτύου καθώς και την απόδοση των μεθόδων επεξεργασίας διαφόρων τύπων ερωτημάτων. Ειδικότερα, δείχνουμε την *κατάσταση* που ένας κόμβος χρειάζεται να διατηρεί, δηλαδή το μέγεθος επικαιροποιημένης πληροφορίας, π.χ. σύνδεσμοι, ζώνες, κ.τ.λ. Η συγκεκριμένη παράμετρος είναι σημαντικό μέτρο κλιμακωσιμότητας (scalability) καθώς ο αριθμός των κόμβων αυξάνεται.

Επιπλέον, μελετάμε την κατανομή των δεδομένων στους κόμβους. Ανάλογα με την κατανομή αυτή υπάρχει το ενδεχόμενο ύπαρξης δυσανάλογου φορτίου για συγκεκριμένους κόμβους στο δίκτυο. Χρησιμοποιούμε ένα μέγεθος φορτίου δεδομένων που αναλογεί στο ποσοστό των δεδομένων που είναι αποθηκευμένο στους $Q\%$ πιο επιφορτισμένους κόμβους. Συνεπώς, η ιδανική του τιμή είναι $Q/100$ κι αντιστοιχεί στην περίπτωση όπου τα δεδομένα είναι κατανεμημένα επί ίσοις όροις σε όλους τους κόμβους. Ακόμα, κάνουμε χρήση του δείκτη ισοροπίας του Jain [76] ως ένα επιπλέον μέτρο εξακρίβωσης της κατανομής φορτίου στους κόμβους. Ο δείκτης αυτός είναι κανονικοποιημένος στο διάστημα $[0,1]$ κι η βέλτιστη τιμή του είναι η μονάδα.

Σχετικά με την απόδοση των μεθόδων επεξεργασίας των διαφόρων τύπων ερωτημάτων υιοθετούμε τρεις μετρικές. Ο μέγιστος αριθμός επαναπροωθήσεων (*latency*) μετράει τον αριθμός από βήματα κι επαναπροωθήσεις που χρειάζονται στα πλαίσια της επεξεργασίας ενός ερωτήματος όπου οι χαμηλές τιμές υποδεικνύουν γρήγορη απόκριση. Η κατανεμημένη επεξεργασία ερωτημάτων επιβάλλει τον διαμοιρασμό του υπολογισμού του ερωτήματος σε πολλαπλούς απομακρυσμένους κόμβους συμπεριλαμβανομένου ενός αριθμού από κόμβους που δεν συμμετέχουν άμεσα στην απάντηση του ερωτήματος αλλά μεσολαβούν προκειμένου τα ερωτήματα να φτάσουν στους σχετικούς με το ερώτημα κόμβους. Δύο μετρικές χρησιμοποιούνται για την ποσοτικοποίηση του μεγέθους αυτού. Η ακρίβεια (*precision*) ορίζεται ως

η αναλογία του αριθμού των σχετικών κόμβων που προσπελαύνονται κατά τη διάρκεια της επεξεργασίας του ερωτήματος προς το συνολικό αριθμό κόμβων που προσπελαύνονται, με την βέλτιστη τιμή του μεγέθους να ισούται με τη μονάδα. Η συμφόρηση (*congestion*) ορίζεται ως ο μέσος αριθμός από ερωτήματα που λαμβάνει ο κάθε κόμβος όταν n τυχαία ερωτήματα εκδίδονται, όπου το n αντιστοιχεί στο μέγεθος του δικτύου κι οι χαμηλές τιμές υποδεικνύουν χαμηλό φορτίο επεξεργασίας.

4.4.6 Αποτελέσματα

Στην ενότητα αυτή περιγράφουμε και συζητάμε εκτενώς τα αποτελέσματα της πειραματικής αποτίμησης των μεθόδων μας.

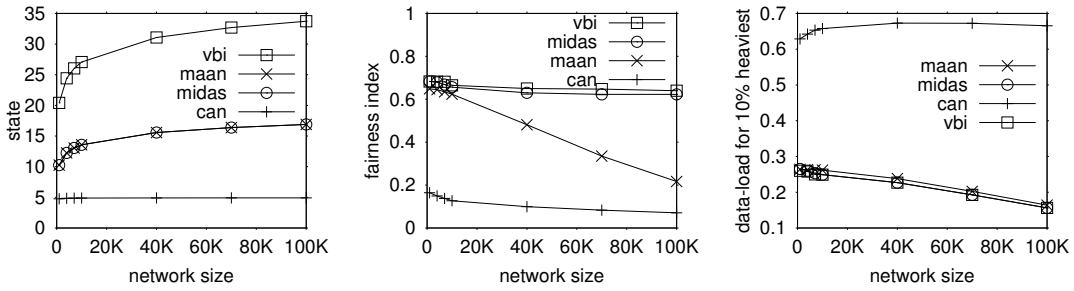
Χαρακτηριστικά Τοπολογίας Δικτύου

Τα Σχήματα 4.5(α΄) και 4.6(α΄) περιγράφουν το μέγεθος της κατάστασης/πληροφορίας που διατηρούν οι κόμβοι επικαιροποιημένη καθώς μεταβάλλεται το μέγεθος του δικτύου, για τα πειράματα σε χωρικά και συνθετικά δεδομένα αντίστοιχα. Το μέγεθος αυτό είναι ευθέως αναλογο της δικτυακής κίνησης που προκαλείται από διαδικασίες συντήρησης του δικτύου, όπως ο εντοπισμός απρόσμενων απωλειών κόμβων, η διατήρηση επικαιροποιημένων πινάκων δρομολόγησης, το κόστος σε μηνύματα των εισαγωγών και αποχωρήσεων κόμβων, κ.α. Το μέγεθος αυτό αυξάνεται με λογαριθμικό ρυθμό για τα MIDAS, MAAN και VBI-tree, όπως φαίνεται στο Σχήμα 4.5(α΄). Αν και στο MIDAS και το MAAN οι κόμβοι διατηρούν παρόμοια κατάσταση, στο VBI-tree διατηρούν περίπου τη διπλάσια κατάσταση συγκριτικά. Ο λόγος έγκειται στο γεγονός ότι στο VBI-tree οι κόμβοι διατηρούν πληροφορία όχι μόνο για τους κόμβους που βρίσκονται στο μονοπάτι προς τη ρίζα αλλά και για τους κόμβους που βρίσκονται στο ίδιο ύψος με αυτούς στην κατανεμημένη ιεραρχία που αντικατοπτρίζουν. Ακόμα, σημειώνουμε ότι η κατάσταση που διατηρεί ο κάθε κόμβος παραμένει αμετάβλητη για τα MIDAS, MAAN και VBI-tree ως προς τον αριθμό των διαστάσεων του χώρου που δεικτοδοτεί το ευρετήριο, όπως δείχνει το Σχήμα 4.6(α΄). Εν αντιθέσει με το CAN του οποίου αν και επηρεάζεται ελάχιστα από το μέγεθος του δικτύου, αυξάνεται γραμμικά σε σχέση με τη διαστασιμότητα του χώρου. Έτσι έχουμε λίγους γείτονες για κάθε κόμβο στο CAN στο Σχήμα 4.5(α΄) που αφορά τα χωρικά δεδομένα. Καθώς όμως αυξάνεται ο αριθμός των διαστάσεων, το ίδιο συμβαίνει και για τους πίνακες δρομολόγησης των κόμβων στο CAN. Για παράδειγμα, έχουμε ότι για περισσότερες των 8 διαστάσεων οι κόμβοι στο CAN διατηρούν περισσότερους συνδέσμους προς άλλους κόμβος από ότι στο MIDAS και στο MAAN.

Τα Σχήματα 4.5(β΄) και 4.5(γ΄) δείχνουν μετρικές ισοκατανομής φορτίου, τον δείκτη του Jain καθώς και το ποσοστό του φορτίου που αντιστοιχεί στο 10% των πιο επιβαρημένων κόμβων του δικτύου καθώς μεταβάλλεται το μέγεθος του δικτύου που δεικτοδοτεί τα χωρικά δεδομένα. Ομοίως, τα Σχήματα 4.6(β΄) και 4.6(γ΄) παρουσιάζουν τις διακυμάνσεις των μεγεθών αυτών καθώς μεταβάλλεται ο αριθμός διαστάσεων των δεδομένων. Από τις γραφικές αυτές συμπεραίνουμε ότι το MIDAS και το VBI-tree είναι εξίσου δίκαια κι ανθεκτικά στις μεταβολές στη τοπολογία του δικτύου και τον αριθμό των διαστάσεων του ευρετηρίου. Από την άλλη, το CAN μειονεκτεί αισθητά. Για παράδειγμα, στο Σχήμα 4.6(γ΄) βλέπουμε ότι μόλις το 10% των πιο επιβαρημένων κόμβων είναι υπεύθυνο για περισσότερα από τα $\frac{3}{4}$ των δεδομένων για τις 13 διαστάσεις! Ο λόγος είναι απλός κι οφείλεται στο πρωτόκολλο εισαγωγής νέων κόμβων σύμφωνα με το οποίο κάθε φορά μοιράζεται στη μέση ο όγκος της ζώνης ενός κόμβου για να φιλοξενηθεί ένας ακόμα, χωρίς να λαμβάνεται υπόψην η κατανομή των δεδομένων. Τέλος, το MAAN καταφέρνει να είναι δίκαιο για μικρές μόνον τοπολογίες επειδή η αντιστοίχιση των πολλών διαστάσεων σε μόλις μία χωρις κάποιον επιτηδευμένο τρόπο, π.χ. space-filling curves, principal components analysis, οδηγεί σε ανισοκατανομές όπως προκύπτει κι από το Σχήμα 4.5(β΄).

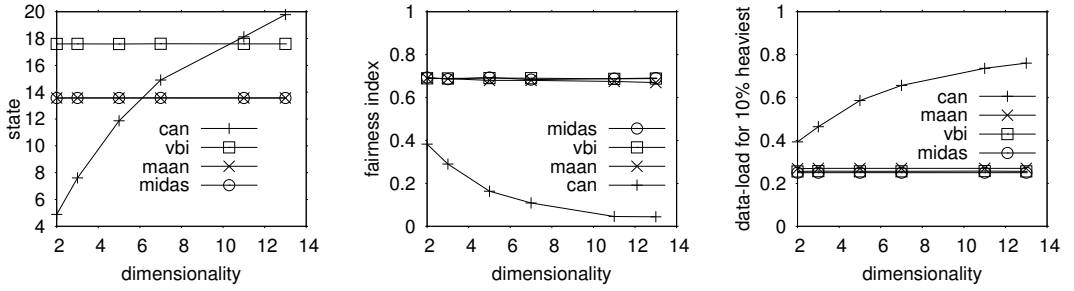
Ερωτήματα Σημείων

Το Σχήμα 4.7 παρουσιάζει την απόδοση των ερωτημάτων σημείων για τα χωρικά δεδομένα καθώς μεταβάλλεται το μέγεθος του δικτύου ενώ το Σχήμα 4.8 για τα συνθετικά δεδομένα



(α΄) μέγεθος πινάκων δρομολόγησης (β΄) δείκτης ισοκατανομής φορτίου (γ΄) ποσοστό φορτίου για το 10% των πιο επιφορτισμένων κόμβων

Σχήμα 4.5: Χαρακτηριστικά δομής δικτύου για χωρικά δεδομένα.

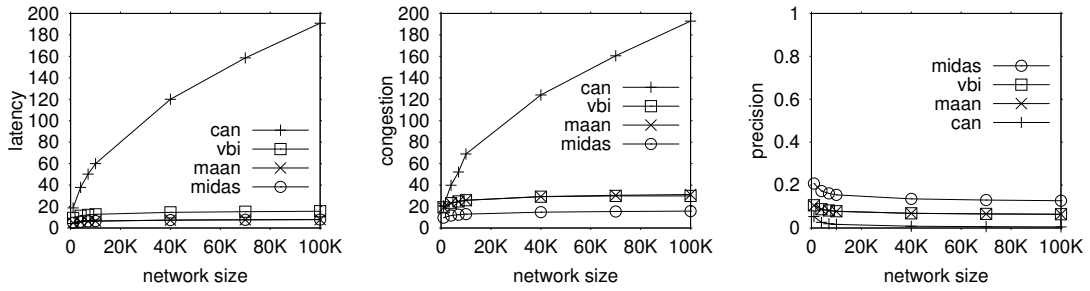


(α΄) μέγεθος πινάκων δρομολόγησης (β΄) δείκτης ισοκατανομής φορτίου (γ΄) ποσοστό φορτίου για το 10% των πιο επιφορτισμένων κόμβων

Σχήμα 4.6: Χαρακτηριστικά δομής δικτύου για συνθετικά δεδομένα.

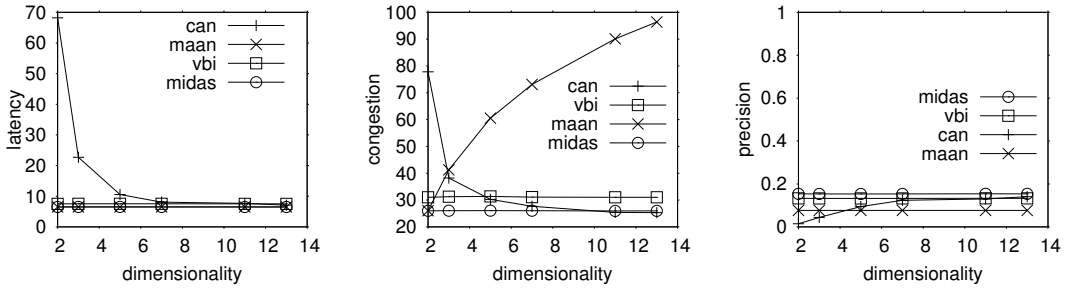
καθώς μεταβάλεται ο αριθμός των διαστάσεων του μετρικού χώρου που δεικτοδοτείται. Ο αριθμός των επαναπροωθήσεων στο MIDAS, MAAN και στο VBI-tree κλιμακώνεται λογαριθμικά καθώς αυξάνεται ο αριθμός των κόμβων στο Σχήμα 4.7(α΄) και παραμένει σχεδόν ανέπαφο από την αύξηση του αριθμού των διαστάσεων στο Σχήμα 4.8(α΄). Σε όλες τις περιπτώσεις τα ερωτήματα αυτά απαιτούν μικρό αριθμό από βήματα για MIDAS και MAAN και υπερτερούν του VBI-tree. Όσον αφορά το CAN, ο αναμενόμενος αριθμός από επαναπροωθήσεις ισούται με $O(\sqrt[d]{n})$. Δηλαδή αν κι αυξάνεται με το μέγεθος του δικτύου, μειώνεται με τον αριθμό των διαστάσεων, καθώς υποδεικνύουν αμφότερα γραφήματα. Παρ' όλα αυτά η κατάσταση που πρέπει να διατηρείται επικαιροποιημένη σε κάθε κόμβο του CAN αυξάνεται γραμμικά με τον αριθμό των διαστάσεων (Σχήμα 4.6(α΄)).

Όσον αφορά όμως το μέτρο της συμφόρησης (congestion) που παρουσιάζουμε στα Σχήματα 4.7(β΄) και 4.8(β΄), το MIDAS υπερτερεί για μέχρι 10 διαστάσεις. Για περισσότερες διαστάσεις το CAN συμπεριφέρεται καλύτερα αν και η συμφόρηση δεν κλιμακώνεται καλά καθώς αυξάνεται ο αριθμός των κόμβων στο δίκτυο. Η συμφόρηση των κόμβων του VBI-tree είναι εξίσου καλή με το MIDAS. Παρόμοια αποτελέσματα παίρνουμε για το μέγεθος της ακρίβειας (precision) όπως βλέπουμε στα Σχήματα 4.7(γ΄) και 4.8(γ΄). Το MAAN παρατηρούμε ότι έχει χαμηλή συμφόρηση για τα χωρικά δεδομένα μία συμπεριφορά που δεν επιτυγχάνει να συντηρήσει για πολλές διαστάσεις. Παρόμοια συμπεριφορά παρατηρούμε και για την ακρίβεια για τα χωρικά δεδομένα κι οφείλεται κυρίως στην αντιστοίχιση των πολυδιάστατων δεδομένων στη μία διάσταση, εξαναγκάζοντας έτσι την προσπέλαση πολλών περισσότερων κόμβων αφού εκ κατασκευής στο MAAN δεν υπάρχει η δυνατότητα για ιδιαίτερα αποτελεσματική δρομολόγηση για μεγάλο αριθμό διαστάσεων.



(α΄) αριθμός επαναπροωθήσεων (β΄) συμφόρηση δικτύου (γ΄) ακρίβεια

Σχήμα 4.7: Απόδοση ερωτημάτων σημείων για χωρικά δεδομένα ως προς το μέγεθος του δικτύου.



(α΄) αριθμός επαναπροωθήσεων (β΄) συμφόρηση δικτύου (γ΄) ακρίβεια

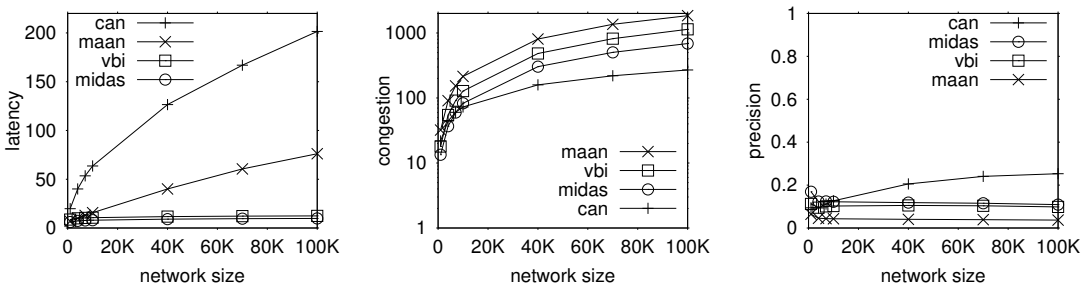
Σχήμα 4.8: Απόδοση ερωτημάτων σημείων για συνθετικά δεδομένα ως προς τον αριθμό των διαστάσεων.

Ερωτήματα Εύρους

Όπως και για τον προηγούμενο τύπο ερωτημάτων έχουμε ότι το latency κλιμακώνεται με λογαριθμικό ρυθμό σε σχέση με το μέγεθος του δικτύου (Σχήμα 4.9(α΄)), και παραμένει σχεδόν ανεπηρέαστο από τον αριθμό των διαστάσεων (Σχήμα 4.10(α΄)). Το MIDAS ξεκάθαρα υπερτερεί του ανταγωνισμού τόσο για χωρικά, όσο και για πολυδιάστατα δεδομένα.

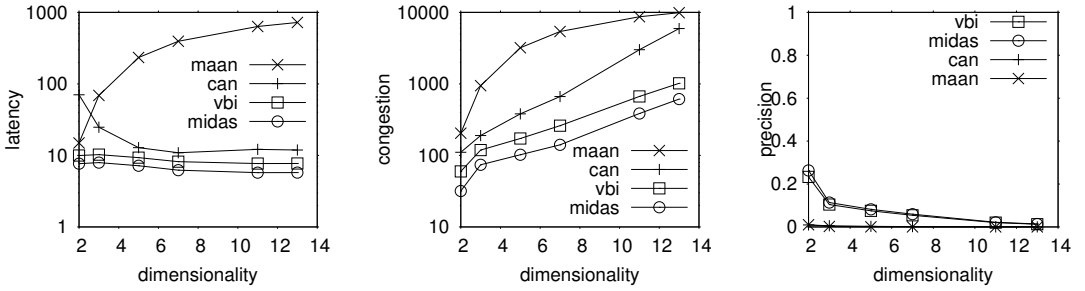
Επιπροσθέτως, η επιλεκτικότητα (selectivity) έχει πολύ μικρό αντίκτυπο στο latency καθώς δείχνουμε στο Σχήμα 4.11(α΄). Εν αντιθέσει, ο αριθμος των επαναπροωθήσεων στο MAAN αυξάνεται με σχεδόν γραμμικό ρυθμό ως προς το μέγεθος του δικτύου. Ειδικότερα, το latency για το MAAN καθορίζεται σε πολύ μεγάλο βαθμό από τον αριθμό των σχετικών προς το ερώτημα κόμβων εξαιτίας της σειριακής επαναπροώθησης του ερωτήματος από κόμβο προς κόμβο. Η κακή απόδοση του MAAN μπορεί επίσης να καταλογιστεί στο γεγονός ότι στην πραγματικότητα η διάταξη των πολυδιάστατων δεδομένων γίνεται επί τη βάσει μίας και μοναδικής διάστασης. Έτσι όλοι οι κόμβοι των οποίων οι ζώνες επικαλύπτουν το ερώτημα ως προς τη μία διάσταση προσπελαύνονται αν και μπορεί αυτό να μην ισχύει ως προς τις υπόλοιπες διαστάσεις. Επιπλέον αυτοί οι κόμβοι προσπελαύνονται σειριακά, γεγονός που προκαλεί έναν αρκετά μεγάλο αριθμό επαναπροωθήσεων για τα ερωτήματα εύρους στον πολυδιάστατο χώρο. Κατά συνέπεια υπό συνθήκες το MAAN γίνεται εξαιρετικά ευάλωτο ως προς τη συμφόρηση και την ακρίβεια καθώς αυξάνεται το μέγεθος του δικτύου (Σχήματα 4.9(β΄) και 4.9(γ΄)), τον αριθμό των διαστάσεων (Σχήματα 4.10(β΄) και 4.10(γ΄)), αλλά και την επιλεκτικότητα των ερωτημάτων (Σχήματα 4.11(β΄) και 4.11(γ΄)).

Τα Σχήματα 4.9(β΄), 4.11(β΄) και 4.10(β΄) επιδεικνύουν τη χαμηλή συμφόρηση στους κόμβους του MIDAS σε σχέση με τις υπόλοιπες μεθόδους. Η συμπεριφορά του MIDAS βασίζεται στο γεγονός ότι τα μηνύματα προωθούνται παράλληλα κάθε φορά προς όλους τους κόμβους των οποίων οι περιοχές που αντιπροσωπεύουν επικαλύπτουν την περιοχή του ερωτήματος. Συνεπώς δημιουργούνται πολλαπλές παράλληλες διαδρομές στα πλαίσια της επεξεργασίας ενός ερωτήματος. Προκειμένου να αποφευχθεί στο VBI-tree η υπέρμετρη επιβάρυνση των



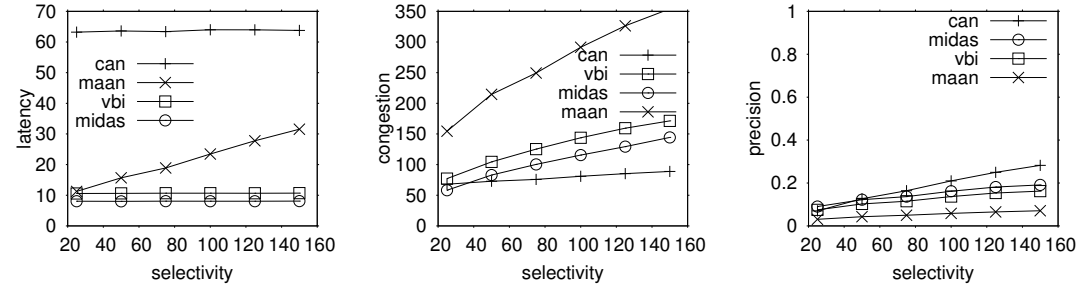
(α΄) αριθμός επαναπροωθήσεων (β΄) συμφόρηση δικτύου (γ΄) ακρίβεια

Σχήμα 4.9: Απόδοση ερωτημάτων εύρους για χωρικά δεδομένα ως προς το μέγεθος του δικτύου.



(α΄) αριθμός επαναπροωθήσεων (β΄) συμφόρηση δικτύου (γ΄) ακρίβεια

Σχήμα 4.10: Απόδοση ερωτημάτων εύρους για συνθετικά δεδομένα ως προς τον αριθμό των διαστάσεων.



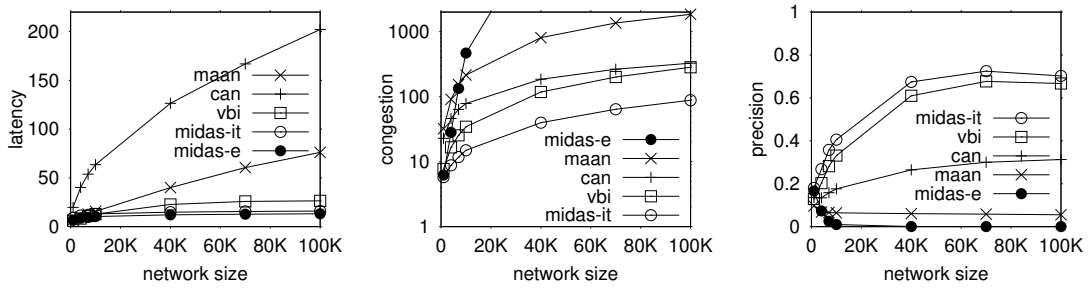
(α΄) αριθμός επαναπροωθήσεων (β΄) συμφόρηση δικτύου (γ΄) ακρίβεια

Σχήμα 4.11: Απόδοση ερωτημάτων εύρους για χωρικά δεδομένα ως προς την επιλεκτικότητα του ερωτήματος.

κόμβων που βρίσκονται πολύ κοντά στη ρίζα του δέντρου χρησιμοποιούνται παράλληλα οριζόντιες διαδρομές μαζί με τις κάθετες. Ακόμα, κάθε ομότιμος κόμβος είναι υπεύθυνος και για έναν ακόμα κόμβο (εσωτερικό κόμβο της κατανεμημένης ιεραρχίας) που αν και στοχεύει στην δρομολόγηση των ερωτημάτων έχει ως αποτέλεσμα το διπλάσιο αριθμό κόμβων στο σύνολο από ότι έχουμε στο MIDAS. Εν κατακλείδι, καταλήγουμε να έχει μεγαλύτερο κόστος μηνυμάτων για την επεξεργασία ερωτημάτων εύρους. Στα Σχήματα 4.9(γ΄) και 4.10(γ΄), η ακρίβεια (precision) των VBI-tree και MAAN υστερεί ως προς το MIDAS καθώς προσπελαύνονται περισσότεροι κόμβοι προκειμένου τα ερωτήματα να φθάσουν στους σχετικούς κόμβους των οποίων οι ζώνες επικαλύπτουν το ερώτημα. Στο Σχήμα 4.11(γ΄) η ακρίβεια βελτιώνεται καθώς αυξάνεται η επιλεκτικότητα (selectivity) των ερωτημάτων κι άρα αυξάνεται ο αριθμός των σχετικών κόμβων με ταχύτερο ρυθμό σε σχέση με το συνολικό αριθμό κόμβων που προσπελαύνονται.

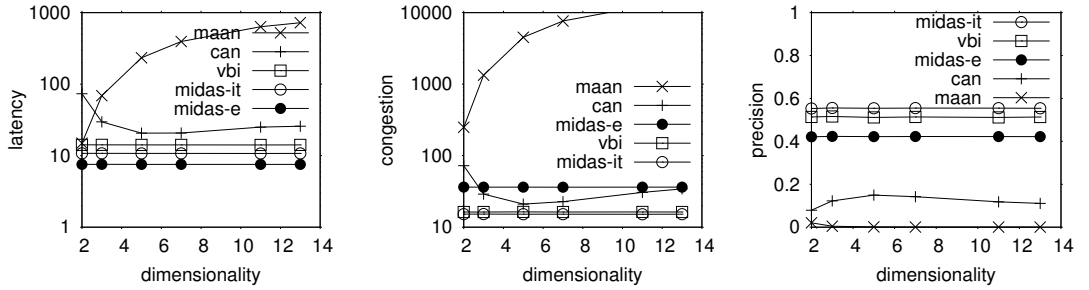
Ερωτήματα Κοντινότερων Γειτόνων

Για το υπόλοιπο της ενότητας αυτής θα εξετάσουμε την απόδοση των μεθόδων για την ανεύρεση των k κοντινότερων γειτόνων. Όπως αναλύσαμε και σε προηγούμενη ενότητα, ο μέγιστος αριθμός επαναπροωθήσεων φράσσεται από $O(\log^2 n)$ για τον iterative processing αλγόριθμο ενώ για τον eager processing αλγόριθμο από $O(\log n)$ βήματα, όπως προβλέπουν και τα Λήμματα 7 και 9. Όμως στο Σχήμα 4.12(α΄) αμφότερες μέθοδοι επιδεικνύουν λογαριθμική συμπεριφορά ως προς το μέγεθος του δικτύου. Το ίδιο ισχύει και για το VBI-tree ενώ για το MAAN το latency αυξάνεται γραμμικά με το μέγεθος του δικτύου. Για τον εντοπισμό των κοντινότερων γειτόνων στο CAN χρησιμοποιούμε την τεχνική από το [56]. Η προσέγγιση αυτή επωφελείται από την τοπικότητα στη δόμηση της τοπολογίας του CAN αφού κόμβοι οι οποίοι είναι υπεύθυνοι για κοντινά σημεία στο επίπεδο απέχουν μικρό αριθμό από βήματα



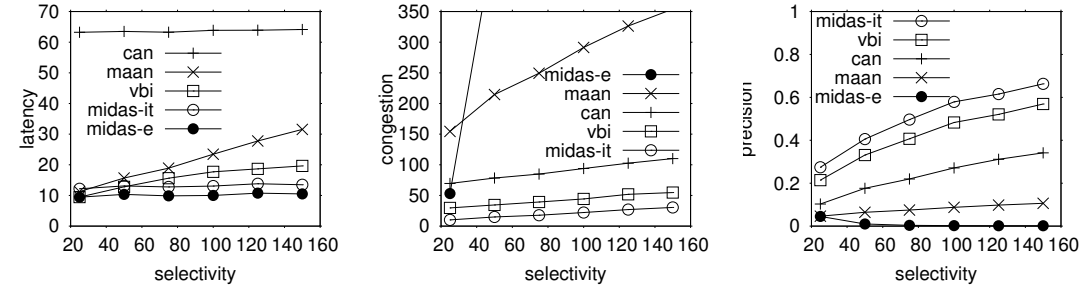
(α΄) αριθμός επαναπροωθήσεων (β΄) συμφόρηση δικτύου (γ΄) ακρίβεια

Σχήμα 4.12: Απόδοση ερωτημάτων κοντινότερων γειτόνων για χωρικά δεδομένα ως προς το μέγεθος του δικτύου.



(α΄) αριθμός επαναπροωθήσεων (β΄) συμφόρηση δικτύου (γ΄) ακρίβεια

Σχήμα 4.13: Απόδοση ερωτημάτων κοντινότερων γειτόνων για συνθετικά δεδομένα ως προς τον αριθμό των διαστάσεων.



(α΄) αριθμός επαναπροωθήσεων (β΄) συμφόρηση δικτύου (γ΄) ακρίβεια

Σχήμα 4.14: Απόδοση ερωτημάτων κοντινότερων γειτόνων για χωρικά δεδομένα ως προς την επιλεκτικότητα του ερωτήματος.

αναμεταξύ τους. Αν κι η προσέγγιση που προτείνουν είναι αποτελεσματική, επηρεάζεται όμως από τα χαρακτηριστικά του δικτύου και για αυτό δεν επιτυγχάνει καλή απόδοση για τα χωρικά δεδομένα στο CAN (Σχήμα 4.12(α΄)).

Το Σχήμα 4.12(β΄) παρουσιάζει το πως κλιμακώνεται η συμφόρηση (congestion) ως προς το μέγεθος του δικτύου. Το μέσο φορτίο ανά κόμβο για την iterative processing μέθοδο του MIDAS είναι το χαμηλότερο, ένα θετικό αποτέλεσμα το οποίο συνοδεύεται από τα υψηλά ποσοστά ακρίβειας του Σχήματος 4.12(γ΄). Η eager processing μέθοδος, αν κι είναι πιο γρήγορη, περιλαμβάνει αρκετά υψηλότερο επικοινωνιακό κόστος εξαιτίας του ότι βασίζεται σε τοπικές εκτιμήσεις του εύρους αναζήτησης χωρίς κάποιο συντονιστή κόμβο. Αξιοσημείωτο το γεγονός ότι η ποιότητα της εκτίμησης αυτής χειροτερεύει καθώς μεγαλώνει το δίκτυο, όπως φαίνεται κι από το Σχήμα 4.12(γ΄).

Αμφότεροι μέθοδοι του MIDAS φαίνονται να μην επηρεάζονται από την αύξηση στον αριθμό των διαστάσεων στο Σχήμα 4.13 και είναι ασυμπτωτικά καλύτερες από τις άλλες μεθόδους. Επιπροσθετως, το CAN είναι σε θέση να εκμεταλευτεί τον μεγάλο αριθμό διαστάσεων με το να εγκαθιδρύει αριθμό συνδέσμων ευθέως ανάλογο κι άρα η διάμετρος του δικτύου μειώνεται δραστικά. Κατά συνέπεια, μειώνεται κι ο μέγιστος αριθμός επαναπροωθήσεων αλλα επίσης βελτιώνεται κι η συμφόρηση. Εν αντιθέσει, η απόδοση του MAAN φθίνει δραματικά ως προς όλες τις μετρικές καθώς αυξανεται ο αριθμός των διαστάσεων.

Τέλος, η επιλεκτικότητα του ερωτήματος επηρεάζει ελαφρά την iterative processing μέθοδο του MIDAS, όπως φαίνεται και στο Σχήμα 4.14, όπου επιδεικνύει το χαμηλότερο αριθμό επαναπροωθήσεων, συμφόρηση αλλά και την καλύτερη ακρίβεια. Η eager processing μέθοδος αν κι έχει σταθερό latency, δεν κλιμακώνεται καλά ως προς την συμφόρηση και την ακρίβεια. Το CAN στον αντίποδα, αν και μειονεκτεί, δείχνει να μην επηρεάζεται από την επιλεκτικότητα του ερωτήματος. Ανάλογα, τα VBI-tree και MAAN επηρεάζονται ελαφρώς. Το Σχήμα 4.14(β΄) παρουσιάζει τη συμφόρηση για όλες τις μεθόδους ως προς την επιλεκτικότητα των ερωτημάτων, με τα CAN, VBI-tree και MIDAS iterative processing να είναι τα πιο σταθερά. Το Σχήμα 4.14(γ΄) υποδεικνύει ότι η ακρίβεια των μεθόδων βελτιώνεται με την επιλεκτικότητα των ερωτημάτων για CAN, VBI-tree και MIDAS iterative processing, καθώς ο αριθμός των σχετικών κόμβων που προσπελαύνονται αυξάνεται ταχύτερα από τον συνολικό αριθμό. Από την άλλη η ακρίβεια (precision) είναι πολύ χαμηλή για την eager processing μέθοδο.

4.5 Συζήτηση

Το MIDAS διαφέρει σημαντικά από άλλα εδραιωμένα συστήματα κι επιτρέπει την επεξεργασία πολυδιάστατων ερωτημάτων παρέχοντας ταυτόχρονα εγγυήσεις ως προς τον χρόνο απόκρισης. Ειδικότερα, δείξαμε με αναδρομικό τρόπο ότι τα ερωτήματα σημείων κι εύρους ολοκληρώνονται εντός $O(\log n)$ βημάτων, όπου με n συμβολίζουμε το μέγεθος του δικτύου. Συγκριτικά με άλλα συστήματα τα οποία αποσκοπούν στην αποθήκευση πολυδιάστατου περιεχομένου, το CAN είναι σε θέση να απαντάει του ιδίου τύπου ερωτήματα σε $O(\sqrt[d]{n})$ βήματα, που σημαίνει ότι για χαμηλή διασταστημότητα, π.χ. για δεδομένα τα οποία αναλύονται σε έως δέκα χαρακτηριστικά, (ήτοι τη συντριπτική πλειονότητα των εφαρμογών) το MIDAS αναδεικνύεται ως καλύτερο. Βέβαια για μεγαλύτερες διαστάσεις το CAN καταφέρνει να αξιοποιεί τον υπερβολικά μεγάλο αριθμό συνδέσμων ανά κόμβο (γραμμικό ως προς το μέγεθος του δικτύου) όταν για το MIDAS έχουμε μόλις $\log n$ συνδέσμους περίπου ανά κόμβο. Δηλαδή, το MIDAS καταφέρνει να συμβιβάζει τη γρήγορη απόκριση με έναν λογικό αριθμό συνδέσμων για κάθε κόμβο αμβλύνοντας έτσι κόστη που έχουν να κάνουν με τη διατήρηση της συνοχής στο δίκτυο.

Επιπλέον, για τα ερωτήματα κοντινότερων γειτόνων παρέχουμε δύο εναλλακτικές με διαφορετικές προδιαγραφές η κάθε μία όμως. Πιο συγκεκριμένα, η πρώτη έχει χαμηλό χρόνο απόκρισης, όπου τα ερωτήματα απαντώνται και πάλι σε $O(\log n)$ βήματα, ενώ η δεύτερη αν κι απαιτεί έως $O(\log^2 n)$ βήματα απασχολεί εν αντιθέσει πολύ μικρότερο αριθμό κόμβων κι οπότε είναι λιγότερο δαπανηρή από άποψη τόσο επεξεργαστικού, όσο κι επικοινωνιακού κόστους. Η πειραματική αποτίμηση επιδεικνύει ότι το MIDAS έχει καλύτερη απόδοση από τους ανταγωνιστές του καθώς επεξεργάζεται γρήγορα τα ερωτήματα ενώ ταυτόχρονα

επιβαρύνονται ελαφρά οι κόμβοι για κάθε ένα ερώτημα που εκδίδεται εξαιτίας της χαμηλής συμφόρησης στο δίκτυο που προκαλεί η επεξεργασία τους.

Κεφάλαιο 5

Ένα Κατανεμημένο Αποθετήριο για RDF(S) Δεδομένα

Cab driver: Hey mister, $18.75.
Groucho: 1875? That's what I thought.
The 1940 models run much smoother.

Groucho Marx

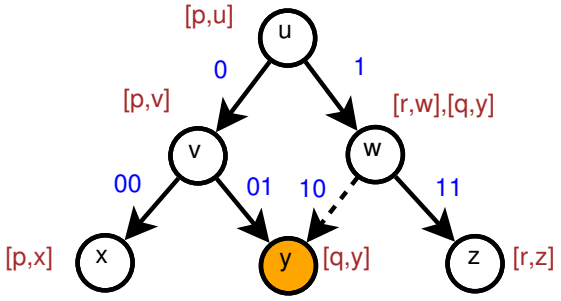
5.1 Μοντέλο Αποθήκευσης

Το MIDAS-RDF αποθηκεύει RDF τριπλέτες χρησιμοποιώντας το κατανεμημένο ευρετήριο MIDAS. Μια RDF τριπλέτα (u, α, v) ανάγεται σε ένα τετραδιάσταστατο κλειδί $\langle u, \alpha, v, \texttt{id}(v)\rangle$, όπου με το $\texttt{id}(v)$ κωδικοποιούμε τις μεταβατικές ιδιότητες της συγκεκριμένης τριπλέτας στο υπογράφημα που αντιστοιχεί στο κατηγόρημα α, δηλαδή το RDF γράφημα που περιλαμβάνει μόνο τις τριπλέτες που έχουν ως κατηγόρημα την ιδιότητα α. Υποθέτουμε επίσης μια λεξικογραφική διάταξη για *υποκείμενα*, *κατηγορήματα* και *αντικείμενα*, καθώς και τη φυσική διάταξη για τα *αναγνωριστικά αντικείμενων των υπογράφων*. Αξίζει να σημειωθεί ότι ένας εικονικός κόμβος (που αντιστοιχεί σε ένα κενό εικονικό υποκείμενο) συνδέεται με το υπογράφημα κάθε κατηγόρηματος με μία ακμή επισημασμένη με την ίδια ιδιότητα. Οπότε υπάρχει έτσι η δυνατότητα τα αναγνωριστικά/ετικέτες που συνοδεύουν τις τριπλέτες να αφορούν είτε ολόκληρο τον γράφο είτε τον υπογράφο για τη συγκεκριμένη ιδιότητα. Για λόγους που θα γίνουν ξεκάθαροι αργότερα, θεωρούμε τον γράφο ως ένα σύνολο από υπογράφους διαφορετικών κατηγορημάτων των οποίων οι μεταβατικές ιδιότητες κωδικοποιούνται στο ξεχωριστό πεδίο που συμβατικά επικαλείται ως ετικέτα (label).

Ως κλειδιά θεωρούνται στο MIDAS-RDF οι πλειάδες της μορφής $\langle u, \alpha, v, \texttt{id}(v)\rangle$. Οπότε το αναγνωριστικό μία τριπλέτας v, $\texttt{id}(v)$ στον ανάλογο υπογράφο, μπορεί να θεωρηθεί ως το μεταθεματικό (postorder) αναγνωριστικό του κόμβου που αντιστοιχεί στο συγκεκριμένο αντικείμενο για τον δεδομένο υπογράφο που περιλαμβάνει όλες τις πλειάδες για το ίδιο κατηγόρημα. Δηλαδή, έχουμε ότι σε κάθε πλειάδα το $\texttt{id}(v)$ αντιστοιχεί στο $\texttt{post}(v)$, που ισούται με τη σειρά με την οποία γίνεται προσπέλαση του συγκεκριμένου κόμβου σύμφωνα με μία μεταθεματική (postorder) διάσχυση του συγκεκριμένου υπογράφου για την ιδιότητα α. Όμως ξέχωρα από τη χρήση της επιπλέον διάστασης για το αναγνωριστικό των RDF τριπλετών, για τη πλήρη υποστήριξη ενός interval-based labeling scheme [17] χρειάζεται να κωδικοποιήσουμε επιπλέον το διάστημα με τα αναγνωριστικά των εννοιών τις οποίες περικλύει (subsumes) ο κάθε κόμβος. Για το λόγο αυτό, κάθε κλειδί του τύπου $\langle u, \alpha, v, \texttt{post(v)}\rangle$ συσχετίζεται στο MIDAS-RDF με ένα σύνθετο διάστημα που αποτελείται από όλα τα θυγατρικά αναγνωριστικά που υπόκεινται στον συγκεκριμένο κόμβο στον υπογράφου του κατηγορήματος α. Ως εκ τούτου, στην περίπτωση αυτή, μια πλειάδα MIDAS-RDF έχει πέντε διαστάσεις, *υποκείμενο*, *κατηγόρημα*, *αντικείμενο*, *αναγνωριστικό τριπλέτας* και *διαστήματα αναγνωριστικών*

θυγατρικών κόμβων. Ωστόσο, μόνο τα πρώτα τέσσερα αποτελούν το κλειδί για το ευρετήριο και χρησιμοποιούνται για την ανάκτηση των ανάλογων διαστημάτων.

Η Εικόνα 5.1 απεικονίζει το παράδειγμα ενός υπογράφου για ένα κατηγόρημα α που περιέχει έξι κόμβους και υποδεικνύονται από τα σύμβολα u έως z. Ακόμα, υποθέτουμε ότι υπάρχουν κι άλλοι κόμβοι κάτω από τους x, y, z τους οποίους δεν δείχνουμε. Για το interval-based scheme, το δέντρο επικάλυψης (spanning tree) εμπεριέχει όλες τις ακμές εκτός από αυτές που δείχνουμε με διακεκομμένη γραμμή. Για ευκολία στην αναφορά, τα σύμβολα στους κόμβους αντιστοιχούν επίσης στα αναγνωριστικά του αντικειμένου σύμφωνα με το interval-based scheme, δηλαδή ισχύει ότι $\texttt{post}(u) = u$. Τα διαστήματα που συνδέονται με κάθε κόμβο δείχνονται με κόκκινο. Παρατηρούμε ότι ο κόμβος w διαθέτει δύο διαστήματα, ένα από την ακμή του δέντρου επικάλυψης (r,w), κι ένα ακόμα (q,y), που προέρχεται από τον κόμβο y.



Σχήμα 5.1: Παράδειγμα ετικετών διαστήματος.

Ο Πίνακας 5.1 δείχνει τις τιμές των χαρακτηριστικών των πλειάδων που αποθηκεύονται στους κόμβους του MIDAS-RDF. Σε αυτό το παράδειγμα, η διάσταση που αντιστοιχεί στο κατηγόρημα παραλείπεται δεδομένου ότι είναι κοινή για όλες τις τριπλέτες. Οπότε έχουμε ότι το κατηγόρημα μαζί με τις στήλες 1, 2 και 3 αποτελούν το κλειδί της πλειάδας.

Subject	Object	Post	Intervals
u	v	v	$[p,v]$
u	w	w	$[r,w],[q,y]$
v	x	x	$[p,x]$
v	y	y	$[q,y]$
w	y	y	$[q,y]$
w	z	z	$[r,z]$

Πίνακας 5.1: Αποθηκευμένες πλειάδες επαυξημένες με την τεχνική επισήμανσης.

5.2 Ερωτήματα RDF

Το MIDAS-RDF υποστηρίζει RDF ερωτήματα τριπλού μοτίβου. Ας εξετάσουμε τους τύπους ερωτημάτων που φαίνονται στον Πίνακα 5.2. Ο συμβολισμός $?s$ (κι αντίστοιχα $?p$, $?o$) είναι δανεισμένος από τη SPARQL [**?**] κι υποδηλώνει ότι η μεταβλητή $?s$ αντιστοιχεί στα πολλά υποκείμενα που ενδεχομένως ικανοποιούν το ερώτημα. Ο τύπος ερωτήματος Q1 αποτιμάται ως ερώτημα σημείου κι απαιτεί λογαριθμικό αριθμό βημάτων. Οι υπόλοιποι τύποι ερωτημάτων αποτιμούνται ως ερωτήματα εύρους, όπου οι μεταβλητές $?s$, $?p$ και $?o$ αντιστοιχούν σε ολόκληρο το εύρος του πεδίου ορισμού του συγκεκριμένου χαρακτηριστικού. Για παράδειγμα, στον γράφο RDF που απεικονίζεται στην Εικόνα 2.4(α΄) το ερώτημα τύπου Q2 `SELECT ?paint WHERE {Picasso paints ?paint}` επιστρέφει όλα τα έργα ζωγραφικής του Picasso. Ένα ερώτημα τύπου Q8 ανακτά ουσιαστικά όλες τις τριάδες RDF που αν και προσπελαύνει όλους τους κόμβους, τους επισκέπτεται σε λογαριθμικό αριθμό βημάτων.

Τα διαζευκτικά ερωτήματα εύρους επιλύονται με την έκδοση ενός ερώτηματος για κάθε μέρος της διάξευξης και στη συνέχεια υπολογίζεται η ένωση των αποτελεσμάτων. Ένας

Name	Subject	Predicate	Object
Q1	s	p	o
Q2	s	p	?o
Q3	s	?p	o
Q4	s	?p	?o
Q5	?s	p	o
Q6	?s	p	?o
Q7	?s	?p	o
Q8	?s	?p	?o

Πίνακας 5.2: Αποθηκευμένες πλειάδες επαυξημένες με την τεχνική επισήμανσης.

τέτοιος μηχανισμός μπορεί να υλοποιηθεί αποτελεσματικά μέσω της έκδοσης ταυτόχρονων ατομικών ερωτημάτων. Ένα συζευκτικό ερώτημα, όπως το `SELECT ?artifact WHERE {{?x type Sculptor} {?x type Painter} {?x creates ?artifact}}`, θα επέστρεφε τα αντικείμενα που δημιουργήθηκαν από έναν καλλιτέχνη που ήταν ζωγράφος *και* γλύπτης, π.χ., ο Δαβίδ του Μιχαήλ Αγγέλου. Ένας απλός τρόπος για να υποστηριχθούν τα συνδετικά ερωτήματα είναι να εκτελεστεί πρώτα το πιο επιλεκτικό (selective) ερώτημα της σύζευξης (αν υποθεσουμε ότι υπάρχει τέτοια πληροφορία, π.χ. μέσω ιστογραμμάτων). Στη συνέχεια, το αποτέλεσμα φιλτράρεται κατάλληλα στον κόμβο που εξέδωσε το ερώτημα.

Μια σημαντική κατηγορία ερωτημάτων αφορά τις μεταβατικές ιδιότητες των κατηγορημάτων. Για παράδειγμα, ας εξετάσουμε τον *sc* υπογράφο του Σχήματος 2.4(β΄). Ένα ερώτημα για τις έννοιες που περικλείουν την έννοια `Cubist` θα ανακτούσε τις έννοιες `Painter` και `Artist`. Η αποτελεσματική επεξεργασία τέτοιων μεταβατικών ερωτημάτων απαιτεί ένα σύστημα επισήμανσης. Έτσι, πρώτα ανακτόνται πλειάδες που συνοδεύονται από τα ανάλογα διαστήματα που υποδηλώνουν τη μεταβατική σχέση, π.χ. $\langle$`Cubist`$,sc,$?o$\rangle$, και στη συνέχεια εκδίδονται ταυτόχρονα τα ερωτήματα εύρους για τις πλειάδες που έχουν αναγνωριστικά εντός των διαστημάτων που ανακτήθηκαν. Οι έννοιες που περιλαμβάνουν το v στο Σχήμα 5.1 έχουν αναγνωριστικά που ανήκουν στο διάστημα $[p,v]$. Αν και τα ερωτήματα είναι μεταβατικά η επεξεργασία τους απαιτεί λογαριθμικό αριθμό βημάτων ανεξάρτητα από το βάθος του γράφου επειδή μετατρέπονται σε ερωτήματα εύρους.

5.3 Μέθοδοι Κατανεμημένης Συλλογιστικής

Η Ενότητα 5.3.1 παρουσιάζει τις βασικές αρχές του επαγωγικού μοντέλου που χρησιμοποιείται στο MIDAS-RDF. Στην Ενότητα 5.3.2 περιγράφουμε τη διαδικασία επεξεργασίας των δεδομένων σε συνδυασμό με τους RDFS κανόνες συμπερασμού που έχουν προταθεί από το W3C.

5.3.1 Επαγωγικό Μοντέλο Εξαγωγής Συμπερασμάτων

Στην ενότητα αυτή περιγράφουμε πως οι RDFS επαγωγικοί κανόνες μπορούν να αντιμετωπιστούν ως ένα σύνολο κανόνων οι οποίοι μπορούν να παράξουν νέες πλειάδες βάσει παλαιότερων. Στο κεφάλαιο αυτό επικεντρωνόμαστε σε ένα υποσύνολο των κανόνων αυτών συμπεριλαμβανομένου και των επεκτεταμένων κανόνων (extensional entailment rules) τους οποίους συνοψίζουμε στον Πίνακα 5.3 που περιλαμβάνει τους πιο δαπανηρούς υπολογιστικά κανόνες. Οι ιδιότητες `sp` και `sc` αποτελούν συντομογραφίες για τις ιδιότητες `subPropertyOf` και `subClassOf` αντίστοιχα.

Σε αντίθεση με άλλες προσεγγίσεις, στο MIDAS-RDF δεν απαιτείται η ξεχωριστή δημιουργία κάποιου πλάνου εκτέλεσης για τον επιμερισμό των μερών της απαιτούμενης επεξεργασίας στους κόμβους. Αντίθετα, οι κόμβοι χωρίζουν το πρόβλημα αυτόνομα καθώς ο καθένας επικεντρώνεται σε ένα τμήμα του προβλήματος προκειμένου να βρει λύσεις οι οποίες γίνονται διαθέσιμες προς όλους. Συγκεκριμένα, ακολουθείται η μέθοδος του forward chaining. Κάθε φορά που κάποιος κόμβος λαμβάνει μία νέα πλειάδα προς αποθήκευση, θα παράξει όλες τις

Rule	Precondition	Generated Triple
rdfs2	$(\alpha, \texttt{domain}, x), (u, \alpha, v)$	$(u, \texttt{type}, x)$
rdfs3	$(\alpha, \texttt{range}, x), (u, \alpha, v)$	$(v, \texttt{type}, x)$
rdfs5	$(\alpha, \texttt{sp}, \beta), (\beta, \texttt{sp}, \gamma)$	$(\alpha, \texttt{sp}, \gamma)$
rdfs7	$(\alpha, \texttt{sp}, \beta), (u, \alpha, v)$	(u, β, v)
rdfs9	$(u, \texttt{sc}, v), (w, \texttt{type}, u)$	$(w, \texttt{type}, v)$
rdfs11	$(u, \texttt{sc}, v), (v, \texttt{sc}, w)$	$(u, \texttt{sc}, w)$
ext1	$(\alpha, \texttt{domain}, u), (u, \texttt{sc}, v)$	$(\alpha, \texttt{domain}, v)$
ext2	$(\alpha, \texttt{range}, u), (u, \texttt{sc}, v)$	$(\alpha, \texttt{range}, v)$
ext3	$(\alpha, \texttt{domain}, u), (\beta, \texttt{sp}, \alpha)$	$(\beta, \texttt{domain}, u)$
ext4	$(\alpha, \texttt{range}, u), (\beta, \texttt{sp}, \alpha)$	$(\beta, \texttt{range}, u)$

Πίνακας 5.3: Συμπερασματικοί κανόνες RDFS του W3C.

πλειάδες που συνάγονται και τις εισάγει με τη σειρά τους χρησιμοποιώντας την υποδομή του δικτύου. Φυσικά, οι παραγόμενες πλειάδες θα συνδυαστούν με κάποιους άλλους κανόνες για περαιτέρω επεξεργασία.

Προκειμένου να αποφύγουμε τη δημιουργία διπλότυπων, ο κάθε κανόνας εκτελείται από τον έναν εκ των δύο κόμβων που έχουν αποθηκευμένες τις πλειάδες εισόδου. Όμως το μέρος που επιλέγεται κάθε φορά γίνεται βάσει κριτηρίων που αφορούν την απόδοση και την επεκτασιμότητα. Οπότε ο σχεδιασμός γίνεται με σκοπό να ελαχιστοποιηθούν τα παραγόμενα μηνύματα κι ως εκ τούτου το απαιτουμενο εύρος ζώνης (bandwidth) που αποτελεί ίσως τον πιο ακριβό πόρο σε ένα κατανεμημένο περιβάλλον.

Το σημαντικό πλεονέκτημα του MIDAS-RDF είναι ο χαμηλός χρόνος εκτέλεσης συγκριτικά με άλλες προσεγγίσεις. Για παράδειγμα, σύμφωνα με τις συμβατικές τεχνικές, η μεταβατική κλειστότητα του κανόνα συνεπαγωγής *rdfs11* σε ένα `subClassOf` υπογράφο βάθους k θα απαιτούσε μία σειρά από τουλάχιστον k λειτουργίες που απαιτούν λογαριθμικό αριθμό από βήματα έκαστη. Από την άλλη, με την τεχνική που παρουσιάσαμε μπορούμε να υπολογίσουμε τη μεταβατική κλειστότητα σε $O(\log n)$ βήματα ανεξάρτητα από το βάθος του γράφου. Αυτό επιτυγχάνεται μέσω κατάλληλων ερωτημάτων εύρους, όπως αυτά περιγράφονται στην Ενότητα 5.2. Ας θεωρήσουμε τον RDF γράφο της Εικόνας 2.4(α΄) κι ας αναλογιστούμε τον κανόνα *rdfs9*. Κι οπότε ένας κόμβος θα παράξει τις πλειάδες `(Picasso, type, Painter)` και `(Picasso, type, Artist)`. Ως αποτέλεσμα, το ερώτημα `SELECT ?x WHERE {?x type Artist}` θα επέστρεφε επίσης το όνομα του Picasso, το οποίο διαφορετικά θα παρελειπόταν αν δεν είχε προηγηθεί η διαδικασία του συμπερασμού.

5.3.2 Κατανεμημένοι Μέθοδοι Επεξεργασίας Κανόνων Συνεπαγωγής

Στην ενότητα αυτή παρουσιάζουμε τις μεθόδους για την επεξεργασία των κανόνων του Πίνακα 5.3 κι επίσης περιγράφουμε τεχνικές ενημερώσεως πλειάδων.

Εφαρμογή Συμπερασματικών Κανόνων rdfs2 και rdfs3

Ο Αλγόριθμος 7 παράγει νέα γνώση χρησιμοποιώντας τους κανόνες συνεπαγωγής *rdfs2* και *rdfs3* του W3C. Αυτή η μέθοδος καλείται να εκτελεστεί όταν μία τριπλέτα της μορφής $(\alpha, \texttt{domain}, u)$ ή $(\alpha, \texttt{range}, v)$ αποθηκεύεται τοπικά στον κόμβο που είναι υπεύθυνος. Μία τέτοια τριπλέτα θα πρέπει να συσχετιστεί με τις εγγραφές του κατηγορήματος α οι οποίες θα ανακτηθούν από τους απομακρυσμένους κόμβους με την κλήση ενός ερωτήματος εύρους. Ο συνδυασμός αυτών με την νέα πλειάδα μέσω οποιουδήποτε από τους δύο κανόνες συνεπαγωγής θα παράξει καινούρια δεδομένα.

Το παράδειγμα που δείχνουμε στον Πίνακα 5.4 αφορά τον κανόνα συμπερασμού *rdfs2* και μπορεί να περιγραφεί εν συντομία από τα ακόλουθα βήματα:

1. Ο κόμβος p_2 εξετάζει την τριπλέτα (u, α, v) και εκδίδει ένα ερώτημα εύρους της μορφής $(\alpha, \texttt{domain}, ?X)$ για να ανακτήσει $(\alpha, \texttt{domain}, x)$ από τον κόμβο p_1.

2. Ο κόμβος p_2 επεξεργάζεται την πλειάδα που έλαβε από το ερώτημα εύρους συνδυάζοντάς την με τον κανόνα *rdfs2* για να παράξει την τριπλέτα $(u,\texttt{type},x)$.

3. Ο κόμβος p_2 εισάγει προς αποθήκευση στο δίκτυο τις παραγόμενες εγγραφές στο MIDAS και στη συνέχεια οι κόμβοι οι οποίοι είναι υπεύθυνοι για τις νεο-εισαχθείσες εγγραφές θα καλέσουν με τη σειρά τους τον Αλγόριθμο 11 προκειμένου να εφαρμόσουν τον επαγωγικό κανόνα *rdfs9*.

Το Σχήμα 5.4 παρουσιάζει μία παρόμοια προσέγγιση για την εφαρμογή του επαγωγικού κανόνα *rdfs3*. Σημειωτέον ότι η κλήση του Αλγορίθμου 11 γίνεται για τις πλειάδες που μόλις έχουν εισαχθεί σε έναν κόμβο. Προκειμένου να αποφευχθεί η δημιουργία διπλότυπων εγγραφών εξετάζουμε ένα προκαθορισμένο εκ των προτέρων μέρος του σώματος του κάθε κανόνα. Παρ' όλα αυτά, καθώς οι νέες τριπλέτες παράγονται και εισάγονται στο MIDAS, υπάρχει η ανάγκη να σχεδιαστούν μέθοδοι για τον χειρισμό κανόνων συνεπαγωγής για το άλλο μέλος του σώματος του κανόνα. Αυτή η περίπτωση καλύπτεται από τον Αλγόριθμο 11 για τον κανόνα συμπερασμού *rdfs9*. Προτείνουμε παρόμοιες μεθόδους με μικρές αλλαγές όπου αυτές απαιτούνται για τους κανόνες που δεν παρουσιάζονται ως προς τα δύο σκέλη τους, όπως οι κανόνες *rdfs7*, *ext1* και *ext2*. Ως έναν βαθμό οι δύο τρόποι συνδυασμού του ίδιου κανόνα είναι ισοδύναμοι κι οι επιλογές γίνονται επί τη βάσει της απόδοσης.

Peer	Local Store	Generated Triples	Rule
p_1	$(\alpha,\texttt{domain},x)$		
p_2	(u,α,v)	$(u,\texttt{type},x),(v,\texttt{type},z)$	*rdfs2,3*
p_3	$(\alpha,\texttt{range},z)$		

Πίνακας 5.4: Παράδειγμα συνεπαγωγής για τους συμπερασματικούς κανόνες *rdfs2* και *rdfs3*.

Algorithm 7: deduceRDFS2/3a: για είσοδο μίας RDF τριπλέτας της μορφής (u,α,v), ανακτά συσχετιζόμενες τριπλέτες σύμφωνα με τους συμπερασματικούς κανόνες *rdfs2* and *rdfs3* κι εξάγει νέα πληροφορία προς αποθήκευση.

```
1 /* Associated rule: RDFS2 */;
2 p.rangeRDFquery (β, domain, ?X) forall the x in X do
3 |  p.insert (u, type, x);
4 /* Associated rule: RDFS3 */ p.rangeRDFquery (β, range, ?Y) forall the y in Y do
5 |  p.insert (v, type, y);
```

Algorithm 8: deduceRDFS2b: πυροδοτείται με την είσοδο μίας RDF τριπλέτας της μορφής $(\alpha,\texttt{domain},u)$, κι ανακτά συσχετιζόμενες τριπλέτες σύμφωνα με το συμπερασματικό κανόνα *rdfs2* κι εξάγει νέα πληροφορία προς αποθήκευση.

```
1 p.rangeRDFquery (?X, α, ?Y) forall the x in X do
2 |  p.insert (x, type, u);
```

Εφαρμογή Συμπερασματικών Κανόνων rdfs5 και rdfs7

Οι Αλγόριθμοι 9 και 10 δικαιολογούν τη σχεδιαστική επιλογή χρήσης τεχνικών επισήμανσης για την επεξεργασία μεταβατικών κανόνων συμπερασμού στο MIDAS-RDF καθώς μειώνουν τον αριθμό των βημάτων από $O(k\log n)$ που ισχύει για τις συμβατικές μεθόδους της βιβλιογραφίας σε μόλις $O(\log n)$. Το παράδειγμα του Αλγορίθμου 9 που δείχνουμε σχηματικά στον Πίνακα 5.5 μπορεί να περιγραφεί συνοπτικά με τα ακόλουθα βήματα:

1. Ο κόμβος p_1 εξετάζει την τριπλέτα $(\alpha, \texttt{subPropertyOf}, \beta)$ κι αποκτά τις τοπικά αποθηκευμένες ετικέτες των τριπλετών που αντιστοιχούν στους κόμβους που συμπεριλαμβάνουν το κατηγόρημα β στον γράφο της `subPropertyOf` ιδιότητας.

2. Ο κόμβος p_1 λαμβάνει όλες τις τριπλέτες που αντιστοιχούν στις ανακτηθείσες ετικέτες σε $O(\log n)$ βήματα από τους κόμβους p_2 και p_3, δηλαδή τις τριπλέτες $(\beta, \texttt{subPropertyOf}, \zeta)$ και $(\zeta, \texttt{subPropertyOf}, \xi)$.

3. Ο κόμβος p_1 εισάγει προς αποθήκευση στο δίκτυο τις παραγόμενες πλειάδες από τον κανόνα *rdfs5*, $(\alpha, \texttt{subPropertyOf}, \zeta)$ και $(\alpha, \texttt{subPropertyOf}, \xi)$.

4. Ομοίως, ο κόμβος p_2 εξετάζει την τριπλέτα $(\beta, \texttt{subPropertyOf}, \zeta)$ για να ανακτήσει την $(\zeta, \texttt{subPropertyOf}, \xi)$ από τον κόμβο p_3 και με την εφαρμογή του κανόνα *rdfs5* παράγει την $(\beta, \texttt{subPropertyOf}, \xi)$ για να την εισάγει προς αποθήκευση στο δίκτυο.

5. Ο κόμβος p_3 εξετάζει την $(\zeta, \texttt{subPropertyOf}, \xi)$ για να διαπιστώσει ότι δεν υπάρχουν άλλες εγγραφές που να εμπίπτουν στο κατηγόρημα ξ.

6. Ο κόμβος p_2 ανακτά την πλειάδα (u, α, v) (Alg. 9) στα πλαίσια της εφαρμογής του κανόνα *rdfs7* με την χρήση των τοπικών ετικετών με τα αναγνωριστικά των εγγραφών που υπόκεινται στο κατηγόρημα ζ.

7. Τότε, ο κόμβος p_2, αφού εκδίδει ένα ερώτημα για όλες τις σχετικές ετικέτες που ανήκουν στο διάστημα που έχει σχηματιστεί, ανακτά την τριπλέτα $(\zeta, \texttt{subPropertyOf}, \xi)$ από τον κόμβο p_3.

8. Με την εφαρμογή του *rdfs7* από τον κόμβο p_2 παράγονται οι πλειάδες (u, ζ, v) και (u, ξ, v) εισάγονται προς αποθήκευση στο δίκτυο.

9. Οι κόμβοι οι οποίοι είναι υπεύθυνοι για τις νεο-εισαχθείσες πλειάδες που παράχθηκαν από τον κόμβο p_2 θα εκτελέσουν τον Αλγόριθμο 7 για τον χειρισμό τους σύμφωνα με τους συμπερασματικούς κανόνες *rdfs2* και *rdfs3*.

Peer	Local Store	Generated Triples	Rule
p_1	$(\alpha, \texttt{sp}, \beta)$	$(\alpha, \texttt{sp}, \zeta), (\alpha, \texttt{sp}, \xi)$	*rdfs5*
p_2	$(\beta, \texttt{sp}, \zeta)$	$(\beta, \texttt{sp}, \xi), (u, \zeta, v), (u, \xi, v)$	*rdfs5,7*
p_3	$(\zeta, \texttt{sp}, \xi)$		
p_4	(u, β, v)		

Πίνακας 5.5: Παράδειγμα συνεπαγωγής για τους συμπερασματικούς κανόνες *rdfs5* και *rdfs7*.

Αντί του να εκδίδουμε αλληλουχίες σειριακών ερωτημάτων, ο κόμβος p_1 ανακτά τον υπογράφο με ένα απλό ερώτημα εύρους για τις RDF εγγραφές με τα σχετικά αναγνωριστικά. Ειδικότερα, ο κόμβος p_4 θα πρέπει να εκδόσει τέσσερα διαφορετικά ερωτήματα προκειμένου να παραχθούν όλες οι δυνατές πλειάδες όσον αφορά την επεξεργασία του κανόνα συνεπαγωγής *rdfs7*. Είναι ιδιαίτερα σημαντικό το γεγονός ότι ο κόμβος p_4 είναι σε θέση να υπολογίσει ανεξάρτητα την μεταβατική κλειστότητα των εγγραφών που επεξεργάζεται χωρίς να είναι ενήμερος για την πρόοδο των άλλων κόμβων, για την περίπτωσή μας οι κόμβοι p_1, p_2, p_3, ή να είναι υπόλογος προς τους άλλους κόμβους για την πρόοδό του, ή να αναμένει τα όποια ενδιάμεσα αποτελέσματα.

Εφαρμογή Συμπερασματικών Κανόνων rdfs11 και rdfs9

Το παράδειγμα του Αλγορίθμου 10 που δείχνουμε σχηματικά στον Πίνακα 5.6 μπορεί να περιγραφεί συνοπτικά από τα ακόλουθα βήματα:

1. Ο κόμβος p_1 εξετάζει την τοπική εγγραφή $(u, \texttt{subClassOf}, v)$ κι ελέγχει τις ετικέτες των τριπλετών που υπόκεινται στο αντικείμενο v.

2. Ο κόμβος p_1 τις πλειάδες: $(v, \texttt{subClassOf}, w)$ από τον κόμβο p_2 και $(w, \texttt{subClassOf}, y)$ από τον κόμβο p_3.

3. Ο κόμβος p_1 εισάγει προς αποθήκευση τις πλειάδες $(u, \texttt{subClassOf}, w)$ και $(u, \texttt{subClassOf}, y)$.

4. Όταν ο κόμβος p_2 εξετάζει την $(v, \texttt{subClassOf}, w)$, θα ανακτήσει από τον κόμβο p_3 την πλειάδα $(w, \texttt{subClassOf}, y)$.

5. Ο κόμβος p_2 εισάγει προς αποθήκευση στο δίκτυο την πλειάδα $(v, \texttt{subClassOf}, y)$.

6. Ο κόμβος p_3 εξετάζει την $(w, \texttt{subClassOf}, y)$ για να διαπιστώσει ότι δεν υπόκειται κάποια άλλη εγγραφή στο αντικείμενο y. Επίσης εκδίδει ένα ερώτημα εύρους της μορφής $(?X, \texttt{type}, w)$ για να ανακτήσει εν τέλει την εγγραφή $(x, \texttt{type}, w)$ από τον κόμβο p_4.

7. Ο κόμβος p_3 εφαρμόζει τον κανόνα συνεπαγωγής *rdfs9* και παράγει την πλειάδα $(x, \texttt{type}, y)$ που εισάγει στο δίκτυο προς αποθήκευση.

Peer	Local Store	Generated Triples	Rule
p_1	$(u, \texttt{sc}, v)$	$(u, \texttt{sc}, w), (u, \texttt{sc}, y)$	*rdfs11*
p_2	$(v, \texttt{sc}, w)$	$(v, \texttt{sc}, y)$	*rdfs11*
p_3	$(w, \texttt{sc}, y)$	$(x, \texttt{type}, y)$	*rdfs9*
p_4	$(x, \texttt{type}, w)$		

Πίνακας 5.6: Παράδειγμα συνεπαγωγής για τους συμπερασματικούς κανόνες *rdfs11* και *rdfs9*.

Εφαρμογή Συμπερασματικών Κανόνων ext1 και ext2

Για την επεξεργασία των κανόνων συνεπαγωγής *ext1* και *ext2*, ακολουθούμε το παράδειγμα της προσέγγισης που ακολουθήσαμε για τον κανόνα *rdfs9*. Το παράδειγμα που δείχνουμε στον Πίνακα 5.7 περιγράφεται σχηματικά παρακάτω:

1. Ο κόμβος p_2 εξετάζει την πλειάδα $(u, \texttt{subClassOf}, v)$ και για να εφαρμόσει τον κανόνα συνεπαγωγής *ext1* ανακτά τις πλειάδες που αντιστοιχούν στις ετικέτες που υπόκεινται στο αντικείμενο v, δηλαδή μόνο την $(v, \texttt{subClassOf}, w)$ για την περίπτωσή μας.

2. Όταν εφαρμόζεται ο κανόνας συνεπαγωγής *ext1* για κάθε μία από τις ανακτηθείσες πλειάδες (Αλγ. 10), ο κόμβος p_2 παράγει κι εισάγει προς αποθήκευση τις πλειάδες $(\alpha, \texttt{domain}, v)$ και $(\alpha, \texttt{domain}, w)$, αφού ανακτήσει από τον κόμβο p_1 την $(\alpha, \texttt{domain}, u)$ μέσω ενός ερωτήματος εύρους.

3. Οι κόμβοι οι οποίοι είναι υπεύθυνοι για τις νεο-εισαχθείσες πλειάδες θα εκτελέσουν τον Αλγόριθμο 8 για την επεξεργασία των δεδομένων όπως επιβάλλει ο κανόνας *rdfs2*.

4. Ομοίως, ο κόμβος p_3 εφαρμόζει τον κανόνα *ext2* στην πλειάδα $(\beta, \texttt{range}, v)$ την οποία ανακτά από τον κόμβο p_4 σε $O(\log n)$ βήματα για να παράξει εν τέλει την πλειάδα $(\beta, \texttt{range}, w)$.

Peer	Local Store	Generated Triples	Rule
p_1	$(\alpha, \texttt{domain}, u)$		
p_2	$(u, \texttt{sc}, v)$	$(\alpha, \texttt{domain}, v), (\alpha, \texttt{domain}, w)$	*ext1*
p_3	$(v, \texttt{sc}, w)$	$(\beta, \texttt{range}, w)$	*ext2*
p_4	$(\beta, \texttt{range}, v)$		

Πίνακας 5.7: Παράδειγμα συνεπαγωγής για τους συμπερασματικούς κανόνες *ext1* και *ext2*.

Εφαρμογή Συμπερασματικών Κανόνων ext3 και ext4

Για την επεξεργασία της γνωσης βάσης σύμφωνα με τους κανόνες συνεπαγωγής *ext3* και *ext4* θα πρέπει να υιοθετηθεί μία διαφορετική προσέγγιση. Οι προαναφερθείσες μέθοδοι αφορούσαν μεταβατικές σχέσεις των οποίων τα αποτελέσματα συνδυάζονταν με σταθερές δηλώσεις. Για τους συγκεκριμένους κανόνες υπολογίζουμε τη μεταβατική κλειστότητα για κάθε δήλωση που εξετάζουμε και συνδυάζουμε κάθε στοιχείο του αποτελέσματος με τις απομακρυσμένες πλειάδες που μπορούν να συσχετιστούν βάσει των κανόνων *ext3* και *ext4*. Στη συνέχεια δίνουμε μία διαισθητική περιγραφή των αρχών που χρησιμοποιούνται στον Αλγόριθμο 9 με το παράδειγμα του Πίνακος 5.8:

1. Ο κόμβος p_3 εξετάζει την πλειάδα $(\zeta, \texttt{subPropertyOf}, \beta)$ και χρησιμοποιεί την συσχετισμένη με το συγκεκριμένο κλειδί πληροφορία προκειμένου να ανακτήσει τα διαστήματα των ετικετών που υπόκεινται στο υποκείμενο ζ ως προς την ιδιότητα `subPropertyOf`.

2. Ο κόμβος p_3 εκδίδει τα αντίστοιχα ερωτήματα εύρους ως προς το πεδίο που χρησιμοποιείται για την επισήμανση για να ανακτήσει την πλειάδα $(\beta, \texttt{subPropertyOf}, \alpha)$ από τον κόμβο p_2.

3. Ο κόμβος p_3 ανακτά όλες τις εγγραφές της μορφής $(\alpha, \texttt{domain}, u)$ για κάθε ένα υποκείμενο α που προκύπτει. Για την περίπτωσή μας είναι οι πλειάδες $(\alpha, \texttt{domain}, u)$ από τον κόμβο p_1 και η $(\alpha, \texttt{range}, v)$ από τον κόμβο p_4.

4. Όταν ο κόμβος p_3 εφαρμόζει τον κανόνα *ext3* θα παράξει την πλειάδα $(\zeta, \texttt{domain}, u)$ και την εισάγει προς αποθήκευση με την χρήση των υποδομών του δικτύου. Κατ' αναλογίαν, ο κόμβος p_3 εξάγει την $(\zeta, \texttt{range}, v)$ αξιοποιώντας τον κανόνα συνεπαγωγής *ext4*.

5. Οι κόμβοι που είναι υπεύθυνοι για τις νεο-εισερχόμενες πλειάδες θα καλέσουν με τη σειρά τους τον Αλγόριθμο 8 για την επεξεργασία των $(\zeta, \texttt{domain}, u)$ και $(\zeta, \texttt{range}, v)$ βάσει των κανόνων *rdfs2* και *rdfs3*.

6. Παρόμοια διαδικασία ακολουθείται κι από τον κόμβο p_2 για να παραχθούν οι $(\beta, \texttt{domain}, u)$ και $(\beta, \texttt{range}, v)$ για τους κανόνες *ext3* και *ext4*.

Peer	Local Store	Generated Triples	Rule
p_1	$(\alpha, \texttt{domain}, u)$		
p_2	$(\beta, \texttt{sp}, \alpha)$	$(\beta, \texttt{domain}, u), (\beta, \texttt{range}, v)$	*ext3*,*4*
p_3	$(\zeta, \texttt{sp}, \beta)$	$(\zeta, \texttt{domain}, u), (\zeta, \texttt{range}, v)$	*ext3*,*4*
p_4	$(\alpha, \texttt{range}, v)$		

Πίνακας 5.8: Παράδειγμα συνεπαγωγής για τους συμπερασματικούς κανόνες *ext3* και *ext4*.

Ενημερώσεις Πραγματικού Χρόνου

Μία ενημέρωση (update) σε κάποια τριπλέτα πυροδοτεί με τη σειρά της αλλαγές και σε όλες τις πλειάδες που παράχθηκαν βασιζόμενες σε αυτή σε συνδυασμό με κάποιο σύνολο κανόνων από αυτούς που παρουσιάσαμε προηγουμένως, δεδομένου ότι οι τεχνικές που έχουν υιοθετηθεί βασίζονται στις αρχές του forward chaining. Έτσι απαιτείται ο σχεδιασμός τεχνικών αναίρεσης ή ενημέρωσης των συμπερασμάτων εφόσον έχει αλλάξει η πληροφορία στην οποία βασίστηκαν. Εφόσον δεν παρέχεται κάποιος μηχανισμός που να παρέχει πληροφορία για την προέλευση των δεδομένων (data provenance), θα ακολουθήσουμε την ανάστροφη διαδικασία των μεθόδων συμπερασμού έχοντας όμως προνοήσει παράλληλα πριν την αναίρεση μίας οποιαδήποτε δήλωσης να εξακριβώσουμε εάν υπάρχουν ή όχι αλλές προϋποθέσεις που να παραμένουν αρραγείς στη βάση γνώσης και να οδηγούν στο ίδιο συμπέρασμα όταν συνδυαστούν με κάποιους RDFS κανόνες συνεπαγωγής. Ο έλεγχος αυτός μπορεί να υποστηριχθεί με την προσθήκη κάποιων επιπλέον βημάτων επεξεργασίας των ad hoc ενημερώσεων.

Algorithm 9: deduceRDFS5/7/EXT3/4: για είσοδο μίας RDF τριπλέτας της μορφής $(\beta, \texttt{sp}, \alpha)$, ανακτά συσχετιζόμενες τριπλέτες σύμφωνα με τους συμπερασματικούς κανόνες *ext3* and *ext4* κι εξάγει νέα πληροφορία προς αποθήκευση.

```
1  Λ = p.getTriplesThatSubsume (α);
2  forall the (s,p,o) in Λ do
3  |  /* Associated rule: RDFS5 */;
4  |  p.insert (β, subPropertyOf, o);
5  |  /* Associated rule: RDFS7 */;
6  |  p.rangeRDFquery (?U, β, ?V);
7  |  forall the (u,v) in (U,V) do
8  |  |  p.insert (u, o, v);
9  |  /* Associated rule: EXT3 */;
10 |  p.rangeRDFquery (o, domain, ?X);
11 |  forall the x in X do
12 |  |  p.insert (β, domain, x);
13 |  /* Associated rule: EXT4 */;
14 |  p.rangeRDFquery (o, range, ?Y);
15 |  forall the y in Y do
16 |  |  p.insert (β, range, y);
```

Το παράδειγμα του Αλγορίθμου 12 στον Πίνακα 5.9 εφαρμόζεται για τις διαγραφές τριπλετών της μορφής $(\alpha, \texttt{subPropertyOf}, \beta)$ ή (u, α, v) από τους κόμβους p_1, p_2 που θέτει το θέμα της αναίρεσης της εγγραφής (u, β, v) που εξήχθει από την επεξεργασία του κανόνα συμπερασμού *rdfs7*. Το πρόβλημα έγκειται στο ότι η αναίρεση στην πραγματικότητα δεν θα πρέπει να πραγματοποιηθεί αφού η ίδια δήλωση έχει εξαχθεί από τον κόμβο p_4 όταν εφάρμοσε τον κανόνα *rdfs7* στα περιεχόμενά του. Οπότε, καθώς άνω του ενός συνδυασμοί κανόνων και εγγραφών μπορούν κάλλιστα να οδηγήσουν στα ίδια συμπεράσματα, στο MIDAS-RDF επιβάλλεται να εξετάσουμε ενδελεχώς για το εάν υπάρχει έστω ένας εναλλακτικός τρόπος που να παράγει την ίδια πρόταση πριν την αναιρέσουμε όταν κάποια από τις προϋποθέσεις της δεν ισχύουν πια. Εν τοιαύτη περιπτώσει επαναλαμβάνουμε την ίδια διαδικασία για την επόμενη πρόταση που έχει εξαχθεί από κάποια εγγραφή πριν ενημερωθεί. Για παράδειγμα, στον Πίνακα 5.9 όταν η πλειάδα (u, α, v) διαγράφεται και εξετάζεται η (u, β, v) θα πρέπει να εκδοθούν ταυτόχρονα όλα τα κατάλληλα αιτήματα τα οποία θα εντοπίζουν εναλλακτικούς τρόπους εξαγωγής του ιδίου συμπεράσματος. Πρώτα θα πρέπει να εξεταστούν όλοι οι κανόνες συνεπαγωγής κι από αυτούς θα κρατήσουμε μόνον εκείνους που μπορούν να συνδυαστούν ώστε να παράξουν την (u, β, v) τριπλέτα. Έπειτα, θα πρέπει να εξακριβώσουμε εάν στην κατανεμημένη γνώση βάσης υπάρχουν αποθηκευμένες άλλες πλειάδες οι οποίες μπορούν σε συνδυασμό με κάποιον από τους υποψήφιους κανόνες να οδηγήσουν στον συμπερασμό της (u, β, v). Στην περίπτωσή μας, ο κανόνας *rdfs7* είναι ο μοναδικός που θα χρειαστεί να ελεγχθεί. Ο έλεγχος αυτός διεξάγεται απλά με την έκδοση ενός ερωτήματος εύρους της μορφής $(u, ?\Gamma, v)$ η επεξεργασία του οποίου θα επιστρέψει την εγγραφή (u, γ, v) που είναι αποθηκευμένη στον κόμβο p_3. Οπότε τώρα θα πρέπει να ελέγξουμε για το εάν το κατηγόρημα β είναι πρόγονος του γ ως προς την ιδιότητα `subPropertyOf`, κάτι που γίνεται απλά χρησιμοποιώντας το σύστημα επισήμανσης. Πιο συγκεκριμένα, εξετάζουμε εάν το μεταθεματικό αναγνωριστικό (postorder identifier) του κόμβου γ στον αντίστοιχο RDF γράφο ανήκει σε οποιοδήποτε από τα διαστήματα που συνοδεύουν την εγγραφή με το κλειδί που αντιστοιχεί στην πρόταση (u, β, v). Φαινομενικά, το κόστος ενός τέτοιου μηχανισμού δεν είναι δραματικό εφόσον υπάρχει η δυνατότητα να αναλυθούν ταυτόχρονα οι όποιες υποψήφιες διαδρομές οι οποίες με τη χρήση μεταβατικών κανόνων οδηγούν στην ίδια πρόταση. Εν κατακλείδι, ο μηχανισμός αναίρεσης που πυροδοτείται όταν προκύπτουν αλλαγές στη βάση γνώσης χρησιμοποιεί τις προαναφερθείσες μεθόδους στις οποίες όμως έχουν αντικατασταθεί οι κλήσεις `p.insert()` με τις αντίστοιχες `p.delete()` κλήσεις. Τέλος, οι ενημερώσεις

μπορούν να θεωρηθούν ως ένα ζεύγος λειτουργιών, τη διαγραφή μίας παλιάς πλειάδας και την εισαγωγή της πλειάδας με τις ενημερωμένες τιμές.

Peer	Local Store	Generated Triples	Rule
p_1	$(\alpha, \texttt{sp}, \beta)$	(u, β, v)	*rdfs7*
p_2	(u, α, v)		
p_3	(u, γ, v)		
p_4	$(\gamma, \texttt{sp}, \beta)$	(u, β, v)	*rdfs7*

Πίνακας 5.9: Παράδειγμα συνεπαγωγής για τον συμπερασματικό κανόνα *rdfs7*.

Algorithm 10: deduceRDFS11/9/EXT1/2: για είσοδο μίας RDF τριπλέτας της μορφής $(u, \texttt{sc}, v)$, ανακτά συσχετιζόμενες τριπλέτες σύμφωνα με τον συμπερασματικό κανόνα *rdfs11* κι εξάγει νέα πληροφορία προς αποθήκευση.

```
1  Λ = p.getTriplesThatSubsume (v);
2  p.rangeRDFquery (?X, type, u);
3  p.rangeRDFquery (?A, domain, u);
4  p.rangeRDFquery (?B, range, u);
5  forall the (s, p, o) in Λ do
6      /* Associated rule: RDFS11 */;
7      p.insert (u, sc, o);
8      /* Associated rule: RDFS9 */;
9      forall the x in X do
10         p.insert (x, type, o);
11     /* Associated rule: EXT1 */;
12     forall the α in A do
13         p.insert (α, domain, o);
14     /* Associated rule: EXT2 */;
15     forall the β in B do
16         p.insert (β, range, o);
```

Algorithm 11: deduceRDFS9b: Αλλαγές που πυροδοτούνται από την εισαγωγή μίας πλειάδας της μορφής $(x, \texttt{type}, u)$ εξαιτίας του συμπερασματικού κανόνα *rdfs9*.

```
1  p.rangeRDFquery (u, subClassOf, ?V);
2  forall the v in V do
3      [ℓ, h] = p.getLabel (u, subClassOf, v);
4      Λ = p.rangeLabelQuery (ℓ, h);
5      forall the (s, p, o) in Λ do
6          p.insert (x, type, o);
```

5.4 Μοντέλο Publish/Subscribe για RDF Δεδομένα

Στην ενότητα αυτή παρουσιάζουμε μία κατανεμημένη υπηρεσία publish-subscribe που βασίζεται στο περιεχόμενο για τη διαχείριση των συνδρομών και των μηνυμάτων. Μία τέτοια υπηρεσία απευθύνεται στις ανάγκες των συμμετεχόντων σε μία κοινότητα που τη χρησιμοποιεί προκειμένου να ενημερώνονται απευθείας σχετικά με νεο-δημοσιευμένο περιεχόμενο. Με άλλα λόγια δίνεται η δυνατότητα έκδοσης συνεχόμενων ερωτημάτων (continuous queries)

Algorithm 12: deleteRDFS7: Αλλαγές που πυροδοτούνται από την εισαγωγή μίας πλειάδας της μορφής $(\alpha, \texttt{subProperty}, \beta)$ εξαιτίας του συμπερασματικού κανόνα *rdfs7*.

```
1 Λ = p.getTriplesThatSubsume (β);
2 forall the (s,p,o) in Λ do
3 |   /* Associated rule: RDFS7 */;
4 |   p.rangeRDFquery (?Γ, subPropertyOf, β);
5 |   p.rangeRDFquery (?U, α, ?V);
6 |   forall the (u,v) in (U,V) do
7 |   |   if ∄γ ∈ Γ, γ ≠ α : (u,γ,v) ∈ KB then
8 |   |   |   p.delete (u,o,v);
```

για συγκεκριμένα θέματα. Το γεγονός ότι η προσέγγιση που υιοθετούμε βασίζεται στο περιεχόμενο παρέχει την ευελιξία στον χρήστη να αποτυπώσει τα ενδιαφέροντά του διατυπώνοντας σύνθετα ερωτήματα που να τα περιγράφουν, π.χ. περιορισμούς για το κατηγόρημα ή το αντικείμενο, σύζευξη, διάζευξη, κ.τ.λ., κι έτσι μόνο τα δημοσιευμένα RDF μηνύματα που τηρούνε τις προδιαγραφές των συνδρομητών καταφτάνουν στους ενδιαφερόμενους.

Κυρίως η αποτελεσματική διαχείριση των συνδρομών καθιστά το MIDAS-RDF ένα στοιβαρό middleware για την μεγάλης κλίμακας διασπορά δεδομένων γεγονότων προς απομακρυσμένους συνδρομητές που έχουν ρητά διατυπώσει το ενδιαφέρον τους. Έτσι έχουμε ότι η περιοχή ενδιαφέροντος ενός συνδρομητή περιγράφεται από ένα πολυδιάστατο διάστημα που δίνεται από ένα κάτω κι ένα άνω φράγμα για κάθε διάσταση του χώρου προκειμένου να ταυτοποιήσει τις σχετικές πλειάδες που απαρτίζουν μία θεματική περιοχή. Συμβατικά τις μεταβλητές μίας τέτοιας συνδρομής, π.χ. τη μοντελοποίηση συνδρομών για γεγονότα δεδομένου κατηγορήματος κι αντικειμένου ανεξαρτήτως υποκειμένου, όπως τα άρθρα που δημοσιεύονται ανά διαστήματα σε ένα συγκεκριμένο επιστημονικό περιοδικό, δηλώνονται από ένα μη φραγμένο πεδίο ορισμού που αφορά ως προς το συγκεκριμένο πεδίο όλες τις πλειάδες που συνηγορούν με το ερώτημα ως προς τις καλά από τον χρήστη ορισμένες διαστάσεις. Για την μοντελοποίηση πολλαπλών περιορισμών πάνω σε μία συγκεκριμένη διάσταση χρησιμοποιούμε τις προαναφερθείσες τεχνικές διάζευξης και σύζευξης διατυπώνοντας ανάλογα υπο-συνδρομές που συνολικά μπορούν να περιγράψουν μία σύνθετη έκφραση. Δίνουμε ιδιαίτερη έμφαση στο γεγονός ότι μία περιοχή ενδιαφέροντος μπορεί να επικαλύπτει τις ζώνες πολλών ομότιμων κόμβων χωρίς να περιορίζεται μόνο σε έναν, ή το αντίστροφο, δηλαδή ο κάθε κόμβος να αναλαμβάνει μία συνδρομή. Ως επί το πλείστον, ένας συνδρομητής ούτε χρειάζεται ούτε είναι σε θέση να γνωρίζει εκ των προτέρων τους κόμβους που είναι υπεύθυνοι για τις περιοχές ενδιαφέροντός του.

Επιπροσθέτως, κάθε κόμβος υποχρεούται να διατηρεί ένα τοπικό πολυδιάστατο ευρετήριο στο οποίο είναι αποθηκευμένες όλες οι συνδρομές που τον αφορούν, ή υπο-συνδρομές για τις περιπτώσεις όπου μία συνδρομή εκτείνεται πέραν της ζώνης ενός κόμβου. Δηλαδή καθώς προωθείται μετά την έκδοσή της μία συνδρομή και προωθείται αναδρομικά σε ολοένα και πιο σχετικούς κόμβους, διαμερίζεται σε επιμέρους υπο-συνδρομές που ορίζονται από τις εκάστοτε τομές των ζωνών των κόμβων που σχετίζονται με τη συνολική περιοχή ενδιαφέροντος που ορίζει μία συνδρομή. Με αυτόν τον τρόπο ο κάθε κόμβος διατηρεί μία ιεραρχική δομή με όλα τα τμήματα των συνδρομών που αφορούν αποκλειστικά τη δική του ζώνη. Στηριζόμενος σε αυτή τη δομή ο κάθε κόμβος όταν λαμβάνει ένα γεγονός δεδομενων ανακτά τοπικά όλες τις συνδρομές που το επικαλύπτουν κι ενημερώνει τους συνδρομητές σχετικά. Κατά συνέπεια η υπηρεσία αυτή χρησιμοποιεί κάποιες από τις λειτουργίες που ήδη έχουμε περιγράψει προηγουμένως και τις επιβαρύνει με μόλις ένα επιπλέον βήμα στη διαδικασία εισαγωγής/ενημέρωσης πλειάδων. Ακόμα, πολλαπλές συνδρομές που αφορούν το ίδιο γεγονός θα προκαλέσουν μία και μόνο ενημέρωση για κάθε συνδρομητή αφου εντοπίζονται εύκολα στην τοπική ιεραρχία.

Ένας διαφορετικός κι όχι τόσο αποδοτικός τρόπος υλοποίησης μίας τέτοιας υπηρεσίας που συναντάται συχνά, απαιτεί από έναν κόμβο/συνδρομητή να εκδίδει συνεχώς ερωτήματα για τις περιοχές ενδιαφέροντός του και στη συνέχεια να τις επεξεργάζεται κατάλληλα προ-

κειμένου να διαπιστώσει εάν και τι είδους αλλαγές έχουν προκληθεί. Προφανώς μία τέτοια προσέγγιση συνοδεύεται από ένα μη αμελητέο επιπλέον επικοινωνιακό κόστος.

5.5 Πειραματική Αποτίμηση

Η Ενότητα 5.5.1 περιγράφει τη μεθοδολογία που ακολουθήσαμε στην πειραματική αποτίμηση των μεθόδων του MIDAS-RDF για την επεξεργασία ατομικών RDF ερωτημάτων μοτίβου (atomic triple pattern queries) αλλά και για την αποτελεσματικότητα του υπολογισμού της μεταβατικής κλειστότητας ενός μέρος του RDF γραφου. Στην Ενότητα 5.5.2 παρουσιάζουμε τα αποτελέσματα.

Συχνότητα	Κατηγορήματα
900440	publication-has-author (author)
438531	contained in proceedings (isIncludedIn)
112303	cites publication
10639	has-homepage (foaf:homepage)
10461	has-publisher (dc:publisher)
7308	has affiliation (foaf:workplaceHomepage)
5850	in series

Πίνακας 5.10: Στατιστικά μεγέθη για τα δεδομένα του DBLP (Resource-to-Resource Triples: 3740438, Resource-to-Literal Triples: 7274180).

5.5.1 Μεθοδολογία

Τοπολογία Δικτύου

Τα πειράματά μας προσομειώνουν ένα δυναμικό, κατανεμημένο περιβάλλον. Η διαδικασία που ακολουθήσαμε για την αξιολόγηση των χαρακτηριστικών των μεθόδων μας αποτελείται από δύο φάσεις. Στην πρώτη φάση αυξάνουμε διαδοχικά το μέγεθος του δικτύου μελετώντας ταυτόχρονα τη συμπεριφορά των μετρικών που έχουμε ορίσει, ενώ η δεύτερη φάση ξεκινάει αφού έχει φτάσει το δίκτυο στο μέγιστο επιτρεπτό μέγεθος και προκαλεί τη σταδιακή αποχώρηση τυχαίων κόμβων έως ότου το δίκτυο φτάσει και πάλι στο αρχικό του μέγεθος. Η κάθε προσομείωση αρχικοποιεί ένα δίκτυο 1000 κόμβων το οποίο αυξάνεται έως να φτάσει τους 100000 κόμβους, διαδικασία η οποία ακολουθείται από την ακριβώς αντίστροφη πορεία έως ότου το δίκτυο φτάσει και παλι τους 1000 κόμβους της αρχικής του κατάστασης.

Μετρικές

Αποτιμούμε την απόδοση των μεθόδων βάσει των παρακάτω μετρικών. Πρώτον, από τον αριθμό των επαναπροωθήσεων ή βημάτων (latency) που απαιτούνται προκειμένου να επεξεργαστούμε ένα ερώτημα αφού το μέγεθος αυτό καθορίζει τη γρήγορη ή μη απόκριση του MIDAS-RDF από την πλευρά του χρήστη. Δεύτερον, από τη συμφόρηση (congestion) στο δικτύου που ορίζεται ως ο μέσος αριθμός ερωτημάτων που επεξεργάζεται ο κάθε κόμβος όταν n ερωτήματα εκδίδονται από τυχαία επιλεγμένους κόμβους. Το μέγεθος αυτό αντικατοπτρίζει την μέση κίνηση που δέχεται ένας κόμβος. Τρίτον, προκειμένου να ποσοτικοποιήσουμε επακριβώς το πόσο γρήγορα απαντάται ένα ερώτημα ως προς κάθε του στάδιο (ποσοστό σχηματισμού της τελικής απάντησης επί τοις εκατώ) θα δείξουμε δύο ακόμα μετρικές. Η ανάκληση (recall) ορίζεται ως η αναλογία μεταξύ του αριθμού των σχετικών κόμβων που προσπελάσονται σε κάθε κύκλο επεξεργασίας ως προς το συνολικό αριθμό των σχετικών κόμβων. Η αποκρισημότητα (responsiveness) ορίζεται ως η αναλογία μεταξύ του αριθμού των ανακτηθέντων πλειάδων του αποτελέσματος σε κάθε βήμα επεξεργασίας του ερωτήματος ως προς το συνολικό μέγεθος του αποτελέσματος.

Συχνότητα	Οντότητες
560792	Πρόσωπα (foaf:Person)
561895	Άρθρα σε πρακτικά
340488	Άρθρα επιστημονικών περιοδικών
10610	Ιστοσελίδες ατόμων
9027	Πρακτικά
2530	Κεφάλαια βιβλίων

Πίνακας 5.11: Στατιστικά μεγέθη για τα δεδομένα του DBLP (Resources: 2395467, Literals: 3064704).

Δεδομένα κι Ερωτήματα

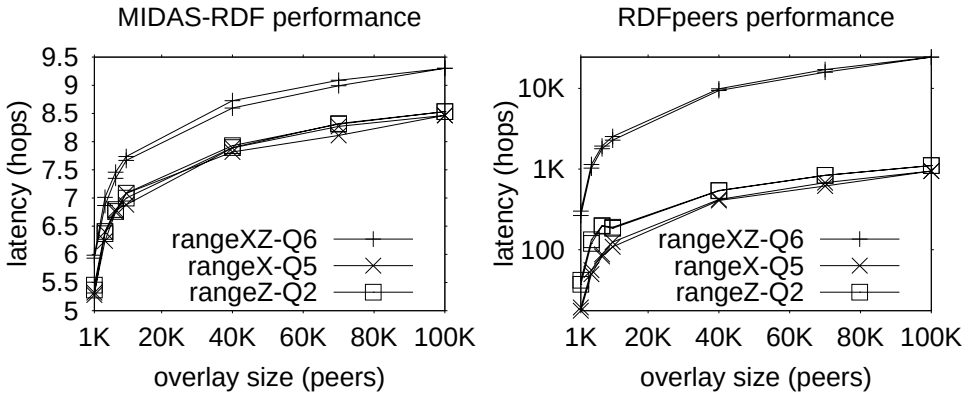
Προκειμένου να μελετήσουμε τα χαρακτηριστικά των μεθόδων υπολογισμού της μεταβατικής κλειστότητας (transitivity computation) και πως αυτές κλιμακώνονται, χρησιμοποιήσαμε συνθετικούς γράφους παραμετροποιήσιμου βάθους (έως 16) που αποτελούνται από 1000000 εγγραφές έκαστος. Στο σενάριο αυτό συγκρίνουμε συμβατικές τεχνικές επεξεργασίας με το MIDAS-RDF που έχει ενσωματώσει έναν αποδοτικό μηχανισμό επισήμανσης προκειμένου να μπορεί γρήγορα να ανακτά τα στοιχεία του RDFS γράφου τα οποία σχετίζονται με μεταβατικές ιδιότητες. Σημειώνουμε ότι ο αποδοτικός υπολογισμός τέτοιων σχέσεων είναι ιδιαίτερα σημαντικός επειδή εμφανίζεται στη συντριπτική πλειοψηφία των RDFS κανόνων συνεπαγωγής του W3C [5].

Επιπλέον αποτιμούμε τις βασικές λειτουργίες που απαρτίζουν τις μεθόδους επεξεργασίας στο MIDAS-RDF πάνω σε πραγματικά δεδομένα. Ειδικότερα, χρησιμοποιήσαμε τη βάση γνώσης του DBLP [1] η οποία αποτελείται από περίπου 11,2 εκατομύρια εγγραφές κι είναι διαθέσιμη σε XML μορφή. Η συγκεκριμένη βάση γνώσης εμπεριέχει έναν πολύ μεγάλο αριθμό από βιβλιογραφικές αναφορές και μεταδεδομένα σχετικά τα πρακτικά της πλειοψηφίας των επιστημονικών περιοδικών και συνεδρίων που αφορούν την επιστήμη υπολογιστών έως και το 2006. Για παράδειγμα, παρέχονται στοιχεία για πάνω από μισό εκατομύριο άρθρα και πολλές χιλιάδες σύνδεσμοι προς ιστοσελίδες των συγγραφέων τους μαζί με πολλές άλλες πληροφορίες φυσικά.

Οι Πίνακες 5.10 και 5.11 παρουσιάζουν στατιστικά στοιχεία της γνώσης βάσης του DBLP. Τα χαρακτηριστικά της βάσης αυτής που αποτελείται από πραγματικά δεδομένα επηρεάζουν σε σημαντικό βαθμό την απόδοση των μεθόδων που εξετάζουμε. Για παράδειγμα, υπάρχουν 2425830 δηλώσεις που χρησιμοποιούν το κατηγόρημα `<rdf:type>`, 1708988 εγγραφές με την ιδιότητα `<http://xmlns.com/foaf/0.1/name>`, 1689330 για τη δήλωση της ιδιότητας `<dc:creator>`, κ.ο.κ. Ενδεικτικά αναφέρουμε ότι μόλις εφτά ιδιότητες παρουσιάζονται στο εξωπραγματικό ποσοστό του 72% των τριπλετών. Δεδομένου αυτών των ακραίων ανισορροπιών που παρουσιάζονται στα πραγματικά δεδομένα κρίνουμε ότι οι συμβατικές μέθοδοι που έχουν υιοθετηθεί μέχρι στιγμής για την κατανεμημένη επεξεργασία τέτοιου είδους ερωτημάτων είναι επιεικώς ανεπαρκής αφού στηρίζονται σε πολύ απλουστευτικές προσεγγίσεις, όπως την αντιστοίχιση κάθε εγγραφής σε έως τρεις διαφορετικούς κόμβους του δικτύου βάσει του υποκειμένου, του κατηγορήματος και του αντικειμένου. Αλλάξαμε ελαφρώς το σχεδιασμό στο RDFPeers [35] που προηγουμένως επιφορτιζόταν με το τριπλάσιο φορτίο δεδομένων για κάθε τριπλέτα που του εισάγαμε. Διαφορετικά δεν είναι καθόλου απίθανο να καταλήγαμε στα πειράματά μας με έστω έναν κόμβο στο να είναι υπεύθυνος RDFPeers για τα μισά περίπου δεδομένα, μία σημαντική σχεδιαστική αδυναμία στην οποία έπρεπε να είχε δοθεί αρκετή έμφαση.

Η αποτίμηση στηρίζεται στους τύπους ερωτημάτων τα οποία συναντάμε συχνότερα στις διαδικασίες συμπερασμού. Ειδικότερα συμβολίζουμε με *rangeXZ* το σύνολο των ερωτημάτων της μορφής `SELECT ?publication ?author WHERE {?publication <dc:creator> ?person}`, όπου μόνο το κατηγόρημα παραμένει σταθερό, όπως είδαμε στον Αλγόριθμο 8. Με *rangeZ* συμβολίζουμε το σύνολο των ερωτημάτων της μορφής `SELECT ?author WHERE {<http://www.w3.org/TR/xquery> <ex:editor> ?author}` όπου ανακτόνται οι τριπλέτες για σταθερό υποκείμενο και κατηγόρημα, όπως στον Αλγόριθμο 7. Τέλος, με *rangeX*

συμβολίζουμε το σύνολο των ερωτημάτων της μορφής `SELECT ?id WHERE {?id date "2002-01-03"}` για την ανάκτηση όλων των υποκειμένων που εμφανίζονται στη βάση γνώσης για συγκεκριμένο κατηγόρημα κι αντικείμενο, όπως δηλαδή απαιτείται στον Αλγόριθμο 10. Τα ερωτήματα δημιουργήθηκαν με τυχαίο τρόπο και βασίστηκαν στη βάση γνώσης του DBLP. Η επιλεκτικότητα (selectivity) δεν είναι σταθερή κι επηρεάζεται από τα χαρακτηριστικά των δεδομένων και μόνον. Τέλος, στα Σχήματα 4.2 και 5.3 δείχνουμε τη μέση τιμή για κάθε μετρική για όλες τις διαφορετικές τοπολογίες δικτύου που δοκιμάζουμε.

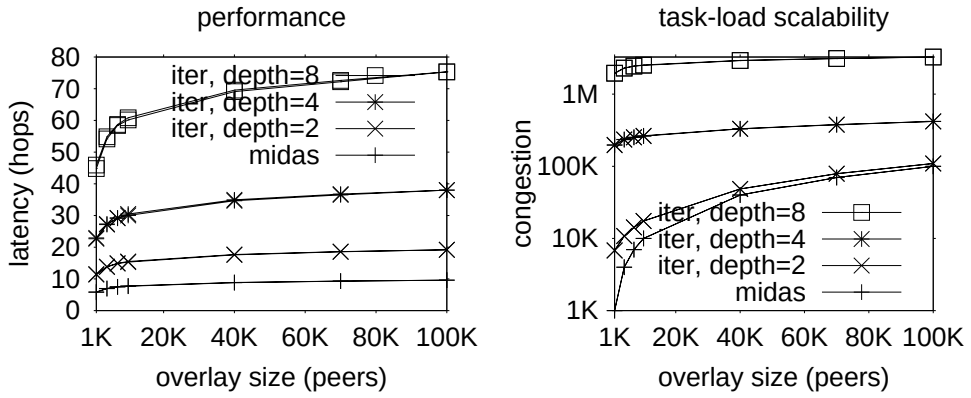


Σχήμα 5.2: Αριθμός επαναπροωθήσεων για MIDAS-RDF και RDFPeers στα ερωτήματα *rangeXZ*, *rangeX*, *rangeZ* για δεδομένα του DBLP.

5.5.2 Αποτελέσματα

Για το υπόλοιπο μέρος του κεφαλαίου παρουσιάζουμε τα αποτελέσματα της πειραματικής αποτιμησης τα οποία μάλιστα επιβεβαιώνουν τη θεωρητική ανάλυση από προηγούμενες ενότητες. Στο Σχήμα 5.2 παρουσιάζουμε τη συμπεριφορά μετρικών χρήσιμων για την αξιολόγηση των μεθόδων επεξεργασίας του MIDAS-RDF για τις κλάσεις ερωτημάτων *rangeX*, *rangeZ* και *rangeXZ*. Ο μέγιστος αριθμός επαναπροωθήσεων (latency) φράσσεται από το $O(\log n)$ στο MIDAS-RDF, όπως βλέπουμε και στο γράφημα. Το RDFPeers όπως αυτό προτάθηκε στην αρχική του μορφή είναι εξίσου γρήγορο με τη δική μας προσέγγιση. Παρ' όλα αυτά εμφανίζονται τεράστιες ανισοκατανομές ως προς τα δεδομένα αλλά κι ως τη συμφόρηση του δικτύου. Εν αντιθέσει, το MIDAS-RDF που βασίζεται σε ένα εγγενώς πολυδιάστατο ευρετήριο αποτρέπει τέτοιου είδους φαινόμενα καθώς λαμβάνει υπόψην την κατανομή του φορτίου κατά το σχηματικό της δομής του δικτύου. Όσον αφορά την ανασχεδιασμένη έκδοση του RDFPeers που στηρίζεται στο MAAN προκειμένου να μειωθεί το επιπλέον αποθηκευτικό κόστος, αυτό μειονεκτεί ως προς την ταχύτητα επεξεργασίας των ερωτημάτων. Για την ακρίβεια, ο αριθμός των απαιτούμενων επαναπροωθήσεων επιδεικνύει γραμμική συμπεριφορά καθώς αυξάνεται το μέγεθος του δικτύου όταν για το MIDAS-RDF η ίδια μετρική παρουσιάζει λογαριθμική κλιμάκωση. Ειδικότερα, το latency στο MAAN κυριαρχείται από τον αριθμό των σχετικών κόμβων λόγω της σειριακής προσπέλασης των σχετικών κόμβων μετά τον εντοπισμό του κόμβου για το άνω ή κάτω φράγμα του ερωτήματος. Αυτό γίνεται επίσης εμφανές για όλους τους τύπους αναζήτησης που εκτελούνται καθώς η απόδοση φθίνει καθώς η επιλεκτικότητα των ερωτημάτων για τα πραγματικά δεδομένα αυξάνεται από τα ερωτήματα του *rangeX* στο *rangeXZ*.

Το Σχήμα 5.3 δείχνουμε την απόδοση των μεθόδων υπολογισμού της μεταβατικής κλειστότητας συνθετικών γράφων μεταβλητού βάθους. Συγκρίνουμε το MIDAS-RDF που όπως φαίνεται αξιοποιεί σε μεγάλο βαθμό τις τεχνικές επισήμανσης που προσαρμόζει με συμβατικές τεχνικές που συναντάμε συχνά στη βιβλιογραφία. Πιο συγκεκριμένα, η υιοθέτηση των τεχνικών επισήμανσης βελτιώνει την απόδοση σε πολύ μεγάλο βαθμό καθώς καθιστά αδιάφορο την παράμετρο της διαμέτρου του γραφήματος. Όσον αφορά τη συμφόρηση του δικτύου,



Σχήμα 5.3: Αριθμός επαναπροωθήσεων και συμφόρηση δικτύου για τον υπολογισμό της μεταβατικής κλειστότητας συνθετικών γράφων μεταβλητού βάθους με τη χρήση τεχνικών επισήμανσης και χωρίς.

το MIDAS-RDF υπερτερεί και πάλι. Προφανώς, οι τεχνικές επισήμανσης είναι εξαιρετικά αποδοτικές για μεγάλους γράφους που παρουσιάζουν ιεραρχική δομή.

5.6 Συζήτηση

Στο κεφάλαιο αυτό παρουσιάσαμε το MIDAS-RDF, ένα κατανεμημένο αποθετήριο για RDF δεδομένα το οποία στηρίζεται εξολοκλήρου στο κατανεμημένο πολυδιάστατο ευρετήριο την αρχιτεκτονική του οποίου παρουσιάσαμε στο προηγούμενο κεφάλαιο. Εξετάσαμε τεχνικές για την επεξεργασία διαφόρων τύπων ερωτημάτων σε λογαριθμικό αριθμό βημάτων ως προς το μέγεθος του δικτύου. Επιπλέον, μελετάμε τον συνδυασμό αυτών τόσο σε συζεύξεις, όσο και σε διαζεύξεις. Ακόμα, με την ενσωμάτωση τεχνικών επισήμανσης αλλά και την αξιοποίηση τους από επαγωγικά μοντέλα συμπερασμού, όπως το forward-chaining μοντέλο που παρουσιάσαμε με το MIDAS-RDF, δείχνουμε ότι η λύση που προτείνουμε υπερτερεί ως προς τις παραδοσιακές μεθόδους που υπάρχουν στη βιβλιογραφία για το συγκεκριμένο πρόβλημα, οι οποίες βασίζονται σε διαδοχικά επαναληπτικά βήματα ο αριθμός των οποίων είναι ευθέως ανάλογος της διαμέτρου του RDF γράφου που έχει αποθηκευτεί στο MIDAS-RDF. Τέλος, η αξιοποίηση της υποδομής αυτής βρίσκει χρήσιμη εφαρμογή σε ένα publish-subscribe μοντέλο που δίνει τη δυνατότητα στους χρήστες να εγγράφονται ως συνδρομητές στα νεο-εισερχόμενα δεδομένα που ικανοποιούν συγκεκριμένες προδιαγραφές ως προς το περιεχόμενο.

Κεφάλαιο 6

Επεξεργασία Ερωτημάτων Κατάταξης με το RIPPLE

The important thing in science
is not so much to obtain new facts,
as to discover new ways of thinking about them.

Sir William Bragg

6.1 Απαιτήσεις και Χαρακτηριστικά του RIPPLE

Η Ενότητα 6 περιγράφει το RIPPLE framework [112] για την κατανεμημένη επεξεργασία ερωτημάτων κατάταξης (rank queries) ενώ η Ενότητα 6.1.2 εστιάζει στην ανάλυση κι εφαρμογή του RIPPLE πάνω στο MIDAS με κάποιες βελτιστοποιήσεις.

6.1.1 Αρχές Σχεδιασμού του RIPPLE

Το RIPPLE framework στοχεύει στην κατανεμημένη επεξεργασία ερωτημάτων κατάταξης (rank queries). Η εφαρμογή του RIPPLE αφορά οποιοδήποτε κατανεμημένο πίνακα κατακερματισμού (distributed hash table). Για κάθε κόμβο w του δικτύου, θεωρούμε με w.link το σύνολο των κόμβων τους οποίους γνωρίζει και στους οποίους προωθεί τα μηνύματα που αφορούν την επεξεργασία ερωτημάτων κι άλλες λειτουργίες του δικτύου. Το μέγεθος του πίνακα αυτού το σημειώνουμε με το $|w.\text{link}|$ ενώ ο μέγιστος αριθμός από γνωστούς κόμβους ισούται με Δ.

Μια πολύ σημαντική έννοια στο RIPPLE είναι αυτή της *περιοχής* (*region*). Στο RIPPLE ο κάθε σύνδεσμος του w συσχετίζεται με μία περιοχή, την οποία γράφουμε ως $w.\text{link}[i].\text{region}$. Οι περιοχές όλων των κόμβων που γνωρίζει ο w αποτελούν μία διαμέριση ολόκληρου του χώρου. Υπάρχει σημαντική διαφορά μεταξύ της περιοχής (region) ενός κόμβου και της ζώνης (zone) του. Υπενθυμίζουμε ότι σε έναν κόμβο ανατίθεται ένα τμήμα ολόκληρου του χώρου το οποίο ονομάζεται ζώνη (zone) κι ο κόμβος υπεύθυνος για αυτήν αποθηκεύει κι απαντάει ερωτήματα σχετικά με όλες τις πλειάδες που αντιστοιχούν στη ζώνη του. Από την άλλη, ειδικα για το RIPPLE, η περιοχή (region) ενός κόμβου αντιστοιχεί σε έναν μεγαλύτερο χώρο ο οποίος συμπεριλαμβάνει και τη ζώνη του κόμβου. Πιο συγκεκριμένα, η περιοχή ενός κόμβου εξαρτάται από την οπτική των κόμβων που τον έχουν συμπεριλάβει στους πίνακες δρομολόγησής τους (routing tables) κι άρα μπορεί για δύο διαφορετικούς κόμβους ο ίδιος σύνδεσμος να αντιστοιχεί σε διαφορετικές περιοχές. Για παράδειγμα, ας εξετάσουμε τους κόμβους w και v, όπου αμφότεροι είναι συνδεδεμένοι με τον κόμβο x, αλλά ο συγκεκριμένος σύνδεσμος εκπροσωπεί διαφορετικές περιοχές για τους w και v, αν κι εμπεριέχουν τη ζώνη του x. Εν ολίγοις, οι δύο διαφορετικές έννοιες μπορούν να συνοψιστούν ως εξής: η *περιοχή* είναι μια

γενικότερη έννοια για την οποία ένας κόμβος είναι υπεύθυνος για την *προώθηση* όλων των μηνυμάτων που αφορούν την περιοχή του αλλά *απαντάει*, *αποθηκεύει* κι *επεξεργάζεται τοπικά* μόνο αιτήσεις που αφορούν τη δική του *ζώνη*.

Ανάλογα με την δικτυακή υποδομή (DHT) πάνω στην οποία λειτουργεί το RIPPLE, υπάρχει συχνά ένας φυσικός τρόπος προκειμένου να ανατεθούν περιοχές στους κόμβους με τους οποίους είναι συνδεδεμένος ένας κόμβος w. Μία περιοχή πρέπει να ικανοποιεί δύο προαπαιτούμενα: (i) θα πρέπει να υπερκαλύπτει τη ζώνη του κόμβου με τον οποίον συσχετίζεται, (ii) η ένωση όλων των περιοχών των συνδεδεμένων κόμβων σχηματίζει ολόκληρο τον αρχικό χώρο του ευρετηρίου.

Ας δούμε όμως πως ακριβώς ορίζονται οι περιοχές για τρεις από τους πιο επιφανής κατανεμημένους πίνακες κατακερματισμού που αποτελούν κι εκπροσώπους των μεγαλύτερων κατηγοριών. Οι ορισμοί για τα υπόλοιπα DHTs είναι ευθέως ανάλογοι. Στο CAN, κάθε κόμβος w διαθέτει τουλάχιστον δύο συνδέσμους για κάθε διάσταση. Πιο συγκεκριμένα, για τις d διαστάσεις, δύο κόμβοι είναι γείτονες εάν οι ζώνες τους αλληλοεπικαλύπτονται κατά $d-1$ διαστάσεις και εφάπτονται ή συνορεύουν κατά μία. Οπότε, ο κάτω (αντίστοιχα άνω) γείτονας κατά την i-οστή διάσταση αντιπροσωπεύει μία περιοχή που προσομοιάζει το κομμάτι μίας πυραμίδας (ένα τραπέζιο για τις δύο διαστάσεις ή για περισσότερες διαστάσεις μία πυραμίδα της οποίας η κορυφή έχει αποκοπεί) που έχει ως βάση την κάτω (αντίστοιχα άνω) συνοριακή επιφάνεια προς τον υπόλοιπο χώρο που είναι κάθετη στη i-ιοστή διάσταση. Οπότε ο κόμβος προωθεί μία αίτηση που λαμβάνει (ή εκδίδει) προς τους κόμβους των οποίων οι περιοχές (όχι οι ζώνες) επικαλύπτουν το ερώτημα.

Στο Chord κάθε κόμβος w γειτονεύει με τους κόμβους των οποίων οι ζώνες έχουν απόστασεις που αυξάνονται εκθετικά από τον w. Τότε, για τον κόμβο w, η περιοχή του i-οστού κόμβου που ξέρει εκτείνεται από την αρχή της ζώνης του και τελειώνει στην αρχή της ζώνης του επόμενου $(i+1)$ γείτονα (ή της ζώνης του ίδιου του w αν πρόκειται για τον τελευταίο σύνδεσμο στον πίνακα δρομολόγησης).

Κάθε κόμβος w του MIDAS έχει έναν γείτονα εντός κάθε συμμετρικού υποδέντρου του οποίου η ρίζα βρίσκεται σε βάθος μέχρι w.depth. Τότε, για τον w η περιοχή που αντιστοιχεί στον i-οστό γείτονα ορίζεται από ολόκληρο το χώρο που δεικτοδοτεί το συγκεκριμένο υποδέντρο βάθους i.

Όσον αφορά την επεξεργασία ερωτημάτων, συμβολίζουμε με Q ένα γενικευμένο ερώτημα. Ορίζουμε ως A το *τοπικό αποτέλεσμα*, π.χ., τις τοπικές πλειάδες που ικανοποιούν το ερώτημα. Η επεξεργασία αρχίζει από τον κόμβο που εξέδωσε το ερώτημα, στον οποίον θα αναφερόμαστε στο εξής ως v. Κάθε κόμβος, συμπεριλαμβανομένου και του v, ο οποίος εμπλέκεται στην επεξεργασία, εκτελεί το ίδιο πρωτόκολλο κι επιστρέφει το δικό του μέρος της απάντησης που απαρτίζεται από τοπικά αποθηκευμένες πλειάδες που ικανοποιούν τους περιορισμούς. Ανάλογα με το ερώτημα, ο κόμβος v θα πρέπει να εκτελέσει επιπλέον λειτουργίες (π.χ. φιλτράρισμα) έως ότου εξαχθεί το τελικό αποτέλεσμα.

Μία ακόμα σημαντική έννοια στο RIPPLE είναι αυτή της *κατάστασης*, που υποδηλώνεται ως S κι αποτελεί μία (μερική) οπτική της προόδου της κατανεμημένης επεξεργασίας του ερωτήματος. Για παράδειγμα, δεδομένου του ερωτήματος και του κατανεμημένου αλγορίθμου, το S θα μπορούσε να είναι ένα σύνολο από τοπικές ή μη πλειάδες ή/και όρια/εγγυήσεις για την ποιότητα του αποτελέσματος. Ξεχωρίζουμε λοιπόν δύο ειδών καταστάσεις: (i) την *τοπική κατάσταση* στον κόμβο w, που σημειώνουμε με S_w^L, κι η οποία εμπεριέχει πληροφορία προσβάσιμη μόνο από τον w, τόσο από τοπικές πλειάδες αλλά και από τις καταστάσεις άλλων κόμβων τις οποίες έχει ζητήσει, (ii) την *ολική κατάσταση* στον κόμβο w που δείχνουμε με S_w^G και περικλύει την τοπική κατάσταση S_w^L ενώ περιλαμβάνει επιπλέον πληροφορία η οποία προωθήθηκε στον w μαζί με το ερώτημα.

Η κύρια ιδέα στην οποία βασίζεται το RIPPLE είναι να συνδυαστούν οι περιοχές με τις καταστάσεις των ερωτημάτων με αποδοτικό τρόπο ώστε να αποκτηθεί γνώση σχετικά με την πρόοδο της επεξεργασίας του ερωτήματος κι έτσι να δρομολογηθεί αποτελεσματικά το ερώτημα προς τους σχετικούς κόμβους σε χαμηλό χρόνο εκτέλεσης και συνολικό αριθμό μηνυμάτων. Πρώτα θα εξετάσουμε το RIPPLE για δύο διαφορετικά σενάρια και στη συνέχεια θα παρουσιάσουμε έναν πιο γενικευμένο παραμετροποιήσιμο αλγόριθμο που δύναται να συνδυάσει τις διαφορετικές στρατηγικές.

Ο πρώτος αλγόριθμος ονομάζεται fast και αποσκοπεί στο να βελτιστοποιήσει τον χρόνο εκτέλεσης. Ο Αλγόριθμος 13 δείχνει το πρωτόκολλο που εκτελεί ο κάθε κόμβος στα πλαίσια της επεξεργασίας ερωτημάτων κατάταξης. Ένας κόμβος w λαμβάνει το ερώτημα Q, την ολική κατάσταση S^G, τα στοιχεία του κόμβου v που εξέδωσε το ερώτημα και την σχετική περιοχή R εντός της οποίας η επεξεργασία του ερωτήματος περιορίζεται. Με τον ορισμό της περιοχής αυτής μπορούμε να εγγυηθούμε ότι κανένας κόμβος δε θα λάβει πάνω από μία φορά το μήνυμα. Δίνουμε έμφαση στο γεγονός ότι όλοι οι αλγόριθμοι σε αυτό το κεφάλαιο αποτελούν *πρότυπα* (*templates*) ή *διεπαφές* (*interfaces*) και εμπεριέχουν κλήσεις σε γενικευμένες συναρτήσεις των οποίων τα στιγμιότυπα (instances) εξαρτώνται από τον τύπο του ερωτήματος και περιγράφονται εκτενώς στις αμέσως επόμενες ενότητες.

Algorithm 13: w.fast(v, Q, S^G, R): Καλείται από τον κόμβο v για την επεξεργασία του ερωτήματος Q εντός της περιοχής R με ολική κατάσταση S^G.

```
1 S_w^L ← w.computeLocalState(Q, S^G) ;
2 S_w^G ← w.computeGlobalState(Q, S^G, S_w^L) ;
3 for each link i do
4 |   if w.isLinkRelevant(i, Q, S_w^G, R) then
5 |   |   w.link[i].fast(v, Q, S_w^G, w.link[i].region∩R) ;
6 A ← w.computeLocalAnswer(Q, S_w^L) ;
7 w.sendLocalAnswerTo(v, A) ;
```

Ο κόμβος w υπολογίζει την τοπική του κατάσταση καλώντας τη συνάρτηση computeLocalState στηριζόμενος στην ολική κατάσταση S^G και τις εγγραφές που είναι αποθηκευμένες τοπικά. Επίσης, επαναπροσδιορίζει την ολική κατάσταση με την computeGlobalState (γραμμή 2). Έπειτα, ο κόμβος w εξετάζει έναν προς έναν τους κόμβους που γνωρίζει (γραμμές 3–5) και καλεί την isLinkRelevant (γραμμή 4) για να ελέγξει εάν η περιοχή του i-οστού γείτονα (i) επικαλύπτει την περιοχή R που αφορά το ερώτημα, και (ii) εμπεριέχει εγγραφές που μπορούν να συνεισφέρουν στο αποτέλεσμα δεδομένου της ολικής κατάστασης S_w^G. Εάν ισχύουν τα ανωτέρω για τον i-οστό σύνδεσμο τότε το ερώτημα προωθείται ανάλογα με την ολική κατάσταση και τη νέα περιοχή του ερωτήματος να ορίζεται ως η τομή της R με εκείνη που αντιστοιχεί στον i-οστό κόμβο (γραμμή 5). Αφού έχουν εξεταστεί όλοι οι σύνδεσμοι, ο κόμβος w υπολογίζει το δικό του μέρος της απάντησης στηριζόμενος στη τοπική του κατάσταση καλώντας τη μέθοδο computeLocalAnswer (γραμμή 6), και στέλνει στον εκδότη του αιτήματος μόνο τις τοπικές εγγραφές που ικανοποιούν τις προδιαγραφές του ερωτήματος (γραμμή 7).

Ο Αλγόριθμος 13 αρχικά καλείται χωρίς περιορισμούς ως προς την περιοχή που αφορά (ολόκληρο το πεδίο ορισμου), κι εν συνεχεία επεξεργάζεται το ερώτημα Q, το οποίο υπόκειται στις μεθόδους της διεπαφής. Αν βέβαια παραλείπαμε το δεύτερο έλεγχο της isLinkRelevant (για το εαν ο σύνδεσμος εμπεριέχει σχετικές πλειάδες δεδομένου της τοπικής κατάστασης), τότε όλοι οι κόμβοι του δικτύου θα λάμβαναν ένα αντίγραφο του ερωτήματος ακριβώς μία φορά κι ο μέγιστος αριθμός από βήματα θα ισούταν με τη διάμετρο του δικτύου.

Ο Αλγόριθμος 13 βελτιστοποιεί τον χρόνο απόκρισης και προσπαθεί να μειώσει τον αριθμό μηνυμάτων κατάλληλα. Στη συνέχεια παρουσιάζουμε το RIPPLE όταν είναι προσαρμοσμένο ώστε να ελαχιστοποιεί τον αριθμό των μηνυμάτων. Έτσι έχουμε τον Αλγόριθμο 14, ονόματι slow, που αποτελεί το πρωτόκολλο που εκτελεί ο κάθε κόμβος στα πλαίσια της επεξεργασίας ερωτημάτων κατάταξης με κριτήριο όμως το κόστος επικοινωνίας αυτή τη φορά. Όπως και προηγουμένως, ο αλγόριθμος περιορίζεται στους σχετικούς κόμβους χρησιμοποιώντας το ίδιο μοντέλο παραμετροποίησης. Η διαφορά έγκειται στο γεγονός ότι η επεξεργασία των ερωτημάτων γίνεται σταδιακά κι η τοπική κατάσταση ενημερώνεται σε κάθε βρόχο. Το σκεπτικό είναι ότι το κόστος επικοινωνίας εξαρτάται από την τοπική πληροφορία που προέρχεται από τους γνωστούς κόμβους και την κατανεμημένη επεξεργασία των ερωτημάτων. Οπότε γίνεται προφανές ότι χρειάζεται μία έξυπνη διαχείριση των πόρων ώστε να προσπελαύνονται κυρίως σχετικοί με το ερώτημα πόροι.

Για τον αλγόριθμο slow, ένας κόμβος w λαμβάνει ένα ερώτημα Q, την τωρινή ολική κατάσταση, S^G, τη διεύθυνση του εκδότη του ερωτήματος, τη διεύθυνση του κόμβου u που

προώθησε την αίτηση και τέλος τις ιδιότητες της περιοχής R. Αρχικά, υπολογίζει την τοπική κατάσταση βάσει της ολικής κατάστασης που έλαβε (γραμμή 1), κι έπειτα την νέα ολική κατάσταση (γραμμή 2). Το επόμενο βήμα όμως διαφέρει από τον Αλγόριθμο 13. Ο κόμβος w δίνει υψηλή προτεραιότητα στους κόμβους με μεγάλη πιθανότητα να συνεισφέρουν στο αποτέλεσμα. Η μέθοδος sortLinks ταξινομεί τους συνδέσμους του w με τη χρήση της συνάρτησης comp, η οποία συγκρίνει το δυναμικό δύο κόμβων στη γραμμή 3.

Έπειτα, ο slow εξετάζει τον κάθε κόμβο που γνωρίζει με σειρά προτεραιότητας (γραμμές 3–8). Ας θεωρήσουμε ότι εξετάζουμε τώρα τον ℓ-ιοστό γείτονα. Με τρόπο παρόμοιο με τον fast, ο κόμβος w καλεί την isLinkRelevant (γραμμή 4) προκειμένου να ελέγξει εάν θα πρέπει να του αποστείλει αντίγραφο του ερωτήματος. Εάν ο έλεγχος επιβάλλει κάτι τέτοιο, τότε το ερώτημα προωθείται περαιτέρω μαζί με την τωρινή εικόνα της ολικής κατάστασης και τις ιδιότητες της περιοχής R που αφορά το ερώτημα (γραμμή 5). Σε αντίθεση με τον Αλγόριθμο 13, ο κόμβος w θα περιμένει την απόκριση του κόμβου. Όταν θα λάβει την απόκριση (γραμμή 6) που συμπεριλαμβάνει την τοπική κατάσταση του απομακρυσμένου κόμβου, ο κόμβος w καλεί την updateLocalState προκειμένου να ενημερώσει τη δική του κατάσταση βάσει της κατάστασης του απομακρυσμένου κόμβου. Επιπλέον επαναυπολογίζει την ολική κατάσταση σύμφωνα με τις ενημερώσεις καλώντας τη συνάρτηση computeGlobalState (γραμμή 8). Τότε συνεχίζει με την εξέταση του σύνδεσμου που έπεται ως προς το δυναμικό. Καθώς συνεχίζεται η επεξεργασία, η τοπική κατάσταση ενημερώνεται διαδοχικά ενσωματώνοντας αναδρομικά καινούρια πληροφορία ως αποτέλεσμα από την επεξεργασία στους απομακρυσμένους κόμβους. Αφού εξετάσει όλους τους συνδέσμους του, ο κόμβος w αποστέλει τη δική του ενημερωμένη πλεόν κατάσταση στον κόμβο u, ο οποίος του προώθησε αρχικά το ερώτημα (γραμμή 9). Ως επακόλουθο, ο κόμβος w υπολογίζει την απάντηση καλώντας τη μέθοδο computeLocalAnswer (γραμμή 10), κι αποστέλει με τη σειρά του στον κόμβο v τις τοπικές εγγραφές που ικανοποιουν το ερώτημα (γραμμή 11).

Algorithm 14: w.slow(v,u,Q,S^G,R): Προωθείται στον κόμβο u από τον κόμβο v για την επεξεργασία του ερωτήματος Q εντός της περιοχής R με ολική κατάσταση S^G.

```
1  S_w^L ← w.computeLocalState(Q, S^G) ;
2  S_w^G ← w.computeGlobalState(Q, S^G, S_w^L) ;
3  for ℓ ∈ w.sortLinks(w.comp(i, j, Q)) do
4      if w.isLinkRelevant(ℓ, Q, S_w^G, R) then
5          w.link[ℓ].slow(v, w, Q, S_w^G, w.link[ℓ].region∩R) ;
6          S^L ← w.receiveRemoteLocalState() ;
7          S_w^L ← w.updateLocalState(Q, {S_w^L, S^L}) ;
8          S_w^G ← w.computeGlobalState(Q, S^G, S_w^L) ;
9  w.sendLocalStateTo(u, S_w^L) ;
10 A ← w.computeLocalAnswer(Q, S_w^L) ;
11 w.sendLocalAnswerTo(v, A) ;
```

Από το γεγονός ότι στον αναδρομικό Αλγόριθμο 14 ο κάθε κόμβος επικοινωνεί με έναν μόνο άλλο κόμβο τη φορά περιμένοντας το αποτέλεσμά του γίνεται πρόδηλο ότι ο μέγιστος χρόνος επεξεργασίας είναι διαφορετικός από τον αλγόριθμο που μελετήσαμε προηγουμένως. Συγκεκριμένα, η απάντηση έρχεται μόνον αφού το μήνυμα έχει προωθηθεί σειριακά στους κόμβους με μεγάλη πιθανότητα για να συνεισφέρουν με πολύ σχετικές πλειάδες. Δεδομένου όμως ότι ο κάθε κόμβος ακολουθεί αναδρομικά την ίδια στρατηγική, ο χρόνος αναμονής (σε βήματα ή μέγιστο αριθμό επαναπροωθήσεων) ισούται με το συνολικό αριθμό κόμβων στην περιοχή R. Αυτό όμως ισχύει αν αναλύσουμε τη διαδικασία για τη χειρότερη περίπτωση και μόνον. Εν γένει, ο χρόνος απόκρισης είναι πολύ μικρότερος λόγω της προτεραιότητας βάσει της οποίας εξετάζουμε τους κόμβους ενώ είναι πολύ σημαντικό το γεγονός ότι η περιοχή αυτή περιορίζεται δραστικά καθώς εμπλουντίζεται η απάντηση. Εν ολίγοις, καθώς ο πιο σχετικός κόμβος προσφέρει όσες πλειάδες ζητήθηκαν είναι εντελώς ανούσιο να προωθηθεί το ερώτημα σε κάποιον λιγότερο σχετικό κόμβο. Ειδικότερα για top-k ερωτήματα, εάν υπάρχει

επιπλέον γνώση για την κατανομή των δεδομένων, π.χ. ότι όλοι οι κόμβοι διαθέτουν αρκετά περισσότερες από k εγγραφές κι ότι οι υπόλοιποι κόμβοι του δικτύου δεν διαθέτουν πλειάδες πιο σχετικές από οποιαδήποτε εγγραφή στο τοπικό αποτέλεσμα του πιο σχετικού κόμβου, θα μπορούσαμε με ευκολία να τεκμηριώσουμε ότι αρκεί να βρούμε τον πιο σχετικό κόμβο σε μόλις $O(\log n)$ βήματα και μηνύματα, παρά το γεγονός ότι θεωρητικά η χειρότερη περίπτωση προσδίδει γραμμικό κόστος ως προς το μέγεθος του δικτύου.

Για το υπόλοιπο μέρος της ενότητας θα παρουσιάσουμε τον γενικευμένο κατανεμημένο αλγόριθμο ripple ο οποίος θα χρησιμοποιηθεί στα επόμενα κεφάλαια προκειμένου να προταθούν μέθοδοι για την επεξεργασία μίας σειράς από τύπους ερωτημάτων κατάταξης. Ο αλγόριθμος αυτός είναι σε θέση να συμβιβάσει τον χρόνο εκτέλεσης με τον απαιτούμενο αριθμό μηνυμάτων μέσω της παραμέτρου r. Προκειμένου να ελαχιστοποιηθεί ο αριθμός των μηνυμάτων, η αναζήτηση αρχικά γίνεται βάσει κριτηρίων προωθώντας ενδελεχώς ερωτήματα στους κόμβους οι οποίοι διαθέτουν πολύ σχετικές εγγραφές με τον τρόπο που περιγράψαμε ανωτέρω για τον slow αλγόριθμο. Επιπλέον, αφού το ερώτημα φτάσει στους κόμβους οι οποίοι απέχουν περισσότερα από r βήματα από τον εκδότη του ερωτήματος, η περαιτέρω προώθηση του ερωτήματος γίνεται με τον τρόπο που περιγράψαμε για τον αλγόριθμο fast, προκειμένου να ελεγχθεί με τον τρόπο αυτό ο μέγιστος χρόνος εκτέλεσης. Ουσιαστικά, ο ripple αλγόριθμος στηρίζεται στο γεγονός ότι τα πρώτα βήματα της αναζήτησης προς κόμβους οι οποίοι είναι ιδιαίτερα σημαντικοί στο να σχηματιστεί γρήγορα ένα πολύ σχετικό με το ερώτημα αποτέλεσμα, το οποίο θα χρησιμοποιηθεί στη συνέχεια ώστε να αποτρέψει τη διαδικασία της αναζήτησης από το να σπαταλήσει χρόνο και πόρους για την ανάκτηση μη σχετικών πλειάδων.

Προκειμένου να επιβάλλουμε το σκεπτικό αυτό, το πρωτόκολλο που εκτελεί ο κάθε κόμβος όταν λαμβάνει ένα ερώτημα υπαγορεύεται από τον slow αλγόριθμο όταν η επαναπροώθηση του ερωτήματος έχει γίνει έως r φορές, ενώ για κόμβους που απέχουν πάνω από r βήματα από τον εκδότη του ερωτήματος η επεξεργασία λαμβάνει χώρα βάσει του αλγορίθμου fast. Συνεπώς, οι ακραίες τιμές που μπορεί να πάρει η μεταβλητή r είναι είτε Δ (ο μέγιστος αριθμός γειτόνων), όπου έτσι ο RIPPLE αλγόριθμος εκφυλίζεται στον slow, είτε 0 όπου ο αλγόριθμος εκφυλίζεται ανάλογα στον fast.

Ο Αλγόριθμος 16 δείχνει τις λεπτομέρειες της τοπικής επεξεργασίας ερωτημάτων για κάθε κόμβο. Αρχικά ένας κόμβος w λαμβάνει το ερώτημα Q μαζί με την τρέχουσα κατάσταση S, τη διεύθυνση του εκδότη του ερωτήματος v, τη διεύθυνση του κόμβου u που του προώθησε το μήνυμα, την περιοχή R του χώρου κλειδιών (keyspace) την οποία αφορά το ερώτημα, και την τιμή της παραμέτρου r. Οπότε ο κόμβος έχει τη δυνατότητα να υπολογίσει την τοπική κατάσταση και την ολική (γραμμές 1–2). Έπειτα, ανάλογα με την τιμή της παραμέτρου r ένας εκ των δύο βρόχων θα εκτελεστεί. Ο πρώτος βρόχος (γραμμές 4–9) αντιστοιχεί στον Αλγόριθμο 14 που μελετήσαμε προηγουμένως, με την εξαίρεση ότι πολλαπλές καταστάσεις μπορούν να ληφθούν (γραμμή 7) που να χρειάζονται επεξεργασία, π.χ. ενημέρωση της τοπικής κατάστασης κι υπολογισμός της ολικής κατάστασης (γραμμές 8–9). Ακόμα, η τιμή της παραμέτρου r μειώνεται στις επαναπροωθήσεις όπου αυτές χρειάζονται. Από την άλλη, ο δεύτερος βρόχος (γραμμές 11–13) αντιστοιχεί στον Αλγόριθμο 13 όπου όλοι οι επακόλουθοι κόμβοι λαμβάνουν αιτήματα με μηδενική r τιμή. Η τοπική απάντηση υπολογίζεται καλώντας τη μέθοδο computeLocalAnswer (γραμμή 15) κι έπειτα οι σχετικές εγγραφές αποστέλονται στον κόμβο που εξέδωσε το ερώτημα (γραμμή 16).

Ο μέγιστος αριθμός από βήματα που απαιτούνται εξαρτάται από την παράμετρο r κι από τον τύπο του κατανεμημένου πίνακα κατακερματισμού που χρησιμοποιείται. Στην αμέσως επόμενη ενότητα αναλύουμε τον χρόνο εκτέλεσης σε βήματα για το MIDAS. Για χαμηλές τιμές του r ο μέγιστος αριθμός επαναπροωθήσεων είναι κοντινότερα στη διάμετρο του δικτύου ενώ για υψηλές τιμές του r απαιτείται αριθμός βημάτων που προσεγγίζει το μέγεθος του δικτύου για τη χειρότερη περίπτωση.

Algorithm 15: w.ripple(v,u,Q,S^G,R,r): Προωθείται από τον κόμβο u για την επεξεργασία του ερωτήματος Q εντός της περιοχής R με ολική κατάσταση S^G και παράμετρο ripple r.

```
1  S_w^L ← w.computeLocalState(Q, S^G) ;
2  S_w^G ← w.computeGlobalState(Q, S^G, S_w^L) ;
3  if r > 0 then
4      for ℓ ∈ w.sortLinks(w.comp(i, j, Q)) do
5          if w.isLinkRelevant(ℓ, Q, S_w^G) then
6              w.link[ℓ].ripple(v, w, Q, S_w^G, w.link[ℓ].region∩R, r − 1);
7              {S_i^L} ← w.receiveRemoteLocalState() ;
8              S_w^L ← w.updateLocalState(Q, {S_w^L, {S_i^L}}) ;
9              S_w^G ← w.computeGlobalState(Q, S^G, S_w^L) ;
10 else
11     for each link i do
12         if w.isLinkRelevant(i, Q) then
13             w.link[i].ripple(v, u, Q, S_w^G, w.link[ℓ].region∩R, 0) ;
14 w.sendLocalStateTo(u, S_w^L) ;
15 A ← w.computeLocalAnswer(Q, S_w^L) ;
16 w.sendLocalAnswerTo(v, A) ;
```

6.1.2 Ανάλυση του RIPPLE για το MIDAS

Αυτή η ενότητα υποθέτει ότι η υποκείμενη δικτυακή υποδομή για το RIPPLE είναι το MIDAS[1]. Οπότε κατά συνέπεια οι περιοχές κι οι περιορισμοί που επιβάλλει ο αλγόριθμος που παρουσιάσαμε προηγουμένως υπόκεινται στα υποδέντρα που αντιστοιχούν στο κατανεμημένο k-d δέντρο το οποίο αντικατοπτρίζει την εικόνα του δικτύου. Άρα η παράμετρος r μπορεί να αντικατασταθεί με το βάθος δ του υποδέντρου που μελετάμε στο οποίο περιορίζεται η αναζήτηση. Απόροια της συγκεκριμένης σύμβασης θα είναι η έκφραση του χρόνου εκτέλεσης σε βήματα ως συνάρτηση του δ, όπως θα δούμε παρακάτω.

Λήμμα 11 *Ο μέγιστος χρόνος εκτέλεσης σε βήματα (επαναπροωθήσεις) του Αλγορίθμου 13 για το MIDAS δεν υπερβαίνει το* $L_f(\delta) = \Delta - \delta$.

Απόδειξη 11 *Θα ακολουθήσουμε την επαγωγική μέθοδο. Αρχικά, παρατηρούμε ότι* $L_f(\Delta) = 0$, *όπου κανένα μήνυμα δεν χρειάζεται να μεταδοθεί. Για την i-οστή εκτέλεση του βρόχου, ο Αλγόριθμος 13 προωθεί το ερώτημα σε ένα σύνδεσμο και το περιορίζει στο συμμετρικό υποδέντρο για βάθος i. Αναδρομικά, η επόμενη διαδοχή από τον κόμβο που έλαβε το μήνυμα απαιτεί* $1 + L_f(i)$ *βήματα. Καθώς οι εκτελέσεις των βρόχων λαμβάνουν χώρα ακριβώς μία φορά, ο πιο απαιτητικός χρόνος ορίζεται από το μακρύτερο μονοπάτι στα συμμετρικά υποδέντρα. Άρα, έχουμε ότι,*

$$L_f(\delta) = 1 + \max_{i=\delta+1}^{\Delta} L_f(i).$$

Αφού $L_f(i) > L_f(i+1)$, *παίρνουμε την αναδρομή που οδηγεί στο*

$$L_f(\delta) = 1 + L_f(\delta+1) = \Delta - \delta.$$

■

Θέτωντας $\delta = 0$, αποκτούμε τον μέγιστο αριθμό βημάτων για την επεξεργασία ερωτημάτων σύμφωνα με τον Αλγόριθμο 13 που ισούται με Δ. Ειδικά για το MIDAS είναι $O(\log n)$, όσο δηλαδή κι η διάμετρος του δικτύου.

[1] Σε πολύ παρόμοια ανάλυση θα καταλήγαμε για το Chord και για μία ολόκληρη οικογένεια από κατανεμημένους πίνακες κατακερματισμού.

Λήμμα 12 *Ο μέγιστος αριθμός βημάτων που απαιτείται από τον Αλγόριθμο 14 για το MIDAS δεν υπερβαίνει το* $L_s(\delta) = 2^{\Delta-\delta} - 1$.

Απόδειξη 12 *Ισχύει ότι* $L_s(\Delta) = 0$, *κι ότι κάθε εκτέλεση του βρόχου για τα συμμετρικά υποδέντρα στο βάθος* ℓ *εισάγει μέγιστο αριθμό βημάτων που απαιτούνται στα* $1 + L_s(\ell)$. *Καθώς ο αλγόριθμος περιμένει την απόκριση του συνδέσμου πριν προχωρήσει με τον επόμενο, ο συνολικός χρόνος εκτέλεσης δίνεται από το άθροισμα των αναδρομικών βημάτων που απαιτούνται ανά διαδοχική εκτέλεση ανεξάρτητα της σειράς κατά την οποία εξετάζονται τα υποδέντρα που αντιστοιχούν σε κάθε σύνδεσμο. Οπότε:*

$$L_s(\delta) = \sum_{\ell=\delta+1}^{\Delta} (1 + L_s(\ell)).$$

Από το οποίο παίρνουμε για την αναδρομή,

$$L_s(\delta) = 1 + 2 \cdot L_s(\delta + 1) = 2^{\Delta-\delta} - 1.$$

■

Θέτοντας $\delta = 0$, παίρνουμε ότι ο μέγιστος αριθμός βημάτων που απαιτείται από τον Αλγόριθμο 14 είναι 2^{Δ} που ισούται με $O(n)$ για το MIDAS. Όμως σημειώνουμε ότι λόγω της υψηλής προτεραιότητας που δίνεται προς τους κόμβους που είναι σχετικοί με το ερώτημα κι έχουν μεγάλη πιθανότητα να συνεισφέρουν στο αποτέλεσμα, ο μέσος απαιτούμενος αριθμός βημάτων που απαιτείται από τον slow είναι αρκετά χαμηλότερος, κάτι που επιβεβαιώνεται κι από την πειραματική μας ανάλυση.

Αρχικά διατυπώνουμε το ακόλουθο λήμμα το οποίο θα μας φανεί χρήσιμο στη συνέχεια για να υπολογίσουμε τον μέγιστο αριθμό βημάτων που απαιτούνται από τον αλγόριθμο ripple, ο οποίος συνδυάζει αμφότερες τεχνικές.

Λήμμα 13 *Για κάθε* $n \in \mathbb{N}^*$ *και* $k \in \mathbb{N}^*$ *με* $n \geq k$ *ισχύει ότι*

$$\binom{n}{k} = \sum_{i=k-1}^{n-1} \binom{i}{k-1} \tag{6.1}$$

Απόδειξη 13 *Χρησιμοποιούμε τον γνωστό αναδρομικό τύπο,*

$$\binom{n}{k} = \binom{n-1}{k-1} + \binom{n-1}{k}$$

κι αντικαθιστούμε αναδρομικά κάθε φορά τον τελευταίο μόνο όρο. Οπότε έχουμε

$$\begin{aligned}
\binom{n}{k} &= \binom{n-1}{k-1} + \binom{n-2}{k-1} + \binom{n-2}{k} \\
&= \binom{n-1}{k-1} + \binom{n-2}{k-1} + \binom{n-3}{k-1} + \binom{n-3}{k} \\
&= \binom{n-1}{k-1} + \cdots + \binom{k}{k-1} + \binom{k}{k} \\
&= \binom{n-1}{k-1} + \cdots + \binom{k}{k-1} + \binom{k-1}{k-1} \\
&= \sum_{i=k-1}^{n-1} \binom{i}{k-1}
\end{aligned}$$

■

Λήμμα 14 *Για κάθε $n,k,x \in \mathbb{N}^*$ με $n \geq k \geq x$ ισχύει ότι*

$$\binom{n}{k} = \underbrace{\sum_{i=k-1}^{n-1} \sum_{j=k-2}^{i-1} \cdots \sum_{y=k-x}}_{\#x} \binom{y}{k-x} \tag{6.2}$$

Απόδειξη 14 *Δουλεύουμε όπως προηγουμένως κι αναπτύσουμε τους όρους από το προηγούμενο λήμμα αναδρομικά χρησιμοποιώντας το x φορές.*

$$\binom{n}{k} = \sum_{i=k-1}^{n-1} \binom{i}{k-1} \overset{k\geq 1}{=} \sum_{i=k-1}^{n-1} \sum_{j=k-2}^{i-1} \binom{j}{k-2} = \underbrace{\sum_{i=k-1}^{n-1} \sum_{j=k-2}^{i-1} \cdots \sum_{y=k-x}}_{\#x} \binom{y}{k-x}$$

■

Ειδικά για την περίπτωση όπου $x = k$ παίρνουμε ότι $\binom{n}{k} = \underbrace{\sum_{i=k-1}^{n-1} \sum_{j=k-2}^{i-1} \cdots \sum_{u=0}}_{\#k} 1$

Λήμμα 15 *Ο αριθμός των κόμβων που προσπελαύνει ο αλγόριθμος* slow *στο MIDAS σε απόσταση ακριβώς δ βημάτων από τον κόμβο που εκδίδει ένα ερώτημα ισούται με $\binom{\Delta}{\delta}$.*

Απόδειξη 15 *Υπενθυμίζουμε ότι σύμφωνα με τον αλγόριθμο δρομολόγησης που παρουσιάσαμε στην Ενότητα 3.2 δεν δημιουργούνται κυκλικές διαδρομές και κανένας κόμβος δε λαμβάνει το ίδιο αίτημα δύο φορές. Θα χρησιμοποιήσουμε τη μέθοδο της επαγωγής για να αποδείξουμε το ζητούμενο.*

$\underline{\delta = 0}$ $\binom{\Delta}{0} = 1$, που ισχύει.

$\underline{\delta = 1}$ $\binom{\Delta}{1} = \Delta$, που ισχύει.

$\underline{\delta = k}$ $\binom{\Delta}{k}$, που θεωρούμε ότι ισχύει από την υπόθεση.

$\underline{\delta = k+1}$ Στο σκέλος αυτό υπολογίζουμε ουσιαστικά πόσοι κόμβοι λαμβάνουν κι επεξεργάζονται το ερώτημα στο βήμα $k+1$ κι άρα έχουν απόσταση ακριβώς $k+1$ από τον κόμβο που εξέδωσε το ερώτημα. Αυτό πρακτικά σημαίνει ότι θα πρέπει να ψάξουμε για υποδέντρα με βάθος τουλάχιστον $k+1$ που αντιστοιχούν στα $\Delta - k$ μεγαλύτερα υποδέντρα για τους πρώτους συνδέσμους που σχετίζονται με τις μεγαλύτερες περιοχές του αρχικού χώρου. Άρα, από τους Δ λοιπόν συνδέσμους που λαμβάνουν το μήνυμα από τον εκδότη μόνο οι $\Delta - k$ αντιστοιχούν στα συγκεκριμένα υποδέντρα ενώ από τους υπόλοιπους συνδέσμους δεν οδηγούμαστε σε υποδέντρα βάθους μεγαλύτερου του k. Άρα, δουλεύοντας αναδρομικά θα έχουμε να βρούμε στη συνέχεια πόσοι κόμβοι στα συγκεκριμένα υποδέντρα αυτά απέχουν ακριβώς $(k+1)-1 = k$ βήματα.

Οπότε μπορούμε να αντιμετωπίσουμε κάθε ένα από τα συγκεκριμένα $\Delta - k$ υποδέντρα ως ένα ανεξάρτητο δίκτυο με μικρότερους πίνακες δρομολόγησης (routing tables) μεγέθους $\Delta - j$ για το υποδέντρο που αντιστοιχεί στον j-οστό σύνδεσμο. Ο λόγος είναι ότι ένας κόμβος που λαμβάνει ένα μήνυμα δεν μπορεί να το προωθήσει σε κόμβους σε υψηλότερα επίπεδα στο δικό του πίνακα δρομολόγησης από αυτό στο οποίο βρίσκεται ο ίδιος ως προς τον κόμβο που του το προώθησε (προκειμένου να αποφευχθούν πολλαπλές λήψεις μηνυμάτων). Το σκεπτικό βασίζεται στο να υπολογίσουμε για κάθε ένα από αυτά τα υποδέντρα τον αριθμό των κόμβων που απασχολούνται και στη συνέχεια να τους προσθέσουμε. Δηλαδή χρησιμοποιώντας το προηγούμενο σκέλος έχουμε $\sum_{j=k}^{\Delta-1} \binom{\Delta-j}{k}$ το οποίο από το προηγούμενο λήμμα ισούται με $\binom{\Delta}{k+1}$.

■

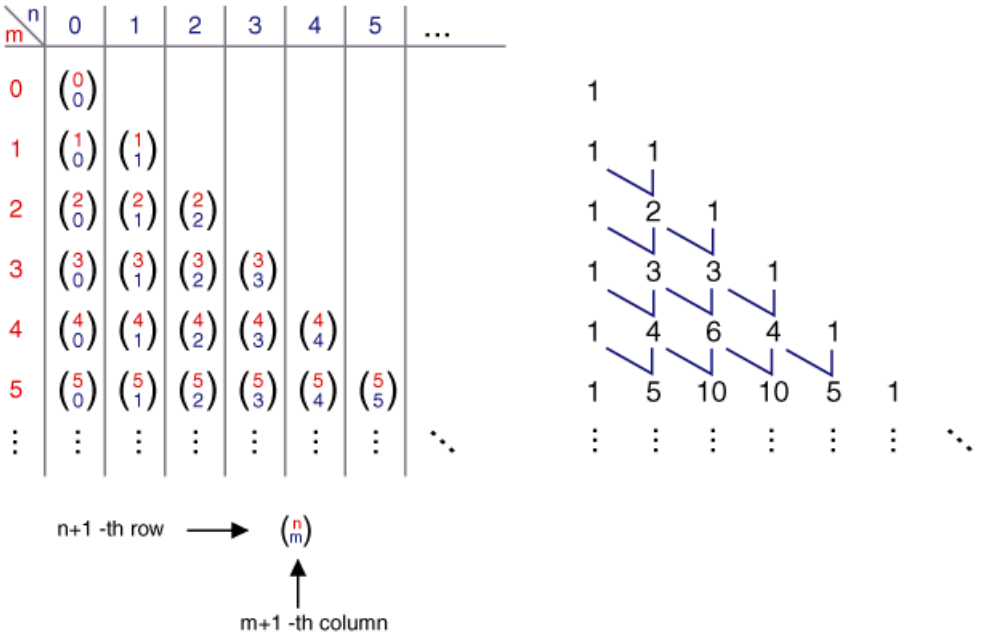
Στην προηγούμενη απόδειξη θα μπορούσαμε να υπολογίσουμε τα μονοπάτια μήκους $k+1$ από ακόμα βαθύτερους συνδέσμους συνδέσμων (επαναπροωθησεις έως $k+1$ βάθος) και να οδηγηθούμε πάλι στο ίδιο αποτέλεσμα. Για παράδειγμα σε ένα από τα επόμενα βήματα επαναπροωθήσεων όπου θα ψάχνουμε πλέον για μονοπάτια μήκους $k-1$, ο πρώτος σύνδεσμος του πρώτου συνδέσμου προωθεί το μήνυμα σε $\Delta-2$ δικούς του συνδέσμους δημιουργώντας $\binom{\Delta-2}{k-1}$ κατάλληλα μονοπάτια, ενώ ο $\Delta-1-(k-1)$ συνδεσμος του πρώτου συνδέσμου δημιουργεί $\binom{k-1}{k-1}$ κατάλληλες διαδρομές. Άρα από τον πρώτο σύνδεσμο έχουμε $\sum_{j=k-1}^{\Delta-2}\binom{j}{k-1}$ διαδρομές ενώ συνολικά για όλους τους συνδέσμους του εκδότη $\sum_{i=k}^{\Delta-1}\sum_{j=k-1}^{i-1}\binom{j}{k-1}$, που μας οδηγεί από το προηγούμενο λήμμα σε $\binom{\Delta}{k+1}$ μονοπάτια κατάλληλου μήκους. Προσέξτε το διπλό άθροισμο όταν υπολογίζουμε τα μονοπάτια κατάλληλου μήκους μετά το δεύτερο βήμα αφού μετράμε τις διαδρομές από τους πρώτους $\Delta-k$ συνδέσμους του εκδότη που περνάνε με τη σειρά τους από τους $\Delta-j-(k-1)$ πρώτους συνδέσμους του j-οστού συνδέσμου του εκδότη. Αξίζει να σημειωθεί ότι καταλήγουμε στο ίδιο αποτέλεσμα και για μεγαλύτερα βάθη αναλόγως αλλά με χρονοβόρους υπολογισμούς αν δεν χρησιμοποιήσουμε το ανωτέρω λήμμα.

Ένας εναλλακτικός πιο διαισθητικός τρόπος για να δείξουμε το ανωτέρω λήμμα βασίζεται στη μελέτη του γράφου που σχηματίζεται με τις ακμές να υποδηλώνουν την προώθηση ενός μηνύματος προς έναν άλλο σύνδεσμο (process tree), όπως δείχνουμε στο Σχήμα 6.2. Οπότε αν και μελετάμε την προώθηση του ερωτήματος μέσω όλων των συνδέσμων, ξεκινάμε να εξετάζουμε τον κόμβο στο πιο χαμηλό επίπεδο που έχει το ίδιο αναγνωριστικό με τον εκδότη εκτός του τελευταίου δυφίου. Όπως δείχνουμε και στον Πίνακα 6.1.2, ο κόμβος αυτός στο επίπεδο 1 απέχει ακριβώς ένα hop και δεν μπορεί να προωθήσει περαιτέρω το μήνυμα αφού το έλαβε με παράμετρο ίση με το βάθος του δέντρου. Ο επόμενος σύνδεσμος στο επίπεδο 2 αντιστοιχεί σε ένα υποδέντρο μεγέθους δύο κόμβων κι άρα όποιος κι από τους δύο να λάβει το μήνυμα στο πρώτο hop, το προωθεί στον άλλον. Δηλαδή απασχολούνται στο νέο υποδέντρο ένας κόμβος για κάθε ένα από τα βήματα 1 και 2 ενώ για το συνολικό αριθμό κόμβων ως το επίπεδο 2 προσθέτουμε και τους κόμβους του επιπέδου 1. Προχωρώντας στα ανώτερα επίπεδα κάθε φορά δουλεύουμε για ένα υποδέντρο διπλάσιου μεγέθους από το προηγούμενο ενώ στο ανάλογο process tree το νέο υποδέντρο έχει την ίδια μορφή που είχε το process tree για όλα μαζί τα προηγούμενα επίπεδα. Οπότε για να υπολογίσουμε τους κόμβους σε κάθε βήμα του επίπεδο i, αθροίζουμε τους κόμβους που απασχολήσαμε στο επίπεδο $i-1$ με την ίδια γραμμή μετατοπισμένη κατά ένα βήμα προς τα δεξιά για το νέο υποδέντρο (με zero filling για την κενή θέση που δημιουργείται). Συνεπώς, ο Πίνακας 6.1.2 προκύπτει με τον ίδιο ακριβώς τρόπο με το τρίγωνο του Pascal (Σχήμα 6.1). Στο βήμα 0 απασχολείται μόνο ο εκδότης που θεωρούμε ότι βρίσκεται στο επίπεδο 0 κατά σύμβαση ενώ μόνον ο αδερφός κόμβος στο επίπεδο 1, κ.ο.κ. Ως γνωστόν παίρνουμε από το τρίγωνο του Pascal που σχηματίζεται ότι για το επίπεδο i και το βήμα j έχουμε ακριβώς $\binom{i}{j}$ κόμβους. Όταν έχουμε φτάσει στο τελευταίο επίπεδο Δ έχουμε τους κόμβους που προσπελαύνει ο αλγόριθμος αναλυτικά για κάθε βήμα j που τώρα ξέρουμε ότι δίνονται από τον τύπο $\binom{\Delta}{j}$.

	βήμα 0	βήμα 1	βήμα 2	βήμα 3	βήμα 4	βήμα 5	βήμα 6
επίπεδο 0	1	0					
επίπεδο 1	1	1	0				
επίπεδο 2	1	2	1	0			
επίπεδο 3	1	3	3	1	0		
επίπεδο 4	1	4	6	4	1	0	
επίπεδο 5	1	5	10	10	5	1	0
επίπεδο 6	1	6	15	20	15	6	1

Πίνακας 6.1: Αναδρομική προσπέλαση κόμβων μέχρι το j-οστό σύνδεσμο ανά βήμα.

Στο Σχήμα 6.2 δείχνουμε τις συμμετρίες στο γράφο που σχηματίζεται με τις ακμές να υποδηλώνουν την προώθηση ενός μηνύματος προς έναν άλλο σύνδεσμο σχηματίζοντας έτσι ένα δέντρο διεργασίας (process tree). Κυκλώνουμε με κόκκινο τη συμμετρική προώθηση του μηνύματος από τον εκδότη στο σύνδεσμο του πρώτου επιπέδου αλλά και την προώθηση του συνδέσμου του πρώτου επιπέδου στο δικό του αδερφό κόμβο. Ομοίως το υποδέντρο



Σχήμα 6.1: Το τρίγωνο του Pascal κι οι συντελεστές του.

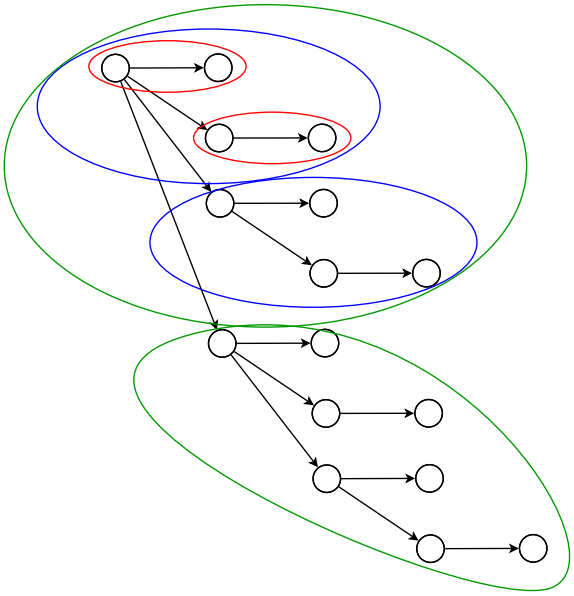
του process tree που προκύπτει από την πρώθηση του αιτήματος στον τρίτο σύνδεσμο έχει την ίδια μορφή του process tree για τα προηγούμενα δύο επίπεδα, όπως άλλωστε δείχνουμε με μπλε χρώμα. Αντίστοιχα συμμετρικά σχηματίζεται και το μέρος του process tree για το τέταρτο επίπεδο που δείχνουμε με πράσινο μέχρι και το Δ επίπεδο.

Στη συνέχεια υπολογίζουμε την απόδοση του RIPPLE. Κατ' αναλογίαν με τα προηγούμενα το μέγεθος αυτό θα είναι ίσο με $L_{\mathsf{slow}}(0) - \binom{\Delta}{r} L_{\mathsf{slow}}(r) + L_{\mathsf{fast}}(r)$. Δηλαδή έχει απόδοση που δίνεται από την εκτέλεση του slow ξεκινώντας από τη ρίζα κι αφαιρωντας το κόστος σε βήματα για τις προωθήσεις στους κόμβους που απέχουν πάνω από r βήματα. Τον συντελεστή που αντιστοιχεί στον αριθμό των προωθήσεων αυτών τον βρίσκουμε εφαρμόζοντας το προηγούμενο λήμμα. Τέλος, έχουμε την εκτέλεση του fast στα υποδέντρα που απέχουν απόσταση μεγαλύτερη του r.

Λήμμα 16 *Ο μέγιστος αριθμός βημάτων που απαιτεί ο Αλγόριθμος 16 για το MIDAS είναι* $L_{ripple}(\Delta, r) = \sum_{\delta=0}^{r} \binom{\Delta}{\delta} - 1 + \Delta - r$.

Απόδειξη 16 *Η απόδειξη έχει δύο μέρη. Πρώτον, δείχνουμε τον αριθμό βημάτων που απαιτούνται από το πρώτο μέρος του RIPPLE όπου εκτελείται ο slow για μέγιστο βάθος μέχρι* r. *Θεωρούμε ότι στο βήμα* $\delta \leq r$ *λαμβάνουν κι επεξεργάζονται το ερώτημα ακριβώς* $\binom{\Delta}{\delta}$ *κόμβοι όπως δείξαμε στο προηγούμενο λήμμα. Σημειωτέον ότι ο slow προσπελαύνει τους κόμβους σειριακά για βάθη από* 0 *έως και* r. *Συνεπώς ο slow απαιτεί* $\sum_{\delta=0}^{r} \binom{\Delta}{\delta} - 1$ *βήματα, δηλαδή όσοι κόμβοι απασχολούνται συνολικά μείον τον αρχικό κόμβο αφού στον εκδότη δεν προωθήθηκε το μήνυμα αν και το προώθησε. Επιπλέον στην προηγούμενη ενότητα δείχνουμε την απόδοση του fast η οποία είναι* $\Delta - r$. *Αθροίζοντας τους παράγοντες καταλήγουμε στο ζητούμενο.* ■

Οπότε για $r = 0$ έχουμε μέγιστο αριθμό βημάτων $L_{\mathsf{ripple}}(\Delta, 0) = \sum_{\delta=0}^{0} \binom{\Delta}{0} - 1 + \Delta - 0 = \Delta$ που αντιστοιχεί στην απόκριση του fast. Αναλόγως για $r = \Delta$ βρίσκουμε ότι $L_{\mathsf{ripple}}(\Delta, \Delta) = \sum_{\delta=0}^{\Delta} \binom{\Delta}{\delta} - 1 + \Delta - \Delta = 2^{\Delta} - 1$ που αντιστοιχεί στο latency του slow.



Σχήμα 6.2: Γράφος προώθησης μηνυμάτων για τον slow σε δίκτυο 2^4 κόμβων.

6.2 Top-k Ερωτήματα

Πρώτα παρουσιάζουμε την εφαρμογή του RIPPLE για top-k ερωτήματα. Δεδομένου μίας παραμέτρου k και μίας unimodal/quasiconvex συνάρτησης κατάταξης f, ο χρήστης ζητάει ένα σύνολο A που απαρτίζεται από k πλειάδες ώστε $\forall \vec{t} \in A, \forall \vec{t'} \notin A : f(\vec{t}) \geq f(\vec{t'})$. Μία τέτοια συνάρτηση παρουσιάζει ένα και μοναδικό τοπικό μέγιστο.

Σε αυτήν την κατηγορία ερωτημάτων, ένα ερώτημα Q εμπεριέχει τόσο τη συνάρτηση f με τα σχετικά βάρη της για κάθε διάσταση, όσο και την παράμετρο k. Η κατάσταση ενός ερωτήματος περιγράφεται από τις παραμέτρους m και τ οι οποίες υποδεικνύουν ότι οι m εγγραφές που έχουν ανακτηθεί έως τώρα για το αποτέλεσμα έχουν σκορ μεγαλύτερο από τ.

Χρησιμοποιώντας το framework που περιγράψαμε στην Ενότητα 6, περιγράφουμε την υλοποίησή των συναρτήσεων της διεπαφής του RIPPLE για την επεξεργασία top-k ερωτημάτων. Η πρώτη συνάρτηση είναι η computeLocalState που δείχνουμε στον Αλγόριθμο 16, η οποία είναι αρμόδια για να ενημερώνει την τοπική κατάσταση δεδομένου της ολικής κατάστασης που προέρχεται από το ερώτημα. Η συνάρτηση αυτή εκτελείται στον κόμβο w και δέχεται ως είσοδο τις παραμέτρους του ερωτήματος Q (f,k) καθώς και την ολική κατάσταση (m^G, τ^G) κι επιστρέφει την τοπική κατάσταση (m_w^L, τ_w^L).

Το σκεπτικό πίσω από την μέθοδο computeLocalState είναι να εντοπιστούν οι πιο κατάλληλες εγγραφές για το αποτέλεσμα προκειμένου να σχηματιστεί το αποτέλεσμα των k πιο σχετικών εγγραφών. Οπότε, αρχικά ο κόμβος w ανακτά μέχρι k τοπικά αποθηκευμένες πλειάδες που επιτυγχάνουν σκορ μεγαλύτερο από τ^G (γραμμή 1) και τις προσθέτει στο αποτέλεσμα A. Εάν ο αριθμός των ανακτημένων εγγραφών μαζί με αυτές που υπάρχουν ήδη στο αποτέλεσμα είναι λιγότερες από k (γραμμή 2), τότε ο κόμβος w επιπροσθέτως βρίσκει τις $k-m$ καλύτερες πλειάδες ακόμα κι αν αυτές επιτυγχάνουν σκορ χειρότερο από τ^G (γραμμή 3). Όταν ολοκληρωθεί η εκτέλεση της computeLocalState, θέτουμε τις παραμέτρους της τοπικής κατάστασης m_w^L σύμφωνα με τον αριθμό των τοπικών πλειάδων που ανακτήθηκαν και τ_w^L με το μικρότερο σκορ που επιτυγχάνεται από τις πλειάδες αυτές.

Η μέθοδος computeGlobalState του Αλγόριθμου 17 παράγει την ολική κατάσταση στον w παίρνοντας υπόψην την προηγούμενη ολική κατάσταση που του είχε προωθηθεί (m^G, τ^G) καθώς και την νέα τοπική κατάσταση στον w (m_w^L, τ_w^L). Για τον συγκεκριμένο τύπο ερωτήματος απλά παραθέτει τον αριθμό των πλειάδων στο αποτέλεσμα και θέτει το όριο τ^G ανάλογα με την χαμηλότερη τιμή των δύο.

Η μέθοδος computeLocalAnswer του Αλγορίθμου 18 εξάγει τις τοπικά αποθηκευμένες

Algorithm 16: w.top-computeLocalState(f, k, m^G, τ^G)

```
1 insert in A up to k local tuples with score better than τ^G ;
2 if m^G + |A| < k then
3 |  insert in A up to k − m^G − |A| highest ranking local tuples ;
4 return (m_w^L, τ_w^L) ← (|A|, f(A)) ;
```

Algorithm 17: w.top-computeGlobalState($f, k, m^G, \tau^G, m_w^L, \tau_w^L$)

```
1 return (m_w^G, τ_w^G) ← (m^G + m_w^L, min{τ^G, τ_w^L}) ;
```

εγγραφές βάσει της τοπικής κατάστασης. Ειδικά για τον τύπο ερωτήματος που μελετάμε στην ενότητα αυτή σημαίνει ότι όλες οι τοπικές εγγραφές που επιτυγχάνουν σκορ μεγαλύτερο από το όριο τ ανακτώνται για περαιτέρω επεξεργασία.

Algorithm 18: w.top-computeLocalAnswer(f, k, m_w^L, τ_w^L)

```
1 insert in A all local tuples with score better than τ_w^L ;
2 return A ;
```

Η επόμενη συνάρτηση που μελετάμε είναι η updateLocalState τις λεπτομέρειες της οποίας δείχνουμε στον Αλγόριθμο 19. Σκοπός της η ενημέρωση της τοπικής κατάστασης δεδομένου ενός συνόλου παλαιότερων τοπικών καταστάσεων. Εκτελείται στον κόμβο w και δέχεται ως είσοδο το ερώτημα Q (f,k) καθώς κι ένα σύνολο τοπικών καταστάσεων ($\{m_i^L, \tau_i^L\}$) ενω επιστρέφει την νέα τοπική κατάσταση (m_w^L, τ_w^L). Διαισθητικά, η μέθοδος updateLocalState προσπαθεί να βρει το υψηλότερο όριο τ το οποίο επιτρέπει την ανάκτηση των καλύτερων k πλειάδων, εφόσον βέβαια αυτές υπάρχουν.

Αρχικά, ο κόμβος w ταξινομεί τις καταστάσεις σε φθίνουσα σειρά σύμφωνα πάντα με τις τ τιμές τους (γραμμή 1) κι αρχικοποιεί τη μεταβλητή για τον υπολογισμό του αποτελέσματος m_w^L (γραμμή 2). Έπειτα εξετάζει την κάθε τοπική κατάσταση από την είσοδο (γραμμές 3–6), προσαυξάνει ανάλογα την παράμετρο m_w^L (γραμμή 4) και θέτει το όριο σύμφωνα με τα όρια των καταστάσεων που εξετάστηκαν (γραμμή 5). Η μέθοδος τερματίζει είτε όταν έχουν εξεταστεί όλες, είτε όταν ο αριθμός των ανακτηθέντων πλειάδων φτάσει τον επιθυμητό αριθμό k (γραμμή 6).

Ο Αλγόριθμος 20 αποφασίζει για το εάν η περιοχή ενός συγκεκριμένου σύνδεσμου του w εμπεριέχει σχετικές εγγραφές δεδομένου της τρέχουσας κατάστασης της ολικής κατάστασης. Το ερώτημα προωθείται σε έναν σύνδεσμο εφόσον ο συνολικός αριθμός των πλειάδων που έχουν ανακτηθεί (όπως γνωστοποιείται στον κόμβο w) είναι μικρότερος από k ή αν η συσχετιζόμενη περιοχή καλύπτει σημεία του χώρου που επιτυγχάνουν σκορ μεγαλύτερο από τ_w^G. Για τον έλεγχο αυτό κάνουμε χρήση της συνάρτησης f^+, η οποία επιστρέφει ένα άνω όριο για το σκορ που μπορεί να έχει μία πλειάδα εντός της περιοχής του υπό εξέταση κόμβου. Σημειωτέον ότι πρόκειται για εκτίμηση που υπολογίζεται χωρίς να εξετάσουμε τις πραγματικές εγγραφές που αφορούν το συγκεκριμένο υποχώρο. Πρόκειται δηλαδή για ένα αυστηρό φράγμα που υπολογίζεται από τα χαρακτηριστικά της περιοχής και το οποίο μας εγγυάται ότι δεν εμπεριέχεται καμία εγγραφή με καλύτερο σκορ από το συγκεκριμένο κατώφλι.

Τέλος, ο Αλγόριθμος 21 συγκρίνει δύο συνδέσμους βάσει της πιθανότητας των περιοχών τους να εμπεριέχουν πολύ σχετικές πλειάδες. Για τον σκοπό αυτό γίνεται και πάλι χρήση της συνάρτησης δυναμικού f^+.

6.3 Ερωτήματα Κορυφογραμμής

Στην Ενότητα 6.3.1, περιγράφουμε την υλοποίηση της διεπαφής του RIPPLE για την κατανεμημένη επεξεργασία ερωτημάτων κορυφογραμμής. Έπειτα, στην Ενότητα 6.3.2 εστιάζουμε

Algorithm 19: w.top-updateLocalState($f, k, \{m_i^L, \tau_i^L\}$)

```
1 sort {m_i^L, τ_i^L} entries descending in their τ_i^L values ;
2 m_w^L ← 0 ;
3 for each entry (m_i^L, τ_i^L) do
4 |   m_w^L ← m_w^L + m_i^L ;
5 |   τ_w^L ← τ_i^L ;
6 |   if m_w^L ≥ k then break ;
7 return (m_w^L, τ_w^L) ;
```

Algorithm 20: w.top-isLinkRelevant(i, f, k, m_w^G, τ_w^G)

```
1 return m_w^G < k or f⁺(w.link[i].region) ≥ τ_w^G ;
```

στο MIDAS και προτείνουμε μία βελτιστοποίηση για το συγκεκριμένο τύπο ερωτημάτων επιβάλλοντας μία διαφορετική δομή αλλάζοντας τον τρόπο διαμόρφωσης του δικτύου.

6.3.1 Υπολογισμός κι Ανάκτηση Κορυφογραμμής

Στην ενότητα αυτή θα παρουσιάσουμε τις λεπτομέρειες της κατανεμημένης επεξεργασίας ερωτημάτων κορυφογραμμής όπως αυτή λαμβάνει χώρα για το RIPPLE. Θεωρούμε ότι μία πλειάδα $\vec{t}$ υπερτερεί μίας άλλης $\vec{t'}$, και συμβολίζουμε με $\vec{t} \succ \vec{t'}$, εάν η $\vec{t}$ έχει καλύτερες ή εξίσου καλές τιμές σε όλες τις διαστάσεις και αυστηρά καλύτερες σε τουλάχιστον μία διάσταση. Δίχως βλάβη της γενικότητας θεωρούμε ως καλύτερες τις μικρότερες τιμές. Έτσι έχουμε ότι το ερώτημα κορυφογραμμής αποτελεί αναζήτηση για όλες τις πλειάδες ως προς τις οποίες δεν υπερτερεί καμία άλλη. Η κατάσταση S ενός ερωτήματος κορυφογραμμής ορίζεται από το σύνολο των πλειάδων ως προς τις οποίες δεν υπερτερεί καμία άλλη πλειάδα στους κόμβους που έχουν λάβει έως τη δεδομένη στιγμή το ερώτημα (μερική κορυφογραμμή).

Πρώτα περιγράφουμε την υλοποίηση της μεθόδου computeLocalState για τον συγκεκριμένο τύπο ερωτήματος. Όπως δείχνουμε στον Αλγόριθμο 22, αρχικά ο κόμβος w υπολογίζει τη δική του κορυφογραμμή, η οποία αποτελεί την τοπική κατάσταση (γραμμή 1). Τότε, ενσωματώνει τις ανακτημένες εγγραφές στην ληφθείσα ολική κατάσταση αφαιρώντας τις πλειάδες που δεν υπερτερούν πια. Με τον τρόπο αυτό σχηματίζουμε την ολική κατάσταση S_w^G στον κόμβο w (γραμμή 2). Η ενημερωμένη τοπική κατάσταση υπολογίζεται από την τομή της τοπικής κορυφογραμμής και της ολικής κατάστασης (γραμμή 3).

Η μέθοδος computeGlobalState που δείχνουμε στον Αλγόριθμο 23 ορίζει την ολική κατάσταση στον κόμβο w. Όπως περιγράψαμε και προηγουμένως, το S_w^G συμβολίζει την ολική κατάσταση και ταυτόχρονα την κορυφογραμμή όπως υπολογίστηκε βάσει της ληφθείσας ολικής κατάστασης και της τοπικής κορυφογραμμής. Επιπλέον, η μέθοδος computeLocalAnswer που δείχνουμε στον Αλγόριθμο 24 επιστρέφει τις τοπικές πλειάδες από την τοπική κατάσταση S_w^L.

Η μέθοδος updateLocalState που δείχνουμε στον Αλγόριθμο 25 δέχεται ως είσοδο ένα σύνολο από τοπικές καταστάσεις $\{S_i^L\}$ και τις συνδυάζει έτσι ώστε να παράξει τη νέα τοπική κατάσταση. Συγκεκριμένα, ο κόμβος w ενσωματώνει όλες τις τοπικές καταστάσεις κι υπολογίζει από αυτές την κορυφογραμμή, η οποία γίνεται η νέα τοπική κατάσταση του κόμβου w.

Σχετικά με τη μέθοδο isLinkRelevant, ο Αλγόριθμος 26 ελέγχει τις πλειάδες της ολικής κατάστασης (γραμμές 1–3). Αν οποιαδήποτε από αυτές υπερτερεί ολόκληρης της περιοχής

Algorithm 21: w.top-comp(i, j, f, k)

```
1 return f⁺(w.link[i].region) > f⁺(w.link[j].region) ;
```

Algorithm 22: w.sky-computeLocalState(S^G)

1 $S_w^L \leftarrow$ the local skyline;
2 $S_w^G \leftarrow$ computeSkyline($S^G \cup S_w^L$);
3 **return** $S_w^L \leftarrow S_w^L \cap S_w^G$;

Algorithm 23: w.sky-computeGlobalState(S^G, S_w^L)

1 **return** $S_w^G \leftarrow$ computeSkyline($S^G \cup S_w^L$);

που αντιστοιχεί στον σύνδεσμο, τότε μπορούμε εκ του ασφαλούς να παραλείψουμε τον συγκεκριμένο σύνδεσμο από την αναζήτηση αφού δεν εμπεριέχει εγγραφές που να ανήκουν στην κορυφογραμμή. Διαφορετικά, ο σύνδεσμος θα πρέπει να συμπεριληφθεί στην αναζήτηση (γραμμή 4).

Τέλος, η μέθοδος comp που δείχνουμε στον Αλγόριθμο 27, συγκρίνει τις περιοχές δύο συνδέσμων με κριτήριο την απόστασή τους από την αρχή των αξόνων $\vec{0}$. Η συνάρτηση d^- υπολογίζει την ελάχιστη δυνατή απόσταση οποιασδήποτε πλειάδας εντός μιας περιοχής από την αρχή των αξόνων $\vec{0}$, αποτελώντας έτσι ένα κατώφλι για το δυναμικό ενός κόμβου να συμμετάσχει στη κορυφογραμμή.

6.3.2 Προσαρμοσμένη Δομή MIDAS για Ερωτήματα Κορυφογραμμής

Στην ενότητα αυτή παρουσιάζουμε έναν τρόπο δόμησης του δικτύου με σκοπό τη βελτιστοποίηση της κατανεμημένης επεξεργασίας ερωτημάτων κορυφογραμμής στο RIPPLE για το MIDAS. Το σκεπτικό βασίζεται στο γεγονός ότι επιθυμούμε τον κόμβο που λαμβάνει την αίτηση να ανήκει στην κορυφογραμμή όσο πιο συχνά γίνεται, δηλαδή να αυξήσουμε την ακρίβεια[2] και την ταχύτητα των μεθόδων που παρουσιάσαμε στην προηγούμενη ενότητα. Με τον τρόπο αυτό μειώνεται το κόστος επικοινωνίας σε μηνύματα κι εξοικονομείται εύρος ζώνης (bandwidth) καθώς κατευθύνεται η αναζήτηση γρήγορα προς τους σχετικούς με το ερώτημα κόμβους. Για να κατανοήσουμε καλυτερα τι απαιτείται προκειμένου να υλοποιηθεί η προσέγγιση αυτή ας μελετήσουμε την πρόοδο ενός τέτοιου ερωτήματος σύμφωνα με όσα περιγράψαμε στην προηγούμενη ενότητα. Παρατηρήστε ότι εκτελώντας το RIPPLE σε έναν κόμβο που βρίσκεται στη μέση του συνολικού χώρου του ευρετηρίου θα δημιουργούσε μία τοπική απάντηση η οποία θα απαρτιζόταν από πλειάδες ως προς τις οποίες υπερτερεί το περιεχόμενο πολλών κόμβων του δικτύου. Από την άλλη, όχι απαραίτητα όλοι οι κόμβοι οι οποίοι βρίσκονται παρά τους άξονες του χώρου ανήκουν στη κορυφογραμμή, αν και πολλοί από αυτούς συνεισφέρουν σε μεγάλο ποσοστό στο τελικό αποτέλεσμα.

Τώρα τίθεται το θέμα πως μπορούμε να εντοπίσουμε τους κόμβους αυτούς που έχουν ισχυρή πιθανότητα να συνεισφέρουν στο αποτέλεσμα ώστε να τους εντάξουμε στη διαδικασία αναζήτησης όσο το δυνατόν συντομότερα. Υπενθυμίζουμε από το Κεφαλαιο 3 ότι το MIDAS αντικατοπτρίζει ένα εικονικό κατανεμημένο k-d δέντρο και κάθε κόμβος w διατηρεί συνδέσμους με κόμβους οι οποίοι βρίσκονται εντός των $\log n$ υποδέντρων που έχουν ως ρίζες τους κόμβους που είναι αδέρφια των κόμβων που βρίσκονται στο μονοπάτι του w προς τη ρίζα ολόκληρου του δέντρου. Ξέχωρα από αυτόν τον περιορισμό δεν έχουμε ορίσει κάποιον άλλον σχετικά με τις ιδιότητες που επιβάλλεται να τηρούν οι σύνδεσμοι ενός κόμβου. Συνεπώς, υπάρχει το περιθώριο κι η ελευθερία να διαμορφώσουμε τη δομή του MIDAS με τέτοιον

[2]Το μέγεθος που εκφράζει το λόγο του αριθμού των κόμβων που έχουμε προσπελάσει κι είναι ταυτόχρονα σχετικοί με το ερώτημα προς το συνολικό αριθμό των κόμβων που έχουμε προσπελάσει

Algorithm 24: w.sky-computeLocalAnswer(S_w^L)

1 **return** $A \leftarrow$ local tuples of S_w^L;

Algorithm 25: w.sky-updateLocalState($\{S_i^L\}$)

```
1 return S_w^L ← computeSkyline(⋃_i S_i^L);
```

Algorithm 26: w.sky-isLinkRelevant(i, S_w^G)

```
1 for each s⃗ ∈ S_w^G do
2     if s⃗ ≻ w.link[i].region then
3         return false;
4 return true;
```

τρόπο ώστε να εξοικονομήσουμε πόρους του δικτύου κατά την επεξεργασία ερωτημάτων κορυφογραμμής.

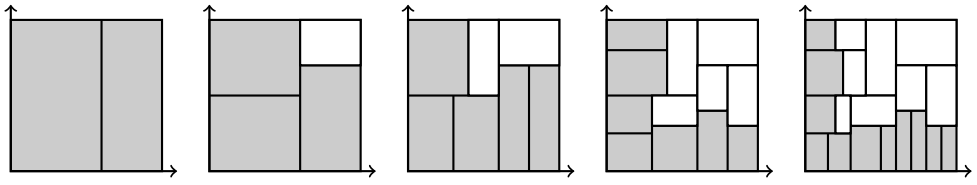
Στο MIDAS έχουμε τη δυνατότητα να ταυτοποιήσουμε τους κόμβους που είναι υπεύθυνοι για ζώνες που βρίσκονται παρά τους άξονες του συνολικού χώρου. Το Σχήμα 6.3 οπτικοποιεί ένα παράδειγμα δύο διαστάσεων όπου οι διαστάσεις που χωρίζουν αναδρομικά το χώρο εναλλάσονται διαδοχικά. Έτσι έχουμε τους κόμβους που δείχνουμε με γκρι των οποίων τα αναγνωριστικά ικανοποιούν έστω μία από τις κανονικές εκφράσεις (regular expressions) $p_h = (X0)^*X?$ και $p_v = (0X)^*0?$, όπου το X μπορεί να πάρει την τιμή είτε 1, είτε 0 ($X \leftarrow (0|1)$). Σημειωτέον ότι οι κόμβοι οι οποίοι έχουν ένα 0 ανά δύο δυφία είναι υπεύθυνοι για τα χαμηλότερα μέρη του χώρου κατά μήκος των οριζοντίων και των καθέτων αξόνων αντίστοιχα. Ανάλογα για d διαστάσεις όπου οι διαστάσεις που διαχωρίζουν τον χώρο εναλλάσονται σειριακά, έχουμε ακριβώς d πρότυπα συνολικά όπου $p_0 = (X0\cdots0)^*X0\{0,d-1\}$, $p_1 = (0X0\cdots0)^*0X0\{0,d-2\}$, $p_2 = (00X0\cdots0)^*00X0\{0,d-3\}$, κ.ο.κ. Δεν είναι δύσκολο να δείξουμε οτι εάν ένας κόμβος έχει αναγνωριστικό το οποίο δεν συνάγει με καμία από τις κανονικές εκφράσεις, τότε κανένας απόγονός του θα το κάνει ανεξαρτήτως του βάθους του. Αυτό οφείλεται στο γεγονός ότι τα αναγνωριστικά των απογόνων κόμβων έχουν αναγνωριστικά τα οποία εκ κατασκευής έχουν ως πρόθεμα τα αναγνωριστικά των προγόνων τους.

Θα επιβάλλουμε από τους συνδέσμους ενός κόμβου να υπακούουν στα αναγνωριστικά σύμφωνα με τις κανονικές εκφράσεις $p_0, \cdots, p_{d-1}$, όταν αυτό είναι δυνατόν. Προκειμένου να επιτύχουμε την επιδίωξή μας αυτή θα ενσωματώσουμε κανόνες στο πρωτόκολλο εισαγωγής κόμβων ώστε οι νέοι σύνδεσμοι να τηρούν τις συγκεκριμένες προδιαγραφές καθώς το δίκτυο επεκτείνεται. Αυτό σημαίνει ότι στην πράξη ο j-ιοστός σύνδεσμος κάθε κόμβου εγκαθιδρύεται με τέτοιον τρόπο ώστε να έχει αναγνωριστικό που να υπακούει σε τουλάχιστον ένα εκ των προτύπων, εάν υπάρχει έστω ένας τέτοιος κόμβος στο αντίστοιχο συμμετρικό υποδέντρο με βάθος j. Αυτό βέβαια δεν έρχεται καθόλου σε αντίθεση με τη συμβατική πολιτική του MIDAS που επιβάλλει συγκεκριμένα χαρακτηριστικά μόνο ως προς τα πρώτα j δυφία των αναγνωριστικών των συνδέσμων. Απλά οι κανονικές αυτές εκφράσεις προστίθονται στους έως τώρα περιορισμούς κι ιδιότητες που θα πρέπει να τηρούν οι σύνδεσμοι. Έτσι οδηγούμαστε στην ακόλουθη πολιτική σχηματισμού του δικτύου καθώς αυτό επεκτείνεται. Όταν ένας κόμβος εισάγεται αυτό σημαίνει ότι κάποιος προ-υπάρχων κόμβος μοίρασε τη ζώνη του στα δύο και του ανέθεσε το ένα μισό. Τότε οι δύο παραγόμενοι κόμβοι βρίσκονται στο ίδιο βάθος ως φύλλα κι η ακόλουθη διαδικασία λαμβάνει χώρα:

1. Κανένας από τους δύο ή αμφότεροι νέοι κόμβοι διαθέτουν αναγνωριστικά τα οποία υπακούουν σε τουλάχιστον μία από τις κανονικές εκφράσεις. Τότε οι κόμβοι οι οποίοι ήταν συνδεδεμένοι στον πρόγονο κόμβο θα συσχετίζονται με οποιονδήποτε από τους δύο με ίση πιθανότητα.

Algorithm 27: w.sky-comp(i, j)

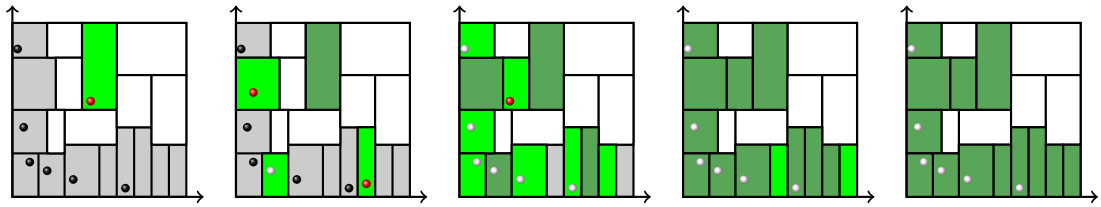
```
1 return d⁻(w.link[i].region, 0⃗) < d⁻(w.link[j].region, 0⃗);
```



(α΄) επίπεδο 1 (β΄) επίπεδο 2 (γ΄) επίπεδο 3 (δ΄) επίπεδο 4 (ε΄) επίπεδο 5

Σχήμα 6.3: Κόμβοι με αναγνωριστικά της μορφής $p_h = (X0)^*X?$ and $p_v = (0X)^*0?$ για δισδιάστατο χώρο.

2. Μόνον ένας από τους δύο κόμβους διαθέτει αναγνωριστικό σύμφωνα με οποιοδήποτε από τα πρότυπα. Τότε όλοι οι κόμβοι που ήταν συνδεδεμένοι με τον αρχικό κόμβο θα συσχετίζονται πλέον με τον κόμβο αυτόν. Έπειτα μόνον ο κόμβος αυτός θα έχει γνώση για τον παραγόμενο κόμβο με το αδιάφορο αναγνωριστικό.



(α΄) έναρξη (β΄) 1ο βήμα (γ΄) 2ο βήμα (δ΄) 3ο βήμα (ε΄) 4ο βήμα

Σχήμα 6.4: Επεξεργασία ερωτημάτων κορυφογραμμής για δύο διαστάσεις με τον αλγόριθμο fast RIPPLE.

Στο Σχήμα 6.4 δείχνουμε με ένα παράδειγμα τον αντίκτυπο της στρατηγικής αυτής στη δομή του MIDAS καθώς και στην επεξεργασία ερωτημάτων κορυφογραμμής σύμφωνα με το RIPPLE για την περίπτωση fast. Με γκρι επισημαίνουμε τους κόμβους των οποίων τα αναγνωριστικά υπακούουν σε κάποιο εκ των δύο προαναφερθέντων προτύπων. Με ανοιχτό πράσινο δείχνουμε τους κόμβους οι οποίοι συμμετέχουν στην επεξεργασία του ερωτήματος για το συγκεκριμένο βήμα (hop), ενώ με σκούρο πράσινο επισημαίνουμε τους κόμβους που έχουν συμμετάσχει μέχρι εκείνη τη στιγμή. Μπορούμε τώρα να παρατηρήσουμε στο παράδειγμα την αποτελεσματικότητα με την οποία το RIPPLE προσπελαύνει τους κόμβους που εμπεριέχουν στοιχεία της απάντησης (μαύρα σημεία που δείχουμε με άσπρο όταν προστίθονται στο αποτέλεσμα). Ομοίως στο Σχήμα 6.5 δείχνουμε την εκτέλεση του slow αλγορίθμου για το ίδιο παράδειγμα.

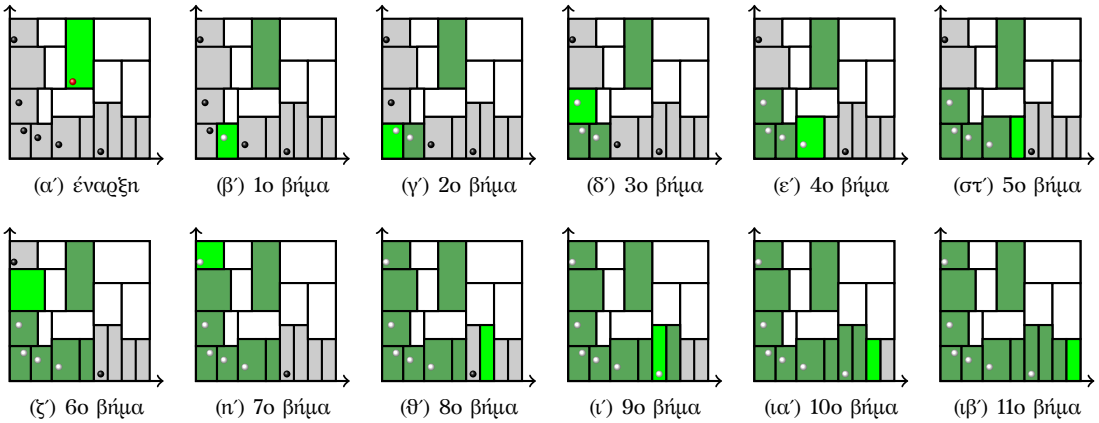
6.4 Αναζήτηση Διαφοροποιημένου Αποτελέσματος

Η Ενότητα 6.4.1 εισάγει απαραίτητες έννοιες σχετικά με το πρόβλημα της διαφοροποίησης αποτελέσματος. Η Ενότητα 6.4.2 παρουσιάζει τον αλγόριθμο βασισμένο στο RIPPLE για ένα σημαντικό υποπρόβλημα στο οποίο βασιζόμαστε στην Ενότητα 6.4.3 για να σχεδιάσουμε μία λύση στο πλήρες πρόβλημα.

6.4.1 Ορισμοί

Δεδομένου ενός ερωτήματος $\vec{q}$, το ερώτημα k-διαφοροποίησης (k-diversification query) αφορά την αναζήτηση του συνόλου O_k από k πλειάδες οι οποίες μεγιστοποιούν την ακόλουθη συνάρτηση:

$$f(O,\vec{q}) = \lambda \max_{\vec{x}\in O} d_r(\vec{x},\vec{q}) - (1-\lambda) \min_{\vec{y},\vec{z}\in O} d_v(\vec{y},\vec{z}). \tag{6.3}$$



Σχήμα 6.5: Επεξεργασία ερωτημάτων κορυφογραμμής για δύο διαστάσεις με τον αλγόριθμο slow RIPPLE.

Το πρώτο μέρος της αντικειμενικής συνάρτησης ορίζεται από τη μέγιστη απόσταση οποιασδήποτε πλειάδας του O από το $\vec{q}$. Μία χαμηλή τιμή υποδεικνύει ότι το σύνολο O αποτελείται από πολύ σχετικές πλειάδες. Το δεύτερο μέρος της αντικειμενικής συνάρτησης ορίζεται από την ελάχιστη απόσταση μεταξύ δύο οποιονδήποτε πλειάδων στο O. Μία υψηλή τιμή υποδεικνύει ότι το σύνολο O εμπεριέχει διαφοροποιημένες πλειάδες. Οι αποστάσεις στην Εξίσωση 6.3 υπολογίζονται μέσω των συναρτήσεων d_r, d_v που ορίζονται από τον χρήστη. Η παράμετρος λ παίρνει τιμές στο διάστημα $[0,1]$ κι ελέγχει τα σχετικά βάρη για τον πρώτο και το δεύτερο παράγοντα που αντιστοιχούν στη σχετικότητα και τη διαφοροποίηση του συνόλου. Η παράμετρος αυτή ορίζεται από τον χρήστη κι ελέγχει το βαθμό με τον οποίο συμβιβάζονται τα δύο διαφορετικά χαρακτηριστικά. Εν γένει, ο στόχος ενός ερωτήματος k-διαφοροποίησης (k-diversification query) αφορά την εύρεση του συνόλου O_k του οποίου τα στοιχεία συμβιβάζουν τη σχετικότητα στο ερώτημα με την αναμεταξύ τους διαφοροποίηση ταυτόχρονα.

Ένα σημαντικό υποπρόβλημα, το οποίο συναντάμε στους περισσότερους αλγορίθμους που προσπαθούν με άπλησστες τεχνικές να λύσουν το πρόβλημα της k-διαφοροποίησης, είναι το ακόλουθο. Δεδομένου ενός ερωτήματος $\vec{q}$ κι ενός συνόλου από αντικείμενα O, το *ερώτημα διαφοροποιημένης πλειάδας* (*single tuple diversification query*) αναζητά την πλειάδα $\vec{t}^* \notin O$ η οποία ελαχιστοποιεί την αντικειμενική συνάρτηση για το σύνολο $O \cup \{\vec{t}^*\}$, δηλαδή,

$$\vec{t}^* = \text{argmin}_{\vec{t} \notin O} f(O \cup \{\vec{t}\}, \vec{q}) \tag{6.4}$$

Διακρίνουμε τέσσερις διαφορετικές περιπτώσεις μελετώντας τον αντίκτυπο του να προσθέσεις μία πλειάδα $\vec{t}$ στο O. Σύμφωνα με την πρώτη, η πλειάδα $\vec{t}$ βρίσκεται εντός της περιοχής που ορίζεται από το λιγότερο σχετικό στοιχείο στο σύνολο O κι επίσης βρίσκεται μακρύτερα από κάθε άλλο στοιχείο του O από ότι η μικρότερη απόσταση μεταξύ οποιονδήποτε στοιχείων του συνόλου. Οπότε η τιμή της f δεν διαφοροποιείται από την εισαγωγή του νέου στοιχείου καθώς τόσο η ελάχιστη σχετικότητα των στοιχείων του συνόλου όσο κι η ελάχιστη διαφοροποίησή τους δεν αλλάζουν.

Όσον αφορά τη δεύτερη περίπτωση, η νέα πλειάδα $\vec{t}$ βρίσκεται μακρύτερα από το ερώτημα $\vec{q}$ συγκριτικά με κάθε άλλο στοιχείο στο O, και μακρύτερα από κάθε στοιχείο του O συγκριτικά με τη μικρότερη απόσταση μεταξύ δύο οποιονδήποτε στοιχείων του O. Συνεπώς, η αντικειμενική τιμή του επαυξημένου συνόλου αυξάνεται κατα τόσο όσο είναι η διαφορά στη σχετικότητα μεταξύ του $\vec{t}$ και της προηγούμενης λιγότερο σχετικής πλειάδας στο O.

Η τρίτη περίπτωση αφορά τις πλειάδες $\vec{t}$ που είναι εγγύτερα στο $\vec{q}$ συγκριτικά με τη λιγότερο σχετική πλειάδα στο O κι η μικρότερη απόστασή της από οποιοδήποτε στοιχείο του O είναι μικρότερη από ότι το κοντινότερο ζεύγος στοιχείων του O. Συνεπώς, η αντικειμενική τιμή της συνάρτησης του επαυξημένου συνόλου αυξάνεται κατά τόσο όσο αντιστοιχεί στην απώλεια διαφοροποίησης εξαιτίας της εισαγωγής του νέου στοιχείου.

Τέλος, εάν το $\vec{t}$ είναι λιγότερο σχετικό από κάθε στοιχείο του O και ταυτόχρονα η απόστασή του από κάθε πλειάδα στο O είναι μικρότερη από το κοντινότερο ζεύγος του O, τότε η τιμή του $f(O \cup \{\vec{t}\})$ αυξάνεται εξαιτίας της απώλειας όσον αφορά τη σχετικότητα του αποτελέσματος αλλά και την απώλεια στη διαφοροποίηση του συνόλου που οφείλεται στην εισαγωγή του $\vec{t}$ στο O.

Λαμβάνοντας τις παρατηρήσεις αυτές υπόψην, δεδομένου ενός συνόλου αντικειμένων O και ενός ερωτήματος $\vec{q}$, ορίζουμε τη συνάρτηση "χρησιμότητας" (utility function) $\phi(\vec{t},\vec{q},O)$ που αφορά μεμωνομένες πλειάδες όπως δείχνουμε στην Εξίσωση 6.5. Οι τέσσερις περιπτώσεις που περιγράψαμε προηγουμένως ανταποκρίνονται στις διακλαδώσεις της συνάρτησης. Είναι εύκολο να δειχτεί ότι η πλειάδα η οποία ελαχιστοποιεί την Εξίσωση 6.5 είναι αυτή που επιλύει το *ερώτημα διαφοροποιημένης πλειάδας* (*single tuple diversification query*) που ορίσαμε προηγουμένως. Επιπλέον, δίνουμε ιδιαίτερη έμφαση στο γεγονός ότι καθίσταται δυνατόν να οριστούν παρόμοιες συναρτήσεις ϕ με διαφορετικά κριτήρια από αυτά που λαμβάνονται υπόψην στην Εξίσωση 6.3, π.χ. για διαφορετικές συναρτήσεις που χρησιμοποιούν αθροίσματα.

$$\phi(\vec{t},\vec{q},O) = \begin{cases} 0, \text{ if } d_r(\vec{t},\vec{q}) \leq \max_{\vec{x}\in O} d_r(\vec{x},\vec{q}) \text{ and } \min_{\vec{x}\in O} d_v(\vec{t},\vec{x}) \geq \min_{\vec{y},\vec{z}\in O} d_v(\vec{y},\vec{z}), \\ \lambda(d_r(\vec{t},\vec{q}) - \max_{\vec{x}\in O} d_r(\vec{x},\vec{q})), \text{ if } d_r(\vec{t},\vec{q}) > \max_{\vec{x}\in O} d_r(\vec{x},\vec{q}) \text{ and } \min_{\vec{x}\in O} d_v(\vec{t},\vec{x}) \geq \min_{\vec{y},\vec{z}\in O} d_v(\vec{y},\vec{z}), \\ (1-\lambda)(\min_{\vec{x},\vec{y}\in O} d_v(\vec{x},\vec{y}) - \min_{\vec{z}\in O} d_v(\vec{t},\vec{z})), \text{ if } d_r(\vec{t},\vec{q}) \leq \max_{\vec{x}\in O} d_r(\vec{x},\vec{q}) \text{ and } \min_{\vec{x}\in O} d_v(\vec{t},\vec{x}) < \min_{\vec{y},\vec{z}\in O} d_v(\vec{y},\vec{z}), \\ \lambda(d_r(\vec{t},\vec{q}) - \max_{\vec{x}\in O} d_r(\vec{x},\vec{q})) + (1-\lambda)(\min_{\vec{y},\vec{z}\in O} d_v(\vec{y},\vec{z}) - \min_{\vec{x}\in O} d_v(\vec{t},\vec{x})), \text{ otherwise.} \end{cases} \tag{6.5}$$

6.4.2 Ανάκτηση Διαφοροποιημένων Πλειάδων

Αυτή η ενότητα περιγράφει την εφαρμογή του RIPPLE για την επίλυση *ερωτημάτων διαφοροποιημένης πλειάδας* (*single tuple diversification query*). Όπως και προηγουμένως, υλοποιούμε τις συναρτήσεις της διεπαφής του RIPPLE. Για το συγκεκριμένο πρόβλημα, το ερώτημα Q εμπεριέχει το σημείο $\vec{q}$, το σύνολο αντικειμένων O, και τη συνάρτηση κατάταξης αντικειμένων ϕ (όπως αυτή εξάγεται από την αντικειμενική συνάρτηση f για σύνολα). Η κατάσταση S εκφράζει ένα όριο διαφοροποίησης τ προς βελτιστοποίηση. Η απάντηση A αποτελεί την πλειάδα $\vec{t}^* \notin O$ που ελαχιστοποιεί τη συνάρτηση κατάταξης.

Πρώτα περιγράφουμε τη μέθοδο computeLocalState που δείχνουμε στον Αλγόριθμο 28, ο οποίος παράγει την τοπική κατάσταση δεδομένου μίας ληφθείσας ολικής κατάστασης. Αρχικά, ο κόμβος w ανακτά την τοπική πλειάδα $\vec{t}$ που ελαχιστοποιεί τη συνάρτηση ϕ (γραμμή 2). Εάν η τιμή της εγγραφής είναι μικρότερη από την ολική κατάσταση/φράγμα τ^G (γραμμή 2), τότε η τοπική κατάσταση αρχικοποιείται στο σκορ της δεδομένης πλειάδας (γραμμή 3). Διαφορετικά, η τοπική κατάσταση παίρνει την τιμή της ολικής κατάστασης (γραμμή 5) υπό την έννοια ότι καμία πλειάδα είναι καλύτερη από αυτή που έχει ήδη ανακτηθεί. Σε κάθε περίπτωση, η μέθοδος computeGlobalState, που δείχνουμε στον Αλγόριθμο 29, ορίζει την ολική κατάσταση στον κόμβο w σύμφωνα με την τοπική κατάσταση.

Algorithm 28: w.div-computeLocalState($\vec{q}, O, \phi, \tau^G$)

```
1 t⃗ ← w.getMostDiverseLocalObject(q⃗, O, φ) ;
2 if φ(t⃗, q⃗, O) < τ^G then
3 |   return τ_w^L ← φ(t⃗, q⃗, O) ;
4 else
5 |   return τ_w^L ← τ^G ;
```

Algorithm 29: w.div-computeGlobalState($\vec{q}, O, \phi, \tau^G, \tau_w^L$)

```
1 return τ_w^G ← τ_w^L ;
```

Η μέθοδος computeLocalAnswer του Αλγορίθμου 30 εξάγει την τοπική πλειάδα η οποία αποτελεί την καλύτερη απάντηση, εφόσον υπάρχει. Αρχικά ο κόμβος w βρίσκει την τοπική πλειάδα $\vec{t}$ με το μικρότερο σκορ (γραμμή 1). Επακόλουθο ότι εάν το σκορ της πλειάδας ισούται με την τοπική κατάσταση (γραμμή 2), η πλειάδα $\vec{t}$ γίνεται η τοπική απάντηση (γραμμή 3). Διαφορετικά, η τοπική απάντηση αφήνεται κενή (γραμμή 5).

Algorithm 30: w.div-computeLocalAnswer($\vec{q}, O, \phi, \tau_w^L$)

```
1 t⃗ ← w.getMostDiverseLocalObject(q⃗, O, φ) ;
2 if φ(t⃗, q⃗, O) = τ_w^L then
3 |   return A ← t⃗ ;
4 else
5 |   return A ← null ;
```

Ο Αλγόριθμος 31 ενημερώνει την τοπική κατάσταση σύμφωνα με το σύνολο των τοπικών καταστάσεων που ελήφθησαν. Ο κόμβος w απλά ορίζει την τοπική κατάσταση ανάλογα με την ελάχιστη από εκείνες που ελήφθησαν. Ο Αλγόριθμος 32 αποφασίζει για το εάν θα πρέπει να προχωρήσει η αναζήτηση στην περιοχή που αντιστοιχεί στον i-ιοστό σύνδεσμο του κόμβου w. Η απόφαση εξαρτάται από το εάν το κάτω φράγμα για το σκορ που θα μπορούσε να επιτύχει η καλύτερη πλειάδα στη συγκεκριμένη περιοχή είναι χαμηλότερο από την ολική κατάσταση. Η συνάρτηση ϕ^- καθορίζει το φράγμα αυτό. Τέλος, ο Αλγόριθμος 33 συγκρίνει τις προτεραιότητες των συνδέσμων του w. Οπότε κι ο σύνδεσμος του οποίου η περιοχή είναι η πιο υποσχόμενη ως προς τη ϕ^- έχει την υψηλότερη προτεραιότητα.

Algorithm 31: w.div-updateLocalState($\vec{q}, O, \phi, \{\tau_i^L\}$)

```
1 return τ_w^L ← min_i{τ_i^L} ;
```

Algorithm 32: w.div-isLinkRelevant($i, \vec{q}, O, \phi, \tau_w^G$)

```
1 return φ⁻(w.link[i].region, q⃗, O) < τ_w^G ;
```

6.4.3 Σχηματισμός Διαφοροποιημένων Συνόλων

Χρησιμοποιώντας τον αλγόριθμο για την επεξεργασία *ερωτημάτων διαφοροποιημένων πλειάδων* (*single tuple diversification query*), που περιγράψαμε στην προηγούμενη ενότητα, προτείνουμε ένα άπλησστο αλγόριθμο για την επεξεργασία ερωτημάτων k-διαφοροποίησης (k-diversification query).

Ο Αλγόριθμος 34 δείχνει τον ψευδοκώδικα που εκτελεί ο εκδότης ενός ερωτήματος, έστω κόμβος v. Αρχικά, ένα σύνολο k πλειάδων ανακτάται από το δίκτυο εκτελώντας τη μέθοδο initialize (γραμμή 1). Η συνάρτηση αυτή μπορεί είτε να δημιουργεί ένα σύνολο από k τυχαίες πλειάδες, είτε να γεμίζει το σύνολο της απάντησης σταδιακά κάθε φορά με το να εκτελεί τον div-ripple αλγόριθμο επεξεργασίας *ερωτημάτων διαφοροποιημένων πλειάδων* (*single tuple diversification query*) k φορές.

Τότε, δεδομένου του αρχικού συνόλου O από k εγγραφές, ο αλγόριθμος επιχειρεί να βελτιώσει την αντικειμενική τιμή του συνόλου σταδιακά (γραμμές 2–5). Κάθε πέρασμα καλεί τη μέθοδο div-improve για να αποκτήσουμε ένα νέο, βελτιωμένο σύνολο (γραμμή 3). Οι επαναλήψεις του αλγορίθμου σταματάνε όταν ο αλγόριθμος div-improve δεν μπορεί να βελτιώσει περαιτέρω το τρέχον αποτέλεσμα (γραμμή 7).

Η μέθοδος div-improve του Αλγορίθμου 35 αποσκοπεί στο να επιλέξει το καλύτερο ζεύγος $\vec{t}_{out}, \vec{t}_{in}$, όπου $\vec{t}_{out} \in O$ το στοιχείο της απάντησης το οποίο αν αντικατασταθεί με το $\vec{t}_{in} \notin O$, η απάντηση θα αποκτήσει καλύτερα χαρακτηριστικά από ότι θα είχε αν αντικαταστούσαμε

Algorithm 33: w.div-comp($i, j, \vec{q}, O, \phi$)

```
1 return φ⁻(w.link[i].region, q, O) < φ⁻(w.link[j].region, q, O) ;
```

Algorithm 34: v.diversify($\vec{q}, k$)

```
1 O ← v.initialize(q, k);
2 for i ← 1 to MAX_ITERS do
3 |   O' ← v.div-improve(q, O);
4 |   if O' ≠ O then
5 |   |   O ← O';
6 |   else
7 |   |   break;
```

οποιοδήποτε άλλο στοιχείο του O με κάποιο άλλο αντικείμενο αποθηκευμένο σε κάποιον κόμβο του δικτύου.

Η μέθοδος div-improve αρχικά θα ταξινομήσει σε φθίνουσα σειρά τα στοιχεία του O (γραμμή 3) σύμφωνα με την τιμή που παίρνει η κάθε πλειάδα $\vec{t}_i \in O$ για τη συνάρτηση $\phi(\vec{t}_i, q, O \setminus \{\vec{t}_i\})$ που υποδηλώνει την ποιότητα του υποσυνόλου του O χωρίς το συγκεκριμένο στοιχείο. Παρατηρούμε επίσης ότι η ταξινόμηση αυτή υπονοεί παράλληλα την ταξινόμηση των υποσυνόλων $O \setminus \{\vec{t}_i\}$ σε αύξουσα σειρά ως προς τις αντικειμενικές τους τιμές με βάσει τη συνάρτηση κατάταξης που έχει επιλεγεί. Οπότε, η πρώτη πλειάδα στη ταξινομημένη λίστα θα έχει το χειρότερο σκορ αλλά το σύνολο που σχηματίζεται όταν το αφαιρούμε θα έχει τα καλύτερα χαρακτηριστικά. Πιο αναλυτικά, ας υποθέσουμε ότι το $\vec{t}_i$ προηγείται του $\vec{t}_j$, αν $\phi(\vec{t}_i, \vec{q}, O \setminus \{\vec{t}_i\}) \geq \phi(\vec{t}_j, \vec{q}, O \setminus \{\vec{t}_j\})$ τότε οδηγούμαστε:

$$
\begin{aligned}
f(O, \vec{q}) = f(O, \vec{q}) \quad &\Leftrightarrow \\
f(O \setminus \{\vec{t}_i\} \cup \{\vec{t}_i\}, \vec{q}) = f(O \setminus \{\vec{t}_j\} \cup \{\vec{t}_j\}, \vec{q}) \quad &\Leftrightarrow \\
f(O \setminus \{\vec{t}_i\}, \vec{q}) + \phi(\vec{t}_i, \vec{q}, O \setminus \{\vec{t}_i\}) = & \\
f(O \setminus \{\vec{t}_j\}, \vec{q}) + \phi(\vec{t}_j, \vec{q}, O \setminus \{\vec{t}_j\}) \quad &\Leftrightarrow \\
f(O \setminus \{\vec{t}_i\}, \vec{q}) \leq f(O \setminus \{\vec{t}_j\}, \vec{q}). &
\end{aligned}
$$

Συνεπώς το $O \setminus \{\vec{t}_i\}$ επιτυγχάνει καλύτερη τιμή. Γενικά βασιζόμαστε στο ότι με το να επεξεργαστούμε τα καλά υποσύνολα νωρίς, θα καταλήξουμε γρήγορα σε ένα ακόμη καλύτερο σύνολο μετά την βέλτιστη αντικατάσταση από ότι θα καταλήγαμε με κάποια άλλη επιλογή.

Algorithm 35: v.div-improve($\vec{q}, O$)

```
1  t_in ← null;
2  t_out ← null;
3  sort tuples in O descending on their φ scores;
4  for each t_i ∈ O do
5  |   if t_in = null then
6  |   |   τ ← φ(t_i, q, O);
7  |   else
8  |   |   τ ← f(O \ {t_out} ∪ {t_in}, q) − f(O, q);
9  |   v.div-ripple(v, v, q, O \ t_i, τ, R, r);
10 |   t_in ← v.receive();
11 |   if t_in ≠ null then
12 |   |   t_out ← t_i;
13 return O \ {t_out} ∪ {t_in};
```

Ο Αλγόριθμος div-improve εξετάζει ένα προς ένα τα ταξινομημένα στοιχεία (γραμμές 4–12). Εν συντομία, κάθε φορά εξετάζουμε την περίπτωση του να αφαιρέσουμε το υπο εξέταση στοιχείο του O κι αναζητούμε το στοιχείο εκτός του O που αποτελεί τον καλύτερο αντικαταστάτη του. Ο αλγόριθμος απαιτεί όμως από το νέο υποδιαμόρφωση σύνολο να επιτυγχάνει καλύτερο σκορ τόσο από αυτό του αρχικού συνόλου όσο και της προηγούμενης αντικατάστασης.

Προκειμένου να βρούμε τον καλύτερο αντικαταστάτη της πλειάδας $\vec{t}_i$, η μέθοδος div-improve καλεί τον αλγόριθμο div-ripple της προηγούμενης ενότητας δεδομένου όμως του υποσυνόλου $O \setminus \{\vec{t}_i\}$ (γραμμή 9). Σημαντική είναι κι η παράμετρος τ της ολικής κατάστασης. Αρχικά, η παράμετρος αυτή παίρνει μία ουδέτερη τιμή, όπως $+\infty$. Όμως στη συνέχεια η τιμή μειώνεται σταδιακά ώστε να επιβάλλει την αναζήτηση για αντικαταστάσεις που θα αποτελέσουν μία καλύτερη λύση από την προηγούμενη. Όταν καμία τέτοια πλειάδα δεν έχει βρεθεί ακόμα (γραμμή 5), η ολική κατάσταση παίρνει την τιμή που αντιστοιχεί στο ϕ σκορ της πλειάδας $\vec{t}_i$ (γραμμή 6). Αυτό καθιστά την αναζήτηση του div-ripple αρχικά για καλύτερα σύνολα του αρχικού. Εάν ο αλγόριθμος έχει ήδη βρει μία πλειάδα $\vec{t}_{out}$ για να αντικαταστήσει με την $\vec{t}_{in}$, η ολική κατάσταση ορίζεται από την αντικειμενική τιμή του βελτιωμένου συνόλου μείον την αντικειμενική τιμή του αρχικού, δηλαδή τη διαφορά στην ποιότητα των δύο συνόλων (γραμμή 8). Διαισθητικά αναζητούμε πλειάδες που βελτιώνουν ακόμα περαιτέρω το αποτέλεσμα των προηγούμενων αντικαταστάσεων. Διαφορετικά, αν δεν υπάρχουν τέτοιες πλειάδες επιστρέφεται το βελτιωμένο σύνολο βάσει τα τρέχοντα στοιχεία προς αντικατάσταση. Αρχικοποιώντας την ολική κατάσταση με αυτόν τον τρόπο βελτιστοποιούμε την αναζήτηση με το να παραλείπουμε μεγάλα μη βέλτιστα κομμάτια του χώρου που δεν μπορούν να συνεισφέρουν άλλωστε στο αποτέλεσμα. Αυτό μεταφράζεται αυτόματα σε λιγότερο χρόνο αναζήτησης και εξοικονόμιση πόρων του δικτύου.

6.5 Πειραματική Αποτίμηση

Προκειμένου να αξιολογήσουμε τις μεθόδους μας διεξάγουμε εκτενή πειράματα προσομειώνοντας δυναμικές τοπολογίες δικτύου και μελετάμε τη συμπεριφορά σημαντικών μεγεθών.

6.5.1 Μεθοδολογία

Για κάθε τύπο ερωτήματος συγκρίνουμε τις μεθόδους μας με εδραιωμένες λύσεις από τη βιβλιογραφία. Όσον αφορά τα ερωτήματα κορυφογραμμής (skyline queries) συγκρίνουμε την απόδοση της μεθόδου μας ως προς το DSL [127] που βασίζεται στο CAN [99], και το SSP [120] που προσαρμόζει μία Z-curve στο BATON [74]. Για τη διαφοροποίηση αποτελέσματος προσαρμόζουμε τον αλγόριθμο από το [85] για το CAN. Αμφότερες μέθοδοι βασίζονται σε παρόμοιες αρχές αν και η baseline δεν χρησιμοποιεί τις βελτιστοποιήσεις που προτείνουμε στο παρόν. Οι δύο μέθοδοι παράγουν ίδιας ποιότητας απαντήσεις κι αυτό μας διευκολύνει στο να συγκρίνουμε την απόκριση και το επικοινωνιακό κόστος της κάθε μίας.

6.5.2 Τοπολογία Δικτύου

Προσομοιώνουμε μία δυναμική τοπολογία όπου ομότιμοι κόμβοι εισέρχονται κι απέρχονται αυθαίρετα από το δίκτυο σε δύο διακριτά στάδια. Στην πρώτη φάση έχουμε συνεχόμενες εισαγωγές κόμβων όπου ξεκινώντας από ένα δίκτυο 1024 κόμβων καταλήγουμε σε 131072 κόμβους. Στη δεύτερη φάση επιλέγουμε τυχαία κόμβους προς αποχώρηση διαδοχικά έως ότου το δίκτυο να φθάσει και πάλι στο αρχικό του μέγεθος των 1024 κόμβων. Καθώς μεταβάλλουμε το μέγεθος του δικτύου δείχνουμε μόνο τη πρώτη φάση ενώ τα αποτελέσματα της δεύτερης είναι ανάλογα και παραλείπονται.

Παράμετρος	Εύρος	Προκαθορισμένη τιμή
Μέγεθος δικτύου n	$2^{10}, 2^{11}, 2^{12}, 2^{13}, 2^{14}, 2^{15}, 2^{16}, 2^{17}$	2^{14}
Αριθμός διαστάσεων d	$2, 3, 4, 5, 6, 7, 8, 9, 10$	$5, 6$
Μέγεθος αποτελέσματος k	10,20,30,40,50,60,70,80,90,100	10
Παράγοντας εξισορρόπησης λ	$0, 0.2, 0.3, 0.5, 0.7, 0.8, 1$	0.5

Πίνακας 6.2: Παράμετροι πειραματικής αποτίμησης.

6.5.3 Παράμετροι

Θα εξετάσουμε τέσσερις κύριες παραμέτρους. Το μέγεθος του δικτύου που μεταβάλλεται από 1024 κόμβους ως τους 131072. Τον αριθμό των διαστάσεων του χώρου από 2 έως τις 10 που δεικτοδοτούν τα ευρετήρια στα οποία εκτελούνται οι διάφορες μέθοδοι για κάθε τύπο ερωτήματος. Επίσης μελετάμε τον αντίκτυπο του αναμενόμενου μεγέθους του αποτελέσματος k μεταξύ 10 και 100 για τα top-k ερωτήματα και τη διαφοροποίηση αποτελέσματος. Ειδικά για τον τελευταίο τύπο ερωτήματος μελετάμε επιπλέον το πως επηρεάζει την απόδοση η παράμετρος λ για την εξισορρόπηση μεταξύ σχετικότητας των στοιχείων του αποτελέσματος και της αναμεταξύ τους διαφοροποίησης. Πιο συγκεκριμένα το λ όπως χρησιμοποιείται στην Εξίσωση 6.5 παίρνει τιμές στο διάστημα $[0, 1]$. Το εύρος της κάθε παραμέτρου παρατίθεται στον Πίνακα 6.2 παράλληλα με τις αντίστοιχες προκαθορισμένες τιμές. Όταν μεταβάλλουμε μία παράμετρο οι υπόλοιπες τίθονται ανάλογα με τις προκαθορισμένες τιμές.

6.5.4 Μετρικές

Όσον αφορά την απόδοση των μεθόδων της επεξεργασίας των ερωτημάτων χρησιμοποιούμε κυρίως δύο μετρικές. Πρώτον, το latency μετράει τον αριθμό των βημάτων κι επαναπροωθήσεων που απαιτούνται στα πλαίσια της επεξεργασίας. Επιπλέον, η κατανεμημένη επεξεργασία των ερωτημάτων επιβάλλει το επιμερισμό του υπολογιστικού φορτίου σε ένα σύνολο από κόμβους του δικτύου που επιβαρύνονται με τη μερική επεξεργασία ή δρομολόγηση των ερωτημάτων. Οπότε κρίνουμε ότι είναι απαραίτητη η μελέτη της συμφόρησης (congestion) του δικτύου που ορίζεται ως ο μέσος αριθμός από ερωτήματα που επεξεργάζονται σε κάθε κόμβο όταν n τυχαία ερωτήματα εκδίδονται από κόμβους που επιλέγουμε τυχαία κι ανεξάρτητα. Υπό μία έννοια αυτή η μετρική αντικατοπτρίζει την κίνηση που δέχεται ένας κόμβος.

6.5.5 Δεδομένα κι Ερωτήματα

Για την αποτίμηση των μεθόδων για τα top-k ερωτήματα και τα ερωτήματα κορυφογραμμής χρησιμοποιούμε στατιστικά δεδομένα για τους παίκτες του NBA[3] που αποτελούνται από 22000 εγγραφές έξι διαστάσεων και καλύπτουν τις σεζόν από το 1946 ως το 2009. Οι εγγραφές αυτές αποτελούνται από στοιχεία όπως rebounds, assists και blocks ανά παιχνίδι. Οπότε έχουμε ότι ένα top-k ερώτημα ανακτά τους καλύτερους συνολικά παίκτες καθώς οι στατιστικές τους συμψηφίζονται με τη χρήση μίας αθροιστικής συνάρτησης βαρών για κάθε διάσταση. Ένα ερώτημα κορυφογραμμής ανακτά τους παίκτες οι οποίοι είναι κορυφαίοι ως προς οποιοδήποτε συνδυασμό στατιστικών χαρακτηριστικών.

Για τα ερωτήματα διαφοροποίησης χρησιμοποιούμε τη MIRFLICKR συλλογή 1000000 εικόνων που συναντάμε συχνά στην αξιολόγηση μεθόδων ανάκτησης εικόνων[4] βάσει του περιεχομένου τους. Εξάγουμε ένα πενταδιάστατο ιστόγραμμα το οποίο είναι συμβατό με το MPEG-7 πρότυπο πάνω στο οποίο χρησιμοποιούμε τη L_1 απόσταση για τον υπολογισμό της ομοιότητας με το ερώτημα και τη διαφοροποίηση των στοιχείων του αποτελέσματος αναμεταξύ τους.

Προκειμένου να μελετήσουμε τον αντίκτυπο του αριθμού των διαστάσεων για όλους τους τύπους ερωτημάτων κατασκευάζουμε 1000000 πολυδιάστατες εγγραφές στο διάστημα $[0, 1]^d$,

[3] διαθέσιμα on-line στο `http://www.basketball-reference.com`
[4] διαθέσιμα στο `http://press.liacs.nl/mirflickr/`

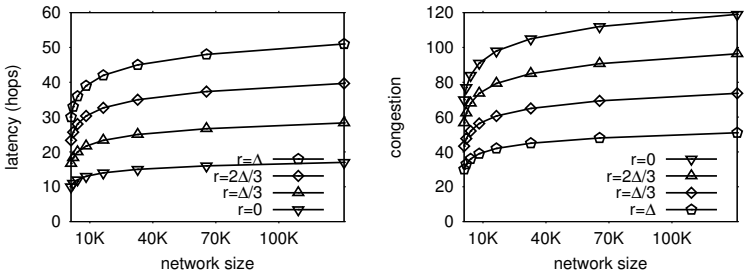
με το d να παίρνει τιμές από το 2 ως το 10, και δείχνουμε στα γραφήματα ως SYNTH. Τα δεδομένα δημιουργήθηκαν γύρω από 50000 κέντρα σύμφωνα με τη zipfian κατανομή με τη παράμετρο $\sigma = 0.1$ (skewness factor). Οι τιμές στα γραφήματα αποτελούν τις μέσες τιμές των μετρικών μετά την επεξεργασία 65536 ερωτημάτων που εκδόθηκα σε 16 διαφορετικές τοπολογίες.

6.5.6 Αποτελέσματα

Top-k Ερωτήματα

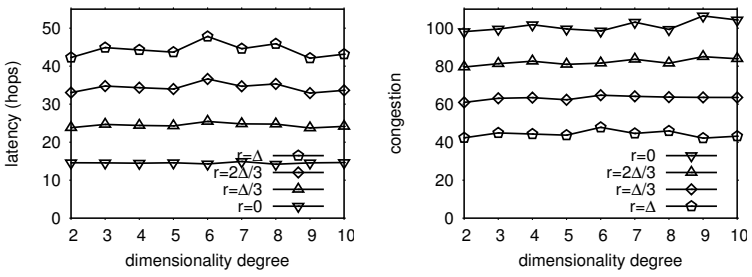
Στην ενότητα αυτή μελετάμε την επίδραση της ripple παραμέτρου r στα top-k ερωτήματα. Ειδικότερα εξετάζουμε τέσσερις τιμές της παραμέτρου: τις δύο ακραίες, 0 όπου το RIPPLE εκτελεί τον αλγόριθμο fast, και Δ όπου εκτελείται ο αλγόριθμος slow, καθώς και δύο ενδιάμεσες τιμές, $\Delta/3$, $2\Delta/3$.

Χρησιμοποιούμε στις προσομειώσεις τα στατιστικά δεδομένα του NBA και παρουσιάζουμε τα αντίστοιχα αποτελέσματα στα Σχήματα 6.6 και 6.8, ενώ τα αποτελέσματα για το συνθετικό



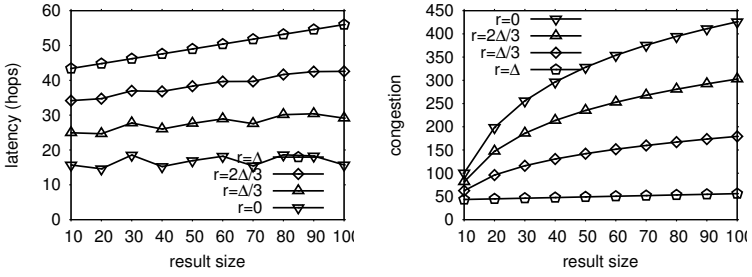
(α′) αριθμός επαναπροωθήσεων (β′) συμφόρηση δικτύου

Σχήμα 6.6: Απόδοση top-k ερωτημάτων ως προς το μέγεθος του δικτύου.



(α′) αριθμός επαναπροωθήσεων (β′) συμφόρηση δικτύου

Σχήμα 6.7: Απόδοση top-k ερωτημάτων ως προς τον αριθμό των διαστάσεων.



(α′) αριθμός επαναπροωθήσεων (β′) συμφόρηση δικτύου

Σχήμα 6.8: Απόδοση top-k ερωτημάτων ως προς το μέγεθος του αποτελέσματος.

πολυδιάστατο dataset SYNTH φαίνονται στο Σχήμα 6.7. Όπως αναμενόταν για χαμηλές τιμές του r κοντά στο 0 έχουμε γρήγορη απόκριση αλλά για μεγαλύτερο επικοινωνιακό κόστος. Για μεγάλες όμως τιμές του r (κοντά στο μέγιστο αριθμό συνδέσμων ανά κόμβων Δ) το επικοινωνιακό κόστος ελαχιστοποιείται καθώς επιβαρύνονται σχεδόν αποκλειστικά σχετικοί κόμβοι στα πλαίσια της επεξεργασίας του ερωτήματος.

Στο Σχήμα 6.6(α΄) το latency κλιμακώνεται καλά καθώς μεγαλώνει ο αριθμός των κόμβων στο δίκτυο. Ακόμα και για υψηλές τιμές του r κοντά στο Δ, ο μέγιστος αριθμός επαναπροωθήσεων στο RIPPLE είναι αρκετά χαμηλότερος από την γραμμική πρόβλεψη της θεωρητικής ανάλυσης για την χειρότερη περίπτωση εξαιτίας της προτεραιότητας με την οποία επισκεπτόμαστε τους κόμβους στο RIPPLE και επιδεικνύει μία πολυλογαριθμική συμπεριφορά. Η αυξημένη συμφόρηση του δικτύου στο Σχήμα 6.6(β΄) για χαμηλές τιμές του r εξηγείται από την παράλληλη μετάδοση των ερωτημάτων στο δίκτυο και τις πολλαπλές διαδρομές που αναπτύσονται.

Η διαστασιμότητα επηρεάζει ελαφρώς την απόδοση όπως δείχνουμε και στο Σχήμα 6.7. Ο λόγος βρίσκεται στο ότι η δομή (αριθμός συνδέσμος ανά κόμβο, μέγεθος δικτύου) της δικτυακής υποδομής που χρησιμοποιείται (για την περίπτωσή μας το MIDAS) θα καθορίσει με τα χαρακτηριστικά της και την απόδοση των μεθόδων επεξεργασίας των ερωτημάτων.

Τέλος, στο Σχήμα 6.8 βλέπουμε ότι η ανάκτηση μεγαλύτερων αποτελεσμάτων έχει αρνητικό αντίκτυπο στην απόδοση καθώς τόσο οι σχετικοί κόμβοι, όσο κι ο συνολικός αριθμός από κόμβους που προσπελαύνονται αυξάνεται. Για παράδειγμα, όταν $k = 100$ προσπελαύνεται σε αναλογία περίπου το 0.5% των δεδομένων.

Ερωτήματα Κορυφογραμμής

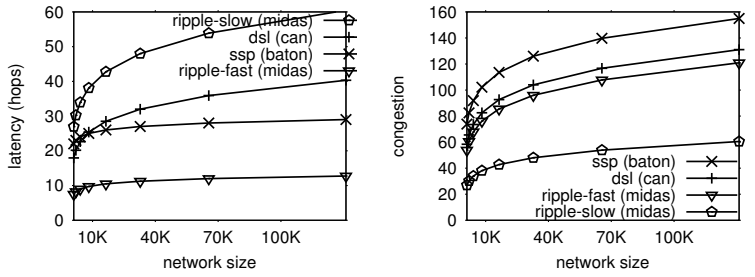
Για το υπόλοιπο της πειραματικής μας αποτίμησης δείχνουμε στα γραφήματα τη συμπεριφορά του RIPPLE μόνο για τις ακραίες τιμές της παραμέτρου r που καθορίζει και τον τρόπο που επεξεργάζονται τα ερωτήματα. Έτσι, συμβολίζουμε ως ripple-fast την περίπτωση όπου $r = 0$, κι ως ripple-slow την περίπτωση όπου $r = \Delta$. Ο αριθμός επαναπροωθήσεων κι η συμφόρηση του δικτύου για άλλες τιμές του r βρίσκεται πάντα μεταξύ των δύο αυτών περιπτώσεων, όπως δείξαμε και για τα top-k ερωτήματα.

Στα Σχήματα 6.9 και 6.10 δείχνουμε την απόδοση των μεθόδων επεξεργασίας ερωτημάτων κορυφογραμμής για τα στατιστικά του NBA και τα συνθετικά δεδομένα. Στο Σχήμα 6.9(α΄) το latency επιδεικνύει λογαριθμική συμπεριφορά για τις μεθόδους ripple-slow και SSP λόγω των χαρακτηριστικών των ευρετηρίων στα οποία βασίζονται. Το RIPPLE στηρίζεται στο εγγενώς πολυδιάστατο MIDAS ενώ το SSP εφαρμόζει μία καμπύλη πληρώσεως του χώρου στο μονοδιάστατο BATON. Κατά συνέπεια προσπελαύνονται περισσότεροι κόμβοι καθώς η δρομολόγηση δεν γίνεται με τόσο αποδοτικό τρόπο για μεγάλο αριθμό διαστάσεων, κάτι που έχει αντίτιμο στην απόκριση και το επικοινωνιακό κόστος της μεθόδου.

Στο Σχήμα 6.9(α΄), το DSL φαίνεται να είναι πιο αργό καθώς τα μηνύματα προωθούνται στους κόμβους που είναι υπεύθυνοι για τα γειτονικά κλειδιά. Όμως, όπως φαίνεται και στο Σχήμα 6.10 είναι σε θέση να εκμεταλευτεί τον αυξανόμενο αριθμό διαστάσεων ελαττώνοντας τη διάμετρο (μέγιστη διαδρομή/μονοπάτι από διαφορετικούς κόμβους) του δικτύου καθώς κάθε κόμβος συνάπτει μεγαλύτερο αριθμό από συνδέσμους που είναι ευθέως ανάλογος του αριθμού των διαστάσεων. Υπό μία έννοια, οι μεγαλύτερες γειτονιές μεταφράζονται στην πράξη σε πιο αποτελεσματική δρομολόγηση καθώς τα ερωτήματα κατευθύνονται σε ιδιαίτερα σχετικούς κόμβους που επιλέγονται από μεγαλύτερη γκάμα συνδέσμων. Η σχεδιαστική αυτή επιλογή όμως έχει και τα αρνητικά της καθώς λειτουργίες του δικτύου όπως οι εισαγωγές κι οι αποχωρήσεις κόμβων αλλά κι ο εντοπισμός απρόσμενων απωλειών κόμβων έχουν μεγαλύτερο επικοινωνιακό κόστος. Σε κάθε περίπτωση όμως η μέθοδος αυτή είναι ξεκάθαρα ακατάλληλη για δεδομένα λίγων χαρακτηριστικών, π.χ. χωρικά, καθώς latency και congestion είναι αρκετά φτωχά στο Σχήμα 6.10.

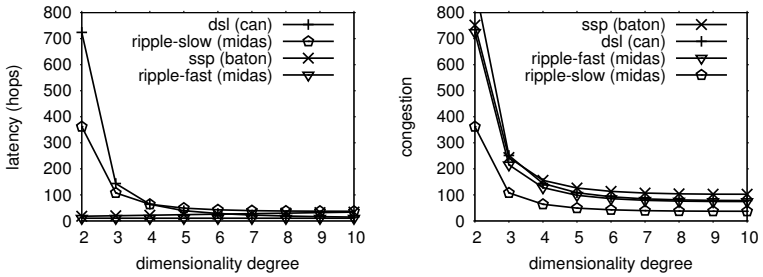
Η μέθοδος ripple-slow αν κι είναι η πιο αργή, καταναλώνει τους λιγότερους πόρους στα Σχήματα 6.9(β΄) και 6.10(β΄). Παρ' όλα αυτά δεν αποδίδει καλά για μικρό αριθμό διαστάσεων ως προς το latency εξαιτίας της σειριακής προσπέλασης ενός μεγάλου αριθμού από σχετικούς κόμβους καθώς το μέγεθος του δικτύου παραμένει σταθερό στο Σχήμα 6.10. Στη θετική πλευρά το ότι αποδίδει καλύτερα από το γραμμικό κόστος που προβλέπεται από

τη θεωρητική ανάλυση εξαιτίας της σειριακής προσπέλασης των κόμβων σύμφωνα με μία συνάρτηση δυναμικού που αναπαριστά την πιθανότητα ενός κόμβου στο να συνεισφέρει στο αποτέλεσμα.



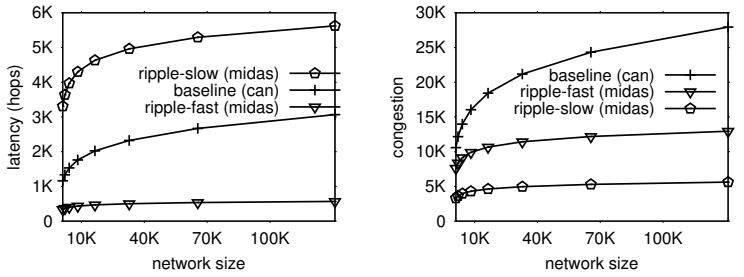
(α΄) αριθμός επαναπροωθήσεων (β΄) συμφόρηση δικτύου

Σχήμα 6.9: Απόδοση ερωτημάτων κορυφογραμμής ως προς το μέγεθος του δικτύου.



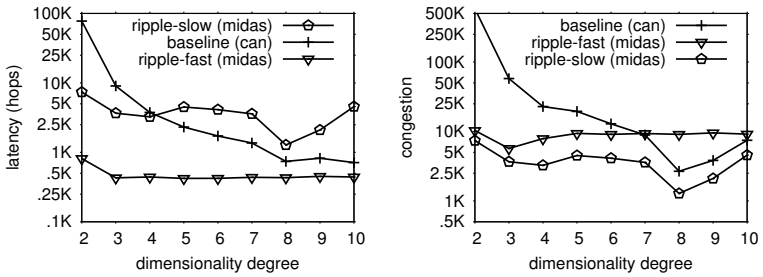
(α΄) αριθμός επαναπροωθήσεων (β΄) συμφόρηση δικτύου

Σχήμα 6.10: Απόδοση ερωτημάτων κορυφογραμμής ως προς τον αριθμό των διαστάσεων.



(α΄) αριθμός επαναπροωθήσεων (β΄) συμφόρηση δικτύου

Σχήμα 6.11: Απόδοση διαφοροποίησης αποτελέσματος ως προς το μέγεθος του δικτύου.



(α΄) αριθμός επαναπροωθήσεων (β΄) συμφόρηση δικτύου

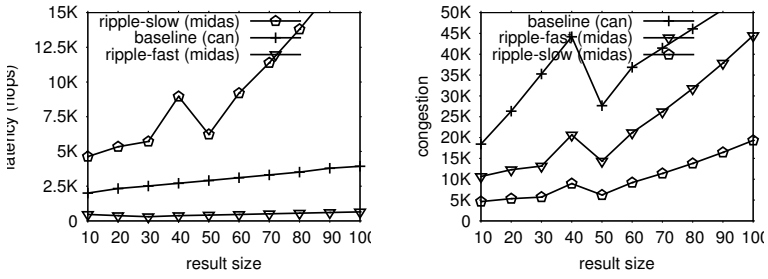
Σχήμα 6.12: Απόδοση διαφοροποίησης αποτελέσματος ως προς τον αριθμό των διαστάσεων.

Εν γένει, η συμφόρηση (congestion) του δικτύου φαίνεται υψηλή για όλες τις μεθόδους. Οι λειτουργίες αυτές είναι φαινομενικά ακριβές αλλά αυτό οφείλεται στον μεγάλο αριθμό σχετικών κόμβων για του συγκεκριμένου τύπου ερώτημα. Προσεγγιστικά υπάρχουν $d\sqrt[d]{n}+m$ σχετικοί κόμβοι, όπου n το μέγεθος του δικτύου, d ο αριθμός των διαστάσεων και m ο αριθμός των σχετικών κόμβων που δεν βρίσκονται στα όρια του χώρου του ευρετηρίου, είτε προσπελαύνονται ως false positives, π.χ. κατά τη διάρκεια των πρώτων σταδίων του αλγορίθμου δεν μπόρεσαν να φιλτραριστούν ως μία σχετικά καλή λύση για τη δεδομένη στιγμή. Συνεπώς, η κύρια πρόκληση στην κατανεμημένη επεξεργασία των ερωτημάτων κορυφογραμμής έγκειται στον τρόπο προσπέλασης του μεγάλου αριθμού σχετικών κόμβων ώστε η απόκριση και το επικοινωνιακό κόστος σε μηνύματα να παραμείνουν σε χαμηλά επίπεδα. Κατ' ουσίαν προτείνουμε μία εργαλειοθήκη για την επεξεργασία ερωτημάτων κορυφογραμμής όπου η εξισορρόπηση latency-congestion είναι παραμετροποιήσιμη και κυμαίνεται από την λογαριθμικού latency ως προς το μέγεθος του δικτύου μέθοδο ripple-fast ως την μέθοδο ripple-slow.

Ερωτήματα k-Διαφοροποίησης

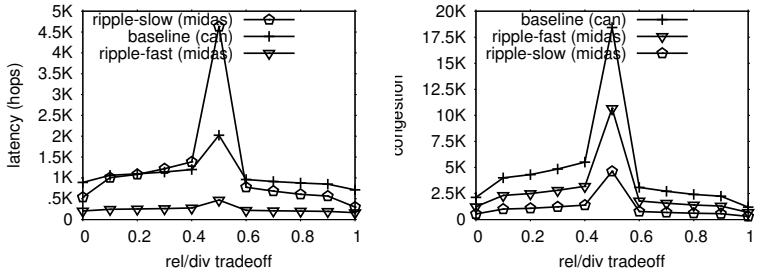
Για το τελευταίο μέρος της πειραματικής μας αποτίμησης θα συγκρίνουμε τη μέθοδο που προτείνεται στα πλαίσια του RIPPLE ως προς τη προσαρμοσμένη baseline μέθοδο. Όπως και προηγουμένως δείχνουμε τη συμπεριφορά του RIPPLE για τις ακραίες τιμές της παραμέτρου r, ήτοι 0 και Δ που αντιστοιχούν στην εκτέλεση της ripple-fast και ripple-slow μεθόδου αντίστοιχα. Το Σχήμα 6.11 παρουσιάζει τα αποτελέσματα για το MIRFLICKR ως προς το μέγεθος του δικτύου. Το Σχήμα 6.12 δείχνει τα αποτελέσματα για τα συνθετικά πολυδιάστατα δεδομένα ενώ στο Σχήμα 6.13 μεταβάλλεται το μέγεθος του αποτελέσματος για το MIRFLICKR. Στα ίδια δεδομένα μελετάμε τη λ παράμετρο εξισορρόπησης μεταξύ σχετικότητας και διαφοροποίησης, όπως δείχνουμε στο Σχήμα 6.14.

Στα Σχήματα 6.11(α΄) και 6.12(α΄) δεν γίνεται να μην προσέξουμε την πολύ γρήγορη απόκριση του ripple-fast για κάθε μέγεθος δικτύου κι αριθμό διαστάσεων. Επιπροσθέτως, τα



(α΄) αριθμός επαναπροωθήσεων (β΄) συμφόρηση δικτύου

Σχήμα 6.13: Απόδοση διαφοροποίησης αποτελέσματος ως προς το μέγεθος του αποτελέσματος.



(α΄) αριθμός επαναπροωθήσεων (β΄) συμφόρηση δικτύου

Σχήμα 6.14: Απόδοση διαφοροποίησης αποτελέσματος ως προς τον παράγοντα εξισορρόπησης/συμβιβασμού λ.

ωφέλη του RIPPLE είναι εμφανή και στο Σχήμα 6.11(β΄) όπου η συμφόρηση στο δίκτυο είναι χαμηλότερη. Ο απαιτούμενος αριθμός από εκτελέσεις του βρόχου έως ότου να συγκλίνει η μέθοδος διαφοροποίησης του RIPPLE παίζει κυρίαρχο ρόλο στην απόδοση. Όμως σημαντικό ρόλο παίζουν κι η αποτελεσματικότητα της μεθόδου καθώς και τα χαρακτηριστικά του κατανεμημένου ευρετηρίου που χρησιμοποιείται. Αυτό γίνεται εμφανές στο Σχήμα 6.12 όπου η απόδοση της baseline μεθόδου βελτιώνεται καθώς ο αριθμός των συνάψεων που εγκαθιδρύει ο κάθε κόμβος προς άλλους κόμβους του δικτύου αυξάνεται αναλογικά με τον αριθμό των διαστάσεων και για αυτό η δρομολόγηση γίνεται με πιο αποτελεσματικό τρόπο. Επίσης σημειώνουμε ότι ο αριθμός των σχετικών κόμβων μειώνεται με κάθε εκτέλεση του βρόχου για τη μεθοδο του RIPPLE. Στα Σχήματα 6.11(β΄) και 6.12(β΄) που δείχνουμε τη συμφόρηση του δικτύου φαίνονται τα ωφέλη από τον περιορισμό της αναζήτησης μόνο στις περιοχές οι οποίες μεγιστοποιούν τη διαφοροποίηση του αποτελέσματος αφού με τον τρόπο αυτό επιτυγχάνουμε τον ίδιο σκοπό με τις συμβατικές τεχνικές με αισθητά λιγότερα μηνύματα.

Το Σχήμα 6.13 παρατηρούμε τον αντίκτυπο του αναμενόμενου μεγέθους του αποτελέσματος στην απόδοση των μεθόδων όπου ενδεικτικά η μεγέθυνση του αποτελέσματος επηρεάζει με αμφίρροπο τρόπο. Πιο συγκεκριμένα, καθώς περισσότερα στοιχεία του αποτελέσματος εξετάζονται προς αντικατάσταση, αναμένουμε γραμμική αύξηση για τον αριθμό επαναπροωθήσεων και τη συμφόρηση του δικτύου εξαιτίας των επιπρόσθετων υπολογισμών για τα υποψήφια προς αντικατάσταση στοιχεία. Όμως στο Σχήμα 6.13(α΄) δεν παρατηρούμε τη συμπεριφορά που περιμέναμε για το ripple-fast όπου αποκαλύπτεται η δράση του συνδυασμού δύο αντίθετων φαινομένων. Σε πιο μεγάλη λεπτομέρεια, όταν εξετάζουμε ένα στοιχείο τη φορά, υπάρχουν άλλα $k-1$ που περιορίζουν τον χώρο αναζήτησης. Άρα προσπελαύνονται μόνον οι κόμβοι οι οποίοι επικαλύπτουν την τομή όλων των $k-1$ περιοχών που βελτιώνουν το αποτέλεσμα ως προς κάθε στοιχείο του. Οπότε τώρα γίνεται προφανές ότι επιβάλλονται περισσότεροι περιορισμοί καθώς το k αυξάνεται κι άρα η διαδικασία αναζήτησης εκτυλίσσεται σε μικρότερο χώρο για τον οποίον είναι υπεύθυνοι λιγότεροι κόμβοι. Ως αποτέλεσμα, η απόδοση φαίνεται να μην επηρεάζεται για τη ripple-fast μέθοδο εν τέλει. Άλλωστε για τους ίδιους λόγους η συμφόρηση αυξάνεται αργά με το k στο Σχήμα 6.13(β΄). Όμως αυτά τα ωφέλη παύουν για πολύ μεγάλες τιμές του k όπου το υπολογιστικό κόστος κυριαρχεί.

Στο Σχήμα 6.14 παρατηρούμε κάτι ενδιαφέρον. Όταν η παράμετρος λ παίρνει πολύ χαμηλές ή υψηλές τιμές, ο αριθμός των κόμβων που προσπελαύνονται μειώνεται θεαματικά όπως κι ο αριθμός βημάτων που χρειάζονται. Ο χώρος αναζήτησης για αυτές τις τιμές του λ είναι κατά πολύ μικρότερος καθώς οι περιοχές που εμπεριέχουν καλύτερους αντικαταστάτες είναι είτε πολύ κοντά στο ερώτημα για $\lambda \to 1$, είτε κατά μήκος των ορίων του χώρου που δεικτοδοτεί το ευρετήριο για $\lambda \to 0$. Άρα, όταν η παράμετρος παίρνει τιμές κοντα στο 0 ή το 1, η αναζήτηση κατευθύνεται αυτόματα προς εκείνες τις περιοχές. Με άλλα λόγια, λίγοι μόνο κόμβοι είναι αυτοί που επικαλύπτουν τις περιοχές αυτές για πολύ χαμηλές ή υψηλές τιμές του λ. Συνεπώς, η αναζήτηση διεξάγεται αισθητά πιο γρήγορα και με μικρότερο κόστος σε μηνύματα, όπως φαίνεται στο Σχήμα 6.14, όπου latency και congestion μειώνονται καθώς απομακρυνόμαστε από την κεντρική τιμή $\lambda = 0.5$.

6.6 Συζήτηση

Στο κεφάλαιο αυτό μελετήσαμε ενδελεχώς μεθόδους για την κατανεμημένη επεξεργασία ερωτημάτων (i) προτίμησης (top-k queries), (ii) κορυφογραμμής (skyline computation), και (iii) διαφοροποιημένου αποτελέσματος (search result diversification), με αποτελεσματικό τρόπο καθότι τα ερωτήματα αυτά καθορίζονται από κριτήρια του χρήστη χωρίς όμως αυτός να ορίζει (ή να μπορεί να ορίσει ούτως η άλλως) μία αυστηρά φραγμένη περιοχή στην οποία να στοχεύει η επεξεργασία τους. Αυτό έχει συγκεκριμένες συνέπειες, για παράδειγμα ο πιο απλός τρόπος επεξεργασίας τους θα απαιτούσε την επικοινωνία με όλους τους κόμβους του δικτύου, κάτι το οποίο προφανώς ένας αποδοτικός αλγόριθμος θα έπρεπε να αποφύγει. Για το σκοπό αυτό προτείνουμε ένα πλαίσιο επεξεργασίας τέτοιων ερωτημάτων, ονόματι RIPPLE. Για πρώτη φορά οι μέθοδοί μας είναι αυτές που μπορούν να συμβιβάζουν με κομψό τρόπο τη γρήγορη απόκριση με την μικρή συμφόρηση μέσω της χρήση μίας συγκεκριμένου παραμέτρου που καθορίζεται κατά την έκδοση του ερωτήματος. Σκοπός της παραμέτρου αυτής είναι η

δρομολόγηση των ερωτημάτων με τέτοιον τρόπο ώστε να δίνεται προτεραιότητα σε ιδιαίτερα υποσχόμενες περιοχές η σειριακή προσέγγιση των οποίων γίνεται με προσεκτικό τρόπο, ενώ εντός των πλαισίων των περιοχών αυτών η επεξεργασία γίνεται με γρήγορο τρόπο όπου τα ερωτήματα προωθούνται ταυτόχρονα σε όλους τους σχετικούς κόμβους. Επιπροσθέτως, παρουσιάζουμε μία θεωρητική ανάλυση για το αθροιστικό κόστος των μεθόδων μας που προέρχεται από τον παραμετροποιημένο συνδυασμό των δύο αυτών ετερόκλητων τεχνικών.

Συνοψίζοντας σε ανώτερο επίπεδο, η κύρια ιδέα στην οποία στηρίζεται το RIPPLE είναι η αξιοποίηση της μερικής εικόνας που έχει ο κάθε κόμβος τοπικά αποθηκευμένη για τη δομή του δικτύου, έτσι ώστε να μπορεί να καθοδηγεί την περαιτέρω επεξεργασία του ερωτήματος με τον καλύτερο δυνατό τρόπο. Η χρήση του RIPPLE για την κατανεμημένη επεξεργασία ερωτημάτων κορυφογραμμής, σε συνδυασμό με τεχνικές αναδόμησης του δικτύου ώστε οι κόμβοι να συνάπτουν συνδέσμους με κόμβους οι οποίοι έχουν υψηλή πιθανότητα να εμπεριέχουν μέρος της απάντησης, αποτελεί μία αποδοτική λύση για παρόμοια ερωτήματα. Τέλος, οι μέθοδοι για την αναζήτηση διαφοροποιημένου αποτελέσματος αποτελεί από τις πρώτες προσπάθειες για την επεξεργασία τέτοιου είδους ερωτημάτων σε κατανεμημένο περιβάλλον.

Κεφάλαιο 7

Συμπεράσματα

Do not seek to follow in the footsteps of the men of old; seek what they sought.

Basho

Στο τελευταίο κεφάλαιο της διατριβής συνοψίζουμε τη συνεισφορά μας ενώ στην τελευταία ενότητα συζητάμε προτάσεις για μελλοντικές εργασίες πάνω στα θέματα που πραγματευτήκαμε στο παρόν.

7.1 Σύνοψη

Στα πλαίσια αυτής της διατριβής, προτείναμε μεθόδους για την επεξεργασία σύνθετων ερωτημάτων για αποκεντρωμένα συστήματα μεγάλης κλίμακας με τη χρήση ενός πολυδιάστατου κατανεμημένου ευρετηρίου ονόματι MIDAS. Η προσέγγισή μας διαφέρει από άλλα παρόμοια ευρετήρια και επιτρέπει την επεξεργασία διαφόρων τύπων ερωτημάτων σε πολυδιάστατα δεδομένα παρέχοντας ταυτόχρονα εγγυήσεις ως προς το μέγιστο αριθμό επαναπροωθήσεων. Η συνεισφορά μας βασίζεται σε τρεις άξονες:

επεξεργασία χωρικών δεδομένων μέσω μεθόδων για (i) ερωτήματα σημείων, (ii) ερωτήματα εύρους, και (iii) k κοντινότερων γειτόνων,

επεξεργασία RDF(S) δεδομένων με μεθόδους για την ανάκτησης τριπλετών αλλά και τον γρήγορο υπολογισμό της μεταβατικής κλειστότητας βασισμένοι σε τεχνικές επισήμανσης,

επεξεργασία ερωτημάτων κατάταξης πιο γνωστά κι ως rank queries, όπως (i) top-k ερωτήματα, (ii) ερωτήματα κορυφογραμμής (skyline queries), και (iii) διαφοροποιημένου αποτελέσματος (result diversification).

Πιο συγκεκριμένα, τα ερωτήματα σημείων και τα ερωτήματα εύρους απαιτούν $O(\log n)$ βήματα, όπου n το μέγεθος του δικτύου. Για ερωτήματα k κοντινότερων γειτόνων παρέχουμε δύο εναλλακτικές: eager processing για γρήγορη απόκριση όπου ο αναμενόμενος αριθμός επαναπροωθήσεων φράσσεται στο $O(\log n)$ ενώ για την iterative processing μέθοδο στο $O(\log^2 n)$ αλλά έχει σημαντικά μικρότερο επικοινωνιακό κόστος. Αξίζει να αναφέρουμε ότι τόσο θεωρητικά όσο και πειραματικά το MIDAS υπερτερεί από παρόμοιες τεχνικές που έχουν προταθεί κατά καιρούς στη βιβλιογραφία.

Όσον αφορά τις μεθόδους επεξεργασίας RDF(S) δεδομένων, εκμεταλευόμαστε τη λειτουργικότητα του MIDAS προκειμένου να κατασκευάσουμε ένα εγγενώς πολυδιάστατο κατανεμημένο RDF αποθετήριο. Το MIDAS-RDF είναι σε θέση να επεξεργαστεί RDF ερωτήματα μοτίβου σε λογαριθμικό αριθμό βημάτων ως προς το μέγεθος του δικτύου. Επιπλέον, με το σχεδιασμό ενός forward-chaining επαγωγικού μοντέλου και τη χρήση τεχνικών επισήμανσης

μπορεί να υπολογίσει ερωτήματα για μεταβατικές ιδιότητες σε χρόνο που πλέον δεν εξαρτάται από τη διάμετρο του αντίστοιχου RDF γράφου για το δεδομένο κατηγόρημα, υπερτερώντας έτσι των συμβατικών μεθόδων που μπορεί να συναντήσει κανείς. Ακόμα, παρέχει ένα publish-subscibe μοντέλο που επιτρέπει στους επιλεκτικούς συνδρομητές την έκφραση σύνθετων ενδιαφερόντων βάσει περιεχομένου (π.χ. όχι λέξεων κλειδιών) και τον ορισμό κριτηρίων σχετικότητας, φιλτραρίσματος, διαζεύξεων, συζεύξεων, κ.α.

Το τελευταίο μέρος είναι αφιερωμένο στην κατανεμημένη επεξεργασία ερωτημάτων κατάταξης, κι ειδικότερα τους πιο σημαντικούς τύπους, top-k ερωτήματα, ερωτήματα κορυφογραμμής, και διαφοροποιημένου αποτελέσματος. Οι μέθοδοί μας μελετάνε την εξισορρόπηση μεταξύ του μέγιστου αριθμού επαναπροωθήσεων και της συμφόρησης του δικτύου με την χρήση μίας ειδικής παραμέτρου η τιμή της οποίας καθορίζει τον τρόπο με τον οποίον επεξεργάζεται το ερώτημα. Έτσι προτείνουμε το συνδυασμό επιλεκτικών best-first search μεθόδων αναζήτησης και γρήγορων shower-like multi-cast τεχνικών που συμβιβάζονται κατάλληλα. Τέλος, για την βελτιστοποιημένη επεξεργασία ερωτημάτων κορυφογραμμής επιβάλλουμε διαφορετικό σχηματισμό στη τοπολογία του MIDAS, ενώ το παρόν αποτελεί την πρώτη προσπάθεια με θέμα τη διαφοροποίηση αποτελέσματος για κατανεμημένα περιβάλλοντα.

7.2 Μελλοντικές Εργασίες

Κατά την εργασία στην παρούσα διατριβή, αναγνωρίσαμε τα ακόλουθα ενδιαφέροντα θέματα τα οποία προτείνουμε για μελλοντική εργασία.

- Μία ενδιαφέρουσα επέκταση για το MIDAS θα αφορούσε τον κατάλληλο ανασχεδιασμό των μεθόδων αποθήκευσης κι αναζήτησης ώστε να καταστεί δυνατή η ανάπτυξη κατανεμημένων τεχνικών για την προσεγγιστική αναζήτηση συγκεκριμένων πεδίων (approximate string matching) ως προς δεδομένα χαρακτηριστικά που επιλέγονται από τον χρήστη δυναμικά (at query time).

- Επίσης ενδιαφέρουσα θα ήταν η ανάπτυξη τεχνικών διαφοροποίησης αποτελέσματος για κατανεμημένα διασυνδεδεμένα δεδομένα (linked data) τα οποία διαφοροποιούνται ως προς το είδος των δεδομένων που απασχολήθηκαμε στο γεγονός ότι παρουσιάζουν μία δομή γράφου.

- Υπάρχει η δυνατότητα κατασκευής ενός κατανεμημένου συστήματος αρχείων (distributed filesystem) πάνω από το MIDAS, το οποίο θα είναι σε θέση να αποθηκεύει σε κάποιο μεγάλης κλίμακας κατανεμημένο σύστημα (π.χ. cloud) τμήματα αρχείων των χρηστών τα οποία θα μπορούν να ανακτώνται βάσει πολλαπλών κριτηρίων (π.χ. όνομα αρχείου, χρήστη, κ.τ.λ.)

Συμπερασματικά, πιστεύουμε ότι υπάρχει μια πληθώρα από ενδιαφέροντα και νέα θέματα που σχετίζονται με την αποδοτική διαχείριση κατανεμημένων δεδομένων. Ελπίζουμε η παρούσα διατριβή να αποτελέσει εφαλτήριο για επιπλέον έρευνα στο συγκεκριμένο πεδίο.

Βιβλιογραφία

[1] The dblp data-set. http://dblp.uni-trier.de/xml.

[2] hadoop. http://hadoop.apache.org/.

[3] Hbase. http://hbase.apache.org/.

[4] Sparql query language for rdf. http://www.w3.org/TR/rdf-sparql-query/.

[5] W3c rdfs rules of entailment. http://www.w3.org/TR/rdf-mt/#rules.

[6] World wide web consortium (w3c). http://www.w3.org/.

[7] Daniel J. Abadi, Adam Marcus, Samuel Madden, and Katherine J. Hollenbach. Scalable semantic web data management using vertical partitioning. In *VLDB*, pages 411–422, 2007.

[8] Karl Aberer. P-grid: A self-organizing access structure for p2p information systems. In *CoopIS*, pages 179–194, 2001.

[9] Karl Aberer. Scalable data access in peer-to-peer systems using unbalanced search trees. In *WDAS*, pages 107–120, 2002.

[10] Karl Aberer, Philippe Cudré-Mauroux, Anwitaman Datta, Zoran Despotovic, Manfred Hauswirth, Magdalena Punceva, and Roman Schmidt. P-grid: a self-organizing structured p2p system. *SIGMOD Record*, 32(3):29–33, 2003.

[11] Karl Aberer, Philippe Cudré-Mauroux, and Manfred Hauswirth. The chatty web: emergent semantics through gossiping. In *WWW*, pages 197–206, 2003.

[12] Karl Aberer, Philippe Cudré-Mauroux, Manfred Hauswirth, and Tim Van Pelt. Gridvine: Building internet-scale semantic overlay networks. In *ISWC*, pages 107–121, 2004.

[13] Karl Aberer, Anwitaman Datta, and Manfred Hauswirth. Efficient, self-contained handling of identity in peer-to-peer systems. *IEEE TKDE*, 16, 2004.

[14] Karl Aberer, Anwitaman Datta, and Manfred Hauswirth. P-grid: Dynamics of self-organizing processes in structured peer-to-peer systems. In *Peer-to-Peer Systems and Applications*, pages 137–153, 2005.

[15] Karl Aberer, Anwitaman Datta, Manfred Hauswirth, and Roman Schmidt. Indexing data-oriented overlay networks. In *VLDB*, pages 685–696, 2005.

[16] Serge Abiteboul, Stephen Alstrup, Haim Kaplan, Tova Milo, and Theis Rauhe. Compact labeling scheme for ancestor queries. *SIAM J. Comput.*, 35(6):1295–1309, 2006.

[17] Rakesh Agrawal, Alexander Borgida, and H. V. Jagadish. Efficient management of transitive relationships in large data and knowledge bases. In *SIGMOD Conference*, pages 253–262, 1989.

[18] Rakesh Agrawal, Sreenivas Gollapudi, Alan Halverson, and Samuel Ieong. Diversifying search results. In *WSDM*, pages 5–14, 2009.

[19] Reza Akbarinia, Esther Pacitti, and Patrick Valduriez. Reducing network traffic in unstructured p2p systems using top-*k* queries. *Distributed and Parallel Databases*, 19(2-3):67–86, 2006.

[20] Jochen Alber and Rolf Niedermeier. On multidimensional curves with hilbert property. *Theory Comput. Syst.*, 33(4):295–312, 2000.

[21] Sofia Alexaki, Vassilis Christophides, Gregory Karvounarakis, Dimitris Plexousakis, and Karsten Tolle. The ics-forth rdfsuite: Managing voluminous rdf description bases. In *SemWeb*, 2001.

[22] Stephen Alstrup and Theis Rauhe. Improved labeling scheme for ancestor queries. In *SODA*, pages 947–953, 2002.

[23] Wolf-Tilo Balke, Wolfgang Nejdl, Wolf Siberski, and Uwe Thaden. Progressive distributed top k retrieval in peer-to-peer networks. In *ICDE*, pages 174–185, 2005.

[24] Dominic Battré, Felix Heine, André Höing, and Odej Kao. On triple dissemination, forward-chaining, and load balancing in dht rdf stores. In *DBISP2P*, pages 343–354, 2006.

[25] Norbert Beckmann, Hans-Peter Kriegel, Ralf Schneider, and Bernhard Seeger. The r*-tree: An efficient and robust access method for points and rectangles. In *SIGMOD Conference*, pages 322–331, 1990.

[26] Jon Louis Bentley. Multidimensional binary search trees used for associative searching. *Commun. ACM*, 18(9):509–517, 1975.

[27] Jon Louis Bentley. K-d trees for semidynamic point sets. In *Symposium on Computational Geometry*, pages 187–197, 1990.

[28] Jon Louis Bentley and Jerome H. Friedman. Data structures for range searching. *ACM Comput. Surv.*, 11(4):397–409, 1979.

[29] Tim Berners-Lee, James Hendler, and Ora Lassila. The semantic web. The Scientific American, May, 17, 2001.

[30] Ashwin R. Bharambe, Mukesh Agrawal, and Srinivasan Seshan. Mercury: supporting scalable multi-attribute range queries. In *SIGCOMM*, pages 353–366, 2004.

[31] Spyros Blanas and Vasilis Samoladas. Contention-based performance evaluation of multidimensional range search in p2p networks. In *InfoScale'07*, pages 1–8, 2007.

[32] Spyros Blanas and Vasilis Samoladas. Contention-based performance evaluation of multidimensional range search in peer-to-peer networks. *Future Generation Comp. Syst.*, 25(1):100–108, 2009.

[33] Christian Böhm, Stefan Berchtold, and Daniel A. Keim. Searching in high-dimensional spaces: Index structures for improving the performance of multimedia databases. *ACM Comput. Surv.*, 33(3):322–373, 2001.

[34] Stephan Börzsönyi, Donald Kossmann, and Konrad Stocker. The skyline operator. In *ICDE*, pages 421–430, 2001.

[35] Min Cai and Martin R. Frank. Rdfpeers: a scalable distributed rdf repository based on a structured peer-to-peer network. In *WWW*, pages 650–657, 2004.

[36] Min Cai, Martin R. Frank, Jinbo Chen, and Pedro A. Szekely. Maan: A multi-attribute addressable network for grid information services. *J. Grid Comp.*, 2(1):3–14, 2004.

[37] Pei Cao and Zhe Wang. Efficient top-k query calculation in distributed networks. In *PODC*, pages 206–215, 2004.

[38] Jaime G. Carbonell and Jade Goldstein. The use of mmr, diversity-based reranking for reordering documents and producing summaries. In *SIGIR*, pages 335–336, 1998.

[39] Michael Christen. Yacy*: Large-scale search engine. `http://yacy.net/`.

[40] Vassilis Christophides, Gregory Karvounarakis, Dimitris Plexousakis, Michel Scholl, and Sotirios Tourtounis. Optimizing taxonomic semantic web queries using labeling schemes. *J. Web Sem.*, 1(2):207–228, 2004.

[41] Vassilis Christophides, Dimitris Plexousakis, Michel Scholl, and Sotirios Tourtounis. On labeling schemes for the semantic web. In *WWW*, pages 544–555, 2003.

[42] Paolo Ciaccia, Marco Patella, and Pavel Zezula. M-tree: An efficient access method for similarity search in metric spaces. In *VLDB*, pages 426–435, 1997.

[43] Charles L. A. Clarke, Maheedhar Kolla, Gordon V. Cormack, Olga Vechtomova, Azin Ashkan, Stefan Büttcher, and Ian MacKinnon. Novelty and diversity in information retrieval evaluation. In *SIGIR*, pages 659–666, 2008.

[44] George P. Copeland and Setrag Khoshafian. A decomposition storage model. In *SIGMOD Conference*, pages 268–279, 1985.

[45] Arturo Crespo and Hector Garcia-Molina. Semantic overlay networks for p2p systems. In *AP2PC*, pages 1–13, 2004.

[46] Anwitaman Datta, Manfred Hauswirth, Renault John, Roman Schmidt, and Karl Aberer. Range queries in trie-structured overlays. In *P2P Computing*, pages 57–66, 2005.

[47] Jeffrey Dean and Sanjay Ghemawat. Mapreduce: simplified data processing on large clusters. *Commun. ACM*, 51(1):107–113, 2008.

[48] Jeffrey Dean and Sanjay Ghemawat. Mapreduce: a flexible data processing tool. *Commun. ACM*, 53(1):72–77, 2010.

[49] Paul F. Dietz. Maintaining order in a linked list. In *STOC*, pages 122–127, 1982.

[50] Paul F. Dietz and Daniel Dominic Sleator. Two algorithms for maintaining order in a list. In *STOC*, pages 365–372.

[51] Christos Doulkeridis, Akrivi Vlachou, Yannis Kotidis, and Michalis Vazirgiannis. Peer-to-peer similarity search in metric spaces. In *VLDB*, pages 986–997, 2007.

[52] Marina Drosou and Evaggelia Pitoura. Search result diversification. *SIGMOD Record*, 39(1):41–47, 2010.

[53] Jin-Hang Du, Haofen Wang, Yuan Ni, and Yong Yu. Hadooprdf: A scalable semantic data analytical engine. In *ICIC (2)*, pages 633–641, 2012.

[54] Amalia Duch, Vladimir Estivill-Castro, and Conrado Martínez. Randomized k-dimensional binary search trees. In *ISAAC*, pages 199–208, 1998.

[55] Ronald Fagin, Amnon Lotem, and Moni Naor. Optimal aggregation algorithms for middleware. *Journal of Computer and System Sciences*, 66(4):614–656, 2003.

[56] Fabrizio Falchi, Claudio Gennaro, and Pavel Zezula. Nearest neighbor search in metric spaces through content-addressable networks. *Inf. Process. Manage.*, 44(1):411–429, 2008.

[57] Qiming Fang, Ying Zhao, Guangwen Yang, and Weimin Zheng. Scalable distributed ontology reasoning using dht-based partitioning. In *ASWC*, pages 91–105, 2008.

[58] Piero Fraternali, Davide Martinenghi, and Marco Tagliasacchi. Top-k bounded diversification. In *SIGMOD*, pages 421–432, 2012.

[59] Jerome H. Friedman, Jon Louis Bentley, and Raphael A. Finkel. An algorithm for finding best matches in logarithmic expected time. *ACM Trans. Math. Softw.*, 3(3):209–226, 1977.

[60] Volker Gaede and Oliver Günther. Multidimensional access methods. *ACM Comput. Surv.*, 30(2):170–231, 1998.

[61] Prasanna Ganesan, Beverly Yang, and Hector Garcia-Molina. One torus to rule them all: Multidimensional queries in p2p systems. In *WebDB*, pages 19–24, 2004.

[62] Sreenivas Gollapudi and Aneesh Sharma. An axiomatic framework for result diversification. *IEEE Da. Eng. Bul.*, 32(4):7–14, 2009.

[63] Claudio Gutiérrez, Carlos A. Hurtado, and Alberto O. Mendelzon. Foundations of semantic web databases. In *PODS*, pages 95–106, 2004.

[64] Antonin Guttman. R-trees: A dynamic index structure for spatial searching. In *SIGMOD Conference*, pages 47–57, 1984.

[65] Alon Y. Halevy, Zachary G. Ives, Peter Mork, and Igor Tatarinov. Piazza: data management infrastructure for semantic web applications. In *WWW*, pages 556–567, 2003.

[66] Jayant R. Haritsa. The kndn problem: A quest for unity in diversity. *IEEE Data Eng. Bull.*, 32(4):15–22, 2009.

[67] Stephen Harris and Nicholas Gibbins. 3store: Efficient bulk rdf storage. In *PSSS*, 2003.

[68] Katja Hose and Akrivi Vlachou. A survey of skyline processing in highly distributed environments. *VLDB J.*, 21(3):359–384, 2012.

[69] Benoit Hudzia, M. Tahar Kechadi, and A. Ottewill. Treep: A tree based p2p network architecture. In *CLUSTER*, pages 1–15, 2005.

[70] Ryan Huebsch, Brent N. Chun, Joseph M. Hellerstein, Boon Thau Loo, Petros Maniatis, Timothy Roscoe, Scott Shenker, Ion Stoica, and Aydan R. Yumerefendi. The architecture of pier: an internet-scale query processor. In *CIDR*, pages 28–43, 2005.

[71] Ryan Huebsch, Joseph M. Hellerstein, Nick Lanham, Boon Thau Loo, Scott Shenker, and Ion Stoica. Querying the internet with pier. In *VLDB*, pages 321–332, 2003.

[72] Adriana Iamnitchi, Ian T. Foster, and Daniel Nurmi. A peer-to-peer approach to resource location in grid environments. In *HPDC*, page 419, 2002.

[73] H. V. Jagadish, Beng Chin Ooi, Kian-Lee Tan, Cui Yu, and Rui Zhang. idistance: An adaptive b^+-tree based indexing method for nearest neighbor search. *ACM Trans. Database Syst.*, 30(2):364–397, 2005.

[74] H. V. Jagadish, Beng Chin Ooi, and Quang Hieu Vu. Baton: A balanced tree structure for peer-to-peer networks. In *VLDB*, pages 661–672, 2005.

[75] H. V. Jagadish, Beng Chin Ooi, Quang Hieu Vu, Rong Zhang, and Aoying Zhou. Vbi-tree: A peer-to-peer framework for supporting multi-dimensional indexing schemes. In *ICDE*, page 34, 2006.

[76] R. Jain, D. Chiu, and W. Hawe. A quantitative measure of fairness and discrimination for resource allocation in shared computer systems. In *DEC Research Report TR-301*, 1984.

[77] Zoi Kaoudi, Iris Miliaraki, and Manolis Koubarakis. Rdfs reasoning and query answering on top of dhts. In *ISWC*, pages 499–516, 2008.

[78] Haim Kaplan, Tova Milo, and Ronen Shabo. A comparison of labeling schemes for ancestor queries. In *SODA*, pages 954–963, 2002.

[79] David Karger, Eric Lehman, Tom Leighton, Rina Panigrahy, Matthew Levine, and Daniel Lewin. Consistent hashing and random trees: Distributed caching protocols for relieving hot spots on the world wide web. In *ACM Symp. on Theory of Comp.*, pages 654–663, 1997.

[80] John Kubiatowicz, David Bindel, Yan Chen, Steven E. Czerwinski, Patrick R. Eaton, Dennis Geels, Ramakrishna Gummadi, Sean C. Rhea, Hakim Weatherspoon, Westley Weimer, Chris Wells, and Ben Y. Zhao. Oceanstore: An architecture for global-scale persistent storage. In *ASPLOS*, pages 190–201, 2000.

[81] Ziyang Liu, Peng Sun, and Yi Chen. Structured search result differentiation. *PVLDB*, 2(1):313–324, 2009.

[82] Eng Keong Lua, Jon Crowcroft, Marcelo Pias, Ravi Sharma, and Steven Lim. A survey and comparison of p2p overlay network schemes. *IEEE Communications Surveys and Tutorials*, 7(1-4):72–93, 2005.

[83] Petar Maymounkov and David Mazières. Kademlia: A peer-to-peer information system based on the xor metric. In *IPTPS*, pages 53–65, 2002.

[84] Sebastian Michel, Peter Triantafillou, and Gerhard Weikum. Klee: A framework for distributed top-k query algorithms. In *VLDB*, pages 637–648, 2005.

[85] Enrico Minack, Wolf Siberski, and Wolfgang Nejdl. Incremental diversification for very large sets: a streaming-based approach. In *SIGIR*, pages 585–594, 2011.

[86] Anirban Mondal, Yi Lifu, and Masaru Kitsuregawa. P2pr-tree: An r-tree-based spatial index for peer-to-peer environments. In *EDBT Workshops*, pages 516–525, 2004.

[87] Wolfgang Nejdl, Boris Wolf, Changtao Qu, Stefan Decker, Michael Sintek, Ambjörn Naeve, Mikael Nilsson, Matthias Palmér, and Tore Risch. Edutella: a p2p networking infrastructure based on rdf. In *WWW*, pages 604–615, 2002.

[88] David Novak and Pavel Zezula. M-chord: a scalable distributed similarity search structure. In *Proceedings of the 1st international conference on Scalable information systems*, InfoScale '06, 2006.

[89] Eyal Oren, Spyros Kotoulas, George Anadiotis, Ronny Siebes, Annette ten Teije, and Frank van Harmelen. Marvin: Distributed reasoning over large-scale semantic web data. *J. Web Sem.*, 7(4):305–316, 2009.

[90] J. A. Orenstein and T. H. Merrett. A class of data structures for associative searching. In *PODS '84*, pages 181–190, New York, NY, USA, 1984. ACM.

[91] Aris M. Ouksel and Gianluca Moro. G-grid: A class of scalable and self-organizing data structures for multi-dimensional querying and content routing in p2p networks. In *AP2PC*, pages 123–137, 2003.

[92] Zhengxiang Pan and Jeff Heflin. Dldb: Extending relational databases to support semantic web queries. In *PSSS*, 2003.

[93] Zhengxiang Pan, Xingjian Zhang, and Jeff Heflin. Dldb2: A scalable multi-perspective semantic web repository. In *Web Intelligence*, pages 489–495, 2008.

[94] Nikolaos Papailiou, Ioannis Konstantinou, Dimitrios Tsoumakos, Panagiotis Karras, and Nectarios Koziris. H2rdf+: High-performance distributed joins over large-scale rdf graphs. In *BigData Conference*, pages 255–263, 2013.

[95] Nikolaos Papailiou, Ioannis Konstantinou, Dimitrios Tsoumakos, and Nectarios Koziris. H2rdf: adaptive query processing on rdf data in the cloud. In *WWW (Companion Volume)*, pages 397–400, 2012.

[96] David Peleg. Informative labeling schemes for graphs. *Theor. Comput. Sci.*, 340(3):577–593, 2005.

[97] C. Greg Plaxton, Rajmohan Rajaraman, and Andréa W. Richa. Accessing nearby copies of replicated objects in a distributed environment. *Theory Comput. Syst.*, 32(3):241–280, 1999.

[98] Himabindu Pucha, David G. Andersen, and Michael Kaminsky. Exploiting similarity for multi-source downloads using file handprints. In *NSDI*, 2007.

[99] Sylvia Ratnasamy, Paul Francis, Mark Handley, Richard Karp, and Scott Schenker. A scalable content-addressable network. In *SIGCOMM '01*, pages 161–172, 2001.

[100] Bruce A. Reed. The height of a random binary search tree. *Journal of the ACM*, 50(3):306–332, 2003.

[101] Sean C. Rhea, Patrick R. Eaton, Dennis Geels, Hakim Weatherspoon, Ben Y. Zhao, and John Kubiatowicz. Pond: The oceanstore prototype. In *FAST*, 2003.

[102] Antony I. T. Rowstron and Peter Druschel. Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems. In *Middleware*, pages 329–350, 2001.

[103] Norvald H. Ryeng, Akrivi Vlachou, Christos Doulkeridis, and Kjetil Nørvåg. Efficient distributed top-k query processing with caching. In *DASFAA (2)*, pages 280–295, 2011.

[104] Hanan Samet. *The Design and Analysis of Spatial Data Structures*. Addison-Wesley, 1990.

[105] Anne Schlicht and Heiner Stuckenschmidt. Distributed resolution for alc. In *Description Logics*, 2008.

[106] Timos K. Sellis, Nick Roussopoulos, and Christos Faloutsos. The r+-tree: A dynamic index for multi-dimensional objects. In *VLDB*, pages 507–518, 1987.

[107] Luciano Serafini and Andrei Tamilin. Drago: Distributed reasoning architecture for the semantic web. In *ESWC*, pages 361–376, 2005.

[108] Yanfeng Shu, Beng Chin Ooi, Kian-Lee Tan, and Aoying Zhou. Supporting multi-dimensional range queries in peer-to-peer systems. In *Peer-to-Peer Computing*, pages 173–180, 2005.

[109] Ion Stoica, Robert Morris, David Liben-Nowell, David R. Karger, M. Frans Kaashoek, Frank Dabek, and Hari Balakrishnan. Chord: a scalable p2p lookup protocol for internet applications. *IEEE/ACM Trans. Netw.*, 11(1):17–32, 2003.

[110] Egemen Tanin, Aaron Harwood, and Hanan Samet. Indexing distributed complex data for complex queries. In *dg.o '04*, pages 1–10, 2004.

[111] Athanasios K. Tsakalidis. Maintaining order in a generalized linked list. *Acta Inf.*, 21:101–112, 1984.

[112] George Tsatsanifos, Dimitris Sacharidis, and Timos Sellis. Ripple: A scalable framework for distributed processing of rank queries. In *EDBT*, pages 259–270, 2014.

[113] George Tsatsanifos, Dimitris Sacharidis, and Timos K. Sellis. Midas: Multi-attribute indexing for distributed architecture systems. In *SSTD*, pages 168–185, 2011.

[114] George Tsatsanifos, Dimitris Sacharidis, and Timos K. Sellis. On enhancing scalability for distributed rdf/s stores. In *EDBT*, pages 141–152, 2011.

[115] Erik Vee, Jayavel Shanmugasundaram, and Sihem Amer-Yahia. Efficient computation of diverse query results. *IEEE Data Eng. Bull.*, 32(4):57–64, 2009.

[116] Marcos Vieira, Humberto Razente, Maria Barioni, Marios Hadjieleftheriou, Divesh Srivastava, Caetano Jr., and Vassilis Tsotras. On query result diversification. In *ICDE*, pages 1163–1174, 2011.

[117] Akrivi Vlachou, Christos Doulkeridis, Kjetil Nørvåg, and Michalis Vazirgiannis. On efficient top-k query processing in highly distributed environments. In *SIGMOD Conf.*, pages 753–764, 2008.

[118] Raphael Volz, Daniel Oberle, Steffen Staab, and Boris Motik. Kaon server - a semantic web management system. In *WWW (Alternate Paper Tracks)*, 2003.

[119] Jinbao Wang, Sai Wu, Hong Gao, Jianzhong Li, and Beng Chin Ooi. Indexing multi-dimensional data in a cloud system. In *Proceedings of the 2010 ACM SIGMOD International Conference on Management of data*, SIGMOD '10, pages 591–602, 2010.

[120] Shiyuan Wang, Beng Chin Ooi, Anthony K. H. Tung, and Lizhen Xu. Efficient skyline query processing on peer-to-peer networks. In *ICDE*, pages 1126–1135, 2007.

[121] Shiyuan Wang, Quang Hieu Vu, Beng Chin Ooi, Anthony K. H. Tung, and Lizhen Xu. Skyframe: a framework for skyline query processing in peer-to-peer systems. *VLDB J.*, 18(1):345–362, 2009.

[122] Sage A. Weil, Scott A. Brandt, Ethan L. Miller, Darrell D. E. Long, and Carlos Maltzahn. Ceph: A scalable, high-performance distributed file system. In *OSDI*, pages 307–320, 2006.

[123] Cathrin Weiss, Panagiotis Karras, and Abraham Bernstein. Hexastore: sextuple indexing for semantic web data management. *PVLDB*, 1(1):1008–1019, 2008.

[124] Kevin Wilkinson, Craig Sayers, Harumi A. Kuno, and Dave Reynolds. Efficient rdf storage and retrieval in jena2. In *SWDB*, pages 131–150, 2003.

[125] Kevin Wilkinson and Kevin Wilkinson. Jena property table implementation. In *SSWS*, 2006.

[126] Bernard Wong, Ymir Vigfusson, and Emin Gün Sirer. Hyperspaces for object clustering and approximate matching in peer-to-peer overlays. In *HotOS*, 2007.

[127] Ping Wu, Caijie Zhang, Ying Feng, Ben Y. Zhao, Divyakant Agrawal, and Amr El Abbadi. Parallelizing skyline queries for scalable distribution. In *EDBT*, pages 112–130, 2006.

[128] Yunjie Xu and Hainan Yin. Novelty and topicality in interactive information retrieval. *JASIST*, 59(2):201–215, 2008.

[129] Peter N. Yianilos. Data structures and algorithms for nearest neighbor search in general metric spaces. In *SODA*, pages 311–321, 1993.

[130] Cong Yu, Laks V. S. Lakshmanan, and Sihem Amer-Yahia. It takes variety to make a world: diversification in recommender systems. In *EDBT*, pages 368–378, 2009.

[131] Cui Yu, Beng Chin Ooi, Kian-Lee Tan, and H. V. Jagadish. Indexing the distance: An efficient method to knn processing. In *VLDB*, pages 421–430, 2001.

[132] ChengXiang Zhai, William W. Cohen, and John D. Lafferty. Beyond independent relevance: methods and evaluation metrics for subtopic retrieval. In *SIGIR*, pages 10–17, 2003.

[133] Mi Zhang and Neil Hurley. Avoiding monotony: improving the diversity of recommendation lists. In *RecSys*, pages 123–130, 2008.

[134] Yi Zhang, James P. Callan, and Thomas P. Minka. Novelty and redundancy detection in adaptive filtering. In *SIGIR*, pages 81–88, 2002.

[135] B.Y. Zhao, John Kubiatowicz, and Anthony D. Joseph. Tapestry: a resilient global-scale overlay for service deployment. *IEEE Journal on Selected Areas in Comm.*, 22(1):41–53, 2004.

[136] Keping Zhao, Yufei Tao, and Shuigeng Zhou. Efficient top-k processing in large-scaled distributed environments. *Data Knowl. Eng.*, 63(2):315–335, 2007.

[137] Cai-Nicolas Ziegler, Sean M. McNee, Joseph A. Konstan, and Georg Lausen. Improving recommendation lists through topic diversification. In *WWW*, pages 22–32, 2005.