ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ
ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ Η/Υ, ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ ΚΑΙ ΔΙΚΤΥΩΝ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ ΜΕ ΘΕΜΑ:

«Κ - ΚΟΡΥΦΑΙΕΣ ΕΡΩΤΗΣΕΙΣ ΣΕ ΒΑΣΕΙΣ ΔΕΔΟΜΕΝΩΝ»

ΕΠΙΒΛΕΠΩΝ :
ΕΠΙΚΟΥΡΟΣ ΚΑΘΗΓΗΤΗΣ ΜΠΟΖΑΝΗΣ ΠΑΝΑΓΙΩΤΗΣ

ΣΥΝΕΠΙΒΛΕΠΩΝ:
ΚΑΘΗΓΗΤΗΣ ΣΤΑΜΟΥΛΗΣ ΓΕΩΡΓΙΟΣ

ΕΚΠΟΝΗΣΗ:
ΣΠΑΘΑΣ ΕΥΑΓΓΕΛΟΣ ΑΜ : 1700020

ΒΟΛΟΣ 2008

ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ
ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ Η/Υ, ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ ΚΑΙ ΔΙΚΤΥΩΝ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ ΜΕ ΘΕΜΑ:

«ΣΥΝΕΧΗΣ ΕΛΕΓΧΟΣ ΚΟΡΥΦΑΙΩΝ Κ ΕΠΕΡΩΤΗΣΕΩΝ ΠΑΝΩ ΑΠΟ ΟΛΙΣΘΑΙΝΟΝΤΑ ΠΑΡΑΘΥΡΑ»

ΕΠΙΒΛΕΠΩΝ :
ΕΠΙΚΟΥΡΟΣ ΚΑΘΗΓΗΤΗΣ ΜΠΟΖΑΝΗΣ ΠΑΝΑΓΙΩΤΗΣ

ΣΥΝΕΠΙΒΛΕΠΩΝ:
ΚΑΘΗΓΗΤΗΣ ΣΤΑΜΟΥΛΗΣ ΓΕΩΡΓΙΟΣ

ΕΚΠΟΝΗΣΗ:
ΣΠΑΘΑΣ ΕΥΑΓΓΕΛΟΣ

ΒΟΛΟΣ 2008

ΠΕΡΙΕΧΟΜΕΝΑ

1.Εισαγωγή

Η παρούσα εργασία πραγματεύεται το θέμα των συνεχόμενων επερωτήσεων πάνω από συνεχείς ροές δεδομένων και πιο συγκεκριμένα μελετά το συνεχή έλεγχο κορυφαίων-k ερωτημάτων πάνω από ολισθαίνοντα παράθυρα σταθερού μεγέθους W με τα πιο πρόσφατα δεδομένα.

Ας υποθέσουμε ένα σύνολο δεδομένων P , όπου κάθε αντικείμενο p ∈ P έχει d αριθμητικά γνωρίσματα $p.x_1, p.x_2 \ldots p.x_d$. Μία κορυφαία-κ επερώτηση q συγκεκριμενοποιεί μία συνάρτηση προτίμησης f που απεικονίζει κάθε πλειάδα p που ανήκει στο σύνολο P σε ένα πραγματικό αριθμό score(p) = $f(p.x_1, p.x_2 \ldots p.x_d)$. Το αποτέλεσμα της q είναι το σύνολο των κ αντικειμένων με τους υψηλότερους βαθμούς(score).

Η πιο πάνω μέθοδος είναι σημαντική για πληθώρα (on-line) εφαρμογές. Ας υποθέσουμε, για παράδειγμα, έναν ISP(Internet Service Provider) που ελέγχει τη κίνηση σε πολλά σημεία(π.χ δρομολογητές) μέσα σε ένα δίκτυο. Τα εργαλεία ελέγχου, όπως το NetFlow της Cisco , δημιουργούν και επικοινωνούν (σε ένα κεντρικό εξυπηρετητή) λεπτομερή αρχεία όπου καταγράφεται η κίνηση (logs) κάθε ροής πακέτων.(Κάθε ροή αποτελείται από ένα σύνολο πακέτων που έχουν την ίδια πηγή και τον ίδιο προορισμό). Τα αποτελέσματα των ροών δεδομένων έχουν πολύ υψηλή συχνότητα και συνήθως υπολογίζονται για εκατοντάδες GBytes δεδομένων καθημερινά.

Μία τυπική πλειάδα στο πιο πάνω σενάριο περιλαμβάνει τη πηγή και τον προορισμό (διεύθυνση IP),το χρόνο έναρξης και λήξης, την ποσότητα πακέτων κ. α . Η διαθεσιμότητα τέτοιων στοιχείων στο κεντρικό εξυπηρετητή κάνει εφικτό το συνεχή έλεγχο μέσω πολυάριθμων ερωτημάτων της κίνησης ,της ασφάλειας του δικτύου , ή της εντόπισης βλαβών. Για παράδειγμα, κάποιος μπορεί να θέλει να παρακολουθήσει σε πραγματικό χρόνο τις κορυφαίες 100 ροές με τη μεγαλύτερη έξοδο. Αν ένα σύνολο από αποτελέσματα αναφέρεται στον ίδιο προορισμό (διεύθυνση IP) , αυτό μπορεί να είναι μία υπόδειξη για το ότι ο προορισμός είναι θύμα κατανεμημένη επίθεσης άρνησης εξυπηρέτησης(Denial of Service attack).Από την άλλη μεριά, κάποιος μπορεί να ρωτήσει «ποιες είναι οι 100 κορυφαίες ροές με τον ελάχιστο αριθμό μεταδιδόμενων πακέτων;»Εδώ αν ένα σύνολο από πλειάδες μοιράζεται την ίδια διεύθυνση IP σαν πηγή ,τότε αυτό θα μπορούσε να είναι σημάδι για σκουλήκι(internet warm) .Και στις δύο περιπτώσεις , η εφαρμογή εξαρτάται από το χρόνο ,έτσι η έγκαιρη και συνεχόμενη αποτίμηση κάθε ερωτήματος είναι ουσιαστικό θέμα.

Πληθώρα τεχνικών έχουν προταθεί για ερωτήματα κορυφαίων κ σε συμβατικές βάσεις δεδομένων και σε κατανεμημένους χώρους αποθήκευσης και αρκετές από αυτές θα δούμε στη συνέχεια με ειδικότερη αναφορά στον αλγόριθμο του Fagin (FA) καθώς και σε άλλους αλγόριθμους (TA, NRA, CA).

Επίσης θα γίνει ειδική αναφορά στη τεχνική του YI για την επεξεργασία κορυφαίων-k όψεων , η οποία θα χρησιμοποιηθεί και παρακάτω κατά την παρουσίαση του TSL (Threshold List Algorithm).

Το βασικό πεδίο μελέτης όμως της παρούσας εργασίας αφορά το ζήτημα της ύπαρξης αλγόριθμων για την επεξεργασία πολλαπλών ερωτημάτων που απαιτούν συνεχόμενη αποτίμηση σε on-line κατάσταση. Ζήτημα που μένει αναπάντητο από τους προγενέστερους αλγόριθμους.

Βασικά ασχολούμαστε με τον συνεχή έλεγχο κορυφαίων-k ερωτημάτων σε συνεχείς ροές δεδομένων ,υποθέτοντας ότι γίνεται χρήση μόνο του append-only μοντέλου ροής δεδομένων. Πλειάδες ρέουν συνεχώς προς το σύστημα και θεωρούνται έγκυρες μόνο όταν ανήκουν σε ένα ολισθαίνων παράθυρο W. Θεωρούμε δύο εκδοχές παραθύρων. Το βασισμένο στη ποσότητα παράθυρο W το οποίο περιέχει τις N πιο πρόσφατες καταχωρήσεις, καθώς και το βασισμένο στο χρόνο ,που περιλαμβάνει όλες τις πλειάδες που καταφθάνουν σε μία καθορισμένη χρονική περίοδο που περιλαμβάνει τις πιο πρόσφατες χρονικές στιγμές. Το καθήκον του επεξεργαστή ερωτημάτων είναι να αναφέρει σταθερά το σύνολο των κορυφαίων κ αποτελεσμάτων μεταξύ των έγκυρων δεδομένων που απαιτεί κάθε ερώτημα ελέγχου. Προτείνεται ένα γενικό πλαίσιο εργασίας που έχει εφαρμογή και στα δύο είδη παραθύρων ,σε αυθαίρετο k και σε οποιαδήποτε μονότονη συνάρτηση βαθμολόγησης.

Προκειμένου να πετύχουμε πραγματικού χρόνου αποτίμηση ερωτήματος , οι έγκυρες πλειάδες είναι αποθηκευμένες στην κύρια μνήμη και δεικτοδοτούνται από ένα ομαλό πλέγμα. Όταν ένα ερώτημα q καταφθάνει πρώτο στο σύστημα, το αποτέλεσμα του υπολογίζεται από το μοντέλο υπολογίσιμου των κορυφαίων κ, το οποίο ψάχνει τον ελάχιστο αριθμό των κελιών που μπορεί να περιέχουν αποτελέσματα εγγραφών. Η διαχείριση των κορυφαίων – κ ασχολείται μόνο με προσθήκες και διαγραφές πλειάδων που βρίσκονται μέσα σε αυτά τα κελιά. Διακρίνουμε δύο πολιτικές και αντιστοίχως προτείνουμε δύο αλγορίθμους : I) Ο αλγόριθμος ελέγχου των κορυφαίων κ(TMA- Top-k Monitoring algorithm) επαναϋπολογίζει την απάντηση ενός ερωτήματος q όταν κάποια από τις συγκεκριμένες κορυφαίες κ πλειάδες λήγει. II) Ο Skyband monitoring algorithm (SMA) μερικώς προϋπολογίζει τα μελλοντικά αποτελέσματα ,μειώνοντας το πρόβλημα σε διατήρηση του skyband πάνω από ένα υποσύνολο έγκυρων καταχωρήσεων.

Αναλυτικά η εργασία οργανώνεται ως εξής : Στο 2ο κεφάλαιο αναφέρονται παλαιότερες προσπάθειες πάνω στο θέμα των κορυφαίων k ερωτημάτων καθώς και διαφορετικοί τύποι ερωτημάτων. Στο 3ο κεφάλαιο γίνεται ειδική αναφορά στο μοντέλο ροής δεδομένων και στην εφαρμογή ερωτημάτων πάνω σε αυτό. Και στο 4ο κεφάλαιο παρουσιάζουμε αρχικά μία τεχνική που θα χρησιμοποιηθεί για να συγκριθεί με τις μεθόδους που παρουσιάζουμε και έπειτα γίνεται αναλυτική παρουσίαση των αλγορίθμων TMA, SMA , της απόδοσης τους με μαθηματικούς τύπους και με πείραμα όπως και ειδική αναφορά σε άλλους τύπους ερωτημάτων.

2. ΠΡΟΗΓΟΥΜΕΝΕΣ ΕΡΓΑΣΙΕΣ

Στον παρόν κεφάλαιο γίνεται η προσπάθεια να παρουσιαστούν υπάρχοντες τεχνικές που διαπραγματεύονται την επεξεργασία κορυφαίων –k ερωτημάτων σε ποικίλα σενάρια.

2.1

Ο Onion και ο Prefer έχουν επεξεργαστεί βασικές τεχνικές για επερωτήσεις κορυφαίων – κ σε συμβατικές βάσεις δεδομένων. Ο Onion συγκεκριμένα , υποκινείται από το γεγονός ότι το σημείο με τον υψηλότερο βαθμό (για κάθε γραμμική συνάρτηση προτίμησης) μπορεί να βρεθεί μέσα στο χώρο του συνόλου δεδομένων. Η μέθοδος υπολογίζει και αποθηκεύει κυρτούς χώρους σε επίπεδα, με τα εξωτερικά επίπεδα γεωμετρικά να κλείνουν τα εσωτερικά. Μία γραμμική επερώτηση κορυφαίων k αποτιμάται επεξεργάζοντας τα επίπεδα προς τα μέσα ξεκινώντας από τα εξωτερικά. Ο Prefer προ-υπολογίζει ένα σύνολο από ταξινομημένες λίστες σύμφωνα με κάποιες αυθαίρετες συναρτήσεις προτιμήσεως , και τις επεξεργάζεται σαν όψεις. Για κάθε δοσμένη συνάρτηση προτίμησης f , διαλέγει τις επεξεργασμένες όψεις ανταποκρινόμενος στη συνάρτηση που είναι πιο κοντά στην f, η επερώτηση τότε μπορεί να απαντηθεί επαρκώς εξετάζοντας ένα υποσύνολο εγγραφών σε αυτή την όψη. Ο Prefer εργάζεται για μη γραμμικές συναρτήσεις ενεργοποίησης , προϋποθέτοντας ότι ένα διαφορετικό σύνολο από όψεις θα διατηρείται για κάθε τύπο λειτουργίας.

Συνεχίζοντας με τις προϋπάρχουσες τεχνικές προσέγγισης των κορυφαίων κ συναντάμε τον Bruno που χρησιμοποιεί πολυδιάστατα ιστογράμματα προκειμένου να απεικονίσει τις επερωτήσεις κορυφαίων κ σε παραδοσιακά πεδία και έτσι να τις επεξεργαστεί μέσα από σχεσιακά συστήματα διαχείρισης βάσεων δεδομένων (RDBMS).

Παρομοίως οι Chen και Ling χρησιμοποιούν μία μέθοδο βασιζόμενη σε δειγματοληψία για να μετατρέψουν τις επερωτήσεις κορυφαίων k σε βέλτιστες συναρτήσεις.

Και στις δύο περιπτώσεις, αν η συνάρτηση δεν παράγει κ αποτελέσματα η διαδικασία επαναλαμβάνεται. Ακόμη ένα καλά μελετημένο πρόβλημα αφορά τις κορυφαίες κ εγγραφές ανάμεσα στα αποτελέσματα μιας ένωσης πάνω από πολλαπλές

συσχετίσεις. Ο Donjerkovic και ο Ramakrishnan εφάρμοσαν πιθανοκρατική βελτιστοποίηση για να απαντήσουν διατεταγμένες επερωτήσεις που έμπλεκαν επιλογές και ενώσεις. Ο Ilyas προτείνει ένα αλγόριθμο με σωληνώσεις ,ιδανικό για υλοποίηση μέσα σε μία ιεραρχία τελεστών ένωσης. Ο ίδιος έπειτα επίσης παρουσίασε ένα διασωληνομένο αλγόριθμο βασισμένο στην ripple –join τεχνική. Ο Ilyas επιπλέον ερευνά την ενσωμάτωση των κατατασσόμενων τελεστών σε συμβατικές βάσεις δεδομένων, εκτιμώντας το κόστος τους σαν μέρος ενός σχεδίου εκτέλεσης επερώτησης. Τέλος ο Tsaparas, δημιούργησε ένα ιεραρχημένο ευρετήριο για να απαντήσει επαρκώς κορυφαίες –κ ενώσεως επερωτήσεις για αυθαίρετες συναρτήσεις προτίμησης.

2.2 Αναλυτική παρουσίαση των αλγορίθμων FA, TA, NRA, CA

Παλαιότερα τα συστήματα βάσεων δεδομένων απαιτούσαν την αποθήκευση μόνο μικρών αλφαριθμητικών όπως οι καταχωρήσεις σε μία πλειάδα σε παραδοσιακή σχεσιακή βάση δεδομένων. Έτσι , τα δεδομένα ήταν κάπως ομογενοποιημένα. Στις μέρες μας, επιθυμούμε τα συστήματα βάσεων δεδομένων να είναι ικανά να αντιμετωπίσουν όχι μόνο αλφαριθμητικά αλλά μία πληθώρα δεδομένων από πολυμέσα (π.χ εικόνα , ήχο κ.τ.λ). Έτσι , τα δεδομένα στα οποία επιθυμούμε να έχουμε πρόσβαση και να συνδυάζουμε μπορεί να ανήκουν σε μία πληθώρα αποθηκών δεδομένων, και επίσης να θέλουμε το σύστημα βάσεων δεδομένων μας να λειτουργεί σαν μεσάζοντας που θα μπορεί να έχει πρόσβαση σε τέτοια δεδομένα.

Μία βασική διαφορά μεταξύ μικρών αλφαριθμητικών και δεδομένων από πολυμέσα είναι ότι τα δεύτερα μπορεί να έχουν χαρακτηριστικά τα οποία δεν ορίζονται με σαφήνεια. Για παράδειγμα, δεν λέμε ότι μία δοσμένη εικόνα είναι «κόκκινη» ή «όχι κόκκινη».Αλλά, υπάρχει μία κλίμακα του κόκκινου που ξεκινάει από 0 = καθόλου κόκκινο και καταλήγει στο 1 = τελείως κόκκινο.

Μία προσέγγιση για να αντιμετωπίσουμε τέτοια θέματα είναι να χρησιμοποιούμε την αθροιστική συνάρτηση t. Αν $x_1, x_2 \ldots, x_m$, (καθένα στο εσωτερικό το [0,1]) είναι οι βαθμοί του αντικειμένου R κάτω από τα m χαρακτηριστικά, τότε $t(x_1, x_2 \ldots, x_m)$ είναι ο γενικός βαθμός του αντικειμένου R.

Μία δημοφιλής επιλογή για την αθροιστική συνάρτηση είναι η min. Στη πραγματικότητα, κάτω από τους κανόνες της fuzzy λογικής , αν ένα αντικείμενο R έχει βαθμό x_1 στο χαρακτηριστικό A1 και βαθμό x_2 στο χαρακτηριστικό A2 τότε ο βαθμός στο χαρακτηριστικό A1∩A2 είναι το $\min(x_1, x_2)$.Μία άλλη δημοφιλής αθροιστική συνάρτηση είναι ο μέσος όρος(ή το άθροισμα σε περίπτωση που δεν μας ενδιαφέρει αν το αποτέλεσμα βρίσκεται στο εσωτερικό του [0,1]).

Λέμε πως μία αθροιστική συνάρτηση είναι μονότονη αν ισχύει ότι $t(x_1, x_2 \ldots, x_m \leq t(x_1', x_2' \ldots, x_m')$ όποτε ισχύει $xi \leq xi'$ για κάθε i.Βέβαια η μονοτονία είναι μία λογική ιδιότητα που πρέπει να απαιτείται από μία αθροιστική συνάρτηση: Αν για κάθε γνώρισμα, ο βαθμός ενός αντικειμένου R' είναι το ίδιο υψηλός όσο του αντικειμένου R ,τότε θα περιμέναμε από το γενικό βαθμό της R' να είναι τουλάχιστον τόσο υψηλός όσο αυτός της R.

Η ιδέα της επερώτησης είναι διαφορετική σε ένα σύστημα πολυμεσικών βάσεων δεδομένων από ότι σε ένα παραδοσιακό σύστημα βάσεων δεδομένων. Δοθέντος μίας επερώτησης σε ένα παραδοσιακό σύστημα βάσεων δεδομένων (όπως ένα σχεσιακό σύστημα βάσεων δεδομένων) , υπάρχει ένα αταξινόμητο σύνολο από απαντήσεις. Εν αντιθέσει , σε ένα σύστημα πολυμεσικών βάσεων δεδομένων, η απάντηση σε μία επερώτηση είναι ένα «βαθμολογημένο» σύνολο. Ένα βαθμολογημένο σύνολο είναι ένα σύνολο ζευγαριών (x,g), όπου το x είναι ένα αντικείμενο και g (ο βαθμός) είναι ένας πραγματικός αριθμός στο εσωτερικό του [0,1]. Τα βαθμολογημένα σύνολα συνήθως παρουσιάζονται σε ταξινομημένη σειρά, κατά βαθμό. Θα αναγνωρίζουμε μία επερώτηση με επιλογή της αθροιστικής συνάρτησης t.Ο χρήστης ενδιαφέρεται να βρει τις κορυφαίες k απαντήσεις , όπου k είναι μία δοσμένη παράμετρος(όπως k = 1 , 10 ή 100).Αυτό σημαίνει ότι θέλουμε σαν αποτέλεσμα k αντικείμενα(στα οποία θα μπορούμε να αναφερόμαστε σαν τα «κορυφαία k αντικείμενα») με τους υψηλότερους βαθμούς σε αυτή την επερώτηση, το καθένα μαζί με το βαθμό του. Για ευκολία, παρακάτω θα θεωρούμε το k ως μία σταθερή τιμή, και θα παρουσιάζουμε αλγορίθμους για την εύρεση των κορυφαίων k απαντήσεων σε βάσεις δεδομένων που περιέχουν τουλάχιστον k αντικείμενα.

Άλλες εφαρμογές: Υπάρχουν και άλλες εφαρμογές εκτός από βάσεις δεδομένων για πολυμέσα όπου κάνουμε χρήση μίας αθροιστικής συνάρτησης προκειμένου να συνδυάσουμε βαθμούς, και όπου θέλουμε να βρούμε τις κορυφαίες k απαντήσεις.

Ένα σημαντικό παράδειγμα είναι η ανάκτηση πληροφορίας όπου τα αντικείμενα R είναι κείμενα, τα m γνωρίσματα είναι όροι ανεύρεσης $s_1, \ldots s_m$ και ο βαθμός xi μετρά τη σχέση του κειμένου R με τους όρους $s_1, \ldots s_m$. Είναι σύνηθες να παίρνουμε την αθροιστική συνάρτηση t να είναι το άθροισμα. Αυτό σημαίνει ότι ο συνολικός βαθμός σχετικότητας του κειμένου R όταν η επερώτηση αποτελείται από τους όρους ανεύρεσης $s_1, \ldots s_m$ να είναι $t(x_1, x_2 \ldots, x_m) = x_1 + x_2 + \ldots + x_m$.

Μία άλλη εφαρμογή αφορά το προγραμματισμό μεγάλης κλίμακας κατ' απαίτηση δεδομένων. Σε αυτή τη περίπτωση κάθε αντικείμενο είναι μία σελίδα και υπάρχουν δύο πεδία. Το πρώτο πεδίο αναπαριστά τη ποσότητα του χρόνου που περιμένει ο πρώτος χρήστης που ζητάει μία σελίδα, και το δεύτερο πεδίο αναπαριστά τον αριθμό των χρηστών που ζητούν μία σελίδα. Κάνουν χρήση της συνάρτηση t με $t(x_1, x_2) = x_1 x_2$, και επιθυμούν να εκπέμψουν τη σελίδα με το υψηλότερο βαθμό.

Το μοντέλο: Υποθέτουμε ότι κάθε βάση δεδομένων αποτελείται από ένα πεπερασμένο σύνολο από αντικείμενα. Θα θέσουμε τυπικά N να αναπαριστούν τον αριθμό των αντικειμένων. Σχετικά με κάθε αντικείμενο R είναι m πεδία $x_1, x_2, \ldots, x_m$, όπου x_i ανήκει στο [0,1] για κάθε i. Μπορούμε να αναφερόμαστε στο x_i σαν το i-οστό πεδίο του R. Η βάση δεδομένων μπορεί να ειδωθεί σαν να αποτελείται από μία μόνο σχέση, όπου η μία στήλη αναφέρεται στο id του αντικειμένου και οι άλλες στήλες ανταποκρίνονται στα m γνωρίσματα του αντικειμένου. Εναλλακτικά, ο τρόπος με τον οποίο θα σκεφτόμαστε για μία βάση δεδομένων είναι σαν αποτελείται από m ταξινομημένες λίστες $L_1, L_2, L_3, .. L_m$,κάθε μία μήκους N(υπάρχει μία καταχώρηση σε κάθε λίστα για κάθε ένα από τα N αντικείμενα).Μπορούμε να αναφερόμαστε στη λίστα L_i σαν λίστα i. Κάθε καταχώρηση της L_i , είναι του τύπου(R, x_i) όπου x_i είναι το ιοστό πεδίο του R. Κάθε λίστα L_i είναι ταξινομημένη κατά φθίνουσα σειρά σύμφωνα με τιμή του γνωρίσματος x_i. Στην πράξη μπορεί να είναι αρκετά ακριβό να υπολογίζουμε τις τιμές των πεδίων, αλλά αγνοούμε το ζήτημα εδώ ,και από εδώ και στο εξής θα παίρνουμε τις τιμές όπως δίνονται.

Θεωρούμε δύο τρόπους για τη προσπέλαση δεδομένων. Ο πρώτος τρόπος προσπέλασης είναι η ταξινομημένη (ή ακολουθιακή) προσπέλαση. Εδώ το σύστημα πετυχαίνει το βαθμό ενός αντικειμένου σε μία από τις ταξινομημένες λίστες προχωρώντας μέσα από την λίστα ακολουθιακά ξεκινώντας από την κορυφή. Έτσι, αν ένα αντικείμενο R έχει το l πιο υψηλό βαθμό στην i λίστα, τότε l ταξινομημένες προσπελάσεις στο ioστό απαιτούνται για να δούμε αυτόν το βαθμό υπό ταξινομημένη προσπέλαση. Ο δεύτερος τρόπος είναι η τυχαία προσπέλαση. Εδώ, το σύστημα απαιτεί το βαθμού του αντικειμένου R στην i λίστα και το πετυχαίνει σε μία τυχαία προσπέλαση. Αν υπάρχουν s ταξινομημένες προσπελάσεις και r τυχαίες προσπελάσεις τότε το κόστος της ταξινομημένης προσπέλασης είναι sc_S, και το κόστος της τυχαίας προσπέλασης είναι rcr, και το μέσο κόστος είναι $sc_S + rc_R$, για κάποιες θετικές σταθερές c_S και c_R.

Αλγόριθμοι: Υπάρχει ένας προφανής απλοϊκός αλγόριθμος για την επίτευξη των κορυφαίων k απαντήσεων. Υπό ταξινομημένη προσπέλαση , σε κάθε καταχώρηση και σε κάθε ένα από τις m ταξινομημένες λίστες, υπολογίζει(χρησιμοποιώντας το t) το γενικό βαθμό κάθε αντικειμένου, και επιστρέφει τις κορυφαίες k απαντήσεις. Ο αλγόριθμος έχει γραμμικό μέσο κόστος(γραμμικό ως προς το μέγεθος της βάσης δεδομένων), και έτσι δεν είναι επαρκής για μεγάλη βάση δεδομένων.

Ο FAGIN πρότεινε έναν αλγόριθμο που πολλές φορές ανταποκρίνεται καλύτερα από τον απλοϊκό αλγόριθμο. Στη περίπτωση που οι εντολές στις ταξινομημένες λίστες είναι ανεξάρτητες (σαν γεγονότα πιθανοτήτων) , ο FA βρίσκει τις κορυφαίες k απαντήσεις , πάνω από μια βάση δεδομένων με N αντικείμενα και με μέσο κόστος $O(N^{(m-1)/m}k^{1/m})$, με αυθαιρέτως μεγάλη

πιθανότητα. Ο Fagin επίσης απέδειξε ότι υπό την υπόθεση ανεξαρτησίας, καθώς και με μία υπόθεση στην αθροιστική συνάρτηση ,κάθε σωστός αλγόριθμος πρέπει, με υψηλή πιθανότητα, να προκαλεί ένα μέσο κόστος στη χειρότερη περίπτωση.

Έπειτα θα παρουσιάσουμε τον «threshold algorithm» ή TA . Αυτός ο αλγόριθμος ανακαλύφθηκε ανεξάρτητα από τουλάχιστον τρεις ομάδες.

Θα δείξουμε ότι ο TA αλγόριθμος είναι καταλληλότερος από τον FA. Θα ορίσουμε αυτήν την έννοια της καταλληλότητας ,η οποία θεωρείται ενδιαφέρουσα.

Καταλληλότητα στιγμιότυπου(instance optimality) : Έστω ότι **A** είναι μία κλάση αλγορίθμων, έστω ότι **D** είναι μία κλάση βάσεων δεδομένων ,και έστω ότι cost(A,D) είναι το μέσο κόστος που προκύπτει από την εκτέλεση του αλγορίθμου A πάνω από τη βάση δεδομένων D. Λέμε ότι ένας αλγόριθμος B είναι κατάλληλος κατά το στιγμιότυπο πάνω από τα **A** και **D** αν B ανήκει στο **A** και αν για κάθε D ανήκει στο **D** έχουμε ότι:

$$cost(B,D) = O(\ cost\ (A,D)\ ).$$

Η παραπάνω ισότητα σημαίνει ότι υπάρχουν σταθερές c ,c′ τέτοιες ώστε το $cost(B,D) \leq c\ cost(A,D) + c'$ για κάθε επιλογή A ανήκει στο **A** και D ανήκει στο **D**. Θα αναφερόμαστε στο c σαν την αναλογία καταλληλόλητας. Διαισθητικά , η καταλληλότητα ενός στιγμιότυπου ανταποκρίνεται στη καταλληλόλητα κάθε στιγμιότυπου, όπως αντιμετωπίζεται στην ακριβώς χειρότερη περίπτωση ή στη μέση περίπτωση. Ο FA είναι κατάλληλος με πιθανότητα χειρότερης περίπτωσης υπό κάποιες προϋποθέσεις. Ο TA είναι βέλτιστος με ισχυρότερη πιθανότητα και χωρίς ένα στοιχειώδες πιθανοκρατικό μοντέλο : είναι βέλτιστος , για κάποιες φυσικές επιλογές των **A** , **D**. Ειδικά , η καταλληλότητα στιγμιότυπου ισχύει όταν **A** θεωρείται η κλάση των αλγορίθμων που κανονικά θα υλοποιηθεί στη πράξη(αφού οι μόνοι αλγόριθμοι που αποκλείονται είναι αυτοί που κάνουν πολύ τυχαίες υποθέσεις), και όταν το **D** θεωρείται να είναι η κλάση όλων των βάσεων δεδομένων. Η καταλληλότητα στιγμιότυπου του TA ισχύει σε αυτή τη περίπτωση για κάθε μονότονη αθροιστική συνάρτηση. Εν αντιθέσει, η καταλληλότητα υψηλής πιθανότητας χειρότερης περίπτωσης του FA ισχύει μόνο υπό την υπόθεση της «αυστηρότητας» (θα ορίσουμε παρακάτω την αυστηρότητα, κατά μία έννοια σημαίνει ότι η αθροιστική συνάρτηση αναπαριστά τη συνένωση)

Βελτιστοποίηση και πρόωρη παύση: Υπάρχουν στιγμές όπου ο χρήστης μπορεί να είναι ικανοποιημένος με μία βέλτιστη λίστα κορυφαίων k. Ας υποθέσουμε θ>1. Ορίζουμε μία θ-βελτιστοποίηση για τις κορυφαίες κ απαντήσεις για την αθροιστική συνάρτηση t να είναι μια συλλογή k αντικειμένων(το κάθε ένα με τους βαθμούς του) έτσι ώστε για κάθε y μεταξύ αυτών των k αντικειμένων και κάθε z όχι μεταξύ αυτών των k αντικειμένων ισχύει $\theta(t(y) \geq t(z)$. Σημειώνουμε ότι ο ίδιος ορισμός με θ = 1 δίνει τις κορυφαίες

k απαντήσεις. Θα δείξουμε πώς να μετατρέψουμε το ΤΑ έτσι ώστε να δίνει τέτοια θ-προσέγγιση .Στη πραγματικότητα, μπορούμε εύκολα να μετατρέψουμε το ΤΑ σε μία αλληλεπιδρώσα διαδικασία όπου σε όλες τις στιγμές το σύστημα θα έδειχνε στο χρήστη τη τρέχουσα όψη της λίστας των κορυφαίων k μαζί με εχέγγυα του βαθμού προσέγγισης της θ στη σωστή απάντηση. Σε κάθε χρονική στιγμή, ο χρήστης μπορεί να αποφασίσει πότε θα ήθελε να σταματήσει την επεξεργασία.

Περιορίζοντας τη τυχαία προσπέλαση

Όπως θα συζητηθεί παρακάτω ,υπάρχουν κάποια συστήματα όπου η τυχαία προσπέλαση είναι αδύνατη. Για να αντιμετωπίσουμε τέτοιες καταστάσεις , θα δούμε παρακάτω πώς να μετατρέψουμε τον ΤΑ προκειμένου να πετύχουμε έναν άλλο αλγόριθμο το NRA (καθόλου τυχαία προσπέλαση) ο οποίος δεν επιτρέπει καθόλου τυχαία προσπέλαση. Θα αποδείξουμε ότι ο NRA είναι κατάλληλος για όλους τους αλγορίθμους που δεν κάνουν τυχαία προσπέλαση σε όλες τις βάσεις δεδομένων. Όσον αφορά τις καταστάσεις όπου δεν υπάρχει τυχαία προσπέλαση, πολύ απλά είναι πολύ ακριβή? Αν και ο ΤΑ είναι βέλτιστος , η αναλογία καταλληλότητας εξαρτάται από το λόγο c_R / c_S του κόστους μία μόνο τυχαίας προσπέλασης στο κόστος μίας μόνο ταξινομημένης προσπέλασης.
Ορίζουμε έναν άλλο αλγόριθμο που είναι συνδυασμός του ΤΑ και του NRA τον οποίο καλούμε CA(συνδυαστικός αλγόριθμος).Ο ορισμός του αλγορίθμου εξαρτάται από το λόγο c_R / c_S .Το κίνητρο είναι να επιτευχθεί ένας αλγόριθμος που δεν είναι μόνο κατάλληλος σύμφωνα με το στιγμιότυπο , αλλά του οποίου ο λόγος καταλληλότητας είναι ανεξάρτητος από το c_R / c_S . Η αρχική μας ελπίδα ήταν ο CA να ήταν βέλτιστος κατά το στιγμιότυπο(με το λόγο καταλληλότητας να είναι ανεξάρτητος από το c_R / c_S) σε αυτές τις περιπτώσεις όπου ο ΤΑ είναι βέλτιστος κατά το στιγμιότυπο. Όχι μόνο αυτή η ελπίδα έχει αποτύχει αλλά επίσης αποδείξαμε ότι δεν υπάρχει κάποιος ντετερμινιστικός αλγόριθμος ,ή πιθανοκρατικός αλγόριθμος που δεν κάνει κάποιο λάθος ,με τον λόγο καταλληλότητας να είναι ανεξάρτητος από το c_R / c_S σε αυτές τις περιπτώσεις. Παρ' όλαυτα , βρήκαμε μία καινούργια περίπτωση όπου ο CA είναι κατάλληλος κατά το στιγμιότυπο, με το λόγο καταλληλότητας να είναι ανεξάρτητος από το λόγο c_R / c_S .

Τρόποι προσπέλασης των δεδομένων

Τα θέματα της επαρκούς αξιολόγησης επερώτησης σε ένα μέσο σύστημα είναι πολύ διαφορετικά από εκείνα σε ένα παραδοσιακό σύστημα βάσεων δεδομένων. Αυτό γιατί το μέσο σύστημα λαμβάνει απαντήσεις σε επερωτήσεις από πληθώρα υποσυστημάτων, τα οποία μπορούν να προσπελαστούν μόνο με

περιορισμένους τρόπους .Τι υποθέτουμε για τη διεπαφή ανάμεσα στο σύστημα και στα υποσυστήματα; Ας θεωρήσουμε το QBIC(«Query by image content») σαν το υποσύστημα(βλέπε [11]). Το QBIC μπορεί να αναζητήσει εικόνες από ποικιλία χαρακτηριστικών όπως το χρώμα και το κείμενο(και μία πειραματική εκδοχή μπορεί να αναζητήσει ακόμη και από το σχήμα). Σε απάντηση σε μία επερώτηση, όπως Color="red", το υποσύστημα θα βγάλει το βαθμολογημένο σύνολο που θα αποτελείται από όλα τα αντικείμενα, ένα προς ένα, το κάθε ένα με το βαθμό του υπό την επερώτηση, σε ταξινομημένη σειρά βασιζόμενη στο βαθμό, μέχρι το σύστημα να πει στο υποσύστημα να σταματήσει. Τότε το σύστημα μπορεί αργότερα να πει στο υποσύστημα να επαναλάβει τη παραγωγή στην έξοδο του βαθμολογημένου συνόλου όπου έχει φύγει. Εναλλακτικά , το σύστημα μεσάζων μπορεί να ρωτήσει το υποσύστημα για, να του πει, τα 10 κορυφαία αντικείμενα σε ταξινομημένη σειρά, το κάθε ένα με τους βαθμούς του. Τότε να ζητά τα επόμενα 10 κ . ο . κ Και στις δύο περιπτώσεις αυτό ανταποκρίνεται σε ότι έχουμε αναφέρει σαν «ταξινομημένη προσπέλαση».

Υπάρχει ένας άλλος δρόμος από τον οποίο θα περιμένουμε το σύστημα να επικοινωνήσει με το υποσύστημα. Ειδικά, το σύστημα μπορεί να ζητήσει από το υποσύστημα το βαθμό κάθε δοθέντος αντικειμένου. Αυτό αναφέρεται σε ότι έχουμε πει μέχρι τώρα ως «τυχαία προσπέλαση». Στη πραγματικότητα , το QBIC επιτρέπει μαζί ταξινομημένη και τυχαία προσπέλαση.

Υπάρχουν κάποιες περιπτώσεις όπου στο σύστημα δεν επιτρέπει τυχαία προσπέλαση σε κάποιο υποσύστημα. Ένα παράδειγμα είναι όταν το σύστημα-μεσάζων είναι ένα σύστημα ανάκτησης κειμένου, και τα υποσυστήματα είναι μηχανές αναζήτησης. Έτσι, δεν μοιάζει να υπάρχει ένας τρόπος για να ζητηθεί από μία μεγάλη μηχανή αναζήτησης στο δίκτυο ο εσωτερικός βαθμός κάποιου κειμένου της επιλογής μας από μία επερώτηση.

Η μέτρηση του κόστους αρχικά αναφέρεται σε αυτό που προκύπτει από σύστημα το οποίο επεξεργάζεται πληροφορίες που έρχονται σε αυτό από ένα υποσύστημα όπως το QBIC. Όπως πριν , αν υπάρχουν s ταξινομημένες προσπελάσεις και r τυχαίες, τότε το μέσο κόστος θεωρείται ως το $sc_S + rc_R$, για κάποιες θετικές σταθερές c_R , c_S . Το ότι τα c_R , c_S μπορεί να διαφέρουν αντανακλά το γεγονός ότι το κόστος σε ένα σύστημα από ταξινομημένη προσπέλαση και από μία τυχαία προσπέλαση μπορεί να διαφέρει.

Ο αλγόριθμος του FAGIN

Σε αυτή τη παράγραφο, θα ασχοληθούμε με τον αλγόριθμο FA(Fagin's Algorithm).Αυτός ο αλγόριθμος υλοποιήθηκε στο Garlic, ένα πειραματικό σύστημα της IBM. Ο FA λειτουργεί ως εξής:

1. Κάνε ταξινομημένη προσπέλαση παράλληλα για κάθε μία από τις m ταξινομημένες λίστες Li.(«παράλληλα» σημαίνει ότι γίνεται

προσπέλαση στα κορυφαία μέλη των λιστών μέσα από ταξινομημένη προσπέλαση, μετά προσπελαύνουμε το δεύτερο μέλος κάθε μίας από τις λίστες κ.ο.κ). Περίμενε μέχρι να υπάρξουν τουλάχιστον k «ταιριάσματα» ,δηλαδή περίμενε μέχρι να υπάρχει ένα σύνολο από τουλάχιστον k αντικείμενα τέτοια ώστε κάθε αντικείμενο να έχει ειδωθεί σε κάθε μία από τις m λίστες.

2. Για κάθε αντικείμενο R που έχει ειδωθεί , κάνε τυχαία προσπέλαση όπως απαιτείται σε κάθε μία από τις λίστες Li, για να βρεις το ιοστό πεδίο x_i του R.
3. Υπολόγισε το βαθμό $t(R) = t(x_1,\ldots,x_m)$ για κάθε αντικείμενο R που έχει ειδωθεί. Έστω Y είναι ένα σύνολο που περιλαμβάνει τα k αντικείμενα που έχουν ειδωθεί με το υψηλότερο βαθμό. Η έξοδος είναι τότε το βαθμολογημένο σύνολο $\{R,t(R))\mid R \in Y\}$.

Είναι εύκολο να δείξουμε ότι αυτός ο αλγόριθμος είναι σωστός για μονότονες αθροιστικές συναρτήσεις t(το οποίο σημαίνει ότι, ο αλγόριθμος με επιτυχία βρίσκει τις κορυφαίες κ απαντήσεις).Αν υπάρχουν N αντικείμενα στη βάση δεδομένων, και αν οι εντολές στις ταξινομημένες λίστες είναι πιθανοκρατικά ανεξάρτητες, τότε το μέσο κόστος του FA είναι $x_1, x_2 \ldots, x_l, 0, \ldots, 0$ $x_i \leq xi$, $O(N^{(m-1)/m}k^{1/m})$ $O(N^{(m-1)/m}k^{1/m})$, με αυθαιρέτως μεγάλη πιθανότητα.

Μία αθροιστική συνάρτηση t είναι αυστηρή εάν η σχέση $t(x_1,\ldots,x_m) = 1$ ισχύει ακριβώς όταν $x_i = 1$ για κάθε i. Έτσι, μία αθροιστική συνάρτηση είναι αυστηρή όταν παίρνει σαν μεγαλύτερη τιμή το 1 ακριβώς όταν κάθε όρισμα παίρνει τη μεγαλύτερη τιμή. Θα περιμέναμε σίγουρα μία αθροιστική συνάρτηση που αναπαριστά την συνένωση να είναι αυστηρή
Ο Fagin δείχνει ότι ο αλγόριθμος του είναι κατάλληλος με υψηλή πιθανότητα στη χειρότερη περίπτωση αν η αθροιστική συνάρτηση είναι αυστηρή(έτσι ώστε, αρχικά, να ασχολούμαστε κατά μία έννοια με τη συνένωση), και αν οι σειρές στις διατεταγμένες λίστες είναι ανεξάρτητες με την έννοια των πιθανοτήτων. Στη πραγματικότητα, το πρότυπο προσπέλασης του FA είναι άσχετο με την επιλογή της αθροιστικής συνάρτησης , και έτσι μέχρι τώρα για κάθε καθορισμένη βάση δεδομένων, το μέσο κόστος του FA είναι ακριβώς το ίδιο άσχετα από το ποια είναι η αθροιστική συνάρτηση. Αυτό αληθεύει ακόμη και για μία σταθερή αθροιστική συνάρτηση .Σε αυτή τη περίπτωση ,φυσικά, υπάρχει ένας ασήμαντος αλγόριθμος που δίνει τις κορυφαίες k απαντήσεις με O(1) μέσο κόστος. Έτσι ο FA δεν είναι βέλτιστος με την οποιαδήποτε έννοια για κάποιες μονότονες αθροιστικές συναρτήσεις t.Σαν ένα ακόμη ενδιαφέρον παράδειγμα , όταν η αθροιστική συνάρτηση είναι η max(η οποία δεν είναι αυστηρή)(βλέπε [12]) ,υπάρχει ένα απλός αλγόριθμος που κάνει το πολύ mk ταξινομημένες προσπελάσεις και καθόλου τυχαίες προσπελάσεις που βρίσκουν τις κορυφαίες k απαντήσεις. Εν αντιθέσει ,όπως θα δούμε, ο

αλγόριθμος ΤΑ είναι βέλτιστος κατά το στιγμιότυπο για κάθε μονότονη αθροιστική συνάρτηση , κάτω από πολύ ασθενείς υποθέσεις.

Ακόμη και στις περιπτώσεις όπου ο FA είναι βέλτιστος, αυτή η καταλληλότητα κρατά μόνο στις χειρότερες περιπτώσεις με υψηλή πιθανότητα. Αυτό αφήνει ανοιχτή τη πιθανότητα να υπάρχουν κάποιοι αλγόριθμοι που έχουν πολύ καλύτερο μέσο κόστος από τον FA πάνω από συγκεκριμένες βάσεις δεδομένες. Ο αλγόριθμος ΤΑ , τον οποίο τώρα συζητάμε είναι ένας τέτοιος αλγόριθμος.

Threshold algorithm (TA)

Τώρα θα παρουσιάσουμε τον ΤΑ.

1. Κάνε ταξινομημένη προσπέλαση παράλληλα σε κάθε μια από τις m ταξινομημένες λίστες L_i. Καθώς ένα αντικείμενο R βλέπεται κάτω από ταξινομημένη προσπέλαση σε κάποιες λίστες , κάνε τυχαία προσπέλαση στις υπόλοιπες λίστες για να βρεις το βαθμό xi του αντικειμένου R σε κάθε λίστα L_i. Τότε υπολόγισε το βαθμό t(R) = $t(x_1,\ldots,x_m)$ του αντικειμένου R. Αν αυτός ο βαθμός είναι ένας από τους k υψηλότερους που έχεις βρει, τότε να θυμάσαι το αντικείμενο R και το βαθμό του t(R).
2. Για κάθε λίστα L_i, έστω xi ότι είναι ο βαθμός του τελευταίου αντικειμένου που είναι ορατός κάτω από ταξινομημένη προσπέλαση. Ορίζουμε τη τιμή κατωφλίου να είναι ίση με t(x1,….xm). Όταν τουλάχιστον κ αντικείμενα έχουν ειδωθεί των οποίων ο βαθμός είναι τουλάχιστον ίσος με το t, τότε σταμάτα.
3. Έστω Y είναι ένα σύνολο που περιέχει τα k αντικείμενα τα οποία έχουν ειδωθεί με τους υψηλότερους βαθμούς. Η έξοδος είναι τότε το βαθμολογημένο σύνολο $\{ (R , t(R)) | R \in Y \}$.

Τώρα θα δείξουμε ότι ο ΤΑ είναι σωστός για κάθε μονότονη αθροιστική συνάρτηση t.

Θεώρημα 2.1 : Αν μία αθροιστική συνάρτηση t είναι μονότονη ,τότε ο ΤΑ σωστά βρίσκει τις κορυφαίες k απαντήσεις.

Απόδειξη:
Έστω ότι Y είναι όπως στο βήμα 3 του αλγορίθμου. Χρειαζόμαστε να δείξουμε ότι κάθε μέλος του Y έχει τουλάχιστον τόσο ψηλά ένα βαθμό όσο κάθε αντικείμενο z που δεν είναι στο Y. Από τον ορισμό του Y,αυτή είναι η περίπτωση για κάθε αντικείμενο z που έχει ειδωθεί όταν τρέχει ο αλγόριθμος ΤΑ. Έτσι υποθέτουμε ότι ο z δεν έχει ειδωθεί. Υποθέτουμε ότι

τα πεδία του z είναι τα $x_1,\ldots,x_m$.Έτσι , $x_i \leq x_i$ για κάθε i. Για αυτό το λόγο $t(z) = t(x_1,\ldots,x_m) \leq t(x_1,\ldots,x_m) = \tau$, όπου η ανισότητα συνεπάγεται από τη μονοτονία του t. Αλλά από τον ορισμό του Υ για κάθε y στο Υ έχουμε $t(y) \geq \tau$. Έτσι για κάθε y στο Υ έχουμε έχουμε $t(y) \geq \tau \geq t(z)$, όπως επιθυμούμε.

Τώρα θα δείξουμε ότι ο κανόνας παύσης του ΤΑ πάντα συμβαίνει τουλάχιστον τόσο νωρίς όσο ο κανόνας παύσης για τον FA. Στον FA, αν το R είναι ένα αντικείμενο που έχει εμφανιστεί κάτω από ταξινομημένη προσπέλαση σε κάθε λίστα ,τότε λόγω μονοτονίας, ο βαθμός του R είναι τουλάχιστον ίσος με τη τιμή του κατωφλίου. Έτσι, όταν υπάρχουν τουλάχιστον k αντικείμενα ,καθένα το οποίο έχει εμφανιστεί κάτω από ταξινομημένη προσπέλαση σε κάθε λίστα(ο κανόνας παύσης του FA) , υπάρχουν τουλάχιστον k αντικείμενα των οποίων ο βαθμός είναι τουλάχιστον ίσος με τη τιμή του κατωφλίου.

Αυτό συνεπάγεται ότι για κάθε βάση δεδομένων, το κόστος ταξινομημένης προσπέλασης για το ΤΑ είναι το πολύ τόσος όσο για τον FA. Αυτό όμως δεν συνεπάγεται ότι το μέσο κόστος του ΤΑ είναι πάντα το πολύ όσο του FA , αφού ο ΤΑ μπορεί να κάνει περισσότερες τυχαίες προσπελάσεις από τον FA.Πάντως, αφού το μέσο κόστος του ΤΑ είναι το πολύ το κόστος ταξινομημένης προσπέλασης επί μία σταθερά (ανεξάρτητη από μέγεθος της βάσης δεδομένων), συνεπάγεται ότι το μέσο κόστος του ΤΑ είναι το πολύ μία σταθερά επί αυτό του FA. Στη πραγματικότητα, θα δείξουμε ότι ο ΤΑ είναι βέλτιστος κατά το στιγμιότυπο κάτω από κανονικές υποθέσεις.

Τώρα ας ασχοληθούμε λίγο πίσω από τον ΤΑ. Για απλότητα, θα συζητήσουμε πρώτα τη περίπτωση όπου k = 1 ,αυτό είναι , όταν ο χρήστης προσπαθεί να καθορίσει τη κορυφαία απάντηση. Υποθέτουμε ότι υπάρχει ένα επίπεδο στον αλγόριθμο όπου δεν έχουμε ακόμη δει κάποιο αντικείμενο που ο βαθμός του είναι τουλάχιστον τόσο μεγάλος όσο η τιμή κατωφλίου τ. Το θέμα είναι ότι σε αυτό το σημείο δεν ξέρουμε τη κορυφαία απάντηση, αφού το επόμενο αντικείμενο που βλέπουμε κάτω από ταξινομημένη προσπέλαση θα μπορούσε να έχει γενικό βαθμό τ, και για αυτό το λόγο μεγαλύτερο βαθμό από κάθε άλλο που έχει ειδωθεί μέχρι τώρα. Επιπλέον, από τη στιγμή που θα δούμε ένα αντικείμενο του οποίου ο βαθμός είναι τουλάχιστον τ, τότε είναι ασφαλές να σταματήσουμε όπως βλέπουμε και στην απόδειξη του θεωρήματος 1.Έτσι, διαισθητικά ο κανόνας παύσης του ΤΑ λέει: «Κάνε παύση όσο νωρίτερα ξέρεις ότι έχεις δει τη κορυφαία απάντηση». Παρομοίως, για k ο κανόνας παύσης του ΤΑ λέει : «Σταμάτα όσο νωρίτερα ξέρεις ότι έχεις δει τις κορυφαίες k απαντήσεις.».Έτσι θα θεωρούμε τον ΤΑ να είναι μία υλοποίηση του ακόλουθου προγράμματος:

Κάνε ταξινομημένη προσπέλαση (και την αντίστοιχη τυχαία προσπέλαση) μέχρι να ξέρεις ότι έχεις δει τις κορυφαίες κ απαντήσεις.

Αυτό το υψηλού-επιπέδου πρόγραμμα είναι γνωστό σαν βασιζόμενο στη γνώση πρόγραμμα. Στη πραγματικότητα, ο ΤΑ σχεδιάστηκε στα όρια αυτού του βασιζόμενου στη γνώση προγράμματος. Το ότι ο ΤΑ ανταποκρίνεται σε αυτό το βασιζόμενο στη γνώση πρόγραμμα είναι το γεγονός πίσω από τη καταλληλότητα στιγμιότυπου του ΤΑ.

Θεώρημα 2.2 : Ο ΤΑ απαιτεί μόνο περιορισμένους χώρους αποθήκευσης (buffers) ,των οποίων το μέγεθος είναι ανεξάρτητο του μεγέθους της βάσης δεδομένων.

Απόδειξη : Όλα όσα πρέπει να θυμάται ο ΤΑ είναι τα συγκεκριμένα κορυφαία k αντικείμενα και οι βαθμοί τους, και (δείκτες σε) τα τελευταία αντικείμενα που έχουν ειδωθεί σε ταξινομημένη σειρά σε κάθε λίστα.

Εν αντιθέσει , ο FA απαιτεί ενδιάμεσους χώρους αποθήκευσης που μεγαλώνουν πάρα πολύ όσο η βάση δεδομένων μεγαλώνει, αφού ο FA πρέπει να θυμάται κάθε αντικείμενο που έχει δει σε διατεταγμένη σειρά σε κάθε λίστα , προκειμένου να ελέγξουμε για αντικείμενα που ταιριάζουν σε πληθώρα λιστών.

Υπάρχει ένα επιπλέον κόστος για τους περιορισμένους χώρους αποθήκευσης . Κάθε φορά που βρίσκουμε ένα αντικείμενο μέσω ταξινομημένης προσπέλασης, ο ΤΑ μπορεί να κάνει m-1 τυχαίες προσπελάσεις (όπου m είναι ο αριθμός των λιστών) , για να βρει το βαθμό των αντικειμένων στις άλλες λίστες. Αυτό συμβαίνει παρά το γεγονός ότι αυτά τα αντικείμενα μπορούν να έχουν ειδωθεί σε αυτές τις άλλες λίστες.

Καταλληλότητα στιγμιότυπου

Προκειμένου να συγκρίνουμε τη καταλληλότητα στιγμιότυπου με άλλες έννοιες από τη βιβλιογραφία, γενικεύουμε τον ορισμό που δώσαμε στην εισαγωγή. Έστω **A** είναι μία κλάση αλγορίθμων, και έστω **D** είναι μία κλάση των νόμιμων εισόδων στους αλγόριθμους. Υποθέτουμε ότι ασχολούμαστε με μία συγκεκριμένη μη αρνητική μέτρηση του κόστους cost(A,D) , η οποία αναπαριστά τη ποσότητα των πηγών που καταναλώνονται προκειμένου να τρέξει ο αλγόριθμος A που ανήκει στη κλάση **A** για είσοδο D (που ανήκει στη κλάση **D**). Αυτό το κόστος μπορεί να είναι ο χρόνος τρεξίματος ενός αλγορίθμου A για είσοδο D ,ή στη παρούσα μελέτη, το μέσο κόστος που προκύπτει τρέχοντας τον αλγόριθμο A πάνω από βάση δεδομένων D.

Λέμε ότι ένας αλγόριθμος B είναι **βέλτιστος κατά το στιγμιότυπο** πάνω από τις κλάσεις **A** και **D** αν ο B ανήκει στο **A** και αν για κάθε A που ανήκει στη κλάση **A** και κάθε D που ανήκει στη κλάση **D** έχουμε:

$$\text{cost}(B,D) = O(\text{cost}(A,D)). \qquad (2)$$

Η ισότητα στη (2) σημαίνει ότι υπάρχουν σταθερές c, c' τέτοιες ώστε το $\text{cost}(B,D) \leq c\ \text{cost}(A,D) + c'$ για κάθε επιλογή του A που ανήκει στη κλάση **A** και D που ανήκει στη κλάση **D** . Αναφερόμαστε στο c σαν το λόγο καταλληλότητας. Είναι παρόμοιο με το λόγο ανταγωνιστικότητας στην ανάλυση ανταγωνιστικότητας. Χρησιμοποιούμε τη λέξη «βέλτιστος» για να αντικατοπτρίσουμε το γεγονός ότι το B είναι ουσιαστικά ο καλύτερος αλγόριθμος της κλάσης **A.**

Διαισθητικά, η καταλληλότητα στιγμιότυπου έχει να κάνει με τη καταλληλότητα κάθε στιγμιότυπου , όπως η χειρότερη περίπτωση ή το μέσο κόστος. Υπάρχουν πολλοί αλγόριθμοι που είναι βέλτιστοι στη χειρότερη περίπτωση αλλά δεν είναι βέλτιστοι κατά το στιγμιότυπο. Ένα παράδειγμα είναι η δυαδική αναζήτηση: Στη χειρότερη περίπτωση , η δυαδική αναζήτηση εγγυάται ότι απαιτεί όχι παραπάνω από log N επαναλήψεις για N στοιχεία δεδομένων. Πάντως, για κάθε στιγμιότυπο, μία θετική απάντηση μπορεί να επιτευχθεί σε μία επανάληψη, και μία αρνητική απάντηση σε δύο.

Θεωρούμε ένα μη ντετερμινιστικό αλγόριθμο σωστό αν σε καμία περίπτωση δεν κάνει λάθος. Παίρνουμε το μέσο κόστος ενός μη ντετερμινιστικού αλγορίθμου να είναι το αρχικό ελάχιστο κόστος πάνω από όλες τις περιπτώσεις του αλγορίθμου όπου σταματά με τις κορυφαίες k απαντήσεις. Παίρνουμε το μέσο κόστος ενός ντετερμινιστικού αλγορίθμου να είναι το αναμενόμενο κόστος (πάνω από όλες τις πιθανές επιλογές του αλγορίθμου). Όταν λέμε ότι ένας ντετερμινιστικός αλγόριθμος B είναι βέλτιστοι κατά το στιγμιότυπο πάνω από το **A** και **D** ,τότε πραγματικά συγκρίνουμε το B με τον καλύτερο μη ντετερμινιστικό αλγόριθμο, ακόμη και αν το **A** περιλαμβάνει μόνο ντετερμινιστικούς αλγορίθμους . Μπορούμε να αντιληφθούμε το κόστος του καλύτερου μη ντετερμινιστικού αλγορίθμου που παράγει τις κορυφαίες k απαντήσεις πάνω από ένα σύνολο από βάσεις δεδομένων σαν το κόστος της συντομότερης απόδειξης για αυτή τη βάση δεδομένων για την οποία αυτές είναι πράγματι οι κορυφαίες k απαντήσεις. Έτσι η καταλληλότητα κατά το στιγμιότυπο είναι δυνατή:το κόστος ενός βέλτιστου κατά το στιγμιότυπο αλγόριθμου είναι ουσιαστικά το κόστος τη συντομότερης απόδειξης. Παρομοίως, μπορούμε να δούμε τον A σαν να περιέχει επίσης πιθανοκρατικούς αλγορίθμους που δεν κάνουν ποτέ λάθος. Για πειστικότητα, στις αποδείξεις μας θα υποθέτουμε πάντα ότι το A περιέχει μόνο ντετερμινιστικούς αλγορίθμους, αφού τα αποτελέσματα μεταφέρονται σε μη ντετερμινιστικούς

αλγόριθμους και πιθανοκρατικούς αλγόριθμους που ποτέ δεν κάνουν λάθος.

Ο ορισμός που έχουμε δώσει για την καταλληλότητα κατά το στιγμιότυπο είναι τυπικά ο ίδιος ορισμός ο οποίος χρησιμοποιείται στην ανταγωνιστική ανάλυση, εκτός από το ότι στην ανταγωνιστική ανάλυση δεν υποθέτουμε ότι ο Β ανήκει στο **A** και ότι το cost(A,D) δεν αναπαριστά τυπικά ένα κόστος απόδοσης. Στην ανταγωνιστική ανάλυση, α) το **D** είναι μια κλάση στιγμιότυπων ενός συγκεκριμένου προβλήματος, β) Το **A** είναι μία κλάση offline αλγορίθμων που δίνουν μία λύση στα στιγμιότυπα του **D** γ) Η cost(A,D) είναι ένας αριθμός που αναπαριστά το πόσο καλή είναι η λύση(όπου μεγαλύτεροι αριθμοί ανταποκρίνονται σε μία χειρότερη λύση) και δ) ο Β είναι ένας ειδικός online αλγόριθμος. Σε αυτή τη περίπτωση, ο online αλγόριθμος λέγεται ανταγωνιστικός. Το θέμα είναι ότι ένας ανταγωνιστικός αλγόριθμος μπορεί να αποδίδει φτωχά σε ορισμένα στιγμιότυπα, αλλά μόνο σε στιγμιότυπα όπου κάθε offline αλγόριθμος θα αποδίδει επίσης φτωχά.

Ένα άλλο παράδειγμα όπου το ζήτημα της καταλληλότητας κατά το στιγμιότυπο εμφανίζεται ,αλλά επίσης χωρίς την υπόθεση ότι ο Β ανήκει στη κλάση **A** και επίσης όπου το cost(A,D) δεν αναπαριστά ένα κόστος απόδοσης είναι στο τομέα των προσεγγιστικών αλγορίθμων. Σε αυτή τη περίπτωση α) Το **D** είναι μια κλάση στιγμιότυπων ενός συγκεκριμένου προβλήματος β)Το **A** είναι μία κλάση αλγορίθμων που επιλύουν τα στιγμιότυπα στη κλάση **D** (σε περίπτωση ενδιαφέροντος, αυτοί οι αλγόριθμοι δεν είναι πολυωνυμικοί ως προς το χρόνο) γ) Το cost(A,D) είναι η τιμή της απάντησης -αποτέλεσμα όποτε ο αλγόριθμος Α εφαρμόζεται σε είσοδο D και δ)ο Β είναι ένας συγκεκριμένος πολυωνυμικός κατά το χρόνο αλγόριθμος.

Υπάρχει επίσης ένα ενδιαφέρον παράδειγμα (Dagnum) για το τι θα ονομάζουμε καταλληλότητα κατά το στιγμιότυπο. Θεωρείται το πρόβλημα σαν ο καθορισμός της έννοιας μίας άγνωστης τυχαίας μεταβλητής με την εκτίμηση Monte Carlo. Σε αυτή τη περίπτωση, α)Το **D** είναι ένα σύνολο από μεταβλητές που μεταβάλλονται στο εσωτερικό [0,1] β)Το **A** είναι η κλάση αλγορίθμων που επανειλημμένως κάνουν ανεξάρτητες αξιολογήσεις μίας τυχαίας μεταβλητής και μετά βρίσκουν το μέσο όρο των αποτελεσμάτων, πετυχαίνουν μία εκτίμηση της έννοιας της τυχαίας μεταβλητής μαζί με μία δοσμένη ακρίβεια και δοσμένη πιθανότητα. γ) Το cost(A,D) είναι ο αναμενόμενος αριθμός ανεξάρτητων αξιολογήσεων της τυχαίας μεταβλητής D υπό τον αλγόριθμο Α και δ)Ο Β είναι αλγόριθμος , που καλείται Α Α για «προσεγγιστικός αλγόριθμος»(«approximation algorithm"). Το βασικό συμπέρασμα είναι ότι ο Α Α είναι κατάλληλος κατά το στιγμιότυπο πάνω από τους Α , D.

Έναν ακόμη παράδειγμα που δίνεται από τον Demaine , είναι ένας αλγόριθμος που είναι κοντά στο να θεωρείται βέλτιστος κατά το

στιγμιότυπο. Έχει να κάνει με το πρόβλημα της εύρεσης της τομής, συνένωσης ή διαφοράς μίας συλλογής ταξινομημένων συνόλων. Σε αυτή τη περίπτωση, α) Η **D** είναι η κλάση των στιγμιότυπων συλλογών διατεταγμένων συνόλων, β)Η **A** είναι η κλάση των αλγορίθμων που συνδυάζουν συγκρίσεις μεταξύ στοιχείων, γ) Το cost(A,D) είναι ο χρόνος τρεξίματος (αριθμός συγκρίσεων) στον τρέχοντα αλγόριθμο της Α στο στιγμιότυπο D και δ) Ο Β είναι ο αλγόριθμος τους. Κατά μία έννοια , αυτός ο αλγόριθμος είναι κοντά σε αυτό που ονομάζουμε βέλτιστος κατα το στιγμιότυπο.

Καταλληλότητα κατά το στιγμιότυπο για τον ΤΑ

Σε αυτή τη παράγραφο , ερευνούμε την καταλληλότητα κατά το στιγμιότυπο του αλγορίθμου threshold algorithm (TA). Ξεκινάμε θεωρώντας ότι ο ΤΑ είναι βέλτιστος κατά το στιγμιότυπο. Αν το Α είναι ένας αλγόριθμος που σταματάει νωρίτερα από τον ΤΑ για κάποια βάση δεδομένων , πριν ο Α βρει k αντικείμενα των οποίων ο βαθμός είναι τουλάχιστον ίσος με τι τιμή κατωφλίου τ , τότε ο Α πρέπει να κάνει ένα λάθος σε κάποιες βάσεις δεδομένων, αφού το επόμενο αντικείμενο σε κάθε λίστα μπορεί να έχει βαθμό xi σε κάθε λίστα i, και έτσι να έχει βαθμό $t(x_1,\ldots,x_m) = \tau$. Αυτό το καινούργιο αντικείμενο ,το οποίο ο Α δεν έχει ακόμη ,έχει μεγαλύτερο βαθμό από κάποιο αντικείμενο στην λίστα των κορυφαίων k το οποίο παρήχθη από το Α και έτσι το Α έσφαλλε που σταμάτησε τόσο νωρίς. Θα θέλαμε να μετατρέψουμε αυτή την αρχική διαίσθηση σε μία απόδειξη για κάθε μονότονη αθροιστική συνάρτηση, ο ΤΑ είναι βέλτιστος κατά το στιγμιότυπο πάνω από όλους τους αλγόριθμους που σωστά βρίσκουν τις κορυφαίες k απαντήσεις, πάνω από τη κλάση όλων των βάσεων δεδομένων. Πρώτα κάνουμε μία αντιστοίχηση ανάμεσα στους αλγορίθμους που κάνουν «άγρια μαντεψιά» (αυτό σημαίνει ότι ,υλοποιούν τυχαία προσπέλαση σε αντικείμενα που δεν έχουν συναντηθεί με διατεταγμένη προσπέλαση) και αυτούς που δεν κάνουν. (ούτε ο FA ούτε ο ΤΑ κάνουν «άγρια μαντεψιά»,ούτε κανένας φυσικός αλγόριθμος στην εργασία μας).Το πρώτο μας θεώρημα (3.1) λέει ότι για κάθε μονότονη αθροιστική συνάρτηση , ο ΤΑ είναι βέλτιστος κατά το στιγμιότυπο πάνω από όλους τους αλγορίθμους που βρίσκουν σωστά τις κορυφαίες k απαντήσεις και δεν κάνουν ισχυρές μαντεψιές, πάνω από τη κλάση όλων των βάσεων δεδομένων. Θα δείξουμε ότι αυτή η αντίθεση (ισχυρές μαντεψιές έναντι ανίσχυρες μαντεψιές) είναι ουσιαστική: αν αλγόριθμοι που κάνουν ισχυρές μαντεψιές επιτρέπονται στη κλάση **A** των αλγορίθμων που ένας αλγόριθμος βέλτιστος κατά το στιγμιότυπο πρέπει να ανταγωνίζεται ,τότε κανένας αλγόριθμος δεν είναι βέλτιστος κατά το στιγμιότυπο.(παράδειγμα 3.3 θεώρημα 3.4). Η ουσία αυτού του παραδείγματος είναι (και το αντίστοιχο θεώρημα) είναι το γεγονός ότι

μπορεί να υπάρχουν πολλαπλά αντικείμενα με το ίδιο βαθμό σε κάποια λίστα. Πράγματι, αν εστιάσουμε τη προσοχή μας σε βάσεις δεδομένων όπου όχι δύο αντικείμενα έχουν την ίδια τιμή στην ίδια λίστα, και κάνουμε ένα φυσικό επιπρόσθετο περιορισμό στην αθροιστική συνάρτηση πέρα από τη μονοτονία ,τότε ο ΤΑ είναι βέλτιστος κατά το στιγμιότυπο πάνω από όλους τους αλγορίθμους που σωστά βρίσκουν τις κορυφαίες k απαντήσεις.(θεώρημα 3.5).

Τώρα θα δώσουμε το πρώτο θετικό αποτέλεσμα στη καταλληλότητα κατά το στιγμιότυπο του ΤΑ. Λέμε ότι ένας αλγόριθμος κάνει ισχυρές μαντεψιές αν κάνει τυχαία προσπέλαση για να βρει βαθμούς κάποιου αντικειμένου R σε κάποια λίστα πριν ο αλγόριθμος δει το R υπό διατεταγμένη προσπέλαση . Αυτό σημαίνει , ότι ένας αλγόριθμος κάνει ισχυρές μαντεψιές αν ο πρώτος βαθμός που επιτυγχάνει για κάποιο αντικείμενο R γίνεται υπό τυχαία προσπέλαση. Δεν θα υλοποιήσουμε αλγορίθμους που κάνουν ισχυρές μαντεψιές. Στη πραγματικότητα, υπάρχουν κάποια περιβάλλοντα όπου δεν θα είναι καν δυνατό να γίνουν ισχυρές μαντεψιές.(ένα τέτοιο περιβάλλον βάσης δεδομένων όπου ο αλγόριθμος δεν ξέρει το όνομα του αντικειμένου δεν έχει ακόμη ειδωθεί.) Πάντως, κάνοντας μία τυχερή ισχυρή μαντεψιά ,μπορεί να βοηθήσει .(όπως θα δούμε και παρακάτω ,παράδειγμα 3.3).

Τώρα θα δείξουμε την καταλληλότητα κατά το στιγμιότυπο του ΤΑ μεταξύ αλγορίθμων που δεν κάνουν ισχυρές μαντεψιές. Σε αυτό το θεώρημα, όταν θέτουμε ως **D** την κλάση όλων των βάσεων δεδομένων , εννοούμε πραγματικά ότι το **D** είναι η κλάση όλων των βάσεων δεδομένων όπου εμπλέκονται διατεταγμένες λίστες που αντιστοιχίζονται στις παραμέτρους της αθροιστικής συνάρτησης t.
Θέτουμε ως k(όπου ψάχνουμε να βρούμε τις κορυφαίες k απαντήσεις) και την αθροιστική συνάρτηση t να είναι σταθερή. Αφού θέτουμε τη συνάρτηση t ως σταθερή , παίρνουμε τον αριθμό m των ορισμάτων της t (εννοείται ο αριθμός των διατεταγμένων λιστών) να είναι αμετάβλητος.

Θεώρημα 3.1. Υποθέτουμε ότι η αθροιστική συνάρτηση t είναι μονότονη. Έστω **D** είναι η κλάση όλων των βάσεων δεδομένων. Έστω **A** είναι η κλάση όλων των αλγορίθμων που ορθά βρίσκουν τις κορυφαίες k απαντήσεις για κάθε t για κάθε βάση δεδομένων και που δεν κάνουν ισχυρές μαντεψιές. Τότε ο ΤΑ είναι κατάλληλος κατά το στιγμιότυπο πάνω από τα **A,D**.

Απόδειξη:

Υποθέτουμε ότι A που ανήκει στη κλάση **A**, και ότι ο αλγόριθμος A τρέχει πάνω από τη βάση δεδομένων D .Υποθέτουμε ότι ο αλγόριθμος A σταματά σε βάθος d (αυτό σημαίνει ότι , αν di είναι ο αριθμός των αντικειμένων που βλέπονται υπό ταξινομημένη προσπέλαση στη λίστα i ,για $1 \leq i \leq m$,τότε $d = \max_i d_i$).Υποθέτουμε ότι το A βλέπει a ξεχωριστά

αντικείμενα (κάποια πιθανώς πολλαπλές φορές). Ειδικά όταν $a \geq d$. Αφότου το Α δεν κάνει ισχυρές μαντεψιές , και βλέπει α ξεχωριστά αντικείμενα , πρέπει να κάνει τουλάχιστον α ταξινομημένες προσπελάσεις, και έτσι το μέσο κόστος του είναι τουλάχιστον $a c_S$. Θα δείξουμε ότι ο ΤΑ σταματά στη D σε μήκος a + k. Έτσι, το μέσο κόστος του ΤΑ είναι το πολύ $(a + k)\ ms_c + (a + k)\ m\ (m-1)\ c_R$, το οποίο είναι $amcs + am(m-1)\ c_R$ συν μία επιπρόσθετη σταθερά , το $kmcs + km(m-1)\ c_R$. Έτσι ο λόγο καταλληλότητας του ΤΑ είναι το πολύ

$$\frac{amc_S + am(m-1)c_R}{ac_S} \frac{amc_S + am(m-1)c_R}{ac_S} = m + m(m-1)\, c_R / c_S$$. Αργότερα, θα δείξουμε ότι αν η αθροιστική συνάρτηση είναι αυστηρή, τότε αυτό είναι ακριβώς ο λόγος καταλληλότητας του ΤΑ, και αυτό είναι το καλύτερο δυνατό.)

Σημειώνουμε ότι για κάθε επιλογή του d',ο αλγόριθμος ΤΑ βλέπει το πολύ d' αντικείμενα μήκους d'(αυτό συμβαίνει γιατί με μήκος d' γίνονται md' ταξινομημένες προσπελάσεις ,και κάθε αντικείμενο προσπελαύνεται το πολύ m φορές κάτω από ταξινομημένη προσπέλαση.).Έστω ότι Υ είναι το παράγωγο σύνολο από το Α (το οποίο αποτελείται από τα κορυφαία k αντικείμενα).Αν υπάρχουν το πολύ k αντικείμενα τα οποία ο Α δεν βλέπει ,τότε ο ΤΑ σταματά σε μήκος α + κ (αφού έχει δει κάθε αντικείμενο), και έχουμε ολοκληρώσει. Έτσι υποθέτουμε ότι υπάρχου τουλάχιστον κ + 1 αντικείμενα που ο Α δεν βλέπει. Αφού το Υ είναι μεγέθους k , υπάρχει κάποιο αντικείμενο V το οποίο ο Α δεν βλέπει και το οποίο δεν είναι στο Υ.

Έστω τΑ είναι η τιμή κατωφλίου όταν ο αλγόριθμος Α σταματά. Αυτό σημαίνει ότι αν ο xi είναι ο βαθμός του τελευταίου αντικειμένου που έχει ειδωθεί υπό ταξινομημένη προσπέλαση στη λίστα i για τον αλγόριθμο Α ,για $1 \leq i \leq m$ τότε $\tau A = t(x_1,\ldots,x_m)$)(αν η λίστα i δεν έχει προσπελαστεί υπό ταξινομημένη προσπέλαση ,θέτουμε $x_i = 1$). Ας καλούμε ένα αντικείμενο R μεγάλο αν $t(R) \geq \tau_A$, και αλλιώς ας το καλούμε μικρό.

Τώρα θα δείξουμε ότι κάθε μέλος R του Υ είναι μεγάλο. Ορίζουμε μία βάση δεδομένων D' να είναι όπως η D , εκτός και αν το αντικείμενο V έχει βαθμό xi στην λίστα i (για $1 \leq i \leq m$). Ο αλγόριθμος Α έχει R ,αλλά όχι V, στην έξοδο του για βάση δεδομένων D'. Αφού οι βαθμοί του V στο D' είναι τΑ, ακολουθείται από την ορθότητα του Α ότι το R είναι μεγάλο, όπως επιθυμούμε.

Υπάρχουν τώρα δύο περιπτώσεις, εξαρτάται από το αν ο αλγόριθμος Α βλέπει ή δεν βλέπει κάθε μέλος του συνόλου εξόδου του Υ.

Περίπτωση 1: Ο αλγόριθμος Α βλέπει κάθε μέλος του Υ. Τότε με μήκος d, ο ΤΑ θα βλέπει κάθε μέλος του Υ. Αφού , όπως θα δείξουμε, κάθε μέλος του Υ είναι μεγάλο, συνεπάγεται ότι ο ΤΑ θα σταματά σε μήκος $d \leq a \leq a+k$,όπως επιθυμούμε.

Περίπτωση 2: ο Αλγόριθμος Α δεν βλέπει κάποια μέλη R του Υ. θα δείξουμε ότι κάθε αντικείμενο R' που δεν βλέπεται από το Α πρέπει να είναι μεγάλο. Ορίζουμε μία βάση δεδομένων D' να είναι σαν τη D για κάθε αντικείμενο που βλέπεται από το Α. Έστω ο βαθμός του V στη λίστα i είναι xi , και έστω ότι τοποθετούμε το V κάτω από όλα τα άλλα αντικείμενα με βαθμό xi στη λίστα i (για $1 \leq i \leq m$). Συνεπώς , ο βαθμός του V στη βάση δεδομένων D' είναι τ_A . Αφού το Α δεν μπορεί να διακριθεί μεταξύ D και D' , έχει την ίδια έξοδο στο D και D'. Αφού το Α δεν βλέπει το R και δεν βλέπει ούτε και το R', δεν έχει καμία πληροφορία για να διακρίνει μεταξύ του R και του R'. Έτσι , πρέπει να είναι ικανή να δώσει R' στην έξοδο χωρίς να κάνει κάποιο λάθος. Αλλά αν το R' είναι στην έξοδο και όχι το V ,τότε με ορθότητα του Α ,συνεπάγεται ότι το R' είναι μεγάλο. Έτσι το R' είναι μεγάλο ,όπως επιθυμούμε.

Αφού ο Α βλέπει α αντικείμενα , και αφού ο ΤΑ βλέπει τουλάχιστον $\alpha + k$ αντικείμενα μήκους $\alpha + k$,συνεπάγεται ότι για μήκος $\alpha + k$, ο ΤΑ βλέπει τουλάχιστον k αντικείμενα τα οποία δεν τα έχει δει ο Α . Δείξαμε ότι κάθε αντικείμενο που δεν βλέπεται από τον του Α , είναι μεγάλο. Έτσι , με μήκος $\alpha + k$, ο ΤΑ βλέπει τουλάχιστον k μεγάλα αντικείμενα. Έτσι , ο ΤΑ σταματά με μήκος $\alpha + k$, όπως είναι το επιθυμητό.

Το επόμενο αποτέλεσμα είναι ένα πόρισμα της απόδειξης του θεωρήματος 1 .Ειδικότερα , στην απόδειξη του θεωρήματος 1 , δείξαμε ότι κάτω από την υπόθεση του θεωρήματος 1 (όχι ισχυρή μαντεψιά), ο λόγος καταλληλότητας του ΤΑ είναι το πολύ $m + m(m - 1)\, c_R / c_S$. Το επόμενο αποτέλεσμα λέει ότι αν η αθροιστική συνάρτηση είναι αυστηρή , τότε ο λόγος καταλληλότητας είναι ακριβώς αυτή η τιμή, και αυτό είναι το καλύτερο δυνατό. Υπενθυμίζουμε εδώ ότι μία αθροιστική συνάρτηση είναι αυστηρή αν η $t(x_1,\ldots,x_m) = 1$ κρατάει ακριβώς όταν $x_i = 1$ για κάθε i. Διαισθητικά , αυστηρότητα σημαίνει ότι η αθροιστική συνάρτηση αναπαριστά κατά μία έννοια τη συνένωση.

ΠΟΡΙΣΜΑ 3.2 Έστω t είναι μία αυθαίρετη μονότονη αυστηρή αθροιστική συνάρτηση με m ορίσματα. Έστω **D** είναι η κλάση όλων των βάσεων δεδομένων. Έστω **A** είναι η κλάση όλων των αλγορίθμων που ορθά βρίσκουν τις κορυφαίες k απαντήσεις για τη t για κάθε βάση δεδομένων και που δεν κάνουν ισχυρές μαντεψιές. Τότε ο ΤΑ είναι βέλτιστος κατά το στιγμιότυπο πάνω από τα **A** και **D** , με λόγο καταλληλότητας $m + m(m-1)\, c_R / c_S$. Κανένας ντετερμινιστικός αλγόριθμος δεν έχει χαμηλότερό λόγο καταλληλότητας.

Δεν μπορούμε να προσπεράσουμε την υπόθεση της αυστηρότητας στο Πόρισμα 3.2. Για παράδειγμα, έστω η αθροιστική συνάρτηση είναι η max. (Η οποία δεν είναι αυστηρή). Είναι εύκολο να δούμε ότι ο ΤΑ σταματά μετά

από k γύρους ταξινομημένης προσπέλασης, και ο λόγος καταλληλότητας είναι ο m (ο οποίος, μπορούμε να προσθέσουμε, είναι το καλύτερο δυνατό για τη max).

Τι γίνεται στη περίπτωση που μας ενδιαφέρει μόνο το κόστος ταξινομημένης προσπέλασης ? Αυτό αντιστοιχίζεται σε $c_R = 0$. Τότε θα δούμε από το πόρισμα 3.2 ότι ο λόγος καταλληλότητας του ΤΑ είναι m.Επιπλέον , αποδεικνύεται επίσης ότι αν η αθροιστική συνάρτηση είναι αυστηρή και αν $c_R = 0$, τότε αυτό είναι το καλύτερο δυνατό: κανένας ντετερμινιστικός αλγόριθμος δεν έχει χαμηλότερο λόγο καταλληλότητας από το m.

Τι γίνεται στη περίπτωση που μας ενδιαφέρει μόνο το κόστος τυχαίας προσπέλασης? Αυτό αντιστοιχίζεται σε $c_S = 0$. Σε αυτή τη περίπτωση, ο ΤΑ είναι μακριά από το να θεωρείται βέλτιστος κατά το στιγμιότυπο. Ο απλοϊκός αλγόριθμος , ο οποίος υλοποιεί ταξινομημένη προσπέλαση σε κάθε αντικείμενο σε κάθε λίστα, δεν κάνει καθόλου τυχαίες προσπελάσεις, και έτσι έχει ταξινομημένη προσπέλαση που κοστολογείται ως 0.

Τώρα θα δείξουμε ότι το να κάνει κανείς μία τυχερή ισχυρή μαντεψιά μπορεί να βοηθήσει.

Παράδειγμα 3.3. Υποθέτουμε ότι υπάρχουν 2 n+1 αντικείμενα , τα οποία θα καλούμε απλά 1, 2...., 2n + 1 και υπάρχουν δύο λίστες L_1 και L_2. Υποθέτουμε ότι στη λίστα L_1, τα αντικείμενα είναι στη σειρά 1, 2...., 2n+1 ,όπου τα κορυφαία n+1 αντικείμενα 1,2....n+1 έχουν όλα βαθμό 1, και τα υπόλοιπα n αντικείμενα n + 2, n + 3 ,..., 2n +1 έχουν όλα βαθμό 0. Υποθέτουμε ότι στη λίστα L_2 , τα αντικείμενα είναι στην αντίθετη σειρά 2n+1, 2n ,.....,1 όπου τα n αντικείμενα 1,....n έχουν όλα τον ίδιο βαθμό 0, και τα υπόλοιπα n+1, n+2.....2n+1 έχουν όλα βαθμό 1. Υποθέτουμε ότι η αθροιστική συνάρτηση είναι η min , και ότι ενδιαφερόμαστε στο να βρούμε την κορυφαία απάντηση (π.χ k = 1). Είναι ξεκάθαρο ότι οι κορυφαίες απαντήσεις είναι τα αντικείμενα n + 1 με γενικό βαθμό 1(κάθε αντικείμενο εκτός από το n + 1 έχει γενικό βαθμό 0).

Ένας αλγόριθμος που κάνει μία ισχυρή μαντεψιά και ψάχνει για το βαθμό του αντικειμένου n+1 σε όλες τις λίστες θα καθορίσει τη σωστή απάντηση και θα είναι ικανός να σταματήσει με ασφαλή τρόπο μετά από δύο τυχαίες προσπελάσεις και όχι ταξινομημένες προσπελάσεις. Παρ' όλαυτα, έστω A να είναι ένας αλγόριθμος (όπως ο ΤΑ) ο οποίος δεν κάνει ισχυρές μαντεψιές. Αφού το αντικείμενο που νικάει n + 1 είναι στο μέσο όλων των ταξινομημένων προσπελάσεων, συνεπάγεται ότι τουλάχιστον n + 1 ταξινομημένες προσπελάσεις θα απαιτούνται πριν ο αλγόριθμος A θα έβλεπε το αντικείμενο που νικάει.

Τι γίνεται στη περίπτωση που θέλουμε να μεγεθύνουμε τη κλάση **A** των αλγορίθμων προκειμένου να επιτρέψουμε επερωτήσεις της φόρμας « Ποιο αντικείμενο έχει το ιοστό βαθμό στη λίστα j, και ποιος είναι ο βαθμός

του στη λίστα j?» . Τότε βλέπουμε από το παράδειγμα 3.3 , όπου αντικαθιστούμε την ισχυρή μαντεψιά με την επερώτηση που ρωτά για το αντικείμενο με (n+1) μεγαλύτερο βαθμό στη κάθε λίστα, τότε ο ΤΑ δεν είναι κατάλληλος κατά το στιγμιότυπο. Έτσι , αυτές οι νέες επερωτήσεις είναι «μόλις τόσο κακές» όπως οι ισχυρές μαντεψιές.

Το παράδειγμα 3.3 δείχνει ότι ο ΤΑ δεν είναι κατάλληλος κατά το στιγμιότυπο πάνω από τη κλάση Α όλων των αλγορίθμων που βρίσκουν τη κορυφαία απάντηση για το min (με δύο ορίσματα) και τη κλάση D όλων των βάσεων δεδομένων. Το επόμενο θεώρημα λέει ότι κάτω από αυτές τις προϋποθέσεις , όχι μόνο ο ΤΑ δεν είναι βέλτιστος κατά το στιγμιότυπο , αλλά ούτε είναι κάποιος άλλος αλγόριθμος.

Θεώρημα 3.4. Έστω **D** είναι η κλάση όλων των βάσεων δεδομένων. Έστω **A** είναι η κλάση όλων των αλγορίθμων που ορθώς βρίσκουν τη κορυφαία απάντηση για τη min (με δύο ορίσματα) για κάθε βάση δεδομένων. Δεν υπάρχει ντετερμινιστικός αλγόριθμος (ή πιθανοκρατικός αλγόριθμος που ποτέ δεν κάνει λάθος) ο οποίος είναι βέλτιστος κατά το στιγμιότυπο πάνω από τα **A,D**.

Απόδειξη. Ας προσπαθήσουμε να επιτύχουμε μία οικογένεια βάσεων δεδομένων, κάθε μία με δύο ταξινομημένες λίστες. Η πρώτη λίστα έχει αντικείμενα 1,2…2n+1 σε κάποια σειρά, με τα κορυφαία n+1 αντικείμενα να έχουν βαθμό 1, και τα εναπομείναντα n αντικείμενα να έχουν βαθμό 0. Η δεύτερη λίστα έχει τα αντικείμενα σε αντίστροφη σειρά , πάλι με τα κορυφαία n+1 αντικείμενα που έχουν βαθμό 1, και τα εναπομείναντα n αντικείμενα που έχουν βαθμό 0. Όπως προηγουμένως υπάρχει ένα μοναδικό αντικείμενο με γενικό βαθμό 1(το αντικείμενο στο μέσο όλων των εντολών),και κάθε εναπομένων αντικείμενο έχει γενικό βαθμό 0.

Έστω Α είναι ένας αυθαίρετος ντετερμινιστικός αλγόριθμος στη κλάση **A**. Θεωρούμε την ακόλουθη διανομή στις βάσεις δεδομένων: κάθε μέλος είναι ως άνω, και η πρώτη λίστα επιλέγεται ανομοιόμορφα με τυχαίο τρόπο (με τη σειρά της δεύτερης λίστας να είναι η αντίστροφη της σειράς από τη πρώτη λίστα). Είναι εύκολο να δούμε ότι ο αναμενόμενος αριθμός των προσπελάσεων (ταξινομημένες και τυχαίες μαζί) του αλγόριθμου Α κάτω από αυτή τη διανομή προκειμένου να δούμε το αντικείμενο που νικάει είναι τουλάχιστον n + 1. Από την άλλη, όπως στο παράδειγμα 3.3 , υπάρχει ένας αλγόριθμος ο οποίος κάνει μόνο 2 τυχαίες προσπελάσεις και καθόλου ταξινομημένες προσπελάσεις. Έτσι, ο λόγος καταλληλότητας μπορεί να είναι αυθαιρέτως μεγάλος.

Για πιθανοκρατικούς αλγόριθμους που ποτέ δεν κάνουν ένα λάθος, μπορούμε να δούμε τη περίπτωση της αρχής του Yao η οποία λέει ότι , το αναμενόμενο κόστος του βέλτιστου ντετερμινιστικού αλγόριθμου για διαδικασία αυθαίρετης εισόδου είναι ένα κάτω όριο στο αναμενόμενο

κόστος του βέλτιστου πιθανοκρατικού αλγορίθμου ο οποίος ποτέ δεν κάνει λάθη.

Αν και , όπως σημειώθηκε νωρίτερα , οι αλγόριθμοι που κάνουν ισχυρές μαντεψιές κανονικά δεν θα υλοποιηθούν στη πράξη ,παραμένει ενδιαφέρον το να ασχοληθούμε μαζί τους. Αν θεωρήσουμε τους αλγορίθμους που επιτρέπουν ισχυρές μαντεψιές , τότε επιτρέπουμε μία μεγαλύτερη κλάση αποδείξεων. Έτσι , στο Παράδειγμα 3.3 , το γεγονός ότι το αντικείμενο n+1 έχει γενικό βαθμό 1 είναι μία απόδειξη ότι βρίσκεται στις κορυφαίες απαντήσεις.

Λέμε ότι μία αθροιστική συνάρτηση t είναι *αυστηρώς μονότονη* αν $t(x_1,\ldots,x_m) < t(x_1',\ldots,x_m')$ όταν $x_i < x_i'$ για κάθε i. Επίσης αν και ο μέσος όρος και το ελάχιστο είναι επίσης αυστηρώς μονότονες συναρτήσεις , υπάρχουν αθροιστικές συναρτήσεις που προτείνονται στη βιβλιογραφία για την αναπαράσταση της συνένωσης και της τομής που είναι μονότονες αλλά όχι αυστηρώς. Λέμε τότε ότι μία βάση δεδομένων D ικανοποιεί την ιδιότητα της ευκρίνειας αν για κάθε i , δεν υπάρχουν δύο αντικείμενα στη D που έχουν τον ίδιο βαθμό στη λίστα L_i , που αυτό συμβαίνει όταν οι βαθμοί στην λίστα L_i είναι ευδιάκριτοι. Τώρα θα δείξουμε ότι αυτές οι συνθήκες εγγυώνται τη καταλληλότητα του ΤΑ ακόμη και για αλγορίθμους που κάνουν ισχυρές μαντεψιές.

ΘΕΩΡΗΜΑ 3.5 Υποθέτουμε ότι η αθροιστική συνάρτηση t είναι αυστηρώς μονότονη. Έστω **D** να είναι η κλάση όλων των βάσεων δεδομένων που ικανοποιούν την ιδιότητα της ευκρίνειας. Έστω **A** να είναι η κλάση όλων των αλγορίθμων που ορθά βρίσκουν τις κορυφαίες k απαντήσεις για κάθε t για κάθε βάση δεδομένων στη κλάση **D**. Τότε ο ΤΑ είναι βέλτιστος κατά το στιγμιότυπο πάνω από τις **A, D**.

Απόδειξη : Υποθέτουμε ότι το A ανήκει στην **A** , και ο αλγόριθμος A τρέχει πάνω από τη βάση δεδομένων D που ανήκει στη **D**. Υποθέτουμε επίσης ότι το A βλέπει α ευδιάκριτα αντικείμενα (κάποια πιθανώς πολλαπλές στιγμές). Θα δείξουμε ότι ο ΤΑ σταματά στη βάση δεδομένων D σε βάθος α + k. Ο ΤΑ κάνει το πολύ $m^2(a+k)$ προσπελάσεις , το οποίο είναι m^2a συν μία προστιθέμενη σταθερά του m^2k. Συνεπάγεται εύκολα ότι ο λόγος καταλληλότητας του ΤΑ είναι το πολύ cm^2, όπου $c = \max\{c_R / c_S , c_S / c_R\}$.

Αν υπάρχουν το πολύ k αντικείμενα τα οποία ο A δεν βλέπει ,τότε ο ΤΑ σταματά σε βάθος a + k (αφού έχει δει κάθε αντικείμενο) , και έχουμε τελειώσει. Έτσι υποθέτουμε ότι υπάρχουν τουλάχιστον k + 1 αντικείμενα τα οποία ο A δεν βλέπει. Αφού το Υ είναι του μεγέθους k , υπάρχει κάποιο αντικείμενο V το οποίο το A δεν το βλέπει και το οποίο δεν είναι στο Υ. Θα δείξουμε ότι ο ΤΑ σταματά στο D σε βάθος α + 1.

Έστω τ είναι η τιμή κατωφλίου του ΤΑ σε βάθος α + 1. Αν , το xi είναι ο βαθμός του (α+1) υψηλότερου αντικειμένου στη λίστα i , τότε τ = $t(x_1,\ldots,x_m)$. Θα ονομάζουμε ένα αντικείμενο R μεγάλο αν ισχύει $t(R) \geq \tau$, στην αντίθετη περίπτωση θα ονομάζουμε το αντικείμενο μικρό. (ας σημειώσουμε εδώ ότι αυτοί οι ορισμού του μεγάλου και μικρού αντικειμένου είναι διαφορετικοί από αυτούς στο θεώρημα 6.1).

Τώρα θα δείξουμε ότι κάθε αντικείμενο R του Υ είναι μεγάλο. Έστω xi' είναι κάποιος βαθμός από του κορυφαίους α+1 βαθμούς στη λίστα i που δεν είναι ο βαθμός στη λίστα i κάποιου αντικειμένου που βλέπεται από το A . Υπάρχει ένας τέτοιος βαθμός ,από τη στιγμή που όλοι οι βαθμοί στη λίστα i είναι ευδιάκριτοι , και το A βλέπει το πολύ α αντικείμενα. Έστω ότι το D' συμφωνεί με το D σε όλα τα αντικείμενα που βλέπονται από το A , και έστω ότι το αντικείμενο V έχει βαθμό xi' στην ιοστή λίστα του D' , για $1 \leq i \leq m$. Για αυτό το λόγο , ο βαθμός του V στο D' είναι $t(x_1',\ldots,x_m') \geq \tau$. Αφού το V δεν το έχουμε δει , και αφού το V έχει προσδιορισμένους τους βαθμούς σε κάθε λίστα στο D' κάτω από το επίπεδο όπου το A είναι προσβάσιμο από ταξινομημένη προσπέλαση , συνεπάγεται ότι ο αλγόριθμος A πραγματοποιεί ακριβώς το ίδιο , και πιο συγκεκριμένα δίνει την ίδια έξοδο , για τις βάσεις δεδομένων D , D' . Έτσι , ο αλγόριθμος A έχει R ,αλλά όχι V, στην έξοδό του για βάση δεδομένων D' .Λόγω της ορθότητας του A συνεπάγεται ότι το R είναι μεγάλο, όπως είναι το ζητούμενο.

Υποστηρίζουμε ότι κάθε μέλος R του Υ είναι ένα από τα κορυφαία α + 1 μέλη κάποιας λίστας i (και έτσι βλέπεται από τον ΤΑ με βάθος α + 1). Υποθέτουμε εν αντιθέσει ότι το R δεν είναι ένα από τα κορυφαία α+1 μέλη της λίστα i , για $1 \leq i \leq m$. Από τις υποθέσεις ότι η αθροιστική συνάρτηση t είναι αυστηρώς μονότονη και ότι το D ικανοποιεί την ιδιότητα της ευκρίνειας, συνεπάγεται εύκολα ότι το R είναι μικρό. Ήδη έχουμε δείξει ότι κάθε μέλος της Υ είναι μεγάλο. Αυτή η αντίφαση αποδεικνύει τον ισχυρισμό. Συνεπάγεται ότι ο ΤΑ σταματά σε βάθος α + 1 , όπως επιθυμούσαμε.

Στην απόδειξη του θεωρήματος 3.5 , δείξαμε ότι κάτω από τις υποθέσεις του θεωρήματος 3.5 (αυστηρή μονοτονία και ιδιότητα της ευκρίνειας) ο λόγος καταλληλότητας του ΤΑ είναι το πολύ $\frac{1}{2\theta^2}\ cm^2$, όπου c = $\max\{c_R / c_S,\ c_S / c_R\}$.

Οι αποδείξεις των θεωρημάτων 3.1 και 3.5 έχουν αρκετές καλές ιδιότητες :

- Οι αποδείξεις θα ίσχυαν ακόμη και αν ήμασταν και σε ένα σενάριο όπου , οποτεδήποτε μία τυχαία προσπέλαση του αντικειμένου R στη λίστα i λαμβάνει χώρα, θα μαθαίναμε όχι μόνο το βαθμό του R στη λίστα i, αλλά ακόμη και τη σχετική σειρά. Έτσι ο ΤΑ είναι κατάλληλος κατά το στιγμιότυπο ακόμη

και όταν επιτρέπουμε η κλάση **A** να περιλαμβάνει επίσης αλγορίθμους που μαθαίνουν και κάνουν χρήση τέτοιας σχετικής σειράς πληροφορίας.

- Όπως θα δείξουμε , μπορούμε να δείξουμε τη καταλληλότητα στιγμιότυπου μεταξύ αλγόριθμων βελτιστοποίησης μίας βέλτιστης έκδοσης του ΤΑ , κάτω από τις υποθέσει του θεωρήματος 3.1 με μόνο μία μικρή αλλαγή στην απόδειξη. (όπως θα δείξουμε ένα τέτοιο θεώρημα δεν ισχύει κάτω από τις προυποθέσεις του θεωρήματος 3.5).

Αντιμετωπίζοντας τα k, m σαν σταθερές

Στα θεωρήματα 3.1 και 3.5 για την καταλληλότητα στιγμιότυπου του ΤΑ αντιμετωπίζουμε το k (όπου ψάχνουμε να βρούμε τις κορυφαίες k απαντήσεις) και το m (ο αριθμός των ταξινομημένων λιστών) σαν σταθερές. Τώρα θα συζητήσουμε για τις υποθέσεις.

Ξεκινάμε πρώτα με την υπόθεση ότι το k είναι σταθερά. Όπως στις αποδείξεις των θεωρημάτων 3.1 και 3.5 , έστω α είναι ο αριθμός των προσπελάσεων από ένα αλγόριθμο A που ανήκει στη κλάση **A** . Αν $\alpha \geq k$,τότε δεν υπάρχει ανάγκη να αντιμετωπίσουμε το k σαν σταθερά. Έτσι, αν έπρεπε να περιορίσουμε τη κλάση **A** των αλγορίθμων να περιέχουν μόνο αλγορίθμους οι οποίοι κάνουν τουλάχιστον k προσπελάσεις για να βρουν τις κορυφαίες k απαντήσεις, τότε δεν θα υπάρχει η ανάγκη να υποθέσουμε ότι το k είναι σταθερά. Πως μπορεί να γίνει έτσι ώστε ένα αλγόριθμος A να μπορεί να βρει τις κορυφαίες k απαντήσεις χωρίς να κάνει τουλάχιστον k προσπελάσεις, και πιο συγκεκριμένα χωρίς να προσπελαστούν τουλάχιστον k αντικείμενα; Θα πρέπει τότε να γίνει ότι ή θα υπάρχουν το πολύ k αντικείμενα στην βάση δεδομένων , ή αλλιώς κάθε αντικείμενο R που ο A δεν έχει δει έχει γενικό βαθμό t(R). Ακόμη και κάτω από αυτές τις περιστάσεις , είναι ακόμη αδικαιολόγητο σε κάποια περιβάλλοντα (όπως είναι σε περιβάλλον βάσης δεδομένων) να επιτρέπεται σε ένα αλγόριθμο A να δείχνει στην έξοδό του ένα αντικείμενο σαν μέλος των κορυφαίων k αντικειμένων χωρίς καν να το έχουμε δει: πως ο αλγόριθμος θα ήξερε το όνομα του αντικειμένου; Αυτό είναι παρόμοιο με ένα θέμα που έχουμε θίξει προηγουμένως σχετικά με τις ισχυρές μαντεψιές.

Τι γίνεται με την περίπτωση που το m είναι σταθερά ; Όπως σημειώσαμε προηγουμένως , αυτή είναι μία δικαιολογημένη περίπτωση , αφού το m είναι ένας αριθμός ορισμάτων της αθροιστικής συνάρτησης, τις οποίες τις θεωρούμε δεδομένες.

Σε αυτή τη περίπτωση του θεωρήματος 3.1 (όχι ισχυρές μαντεψιές) το πόρισμα 3.2 λέει ότι τουλάχιστον για αυστηρές αθροιστικές συναρτήσεις, αυτή η εξάρτηση από το λόγο καταλληλότητας στο m είναι αναπόφευκτη.

Παρομοίως, στη περίπτωση των υποθέσεων του θεωρήματος 3.5 (ισχυρή

μονοτονία και ιδιότητα ευκρίνειας) , το θεώρημα 3.2 μας λέει ότι τουλάχιστον για κάποιες αθροιστικές συναρτήσεις , αυτή η εξάρτηση του λόγου καταλληλότητας στο m είναι αναπόφευκτη.

Μετατρέποντας τον ΤΑ σε προσεγγιστικό αλγόριθμο και επιτρέποντας νωρίτερη παύση

Ο ΤΑ μπορεί εύκολα να μετατραπεί σε ένα προσεγγιστικό αλγόριθμο. Θα μπορεί τότε να χρησιμοποιηθεί σε περιπτώσεις όπου νοιαζόμαστε μόνο για τις κορυφαίες k απαντήσεις. Έτσι, έστω ότι δίνεται $\theta > 1$. Ορίζουμε ως θ – προσέγγιση στις κορυφαίες k απαντήσεις (για t πάνω από τις βάσεις δεδομένων D) να είναι μία συλλογή από k αντικείμενα (και οι βαθμοί τους) τέτοιοι ώστε για κάθε y μεταξύ αυτών των k αντικειμένων και κάθε z όχι ανάμεσα αυτών των k αντικειμένων, ισχύει ότι $\theta t(y) \geq t(z)$. Μπορούμε να τροποποιήσουμε τον ΤΑ προκειμένου να βρούμε μία θ-προσέγγιση στις κορυφαίες k απαντήσεις αλλάζοντας το κανόνας παύσης στο δεύτερο βήμα ώστε να είναι «Τόσο σύντομα όσο τουλάχιστον k αντικείμενα να έχουν ειδωθεί των οποίων ο βαθμός είναι τουλάχιστον ίσος με τ / θ ,τότε κάνε παύση.» Ας ονομάζουμε αυτή τη προσέγγιση αλγόριθμο TA_θ .

ΘΕΩΡΗΜΑ 3.6 .Ας υποθέσουμε $\theta > 1$ και ότι η αθροιστική συνάρτηση t είναι μονότονη. Τότε ο TA_θ ορθά βρίσκει μία θ-προσέγγιση στις κορυφαίες k απαντήσεις της t.

Απόδειξη: Αυτό προκύπτει από μία άμεση τροποποίηση της απόδειξης του θεωρήματος 2.1.

Το επόμενο θεώρημα λέει ότι όταν εφιστούμε τη προσοχή μας σε αλγορίθμους που δεν κάνουν ισχυρές μαντεψιές ,τότε ο αλγόριθμος TA_θ είναι κατάλληλος κατά το στιγμιότυπο.

ΘΕΩΡΗΜΑ 3.7.Υποθέτουμε ότι $\theta > 1$ και ότι η αθροιστική συνάρτηση t είναι μονότονη. Έστω **D** είναι η κλάση όλων των βάσεων δεδομένων. Έστω **A** είναι η κλάση όλων των αλγορίθμων που βρίσκουν μία θ-προσέγγιση στις κορυφαίες κ απαντήσεις της t για κάθε βάση δεδομένων και ότι που δεν κάνουν ισχυρές μαντεψιές. Τότε ο αλγόριθμος TA_θ είναι κατάλληλος κατά το στιγμιότυπο πάνω από τις κλάσεις **A** , **D**.

Απόδειξη: Από την απόδειξη του θεωρήματος 3.1 αλλάζουμε τον ορισμό του αντικειμένου R που είναι «μεγάλο» να ισχύει $\theta t(R) \geq \tau A$.

Το θεώρημα 3.7 δείχνει ότι το ανάλογο του θεωρήματος 1 ισχύει για TA_θ .Το επόμενο παράδειγμα , που είναι μία τροποποίηση του παραδείγματος 3.3 ,δείχνει ότι το ανάλογο του θεωρήματος 3.5 δεν ισχύει για TA_θ .Μία εξήγηση για αυτά τα αποτελέσματα είναι ότι το θεώρημα 3.1 είναι επαρκώς ισχυρό και έτσι μπορεί να προσαρμόζεται στην ύπαρξη προσεγγίσεων ενώ το θεώρημα 3.5 όχι.

Παράδειγμα 3.8: Ας υποθέσουμε ότι $\theta > 1$, και ότι υπάρχουν $2n + 1$ αντικείμενα ,τα οποία θα καλούμε απλά σαν $1,2,\ldots,2n + 1$, και ότι υπάρχουν δύο λίστες L1 και L2 δες εικόνα 2 .Ας επίσης υποθέσουμε ότι στη λίστα L1, οι βαθμοί ανατίθεται έτσι ώστε όλοι οι βαθμοί να είναι διαφορετικοί , η σειρά των αντικειμένων κατά βαθμό είναι $1, 2, \ldots 2n + 1$, το αντικείμενο $n + 1$,έχει βαθμό $1/\theta$,και το αντικείμενο $n + 2$ έχει βαθμό $\frac{1}{2\theta^2}$.Υποθέτουμε ότι στη λίστα L2 , οι βαθμοί ανατίθενται έτσι ώστε όλοι να είναι διαφορετικοί , η σειρά των αντικειμένων τότε είναι $2n + 1, 2n, \ldots, 1$ (το αντίστροφο από τη σειρά στη λίστα L1), το αντικείμενο $n + 1$ έχει βαθμό $1 / \theta$ και το αντικείμενο n έχει βαθμό $\frac{1}{2\theta^2}$. Αν η αθροιστική συνάρτηση είναι η min ,και ότι $k = 1$ (έτσι ώστε να ενδιαφερόμαστε για την εύρεση μίας θ- προσέγγισης για τη κορυφαία απάντηση). Ο συνολικός βαθμός κάθε αντικειμένου εκτός του $n + 1$ είναι το πολύ $\alpha = \frac{1}{2\theta^2}$. Αφού $\theta\alpha = \frac{1}{2\theta}$, το οποίο είναι λιγότερο από το βαθμό $1/\theta$ του αντικειμένου $n +1$, συνεπάγεται ότι το μοναδικό αντικείμενο το οποίο μπορεί να επιστραφεί από έναν αλγόριθμο σαν το TA_θ που ορθά βρίσκει μία θ-προσέγγιση στη κορυφαία απάντηση είναι το αντικείμενο $n + 1$.
Ένας αλγόριθμος που κάνει μία ισχυρή μαντεψιά και ζητά το βαθμό του αντικειμένου $n + 1$ και στις δύο λίστες θα καθορίζει την ορθή απάντηση και θα είναι ικανός να σταματά μετά από δύο τυχαίες προσπελάσεις και όχι ταξινομημένες προσπελάσεις. Ο αλγόριθμος μπορεί να κάνει ασφαλή παύση ,αφού γνωρίζει ότι βρέθηκε ένα αντικείμενο R τέτοιο ώστε να ισχύει $\theta t(R) = 1$, και έτσι το $\theta t(R)$ να είναι τόσο μεγάλο όσο κάθε πιθανός βαθμός. Πάντως, υπό ταξινομημένη προσπέλαση για τη λίστα L1, ο αλγόριθμος TA_θ θα έβλεπε τα αντικείμενα στη σειρά $1,2\ldots..2n + 1$, ενώ κάτω από ταξινομημένη προσπέλαση για τη λίστα L2, ο αλγόριθμος θα έβλεπε τα αντικείμενα σε αντίστροφη σειρά. Αφού το αντικείμενο που «νικάει» είναι στη μέση και των δύο ταξινομημένων λιστών , συνεπάγεται ότι τουλάχιστον $n + 1$ ταξινομημένες προσπελάσεις θα απαιτούνται πριν ο TA_θ δει καν το αντικείμενο που «νικάει».

ΘΕΩΡΗΜΑ 3.9 Ας υποθέσουμε ότι $\theta > 1$. Έστω **D** είναι η κλάση όλων των βάσεων δεδομένων που ικανοποιούν την ιδιότητα της ευκρίνειας. Έστω **A** είναι η κλάση των αλγορίθμων που βρίσκουν μία θ-προσέγγιση στη κορυφαία

απάντηση για τη min για κάθε βάση δεδομένων στη κλάση **D**. Δεν υπάρχει ντετερμινιστικός αλγόριθμος (ή πιθανοκρατικός αλγόριθμος που δεν κάνει ποτέ λάθος) ο οποίος είναι κατάλληλος κατά το στιγμιότυπο πάνω από τα **A** ,**D**.

L1	L2
(1,.)	(2n + 1,.)
(2,.)	(2n ,.)
….	…
(n + 1, $\frac{1}{\theta}$)	(n + 1, $\frac{1}{\theta}$)
(n + 2, $\frac{1}{2\theta^2}$)	(n , $\frac{1}{2\theta^2}$)
(n + 3,.)	(n – 1, .)
..	…..
(2n + 1,.)	(1,.)

ΕΙΚΟΝΑ 2 (η τελεία αντιπροσωπεύει μία οποιαδήποτε τιμή)

Ελαχιστοποιώντας τη τυχαία προσπέλαση

Μέχρι τώρα δεν επικεντρωθήκαμε στον αριθμό των τυχαίων προσπελάσεων. Για κάθε ταξινομημένη προσπέλαση στο ΤΑ ,πάνω από m-1 τυχαίες προσπελάσεις παίρνουν μέρος. Αν αποκαλούμε s τον αριθμό των διαταγμένων προσπελάσεων και r τον αριθμό των τυχαίων προσπελάσεων ,τότε το μέσο κόστος είναι $sc_S + rc_R$, για κάποιες θετικές σταθερές c_S , c_R . Η θεωρησή μας για τη καταλληλότητα αντικειμένου αγνοεί σταθερούς παράγοντες όπως ο m και ο c_R (είναι απλοί πολλαπλασιαστικοί παράγοντες στο λόγο καταλληλότητας). Έτσι, δεν υπάρχει κίνητρο ως εδώ για να απασχολούμαστε με τον αριθμό των τυχαίων προσπελάσεων.

Υπάρχουν, ωστόσο ,κάποια σενάρια ,όπου πρέπει να επιστήσουμε τη προσοχή μας στον αριθμό των τυχαίων προσπελάσεων. Το πρώτο σενάριο είναι όταν οι τυχαίες προσπελάσεις είναι αδύνατες (το οποίο σημαίνει ότι $c_R = \infty$) . Ένα παράδειγμα για αυτή τη περίπτωση είναι όταν το μέσο κόστος είναι ένα σύστημα ανάκτησης κειμένου, και οι ταξινομημένες λίστες είναι το αποτέλεσμα των μηχανών αναζήτησης. Ένα άλλο σενάριο είναι όταν οι τυχαίες προσπελάσεις είναι δυνατές, αλλά λιγότερο ακριβές από τις ταξινομημένες προσπελάσεις. Ένα παράδειγμα για αυτή τη περίπτωση είναι

όταν το κόστος ανταποκρίνεται στη προσπέλαση δίσκου(σειριακά αντί για τυχαία).Τότε θα θέλαμε ο λόγος καταλληλότητας να είναι ανεξάρτητος από το c_R/c_S. Αυτό γιατί, αν επιτρέπαμε τα c_R, c_S να ποικίλουν, αντί να τα θεωρούμε σταθερές , θα θέλαμε ο λόγος καταλληλότητας να φράσσεται.

Παρακάτω θα περιγράψουμε δύο αλγόριθμους. Ο ένας δεν κάνει χρήση τυχαίων προσπελάσεων, και έτσι αποκαλείται NRA ("No Random Access") .Και ο δεύτερος λαμβάνει υπόψιν το κόστος των τυχαίων προσπελάσεων. Είναι ένας συνδυασμός του NRA και του TA, και έτσι τον αποκαλούμε CA("Combined Algorithm").

Και οι δύο αλγόριθμοι προσπελαύνουν τη πληροφορία με φυσικό τρόπο , και με το πνεύμα των βασιζόμενων στη γνώση προγραμμάτων σταματούν όταν γνωρίζουν ότι καμία βελτίωση δεν παίρνει μέρος. Γενικά, σε κάθε σημείο στην εκτέλεση των δύο αλγορίθμων όπου ένα πλήθος από ταξινομημένες και τυχαίες προσπελάσεις έχουν λάβει μέρος, για κάθε αντικείμενο R υπάρχει ένα υποσύνολο $S(R)= \{i_1, i_2 \ldots i_l\} \subseteq \{1,\ldots,m\}$ των πεδίων του R όπου ο αλγόριθμος έχει καθορίσει τις τιμές $x_{i_1}, x_{i_2} \ldots\ldots x_{i_l}$ αυτών των πεδίων. Δοθέντος αυτής της πληροφορίας, καθορίζουμε λειτουργίες αυτής της πληροφορίας που είναι κάτω και άνω φράγματα στη τιμή t(R)που μπορούν να επιτευχθούν. Ο αλγόριθμος προχωρά μέχρι να μην υπάρχουν αντικείμενα που το άνω φράγμα τους να είναι καλύτερο από το συγκεκριμένο κ μεγαλύτερο κάτω φράγμα.

Κάτω φράγμα: Δοθέντος ενός αντικειμένου R και ενός υποσυνόλου $S(R)= \{i_1, i_2 \ldots i_l\} \subseteq \{1,\ldots,m\}$ γνωστών πεδίων του R ,με τιμές $x_{i_1}, x_{i_2} \ldots\ldots x_{i_l}$ για αυτά τα γνωστά πεδία , ορίζουμε Ws(R) (ή W(R) αν το υποσύνολο S=S(R)) σαν την ελάχιστη (ή χειρότερη) τιμή που η αθροιστική συνάρτηση μπορεί να φτάσει για ένα αντικείμενο R. Όταν η t είναι μονότονη, αυτή η ελάχιστη τιμή επιτυγχάνεται αντικαθιστώντας για κάθε πεδίο που λείπει i($\{1,\ldots m\}$ to 0 και αντιστοιχώντας το t στο αποτέλεσμα. Για παράδειγμα, αν το $S=\{1,\ldots l\}$ τότε το $Ws(R)=t(x_1, x_2 \ldots., x_l, 0, \ldots, 0\}$.

Πρόταση 4.1.
Αν S είναι το σύνολο των γνωστών πεδίων του αντικειμένου R , τότε $t(R) \geq Ws(R)$.

Με άλλα λόγια, το W(R) αναπαριστά ένα κάτω φράγμα στην t(R). Είναι αυτό το καλύτερο δυνατό? Ναι, εκτός και αν έχουμε επιπρόσθετες πληροφορίες, όπως ότι η τιμή 0 δεν εμφανίζεται στις λίστες. Γενικά, όσο ένας αλγόριθμος προχωρά και μαθαίνουμε περισσότερα πεδία ενός αντικειμένου R, η τιμή του W γίνεται μεγαλύτερη (ή τουλάχιστον όχι μικρότερη). Για κάποιες αθροιστικές συναρτήσεις t η τιμή W(R) δεν αποδίδει τιμή μέχρι το S να περιέχει όλα τα πεδία: π.χ όταν το t είναι το μέσο των τριών πεδίων, τότε όσο πιο νωρίς γνωρίζουμε δύο W(R) από αυτά είναι τουλάχιστον το μικρότερο από τα δύο.

Άνω φράγμα : Η καλύτερη τιμή που ένα αντικείμενο μπορεί να επιτύχει εξαρτάται από άλλες πληροφορίες που έχουμε. Θα χρησιμοποιήσουμε μόνο τις τιμές σε κάθε πεδίο, όπως ορίζονται στο ΤΑ : Δοθέντος ενός αντικειμένου R και ενός υποσυνόλου
$s(R) = \{i_1, i_2 \ldots i_l\} \subseteq \{1,\ldots,m\}$ γνωστών πεδίων του R ,με τιμές $x_{i_1}, x_{i_2} \ldots\ldots x_{i_l}$ για αυτά τα γνωστά πεδία , ορίζουμε Bs(R) (ή B(R) αν το υποσύνολο S είναι κατανοητό από το context) σαν τη μέγιστη (καλύτερη) τιμή την οποία η αθροιστική συνάρτηση t μπορεί να επιτύχει για αντικείμενο R. Όταν η t είναι μονότονη ,αυτή η μέγιστη τιμή επιτυγχάνεται αντικαθιστώντας για κάθε πεδίο που λείπει $i \in \{1,\ldots m\} \setminus S$ η τιμή x_i και εφαρμόζοντας το t στο αποτέλεσμα. Για παράδειγμα, αν το $S = \{1,\ldots l\}$,τότε το Bs(R) = $t(x_1, x_2 \ldots., x_l, \underline{x}_{l+1}, \ldots, \underline{x}_m)$. Η ακόλουθη πρόταση προκύπτει από τον ορισμό.

Πρόταση 4.2
Αν S είναι το σύνολο των γνωστών πεδίων του αντικειμένου R , τότε ισχύει $t(R) \leq Bs(R)$.

Με άλλα λόγια, το B(R) αναπαριστά ένα άνω φράγμα στη τιμή t(R)(ή την καλύτερη τιμή του t(R) που μπορεί να επιτευχθεί) δοθέντος τη πληροφορία που έχουμε μέχρι τώρα. Είναι αυτό το καλύτερο άνω φράγμα που μπορούμε να επιτύχουμε? Αν οι λίστες μπορούν να περιέχουν ίσες τιμές (τις οποίες γενικά υποθέτουμε ότι περιέχουν) , τότε δοθέντος της πληροφορίας που έχουμε είναι δυνατό ότι t(R)= B(R). Αν η ιδιότητα της ευκρίνειας ισχύει (ισότητες δεν επιτρέπονται σε μία λίστα),τότε για συνεχόμενες αθροιστικές συναρτήσεις t είναι η περίπτωση όπου B(R) είναι το καλύτερο πάνω φράγμα για τη τιμή t που μπορούμε να έχουμε στο R. Γενικά ,όσο ένας αλγόριθμος προχωρά και μαθαίνουμε περισσότερα πεδία ενός αντικειμένου R και οι τιμές βάσης xi, μειώνονται, τότε το B(R) μπορεί μόνο να μειώνεται (ή να παραμένει το ίδιο).
Μία σημαντική ειδική περίπτωση είναι όταν ένα αντικείμενο R το οποίο δεν έχει καταμετρηθεί καθόλου. Σε αυτή τη περίπτωση $B(R)= t\{x_1, x_2 \ldots., x_m\}$.Ας σημειώσουμε ότι είναι το ίδιο με τη τιμή σκανδάλης στον ΤΑ.

NRA. (No Random Access Algorithm)

Όπως έχουμε πει και προηγουμένως , υπάρχουν περιπτώσεις που οι τυχαίες προσπελάσεις είναι αδύνατες, σε αυτή τη περίπτωση αλλάζουμε το κριτήριο μας για την επιθυμητή έξοδο. Προηγουμένως ,απαιτούσαμε στην έξοδο να είναι οι «κορυφαίες κ απαντήσεις» ,το οποίο είναι τα κορυφαία κ αντικείμενα, μαζί με τους συνολικούς βαθμούς τους. Ο λόγος είναι ότι αφού οι τυχαίες προσπελάσεις είναι αδύνατες , μπορεί να είναι πολύ φθηνότερο (το οποίο είναι, αφού απαιτεί λιγότερες προσπελάσεις) να βρούμε τα κορυφαία κ αντικείμενα χωρίς τους βαθμούς τους. Αυτό γιατί, όπως θα δείξουμε με

παράδειγμα, μπορούμε μερικές φορές να πετύχουμε αρκετή μερική πληροφορία για τους βαθμούς έτσι ώστε να γνωρίζουμε αν ένα αντικείμενο είναι στα κορυφαία κ χωρίς να ξέρουμε τον ακριβή βαθμό.

Παράδειγμα: Ας θεωρήσουμε το ακόλουθο σενάριο, όπου η αθροιστική συνάρτηση είναι ο μέσος όρος, και όπου k = 1(έτσι ώστε να ενδιαφερόμαστε μόνο για τα κορυφαία αντικείμενα). Υπάρχουν μόνο δύο διατεταγμένες λίστες L1, L2 και ο βαθμός κάθε αντικειμένου μαζί και στις δύο είναι 1/3 ,εκτός από το ότι το αντικείμενο R έχει βαθμό 1 στην L1, και βαθμό 0 στην L0. Μετά από δύο ταξινομημένες προσπελάσεις στην L1 και μία ταξινομημένη προσπέλαση στην L2, υπάρχει αρκετή πληροφορία για να γνωρίζουμε αν το αντικείμενο R είναι στα κορυφαία αντικείμενα(ο μέσος όρος του βαθμού είναι τουλάχιστον 1/2, και κάθε άλλο αντικείμενο έχει μέσο όρο το πολύ 1/3). Αν επιθυμούμε να βρούμε το βαθμό ενός αντικειμένου R, θα χρειαστούμε να κάνουμε ταξινομημένη προσπέλαση σε όλη την L2.

L1	L2
(R,1) (.,1/3) .... (.,1/3)	(.,1/3) ... (.,1/3) (R,0)

Εικόνα 4. Βάση δεδομένων για το προηγούμενο παράδειγμα

Ας σημειώσουμε ότι χρειαζόμαστε μόνο το ότι η έξοδος αποτελείται από τα κορυφαία κ αντικείμενα, χωρίς να δίνεται πληροφορία για τη σειρά ταξινόμησης (ταξινομημένα κατά βαθμούς). Αν επιθυμούμε να γνωρίζουμε τη σειρά ταξινόμησης , αυτό μπορεί εύκολα να καθοριστεί βρίσκοντας τα κορυφαία αντικείμενα, τα κορυφαία 2 αντικείμενα ,κ.τ.λ... Έστω Ci, είναι το κόστος της εύρεσης των κορυφαίων κ αντικειμένων. Είναι ενδιαφέρον να σημειώσουμε ότι δεν υπάρχει υποχρεωτική σχέση μεταξύ των Ci και Cj, για i < j . Για παράδειγμα , στο προηγούμενο παράδειγμα είχαμε C1 < C2 , αν έπρεπε να τροποποιήσουμε το παράδειγμα έτσι ώστε να υπάρχουν δύο αντικείμενα R , R' με βαθμό 1 στην L1 , όπου ο βαθμός της R στην L2 είναι 0 , και ο βαθμός του R' στην L2 είναι 1/4.(και έτσι ,όπως πριν, όλοι οι παραμένοντες βαθμοί και στις δύο λίστες είναι 1/3), τότε ισχύει C2 < C1.

Το κόστος της εύρεσης των κορυφαίων κ αντικειμένων στη ταξινομημένη προσπέλαση είναι το πολύ k maxi Ci. Αφού θεωρούμε το k ως σταθερά, συνεπάγεται εύκολα ότι μπορούμε να μετατρέψουμε τον κατάλληλο στιγμιότυπου αλγόριθμο για την εύρεση των κορυφαίων κ αντικείμενων σε

ταξινομημένη σειρά. Πρακτικά, συνήθως είναι αρκετά καλό να γνωρίζουμε τα κορυφαία κ αντικείμενα σε ταξινομημένη σειρά, χωρίς να γνωρίζουμε τους βαθμούς. Στη πραγματικότητα, η μεγαλύτερη μηχανή αναζήτησης στο δίκτυο δεν δίνει πλέον βαθμούς.(πιθανό για να αποφύγει μηχανισμούς ακύρωσης).

Ο αλγόριθμος NRA έχει ως ακολούθως:

Κάνε ταξινομημένη προσπέλαση παράλληλα σε κάθε μία από τις m ταξινομημένες λίστες Li. Σε κάθε μήκος d (όταν τα d αντικείμενα έχουν προσπελαστεί υπό ταξινομημένη προσπέλαση σε κάθε λίστα)

1

- Διατήρησε τις βασικές τιμές $\underline{x}_1^{(d)}$, $\underline{x}_2^{(d)}$,....$\underline{x}_m^{(d)}$ που καταμετρούνται στις λίστες.
- Για κάθε αντικείμενο R με πεδία που έχουν ανακαλυφτεί $S = S^{(d)}(R) \subseteq \{1,...m\}$, υπολόγισε τις τιμές Wd (R)= Ws(R) και Bd (R)= Bs(R).(για αντικείμενα R που δεν έχουν ειδωθεί, αυτές οι τιμές έχουν εικονικά υπολογιστεί ως Wd(R) = t(0,....,0) και Bd(R) =t($x_1, x_2 \ldots, x_m$), η οποία είναι η τιμή σκανδάλης).
- Έστω Tk είναι η συγκεκριμένη λίστα των κορυφαίων κ, η οποία περιέχει κ αντικείμενα με το μεγαλύτερο Wd τα οποία έχουν ειδωθεί μέχρι τώρα(και οι βαθμοί τους) , αν δύο αντικείμενα έχουν την ίδια τιμή $W^{(d)}$, τότε το αντικείμενο με το υψηλότερο $B^{(d)}$ κερδίζει. Έστω το $Mk^{(d)}$ είναι η κ μεγαλύτερη $W^{(d)}$ τιμή στο $Tk^{(d)}$

2

Καλούμε ένα αντικείμενο R βιώσιμο αν $B^{(d)}(R) > Mk^{(d)}$. Σταμάτα όταν (α) τουλάχιστον κ ευδιάκριτα αντικείμενα έχουν ειδωθεί (έτσι ώστε σε κάθε $Tk^{(d)}$ να περιέχονται k αντικείμενα) και β) δεν υπάρχουν βιώσιμα αντικείμενα έξω από το $Tk^{(d)}$, το οποίο ισχύει όταν , $B^{(d)}(R) \leq Mk^{(d)}$ για κάθε $R \notin Tk^{(d)}$. Επέστρεψε τα αντικείμενα στο $Tk^{(d)}$.

ΘΕΩΡΗΜΑ 4.4

Αν η αθροιστική συνάρτηση t είναι μονότονη τότε ο αλγόριθμος NRA βρίσκει ορθώς τα κορυφαία κ αντικείμενα.

Απόδειξη

Ας υποθέσουμε ότι ο NRA διακόπτει μετά από d ταξινομημένες προσπελάσεις σε κάθε λίστα, και ότι $Tk = \{R_1, R_2 \ldots..R_k\}$. Έτσι, η έξοδος των αντικειμένων από

τον NRA είναι $R_1, R_2 \ldots\ldots R_k$. Έστω R είναι ένα αντικείμενο όχι ανάμεσα στα $R_1, R_2 \ldots\ldots R_k$. Πρέπει να δείξουμε ότι t(R)≤ t(Ri) για κάθε i.

Αφού ο αλγόριθμος σταματά σε βάθος d ,γνωρίζουμε ότι το R είναι μη βιώσιμο σε βάθος d ,το οποίο είναι, $B^{(d)}(R) \leq M_k^{(d)}$.Τώρα ισχύει $t(R) \leq B(R)$ για κάθε i.Επίσης για κάθε ένα από τα k αντικείμενα Ri , έχουμε $Mk \leq Wd(Ri) \leq t(Ri)$ (από προηγούμενη πρόταση και τον ορισμό του Mk).Συνδυάζοντας τις ανισότητες έχουμε :

$t(R) \leq B^{(d)}(R) \leq Mk^{(d)} \leq W^{(d)}(Ri) \leq t(Ri)$ για κάθε i , όπως επιθυμούμε.

Ας σημειώσουμε επίσης ότι μηχανισμός σπασίματος δεν χρειαζόταν για την ορθότητα (αλλά θα χρειαστεί για τη καταλληλότητα στιγμιότυπου), Τώρα θα δείξουμε τη καταλληλότητα στιγμιοτύπου του NRA πάνω από όλους τους αλγορίθμους που δεν χρησιμοποιούν τυχαία προσπέλαση:

ΘΕΩΡΗΜΑ 4.5. Υποθέτουμε ότι η αθροιστική συνάρτηση t είναι μονότονη. Έστω **D** είναι η κλάση όλων των βάσεων δεδομένων. Έστω **A** είναι η κλάση όλων των αλγορίθμων οι οποίοι ορθώς βρίσκουν τα κορυφαία κ αντικείμενα για το t για κάθε βάση δεδομένων και που δεν κάνουν τυχαίες προσπελάσεις. Τότε ο NRA είναι κατάλληλος κατά το στιγμιότυπο πάνω από τα **A ,D**.

Απόδειξη: Ας υποθέσουμε ότι A ε **A.** Αν ο αλγόριθμος NRA σταματά σε βάθος d ,και ο NRA δείχνει τουλάχιστον k ευδιάκριτα αντικείμενα για πρώτη φορά με μήκος d , τότε ο NRA κάνει μόνο ένα σταθερό αριθμό από προσπελάσεις (το πολύ km^2) σε αυτή τη βάση δεδομένων. Έτσι αν υποθέσουμε ότι σε κάποιες βάσεις δεδομένων D,ο αλγόριθμος NRA σταματά σε βάθος d ,και ότι ο NRA δείχνει τουλάχιστον k ευδιάκριτα αντικείμενα με μήκος d-1. Ισχυριζόμαστε ότι ο A πρέπει να φτάσει μήκος d σε τουλάχιστον μία από τις λίστες. Τότε συνεπάγεται ότι ο λόγος καταλληλότητας του NRA είναι το πολύ m. Ας υποθέσουμε ότι ο ισχυρισμός αποτυγχάνει. Τότε από το γεγονός ότι ο αλγόριθμος NRA δεν σταματά σε μήκος d-1 υπάρχει ένα αντικείμενο $R \notin Tk^{(d-1)}$ τέτοιο ώστε $B^{(d-1)}(R) > M_k^{(d-1)}$.Γνωρίζουμε ότι $W^{(d-1)}(R) \leq M_k^{(d-1)}$, αφού $R \neq T_k^{(d-1)}$. Επιπλέον ,γνωρίζουμε από το μηχανισμό σπασίματος ότι αν $W(d-1)(R) = M_k^{(d-1)}$, τότε για κάθε Ri ε $T_k^{(d-1)}$ τέτοιο ώστε $W^{(d)}(R_i) = M_k^{(d)}$ αναγκαία ισχύει $B^{(d-1)}(R_i) \geq B^{(d-1)}(R)$. Υπάρχουν τώρα δύο περιπτώσεις, εξαρτάται από το πώς ή όχι ο αλγόριθμος A βγάζει έξοδο το R σαν ένα από τα κορυφαία k αντικείμενα. Σε άλλη περίπτωση ,κατασκευάζουμε μία βάση δεδομένων στην οποία το A σφάλλει.

Περίπτωση 1: Ο αλγόριθμος Α βγάζει έξοδο R σαν ένα από τα κορυφαία αντικείμενα. Κατασκευάζουμε μία βάση δεδομένων D' όπου το Α σφάλλει. Η βάση δεδομένων D' είναι ιδανική για το D για μήκος πάνω από d-1 (αυτό γιατί ,για κάθε i τα κορυφαία d-1 αντικείμενα και οι βαθμοί τους είναι τα ίδια στη λίστα L_i για D' και D). Για κάθε R_i και για κάθε πεδίο που λείπει j ανήκει στο $\{1,\ldots,m\} \setminus S^{(d-1)}(R_i)$ η τιμή $\underline{x}_j^{(d-1)}$.Για το αντικείμενο R αναθέτουμε όλα τα πεδίου που λείπουν στο $\{1,\ldots,m\} \setminus S^{(d-1)}(R)$ τη τιμή 0. Τώρα θα δείξουμε ότι $t(R) < t(R_j)$ για κάθε j με $1 \leq j \leq k$. Έτσι, το R δεν είναι ένα από τα κορυφαία k αντικείμενα, και έτσι ο αλγόριθμος Α έσφαλλε. Πρώτα έχουμε

$$t(R) = W^{(d-1)}(R) \leq M_k^{(d-1)}.(3)$$

Επίσης , για όλα τα i με $1 \leq i \leq k$ έχουμε

$$M_k^{(d-1)} \leq W^{(d-1)}(R_i) \leq B^{(d-1)}(R_i) = t(R_i). \quad (4)$$

Αν $W^{(d-1)}(R) < M_k^{(d-1)}$, τότε έχουμε από τη (3) και (4) ότι $t(R) < t(R_i)$ για κάθε i, όπως επιθυμούμε. Έτσι υποθέτουμε ότι $W^{(d-1)}(R) = M_k^{(d-1)}$. Πάλι ,επιθυμούμε να δείξουμε ότι t(R) < t(Ri) για κάθε i. Θεωρούμε ξεχωριστά σε δύο υποπεριπτώσεις αυτά τα i για τα οποία ισχύει $M_k^{(d-1)} = W^{(d-1)}(R_i)$ και αυτά για τα οποία ισχύει $M_k^{(d-1)} \neq W^{(d-1)}(R_i)$.

Υποπερίπτωση 1: $M_k^{(d-1)} = W^{(d-1)}(R_i)$. Τότε ισχύει $t(R) \leq M_k^{(d-1)} < B^{(d-1)}(R_i) = t(R_i)$, όπως επιθυμούμε, όπου η τελευταία ανισότητα συνεπάγεται από το μηχανισμό σπασίματος.

Υποπερίπτωση 2: $M_k^{(d-1)} \neq W^{(d-1)}(R_i)$ και έτσι $M_k^{(d-1)} < W^{(d-1)}(R_i)$. Από τις ανισότητες στη (4), βλέπουμε ότι $M_k^{(d-1)} < t(R_i)$. Έτσι από τη (3) έχουμε $t(R) < t(R_i)$,όπως επιθυμούμε.

Περίπτωση 2: Ο αλγόριθμος Α δεν βγάζει έξοδο R σαν ένα από τα κορυφαία κ αντικείμενα. Κατασκευάζουμε μία βάση δεδομένων D'' όπου Α σφάλλει ως ακολούθως. Η βάση δεδομένων D'' είναι ιδανική για D σε μήκος πάνω από d-1

Σε μήκος d δίνει σε κάθε υπολειπόμενο πεδίο ιε{1,...m} S(d-1)(R) του R τη τιμή xi(d-1). Για όλα τα υπολειπόμενα πεδία που λείπουν ,συμπεριλαμβανομένων των πεδίων R1,...Rk , αναθέτει τη τιμή 0. Τώρα το $t(R) = B^{(d-1)}(R) > M_k^{(d-1)}$, όπου α) Για τουλάχιστον ένα Ri(ονομαστικά, το Ri για το οποίο ισχύει $W^{(d)}(R_i) = M_k^{(d)}$ έχουμε $t(R_i) = m_k^{(d-1)}$, και b) για κάθε αντικείμενο R' όχι ανάμεσα $R_1, R_2 \ldots R_k$ ή R που έχουμε έτσι ώστε $t(R') \leq M_k^{(d-1)}$. Έτσι, ο αλγόριθμος Α έσφαλλε στο να μη βγάλει το R σαν ένα από τα κορυφαία κ αντικείμενα.

Ας σημειώσουμε ότι το θέμα της «ισχυρής μαντεψιάς» δεν είναι σχετικό με την υπάρχουσα μελέτη καθώς εφιστούμε τη προσοχή μας σε αλγόριθμους που δεν κάνουν τυχαίες προσπελάσεις(και έτσι δεν γίνονται μαντεψιές).

Το επόμενο, είναι ένα πόρισμα της απόδειξης του θεωρήματος 4.4.Ειδικότερα, στην απόδειξη του θεωρήματος 4.4, δείξαμε ότι ο λόγος καταλληλότητας του NRA είναι το πολύ m. Το επόμενο αποτέλεσμα λέει ότι αν η αθροιστική συνάρτηση είναι αυστηρή, τότε ο λόγος καταλληλότητας είναι ακριβώς m , και αυτό είναι το καλύτερο δυνατό.

Πόρισμα 4.6.

Έστω t είναι μία μονότονη, αυστηρή ,αθροιστική συνάρτηση με m ορίσματα. Έστω **D** είναι μία κλάση όλων των βάσεων δεδομένων. Έστω **A** είναι η κλάση όλων των αλγορίθμων που ορθά βρίσκουν τα κορυφαία κ αντικείμενα για t για κάθε βάση δεδομένων και που δεν κάνουν τυχαίες προσπελάσεις. Τότε ο NRA είναι κατάλληλος κατά το στιγμιότυπο πάνω από τα **A** ,**D**, με λόγο καταλληλότητας m. Κανένας ντετερμινιστικός αλγόριθμος δεν έχει χαμηλότερο λόγο καταλληλότητας.

Σημείωση

Δυστυχώς, η εκτέλεση του NRA μπορεί να απαιτεί πολλές αποθηκεύσεις σε κάθε βήμα, αφού όταν ο NRA εκτελεί ταξινομημένη προσπέλαση σε βήμα l (1 $\leq l \leq d$),η τιμή του $B^{(l)}(R)$ πρέπει να ανανεωθεί για κάθε αντικείμενο R που έχει ειδωθεί μέχρι τώρα. Αυτό μπορεί να είναι πάνω από lm ανανεώσεις για κάθε μήκος l, το οποίο αποφέρει συνολικά Ω(d^2 m)ανανεώσεις με μήκος d. Επιπλέον, ανόμοια με το TA, δεν είναι πλέον αρκετό να έχουμε φραγμένους buffers. Οπότε, για συγκεκριμένη λειτουργία όπως το min είναι δυνατό χρησιμοποιώντας κατάλληλες δομές δεδομένων ο υπολογισμός μπορεί να είναι αρκετά απλοποιημένος. Αυτό είναι ένα θέμα για περαιτέρω έρευνα.

Λαμβάνοντας υπόψιν το κόστος τυχαίας προσπέλασης(αλγόριθμος CA)

Τώρα θα παρουσιάσουμε το συνδυασμένο αλγόριθμο CA ο οποίος κάνει χρήση τυχαίων προσπελάσεων, αλλά παίρνει υπόψιν το κόστος τους. Όπως πριν , έστω cs είναι το κόστος μία ταξινομημένης προσπέλασης και cR είναι το κόστος μία τυχαίας προσπέλασης. Το μέσο κόστος ενός αλγορίθμου ο οποίος κάνει s ταξινομημένες προσπελάσεις και r τυχαίες είναι $sc_S + rc_R$. Γνωρίζουμε ότι ο TA είναι κατάλληλος κατά το στιγμιότυπο , οπότε, ο λόγος καταλληλότητας είναι μία συνάρτηση του σχετικού κόστος της τυχαίας προσπέλασης σε ταξινομημένη προσπέλαση, το οποίο είναι c_R / c_S . Ο στόχος μας είναι να βρούμε έναν αλγόριθμο που είναι κατάλληλος κατά το

στιγμιότυπο και όπου ο λόγος καταλληλότητας είναι ανεξάρτητος από το c_R / c_S. Κανείς μπορεί να δει τον CA σαν μία ένωση ανάμεσα στους TA και NRA. Έστω $h = \lfloor c_R / c_S \rfloor$. Υποθέτουμε ότι $c_R \geq c_S$, έτσι ώστε $h \geq 1$. Η ιδέα του CA είναι να τρέξουμε τον NRA, αλλά κάθε h βήματα να τρέχουμε μία φάση τυχαίας προσπέλασης και να ανανεώνουμε τη πληροφορία (τα άνω και κάτω φράγματα B και W).Και εδώ όπως και πριν χρειαζόμαστε μόνο το ότι η έξοδος αποτελείται από τα κορυφαία κ αντικείμενα χωρίς τους βαθμούς τους. Αν επιθυμούμε να έχουμε και τους βαθμούς, αυτό θα απαιτεί μόνο ένα σταθερό αριθμό από επιπρόσθετες τυχαίες προσπελάσεις, και έτσι δεν θα έχει αποτέλεσμα στη καταλληλότητα στιγμιότυπου.

Ο αλγόριθμος CA έχει ως εξής:

1. Κάνε ταξινομημένες προσπελάσεις παράλληλα σε κάθε μία από τις m ταξινομημένες λίστες Li. Σε κάθε μήκος d(όποτε τα d αντικείμενα έχουν προσπελαστεί κάτω από ταξινομημένη προσπέλαση σε κάθε λίστα):
   - Διατήρησε τις βασικές τιμές $x_1^{(d)}, x_2^{(d)}, \ldots\ldots, x_m^{(d)}$ καταμετρημένες στις λίστες.
   - Για κάθε αντικείμενο R με πεδία που έχουν ανακαλυφτεί $S = S^{(d)} \leq \{1,\ldots m\}$, υπολόγισε τις τιμές $W^{(d)}(R) = W_S(R)$ *και* $B^{(d)}(R) = B_S(R)$.(Για αντικείμενα R που δεν έχουν ειδωθεί ακόμη, αυτές οι τιμές έχουν εικονικά υπολογιστεί σαν $W^{(d)}(R) = W_S(R)$ *και* $B^{(d)}(R) = B_S(R)$.(που είναι η τιμή σκανδάλης.)
   - Έστω $T_k^{(d)}$, η συγκεκριμένη λίστα κορυφαίων k ,που περιέχει τα κ αντικείμενα με τη μεγαλύτερη τιμές W(d) που έχουν ειδωθεί μέχρι τώρα(και τους βαθμούς τους), αν δύο αντικείμενα έχουν την ίδια $W^{(d)}$ τιμή , τότε οι δεσμοί έχουν σπάσει χρησιμοποιώντας τις $B^{(d)}$) τιμές, τέτοια ώστε το αντικείμενο με τη μεγαλύτερη $B^{(d)}$ τιμή κερδίζει. Έστω $M_k^{(d)}$ είναι η κ μεγαλύτερη $W^{(d)}$ τιμή της $T_k^{(d)}$.
2. Καλούμε ένα αντικείμενο R βιώσιμο αν $B^{(d)}(R) > M_k^{(d)}$. Κάθε $h = \lfloor c_R / c_S \rfloor$ βήματα(αυτό είναι κάθε φορά που το μήκος της ταξινομημένης προσπέλασης αυξάνεται κατά h), κάνε τα ακόλουθα: μάζεψε τα βιώσιμα αντικείμενα που έχουν ειδωθεί για τα οποία δεν είναι όλα τα πεδία γνωστά και των οποίων η $B^{(d)}$ τιμή είναι τόσο μεγάλη όσο γίνεται (οι δεσμοί έχουν σπάσει). Υλοποίησε τυχαίες προσπελάσεις για όλα από αυτά (το πολύ m-1) χαμένα πεδία. Αν δεν υπάρχει τέτοιο αντικείμενο , τότε μην κάνει τυχαία προσπέλαση σε αυτό το βήμα.

3. Σταμάτα όταν α) τουλάχιστον κ ευδιάκριτα αντικείμενα έχουν ειδωθεί (έτσι ώστε ειδικά $T_k^{(d)}$. περιέχουν κ αντικείμενα) και b) δεν υπάρχουν βιώσιμα αντικείμενα που έχουν μείνει απέξω από το $T_k^{(d)}$. , το οποίο ισχύει όταν $B^{(d)}(R) \leq M_k^{(d)}$ για κάθε R που δεν ανήκει στο $T_\kappa^{(d)}$. Επέστρεψε τα αντικείμενα στο $T_\kappa^{(d)}$.

Ας σημειώσουμε ότι αν το h είναι πολύ μεγάλο (λέμε μεγαλύτερο από τον αριθμό των αντικειμένων στη βάση δεδομένων), τότε ο αλγόριθμος CA είναι το ίδιο σαν τον NRA, αφού καμία τυχαία προσπέλαση δεν υλοποιείται. Αν h = 1 ,τότε ο αλγόριθμος CA είναι όμοιος με τον TA, αλλά διαφορετικός σε ορισμένες περιπτώσεις . Για κάθε βήμα που γίνεται ταξινομημένη προσπέλαση παράλληλα, ο CA υλοποιεί τυχαίες προσπελάσεις για όλα τα υπολειπόμενα πεδία κάποιων αντικειμένων. Αντί αυτού ο TA υλοποιεί τυχαίες προσπελάσεις για όλα τα υπολειπόμενα πεδία κάθε αντικειμένου που έχει ειδωθεί με ταξινομημένη προσπέλαση.

Για λογικές τιμές του h δεν ισχύει η περίπτωση όπου ο CA είναι ισοδύναμος , στον ασυνεχή αλγόριθμο που εκτελεί h βήματα του NRA και ένα βήμα του TA. (Αυτό γιατί ,ο ασυνεχής αλγόριθμος εκτελεί τυχαίες προσπελάσεις στην ίδια χρονική σειρά με τον TA, αλλά απλά καθυστερεί, έτσι ώστε να κάνει τυχαίες προσπελάσεις κάθε h βήματα.) Θα δείξουμε αργότερα ένα παράδειγμα όπου ο μέσος αλγόριθμος απόδιδει χειρότερα από τον CA. Οι διαφορές μεταξύ των αλγορίθμων είναι ότι ο CA διαλέγει τυχαία σε ποια αντικείμενα θα υλοποιήσει τη τυχαία προσπέλαση, ονομαστικά, σύμφωνα με τις B(d) τιμές τους. Έτσι, για να σχεδιάσουμε τον κατάλληλο κατά το στιγμιότυπο αλγόριθμο CA χρειαζόμαστε επίσης μία αρχή πάνω σε ποια αντικείμενα θα υλοποιήσομε τη τυχαία προσπέλαση. Αυτό δεν το συναντήσαμε σα ζήτημα καθώς σχεδιάζαμε τον TA, αφού τότε ,οι τυχαίες προσπελάσεις αύξαναν το κόστος μόνο με σταθερό πολλαπλάσιο.

Η ορθότητα του CA είναι ουσιαστικά η ίδια με του NRA, αφού τα ίδια άνω και κάτω όρια διατηρούνται.

Θεώρημα 4.8

Αν η αθροιστική συνάρτηση t είναι μονότονη, τότε ο CA ορθά βρίσκει τα κορυφαία κ αντικείμενα.

Στην επόμενη παράγραφο , θεωρούμε σενάρια κάτω από τα οποία ο CA είναι κατάλληλος κατά το στιγμιότυπο, με λόγο καταλληλότητας ανεξάρτητο από το c_R / c_S .

Καταλληλότητα στιγμιότυπου του CA

Προηγουμένως είχαμε δώσει δύο περιπτώσεις υπό τις οποίες ο ΤΑ είναι κατάλληλος κατά το στιγμιότυπο πάνω από τους A,D. Στη πρώτη περίπτωση η αθροιστική συνάρτηση είναι μονότονη ,η D είναι η κλάση όλων των βάσεων δεδομένων και A είναι η κλάση όλων των αλγορίθμων που ορθώς βρίσκουν τα κορυφαία κ αντικείμενα για t για κάθε βάση δεδομένων και που δεν κάνουν ισχυρές μαντεψιές . Στο δεύτερο σενάριο η αθροιστική συνάρτηση t είναι αυστηρώς μονότονη, η D είναι η κλάση όλων των βάσεων δεδομένων που ικανοποιεί την ιδιότητα της ευκρίνειας και A είναι η κλάση όλων των αλγορίθμων που ορθά βρίσκουν τα κορυφαία κ αντικείμενα για το t για κάθε βάση δεδομένων στο D. Μπορεί να ελπίζουμε ότι κάτω από όποιο από αυτά τα δύο σενάρια, ο CA είναι κατάλληλος κατά το στιγμιότυπο, με λόγο καταλληλότητας. Δυστυχώς , αυτή η ελπίδα είναι λανθασμένη, και στα δύο σενάρια. Στη πραγματικότητα, όχι μόνο ο CA αποτυγχάνει να καλύψει αυτή την ελπίδα, αλλά έτσι κάνει κάθε άλλος αλγόριθμος. Με άλλα λόγια, κανένα από αυτά τα σενάρια δεν είναι αρκετό για να διασφαλίσει την ύπαρξη ενός αλγορίθμου με λόγο καταλληλότητας ανεξάρτητο από το cR/cs.

Αποδεικνύεται, ότι αν ελαφρώς ισχυροποιήσουμε την υπόθεση στο t στο δεύτερο σενάριο, ο CA γίνεται βέλτιστος κατά το στιγμιότυπο , με λόγο καταλληλότητας ανεξάρτητο από το cR/cs. Ας πούμε ότι η αθροιστική συνάρτηση t είναι αυστηρώς μονότονη σε κάθε όρισμα αν οποτεδήποτε ένα όρισμα αυξάνεται αυστηρά και τα εναπομείναντα ορίσματα κρατιούνται καθορισμένα , τότε η τιμή της αθροιστικής συνάρτησης είναι αυστηρώς αύξουσα. Αυτό γιατί, η t είναι αυστηρώς μονότονη σε κάθε όρισμα αν xi<xi΄ το οποίο συνεπάγεται ότι

$$t(x_1,\ldots,x_{i-1},x_i,x_{i+1},\ldots,x_m) < t(x_1,\ldots,x_{i-1},x_{i'},x_{i+1},\ldots,x_m).$$

Ο μέσος όρος (ή άθροισμα) είναι αυστηρώς μονότονο σε κάθε όρισμα, ενώ το ελάχιστο (min) δεν είναι.

Αποδεικνύεται ότι στο δεύτερο σενάριο ,αν αντικαταστήσουμε το «την αθροιστική συνάρτηση t είναι αυστηρώς μονότονη» ,με το << η αθροιστική συνάρτηση είναι αυστηρώς μονότονη για κάθε όρισμα>> τότε ο CA είναι κατάλληλος κατά το στιγμιότυπο , με λόγο καταλληλότητας ανεξάρτητο από το c_R / c_S .Αποδεικνύεται επίσης ότι ίδιο αποτέλεσμα ισχύει αν αντί , απλά πάρουμε το t να είναι η min , ακόμη και αν η min δεν είναι αυστηρώς μονότονη σε κάθε όρισμα.

ΘΕΩΡΗΜΑ 4.9

Υποθέτουμε ότι η αθροιστική συνάρτηση t είναι αυστηρώς μονότονη για κάθε όρισμα. Έστω D είναι η κλάση όλων των βάσεων δεδομένων που ικανοποιούν την ιδιότητα της ευκρίνειας. Έστω A είναι η κλάση όλων των

αλγορίθμων που ορθά βρίσκουν τα κορυφαία κ αντικείμενα για t για κάθε βάση δεδομένων στο D. Τότε ο CA είναι κατάλληλος κατά το στιγμιότυπο πάνω από τον A και D, με λόγο καταλληλότητας ανεξάρτητο από το c_R / c_S .

ΘΕΩΡΗΜΑ 4.10

Έστω D είναι η κλάση όλων των βάσεων δεδομένων η οποία ικανοποιεί την ιδιότητα της ευκρίνειας . Έστω A είναι η κλάση όλων των αλγορίθμων η οποία ορθά βρίσκει τα κορυφαία κ αντικείμενα για ελάχιστο για κάθε βάση δεδομένων στο D. Τότε ο CA είναι κατάλληλος κατά το στιγμιότυπο πάνω από τα A,D με λόγο καταλληλότητας ανεξάρτητο από το c_R / c_S .

2.3 Η τεχνική του YI για επεξεργασία όψεων

Ένας αποτελεσματικός τρόπος για την βελτίωση της απόδοσης ακριβών επερωτήσεων κορυφαίων κ είναι να διατηρούμε τα αποτελέσματά τους σαν επεξεργασμένες όψεις. Αν και, η αυξανόμενη διατήρηση επεξεργασμένων κορυφαίων κ όψεων είναι ένα σχετικά ανεξερεύνητο πρόβλημα. Το κύριο σημείο αυτού του προβλήματος είναι ότι μία κορυφαία –κ όψη δεν αυτοανανεώνεται όσον αφορά τις διαγραφές και τις ενημερώσεις στο βασικό πίνακα. Και από αυτό προκύπτει ότι πρέπει να εκτελούμε επερωτήσεις προς το βασικό πίνακα προκειμένου να διατηρούμε τη κορυφαία κ όψη σωστή. Η όψη από μόνη της δεν περιέχει αρκετή πληροφορία για την διατήρηση της.

Για παράδειγμα , ας θεωρήσουμε μία επεξεργασμένη όψη που περιέχει δέκα μετοχές με τη μεγαλύτερη τιμή . Ας υποθέσουμε ότι μία από αυτές τις μετοχές πέφτει και η τιμή της πέφτει κάτω από τη κορυφαία δεκάδα. Μετά από αυτή την ανανέωση , η όψη ακόμη περιέχει τις κορυφαίες 9 μετοχές ,αλλά προκειμένου να έχουμε τη δέκατη στη σειρά μετοχή πρέπει να <<ρωτήσουμε>> το βασικό πίνακα. Αυτή η επερώτηση την οποία ονομάζουμε «επερώτηση ξαναγεμίσματος» μπορεί να είναι πολύ ακριβή για πληθώρα λόγων π.χ ο βασικός πίνακας μπορεί να είναι μεγάλος , το κριτήριο εύρεσης μπορεί να εμπλέκει ακριβές λειτουργίες.

Προκειμένου να αποφύγουμε ακριβές επερωτήσεις πάνω από το βασικό πίνακα ,μπορούμε να κάνουμε μία κορυφαία-κ όψη να αυτοενημερώνεται προσθέτοντας της βοηθητικά δεδομένα . Για παράδειγμα , μπορούμε να κρατάμε τη (κ+1)ιοστή πλειάδα σαν βοηθητικό δεδομένο για να βοηθήσουμε τη διατήρηση μίας κορυφαίας κ –όψης , στη περίπτωση που μια πλειάδα πέφτει έξω από τα κορυφαία κ. Προκειμένου όμως να διατηρούμε και τα βοηθητικά

δεδομένα χρειαζόμαστε και τη (κ+2)ιοστή πλειάδα ...δηλαδή χρειαζόμαστε ένα αντίγραφο όλης της βάσης .

Τώρα ερχόμαστε αντιμέτωποι με ένα δίλημμα. Η μία επιλογή είναι να διατηρούμε την αυθεντική κορυφαία-κ όψη ,η οποία μπορεί να απαιτεί συχνές δαπανηρές επερωτήσεις επαναγεμίσματος. Η άλλη επιλογή είναι να διατηρούμε ένα ταξινομημένο ευρετήριο πάνω σε όλη τη βάση το οποίο αποφεύγει τις επερωτήσεις επαναγεμίσματος όλες μαζί. Καμία επιλογή δεν φαίνεται να ξεχωρίζει.

Ευτυχώς , έχουμε ένα ενδιάμεσο επίπεδο να εξερευνήσουμε ανάμεσα στις δύο επιλογές, χωρίς να αποκλείουμε τις διαγραφές και τις ενημερώσεις. Πρώτα ,αντί της απαίτησης για ολοκληρωμένη αυτοενημέρωση θα προσπαθήσουμε να επιτύχουμε αυτοενημέρωση τη τρέχουσα χρονική στιγμή με υψηλή πιθανότητα. Η δεύτερη παρατήρηση είναι ότι μία επεξεργασμένη όψη μπορεί να έχει ένα δυναμικό ορισμό. Αντί να διατηρούμε μία κορυφαία-κ όψη διατηρούμε μία κορυφαία κ' όψη ,όπου το κ' μπορεί να αλλάζει δυναμικά μεταξύ του k και κάποιου kmax ≥ k. Ξεκινάμε με k'= kmax . Αυξάνουμε το k' κατά ένα όταν μία προσθήκη ή μία ενημέρωση ωθεί μία πλειάδα να μπει στη κορυφαία k' (εκτός και αν το k' είναι ήδη ίσο με το kmax) και μειώνουμε το k' κατά ένα όταν μία διαγραφή ή μία ενημέρωση ωθεί μία πλειάδα να φύγει από τη κορυφαία – k'. Ρωτάμε μόνο το βασικό πίνακα όταν το k' πέφτει κάτω από το k. Ξεκινώντας από το kmax αντί του k ,ελπίζουμε να χαμηλώσουμε τη συχνότητα επαναγεμίσματος και το κόστος διατήρησης της όψης.

Ο αλγόριθμος

Υποθέτουμε ότι ενδιαφερόμαστε για τις κορυφαίες κ πλειάδες από ένα βασικό πίνακα R μεγέθους N. Υποθέτουμε ότι το k είναι μία σταθερά πολύ μικρότερη από το N, αφού τυπικά οι χρήστες ενδιαφέρονται μόνο για ένα μικρό υποσύνολο του R που είναι πιο σημαντικό π.χ τα δέκα πιο δημοφιλή τραγούδια ή τις 100 πιο συχνά προσπελάσιμες ιστοσελίδες. Υποθέτουμε ότι οι πλειάδες στον R ταυτοποιούνται από μία στήλη με id και κατατάσσονται σύμφωνα με τιμή σε μία στήλη val. Πλειάδες με μεγαλύτερες τιμές , κατατάσσονται υψηλότερα . Για απλότητα, θεωρούμε ότι όλες οι τιμές είναι ευδιάκριτες . Η στήλη val μπορεί να αποθηκευτεί στον R ή να υπολογιστεί στον αέρα από κάποια οριζόμενη από το χρήστη λειτουργία. Θεωρούμε ότι δεν υπάρχει δείκτης στο R.val.

Ο αλγόριθμος μας είναι πολύ απλός. Κρατάμε τις κορυφαίες k' πλειάδες (με τα id, val) σε μία επεξεργασμένη όψη V ,όπου k' μπορεί να ποικίλει μεταξύ του κ και κάποιου kmax ≥ k. Αφού k ≤ κ' , μπορούμε να απαντήσουμε τις κορυφαίες κ επερωτήσεις χρησιμοποιώντας τα περιεχόμενα του V.

Πρέπει να διατηρούμε το V με δοσμένες τις αλλαγές στο βασικό πίνακα R. Για να κρατάμε την ανάλυση μας καθαρή , θεωρούμε μόνο τις ενημερώσεις

στο R. Αυτό σημαίνει ότι θεωρούμε τις ταυτότητες των N πλειάδων στο R να παραμένουν διορθωμένες όταν αλλάζουν οι τιμές τους κατά τη διάρκεια του χρόνου. Προκύπτει εύκολα η γενίκευση του αλγορίθμου μας προκειμένου να χειριζόμαστε και διαγραφές και προσθήκες στο R.

Έστω uk' είναι η τιμή της χαμηλότερης στη κατάταξη πλειάδας στο V. Θεωρούμε ότι μία ενημέρωση στο R έχει τη φόρμα (id,val), όπου val είναι η νέα τιμή της πλειάδας που αυθεντικοποιείται από το id. Για κάθε ενημέρωση στο R, υλοποιούμε μία ενημέρωση στο V.

Υπάρχουν τέσσερις περιπτώσεις για να λάβουμε υπόψιν:

- Η πλειάδα που αναγνωρίζεται από το id δεν είναι στο V, και val < vk'. Αυτή η ενημέρωση δεν έχει επίδραση στο V.Την ονομάζουμε *αγνοήσιμη* ενημέρωση.
- Η πλειάδα που αναγνωρίζεται από το id είναι στο V, και val > vk'. Ανανεώνουμε τη τιμή αυτής της πλειάδας στο V στο val. Καλούμε αυτή την ενημέρωση ουδέτερη (« ουδέτερη» με την έννοια ότι δεν αλλάζει τη τιμή του k').
- Η πλειάδα που αναγνωρίζεται από το id δεν είναι στο V, και val > vk'.
  Εισάγουμε (id,val) στο V. Καλούμε αυτή την ενημέρω καλή ενημέρωση («καλή» με την έννοια ότι αυξάνει το k' κατά ένα). Αν το k' υπερβαίνει το kmax , διαγράφουμε τη χαμηλότερη στα κατάταξη πλειάδα στο V.
- Η πλειάδα που αναγνωρίζεται από το id είναι στο V, και val < vk'. Διαγράφουμε την νέα πλειάδα από το V. Καλούμε αυτή την ενημέρωση κακή («κακή» με την έννοια ότι μειώνει το k' κατά ένα).Αν το k' πέσει κάτω από το k, εκτελούμε επαναγέμισμα όπως περιγράφτηκε προηγουμένως.

Η διαδικασία ξαναγεμίσματος ρωτάει το βασικό πίνακα και αποθηκεύει το μέγεθος της όψης στο kmax. Αυτή η λειτουργία αποτελείται από τα ακόλουθα δύο βήματα:

- Εκτίμησε την επερώτηση επαναγεμίσματος πάνω από το R , η οποία επιστρέφει όλες τις πλειάδες που κατατάσσονται μεταξύ k ,kmax. Σημειώνουμε ότι στο χρόνο επαναγεμίσματος αν k>1 , το V περιέχει την (k-1) στη σειρά πλειάδα. Μπορούμε να χρησιμοποιήσουμε τη τιμή αυτής της πλειάδας ,vk-1 ,για να βελτιώσουμε τη επερώτηση επαναγεμίσματος ως εξής : «επέστρεψε τις κορυφαίες kmax-k+1 πλειάδες μεταξύ αυτών των οποίων οι τιμές είναι μικρότερες από vk-1.»
- Πρόσθεσε το αποτέλεσμα της επερώτησης επαναγεμίσματος στο V.

2.4 Τεχνική των Babcock , Olston

Οι προηγούμενες μέθοδοι προϋποθέτουν ότι όλα τα σχετικά δεδομένα είναι διαθέσιμα (τοπικά ή σε κατανεμημένους εξυπηρετητές) πριν την επεξεργασία. Επιπλεόν , αναφέρουν ένα αποτέλεσμα και τερματίζουν. Από την άλλη σε περιβάλλοντα ροών τα δεδομένα δεν είναι γνωστά κατά τη διάρκεια της επεξεργασίας ,αλλά αλλάζουν καθώς νέες πλειάδες φθάνουν και παλιές λήγουν. Ο σκοπός μας είναι να ελέγχουμε διαρκώς τις κορυφαίες κ πλειάδες επερωτήσεων σύμφωνα με τις πλειάδες που φθάνουν και που λήγουν. Η μόνη σχετική εργασία είναι αυτή των Babcock και Olston οι οποίοι εισάγουν την έννοια του κατανεμημένου ελέγχου κορυφαίων-κ. Ο στόχος τους είναι να ενημερώνουν συνεχώς τα κ αντικείμενα με το μεγαλύτερο αθροιστικά βαθμό πάνω από ένα σύνολο από πηγές ροών. Προκειμένου να μειώσουν το κόστος επικοινωνίας , διατηρούν αριθμητικούς περιορισμούς στις πηγές των ροών για να διασφαλίσουν ότι οι πιο πρόσφατα αναφερόμενες απαντήσεις παραμένουν έγκυρες. Όταν ένας περιορισμός παραβιάζεται, η αντίστοιχη πηγή το αναφέρει στο κεντρικό εξυπηρετητή, ο οποίος ανανεώνει το σύνολο των κορυφαίων κ και αναθέτει νέους περιορισμούς στις πηγές.

Το θέμα όμως με το οποίο ασχολούμαστε εμείς είναι διαφορετικό γιατί αντιμετωπίζουμε πολλαπλές κανονικές επερωτήσεις κορυφαίων κ πάνω από μία μόνο πολυδιάστατη ροή. Και επιπλέον ενώ οι Babcock και Olston σκοπό έχουν να ελαχιστοποιήσουν το φόρτο στο δίκτυο ,εμείς ασχολούμαστε με την ελαχιστοποίηση της χρήσης της CPU από τη πλευρά του εξυπηρετητή.

2.5 Συνεχόμενος έλεγχος για σχετικούς τύπους ερωτημάτων

Τα κορυφαία κ ερωτήματα είναι σχετικά με τα skylines, και άλλα παρόμοια προβλήματα όπως η πολύαντικειμενική βελτιστοποίηση και οι μέγιστοι πίνακες.

Το skyline ενός d-διάστατου συνόλου δεδομένων , περιέχει τα σημεία που δεν κυριαρχούνται από άλλα σημεία σε όλες τις διαστάσεις. Ένα σημείο p_i ,λέγεται ότι κυριαρχεί επάνω σε ένα άλλο σημείο p_j, αν και μόνο αν το p_i είναι προτιμότερο από το p_j για κάθε γνώρισμα. Αυτό σημαίνει ότι το p_i έχει μεγαλύτερο βαθμό από το p_j σύμφωνα με κάποια συνάρτηση προτίμησης , η οποία είναι μονότονη σε όλα τα γνωρίσματα. Ο υπολογισμός του skyline έχει πρόσφατα δεχτεί αρκετή προσοχή από τη κοινότητα βάσεων δεδομένων, ειδικά όσον αφορά αλγορίθμους που επιστρέφουν γρήγορα τα αρχικά αποτελέσματα χωρίς να πρέπει να διαβάσουν όλη τη βάση.
Ο skyline τελεστής είναι σημαντικός για αρκετές εφαρμογές που επιδιώκουν λήψη αποφάσεων με πολλά κριτήρια.

Η διαχείριση του skyline δρομολογείται με αλγόριθμους που διαγράφουν καταχωρήσεις που δεν μπορούν να συμμετάσχουν στο skyline μέχρι την λήξη τους.

Τα ερωτήματα κορυφαίων –k (όπως και τα skylines) έχουν ένα πολυδιάστατο καθήκον ,αφού η κάθε καταχώρηση p μπορεί να ειδωθεί σαν ένα σημείο (στο d-διάστατο χώρο) που καθορίζεται από τις τιμές γνωρισμάτων $p.x_1$, $p.x_2$, $p.x_3$ ...$p.x_d$.

3. ΜΟΝΤΕΛΑ ΚΑΙ ΘΕΜΑΤΑ ΣΧΕΤΙΚΑ ΜΕ ΣΥΣΤΗΜΑΤΑ ΡΟΗΣ ΔΕΔΟΜΕΝΩΝ

3.1

Τα τελευταία χρόνια μία καινούργια κλάση εφαρμογών δεδομένων έχει γίνει ιδιαίτερα γνωστή: εφαρμογές στις οποίες τα δεδομένα μοντελοποιούνται καλύτερα όχι σαν μόνιμες σχέσεις αλλά σαν προσωρινές ροές δεδομένων. Παραδείγματα τέτοιων εφαρμογών περιλαμβάνουν το τομέα της χρηματοοικονομίας , δίκτυα αισθητήρων , ασφάλεια, διαχείριση τηλεπικοινωνιακών δεδομένων, εφαρμογές web κ.α. Στο μοντέλο ροών δεδομένων , ατομικά στοιχεία δεδομένων μπορεί να είναι σχεσιακές πλειάδες π. χ. μετρήσεις δικτύου , επισκέψεις σε ιστοσελίδες, διαβάσματα αισθητήρων κ.α. Αλλά οι συνεχόμενες αφίξεις σε πολλαπλές ,γρήγορες , πιθανώς απροσδιόριστες και μη φραγμένες ροές δημιουργούν καινούργια προβλήματα.

Σε όλες τις εφαρμογές που παρουσιάστηκαν προηγουμένως , δεν είναι εφικτό να φορτώσουμε απλά τα δεδομένα που καταφθάνουν σε ένα παραδοσιακό σύστημα διαχείρισης βάσεων δεδομένων και να λειτουργήσουμε εκεί. Τα παραδοσιακά DBMS δεν είναι σχεδιασμένα για γρήγορο και συνεχόμενο φόρτωμα δεδομένων , και δεν υποστηρίζουν τις συνεχόμενες επερωτήσεις που είναι βασικό στοιχείο σε εφαρμογές ροών δεδομένων.

Στη συνέχεια , δεν είναι επί του θέματος μας να παρουσιάσουμε κάποιο άλλο ολοκληρωμένο σύστημα για τη διαχείριση ροών δεδομένων ,θα περιοριστούμε στην παρουσίαση και επεξήγηση εννοιών όπως: το μοντέλο ροής δεδομένων , επερωτήσεις πάνω από ροές , αλλά και ζητήματα πιο ειδικά της επεξεργασίας επερωτήσεων όπως απαίτηση για μνήμη χώρις όριο , ολισθαίνοντα παράθυρα, επεξεργασία κατά τμήματα, δειγματοληψία και σύνοψη, τελεστές blocking , χρονοαποτυπώματα.

Μοντέλο ροής δεδομένων

Στο μοντέλο ροής δεδομένων ένα μέρος ή το σύνολο των δεδομένων που έρχονται στην είσοδο για να επεξεργασθούν δεν είναι διαθέσιμα για τυχαία προσπέλαση από δίσκο ή μνήμη ,αλλά έρχονται σαν συνεχόμενες ροές δεδομένων. Η διαφορά με παλιότερα μοντέλα αφορά κυρίως ότι :

- Τα δεδομένα φθάνουν online.
- Το σύστημα δεν ελέγχει τη σειρά με την οποία καταφθάνουν τα δεδομένα για να τεθούν σε επεξεργασία .
- Οι ροές δεδομένων είναι εν δυνάμει δίχως όριο στο μέγεθος τους.
- Όταν ένα στοιχείο μίας ροής δεδομένων έχει τεθεί σε επεξεργασία , είτε απορρίπτεται είτε εγκρίνεται - δεν μπορεί να ανακτηθεί εύκολα εκτός και αν είναι επαρκώς αποθηκευμένο στη μνήμη, χώρος που τυπικά είναι μικρότερος σε σχέση με το μέγεθος της ροής δεδομένων.

Η λειτουργία με το μοντέλο της ροής δεδομένων δεν αποκλείει την παρουσία ορισμένων δεδομένων που είναι αποθηκευμένα με συμβατικούς τρόπους. Συχνά, τα ερωτήματα σε ροές δεδομένων μπορεί να επιτυγχάνουν ενώσεις ανάμεσα σε ροές δεδομένων και αποθηκευμένων σχεσιακών δεδομένων. Θα υποθέσουμε ότι αν χρησιμοποιούνται συμβατικές σχέσεις αποθήκευσης ,η αναλογία τους παραμένει στατική. Έτσι ,αποκλείουμε κάθε ζήτημα πιθανής επεξεργασίας θεμάτων που μπορεί να προκύπτουν από τη παρουσία ενημερώσεων σε σχέσεις που δρουν παράλληλα με την επεξεργασία ροών δεδομένων.

Επερωτήσεις

Οι επερωτήσεις πάνω σε συνεχόμενες ροές δεδομένων έχουν πολλά κοινά με ερωτήματα σε ένα παραδοσιακό σύστημα επεξεργασίας δεδομένων. Υπάρχουν όμως δύο σημαντικές διακρίσεις ασυνήθιστες στο μοντέλο της ροής δεδομένων .Η πρώτη είναι ανάμεσα σε ενός χρόνου ερωτήματα και σε συνεχόμενα ερωτήματα:

- Τα ενός χρόνου ερωτήματα (μία κλάση η οποία περιλαμβάνει παραδοσιακά ερωτήματα DBMS(Data Base Management System)) είναι ερωτήματα που εκτιμώνται μία φορά πάνω σε ένα σημείο-χρόνο στιγμιότυπο του συνόλου δεδομένων ,με την απάντηση να επιστρέφει στο χρήστη .
- Τα συνεχόμενα ερωτήματα ,από την άλλη, εκτιμώνται συνεχόμενα όσο ροές δεδομένων συνεχίζουν να καταφθάνουν. Τα συνεχόμενα ερωτήματα είναι η πιο ενδιαφέροντα κλάση των ερωτημάτων σε ροές δεδομένων , και εκεί θα αφιερώσουμε τη περισσότερη προσοχή μας. Η απάντηση σε συνεχόμενα ερωτήματα παράγεται κάθε φορά ,πάντα

ανάλογα με τη ροή δεδομένων που ειδώθηκε μέχρι πρόσφατα. Οι συνεχόμενες απαντήσεις ερωτημάτων μπορούν να αποθηκεύονται και να ενημερώνονται καθώς νέα δεδομένα καταφθάνουν ,ή μπορεί να παράγονται σαν ροές δεδομένων οι ίδιες.

Κάποιες φορές ο ένας ή άλλος τρόπος προτιμάται. Για παράδειγμα , οι αθροιστικές επερωτήσεις μπορεί να εμπλέκουν συχνές αλλαγές προκειμένου να απαντήσουν πλειάδες ,υπαγορεύοντας έτσι την προσέγγιση αποθήκευσης. Από την άλλη οι ενώσεις ερωτημάτων μπορούν να παράγουν γρήγορα, όιχως όρια απαντήσεις, υπαγορεύοντας την προσέγγιση της ροής.
Η δεύτερη διάκριση είναι ανάμεσα σε προκαθορισμένα ερωτήματα και σε ad hoc:

- Ένα προκαθορισμένο ερώτημα είναι αυτό που προμηθεύεται στο σύστημα πριν έρθει κάποια σχετική πληροφορία .Τα προκαθορισμένα ερωτήματα είναι γενικά συνεχόμενα ερωτήματα, αν και προγραμματισμένα ενός χρόνου ερωτήματα μπορούν και αυτά να προκαθοριστούν.
- Τα Ad hoc ερωτήματα ,από την άλλη, προκύπτουν online αφού οι ροές δεδομένων έχουν ήδη ξεκινήσει. Ad hoc ερωτήματα μπορούν είτε να είναι ενός χρόνου είτε συνεχόμενα. Τα Ad hoc ερωτήματα περιπλέκουν το σχεδιασμό ενός συστήματος διαχείρισης ροών δεδομένων ,γιατί η σωστή απάντηση για ένα ad hoc ερώτημα μπορεί να απαιτεί την αναφορά σε στοιχεία δεδομένων που ήδη έχουν φτάσει στις ροές δεδομένων (και εν δυνάμει έχουν αποβληθεί).

Παραδείγματα

Μπορούν να βρεθούν αρκετά παραδείγματα που να υποδεικνύουν την ανάγκη για συστήματα που σχετίζονται με ερωτήματα πάνω σε συνεχόμενες ροές δεδομένων

- Το Traderbot είναι μία δικτυακή μηχανή αναζήτησης που αποτιμάει ερωτήματα πάνω σε πραγματικού χρόνου ροές οικονομικών δεδομένων. Το Traderbot δίνει αρκετά παραδείγματα ενός χρόνου και συνεχόμενων ροών δεδομένων που κυρίως τίθενται από τους πελάτες.
- Σύγχρονες εφαρμογές ασφάλειας εφαρμόζουν κανόνες πάνω σε ροές δικτυακών πακέτων.Π.χ iPolicy Network
- Μεγάλες ιστοσελίδες παρακολουθούν τη κίνηση στο δίκτυο προκειμένου να καθιστούν εφικτές διάφορες εφαρμογές όπως εξατομίκευση , έλεγχος απόδοσης.
- Υπάρχουν πολλές εφαρμογές στο χώρο της παρακολούθησης αισθητήρων ,όπου ένας μεγάλος αριθμός αισθητήρων κατανέμεται στο

φυσικό κόσμο και γεννάει ροές δεδομένων που πρέπει να παρακολουθούνται , να αναλύονται.

Ο τομέας εφαρμογών που χρησιμοποιούμε για πιο λεπτομερή παραδείγματα είναι η διαχείριση κίνησης του δικτύου, που εμπλέκει παρακολούθηση πληροφορίας πακέτων (network packet header information) κατά μήκος ενός συνόλου από δρομολογητές προκειμένου να αποκτήσουμε πληροφορίες για τη κίνηση.

Ας θεωρήσουμε το σύστημα διαχείρισης της δικτυακής κίνησης ενός μεγάλου δικτύου , π.χ το δικτυακό κορμό(backbone) ενός (ISP) παροχέα δικτύου. Τέτοια συστήματα παρακολουθούν μία ποικιλία συνεχόμενων ροών δεδομένων που μπορούν να χαρακτηρίζονται και ως απρόβλεπτες και να καταφθάνουν με υψηλή συχνότητα, περιλαμβάνοντας ίχνη πακέτων και μέτρα απόδοσης δικτύου. Τυπικά, υπαρκτά εργαλεία διαχείρισης της κίνησης είτε βασίζονται σε ένα σύστημα ειδικού σκοπού που πραγματώνει online επεξεργασία απλών συνεχόμενων ερωτημάτων ,ή απλά διακόπτει την κίνηση στο δίκτυο και κάνει επερωτήσεις κατά περιόδους. Τα συνήθη συστήματα διαχείρισης βάσεων δεδομένων κρίνουν ανεπαρκές να παρέχουν το είδος της online επεξεργασίας συνεχόμενων επερωτήσεων που θα ήταν περισσότερο ωφέλιμη σε αυτό το τομέα. Ένα σύστημα ροής δεδομένων που μπορεί να παρέχει αποτελεσματική online επεξεργασία συνεχόμενων επερωτήσεων πάνω σε ροές δεδομένων θα επέτρεπε σε τελεστές δικτύου να εγκαταστήσουν , μετατρέψουν ή να διαγράψουν τις κατάλληλες επερωτήσεις παρακολούθησης για να επιτρέπουν έτσι αποτελεσματική διαχείριση των πηγών του παροχέα δικτύου(ISP).

Ας θεωρήσουμε τα ακόλουθα :

Ίχνη δικτυακών πακέτων συλλέγονται από έναν αριθμό από συνδέσεις στο δίκτυο. Το ζήτημα είναι σε δύο συγκεκριμένες συνδέσεις: μία σύνδεση πελάτη,C, η οποία συνδέει το δίκτυο ενός πελάτη στο δίκτυο του παροχέα δικτύου, και μία σύνδεση ενός δικτυακού κορμού, B , η οποία συνδέει δύο δρομολογητές με το δίκτυο του δικτυακού κορμού(backbone) του παροχέα δικτύου. Έστω ότι τα C,B συμβολίζουν δύο ροές ιχνών πακέτων που ανταποκρίνονται σε αυτές τις δύο συνδέσεις. Υποθέτουμε ότι, για απλότητα, τα ίχνη περιέχουν μόλις τα πέντε πεδία της επικεφαλίδας του πακέτου που φαίνονται παρακάτω :

src: διεύθυνση IP του αποστολέα του πακέτου

dest: διεύθυνση IP του παραλήπτη του πακέτου

Id: αριθμός αυθεντικοποίησης που δίνεται από τον αποστολέα έτσι ώστε ο προορισμός να μπορεί με μοναδικό τρόπο να αναγνωρίζει το κάθε πακέτο

Len: μήκος του πακέτου

Time: χρόνος που η επικεφαλίδα του πακέτου έχει καταγραφεί

Έστω πρώτα η συνεχόμενη επερώτηση Q1, η οποία υπολογίζει το φορτίο στη σύνδεση B υπολογισμένο μεταξύ ενός λεπτού παύσεων , σημειώνοντας το τελεστή του δικτύου όταν το φορτίο ξεπεράσει ένα συγκεκριμένο κατώφλι t. Οι συνάρτησεις getminute και notifyoperator έχουν τη φυσική σημασία.

Q1: SELECT notifyoperator(sum(len))
FROM B
GROUP BY getminute(time)
HAVING sum(len)>t

Ενώ η λειτουργικότητα μίας τέτοιας επερώτησης μπορεί πιθανό να επιτευχθεί σε ένα DBMS μέσω της χρήσης μηχανισμού σκανδαλών , μπορούμε να προτιμήσουμε τη χρήση συγκεκριμένων τεχνικών για λόγους αποδοτικότητας. Για παράδειγμα, θεωρούμε τη περίπτωση όπου η σύνδεση B έχει πολύ υψηλή παραγωγή(π.χ αν είναι οπτική σύνδεση).Σε αυτή τη περίπτωση , μπορούμε να διαλέξουμε να υπολογίσουμε μία περίπου σωστή απάντηση στο Q1 προχωρώντας σε τυχαία δειγματοληψία στη ροή.(που είναι ένα καθήκον έξω από τους στάνταρ μηχανισμούς σκανδαλών).

Η δεύτερη επερώτηση Q2 απομονώνει ροές στη σύνδεση του δικτυακού κορμού (backbone) και καθορίζει το ποσό της κίνησης που γεννιέται σε κάθε ροή. Μία ροή ορίζεται σαν μία σειρά από πακέτα ομαδοποιημένα σύμφωνα με το χρόνο, που έχουν επίσης αποσταλεί από συγκεκριμένη πηγή και κατευθύνονται σε συγκεκριμένο προορισμό.

Q2: SELECT flowid,src, dest, sum(len) AS flowlen
FROM (SELECT src,dest, len, time
FROM B
ORDER BY time)
GROUP BY src, dest, getflowid(src,dest,time)
AS flowid

Εδώ η getflowid είναι μία συνάρτηση που ορίζεται από το χρήστη η οποία παίρνει τη IP διεύθυνση πηγή, και ένα αποτύπωμα ενός πακέτου, και επιστρέφει τον αναγνωριστή της ροής στην οποία το πακέτο ανήκει.

Υποθέτουμε ότι τα δεδομένα στην όψη(ή έκφραση πίνακα) στη πρόταση της FROM περνιούνται στην getflowid σύμφωνα με τη σειρά που ορίζεται από την ORDER BY.

Παρατηρούμε ότι θέτοντας την επερώτηση Q2 πάνω από τη ροή B είναι μερικώς προκλητικό λόγω της παρουσίας GROUP BY και ORDER BY στοιχείων, το οποίο οδηγεί σε τελεστές blocking σε ένα σχέδιο εκτέλεσης επερώτησης.

Θεωρούμε τώρα το καθήκον του καθορισμού του κλάσματος της κίνησης της σύνδεσης του backbone που μπορεί να αποδοθεί στο δίκτυο του πελάτη. Αυτή η επερώτηση ,η Q3 , είναι ένα παράδειγμα του είδους των ad hoc συνεχόμενων επερωτήσεων που μπορούν να καταγραφούν κατά περιόδους συμφόρησης προκειμένου να καθοριστεί αν το δίκτυο του πελάτη είναι η φυσική αιτία.

```
Q3: SELECT( count(*)
    FROM    C,B
    WHERE   C.src= B.src and C.dest = B.dest
            and  C.id= B.id)/
   (SELECT count(*) FROM B)
```

Παρατηρούμε ότι το Q3 ενώνει τις ροές C,B βάση των κλειδιών τους προκειμένου να πετύχει ένα άθροισμα του αριθμού των κοινών πακέτων. Από τη στιγμή που η ένωση των δύο ροών μπορεί εν δυνάμει να χρειαστεί πολύ μεγάλη ενδιάμεση αποθήκευση(για παράδειγμα αν δεν υπάρχει όριο στην καθυστέρηση ανάμεσα σε ένα πακέτο που εμφανίζεται στις δύο συνδέσεις),ο χρήστης πιθανό να προτιμά να υπολογίσει μία περίπου σωστή απάντηση.

Το τελικό μας παράδειγμα Q4, είναι μία συνεχής επερώτηση για έλεγχο των ζευγών πηγή-προορισμός στα κορυφαία 5 στα πλαίσια της κίνησης στο δίκτυο. Για ευκολία χρησιμοποιούμε την εντολή WITH. (βλέπε SQL-99)

```
Q4:WITH Load AS
     (SELECT   src, dest, sum(len) AS traffic
      FROM      B
      GROUP       BY  src,dest)
 SELECT       src,dest,traffic
 FROM         LOAD AS L1
 WHERE        (SELECT count(*)
                 FROM   Load AS L2
                 WHERE  L2.traffic < L1.traffic)>
                (SELECT 0.95 x count (*)FROM Load)
    ORDER BY    traffic
```

3.2 Ερωτήματα πάνω από ροές δεδομένων (επιπλέον ζητήματα)

Η επεξεργασία ερωτημάτων στο μοντέλο της ροής δεδομένων φέρει διάφορα σημαντικά ζητήματα .Παρακάτω θα δούμε τα πιο ενδιαφέροντα από αυτά ,και θα περιγράψουμε διάφορες εναλλακτικές προσεγγίσεις για την επίλυση τους.

Απαιτήσεις σε μνήμη δίχως όρια

Αφού οι ροές δεδομένων είναι εν δυνάμει μη φραγμένες στο μέγεθος, η ποσότητα αποθήκευσης που χρειάζεται για τον υπολογισμό μιας ακριβούς απάντησης σε ένα ερώτημα ροής δεδομένων μπορεί επίσης να μεγαλώνει δίχως όρια. Ενώ έχουν μελετηθεί αλγόριθμοι εξωτερικής μνήμης για χειρισμό συνόλων δεδομένων μεγαλύτερων από την κύρια μνήμη, αυτοί οι αλγόριθμοι δεν ταιριάζουν επαρκώς σε εφαρμογές με ροές δεδομένων επειδή δεν υποστηρίζουν συνεχόμενες ουρές και είναι τυπικώς πολύ αργοί για πραγματικού χρόνου απαντήσεις. Το μοντέλο ροής συνεχόμενων δεδομένων είναι πιο εφαρμόσιμο σε προβλήματα όπου ο χρόνος στην απάντηση των ερωτημάτων είναι σημαντικός και υπάρχει μεγάλος όγκος δεδομένων που παράγεται συνεχώς σε υψηλή συχνότητα. Νέα δεδομένα σταθερά καταφθάνουν ακόμη και όταν τα παλιά δεδομένα βρίσκονται υπό επεξεργασία . Η ποσότητα του υπολογίσιμου χρόνου προς τα δεδομένα πρέπει να είναι χαμηλή , αλλιώς ο αλγόριθμος δεν θα μπορεί να συμβαδίσει με τη ροή των δεδομένων. Για αυτό το λόγο ενδιαφερόμαστε για αλγορίθμους που είναι ικανοί να περιορίζουν τους εαυτούς τους στη κύρια μνήμη χωρίς να έχουν πρόσβαση σε δίσκο.

Προσεγγίζοντας απαντήσεις επερωτήσεων

Όταν περιοριζόμαστε σε συγκεκριμένη ποσότητα μνήμης δεν είναι πάντα δυνατό να παράγουμε ακριβής απαντήσεις για ερωτήματα ροών πληροφορίας. Πάντως , υψηλής ποιότητας προσέγγισης απαντήσεις είναι συχνά αποδεκτές σαν ακριβείς απαντήσεις.
Προσεγγιστικοί αλγόριθμοι για προβλήματα που ορίζονται πάνω σε ροές δεδομένων έχουν γίνει καρποφόρα περιοχή για έρευνες στην κοινότητα των αλγορίθμων τα τελευταία χρόνια . Υπάρχουν κάποιες γενικές τεχνικές για απλοποίηση δεδομένων και δημιουργία σύνοψης: περιγράμματα (sketches),τυχαία δειγματοληψία , ιστογράμματα ,κύματα (wavelets).

Βασιζόμενοι σε αυτές τις τεχνικές περίληψης, έχει υπάρξει αρκετή πρόοδο στην προσέγγιση απαντήσεων σε επερωτήσεις.

Ολισθαίνοντα παράθυρα

Μία τεχνική για τη παραγωγή ενός προσεγγιστικού αλγορίθμου σε ένα ερώτημα ροής δεδομένων είναι να αποτιμηθεί το ερώτημα όχι πάνω από όλη την ιστορία των ροών δεδομένων, αλλά μόνο πάνω σε ολισθαίνοντα παράθυρα των πρόσφατων δεδομένων από τις ροές. Για παράδειγμα ,μόνο δεδομένα από την τελευταία εβδομάδα θα μπορούσαν να λαμβάνονται υπόψιν για τη παραγωγή απαντήσεων σε ερωτήματα ,με δεδομένα παλιότερα από μία εβδομάδα να απορρίπτονται.

Επιβάλλοντας ολισθαίνοντα παράθυρα σε ροές δεδομένων είναι μία φυσική μέθοδος για προσέγγιση που έχει αρκετές γοητευτικές ιδιότητες. Είναι καλά ορισμένη και εύκολα κατανοητή .Τα σεμαντικά της προσέγγισης είναι καθαρά ,έτσι ώστε οι χρήστες του συστήματος είναι σίγουρο ότι θα καταλαβαίνουν τι χρειάζεται προκειμένου να παραχθεί η προσεγγιστική απάντηση. Είναι ντετερμινιστικό, έτσι δεν υπάρχει κίνδυνος ότι ακατάλληλες τυχαίες επιλογές θα παράγουν κακή προσέγγιση. Και το πιο σημαντικό είναι ότι δίνει έμφαση σε πρόσφατα δεδομένα, τα οποία στη πλειοψηφία των εφαρμογών του πραγματικού-κόσμου είναι πιο σημαντικά και κατάλληλα από τα παλιά δεδομένα. Αν κανείς προσπαθεί σε πραγματικό χρόνο να κατανοήσει δείγματα της κίνησης σε δίκτυο , ή τηλεφωνικής κλήσης, ή εγγραφές διεκπεραίωσης ,ή σε επιστημονικό αισθητήρα ανίχνευσης, τότε σε γενικές γραμμές η γνώση που βασίζεται στο πρόσφατο παρελθόν θα είναι περισσότερο χρήσιμη από παλιότερη γνώση. Στη πραγματικότητα για αρκετές τέτοιες εφαρμογές ,τα ολισθαίνοντα παράθυρα μπορούν να ειδωθούν όχι σαν προσεγγιστική τεχνική που απρόθυμα επιβάλλεται λόγω της μη πρακτικότητας του υπολογισμού σε όλα τα προηγούμενα δεδομένα, αλλά σαν μέρος των επιθυμητών σεμαντικών που ρητώς εκφράζονται σαν μέρος της επερώτησης του χρήστη. Για παράδειγμα , οι επερωτήσεις Q3,Q4 από τη παράγραφο 2.2 , που εντοπίζουν κίνηση στο backbone δίκτυο, θα εφαρμοζόταν όχι σε όλη τη κίνηση σε όλο το χρόνο , αλλά στη κίνηση στο πρόσφατο παρελθόν.

Υπάρχει μία πληθώρα θεμάτων προς έρευνα για τη χρήση των ολισθαινόντων παραθύρων πάνω από ροές δεδομένων.

- Υπάρχει το βασικό ζήτημα για το πώς ορίζουμε "χρονοαποτυπώματα"(όπως θα δούμε και παρακάτω) πάνω στις ροές για να διευκολύνουμε τη χρήση παραθύρων.

- Η υλοποίηση επερωτήσεων ολισθαινόντων παραθύρων και ο αντίκτυπος τους στη βελτιστοποίηση επερωτήσεων είναι μια τεράστια ανέγγιχτη περιοχή.
- Στη περίπτωση που τα ολισθαίνοντα παράθυρα είναι αρκετά μεγάλα έτσι ώστε όλο το περιεχόμενο των παραθύρων δεν μπορεί να μεταφερθεί στη μνήμη, υπάρχουν επίσης θεωρητικές προκλήσεις για το σχεδιασμό αλγορίθμων που μπορούν να δώσουν βέλτιστες απαντήσεις χρησιμοποιώντας μόνο τη διαθέσιμη μνήμη.

Ενώ οι υπάρχουσες εργασίες σε προσωρινές βάσεις δεδομένων έχουν αντιμετωπίσει πολλά από τα θέματα που εμπλέκονται στις επερωτήσεις που είναι χρονικά ευαίσθητες.(μία κλάση που περιλαμβάνει επερωτήσεις ολισθαινόντων παραθύρων) σε ένα περιβάλλον σχεσιακών βάσεων δεδομένων που διαφοροποιείται στο μοντέλο υπολογισμού ροών δεδομένων εμφανίζονται νέες προκλήσεις. Η έρευνα σε προσωρινές βάσεις δεδομένων ασχολείται πρωτίστως με τη διαχείριση ολόκληρης της ιστορίας του κάθε δεδομένου στο χρόνο, ενώ σε ένα σύστημα ροών δεδομένων ασχολούμαστε πρωτίστως με την επεξεργασία νέων στοιχείων δεδομένων .Οι ακολουθιακές βάσεις δεδομένων επιχειρούν να παράγουν σχέδια επερωτήσεων που θα επιτρέπουν την πρόσβαση σε ροές ,εννοώντας πως ένα μόνο διάβασμα των δεδομένων εισόδου θα είναι επαρκές για να αξιολογηθεί το σχέδιο και η ποσότητα της μνήμης που χρειάζεται για την αξιολόγηση του σχεδίου . Αυτό το μοντέλο υποθέτει ότι το σύστημα βάσεων δεδομένων έχει τον έλεγχο πάνω σε ποια ακολουθία να επεξεργαστεί πλειάδες π.χ όταν διασπάμε πολλαπλές ακολουθίες , τις οποίες δεν μπορούμε να επεξεργαστούμε σε ένα σύστημα διαχείρισης βάσεων δεδομένων.

Επεξεργασία κατά τμήματα (batch processing), Δειγματοληψία και Σύνοψη

Μία ακόμη κλάση τεχνικών για τη παραγωγή βέλτιστων απαντήσεων είναι να δίνονται για επεξεργασία κάθε μέρα στοιχεία που καθώς καταφθάνουν ,θα υπόκεινται σε κάποιο είδος δειγματοληψίας ή τμηματικής επεξεργασίας προκειμένου να επιταχυνθεί η εκτέλεση των επερωτήσεων. Ας υποθέσουμε ότι μία επερώτηση ροής δεδομένων απαντιέται χρησιμοποιώντας μία δομή δεδομένων που θα συντηρείται σταδιακά. Η πιο γενική περιγραφή μία τέτοιας δομής δεδομένων είναι το ότι υποστηρίζει δύο λειτουργίες :update(tuple) και computeAnswer(). Η πρώτη λειτουργία υπάρχει για να ανανεώνει τη δομή δεδομένων καθώς νέα δεδομένα καταφθάνουν, και η δεύτερη λειτουργία παράγει καινούργια ή ενημερωμένα αποτελέσματα στην επερώτηση. Όταν επεξεργαζόμαστε συνεχόμενες επερωτήσεις, το καλύτερο σενάριο είναι ότι όλες μαζί οι λειτουργίες είναι γρήγορες συγκρινόμενες με την

συχνότητα άφιξης των στοιχείων στη ροή δεδομένων. Σε αυτή τη περίπτωση, δεν είναι απαραίτητες ειδικές τεχνικές προκειμένου να αντεπεξέλθουμε με τις ροές δεδομένων και να παράγουμε χρονικές απαντήσεις. Καθώς κάθε ένα στοιχείο δεδομένων καταφθάνει , είναι χρήσιμο να ανανεώνεται η δομή δεδομένων, και τότε νέα αποτελέσματα υπολογίζονται από τη δομή δεδομένων, όλα σε λιγότερο από το μέσο όρο του χρόνου άφιξης των στοιχείων δεδομένων. Αν μία ή όλες οι λειτουργίες των δομών δεδομένων είναι αργές, τότε η παραγωγή μία ακριβής απάντησης η οποία θα ανανεώνεται συνεχώς δεν είναι δυνατή.

Επεξεργασία κατά τμήματα

Το πρώτο σενάριο είναι ότι η λειτουργία update() είναι γρήγορη αλλά η computeAnswer() είναι αργή. Σε αυτή τη περίπτωση η φυσική λύση είναι να επεξεργαστούμε τα δεδομένα σε τμήματα. Αντί να παράγεται μία απάντηση που συνεχώς ενημερώνεται , τα στοιχεία των δεδομένων αποθηκεύονται καθώς καταφθάνουν, και η απάντηση στην επερώτηση υπολογίζεται με περιοδικό τρόπο όσο ο χρόνος το επιτρέπει. Η απάντηση στην επερώτηση μπορεί να θεωρηθεί βέλτιστη με την έννοια ότι δεν είναι επίκαιρη, π.χ αναπαριστά την ακριβή απάντηση σε ένα σημείο στο πρόσφατο παρελθόν αντί την ακριβή απάντηση στη παρούσα στιγμή. Αυτή η προσέγγιση βελτιστοποίησης μέσα από επεξεργασία τμημάτων είναι ελκυστική ακριβώς επειδή δεν προκαλεί καμία αμφιβολία σχετικά με την ακρίβεια της απάντησης ,θυσιάζοντας όμως την επικαιρότητα. Μία ειδική χρήση αυτής της προσέγγισης είναι όταν ένας αλγόριθμος που δεν μπορεί να χειριστεί υψηλές συχνότητες ροής δεδομένων αλλά μπορεί να αντεπεξέλθει κανονικά σε μέσες συχνότητες ροών και αποθηκεύει τις ροές υψηλών συχνοτήτων προκειμένου να τις επεξεργαστεί έπειτα, κατά τη διάρκεια χαμηλών συχνοτήτων. (Xjoin algorithm)

Δειγματοληψία

Στο δεύτερο σενάριο, η λειτουργία computeAnswer μπορεί να είναι γρήγορη ,αλλά η λειτουργία update να είναι αργή- χρειάζεται περισσότερο χρόνο από το μέσο όρο άφιξης των στοιχείων δεδομένων. Είναι ανώφελο να επιχειρήσουμε να κάνουμε χρήση όλων των δεδομένων καθώς υπολογίζουμε μία απάντηση, διότι τα δεδομένα καταφθάνουν πιο γρήγορα από ότι μπορούν να επεξεργαστούν. Απεναντίας, κάποιες πλειάδες πρέπει να υπερπηδηθούν ,έτσι ώστε η επερώτηση να αξιολογηθεί πάνω από ένα δείγμα της ροής δεδομένων αντί όλης της ροής. Πετυχαίνουμε μία βέλτιστη απάντηση μέσω δειγματοληψίας, αλλά σε κάποιες περιπτώσεις υπάρχουν όρια στην εμπιστοσύνη που μπορούμε να δείξουμε στο βαθμό του λάθους που εισάγεται

από τη διαδικασία δειγματοληψίας. Δυστυχώς, για πολλές καταστάσεις οι προσεγγίσεις που βασίζονται στη δειγματοληψία δεν μπορούν να δώσουν αξιόπιστα εχέγγυα βελτιστοποίησης. Ο σχεδιασμός αλγορίθμων βασιζόμενων σε δειγματοληψία που μπορούν να παράγουν βέλτιστες απαντήσεις που είναι αποδεκτως κοντά στην ακριβή απάντηση είναι μία σημαντική και ενεργή περιοχή έρευνας.

Σύνοψη δομών δεδομένων

Προφανώς οι δομές δεδομένων όπου και οι δυο λειτουργίες, update computeAnswer, είναι γρήγορες και είναι και οι πιο επιθυμητές. Για κλάσεις, επερωτήσεων ροής δεδομένων όπου καμία ακριβή δομή δεδομένων με τις επιθυμητές απαιτήσεις δεν υπάρχει, κανείς μπορεί συχνά να σχεδιάσει μία βέλτιστη δομή δεδομένων που διατηρεί μία μικρή σύνοψη ή περίγραμμα των δεδομένων αντί μία ακριβή αναπαράσταση , και έτσι να είναι ικανός να κρατήσει τον υπολογισμό κατά στοιχείο δεδομένων ελάχιστο. Υλοποιώντας την αναγωγή των δεδομένων μέσα από σύνοψη δομών δεδομένων σαν κάτι το εναλλακτικό από την επεξεργασία κατά τμήματα ή τη δειγματοληψία είναι μία καρποφόρα περιοχή έρευνας με συγκεκριμένη αρμοδιότητα στο μοντέλο υπολογισμού ροής δεδομένων.

Τελεστές Blocking

Ένα *blocking τελεστής επερώτησης* είναι ένας τελεστής επερώτησης που δεν μπορεί να παράγει τη πρώτη πλειάδα εξόδου του εκτός και αν έχει δει όλη την είσοδο του. Η ταξινόμηση είναι ένα παράδειγμα ενός blocking τελεστή , καθώς έχει αθροιστικούς τελεστές όπως SUM,COUNT,MIN,MAX ,AVG. Αν κανείς προχωρεί σε αξιολόγηση συνεχόμενων επερωτήσεων σε ροές χρησιμοποιώντας ένα παραδοσιακό δέντρο τελεστών επερωτήσεων, όπου οι ροές δεδομένων εισάγονται στα φύλα και οι απαντήσεις στις τελικές επερωτήσεις παράγονται στη ρίζα, τότε η ενσωμάτωση των blocking τελεστών στο δέντρο επερωτήσεων προκαλεί προβλήματα. Από τη στιγμή που οι συνεχόμενες ροές δεδομένων μπορούν να είναι απεριόριστες ,ένας blocking τελεστής που έχει μία ροή δεδομένων σαν μία από τις εισόδους του δεν θα δει ποτέ όλη την είσοδο του και έτσι δεν θα μπορεί ποτέ να παράγει έξοδο. Οι blocking τελεστές λοιπόν δεν είναι πολύ βολικοί στο μοντέλο υπολογισμού ροών δεδομένων. Καταργώντας τους blocking τελεστές θα ήταν προβληματικό, αλλά από την άλλη η ενασχόληση με αποτελεσματικό τρόπο είναι μία από τις σύγχρονες προκλήσεις στο χώρο του υπολογισμού ροών δεδομένων.

Οι BLOCKING τελεστές που είναι η ρίζα ενός δέντρου τελεστών επερώτησης είναι πιο βολικοί από blocking τελεστές που βρίσκονται στο εσωτερικό δέντρου, παράγοντας ενδιάμεσα αποτελέσματα που παρέχονται σε

άλλους τελεστές για παραπέρα επεξεργασία. Όταν έχουμε έναν blocking αθροιστικό τελεστή στη ρίζα ενός δέντρου επερώτησης, αν ο τελεστής παράγει μία μόνο τιμή ή έναν μικρό αριθμό από τιμές ,τότε οι ενημερώσεις στην απάντηση μπορούν να χαθούν καθώς παράγονται. Όταν όμως η απάντηση είναι μεγαλύτερη ,όπως όταν η απάντηση στην ερώτηση είναι μία σχέση που πρέπει να παραχθεί σε ταξινομημένη σειρά, είναι πιο πρακτικό να διατηρούμε μία δομή δεδομένων με την ενημερωμένη απάντηση , αφού θα υπάρχουν συνεχώς επαναμεταδόσεις όλης της απάντησης.

Καμία από αυτές τις δύο προσεγγίσεις δεν λειτουργεί καλά με blocking τελεστές που παράγουν ενδιάμεσα αποτελέσματα. Το βασικό πρόβλημα είναι πως τα αποτελέσματα που παράγονται από blocking τελεστές μπορούν να αλλάζουν κατά τη διάρκεια του χρόνου ωσότου όλα τα δεδομένα έχουν διαβαστεί, όποτε και οι τελεστές που καταναλώνουν αυτά τα αποτελέσματα δεν μπορούν να πάρουν αξιόπιστες αποφάσεις που να βασίζονται σε ενδιάμεσα στάδια την εκτέλεσης της επερώτησης.

Μία προσέγγιση για να αντιμετωπιστούν οι blocking τελεστές σαν εσωτερικοί κόμβοι σε ένα δέντρο επερώτησης είναι να αντικατασταθούν με non-blockings που διεκπεραιώνουν με παραπλήσιο τρόπο τα ίδια καθήκοντα. Ένα τέτοιο παράδειγμα είναι ο juggle τελεστής , ο οποίος είναι μία non-blocking εκδοχή της ταξινόμησης:Σκοπό έχει να ξαναταξινομήσει τοπικά μία ροή δεδομένων έτσι ώστε πλειάδες που έρχονται νωρίτερα στην επιθυμητή ταξινόμηση παράγονται πριν πλειάδων που έρχονται αργότερα στην ταξινόμηση, παρ' όλαυτα κάποιες πλειάδες μπορούν να παραδοθούν έξώ από τη σειρά. Ένα ενδιαφέρον ανοιχτό πρόβλημα είναι το πώς θα επεκτείνουμε αυτή τη δουλειά σε άλλους τύπους blocking τελεστών, όπως και να ποσοτικοποιήσουμε το λάθος που εισάγεται από την προσέγγιση blocking τελεστών με non-blocking.

Έχει προταθεί μία διαφορετική προσέγγιση για τελεστές blocking. Έχουν προταθεί ροές δεδομένων με διαβεβαιώσεις για το τι μπορεί ή δεν μπορεί να εμφανιστεί στο υπόλοιπο των ροών δεδομένων. Αυτές οι διαβεβαιώσεις(που καλούνται punctuations) βρίσκονται ανάμεσα σε στοιχεία δεδομένων στις ροές.

Ένα ενδιαφέρον ανοιχτό πρόβλημα είναι να τυποποιήσουμε τη σχέση μεταξύ punctuation και τις απαιτήσεις μνήμης μίας επερώτησης – π.χ μία επερώτηση που μπορεί αλλιώς να απαιτεί απεριόριστη μνήμη μπορεί να αποδειχθεί να είναι εφαρμόσιμη σε περιορισμένη μνήμη αν δίνει εχέγγυα για τη παρουσία του κατάλληλου punctuation

Επερωτήσεις που αναφέρονται σε παλιά δεδομένα

Στο μοντέλο ροής δεδομένων υπολογισμού, όταν ένα στοιχείο δεδομένων έχει τεθεί σε επεξεργασία δεν μπορούμε να το ξαναεπισκεφτούμε . Αυτός ο περιορισμός σημαίνει ότι σε ad hoc επερωτήσεις που έχουν προκύψει σαν ζήτημα μετά από κάποια δεδομένα που έχουν απορριφτεί μπορεί να είναι αδύνατο να δοθούν απαντήσεις με ακρίβεια. Μία απλή λύση σε αυτό το πρόβλημα είναι να δίνουμε εγγυήσεις ότι οι ad hoc επερωτήσεις είναι αδύνατες για επακριβείς απαντήσεις: αξιολογούνται σαν οι ροές δεδομένων να αρχίζουν στο σημείο όπου η επερώτηση τέθηκε, και κάθε παλιότερα στοιχεία ροών να μη λαμβάνονται υπόψιν (σαν προϋπόθεση για αυτή την επερώτηση). Ενώ αυτή η λύση μπορεί να εμφανίζεται σαν όχι και πολύ ικανοποιητική , μπορεί να γίνει πολύ καλά αποδεκτή από πολλές εφαρμογές.

Μία περισσότερο αμφιλεγόμενη προσέγγιση για τη διαχείριση ad hoc επερωτήσεων που αναφέρονται στο πρόσφατο παρελθόν είναι η διατήρηση περιλήψεων των ροών δεδομένων ,αυτό μπορεί να χρησιμοποιηθεί για να δίνονται κατά προσέγγιση απαντήσεις σε μελλοντικές ad hoc επερωτήσεις. Λαμβάνοντας αυτή τη προσέγγιση χρειάζεται να παρθεί μία απόφαση για το καλύτερο τρόπο να χρησιμοποιηθούν πηγές μνήμης προκειμένου να δοθούν καλές κατά προσέγγιση απαντήσεις . Το πρόβλημα είναι παρόμοιο κατά κάποιο τρόπο με προβλήματα που εμφανίζονται σε φυσικές βάσεις δεδομένων όπως η επιλογή από δεικτοδοτούμενες και επεξεργασμένες όψεις. Εδώ όμως υπάρχει μία μεγάλη διαφορά: σε ένα παραδοσιακό σύστημα βάσεων δεδομένων όταν ένας δείκτης ή μία όψη λείπει τότε μπορούμε να πάμε στη βασική σχέση, με μεγάλο όμως κόστος. Στο μοντέλο υπολογισμού ροών δεδομένων ,αν η κατάλληλη περίληψη δομής δεν υπάρχει ,τότε καμία άλλη διέξοδος δεν είναι διαθέσιμη.

Ακόμη και αν οι ad hoc επερωτήσεις είναι δηλωμένες μόνο να αναφέρονται σε μελλοντικά δεδομένα, υπάρχουν ακόμη και τότε ζητήματα για το πώς θα τα επεξεργαστούμε με το καλύτερο τρόπο. Στις εφαρμογές ροών δεδομένων ,όπου οι περισσότερες επερωτήσεις είναι μακρόβιες συνεχόμενες επερωτήσεις αντί ενός χρόνου επερωτήσεις, τα κέρδη που μπορούν να επιτευχθούν από τη βελτιστοποίηση πολλών επερωτήσεων μπορεί να είναι σε σημαντικότερο βαθμό από αυτά που είναι δυνατά σε παραδοσιακές βάσεις δεδομένων. Η παρουσία ad hoc επερωτήσεων μεταμορφώνει το πρόβλημα της εύρεσης του καλύτερου σχεδίου επερώτησης για ένα σύνολο επερωτήσεων από ένα offline πρόβλημα σε online. Οι ad hoc επερωτήσεις επίσης θέτουν το ζήτημα της προσαρμοστικότητας στα σχέδια επερωτήσεων.

Χρονοαποτυπώματα σε ροές

Προηγουμένως , τα ολισθαίνοντα παράθυρα ορίστηκαν σε σχέση με ένα χρονοαποτύπομα ή ακολουθία αριθμού γνωρισμάτων που αναπαριστούν τον χρόνο αφίξεως μίας πλειάδας. Αυτή η προσέγγιση είναι αναμφίβολα κατάλληλη για πλειάδες που έρχονται από μία μόνο ροή, αλλά είναι λιγότερο σαφές όταν επιχειρούμε να εφαρμόσουμε ολισθαίνοντα παράθυρα σε σύνθετες πλειάδες που αντλούνται από πλειάδες από πολλαπλές ροές (π.χ παράθυρο στην έξοδο ενός τελεστή ένωσης) . Ποιο θα είναι το χρονοαποτύπομα μίας πλειάδας όταν τα χρονοαποτυπώματα των πλειάδων που ενώνουμε είναι διαφορετικά; Ζητήματα χρονοαποτυπωμάτων επίσης προκύπτουν όταν ένα σύνολο από κατανεμημένες ροές ενώνονται σε μία μόνο ροή .

Στη προηγούμενη παράγραφο για τα χρονοαποτυπώματα εισάγαμε την έννοια των έμμεσων χρονοαποτυπωμάτων ,στα οποία το σύστημα προσθέτει ένα ειδικό πεδίο σε κάθε εισαγόμενη πλειάδα, και την έννοια των άμεσων χρονοαποτυπωμάτων για την οποία ένα γνώρισμα των δεδομένων καθορίζεται όπως το χρονοαποτύπωμα. Τα σαφή χρονοαποτυπώματα χρησιμοποιούνται όταν κάθε πλειάδα ανταποκρίνεται σε ένα γεγονός του πραγματικού κόσμου σε συγκεκριμένο χρόνο που εξαρτάται από τη σημαντικότητα της πλειάδας. Ενώ τα έμμεσα χρονοαποτυπώματα χρησιμοποιούνται όταν η πηγή δεδομένων δεν περιλαμβάνει ήδη τη πληροφορία χρονοαποτυπώματος, ή όταν η ακριβής στιγμή στο χρόνο που σχετίζεται με μία πλειάδα δεν είναι σημαντική ,αλλά γενικές θεωρήσεις όπως «πρόσφατες» ή «παλιές» είναι πιο σημαντικές.

Τα σαφή χρονοαποτυπώματα έχουν το μειονέκτημα ότι οι πλειάδες μπορεί να μην φθάνουν στην ίδια σειρά με τα χρονοαποτυπώματα .(πλειάδες με ύστερα χρονοαποτυπώματα μπορεί να φθάνουν πριν πλειάδες με νωρίτερα χρονοαποτυπώματα). Αυτή η έλλειψη εγγύησης στην σειρά κάνει δύσκολο να υλοποιήσουμε υπολογισμούς ολισθαινόντων παραθύρων . Αλλά ,από τη στιγμή που μία ροή –είσοδος είναι «σχεδόν ταξινομημένη» με χρονοαποτύπωμα , τότε πλειάδες εκτός σειράς μπορούν εύκολα να επιδιορθωθούν. Φαίνεται λογικό να συμπεραίνουμε ότι ακόμη και όταν σαφή χρονοαποτυπώματα χρησιμοποιούνται ,οι πλειάδες θα παραλαμβάνονται με αυστηρή αύξουσα σειρά σύμφωνα με το χρονοαποτύπωμα.

Ας δούμε τώρα το πώς θα αναθέτουμε το κατάλληλο χρονοαποτύπωμα σε πλειάδες που είναι έξοδοι δυαδικών τελεστών χρησιμοποιώντας την ένωση σαν παράδειγμα. Υπάρχουν πολλές πιθανές προσεγγίσεις που μπορούν να υλοποιηθούν. Θα συζητήσουμε δύο.

- Η πρώτη ,η οποία ταιριάζει περισσότερο με έμμεσα χρονοαποτυπώματα, είναι να μη παρέχουμε εγγυήσεις για την σειρά εξόδου των πλειάδων από ένα τελεστή ένωσης, αλλά απλώς να υποθέτουμε ότι οι πλειάδες που καταφθάνουν νωρίτερα θα περνούν από την ένωση νωρίτερα. Η

ακριβής σειρά μπορεί να εξαρτάται από την υλοποίηση και συγκεκριμένα χαρακτηριστικά του προγραμματισμού. Σε κάθε πλειάδα που παράγεται από ένα τελεστή ένωσης ανατίθεται ένα έμμεσο χρονοαποτύπωμα το οποίο αρχικοποιείται στο χρόνο που παράχθηκε από το τελεστή ένωσης. Αυτή η «καλύτερου φορτίου» προσέγγιση έχει το πλεονέκτημα ότι μεγιστοποιεί την ευελιξία της υλοποίησης. Άλλα έχει και το μειονέκτημα είναι ότι κάνει αδύνατο να επιβληθούν ακριβώς ορισμένα , ντετερμινιστικά ολισθαίνοντα παράθυρα στα αποτελέσματα των υποερωτήσεων.

- Η δεύτερη προσέγγιση ,η οποία ταιριάζει με είτε σαφή είτε έμμεσα χρονοαποτυπώματα ,είναι να έχουμε το χρήστη να ορίζει σαν μέρος της επερώτησης ποιο χρονοαποτύπωμα θα ανατεθεί σε πλειάδες που είναι αποτέλεσμα πολλαπλών ροών. Μία απλή πολιτική είναι ότι η σειρά κατά την οποία οι ροές ταξινομούνται στη FROM της επερώτησης αναπαριστά τη προτεραιότητα των ροών. Το χρονοαποτύπωμα για μία πλειάδα που είναι έξοδος ένωσης θα πρέπει να είναι το χρονοαποτύπωμα της πλειάδας ένωσης από τη ροή εισόδου που ταξινομείται πρώτη .Αυτή η προσέγγιση μπορεί να έχει αποτέλεσμα σε πολλαπλές πλειάδες με το ίδιο χρονοαποτύπωμα. Για παράδειγμα , αν η επερώτηση είναι:

```
SELECT   *
FROM     S1 [ROWS 1000 PRECEDING],
         S2 [ROWS 100 PRECEDING]
WHERE    S1.A = S2.B
```

τότε οι εξερχόμενες πλειάδες πρώτα θα ταξινομηθούν σύμφωνα με το χρονοαποτύπωμα του S1, και μετά σύμφωνα με το χρονοαποτύπωμα του S2.

Η δεύτερη , πιο αυστηρή προσέγγιση για την ανάθεση χρονοαποτυπωμάτων σε αποτελέσματα δυαδικών τελεστών μπορεί να έχει ένα μειονέκτημα από σχεδιαστική άποψη. Αν επιθυμούμε , η έξοδος από μία ένωση να ταξινομηθεί από χρονοαποτύπωμα , ο τελεστής ένωσης χρειάζεται να αποθηκεύσει τις πλειάδες εξόδου μέχρι να είναι σίγουρο ότι μελλοντικές πλειάδες δεν θα διαταράξουν τη σειρά των εξερχόμενων πλειάδων. Για παράδειγμα αν το χρονοαποτύπωμα του S1 έχει προτεραιότητα πάνω από το αντίστοιχο του S2 και μία πρόσφατη πλειάδα του S1 ενώνεται με μία του S2 ,είναι δυνατό ότι μία μελλοντική πλειάδα θα ενωθεί με μία παλιότερη S1 πλειάδα η οποία πέφτει επίσης στο συγκεκριμένο παράθυρο. Σε αυτή τη περίπτωση ,η πλειάδα ένωσης που παρήχθη δεύτερη ανήκει πριν από τη πλειάδα ένωσης που παρήχθη πρώτη. Αν οι είσοδοι στο τελεστή ένωσης δεν έχουν ολισθαίνοντα παράθυρα καθόλου, τότε ο τελεστής ένωσης δεν μπορεί ποτέ εμπιστευτικά να παράγει εξόδους με σειρά ταξινόμησης.

Όπως αναφέρθηκε και προηγουμένως ,τα ολισθαίνοντα παράθυρα υπηρετούν δύο διακριτούς σκοπούς: κάποιες φορές είναι ένα σημαντικό μέρος των σεμαντικών της επερώτησης ,και κάποιες άλλες είναι ένα σχήμα βελτιστοποίησης προκειμένου να βελτιώσουμε την επάρκεια της επερώτησης και να μειώσουμε τις ποσότητες των δεδομένων σε διαχειρίσιμο μέγεθος. Όταν το ολισθαίνων παράθυρο εξυπηρετεί περισσότερο την αύξηση επάρκειας της επεξεργασίας επερώτησης ,τότε η προσέγγιση καλύτερου φορτίου ,η οποία επιτρέπει εκτεταμένο πλάτος πάνω από τις σειρές των πλειάδων , είναι συνήθως αποδεκτή. Από την άλλη, όταν οι σειρές των πλειάδων παίζουν συγκεκριμένο ρόλο στην έννοια της επερώτησης, όπως για ολισθαίνοντα παράθυρα οριζόμενης επερώτησης , τότε η αυστηρότερη προσέγγιση μπορεί να προτιμηθεί ακόμη και υπό το κόστος της λιγότερο επαρκούς υλοποίησης. Ένα γενικού σκοπού σύστημα επεξεργασίας ροών δεδομένων πρέπει να υποστηρίζει μαζί και τους δύο τύπους ολισθαινόντων παραθύρων , και η γλώσσα επερωτήσεων θα πρέπει να επιτρέπει στο κάθε χρήστη να επιλέξει ποιος τύπος είναι ο κατάλληλος.

4.ΠΑΡΟΥΣΙΑΣΗ ΤΩΝ ΠΡΟΤΕΙΝΟΜΕΝΩΝ ΤΕΧΝΙΚΩΝ

Οι μέθοδοι που προτείνονται για κάθε ερώτημα ελέγχου των κορυφαίων k , προϋποθέτουν ότι η συνάρτηση κέρδους f είναι μονότονη σε όλα τα γνωρίσματα. Μία συνάρτηση f είναι αύξουσα στη διάσταση xi όταν για κάθε ζεύγος σημείων (p_1 , p_2) με p_1 .xi ≥ p_2 .xi και p_1 .xj = p_2 .xj για κάθε j != i , ισχύει ότι score(p_1)=f($p_1 . x_1$, $p_1 . x_2$ .... $p_1 . x_d$) ≥ score(p_2) = f($p_2 . x_1$, $p_2 . x_2$ .... $p_2 . x_d$). Ομοίως η f είναι φθίνουσα στο xi όταν για κάθε για κάθε ζεύγος σημείων (p_1 , p_2) με p_1 .xi ≥ p_2 .xi και p_1 .xj = p_2 .xj για κάθε j!=i , έχουμε ότι score(p_1)=f($p_1 . x_1$, $p_1 . x_2$ .... $p_1 . x_d$) ≤ score(p_2) = f($p_2 . x_1$, $p_2 . x_2$ .... $p_2 . x_d$). Επίσης μια συνάρτηση μπορεί να είναι αύξουσα σε κάποιες διαστάσεις και φθίνουσα στις υπόλοιπες. Χωρίς απώλεια γενίκευσης ,συγκεντρώνουμε τη προσοχή μας σε δισδιάστατους χώρους(π.χ εγγραφές με δύο γνωρίσματα που οι τιμές τους είναι μεταξύ 0 , 1) και συναρτήσεις κέρδους αύξουσες και στις δύο διαστάσεις. Παρακάτω θα δούμε ορισμένες απαιτήσεις που επιτρέπουν τον επαρκή υπολογισμό και διαχείριση των αποτελεσμάτων. Ενώ επίσης παρουσιάζεται μία μέθοδος που συναγωνίζεται αυτές που θα παρουσιάσουμε στη συνέχεια συνδυάζοντας παλιότερο υλικό προκειμένου να επεξεργαστούμε συνεχόμενα ερωτήματα κορυφαίων κ.

4.1 ΠΡΟΑΠΑΙΤΟΥΜΕΝΑ

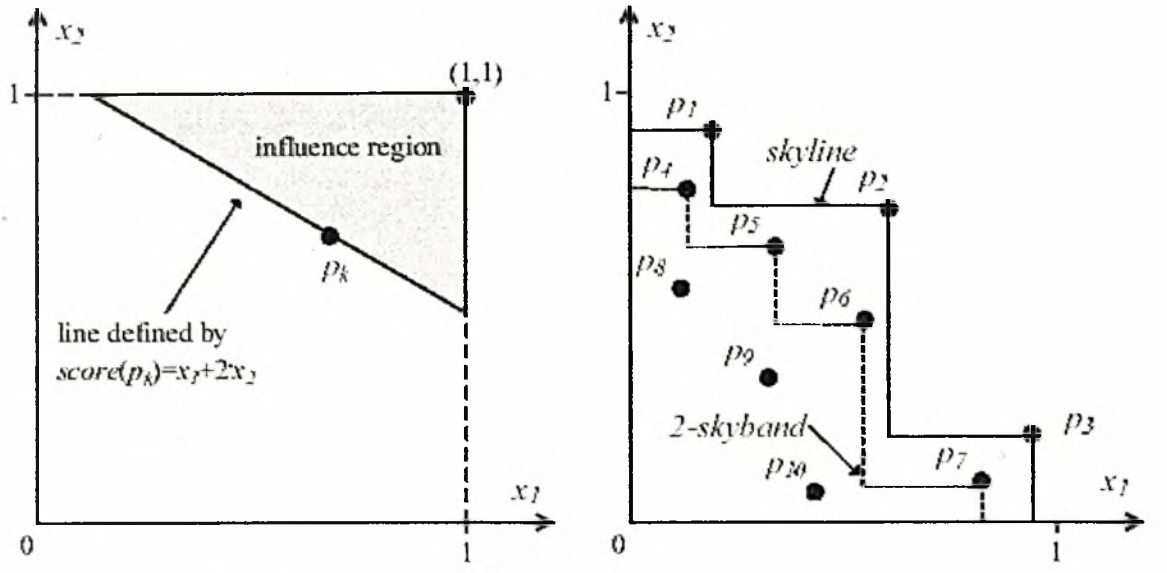


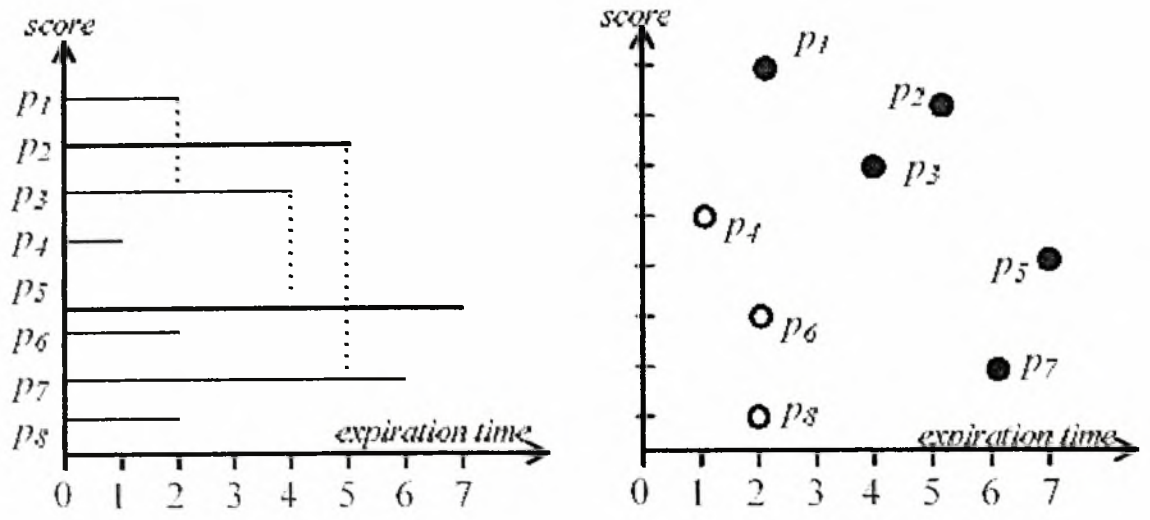
Figure 1: Geometric observations on the top-k problem

Στο πιο πάνω παράδειγμα της εικόνας 1 , το p_k είναι το αντικείμενο με το k πιο υψηλό βαθμό(για ένα ερώτημα q με συνάρτηση προτίμησης $f(x_1, x_2) = x_1 + 2 * x_2$. Η γραμμή που ορίζεται από το $score(p_k) = x_1 + 2 * x_2$ χωρίζει τον χώρο δεδομένων σε δύο μέρη. Η σκιώδης περιοχή ανταποκρίνεται στην περιοχή επιρροής του q ,με κάθε ενημέρωση που πέφτει στη περιοχή επιρροής να αλλάζει το αποτέλεσμα του ερωτήματος. Συγκεκριμένα, μία εισαγωγή θα προκαλέσει τη μετατόπιση του p_k, επομένως τη συρρίκνωση της περιοχής επιρροής. Ενώ μία διαγραφή θα προκαλέσει την επέκταση. Από την άλλη, αλλαγές έξω από τη περιοχή επιρροής είναι άσχετες με το q αφού τα αντίστοιχα αντικείμενα έχουν βαθμό κάτω από το $score(p_k)$ και δεν μεταβάλλουν το αποτέλεσμα.

Για κάθε συνάρτηση βαθμολόγησης(scoring) f (αύξουσα στα σημεία x_1, x_2) ,το σημείο με το μέγιστο βαθμό είναι στη γωνία (1,1) του χώρου δεδομένων. Σε συμφωνία με το skyline,το σημείο (1,1) κυριαρχεί επί κάθε άλλου σημείου. Ή αλλιώς , κάθε σημείο που πέφτει μέσα σε τετράγωνο R κυριαρχείται από την πάνω-δεξιά γωνία. Το score αυτού του σημείου ,ονομάζεται maxscore(R) και είναι το άνω όριο όλων των σημείων στο R.

Γενικά, το skyline ενός συνόλου δεδομένων περιέχει όλα τα σημεία που ανήκουν στο αποτέλεσμα κάποιου top-1 ερωτήματος με μονότονη συνάρτηση. Για παράδειγμα, στο σχήμα β της παραπάνω εικόνας, το skyline περιλαμβάνει τα p_1, p_2 και p_3 ,εννοώντας ότι κάθε top-1 ερώτημα (στα $p_1,\ldots p_{10}$) επιστρέφει ένα από αυτά τα τρία σημεία. Προκειμένου να γενικεύσουμε σε αυθαίρετες τιμές του k χρησιμοποιούμε τον ορισμό του k-skyband. Συγκεκριμένα το k-sky band περιέχει τα σημεία τα οποία κυριαρχούνται από άλλα k-1 σημεία. Σύμφωνα με αυτόν τον ορισμό , το skyline είναι ένα συγκεκριμένο στιγμιότυπο του skyband για k = 1. Στην εικόνα 1(b) το skyline περιλαμβάνει τα p_1, p_2 , p_3 ,εννοώντας ότι κάθε επερώτηση κορυφαίων-1 (στα $p_1,\ldots p_{10}$) επιστρέφει μία από αυτές τις τρεις πλειάδες. Προκειμένου να γενικεύσουμε σε αυθαίρετες τιμές του k, χρησιμοποιούμε την ιδέα του k-skyband. Ειδικότερα, το k-skyband περιέχει τα σημεία που κυριαρχούνται από το πολύ κ-1 άλλα σημεία. Σύμφωνα με αυτό τον ορισμό ,το skyline είναι ένα ειδικό στιγμιότυπο του skyband ,όπου k = 1. Στην εικόνα 1(b) ,το 2 – skyband αποτελείται από όλα τα σημεία ($p_1,\ldots p_7$) επάνω ή δεξιά από την διακεκομμένη γραμμή. Οι πλειάδες ($p_8,\ldots. p_{10}$) που δεν ανήκουν στο 2-skyband δεν μπορούν να είναι στο αποτέλεσμα κάποιας επερώτησης κορυφαίων-2 ,αφού πάντα κυριαρχούνται από δύο ή περισσότερα σημεία.

Τώρα ας υποθέσουμε ότι κάθε σημείο(πλειάδα) συσχετίζεται με ένα χρόνο ¨θανάτου¨ .Η εικόνα 2(α) δείχνει ένα παράδειγμα, όπου στη χρονική στιγμή t=0 υπάρχουν 8 έγκυρα σημεία. Ο οριζόντιος άξονας αναπαριστά το χρόνο ζωής των πλειάδων και ο κάθετος τους βαθμούς (score) τους σύμφωνα με μία συγκεκριμένη συνάρτηση βαθμολόγησης. Υποθέτοντας ότι δεν υπάρχουν άλλες αφίξεις ,μπορούμε να προβλέψουμε όλα τα μελλοντικά αποτελέσματα .Το σύνολο των 2 κορυφαίων τη χρονική στιγμή 0 είναι{ p_1, p_2 }.Ενώ το p_1 λήγει τη χρονική στιγμή 2 , αντικαθίσταται από το p_3. Τη χρονική στιγμή 4, το p_3 λήγει και το αποτέλεσμα γίνεται { p_2, p_5 }.Τελικά, τη χρονική στιγμή 5, το p_7 αντικαθιστά το p_2. Σημαντική παρατήρηση είναι ότι τα αντικείμενα που εμφανίζονται σε κάποιο αποτέλεσμα είναι αυτά που ανήκουν στο 2-skyband στο χώρο των βαθμός(score)-χρόνος. Αυτά τα αντικείμενα φαίνονται στην εικόνα 2β. Ας θεωρήσουμε , για παράδειγμα, μία πλειάδα p που θα ανήκει σε κάποιο μελλοντικό αποτέλεσμα κορυφαίων - 2. Τότε, υπάρχει κάποια χρονική στιγμή που το p έχει μικρότερο βαθμό (κυριαρχείται δηλαδή)από τουλάχιστον 1 άλλα έγκυρα αντικείμενα. Έτσι, το p είναι μέρος του 2-skyband. Αντιστρόφως , ας θεωρήσουμε ότι το p ανήκει στο 2-skyband στο χώρο βαθμός-χρόνος. Αυτό συνεπάγεται ότι υπάρχει το πολύ ένα αντικείμενο με βαθμό υψηλότερο από το p που λήγει μετά το p.Άρα υπάρχει κάποια χρονική στιγμή όπου το p είναι μέλος των κορυφαίων 2 αποτελεσμάτων.



(a) Record lifespans (b) 2-skyband

Figure 2: Top-2 query as 2-skyband in *score-time* space

Η αναγωγή από κ κορυφαία σε κ-skyband βρίσκει εφαρμογή και στα δύο είδη ολισθαινόντων παραθύρων και είναι ανεξάρτητη από τη διαστατικότητα d π.χ το skyband πάντα υπολογίζεται σε δισδιάστατο χώρο (βαθμό-χρόνο) ακόμη και αν οι πλειάδες έχουν περισσότερα από 2 γνωρίσματα.

4.2 Threshold Sorted List Algorithm

Τώρα θα εξετάσούμε μία εναλλακτική μέθοδο για το συνεχή έλεγχο των κορυφαίων κ ερωτημάτων ο οποίος στηρίζεται σε αλγορίθμους που περιγράψαμε προηγουμένως. Ελλείψη άλλων προτάσεων αυτή η μέθοδος θα χρησιμοποιηθεί προκειμένου να τη συγκρίνουμε με τις τεχνικές που παρουσιάζονται στη παρούσα εργασία.

Οι Τοπ-κ τεχνικές πρέπει να ενσωματώνουν δύο στοιχεία: α) Μία μέθοδο για τον αρχικό υπολογισμό του τοπ-κ συνόλου και β)έναν επαρκή μηχανισμό διαχείρισης για να ενημερώνει το αποτέλεσμα σε περίπτωση εισαγωγής/ διαγραφής .Ο ΤΑ (Threshold Algorithm) ικανοποιεί την πρώτη απαίτηση εξαιτίας της καλής του απόδοσης. Για τη δεύτερη απαίτηση θα χρησιμοποιήσουμε την Τεχνική του Yi, η οποία είναι εφαρμόσιμη σε κάθε τοπ-κ υπολογιστική μέθοδο. Ονομάζουμε αυτόν το συνδυασμό Threshold Sorted List Algorithm.

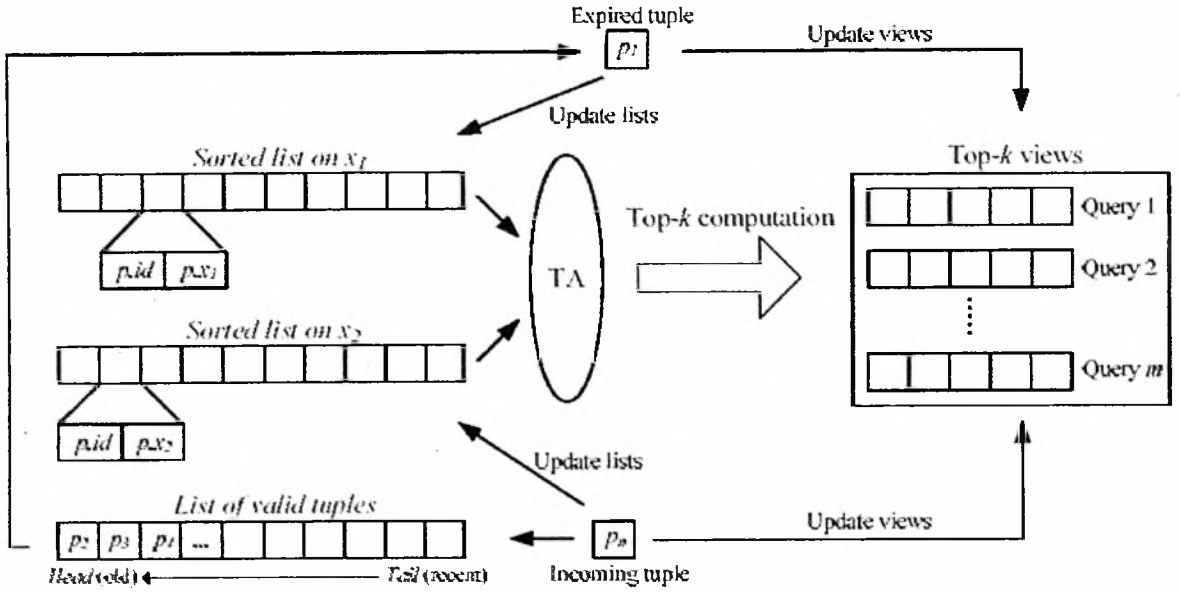


Figure 3: Threshold sorted list mechanism

Στην εικόνα 3 βλέπουμε την αρχιτεκτονική του TSL αλγόριθμου, υποθέτοντας δισδιάστατο χώρο. Όλες οι έγκυρες τιμές είναι αποθηκευμένες σε μία FIFO λίστα ,όπου κάθε p που εισέρχεται είναι ένα σύνολο <p.id,p.x_1,p.x_2,p.t>. (π.χ ένα μοναδικό id, οι τιμές των δύο γνωρισμάτων x1,x2 και ο χρόνος αφίξεως). Επιπρόσθετα ,ο TSL, διατηρεί 2 λίστες από όλα τα σημεία που υπάρχουν στις δύο διαστάσεις.

Όταν ένα καινούργιο top-k ερώτημα q φθάνει στο σύστημα ,ο TA υπολογίζει το αρχικό αποτέλεσμα, το οποίο περιέχει kmax > k καταχωρήσεις(ας θυμηθούμε εδώ από τη τεχνική του Yi, ότι η διατήρηση πιο πολλών αποτελεσμάτων από k βοηθάει στη μείωση των επαναυπολογισμών) .Ο TA έχει πρόσβαση στις δύο ταξινομημένες λίστες με κυκλικό τρόπο. Για κάθε ανακτώμενο p ,λαμβάνει χώρα τυχαία προσπέλαση στα άλλα γνωρίσματα του p, και ο βαθμός του υπολογίζεται. Αν ο βαθμός του p είναι μεγαλύτερος από το συγκεκριμένο Kmax με το καλύτερο βαθμό ,προστίθεται στο top-Kmax αποτέλεσμα. Επιπλέον, μετά από κάθε γύρο, ένα κατώφλι τ υπολογίζεται βασιζόμενο στις τελευταίες τιμές γνωρισμάτων που ειδώθηκαν κατά μήκος όλων των διατεταγμένων λιστών.(το τ είναι ο μέγιστος βαθμός που μπορεί να επιτευχθεί από κάθε σημείο που δεν έχουμε επισκεφτεί ακόμη) .Αν το κ με το μεγαλύτερο βαθμό είναι μεγαλύτερο από το τ, τότε ο αλγόριθμος τερματίζει και η όψη που περιλαμβάνει τα Kmax αποτελέσματα υλοποιείται.(βλέπε εικόνα 3)

Μετά την άφιξη ενός καινούργιου δεδομένου p_n, ο TSL ι) εισάγει το $p_n.x_1$, $p_n.x_2$ στην αντίστοιχη διατεταγμένη λίστα, και ιι) υπολογίζει το score για κάθε μία από τα m ενεργά ερωτήματα. Αν κάθε μία από τις συγκεκριμένες m

όψεις έχουν επηρεαστεί το αντίστοιχο κορυφαίο-k' σύνολο ενημερώνεται (κ' είναι ο αριθμός των εγγραφών στη συγκεκριμένη όψη με $k \leq k' \leq$ kmax).Συγκεκριμένα το p_n εισάγεται στην όψη , και αν k' = kmax το προηγούμενο στοιχείο στη θέση Kmax μετακινείται. Από την άλλη όταν ένα υπαρκτό στοιχείο π.χ p_1 αποσύρεται ο TSL ι) διαγράφει τις αντίστοιχες εγγραφές από τις δύο λίστες, και ιι) μετακινεί το p_1 από κάθε μία από τις m όψεις που το περιέχουν. Αν ο αριθμός των εγγραφών σε μία συγκεκριμένη όψη πέφτει κάτω από k, ο TA αλγόριθμος ξαναγεμίζει την όψη με Kmax εγγραφές.

4.3 Αλγόριθμος ελέγχου των κορυφαίων k (TMA)

Παρακάτω εστιάζουμε στον top-k monitoring algorithm ή αλλιώς TMA . Συγκεκριμένα στη παράγραφο 4.1 παρουσιάζουμε τις δομές ευρετηρίου και book-keeping ενώ στις παραγράφους 4.2 ,4.3 παρουσιάζουμε τα μοντέλα υπολογισμού και διαχείρισης αντίστοιχα.

Δομές ευρετηρίου και book-keeping

Ας υποθέσουμε ένα δισδιάστατο χώρο, κάθε εγγραφή p αναπαρίσταται σαν ένα σύνολο <p.id, p. x_1, p. x_2, p.t> ,όπου το p.id είναι ένα μοναδικό χαρακτηριστικό(κλειδί) , τα p. x_1 , p. x_2 είναι οι x_1, x_2 τιμές γνωρισμάτων του p, και το p.t είναι ο χρόνος άφιξης.

Παρόμοια με προϋπάρχουσες προσεγγίσεις που χειρίζονται ροές πολυδιάστατων σημείων , χρησιμοποιούμε ένα κανονικό πλέγμα για να δεικτοδοτήσουμε τις έγκυρες εγγραφές, αφού μία πιο περίπλοκη μέθοδος προσπέλασης είναι πολύ ακριβή για δυναμική διαχείριση(π.χ ένα R-δέντρο κύριας μνήμης). Η έκταση κάθε κελιού σε κάθε διάσταση είναι δ, έτσι ώστε το κελί $c_{i,j}$ στην γραμμή i και στήλη j(ξεκινώντας από την κάτω χαμηλή γωνία του χώρου εργασίας) περιλαμβάνει όλα τα στοιχεία με x_1 γνώρισμα σε ακτίνα (i * δ , (i + 1) * δ) και χ2 γνώρισμα σε ακτίνα(j * δ, (j + 1) * δ). Αντίστροφα , δοθέντος μίας εγγραφής p με γνωρίσματα (p. x_1,p. x_2) το επικαλύπτον κελί μπορεί να καθοριστεί σαν ci, j όπου i = $\lfloor p.x_1 / \delta \rfloor$ και j=$\lfloor p.x_2 / \delta \rfloor$.

Είναι σημαντικό να παρέχουμε έναν επαρκή μηχανισμό για να <<πετάμε>> ληγμένες εγγραφές. Και στις δύο εκδοχές των ολισθαινόντων παραθύρων (βασιζόμενα στη ποσότητα ,βασιζόμενα στο χρόνο) ,οι εγγραφές τίθενται σε επεξεργασία με τη μέθοδο της ουράς (FIFO) , αφού το παράθυρο W περιλαμβάνει τις πιο πρόσφατες. Όλες οι έγκυρες εγγραφές αποθηκεύονται σε μία μοναδική λίστα. Οι νέες αφίξεις τοποθετούνται στο τέλος της λίστας, και τα στοιχεία που βρίσκονται έξω από το παράθυρο , αποβάλλονται από τη κεφαλή της λίστας. Κάθε κελί περιλαμβάνει μία λίστα από δείκτες στις αντίστοιχες(έγκυρες) εγγραφές, όπως φαίνεται στην εικόνα 4. Από τη στιγμή

που οι εισαγωγές και οι διαγραφές επίσης αντιμετωπίζονται με ουρά(FIFO), κάθε λειτουργία στη λίστα δεικτών (Point list) δαπανά O(1) χρόνο.

Τα τρέχοντα ερωτήματα q αποθηκεύονται σε ένα πίνακα ερωτημάτων QT. Ο πίνακας QT διατηρεί για κάθε q ένα μοναδικό αναγνωριστή q.id, τη συνάρτηση βαθμολόγησης(scoring) q.f , τον αριθμό των εγγραφών q.k , και το αντίστοιχο αποτέλεσμα q.top_list. Ο βαθμός του κ σημείου στη λίστα q.top_list(που αναφέρεται ως q.top_score καθορίζει εμμέσως την περιοχή επιρροής του q) . Για να περιορίσουμε την εμβέλεια της διατήρησης των top-k αλγορίθμων ,κάθε κελί c σχετίζεται με μία λίστα επιρροής ILc που περιλαμβάνει μία καταχώρηση για κάθε ερώτημα q που η περιοχή επιρροής του επικαλύπτει το c. Αφού οι περιοχές επιρροής του ερωτήματος αλλάζουν δυναμικά, η λίστα ILc οργανώνεται σαν πίνακας κατακερματισμού (hash-table) στα id των ερωτημάτων για την υποστήριξη λειτουργιών γρήγορης αναζήτησης, εισαγωγής , και διαγραφής.

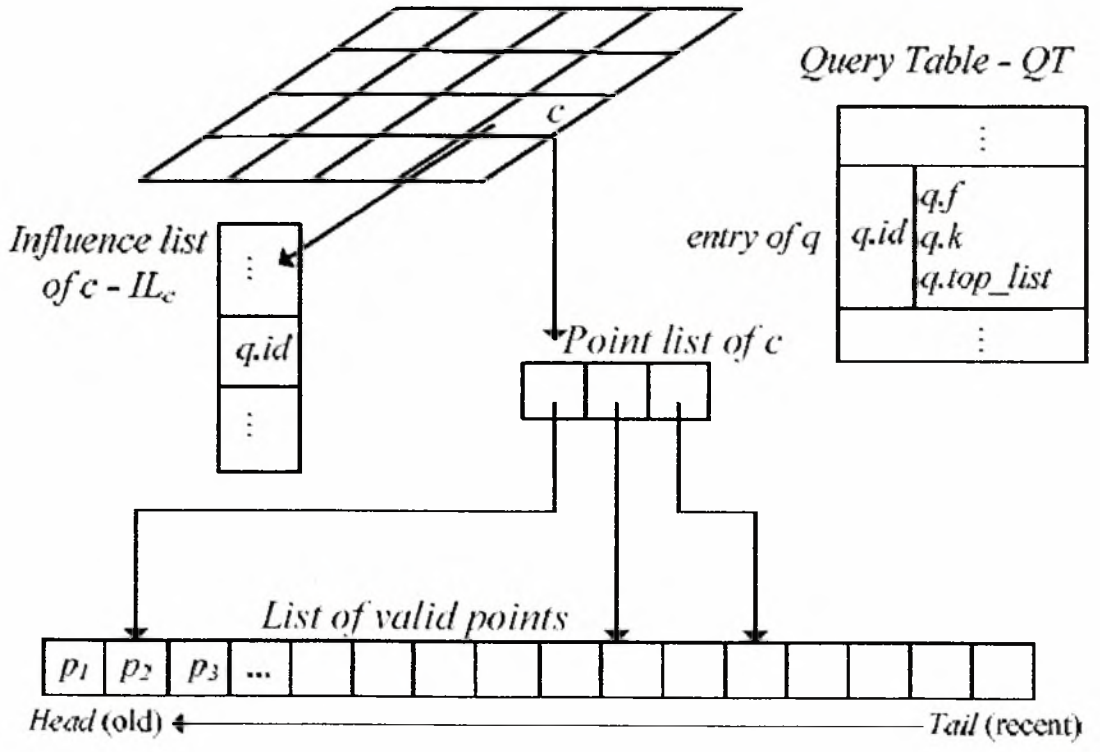


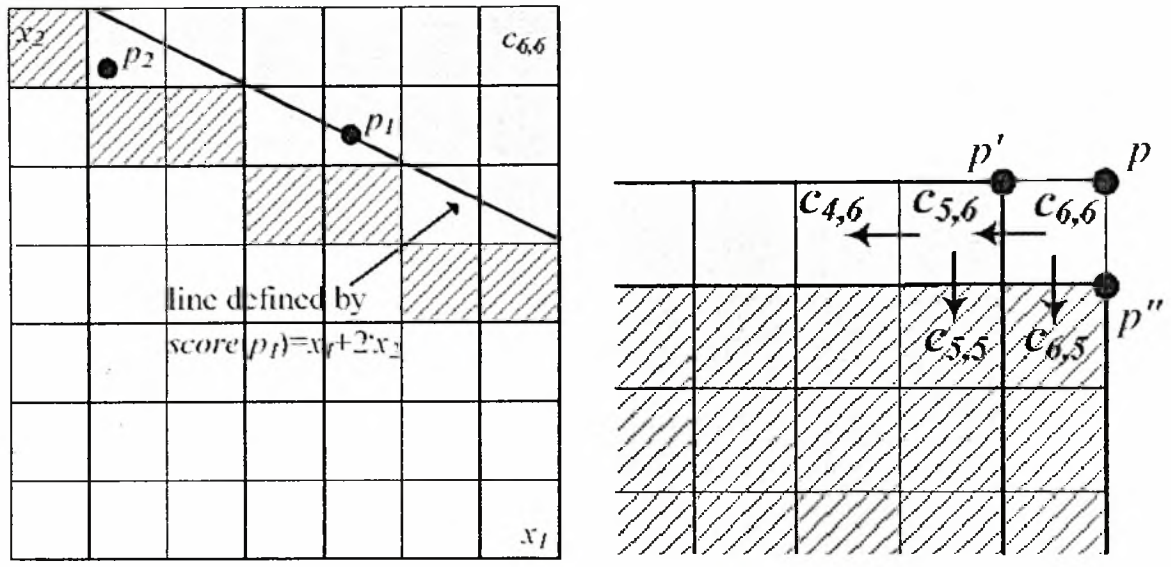
Figure 4: Index and book-keeping structures

Το μοντέλο υπολογισμού

Ένας απλός τρόπος για να επιτύχουμε την απάντηση για ένα ερώτημα q είναι να ταξινομήσουμε όλα τα κελιά c σύμφωνα με το maxscore(c) , και να τα

επεξεργαστούμε σε σειρά από το μεγαλύτερο προς το μικρότερο σύμφωνα με το maxscore(c). Για κάθε κελί που επισκεπτόμαστε ,υπολογίζουμε το score(c) για κάθε σημείο p μέσα σε αυτό ,και ενημερώνουμε το q.top_list αντίστοιχα. Η αναζήτηση σταματάει όταν το κελί c έχει maxscore(c) ≤ q.top_score (π.χ το score του κ στοιχείου στην q.top_list). Η εικόνα 5α δείχνει την επεξεργασία ενός top-1 ερωτήματος q με $f(x_1, x_2) = x_1 + 2 * x_2$ σε ένα 7×7 πλέγμα.
Ο αλγόριθμος επεξεργάζεται τα σκιώδη κελιά, συναντά δύο σημεία p_1, p_2 και επιστρέφει το p_1 σαν αποτέλεσμα. Μπορεί να αποδειχθεί εύκολα ότι είναι βέλτιστο με την έννοια να λαμβάνονται υπόψιν μόνο τα κελιά που ενδιαφέρουν την περιοχή επιρροής.
Όπως συζητήθηκε στην παράγραφο 4.1 , αυτά τα κελιά πρέπει έτσι και αλλιώς να τα επισκεφτούμε προκειμένου να αποφύγομε λανθασμένες απώλειες. Στη πράξη όμως μπορεί να είναι ιδιαίτερα ακριβό επειδή χρειάζεται να υπολογίσουμε το μέγιστο βαθμό (maxscore) για όλα τα κελιά και ακολούθως να τα ταξινομήσουμε.

Η εικόνα 5 β δείχνει πώς να καθορίσουμε την σειρά επίσκεψης δίχως να υπολογίσουμε το μέγιστο βαθμό (maxscore) για όλα τα κελιά. Αφού το σημείο (1,1) μεγιστοποιεί κάθε συνάρτηση f (αύξουσα και στις δύο διαστάσεις),το πάνω δεξιά κελί(σε αυτή τη περίπτωση το c6,6)έχει το υψηλότερο μέγιστο βαθμό και το επεξεργαζόμαστε πρώτο. Ας θεωρήσουμε τώρα τα σημεία p' , p'' (η πάνω αριστερά και κάτω δεξιά γωνία του c6,6 αντίστοιχα). Το σημείο p' έχει το υψηλότερο score από κάθε άλλο σημείο πάνω στη σκιώδη περιοχή. Με άλλα λόγια ,για όλα τα μη επεξεργασμένα κελιά ισχύει maxscore(c) ≤ max(score(p'),score(p'')). Δηλαδή , το κελί που θα επισκεφτούμε μετά το $c_{6,6}$ είναι ένα εκ των $c_{5,6}$ και $c_{6,5}$.Υποθέτοντας ότι score(p') ≥ score(p''), το $c_{5,6}$ το επεξεργαζόμαστε δεύτερο. Ομοίως, το κελί με το τρίτο μεγαλύτερο maxscore επιλέγεται μεταξύ των $c_{6,5}$, $c_{4,6}$, $c_{5,5}$



(a) Processed cells and points (b) Cell visiting order

Figure 5: A top-k computation example

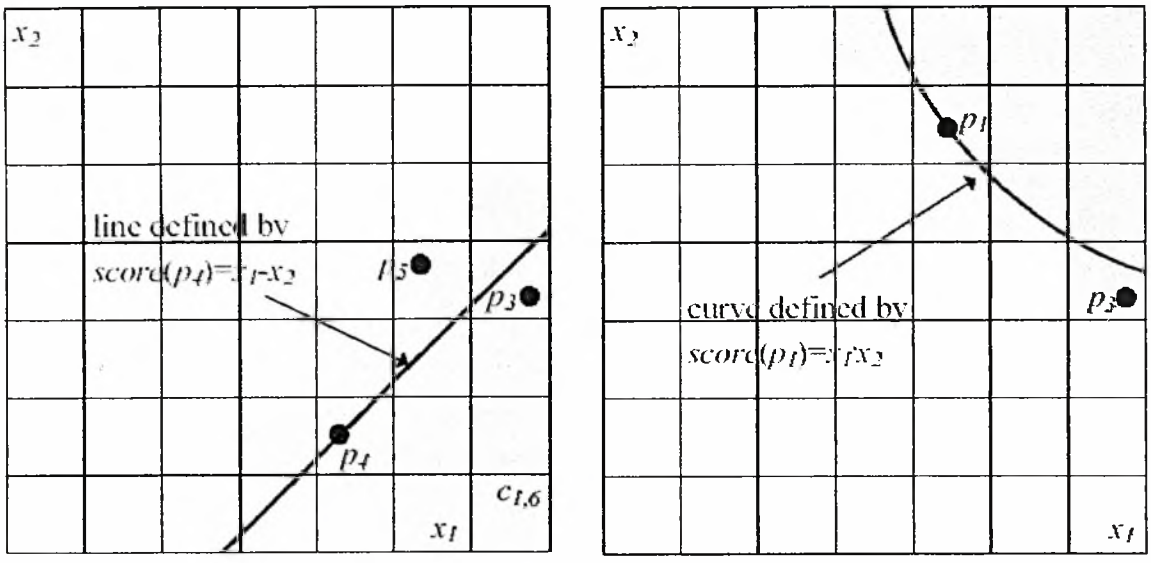
Το μοντέλο υπολογισμού των κορυφαίων k του TMA(βλέπε εικόνα 6) χρησιμοποιεί την άνωθεν μέθοδο για την επίσκεψη των κελιών με σειρά από το μεγαλύτερο προς το μικρότερο σύμφωνα με τους μέγιστους βαθμούς, διατηρώντας την ιδιότητα της επεξεργασίας του ελάχιστου συνόλου των κελιών. Αρχικά ,εισάγει σε ένα άδειο μέγιστο σωρό Η το κελί στην πάνω δεξιά γωνία με το μέγιστο βαθμό σαν το κλειδί ταξινόμησης. Μετά ξεκινάει να βγάζει από το σωρό κελιά με επαναληπτικό τρόπο. Για κάθε κελί που βγάζει εξετάζει τα σημεία μέσα και ενημερώνει το q.top_list. Έπειτα εισάγει στο σωρό τα $c_{i-1,j}$, $c_{i,j-1}$ με τους μέγιστους βαθμούς σαν τα κλειδιά ταξινόμησης, υπό τον όρο ότι δεν έχουν ξαναεισαχθεί. Μία καταχώρηση για το q εισάγεται στη λίστα επιρροής των κελιών που είναι υπό επεξεργασία για να χρησιμοποιηθεί για το χειρισμό των αφίξεων και των <<θανάτων>> .Η επεξεργασία τελειώνει όταν η επόμενη καταχώρηση στο Η έχει κλειδί χαμηλότερο ή ίσο με q.top_score. Να σημειώσουμε επίσης ότι ο αλγόριθμος μπορεί να εισάγει κελί c για το οποίο ισχύει maxscore(c) ≤ q.top_score. Τέτοια κελιά δεν τίθενται σε επεξεργασία (π.χ δεν εξάγονται από το σωρό) και έτσι μπορούν να αποβληθούν χωρίς να εισαχθούν στο σωρό Η. Συνεχίζοντας το παράδειγμα της εικόνας 5 α ο αλγόριθμος υπολογίζει το maxscore και εισάγει τα σκιώδη και ραβδωτά κελιά. Και εξάγει μόνο τα σκιώδη.

Στη πράξη υπάρχουν πολλές επερωτήσεις στο σύστημα η καθεμία με διαφορετική συνάρτηση προτίμησης και αυθαίρετη τιμή του k. Ο αλγόριθμος της εικόνας 6 μπορεί να απαντήσει κάθε επερώτηση, υπό τη προϋπόθεση ότι η αντίστοιχη συνάρτηση προτίμησης f είναι μονότονη σε κάθε άξονα.

Top-k Computation (q)
//q: top-k query
1. $q.top\_score = -\infty$; $q.top\_list = NULL$;
2. Initialize an empty max-heap H
3. Let c be the cell in the top-right corner of the workspace
4. Insert $<c, maxscore(c)>$ into H
5. While next entry has key $> q.top\_score$ and H not empty
6. De-heap the next entry $<c_{i,j}, maxscore(c_{i,j})>$ of H
7. For each point p in $c_{i,j}$
8. If $score(p) > q.top\_score$, update $q.top\_list$
9. If $c_{i-1,j}$ has not been en-heaped before
10. En-heap $<c_{i-1,j}, maxscore(c_{i-1,j})>$
11. If $c_{i,j-1}$ has not been en-heaped before
12. En-heap $<c_{i,j-1}, maxscore(c_{i,j-1})>$
13. Insert an entry for q into the influence list of $c_{i,j}$
14. End while

Figure 6: The top-k computation module

Ας θεωρήσουμε το παράδειγμα της εικόνας 7 α ,όπου f(x_1 , x_2) = x_1 - x_2 (η f είναι αύξουσα στο x_1 και φθίνουσα στο x_2).Ένας top-2 έλεγχος ξεκινάει με το κελί στη κάτω δεξιά γωνία ,και εισάγει το κελί $c_{i,j+1}$ αντί του $c_{i,j-1}$ στις γραμμές 11-12 του πιο πάνω κώδικα. Επιστρέφει τα p_3 και p_4 σαν αποτέλεσμα, μετά την επεξεργασία των κελιών που τέμνουν ή βρίσκονται κάτω από τη γραμμή score(p_4)= x_1 - x_2 . Στην εικόνα 7 β από την άλλη, φαίνεται ένα παράδειγμα top-1 υπολογισμού γραμμική συνάρτηση f(x_1 , x_2)= x_1 * x_2 , η οποία είναι αύξουσα και στους δύο άξονες. Ο αλγόριθμος επισκέπτεται τα σκιασμένα κελιά συναντά τα σημεία p_1 , p_3 και επιστρέφει το p_1 . Η περιοχή επιρροής του ερωτήματος καθορίζεται από την καμπύλη score(p_1) = x_1 * x_2 .Τέλος, η μέθοδος που παρουσιάσαμε συνήθως επεκτείνεται σε μεγαλύτερες διαστάσεις. Για παράδειγμα, σε τρισδιάστατο χώρο η μόνη αλλαγή είναι ότι μετά την επεξεργασία του κελιού $c_{i,j,w}$ τα κελιά $c_{i-1,j,w}$, $c_{i,j-1,w}$, $c_{i,j,w-1}$ εισάγονται στο σωρό H. (προφανώς εφόσον δεν έχουν ξαναεισαχθεί)



(a) $f(x_1, x_2) = x_1 - x_2$, $k = 2$ (b) $f(x_1, x_2) = x_1 \cdot x_2$, $k = 1$

Figure 7: Top-k computation for alternative functions

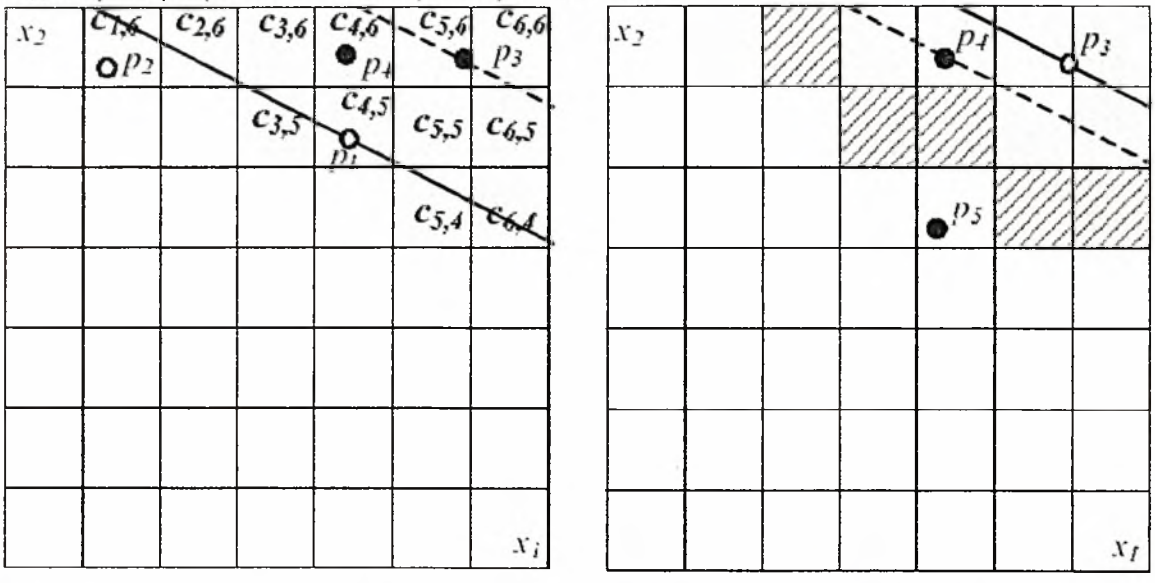
Το μοντέλο διαχείρισης

Μετά τον υπολογισμό του αρχικού αποτελέσματος, νέες πλειάδες φθάνουν στο σύστημα ενώ άλλες λήγουν .Έστω Pins είναι οι εισερχόμενες και Pdel αυτές που έχουν λήξει. Ο TMA επεξεργάζεται τις Pins πρώτα. Για κάθε p που ανήκει στις Pins αρχικά εισάγει το (ένα δείκτη στο) p στη λίστα σημείων του αντίστοιχου κελιού c. Τότε, ανιχνεύει την λίστα επιρροής ILc του κελιού c και ενημερώνει το αποτέλεσμα κάθε ερωτήματος q που ανήκει στην λίστα ILc για το οποίο ισχύει ότι score(p)>= q.top_score. Παρατηρώντας τα ληγμένα σημεία για κάθε p που ανήκει στις Pdel, ο TMA διαγράφει τα p από τα αντίστοιχο κελί c. Το σβησμένο σημείο p μπορεί να είναι μέρος του αποτελέσματος για κάποια από τα ερωτήματα στην λίστα ILc. Για κάθε ερώτημα q στην ILc ,αν το p ανήκει στην q.top_list , το q σημειώνεται σαν <<επηρεασμένο>>(affected) , εννοώντας έτσι ότι το αποτέλεσμα του πρέπει να υπολογιστεί από την αρχή όταν η επεξεργασία των Pdel ολοκληρωθεί.

Επιστρέφοντας στο παράδειγμα της εικόνας 5 α ,ας υποθέσουμε ότι οι πλειάδες p_3, p_4 φθάνουν στο σύστημα , και την ίδια στιγμή οι p_1, p_2 λήγουν ,όπως φαίνεται στην εικόνα 8α .Τα Pins ={ p_3, p_4 } τίθενται πρώτα σε επεξεργασία. Το κελί c5,6 του p_3 έχει μία καταχώρηση για το q στη λίστα επιρροής του , και έτσι το score(p_3) συγκρίνεται με το q.top_score. Από τη στιγμή που το score(p_3) > q.top_score ,το p_3 γίνεται το αποτέλεσμα του q . Ακόμη και αν το q.top_score αλλάζει μετά την είσοδο του p_3 , δεν ανανεώνουμε τις λίστες επιρροής των κελιών που δεν επηρεάζουν πλέον (π.χ

, $c_{1,6}, c_{2,6}, c_{3,6}, c_{3,5}, c_{4,5}, c_{5,5}, c_{5,4}, c_{6,4}$) .Οι λίστες επιρροής ανανεώνονται μόνο μετά από υπολογισμό των κορυφαίων κ από την αρχή. Αυτή η «περίεργη» προσέγγιση δεν επηρεάζει την ορθότητα του αλγορίθμου γιατί τυχόν προσθέσεις ή διαγραφές σε αυτά τα κελιά πολύ απλά δεν λαμβάνονται υπόψιν.

Η προσθήκη του p_4 γίνεται ομοίως, αλλά δεν επιφέρει αλλαγές επειδή το score(p_4) είναι χαμηλότερο από το καινούργιο q.top_score. Μετά, ο ΤΜΑ επεξεργάζεται το θάνατο του p_1 που πλέον δεν ανήκει στο αποτέλεσμα του q. Έτσι, εύκολα μετακινείται από τη λίστα σημείων του κελιού του. Ο αλγόριθμος διαχείρισης προχωράει με το p_2, το διαγράφει από το $c_{1,6}$ και τερματίζει. Ας σημειώσουμε ότι αν τα Pdel επεξεργαστούν πριν τα Pins , το q θα σημειωθεί ως <<επηρεασμένο>> και το αποτέλεσμα του θα πρέπει να υπολογιστεί από την αρχή(εκτός την προσθήκη του p_3) .Αυτός είναι λόγος που πρέπει πρώτα να ασχοληθούμε με τα Pins και μετά με τα Pdel.



(a) $P_{ins} = \{p_3, p_4\}$, $P_{del} = \{p_1, p_2\}$ (b) $P_{ins} = \{p_5\}$, $P_{del} = \{p_3\}$

Figure 8: Handling of updates

Ας θεωρήσουμε τώρα ότι σε επόμενη χρονική στιγμή Pins = { p_5 }, και Pdel = { p_3 },όπως φαίνεται στην εικόνα 8 β .Η επεξεργασία ξεκινάει με το p_5, το οποίο δεν δημιουργεί τυχόν αλλαγές. Από την άλλη ,η διαγραφή του p_3 ακυρώνει το συγκεκριμένο αποτέλεσμα του q. Ο ΤΜΑ θέτει σε λειτουργία το μοντέλο υπολογισμού των κ κορυφαίων (εικόνα 6) που επιστρέφει το p_4. Η νέα περιοχή επιρροής του q περιέχει όλα τα κελιά που τέμνουν ή βρίσκονται κάτω

από τη γραμμή score(p_4) = x_1 + 2 * x_2 .(διακεκομμένη γραμμή στην εικόνα 8b). Έτσι το q πρέπει να μετακινηθεί από την ILc όλων των κελιών($c_{1,6}, c_{2,6}, c_{3,6}, c_{3,5}, c_{4,5}, c_{5,5}, c_{5,4}, c_{6,4}$) που πλέον δεν επηρεάζουν το q . Η ενημέρωση των λιστών επιρροής ξεκινάει με τα κελιά που παραμένουν στο σωρό Η μετά το τερματισμό του υπολογισμού των κ κορυφαίων και συνεχίζει με ένα τρόπο όμοιο με της εικόνας 5β .Η διαφορά είναι ότι η σειρά ανανέωσης των λιστών επιρροής δεν παίζει ρόλο. Έτσι αντί για σωρό χρησιμοποιούμε λίστα που αρχικά περιέχει τα κελιά που παραμένουν στο Η. Για κάθε κελί $c_{i,j}$ στη λίστα, αν q ανήκει στην IL $c_{i,j}$ διαγράφουμε το q από το Il $c_{i,j}$ και προσθέτουμε τα κελιά $c_{i-1,j}$ $c_{i,j-1}$ στη λίστα, υπό τη προϋπόθεση ότι δεν έχουν ξαναπροστεθεί στη λίστα. Η επεξεργασία τερματίζει όταν η λίστα είναι κενή.

Εν κατακλείδι, η μόνη περίπτωση που πρέπει να γίνει ο υπολογισμός από την αρχή σε ένα ερώτημα είναι όταν κάποιες από τις κ κορυφαίες πλειάδες λήξουν και οι νέες αφίξεις έχουν χαμηλότερο score από τις εγγραφές που έχουν λήξει (έτσι ώστε η περιοχή επιρροής να εξαπλώνεται) .Όταν ένα ερώτημα q τερματίζει , το διαγράφουμε από το πίνακα ερωτημάτων ,και μετακινούμε το q από όλες τις λίστες επιρροής του πλέγματος. Το επόμενο καθήκον εκτελείται αρχικοποιώντας τη λίστα για να περιέχει ως γωνιακό κελί αυτό με το μέγιστο βαθμό.

Algorithm TMA

1. In every processing cycle do
2. P_{ins}=set of arriving points; P_{del}=set of expiring points
3. For every point p in P_{ins}
4. Insert p into the point list of the corresponding cell c
5. For each q in IL_c
6. If $score(p) \geq q.top\_score$
7. Insert p into $q.top\_list$
8. For every point p in P_{del}
9. Delete p from the point list of the corresponding cell c
10. For each q in IL_c
11. If $p \in q.top\_list$ mark q as affected
12. For each affected query q
13. Invoke Top-k Computation (q)
14. Insert the cells remaining in H into an empty $list$
15. Repeat
16. Remove next cell $c_{i,j}$ from $list$
17. If $q \in IL_{c_{i,j}}$
18. Delete q from $IL_{c_{i,j}}$
19. If $c_{i-1,j}$ is not in $list$, append $c_{i-1,j}$ to $list$
20. If $c_{i,j-1}$ is not in $list$, append $c_{i,j-1}$ to $list$
21. Until $list$ is empty
22. Report changes to the client

Figure 9: The TMA algorithm

4.4 SKYBAND MONITORING ALGORITHM (SMA)

Ο Skyband Top-k Monitoring Algorithm (SMA) εφαρμόζει την αναγωγή από top-k σε k-skyband ερωτήματα προκειμένου να αποφευχθεί υπολογισμός από την αρχή όταν κάποια αποτελέσματα λήγουν. Ας θεωρήσουμε ,για παράδειγμα, την εικόνα 10α, όπου οι εγγραφές θεωρούνται εσωτερικές ως προς το δισδιάστατο χώρο βαθμός-χρόνος. Ο αριθμός στη παρένθεση ανταποκρίνεται στο μετρητή κυριαρχίας DC κάθε καταχώρησης p π.χ ο αριθμός των καταχωρήσεων με υψηλότερο βαθμό που φθάνουν μετά το p(Ας σημειώσουμε εδώ ότι και στα δύο είδη παραθύρων η σειρά άφιξης είναι ίδια με τη σειρά λήξης). Σε χρόνο 0, το αποτέλεσμα ενός κορυφαίου -2 ερωτήματος περιέχει τα p_2, p_3 ενώ το 2-skyband περιέχει τα p_2, p_3, p_5, p_7. Σε χρόνο 3, το p_9 φθάνει και λήγει μετά από κάθε άλλη εγγραφή στο σύστημα. Ακολουθεί ότι (ι) το p_9 δεν υριαρχείται από κανένα σημείο (π.χ p_9.DC =0) και (ιι) όλα τα

σημειώσουμε εδώ ότι και στα δύο είδη παραθύρων η σειρά άφιξης είναι ίδια με τη σειρά λήξης). Σε χρόνο 0, το αποτέλεσμα ενός κορυφαίου -2 ερωτήματος περιέχει τα p_2, p_3 ενώ το 2-skyband περιέχει τα p_2, p_3, p_5, p_7. Σε χρόνο 3, το p_9 φθάνει και λήγει μετά από κάθε άλλη εγγραφή στο σύστημα. Ακολουθεί ότι (ι) το p_9 δεν υριαρχείται από κανένα σημείο (π.χ p_9.DC =0) και (ιι) όλα τα σημεία p με score(p)<=score(p_9) κυριαρχούνται από το p_9. Έτσι, οι μετρητές κυριαρχίας(DC) των p_5, p_3, p_7 αυξάνονται κατά ένα, π.χ p_5.DC=1 και p_3.DC = p_7.DC =2. Επομένως τα p_3 και p_7 μετακινούνται από το 2-skyband σε χρόνο 3. Το ανανεωμένο 2-skyband που φαίνεται στην εικόνα 10(β), περιέχει τα p_2, p_9 και p_5. Το νέο top-2 σύνολο που αποτελείται από τα δύο στοιχεία στο skyband με τους υψηλότερους βαθμούς (π.χ p_2 και p_9).Μετά τη λήξη του p_2 (στη χρονική στιγμή 5) τα κορυφαία 2 αποτελέσματα είναι { p_5, p_9 }.

Γενικά , ο έλεγχος των μελλοντικών κορυφαίων κ αποτελεσμάτων προσδιορίζεται σαν μία ενέργεια διαχείρισης κ-skyband. Ο SMA περιορίζει τη διαχείριση του skyband για ένα ερώτημα q σε σημεία που πέφτουν μέσα στη περιοχή επιρροής. Συγκεκριμένα, τα αρχικά κορυφαία κ αποτελέσματα του q (και των βαθμών τους) ανακτώνται από το μοντέλο υπολογισμού των κορυφαίων κ της εικόνας 6 και εισάγονται στο q.skyband , το οποίο περιέχει καταχωρήσεις του τύπου <p.id,p.score,p.DC> σε σειρά από το μεγαλύτερο προς το μικρότερο σύμφωνα με το p.score .Τότε ο SMA ανιχνεύει το q.skyband και για κάθε καταχώρηση p υπολογίζει το p.DC.Για να επιταχύνουμε τον υπολογισμό του μετρητή κυριαρχίας, ο χρόνος αφίξεως κάθε επεξεργασμένου στοιχείου του q.skyband είναι αποθηκευμένος σε ένα δέντρο ισορροπίας το BT που είναι ταξινομημένο σε σειρά ελάττωσης σύμφωνα με το βαθμό. Επιπλέον, το p.DC είναι απλώς ο αριθμός των καταχωρήσεων που προηγούνται του p στο BT(αφού οι εγγραφές επεξεργάζονται σε σειρά από το μεγαλύτερο προς το μικρότερο με βάση το βαθμό, αυτές οι καταχωρήσεις είναι προτιμότερες από το p υπό τους όρους ταυτόχρονα του βαθμού και του χρόνου λήξης) . Ένας εσωτερικός κόμβος στο BT περιέχει τον αριθμό στοιχείων του συνόλου του υποδέντρου που έχουν ρίζα αυτό το κόμβο έτσι ώστε ο υπολογισμός των μετρητών κυριαρχίας να χρειάζεται συνολικά O(k * logk) χρόνο. Μετά από τον υπολογισμό του μετρητή κυριαρχίας, το BD αποβάλλεται και το q.skyband περιέχει ακριβώς κ καταχωρήσεις.(q.top-score είναι ο βαθμός της k πλειάδας).

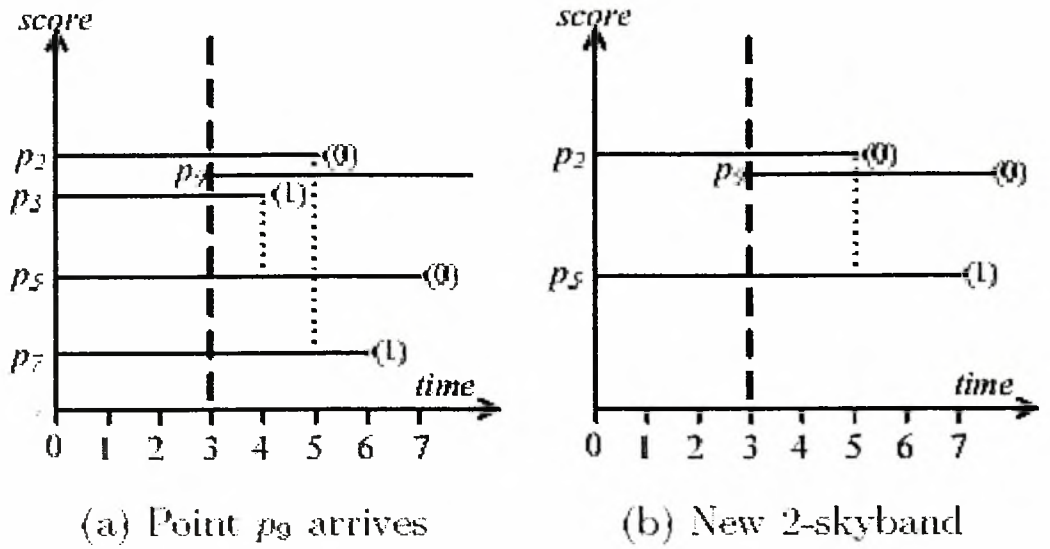


Figure 10: Skyband maintenance

Η διαδικασία διαχείρισης του skyband κρατάει μόνο τις πλειάδες με score(p) ≥ q.top_score. Όταν μία τέτοια πλειάδα φθάνει στο σύστημα εισάγεται στο q.skyband αυξάνοντας τον αριθμό των στοιχείων του συνόλου. Οι πρώτες κ εγγραφές του skyband συνιστούν το q.top_list (σε συμφωνία με την ορολογία του TMA), το οποίο δεν αποθηκεύεται επακριβώς . Ο μετρητής κυριαρχίας όλων των εγγραφών βαθμό μικρότερο από το score(p) αυξάνεται κατά 1 και οι εγγραφές που ο μετρητής τους φτάνει το κ , αποβάλλονται . Παρατηρώντας τις διαγραφές , το στοιχείο p του q.skyband με το νωρίτερο χρόνο άφιξης (π.χ αυτό που λήγει πρώτο) ανήκει στο συγκεκριμένο αποτέλεσμα .Έτσι, όταν το p λήγει ,μετακινείται και τα πρώτα κ στοιχεία του q.skyband αναφέρονται σαν η νέα q.top_list. Ας σημειώσουμε εδώ ότι το p δεν κυριαρχεί σε κανένα σημείο, και παρ' όλαυτα οι μετρητές κυριαρχίας των εναπομεινάντων στοιχείων στο q.skyband δεν επηρεάζονται. Ο αλγόριθμος SMA φαίνεται στην εικόνα 11.

Μία σημαντική περίπτωση είναι όταν το skyband περιέχει λιγότερα από κ σημεία. Αυτό το σενάριο τυγχάνει όταν κάποια από τα κορυφαία κ αποτελέσματα λήγουν και οι πρόσφατες αφίξεις δεν έχουν εισαχθεί στο skyband , γιατί το score τους είναι κάτω από το q.top_score .Σε αυτή τη περίπτωση , πρέπει να ξαναεφαρμόσουμε τον αλγόριθμο της εικόνας 6 και να υπολογίσουμε το skyband από την αρχή.

Algorithm SMA

1. In every processing cycle do
2. P_{ins} = set of arriving points
3. P_{del} = set of expiring points
4. For every point p in P_{ins}
5. Insert p into the point list of the corresponding cell c
6. For each q in IL_c
7. If $score(p) \geq q.top\_score$ //*score of the kth element after the last application of top-k computation (q)*
8. Insert p into $q.skyband$ and set $p.DC = 0$
9. For each p' in $q.skyband$ with $score(p') \leq score(p)$
10. $p'.DC = p'.DC + 1$
11. If $p'.DC = k$ evict p' from $q.skyband$;
12. For every point p in P_{del}
13. Delete p from the point list of the corresponding cell c
14. For each q in IL_c
15. If $p \in q.top\_list$
16. Delete p from $q.skyband$
17. For each query q whose skyband has changed
18. If $q.skyband$ has at least k points
19. $q.top\_list$ = the first k elements of $q.skyband$
20. Else //$q.skyband$ *has fewer than k points*
21. Invoke Top-k Computation (q)
22. Form $q.skyband$ and compute dominance counters
23. Report changes to the client

Figure 11: The SMA algorithm

Ο ψευδοκώδικας της εικόνας 11 χειρίζεται αυτή τη περίπτωση στις σειρές 20-22. Ο SMA περιμένουμε να είναι ταχύτερος από τον TMA ,αφού εμπλέκει λιγότερο συχνές κλήσεις στη μονάδα υπολογισμού των κορυφαίων κ. Για παράδειγμα στην εικόνα 8 β , ο SMA θα κρατούσε το σημείο p_4 στο skyband , και θα αναφερόταν σε αυτό σαν το αποτέλεσμα όταν έληγε το σημείο p_3.Από την άλλη, οι χωρικές απαιτήσεις του αλγόριθμου SMA είναι υψηλότερες από αυτές του TMA, αφού διαχειρίζεται το skyband (το οποίο είναι υπερσύνολο του συγκεκριμένου συνόλου κορυφαίων k) κάθε ερωτήματος. Στην επόμενη παράγραφο θα συγκρίνουμε αναλυτικά την απόδοση και τις απαιτήσεις σε χώρο και χρόνο των πιο πάνω αλγορίθμων.

4.5 ΑΝΑΛΥΣΗ ΑΠΟΔΟΣΗΣ

Με τον ίδιο τρόπο ,που γίνεται και σε άλλες εργασίες η ανάλυση, υποθέτουμε ότι (ι) η πληθικότητα του συνόλου δεδομένων σε κάθε χρονική στιγμή είναι N (ιι) οι καταχωρήσεις είναι κατανεμημένες με ανόμοιο τρόπο σε χώρο d διαστάσεων (iii) η συχνότητα ροής είναι ,κατά μέσο όρο, r πλειάδες ανά κύκλο επεξεργασίας. Αν δ είναι το μήκος της πλευράς του κάθε κελιού ,ο συνολικός αριθμός των κελιών είναι$(1/\delta)^d$,και κάθε κελί περιέχει κατά μέσο όρο $N * \delta^d$ σημεία. Πρώτα, αναλύουμε τον χρόνο του υπολογισμού των κ κορυφαίων(που εμπλέκεται και στο TMA και στο SMA). Υπενθυμίζοντας ότι ο αλγόριθμος επισκέπτεται μόνο τα κελιά που επικαλύπτουν τη περιοχή επιρροής ενός ερωτήματος. Η περιοχή επιρροής περιέχει κ(έξω από τα N)εγγραφές και , σύμφωνα με τη προσέγγιση ανομοιότητας έχει ένταση κ/ N. Έτσι, ο αριθμός των επεξεργασμένων κελιών είναι C = O([κ/ $(N * \delta^d)$]. Κάθε κελί εισάγεται σε ένα σωρό (διαγράφεται από) με λογαριθμικό κόστος, έχοντας σαν αποτέλεσμα ότι το κόστος είναι O(C * logC) για τις λειτουργίες του σωρού. Ο αριθμός των σημείων στα επεξεργασμένα κελιά είναι |C| = O(C*N* δ^d). Καθένα από αυτά τα σημεία λαμβάνεται υπόψιν για εισαγωγή στην q.top_list (ή q.skyband).Με μία υλοποίηση Ερυθρόμαυρου δέντρου, μία ενημέρωση της q.top_list κοστίζει O(logk) χρόνο (π.χ η διαγραφή των κ στοιχείων, ακολουθούμενη από την εισαγωγή ενός καινούργιου. Έτσι, ο χρόνος <<τρεξίματος>> της αναζήτησης των κορυφαίων k είναι T_{comp} = O(C*logC + |C|*logk).

Λαμβάνοντας υπόψιν το κόστος συντήρησης του αλγόριθμου TMA σε κάθε κύκλο επεξεργασίας, r νέα στοιχεία καταφθάνουν στο σύστημα ενώ r παλιά λήγουν .Έτσι , ο χρόνος ανανέωσης του πλέγματος είναι O(r). Κάθε κελί λαμβάνει r * δ^d προσθήκες r * δ^d διαγραφές. Έτσι , η περιοχή επιρροής ενός ερωτήματος κορυφαίων κ q επηρεάζεται από 2 * C * r * δ^d γεγονότα. Ο χρόνος που απαιτείται για να ελεγχθεί αν τα αντίστοιχα σημεία ανήκουν στο συγκεκριμένο αποτέλεσμα είναι O(C * r * δ^d) (συγκρίνοντας με το q.top_score). Μεταξύ τους, κ * r/N νέα σημεία λαμβάνονται για εισαγωγή στην q.top_list και κ * r/N παλιά σημεία διαγράφονται από αυτήν με κόστος O * (κ * r * log k>N) .Ας θέσουμε Pr_{rec} να είναι η πιθανότητα το ερώτημα να πρέπει να επεξεργασθεί από την αρχή μετά την ενημέρωση. Ισχύει ότι $Pr_{rec} \leq$ 1- (1- $(1-(r/N))^k$, όπου $(1-(r/N))^k$ είναι η πιθανότητα ότι κανένα από τα συγκεκριμένα κορυφαία κ στοιχεία δεν λήγουν. Ανακεφαλαιώνοντας ,ο χρόνος πολυπλοκότητας του αλγόριθμου TMA για ένα κύκλο επεξεργασίας είναι TTMA = O(r + Q (C r δd +k r log k/N + Pr_{rec} T_{comp})),όπου Q είναι ο αριθμός των τρεχόντων ερωτημάτων.

Για τον SMA ,το κόστος ενημέρωσης του index είναι το ίδιο με του TMA(π.χ O(r)). Επίσης , ο αριθμός των σημείων που καταφθάνουν (ή λήγουν)

στα κελιά και επικαλύπτουν τη περιοχή επιρροής ενός ερωτήματος q, είναι $O(C * r * \delta^d)$.Αρχικά , (μετά την εφαρμογή του υπολογιστικού μοντέλου των κορυφαίων k) το skyband περιέχει k στοιχεία. Ανάμεσα στα εισιχθέντα (ή διαγραφέντα), O(k * r/N) έχουν βαθμό μεγαλύτερο ή ίσο με το q.top_score και έχουν πρέπει να περιέχονται (ή αποκλείονται) από το skyband.Μία εισαγωγή στο q.skyband απαιτεί O(k) χρόνο, επειδή πρέπει να διατηρούμε τη σειρά σύμφωνα με το βαθμό), και ταυτόχρονα να ενημερώνουμε τους μετρητές κυριαρχίας των καταχωρήσεων με βαθμό χαμηλότερο από αυτόν της καινούργιας εγγραφής. Με τον ίδιο τρόπο, κάθε διαγραφή έχει κόστος O(k).Ας σημειώσουμε ότι για ανομοιόμορφη κατανομή δεδομένων , ο αριθμός των προσθέσεων στη περιοχή επιρροής του q είναι ίσος με τον αριθμό των διαγραφών , και το k-skyband περιέχει ακριβώς κ καταχωρήσεις. Έτσι , ο SMA δεν καταφεύγει σε υπολογισμό κορυφαίων k από την αρχή (αυτή η παρατήρηση επίσης προκύπτει και από πειράματα). Σε αυτή τη περίπτωση ο συνολικός χρόνος «τρεξίματος»είναι
cm^2 $T_{SMA} = O(r + Q * (C * r * \delta^d + k^2 * r / N))$, για κάθε κύκλο επεξεργασίας.

Τελος αναλύουμε τις απαιτήσεις σε μνήμη των προτεινόμενων μεθόδων. Το index έχει O(N * d + N + Q * C) μέγεθος , όπου O(N * d), O(N) και O(Q * C) είναι οι ποσότητες αποθήκευσης που απαιτούνται για τα N έγκυρα d-διάστατα σημεία, για N δείκτες (στις λίστες σημείων των κελιών), και για τις λίστες επιρροής των Q επερωτήσεων. Κάθε καταχώρηση σε πίνακα επερώτηση για το TMA έχει μέγεθος O(d + 2*k), για να αποθηκευτούν οι παράμετροι της συνάρτησης βαθμολόγησης και η πλειάδα <p.id ,score(p)> για κάθε σημείο p στο αποτέλεσμα. Για τον SMA κάθε καταχώρηση του QT χρειάζεται O(d + 3*k) ,αφού επιπρόσθετα στο id και στο βαθμό (score) ,το q.skyband επίσης περιέχει τους μετρητές κυριαρχίας των σημείων. Υπενθυμίζουμε εδώ ότι ο SMA δεν χρειάζεται να αποθηκευτεί επακριβώς η q.top_list ,διότι το αποτέλεσμα περιέχει τις πρώτες κ καταχωρήσεις του q.skyband. Ανακεφαλαιώνοντας, οι απαιτήσεις σε χώρο του TMA και του SMA είναι $S_{TMA} = O(N * (d + 1) + Q * (C + d + 2 * k))$ και $S_{SMA} = O(N * (d + 1) + Q * (C + d + 3 * k))$, αντίστοιχα.

Γενικά, ο SMA περιμένουμε να είναι πιο ταχύς από τον TMA γιατί ο τελευταίος καταφεύγει πιο εύκολα σε υπολογισμό από την αρχή των κορυφαίων κ. Από την άλλη το αποτέλεσμα ενημέρωσης του TMA είναι περισσότερο επαρκές από ότι διατήρηση του skyband του SMA (με πολυπλοκότητα χρόνου O(k * r * logk/N) και $O(k^2 * r/N)$ ανά ερώτημα αντιστοίχως). Έτσι αν το Pr_{rec} είναι πολύ μικρό ο TMA υπερτερεί του SMA .Όπως φαίνεται όμως και στην πειραματική αξιολόγηση, η περίπτωση είναι σπάνια. Λαμβάνοντας υπόψιν τον απαιτούμενο χώρο , ο SMA χρησιμοποιεί περισσότερο χώρο από τον TMA γιατί ι)το skyband αποθηκεύει επιπλέον πληροφορίες για τους μετρητές κυριαρχίας .ιι)Στην πράξη το κ-skyband μπορεί να περιέχει περισσότερες από κ καταχωρήσεις .Η απόδοση και των δύο

αλγορίθμων εξαρτάται από το μέγεθος της πλευράς του κελιού, δ. Μεγάλα κελιά ελαχιστοποιούν το χρόνο που δαπανάται σε λειτουργίες του σωρού, αλλά οδηγούν σε μη αναγκαία επεξεργασία των σημείων που είναι έξω από τη περιοχή επιρροής. Μεγάλο μέγεθος δ επίσης συνεπάγεται χαμηλή κατανάλωση χώρου, γιατί τα ερωτήματα επηρεάζονται από λιγότερα κελιά, και οι λίστες επιρροής των κελιών χρειάζονται λιγότερη μνήμη. Ο χρόνος εκτέλεσης των προτεινόμενων τεχνικών αυξάνεται με τα k, Q , N και r.Το ίδιο ισχύει για την επέκταση του χώρου με εξαίρεση το r.

4.6 ΑΛΛΑ ΕΙΔΗ ΕΡΩΤΗΜΑΤΩΝ ΚΑΙ ΜΟΝΤΕΛΑ ΡΟΩΝ

Σε αυτήν τη παράγραφο θα συζητηθεί η επέκταση των προηγούμενων αλγορίθμων σε ειδικές περιπτώσεις του ελέγχου των κορυφαίων κ , όπως και η υιοθέτηση τους σε άλλο μοντέλο ροής δεδομένων. Ένα δεσμευμένο κορυφαίο κ ερώτημα q ελέγχει μόνο σημεία που πέφτουν στον υποχώρο, που ορίζεται από ένα σύνολο περιορισμών εισόδου. Τυπικά , κάθε περιορισμός διαμορφώνει ένα υπέρ-τετράγωνο(που αναφέρεται σαν περιοχή περιορισμού) στον d διαστάσεων χώρο. Η εικόνα 12 δείχνει ένα παράδειγμα ενός τοπ-1 ερωτήματος με $f(x_1, x_2)= x_1+2 * x_2$,όπου η περιοχή περιορισμού είναι ένα τετράγωνο R παράλληλο στους άξονες x_1, x_2 . Για τον υπολογισμό του αρχικού αποτελέσματος του q, το μοντέλο υπολογισμού των κορυφαίων κ ξεκινά με το κελί c5,5 που μεγιστοποιεί την f στο R. Μετά , συνεχίζει με το $c_{4,5}$ που συναντά το p_1 . Από τη στιγμή που το p_1 δεν βρίσκεται στην R αποκλείεται από τη μελέτη. Τελικά το p_2 ,επιστρέφει σαν το αποτέλεσμα. Τα κελιά που έχουμε επισκεφτεί(που είναι σκιασμένα) λαμβάνουν μία καταχώρηση για την επερώτηση q στις λίστες επιρροής τους. Μεταξύ των σημείων εισαγωγής και διαγραφής σε αυτά τα κελιά, μόνο αυτά που πέφτουν στην R τίθενται σε επεξεργασία από τον αλγόριθμο διατήρησης.

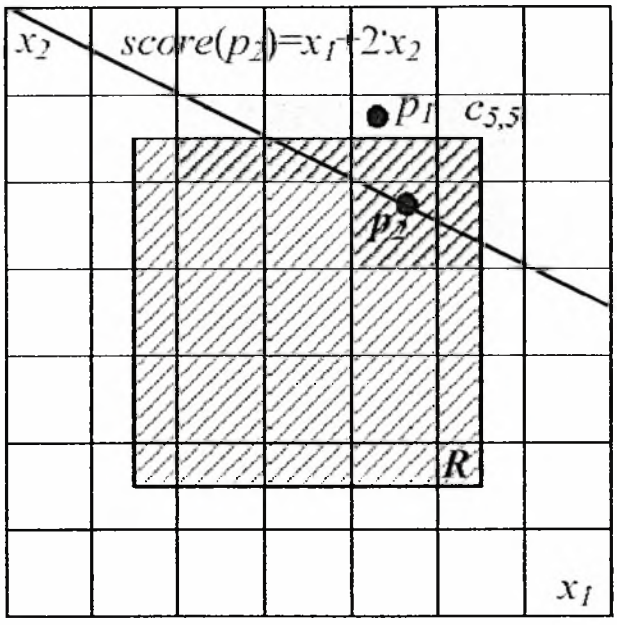


Figure 12: A constrained top-1 query example

Ένας ακόμη ενδιαφέρων τύπος ανάκτησης βασισμένος στη προτίμηση αφορά ερωτήματα που απαιτούν τον έλεγχο όλων των σημείων με βαθμό κάτω από ένα προσδιορισμένο από το χρήστη κατώφλι. Μία απλή , μα ακριβή μέθοδος είναι να ελέγξουμε κάθε καταχώρηση που εισέρχεται ή διαγράφεται έναντι όλων των ερωτημάτων. Από την άλλη το προτιμώμενο πλαίσιο εργασίας, παρέχει μία επαρκή μέθοδο επεξεργασίας. Δοθέντος ενός καινούργιου ερωτήματος κατωφλίου q ,ο υπολογισμός του αρχικού αποτελέσματος ξεκινάει με το κελί στη πάνω δεξιά γωνία του χώρου εργασίας και συνεχίζει με τα γειτονικά κελιά ,για το οποία ισχύει ότι το μέγιστο score τους είναι μεγαλύτερο από το κατώφλι. Αφού η σειρά επίσκεψης δεν είναι σημαντική , η αναζήτηση μπορεί να υλοποιηθεί με μία λίστα(αντί ενός σωρού) με τον ίδιο τρόπο που παρουσιάσαμε στη παράγραφο 4.3. Μία καταχώρηση για το q εισάγεται στη λίστα επιρροής του κάθε επεξεργασμένου κελιού. Η μονάδα συντήρησης απλά αναφέρει τις εισαγωγές και διαγραφές σημείου σε αυτά τα κελιά για σημεία p που ο βαθμός τους είναι μεγαλύτερος από το κατώφλι.

Μέχρι στιγμής έχουμε υποθέσει το append-only μοντέλου ροής δεδομένων. Στη περίπτωση ροών που περιλαμβάνουν σαφείς διαγραφές, τα δεδομένα δεν λήγουν πλέον σαν ουρά (FIFO).Έτσι ,δεν διατηρούμε τη λίστα έγκυρων δεδομένων που περιγράφεται στην παράγραφο 4.3.

Σε αντίθεση , όταν μία καινούργιο πλειάδα p φθάνει στο σύστημα απευθείας τοποθετείται στο κελί του στο πλέγμα. Όταν μία ανανέωση από διαγραφή λαμβάνει χώρα για το p,διαγράφεται από το αντίστοιχο κελί. Οι λίστες σημείων των κελιών υλοποιούνται σαν πίνακες κατακερματισμού (hashtables) για την υποστήριξη τυχαίων προσθηκών και διαγραφών σε σταθερό

αναμενόμενο χρόνο. Ο TMA εφαρμόζεται απευθείας σε αυτό το σενάριο ,όταν κάποια από τα κορυφαία κ σημεία ενός ερωτήματος q διαγράφονται, τότε το αποτέλεσμα υπολογίζεται από την αρχή .Από την άλλη ο υπολογισμός του skyband και η διαχείριση του SMA δεν είναι δυνατή γιατί η σειρά λήξης των πλειάδων δεν είναι γνωστή κατά τη πρόοδο.

4.7 ΑΞΙΟΛΟΓΗΣΗ ΜΕ ΠΕΙΡΑΜΑ

Σε αυτή τη παράγραφο αξιολογούμε πειραματικά τις προτεινόμενες μεθόδους χρησιμοποιώντας ροές ανεξάρτητων (IND) και ασυσχέτιστων (ANT) δεδομένων που διανέμονται σε χώρο δεδομένων διαστάσεων d που κυμαίνονται από 2 εώς 6. Για IND δεδομένα , οι τιμές των γνωρισμάτων κάθε καταχώρησης δημιουργούνται ανεξάρτητα, ακολουθώντας ομοιόμορφη κατανομή. Τα ANT δεδομένα αναπαριστούν ένα περιβάλλον όπου τα σημεία που έχουν μεγάλες τιμές στη μία διάσταση, έχουν μικρές τιμές σε μία ή όλες από τις υπόλοιπες διαστάσεις. Τα IND και ANT είναι κοινά σημεία αναφοράς μετρήσεων για ερωτήματα βασισμένα στη προτίμηση. Η εικόνα 13 δείχνει δύο παραδείγματα συνόλων δεδομένων.

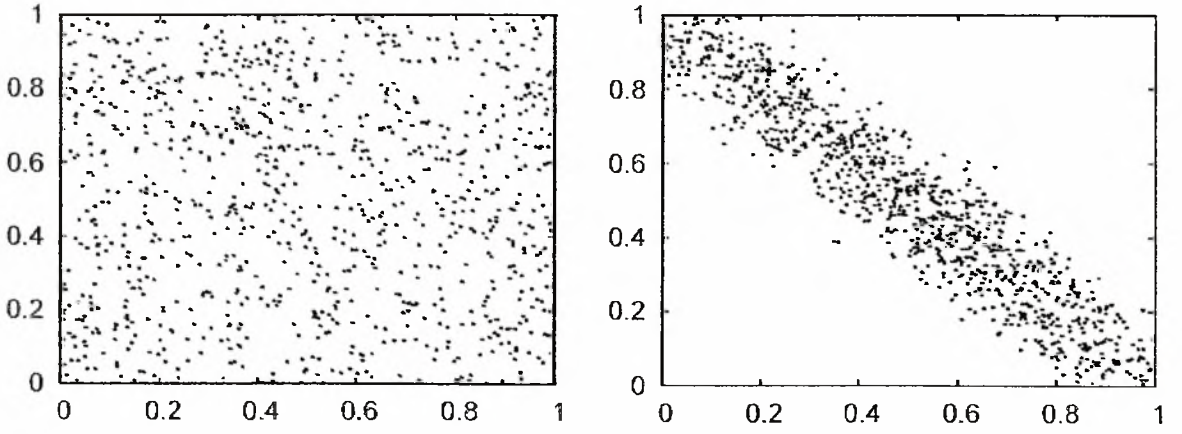


Figure 13: Datasets ($d = 2$)

Θεωρούμε ένα βασιζόμενο στη ποσότητα παράθυρο με μέγεθος N ανάμεσα σε 1 και 5 εκατομμύρια πλειάδων. Κατά τη διάρκεια κάθε χρονικής στιγμής , r καινούργια σημεία φθάνουν στο σύστημα . Δημιουργούμε Q κορυφαία κ ερωτήματα ελέγχου με συνάρτηση βαθμολόγησης από τον τύπο f(p)= $\sum_{i=1}^{d} a_i * p.x_i$ όπου οι a_i συντελεστές επιλέγονται τυχαία ανάμεσα από τα 0 ,1.

Το μήκος της προσομοίωσης είναι 100 χρονοαποτυπώματα (timestamps). Συγκρίνουμε την απόδοση των τριών αλγορίθμων ι) Threshold sorted list (TSL) ii) Top –k Monitoring Algorithm (TMA) iii)Skyband Monitoring Algorithm (SMA).

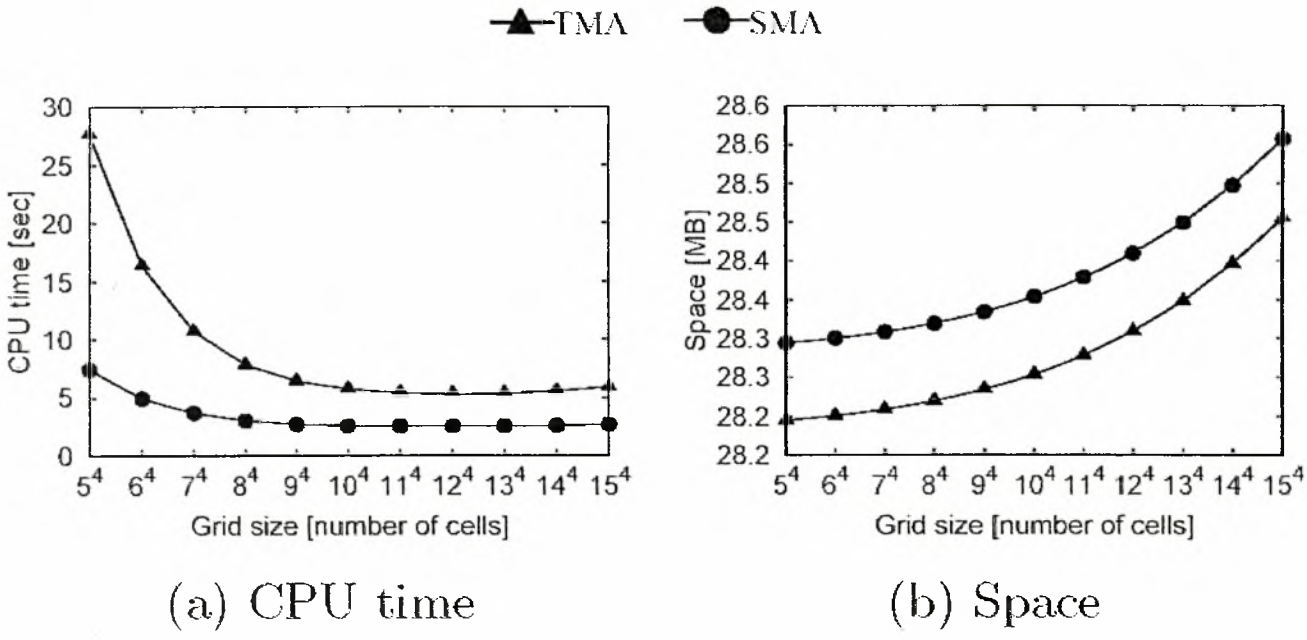
Ο πίνακας 1 περιέχει τις παραμέτρους που είναι υπό εξερεύνηση, μαζί με τις αποστάσεις τους και τις προκαθορισμένες τιμές τους. Σε κάθε πείραμα αλλάζουμε μία μόνο παράμετρο ,ενώ θέτουμε στις άλλες τις προκαθορισμένες τιμές τους.

Για όλες τις προσομοιώσεις χρησιμοποιούμε ένα Pentium 3.2GHz CPU με 1 GByte μνήμη. Πρώτα , μελετάμε την επιρροή τη κοκκώδους μορφής του πλέγματος στον TMA και SMA για IND και τις προκαθορισμένες τιμές τους (d = 4,N = 1M, r = 10K,Q = 1K , k = 20).

Table 1: System parameters

Parameter	Default	Range
Data dimensionality (d)	4	2,3,4,5,6
Data cardinality (N)	1M	1,2,3,4,5 (M)
Arrival rate (r)	10K	1,5,10,50,100 (K)
Query cardinality (Q)	1K	100,500,1K,2K,5K
Result cardinality (k)	20	1,5,10,20,50,100

Κάθε άξονας διαιρείται σε έναν αριθμό ίσων διαστημάτων που ποικίλουν μεταξύ 5 και 15(αντίστοιχο με ένα κελί από 5^4 σε 15^4 κελιά).Η εικόνα 14 δείχνει τον ολικό χρόνο τρεξίματος των μεθόδων καθώς και τις χωρικές απαιτήσεις τους. Ένα πλέγμα 12^4 κελιών είναι η καλύτερη επιλογή υπό όρους που έχουν να κάνουν με την ταχύτητα και για τους δύο αλγορίθμους. Ένα πυκνό πλέγμα είναι πιο δαπανηρό εξαιτίας των λειτουργιών του σωρού στα κελιά(κάποια από τα οποία είναι κενά), από την άλλη ένα αραιό πλέγμα οδηγεί σε μη αναγκαία επεξεργασία σημείων έξω από την περιοχή επιρροής του ερωτήματος .Παρατηρώντας τα <<αποτυπώματα>> της μνήμης , ένα πυκνό πλέγμα καταλήγει σε μεγαλύτερη δαπάνη χώρου, κυρίως εξαιτίας του book-keeping.Τα διαγράμματα για ANT ακολουθούν την ίδια γενική κατεύθυνση και παραβλέπονται.



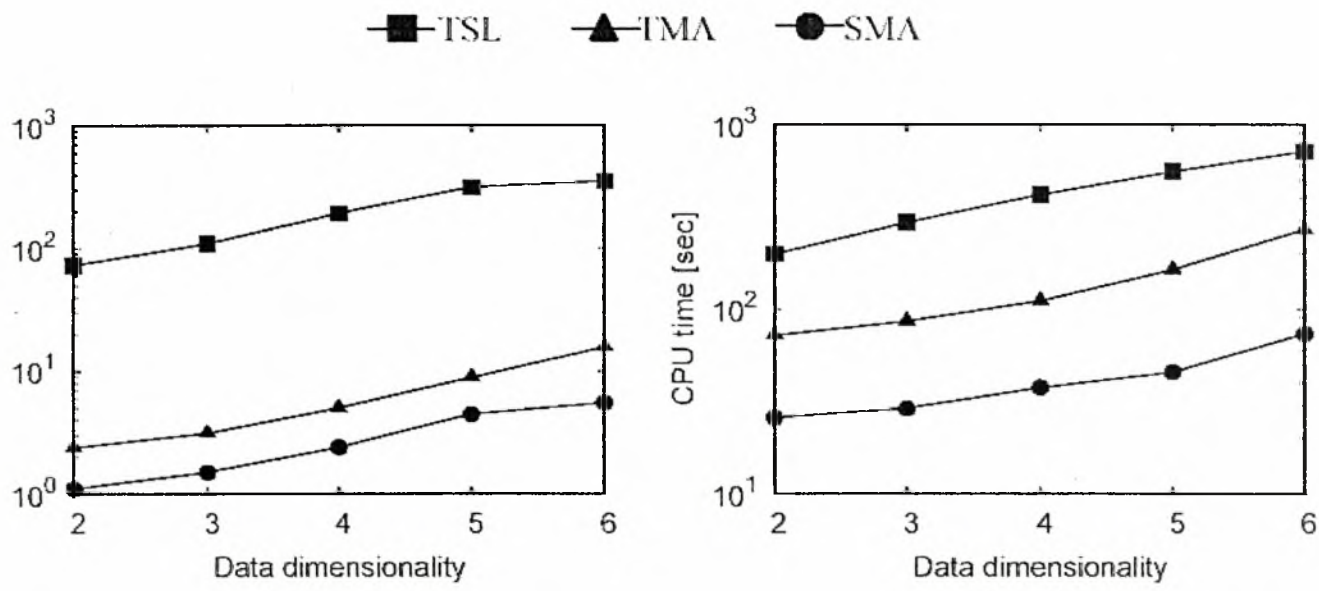
(a) CPU time (b) Space

Figure 14: Performance vs. grid granularity (IND)

Για να είναι δίκαιο, επίσης εναρμονίζουμε τη τιμή του kmax στον αλγόριθμο TSL για τις διάφορες τιμές του k. Χρησιμοποιώντας τις προκαθορισμένες τιμές και το σύνολο δεδομένων του IND , αναγνωρίζουμε τις βέλτιστες τιμές (4,10,20,30,70,120) για το kmax , που ανταποκρίνονται στις τιμές (1,5,10,20,50,100) του k.

Στην εικόνα 15 θέτουμε όπου N = 1M και r = 10K, και μετράμε τον τρέχοντα χρόνο για δεδομένα διαφορετικών διαστάσεων d (που κυμαίνεται από 2 εώς 6).Η έκταση του κελιού επιλέγεται έτσι ώστε το πλέγμα να περιέχει περίπου 12^4 κελιά, επειδή όπως φαίνεται και στην εικόνα 14, αυτή η τιμή παράγει τη καλύτερη απόδοση με λογικές απαιτήσεις σε χώρο. Το κόστος όλων των αλγορίθμων αυξάνεται με το d.Για τους SMA ,TMA αυτό συμβαίνει επειδή περισσότερα κελιά τίθενται σε επεξεργασία κατά τη διάρκεια του υπολογισμού των κορυφαίων κ από την αρχή(μετά την απομάκρυνση από το σωρό ενός κελιού , εισάγουμε στο σωρό d γειτονικά). Για τον TSL αυτό συμβαίνει γιατί ο αριθμός των λιστών και το κόστος των TA υπολογισμών είναι ανάλογα του d .Το σημαντικό κέρδος που προσφέρει ο TMA σε σχέση με το TSL δείχνει τα πλεονεκτήματα του τοπ-κ υπολογισμού συγκρινόμενα με τον αλγόριθμο TA(ας σημειώσουμε ότι ο TMA διατηρεί ακριβώς κ αποτελέσματα και έτσι εκτελεί περισσότερους τοπ-κ υπολογισμούς από τον

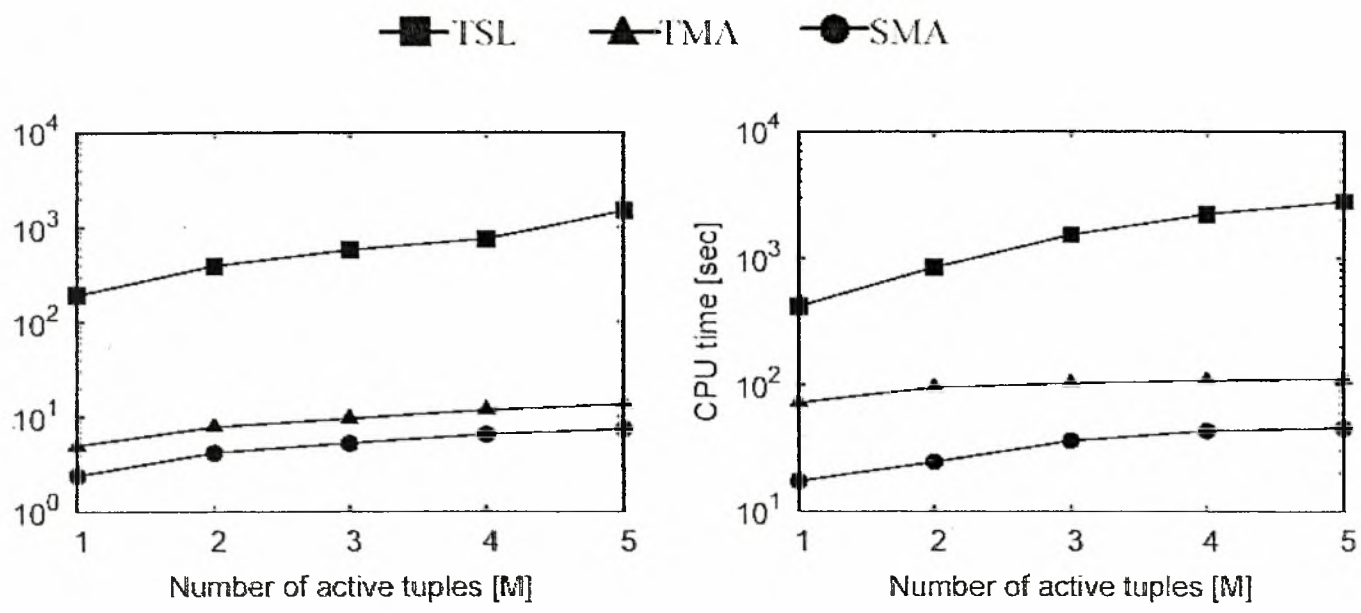
TSL).Το πλεονέκτημα του SMA επί του TMA , έχει να κάνει με τους λιγότερο συχνούς επαναυπολογισμούς (των κορυφαίων –k αποτελεσμάτων) εξ' αρχής.



(a) IND (b) ANT

Figure 15: CPU time vs. d

Στην εικόνα 16 το N κυμαίνεται από 1M έως 5M, και θέτουμε την συχνότητα αφίξεων r σε N/100 πλειάδες ανά χρονική στιγμή. Με άλλα λόγια κατά τη διάρκεια κάθε χρονικής στιγμή το 1% των δεδομένων αντικαθίσταται με νέα. Όπως είναι αναμενόμενο η απόδοση όλων των αλγορίθμων αλλάζει με το N. Πάντως, και οι δύο οι μέθοδοι είναι καλύτερες από το TSL , και είναι πολύ πιο γρήγορες στις περισσότερες περιπτώσεις. Μία ενδιαφέρουσα παρατήρηση , που προκύπτει σε όλα τα πειράματα , είναι ότι το κόστος αυξάνει για ασυσχέτιστα δεδομένα. Αυτό συμβαίνει γιατί για ANT, τα δεδομένα συγκεντρώνονται κοντά στο επίπεδο που περνάει από τα σημεία (0.5,0.5,0.5,0.5) και είναι κάθετο στη γραμμή που διασχίζει τα (0,0,0,0), και (1,1,1,1).Έτσι, ο υπολογισμός των τοπ-κ(που ξεκινάει από τη γωνία πάνω από το χώρο δεδομένων) πρέπει να επεξεργαστεί πολλά κελιά προτού ανακτήσει τα κ αποτελέσματα.

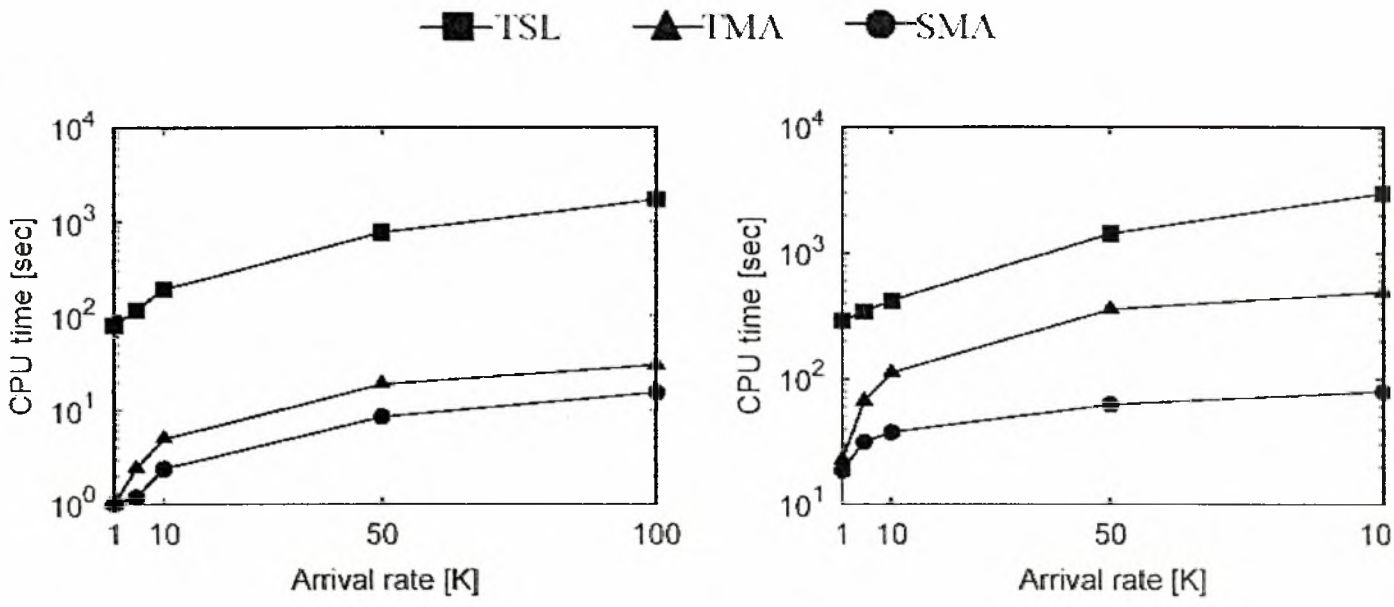


(a) IND (b) ANT

Figure 16: CPU time vs. N ($r = N/100$)

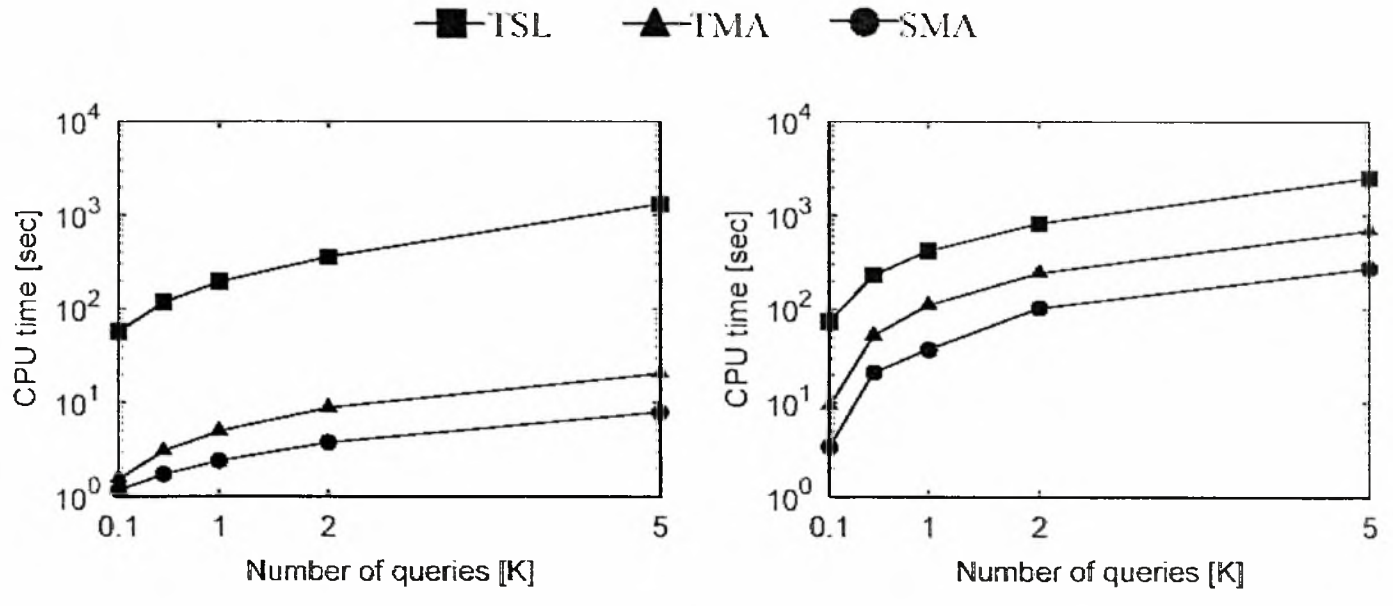
Έν συνεχεία, θέτουμε όπου N = 1M, και κυμαίνουμε το r από 1K έως 100K (0.1% έως 10% των έγκυρων πλειάδων αντικαθίστανται ανά χρονική στιγμή). Όπως φαίνεται στην εικόνα 17, το κόστος του TMA, και SMA αυξάνει με το r, επαληθεύοντας την ανάλυση σε προηγούμενο εδάφιο. Για όλες τις τιμές, και οι δύο οι αλγόριθμοι ελέγχου υπερτερούν του TSL ,δείχνοντας καλύτερη ανθεκτικότητα στις συχνές ενημερώσεις. Αυτό συμβαίνει εξαιτίας του κόστους ενημέρωσης του TSL που εμπλέκει την απευθείας διαχείριση των d ταξινομημένων λιστών. Επιπλέον ,ο SMA αποδίδει καλύτερα από τον TMA για ασυσχέτιστα δεδομένα , αφού το κόστος των συχνών τοπ-κ υπολογισμών είναι υψηλότερο(όπως εξηγείται στην εικόνα 16).Η εικόνα 18 μετρά το αποτέλεσμα του αριθμού του συνόλου των επερωτήσεων ,που κυμαίνεται από 100 έως 5 K. Ο τρέχοντας χρόνος από όλες τις μεθόδους αναπτύσσεται γραμμικά με το Q,ενώ η σχετική απόδοση τους είναι η ίδια με προηγούμενα πειράματα.

Στην εικόνα 19, φαίνεται ο χρόνος εκτέλεσης έναντι του k για IND και ANT. Οι περιοχές επιρροής των ερωτημάτων ,και επομένως και ο αριθμός των επεξεργασμένων κελιών μεγαλώνει με το κ, συνεπάγοντας υψηλότερο υπολογισμό αποτελέσματος και γενική διαχείριση.



(a) IND (b) ANT

Figure 17: CPU time vs. r



(a) IND (b) ANT

Figure 18: CPU time vs. Q

Αρχικά, τα κόστη των TMA και SMA είναι όμοια, αλλά το χάσμα στην απόδοση τους μεγαλώνει με το k. Αυτό συμβαίνει επειδή υψηλές τιμές του κ αυξάνει τη πιθανότητα Pr_{rec} που κάποια από τα συγκεκριμένες τοπ-κ σημεία λήγουν. Για κ=100 και ANT ,το κόστος του TMA είναι σχεδόν τόσο υψηλό όσο αυτό του TSL, εξαιτίας των πολύ συχνών και ακριβών επανυπολογισμών των κορυφαίων κ αποτελεσμάτων από την αρχή.

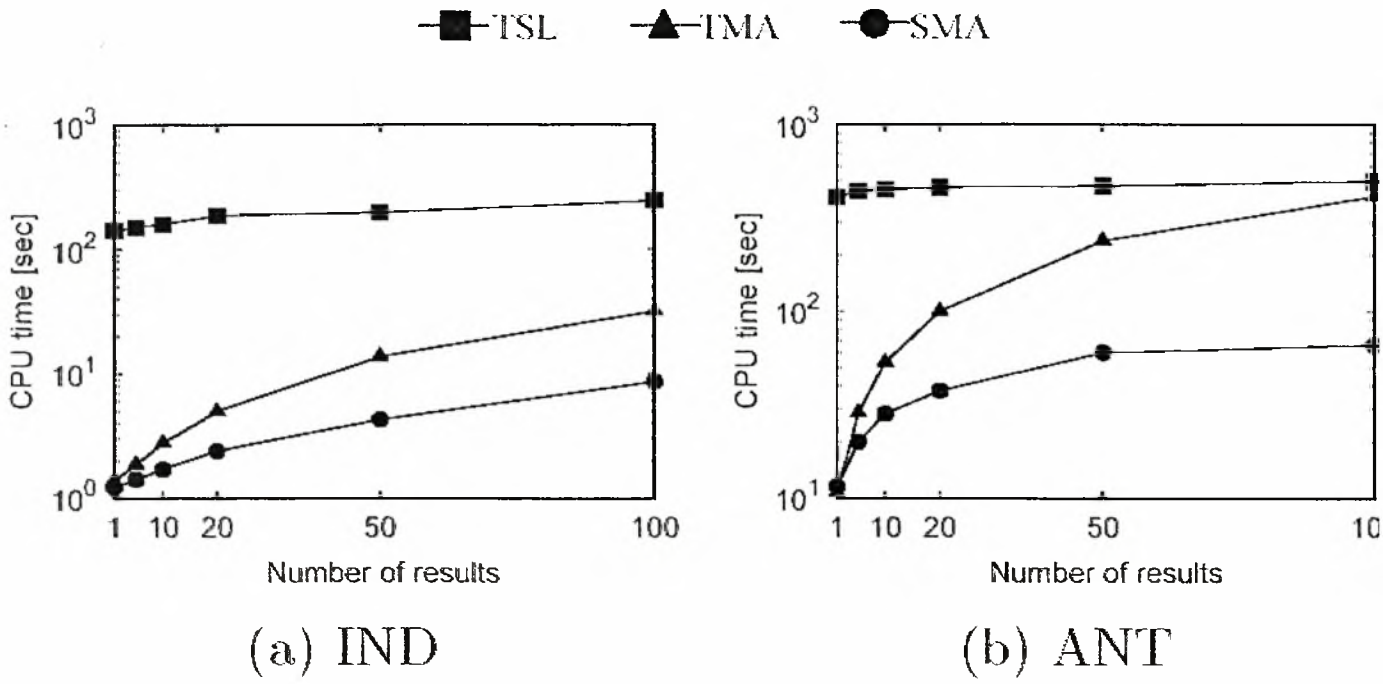
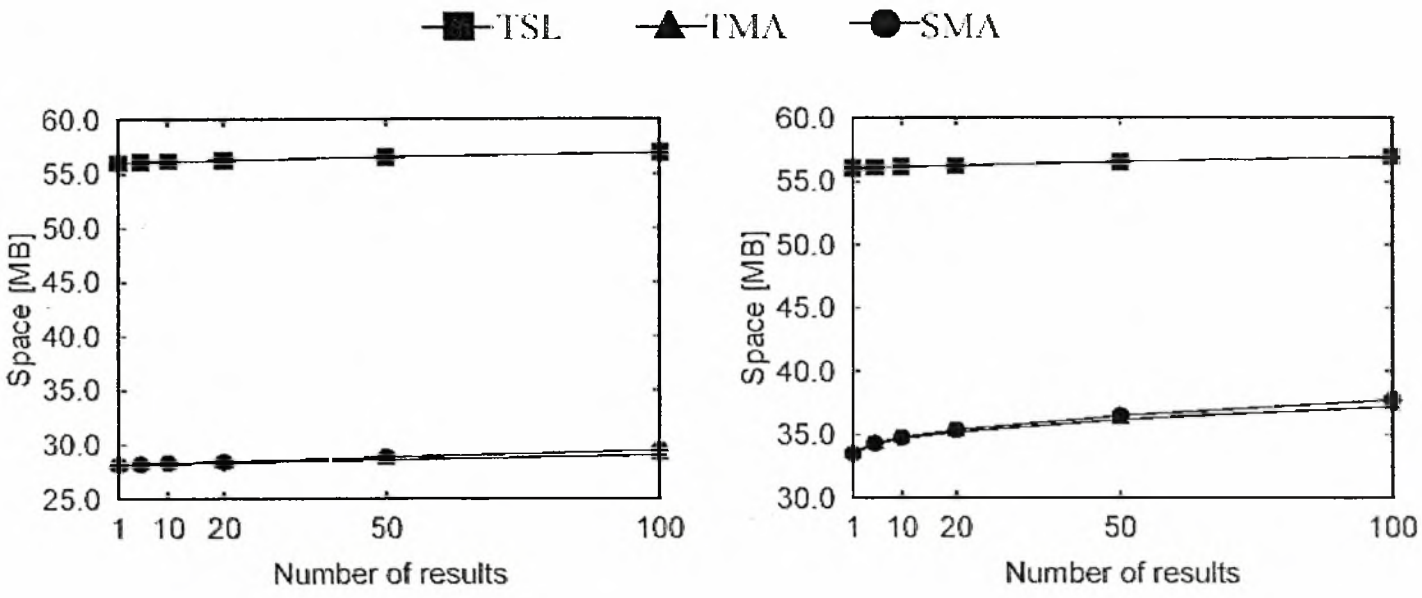


Figure 19: CPU time vs. k

Η εικόνα 20 επιδεικνύει τον χώρο που απαιτείται για το ίδιο πείραμα. Οι TMA και SMA χρειάζονται να διατηρούν για κάθε κελί ι)τις λίστες επιρροής,ιι)την λίστα των αντίστοιχων σημείων. Τα γενικά κόστη των αλγορίθμων αυξάνουν με το k επειδή πρέπει να αποθηκεύσουν περισσότερα αποτελέσματα ανά ερώτημα. Επιπλέον ,για τους TMA, SMA ,οι λίστες επιρροής μεγαλώνουν με το k. Ο TSL, από την άλλη δαπανά περισσότερο χώρο συγκρινόμενα με τις μεθόδους μας εξαιτίας των επιπρόσθετων d ταξινομημένων λιστών . Ο πίνακας 2 δείχνει το μέσο όρο της πληθικότητας των συνόλων των επεξεργασμένων όψεων και των skybands, για TSL ,SMA, αντιστοίχως. Ο SMA, διατηρεί πολύ λίγα επιπλέον σημεία. Ας σημειώσουμε ότι ο SMA διατηρεί

Λιγότερα σημεία στη κορυφαία –k όψη από τον TSL,αφού συνεχώς αποβάλλει τις πλειάδες που δεν εμφανίζονται στο αποτέλεσμα.



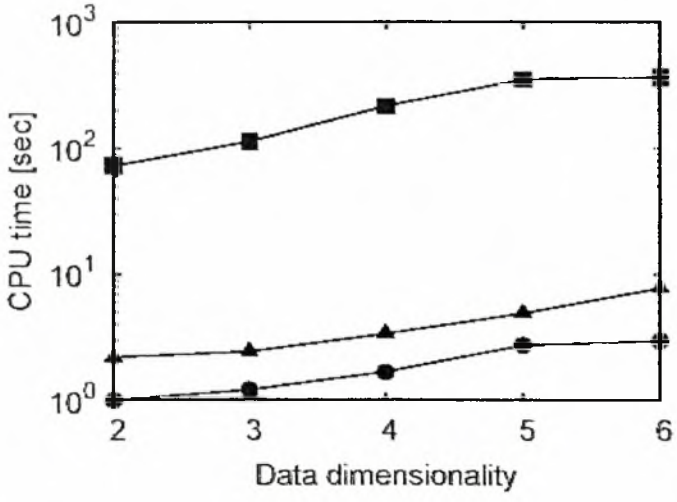
(a) IND (b) ANT

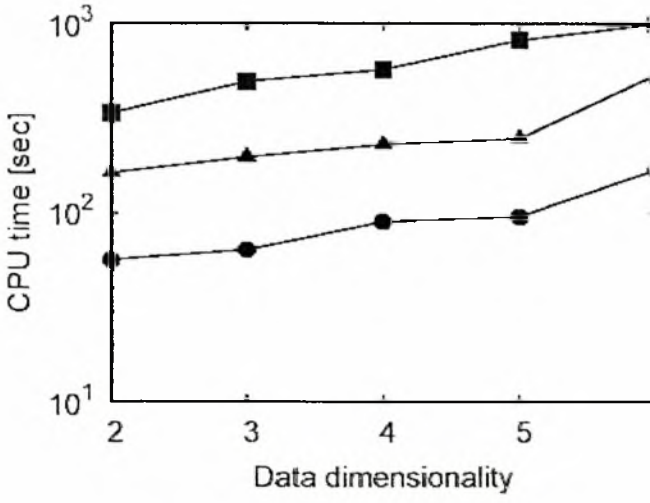
Figure 20: Space requirements vs. k

Table 2: Average view/skyband size per query

k	IND		ANT	
	TSL	SMA	TSL	SMA
1	3.3	1.1	3.1	1.1
5	8.6	5.9	8.4	5.9
10	17.1	11.2	17.2	11.5
20	26.7	21.6	26.9	22.4
50	63.0	53.3	64.4	54.4
100	113.2	104.6	113.6	106.5

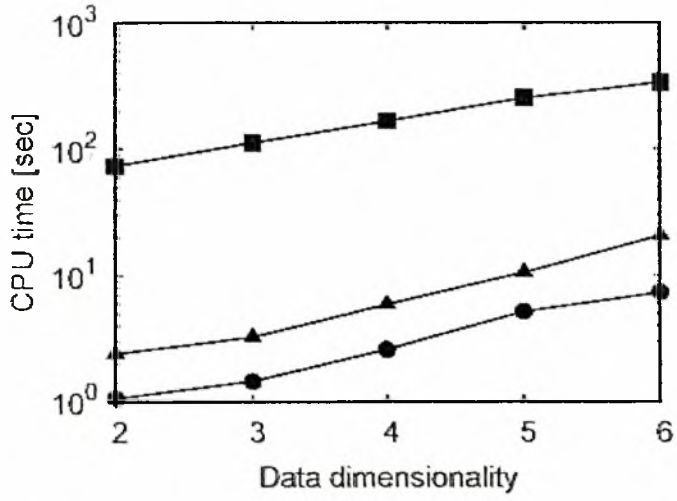
Τελικά, δείχνουμε την αποτελεσματικότητα των προτεινόμενων μεθόδων για μη γραμμικές συναρτήσεις. Οι εικόνες 21(α), 21(β) εκτιμούν το κόστος των ερωτημάτων με το score(p)=$\prod_{i=1}^{d}(a_i + p.x_i)$ όπου $a_i \in [0,1]$ σαν συνάρτηση του d για IND και ANT. Οι εικόνες 21(c),21(d) επαναλαμβάνουν το πείραμα για 1K ερωτήματα με score(p) = $\sum_{i=1}^{d} a_i * p.x_i^2$. Η σχετική απόδοση των αλγορίθμων είναι παρόμοια με την περίπτωση των γραμμικών συναρτήσεων (εικόνα 15), δείχνοντας έτσι τη γενικότητα των μεθόδων.

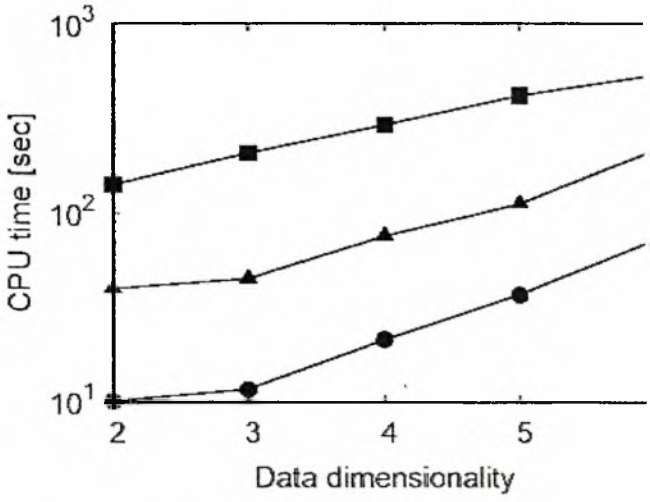




(a) $f(p) = \prod_{i=1}^{d}(a_i + p.x_i)$ (IND)

(b) $f(p) = \prod_{i=1}^{d}(a_i + p.x$ (ANT)





(c) $f(p) = \sum_{i=1}^{d} a_i \cdot p.x_i^2$ (IND)

(d) $f(p) = \sum_{i=1}^{d} a_i \cdot p.x$ (ANT)

Figure 21: CPU time vs. d for non-linear f

ΠΗΓΕΣ

1. "Continuous Monitoring of Top-k Queries over Sliding Windows" Kyriakos Mouratidis, Spyridon Bakiras, Dimitris Papadias 2006.

2. "Models and issues in data stream systems" B.Babcock , S.Babu , M.Datar , R.Motwani, J.Widom 2002.

3. "The Skyline operator" S.Börzsönyi, D.Kossmann, K.Stocker 2001

4. "Optimal aggregation algorithms for middleware" R.Fagin ,A.Lotem, M.Naor 2001

5. "Efficient maintenance of materialized top-k views" K. Yi, H. Yu, J. Yang , G. Xia , Y. Chen 2003

6. "Continues queries over data streams" S.Babu ,J.Wildom 2001
7. "Continues monitoring over append-only databases" D.Terry, D.Goldberg , D.Nichols ,B.Oki 1992

8. Tradebort home page http//www.traderbot.com

9. "A First Course in Database System" J. Ullman , J.Widom 1997

10. "The Onion technique: Indexing for linear optimization queries" Y.-C.Chang,L. D. Bergman, V. Castelli, C.-S. Li, M.-L. Lo, J. R. Smith. 2000

11."Probabilistic optimization of top N queries" , D. Donjerkovic ,R. Ramakrishman 1999

12." The QBIC project: querying images by content using color, texture and shape " W.Niblack,R.Barber ,W.Equitz, M.Flickner,E. Glasman, D.Petkovic,P.Yanker 1993

13. "Combining fuzzy information from multiple systems" , R. Fagin 1999