ΣΥΝΕΧΗΣ ΥΠΟΛΟΓΙΣΜΟΣ ΓΕΝΙΚΕΥΜΕΝΩΝ SKYLINE ΕΡΩΤΗΜΑΤΩΝ ΣΕ ΚΑΤΑΝΕΜΗΜΕΝΑ ΣΥΣΤΗΜΑΤΑ

Η
ΜΕΤΑΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ ΕΞΕΙΔΙΚΕΥΣΗΣ

Υποβάλλεται στην

ορισθείσα από την Γενική Συνέλευση Ειδικής Σύνθεσης
του Τμήματος Πληροφορικής
Εξεταστική Επιτροπή

από την

Αικατερίνη Φωτιάδου

ως μέρος των Υποχρεώσεων

για τη λήψη

του

ΜΕΤΑΠΤΥΧΙΑΚΟΥ ΔΙΠΛΩΜΑΤΟΣ ΣΤΗΝ ΠΛΗΡΟΦΟΡΙΚΗ

ΜΕ ΕΞΕΙΔΙΚΕΥΣΗ ΣΤΟ ΛΟΓΙΣΜΙΚΟ

Σεπτέμβρης 2007

ΕΥΧΑΡΙΣΤΙΕΣ

Ευχαριστώ θερμά την επιβλέπουσα καθηγήτρια μου Ευαγγελία Πιτουρά για την βοήθεια της, τον χρόνο της και τις πολύτιμες γνώσεις που μου μετέδωσε. Επίσης θέλω να ευχαριστήσω την οικογένεια μου για την κατανόηση και υποστήριξη της. Ιδιαίτερα ευχαριστώ το σύζυγο μου Αχιλλέα που πίστεψε σε εμένα και μου έδωσε τη δύναμη να συνεχίσω σε κάθε δυσκολία που αντιμετώπισα.

ΠΕΡΙΕΧΟΜΕΝΑ

ΕΥΡΕΤΗΡΙΟ ΠΙΝΑΚΩΝ

ΕΥΡΕΤΗΡΙΟ ΣΧΗΜΑΤΩΝ

Σχήμα Σελ.

ΕΥΡΕΤΗΡΙΟ ΑΛΓΟΡΙΘΜΩΝ

ΠΕΡΙΛΗΨΗ

Φωτιάδου Αικατερίνη του Γρηγορίου και της Βικτώριας.
MSc, Τμήμα Πληροφορικής, Πανεπιστήμιο Ιωαννίνων, Σεπτέμβρης, 2007.
Συνεχής Υπολογισμός Γενικευμένων Skyline Ερωτημάτων σε Κατανεμημένα Συστήματα.
Επιβλέπουσα: Ευαγγελία Πιτουρά

Ένα σύστημα ομότιμων κόμβων αποτελείται από ένα σύνολο αυτόνομων υπολογιστικών κόμβων στο Διαδίκτυο, οι οποίοι συνεργάζονται με σκοπό την ανταλλαγή δεδομένων. Στα συστήματα αυτά ο όγκος της πληροφορίας που διαμοιράζεται είναι τεράστιος. Οι ανάγκες των χρηστών ξεπερνούν τους παραδοσιακούς τρόπους αναζήτησης και ζητούν απαντήσεις οι οποίες περιλαμβάνουν «ενδιαφέρουσες» μόνο πληροφορίες. Τις απαντήσεις αυτές μπορούν να δώσουν οι Skyline ερωτήσεις. Το Skyline ενός συνόλου σημείων ορίζεται ως τα σημεία που δεν κυριαρχούνται (not dominated) από κανένα άλλο σημείο. Ένα σημείο p κυριαρχεί ενός άλλου σημείου q αν το p δεν είναι χειρότερο σε καμία διάσταση (από αυτές που μας ενδιαφέρουν) από το q και είναι καλύτερο σε τουλάχιστον μια. Υπάρχει πλήθος αλγόριθμων στην βιβλιογραφία για υπολογισμό του Skyline. Παρόλα αυτά είναι ελάχιστες και πολύ πρόσφατες οι εργασίες που αναφέρονται στον υπολογισμό του Skyline σε συστήματα ομότιμων. Για αυτό το λόγο είναι σκόπιμο να προταθούν νέοι τρόποι για τη λύση του προβλήματος. Οι λύσεις αυτές θα πρέπει να σέβονται τα ιδιαίτερα χαρακτηριστικά των συστημάτων ομότιμων όπως η απουσία ολικής ή κεντρικής πληροφορίας, η δυναμικότητα και συνεχής εξέλιξη του συστήματος, το υψηλό υπολογιστικό κόστος όταν πρόκειται για ακριβή αποτελέσματα.

Σε αυτή την εργασία καλούμαστε να λύσουμε το πρόβλημα εύρεσης του Skyline σε ένα σύστημα ομότιμων κόμβων. Επιλέγουμε ένα σύστημα ομότιμων με Super peer αρχιτεκτονική. Δίνουμε αποδοτικές δομές και αναπαράσταση με bit των δεδομένων έτσι ώστε να επισπεύσουμε τους υπολογισμούς. Ορίζουμε επιπλέον πληροφορία έτσι ώστε να είναι δυνατόν να απαντηθούν όλα τα πιθανά Skyline ερωτήματα ενός συνόλου σημείων, επιβαρύνοντας όσο το δυνατόν λιγότερους ομότιμους. Επιπλέον, ορίζουμε μία cache Skyline ερωτημάτων έτσι ώστε να επισπεύσουμε ακόμη περισσότερο την απάντηση των ερωτήσεων και να μειώσουμε το επικοινωνιακό κόστος. Τέλος ορίζουμε τρόπο υποστήριξης continuous Skyline ερωτημάτων στο σύστημα. Η λύση μας είναι απλή, επιβαρύνει μόνο τους κόμβους με επιπλέον αρμοδιότητες (Super peers), δίνει ακριβή αποτελέσματα, αποφεύγει τους κύκλους, έχει γρήγορες ενημερώσεις και ρωτά όσο το δυνατόν λιγότερους ομότιμους του συστήματος.

EXTENDED ABSTRACT IN ENGLISH

Fotiadou Aikaterini.
MSc, Computer Science Department, University of Ioannina, Greece.
September 2007
Continuous Extended Skyline Queries Computation on Peer-to-Peer Systems.
Supervisor: Evaggelia Pitoura

Peer-to-peer systems are a federation of peers with diverse connectivity. Information retrieval in p2p systems has gained attention since usually a massive quantity of data is involved in a query. Traditional queries as MAX or MIN are not enough to satisfy the increasing needs of the users. Skyline queries help users make intelligent decisions. These queries are used to find a set of non dominated data points in a multidimensional data set. One data item dominates another if it is as good or better in all dimensions and better in at least one dimension, where the meaning of "goodness" depends on the preferences of the user. Skyline queries in the database context were first introduced in [BoKS01]. Much work has been done since then in this field but less in Skyline computation in p2p systems. Past approaches do not respect the unique characteristics of p2p systems such as strict decentralization and limited knowledge. In this work, we aim at solving the problem of efficiently processing Skyline queries in an unstructured p2p system with Super peers, taking into consideration all the challenges that arise from the nature of these systems.

Our approach is called BitPeer. The name was inspired by the bit representation of the data that the peers of the system use. The p2p system we rely on to solve the Skyline computation problem follows a Super peer architecture. The nodes are not equal, but some of them have more power. These nodes are called Super Nodes (SN) while the rest are called Ordinary Nodes (ON). Each peer stores its data using a Bitmap representation, which is then used to compute extended Skyline. Extended Skyline was introduced by [VDKV07] and is an extension of Skyline that has a beautiful property: "The extended Skyline of a set of data is a super-set of the union of all subspace Skyline". Each peer computes extended Skyline and sends the results to the SN to which it is connected. Each SN computes the total extended Skyline of its ONs and stores it in Bitmap representation. The next time the SN receives a Skyline query, it does not have to ask all its ONs and wait a lot of time. The algorithm that each peer uses to compute Skyline locally is the Bitmap algorithm proposed by [TaEO01] converted such as to compute extended Skyline. Using this conversion we can compute Skyline without having to scan all data separately. Thus we can produce results very fast. We also investigate how the number of buckets of the Bitmap representation affects the accuracy of the Skyline and find a heuristic way to have better results, although having the same number of buckets. The bitmap-based approach that we follow in solving the problem has several advantages; first of all it doesn't need a lot of additional space to store the data.

Second, it saves bandwidth when data travel all over the network. Moreover, the structure we use is easily updated and finally, the computation is very fast and simple.

The routing of a query is very simple too. When a node wants to pose a Skyline query, it just asks all its neighbors (only SNs), waits for the answer and computes the total result. When someone receives a query, it checks if it has already received a query with the same id (within a time period) and if yes it does nothing. This prevents cycles or round trips. If it is the first query, it forwards it to its neighbors, waits for answers, computes result and sends it back to the one who asked it.

The update of the Bitmap structures has a small cost when an insertion happens but such cost increases in the case of a deletion. Updates occur in the extended Skyline that a SN stores only when u% of the data of an ON change. If this change is an insertion, we check if it affects the Skyline. If yes, we recompute the Skyline to see whether the new points dominate others. If a deletion happens that affects Skyline the SN have to ask all its ON to send their data, so as to compute again the extended Skyline. This is necessary because data that were dominated by the deleted points might now belong to the extended Skyline. Experiments show that the cost of the updates is small when we have Correlated distribution of data, but it is grows with Anti-correlated distribution. This happens because updates in data always affect extended Skyline with Anti-Correlated but rarely with Correlated distribution.

To speed up Skyline query processing, we propose a caching scheme for the SNs. Every SN has a cache which stores Skyline queries. Each time a SN computes a local result, it stores it to its cache. The next time it receives a query, it first checks its cache. In this way, we save a lot of messages, one for each neighbor whose data are already in cache. Moreover, we also save the messages that the neighbors would send to their neighbors etc. Experiments show that message gain is up to 45% (when compared with our initial method and not counting the messages that a naïve technique would send to all ONs). Moreover we propose reusing of already computed queries. Caches store extended Skyline which has two properties: i) extended Skyline of x dimensions is subset of extended Skyline of y dimensions when x is a subset of y, ii) extended Skyline of x dimensions is hyper set of extended Skyline of y dimensions when x is a hyper set of y. So, if a query is stored in cache, whose dimensions are a hyper set of the dimensions of the new Skyline query that we need to compute, the old result can be used to compute the new one, instead of using extended Skyline. We save messages, bandwidth, and computation time because we do not have to compute again points that are already excluded from the result. Similarly, we introduce “fast Skyline queries”, which are queries that do not require exact answers but need to be answered very fast. For these answers, we use computed queries that already exist in cache and that involve a subset of the dimensions required. If none such query exists, we can also use a hyper set as before.

Additionally, we propose an effective way to support continuous queries. Continuous queries allow users to obtain new results without having to issue the same query repeatedly. They are mostly used in peer-to-peer systems for monitoring. In our approach, we have two kinds of continuous queries, change-based and timer-based. The first updates the result of the query every time there is a data change. The second updates the result only at time intervals specified by the user. In each case, we group the continuous queries in a different way, so that less data are moved around the system, less computation happens and results can be shared by related computation.

ΚΕΦΑΛΑΙΟ 1. ΕΙΣΑΓΩΓΗ

Στο κεφάλαιο αυτό περιγράφεται συνοπτικά το πρόβλημα το οποίο καλούμαστε να επιλύσουμε. Περιγράφεται το Skyline και το subspace Skyline ενός συνόλου σημείων και δίνεται ο λόγος για τον οποίο είναι σημαντικός ο υπολογισμός του σε συστήματα ομότιμων κόμβων. Στη συνέχεια δίνουμε συνοπτικά τα χαρακτηριστικά των συστημάτων ομότιμων τα οποία πρέπει να σεβαστούμε για την επιτυχία της μεθόδου που προτείνουμε. Έπειτα γίνεται μία συνοπτική περιγραφή της προσέγγισης που ακολουθείται. Τέλος δίνεται η δομή της εργασίας αυτής.

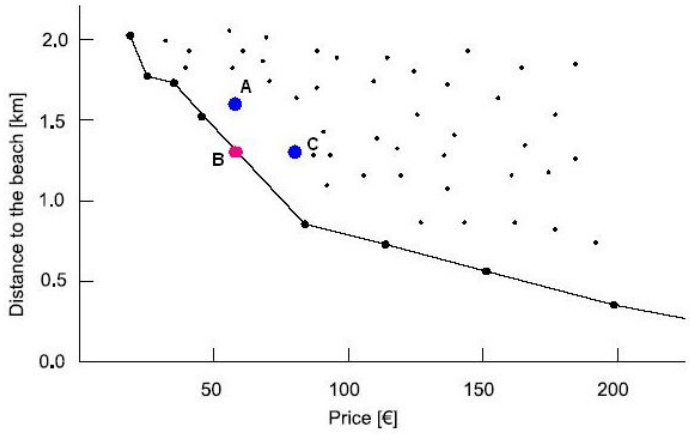
1.1. Υπολογισμός Skyline σε Συστήματα Ομότιμων

Στη σημερινή κοινωνία το Διαδίκτυο έχει μπει για τα καλά στη ζωή των ανθρώπων. Όλο και περισσότεροι έχουν τη δυνατότητα να χρησιμοποιούν τον τεράστιο όγκο πληροφορίας που αυτό διαθέτει. Πλέον οι χρήστες έχουν την δυνατότητα να ανταλλάσσουν πληροφορία μεταξύ τους, είτε να θέτουν ένα ερώτημα σε ένα σύστημα όπου μόνο κάποιοι συγκεκριμένοι χρήστες συμμετέχουν. Οι έως τώρα διαθέσιμες ερωτήσεις τύπου MAX ή MIN δεν επαρκούν για τις ολοένα αυξανόμενες ανάγκες των χρηστών. Είναι αρκετά συχνό φαινόμενο το να θέλουν να βελτιώσουν τις επιλογές τους αποφασίζοντας με γνώμονα πάνω από ένα χαρακτηριστικά των δεδομένων. Ένα ερώτημα που μπορεί να ικανοποιήσει αυτές τις ανάγκες είναι το Skyline ερώτημα, το οποίο επιστρέφει ένα σύνολο από ενδιαφέρουσες απαντήσεις, δοθέντων των

χαρακτηριστικών που μας ενδιαφέρουν. Συγκεκριμένα το Skyline ενός συνόλου σημείων ορίζεται ως τα σημεία που δεν κυριαρχούνται (not dominated) από κανένα άλλο σημείο. Ένα σημείο p κυριαρχεί ενός άλλου σημείου q αν το p δεν είναι χειρότερο σε καμία διάσταση(από αυτές που μας ενδιαφέρουν) από το q και είναι καλύτερο σε τουλάχιστον μια. Η σύνταξη των Skyline ερωτημάτων όπως ορίστηκε από τους [BoKS01] είναι η εξής:

SELECT … FROM … WHERE …
GROUP BY … HAVING …
SKYLINE OF [DISTINCT] d1 [MIN |MAX |DIFF], …, dm [MIN |MAX |DIFF]
ORDER BY …

Όπου όταν έχουμε MIN θέλουμε να ελαχιστοποιήσουμε το χαρακτηριστικό, MAX να το μεγιστοποιήσουμε και DIFF να το διαφοροποιήσουμε.



Σχήμα 1.1 SELECT * FROM Hotels WHERE city= “Nassau” SKYLINE OF price MIN, distance MIN;

Ένα παράδειγμα [KoRR02] μιας Skyline ερώτησης φαίνεται στο Σχήμα 1.1. Σε αυτό το παράδειγμα ενδιαφερόμαστε για ξενοδοχεία τα οποία να βρίσκονται κοντά στην παραλία αλλά να είναι και φθηνά. Το Skyline των ξενοδοχείων περιέχει το σύνολο των απαντήσεων, δηλαδή όλα τα ξενοδοχεία που δεν κυριαρχούνται από κάποιο άλλο. Π.χ. το σημείο Α δεν

ανήκει στο Skyline αφού υπάρχει το σημείο B το οποίο είναι πιο κοντά στην παραλία από το A ενώ έχουνε την ίδια τιμή. Αντίστοιχα το C δεν ανήκει στο Skyline αφού είναι στην ίδια απόσταση με το B αλλά το B είναι πιο φθηνό.

Μερικές φορές οι χρήστες ενδιαφέρονται για ένα μόνο υποσύνολο των διαστάσεων. Για παράδειγμα, για τα ξενοδοχεία μπορεί να υπάρχει πλήθος πληροφορίας, όπως η τιμή, η απόσταση, η κατηγορία του, η διαθεσιμότητα δωματίων κτλ. Κάποιοι χρήστες μπορεί να ενδιαφέρονται για το Skyline της τιμής με την απόσταση (όπως στο Σχήμα 1.1), ενώ άλλοι μπορεί να θέσουν Skyline ερώτημα ως προς την κατηγορία του ξενοδοχείου και την τιμή του. Όμοια με το Skyline ορίζουμε το subspace Skyline ενός συνόλου σημείων το οποίο αφορά ένα μόνο υποσύνολο των διαστάσεων των δεδομένων. Στην περίπτωση του subspace Skyline αγνοούμε πλήρως τις τιμές που έχουν τα δεδομένα στις διαστάσεις που μας ενδιαφέρουν και υπολογίζουμε το Skyline των υπόλοιπων διαστάσεων.

Τέτοιου είδους ερωτήσεις που αφορούν διαφορετικά υποσύνολα των διαστάσεων μπορούν να τεθούν πολλές και συγχρόνως στο σύστημα. Συγκεκριμένα αν έχουμε d διαστάσεις υπάρχουν 2^d-1 πιθανές υποσύνολα των διαστάσεων. Οι υπάρχουσες τεχνικές μπορούν να υπολογίσουν το Skyline κάθε τέτοιας ερώτησης ξεχωριστά. Γίνεται όμως σαφές ότι η τεχνική αυτή δεν είναι τόσο αποδοτική όσον αφορά το χρόνο απόκρισης αλλά και την απόδοση του συστήματος. Στα [YLL+05] και [PJET05] προτείνονται αποδοτικές τεχνικές υπολογισμού subspace Skyline ερωτήσεων, με τέτοιο τρόπο ώστε «παρόμοιες» ερωτήσεις να μπορούν να μοιραστούν κάποιους υπολογισμούς.

Έχουν προταθεί πολλοί αλγόριθμοι υπολογισμού του Skyline. Κάποιοι από αυτούς χρησιμοποιούν ένα παράθυρο το οποίο χωρά στην μνήμη και μέσα σε αυτό κάνουν ελέγχους κυριαρχίας για το Skyline. Παράδειγμα τέτοιου αλγόριθμου είναι ο BNL και διάφορες παραλλαγές του που προτείνονται από τους [BoKS01]. Το μειονέκτημα αυτών των αλγορίθμων είναι ότι έχουν μεγάλο υπολογιστικό κόστος και πολύ μεγάλο χρόνο απόκρισης. Μία άλλη τεχνική είναι η χρήση ευρετηρίων και κυρίως B-δένδρων ή R-δένδρων. Τέτοιου είδους αλγόριθμοι είναι π.χ. οι NN [KoRR02], BBS [PTFS03] κτλ. Οι αλγόριθμοι αυτοί, έχουν καλύτερους χρόνους απόκρισης επειδή μπορούν να αποκλείουν κάποια σημεία που δεν

ανήκουν στο Skyline χωρίς να τα ελέγξουν. Επιπλέον, οι περισσότεροι από αυτούς είναι προοδευτικοί. Το κόστος των πλεονεκτημάτων τους όμως είναι η συντήρηση των ευρετηρίων. Ακόμη μία σημαντική μέθοδος είναι αυτή του «Διαίρει και Βασίλευε», η οποία έχει διάφορες παραλλαγές και συνήθως υλοποιείται με τη χρήση ευρετηρίων όποτε και εμπεριέχει τα πλεονεκτήματα και τα μειονεκτήματα αυτών των μεθόδων. Άλλη μια μέθοδος είναι αυτή της Bitmap αναπαράστασης που παρουσιάζουν οι [TaEO01]. Επιπλέον έχει προταθεί πλήθος αλγόριθμων οι οποίοι να έχουν κάποιες επιπλέον ιδιότητες, όπως το να είναι «προοδευτικοί» ή να μπορούν να χρησιμοποιηθούν για δεδομένα που δεν μπορούν πλήρως να ταξινομηθούν.

Η εφαρμογή Skyline ερωτήσεων σε ένα σύστημα ομότιμων δεν είναι κάτι το προφανές. Ένα σύστημα ομότιμων είναι ένα σύνολο κόμβων οι οποίοι συνεργάζονται για ανταλλαγή πληροφορίας. Κάθε κόμβος είναι υπεύθυνος για τα δικά του δεδομένα. Όταν λοιπόν, ένας κόμβος ζητά πληροφορία για την οποία απαιτείται υπολογισμός που να συμπεριλαμβάνει όλα σχεδόν τα δεδομένα του δικτύου, οι κόμβοι πρέπει να συνεργαστούν κατάλληλα, να κάνουν επιμέρους υπολογισμούς, να ανταλλάξουν τα δεδομένα τους, ώσπου ο κόμβος που ενδιαφέρεται να μπορέσει να λάβει τα αποτελέσματα που τον ενδιαφέρουν. Τέτοιου είδους ενέργειες απαιτούν και οι Skyline ερωτήσεις. Στόχος μας είναι η εύρεση ενός κατάλληλου αλγόριθμου δρομολόγησης έτσι ώστε να μπορέσει να απαντηθεί ένα Skyline ερώτημα σε ένα σύστημα ομότιμων. Συνοπτικά κάποια από τα χαρακτηριστικά των συστημάτων ομότιμων είναι [AH01]:

- Όλοι οι ομότιμοι είναι ίσοι υπό την έννοια ότι ο καθένας μπορεί να δρα ως πελάτης και ως εξυπηρέτης (client/ server). Καθένας από αυτούς μπορεί να θέσει ένα ερώτημα ή να απαντήσει σε ένα ερώτημα.
- Απουσία ολικής (global) πληροφορίας. Κάθε ομότιμος γνωρίζει μόνο πληροφορία την οποία κατέχει ο ίδιος και ίσως πληροφορία για κάποιους από τους γείτονες του.
- Ολική πληροφορία ή ολική συμπεριφορά εξάγεται από αλληλεπιδράσεις γειτονικών ομότιμων.
- Το σύστημα είναι δυναμικό, δηλαδή οι ομότιμοι εισέρχονται και εξέρχονται κατά βούληση.

- Είναι σχεδόν αδύνατο να εγγυηθούμε ολοκληρωμένες και ακριβείς απαντήσεις χωρίς εξοντωτική αναζήτηση.

Οι αλγόριθμοι που έχουν ήδη προταθεί για τον υπολογισμό του Skyline βασίζονται σε παραδοχές που δεν ισχύουν σε αυτά τα συστήματα. Η απευθείας εφαρμογή κάποιων από αυτών σε συστήματα ομότιμων θα οδηγούσε σε εξοντωτικές αναζητήσεις, πολλά round trips και την ανάγκη να υπάρχει κάπου κεντρικά ολική πληροφορία. Όλα αυτά είναι καταστροφικά για ένα τέτοιο σύστημα, μη αποδεκτά και συνήθως μη υλοποιήσιμα. Για αυτό το λόγο είναι σκόπιμο να προταθούν νέοι τρόποι για τη λύση του προβλήματος οι οποίοι να λαμβάνουν υπόψη τα ειδικά χαρακτηριστικά των συστημάτων ομότιμων. Συγκεκριμένα έχουμε τις εξής απαιτήσεις:

- Ο αλγόριθμος να μην στηρίζεται σε γνώση ολικής πληροφορίας
- Αποδοτική εύρεση απαντήσεων όσον αφορά το υπολογιστικό κόστος. Δηλαδή:
- Ρωτάμε μόνο μερικούς ομότιμους και
- Τους ρωτάμε μόνο μία φορά
- Στέλνουμε όσον το δυνατόν λιγότερα μηνύματα
- Τα μηνύματα θέλουμε να είναι μικρού μεγέθους – οικονομία bandwidth
- Οι ενημερώσεις θέλουμε να είναι απλές και γρήγορες

Ένα χαρακτηριστικό αυτών των συστημάτων είναι ότι τα δεδομένα στο δίκτυο αλλάζουν συνεχώς είτε λόγω προσθήκης νέων δεδομένων στους κόμβους είτε λόγω της συνεχής άφιξης και αποχώρησης κόμβων. Στόχος μας είναι να βρούμε έναν αλγόριθμο, ο οποίος κατανέμει το φορτίο στους κόμβους του δικτύου, θα απαιτεί μικρό αριθμό hops και κανένα round trip. Τέλος, θέλουμε οι κόμβοι του δικτύου να κρατούν για αυτό το ερώτημα μόνο χρήσιμη πληροφορία, η οποία θα διαγράφεται όταν το ερώτημα (ή αυτός που το θέτει) πάψει να υπάρχει στο δίκτυο.

1.2. Επισκόπηση Προσέγγισης

Το πρόβλημα που μας απασχολεί λοιπόν, είναι η εύρεση του Skyline ενός συνόλου σημείων σε συστήματα ομότιμων. Λαμβάνοντας υπόψη τα ιδιαίτερα χαρακτηριστικά αυτών των συστημάτων εξετάζουμε το κατά πόσον επηρεάζουν το πρόβλημα διάφοροι παράμετροι.

Αρχικά, ορίζουμε μία τοπολογία με Super Nodes. Σε αυτήν την τοπολογία μελετάμε το πρόβλημα εύρεσης του Skyline. Τα δεδομένα ανατίθενται στους κόμβους ομοιόμορφα και παράγονται από τρεις κατανομές τις οποίες εξετάζουμε ξεχωριστά. Οι Super Nodes του συστήματος διατηρούν επιπλέον πληροφορία έτσι ώστε να μπορούν να απαντήσουν σε οποιαδήποτε Skyline ερώτηση. Χρησιμοποιούμε το extended Skyline όπως αυτό ορίστηκε από τους [VDKV07], για τη διατήρηση της πληροφορίας αυτής. Σύμφωνα με την τοπολογία αυτή και σεβόμενοι τις απαιτήσεις των συστημάτων ομότιμων ορίζουμε έναν αλγόριθμο δρομολόγησης Skyline ερωτημάτων. Η δρομολόγηση γίνεται μόνο στο δίκτυο στο οποίο συνδέονται οι SN. Ο αλγόριθμος επιστρέφει ακριβή αποτελέσματα, ρωτά μόνο μία φορά τους κόμβους του δικτύου και ρωτά όσο το δυνατόν λιγότερους κόμβους.

Εκτός από τον αλγόριθμο δρομολόγησης απομένει να ορίσουμε και έναν κεντρικοποιημένο αλγόριθμο τον οποίο θα χρησιμοποιεί κάθε ομότιμος ξεχωριστά έτσι ώστε να μπορεί τοπικά να υπολογίζει το Skyline. Οι αλγόριθμοι που έχουν προταθεί είναι πολλοί και κάθε ένας από αυτούς έχει τα δικά του πλεονεκτήματα και μειονεκτήματα. Από αυτούς επιλέξαμε τον Bitmap των [TaEO01]. Ο κύριος λόγος επιλογής αυτού του αλγόριθμου είναι ότι χρησιμοποιεί bit αναπαράσταση των δεδομένων. Με αυτήν την αναπαράσταση και πράξεις AND και OR υπολογίζεται πολύ εύκολα το Skyline ενός συνόλου σημείων. Τροποποιούμε αυτόν τον αλγόριθμο έτσι ώστε να υπολογίζει και το extended Skyline ενός συνόλου σημείων. Λόγω της αναπαράστασης σε bit απαιτείται λίγος χώρος στο δίσκο για αποθήκευση των δεδομένων σε σχέση π.χ. με μία μέθοδο που χρησιμοποιεί ευρετήριο. Επίσης σημαντικό είναι ότι μεταφέρονται πολύ λιγότερα δεδομένα στο δίκτυο από ότι αν θέλαμε να μεταφέρουμε τα πραγματικά δεδομένα. Επιπλέον, η ενημέρωση της δομής αυτής είναι πολύ απλή και εύκολη ειδικά αν τη συγκρίνουμε με τον τρόπο ενημέρωσης ενός ευρετηρίου. Το κόστος των ενημερώσεων είναι πολύ σημαντικός παράγοντας σε έναν σύστημα ομότιμων το

οποίο είναι δυναμικό και τα δεδομένα αλλάζουν διαρκώς. Ένα ακόμη πλεονέκτημα του αλγόριθμου που επιλέξαμε είναι ότι μπορεί να εξάγει με γρήγορο τρόπο τα πρώτα αποτελέσματα του Skyline. Μία από τις καθοριστικές του ιδιότητες είναι η άμεση εφαρμογή subspace Skyline υπολογισμού. Αυτό με λίγα λόγια σημαίνει ότι μπορούμε να χρησιμοποιήσουμε τη δομή ως έχει για να υπολογίσουμε το Skyline οποιουδήποτε υποσυνόλου των διαστάσεων. Δεν χρειάζεται καμία επιπλέον δομή, καμία τροποποίηση, και κανένας περίπλοκος υπολογισμός.

Στη συνέχεια της εργασίας εξετάζουμε τον τρόπο με τον οποίο γίνονται ενημερώσεις του extended Skyline στο σύστημα και πώς μπορούμε να τον βελτιώσουμε. Αν οι ενημερώσεις δεν επηρεάζουν το Skyline τότε δεν γίνεται καμία ενέργεια. Στην περίπτωση που έχουμε εισαγωγή δεδομένων τα πράγματα είναι απλά, αφού απαιτείται απλά επαναυπολογισμός του Skyline για έλεγχο κυριαρχίας με τα νέα δεδομένα. Στην περίπτωση της διαγραφής τα πράγματα είναι πιο περίπλοκα, αφού θα πρέπει να εξεταστούν και στοιχεία που πριν δεν άνηκαν στο Skyline. Το κόστος των ενημερώσεων εξαρτάται κατά έναν μεγάλο βαθμό από την κατανομή με την οποία έχουν παραχθεί τα δεδομένα.

Ως βελτίωση στο σύστημα, προσθέτουμε μία cache Skyline ερωτημάτων. Την cache αυτή τη διατηρούν μόνο οι SN του συστήματος. Η δρομολόγηση των Skyline ερωτημάτων τροποποιείται έτσι ώστε να γίνεται χρησιμοποίηση των αποτελεσμάτων της cache. Η χρησιμοποίηση της cache έχει ως στόχο τη μείωση του επικοινωνιακού κόστους και την αύξηση του χρόνου απόκρισης. Επίσης εισάγουμε τα “fast Skyline queries” τα οποία είναι ερωτήματα που δεν απαιτούν ακριβή αποτελέσματα, αλλά πολύ γρήγορο χρόνο απόκρισης. Για τα αποτελέσματα των ερωτημάτων αυτών δεν γίνεται κανένας υπολογισμός, αλλά χρησιμοποιούνται τα πιο «κοντινά» αποτελέσματα που υπάρχουν στην cache.

Το σύστημα υποστηρίζει και continuous ερωτήματα. Τα continuous ερωτήματα, είναι ερωτήματα τα οποία τίθενται μία μόνο φορά στο σύστημα και εκτελούνται κάθε φορά που συμβαίνει ένα γεγονός. Χρησιμοποιούνται κυρίως για παρακολούθηση του δικτύου. Χωρίζονται σε change-based και timer-based. Στα πρώτα, το ερώτημα εκτελείται κάθε φορά που συμβαίνει κάποια αλλαγή- ενημέρωση στα δεδομένα. Στα δεύτερα εκτελείται ανά κάποια

χρονική περίοδο την οποία ορίζει ο χρήστης. Η πρώτη τεχνική δίνει σαφώς πιο ακριβή αποτελέσματα αλλά επιβαρύνει πολύ το δίκτυο. Σε κάθε μία από αυτές γίνεται διαφορετική ομαδοποίηση των continuous ερωτημάτων ανά SN έτσι ώστε να αυξηθεί η απόδοση του συστήματος. Η κύρια ιδέα είναι ο χωρισμός των ερωτημάτων είτε ανάλογα με τις διαστάσεις του για να πετύχουμε επαναχρησιμοποίηση αποτελεσμάτων, είτε ανάλογα με την περίοδο ενημέρωσης έτσι ώστε να βελτιωθεί το throughput του συστήματος.

Περιληπτικά τα θέματα στα οποία συνεισφέρουμε σε αυτή την εργασία είναι: α) εισαγωγή κάδων στη Bitmap αναπαράσταση για αναπαράσταση συνεχών και όχι μόνο διακριτών τιμών, β) επέκταση του Bitmap αλγόριθμου για υπολογισμό extended Skyline, γ) υπολογισμός Skyline ερωτημάτων σε σύστημα ομότιμων με Super peer αρχιτεκτονική, δ) προσθήκη cache Skyline ερωτημάτων και επαναχρησιμοποίηση επιμέρους αποτελεσμάτων, ε) ομαδοποίηση continuous Skyline ερωτημάτων ανάλογα με τις διαστάσεις τους ή ανάλογα με τη συχνότητα ενημέρωσης.

1.3. Δομή της Διατριβής

Στο Κεφάλαιο 2 δίνονται ορισμοί του Skyline και extended Skyline ενός συνόλου σημείων. Περιγράφεται αναλυτικά η αναπαράσταση Bitmap των δεδομένων και δίνονται αλγόριθμοι εύρεσης του Skyline και extended Skyline ενός συνόλου σημείων με βάση την αναπαράσταση αυτή. Στο Κεφάλαιο 3 περιγράφεται αναλυτικά το σύστημα στο οποίο επιλύουμε το πρόβλημα και η μέθοδος επίλυσης που προτείνουμε. Τη μέθοδο αυτή την ονομάζουμε BitPeer. Συγκεκριμένα, ορίζεται το σύστημα ομότιμων στο οποίο επιλύεται το πρόβλημα, και ο αλγόριθμος δρομολόγησης που χρησιμοποιείται έτσι ώστε να απαντηθεί ένα Skyline ερώτημα. Επίσης αναλύεται ο τρόπος με τον οποίο ενημερώνονται οι δομές τις οποίες έχουν οι ομότιμοι και το κόστος αυτών των ενημερώσεων. Τέλος γίνεται μία σύγκριση της προσέγγισης αυτής σε σχέση με παλαιότερες προσεγγίσεις. Στο Κεφάλαιο 4 προσθέτουμε μία cache σε κάθε Super Node έτσι ώστε να επισπευτεί η απάντηση των Skyline ερωτήσεων. Περιγράφεται η δομή της και ο νέος τρόπος δρομολόγησης των ερωτημάτων. Επίσης περιγράφεται και ένας τρόπος επαναχρησιμοποίησης ερωτημάτων που ήδη έχουν υπολογιστεί

έτσι ώστε να απαντηθούν «γρήγορα» ερωτήματα Skyline. Στο Κεφάλαιο 5 δίνεται ένας τρόπος με τον οποίο το σύστημα μας μπορεί να υποστηρίξει continuous ερωτήματα. Ορίζονται δύο τύποι continuous ερωτημάτων, τα change-based και τα timer-based και γίνεται μία σύγκριση μεταξύ τους. Το Κεφάλαιο 6 περιέχει τα πειραματικά αποτελέσματα των προηγούμενων κεφαλαίων, ενώ στο Κεφάλαιο 7 περιγράφονται συνοπτικά οι σχετικές εργασίες και γίνεται μία σύγκριση αυτών. Αναλυτικές περιγραφές σχετικών εργασιών βρίσκονται στο παράρτημα.

ΚΕΦΑΛΑΙΟ 2. ΥΠΟΛΟΓΙΣΜΟΣ ΓΕΝΙΚΕΥΜΕΝΩΝ ΕΡΩΤΗΜΑΤΩΝ ΤΥΠΟΥ SKYLINE

Στο κεφάλαιο αυτό δίνεται ο ορισμός των ερωτημάτων Skyline. Επίσης, ορίζεται το extended Skyline ενός συνόλου σημείων, το οποίο είναι μία επέκταση του Skyline, με την ιδιότητα ότι από αυτό μπορεί να εξαχθεί το Skyline όλων των δυνατών συνδυασμών των διαστάσεων. Στη συνέχεια, περιγράφεται η Bitmap αναπαράσταση την οποία χρησιμοποιούν τα δεδομένα του συστήματος μας. Έπειτα εξετάζεται συνοπτικά το πώς επηρεάζει ο αριθμός των κάδων της αναπαράστασης την ακρίβεια του Skyline ενός συνόλου σημείων. Τέλος, δίνονται αλγόριθμοι υπολογισμού Skyline και extended Skyline.

2.1. Ορισμός Skyline

Η σημερινή υποδομή των πληροφοριακών συστημάτων παρέχει στους καταναλωτές της ένα τεράστιο πλήθος πληροφορίας. Οι χρήστες θέλουν να λαμβάνουν αποφάσεις οι οποίες εξαρτώνται συνήθως από πολύ μεγάλα σύνολα δεδομένων. Μάλιστα είναι αρκετά συχνό φαινόμενο το να θέλουν να βελτιώσουν τις επιλογές τους αποφασίζοντας με γνώμονα παραπάνω από ένα χαρακτηριστικό των δεδομένων. Εδώ κάνουν την εμφάνιση τους τα Skyline ερωτήματα.

Ορίζουμε ως S ένα σύνολο σημείων, d διαστάσεων. Κάθε διάσταση i, $1 \le i \le d$ αναπαρίσταται ως d_i. Ονομάζουμε D το σύνολο των διαστάσεων $\{d_1, \ldots, d_d\}$. Η τιμή του σημείου p για τη διάσταση d_i γράφεται ως $p(d_i)$.

Ορισμός 2.1 Skyline

Το Skyline ενός συνόλου σημείων ορίζεται ως τα σημεία που δεν κυριαρχούνται (not dominated) από κανένα άλλο σημείο. Ένα σημείο p κυριαρχεί ενός άλλου σημείου q αν το p δεν είναι χειρότερο σε καμία διάσταση (από αυτές που μας ενδιαφέρουν) από το q και είναι καλύτερο σε τουλάχιστον μια.

Ορισμός 2.2 Κυριαρχία

Δοθέντος ενός συνόλου S σημείων και n score functions $F=\{f_1, f_2, \ldots, f_n\}$ μπορεί να οριστεί μία μερική διάταξη των στοιχείων του S σύμφωνα με τον ορισμό της κυριαρχίας, όπου το p *κυριαρχεί* του q ως προς f, αν και μόνο αν:

$$p \succ_f q :\Leftrightarrow \begin{matrix} (i)\forall f \in F : f(p) \ge f(q), and \\ (ii)\exists f \in F : f(p) > f(q) \end{matrix}$$

Δηλαδή, σύμφωνα με τον ορισμό, για να κυριαρχεί το p του q, θα πρέπει να είναι τόσο καλό όσο το q για όλες τις score functions και καλύτερο τουλάχιστον σε μία. Αν για δύο αντικείμενα δεν ισχύει $p \succ q$, αλλά ούτε $q \succ p$, τότε λέμε ότι τα σημεία αυτά είναι *μη συγκρίσιμα.*

Η σχέση κυριαρχίας ικανοποιεί τις εξής ιδιότητες:

- Μη αυτοπαθής: $p \not\succ p,\ \forall p \in S$.
- Αντι-συμμετρική: Αν ισχύει $p \succ q$, τότε $q \not\succ p$.
- Μεταβατική: Αν $p \succ q$ και $q \succ v$, τότε ισχύει $p \succ v$.

Οι παραπάνω ιδιότητες αποδεικνύονται πολύ εύκολα από τον ορισμό της κυριαρχίας.

Το Skyline ενός συνόλου σημείων προτάθηκε ως τελεστής SQL σε βάσεις δεδομένων από τους [BoKS01]. Δίνεται η σύνταξη των Skyline ερωτημάτων όπως ορίστηκε από τους παραπάνω:

SELECT … FROM … WHERE …
GROUP BY … HAVING …
SKYLINE OF [DISTINCT] d1 [MIN |MAX |DIFF], …, dm [MIN |MAX |DIFF]
ORDER BY …

Όπου η πλειάδα *p* κυριαρχεί της *q* για ένα Skyline ερώτημα "SKYLINE OF d_1 MIN, …, d_k MIN, d_{k+1} MAX, …, d_l MAX, d_{l+1} DIFF, …, d_m DIFF", αν ισχύουν οι παρακάτω τρεις συνθήκες:

- $p(d_i) \leq q(d_i)$ για όλα τα i = 1, … , k
- $p(d_i) \geq q(d_i)$ για όλα τα i = (k + 1) , …, l
- $p(d_i) = q(d_i)$ για όλα τα i = (l + 1) , …, m

Όταν έχουμε MIN θέλουμε να ελαχιστοποιήσουμε το χαρακτηριστικό, MAX να το μεγιστοποιήσουμε και DIFF να το διαφοροποιήσουμε. Σημειώνουμε ότι η εύρεση του Skyline μίας μόνο διάστασης ισοδυναμεί με απλό MIN, MAX ή DIFF. Για λόγους απλότητας στη συγκεκριμένη εργασία θα υποθέσουμε ότι στόχος μας είναι μόνο η μεγιστοποίηση κάποιων χαρακτηριστικών, δηλαδή έχουμε μόνο MAX. Επίσης σημειώνουμε ότι ενδιαφερόμαστε μόνο για το SKYLINE OF κομμάτι του ερωτήματος, και συγκεκριμένα για τις διαστάσεις που αυτό περιλαμβάνει. Θεωρούμε δηλαδή ότι έχουμε WHERE=TRUE, HAVING=TRUE, δεν γίνεται GROUP BY κτλ.

2.1.1 Ερωτήματα Skyline σε Υποχώρους

Έστω ένα υποσύνολο U των διαστάσεων των δεδομένων, $U \subseteq D$. Το U σχηματίζει έναν υποχώρο (subspace). Η προβολή ενός σημείου σε αυτόν το υποχώρο είναι μία πλειάδα αποτελούμενη από |U| τιμές. Κάθε μία από αυτές τις τιμές είναι η τιμή του αρχικού σημείου

στην αντίστοιχη διάσταση που περιλαμβάνεται στο U. Η προβολή ενός σημείου p στο U συμβολίζεται ως p_U. Το p_U είναι μία πλειάδα ($p(d_{i1})$, …, $p(d_{i|U|})$), όπου οι d_{i1}, …,$d_{iU} \in U$. Ο ορισμός του Skyline ενός συνόλου σημείων σε έναν υποχώρο U μπορεί να εξαχθεί απευθείας από τον γενικό ορισμό του Skyline [PJET05].

Ορισμός 2.3 Subspace Skyline

Το Skyline ενός συνόλου σημείων όσον αφορά ένα υποσύνολο U των διαστάσεων, ορίζεται ως το Skyline των προβολών των σημείων σε αυτόν τον υποχώρο. Συγκεκριμένα, η προβολή ενός σημείου p ($p \in S$) στον υποχώρο $U \subseteq D$ ανήκει στο Skyline του υποχώρου U, αν το p_U δεν κυριαρχείται από κανένα $q_U \in U$, για όλα τα $q \in S$.

Θεωρούμε ότι οι Skyline ερωτήσεις που γίνονται στο σύστημα μπορεί να αφορούν οποιοδήποτε υποσύνολο U των διαστάσεων των σημείων. Ο αριθμός των πιθανών υποχώρων για d αριθμό διαστάσεων είναι 2^d-1.

Θεώρημα 2.1

Έστω ένα σύνολο σημείων S, με d διαστάσεις ενός συνόλου D. Έστω και δύο υποχώροι $U,V \subseteq D$ και $U \subset V$. Στον υποχώρο V, για κάθε σημείο q που ανήκει στο Skyline ενός συνόλου σημείων στον υποχώρο U ισχύει:

- είτε κυριαρχείται από ένα σημείο p που ανήκει στο Skyline του U
- είτε ανήκει στο Skyline του V

Το παραπάνω θεώρημα αποδεικνύεται στο [YLL+05]. Με απλά λόγια σημαίνει ότι δεν υπάρχει κάποια σχέση «περιεκτικότητας» ανάμεσα στα Skyline διαφορετικών υποχώρων, ακόμα και αν η ιδιότητα αυτή ισχύει για τις διαστάσεις που οι υποχώροι περιλαμβάνουν. Δηλαδή για δύο υποχώρους $U,V \subseteq D$ και $U \subset V$ δεν ισχύει ότι το Skyline του U είναι υποσύνολο του Skyline του V και αντίστροφα.

Πόρισμα 2.1 (Συνθήκη μοναδικής Τιμής)
Δοθέντος ενός συνόλου σημείων S με διαστάσεις D. Για δύο σημεία *p* και *q*, αν $p(d_i) \neq q(d_i)$ $\forall d_i \in D$, τότε για δύο υποχώρους U,V, $U, V \subseteq D$ και $U \subset V$ ισχύει ότι το Skyline του U είναι υποσύνολο του Skyline του V.

2.2. Επίδραση Κατανομής Δεδομένων στο Skyline

Ο αριθμός των αποτελεσμάτων του Skyline ενός συνόλου σημείων δεν εξαρτάται μόνο από το αρχικό πλήθος των δεδομένων. Εξαρτάται κυρίως από τη συσχέτιση που έχουν οι τιμές των δεδομένων στις διαστάσεις τους. Στην εργασία αυτή χρησιμοποιούνται τρεις διαφορετικές κατανομές για την παραγωγή των δεδομένων.

- Independent: στην κατανομή αυτή οι τιμές των διαστάσεων των δεδομένων παράγονται τυχαία. Με άλλα λόγια δεν υπάρχει καμία συσχέτιση ανάμεσα στην τιμές που έχει ένα σημείο στις επιμέρους διαστάσεις του.
- Correlated: εδώ οι τιμές που έχουν τα σημεία στις διαστάσεις τους είναι θετικά συσχετισμένες. Δηλαδή, σημεία που είναι καλά σε μία διάσταση είναι καλά και στις υπόλοιπες διαστάσεις και το αντίθετο. Συνήθως, όταν τα δεδομένα παράγονται με αυτή την κατανομή, τα σημεία που ανήκουν στο Skyline είναι πολύ λίγα.
- Anti-correlated: οι τιμές των διαστάσεων των σημείων είναι αρνητικά συσχετισμένες. Δηλαδή, τα σημεία που είναι καλά σε μία διάσταση τείνουν να μην είναι καλά στις υπόλοιπες. Όταν τα δεδομένα παράγονται με αυτή την κατανομή, τότε τα σημεία που ανήκουν στο Skyline είναι πάρα πολλά.

Το «καλά» στους παραπάνω ορισμούς εξαρτάται κάθε φορά από τις προτιμήσεις του χρήστη. Σε αυτή την εργασία, τα δεδομένα όλων των ομότιμων παράγονται από την ίδια κάθε φορά κατανομή. Γραφικές παραστάσεις και αναλυτική περιγραφή των κατανομών δίνεται στο Κεφάλαιο 6.

2.3. Bitmap

Ο αλγόριθμος που χρησιμοποιούμε για την εύρεση του Skyline ενός συνόλου σημείων έχει βασιστεί στον Bitmap αλγόριθμο που παρουσιάζεται στο [TaEO01]. Ο αλγόριθμος ονομάζεται έτσι επειδή χρησιμοποιεί ως είσοδο μία αναπαράσταση των δεδομένων ως ακολουθία από bit. Υποθέτουμε ότι τα δεδομένα είναι αποθηκευμένα ανά σημεία d διαστάσεων.

2.3.1 Bitmap Αναπαράσταση

Η αναπαράσταση αφορά και τις d διαστάσεις. Κάθε διάσταση i, $1 \le i \le d$ μπορεί να πάρει k_i διαφορετικές τιμές. Με v_{ij} συμβολίζουμε την j-οστή τιμή της i-οστής διάστασης. Υποθέτουμε ότι $v_{i1}>v_{12}>\ldots>v_{iki}$.

Ένα σημείο p (d_1, …, d_d) αναπαρίσταται ως ένα διάνυσμα των m-bit με τον τρόπο που ακολουθεί:

- η διάσταση d_i αναπαρίσταται από k_i bits. Άρα $m = \sum_{j=1}^{d} k_j$.
- το j-οστό bit αντιστοιχεί στην v_{ij}. Σημειώνουμε ότι επειδή ενδιαφερόμαστε για το MAX το πρώτο bit αντιστοιχεί στην v_{i1}, το δεύτερο στο v_{i2} κτλ. Αν το $p(d_i)$ είναι η v_{il}-οστή διαφορετική τιμή της διάστασης d_i, τότε τα k_i bits που αναπαριστούν το $p(d_i)$ τίθενται σε bit με τον τόπο που ακολουθεί: τα bits από 1 εώς l-1 τίθενται σε 0, και τα bits από l εώς k_i τίθενται σε 1.

Με απλά λόγια, για κάθε διάσταση υπάρχει ένας πίνακας από bit με γραμμές όσες και τα σημεία που έχουμε (τα οποία θέλουμε να ελέγξουμε αν ανήκουν στο Skyline) και στήλες όλες τις διαθέσιμες τιμές που μπορεί να υπάρχουν σε κάθε διάσταση. Αν το σημείο p στην διάσταση d_i έχει την τιμή j τότε τα bit από 1 έως j-1 τίθενται σε 0 και όλα τα υπόλοιπα σε 1. Κάθε πίνακας αποθηκεύεται ανά στήλη (bit slices).

Παραθέτουμε ένα παράδειγμα Bitmap αναπαράστασης, όπως αυτό αρχικά περιγράφηκε από τους [TaEO01].

Πίνακας 2.1 Αρχικά Σημεία

d1	d2	d3
1	1	2
3	2	1
4	1	1
2	3	2

Πίνακας 2.2 Η Bitmap αναπαράσταση των σημείων του Πίνακα 2.1

d1				d2			d3	
4	3	2	1	3	2	1	2	1
0	0	0	1	0	0	1	1	1
0	1	1	1	0	1	1	0	1
1	1	1	1	0	0	1	0	1
0	0	1	1	1	1	1	1	1

Στον Πίνακα 2.1 βλέπουμε τέσσερα σημεία τριών διαστάσεων. Η πρώτη διάσταση μπορεί να πάρει τέσσερις διαφορετικές τιμές, η δεύτερη τρεις, ενώ η τρίτη διάσταση μπορεί να πάρει δύο διαφορετικές τιμές. Στον Πίνακα 2.2 βλέπουμε την αναπαράσταση των τεσσάρων αυτών σημείων σε Bitmap μορφή. Για κάθε διάσταση υπάρχει ένας διαφορετικός Bitmap πίνακας. Υπάρχει μία στήλη σε κάθε πίνακα για κάθε πιθανή τιμή που μπορεί να πάρει η διάσταση αυτή. Π.χ. η διάσταση d1 έχει τέσσερις στήλες, ενώ η d2 έχει μόνο τρεις. Κάθε γραμμή αντιστοιχεί σε ένα σημείο. Έστω το σημείο (3, 2, 1) το οποίο εμφανίζεται δεύτερο στον Πίνακα 2.1. Στη διάσταση d1 το σημείο αυτό έχει τιμή 3. Έτσι τα bit των τιμών 3, 2 και 1 γίνονται 1 (άσσος), ενώ το bit που αντιστοιχεί στη διάσταση 4 παραμένει 0 (μηδέν). Αντίστοιχα στη διάσταση d2, τα bit των τιμών 2 και 1 γίνονται 1 ενώ στην τιμή 3 βάζουμε 0. Η ιδιότητα της αναπαράστασης αυτής είναι ότι πολύ εύκολα μπορούμε να δούμε τις τιμές για

κάθε διάσταση για τις οποίες το σημείο είναι το ίδιο καλό ή καλύτερο (είναι οι τιμές που έχουν άσσο).

2.3.2 Bitmap Αλγόριθμος Υπολογισμού Skyline

Σε αυτήν την παράγραφο περιγράφεται ο αλγόριθμος υπολογισμού του Skyline ενός συνόλου σημείων με βάση τη Bitmap αναπαράσταση που δόθηκε από τους [TaEO01].

Αλγόριθμος εύρεσης Skyline

Έστω ότι ενδιαφερόμαστε να δούμε αν το σημείο *p* ανήκει στο Skyline ενός συνόλου σημείων. Εκτελούμε τα παρακάτω βήματα:

- Παίρνουμε τις στήλες που αναφέρονται στις τιμές που έχει αυτό το σημείο για κάθε διάσταση και εκτελούμε την πράξη AND. Έστω ότι το αποτέλεσμα αποθηκεύεται στο A. Το A είναι μία ακολουθία από bit (ο αριθμός των οποίων ισούται με τον αριθμό όλων των σημείων που έχουμε), για τα οποία ισχύει η παρακάτω ιδιότητα: Το n-οστό bit είναι 1 αν και μόνο αν το n-οστό σημείο σε κάθε διάσταση έχει τιμή μεγαλύτερη ή ίση από την αντίστοιχη του σημείου p.
- Έπειτα παίρνουμε τις στήλες που αναφέρονται στις αμέσως μεγαλύτερες τιμές από αυτές που έχει το σημείο p σε κάθε διάσταση και εκτελούμε την πράξη OR. Αν δεν υπάρχει τέτοια στήλη θεωρούμε μία που έχει όλα τα bit ίσα με 0. Έστω ότι το αποτέλεσμα αποθηκεύεται στο B. Το B είναι μία ακολουθία από bit για τα οποία ισχύει η παρακάτω ιδιότητα: Το n-οστό bit είναι 1 αν και μόνο αν το n-οστό σημείο σε κάποια διάσταση έχει τιμή μεγαλύτερη από την αντίστοιχη του σημείου p.
- Τέλος εκτελούμε μία πράξη AND μεταξύ των A και B. Έστω C το αποτέλεσμα αυτής της πράξης. Από τις ιδιότητες των A και B προκύπτει ότι αν όλα τα bit του C είναι 0, τότε το σημείο p ανήκει σίγουρα στο Skyline.

Πρέπει να σημειώσουμε ότι αν κάποιος ενδιαφέρεται για ένα υποσύνολο των διαστάσεων τότε εκτελώντας τις παραπάνω πράξεις σε αυτό το υποσύνολο μπορεί να εξαχθεί το Skyline

των συγκεκριμένων μόνο διαστάσεων (άμεση εφαρμογή subspace Skyline Computation). Αυτό είναι πολύ εύκολο καθώς όπως αναφέρθηκε παραπάνω η αναπαράσταση αυτή αποθηκεύεται ανά στήλες. Ακολουθεί ένα παράδειγμα εφαρμογής του αλγόριθμου.

Παράδειγμα 1. Εύρεση Skyline με Bitmap αλγόριθμο

Έστω ότι θέλουμε να δούμε αν το σημείο (3, 2, 1) ανήκει στο Skyline του συνόλου του Πίνακα 2.1. Ακολουθώντας τον αλγόριθμο που περιγράφηκε παραπάνω ακολουθούμε τα παρακάτω βήματα:

- Βήμα 1: A= 0110 AND 0101 AND 1111 = 0100
- Βήμα 2: B= 0010 OR 0001 OR 1001 = 1011
- Βήμα 3: C= A AND B = 0100 AND 1011 = 0000

Το αποτέλεσμα (όλα μηδενικά) σημαίνει ότι το σημείο (3, 2, 1) ανήκει στο Skyline. Αντίθετα αν εκτελέσουμε την παραπάνω διαδικασία για το σημείο (1, 1, 2) έχουμε:

- Βήμα 1: A= 1111 AND 1111 AND 1001 = 1001
- Βήμα 2: B= 0111 OR 0101 OR 0000 = 0111
- Βήμα 3: C= A AND B = 1001 AND 0111 = 0001

Επειδή στο αποτέλεσμα δεν είναι όλα μηδέν, το σημείο (1, 1, 2) δεν ανήκει στο Skyline. Το ότι το τέταρτο bit είναι 1 σημαίνει ότι το τέταρτο σημείο (δηλαδή το (2, 3, 2)) έχει σε όλες του τις διαστάσεις τιμές μεγαλύτερες ή ίσες του (1, 1, 2) και σε τουλάχιστον μία τιμή μεγαλύτερη της αντίστοιχης του (1, 1, 2).

2.3.3 Χρήση Κάδων

Ο αλγόριθμος Bitmap υποθέτει ότι γνωρίζουμε όλες τις τιμές που μπορεί να πάρει μία διάσταση. Αυτό μπορεί να ισχύει για κάποιες εφαρμογές, όπως π.χ. στην περίπτωση των κατηγορικών ορισμάτων (για παράδειγμα ένα ξενοδοχείο στο πεδίο κατηγορία, γνωρίζουμε ότι μπορεί να έχει από ένα έως πέντε αστέρια), δεν είναι όμως αληθές όταν αναφερόμαστε σε

συνεχή πεδία όπως π.χ. η απόσταση η οποία μπορεί να πάρει άπειρες τιμές. Για τα τελευταία προβλήματα μία λύση που γενικά χρησιμοποιείται είναι να χωρίσουμε το διάστημα τιμών σε έναν αριθμό από κάδους. Σε κάθε κάδο ανατίθεται ένα εύρος τιμών, το οποίο του αντιστοιχεί. Πλέον όλες οι τιμές που έχουν αντιστοιχιστεί σε ένα κάδο θεωρούνται ίσες. Όσο αυξάνεται ο αριθμός των κάδων τόσο μεγαλύτερη είναι και η ακρίβεια του Skyline επειδή κάθε πιθανή τιμή τείνει να αντιστοιχιστεί με ένα μόνο κάδο.

Αν αυξήσουμε υπερβολικά το πλήθος των κάδων τότε χάνουμε μία από τις καλές ιδιότητες της Bitmap αναπαράστασης, η οποία είναι το μικρό μέγεθος της. Μπορούμε να κρατήσουμε μικρό τον αριθμό των κάδων, αλλά να μην έχουνε ίσο μέγεθος. Στη συγκεκριμένη περίπτωση που μας ενδιαφέρει το MAX για τον υπολογισμό του Skyline μπορούμε να έχουμε περισσότερους κάδους στις μεγαλύτερες τιμές, ενώ να έχουμε λιγότερους στις λίγες. Για παράδειγμα αν έχουμε 16 κάδους και σημεία στο (0, 10) μπορούμε να αναθέσουμε τις τιμές 8-10 στους 8 κάδους, και τις τιμές 0-8 στους υπόλοιπους 8. Την προσέγγιση αυτή την ονομάζουμε “heuristic buckets”.

Ο συγκεκριμένος διαμοιρασμός των κάδων εξετάστηκε πειραματικά και τα αποτελέσματα εμφανίζονται στο Κεφ. 6. Πράγματι, το πλήθος των σημείων που ανήκουν στο Skyline μειώνεται αρκετά για την Independent και Correlated κατανομή. Δε ισχύει όμως το ίδιο και στην Anti-Correlated κατανομή, για την οποία η περίπτωση με κάδους ίσου μεγέθους δουλεύει αρκετά καλύτερα. Αυτό συμβαίνει επειδή στην Anti-Correlated κατανομή τα σημεία τα οποία έχουν μεγάλη τιμή στη μία διάσταση, έχουν μικρή τιμή στην άλλη διάσταση. Έτσι για ένα σημείο αυτόματα χάνουμε την επιπλέον ακρίβεια την οποία είχαμε για την διάσταση με τη μεγάλη τιμή.

Ένας διαφορετικός τρόπος με τον οποίο μπορούν να κατανεμηθούν οι κάδοι στην Bitmap αναπαράσταση είναι να ακολουθούν λογαριθμική κατανομή. Δηλαδή το μέγεθος των κάδων να μειώνεται λογαριθμικά όσο πάμε από τις μεγαλύτερες στις μικρότερες τιμές των διαστάσεων. Με αυτόν τον τρόπο γίνεται πιο ομαλή μετάβαση στο μέγεθος των κάδων και έτσι πιθανότατα να έχουμε καλύτερα αποτελέσματα.

2.4. Extended Skyline

Το πρόβλημα που προκύπτει με τους κάδους όσον αφορά τον υπολογισμό του Skyline ενός συνόλου σημείων είναι ότι έχουμε false negatives. Με άλλα λόγια, λόγω της ύπαρξης των κάδων, κάποιες τιμές εξισώνονται και αποκλείονται από το Skyline σημεία τα οποία στην πραγματικότητα ανήκουν σε αυτό. Π.χ. έστω σημεία δύο διαστάσεων d_1, d_2. Έστω από αυτά, ότι θέλουμε να ελέγξουμε αν τα σημεία (7.56, 5.30) και (7.23, 6.71) ανήκουν στο Skyline. Αν έχουμε 10 κάδους και οι διαστάσεις παίρνουν τιμές στο διάστημα (0, 10) τότε: το d_1 θεωρείται ίδιο και για τα δύο σημεία, ενώ το d_2 μεγαλύτερο για το δεύτερο σημείο. Έτσι μόνο το δεύτερο σημείο ανήκει στο Skyline ενώ θα έπρεπε να ανήκουν και τα δύο.
Η λύση στο να μην έχουμε false negatives είναι αντί υπολογίζουμε το Skyline στο οποίο ανήκουν τα σημεία που είναι καλύτερα σε τουλάχιστον μία διάσταση, να υπολογίζουμε ένα άλλο σύνολο στο οποίο θα ανήκουν και τα σημεία που είναι ίσα σε τουλάχιστον μία διάσταση. Με αυτόν τον τρόπο αποφεύγουμε τα false negatives, έχουμε όμως false positives αφού κάποια από τα σημεία του νέου συνόλου δεν ανήκουν στο Skyline σύμφωνα με τον ορισμό. Υποθέτουμε ότι τα σημεία αυτά ενδιαφέρουν το χρήστη, αν το μέγεθος των κάδων είναι σχετικά μικρό. Αν υποθέσουμε το παράδειγμα με τα ξενοδοχεία, τότε αν το ξενοδοχείο Α με τιμή 50 ευρώ και απόσταση από την παραλία 200μ. ανήκει στο Skyline, τότε το ξενοδοχείο Β με τιμή 52 ευρώ και απόσταση 200μ. δεν θα ανήκει και δεν θα εμφανιστεί στο χρήστη. Πιθανόν όμως ο χρήστης να ενδιαφέρεται για το ξενοδοχείο Β παρόλο που είναι λίγο πιο ακριβό (για παράδειγμα επειδή γνωρίζει ότι έχει καλύτερη εξυπηρέτηση). Αν έχουμε false positives το ξενοδοχείο Β θα εμφανιστεί στο χρήστη αν έχουμε κάδο στον οποίο ανήκουν οι τιμές από 50 έως 60 ευρώ.

Αυτό που απομένει λοιπόν είναι να ορίσουμε ένα νέο σύνολο το οποίο θα υπολογίζουμε. Το σύνολο αυτό θα έχει την ιδιότητα που προαναφέρθηκε: θα ανήκουν όλα τα σημεία τα οποία είναι καλύτερα ή ίσα σε τουλάχιστον μία διάσταση.

Το extended Skyline ενός συνόλου σημείων ορίζεται ως τα σημεία που δεν ext - κυριαρχούνται (not ext-dominated) από κανένα άλλο σημείο. Ο ορισμός της κυριαρχίας εδώ είναι διαφορετικός από τον ορισμό κυριαρχίας του απλού Skyline. Ένα σημείο *p* ext-

κυριαρχεί ενός άλλου σημείου *q* αν το *p* είναι καλύτερο από το *q* σε όλες τις διαστάσεις που μας ενδιαφέρουν. Δίνεται ο ορισμός όπως δόθηκε από τους [VDKV07]:

Ορισμός 2.4 Extended Skyline

Για κάθε σύνολο U οποιασδήποτε διάστασης, το *p* κάνει ext-dominate το *q* αν για κάθε διάσταση d_i του U, ισχύει ότι $p(d_i)>q(d_i)$. Το extended Skyline είναι το σύνολο των σημείων που δεν γίνονται ext-dominate από κανένα άλλο.

Με άλλα λόγια, δύο σημεία ανήκουν στο extended Skyline ενός συνόλου σημείων αν είναι καλύτερα σε όλες τις διαστάσεις σε σχέση με τα σημεία που δεν ανήκουν στο extended Skyline και μεταξύ τους είτε ισχύει ο ορισμός κυριαρχίας του κανονικού Skyline είτε έχουν ίδια τιμή σε τουλάχιστον μία διάσταση.

Ιδιότητα 2.1

Κάθε σημείο που ανήκει στο Skyline ενός συνόλου διαστάσεων U ανήκει και στο extended Skyline του U, δηλ. $SKY_U \subseteq ext - SKY_U$.

Ιδιότητα 2.2

Κάθε σημείο που ανήκει στο Skyline ενός υποσυνόλου των διαστάσεων V, $V \subseteq U$, ανήκει και στο extended Skyline του U, δηλ. $SKY_V \subseteq ext - SKY_U, V \subseteq U$.

Με απλά λόγια, το extended Skyline όλων των διαστάσεων είναι τέτοιο ώστε για οποιοδήποτε υποσύνολο των διαστάσεων να μπορεί να υπολογιστεί το Skyline.

2.4.1 Bitmap Αλγόριθμος Υπολογισμού Extended Skyline

Στην παράγραφο αυτή τροποποιούμε τον Bitmap αλγόριθμο έτσι ώστε να υπολογίζει το extended Skyline ενός συνόλου σημείων, όπως αυτό ορίστηκε στην προηγούμενη παράγραφο.

Αλγόριθμος εύρεσης extended Skyline

Για να βρούμε το extended Skyline αφήνουμε ίδια την Bitmap αναπαράσταση, τροποποιούμε όμως τον Bitmap αλγόριθμο. Πλέον απαιτείται μόνο ένα βήμα για κάθε σημείο p:

- Παίρνουμε τις στήλες που αναφέρονται στις αμέσως μεγαλύτερες τιμές από αυτές που έχει το σημείο p σε κάθε διάσταση και εκτελούμε την πράξη AND. Αν δεν υπάρχει τέτοια στήλη θεωρούμε μία που έχει όλα τα bit ίσα με 0. Έστω ότι το αποτέλεσμα αποθηκεύεται στο C. Το C είναι μία ακολουθία από bit για τα οποία ισχύει η παρακάτω ιδιότητα: Το n-οστό bit είναι 1 αν και μόνο αν το n-οστό σημείο έχει σε κάθε διάσταση τιμή μεγαλύτερη από την αντίστοιχη του σημείου p.
- Αν όλα τα bit του C είναι μηδενικά, τότε το σημείο ανήκει στο extended Skyline, αλλιώς αν υπάρχει έστω και ένας άσσος, το σημείο δεν ανήκει στο extended Skyline.

Ακολουθεί ένα παράδειγμα εφαρμογής του Bitmap αλγόριθμου για υπολογισμό του extended Skyline.

Παράδειγμα 2. Εύρεση extended Skyline με Bitmap αλγόριθμο

Για να βρούμε αν ένα σημείο ανήκει στο extended Skyline ακολουθούμε τον αλγόριθμο που περιγράφηκε για αυτήν την περίπτωση. Έστω ότι ελέγχουμε πάλι για το σημείο (3, 2, 1). Τότε:

- Βήμα 1: C= 0010 AND 0001 AND 1001 = 0000

Το αποτέλεσμα σημαίνει ότι το σημείο (3, 2, 1) ανήκει στο extended Skyline. Το αποτέλεσμα είναι σωστό διότι το Skyline ενός συνόλου σημείων είναι υποσύνολο του extended Skyline. Εκτελούμε την ίδια διαδικασία και για το σημείο (1, 1, 2):

- Βήμα 1: C= 0111 AND 0101 AND 0000 = 0000

Όπως παρατηρούμε και το σημείο (1, 1, 2) ανήκει στο extended Skyline παρόλο που δεν ανήκει στο Skyline. Αυτό συμβαίνει επειδή το (1, 1, 2) έχει στην τρίτη διάσταση ίδια τιμή με ένα σημείο το οποίο ανήκει στο Skyline και αυτό είναι το (2, 3, 2).

2.4.1.1. Βελτίωση του Bitmap Αλγόριθμου Υπολογισμού του Extended Skyline

Ο αλγόριθμος που μόλις περιγράφηκε είναι αρκετά απλός και πολύ γρήγορος στην εξαγωγή αρχικών αποτελεσμάτων. Το κύριο μειονέκτημα του είναι ότι για να εξαχθεί το τελικό extended Skyline θα πρέπει ο έλεγχος να γίνει για ένα προς ένα όλα τα σημεία με αποτέλεσμα να είναι αργός στην παραγωγή του τελικού αποτελέσματος. Αν εκμεταλλευτούμε πλήρως τον ορισμό του extended Skyline ενός συνόλου σημείων μπορούμε να έχουμε καλύτερες αποδόσεις.

Ας υποθέσουμε ότι χρησιμοποιείται ο παραπάνω αλγόριθμος και βρίσκουμε ένα σημείο *p* το οποίο ανήκει στο extended Skyline. Τότε, σύμφωνα με τον ορισμό, α) στο extended Skyline θα ανήκουν και όλα τα σημεία τα οποία έχουν τιμή σε τουλάχιστον μία διάσταση μεγαλύτερη με την αντίστοιχη τιμή του *p*, β) τα σημεία που έχουν σε όλες τις διαστάσεις τιμές μικρότερες του *p* είναι σίγουρο ότι δεν ανήκουν στο extended Skyline. Το συμπέρασμα είναι ότι δεν χρειάζεται να γίνει κανένας έλεγχος για τα σημεία που ικανοποιούν την ιδιότητα α), και άρα μπορούν να εισαχθούν απευθείας στο extended Skyline. Επίσης δεν χρειάζεται να γίνει κανένας έλεγχος για τα σημεία που έχουν την ιδιότητα β) και έτσι αυτά μπορούν απευθείας να αποκλειστούν από το extended Skyline. Το ζήτημα λοιπόν είναι να βρεθεί ένας αποδοτικός τρόπος υπολογισμού των σημείων που ικανοποιούν το α) και αυτών που ικανοποιούν το β).

Εφόσον ένα σημείο *p* ανήκει στο extended Skyline, εκτελούμε δύο επιπλέον βήματα που βασίζονται στη Bitmap αναπαράσταση των δεδομένων:

- Παίρνουμε τις στήλες που αναφέρονται στις τιμές που έχει το σημείο *p* σε κάθε διάσταση και εκτελούμε την πράξη OR. Έστω ότι το αποτέλεσμα αποθηκεύεται στο C. Το C είναι μία ακολουθία από bit για τα οποία ισχύει η παρακάτω ιδιότητα: Το n-οστό bit είναι 1 αν και μόνο αν το n-οστό σημείο έχει σε τουλάχιστον μία διάσταση τιμή μεγαλύτερη ή ίση από την αντίστοιχη του σημείου *p*. Αντίστοιχα το n-οστό bit είναι 0 αν και μόνο αν το n-οστό σημείο έχει μικρότερες τιμές από το *p* σε όλες του τις διαστάσεις. Όλα τα σημεία για τα οποία η τιμή του C είναι 0 μπορούν απευθείας να αποκλειστούν από το extended Skyline. Για τα υπόλοιπα θα πρέπει να γίνει εκ νέου έλεγχος.

- Παίρνουμε τις στήλες που αναφέρονται στις αμέσως μεγαλύτερες τιμές από αυτές που έχει το σημείο *p* σε κάθε διάσταση και εκτελούμε την πράξη OR. Έστω ότι το αποτέλεσμα αποθηκεύεται στο D. Το D είναι μία ακολουθία από bit για τα οποία ισχύει η παρακάτω ιδιότητα: Το n-οστό bit είναι 1 αν και μόνο αν το n-οστό σημείο έχει σε τουλάχιστον μία διάσταση τιμή μεγαλύτερη από την αντίστοιχη του σημείου *p*. Όλα τα σημεία για τα οποία η τιμή του D είναι 1 είναι σίγουρο ότι ανήκουν στο extended Skyline και μπορούμε απευθείας να τα εισάγουμε σε αυτό.

Η επιτυχία της μεθόδου αυτής εξαρτάται από δύο παράγοντες: α) από το πόσο γρήγορα θα βρούμε το πρώτο σημείο το οποίο ανήκει στο extended Skyline, β) από το πόσο μεγάλες τιμές θα έχει αυτό το σημείο σε όλες του τις διαστάσεις. Αν βρεθεί γρήγορα ένα σημείο το οποίο ανήκει στο extended Skyline, τότε μπορούμε να εφαρμόσουμε την παραπάνω τεχνική, ενώ αν το σημείο που θα βρεθεί έχει μεγάλες τιμές σε όλες τις διαστάσεις, τότε μπορούν χωρίς έλεγχο να αποκλειστούν αρκετά σημεία.

Αν το σύνολο των σημείων που έχουμε είναι πολύ μεγάλο τότε συμφέρει να βρούμε εξ αρχής κάποιο σημείο που ανήκει στο extended Skyline.

- Επιλέγουμε τυχαία μία διάσταση.
- Από τη διάσταση αυτή επιλέγουμε τη στήλη που αντιστοιχεί στη μεγαλύτερη τιμή. Από τη στήλη επιλέγουμε έναν άσσο ο οποίος έστω ότι βρίσκεται στη θέση n. Επιλέγουμε λοιπόν το n-οστό στοιχείο για να κάνουμε τον πρώτο έλεγχο για το αν ανήκει στο extended Skyline. Το σημείο αυτό θα ανήκει σίγουρα στο extended Skyline αφού έχει τη μεγαλύτερη τιμή σε τουλάχιστον μία διάσταση.

Όταν εφαρμόσουμε σε αυτό το σημείο τη βελτίωση που περιγράφηκε, κάποια σημεία θα αποκλειστούν. Επίσης κάποια σημεία θα είναι σίγουρο ότι ανήκουν στο extended Skyline. Έπειτα συνεχίζουμε τον έλεγχο για τα σημεία που δεν γνωρίζουμε ακόμα αν ανήκουν ή όχι στο extended Skyline, με τον ίδιο τρόπο που το κάναμε και για αυτό το σημείο.

Στον αλγόριθμο που περιγράφηκε μπορεί να γίνει μία επιπλέον βελτίωση για πολύ γρήγορη εξαγωγή αρχικών αποτελεσμάτων χρησιμοποιώντας την παρακάτω ιδιότητα: Κάθε σημείο το

οποίο έχει σε τουλάχιστον μία διάσταση την μεγαλύτερη δυνατή τιμή ανήκει σίγουρα στο extended Skyline. Πριν την εκτέλεση του αλγόριθμου εκτελούμε τα παρακάτω τρία βήματα:

- Επιλέγουμε τις στήλες που αντιστοιχούν στις μεγαλύτερες τιμές των διαστάσεων (ή στον κάδο που αντιστοιχεί στις τιμές αυτές)
- Ελέγχουμε ποια σημεία έχουν άσσο σε αυτές τις στήλες. Τα σημεία αυτά μπορούμε να τα εμφανίσουμε απευθείας στο χρήστη.
- Από τα σημεία αυτά επιλέγω αυτό το οποίο έχει τους περισσότερους άσσους και εκτελώ για αυτό την βελτίωση που περιγράφηκε παραπάνω για αποκλεισμό σημείων. Το σημείο αυτό είναι πιθανό να αποκλείσει πολλά σημεία, λόγω του ότι θα έχει μεγάλες τιμές σε πολλές διαστάσεις.

Τα δύο πρώτα βήματα αναμένεται να δουλέψουν πολύ καλά σε όλες τις κατανομές των δεδομένων. Ειδικά στην Anti-Correlated κατανομή θα έχουμε σίγουρα πολλά σημεία που ικανοποιούν αυτή την ιδιότητα και μπορούν να εμφανιστούν στο χρήστη με ελάχιστους μόνο υπολογισμούς. Στην Correlated κατανομή θα δουλέψει πολύ καλά το τρίτο βήμα της βελτίωσης. Επειδή τα δεδομένα είναι θετικά συσχετισμένα υπάρχει τουλάχιστον ένα σημείο που έχει πολύ μεγάλες τιμές σε όλες τις διαστάσεις. Το σημείο αυτό θα αποκλείσει πάρα πολλά σημεία τα οποία θα έχουν μικρότερες από αυτό τιμές σε όλες τις διαστάσεις.

ΚΕΦΑΛΑΙΟ 3. BITPEER

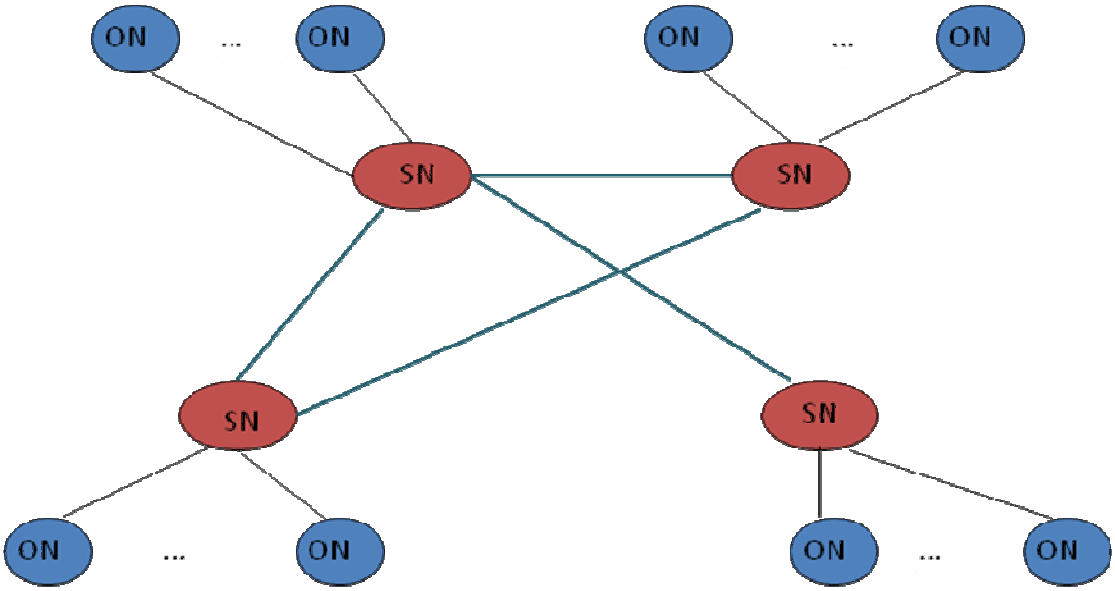
Σε αυτό το κεφάλαιο περιγράφεται η προσέγγιση αυτής της εργασίας την οποία έχουμε ονομάσει BitPeer. Αρχικά ορίζεται η τοπολογία του δικτύου ομότιμων στην οποία επιλύουμε το πρόβλημα. Περιγράφεται το πώς δρομολογείται μία Skyline ερώτηση στο σύστημα μας, έτσι ώστε να γίνει ο τελικός υπολογισμός και δίνεται ένα παράδειγμα δρομολόγησης. Έπειτα εξετάζεται ο τρόπος με τον οποίο γίνονται ενημερώσεις σε ένα τέτοιο σύστημα και αναλύονται οι παράγοντες που επηρεάζουν το κόστος μίας ενημέρωσης. Στη συνέχεια δίνεται ένας τρόπος ενημέρωσης για ένα σύστημα με διαφορετική κατανομή των δεδομένων στους κόμβους. Τέλος γίνεται μία σύγκριση της BitPeer προσέγγισης σε σχέση με μεθόδους που χρησιμοποιούν Index.

3.1. Τοπολογία

Σκοπός αυτής της ενότητας είναι να οριστεί το σύστημα ομότιμων στο οποίο εξετάζουμε το πρόβλημα υπολογισμού ερωτήσεων Skyline. Οι κύριες κατηγορίες στις οποίες χωρίζονται τα μη κεντρικοποιημένα συστήματα ομότιμων σύμφωνα με την αρχιτεκτονική τους είναι δύο: δομημένα και αδόμητα. Στα δομημένα συστήματα υπάρχει μία προκαθορισμένη δομή η οποία

πρέπει να τηρείται. Κάθε κόμβος που θέλει να συμμετέχει στο δίκτυο πρέπει να εισαχθεί σε συγκεκριμένο σημείο του δικτύου έτσι ώστε να μην καταστρέφει τη δομή που έχει οριστεί. Αντίθετα τα αδόμητα συστήματα οι κόμβοι συνδέονται με τυχαίο τρόπο (ή σχεδόν με τυχαίο) ακολουθώντας κάποιους χαλαρούς κανόνες. Τα συστήματα αυτά δεν εγγυώνται για την επιτυχία μιας αναζήτησης. Παρόλο που όλοι οι κόμβοι θεωρούνται ίσοι σε αυτά τα συστήματα, κάτι τέτοιο δεν ισχύει. Κάποιοι από τους κόμβους έχουν μεγαλύτερη διαθεσιμότητα, υπολογιστική και αποθηκευτική ισχύ και πιθανότατα πιο γρήγορη σύνδεση δικτύου. Όταν ενδιαφερόμαστε για τον υπολογισμό ενός Skyline ερωτήματος, αυτοί οι κόμβοι μπορούν να αναλάβουν περισσότερους υπολογισμούς από τους υπόλοιπους έτσι ώστε να γίνει πιο αποδοτικός ο συνολικός υπολογισμός του Skyline των δεδομένων του δικτύου. Για να μπορέσουμε να εκμεταλλευτούμε την επιπλέον ισχύ των κόμβων αυτών δεν αρκούν οι αρχιτεκτονικές που μόλις περιγράψαμε.

Το δίκτυο το οποίο υποθέτουμε για αυτήν την εργασία είναι ένα αδόμητο P2P σύστημα αποτελούμενο από N ομότιμους. Κάποιοι από αυτούς τους ομότιμους έχουν επιπλέον αρμοδιότητες λόγω των επιπλέον ικανοτήτων τους. Αυτοί οι ομότιμοι ονομάζονται Super Nodes (SNs) και αποτελούν ένα μικρό μόνο μέρος των κόμβων του δικτύου. Οι υπόλοιποι ομότιμοι ονομάζονται Ordinary Nodes (ONs). Κάθε ON ανήκει σε έναν SN, δηλαδή κάθε ON διατηρεί μια και μοναδική σύνδεση με τον SN στον οποίο ανήκει. Αντίθετα οι SN είναι συνδεδεμένοι με πολλούς ON αλλά και με αρκετούς SN. Η συνδεσμολογία μεταξύ των SN μπορεί να ποικίλει ανάλογα με τις ανάγκες της εφαρμογής. Μία περίπτωση είναι να υπάρχει μεταξύ τους πλήρης συνδεσιμότητα ή για παράδειγμα να υπάρχει μεταξύ τους τυχαία συνδεσμολογία. Επιλέγουμε την γενική περίπτωση στην οποία το δίκτυο μεταξύ των SN είναι αδόμητο σύστημα ομότιμων, το οποίο δημιουργείται με τυχαία συνδεσμολογία. Θεωρούμε δεδομένη την τυχαία συνδεσμολογία μεταξύ των SN για την επίλυση του προβλήματος και δεν εξετάζουμε το πώς αυτή θα επηρέαζε το πρόβλημα αν ήταν διαφορετική. Σημειώνουμε ότι οι αλγόριθμοι που προτείνονται σε αυτή την εργασία δουλεύουν ανεξάρτητα της συνδεσμολογίας μεταξύ των SN. Κάθε φορά αλλάζει η δρομολόγηση των ερωτήσεων λόγω του ότι το δίκτυο είναι διαφορετικό όμως το αποτέλεσμα παραμένει σωστό.



Σχήμα 3.1 Η Τοπολογία του Δικτύου.

Κάθε κόμβος διατηρεί τα δεδομένα του σε Bitmap μορφή. Οι SN διατηρούν το extended Skyline όλων των ON κόμβων με τους οποίους συνδέονται. Οι ON μόλις συνδεθούν στο δίκτυο στέλνουν τα δεδομένα τους στον SN με τον οποίο συνδέθηκαν και αυτός κάνει τους υπολογισμούς. Σε επόμενη παράγραφο περιγράφεται η περίπτωση που οι κόμβοι έχουν ενημερώσεις στα δεδομένα τους. Ο SN χρησιμοποιεί το extended Skyline έτσι ώστε όταν λάβει ένα Skyline ερώτημα οποιουδήποτε υποσυνόλου των διαστάσεων, να μπορεί να απαντήσει χωρίς να χρειαστεί να προωθήσει το μήνυμα σε όλους τους ON που είναι συνδεδεμένοι με αυτόν.

Όπως γνωρίζουμε τα P2P συστήματα χαρακτηρίζονται για τις άναρχες εισόδους και εξόδους των κόμβων στο δίκτυο. Στη συνέχεια περιγράφεται συνοπτικά ο τρόπος με τον οποίο εκτελούνται αυτές οι ενέργειες. Όταν ένας SN φύγει από το δίκτυο τότε ενημερώνει τα παιδιά του και αυτά επαναλαμβάνουν εκ νέου την διαδικασία εισόδου στο δίκτυο. Όταν ένας ON αποφασίσει να φύγει από το δίκτυο τότε ενημερώνει τον πατέρα του για την απόφασή του και ο πατέρας του απλά ενημερώνει την πληροφορία που διατηρεί για τα δεδομένα των παιδιών του.

Στην περίπτωση κατά την οποία ένας κόμβος θέλει να εισέλθει στο δίκτυο αυτός θα μπει αρχικά ως ON. Ακόμη πρέπει -για να μπει επιτυχώς- να γνωρίζει τουλάχιστον έναν κόμβο που εκείνη τη στιγμή είναι στο δίκτυο. Για να πετύχουμε καλό load balance ο κόμβος που θέλει να εισέλθει αρχικοποιεί μια προσωρινή επικοινωνία με κάποιους SN του δικτύου, με τη βοήθεια του ενός τουλάχιστον κόμβου που γνωρίζει, και επιλέγει τελικά να συνδεθεί με τον SN που έχει τα λιγότερα παιδιά. Για τη διατήρηση της τοπολογίας μετά από συνεχείς εισόδους και εξόδους εκτελείται μια περιοδική ρουτίνα ώστε αν σε κάποιους SNs έχουν ανατεθεί ON οι οποίοι να ξεπερνούν ένα άνω όριο τότε αποφασίζει και μετατρέπει έναν ή περισσότερους ONs σε SNs ώστε να εξισορροπηθεί το φορτίο του δικτύου. Αντίστοιχα υπάρχει και κάτω όριο στον αριθμό των ON που ένας SN μπορεί να έχει. Αν ο αριθμός των ON πέσει κάτω από αυτό το όριο τότε ο SN αποσυνδέεται από το δίκτυο και επανασυνδέεται ως ON σε κάποιον άλλο SN.

Υπάρχει επίσης το ενδεχόμενο κατάρρευσης ενός κόμβου. Αν ο κόμβος αυτός είναι ON, τότε ο πατέρας του (SN) θα πρέπει να διαγράψει την πληροφορία που διατηρεί για αυτόν για να μην υπάρχει ασυνέπεια στο σύστημα. Υπάρχει μία περιοδική ρουτίνα και για αυτήν την λειτουργία. Τα προβλήματα είναι σοβαρότερα αν αποτύχει ένας SN. Τότε όλα τα παιδιά του (ON) βρίσκονται εκτός δικτύου και όταν το αντιληφθούν πρέπει να αρχίσουν μόνα τους την είσοδο στο δίκτυο.

3.2. Δρομολόγηση και Υπολογισμός Ερώτησης

Έστω ότι ένας οποιοσδήποτε ομότιμος του συστήματος (SN ή ON) θέλει να υπολογίσει το Skyline του συνόλου των σημείων που βρίσκονται εκείνη τη στιγμή στο σύστημα. Για να λάβει μία απάντηση, θα πρέπει να επικοινωνήσει με τους γείτονές του και αυτοί αντίστοιχα με τους δικούς τους, έτσι ώστε να έχουμε αποτέλεσμα που να περιλαμβάνει όλα τα σημεία του συστήματος ομότιμων. Για αποδοτικότητα θέλουμε επίσης κάθε σημείο να περιλαμβάνεται μόνο μία φορά στον υπολογισμό, δηλαδή να ρωτάμε μία μόνο φορά κάθε ομότιμο και να μη γίνονται κύκλοι στο δίκτυο. Οι SN έχουν υπολογισμένο το extended

Skyline όλων των ON τους και έτσι δεν προωθούν το ερώτημα στους ON, αφού μπορούν οι ίδιοι να απαντήσουν σε οποιοδήποτε Skyline ερώτημα τους τεθεί. Με άλλα λόγια, δεν υπάρχει δρομολόγηση του ερωτήματος μεταξύ ON και SN, αλλά μόνο στο δίκτυο στο οποίο συνδέονται οι SN μεταξύ τους. Αν ένας ON θέλει να θέσει ένα Skyline ερώτημα, ενημερώνει τον SN με τον οποίο συνδέεται και αυτός το δρομολογεί στο δίκτυο των SN.

Στον αλγόριθμο δρομολόγησης που προτείνεται παρακάτω πρέπει να αντιμετωπιστούν και τα εξής: α) κύκλοι, β) ασύγχρονη λειτουργία, γ) αποτυχίες κόμβων. Για αποφυγή των κύκλων κάθε ερώτηση έχει το δικό της μοναδικό id. Αν ένας κόμβος λάβει την ερώτηση με το ίδιο id για δεύτερη φορά, την αγνοεί. Οι αποτυχίες κόμβων αντιμετωπίζονται με έναν χρόνο αναμονής t1. Δηλαδή, ένας κόμβος περιμένει να του αποκριθεί ένας άλλος το πολύ χρόνο t1. Η ασυγχρονία του συστήματος αντιμετωπίζεται με τον ίδιο χρόνο αναμονής t1. Ορίζουμε λοιπόν το t1 ως το χρόνο αναμονής απόκρισης ενός κόμβου.

Παρακάτω περιγράφεται ο αλγόριθμος δρομολόγησης που προτείνουμε. Ο αλγόριθμος αυτός πετυχαίνει τους παραπάνω στόχους. Υποθέτουμε ότι οι κόμβοι είναι ενημερωμένοι για το αν κάποιοι γείτονες τους δεν συμμετέχουν πια στο σύστημα ομότιμων.

Ξεχωρίζουμε τις εξής τρεις περιπτώσεις:

1 Ένας SN θέτει ένα ερώτημα

2 Ένας ON θέτει ένα ερώτημα

3 Ένας SN λαμβάνει ένα ερώτημα

Στην 1η περίπτωση ο SN προωθεί το μήνυμα σε όλους τους SN τους οποίους γνωρίζει. Έπειτα υπολογίζει το τοπικό αποτέλεσμα από το extended Skyline το οποίο έχει αποθηκευμένο. Κάθε φορά που λαμβάνει ένα αποτέλεσμα από έναν γείτονα του, το συνυπολογίζει στο τελικό Skyline. Όταν λάβει αποτέλεσμα από όλους τους γείτονες του, τότε το Skyline ερώτημα έχει απαντηθεί. Αν δεν λάβει αποτελέσματα μέσα σε κάποιο χρονικό διάστημα t1 τότε ο SN υπολογίζει μόνο τα αποτελέσματα που έχει ήδη λάβει (ο κόμβος που δεν έστειλε αποτέλεσμα είτε έχει πέσει, είτε έχει δεχθεί το ερώτημα πρωτύτερα από κάποιον άλλον κόμβο του δικτύου μέσω του οποίου θα λάβουμε έτσι κι αλλιώς τα αποτελέσματα).

Επισημαίνουμε πως οι γείτονες του, στέλνουν το αποτέλεσμα σε Bitmap μορφή, έτσι ώστε να είναι δυνατόν να χρησιμοποιηθεί ο Bitmap αλγόριθμος για τον υπολογισμό του Skyline.

Στην 2η περίπτωση ο ON που θέτει το ερώτημα προωθεί την ερώτηση στον SN με τον οποίο είναι συνδεδεμένος. Πλέον ο SN είναι υπεύθυνος για την απάντηση του ερωτήματος. Έπειτα περιμένει ώσπου ο SN να του στείλει τα αποτελέσματα της ερώτησης. Αν δεν λάβει αποτελέσματα μέσα σε κάποιο χρονικό διάστημα t1 τότε ο κόμβος ελέγχει αν ο SN έχει πέσει και είτε θέτει ξανά το ερώτημα, είτε πρέπει να επανασυνδεθεί στο δίκτυο και να θέσει ξανά την ερώτηση στον νέο κόμβο που θα συνδεθεί.

Στην 3η περίπτωση ο SN ελέγχει αν έχει ξαναδεχθεί ερώτημα με ίδιο id. Αν ναι, τότε απλά το αγνοεί. Αν το ερώτημα είναι το πρώτο με αυτό το id τότε ο SN συμπεριφέρεται ακριβώς όπως στην 1η περίπτωση με τη διαφορά ότι α) όταν ρωτά τους γείτονες του, δεν προωθεί το ερώτημα στον κόμβο ο οποίος τον ρώτησε και β) στο τέλος στέλνει τα αποτελέσματα του Skyline στον κόμβος ο οποίος των ρώτησε. Ο έλεγχος για το αν έχει λάβει ξανά το ερώτημα γίνεται για να μην υπάρχουν κύκλοι.

Παρακάτω δίνονται οι αντίστοιχοι αλγόριθμοι. Ο Αλγόριθμος 3.1 περιγράφει την περίπτωση που ένας κόμβος θέτει ένα ερώτημα. Λαμβάνει ως ορίσματα το είδος του κόμβου που θέτει το ερώτημα, το ερώτημα και το αναγνωριστικό του. Επίσης λαμβάνει ως όρισμα έναν κόμβο P. Ο κόμβος αυτός είναι Null στην περίπτωση που η συνάρτηση καλείται από τον πρώτο κόμβο που αποφασίζει να θέσει το ερώτημα. Αν όμως ο κόμβος έχει δεχθεί το ερώτημα από έναν κόμβο P1 και καλέσει την PoseQuery για να ρωτήσει τους γείτονες του, ο κόμβος P είναι ίσος με P1 έτσι ώστε το ερώτημα να μην σταλεί πάλι πίσω σε αυτόν που ήδη έχει ρωτήσει. Ο Αλγόριθμος 3.2 περιγράφει την περίπτωση που ένας κόμβος λαμβάνει ένα ερώτημα.

Αλγόριθμος 3.1 Pose_Query

Input: node, Query, QueryId, P: NULL if it is the first node that poses the query
Output: FinalSkyline

```
if node==ON then
    Send Query to SN;                                   //θέτει το ερώτημα στον SN
    Wait(t1);                                           //περιμένει
    S=collectAnswers();                                 //λαμβάνει τις απαντήσεις
    FinalSkyline=Bitmap(Query, S);                      //και υπολογίζει το τελικό Skyline
else if node==SN then
    Send Query to Neighbor[node]-P;      //στέλνει το ερώτημα στους γείτονες του
    LocalSkyline=Bitmap(Query, extended Skyline)      //υπολογίζει το τοπικό
    Wait(t1);                                                   //περιμένει
    S=collectAnswers();                                 //λαμβάνει τις απαντήσεις από όλους
    FinalSkyline=Bitmap(Query, S, LocalSkyline);     //και υπολογίζει
                                                                //το τελικό Skyline
end if
return FinalSkyline;
```

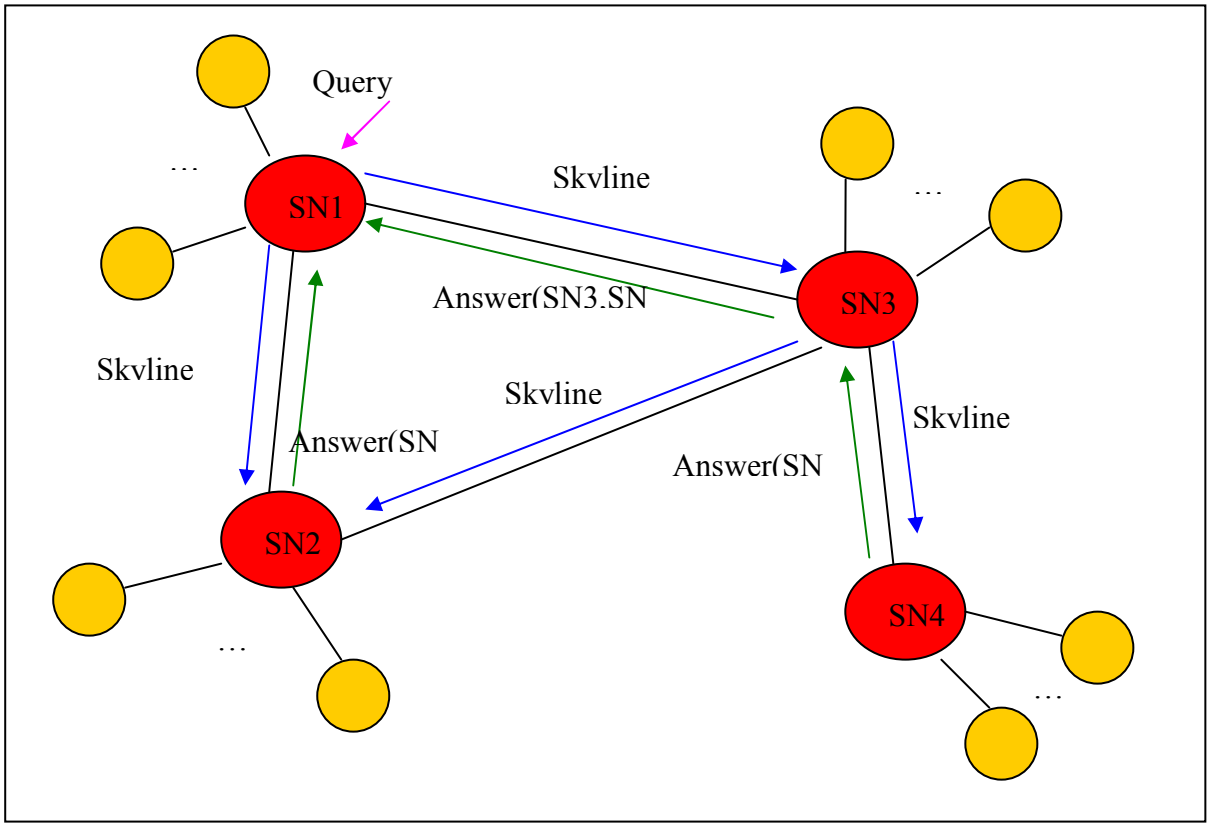
Αλγόριθμος 3.2 Receive_Query from P

Input: node, Query, QueryId, P
Output: PartialSkyline

```
if node==SN then                                        //αν είναι SN
    if QueryId is the first then       //αν δεν έχει λάβει την ερώτηση με αυτό το id
        PartialSkyline=PoseQuery(SN, Query, QueryId, P);
        Send PartialSkyline to P;                //στέλνει το αποτέλεσμα στον P
    end if
end if
```

Παράδειγμα Δρομολόγησης

Ο αλγόριθμος που δόθηκε στην προηγούμενη παράγραφο δίνεται σχηματικά στο Σχήμα 3.2. Έχουμε ένα σύστημα ομότιμων με τέσσερις SN στον καθένα από τους οποίους αντιστοιχεί ένα σύνολο από ON. Ο SN1 συνδέεται με τους SN2 και SN3, ενώ ο SN4 συνδέεται μόνο με τον SN3. Έστω ότι ο SN1 θέλει να θέσει ένα Skyline ερώτημα. Ρωτάει όλους τους γείτονες του (δηλαδή τους SN2 και SN3) και περιμένει απαντήσεις. Ο SN2 λαμβάνει το ερώτημα και αφού δεν έχει γείτονες να το προωθήσει στέλνει τα αποτελέσματα στον SN1. Το αποτέλεσμα το υπολογίζει από το extended Skyline που ο ίδιος έχει υπολογισμένο και περιλαμβάνει και τα δεδομένα όλων των ON του. Για αυτό δεν χρειάζεται να ρωτήσει κανέναν από τους ON του. Ο SN3 λαμβάνει και αυτός το ερώτημα από τον SN1 και ομοίως το προωθεί στους γείτονες του SN2 και SN4. Ο SN2 έχει ήδη απαντήσει στην ερώτηση το συγκεκριμένο id και έτσι δεν του στέλνει απαντήσεις. Ο SN4 υπολογίζει το extended Skyline του και του απαντά. Όταν ο SN3 λάβει αποτελέσματα από τον SN4 και λήξει ο χρόνος που περιμένει αποτελέσματα από τον SN2, συνυπολογίζει τα δικά του αποτελέσματα με αυτά του SN4 και τα στέλνει στον SN1. Ο SN1 αφού έλαβε αποτελέσματα από όλους τους γείτονες του υπολογίζει το τελικό αποτέλεσμα. Αφού υπολογίσει το συνολικό αποτέλεσμα στέλνει μήνυμα στους κόμβους που έχουν τα δεδομένα του τελικού αποτελέσματος και ζητά να του τα στείλουν για να τα εμφανίσει στο χρήστη. Αυτό το κάνει επειδή έχει την Bitmap αναπαράσταση των δεδομένων και όχι τα πραγματικά δεδομένα. Αν στην αναπαράσταση αυτή έχουμε έναν κάδο για κάθε τιμή, τότε το τελευταίο βήμα παραλείπεται.



Σχήμα 3.2 Δρομολόγηση Skyline Ερωτήσεων

3.3. Ενημερώσεις

Η ενημέρωση των ON επαφίεται στον κάθε ομότιμο ξεχωριστά και γίνεται σε batches. Δηλαδή δεν ενημερώνει τον SN κάθε φορά που διαγράφεται ή εισάγεται κάποιο δεδομένο, αλλά κάθε φορά που τροποποιείται το u% των δεδομένων. Αυτό γίνεται για να μην έχουμε συμφόρηση στο δίκτυο λόγω των ενημερώσεων. Η συχνότητα με την οποία γίνονται οι ενημερώσεις περιγράφεται πιο αναλυτικά στην Ενότητα 3.5.

Όταν συνδέεται ένας ομότιμος σε έναν SN, υπολογίζει το extended Skyline του και του στέλνει το αποτέλεσμα. Όταν ένας ομότιμος πέσει ή αποσυνδεθεί, τότε τα δεδομένα αυτού του ομότιμου διαγράφονται από τον SN.

Διαγραφή δεδομένων από ομότιμο

Αν σε έναν ομότιμο διαγραφούν κάποια δεδομένα τα οποία ανήκουν στο extended Skyline, τότε τον ενημερώνει. Αν τα δεδομένα αυτά δεν ανήκουν στο extended Skyline του SN, ο SN αγνοεί την ενημέρωση. Διαφορετικά θα πρέπει να ζητήσει από τους ομότιμους του να στείλουν τα Skyline τους για υπολογιστεί ξανά το ολικό Skyline. Αυτό είναι απαραίτητο, διότι τα δεδομένα που διαγράφηκαν μπορεί να απέκλεισαν κάποια σημεία από το extended Skyline και τα σημεία αυτά πρέπει τώρα να προστεθούν. Αν δεν χρειάζεται οι SN να είναι απολύτως ακριβείς στις απαντήσεις τους, τότε ο υπολογισμός αυτός μπορεί να παραληφθεί και να γίνει σε κάποιο χρονικό διάστημα που ο SN θα είναι ανενεργός.

Εισαγωγή δεδομένων σε ομότιμο

Όταν σε έναν ομότιμο προστεθούν νέα δεδομένα, αυτός ελέγχει αν ανήκουν στο extended Skyline του, συγκρίνοντας τα (με τον Bitmap πάντα αλγόριθμο), μόνο με τα δεδομένα που ήδη ανήκουν στο extended Skyline του SN. Αν κανένα από τα νέα δεδομένα, δεν προστεθεί στο extended Skyline, τότε δεν χρειάζεται να γίνει κανένας έλεγχος. Αν όμως έστω και ένα νέο σημείο προστεθεί, τότε ελέγχουμε και τα σημεία που άνηκαν εξαρχής στο extended Skyline, αφού υπάρχει περίπτωση το νέο σημείο να κυριαρχεί κάποιο ή κάποια από τα προηγούμενα. Δηλαδή, ο ON στέλνει τα νέα δεδομένα του στον SN με τον οποίο είναι συνδεδεμένος. Ο SN θα πρέπει να υπολογίσει ξανά το extended Skyline σύμφωνα με τα νέα σημεία.

Τροποποίηση δεδομένων σε ομότιμων

Αν τροποποιηθούν κάποια δεδομένα σε έναν ομότιμο, τότε αυτό αντιμετωπίζεται ως διαγραφή και εισαγωγή δεδομένων σύμφωνα με τις παραπάνω περιπτώσεις.

Ακολουθεί αναλυτικός αλγόριθμος ενημέρωσης του extended Skyline.

Αλγόριθμος 3.3 Update_extended_Skyline

Input: node, Type

```
if node==SN then
        switch (Type)
            case Insertion:
                    if (they belong to extended Skyline) then
                            reCompute extended Skyline;
                    else do nothing;
                    end if
            case Deletion:
                    if (they belong to extended Skyline) then
                        ask ONs to send again their Skylines;
                        reCompute extended Skyline;
                    else do nothing;
                    end if
            default: Invalid Update Type
                    break;
        end switch
else if node==ON then
        switch (Type)
            case Insertion:
                    if (they belong to  SNs extended Skyline) then
                        sent new data to SN;
                        call Update_extended_Skyline(SN, Insertion);
                    else do nothing;
                    end if
```

```
            case Deletion:
                if (they belong to  SNs extended Skyline) then
                    send new Skyline to SN
                    call Update_extended_Skyline(SN, Deletion);
                else do nothing;
                end if
            default: Invalid Update Type
                break;
        end switch
    end if
end if
```

3.4. Κόστος Ενημερώσεων

Κάθε φορά που γίνεται μία ενημέρωση σε έναν κόμβο χρειάζεται να σταλούν δεδομένα ανάλογα με το εάν i) τα δεδομένα ανήκαν στο Skyline, ii) διαγράφηκαν ή προστέθηκαν δεδομένα. Ισχύει ότι i) αν τα δεδομένα που ενημερώθηκαν δεν άνηκαν στο Skyline τότε δεν χρειάζεται να γίνει τίποτα, ii) η διαγραφή δεδομένων έχει πολύ μεγαλύτερο κόστος από ότι η εισαγωγή. Εξετάζουμε τι επίπτωση έχει η κατανομή των δεδομένων στις ενημερώσεις και ποιο είναι το κόστος υπολογισμού και το κόστος σε bandwidth στην περίπτωση της εισαγωγής και διαγραφής. Επίσης αναφέρουμε το πώς βοηθά η Bitmap αναπαράσταση στο να χρησιμοποιούμε λιγότερο bandwidth κατά τη διάρκεια των ενημερώσεων. Στα παρακάτω δεν εξετάζουμε την περίπτωση της τροποποίησης δεδομένων, επειδή όπως προαναφέρθηκε είναι συνδυασμός των άλλων δύο περιπτώσεων.

3.4.1 Κατανομή Δεδομένων

Η κατανομή από την οποία παράγονται τα δεδομένα επηρεάζει τον αριθμό των δεδομένων που ανήκουν στο Skyline.

Αν τα δεδομένα ακολουθούν Correlated κατανομή, τότε πολύ λίγα δεδομένα ανήκουν στο extended Skyline και έτσι υπάρχει μικρότερη πιθανότητα τα νέα ή διαγραφέντα σημεία να ανήκουν στο extended Skyline. Αν ακολουθούν Independent κατανομή, τότε η πιθανότητα τα νέα σημεία να ανήκουν στο extended Skyline είναι αρκετά μεγαλύτερη, ενώ αν ακολουθούν Anti-correlated τότε είναι σίγουρο ότι θα ανήκουν. Ο αριθμός των σημείων που ανήκουν στο extended Skyline μας ενδιαφέρει ιδιαίτερα στην περίπτωση των ενημερώσεων. Όπως αναφέρθηκε και στην αρχή της παραγράφου, αν τα δεδομένα που ενημερώθηκαν δεν ανήκουν στο extended Skyline τότε δεν χρειάζεται μα γίνει καμία ενέργεια. Έτσι όσο λιγότερα είναι τα δεδομένα που ανήκουν στο extended Skyline λόγω της κατανομής των δεδομένων, τόσο λιγότερες είναι και οι περιπτώσεις που θα χρειαστεί να σταλούν δεδομένα και να γίνει κάποιος επιπλέον υπολογισμός. Στην Παράγραφο 6.4 μελετάται το ποσοστό των δεδομένων που ανήκουν στο extended Skyline για κάποιο αριθμό ενημερώσεων που γίνεται σε έναν SN.

3.4.2 Κόστος Επικοινωνίας

Στην παράγραφο αυτή εξετάζουμε το κόστος επικοινωνίας των ενημερώσεων. Στα παρακάτω υποθέτουμε ότι γίνεται ενημέρωση κάθε φορά που αλλάζει το *u%* των δεδομένων ενός κόμβου.

Εισαγωγή

Ο ομότιμος στον οποίο προστέθηκαν δεδομένα, στέλνει τα δεδομένα του στον SN. Αν η ενημέρωση είχε γίνει στον SN και όχι στον ON, τότε δεν στέλνονται καθόλου δεδομένα. Δηλ. στέλνεται το πολύ *u%* δεδομένων.

Διαγραφή

Ο ομότιμος στον οποίο προστέθηκαν δεδομένα, στέλνει τα δεδομένα του στον SN. Αν η ενημέρωση είχε γίνει στον SN και όχι στον ON, τότε δεν στέλνονται καθόλου δεδομένα. Αν τα παραπάνω δεδομένα ανήκουν στο extended Skyline, ο SN ζητά από τους ON του να του στείλουν τα extended Skyline τους, άρα αν έχει N ομότιμους και κάθε ομότιμος *i* έχει *extended[i]* δεδομένα που ανήκουν στο extended Skyline τους, τότε σε αυτή τη φάση

στέλνονται $\sum_{i=1}^{N} extended[i]$. Συνολικό κόστος bandwidth διαγραφής *u%* δεδομένων + $\sum_{i=1}^{N} extended[i]$.

3.4.3 Κόστος Υπολογισμού

Εισαγωγή

Αρχικά γίνεται έλεγχος για τον αν τα νέα δεδομένα ανήκουν στο extended Skyline. Στον έλεγχο αυτό, ελέγχονται μόνο τα νέα δεδομένα σε σχέση με τη Bitmap αναπαράσταση μόνο των δεδομένων που ανήκουν στο extended Skyline, έστω *X*. Αν τα παραπάνω ανήκουν, τότε υπολογίζεται και για τα υπόλοιπα δεδομένα αν ανήκουν ακόμη στο extended Skyline. Δηλαδή γίνεται ο έλεγχος για *X* δεδομένα σε σχέση με *X+u%* δεδομένα (το πολύ). Συνολικά δηλαδή έχουμε υπολογισμό του extended Skyline για το πολύ *X+u%* δεδομένα

Διαγραφή

Αρχικά γίνεται έλεγχος για τον αν τα νέα δεδομένα ανήκουν στο extended Skyline. Στον έλεγχο αυτό, ελέγχονται μόνο τα νέα δεδομένα σε σχέση με τη Bitmap αναπαράσταση μόνο των δεδομένων που ανήκουν στο extended Skyline, έστω *X*. Αν τα παραπάνω ανήκουν, τότε κάθε ON πρέπει να υπολογίσει το δικό του extended Skyline για να το στείλει. Αν έχουμε N ομότιμους και κάθε ομότιμος *i* έχει *data[i]* δεδομένα, τότε έχουμε N υπολογισμούς extended Skyline για *data[i]* δεδομένα. Τέλος γίνεται από τον SN ένας ακόμη ολικός υπολογισμός του extended Skyline για *A* δεδομένα (όπου *A* το συνολικό άθροισμα των δεδομένων που ανήκουν σε όλα τα extended Skyline των ON). Σημειώνουμε ότι *A>>X*.

3.4.4 Χρησιμοποίηση Bitmap

Στους παραπάνω υπολογισμούς δεν λάβαμε υπόψη μας το ότι τα δεδομένα δεν στέλνονται στο δίκτυο με την αυθεντική τους μορφή αλλά με Bitmap αναπαράσταση. Σε αυτή την

παράγραφο εξετάζουμε το κατά πόσο μειώνεται το μέγεθος των δεδομένων που στέλνουμε, αν χρησιμοποιηθεί αυτή η αναπαράσταση.

Το κατά πόσο κερδίζουμε χώρο χρησιμοποιώντας την Bitmap αναπαράσταση εξαρτάται από τον αριθμό των κάδων που επιλέγουμε να χρησιμοποιήσουμε. Όσο λιγότερους κάδους έχουμε, τόσο λιγότερο χώρο χρειαζόμαστε, χάνουμε όμως κατά πολύ σε ακρίβεια. Για το παράδειγμα μας θα χρησιμοποιήσουμε Bitmap με 10 κάδους για αναπαράσταση float αριθμών. Για την αναπαράσταση double χρειαζόμαστε το λιγότερο 64 bit. Στην δική μας αναπαράσταση χρειαζόμαστε 10 bit (1 για κάθε bucket). Για να γίνει ίσο το Bitmap από άποψη χώρου σε σχέση με την κανονική αναπαράσταση χρειαζόμαστε 64 κάδους. Έστω ότι μία πλειάδα έχει 6 διαστάσεις, τότε θα χρειαζόμασταν κανονικά 6* 64=384 bits, ενώ με Bitmap χρειάζομαι 6*10=60bits(!), έχουμε δηλαδή 84% κέρδος σε χώρο. Το κέρδος αυτό αυξάνει κατά πολύ αν σκεφτείτε το πλήθος των πλειάδων.

3.5. Συχνότητα Ενημέρωσης

Έως τώρα έχουμε υποθέσει ότι τα σημεία ανατίθενται ομοιόμορφα σε όλους τους ON. Έτσι ο ρυθμός με τον οποίο οι ON στέλνουν ενημερώσεις στον SN με τον οποίο είναι συνδεδεμένοι είναι ίδιος για όλους. Αν όμως οι ON δεν έχουν όλοι τον ίδιο αριθμό δεδομένων (κάτι που συμβαίνει στην πραγματικότητα), τότε αν στέλνουν όλοι με τον ίδιο ρυθμό ενημερώσεις συμβαίνει ένα από τα εξής: α) είτε ο SN δεν έχει ενημερωμένο το extended Skyline του για χρονικό διάστημα στο οποίο γίνονται αρκετές ενημερώσεις, β) είτε ο SN κατακλύζεται συνεχώς από ενημερώσεις οι οποίες δεν είναι πολύ σημαντικές ή είναι μικρού μεγέθους. Αυτό είναι λογικό διότι οι κόμβοι που έχουν περισσότερα δεδομένα, έχουν και περισσότερες ενημερώσεις, οι οποίες επηρεάζουν το extended Skyline του SN.

Το κατά πόσον επηρεάζει το extended Skyline ενός SN ένας ομότιμος σχετίζεται ιδιαίτερα με τον αριθμό των δεδομένων που έχει. Έτσι, ένας απλός τρόπος λύσης του προβλήματος είναι οι κόμβοι που έχουν πολλά δεδομένα να στέλνουνε πιο συχνά ενημερώσεις από ότι οι

υπόλοιποι κόμβοι. Επειδή όμως ένας κόμβος δεν έχει ολική γνώση του δικτύου, δεν μπορεί να γνωρίζει από μόνος του αν θεωρείται κόμβος με πολλά δεδομένα.

Ένας πιο έξυπνος τρόπος είναι οι SN να κρατούν στατιστικά για τους ομότιμους τους ανάλογα με τις ενημερώσεις που έχουν κάνει στο παρελθόν. Ανάλογα με τα στατιστικά αυτά, μπορούν να δίνουν feedback στους ομότιμους τους για το αν θα πρέπει να κάνουν περισσότερες ενημερώσεις. Δηλαδή κάθε φορά που ένας ON ενημερώνει τον αντίστοιχο SN, ο SN κρατά πληροφορία για το κατά πόσο το extended Skyline του επηρεάστηκε. Ανά κάποια χρονικά διαστήματα οι SN ελέγχουν τα στατιστικά τους και αν ένας ομότιμος φαίνεται να επηρεάζει αρκετά το extended Skyline (αν η αντίστοιχη τιμή του π.χ. είναι μεγαλύτερη από κάποιο κατώφλι), τότε ο SN του ζητά να στείλει επιπλέον ενημερώσεις. Η προσέγγιση που μόλις περιγράφηκε παρουσιάζει αρκετά προβλήματα. Ένα από αυτά είναι το πώς «μετράμε» την ομοιότητα του «extended Skyline πριν την ενημέρωση» με το «extended Skyline μετά την ενημέρωση». Μία σκέψη θα μπορούσε να είναι για παράδειγμα το πλήθος των πλειάδων που διαφέρουν σε σχέση με το προηγούμενο extended Skyline προς τον συνολικό αριθμό πλειάδων. Για να γίνει όμως ένας τέτοιος υπολογισμός (για έναν ομότιμο σε κάθε ενημέρωση που κάνει) απαιτείται αρκετός χρόνος και μεγάλο υπολογιστικό κόστος, καθώς και το να έχουμε αποθηκεμένο και το παλαιό extended Skyline.

Επειδή η παραπάνω προσέγγιση είναι αρκετά ακριβή (σε υπολογιστικό κόστος) εξετάζουμε μία ακόμη προσέγγιση. Όπως περιγράφεται και παραπάνω, όταν ένας ON έχει μία ενημέρωση, πρώτα ελέγχει αν αυτή η ενημέρωση επηρεάζει το extended Skyline του αντίστοιχου SN και αν ναι τότε μόνο προχωρά στην ενημέρωση και του extended Skyline. Άρα για κάθε μία ενημέρωση που κάνει ένας ON είναι γνωστό το αν επηρέασε ή όχι το extended Skyline. Η διαφορά λοιπόν με τα προηγούμενα είναι ότι αντί να μετράμε το «πόσο» μία ενημέρωση επηρέασε το extended Skyline, μετράμε το «αν» μία ενημέρωση το επηρέασε. Αν λοιπόν οι ενημερώσεις ενός ON τείνουν να επηρεάζουν συχνά το extended Skyline τότε ζητείται από τον συγκεκριμένο ON να στέλνει πιο συχνά ενημερώσεις. Αυτός ο υπολογισμός είναι πολύ απλός και είναι εύκολο να γίνει τοπικά σε κάθε ON, αφού η πληροφορία που χρειάζεται υπάρχει ήδη.

Κάθε SN διατηρεί για κάθε ON που είναι συνδεδεμένος με αυτόν, έναν αριθμό update_ const που σχετίζεται με την ενημέρωση. Ο αριθμός αυτός είναι ίδιος για κάθε ON, ο οποίος ακόμη δεν έχει καμία ενημέρωση και καθορίζεται από το σύστημα. Αυτό σημαίνει ότι ο ON κάνει ενημερώσεις με έναν συγκεκριμένο ρυθμό που είναι αρχικά ίσος με update_ const % των δεδομένων του (αυτό σημαίνει ότι κάθε φορά που αλλάζει το update_ const % των δεδομένων του κόμβου, αυτός στέλνει μία ενημέρωση).

Αν η επόμενη ενημέρωση που θα κάνει ο κόμβος επηρεάσει το extended Skyline τότε το update_ const μειώνεται κατά h (π.χ αν h=2 και update_ const =10 τότε update_ const=8). Από εκείνη τη στιγμή ο κόμβος θα στέλνει ενημερώσεις κάθε φορά που αλλάζει το νέο update_ const % των δεδομένων του (π.χ. 8%), δηλαδή πιο συχνά από πριν. Το update_const συνεχίζει να μειώνεται, ο ρυθμός ενημέρωσης όμως σταματά να αυξάνεται όταν φτάσει σε κάποιο συγκεκριμένο όριο θ1 (δηλαδή στέλνει ενημέρωση ανά το λιγότερο θ1% αλλαγές στα δεδομένα του). Αυτό γίνεται για να μην υπάρχει κατακλυσμός ενημερώσεων από κάποιον συγκεκριμένο κόμβο.

Αν η ενημέρωση που θα κάνει ο κόμβος δεν επηρεάσει το extended Skyline, τότε αντίστοιχα με τα προηγούμενα, το update_ const αυξάνεται κατά g, όπου g<h. Επίσης αντίστοιχα υπάρχει ένα όριο θ2. Το g είναι μικρότερο του h, έτσι ώστε ο ρυθμός ενημέρωσης να επηρεάζεται περισσότερο από τις ενημερώσεις που μας ενδιαφέρουν, παρά από εκείνες που δεν έχουν σημασία.

Παρακάτω δίνουμε τον αλγόριθμο ενημέρωσης του update_const, όπου τα g, h και τα κατώφλια θ1, θ2 καθορίζονται από το εκάστοτε πρόβλημα.

Αλγόριθμος 3.4 Update update_const

Input: change, update_const(ON), g, h, θ1, θ2
Output: update_const(ON)

```
if (change=yes AND update_const(ON)-h>θ1) then   //αν επηρεάζει το Skyline
        update_const(ON)=update_const(ON)-h;
else if (change=no AND update_const(ON)+g<θ2) then  //αν δεν επηρεάζει το Skyline
        update_const(ON)=update_const(ON)+g;
     end if
end if
```

3.6. Bitmap εναντίον άλλων Μεθόδων

Στη βιβλιογραφία, έχει προταθεί πλήθος μεθόδων οι οποίες χρησιμοποιούν κάποιο ευρετήριο (index) για την εύρεση του Skyline. Κύριο πλεονέκτημα αυτών των μεθόδων είναι η progressive (προοδευτική) ιδιότητα τους. Η ιδιότητα αυτή απορρέει από την μερική διάταξη με την οποία αποθηκεύονται στο ευρετήριο τα δεδομένα. Η προοδευτική ιδιότητα έχει δύο έννοιες. Η πρώτη είναι ότι μπορείς να εξάγεις αρχικά αποτελέσματα (σημεία δηλαδή που ανήκουν στο Skyline) χωρίς να χρειαστεί να εξετάσεις το σύνολο των δεδομένων. Η δεύτερη είναι ότι μπορείς εξ αρχής να αποκλείσεις κάποια δεδομένα χωρίς να χρειαστεί να τα εξετάσεις καθόλου. Σε αυτή την παράγραφο γίνεται μία σύγκριση των μεθόδων που χρησιμοποιούν ευρετήριο με τη μέθοδο που χρησιμοποιεί Bitmap, με τον τρόπο που αυτό περιγράφηκε παραπάνω. Σκοπός είναι να παρουσιαστούν οι λόγοι που μας έκαναν να επιλέξουμε σε αυτή την εργασία, τη δομή Bitmap για την επίλυση του προβλήματος. Για αυτό γίνεται μία σύγκριση του Bitmap με τις μεθόδους που χρησιμοποιούν ευρετήρια.

Ο πρώτος λόγος επιλογής του Bitmap είναι το ότι η δομή του καταλαμβάνει πολύ λίγο χώρο στο δίσκο. Ο χώρος μας ενδιαφέρει ιδιαίτερα στην συγκεκριμένη τοπολογία στην οποία επιλύουμε το πρόβλημα. Οι SN χρειάζεται να αποθηκεύσουν αποτελέσματα από τα δεδομένα όλων των ON που συνδέονται με αυτούς. Ο όγκος της πληροφορίας αυτής μπορεί να είναι αρκετά μεγάλος, ειδικά στην περίπτωση που αποθηκεύουν το extended Skyline το οποίο

αποδεικνύεται να περιέχει πολύ περισσότερα αποτελέσματα από ότι το απλό Skyline (αυτό συμβαίνει επειδή μπορεί να απαντήσει σε όλες τις πιθανές ερωτήσεις). Η Bitmap δομή καταλαμβάνει χώρο ο οποίος στη συνήθη περίπτωση είναι λιγότερος από το μισό του χώρου που καταλαμβάνουν τα πραγματικά δεδομένα. Αντίθετα τα ευρετήρια καταλαμβάνουν αρκετό χώρο και μάλιστα χώρο ο οποίος είναι κατά πολύ μεγαλύτερος από αυτόν που θα καταλάμβαναν μόνο τα πραγματικά δεδομένα. Μάλιστα μερικές μέθοδοι, όπως π.χ. η Index μέθοδος των [TaEO01] χρειάζονται ένα ευρετήριο για κάθε συνδυασμό των διαστάσεων! Όσον αφορά τη Bitmap αναπαράσταση, παραθέτουμε ένα γενικό παράδειγμα:

Έστω ότι έχουμε k διαστάσεις από float αριθμούς σε μηχανή που αναπαριστά τους double με 64 bits και έχουμε N σημεία. Τότε αν στείλουμε τα δεδομένα χωρίς αναπαράσταση, θα στείλουμε 64xkxN bits. Αντίθετα αν στείλουμε Bitmap αναπαράσταση στην οποία έχουμε 10 κάδους για κάθε διάσταση, τότε θα στείλουμε 10xkxN bits. Το μέγεθος των δεδομένων βλέπουμε ότι μειώνεται σημαντικά. Θα χρειαστούμε 64 κάδους για κάθε διάσταση για να φτάσουμε σε μέγεθος τα κανονικά δεδομένα.

Ο παραπάνω λόγος επιλογής Bitmap, δηλαδή το ότι η αναπαράσταση καταλαμβάνει λίγο χώρο, έχει και μία επιπλέον προέκταση. Όταν μεταφέρουμε πληροφορία στο δίκτυο χρησιμοποιούμε αυτήν την αναπαράσταση με αποτέλεσμα να απαιτείται λιγότερο bandwidth από ότι θα απαιτούταν αν στέλναμε την πραγματική πληροφορία. Στις υπόλοιπες μεθόδους, η πληροφορία στέλνεται ως έχει και έτσι δεν εξοικονομούν το bandwidth του δικτύου.

Ένας ακόμη σημαντικός λόγος επιλογής του Bitmap είναι ο τρόπος ενημέρωσης του. Αν σε έναν ομότιμο προστεθούν ή διαγραφούν δεδομένα τότε η αντίστοιχη ενημέρωση της Bitmap αναπαράστασης θα είναι μία εισαγωγή ή διαγραφή σημείου. Αντίθετα, η ενημέρωση ενός ευρετηρίου είναι μία αρκετά χρονοβόρα διαδικασία και πολλές φορές απαιτεί την πλήρη αναδιάρθρωση του ευρετηρίου. Σκεφτείτε τι γίνεται στην περίπτωση ενός συστήματος ομότιμων, όπου οι ενημερώσεις είναι αρκετά συχνές.

Σημαντικό χαρακτηριστικό της Bitmap αναπαράστασης είναι η ευκολία με την οποία μπορεί να επεκταθεί σε υπολογισμό του Skyline (και extended Skyline) για οποιοδήποτε υποσύνολο

διαστάσεων. Αυτό δεν ισχύει στις Index μεθόδους, οι οποίες έχουν σχεδιαστεί καθαρά για παραγωγή ολόκληρου του Skyline. Σε πολλές περιπτώσεις οι μέθοδοι αυτοί απαιτούν επιπλέον προεπεξεργασία των δεδομένων για να είναι ναι σε θέση να απαντήσουν σε όλα τα ερωτήματα.

Κοινό χαρακτηριστικό των μεθόδων που συγκρίνουμε είναι ότι μπορούν να παράγουν πολύ γρήγορα κάποια αρχικά αποτελέσματα. Η Bitmap μέθοδος παράγει πολύ γρήγορα αρχικά αποτελέσματα επειδή χρησιμοποιεί πράξεις σε bit, οι οποίες είναι πολύ γρήγορες. Οι μέθοδοι με τα ευρετήρια έχουν την ίδια ιδιότητα λόγω του ότι υπάρχει μερική κατάταξη στα δεδομένα και έτσι μπορούν εύκολα να εξάγουν κάποια πρώτα αποτελέσματα. Η σειρά των πρώτων αυτών αποτελεσμάτων εξαρτάται στην περίπτωση του Bitmap από τη σειρά με την οποία είναι αποθηκευμένα τα δεδομένα, ενώ στην περίπτωση των μεθόδων με ευρετήριο από την κατανομή των δεδομένων.

Όπως είναι αναμενόμενο, τα παραπάνω προτερήματα του Bitmap συνδυάζονται με κάποια μειονεκτήματα. Το πρώτο είναι ότι είναι αργός στην παραγωγή ολόκληρου του Skyline. Αυτό συμβαίνει επειδή πρέπει για κάθε σημείο ξεχωριστά να ελέγξεις αν ανήκει στο Skyline. Δεν μπορείς δηλαδή να αποκλείσεις κάποια σημεία τα οποία δεν πρόκειται να ανήκουν στο Skyline και να μην τα εξετάσεις καθόλου. Όπως περιγράφηκε και στην αρχή της παραγράφου, τα παραπάνω γίνονται πολύ εύκολα με τη χρήση ευρετηρίων. Παρόλα αυτά η κύρια χρήση του Bitmap αλγόριθμου στο σύστημα μας είναι για τον υπολογισμό του extended Skyline. Σύμφωνα με αυτόν τον αλγόριθμο μπορεί να γίνει απαλοιφή κάποιων σημείων χωρίς αυτά να ελεγχθούν. Επίσης μπορεί να γίνει απευθείας εισαγωγή κάποιων σημείων στο Skyline. Με αυτό τον τρόπο οι έλεγχοι κυριαρχίας μειώνονται αρκετά και ο χρόνος υπολογισμού ολόκληρου του extended Skyline μειώνεται κατά πολύ.

Ένα ακόμη μειονέκτημα του είναι το χάσιμο πληροφορίας. Αυτό συμβαίνει επειδή χρησιμοποιεί κάδους. Για να μην χάσουμε καθόλου πληροφορία πρέπει να έχουμε έναν κάδο για κάθε πιθανή τιμή της διάστασης. Το παραπάνω δεν είναι εφικτό, διότι σε πολλές περιπτώσεις δεν είναι δυνατόν να γνωρίζουμε όλες τις πιθανές τιμές. Στην προσέγγιση μας αντιμετωπίζουμε αυτό το πρόβλημα χρησιμοποιώντας μικρό αριθμό από κάδους και extended

Skyline για να βρούμε το ολικό Skyline του συστήματος χωρίς να χάσουμε πληροφορία (αντιθέτως έχουμε επιπλέον πληροφορία). Αν ο ομότιμος που θέτει το ερώτημα ενδιαφέρεται για ακριβή αποτελέσματα, επικοινωνεί με τους ομότιμους των οποίων τα σημεία ανήκουν στο extended για να λάβει την ακριβή πληροφορία. Η διαδικασία αυτή δεν είναι τόσο αργή λόγω της προεπεξεργασίας που έχει στα δεδομένα. Δηλαδή τα δεδομένα που τελικά θα λάβει ο ομότιμος που θέτει το Skyline ερώτημα είναι κατά πολύ λιγότερα χρησιμοποιώντας τη Bitmap μέθοδο από ότι αν χρησιμοποιούνταν μία naïve τεχνική.

Πίνακας 3.1 Bitmap εναντίον index μεθόδων

Bitmap	***Μέθοδοι που χρησιμοποιούν index***
καταλαμβάνει πολύ λίγο χώρο στο δίσκο	σπαταλά το χώρο των ομότιμων
οι ενημερώσεις είναι πολύ γρήγορες και απαιτούν απλά μία εισαγωγή ή διαγραφή γραμμών του πίνακα	οι ενημερώσεις είναι χρονοβόρες και πολλές φορές απαιτούν πλήρη αναδιάρθρωση του συστήματος (ειδικά όταν πρόκειται για συστήματα ομότιμων)
απαιτείται λίγο bandwidth όταν πρόκειται για μεταφορά πληροφορίας	μεταφέρεται η πληροφορία ως έχει, χωρίς να χρησιμοποιείται αναπαράσταση και έτσι χρειάζεται περισσότερο bandwidth
γρήγορος στον υπολογισμό του extended Skyline – γίνεται απαλοιφή σημείων	λιγότερο γρήγοροι όσον αφορά τον υπολογισμό του extended Skyline
άμεση εφαρμογή subspace Skyline computation	σχεδιασμένοι για τον υπολογισμό ολόκληρου του Skyline, απαιτείται επιπλέον προεπεξεργασία
δεν διαγράφονται εξ'αρχής κάποια σημεία που σίγουρα δεν ανήκουν στο Skyline πρέπει να ελέγχουν ένα ένα όλα τα σημεία για να δούμε αν ανήκουν στο Skyline	λόγω της διάταξης των δεδομένων κάποια από αυτά μπορούν αμέσως να διαγραφούν δεν χρειάζεται να γίνει έλεγχος για όλα τα σημεία
πολύ αργός για την παραγωγή όλου του Skyline λόγω της παραπάνω ιδιότητας	γρήγοροι στην παραγωγή ολόκληρου του Skyline
γρήγορος στην εξαγωγή πρώτων-αρχικών αποτελεσμάτων	γρήγορες στην εξαγωγή πρώτων-αρχικών αποτελεσμάτων
η σειρά εξαγωγής αποτελεσμάτων εξαρτάται από τη σειρά με την οποία υπάρχουν τα δεδομένα στην αναπαράσταση	η σειρά εξαγωγής αποτελεσμάτων συνήθως εξαρτάται από την κατανομή των δεδομένων
για τη Bitmap αναπαράσταση απαιτείται να γνωρίζουμε κάθε πιθανή τιμή που μπορούν να πάρουν τα σημεία για κάθε διάσταση	απαιτείται ένα ευρετήριο για κάθε διάσταση ή ακόμη και ένα ευρετήριο για κάθε πιθανό συνδυασμό των διαστάσεων

ΚΕΦΑΛΑΙΟ 4. CACHING SKYLINE

Σε αυτό το κεφάλαιο περιγράφεται η λειτουργία του συστήματος υπό την παρουσία cache. Στην τοπολογία των Super Nodes που έχουμε υποθέσει στην εργασία, προστίθεται μία cache Skyline ερωτημάτων σε κάθε SN. Δηλαδή, αν ένας SN κληθεί να απαντήσει μία Skyline ερώτηση, πριν στείλει το αποτέλεσμα στον κόμβο που τον ρώτησε, το αποθηκεύει στην cache του. Την επόμενη φορά που ο SN αυτός θα κληθεί να απαντήσει στο ίδιο ερώτημα, χρησιμοποιεί το αποτέλεσμα που ήδη έχει στην cache του. Στόχος της τροποποίησης αυτής είναι α) να επισπευτούν οι απαντήσεις Skyline ερωτημάτων και β) να ερωτώνται λιγότεροι SN για να απαντηθεί ένα ερώτημα. Περιγράφεται η δομή της cache που χρησιμοποιείται, η διατήρηση της και η δρομολόγηση των ερωτημάτων υπό την παρουσία της. Τέλος, χρησιμοποιούμε τα αποτελέσματα που ήδη υπάρχουν στην cache (επαναχρησιμοποίηση ερωτήσεων) για να επισπεύσουμε ακόμη περισσότερο τη διαδικασία των απαντήσεων και ορίζουμε ένα νέο τύπο ερωτήσεων, τις fast Skyline queries.

4.1. Cache Skyline Ερωτημάτων

Στην επιστήμη των υπολογιστών ως cache ορίζεται μία συλλογή δεδομένων τα οποία είναι πιστά αντίγραφα πληροφορίας η οποία αποθηκεύεται κάπου αλλού ή έχει υπολογιστεί πρωτύτερα. Στην περίπτωση της cache η αυθεντική πληροφορία είναι δύσκολο να ανακτηθεί ή να υπολογιστεί σε σχέση με το κόστος διαβάσματος της cache. Με άλλα λόγια, η cache

είναι μία προσωρινή περιοχή αποθήκευσης όπου αποθηκεύονται δεδομένα τα οποία προσπελαύνονται συχνά, έτσι ώστε να έχουμε γρήγορη πρόσβαση σε αυτά. Από τη στιγμή που κάποια δεδομένα έχουν αποθηκευτεί στην cache, η επόμενη χρήση τους γίνεται απευθείας από αυτήν, αντί να επαναϋπολογιστεί το αποτέλεσμα.

Στο σύστημα μας η cache διατηρεί υπολογισμένα ένα σύνολο από extended Skyline ερωτήματα. Τα ερωτήματα αυτά είχαν τεθεί κάποια στιγμή στο παρελθόν στο σύστημα. Μόνο οι SN διατηρούν cache Skyline ερωτημάτων. Η cache αυτή τη χρησιμοποιείται από τους SN στη δρομολόγηση των Skyline ερωτημάτων. Σε αυτή την παράγραφο περιγράφεται η δομή της cache αλλά και ο τρόπος με τον οποίο γίνεται η δρομολόγηση των ερωτημάτων υπό την παρουσία της.

Σε μία cache διατηρείται ένας αριθμός απαντήσεων Skyline ερωτήσεων. Για κάθε ερώτηση διατηρείται:

- ένα id: το οποίο είναι ίδιο για όλες τις ερωτήσεις που αφορούν τις ίδιες διαστάσεις
- το αποτέλεσμα της ερώτησης σε μορφή Bitmap
- μία λίστα από SN κόμβους των οποίων τα δεδομένα συμπεριλαμβάνονται στο αποτέλεσμα της cache
- ο χρόνος τον οποίο χρησιμοποιήθηκε τελευταία φορά το αποτέλεσμα της cache για αυτή την ερώτηση
- ο χρόνος τον οποίο «λήγει» το περιεχόμενο της cache (μετά την πάροδο αυτού του χρόνου το περιεχόμενο της cache δεν χρησιμοποιείται)

Η δρομολόγηση των Skyline ερωτημάτων τροποποιείται έτσι ώστε να χρησιμοποιούνται τα αποτελέσματα της cache. Έστω ότι σε έναν ομότιμο φτάνει ένα ερώτημα (ή το θέτει ο ίδιος). Αρχικά ελέγχει αν το αποτέλεσμα υπάρχει αποθηκευμένο στην cache του (ο έλεγχος γίνεται σύμφωνα με το id της ερώτησης). Αν το ερώτημα δεν υπάρχει στην cache τότε προωθείται στους γείτονες με τον τρόπο που γίνεται στην απλή-κανονική περίπτωση. Όταν ο ομότιμος λάβει το αποτέλεσμα, πριν το προωθήσει σε αυτόν που τον ρώτησε, το αποθηκεύει στην cache του (μαζί με όλα τα στοιχεία της cache όπως περιγράφηκαν παραπάνω). Επειδή η cache είναι προκαθορισμένου μεγέθους, μπορεί τη δεδομένη χρονική στιγμή η cache να είναι

γεμάτη. Αν δεν υπάρχει χώρος για να αποθηκευτούν τα αποτελέσματα του νέου ερωτήματος, τότε από αυτά που είναι ήδη αποθηκευμένα στην cache διαγράφεται αυτό που χρησιμοποιήθηκε παλαιότερα (least recently used) και το αντικαθιστώ με το νέο.

Αν το αποτέλεσμα της ερώτησης υπάρχει αποθηκευμένο στην cache, τότε προωθείται το ερώτημα μόνο στους κόμβους οι οποίοι δεν περιέχονται στη λίστα του ερωτήματος που υπάρχει στην cache (δηλαδή μόνο σε αυτούς των οποίων τα δεδομένα δεν εμπεριέχονται στο αποτέλεσμα). Οι υπόλοιποι κόμβοι θεωρούμε ότι έχουν απαντήσει στο ερώτημα. Έπειτα το περιεχόμενο της cache ανανεώνεται σύμφωνα με τα νέα αποτελέσματα.

Επειδή στο σύστημα γίνονται ενημερώσεις και τα δεδομένα αλλάζουν, πρέπει και το αποτέλεσμα της cache περιοδικά να ανανεώνεται. Για αυτό το λόγο υπάρχει ημερομηνία λήξης. Αν ζητηθεί από τον SN το αποτέλεσμα ενός ερωτήματος το οποίο υπάρχει στην cache, αλλά έχει λήξει, τότε το ερώτημα προωθείται κανονικά (σαν να μην υπήρχε η cache) και η cache ανανεώνεται. Σημειώνουμε ότι η cache ανανεώνεται μόνο εφόσον ζητηθεί αυτό το αποτέλεσμα για να μην φορτώνεται άσκοπα το δίκτυο.

Παρακάτω περιγράφονται οι αλγόριθμοι δρομολόγησης υπό την παρουσία cache. Χωρίζουμε δύο περιπτώσεις: α) ένας κόμβος λαμβάνει ερώτημα και β) ένας κόμβος θέτει ερώτημα, για να έχουμε αντιστοιχία με τους αλγορίθμους της Παραγράφου 3.2. Οι παρακάτω αλγόριθμοι αφορούν μόνο SN κόμβους. Η συνάρτηση Update_Cache() ενημερώνει τα αποτελέσματα της cache, τους χρόνους και τη λίστα των κόμβων που συμπεριλαμβάνονται στην απάντηση. Η Create_Cache() δημιουργεί στην cache την απάντηση ενός νέου ερωτήματος. Αν δεν υπάρχει αρκετός χώρος στην cache για την αποθήκευση των νέων αποτελεσμάτων, διαγράφουμε το ερώτημα το οποίο χρησιμοποιήθηκε πιο παλιά από τα υπόλοιπα (least recently used).

Αλγόριθμος 4.1 Pose_Query_using_Cache

Input: node, Query, QueryId, P: NULL if it is the first node that poses the query

Output: FinalSkyline

```
ckeck=Check_Cache(QueryId);              //ελέγχει αν το ερώτημα υπάρχει στην cache
if (check==exist && not expired) then
    /*στέλνει το ερώτημα στους γείτονες του εκτός από αυτούς
                                που υπάρχουν ήδη στην cache*/
    Send Query to Neighbor[node]-P-Cache_neighbors;
    LocalSkyline=Cached_Skyline;
    Wait(t1);                                                    //περιμένει
    S=collectAnswers();                                  //λαμβάνει τις απαντήσεις
    FinalSkyline=Bitmap(Query, S, LocalSkyline); //υπολογίζει το τελικό Skyline
    call Update_Cache(FinalSkyline); //και ενημερώνει την cache
else
    FinalSkyline=PoseQuery(SN, Query, QueryId, Null); //θέτει το ερώτημα
    if (check==exist && not expired) then
        call Update_Cache(FinalSkyline);    //ενημερώνει την cache
    else
        call Create_Cache(FinalSkyline);    //ή δημιουργεί cache
    end if
end if
return FinalSkyline;
```

Αλγόριθμος 4.2 Receive_Query_using_Cache

Input: node, Query, QueryId, P
Output: PartialSkyline

```
if QueryId is the first in t1 then        //αν δεν έχει λάβει την ερώτηση σε t3 διάστημα
    ckeck=Check_Cache(QueryId);           //check if query exists in cache
    if (check==exist && not expired) then
        PartialSkyline=Pose_Query_using_Cache(SN, Query, QueryId,P);
        Send PartialSkyline to P;                    //στέλνει το αποτέλεσμα στον P
    else
        PartialSkyline=PoseQuery(SN, Query, QueryId,P);
      call Create_Cache(PartialSkyline);          //δημιουργεί cache
      Send PartialSkyline to P;          //στέλνει το αποτέλεσμα στον P
    end if
end if
```

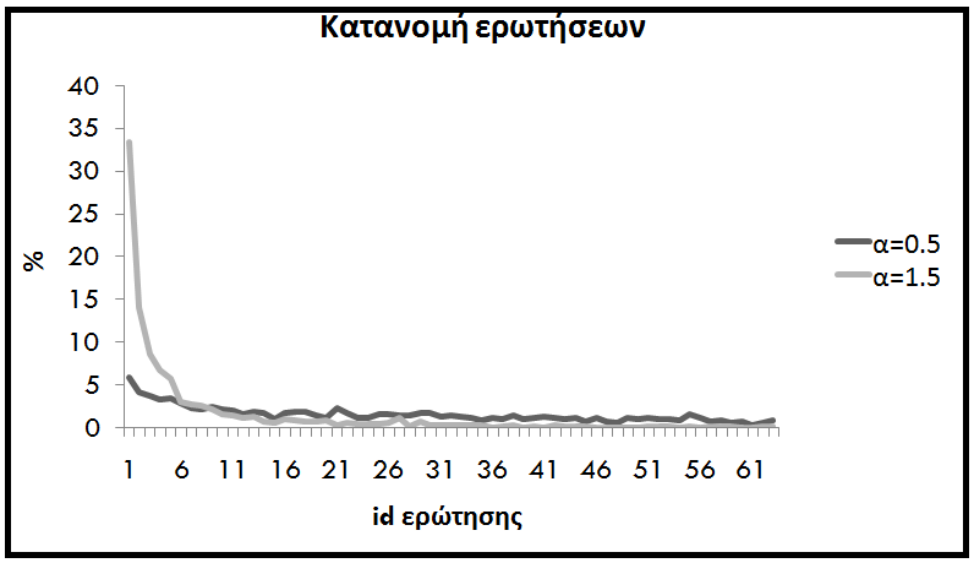
4.2. Απόδοση της Cache

Σε αυτή την παράγραφο παρουσιάζονται οι παράμετροι που επηρεάζουν την απόδοση της cache στο σύστημα που περιγράφουμε. Μερικοί από αυτούς εξαρτώνται από την ίδια την cache και είναι το μέγεθος της, ή ο χρόνος λήξης των περιεχομένων της, ενώ άλλοι εξαρτώνται από το σύστημα, όπως π.χ. η κατανομή των ερωτήσεων του συστήματος.

4.2.1 Κατανομή Ερωτήσεων

Ένας παράγοντας που επηρεάζει την απόδοση της cache είναι η κατανομή ή αλλιώς ο φόρτος των ερωτήσεων του συστήματος. Ο παράγοντας αυτός έχει διάφορες εκδοχές όπως το είδος των ερωτήσεων που γίνονται (τι αριθμό διαστάσεων έχουν;), η συχνότητα με την οποία γίνονται οι ερωτήσεις (αρκετά συχνά ώστε να χρησιμοποιηθεί το περιεχόμενο της cache;) και η σειρά εμφάνισης των ερωτήσεων (εμφανίζονται σε συνεχόμενες φορές οι ίδιες ερωτήσεις;).

Όσον αφορά το είδος των ερωτήσεων, εννοούμε τον αριθμό των διαστάσεων που οι ερωτήσεις αυτές εμπεριέχουν. Ο αριθμός των διαστάσεων μιας ερώτησης επηρεάζει την απόδοση της cache με τον παρακάτω τρόπο: όσο περισσότερες είναι οι διαστάσεις που εμπεριέχει ένα ερώτημα, τόσο περισσότερα είναι και τα αποτελέσματα του extended Skyline, δηλαδή τα αποτελέσματα που αποθηκεύονται στην cache. Αν λοιπόν έχουμε ερωτήματα πολλών διαστάσεων, αυτά χρησιμοποιούν πολύ χώροι στην cache, μειώνοντας τη διαθεσιμότητα της για άλλα ερωτήματα.



Σχήμα 4.1 Κατανομή Ερωτήσεων

Ένας ακόμη παράγοντας που επηρεάζει την απόδοση της cache είναι η συχνότητα με την οποία τίθενται ερωτήματα στο σύστημα. Αν τα ερωτήματα τίθενται σε αραιά διαστήματα, τότε η cache δεν χρησιμοποιείται σχεδόν ποτέ. Επίσης σημαντικό είναι το πόσο συχνά τίθεται το ίδιο ερώτημα στο σύστημα, ώστε η cache να μπορεί να χρησιμοποιήσει τα αποτελέσματα που έχει αποθηκεύσει.

Στο σύστημα μας οι ερωτήσεις ακολουθούν Zipf κατανομή. Η Zipf κατανομή δίνεται από τον τύπο:

$$P(k,a) = 1/[k^a * \zeta(a)]$$

και εκφράζει την πιθανότητα εμφάνισης της *k*-οστής Skyline ερώτησης στο σύστημα ομότιμων. Το *ζ(α)* είναι η συνάρτηση ζήτα του Riemann και το α είναι η παράμετρος skewness της κατανομής.

Το Σχήμα 4.1 δείχνει τον αριθμό εμφάνισης κάθε αριθμού (id ερώτησης) αν ο συντελεστής κατανομής είναι α=0.5 ή α=1.5. Όσο μεγαλύτερο είναι το α, τόσο πιο συχνά εμφανίζονται τα πρώτα id σε σχέση με τα τελευταία. Όταν κάποιες ερωτήσεις εμφανίζονται πολύ πιο συχνά από κάποιες άλλες, είναι λογικό να αναμένουμε ότι το περιεχόμενο της cache θα χρησιμοποιείται πιο συχνά και έτσι θα στέλνονται πολύ λιγότερα μηνύματα στο δίκτυο. Άρα όσο πιο μεγάλο είναι το α, τόσο μεγαλύτερη είναι η απόδοση της cache.

4.2.2 Χρόνος «λήξης» των Ερωτημάτων της Cache

Αν το αποτέλεσμα κάποιου ερωτήματος υπάρχει ήδη στην cache αλλά έχει λήξει, τότε η cache αγνοείται και δεν μειώνεται ο αριθμός των μηνυμάτων για τη δρομολόγηση ενός ερωτήματος. Άρα όσο μεγαλύτερος είναι ο χρόνος λήξης, τόσο πιο πολύ χρησιμοποιείται το αποτέλεσμα που υπάρχει στην cache και τόσο λιγότερα μηνύματα στέλνονται στο δίκτυο.

4.2.3 Χωρητικότητα της Cache

Όσο περισσότερα αποτελέσματα ερωτήσεων χωράνε στην cache, τόσο λιγότερα μηνύματα θα χρειαστεί να σταλούν, αφού δεν θα σβήνεται συχνά περιεχόμενο της cache που μπορεί να χρησιμοποιηθεί στα επόμενα ερωτήματα.

4.2.4 Κατανομή Δεδομένων

Η κατανομή των δεδομένων σχετίζεται άμεσα με τη χωρητικότητα της cache. Αυτό συμβαίνει επειδή τα αποτελέσματα των ερωτήσεων είναι πολύ περισσότερα όταν έχουμε Independent και Anti-Correlated κατανομή και αρκετά λιγότερα όταν έχουμε Correlated κατανομή.

4.2.5 Αριθμός Δεδομένων

Για τους ίδιους λόγους που αναφέραμε παραπάνω, η απόδοση της cache εξαρτάται και από των αριθμό των σημείων κάθε κόμβου ή καλύτερα από τον αριθμό των σημείων για τα οποία είναι υπεύθυνος ένας SN.

4.2.6 Τοπολογία του Δικτύου

Σε αυτό το σημείο πρέπει να σημειώσουμε ότι και η τοπολογία του δικτύου και συγκεκριμένα η συνδεσμολογία των SN παίζει σημαντικό ρόλο στην απόδοση της cache. Εάν το δίκτυο είναι φτιαγμένο με τέτοιο τρόπο ώστε οι SN να έχουν πολλούς γείτονες (να συνδέονται δηλαδή, με αρκετούς SN), τότε κατά τη δρομολόγηση ενός ερωτήματος αποθηκεύουν ενδιάμεσα αποτελέσματα πολλών άλλων SN. Όσο περισσότερα ενδιάμεσα αποτελέσματα αποθηκεύει μία cache, τόσο αυξάνεται η απόδοση της, αφού το ερώτημα θα χρειαστεί να δρομολογηθεί σε λιγότερους SN.

4.3. Επαναχρησιμοποίηση Ερωτήσεων

Λόγω του ότι χρησιμοποιείται το extended Skyline ενός συνόλου σημείων και όχι το απλό Skyline έχουμε στη διάθεση μας τις Ιδιότητες 2.1 και 2.2 που αναφέρονται στο Κεφάλαιο 2. Υπενθυμίζουμε εδώ τις ιδιότητες αυτές. Επιπλέον προσθέτουμε και μία ακόμη ιδιότητα του extended Skyline.

Ιδιότητα 2.1
Κάθε σημείο που ανήκει στο Skyline ενός συνόλου διαστάσεων U ανήκει και στο extended Skyline του U, δηλ. $SKY_U \subseteq ext - SKY_U$.

Ιδιότητα 2.2

Κάθε σημείο που ανήκει στο Skyline ενός υποσυνόλου των διαστάσεων V, $V \subseteq U$, ανήκει και στο extended Skyline του U, δηλ. $SKY_V \subseteq ext-SKY_U, V \subseteq U$.

Ιδιότητα 4.1

Κάθε σημείο που ανήκει στο extended Skyline ενός υποσυνόλου των διαστάσεων V, $V \subseteq U$, ανήκει και στο extended Skyline του U, δηλ. $ext-SKY_V \subseteq ext-SKY_U, V \subseteq U$.

Η Ιδιότητα 4.1 είναι πολύ εύκολο να αποδειχθεί. Έστω ένα σημείο *p* ανήκει στο extended Skyline του V. Αν δεν ανήκει στο extended Skyline του U, τότε υπάρχει ένα σημείο *q* το οποίο το κάνει ext-dominate. Δηλαδή το *q* έχει σε όλες τις διαστάσεις του U τιμές μεγαλύτερες από αυτές του *p*. Αν όμως ισχύει αυτό, τότε το *q* έχει μεγαλύτερες τιμές και σε όλες τις διαστάσεις του συνόλου V. Άρα το *q* κυριαρχεί του *p* και στο σύνολο V και άρα το *p* δεν ανήκει στο extended Skyline του V. Καταλήγουμε σε άτοπο.

Σύμφωνα με τις ιδιότητες αυτές καταλήγουμε στα εξής συμπεράσματα: α) από το extended Skyline όλων των διαστάσεων, μπορεί να υπολογιστεί το Skyline οποιουδήποτε υποσυνόλου των διαστάσεων ζητηθεί, β) αν θέλουμε να υπολογίσουμε το Skyline ενός συνόλου διαστάσεων τότε αρκεί να έχουμε το extended Skyline οποιουδήποτε υπερσυνόλου των διαστάσεων αυτών, γ) αν γνωρίζουμε το extended Skyline ενός υποσυνόλου διαστάσεων τότε γνωρίζουμε ένα κομμάτι του extended Skyline οποιουδήποτε υπερσυνόλου των διαστάσεων αυτών.

Οι παραπάνω ιδιότητες χρησιμοποιούνται ως βάση του συστήματος μας. Οι SN διατηρούν το extended Skyline έτσι ώστε να μπορούν να απαντήσουν σε ερωτήματα οποιασδήποτε διάστασης. Υπενθυμίζουμε επίσης, ότι επειδή χρησιμοποιούμε Bitmap αναπαράσταση, είναι απαραίτητο να υπολογίζουμε το extended Skyline σε κάθε βήμα έτσι ώστε να μην έχουμε false negative. Το απλό Skyline, υπολογίζεται μόνο στο τελευταίο βήμα από τον κόμβο που το επιθυμεί. Άρα και στα ενδιάμεσα αποτελέσματα, τα οποία αποθηκεύονται στην cache,

υπάρχει υπολογισμένο το extended Skyline. Συνεπώς οι ιδιότητες του extended Skyline μπορούν εύκολα να εφαρμοστούν για την πιο αποδοτική χρησιμοποίηση της cache.

Αυτό που κάνουμε είναι να χρησιμοποιούμε προηγούμενα αποτελέσματα της cache για να απαντήσουμε νέες ερωτήσεις που δεν υπάρχουν υπολογισμένες σε αυτήν. Αυτό γίνεται με δύο τρόπους, α) είτε χρησιμοποιούμε υπολογισμένες ερωτήσεις μεγαλύτερων διαστάσεων, β) είτε χρησιμοποιούμε υπολογισμένες ερωτήσεις μικρότερων διαστάσεων. Επίσης εισάγουμε ένα νέο τύπο Skyline ερωτήσεων που ονομάζονται fast Skyline queries.

4.3.1 Χρησιμοποίηση Ερωτήσεων Μεγαλύτερων Διαστάσεων

Όπως αναφέρεται στην Ιδιότητα 4.1 το extended Skyline ενός αριθμού διαστάσεων είναι υπερσύνολο του extended Skyline μικρότερου αριθμού διαστάσεων. Για αυτό το λόγο, οι SN διατηρούν το extended Skyline όλων των διαστάσεων των δεδομένων. Όταν πρόκειται να υπολογίσουν το extended Skyline οποιουδήποτε συνδυασμού διαστάσεων χρησιμοποιούν το extended Skyline για να εξάγουν τα αποτέλεσμα. Στο σύστημα μας, η cache αποθηκεύει επίσης extended Skyline. Τα αποτελέσματα της μπορούν να χρησιμοποιηθούν σαν να ήταν η cache ο αντίστοιχος SN. Για παράδειγμα, έστω ότι η cache από προηγούμενο ερώτημα έχει αποθηκευμένο το extended Skyline τριών διαστάσεων. Έστω επίσης, ότι γίνεται ένα Skyline ερώτημα που αφορά δύο διαστάσεις οι οποίες είναι υποσύνολο των τριών. Αυτό που θα έκανε ο SN είναι να χρησιμοποιούσε το extended Skyline όλων των διαστάσεων για να εξάγει το αντίστοιχο των δύο. Αφού όμως έχει ήδη αποκλείσει κάποια σημεία για να βρει το extended Skyline των τριών διαστάσεων, δεν υπάρχει λόγος να κάνει ξανά τον υπολογισμό για τα σημεία που ήδη απέκλεισε.

Έτσι όταν ένας SN λαμβάνει ερώτημα ελέγχει τα παρακάτω:

- Αν υπάρχει υπερσύνολο του ερωτήματος υπολογισμένο στην cache.
- Αν υπάρχουν περισσότερα από ένα επιλέγει αυτό με τις λιγότερες διαστάσεις. Χρησιμοποιεί αυτό το υπερσύνολο για να υπολογίσει το ερώτημα που έλαβε. Με αυτό τον τρόπο γλιτώνει άσκοπους υπολογισμούς για ήδη αποκλεισμένα σημεία.

- Αν δεν υπάρχει υπερσύνολο το υπολογίζει κανονικά από το extended Skyline όλων των διαστάσεων.

4.3.2 Χρησιμοποίηση Ερωτήσεων Μικρότερων Διαστάσεων για Απάντηση Γρήγορων Ερωτημάτων Skyline

Αντίστοιχα με τα παραπάνω ένας SN μπορεί να έχει ήδη υπολογισμένο στην cache του το αποτέλεσμα μιας Skyline ερώτησης το οποίο εμπεριέχει λιγότερες διαστάσεις από αυτές που του ζητούνται. Αυτό το αποτέλεσμα είναι σίγουρα ελλιπές, όμως τα στοιχεία του είναι κομμάτι της απάντησης. Μάλιστα για να βρεθεί αυτό το κομμάτι της απάντησης, δεν χρειάζεται να γίνει κανένας υπολογισμός διότι είναι ήδη υπολογισμένο. Επιπλέον, αν υπάρχουν περισσότερα του ενός ερωτήματα, τα οποία εμπεριέχουν διαστάσεις οι οποίες είναι υποσύνολο του ζητούντος ερωτήματος, τότε τα αποτελέσματα των ερωτημάτων αυτών είναι όλα υποσύνολα της απάντησης. Τα υποσύνολα αυτά δεν είναι ξένα μεταξύ τους. Η ένωση αυτών των απαντήσεων, ακόμα και αν οι επί μέρους απαντήσεις περιέχουν όλες τις διαστάσεις του ερωτήματος, δεν είναι το αποτέλεσμα του extended Skyline που αφορά όλες μαζί τις διαστάσεις. Θυμηθείτε το παράδειγμα της Παραγράφου 2.4. Υπενθυμίζουμε ότι έχουμε τέσσερα σημεία α(1,1,2), β(3,2,1), γ(4,1,1), δ(2,3,2). Το αποτέλεσμα του extended Skyline που αναφέρεται στις δύο πρώτες διαστάσεις, περιέχει τα σημεία β, γ και δ. Το αποτέλεσμα του αντίστοιχου ερωτήματος που αφορά την πρώτη μόνο διάσταση είναι το σημείο γ, ενώ το ερώτημα που αφορά τη δεύτερη είναι μόνο το σημείο δ. Όπως παρατηρούμε από το παράδειγμα η ένωση των δύο τελευταίων συνόλων δεν είναι ολοκληρωμένη απάντηση του ερωτήματος που αφορά και τις δύο διαστάσεις. Αν τώρα ζητάμε το extended Skyline όλων των διαστάσεων του οποίου το αποτέλεσμα είναι όλα τα σημεία και έχουμε υπολογισμένο το extended Skyline των δύο πρώτων διαστάσεων (σημεία β, γ, δ) και το extended Skyline της τελευταίας διάστασης (σημεία α, δ), παρατηρούμε ότι η ένωση τους θα περιέχει το σημείο δ δύο φορές με αποτέλεσμα να εμφανίζονται διπλότυπα στο χρήστη.

Συγκεντρωτικά, έστω ότι έχουμε δύο ερωτήματα q1 και q2 με διστάσεις U και V αντίστοιχα. Έστω επίσης ένα ερώτημα q3 με διαστάσεις $D = U \cup V$. Για την ένωση των extended Skyline των q1 και q3 ισχύουν:

- το αποτέλεσμα μπορεί να είναι ελλιπές σε σχέση με το extended Skyline του q3
- το αποτέλεσμα μπορεί να περιέχει διπλότυπα

Παρότι τα αποτελέσματα των παραπάνω ενεργειών δεν είναι ακριβή και ολοκληρωμένα αποτελέσματα ενός extended Skyline ερωτήματος, έχουν μία πολύ καλή ιδιότητα: «Δεν χρειάζεται να γίνει κανένας υπολογισμός για την εξαγωγή τους». Αυτό είναι και το χαρακτηριστικό που μας ενδιαφέρει ιδιαίτερα. Εισάγουμε την έννοια του fast Skyline query.

Ορισμός 5.1
Fast Skyline query είναι ένα Skyline ερώτημα για το οποίο δεν απαιτούνται ακριβείς και ολοκληρωμένες απαντήσεις.

Όταν ένας SN λαμβάνει ένα τέτοιο ερώτημα απαιτείται να δώσει όσο πιο γρήγορα μπορεί μία απάντηση, η οποία μπορεί να μην είναι ολοκληρωμένη, είναι όμως είτε μέρος της απάντησης του ερωτήματος. Αν ο κόμβος που θέτει το Skyline ερώτημα είναι ιδιαίτερα βιαστικός και να δεν ενδιαφέρεται για ακριβείς απαντήσεις μπορεί να θέσει ένα fast Skyline query.

Αν ένας SN λάβει ένα fast Skyline query τότε:

- Ελέγχει αν υπάρχει υπολογισμένη απάντηση για υποσύνολο των διαστάσεων του ερωτήματος.
- Αν υπάρχουν περισσότερες από μία, τότε επιλέγει τόσες ερωτήσεις έτσι ώστε ο συνδυασμός τους αλλά και κάθε μία ξεχωριστά να περιέχει όσο το δυνατόν περισσότερες διαστάσεις. Κίνητρο αυτής της τακτικής είναι ότι το extended Skyline περιέχει περισσότερα σημεία όταν αφορά περισσότερες διαστάσεις. Φυσικά κάποιες διαστάσεις έχουν αδικηθεί σε αυτά τα αποτελέσματα, αλλά δεν υπάρχει γρήγορη απάντηση χωρίς κάποιο κόστος.

- Αν δεν υπάρχει απάντηση για κανένα υποσύνολο των διαστάσεων, τότε κάνει ότι και στην περίπτωση ενός απλού ερωτήματος με επαναχρησιμοποίηση αποτελεσμάτων μεγαλύτερης διάστασης.

Δίνουμε ένα παράδειγμα για να γίνει πιο κατανοητή η επιλογή των ερωτημάτων που θα χρησιμοποιηθούν στην απάντηση. Ζητάμε ερώτημα τριών διαστάσεων και έχουμε υπολογισμένα τα εξής: α) το αποτέλεσμα για d1, β) d2, γ) d3, δ) d1 και d2, ε) d2 και d3. Θα μπορούσαν να χρησιμοποιηθούν τα α, β, και γ ή τα α και ε ή τα γ και δ. Με αυτόν τον τρόπο θα είχαμε και τις τρεις διαστάσεις στο αποτέλεσμα. Όμως όπως περιγράφηκε πριν, πρέπει να επιλέξουμε τον συνδυασμό εκείνον, ο οποίος έχει τις περισσότερες διαστάσεις συνολικά (εδώ όλα έχουν τρεις), αλλά και τις περισσότερες ξεχωριστά. Αν επιλέξουμε τα α, β, και γ ξεχωριστά έχουμε από μία μόνο διάσταση. Αν επιλέξουμε τις άλλες περιπτώσεις (α και ε ή γ και δ) έχουμε ξεχωριστά δύο και μία διάσταση (δηλαδή είναι καλύτερες από την προηγούμενη επιλογή). Μπορούμε όμως αντί αυτών να επιλέξω τα δ και ε, τα οποία εμπεριέχουν συνολικά όλες τις διαστάσεις και ξεχωριστά έχουν περισσότερες διαστάσεις από ότι οι προηγούμενες επιλογές (δύο και δύο). Αυτή η τελευταία είναι και η επιλογή που θα κάνουμε. Όσο περισσότερες είναι οι διαστάσεις που εμπεριέχονται στα υποσύνολα που θα επιλέξουμε, τόσο πιο ολοκληρωμένη θα είναι και η απάντηση που θα λάβουμε.

Ακρίβεια αποτελέσματος

Πάμε τώρα λίγο πίσω στην περίπτωση που δεν υπάρχει συνδυασμός υποσυνόλων που να περιέχει όλες τις διαστάσεις. Σε αυτή την περίπτωση απαντάμε με ένα ή περισσότερα υποσύνολα. Η ακρίβεια της απάντησης εξαρτάται καθαρά από το ποιες και πόσες θα είναι οι διαστάσεις του υποσυνόλου, καθώς και από την κατανομή που ακολουθούν τα δεδομένα. Έστω ότι έχουμε δεδομένα που ακολουθούν Independent κατανομή. Τότε είναι ξεκάθαρο ότι η ακρίβεια του αποτελέσματος εξαρτάται μόνο από το πλήθος των διαστάσεων και όχι από την επιλογή των διαστάσεων. Όπως πολλές φορές έχουμε αναφέρει όσο περισσότερες είναι οι διαστάσεις που εμπλέκονται τόσο περισσότερα είναι και τα δεδομένα που ανήκουν στο extended Skyline. Στην προκειμένη περίπτωση επειδή μιλάμε για υποσύνολο της απάντησης είναι αυτονόητο ότι με το πλήθος συνεπάγεται και η ακρίβεια. Αντίθετα όταν μιλάμε για Anti-Correlated κατανομή ή Correlated, τα αποτελέσματα είναι διαφορετικά. Στην Correlated

κατανομή η επιλογή μίας και μόνο διάστασης μπορεί να αρκεί – δίνει περίπου ίδια αποτελέσματα με την επιλογή παραπάνω διαστάσεων. Αυτό συμβαίνει επειδή λόγω της κατανομής, αν ένα σημείο είναι καλό σε μία διάσταση θα είναι καλό και στις υπόλοιπες. Άρα αν εφαρμοστεί η τεχνική αυτή όταν έχουμε Correlated κατανομή μπορεί να τροποποιηθεί λίγο (να μην επιλέγουμε δηλαδή όσο περισσότερες διαστάσεις μπορούμε, αλλά να αρκεί είτε απλά να περιέχονται όλες ή ακόμα και ένα υποσύνολο αυτών) έτσι ώστε να μην εμφανίζονται πολλά διπλότυπα στο χρήστη. Στην περίπτωση της Anti-Correlated κατανομής τα πράγματα είναι λίγο πιο περίπλοκα. Αν έχουμε πολλές διαστάσεις τότε μία εξ αυτών μπορεί να είναι πολύ χειρότερη ή πολύ καλύτερη από τις υπόλοιπες. Αν αυτή η διάσταση συμπεριλαμβάνεται στο ερώτημα, αλλά δεν συμπεριληφθεί στην απάντηση της fast query τότε τα αποτελέσματα θα είναι πολύ χειρότερα από όσο περιμέναμε επειδή η διάσταση αυτή κυρίως επηρεάζει το αποτέλεσμα. Αν γνωρίζουμε την ταυτότητα αυτής της διάστασης μπορούμε να επιλέξουμε η διάσταση αυτή να συμπεριλαμβάνεται αν είναι δυνατόν στην απάντηση, ακόμα και αν οι διαστάσεις που εμπλέκονται είναι λιγότερες σε αριθμό από κάποια άλλη πιθανή επιλογή. Αν η διάσταση αυτή δεν συμπεριλαμβάνεται στο αρχικό ερώτημα, τότε τα αποτελέσματα είναι παρόμοια με αυτά της Independent κατανομής και εξαρτώνται καθαρά από τον αριθμό των διαστάσεων.

ΚΕΦΑΛΑΙΟ 5. CONTINUOUS SKYLINE QUERIES

Σε αυτό το κεφάλαιο δίνεται ο ορισμός των συνεχών (continuous) ερωτημάτων και περιγράφεται ο τρόπος με τον οποίο το σύστημα μας μπορεί να υποστηρίξει τέτοια ερωτήματα. Δίνονται αναλυτικά οι επιπλέον δομές που πρέπει να χρησιμοποιηθούν καθώς και οι ενέργειες που πρέπει να εκτελεστούν ώστε το ερώτημα να είναι συνεχώς ενημερωμένο. Υποθέτουμε ότι υπάρχουν δύο κύριοι τρόποι ενημέρωσης των ερωτημάτων: α) είτε κάθε φορά που γίνεται μία ενημέρωση σε κάποιον κόμβο, β) είτε περιοδικά με χρονική περίοδο που ορίζεται από τον χρήστη.

5.1. Continuous Skyline Ερωτήματα

Ένα continuous query, ορίζεται ως το ερώτημα το οποίο τίθεται μία μόνο φορά στο σύστημα και έπειτα τρέχει συνεχώς στο σύστημα, σε αντίθεση με ένα απλό query, το οποίο τίθεται μία φορά και τρέχει μέχρι να ολοκληρωθεί η απάντηση [TGNO92]. Το σύστημα μας μπορεί να υποστηρίξει και continuous ερωτήσεις.

Η παραπάνω τακτική έχει αρκετά πλεονεκτήματα σε σχέση με το να τίθεται συνεχώς το ερώτημα. Πρώτον, δεν τίθενται ερωτήσεις συνεχώς στο δίκτυο, αλλά στέλνονται μόνο αποτελέσματα. Δεύτερον, τα αποτελέσματα στέλνονται μόνο όταν αυτό είναι απαραίτητο. Η

χρησιμότητα των continuous ερωτήσεων έγκειται κυρίως στο “monitoring”, δηλαδή στην παρακολούθηση των αλλαγών που συμβαίνουν στο σύστημα ανά κάποια χρονικά διαστήματα. Ως παράδειγμα μπορούμε να θέσουμε αυτό ενός ταξιδιωτικού γραφείου το οποίο π.χ. θέλει να ενημερώνεται συνεχώς για τη διαθεσιμότητα δωματίων σε ξενοδοχεία ή εισιτηρίων σε πτήσεις. Ένα ακόμη παράδειγμα θα μπορούσε να είναι η παρακολούθηση αλλαγών στο δίκτυο, ανίχνευση ανωμαλιών και επιπτώσεις αυτών.

Τα continuous ερωτήματα μπορούν να χωριστούν σε δύο κατηγορίες ανάλογα με τα γεγονότα τα οποία ξεκινούν την εκτέλεσή τους. Υπάρχουν τα change-based ερωτήματα και τα timer-based. Στα timer-based, ο χρήστης καθορίζει τη συχνότητα ενημέρωσης που τον ενδιαφέρει. Στο σύστημα γίνονται κατάλληλες ενέργειες έτσι ώστε στη δεδομένη χρονική στιγμή ο χρήστης να έχει ενημερωμένες τις απαντήσεις που ζητά. Ο τρόπος αυτός εμπεριέχει τον κίνδυνο να μην δούμε έγκαιρα ενημερώσεις οι οποίες μπορεί να είναι πολύ σημαντικές. Στα change-based ερωτήματα ο χρήστης θέλει να είναι ενήμερος για κάθε αλλαγή που συμβαίνει στο σύστημα. Με αυτόν τρόπο έχει έγκαιρα όλες τις ενημερώσεις, αλλά αυξάνεται ιδιαίτερα το υπολογιστικό κόστος, κυρίως όταν μιλάμε για πολύ μεγάλα συστήματα με συνεχείς ενημερώσεις (όπως είναι ένα δίκτυο ομότιμων). Η περίπτωση αυτή μπορεί λίγο να επεκταθεί – βελτιωθεί αν ο χρήστης αντί να βλέπει κάθε αλλαγή που συμβαίνει θέλει να ενημερώνονται τα στοιχεία που τον ενδιαφέρουν όταν π.χ. γίνουν k αλλαγές στο σύστημα. Ένας επιπλέον τρόπος ανανέωσης των continuous ερωτημάτων είναι όταν γίνεται μία αλλαγή (ή και περισσότερες) να μετράμε την απόσταση που έχει αυτή από το σύνολο των δεδομένων που βλέπει ο χρήστης. Μόνο αν αυτή η αλλαγή επηρεάζει αρκετά το σύνολο των δεδομένων (αλλάζει τη συνολική του εικόνα) τότε εμφανίζεται στο χρήστη.

Συνοψίζοντας αναφέρουμε συγκεντρωτικά τους τρόπους με τους οποίους μπορεί να ενημερώνεται ένα continuous ερώτημα, α) ανά κάποιο χρονικό διάστημα, β) κάθε φορά που συμβαίνουν k τροποποιήσεις στα δεδομένα, γ) κάθε φορά που η απάντηση επηρεάζεται κατά k% από τις αλλαγές. Στην παρούσα εργασία εξετάζουμε τις δύο πρώτες περιπτώσεις. Παρακάτω το α) αναφέρεται και ως change-based, ενώ το β) ως timer-based. Ανάλογα με το ποιος τρόπος χρησιμοποιείται γίνεται και διαμοιρασμός των continuous ερωτημάτων σε SN έτσι ώστε να είναι πιο αποδοτική η διαχείριση τους.

5.2. Continuous Ερωτήματα σε Σύστημα Ομότιμων

Υποθέτουμε ότι στο σύστημα υπάρχει ένα σύνολο από continuous queries οι οποίες είναι γνωστές και για τις οποίες ενημερώνεται κάθε κόμβος από τη στιγμή που αυτός θα συνδεθεί στο δίκτυο. Για κάθε continuous query είναι υπεύθυνος και ένας SN. Οι υπόλοιποι SN πρέπει να ενημερώνουν τον SN που είναι υπεύθυνος για ένα τέτοιο ερώτημα κάθε φορά που αλλάζει κάτι στα δεδομένα τους, το οποίο επηρεάζει την απάντηση του ερωτήματος. Ο SN που είναι υπεύθυνος έχει την υποχρέωση να διαχειρίζεται τις ενημερώσεις, έτσι ώστε να είναι συνεχώς σωστή η απάντηση στο ερώτημα. Αν κάποιος κόμβος του συστήματος θέλει την απάντηση ενός ερωτήματος που είναι γνωστό ότι είναι continuous, τότε δεν δρομολογεί το ερώτημα στο δίκτυο, αλλά απευθύνεται στον υπεύθυνο SN του ερωτήματος. Παρακάτω περιγράφουμε αναλυτικά πώς γίνονται αυτές οι λειτουργίες.

5.2.1 Δομές Αποθήκευσης Continuous Ερωτημάτων

Σε κάθε κόμβο υπάρχει μία λίστα από continuous ερωτήματα. Για κάθε τέτοιο ερώτημα υπάρχει πληροφορία για τις διαστάσεις που το αφορούν και πληροφορία για το ποιος SN είναι υπεύθυνος για αυτό.

Κάθε SN, ο οποίος είναι υπεύθυνος για ένα ή περισσότερα continuous ερωτήματα έχει μία ξεχωριστή cache που διατηρεί αυτά τα ερωτήματα. Την cache αυτή θα την ονομάζουμε continuous cache έτσι ώστε να γίνει σαφής διαχωρισμός σε σχέση με την απλή cache που διατηρούν οι SN. Για κάθε continuous ερώτηση που βρίσκεται στην cache, διατηρούμε ένα id, το οποίο είναι ίδιο για όλες τις ερωτήσεις που αφορούν τις ίδιες διαστάσεις. Επίσης αποθηκεύεται και το αποτέλεσμα της ερώτησης σε μορφή Bitmap.

5.2.2 Διαχείριση Continuous Ερωτημάτων

Για να διατηρείται ενημερωμένο ένα continuous ερώτημα πρέπει να οριστούν κάποιες λειτουργίες. Η ενημέρωση των continuous ερωτημάτων είναι παρόμοια με την ενημέρωση του extended Skyline των SN που περιγράφεται στην Ενότητα 3.3.

Όταν ένας SN λαμβάνει κάποιες ενημερώσεις, οι οποίες επηρεάζουν το τοπικό extended Skyline του, τότε στέλνει τις ενημερώσεις του σε όλους τους κόμβους οι οποίοι είναι υπεύθυνοι για κάποιο continuous ερώτημα. Με τη σειρά του, ο κόμβος ο οποίος είναι υπεύθυνος για κάποιο ερώτημα, όταν λαμβάνει ενημερώσεις για ένα συγκεκριμένο ερώτημα εκτελεί τα εξής βήματα:

- Αν λάβει επιπλέον δεδομένα, τότε ελέγχει αν ανήκουν στο extended Skyline του ερωτήματος. Αν κανένα από τα νέα δεδομένα, δεν προστεθεί στο extended Skyline, τότε δεν χρειάζεται να γίνει κανένας άλλος έλεγχος. Αν όμως έστω και ένα νέο σημείο προστεθεί, τότε προχωρά σε επανυπολογισμό του extended Skyline, αφού υπάρχει περίπτωση το νέο σημείο να κυριαρχεί κάποιο ή κάποια από τα προηγούμενα.
- Αν η ενημέρωση αφορά τη διαγραφή κάποιων σημείων, ο κόμβος που είναι υπεύθυνος για το continuous ερώτημα, ελέγχει αν τα δεδομένα που διαγράφηκαν ανήκουν στο extended Skyline του ερωτήματος. Αν τα δεδομένα αυτά δεν ανήκουν στο extended Skyline, ο SN αγνοεί την ενημέρωση. Διαφορετικά θα πρέπει να ζητήσει από όλους τους SN να στείλουν τα extended Skyline τους για υπολογιστεί ξανά το ολικό Skyline του continuous ερωτήματος. Αυτό είναι απαραίτητο, διότι τα δεδομένα που διαγράφηκαν μπορεί να απέκλεισαν κάποια σημεία από το ανήκουν στο extended Skyline και τα σημεία αυτά πρέπει τώρα να προστεθούν.

Τα κόστη των παραπάνω ενεργειών είναι ανάλογα των ενημερώσεων του extended Skyline που διατηρεί κάθε SN, μόνο που εδώ είναι σε μεγαλύτερη κλίμακα διότι το αποτέλεσμα αφορά όλο το δίκτυο.

5.3. Διαμοιρασμός Ερωτημάτων με Βάση την Ομοιότητα τους

Στην change-based προσέγγιση τα continuous ερωτήματα ενημερώνονται κάθε φορά που γίνεται μία ενημέρωση σε έναν SN. Δηλαδή αν κάποιος έχει ενημερώσεις, τότε αναλαμβάνει να τις στείλει και στους SN οι οποίοι είναι υπεύθυνοι για continuous ερωτήματα.

5.3.1 Αλγόριθμος Ομαδοποίησης

Έχουμε ένα σύνολο continuous ερωτημάτων τα οποία διαμοιράζονται στους SN του συστήματος. Ο διαμοιρασμός των ερωτημάτων μπορεί να είναι τυχαίος, μπορούμε όμως να διαμοιράσουμε τις ερωτήσεις με τέτοιον τρόπο, ώστε να εκμεταλλευτούμε τις ιδιότητες του extended Skyline, όπως αυτές αναφέρθηκαν σε προηγούμενες παραγράφους. Συγκεκριμένα, στην Παράγραφο 4.4 αναφερόμαστε στις ιδιότητες του extended Skyline όσον αφορά την επαναχρησιμοποίηση ερωτήσεων. Αν λοιπόν αναθέσουμε στους SN ερωτήσεις οι οποίες είναι «παρόμοιες» μεταξύ τους, τότε πιθανότατα δεν θα χρειαστεί οι υπόλοιποι SN του συστήματος να κάνουν υπολογισμούς και να στέλνουν πληροφορία στο δίκτυο για όλες τις ερωτήσεις, αλλά αντίθετα θα υπολογίζουν μόνο ένα υποσύνολο αυτών, το οποίο θα επαρκεί και για τον υπολογισμό των υπολοίπων.

Έστω ένα σύνολο C continuous ερωτημάτων και ένα σύνολο N από SNs. Μοιράζουμε τα ερωτήματα του συνόλου C σε |N| ξένα μεταξύ τους υποσύνολα, έτσι ώστε κάθε ένα από αυτά να ικανοποιεί το εξής:

Συνθήκη 4.1
Αν $c_1, c_2, ..., c_k$ τα στοιχεία ενός υποσυνόλου τότε πρέπει να υπάρχει διάταξη των ερωτημάτων τέτοια ώστε $c_1 \subset c_2 \subset ... \subset c_k$.

Αν το παραπάνω δεν είναι εφικτό, τότε αρκεί για τα στοιχεία των υποσυνόλων να ισχύει το εξής:

Συνθήκη 4.2

Αν $c_1, c_2, ..., c_k$ τα στοιχεία ενός υποσυνόλου τότε πρέπει να υπάρχει c_k τέτοιο ώστε $c_1 \subset c_k, c_2 \subset c_k, ...c_{k-1} \subset c_k$. Δηλαδή τουλάχιστον ένα στοιχείο να είναι υπερσύνολο όλων των άλλων.

Για να πετύχουμε το διαμοιρασμό αυτό χρησιμοποιούμε έναν πολύ απλό (και όχι βέλτιστο) τρόπο.

- Χωρίζουμε το σύνολο C των ερωτημάτων ανάλογα με τον αριθμό των διαστάσεων που το κάθε ένα περιλαμβάνει. Δηλαδή σε σύνολα των 1, 2 κτλ διαστάσεων. Έστω ότι τα ονομάζουμε C_1, C_2, …, C_N, όπου N είναι ο μέγιστος αριθμός διαστάσεων που μπορούμε να έχουμε. Το κάθε ένα από αυτά τα σύνολα δεν μπορεί να ικανοποιεί ούτε την Συνθήκη 4.1, ούτε τη Συνθήκη 4.2.
- Από αυτά τα σύνολα επιλέγουμε αυτό με τα περισσότερα στοιχεία έστω C_k. Κάθε στοιχείο αυτού του συνόλου θα αποτελέσει την αρχή για τα σύνολα που θέλουμε να δημιουργήσουμε. Δημιουργούμε δηλαδή $|C_k|$ σύνολα, κάθε ένα από τα οποία περιλαμβάνουν ένα στοιχείο του C_k.
- Έπειτα παίρνουμε όλα τα υπόλοιπα σύνολα C_1, C_2, …, C_i, i=1,..N i≠k με τη σειρά, ξεκινώντας από αυτό που έχει τις λιγότερες διαστάσεις. Παίρνουμε ένα ένα τα στοιχεία των συνόλων και τα αναθέτουμε στα $|C_k|$ σύνολα, έτσι ώστε να τηρείται η Συνθήκη 4.1 είτε η 4.2. Αν δεν ισχύει η συνθήκη για καμία ανάθεση, τότε δημιουργούμε νέα σύνολα του ενός στοιχείου.

Αν τα σύνολα είναι λιγότερα από τον αριθμό |N| τότε κάποιοι SN δεν θα είναι υπεύθυνοι για κανένα ερώτημα. Αν ο αριθμός των συνόλων που έχουμε δημιουργήσει υπερβαίνει τον αριθμό |N| των SNs, τότε κάποια σύνολα θα πρέπει να συνενωθούν.

Αν απαιτείται περαιτέρω συνένωση των συνόλων των continuous ερωτημάτων είτε για εξοικονόμηση bandwidth είτε για άλλους λόγους που συντρέχουν στο δίκτυο τότε τα σύνολα που ικανοποιούν την Συνθήκη 4.2 συνενώνονται περαιτέρω ώστε να ισχύει:

Συνθήκη 4.2.1

Αν C_1, C_2, …, C_N, είναι σύνολα continuous ερωτημάτων που ικανοποιούν τη Συνθήκη 4.2 τα σύνολα συνενώνονται σε M σύνολα με τέτοιον τρόπο ώστε στις διαστάσεις των ερωτημάτων κάθε συνόλου να ισχύει η μέγιστη δυνατή επικάλυψη.

Αν $c_1, c_2, ..., c_n$ τα ερωτήματα τα οποία αποτελούν υπερσύνολο όλων των διαστάσεων για κάθε ένα σύνολο C_1, C_2, …, C_N, τότε γ ια να δημιουργηθούν σύνολα τα οποία ικανοποιούν τη Συνθήκη 4.2.1 γίνεται έλεγχος μέγιστης επικάλυψης μόνο για τα ερωτήματα αυτά.

Σημειώνουμε ότι αν τα δεδομένα έχουν 6 διαστάσεις, τότε μπορεί να γίνει αποδοτικός διαμοιρασμός των ερωτήσεων έτσι ώστε να έχουμε μόνο σύνολα που ικανοποιούν τη Συνθήκη 4.1. Πιθανότατα θα μπορούσε να βρεθεί πιο αποδοτικός αλγόριθμος διαμοιρασμού των ερωτημάτων, όμως κάτι τέτοιο δεν συμπεριλαμβάνεται στους σκοπούς αυτής της εργασίας.

5.3.2 Υπολογισμός

Κάθε SN διατηρεί μία cache continuous ερωτημάτων. Η cache αυτή περιλαμβάνει ένα ή περισσότερα σύνολα continuous ερωτήσεων έτσι όπως τα σύνολα αυτά ορίστηκαν στην προηγούμενη παράγραφο. Για κάθε ένα δηλαδή σύνολο, ισχύει είτε η Συνθήκη 4.1 είτε η Συνθήκη 4.2 είτε η Συνθήκη 4.2.1. Για κάθε σύνολο ορίζουμε έναν αντιπρόσωπο. Ο αντιπρόσωπος είναι η ερώτηση για την οποία ισχύει ότι το σύνολο των διαστάσεων που περιλαμβάνει είναι υπερσύνολο όλων των διαστάσεων των ερωτημάτων που ανήκουν σε αυτό το σύνολο. Στην περίπτωση της Συνθήκης 4.2.1 ορίζουμε ως αντιπρόσωπο ένα υποθετικό continuous ερώτημα το οποίο περιλαμβάνει όλες τις διαστάσεις που εμπεριέχονται στα ερωτήματα του συνόλου. Λόγω των ιδιοτήτων του extended Skyline, ισχύει ότι μέσω των αποτελεσμάτων του αντιπρόσωπου, μπορούμε να υπολογίσουμε και όλα τα αποτελέσματα των ερωτήσεων που ανήκουν στο ίδιο σύνολο με τον αντιπρόσωπο.

Κάθε κόμβος που συνδέεται ως SN στο δίκτυο, δεν ενημερώνεται για το σύνολο των continuous ερωτημάτων αλλά μόνο για τους αντιπροσώπους αυτών. Έτσι στέλνει ενημερώσεις που αφορούν αυτά μόνο τα ερωτήματα. Από την άλλη πλευρά, οι SN που είναι υπεύθυνοι continuous ερωτημάτων αναλαμβάνουν τον υπολογισμό των απαντήσεων για όλα τα υπόλοιπα ερωτήματα που ανήκουν στα σύνολα των αντιπροσώπων.

Για τα σύνολα για τα οποία ικανοποιείται η Συνθήκη 4.1, τα πράγματα είναι αρκετά εύκολα όσον αφορά τους υπολογισμούς. Οι ερωτήσεις εκτελούνται με τη σειρά από αυτές που περιλαμβάνουν πολλές διαστάσεις σε αυτές που περιλαμβάνουν λιγότερες. Τα σημεία που αποκλείονται από τον υπολογισμό της πρώτης ερώτησης δεν ελέγχονται ξανά στον υπολογισμό της δεύτερης αφού ήδη έχουν αποκλειστεί κ.ο.κ. Με αυτό τον τρόπο γλιτώνουμε πολλούς υπολογισμούς τους οποίους θα κάναμε αν υπολογίζαμε κάθε ερώτημα ξανά από την αρχή. Για παράδειγμα, έχουμε ένα σύνολο στο οποίο ανήκουν 6 ερωτήσεις, 6 διαφορετικών διαστάσεων, που ικανοποιούν τη Συνθήκη 4.1. Έστω ότι έχουμε ένα σύνολο 10000 σημείων τα οποία έχουν παραχθεί από Independent κατανομή. Αν για κάθε μία ερώτηση κάναμε εξ αρχής υπολογισμό τότε θα έπρεπε να ελέγξω 6*10000=60000 σημεία. Αν όμως ο υπολογισμός γίνει με τον τρόπο που μόλις περιγράφηκε, τότε λόγω των διαδοχικών σταδίων απαλοιφής χρειάζεται να εξεταστούν περίπου 26.700, δηλαδή 55,5% λιγότερα από ότι στην απλή περίπτωση. (Το αποτέλεσμα αυτό προέκυψε από πείραμα που μετρά πόσα περίπου σημεία ανήκουν στο extended Skyline with buckets για διαφορετικό αριθμό διαστάσεων).

Για τα σύνολα που ικανοποιούν τη Συνθήκη 4.2 ή την Συνθήκη 4.2.1 δεν μπορεί να γίνει η παραπάνω «απαλοιφή» σημείων. Παρόλα αυτά και σε αυτή την περίπτωση, όπως και στην προηγούμενη γλιτώνουμε τους υπολογισμούς που θα έκανε ξεχωριστά κάθε SN για την ενημέρωση πολύ περισσότερων ερωτημάτων. Ακόμη πιο σημαντικό είναι ότι εξοικονομούμε το bandwidth του δικτύου από τη στιγμή που στέλνονται ακριβώς τόσα δεδομένα όσα επαρκούν για τον υπολογισμό των ερωτημάτων και όχι επιπλέον πληροφορία.

Χρησιμοποιώντας ομαδοποίηση στην change-based περίπτωση, οι υπολογισμοί (όσον αφορά τα πόσα σημεία ελέγχουμε για να δούμε αν ανήκουν στο Skyline) μειώνονται αρκετά σε σχέση με την περίπτωση που έχουμε change-based ερωτήματα αλλά δεν χρησιμοποιούμε

σύνολα. Το κέρδος υπολογισμών που έχουμε σε αυτή την περίπτωση εξαρτάται από το πόσο επικαλύπτονται οι διαστάσεις των continuous ερωτημάτων που θέλουμε να υπολογίσουμε. Στο Κεφάλαιο 6, δείχνουμε ότι όταν δεν υπάρχει επικάλυψη δεν έχουμε κέρδος σε υπολογισμούς, ενώ όταν υπάρχει πλήρης επικάλυψη μπορούμε να φτάσουμε σε μέχρι και 50% λιγότερους υπολογισμούς σημείων.

5.4. Διαμοιρασμός Ερωτημάτων με Βάση την Περίοδο Ενημέρωσης

Μερικές φορές τα continuous ερωτήματα δεν θέλουν να είναι συνεχώς ενημερωμένα, αλλά να ενημερώνονται ανά κάποια χρονική περίοδο. Αυτό γίνεται κυρίως για να μην υπάρχει πολύ φόρτος στο δίκτυο λόγω των ερωτημάτων αυτών. Σε αυτή την περίπτωση οι SN ελέγχουν αν έχουν ενημερώσεις την χρονική περίοδο που απαιτείται και μόνο αυτοί που έχουν στέλνουν δεδομένα στους υπεύθυνους για τα continuous ερωτήματα SNs.

5.4.1 Αλγόριθμος Ομαδοποίησης

Εδώ δεν μπορούν να χρησιμοποιηθούν απευθείας τα σύνολα ερωτήσεων που ορίσαμε στην Παράγραφο 5.3.1. Αυτό συμβαίνει επειδή κάθε ερώτηση μπορεί να θέλει να ενημερώνεται σε διαφορετική χρονική περίοδο από την περίοδο ενημέρωσης του αντιπροσώπου. Μπορούμε βέβαια να ορίσουμε ως περίοδο ενημέρωσης του συνόλου, την μικρότερη περίοδο ενημέρωσης που υπάρχει στα ερωτήματα του συνόλου. Αυτό όμως δεν είναι πολύ καλή λύση επειδή θα μεταφέρεται πολύ συχνά στο δίκτυο πληροφορία η περισσότερη από την οποία θα είναι άχρηστη. Ένας ακόμη τρόπος είναι αφού χωρίσουμε τις ερωτήσεις σε σύνολα με τον τρόπο που περιγράφηκε παραπάνω, να ορίσουμε εμείς χρόνο ενημέρωσης όλου του συνόλου. Έτσι θα μπορούμε να οργανώσουμε τις ερωτήσεις για να μην φορτώνονται όλοι οι κόμβοι του δικτύου την ίδια χρονική στιγμή. Παρόλα αυτά, συνήθως οι χρόνοι ενημέρωσης continuous ερωτημάτων καθορίζονται από το χρήστη. Στην παράγραφο αυτή προσπαθούμε να κάνουμε έναν διαφορετικό διαμοιρασμό των continuous ερωτημάτων έτσι ώστε να μας βοηθά στην περίπτωση που κάθε ένα από αυτά έχει μία διαφορετική περίοδο ενημέρωσης.

Έστω ένα σύνολο C continuous ερωτημάτων και ένα σύνολο N από SNs. Μοιράζουμε τα ερωτήματα του συνόλου C σε |N| ξένα μεταξύ τους υποσύνολα, έτσι ώστε κάθε ένα από αυτά να ικανοποιεί το εξής:

Συνθήκη 4.3

Αν $t_1, t_2, ..., t_k$ οι περίοδοι ενημέρωσης των στοιχείων ενός υποσυνόλου τότε πρέπει να ισχύει ότι $t_1 = t_2 = ... = t_k$.

Αν το παραπάνω δεν είναι εφικτό, τότε αρκεί για τα στοιχεία των υποσυνόλων να ισχύει το εξής:

Συνθήκη 4.4

Αν $t_1, t_2, ..., t_k$ οι περίοδοι ενημέρωσης των στοιχείων ενός υποσυνόλου τότε πρέπει να υπάρχουν ακέραιοι $a_i > 0$ έτσι ώστε να ισχύει: $a_1 * t_1 = a_2 * t_2 = ... = a_k * t_k$.

Δηλαδή όταν ισχύει η Συνθήκη 4.4, οι περίοδοι ενημέρωσης των ερωτήσεων ενός συνόλου είναι η μία πολλαπλάσια της άλλης. Με άλλα λόγια, αν έχουμε ερωτήματα με περίοδο 2, 4, αυτά θα μπούνε μαζί και σε άλλο σύνολο θα μπούνε ερωτήματα με περίοδο 3, 6, 9.

Επιπλέον, μπορούμε να δημιουργήσω σύνολα τα οποία ικανοποιούν τη Συνθήκη 4.3 και επιπλέον ικανοποιούν και τη Συνθήκη 4.1. Με αυτόν τον τρόπο μπορούμε να χρησιμοποιήσουμε και σε αυτή την περίπτωση τις καλές ιδιότητες που η Συνθήκη 4.1 μου προσφέρει. Ακόμα και αν αυτό φαίνεται δύσκολο και ότι είναι δύσκολο να ισχύει, μπορούμε εύκολα να πούμε ότι σίγουρα δεν χάνουμε αν ακολουθήσουμε αυτήν την τακτική. Σκεφτείτε τα παρακάτω: Στο σύστημα που έχουμε υποθέσει στην εργασία αυτή τα δεδομένα έχουνε το πολύ 6 διαστάσεις. Θεωρήσαμε ότι ο αριθμός αυτός είναι αρκετός για να περιγράψει τα πιο συνηθισμένα σύνολα δεδομένων. Με 6 διαστάσεις μπορούμε να έχουμε το πολύ 63 διαφορετικές Skyline ερωτήσεις. Επίσης υποθέτουμε ότι για μία συγκεκριμένη ερώτηση υπάρχει συγκεκριμένος χρόνος ανανέωσης ο οποίος είναι ίδιος για όλο το δίκτυο (δεν μπορούμε δηλαδή να έχουμε 2 ίδιες ερωτήσεις με διαφορετικό χρόνο ανανέωσης). Τέλος στα

περισσότερα πειράματα μας έχουμε υποθέσει ότι ο αριθμός των SN είναι ίσος με 100. Σε ένα πραγματικό δίκτυο ο αριθμός αυτός θα είναι κατά πολύ μεγαλύτερος. Το συμπέρασμα από όλα αυτά είναι ότι ακόμη και αν κάθε ερώτηση έχει διαφορετικό χρόνο ανανέωσης και η Συνθήκη 4.3 δεν μπορεί να ισχύει, τότε μπορώ να έχουμε 63 διαφορετικά σύνολα, τα οποία θα αναθέσω σε 63 διαφορετικούς κόμβους. Αν όμως η Συνθήκη 4.3 ισχύει για αρκετές ερωτήσεις, τότε είναι πολύ πιθανό να βρούμε σύνολα για τα οποία να ισχύει και η Συνθήκη 4.1.

Ο τρόπος με τον οποίο μπορεί να γίνει ο διαμοιρασμός αυτός είναι απλός.

5.4.2 Υπολογισμός

Ως αντιπρόσωπο του κάθε συνόλου ορίζουμε ένα υποθετικό continuous Skyline ερώτημα το οποίο ρωτά για όλες τις διαστάσεις που συμπεριλαμβάνονται στα ερωτήματα του συνόλου. Το extended Skyline του αντιπροσώπου αρκεί για να υπολογίσω τα αποτελέσματα όλων των ερωτήσεων που συμπεριλαμβάνονται σε ένα σύνολο. Ως περίοδο ενημέρωσης του συνόλου ορίζουμε την μικρότερη περίοδο ενημέρωσης των ερωτημάτων που συμπεριλαμβάνει. Όμοια με την change-based περίπτωση, οι SN γνωρίζουν μόνο τον αντιπρόσωπο κάθε συνόλου και στέλνουν ενημερώσεις μόνο για αυτό.

Ανάλογα με το χρόνο στον οποίο λαμβάνεται μια ενημέρωση από έναν SN ο οποίος είναι υπεύθυνος για ένα υποσύνολο, υπολογίζονται τα αντίστοιχα ερωτήματα. Το μειονέκτημα αυτής της τεχνικής είναι ότι στέλνω περισσότερα δεδομένα ακόμα και σε περιόδους που δεν τα χρειάζομαι. Π.χ. έχουμε ένα σύνολο στο οποίο ανήκουν δύο ερωτήματα, από τα οποία το πρώτο αφορά τις διαστάσεις 1, 2, 3 ενώ το δεύτερο αφορά μόνο τη διάσταση 1. Το πρώτο έχει περίοδο 4 ενώ το δεύτερο 2. Το πρώτο ερώτημα είναι και ο αντιπρόσωπος του συνόλου επειδή ικανοποιεί τη συνθήκη ότι συμπεριλαμβάνει τις διαστάσεις όλων των ερωτημάτων του συνόλου. Άρα κάθε χρόνο 2 θα έρχονται ενημερώσεις οι οποίες αφορούν τις διαστάσει 1,2,3. Παρόλα αυτά, τις μισές φορές οι διαστάσεις 2 και 3 θα ήταν άχρηστες. Το συγκεκριμένο παράδειγμα είναι πολύ απλό και θα μπορούσε προφανώς να δρομολογηθεί με πιο έξυπνο

τρόπο. Συνήθως όμως οι περιπτώσεις που έχουν να αντιμετωπιστούν είναι αρκετά περίπλοκες. Ένας απλός χειρισμός είναι καλή λύση για ένα τεράστιο δίκτυο όπου πρέπει να συντονιστούν εκατοντάδες κόμβοι.

Όσον αφορά τους υπολογισμούς του Skyline, είναι προφανές ότι στην timer-based περίπτωση και με χρόνο ενημέρωσης μεγαλύτερο από το χρόνο ενημέρωσης του συστήματος, θα γίνουν πολύ λιγότερες φορές οι υπολογισμοί για ένα δοθέν ερώτημα. Αν διπλασιάσω π.χ. την περίοδο ενημέρωσης, τότε θα γίνουν οι μισοί υπολογισμοί κτλ. Βέβαια το περιεχόμενο του ερωτήματος δεν θα είναι πάντα φρέσκο, αλλά το πόσο ενημερωμένο πρέπει να είναι εξαρτάται κυρίως από τις ανάγκες του χρήστη του συστήματος.

ΚΕΦΑΛΑΙΟ 6. ΠΕΙΡΑΜΑΤΙΚΑ ΑΠΟΤΕΛΕΣΜΑΤΑ

Στο κεφάλαιο αυτό περιγράφονται τα πειράματα που έγιναν για τους σκοπούς αυτής της εργασίας και δίνονται γραφικά τα αποτελέσματα τους. Αρχικά περιγράφεται ο τρόπος με τον οποίο παρήχθησαν τα δεδομένα που χρησιμοποιήθηκαν στα πειράματα. Έπειτα συγκρίνονται οι διαφορές του extended Skyline με το Skyline, και περιγράφεται το πώς ο αριθμός των κάδων της Bitmap αναπαράστασης επηρεάζει τα αποτελέσματα του Skyline και extended Skyline αντίστοιχα. Περιλαμβάνεται ένα πείραμα που αφορά τον ευριστικό τρόπο κατανομής των κάδων. Όσον αφορά τις ενημερώσεις, εξετάζεται το πώς το κόστος τους εξαρτάται από την κατανομή των δεδομένων. Επίσης, εξετάζεται το πώς επηρεάζουν διάφοροι παράγοντες την απόδοση της cache. Τέλος παρουσιάζονται γραφικές παραστάσεις οι οποίες συγκρίνουν την απόδοση των δύο μεθόδων που προτείνουμε για υποστήριξη continuous ερωτημάτων.

6.1. Παραγωγή Δεδομένων

Για τη διεξαγωγή των πειραμάτων, πρέπει να παραχθούν δεδομένα σε κάθε έναν από τους ομότιμους. Τα δεδομένα μπορούν να παραχθούν αλλάζοντας τη μεταξύ τους συσχέτιση, τον αριθμό των διαστάσεων, και το πλήθος-μέγεθος των δεδομένων. Στην εργασία αυτή

χρησιμοποιήθηκαν τρεις διαφορετικές κατανομές για την παραγωγή των δεδομένων. Οι κατανομές αυτές περιγράφονται αναλυτικά στο [BoKS01].

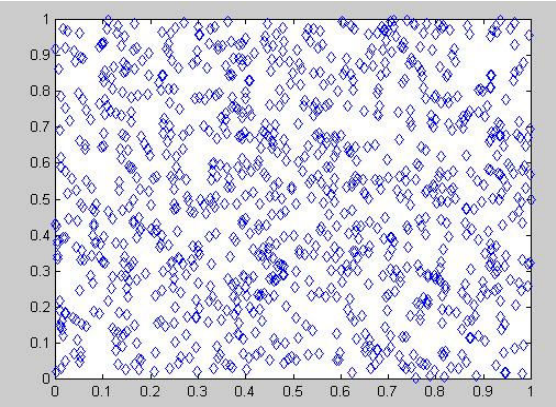
6.1.1 Κατανομές

- Independent: για αυτόν τον τύπο της βάσης δεδομένων, όλα τα χαρακτηριστικά-γνωρίσματα παράγονται ανεξάρτητα χρησιμοποιώντας ομοιόμορφη κατανομή.
- Correlated: μία συσχετισμένη βάση αναπαριστά ένα περιβάλλον στο οποίο τα σημεία που είναι καλά σε μία διάσταση είναι καλά και στις υπόλοιπες διαστάσεις. Αυτό γίνεται με τον εξής τρόπο: αρχικά χρησιμοποιώντας κανονική κατανομή βρίσκουν ένα σημείο στο διάστημα (0,1). Η κατανομή αυτή είναι φτιαγμένη έτσι ώστε τα σημεία που βρίσκονται στη μέση να επιλέγονται με μεγαλύτερη πιθανότητα σε σχέση με αυτά που βρίσκονται στα άκρα. Το επόμενο σημείο παράγεται χρησιμοποιώντας πάλι κανονική κατανομή. Αυτή τη φορά όμως μεγαλύτερη είναι η πιθανότητα να επιλεγεί σημείο κοντά στο προηγούμενο. Συνήθως, όταν τα δεδομένα παράγονται με αυτή την κατανομή, τα σημεία που ανήκουν στο Skyline είναι πολύ λίγα.
- Anti-correlated: μία αντί-συσχετισμένη βάση δεδομένων αναπαριστά ένα περιβάλλον στο οποίο τα σημεία που είναι καλά σε μία διάσταση τείνουν να μην είναι καλά στις υπόλοιπες. Χρησιμοποιείται κανονική κατανομή με μικρή διακύμανση, έτσι ώστε όλα τα σημεία να βρίσκονται κοντά στη μέση. Οι τιμές των χαρακτηριστικών παράγονται χρησιμοποιώντας ομοιόμορφη κατανομή. Όταν τα δεδομένα παράγονται με αυτή την κατανομή, τότε τα σημεία που ανήκουν στο Skyline είναι πάρα πολλά.

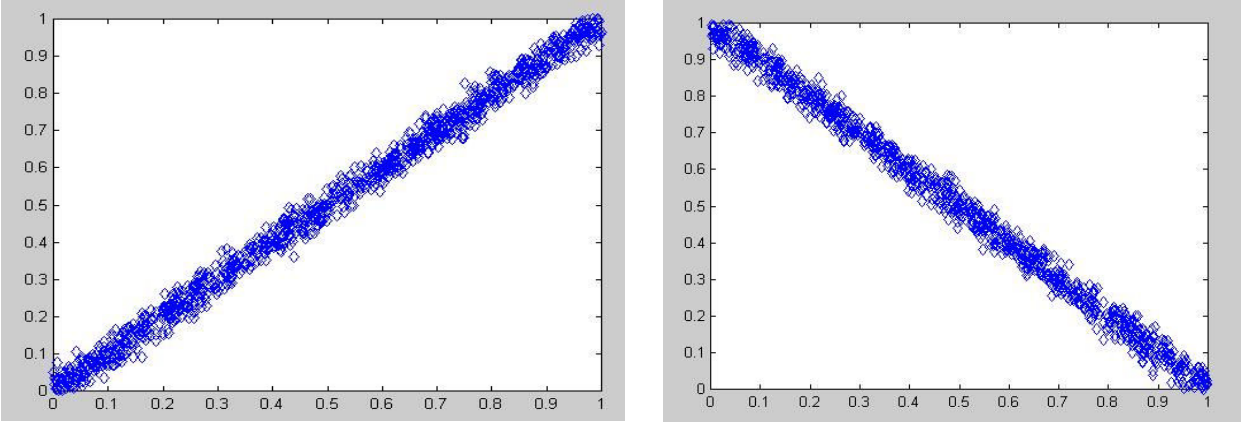
Τα παραπάνω χρησιμοποιούνται για παραγωγή δεδομένων σε μία μόνο βάση. Σε ένα σύστημα ομότιμων μπορούν να χρησιμοποιηθούν συνδυασμοί των παραπάνω, όπως για παράδειγμα να χρησιμοποιείται μία μόνο περίπτωση για όλους τους ομότιμους (π.χ. όλα τα δεδομένα παράγονται ανεξάρτητα) ή να έχουμε συνδυασμούς περιπτώσεων (π.χ. κάποιοι ομότιμοι έχουν ανεξάρτητα δεδομένα, κάποιοι συσχετισμένα και κάποιοι αντί-συσχετισμένα, όπου τα ποσοστά θα αλλάζουν ανάλογα με την περίσταση). Σε αυτή την εργασία μελετάμε

μόνο περιπτώσεις όπου τα δεδομένα όλων των ομότιμων έχουν παραχθεί από την ίδια κατανομή.

Στα Σχήματα 6.1, και 6.2 απεικονίζονται γραφικά οι κατανομές αυτές για 100 σημεία δύο διαστάσεων που έχουν παραχθεί στο διάστημα [0,1]. Για την Independent κατανομή του Σχλήματος 6.1, ο βαθμός συσχέτισης (correlation) είναι 0.011455, δηλαδή οι τιμές των δύο διαστάσεων δεν συσχετίζονται μεταξύ τους. Αντίθετα για την Correlated ο βαθμός συσχέτισης είναι 0.996388, κάτι που σημαίνει ότι οι δύο διαστάσεις είναι πλήρως συσχετισμένες ενώ για την Anti-corelated ο βαθμός συσχέτισης είναι -0.996045, δηλαδή το αντίθετο.



Σχήμα 6.1 Independent Distribution



Σχήμα 6.2 Correlated και Anti-Correlated Distribution

6.1.2 Υλοποίηση

Από το βιβλίο "Numerical Recipes" [PFTV92] πήραμε τη συνάρτηση ran1 για παραγωγή ομοιόμορφη κατανομής. Η ran1 σε αντίθεση με τη rand που προσφέρει το σύστημα, παράγει τυχαίους αριθμούς που δεν είναι συσχετισμένοι γραμμικά. Η ran1 χρησιμοποιείται για την παραγωγή ανεξάρτητων δεδομένων. Επίσης χρησιμοποιείται ως βάση για να φτιαχτούν οι άλλες κατανομές.

Για την παραγωγή δεδομένων που χρησιμοποιούν Independent κατανομή, χρησιμοποιείται απλά η ran1.

Για την παραγωγή θετικά συσχετισμένων δεδομένων γίνεται το εξής: Αν για παράδειγμα τα δεδομένα έχουν δύο διαστάσεις, τότε η πρώτη διάσταση παράγεται με ομοιόμορφη κατανομή χρησιμοποιώντας τη ran1, ενώ η δεύτερη διάσταση παράγεται από γκαουσιανή κατανομή με μέση τιμή την τιμή της πρώτης διάστασης και τυπική απόκλιση ανάλογη με το εύρος τιμών των δεδομένων (π.χ. αν τα δεδομένα είναι στο διάστημα (0, 100) μία καλή τιμή θα μπορούσε να είναι το 5). Αν οι διαστάσεις είναι πάνω από δύο, η τυπική απόκλιση μένει η ίδια και μέση τιμή παραμένει αυτή της πρώτης διάστασης.

Για την παραγωγή αρνητικά συσχετισμένων δεδομένων παράγουμε την πρώτη διάσταση με ομοιόμορφη κατανομή. Έστω x η τιμή της πρώτης διάστασης. Για την τιμή της δεύτερης διάστασης χρησιμοποιούμε γκαουσιανή κατανομή με τυπική απόκλιση ανάλογη με το εύρος των δεδομένων (π.χ. 0.025 αν τα δεδομένα ανήκουν στο (0,1)) και μέση τιμή το k-x αν τα δεδομένα ανήκουν στο (0,k). Αν τα σημεία έχουν πάνω από δύο διαστάσεις τότε χρησιμοποιούμε την ίδια γκαουσιανή και για τις υπόλοιπες διαστάσεις.

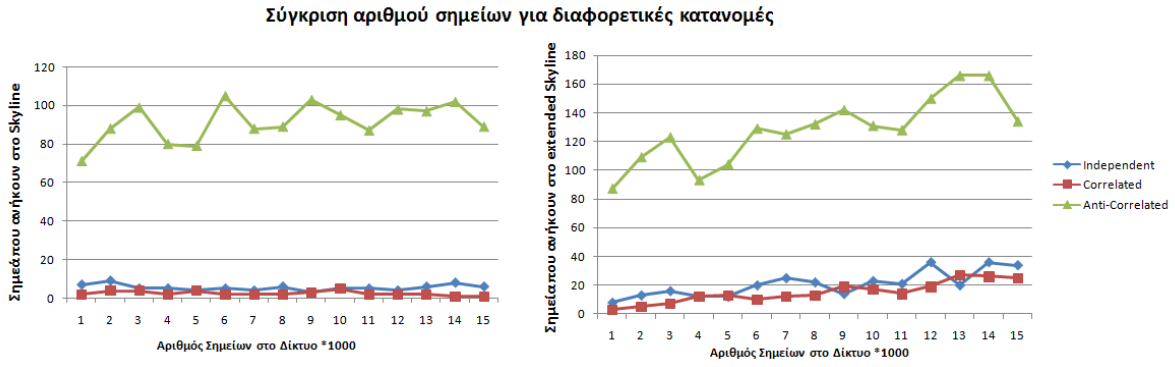
6.2. Extended Skyline

Σε αυτή την παράγραφο εξετάζεται πιο αναλυτικά το πώς η κατανομή από την οποία έχουν παραχθεί τα δεδομένα των κόμβων του δικτύου επηρεάζει τον αριθμό των δεδομένων που

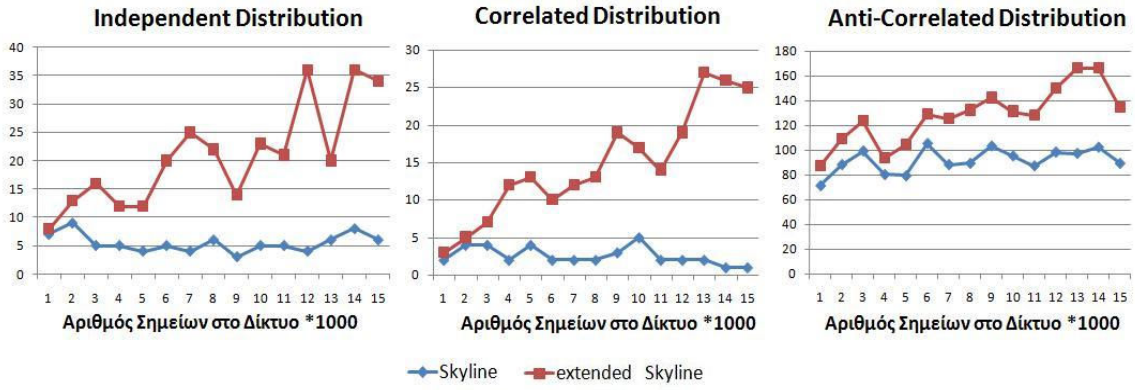
ανήκουν στο Skyline καθώς και οι διαφορές του Skyline και extended Skyline ενός συνόλου σημείων. Η υλοποίηση έγινε σε C. Στο σύστημα ομότιμων που δημιουργήθηκε κάθε SN συνδέεται τυχαία με το 5% περίπου των SN που βρίσκονται στο δίκτυο εκείνη τη στιγμή. Οι SN αποτελούν το 10% των κόμβων του δικτύου.

Για τις γραφικές παραστάσεις αυτής της παραγράφου χρησιμοποιήθηκε σύστημα ομότιμων 1000 κόμβων. Οι 100 από αυτούς είναι SN ενώ οι υπόλοιποι ON. Μεταβάλλαμε τον αριθμό των δεδομένων του δικτύου από 1000 σε 15000, ενώ κάθε φορά στους κόμβους ανατίθεται ίσος αριθμός δεδομένων. Ο υπολογισμός του Skyline χωρίς κάδους έγινε με τον BNL αλγόριθμο [BoKS01] και του extended Skyline χωρίς κάδους με μία παραλλαγή αυτού (ο αλγόριθμος μένει ίδιος αλλά αλλάζει ο ορισμός κυριαρχίας). Τα σημεία είναι δύο διαστάσεων και παράγονται στο διάστημα (0, 10).

Στο Σχήμα 6.3 εμφανίζεται το πλήθος των σημείων που ανήκουν στο Skyline και για τις τρεις κατανομές των δεδομένων (αριστερή γραφική παράσταση), ενώ στην δεξιά γραφική παράσταση εμφανίζεται ο αριθμός των σημείων που ανήκουν στο extended Skyline. Όπως παρατηρούμε και στις δύο περιπτώσεις, τα σημεία που ανήκουν στο Skyline (είτε είναι extended είτε όχι) είναι πολύ περισσότερα στην περίπτωση της Anti-Correlated κατανομής, ενώ είναι αρκετά λιγότερα στις άλλες δύο, με την Correlated να έχει λίγο λιγότερα από την Independent.



Σχήμα 6.3 Αριθμός Σημείων που ανήκουν στο Skyline και extended Skyline για Διαφορετικές Κατανομές Δεδομένων



Σχήμα 6.4 Αριθμός Σημείων που ανήκουν στο Skyline και extended Skyline χωρίς Κάδους για Διαφορετικές Κατανομές Δεδομένων

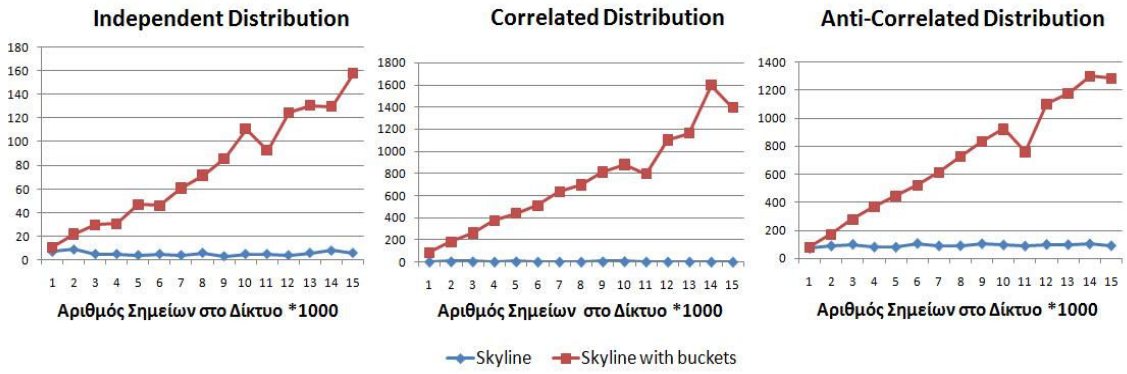
Στο Σχήμα 6.4 παρουσιάζεται πιο αναλυτικά η διαφορά των σημείων που ανήκουν στο extended Skyline και στο Skyline για κάθε κατανομή ξεχωριστά. Ανεξάρτητα από την κατανομή το πλήθος των σημείων του extended Skyline υπερβαίνει κατά πολύ το πλήθος των σημείων του Skyline. Το αποτέλεσμα αυτό είναι λογικό καθότι από το extended Skyline μπορούμε να υπολογίσουμε οποιοδήποτε υποσύνολο των διαστάσεων μας ζητηθεί.

6.3. Buckets και Skyline

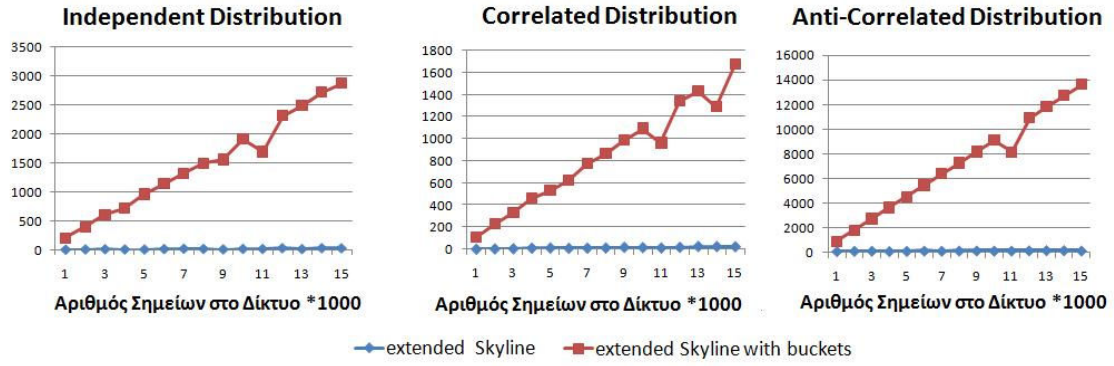
Σε αυτήν την παράγραφο εξετάζεται το κατά πόσον ο υπολογισμός του Skyline με κάδους επηρεάζει τον αριθμό των σημείων που ανήκουν σε αυτό. Για να γίνουν οι γραφικές παραστάσεις της παραγράφου αυτής χρησιμοποιήθηκε το σύστημα που περιγράφηκε στην Παράγραφο 6.2. Ο υπολογισμός των Skyline και extended Skyline με κάδους γίνεται με τον Bitmap αλγόριθμο που περιγράφεται στο Κεφάλαιο 2 όπου ο αριθμός των κάδων έχει οριστεί να είναι 10 για κάθε διάσταση.

Στο Σχήμα 6.5 εμφανίζεται ο αριθμός των σημείων του Skyline σε σχέση με τα σημεία που ανήκουν στο Skyline with buckets. Στο Skyline with buckets παρατηρούμε ότι ανήκουν περισσότερα σημεία από ότι κανονικά θα έπρεπε να ανήκουν. Στην Independent κατανομή ο αριθμός αυτός φτάνει περίπου 150, στην Correlated έως 1600, ενώ στην Anti-Correlated

φτάνει τα 1400. Αυτό συμβαίνει επειδή έχουμε μεγάλο εύρος τιμών (0, 10) με δύο δεκαδικά ψηφία, δηλ. 1000 διαφορετικές τιμές, ενώ έχουμε μικρό αριθμό από κάδους (μόνο 10). Είναι αναμενόμενο να μην έχουμε την επιθυμητή ακρίβεια επειδή σε κάθε κάδο αντί για μία, αντιστοιχούν 100 τιμές, οι οποίες εξισώνονται.



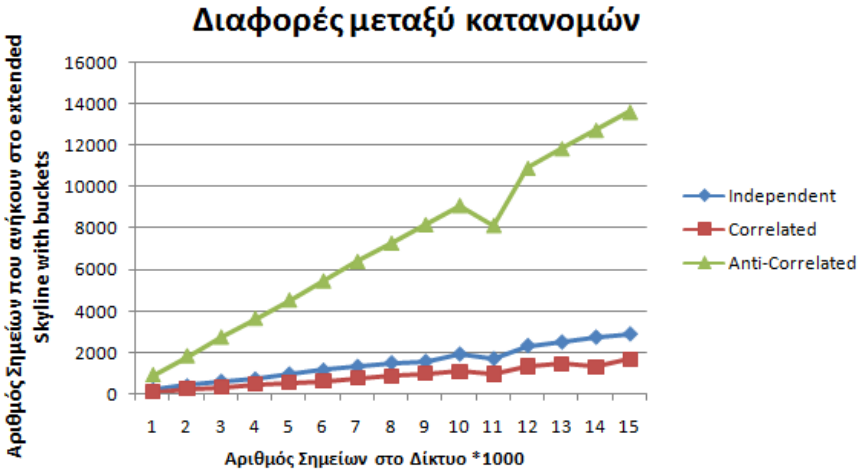
Σχήμα 6.5 Αριθμός Σημείων που ανήκουν στο Skyline και Skyline with buckets για Διαφορετικές Κατανομές Δεδομένων



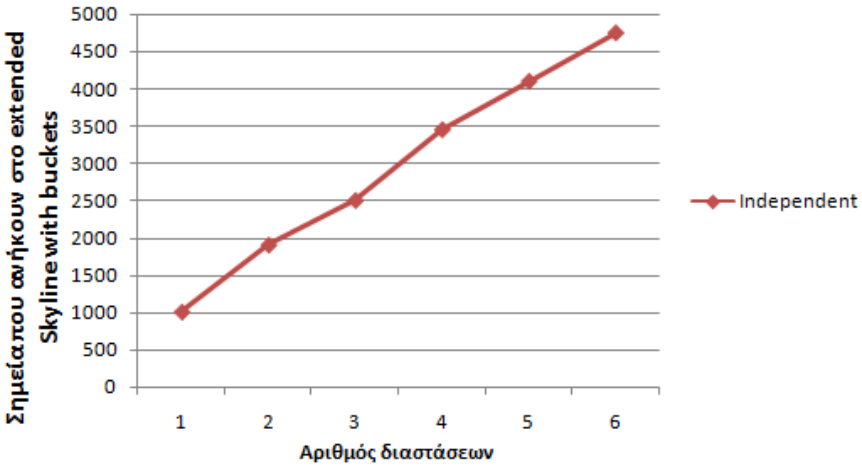
Σχήμα 6.6 Αριθμός Σημείων που ανήκουν στο extended Skyline και extended Skyline with buckets για Διαφορετικές Κατανομές Δεδομένων

Παρόμοια αποτελέσματα φαίνονται και στο Σχήμα 6.6 για το extended Skyline και το extended Skyline με buckets μόνο που σε αυτή την περίπτωση η προσθήκη κάδων κάνει πολύ χειρότερα τα αποτελέσματα. Για παράδειγμα στην Anti-Correlated κατανομή φτάνουμε στα

14000 επιπλέον σημεία. Στο Σχήμα 6.7 απεικονίζονται συγκεντρωτικά οι διαφορές μεταξύ των τριών κατανομών, όταν πρόκειται για υπολογισμό του extended Skyline με buckets. Τέλος στο Σχήμα 6.8 δείχνουμε πώς επηρεάζει τον αριθμό των δεδομένων που ανήκουν στο extended Skyline with buckets ο αριθμός των διαστάσεων των δεδομένων για Independent μόνο κατανομή.



Σχήμα 6.7 Αριθμός Σημείων που ανήκουν στο extended Skyline with buckets για Διαφορετικές Κατανομές Δεδομένων



Σχήμα 6.8 Αριθμός Σημείων που ανήκουν στο extended Skyline with buckets για Διαφορετικό Αριθμό Διαστάσεων

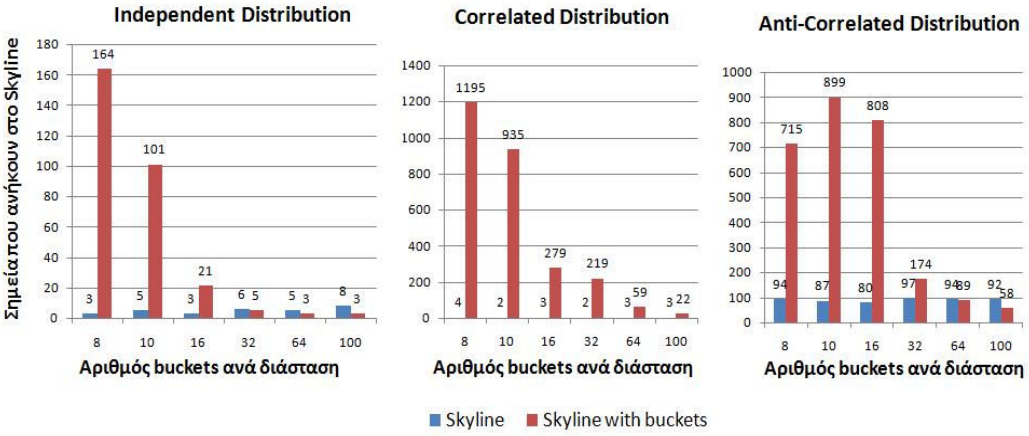
6.3.1 Μεταβολή του Αριθμού - Μεγέθους των Κάδων

Σε αυτή την παράγραφο ο αριθμός των δεδομένων στο σύστημα μένει σταθερός και μεταβάλλεται ο αριθμός των κάδων που χρησιμοποιούνται για την Bitmap αναπαράσταση. Όπως και πριν, στο σύστημα υπάρχουν 1000 κόμβοι, 100 SN και 900 ON και 10000 σημεία συνολικά, τα οποία ανατίθενται ομοιόμορφα στους κόμβους. Τα σημεία παράγονται στο (0, 10) και έχουν ακρίβεια δύο δεκαδικών ψηφίων. Ο αριθμός των κάδων είναι ίδιος για κάθε διάσταση και μεταβάλλεται σε 8, 10, 16, 32, 64, και 100.

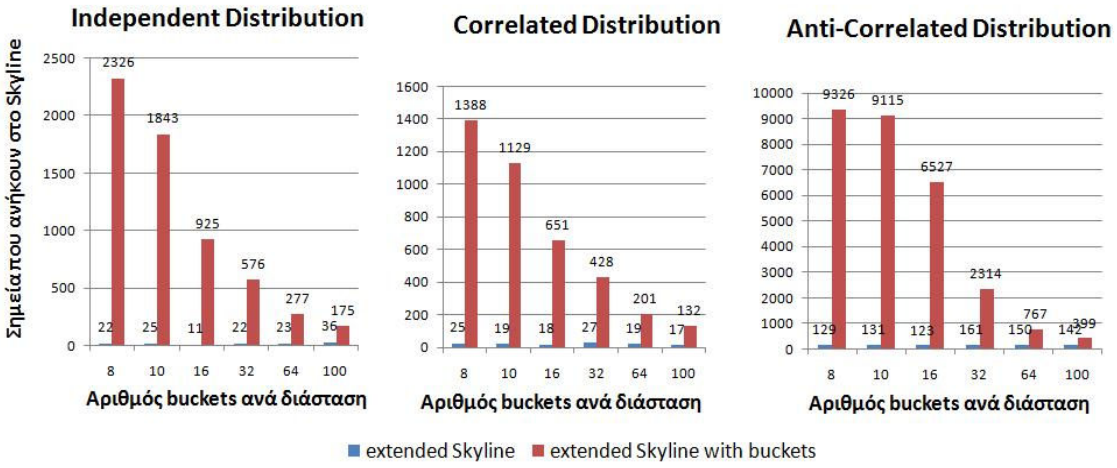
Όσο αυξάνεται ο αριθμός των κάδων τόσο μεγαλύτερη είναι και η ακρίβεια του Skyline. Μάλιστα παρατηρούμε ότι σε μερικές περιπτώσεις (Σχήμα 6.9) τα σημεία που περιέχονται στο Skyline with buckets είναι λιγότερα από αυτά που περιέχονται στο Skyline. Υπάρχουν δηλαδή false negatives. Π.χ. έστω ότι ελέγχω τα σημεία (7.56, 5.30) και (7.23, 6.71). Αν έχουμε10 κάδους και οι διαστάσεις παίρνουν τιμές στο διάστημα (0, 10) τότε: η πρώτη διάσταση θεωρείται ίδια και για τα δύο σημεία, ενώ η δεύτερη μεγαλύτερη για το δεύτερο σημείο. Έτσι μόνο το δεύτερο σημείο ανήκει στο Skyline ενώ θα έπρεπε να ανήκουν και τα δύο.

Η λύση στο να μην έχουμε false negatives είναι αντί να υπάρχει ένα σύνολο στο οποίο ανήκουν τα σημεία που είναι καλύτερα σε τουλάχιστον μία διάσταση, να υπάρχει ένα άλλο σύνολο στο οποίο να ανήκουν και αυτά που είναι ίσα σε τουλάχιστον μία διάσταση. Αυτός είναι ο ορισμός του extended Skyline, το οποίο υποθέτουμε ότι υπολογίζουν οι SN στο σύστημα μας και έτσι δεν έχουμε να αντιμετωπίσουμε αυτό το πρόβλημα. Τα αποτελέσματα για το extended Skyline εμφανίζονται στο Σχήμα 6.10. Στο Σχήμα 6.11 γίνεται μία σύγκριση του Skyline με buckets extended Skyline με buckets.

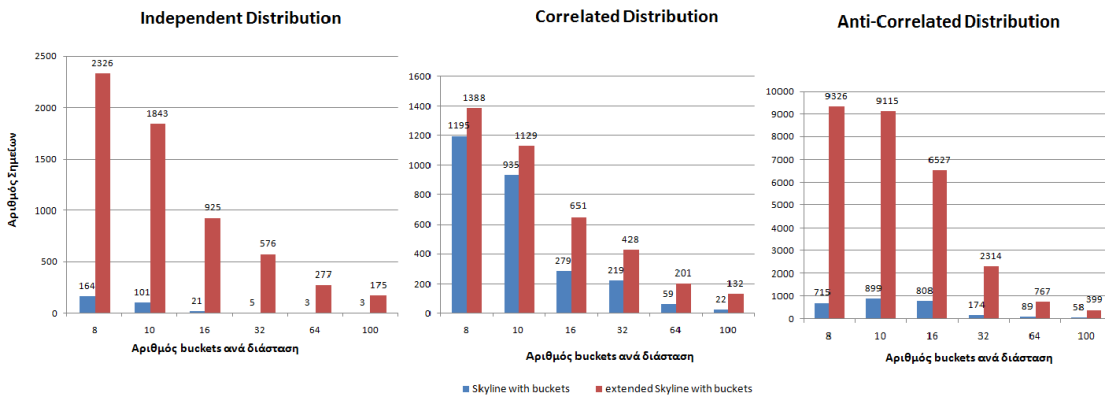
Ο αριθμός των κάδων που θα επιλέξουμε για την εκάστοτε εφαρμογή, εξαρτάται από τις προτιμήσεις του χρήστη, τη φύση των δεδομένων και την επιθυμητή ακρίβεια. Καλό είναι ο αριθμός των κάδων να μένει σε μέτρια επίπεδα, π.χ. για double να έχω 32 κάδους, όπου το κόστος της ακρίβειας δεν είναι πολύ μεγάλο.



Σχήμα 6.9 Αριθμός Σημείων που ανήκουν στο Skyline with buckets για Διαφορετικές Κατανομές Δεδομένων και Διαφορετικό Αριθμό Κάδων



Σχήμα 6.10 Αριθμός Σημείων που ανήκουν στο extended Skyline with buckets για Διαφορετικές Κατανομές Δεδομένων και Διαφορετικό Αριθμό Κάδων

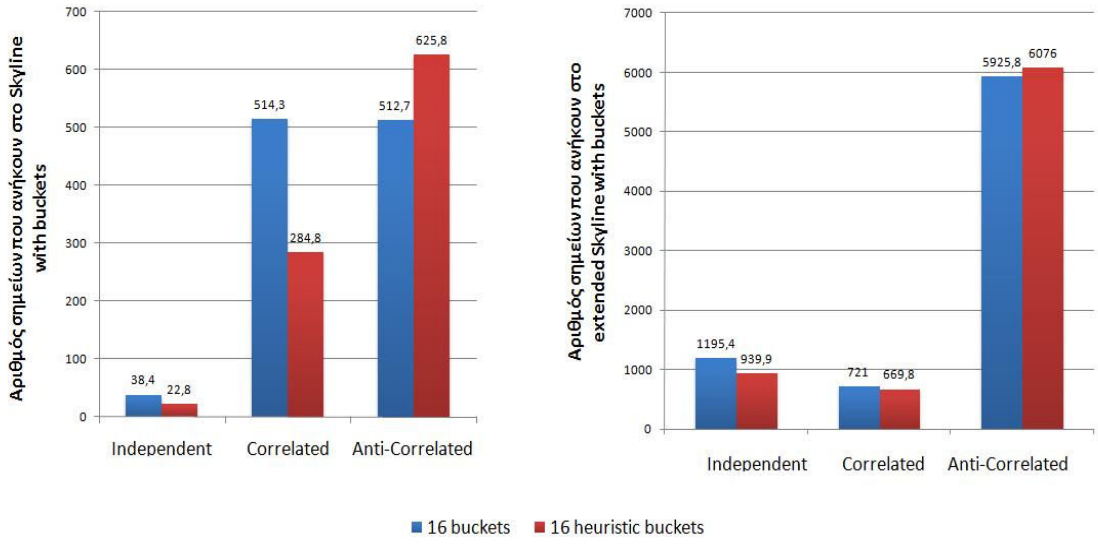


Σχήμα 6.11 Αριθμός Σημείων που ανήκουν στο Skyline with buckets και στο extended Skyline with buckets για Διαφορετικές Κατανομές Δεδομένων και Διαφορετικό Αριθμό Κάδων

6.3.2 Heuristic Buckets

Στην παράγραφο αυτή απεικονίζονται τα αποτελέσματα της heuristic bucket τεχνικής που περιγράφηκε στην Ενότητα 2.3. Στο πείραμα χρησιμοποιείται το ίδιο σύστημα με αυτό της Παραγράφου 6.3.1. Η διαφορά είναι ότι εδώ υπάρχει συγκεκριμένος αριθμός από κάδους που είναι 16. Με αυτόν τον αριθμό των κάδων έχουμε δύο περιπτώσεις. Στην πρώτη περίπτωση υπάρχουν 16 κάδοι ίσου μεγέθους, ενώ στη δεύτερη υπάρχουν 16 κάδοι που χωρίζονται με ευριστικό τρόπο. Επειδή έχουμε υποθέσει ότι ψάχνουμε το MAX για το Skyline, οι 8 κάδοι ανατίθενται στις τιμές 8-10, και οι υπόλοιποι 8 στις τιμές 0-8. Αυτό γίνεται επειδή θέλουμε στις τιμές που είναι μεγαλύτερες να έχουμε μεγαλύτερη ακρίβεια στον υπολογισμό του Skyline. Πράγματι, όπως βλέπουμε στο Σχήμα 6.12 ο αριθμός των false positive μειώνεται αρκετά για την Independent και Correlated κατανομή. Δε συμβαίνει όμως το ίδιο και για την Anti-Correlated για την οποία η περίπτωση με τους κάδους ίσου μεγέθους δουλεύει αρκετά καλύτερα. Αυτό συμβαίνει επειδή στην Anti-Correlated κατανομή τα σημεία τα οποία έχουν μεγάλη τιμή στη μία διάσταση, έχουν μικρή τιμή στην άλλη διάσταση. Έτσι για ένα σημείο αυτόματα χάνουμε την επιπλέον ακρίβεια την οποία είχαμε για την διάσταση με τη μεγάλη τιμή.

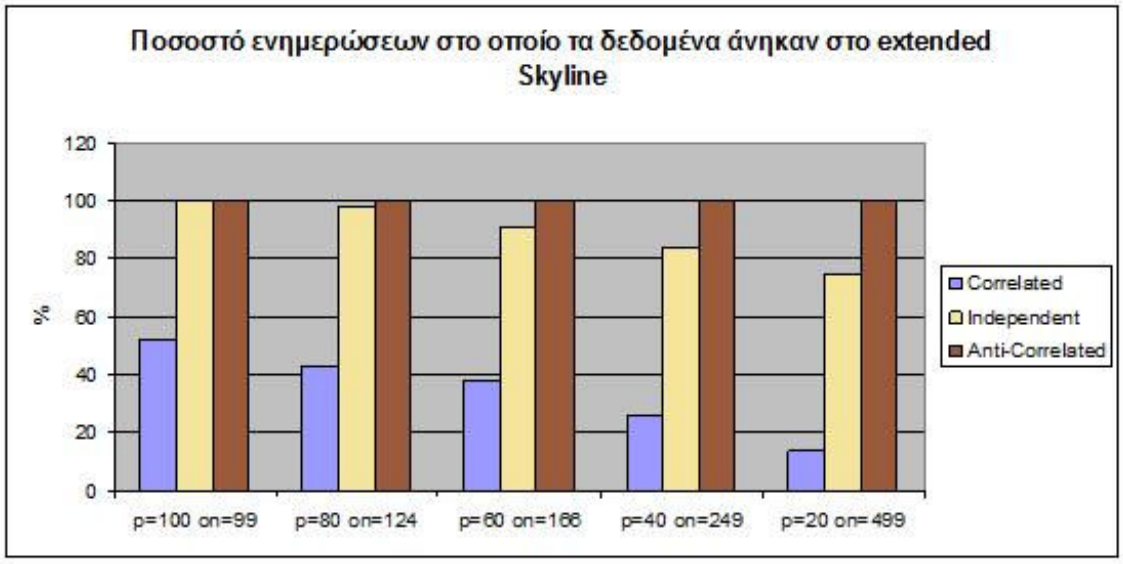
Αν η κατανομή των δεδομένων του συστήματος είναι γνωστή από πριν, μπορούμε να επιλέξουμε ποιος τρόπος κατανομής κάδων θα χρησιμοποιηθεί. Τα πράγματα γίνονται πιο περίπλοκα αν εκτός από το MAX, υπολογίζουμε MIN και DIFF. Αν δεν υπάρχει περιορισμός χώρου στο σύστημα μπορούμε να κρατάμε τα δεδομένα σε παραπάνω από μία μορφές και να χρησιμοποιούμε κάθε φορά αυτή που μας συμφέρει.



Σχήμα 6.12 Αριθμός Σημείων που ανήκουν στο Skyline with buckets και στο extended Skyline with buckets για Διαφορετικές Κατανομές Δεδομένων και 16 buckets ή 16 heuristic buckets

6.4. Ενημερώσεις

Σε προηγούμενη παράγραφο αναφέρθηκε ότι η κατανομή από την οποία έχουν παραχθεί τα δεδομένα επηρεάζει τον αριθμό των σημείων που ανήκουν στο Skyline ενός συνόλου σημείων. Λόγω αυτής της ιδιότητας, η κατανομή αυτή επηρεάζει και το κόστος των ενημερώσεων. Αν οι ενημερώσεις που γίνονται σε έναν κόμβο δεν ανήκουν στο extended Skyline (είτε πρόκειται για εισαγωγή, είτε για διαγραφή δεδομένων) δεν χρειάζεται να γίνει καμία ενέργεια. Το πόσες φορές, λοιπόν, οι ενημερώσεις ανήκουν στο extended Skyline έχει μεγάλη σημασία.



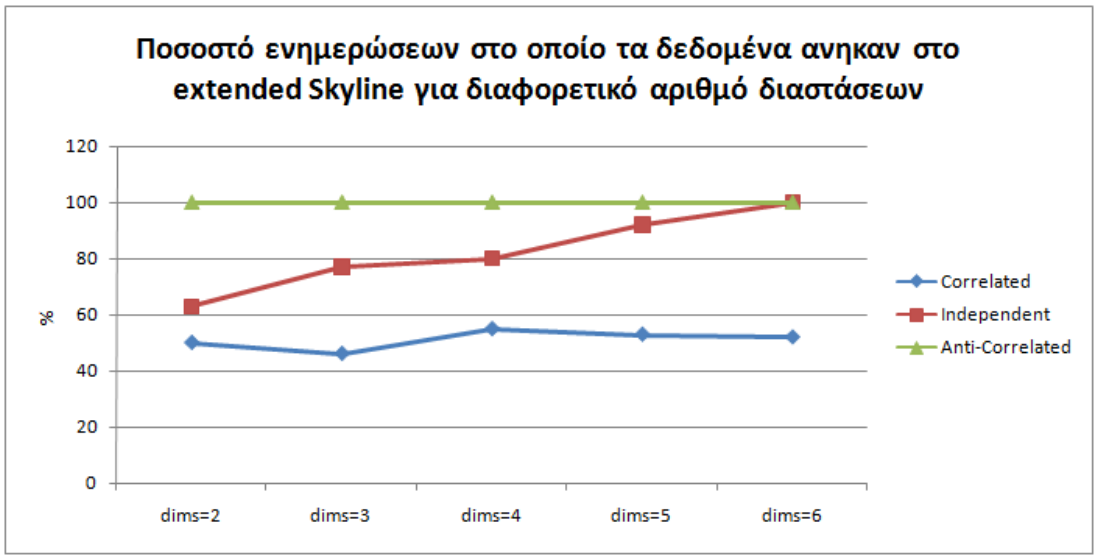
Σχήμα 6.13 Ποσοστό Ενημερώσεων στο οποίο τα Δεδομένα ανήκαν στο extended Skyline

Στο Σχήμα 6.13 βλέπουμε ποιο ποσοστό των ενημερώσεων που γίνονται στους ON ενός SN ανήκει στο extended Skyline. Έχουμε 10.000 σημεία σε έναν SN (περικλείονται δηλαδή και τα σημεία που ανήκουν στους ON του). Υποθέτουμε ότι γίνεται ενημέρωση του extended Skyline του SN κάθε φορά που γίνεται ενημέρωση στο 10% των δεδομένων ενός κόμβου. Μετράμε το ποσοστό των ενημερώσεων για το οποίο τα σημεία ανήκαν στο Skyline. Η παραπάνω διαδικασία γίνεται ξεχωριστά για κάθε κατανομή, έτσι ώστε να μπορεί να γίνει σύγκριση μεταξύ τους. Θεωρούμε ότι γίνονται μόνο εισαγωγές και διαγραφές που γίνονται με την ίδια πιθανότητα. Τα δεδομένα έχουν 6 διαστάσεις.

Όπως είναι αναμενόμενο, στην Anti-Correlated κατανομή, σε κάθε ενημέρωση υπάρχει κάποιο σημείο που ανήκει στο Skyline. Η Correlated κατανομή δίνει καλά αποτελέσματα σε όλες τις περιπτώσεις, ενώ η Independent κατανομή δίνει μέτρια αποτελέσματα. Όσο λιγότερα είναι τα σημεία που αναθέτουμε σε κάθε κόμβο τόσο καλύτερα είναι τα αποτελέσματα. Αυτό συμβαίνει επειδή ενημέρωση γίνεται στο 10% των δεδομένων κάθε κόμβου (εξαρτάται δηλ. η ενημέρωση από τον αριθμό των δεδομένων).

Σημειώνουμε ότι στο παραπάνω πείραμα υποθέσαμε ότι ο αριθμός των εισαγωγών-διαγραφών είναι περίπου 50-50. Αν υποθέσουμε ότι έχουμε μόνο εισαγωγές τα αποτελέσματα

θα είναι χειρότερα (αυξάνεται συνεχώς ο αριθμός των σημείων που ανήκουν στο extended Skyline και ο αριθμός των σημείων, αφού η ενημέρωση γίνεται στο 10% των σημείων που κάθε φορά υπάρχουν). Αντίστοιχα αν αυξήσουμε τον αριθμό των διαγραφών τα αποτελέσματα θα είναι αρκετά καλύτερα.



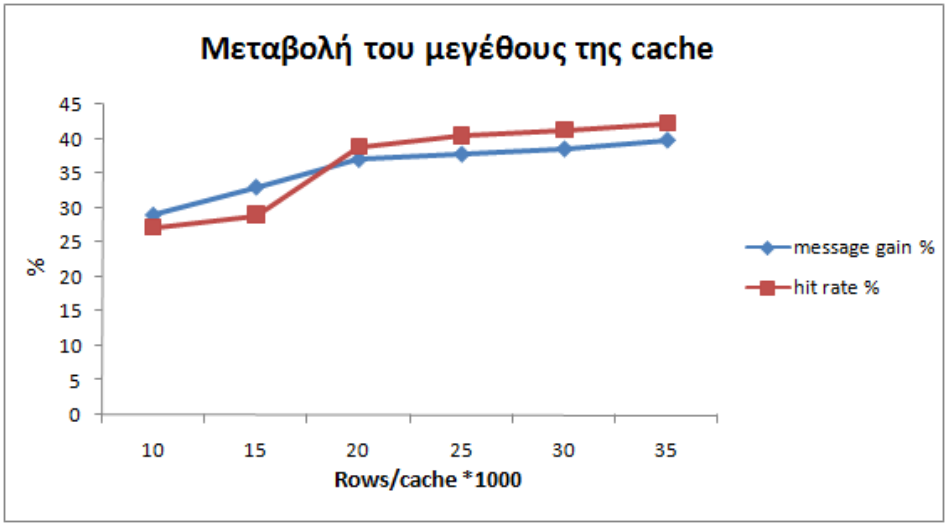
Σχήμα 6.14 Ποσοστό Ενημερώσεων στο οποίο τα Δεδομένα ανήκαν στο extended Skyline για Διαφορετικό Αριθμό Διαστάσεων.

Τα αποτελέσματα ενός παρόμοιου πειράματος εμφανίζονται στο Σχήμα 6.14. Μετράμε το ποσοστό των ενημερώσεων που άνηκαν στο extended Skyline ενός SN. Αυτή τη φορά όμως μεταβάλλονται οι διαστάσεις των δεδομένων. Ξεκινάμε με δυσδιάστατα δεδομένα και καταλήγουμε σε δεδομένα των 6 διαστάσεων. Έχουμε 100 σημεία ανά ON και 99 ON. Όπως παρατηρούμε η μεταβολή στα αποτελέσματα της Correlated κατανομής, είναι πολύ μικρή και δεν επηρεάζεται ιδιαίτερα από τον αριθμό των διαστάσεων. Η Anti-Correlated κατανομή, όπως και πριν είναι μονίμως στο 100%, δηλαδή σε κάθε ενημέρωση πρέπει να γίνει κάποια ενέργεια. Αντίθετα με τις παραπάνω κατανομές, η Independent βλέπουμε ότι επηρεάζεται από τον αριθμό των διαστάσεων των δεδομένων. Όσο μικρότερες είναι οι διαστάσεις τόσο λιγότερα είναι τα δεδομένα που ανήκουν στο extended Skyline, ενώ όσο αυξάνονται, αυξάνονται και τα δεδομένα.

6.5. Caching Skyline Ερωτήματα

Στην παράγραφο αυτή εξετάζουμε το κατά πόσον επηρεάζουν κάποιοι παράγοντες την απόδοση της cache. Οι παράγοντες αυτοί έχουν εξεταστεί πιο αναλυτικά στην Ενότητα 4.2. Στις παρακάτω γραφικές παραστάσεις απεικονίζεται το «message gain» και το «hit rate». Το «message gain» είναι ο αριθμός των μηνυμάτων που κερδίζω αν χρησιμοποιήσω cache. Αν δεν υπάρχει cache, τότε κατά τη δρομολόγηση, ένας SN πρέπει να στείλει μήνυμα (Skyline query) σε όλους τους γείτονες του. Υπό την παρουσία όμως της cache, ο SN στέλνει μήνυμα μόνο σε αυτούς τους γείτονες, οι οποίοι δεν περιέχονται στα αποτελέσματα της cache του. Το «message gain» είναι το ποσοστό των λιγότερων μηνυμάτων που στέλνονται στο σύστημα σε σχέση με αυτά που θα στελνόντουσαν αν δεν υπήρχε η cache. Το «hit rate» είναι το ποσοστό των φορών που ο αρχικός κόμβος που θέτει το ερώτημα και βρίσκει το αποτέλεσμα στην cache του (είτε αυτό είναι μερικό είτε είναι ολικό).

Στο σύστημα των πειραμάτων συμμετέχουν 1000 κόμβοι, οι οποίοι έχουν από 10 σημεία. 900 από αυτούς είναι ON, ενώ 100 είναι SN. Τα δεδομένα των κόμβων ακολουθούν Independent κατανομή και είναι 6 διαστάσεων. Η Zipf κατανομή των ερωτήσεων έχει συντελεστή α=1.2 και η cache έχει χωρητικότητα 20000 πλειάδων (δηλ. 6*64*20000=7680000bit= 7.68MB). Ο χρόνος λήξης των δεδομένων της cache είναι ο χρόνος δημιουργίας συν 20, όπου ο χρόνος μετράται σύμφωνα με τις ενέργειες-βήματα που έχουν γίνει στο σύστημα. Δεδομένου ότι στο σύστημα γίνονται ενημερώσεις και ερωτήσεις με ποσοστό 30% και 70% αντίστοιχα, στο διάστημα αυτό θα έχουν γίνει περίπου 6 ενημερώσεις σε όλο το σύστημα. Το πείραμα έτρεξε για χρόνο 1000, ενώ η cache είναι cold, δηλαδή το πείραμα ξεκινά με την cache άδεια.

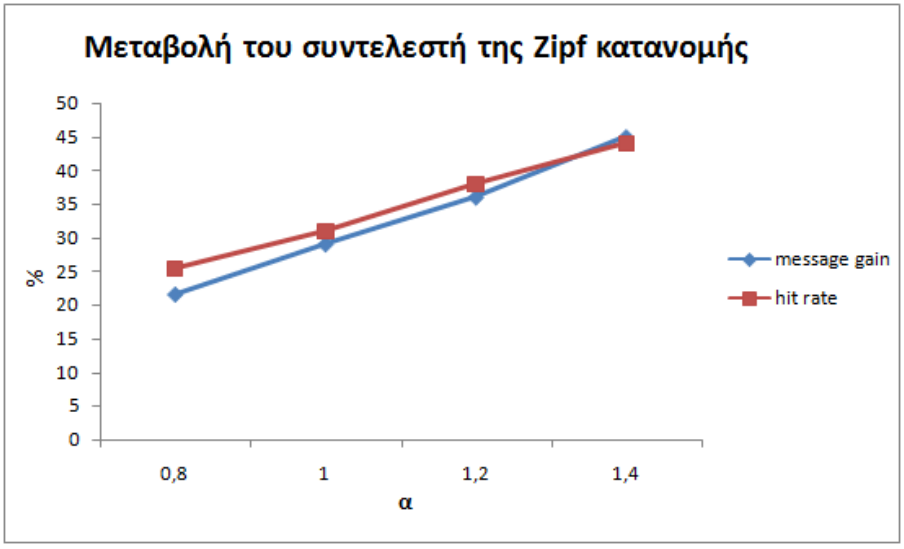


Σχήμα 6.15 Μεταβολή του Μεγέθους της Cache

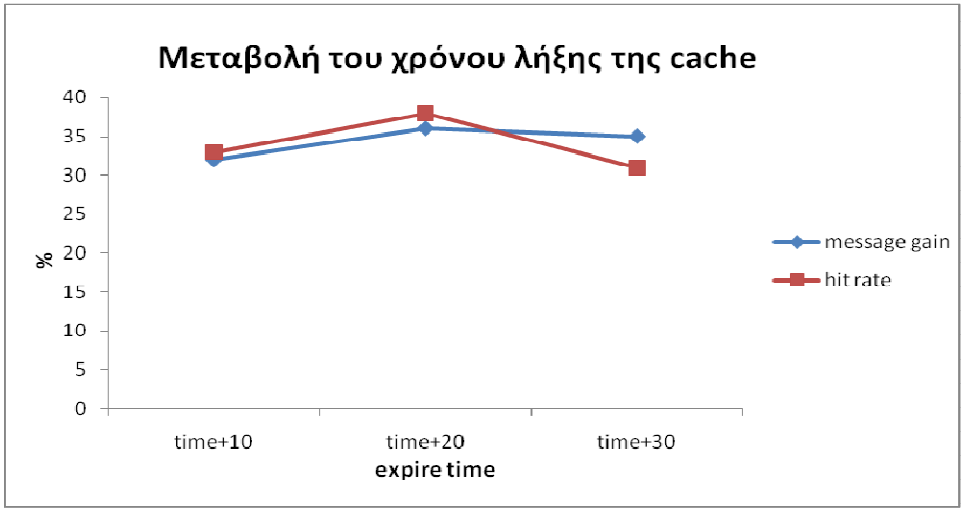
Στο Σχήμα 6.15 βλέπουμε πώς το «message gain» και το «hit rate» μεταβάλλονται ανάλογα με το μέγεθος της cache. Παρατηρούμε ότι ακόμη και μικρό μέγεθος cache έχουμε αρκετό κέρδος 27% και 29% αντίστοιχα ενώ όσο μεγαλώνει το μέγεθος της cache αυξάνεται και το κέρδος που έχουμε.

Στο Σχήμα 6.16 μεταβάλλεται ο συντελεστής της Zipf κατανομής των ερωτήσεων. Ο συντελεστής αυτός καθορίζει το πόσο συχνά γίνονται οι ίδιες ερωτήσεις στο σύστημα. Όσο πιο συχνά τίθενται τα ίδια ερωτήματα στο σύστημα, τόσο αυξάνεται η απόδοση της cache.

Στο Σχήμα 6.17 μεταβάλλεται ο χρόνο λήξης των ερωτημάτων της cache. Παρατηρούμε ότι ενώ αρχικά βλέπουμε μία αύξηση του κέρδους αυτό ξαφνικά μειώνεται στην περίπτωση που ο χρόνος λήξης ισούται με τον χρόνο δημιουργίας των ερωτημάτων +30. Αυτό συμβαίνει επειδή το μέγεθος της cache είναι μικρό, και έτσι παρόλο που τα δεδομένα δεν λήγουνε, πρέπει να διαγραφούν λόγω έλλειψης χωρητικότητας.



Σχήμα 6.16 Μεταβολή της Κατανομής των Ερωτήσεων



Σχήμα 6.17 Μεταβολή του Χρόνου Λήξης της Cache

6.6. Continuous Skyline Ερωτήματα

Στην παράγραφο αυτή γίνονται συγκρίσεις για την απόδοση που έχουν οι δύο διαφορετικοί μέθοδοι ενημέρωσης continuous ερωτημάτων που εξετάσαμε στο Κεφάλαιο 5. Αυτό που συγκρίνουμε είναι το υπολογιστικό και επικοινωνιακό τους κόστος. Η απόδοση της timer-based περίπτωσης σε σχέση με την change-based περίπτωση εξαρτάται κατά πολύ από την

χρονική περίοδο ενημέρωσης των ερωτημάτων. Για να γίνει πιο σαφές θα το εξετάσουμε με ένα απλό παράδειγμα. Αρχικά εξετάζουμε το επικοινωνιακό κόστος.

Υποθέτουμε ότι έχουμε 100 SN. Σε αυτούς συνολικά στο 30 % του χρόνου γίνονται ενημερώσεις. Αυτό σημαίνει ότι έχουμε μία ενημέρωση ανά χρόνο περίπου 3,3. Δηλαδή, αν τρέχει το σύστημα για χρόνο 100, τότε θα έχουμε μόνο 30 ενημερώσεις, δηλ. θα γίνει ενημέρωση σε 30 SN από τους 100. Σε χρόνο 1000 θα έχουμε 300 ενημερώσεις, δηλαδή κάθε SN θα έχει ενημερωθεί περίπου 3 φορές.

Στην change-based περίπτωση ενημερώνω τα continuous ερωτήματα κάθε φορά που έχουμε μία ενημέρωση σε SN. Έτσι, σε χρόνο 100 θα έχουμε 30 ενημερώσεις ερωτήσεων (δηλαδή 30 μηνύματα), ενώ σε χρόνο 1000 θα έχουμε 300 μηνύματα. Σημειώνουμε βέβαια ότι σε αυτό το διάστημα οι κόμβοι που δεν έχουν κάποια ενημέρωση δεν χρειάζεται να κάνουν τίποτα.

Για να γίνει σύγκριση με την timer-based περίπτωση πρέπει να ορίσω μία περίοδο ενημέρωσης ερωτημάτων. Για ευκολία του παραδείγματος ορίζουμε την ίδια περίοδο ενημέρωσης για όλα ερωτήματα. Υπάρχουν τρεις περιπτώσεις:

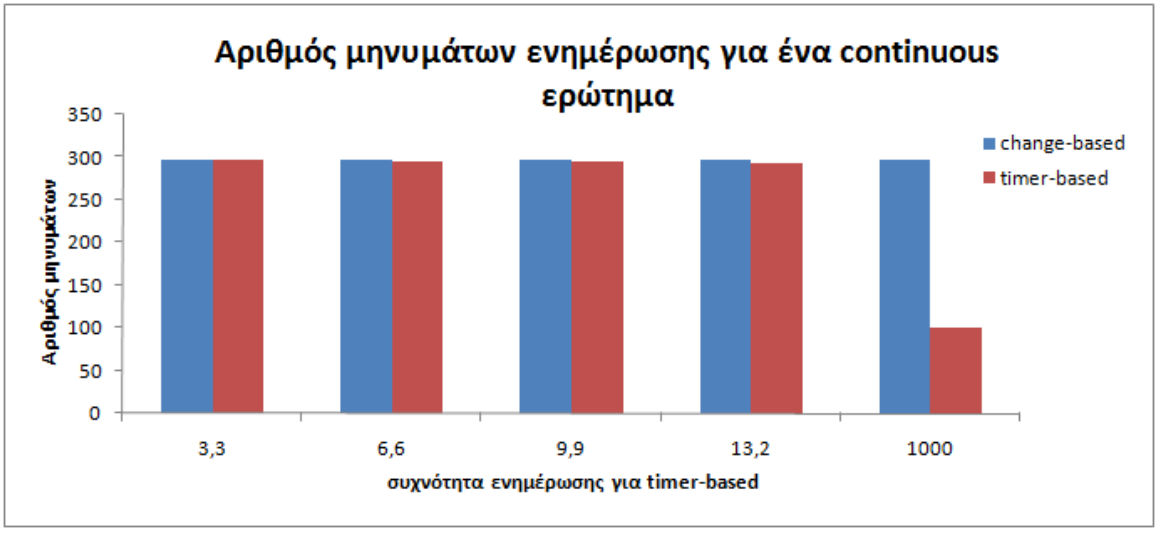
- Αν ορίσουμε περίοδο ενημέρωσης όλων των ερωτημάτων 3,3 τότε σε χρόνο 100 θα σταλούν 30 μηνύματα και σε χρόνο 1000 θα σταλούν 300 μηνύματα, όπως ακριβώς και στην change-based περίπτωση. Επιπλέον σε αυτή την περίοδο όλοι οι SN θα πρέπει να κάνουν έλεγχο για ενημερώσεις. Όσον αφορά το κόστος υπολογισμού και το μέγεθος των μηνυμάτων, αυτό θα είναι σαφώς μεγαλύτερο από αυτό της change-based περίπτωσης. Αυτό όμως θα το εξετάσουμε αργότερα.

- Αν η περίοδος ενημέρωσης των ερωτημάτων είναι μικρότερη από 3,3, τότε τα πράγματα είναι χειρότερα από την άποψη ότι γίνονται συνεχώς περιττοί έλεγχοι. Τα μηνύματα που θα σταλούν όμως είναι τα ίδια όπως και πριν.

- Αν η περίοδος ενημέρωσης είναι μεγαλύτερη από 3,3 τότε και πάλι τα μηνύματα είναι ίδια με αυτά που στέλνονται στην change-based περίπτωση, ή ίσως ελάχιστα λιγότερα. Για να έχουμε λιγότερα μηνύματα θα πρέπει, ο ίδιος SN να προλάβει να

ενημερωθεί πάνω από μία φορά στο χρονικό αυτό διάστημα. Π.χ. ακόμα και αν διπλασιάσουμε την περίοδο από 3,3 σε 6,6 τότε λόγω του πλήθους των SN είναι σχεδόν απίθανο να προλάβει ένας SN να ενημερωθεί πάνω από μία φορά σε τόσο μικρό διάστημα. Αν η περίοδος ενημέρωσης είναι 1000, τότε θα σταλούν 100 μηνύματα επειδή κάθε SN θα στείλει μία μόνο φορά ενημερώσεις αντί να στείλει 3. Δηλαδή αντί να σταλούν 300 μηνύματα που στέλνονται στην περίπτωση 1 στέλνεται το 1/3. Αν όμως η περίοδος είναι τόσο μεγάλη, τότε δε μιλάμε για continuous ερωτήματα. Επίσης σημειώνουμε ότι ακόμα και αν έχουμε περίοδο 2000 τότε πάλι θα στέλναμε 100 μηνύματα. Θα δούμε παρακάτω ότι αυτό που κερδίζουμε στην timer-based περίπτωση με περίοδο ενημέρωσης πάνω από 3,3 (σε λογικά φυσικά πλαίσια) είναι ότι κάνουμε λιγότερους υπολογισμούς.

Το Σχήμα 6.18 απεικονίζει τον αριθμό των μηνυμάτων που στέλνονται για την ενημέρωση ενός continuous ερωτήματος. Στο σύστημα υπάρχουν 100 SN. Στο 30% του χρόνου γίνονται ενημερώσεις, ενώ στο70% ερωτήσεις. Δηλαδή η συχνότητα ενημέρωσης είναι περίπου 3,3. Αφήνουμε το σύστημα να τρέξει για χρόνο 1000 και συγκρίνουμε τον αριθμό των μηνυμάτων που στέλνονται για την ενημέρωση ενός ερωτήματος στην περίπτωση που έχουμε change-based ερωτήματα και στην περίπτωση που έχουμε timer-based ερωτήματα. Μεταβάλουμε την χρονική περίοδο ενημέρωσης των timer-based ερωτημάτων και παρατηρούμε ότι ο αριθμός των μηνυμάτων δεν μειώνεται σημαντικά. Πρέπει να αυξήσουμε κατά πάρα πολύ την περίοδο ενημέρωσης έτσι ώστε να μειώσουμε τον αριθμό των μηνυμάτων.



Σχήμα 6.18 Αριθμός Μηνυμάτων Ενημέρωσης για ένα Continuous Ερώτημα

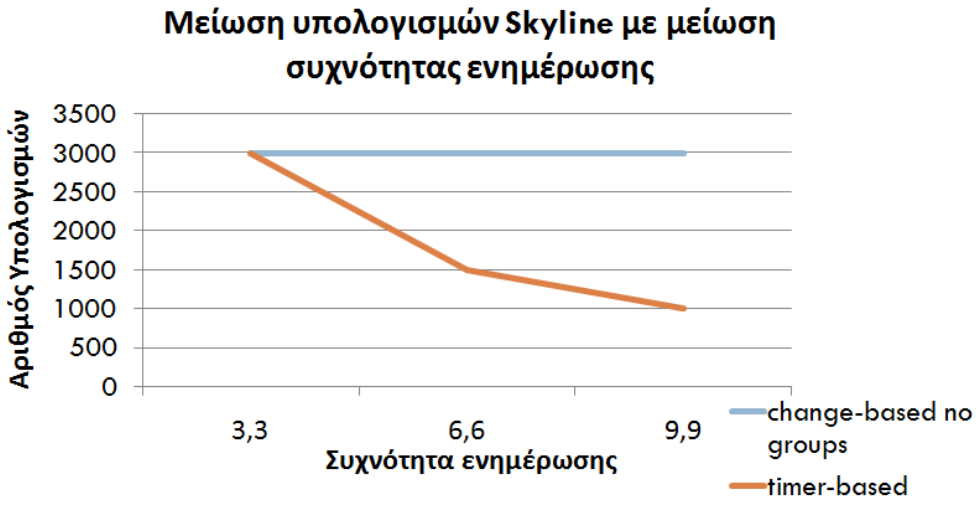
Στο Σχήμα 6.19 συγκρίνουμε τους υπολογισμούς που κάνουμε ανά σημείο για την change-based περίπτωση χωρίς διαμοιρασμό ερωτήσεων σε σύνολα, και για την timer-based περίπτωση. Μετράμε δηλαδή πόσα σημεία πρέπει να ελεγχθούν για να υπολογιστεί το extended Skyline. Υποθέτουμε ότι το σύστημα τρέχει για χρόνο 100, δηλαδή γίνονται 30 ενημερώσεις. Επίσης υποθέτουμε ότι υπάρχουν 100 σημεία για τα οποία γίνεται ο υπολογισμός και ότι έχουμε ένα μόνο continuous ερώτημα. Τα σημεία παράγονται με Independent κατανομή. Η συχνότητα ενημέρωσης για την change-based περίπτωση μένει σταθερή στο 3,3 ενώ για την περίπτωση 2 μεταβάλλεται. Παρατηρούμε ότι η timer-based περίπτωση χρειάζεται να κάνει πολύ λιγότερους υπολογισμούς συνολικά καθώς μειώνουμε το χρόνο ενημέρωσης των ερωτήσεων.

Στο Σχήμα 6.20 μπορούμε να δούμε πώς αυξάνεται το κέρδος σε υπολογισμούς του extended Skyline ανάλογα με το πόσο επικαλύπτονται οι διαστάσεις των continuous ερωτημάτων που μας ενδιαφέρουν. Ουσιαστικά συγκρίνουμε το πόσα σημεία (πλειάδες) ελέγχονται στην απλή περίπτωση, δηλαδή στην περίπτωση όπου κάθε ένα continuous ερώτημα νοείται ως ξεχωριστό και στην περίπτωση που χωρίζουμε τα ερωτήματα αυτά σε σύνολα ή ομάδες έτσι ώστε να ικανοποιούν τη Συνθήκη 4.1, όπως αυτή περιγράφηκε στο Κεφάλαιο 5. Υποθέτουμε ότι στο σύστημα μας υπάρχουν 100 SN. Η συχνότητα ενημέρωσης είναι 3,3 (σε αυτή τη

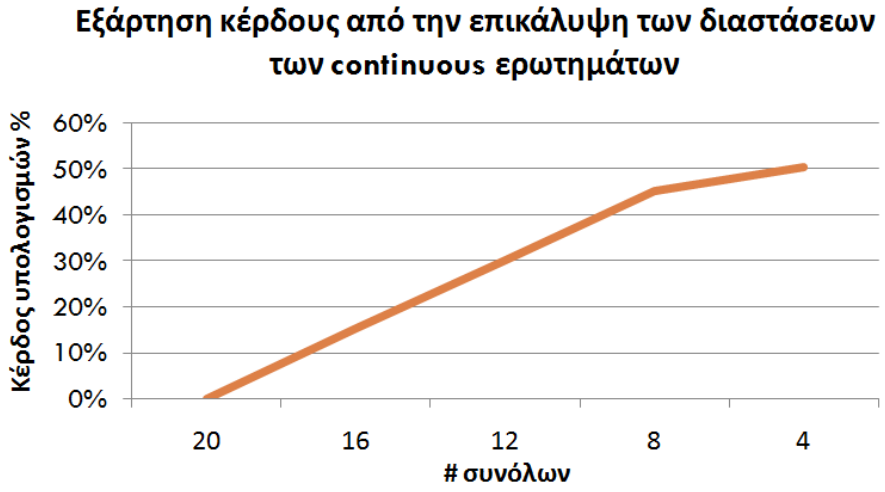
συχνότητα ενημερώνεται ένας SN του συστήματος). Επίσης υποθέτουμε ότι υπάρχει ένα σύνολο 20 continuous ερωτήσεων, οι οποίες διαμοιράζονται στους SN. Στην απλή περίπτωση θεωρούμε ότι 20 τυχαία επιλεγμένοι SN αναλαμβάνουν από ένα ερώτημα. Στην περίπτωση που έχουμε διαμοιρασμό σε σύνολα, μεταβάλουμε τον αριθμό των συνόλων και συνεπώς την επικάλυψη των διαστάσεων των ερωτημάτων. Σε κάθε ένα σύνολο απαιτείται να ισχύει η Συνθήκη 4.1, ενώ θεωρούμε ότι οι διαστάσεις των δεδομένων είναι 6. Αρχικά δεν έχουμε καθόλου επικάλυψη, δηλαδή έχουμε 20 σύνολα και έτσι οι δύο περιπτώσεις συμπίπτουν (0% κέρδος). Σιγά σιγά αυξάνουμε την επικάλυψη μέχρι που φτάνουμε στην πλήρη επικάλυψη, δηλαδή έχουμε 4 σύνολα κάθε ένα από τα οποία περιλαμβάνει 5 ερωτήσεις. Σε αυτό το σημείο φτάνουμε σε πάνω από 50% κέρδος, όσον αφορά τα σημεία που συνολικά ελέγχουμε για το αν ανήκουν στο extended Skyline. Παρατηρούμε ότι δεν έχουμε κάτι να χάσουμε από την εφαρμογή της μεθόδου των συνόλων, από τη στιγμή που η απόδοση της είναι είτε ίδια είτε καλύτερη από αυτή της απλής περίπτωσης. Επίσης πρέπει να σημειώσουμε ότι χρησιμοποιώντας τον απλό τρόπο ανάθεσης ερωτημάτων σε σύνολα είναι σχεδόν σίγουρο ότι θα έχουμε κέρδος. Στις 6 διαστάσεις που χρησιμοποιούμε στα πειράματα μας, όλες οι πιθανές ερωτήσεις είναι 63, ενώ τα σύνολα που ικανοποιούν την Συνθήκη 4.1 είναι 20 στη χειρότερη περίπτωση. Σημειώνουμε, ότι αν έχουμε και τις 63 ερωτήσεις στο σύστημα ως continuous και χρησιμοποιήσουμε έναν διαμοιρασμό σε σύνολα τότε έχουμε συνολικό κέρδος περίπου 48,21%.

Στο Σχήμα 6.21 συγκρίνουμε την change-based περίπτωση με την timer-based περίπτωση. Αυτή τη φορά υποθέτουμε ότι έχουμε ένα σύνολο τριών continuous ερωτημάτων. Στην change-based περίπτωση, το σύνολο αυτό ικανοποιεί την Συνθήκη 4.1. Στην timer-based περίπτωση, το σύνολο απλώς έχει την ίδια συχνότητα ενημέρωσης για όλες τις ερωτήσεις που ανήκουν σε αυτό. Τρέχουμε το σύστημα αυτό για χρόνο 500 και μετράμε τον συνολικό αριθμό σημείων που ελέγχθηκαν για κυριαρχία στο Skyline. Παρατηρούμε ότι όταν η συχνότητα ενημέρωσης της timer–based περίπτωσης είναι ίδια με αυτή της change-based περίπτωσης, τότε στην change-based περίπτωση χρειάζεται να κάνουμε πολύ λιγότερους υπολογισμούς. Μόλις όμως μειώσουμε στο μισό τη συχνότητα ενημέρωσης της timer–based περίπτωσης, αμέσως οι υπολογισμοί που χρειάζονται γίνονται λιγότεροι από αυτούς της change-based περίπτωσης. Όσο μειώνουμε τη συχνότητα ενημέρωσης τόσο λιγότεροι

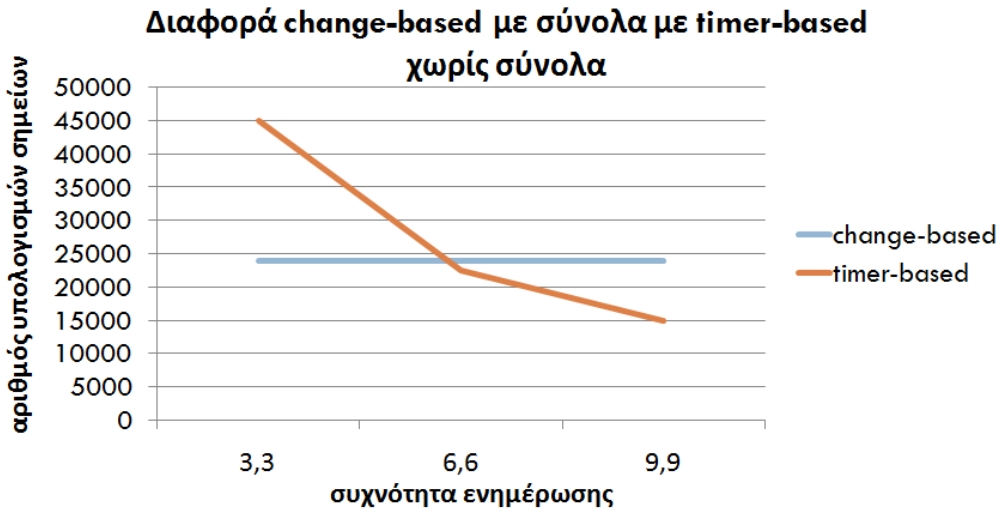
υπολογισμοί χρειάζονται. Σημειώνουμε βέβαια ότι μειώνεται η ακρίβεια των αποτελεσμάτων που εμφανίζονται στο χρήστη.



Σχήμα 6.19 Αριθμός Skyline Υπολογισμών



Σχήμα 6.20 Κέρδος Μηνυμάτων ανάλογα με την Επικάλυψη των Διαστάσεων continuous Ερωτημάτων



Σχήμα 6.21 Διαφορά σε Υπολογισμούς Σημείων μεταξύ change-based και timer-based Ερωτημάτων

ΚΕΦΑΛΑΙΟ 7. ΣΧΕΤΙΚΕΣ ΕΡΓΑΣΙΕΣ

Όσον αφορά την βιβλιογραφία του Skyline, οι μέθοδοι που προτείνονται μπορούν να χωριστούν σε διάφορες κατηγορίες. Αρχικά, υπάρχουν αυτοί που πρότειναν το Skyline ως μαθηματικό-γεωμετρικό πρόβλημα („the maximum vector problem“), Έπειτα αυτοί που ορίζουν το πρόβλημα σε βάσεις δεδομένων και προσπαθούν να βρουν τη λύση του (με τη διαφορά ότι πλέον η διαθέσιμη μνήμη είναι περιορισμένη) και τέλος αυτοί που προσπαθούν το λύσουν το πρόβλημα σε συστήματα ομότιμων. Σε αυτό το κεφάλαιο παρουσιάζεται η σχετική εργασία και γίνεται μία ομαδοποίηση των εργασιών ανά κατηγορία. Τέλος παρουσιάζονται δύο συγκεντρωτικοί και συγκριτικοί πίνακες των εργασιών που εξετάστηκαν. Στο Παράρτημα μπορεί κανείς να δει περιλήψεις των εργασιών αυτών.

7.1. Το Skyline στο Περιβάλλον των Βάσεων Δεδομένων

Το πρόβλημα προτάθηκε στις βάσεις δεδομένων από τους [BoKS01], οι οποίοι προτείνουν και αρκετούς αλγόριθμους για τη λύση του Skyline προβλήματος. Αρχικά προτείνονται διάφορες παραλλαγές αλγόριθμων που χρησιμοποιούν ένα „παράθυρο“ (window), τέτοιου μεγέθους ώστε τα δεδομένα να χωράνε στη μνήμη για υπολογισμό. Έπειτα προτείνουν αναδρομική λύση του προβλήματος („Divide and Conquer“) χωρίζοντας το σύνολο των

σημείων σε υποσύνολα. Τέλος προτείνουν μεθόδους που χρησιμοποιούν δενδρικές δομές για τον υπολογισμό του Skyline και συγκεκριμένα δίνουν έναν αλγόριθμο που χρησιμοποιεί B-δένδρα και έναν ακόμη που χρησιμοποιεί R-δένδρα. Η ιδέα πίσω από την χρησιμοποίηση τέτοιου είδους ευρετηρίων είναι να χρησιμοποιηθεί η διάταξη των δεδομένων για να μπορέσουν να απαλειφθούν πολύ γρήγορα εγγραφές που δεν ανήκουν στο Skyline.

7.2. Προοδευτικοί Αλγόριθμοι Εύρεσης του Skyline

Τα ευρετήρια που αναφέρθηκαν παραπάνω χρησιμοποιήθηκαν από αρκετούς για να λύσουν διάφορα προβλήματα που αφορούν το Skyline, όπως π.χ. το να είναι progressive, δηλαδή τα αποτελέσματα να μην παράγονται όλα μαζί, αλλά να μπορούμε να δίνουμε σταδιακά αποτελέσματα στο χρήστη. Συγκεκριμένα, B-δένδρα χρησιμοποιούνται από τους [TaEO01] στον αλγόριθμο που ονομάζουν Index για να υπολογίσουν το Skyline έτσι ώστε να δίνουν στο χρήστη τα σημεία που ανήκουν στο Skyline αμέσως μόλις τα συναντήσουν. Όπως παρουσιάζεται στα πειράματα τους, ο Index αλγόριθμος φαίνεται να υπερτερεί έναντι των αλγορίθμων με το „παράθυρο“, αλλά και του D&C. Δυστυχώς όμως ο αλγόριθμος τους απαιτεί ένα B-δένδρο για κάθε πιθανό συνδυασμό των διαστάσεων των δεδομένων, κάτι που είναι πολύ ακριβό χωρικά και υπολογιστικά. Οι [KoRR02] προτείνουν έναν καλύτερο progressive Skyline αλγόριθμο, χρησιμοποιώντας τη Nearest neighbor τεχνική, η οποία βασίζεται σε R-δένδρο αυτή τη φορά. Η τεχνική τους παρουσιάζει αρκετά μειονεκτήματα με κυριότερα τις πολλαπλές προσβάσεις στον ίδιο κόμβο και την απαίτηση μεγάλου όγκου χώρου για να τρέξει. Οι [PTFS03] λύνουν αυτά τα προβλήματα προτείνοντας μία παραλλαγή του παραπάνω αλγόριθμου, η οποία επίσης χρησιμοποιεί R-δένδρο και ονομάζεται BBS. Ο BBS ξεπερνά όλα τα μειονεκτήματα του NN, ενώ διατηρεί τα πλεονεκτήματα του.

7.3. Άλλοι Αλγόριθμοι Εύρεσης του Skyline

Μία παραλλαγή του BBS χρησιμοποιείται από τους [ChET05] για να λύσει το πρόβλημα του Skyline για δεδομένα που δεν μπορούν πλήρως να ταξινομηθούν. Οι τελευταίοι προτείνουν ακόμη έναν αλγόριθμο, ο οποίος χωρίζει τα δεδομένα σε υποσύνολα και χρησιμοποιεί

γράφους κυριαρχίας για βρουν ποια δεδομένα μπορεί να ανήκουν στο Skyline. Το κάθε υποσύνολο είναι ευρετηριοποιημένο με τη βοήθεια R-δένδρου.

Εκτός από τις μεθόδους που χρησιμοποιούν δένδρα για τον υπολογισμό του Skyline, υπάρχουν και άλλες δύο διαφορετικές περιπτώσεις. Η πρώτη έχει προταθεί από τους [BaGZ04] και απαιτεί την ύπαρξη n ταξινομημένων λιστών, όπου n είναι ο αριθμός των διαστάσεων. Η δεύτερη προτάθηκε από τους [TaEO01] και χρησιμοποιεί Bitmap αναπαράσταση των δεδομένων καθώς και πράξεις σε bit για να υπολογίσει τα σημεία που ανήκουν στο Skyline.

7.4. Εύρεση του Skyline σε Συστήματα Ομότιμων

Από τη στιγμή που είχαν προταθεί αρκετές λύσεις για κεντρικοποιημένα συστήματα άρχισαν να προτείνονται λύσεις για συστήματα ομότιμων. Στα συστήματα αυτά οι σχεδιαστές αλγόριθμων έχουν να αντιμετωπίσουν περισσότερες προκλήσεις, όπως ότι έχουμε να κάνουμε με ένα κατανεμημένο περιβάλλον, με συχνές αναχωρήσεις και αφίξεις κόμβων, κάθε κόμβος μπορεί να θέσει ένα ερώτημα και οι κόμβοι μπορεί να έχουν διαφορετικό σχήμα στη βάση δεδομένων τους.

Στις λύσεις για συστήματα ομότιμων, οι συγγραφείς των άρθρων πρέπει να κάνουν μία εξ αρχής υπόθεση για το σύστημα στο οποίο θέλουνε να λύσουν το πρόβλημα (δομημένο ή αδόμητο) και την κατανομή των δεδομένων στους κόμβους. Τα δεδομένα είτε είναι τυχαία διαμοιρασμένα στους κόμβους, είτε θεωρούνε ότι τα δεδομένα ανατίθενται σε αυτούς με συγκεκριμένο τρόπο οπότε και το σύστημα είναι δομημένο. Υπάρχει και η περίπτωση όπου τα παραπάνω συνδυάζονται, έχουμε δηλαδή τυχαία κατανομή των δεδομένων στους κόμβους, αλλά υπάρχει ευρετήριο και δείκτες για τα δεδομένα που θα έπρεπε να ανήκουν σε άλλο σημείο του ευρετηρίου.

Στις περισσότερες εργασίες, οι σχεδιαστές δεν ασχολούνται με τον καθαυτό υπολογισμό του Skyline σε κάθε ομότιμο, αλλά προσπαθούν να βρουν αποδοτικούς τρόπους (π.χ. ευρετήρια)

για την δρομολόγηση του ερωτήματος. Ο υπολογισμός του Skyline μπορεί να γίνει σε κάθε ομότιμο με όποιον κεντρικοποιημένο αλγόριθμο αυτός επιθυμεί. Ένα τέτοιο παράδειγμα είναι αυτό της [Hose05]. Εκεί χρησιμοποιούνται φίλτρα δρομολόγησης για να αποφασιστεί το σε ποιους ομότιμους θα σταλεί το Skyline ερώτημα. Το αποτέλεσμα του ερωτήματος δεν είναι απόλυτα σωστό, αλλά πιθανοτικό. Το σύστημα στο οποίο προτείνεται αυτή η λύση είναι ένα πλήρως αδόμητο σύστημα ομότιμων. Στο [LiTL06] προτείνονται δύο αλγόριθμοι. Ο πρώτος υποθέτει ένα δομημένο σύστημα όπου οι ομότιμοι ανατίθενται σε cluster. Κατά τον υπολογισμό του Skyline κάποια cluster μπορούν να διαγραφούν αμέσως, αφού δεν υπάρχει περίπτωση να ανήκουν τα δεδομένα τους στο Skyline. Ο αλγόριθμος αυτός παρουσιάζει αρκετές ομοιότητες με τον κεντρικοποιημένο BBS αλγόριθμο. Ο δεύτερος αλγόριθμος που προτείνουν απευθύνεται σε ένα πλήρως αδόμητο σύστημα. Ασχολείται και αυτός με την προώθηση μόνο του ερωτήματος σε ομότιμους και είναι πιθανοτικός. Οι [WZF+06] προσπαθούν να λύσουν το πρόβλημα σε δομημένο σύστημα ομότιμων και συγκεκριμένα στο CAN. Χρησιμοποιούν τη δομή του συστήματος – κατανομή δεδομένων για να χωρίσουν σε υποπεριοχές τους ομότιμους και να αποκλείσουν εξαρχής κάποιους από αυτούς. Οι [VDKV07] προσπαθούν να λύσουν ένα ακόμη πρόβλημα που προκύπτει στα συστήματα ομότιμων όσον αφορά τον υπολογισμό του Skyline: το κατά πόσον μπορεί να υπολογιστεί το Skyline οποιουδήποτε υποσυνόλου των διαστάσεων. Αυτό το πετυχαίνουν σε ένα αδόμητο σύστημα στο οποίο υπάρχουν Super Peers, οι οποίοι διατηρούν όχι μόνο το Skyline των δεδομένων, αλλά παραπάνω πληροφορία, ή όπως το ονομάζουν το extended Skyline. Τέλος, οι [WOTX07] επιλύουν το πρόβλημα εύρεσης του Skyline ενός συνόλου σημείων στο σύστημα BATON. Στη δομή αυτού του συστήματος κάθε ομότιμος είναι υπεύθυνος για μία συγκεκριμένη περιοχή των δεδομένων. Λόγω αυτής της ιδιότητας είναι εύκολο να βρεθεί ποιοι κόμβοι έχουν δεδομένα τα οποία ανήκουν στο Skyline. Βασίζονται στις ιδιότητες του BATON για τις «καλές» ιδιότητες του αλγόριθμου που προτείνουν, όπως η ελαχιστοποίηση των μηνυμάτων, η εμπλοκή όσο το δυνατόν λιγότερων κόμβων και το καλό load balance του συστήματος.

7.5. Εφαρμογή Άλλων Μεθόδων σε Συστήματα Ομότιμων Κόμβων

Οι αλγόριθμοι BNL και D&C που προτάθηκαν στο [BoKS01], έχουν σχεδιαστεί με σκοπό να μειώσουν την χρησιμοποίηση-κατανάλωση της κύριας μνήμης σε κεντρικοποιημένα συστήματα βάσεων δεδομένων. Η εφαρμογή τους προϋποθέτει την γνώση όλων των στοιχείων της βάσης και έτσι δεν μπορούν να εφαρμοστούν σε συστήματα ομότιμων, όπου είναι γνωστή μόνο τοπική πληροφορία. Το να συγκεντρώσουμε όλα τα στοιχεία του δικτύου σε έναν ομότιμο για να μπορέσουμε να εφαρμόσουμε έναν από αυτούς τους αλγόριθμους δεν αποδεικνύεται καλή λύση, αφού προϋποθέτει υψηλό bandwidth και το πιθανότερο είναι να μην φτάνει η τοπική κύρια μνήμη του ομότιμου για την εκτέλεση του ερωτήματος. Η εφαρμογή του «Διαίρει και Βασίλευε» προτύπου αλγόριθμου είναι εφικτή σε συστήματα ομότιμων, όπου κάθε ομότιμος υπολογίζει το τοπικό του Skyline και τα αποτελέσματα συνενώνονται καθώς επιστρέφουν στον ομότιμο που έθεσε το ερώτημα. Το αρνητικό αυτής της προσέγγισης είναι ότι πρέπει να πλημμυρίσουμε το δίκτυο.

Οι αλγόριθμοι που βασίζονται σε κάποιου είδους ταξινόμησης των δεδομένων, όπως είναι οι B-tree και R-tree που προτείνουν οι [BoKS01] και ο Bitmap και ο Index που προτείνουν οι [TaEO01] απαιτούν συγκεκριμένα ευρετήρια για την εξαγωγή Skyline ερωτημάτων. Τα ευρετήρια αυτά είναι «από πριν» υπολογισμένα και έχουν σχεδιαστεί για δεδομένα που είναι διαθέσιμα τοπικά (περιλαμβάνουν πληροφορία για όλες τις πιθανές διαστάσεις του ερωτήματος). Λόγω της δυναμικότητας των συστημάτων ομότιμων η διατήρηση τέτοιου είδους ευρετηρίων θα εμπεριείχε πολύ μεγάλο κόστος. (ουσιαστικά αυτοί οι αλγόριθμοι μπορούν να εφαρμοστούν μόνο τοπικά σε κάθε ομότιμο και όχι συνολικά σε όλο το δίκτυο).
Οι αλγόριθμοι που βασίζονται στη nearest neighbor αναζήτηση, όπως ο NN [KoRR02] και ο BBS [PTFS03] είναι γενικά μία καλή ιδέα λόγω του ότι μπορούν να παράγουν αμέσως κάποια πρώτα αποτελέσματα. Αυτό το χαρακτηριστικό τους είναι πολύ σημαντικό, ειδικά στην περίπτωση των συστημάτων ομότιμων, όπου ο χρήστης θέλει να δει αμέσως κάποια αποτελέσματα χωρίς να χρειαστεί να περιμένει ώρα. Γενικά το να χωρίσουμε το χώρο και να ψάχνουμε αναδρομικά σε κάθε κομμάτι του για σημεία του Skyline είναι κάτι που ίσως θα δούλευε και σε συστήματα ομότιμων. Αυτή όμως η λύση οδηγεί σε πολλαπλά round trips κάτι που δεν είναι αποδεκτό λόγω της καθυστέρησης και του φόρτου στο δίκτυο.

Ο πρώτος αλγόριθμος που προτάθηκε για εφαρμογή σε κατανεμημένα συστήματα και συγκεκριμένα για Web Information Systems, είναι αυτός που προτείνεται στο [BaGZ04]. Η διαφορά αυτών των συστημάτων με τα συστήματα ομότιμων είναι ότι τα πρώτα υποθέτουν ένα μικρό σύνολο ομότιμων στους οποίους έχει πρόσβαση κάποια κεντρική μονάδα, ενώ στα δεύτερα έχουμε μεγάλο πλήθος ομότιμων οι οποίοι είναι συνδεδεμένοι μεταξύ τους πάνω σε κάποια τυχαία (ή όχι) δομή στην οποία δύο ομότιμοι μπορεί να βρίσκονται σε πολύ μακρινή απόσταση. Έτσι στην πρώτη περίπτωση δεν κοστίζει ιδιαίτερα να κρατάμε ταξινομημένες λίστες των δεδομένων μας, ενώ στην δεύτερη δεν είναι δυνατόν, αφού απαιτείται γνώση ολικής πληροφορίας. Ουσιαστικά αυτό που υποθέτει είναι κάθετη κατανομή των δεδομένων (όσον αφορά δηλαδή τις διαστάσεις) και όχι οριζόντια, όπως συνήθως γίνεται στα συστήματα ομότιμων.

Μία τροποποίηση του αλγόριθμου των [HKSZ05] για Skyline ερωτήματα σε συστήματα ομότιμων είναι αποδεκτή, αφού η χρησιμοποίηση ευρετηρίων είναι κάτι που φαίνεται να έχει εδραιωθεί όσον αφορά την αναζήτηση σε συστήματα ομότιμων. Αυτή ακριβώς η τροποποίηση παρουσιάζεται στο [Hose05] όπου τα ίδια ευρετήρια χρησιμοποιούνται για να απαντήσουν Skyline ερωτήματα. Οι διαφορά των δύο αυτών τεχνικών σε σχέση με τις υπόλοιπες είναι ότι δίνουν πιθανοτικές εγγυήσεις για τα αποτελέσματα (το αποτέλεσμα δηλ. είναι σωστό με συγκεκριμένο ποσοστό ορθότητας). Επίσης η τεχνική του [Hose05] μπορεί να εφαρμοστεί σε οποιοδήποτε σύστημα ομότιμων (δομημένο ή αδόμητο). Άποψη μας είναι ότι μπορούμε να πετύχουμε καλύτερα αποτελέσματα όταν σχεδιάζουμε αλγόριθμο για ένα σύστημα με συγκεκριμένη γνωστή δομή (μπορούμε να αποφύγουμε δηλ. την πιθανοτική και να έχουμε μία 100% ορθή απάντηση).

Οι αλγόριθμοι που παρουσιάζονται στο [ChET05] αφορούν δεδομένα που δεν μπορούν πλήρως να ταξινομηθούν. Οι λύσεις που προτείνουν στοχεύουν στη λύση του προβλήματος σε κεντρικοποιημένα συστήματα. Για την αντιστοίχιση των δεδομένων σε άλλα που μπορούν πλήρως να ταξινομηθούν απαιτείται γνώση ολικής πληροφορίας. Ακόμη και αν κάνουμε την αντιστοίχιση τοπικά σε κάθε ομότιμο, ο αλγόριθμος δεν θα μπορεί να τρέξει έπειτα σε συνολικό επίπεδο αφού η αντιστοίχιση δεν θα είναι πλέον έγκυρη. Αν υποθέσουμε ότι

υπάρχει τρόπος να γίνει κάποια άλλη αντιστοίχιση που να λύνει αυτό το πρόβλημα, ο αλγόριθμος δεν μπορεί να λειτουργήσει επειδή οι γράφοι –υποσύνολα στα οποία χωρίζει τα δεδομένα του συνεχίζουν να υποθέτουν ότι η πληροφορία βρίσκεται κάπου συγκεντρωμένη και είναι γνωστή σε όλους.

Στο [WZF+06] περιγράφεται ένας προοδευτικός αλγόριθμος, ο οποίος επιλύει επιτυχώς το πρόβλημα των Skyline ερωτήσεων σε συστήματα ομότιμων όπως το CAN. Ο αλγόριθμος υποθέτει ότι τα δεδομένα διαμοιράζονται στους κόμβους σύμφωνα με το περιεχόμενο τους. Σε αυτήν την υπόθεση βασίζεται για να επιτύχει την προοδευτικότητα του και γενικά όλες τις καλές του ιδιότητες. Στην πραγματικότητα όμως, λίγα είναι τα συστήματα στα οποία τα δεδομένα διαμοιράζονται με αυτόν τον τρόπο. Συνήθως κάθε κόμβος ενός συστήματος ομότιμων διατηρεί τυχαία κάποια δεδομένα. Ένα ακόμα ζήτημα είναι το κατά πόσον τα δεδομένα χωρίζονται σύμφωνα με τις διαστάσεις των Skyline ερωτημάτων. Έτσι θα μπορούσαμε να δούμε τη λύση που προτείνουν, ως μία καλή λύση σε θεωρητικό επίπεδο και όχι στην πράξη.

Στο [VDKV07] παρουσιάζεται μία αρκετά καλή και απλή προσέγγιση για τον υπολογισμό Skyline ερωτημάτων σε συστήματα ομότιμων. Η λύση που προτείνεται απευθύνεται σε ένα αδόμητο σύστημα ομότιμων με Super peer αρχιτεκτονική. Παρουσιάζει αρκετά καλές αποδόσεις σε συνδυασμό όμως με κάποια σημαντικά μειονεκτήματα. Πρώτα από όλα, η πληροφορία που κρατά ένας Super peer για τους ομότιμους που του αντιστοιχούν είναι πολύ μεγάλη. Ουσιαστικά κρατά το Skyline του σε όλες τις διαστάσεις και κάποια επιπλέον δεδομένα που αφορούν το extended Skyline. Με αυτόν τον τρόπο αποφεύγεται η ερώτηση σε κάθε ομότιμο, όμως φορτώνεται πολύ ο Super peer με πληροφορία που πολλές φορές μπορεί να είναι σχεδόν όλο το σύνολο των δεδομένων. Η τεχνική στην οποία βασίζεται είναι να κρατάει για κάθε πλειάδα την μικρότερη τιμή από τις διαστάσεις της και σύμφωνα με αυτήν να «κλαδεύει» κάποια δεδομένα που σίγουρα δεν ανήκουν στο Skyline. Στις μετρήσεις τους δεν παρουσιάζεται το κατά πόσον είναι καλή αυτή η τεχνική (αν δηλαδή αποκλείονται αρκετά δεδομένα). Σκεφτείτε για παράδειγμα την περίπτωση όπου η μικρότερη τιμή δεν ανήκει στις διαστάσεις του ερωτήματος και οι υπόλοιπες διαστάσεις έχουν αρκετά μεγάλες τιμές.

7.6. Συγκεντρωτικοί Πίνακες

Σε αυτή την παράγραφο δίνονται δύο συγκεντρωτικοί πίνακες των σχετικών εργασιών που μελετήσαμε. Ο Πίνακας 7.1 περιλαμβάνει τις εργασίες που αναφέρονται σε κεντρικοποιημένα συστήματα, ενώ ο Πίνακας 7.2 τις εργασίες που αναφέρονται σε κατανεμημένα.

Πίνακας 7.1 Κεντρικοποιημένα συστήματα

paper	*Algorithm and structures*	*progressive*	*problem that it solves*	*disadvantages*	*additional features*
[BoKS01]	BNL and other window algthms (self-organizing list or LRU)		computes Skyline on systems where memory is limited	compares every tuple with almost every other tuple, requires at least one pass over the input before producing the first results	Skyline can be combined with JOIN and TopN
	D&C			sensitive to the curse of dimensionality, bad performance for small memory systems	
	Other (B-tree, R-tree)			not effective for many dimensions, limited applicability with JOIN or GROUP BY	
[TaEO01]	Bitmap (Bitmap representation of data)	✓	returns interesting points as they are identified, operations are very fast, very simple	requires a bitmap column for every distinct attribute value over all dimensions, very slow to process hole Skyline, the order in which points are returned depends on the clustering of the data	
	Index (B-tree)	✓	returns Skyline points in batches, good performance for small memory systems	requires a B-tree for every combination, the order in which points are returned depends on the value distribution of the data	
[KoRR02]	NN (R-tree)	✓	produces more and more results continuously, allows user to give preferences during the running time of the algorithm	multiple accesses of the same node, large space overhead, need of duplicate elimination, for higher than tree dimension the cost increases	
[PTFS03]	BBS (R-tree, list, heap)	✓	performs a single access only to those nodes that may contain Skyline points		Ranked, constrained, dynamic, enumerating, K-dominating Skyline

paper	*Algorithm and structures*	*progressive*	*problem that it solves*	*disadvantages*	*additional features*
[ChET05]	BBS		Skylines with partially ordered domains		
	SDC (R-tree, DAG, MST)				

Πίνακας 7.2 Κατανεμημένα συστήματα

paper	***System***	***additional structures***	***distribution***	***problem that it solves***	***disadvantages***	***additional features***
[BaGZ04]	Web information system	N-sorted lists		finds Skyline in web information systems, reduces number of accesses on project	cannot be applied on P2P	performs sampling
[Hose05]	pure P2P not structured	routing filters	random (or doesn't matter)	ask as less peer as possible in a pure P2P	probabilistic results	
[LiTL06]	structured		peers belong to clusters	finds a good way to discard data that don't belong in Skyline on a distributed environment	too costly and complicated	
	pure P2P	computes a score for every neighbor as a function of its data centroid	random (or doesn't matter)	ask as less peer as possible in a pure P2P, finds an approximate algorithm where an exact one is too costly	probabilistic results	
[WZF+06]	CAN		data is assigned to peer	discovers Skyline points progressively, pipelines query execution, minimizes inter-machine communication	not general, can only be applied on CAN	Constrained Skyline, Progressive
[VDKV07]	unstructured with Super peers	extended Skyline	random (or doesn't matter)	computes extended Skyline (for every combination of dimensions)	Super peers need to store a lot of data	
[WOTX07]	BATON		peers store a specific area of the data	alleviates :“hot spots“ problem, limits number of peers that are involved in the query, limits the amount of messages sent over the network	not general, not easy to update	
BitPeer	unstructured with Super peers	stores extended Skyline, uses Bitmap representation	Independent, Correlated, Anti-Correlated	extended Skyline, uses less bandwidth and space, easy to update structure, reuses already computed queries, small number of msgs	not progressive, asks again ONs for original data, Bitmap’s performance depends on the nature of the data	Cache, Continuous, fast queries

ΚΕΦΑΛΑΙΟ 8. ΣΥΜΠΕΡΑΣΜΑΤΑ ΚΑΙ ΜΕΛΛΟΝΤΙΚΕΣ ΕΠΕΚΤΑΣΕΙΣ

Στην εργασία αυτή έγινε μία μελέτη προσαρμογής της Bitmap προσέγγισης υπολογισμού του Skyline σε κατανεμημένη εκδοχή και συγκεκριμένα σε συστήματα ομότιμων κόμβων. Την προσέγγιση αυτή την ονομάσαμε BitPeer. Για να εφαρμοστεί η Bitmap προσέγγιση σε δεδομένα τα οποία δεν έχουν διακριτές τιμές, εισάγαμε την έννοια των κάδων στην Bitmap αναπαράσταση. Συμπεράναμε πως αν υπολογιστεί το Skyline με την Bitmap προσέγγιση στην οποία χρησιμοποιούνται κάδοι, τότε το αποτέλεσμα παύει να είναι σωστό και υπάρχουν false negatives. Το πρόβλημα αυτό αντιμετωπίστηκε με το extended Skyline. Το extended Skyline έχει την ωραία ιδιότητα ότι από αυτό μπορεί να υπολογιστεί το Skyline οποιουδήποτε υποσυνόλου των διαστάσεων.

Λόγω της ιδιότητας του extended Skyline, περιλαμβάνονται σε αυτό πολύ περισσότερα σημεία από όσα στο Skyline των ίδιων διαστάσεων. Επιπλέον η χρησιμοποίηση κάδων έχει ως αποτέλεσμα να μειώνεται η ακρίβεια των αποτελεσμάτων και να ανήκουν στο extended Skyline (αλλά και στο Skyline) ακόμα περισσότερα δεδομένα. Όσο αυξάνεται το πλήθος των κάδων, τόσο αυξάνεται και η ακρίβεια στα αποτελέσματα. Αν ο αριθμός των κάδων εξισωθεί με τον αριθμό των διακριτών τιμών που έχουν τα δεδομένα σε κάθε διάσταση, τότε το αποτέλεσμα είναι απολύτως ακριβές. Αρχικά υποθέσαμε ότι όλοι οι κάδοι είναι ίσου μεγέθους. Έπειτα προτείναμε έναν ευριστικό τρόπο ανάθεσης των κάδων, έτσι ώστε με ίσο αριθμό κάδων, τα αποτελέσματα να είναι πιο ακριβή. Στις μεγαλύτερες τιμές των διαστάσεων ανατέθηκαν κάδοι μικρού μεγέθους ώστε να αυξηθεί η ακρίβεια για υπολογισμό του MAX, ενώ στις μικρές διαστάσεις ανατέθηκαν κάδοι μεγάλου μεγέθους. Στην Independent και στην Correlated κατανομή τα δεδομένα που άνηκαν στο extended Skyline μειώθηκαν με αυτή την τεχνική. Δεν έγινε όμως το ίδιο και στην Anti-Correlated κατανομή, όπου τα δεδομένα του extended Skyline αντί να μειωθούν, αυξήθηκαν. Στην τελευταία κατανομή είναι κανόνας ότι τα δεδομένα που έχουν μεγάλη τιμή σε μία διάσταση έχουν μικρή τιμή σε άλλη, άρα την

ακρίβεια που προσθέτουμε στις μεγάλες τιμές, την χάνουμε στις μικρές. Όσον αφορά τον αριθμό των σημείων που ανήκουν στο extended Skyline, εξαρτάται πολύ και από την κατανομή που ακολουθούν τα δεδομένα. Στην Independent κατανομή ανήκει μέτριος αριθμός δεδομένων, στην Correlated πολύ λίγα, ενώ στην Anti-Correlated πάρα πολλά. Επίσης, όσο αυξάνεται ο αριθμός των διαστάσεων ενός extended Skyline ερωτήματος τόσο αυξάνεται και ο αριθμός των αποτελεσμάτων.

Για την εφαρμογή της Bitmap προσέγγισης σε συστήματα ομότιμων επιλέξαμε ένα σύστημα με Super Peer αρχιτεκτονική. Η αρχιτεκτονική αυτή συνδυάζει τα υπέρ των κεντρικοποιημένων συστημάτων και των αδόμητων συστημάτων ομότιμων. Κάθε Super Peer διατηρεί το extended Skyline όλων των Ordinary Peers που συνδέονται με αυτόν. Με αυτό τον τρόπο απαντά σε οποιοδήποτε Skyline ερώτημα τεθεί στο σύστημα χωρίς να επικοινωνήσει με τους Ordinary Peers που συνδέεται. Για να είναι το extended Skyline πάντα ενήμερο με τις αλλαγές που γίνονται στο σύστημα περιγράψαμε μία τεχνική ενημερώσεων. Το κόστος αυτών των ενημερώσεων εξαρτάται κυρίως από το ποσοστό των δεδομένων που ανήκουν στο Skyline, δηλαδή από την κατανομή με την οποία παράγονται τα δεδομένα. Στην αρχιτεκτονική αυτή προτείναμε κατάλληλο αλγόριθμο δρομολόγησης Skyline ερωτημάτων έτσι ώστε να αποφεύγονται οι κύκλοι, να ρωτάμε όσο λιγότερους ομότιμους γίνεται και να επιστρέφεται ολική πληροφορία.

Για να επισπεύσουμε τον υπολογισμό των ερωτημάτων Skyline, αλλά και για να μειώσουμε το επικοινωνιακό κόστος στην BitPeer προσέγγιση, προσθέσαμε μία cache Skyline ερωτημάτων. Την cache αυτή τη διατηρούν μόνο οι Super Peers του συστήματος και ανατρέχουν σε αυτή αν τους τεθεί κάποιο Skyline ερώτημα. Τα πειράματα έδειξαν ότι με την προσθήκη της cache το επικοινωνιακό κόστος μειώνεται έως και 45%. Ο αριθμός αυτός εξαρτάται από διάφορους παράγοντες όπως η κατανομή των Skyline ερωτήσεων του συστήματος, η χωρητικότητα της cache, ο χρόνος «λήξης» των αποθηκευμένων ερωτημάτων κτλ. Επιπροσθέτως, προτείναμε έναν πιο αποδοτικό τρόπο χρησιμοποίησης των αποτελεσμάτων της cache. Χρησιμοποιούμε ήδη υπολογισμένα ερωτήματα για να δώσουμε απάντηση σε νέα. Με αυτόν τον τρόπο μειώνονται οι υπολογισμοί. Επιπλέον εισάγαμε τα «γρήγορα» Skyline ερωτήματα, τα οποία χρησιμοποιούν αυτούσια αποτελέσματα από την cache θέτοντας ως προτεραιότητα τους το χρόνο και όχι την ακρίβεια των αποτελεσμάτων.

Επιπλέον, η προσέγγιση μας υποστηρίζει continuous Skyline ερωτήματα. Οι τύποι των ερωτημάτων αυτών είναι δύο: change-based και timer-based. Για κάθε μία από τις κατηγορίες αυτές δόθηκε ένας διαφορετικός τρόπος ομαδοποίησης των continuous ερωτημάτων. Επίσης δόθηκαν τρόποι υπολογισμού για κάθε μία από τις ομαδοποιήσεις αυτές. Με την ομαδοποίηση των ερωτημάτων γίνονται λιγότεροι υπολογισμοί και πιο είναι αποδοτικός ο υπολογισμός τους. Στα change-based ερωτήματα έχουμε έως και 50% λιγότερους υπολογισμούς όταν γίνεται ομαδοποίηση των ερωτημάτων. Τα timer-based ερωτήματα επιβαρύνουν το σύστημα πολύ λιγότερο από ότι τα change-based, όμως δίνουν λιγότερο ακριβή αποτελέσματα.

Η έρευνα που κάναμε μπορεί να συνεχιστεί σε διάφορους τομείς. Η Bitmap προσέγγιση μπορεί να επεκταθεί έτσι ώστε με την ίδια αναπαράσταση να μπορεί να γίνει αποδοτικός ο υπολογισμός του MIN, του DIFF, αλλά και συνδυασμού αυτών. Επίσης μπορεί να γίνει επιπλέον μελέτη με σκοπό την εύρεση καλύτερου τρόπου κατανομής των κάδων έτσι ώστε υπάρχει μεγαλύτερη ακρίβεια. Ο αλγόριθμος υπολογισμού του extended Skyline μπορεί να εξεταστεί περαιτέρω έτσι ώστε να γίνεται απαλοιφή περισσότερων σημείων. Το ίδιο μπορεί να γίνει και για τον Bitmap αλγόριθμο υπολογισμού του Skyline, ο οποίος εξετάζει όλα τα σημεία για κυριαρχία χωρίς να αποκλείει κανένα. Θα ήταν επίσης χρήσιμο να εξεταστεί το κατά πόσο μπορεί να μειωθεί το πλήθος των σημείων που ανήκουν στο extended Skyline, αφήνοντας αναλλοίωτες τις ιδιότητες του.

Το σύστημα υπολογισμού των Skyline ερωτήσεων μπορεί να επεκταθεί έτσι ώστε να υπολογίζει constrained και TopK ερωτήματα. Constrained Skyline ερωτήματα είναι τα ερωτήματα στα οποία ο χρήστης εκτός από MIN, MAX και DIFF, μπορεί να ζητήσει και επιπλέον περιορισμό στις τιμές που θα επιστραφούν (π.χ. η τιμή ενός ξενοδοχείου να είναι <100). Τα TopK ερωτήματα επιστρέφουν τα K καλύτερα αποτελέσματα, όπου το K καθορίζεται από το χρήστη. Ο χρήστης σε αυτή την περίπτωση καθορίζει και την βαρύτητα που δίνει σε κάθε μία από τις διαστάσεις των ερωτημάτων.

Μπορεί επίσης να εξεταστεί τεχνική δειγματοληψίας στα τελικά αποτελέσματα του Skyline. Το πλήθος των τελικών αποτελεσμάτων είναι μερικές φορές πολύ μεγάλο. Αυτό έχει ως αποτέλεσμα ο χρήστης να κατακλύζεται από πλήθος δεδομένων, τα οποία δεν μπορεί να χρησιμοποιήσει. Για να αποφευχθεί αυτό, ο χρήστης θα μπορούσε να εισάγει κάποιου είδους

προτιμήσεις ως προς το τι τον ενδιαφέρει περισσότερο. Σύμφωνα με τις προτιμήσεις αυτές θα μπορούσε να γίνει μία δειγματοληψία των αποτελεσμάτων έτσι ώστε να εμφανιστούν στο χρήστη μόνο τα δεδομένα που τον ενδιαφέρουν περισσότερο. Αντί για δειγματοληψία, θα μπορούσε επίσης να γίνει μία κατηγοριοποίηση ή ταξινόμηση των αποτελεσμάτων, ώστε να μπορεί ο χρήστης να τα χειριστεί καλύτερα.

Επιπλέον, μπορεί να υλοποιηθεί μία παραλλαγή της προσέγγισης μας στην οποία οι Super Peer θα διατηρούν μόνο cache Skyline ερωτημάτων και όχι το extended Skyline. Θα ήταν αρκετά ενδιαφέρον να μελετηθεί η απόδοση του νέου συστήματος και να γίνει σύγκριση με την BitPeer προσέγγιση. Ένα ακόμη ερώτημα το οποίο μπορεί να τεθεί υπό εξέταση είναι το κατά πόσον μπορεί να γίνει αποδοτική δρομολόγηση των ερωτήσεων αν οι ομότιμοι ομαδοποιούνται ανάλογα με τα χαρακτηριστικά των δεδομένων τους. Τέλος μπορεί να εξεταστεί το πώς η τεχνική που προτείνουμε να εφαρμόζεται με κατάλληλες τροποποιήσεις έτσι ώστε να λειτουργεί σε σύστημα στο οποίο τα δεδομένα δεν χωρίζονται οριζόντια στους κόμβους, αλλά κάθετα.

ΑΝΑΦΟΡΕΣ

[AH01] K. Aberer, M. Hauswirth. "Peer to peer information systems: concepts and models, state of the rot, and future Systems". ACM SIGSOFT Software Engineering Notes, Volume 26, Issue 5, Vienna, Austria (September 2001).

[BaGZ04] W.-T. Balke, U. Guntzer, J.X. Zheng. “Efficient Distributed Skylining for Web Information systems”. In Proc. of EDBT, pp. 256-273, 2004.

[BoKS01] S. Borzsonyi, D. Kossmann and K. Stocker. “The Skyline Operator”. In Proc. IEEE Conf. on Data Engineering, pages 421-430, Heidelberg, Germany, 2001.

[ChET05] C.Y. Chan, P.-K. Eng, K.-L. Tan. “Stratified Computation of Skylines with Partially-Ordered Domains”. Proceedings of ACM SIGMOD International Conference on Management of Data, pages 203-214, Baltimore, MD, June 2005.

[HKSZ05] K. Hose, M. Karnstedt, K.-U. Sattler, D. Zinn. “Processing Top-N Queries in P2P-based Web Integration Systems with probabilistic Guarantees”. Proceedings of the Eighth International Workshop. on the Web and Databases. WebDB 2005, pp. 109-114, Baltimore, Maryland, June 16-17, 2005.

[HoLS06] K. Hose, C. Lemke, K. Sattler. “Processing Relaxed Skylines in PDMS Using Distributed Data Summaries”. ACM Fifteenth Conference on Information and Knowledge Management (CIKM), pp. 425-434, Arlington, USA., 2006.

[Hose05] K. Hose. “Processing Skyline Queries in P2P Systems”. VLDB 2005 PhD Workshop, pp. 36-40, Trondheim, 2005.

[KoRR02] D. Kossmann, F. Ramsak and S. Rost. “Shooting Stars in the Sky: An Online Algorithm for Skyline Queries”. VLDB 2002: 275-286.

[LiTL06] H. Li, Q. Tan, W.C. Lee. “Efficient Progressive Processing of Skyline Queries in Peer-to-Peer Systems”. Proceedings of the 1st ACM International Conference on Scalable Information Systems (InfoScale), Hong Kong, May 30 - June 1, 2006.

[PFTV92] W. H. Press, B. P. Fannery, S.A. Teukolskv, W. T. Vetterling. “Numerical Recipes in C: The Art of Scientific Computing”. Cambridge University Press; 2nd edition (October 30, 1992). Available at www.nrbook.com/a/bookcpdf.php.

[PJET05] J. Pei, W. Jin, M. Ester, and Y. Tao. “Catching the Best Views of Skyline: A Semantic Approach Based on Decisive Subspaces”. In Proceedings of the 31st International Conference on Very Large Data Bases (VLDB), Trondheim, Norway, 2005.

[PTFS03] D. Papadias, Y. Tao, G. Fu and B. Seeger. “An Optimal and Progressive Algorithm for Skyline Queries”. In SIGMOD, pp. 467-478, Jun. 2003.

[TaEO01] K.-L. Tan, P.-K. Eng and B.C. Ooi. “Efficient Progressive Skyline Computation”. In Proceedings of. the Very Large Data Bases Conference (VLDB), pp. 301–310, Rome, Italy, Sep. 11–14, 2001.

[TGNO92] D. B. Terry, D. Goldberg, D. Nichols, and B. M. Oki. “Continuous queries over append-only databases”. In Proc. of the 1992 ACM SIGMOD Intl. Conf. on Management of Data, pages 321–330, June 1992.

[VDKV07] A. Vlachou, C. Doulkeridis, Y. Kotidis, M. Vazirgiannis. “SKYPEER: Efficient Subspace Skyline Computation over Distributed Data”. In the Proceedings of the 23rd IEEE International Conference on Data Engineering (ICDE), Istanbul, Turkey, April 2007.

[WOTX07] S. Wang, B. C. Ooi, A. Tung, L. Xu. “Efficient Skyline Query Processing on Peer-to-Peer Networks”. 23rd IEEE International Conference on Data Engineering (ICDE), 2007 .

[WZF+06] P. Wu, C. Zhang, Y. Feng, B. Y. Zhao, D. Agrawal and A. El Abbadi. “Parallelizing Skyline Queries for Scalable Distribution”. EDBT 2006.

[YLL+05] Y. Yuan, X. Lin, Q. Liu, W. Wang, J. X. Yu, and Q. Zhang. “Efficient Computation of the Skyline Cube”. In In Proceedings of the 31st

International Conference on Very Large Data Bases (VLDB), Trondheim, Norway, 2005.

ΠΑΡΑΡΤΗΜΑ

Γίνεται μία συνοπτική περιγραφή των σημαντικότερων εργασιών που ασχολούνται με το πρόβλημα εύρεσης του Skyline σε συστήματα Βάσεων Δεδομένων και συστήματα Ομότιμων κόμβων. Σκοπό έχει να παρουσιάσει λίγο πιο αναλυτικά στον αναγνώστη τον τρόπο με τον οποίο αντιμετωπίζεται το πρόβλημα εύρεσης του Skyline.

Π.1. Εύρεση του Skyline σε Κεντρικοποιημένο Περιβάλλον

Το επιπλέον πρόβλημα που προκύπτει όταν προσπαθήσουμε να εισάγουμε το Skyline πρόβλημα στο χώρο των βάσεων δεδομένων είναι ο περιορισμός της μνήμης. Οι παραδοσιακές λύσεις που έχουν προταθεί μπορεί να είναι αρκετά καλές όσον αφορά την χρονική πολυπλοκότητα, δεν λαμβάνουν όμως υπόψη τους προβλήματα όπως είναι η έλλειψη χώρου στη μνήμη. Σε αυτή την παράγραφο περιγράφονται συνοπτικά εργασίες οι οποίες προσπάθησαν να λύσουν το Skyline πρόβλημα σε βάσεις δεδομένων με σημαντικότερη εξ αυτών την πρώτη που περιγράφεται [BoKS01] η οποία είναι και αυτή που εισήγαγε το πρόβλημα του Skyline στο χώρο των βάσεων δεδομένων. Όλες οι μέχρι τότε δουλειές υπέθεταν ότι το σύνολο των σημείων χωράει στην μνήμη και όχι ότι είναι το αποτέλεσμα ενός ερωτήματος (query).

Π.1.1. The Skyline Operator

Οι [BoKS01] είναι οι πρώτοι που εισήγαγαν το γνωστό έως τότε ως “The maximum vector problem” ως πρόβλημα εύρεσης του Skyline σε βάσεις δεδομένων. Στόχος των συγγραφέων είναι η επέκταση της SQL κατά έναν Skyline operator. Παράλληλα παρουσιάζουν διάφορους αλγόριθμους για την υλοποίηση του και μετρούν πειραματικά τα χαρακτηριστικά τους. Τέλος

κάνουν μία μικρή αναφορά για το πώς μπορεί να συνδυαστεί με άλλες λειτουργίες σε βάσεις δεδομένων και συγκεκριμένα με το JOIN και το TopN.

Αρχικά γίνεται μια εκτενής περιγραφή του Skyline τελεστή και το τι δύναται αυτός να επιστρέψει σε ένα ερώτημα. Σημειώνεται ότι το Skyline που έχει μόνο μία διάσταση είναι ισοδύναμο με ένα MIN, MAX, ή DISTINCT ερώτημα το οποίο δεν περιλαμβάνει τον SKYLINE τελεστή.

Έπειτα περιγράφονται επτά διαφορετικοί αλγόριθμοι με τους οποίους ένα SKYLINE ερώτημα μπορεί να υλοποιηθεί. Ο πρώτος αλγόριθμος που προτείνεται είναι η μετάφραση του SKYLINE ερωτήματος σε ένα εμφωλευμένο SQL ερώτημα. Αυτή η τεχνική αν και απλή έχει πολύ κακή απόδοση, μια και αυτού του τύπου οι εμφωλευμένες ερωτήσεις ανήκουν σε αυτές που δεν μπορούν να αναδιπλωθούν (unnested). Ο δεύτερος αλγόριθμος αφορά μόνο SKYLINE ερωτήματα δύο διαστάσεων. Αν έχουμε μόνο δύο διαστάσεις η εύρεση του SKYLINE απλουστεύεται ιδιαίτερα ταξινομώντας τις δυάδες και απαλείφοντας αυτές που κυριαρχούνται από κάποια άλλη. Δυστυχώς αν έχουμε πάνω από δύο διαστάσεις η τεχνική αυτή δεν δουλεύει.

Στη συνέχεια προτείνεται ο BNL (Block Nested Loops) αλγόριθμος και δύο παραλλαγές του. Η ιδέα αυτού του αλγόριθμου είναι πολύ απλή: Συγκρίνουμε κάθε δυάδα με όλες τις υπόλοιπες. Για να το κάνουμε αυτό όσο το δυνατό πιο αποδοτικά κρατάμε ένα παράθυρο (window) από εγγραφές (tuples) στην κύρια μνήμη. Όταν διαβάζουμε μία νέα εγγραφή, αυτή συγκρίνεται με όλες τις εγγραφές που υπάρχουν στο παράθυρο. Αν κάποια την κυριαρχεί, τότε η νέα εγγραφή διαγράφεται. Αν η νέα εγγραφή κυριαρχεί κάποια άλλη, τότε αυτή και όσες άλλες εγγραφές του παραθύρου κυριαρχούνται διαγράφονται και η νέα εγγραφή εισάγεται στο παράθυρο. Αν η νέα εγγραφή δεν μπορεί να συγκριθεί με καμία εγγραφή του παραθύρου (είναι incomparable) τότε αυτή εισάγεται στο παράθυρο αν υπάρχει χώρος, ενώ αν δεν υπάρχει χώρος γράφεται σε ένα προσωρινό αρχείο για να χρησιμοποιηθεί αργότερα. Μετά από κάθε επανάληψη αφαιρούμε από το παράθυρο τις εγγραφές που έχουν συγκριθεί με όλες τις εγγραφές που βρίσκονται στο προσωρινό αρχείο. Αυτές ανήκουν σίγουρα στο Skyline. Για να βρούμε ποιες είναι αυτές οι εγγραφές χρησιμοποιούμε κάποιου είδους timestamps. Λόγω του ότι χάνεται πολύ χρόνος όταν συγκρίνουμε μία εγγραφή με όλες τις άλλες του παραθύρου έχουν προταθεί δύο ακόμη παραλλαγές αυτού του αλγορίθμου. Η

πρώτη διατηρεί το παράθυρο ως self-organizing list ενώ στη δεύτερη το παράθυρο διατηρείται ως LRU (Least Recently Used) αντικατάστασης.

Μία ακόμη λύση είναι ο D&C (Divide and Conquer) αλγόριθμος, του οποίου η πολυπλοκότητα είναι καλύτερη από αυτήν του BNL στη χειρότερη περίπτωση αλλά πολύ χειρότερη στην καλύτερη περίπτωση. Η ιδέα αυτού του αλγορίθμου είναι να χωρίσουμε το σύνολο των εγγραφών σε δύο υποσύνολα όπου το ένα είναι καλύτερο από το άλλο στη μία διάσταση του Skyline ερωτήματος. Υπολογίζουμε αναδρομικά το Skyline των δύο συνόλων μέχρι τα σύνολα να περιέχουν ένα μόνο στοιχείο. Στο τέλος υπολογίζεται το ολικό Skyline των δύο αρχικών συνόλων. Ο αλγόριθμος αυτός έχει τρομερά κακή απόδοση όταν η είσοδος του δεν χωρά στην κύρια μνήμη. Έτσι, προτείνεται μία τροποποίηση όπου το αρχικό σύνολο δεν χωρίζεται σε δύο υποσύνολα, αλλά σε m, έτσι ώστε κάθε ένα από αυτά να χωράει στην κύρια μνήμη. Ένας άλλος τρόπος να βελτιώσουμε την απόδοση είναι να φορτώσουμε στην μνήμη όσες εγγραφές χωράνε και εφαρμόσουμε σε αυτές τον αρχικό αλγόριθμο, βρίσκοντας έτσι ένα αρχικό Skyline και απαλείφοντας αμέσως αρκετές εγγραφές.

Επίσης, προτείνονται άλλοι δύο αλγόριθμοι οι οποίοι βασίζονται στην χρησιμοποίηση ευρετηρίων. Ο ένας χρησιμοποιεί B-δένδρα, ενώ ο άλλος R-δένδρα. Η ιδέα σε αυτήν την περίπτωση είναι να χρησιμοποιήσουμε την διάταξη των εγγραφών για να απαλείψουμε εγγραφές οι οποίες σίγουρα δεν ανήκουν στο Skyline. Στην περίπτωση των B-δένδρων υποθέτουμε ότι υπάρχει ένα ευρετήριο για κάθε διάσταση και υπολογίζουμε το Skyline με τον τρόπο που ακολουθεί: Σκανάρουμε όλα τα ευρετήρια ταυτόχρονα μέχρι να ανακαλύψουμε την πρώτη εγγραφή που βλέπουμε σε όλα τα ευρετήρια κατά τη διάρκεια του σκαναρίσματος. Αυτή η εγγραφή είναι σίγουρα μέρος του Skyline και μπορεί να επιστραφεί στο χρήστη. Οι εγγραφές που δεν εμφανίζονται σε κανένα ευρετήριο πριν την πρώτη εγγραφή μπορούν να διαγραφούν. Για όλες τις υπόλοιπες εγγραφές μπορεί να εφαρμοστεί ένας από τους υπόλοιπους αλγόριθμους. Το μειονέκτημα αυτού του αλγόριθμου είναι ότι δεν έχει καλή απόδοση όταν έχουμε μεγάλο αριθμό διαστάσεων ή όταν ο αριθμός των εγγραφών που ανήκουν στο Skyline είναι μεγάλος. Τέλος αναφέρεται με ποιον τρόπο μπορεί να συνδυαστεί ο SKYLINE τελεστής με τα JOIN και TopN.

Στα πειράματα που γίνονται για μέτρηση της απόδοσης των αλγόριθμων μεταβάλλονται ο τύπος των δεδομένων που χρησιμοποιούνται, ο αριθμός των διαστάσεων του ερωτήματος και

οι διαθέσιμοι buffers κύριας μνήμης. Τα στοιχεία μιας βάσης μπορούν να είναι είτε ανεξάρτητα, είτε correlated, είτε anti-correlated. Στην περίπτωση της correlated βάσης, τα στοιχεία που ανήκουν στο Skyline είναι πολύ λίγα, ενώ αυξάνονται καθώς πάμε στην ανεξάρτητη και anti-correlated. Σύμφωνα με τα πειράματα οι διάφορες παραλλαγές των BNL αλγόριθμων είναι αποδοτικές όταν το μέγεθος του Skyline είναι μικρό (δηλ. στην correlated περίπτωση). Δεν λειτουργεί όμως αποδοτικά όσο αυξάνεται το πλήθος των διαστάσεων του ερωτήματος. Οι παραλλαγές του D&C αλγόριθμου είναι λιγότερο ευαίσθητες στον αριθμό των διαστάσεων και στο correlation της βάσης (των στοιχείων).

Π.1.2. Efficient Progressive Skyline Computation

Στην εργασία αυτήν [TaEO01] παρουσιάζονται δύο αλγόριθμοι για τον υπολογισμό του Skyline ενός συνόλου σημείων, ο Bitmap και ο Index. Η διαφορά τους από τους υπόλοιπους αλγόριθμους είναι ότι δεν απαιτούν ένα πέρασμα από όλα τα σημεία εισόδου για να παράγουν το πρώτο σημείο του Skyline, αλλά επιστρέφουν ενδιαφέροντα σημεία, ακριβώς την στιγμή που θα τα συναντήσουν (είναι δηλ. progressive). Η τεχνική αυτή είναι ιδιαίτερα ενδιαφέρουσα και χρήσιμη, ιδιαίτερα σε εφαρμογές όπου ο χρήστης ενδιαφέρεται να δει πολύ γρήγορα κάποια πρώτα αποτελέσματα.

Ο πρώτος αλγόριθμος που παρουσιάζεται είναι ο Bitmap. Σε αυτόν τον αλγόριθμο κάθε εγγραφή αντιστοιχίζεται σε ένα m-bit διάνυσμα, όπου m είναι το άθροισμα του αριθμού των διαφορετικών τιμών των attributes σε όλες τις διαστάσεις. Αυτή η bitmap δομή είναι μία δομή που υπολογίζεται μία φορά εξ αρχής και περιέχει περισσότερη πληροφορία από ότι θα περιείχε μία απλή bitmap δομή ολόκληρης της βάσης δεδομένων. Στη διάσταση i αντιστοιχίζονται τόσα bit όσες είναι οι διαφορετικές τιμές των attributes σε αυτή τη διάσταση. Το πρώτο bit αντιστοιχεί στη μεγαλύτερη τιμή της διάστασης, το δεύτερο στη δεύτερη μεγαλύτερη κ.ο.κ. Αν η τιμή του σημείου σε αυτή την διάσταση έχει την j-οστή τιμή τότε τα bit από 1 έως j τίθενται σε 0 και όλα τα υπόλοιπα σε 1. Ο πίνακας που αναπαράστασης των σημείων αποθηκεύεται σε τμήματα (slices), όπου το κάθε τμήμα αναπαριστά την j-οστή τιμή της i-οστής διάστασης. Με πράξεις AND και OR πάνω στο κάθε τμήμα μπορούμε πολύ γρήγορα να καθορίσουμε αν ένα σημείο ανήκει ή όχι στο Skyline

χωρίς να χρειαστεί να εξετάσουμε τις υπόλοιπες εγγραφές. Αυτή η διαδικασία μπορεί γίνεται πολύ γρήγορα αφού όλες οι πράξεις είναι πράξεις με bit.

Ο δεύτερος αλγόριθμος που παρουσιάζεται είναι ο Index και βασίζεται στα B-δένδρα. Κάθε σημείο αντιστοιχίζεται σε σημείο του μονοδιάστατου χώρου, σύμφωνα με τη μέγιστη τιμή του σε κάθε διάσταση. Δηλαδή έχουμε τόσες ομάδες, όσες είναι και οι διαφορετικές διαστάσεις και σε κάθε μία ομάδα ανήκουν τα σημεία που έχουν μέγιστη τιμή στην αντίστοιχη διάσταση. Κάθε μία από τις ομάδες ταξινομείται ως προς την διάσταση που αντιστοιχεί. Μετά τον μετασχηματισμό αυτό, χρησιμοποιούμε ένα B-δένδρο ως ευρετήριο, του οποίου όμως τα φύλλα έχουν δείκτες και στο δεξί και στο αριστερό αδέρφι τους. Παρατηρούμε ότι:

- τα ενδιαφέροντα σημεία βρίσκονται στην κορυφή. Ελέγχουμε πρώτα τις εγγραφές των οποίων οι τιμές είναι οι μέγιστες. Κάποιες από αυτές είναι ξεκάθαρο ότι ανήκουν στο Skyline.
- αρκετά σημεία μπορούν να απαλειφθούν εύκολα, χωρίς καν να τα εξετάσουμε. Αν η μικρότερη τιμή μιας εγγραφής είναι μεγαλύτερη από την μεγαλύτερη κάποιας άλλης, τότε η δεύτερη δεν ανήκει στο Skyline και πρέπει να απαλειφτεί. Μαζί της συμπαρασύρει και όλες όσες βρίσκονται κάτω από αυτήν στη διάταξη της διάστασης στην οποία παρατηρείται η μέγιστη τιμή.
- στην χειρότερη περίπτωση εφαρμόζεται μία από τις υπάρχουσες τεχνικές σκανάροντας τα φύλλα. Και σε αυτή την περίπτωση υπάρχει κέρδος, διότι εμπλέκονται μόνο διαστάσεις.
- η εσωτερική δομή του δένδρου βελτιώνεται αν ταξινομήσουμε τα σημεία που έχουν την ίδια μέγιστη τιμή σύμφωνα με την μικρότερη τιμή τους.
- δεν απαιτείται μεγάλη κύρια μνήμη.

Οι αλγόριθμοι αυτοί συγκρίνονται με τους α) BNL όπου το παράθυρο υλοποιείται ως self-organizing λίστα, β) D&C όπου η είσοδος χωρίζεται σε m υποσύνολα, γ) με τον βασισμένο σε B-δένδρα αλγόριθμο σε συνδυασμό με τον BNL. Τα δεδομένα στις βάσεις παράγονται ακριβώς με τον ίδιο τρόπο όπως και στο [BoKS01], με την διαφορά ότι αντί να παράγουνε double στο [0, 1], παράγουνε ακέραιους στο [0, 100]. Γίνονται τέσσερις μετρήσεις. Αρχικά οι αλγόριθμοι συγκρίνονται όσον αφορά τον ολικό χρόνο που χρειάζονται για τον υπολογισμό

του Skyline. Ο Index αλγόριθμος υπερτερεί έναντι των άλλων σε όλες τις περιπτώσεις, ενώ και ο BNL αλγόριθμος έχει καλή απόδοση στην περίπτωση της correlated βάσης. Η δεύτερη μέτρηση που γίνεται αφορά το πόσο γρήγορα επιστρέφονται οι πρώτες απαντήσεις. Οι Bitmap και Index φαίνεται να κερδίζουν τον αγώνα όσον αφορά κυρίως τις πρώτες απαντήσεις. Ο Index είναι καλύτερος από όλους σε όλες τις περιπτώσεις, ενώ ο Bitmap μαζί με τον D&C είναι αρκετά αργοί στην περίπτωση της correlated βάσης δεδομένων. Στην επόμενη μέτρηση παρατηρούμε ότι η απόδοση των Index και Bitmap δεν επηρεάζεται ιδιαίτερα από το μέγεθος των buffers, ενώ στην τελευταίο ότι ο αριθμός των διαστάσεων επηρεάζει αρκετά την απόδοση των BNL και Bitmap και πολύ λιγότερο αυτή των Index και D&C.

Π.1.3. Shooting Stars in The Sky: An Online Algorithm for Skyline Queries

Σε αυτήν την εργασία [KoRR02] παρουσιάζεται ένας online αλγόριθμος για τον υπολογισμό του Skyline ενός συνόλου σημείων. Αρχικά παρουσιάζεται ο τρόπος που δουλεύει ο αλγόριθμος στις δύο διαστάσεις και έπειτα επεκτείνεται σε περισσότερες. Για λόγους ευκολίας οι συγγραφείς υποθέτουν ότι όλες οι τιμές είναι θετικοί, πραγματικοί αριθμοί, δεν υπάρχουν διπλοεγγραφές και ψάχνουμε ελάχιστα σημεία (MIN). Ο αλγόριθμος αυτός μπορεί να επεκταθεί έτσι ώστε να βρίσκει και μέγιστα σημεία, όμως δεν δουλεύει στην περίπτωση του DIFF.

Σύμφωνα με τους συγγραφείς ένας online αλγόριθμος πρέπει να πληροί συνοπτικά τις εξής προϋποθέσεις: α) να επιστρέφει πολύ γρήγορα τα πρώτα αποτελέσματα, β) να παράγει όλο και περισσότερα αποτελέσματα, ώσπου να υπολογίσει ολόκληρο το Skyline, γ) να επιστρέφει μόνο σημεία που ανήκουν στο Skyline και όχι απλά «καλά» σημεία, δ) να είναι δίκαιος, να μην επιστρέφει δηλαδή αρχικά μόνο σημεία που είναι ιδιαίτερα καλά σε μία διάσταση, ε) ο χρήστης να μπορεί να ελέγχει τη διαδικασία και στ) να είναι Universal όσον αφορά τα ερωτήματα και το σύνολο των δεδομένων.

Κανείς από τους αλγόριθμους που παρουσιάστηκαν στο [BoKS01] δεν πληροί αυτές τις προϋποθέσεις. Όσον αφορά αυτούς του [TaEO01], πληρούν τις πρώτες τρεις μόνο προϋποθέσεις. Δεν είναι δίκαιοι διότι η σειρά με την οποία επιστρέφονται τα δεδομένα στον

Bitmap εξαρτάται από τη σειρά με την οποία τα δεδομένα είναι ομαδοποιημένα, ενώ στον Index από την κατανομή των δεδομένων. Επιπλέον κανείς από τους δύο δεν επιτρέπει αλληλεπίδραση με το χρήστη. Τέλος, όσον αφορά τον Index, η εφαρμογή του είναι περιορισμένη λόγω του ότι απαιτεί ένα B-δένδρο για κάθε συνδυασμό των διαστάσεων που μπορεί να έχει το ερώτημα. Επίσης κανείς από τους δύο δεν μπορεί να εφαρμοστεί σε δυναμικό (κινητό) περιβάλλον όπου κάποιες ιδιότητες είναι παραμετρικές, όπως π.χ. η απόσταση από τον χρήστη.

Ο αλγόριθμος που παρουσιάζεται σε αυτήν την εργασία ονομάζεται NN αλγόριθμος επειδή βασίζεται στην Nearest Neighbor αναζήτηση. Ο αλγόριθμος αυτός δεν εφαρμόζεται αν το Skyline ερώτημα περιέχει διάσταση που δεν είναι στο ευρετήριο, ή αν το ερώτημα περιλαμβάνει μεγάλα JOIN ή GROUP BY. Αρχικά γίνονται δύο παρατηρήσεις για το δισδιάστατο χώρο: α) ο nearest neighbor του O(0, 0), σύμφωνα με κάποια μονότονη συνάρτηση f, ανήκει σίγουρα στο Skyline, β) αν κάνουμε nearest neighbor αναζήτηση σε μία περιοχή γύρω από το O(0, 0), τότε και ο νέος nearest neighbor θα ανήκει στο Skyline. Με λίγα λόγια ο αλγόριθμος δουλεύει ως εξής: Αρχικά βρίσκεται ο nearest neighbor του O(0, 0) και χωρίζουμε το επίπεδο σε τρία μέρη σύμφωνα με αυτόν. Το ένα από αυτά δεν χρειάζεται να εξεταστεί καθόλου, ενώ στα άλλα δύο βρίσκουμε του αντίστοιχους nearest neighbors και χωρίζουμε το επίπεδο σε επιπλέον υποεπίπεδα σύμφωνα με αυτούς. Ο αλγόριθμος συνεχίζει έτσι αναδρομικά μέχρι που κάποιο τμήμα να μην έχει άλλο σημείο.

Ο αλγόριθμος αυτός μπορεί με τον ίδιο τρόπο να εφαρμοστεί και στις d διαστάσεις. Το πρόβλημα όμως που προκύπτει είναι πως ένα σημείο μπορεί να βρίσκεται σε πάνω από μία υποπεριοχές. Προτείνονται λοιπόν κάποιες τεχνικές επίλυσης του προβλήματος των διπλοεγγραφών. Αρχικά προτείνεται η τεχνική “Laisser-faire”. Σε αυτή την τεχνική τα σημεία του Skyline κρατούνται σε ένα hash table. Κάθε φορά που βρίσκουμε έναν nearest neighbor ελέγχουμε αν υπάρχει στο hash table. Αν δεν υπάρχει σημαίνει ότι είναι νέο σημείο του Skyline και το εμφανίζουμε στο χρήστη. Αν υπάρχει τότε δεν εμφανίζεται. Και στις δύο περιπτώσεις συνεχίζουμε να χωρίζουμε το χώρο. Το μειονέκτημα αυτού του αλγόριθμου είναι ότι καθυστερεί πολύ διότι μπορεί να βρει το ίδιο σημείο μέχρι και d-1 φορές. Επίσης στην περίπτωση που υπάρχουν πραγματικά διπλότυπα, θα πρέπει να δώσει σε όλες τις εγγραφές διαφορετικό αναγνωριστικό.

Ένας άλλος τρόπος είναι να αντιμετωπίσουμε το πρόβλημα πριν συμβεί (Propagation). Κάθε φορά που βρίσκουμε ένα νέο σημείο του Skyline, ψάχνουμε να βρούμε όλες τις υποπεριοχές στις οποίες το σημείο αυτό ανήκει και τις χωρίζουμε σύμφωνα με αυτό το σημείο. Το πρόβλημα είναι ότι χρειάζεται αρκετή δουλειά για να το κάνουμε αυτό. Μία τρίτη τεχνική που μπορεί να χρησιμοποιηθεί είναι η συνένωση ή απαλοιφή συγκεκριμένων περιοχών. Στην τελευταία τεχνική που περιγράφηκε μπορούμε να απαλείψουμε περιοχές που κυριαρχούνται από κάποιες άλλες, ενώ στην πρώτη περίπτωση συνενώνουμε περιοχές που προέκυψαν από το ίδιο σημείο του Skyline. Μία ακόμη τεχνική είναι να χωρίζουμε την περιοχή σε υποπεριοχές μεταξύ των οποίων δεν υπάρχει τομή. Τέλος μπορούμε να συνδυάσουμε κάποιες από τις παραπάνω τεχνικές. Για παράδειγμα μπορούμε να χρησιμοποιήσουμε την propagation τεχνική και όταν φτάσουμε να έχουμε αρκετές υποπεριοχές να υλοποιήσουμε την laisser-faire τεχνική.

Στις μετρήσεις τους χρησιμοποιούν βάσεις δεδομένων με διαφορετικό μέγεθος, διαφορετική κατανομή τιμών και διαφορετικό αριθμό διαστάσεων. Μετράνε τον χρόνο υπολογισμού των αρχικών αποτελεσμάτων, αλλά και του ολικού Skyline και τα συγκρίνουν με άλλες μεθόδους. Συγκεκριμένα συγκρίνονται οι NN, D&C, BNL, Bitmap, και B-tree για ερωτήσεις στις δύο και ερωτήσεις σε υψηλότερες διαστάσεις. Ο NN αλγόριθμος είναι ο νικητής, ειδικά όταν πρόκειται για χρόνο επιστροφής των πρώτων αποτελεσμάτων. Τέλος ανάμεσα στις τεχνικές που προτείνονται για την αντιμετώπιση των διπλοεγγραφών, υπερτερούν οι τεχνική της συνένωσης και η υβριδική.

Π.1.4. An Optimal and Progressive Algorithm for Skyline Queries

Σε αυτήν την εργασία [PTFS03] παρουσιάζεται ο Branch and Bound Skyline αλγόριθμος (BBS). Ο αλγόριθμος αυτός είναι ένας progressive (online) αλγόριθμος που βασίζεται στην Nearest Neighbor αναζήτηση όπως και ο NN που παρουσιάζεται στο [KoRR02] με την διαφορά ότι ο BBS όπως ισχυρίζονται οι συγγραφείς είναι I/O optimal. Για τις Skyline ερωτήσεις θεωρούν μόνο το MIN.

Αρχικά περιγράφονται συνοπτικά οι αλγόριθμοι των [BoKS01], [TaEO01], και [KoRR02]. Κίνητρο τους για την συγγραφή της εργασίας, ήταν το να βελτιώσουν τον NN αλγόριθμο ο

οποίος παρουσιάζει αρκετά μειονεκτήματα. Κάποια από αυτά είναι α) η ανάγκη απαλοιφής των διπλοεγγραφών, β) οι πολλαπλές προσβάσεις στον ίδιο κόμβο και γ) ο μεγάλος όγκος χώρου που χρειάζεται για να τρέξει. Αντιθέτως ο BBS αλγόριθμος α) εκτελεί μία μόνο πρόσβαση στον κάθε κόμβο, και μάλιστα μόνο σε κόμβους που μπορεί να περιέχονται στο Skyline, β) δεν επιστρέφει διπλότυπα, γ) χρειάζεται λιγότερο χώρο για να τρέξει από ότι ο NN, δ) είναι ευκολότερα υλοποιήσιμος και ε) μπορεί να εφαρμοστεί αποδοτικά σε ευρεία γκάμα Skyline ερωτήσεων. Επιπλέον ο BBS αλγόριθμος πληροί όλες τις προϋποθέσεις ενός online αλγόριθμου, όπως αυτές παρουσιάστηκαν στο [KoRR02]. Παρακάτω δίνουμε μία συνοπτική περιγραφή του αλγόριθμου.

Στον αλγόριθμο χρησιμοποιούνται τα R-δένδρα λόγω του ότι είναι πολύ απλά και πολύ δημοφιλή. Ο αλγόριθμος μοιάζει με τον αλγόριθμο εύρεσης του κοντινότερου γείτονα, μόνο που συνεχίζει να εκτελείται μετά την εύρεση του πρώτου nearest neighbor. Τα σημεία του συνόλου δεδομένων αποθηκεύονται σε ένα R-δένδρο με τον τρόπο που ακολουθεί: τα σημεία βρίσκονται στα φύλλα του δένδρου, ενώ οι εσωτερικοί κόμβοι αντιστοιχούν στο Minimum Bounding Rectangle (MBR) των φύλλων που τους αντιστοιχούν. Οι αποστάσεις υπολογίζονται ως εξής: η απόσταση ενός σημείου είναι το άθροισμα των συντεταγμένων και η απόσταση ενός MBR είναι η απόσταση του κάτω αριστερά σημείου του. Ο αλγόριθμος ξεκινά από την ρίζα του δένδρου και εισάγει τις εκεί εγγραφές σε έναν σωρό σύμφωνα με την απόσταση τους. Έπειτα επιλέγει την εγγραφή με την μικρότερη απόσταση και την επεκτείνει. Αυτό σημαίνει ότι αφαιρεί την εγγραφή από το σωρό και εισάγει σε αυτόν τα παιδιά της. Αυτό συνεχίζεται μέχρι να βρεθεί ο πρώτος nearest neighbor, ο οποίος βγαίνει από το σωρό και εισάγεται στη λίστα των σημείων που ανήκουν στο Skyline. Μετά από το σημείο συνεχίζουμε να επεκτείνουμε τους κόμβους με τη διαφορά ότι μόνο εγγραφές που δεν κυριαρχούνται από τα σημεία που ανήκουν στη λίστα με τα Skyline εισάγονται στο σωρό. Ο αλγόριθμος συνεχίζει με αυτόν τον τρόπο μέχρι ο σωρός να αδειάσει τελείως. Σημειώνουμε ότι ο έλεγχος για κυριαρχία γίνεται δύο φορές, μία πριν η εγγραφή μπει στο σωρό και μία πριν επεκταθεί.

Ο έλεγχος για κυριαρχία είναι ένα πολύ σημαντικό θέμα καθώς μπορεί να αποβεί πολύ ακριβό όταν έχουμε πολλά σημεία που να ανήκουν στο Skyline. Για να επιταχύνουν αυτή τη διαδικασία, οργανώνουν τη λίστα με τα σημεία του Skyline ως ένα R-δένδρο που χωρά στην κύρια μνήμη. Έτσι, ο έλεγχος για κυριαρχία μπορεί να γίνει με παρόμοιο τρόπο όπως και τα

windowed queries. Μία εγγραφή κυριαρχείται από κάποιο σημείο p του Skyline, αν το κάτω αριστερό σημείο του πέφτει μέσα στην περιοχή κυριαρχίας του p (το ορθογώνιο που ορίζεται από το p και το τέλος του σύμπαντος.

Στη συνέχεια της εργασίας αποδεικνύεται η ορθότητα του αλγορίθμου καθώς και οι ισχυρισμοί των συγγραφέων ότι επισκέπτεται κάθε κόμβο μία μόνο φορά και ότι επισκέπτεται μόνο κόμβους οι οποίοι μπορεί να ανήκουν στο Skyline.

Τέλος αναφέρουνε το πώς ο αλγόριθμος μπορεί να τροποποιηθεί έτσι ώστε να υποστηρίζει παραλλαγές των Skyline ερωτήσεων. Συγκεκριμένα, μπορεί αποδοτικά να υποστηρίξει ranked Skyline queries, αν η συνάρτηση απόστασης που ορίστηκε παραπάνω τροποποιηθεί σύμφωνα με την συνάρτηση την οποία θέλουμε κάθε φορά να επιστρέφουμε τα αποτελέσματα. Επιπλέον ο αλγόριθμος θα τερματίζει αφού έχει βρει τα K πρώτα σημεία. Από τους αλγόριθμους που παρουσιάστηκαν στις άλλες εργασίες κανείς δεν υποστηρίζει τις ranked Skyline ερωτήσεις αποδοτικά. Απαιτούν πρώτα υπολογισμό ολόκληρου του Skyline, και έπειτα ταξινομούν για να πάρουν τα K πρώτα στοιχεία. Παρόμοιο κόστος έχει ο NN ο οποίος δεν μπορεί να βρει ένα καλό κριτήριο τερματισμού λόγω του ότι ανήκει σε «Διαίρει και βασίλευε» αλγόριθμους. Ο BBS υποστηρίζει επίσης αποδοτικά constrained Skyline ερωτήματα. Η διαφορά σε αυτή την περίπτωση με τον αρχικό αλγόριθμο είναι ότι στο σωρό εισάγονται μόνο εγγραφές που τέμνονται την constrained περιοχή. Επίσης με μικρές τροποποιήσεις, μπορεί να υπολογίσει δυναμικά Skyline ερωτήματα, enumerating και K-dominating ερωτήματα.

Τέλος κάνουνε σύγκριση του BBS αλγόριθμου με τον NN όσον αφορά τη συμπεριφορά τους ανάλογα με τον αριθμό των διαστάσεων του ερωτήματος, το πόσο γρήγορα επιστρέφουν τα πρώτα αποτελέσματα, και την αποδοτικότητα τους όσον αφορά τα constrained ερωτήματα. Ο BBS αλγόριθμος συμπεριφέρεται καλύτερα όσον αφορά τον αριθμό των κόμβων που επισκέπτεται και τον χρόνο CPU που χρειάζεται.

Π.1.5. Stratified Computation of Skylines with partially-Ordered Domains

Το πρόβλημα που προσπαθεί αυτό το άρθρο [ChET05] να επιλύσει είναι η εύρεση του Skyline ενός συνόλου, όταν τα χαρακτηριστικά πάνω στα οποία τίθεται το ερώτημα δεν μπορούν πλήρως να ταξινομηθούν. Η πιο απλή λύση θα ήταν η επίλυση του προβλήματος με τον BNL αλγόριθμο [BoKS01]. Το πρόβλημα με αυτόν τον αλγόριθμο είναι ότι δεν μπορεί να επιστρέψει αποτελέσματα αν δεν υπολογίσει πρώτα ολόκληρο το Skyline. Επίσης, έχουν προταθεί άλλοι αλγόριθμοι οι οποίοι βασίζονται σε ευρετήρια (Index [TaEO01], BBS [PTFS03]), οι οποίοι αποδεικνύεται να έχουν πολύ καλύτερη απόδοση από τον BNL αφού δεν συγκρίνουν όλες τις εγγραφές μεταξύ τους. Η άμεση εφαρμογή των αλγόριθμων που βασίζονται σε ευρετήρια δεν είναι εφικτή αφού πλέον τα χαρακτηριστικά του ερωτήματος δεν μπορούν πλήρως να ταξινομηθούν. Μία αντιστοίχιση των χαρακτηριστικών αυτών σε άλλα, που μπορούν πλήρως να ταξινομηθούν με χρήση boolean τιμών, δεν αποδεικνύεται τόσο καλή λύση λόγω του "curse of dimensionality".

Σε αυτό το άρθρο προτείνονται τρεις αλγόριθμοι για την επίλυση του προβλήματος, ο BBS+ που είναι μια παραλλαγή του BBS, ο SDC (Stratisfaction by Dominance Classification) και SDC+ καθώς και δύο βελτιώσεις πάνω στον τελευταίο.

Η βασική ιδέα των συγγραφέων είναι να αντιστοιχίσουν κάθε χαρακτηριστικό με ένα διάστημα (για κάθε attribute δηλαδή έχουμε πλέον δύο αριθμούς). Έπειτα χτίζουν ένα ευρετήριο σύμφωνα με τα νέα δεδομένα και τρέχουν έναν αλγόριθμο που βασίζεται σε αυτό το ευρετήριο. Σημειώνουμε ότι ο αλγόριθμος δεν εξαρτάται από τον τρόπο με τον οποίο αντιστοιχίζουμε τα δεδομένα. Από τη στιγμή που ορίζουνε την αντιστοίχιση με τα διαστήματα, δίνουν και τον νέο ορισμό κυριαρχίας για τα μετασχηματισμένα δεδομένα Ένα διάστημα κυριαρχεί ένα άλλο, αν το δεύτερο διάστημα εμπεριέχεται πλήρως στο πρώτο. Τα διαστήματα για τα οποία δεν ισχύει κάτι τέτοιο, δεν μπορούν να συγκριθούν μεταξύ τους. Στην πραγματικότητα όμως (δηλαδή πριν κάνουμε αυτήν την αντιστοίχιση), τα δεδομένα αυτά μπορεί να ήταν και συγκρίσιμα και το ένα να κυριαρχεί το άλλο (false positives). Έτσι, αφού τρέξουμε τον αλγόριθμο, θα πρέπει να κάνουμε έναν επιπλέον έλεγχο για να βρούμε αυτά τα δεδομένα και να τα διαγράψουμε. Τέλος λόγω της αντιστοίχισης πρέπει να ορίσουν μία άλλη μορφή κυριαρχίας, την m-dominance η οποία με λίγα λόγια λέει το εξής: μία εγγραφή για να κυριαρχεί μία άλλη, θα πρέπει να ισχύει ο κανονικός ορισμός κυριαρχίας για

τα χαρακτηριστικά της που είναι πλήρως ταξινομήσιμα και ο ορισμός κυριαρχίας που δόθηκε για μετασχηματισμένα δεδομένα.

Ο αλγόριθμος BBS+ είναι ίδιος με τον BBS με την διαφορά ότι ο έλεγχος κυριαρχίας αντικαθίσταται με έλεγχο m-κυριαρχίας, και όταν προσθέτουμε δεδομένα στο Skyline γίνεται επιπλέον έλεγχος έτσι ώστε να ανιχνευτούν και να διαγραφούν τα false positives. Λόγω αυτής της λειτουργίας δεν μπορούμε να εμφανίζουμε στο χρήστη αμέσως τα πρώτα αποτελέσματα διότι δεν είμαστε σίγουροι για το αν θα ανήκουν στο Skyline. Ένα ακόμη μειονέκτημα του BBS+ είναι ότι κάνει πολλούς (και πιθανότατα περιττούς) ελέγχους για κυριαρχία όταν πάει να προσθέσει ένα νέο στοιχείο στο Skyline.

Για την αποφυγή των μειονεκτημάτων του BBS+ προτείνεται ο SDC αλγόριθμος. Η βάση αυτού του αλγόριθμου είναι μία κατηγοριοποίηση των μετασχηματισμένων δεδομένων, με τη βοήθεια του σκελετικού τους δένδρου (αυτό το δέντρο υπάρχει και χρησιμοποιείται για τον μετασχηματισμό των δεδομένων). Τα δεδομένα χωρίζονται σε τέσσερις κατηγορίες και για αυτές τις κατηγορίες ορίζεται ένας γράφος κυριαρχίας. Δοθέντος αυτού του γράφου προσπαθούν αρχικά να μειώσουν τις συγκρίσεις μεταξύ των δεδομένων (συγκρίνουν μόνο αυτές που χρειάζεται, σύμφωνα με το γράφο) και έπειτα να αυξήσουν τις συγκρίσεις μεταξύ των μετασχηματισμένων δεδομένων, σε σχέση με αυτές που γίνονται στα αρχικά δεδομένα (τις οποίες χρησιμοποιούν μόνο ως έσχατη λύση). Αυτό συμβαίνει επειδή η εύρεση κυριαρχίας για δύο δεδομένα είναι κάτι ακριβό σε σχέση με την εύρεση κυριαρχίας για τα μετασχηματισμένα δεδομένα, όπου απλώς συγκρίνουμε δύο ακέραιους. Τέλος, ο αλγόριθμος μπορεί να γίνει προοδευτικός, αφού σύμφωνα με τον γράφο κυριαρχίας, κάποια δεδομένα ανήκουν σίγουρα στο Skyline και μπορούν αμέσως να εξαχθούν στο χρήστη.

Αυτό που ουσιαστικά κάνει ο SDC είναι να χωρίζει τα δεδομένα σε δύο strata, όπου το ένα ανήκει σίγουρα στο Skyline και μπορεί να εξαχθεί αμέσως στο χρήστη, ενώ για το άλλο πρέπει να γίνουν οι απαραίτητοι έλεγχοι. Επειδή το πρώτο strata συνήθως είναι αρκετά μικρό και έτσι δεν πετυχαίνουμε μεγάλη προοδευτικότητα, σχεδιάστηκε ο SDC+ αλγόριθμος, ο οποίος έχει ως στόχο να αυξήσει την προοδευτικότητα του αλγόριθμου χωρίζοντας τα δεδομένα σε μεγαλύτερο αριθμό από strata. Η ιδέα είναι να έχουμε κάποια strata (μία ακολουθία υποσυνόλων), για τα οποία θα υπολογίζουμε ξεχωριστά το Skyline σύνολο του καθενός, έτσι ώστε ένα σημείο που ανήκει στο Skyline ενός strata να μην κυριαρχείται από τα

σημεία των Skyline των επόμενων strata. Η πιο προφανής λύση είναι να χωρίσουμε τα δεδομένα σε τέσσερα σε τέσσερα strata σύμφωνα με τις κατηγορίες του γράφου κυριαρχίας. Πάλι όμως τα δύο τελευταία strata είναι αρκετά μεγάλα και μειώνουν την προοδευτικότητα του αλγόριθμου. Έτσι οι συγγραφείς ορίζουν έναν αριθμό (uncovered level) για κάθε χαρακτηριστικό (το πλήθος των κατευθυνόμενων ακμών προς αυτόν τον κόμβο που υπάρχουν στο DAG των δεδομένων, αλλά όχι στο σκελετικό τους δένδρο). Το uncovered level ενός σημείου είναι το μέγιστο uncovered level των χαρακτηριστικών του. Ένα σημείο δεν μπορεί να κυριαρχείται από κάποιο άλλο αν το uncovered level του είναι μεγαλύτερο. Έπειτα χωρίζουν το κάθε ένα από τα δύο σύνολα σε τόσα υποσύνολα όσος είναι και το μέγιστο uncovered level των σημείων του. Αυτά τα strata ικανοποιούν την ιδιότητα που θέσαμε στην αρχή της παραγράφου. Κάθε strata είναι ευρετηριοποιημένο σε ένα R-tree. Το Skyline κάθε strata υπολογίζεται έχοντας ως παραμέτρους το R-tree του και το Skyline του προηγούμενου strata.

Αυτό που οι συγγραφείς παρατηρούν είναι ότι ανάλογα με τον τρόπο που χτίζουμε το σκελετικό δένδρο, αλλάζει η δομή των συνόλων (strata) που παίρνουμε. Έτσι προτείνουν δύο βελτιώσεις, τις MinPC και MaxPC, όπου στην πρώτη μειώνονται οι συγκρίσεις για κυριαρχία, ενώ στην δεύτερη αυξάνονται οι συγκρίσεις που γίνονται για m-dominance.

Στις μετρήσεις που κάνουν συγκρίνουν τους BNL, BNL+ (πρώτα τρέχει ο κανονικός BNL χρησιμοποιώντας τις μετασχηματισμένες τιμές και μετά τρέχει ένας δεύτερος για να απαλείψει τα false positives), BBS, BBS+, SDC και SDC+, όσον αφορά το χρόνο που επιστρέφουν τα πρώτα δεδομένα και τον χρόνο που κάνουν για όλο το Skyline σε σχέση με διαφορετικό αριθμό δεδομένων. Τέλος συγκρίνουν και τις βελτιώσεις που έκαναν και βρίσκουν ότι ο αλγόριθμος που συμπεριφέρεται καλύτερα είναι ο SDC+ με το MinPC.

Π.2. Εύρεση του Skyline σε Κατανεμημένο Περιβάλλον

Οι ανάγκες και τα χαρακτηριστικά ενός κατανεμημένου περιβάλλοντος παρουσιάζουν αρκετές διαφορές με αυτά των παραδοσιακών βάσεων δεδομένων. Ιδιαίτερα όταν μιλάμε για συστήματα ομότιμων, οι διαφορές αυξάνονται τόσο πολύ ώστε αλλάζουν εξ' ολοκλήρου οι παράμετροι του προβλήματος. Συνοπτικά, έχουμε να κάνουμε με ένα κατανεμημένο

περιβάλλον όπου είναι συχνές οι αφίξεις και αναχωρήσεις ομότιμων, δεν υπάρχει κεντρική πληροφορία, όλοι ομότιμοι μπορούν να θέσουν ένα ερώτημα, και είναι πιθανό να υπάρχουν ομότιμοι με διαφορετικό σχήμα στη βάση δεδομένων τους. Οι τεχνικές που είχαν πρωτύτερα προταθεί, δεν σέβονται τα ιδιαίτερα χαρακτηριστικά των συστημάτων ομότιμων και έτσι δεν μπορούν να εφαρμοστούν σε αυτά. Είτε απαιτούν μία κεντρική μονάδα, είτε βασίζονται σε ευρετήρια που αναπαριστούν κεντρική πληροφορία, είτε χρειάζονται αρκετά round-trips (ρωτούν πολλές φορές τον ίδιο ομότιμο) που καταλήγουν σε υψηλά κόστη εκτέλεσης. Σε αυτήν την παράγραφο εξετάζουμε τεχνικές οι οποίες ασχολούνται αποκλειστικά με την επίλυση του προβλήματος σε κατανεμημένο περιβάλλον.

Π.2.1. Efficient Distributed Skylining for Web Information Systems

Στόχος αυτού του άρθρου [BaGZ04] είναι να επεκτείνει την εκφραστικότητα των ερωτήσεων που μπορούν να γίνουν σε Web Information Systems. Οι περισσότεροι αλγόριθμοι που έχουν προταθεί για αυτά τα συστήματα, αφορούν top k ερωτήσεις, οι οποίες βασίζονται σε μία μόνο συνάρτηση που χρησιμοποιείται για να αντισταθμίσει τα σκορ μεταξύ διάφορων τμημάτων της ερώτησης. Σε αντίθεση με τις top k ερωτήσεις, οι Skyline έχουν μεγαλύτερη εκφραστικότητα, αφού μπορείς να ορίσεις συναρτήσεις που δίνουν ένα επιθυμητό «βάρος» σε κάθε διάσταση. Οι αλγόριθμοι που έως τώρα έχουν προταθεί στοχεύουν σε κεντρικοποιημένες βάσεις δεδομένων και έτσι κρίνονται ακατάλληλοι για Web Information Systems. Έτσι, οι συγγραφείς προτείνουν έναν κατανεμημένο αλγόριθμο για την εφαρμογή Skyline ερωτήσεων σε Web συστήματα. Αυτός ο αλγόριθμος στοχεύει πλέον στη μείωση του αριθμού των προσβάσεων που γίνονται στα αντικείμενα και όχι στην μείωση της χρησιμοποίησης της κύριας μνήμης.

Αρχικά προτείνουν έναν βασικό κατανεμημένο αλγόριθμο και έπειτα τον βελτιώνουν με την βοήθεια κάποιων ευριστικών. Ο βασικός κατανεμημένος αλγόριθμος για υπολογισμό του Skyline (Basic Distributed Skyline Algorithm) υποθέτει την ύπαρξη n ταξινομημένων λιστών, όπου κάθε λίστα είναι ταξινομημένη ως προς μία διάσταση. Έχει τα εξής τέσσερα βήματα: πρώτα, κάνει ταξινομημένες προσβάσεις στις λίστες, μέχρι να βρει όλα τα σημεία που μπορεί να ανήκουν στο Skyline, έπειτα επεκτείνει τις προσβάσεις σε όλα τα αντικείμενα που έχουν τα μικρότερα σκορ στις λίστες και κλαδεύει όλα τα υπόλοιπα αντικείμενα της βάσης. Τέλος

κάνει τυχαίες προσβάσεις σε συγκεκριμένα αντικείμενα, έτσι ώστε να απαλείψει όλα τα αντικείμενα που κυριαρχούνται από κάποια άλλα. Ουσιαστικά προσπαθούν να βρουν το αντικείμενο που εμφανίζεται πρώτο σε όλες στις λίστες, μετά απαλείφουν όλα τα αντικείμενα που δεν είδανε και ελέγχουν το τι γίνεται μεταξύ των στοιχείων που έχουνε δει. Για να μπορεί να γνωρίζει ποια αντικείμενα έχουμε προσβεί, χρησιμοποιεί μία δομή που περιέχει όλη την απαραίτητη πληροφορία (συγκεκριμένα χρησιμοποιεί έναν πίνακα στον οποίο φαίνεται το ποια αντικείμενα έχουμε προσβεί και από ποιες λίστες προέκυψαν). Επίσης χρησιμοποιεί n επιπλέον πίνακες K στους οποίους στο τέλος του αλγορίθμου βρίσκονται όλα τα στοιχεία τα οποία πρέπει να ελέγξουμε αν κυριαρχούνται από κάποια άλλα. Το ωραίο αυτού του αλγόριθμου κρύβεται στο γεγονός ότι στο τέλος απομένουν λίγα μόνο αντικείμενα τα οποία πρέπει να ελεγχθούν για κυριαρχία και έτσι μπορούν γρήγορα να επιστραφούν γρήγορα κάποια σημεία που ανήκουν στο Skyline.

Έπειτα αποδεικνύεται η ορθότητα του αλγόριθμου βασιζόμενη σε δύο λήμματα. Το πρώτο μας λέει ότι είναι ασφαλές να διαγράψουμε όλα τα αντικείμενα τα οποία δεν έχουμε δει διότι δεν πρόκειται να ανήκουν στο Skyline, ενώ το δεύτερο ότι τα αντικείμενα μπορεί να κυριαρχούνται μόνο από κάποιο αντικείμενο που ανήκει στον ίδιο με αυτά πίνακα K. Τέλος αποδεικνύουν το βέλτιστο του αλγόριθμου ως προς τις ταξινομημένες προσβάσεις που κάνουν στα αντικείμενα.

Για να βελτιώσουν την απόδοση του αλγόριθμου κάνουν τη χρήση κάποιων ευριστικών. Αρχικά προσπαθούν να βρουν όσο το δυνατόν πιο γρήγορα το στοιχείο που εμφανίζεται πρώτο σε όλες τις λίστες, έτσι ώστε να μειωθούν οι συγκρίσεις που γίνονται μετά στους πίνακες K. Για κάθε αντικείμενο που προσβαίνουν ταξινομημένα στις λίστες κάνουν μία τυχαία πρόσβαση ώστε να αποκτήσουν τις τιμές (σκορ) που έχει σε όλα τα πεδία του. Έπειτα υπολογίζουν για αυτό το άθροισμα των διαφορών όλων των τιμών του στις λίστες που αυτό ακόμη δεν εμφανίστηκε μείον την τιμή που εθεάθη τελευταία στην αντίστοιχη λίστα. Αυτό που έχει την μικρότερη τιμή θεωρείται το πιο πιθανό και έτσι σταματάμε τις προσβάσεις στη λίστα που το βρήκαμε και προσπαθούμε να το βρούμε στις υπόλοιπες λίστες. Σημειώνουμε ότι η τεχνική αυτή δεν προσθέτει επιπλέον κόστος λόγω των τυχαίων προσβάσεων, διότι αυτές θα γινόταν έτσι κι αλλιώς αργότερα στον αλγόριθμο.

Η δεύτερη ευριστική που χρησιμοποιούν αφορά των αριθμό των συγκρίσεων που γίνονται μεταξύ των πινάκων K. Εκεί παρατηρούν ότι μπορούν να γίνουν αρχικά έλεγχοι μεταξύ των αντικειμένων που έχουν τις ίδιες τιμές (έτσι ώστε να απαλειφθούν πολλά αντικείμενα) και έπειτα να γίνουν έλεγχοι κυριαρχίας μεταξύ των αντικειμένων που προέκυψαν. Με αυτόν τον τρόπο γλιτώνουμε αρκετές επιπλέον συγκρίσεις. Σύμφωνα με τις δύο αυτές ευριστικές προτείνουν ένα βελτιωμένο αλγόριθμο βασισμένο στον αρχικό τους (Improved Distributed Skyline Algorithm).

Μετά από την περιγραφή των αλγόριθμων κάνουν κάποιες μετρήσεις. Οι αλγόριθμοι τους δεν μπορούν να συγκριθούν με κάποιον άλλον, επειδή έως τότε δεν είχε προταθεί κάποιος άλλος κατανεμημένος αλγόριθμος. Έτσι κάνουνε μετρήσεις του ίδιου του αλγόριθμου όσον αφορά τον αριθμό των προσβάσεων που γίνονται σε αντικείμενα, τον συνολικό αριθμό των αντικειμένων στη βάση δεδομένων, και το πόσο καλύτερος είναι ο βελτιωμένος αλγόριθμος που προτείνουν. Αρχικά μετρούν το πόσο καλύτερος είναι ο βελτιωμένος τους αλγόριθμος όσον αφορά τις προσβάσεις σε αντικείμενα και βρίσκουν ότι είναι καλύτερος κατά έναν παράγοντα 1,5 με 2,5 ανάλογα με των αριθμό των λιστών που χρησιμοποιούνται. Έπειτα δείχνουν ότι ο αριθμός των αντικειμένων που μπορούν να «κλαδευτούν» δεν εξαρτάται από το μέγεθος της βάσης δεδομένων, αλλά εξαρτάται από τον αριθμό των λιστών που χρησιμοποιούνται (δηλαδή από τον αριθμό των διαστάσεων του ερωτήματος). Αυτό συμβαίνει επειδή όταν χρησιμοποιούμε περισσότερες λίστες είναι μεγαλύτερος ο αριθμός των αντικειμένων που μπορεί να ανήκουν στο Skyline.

Στην τελευταία παράγραφο του άρθρου προτείνεται μία τεχνική δειγματοληψίας. Σύμφωνα με τις μετρήσεις που κάνανε, όσο αυξάνεται ο αριθμός των λιστών αυξάνονται και τα στοιχεία του Skyline με αποτέλεσμα μέχρι και 50% περίπου των στοιχείων μιας βάσης δεδομένων να ανήκουν στο Skyline. Το να επιστραφούν όλα αυτά τα στοιχεία στο χρήστη προφανώς δεν είναι κάτι το επιθυμητό αφού δεν θα μπορεί να τα διαχειριστεί. Αυτό που κάνουν είναι να ελέγχουν αν υπάρχει κάποια συσχέτιση (correlation) μεταξύ κάποιων διαστάσεων έτσι ώστε αυτές να μπορέσουν να συνενωθούν σε μία. Επιλέγουν q υποσύνολα των λιστών, υπολογίζουν το Skyline αυτών και τέλος συνθέτουν τα αποτελέσματα ώστε να παραχθεί το τελικό Skyline. Σύμφωνα με μετρήσεις που κάνανε η τεχνική αυτή επιστρέφει ένα υποσύνολο του Skyline, με τον ίδιο σχεδόν τρόπο όπως αν επιλέγαμε με τυχαίο τρόπο ένα υποσύνολο αυτού.

Π.2.2. Processing Skyline Queries in P2P Systems

Το άρθρο αυτό [Hose05] είναι από τα πρώτα που πρότειναν εφαρμογή της Skyline λειτουργίας σε συστήματα ομότιμων. Αρχικά αναφέρεται στις προκλήσεις που έχουν να αντιμετωπίσουν οι σχεδιαστές αλγορίθμων για συστήματα ομότιμων καθώς και τους λόγους για τους οποίους οι υπάρχουσες τεχνικές δεν είναι ικανοποιητικές. Σκεπτόμενοι τα παραπάνω οι συγγραφείς του άρθρου προτείνουν μία τεχνική εύρεσης του Skyline σε συστήματα ομότιμων, η οποία δεν βασίζεται σε ολική πληροφορία, και επεξεργάζεται αποδοτικά τα ερωτήματα όσον αφορά το κόστος εκτέλεσης (ρωτά μόνο μερικούς ομότιμους και τους ρωτά μόνο μία φορά).

Γίνεται ένας διαχωρισμός των στρατηγικών που χρησιμοποιούνται για υπολογισμό σε συστήματα ομότιμων. Χωρίζονται σε: α) Naive: συλλέγει όλα τα δεδομένα σε αυτόν που έθεσε την ερώτηση και υπολογίζει τοπικά το αποτέλεσμα, β) Nearest Neighbor: υπολογίζει τον nearest neighbor (NN) πλημμυρίζοντας το δίκτυο, διαχωρίζει το χώρο σύμφωνα με τον NN και βρίσκει αναδρομικά για NN σε κάθε κομμάτι, γ) D&C: πλημμυρίζει το δίκτυο, κάθε ομότιμος υπολογίζει τοπικά το Skyline και ενώνουν τις απαντήσεις για να σταλούν σε αυτόν που το έθεσε, δ) Probabilistic: χρησιμοποιούνται φίλτρα δρομολόγησης για να αποφασιστεί σε ποιους ομότιμους θα σταλεί η ερώτηση.

Σε αυτό το άρθρο χρησιμοποιείται η τελευταία στρατηγική. Γενικά αυτό που προσπαθούν να κάνουν για να μειώσουν το κόστος είναι μην ρωτάνε τους ομότιμους που δεν μπορούν να συνεισφέρουν στο τελικό αποτέλεσμα. Κάθε ομότιμος διατηρεί ευρετήρια που αφορούν τις διαστάσεις της ερώτησης. Σύμφωνα με αυτά τα ευρετήρια, η ερώτηση προωθείται μόνο στους ομότιμους που μπορεί να περιέχουν σχετικά με την ερώτηση δεδομένα. Επειδή τα ευρετήρια περιέχουν μόνο μία προσέγγιση της πληροφορίας των ομότιμων δεν μπορούμε να είμαστε 100% σίγουροι για το αποτέλεσμα. Έτσι η απάντηση που δίνεται στον χρήστη εμπεριέχει και το ποσοστό ορθότητας της απάντησης.

Τα ευρετήρια που χρησιμοποιούν οι συγγραφείς είναι ιστογράμματα. Κάθε ομότιμος διατηρεί ένα τέτοιο ευρετήριο για κάθε ομότιμο με τον οποίο συνδέεται. Το ευρετήριο αυτό περιγράφει όλα τα XML δεδομένα τα οποία μπορούμε να προσβούμε αν προωθήσουμε το ερώτημα σε αυτόν τον ομότιμο. Το διάστημα τιμών ενός attribute χωρίζεται σε κάδους

(τμήματα), και για κάθε τμήμα κρατάμε τη μέση συχνότητα όλων των τιμών των attributes που ανήκουν σε αυτό το τμήμα. Κάθε ομότιμος αποφασίζει ανεξάρτητα από τους άλλους και σύμφωνα με τα ευρετήρια του, σε ποιους ομότιμους θα προωθήσει την ερώτηση.

Ο αλγόριθμος παίρνει ως είσοδο το Skyline του ομότιμου που θέτει την ερώτηση και την πιθανότητα λάθους την οποία θέλει ο χρήστης να έχουν τα αποτελέσματα. Κάθε ομότιμος που λαμβάνει μία ερώτηση υπολογίζει το Skyline του και έπειτα ελέγχει για κυριαρχία σε σχέση με το Skyline που του έχει δοθεί ως είσοδος. Υπολογίζει το σύνολο των γειτονικών του ομότιμων, οι οποίοι περιέχουν σχετική με την ερώτηση πληροφορία. Έπειτα στέλνει ένα μέρος του αρχικού Skyline που έχει υπολογίσει σε αυτόν που του έκανε την ερώτηση, έτσι ώστε να είναι δυνατόν να λάβει ο χρήστης γρήγορα κάποια πρώτα αποτελέσματα. Μετά επιλέγει τους ομότιμους στους οποίους θα προωθήσει την ερώτηση σύμφωνα με την πιθανότητα λάθους που ο χρήστης έχει εισάγει. Τέλος όταν λάβει τα αποτελέσματα από τους γείτονες του, τα συνδυάζει και τα στέλνει στον ομότιμο που του έθεσε την ερώτηση.

Τέλος θεωρεί το Skyline ως πολυδιάστατη επέκταση των top-N ερωτήσεων και παρουσιάζει την τεχνική και τα αποτελέσματα του [HKSZ05] όσον αφορά το λόγο της ορθότητας με την πιθανότητα ορθότητας που δίνει ο χρήστης.

Π.2.3. Efficient Progressive Processing of Skyline Queries in Peer-to-Peer Systems

Σε αυτό το άρθρο [LiTL06] παρουσιάζονται δύο αλγόριθμοι υπολογισμού του Skyline. Ο πρώτος υποθέτει ότι τα δεδομένα ανήκουν σε clusters και υπολογίζει το ακριβές Skyline εξαιρώντας εξαρχής κάποια cluster- ομότιμους οι οποίοι δεν περιέχουν δεδομένα που να ανήκουν στο Skyline. Ο δεύτερος θεωρεί ότι το σύστημα ομότιμων είναι πλήρως αδόμητο (pure P2P) και δίνει έναν αλγόριθμο ο οποίος δίνει μία προσέγγιση του Skyline των δεδομένων. Παρακάτω περιγράφεται συνοπτικά ο τρόπος με τον οποίο αυτοί οι αλγόριθμοι δουλεύουν.

Στο μοντέλο που υποθέτει ο πρώτος αλγόριθμος, ο χώρος χωρίζεται σε clusters κάθε ένα από τα οποία έχει ένα κομμάτι του χώρου. Οι ομότιμοι ανατίθενται σε ένα cluster ανάλογα με το κεντροειδές (centroid) των δεδομένων τους. Με αυτόν τον τρόπο ένας ομότιμος ενδέχεται να

περιέχει δεδομένα τα οποία δεν ανήκουν στο cluster στο οποίο έχει ανατεθεί. Για αυτά τα δεδομένα υπάρχουν ευρετήρια στον απομακρυσμένο ομότιμο ο οποίος είναι υπεύθυνος για αυτά τα δεδομένα. Κατά την ανάθεση των ομότιμων σε clusters λαμβάνεται υπόψη η τοπικότητα κάθε ομότιμου έτσι ώστε ομότιμοι που βρίσκονται κοντά στην πραγματικότητα να βρίσκονται κοντά και στο σύστημα. Επιπλέον, κάθε ομότιμος διατηρεί μία λίστα στην οποία βρίσκονται ένας ή περισσότεροι γειτονικοί του ομότιμοι. Σημειώνουμε ότι κάθε cluster αντιπροσωπεύεται από την τιμή – ubound που έχουν οι βέλτιστες διαστάσεις που ανήκουν σε αυτό και από τον ομότιμο που τις έχει (π.χ. στο παράδειγμα με τα ξενοδοχεία ένα cluster θα αντιπροσωπεύεται από το ξενοδοχείο με την χαμηλότερη τιμή που είναι και πιο κοντά στην παραλία, ή αν θεωρήσουμε το κάτω αριστερά σημείο του ως (0,0) θα αντιπροσωπεύεται από τον Nearest Neighbor του (0,0)).

Ο αλγόριθμος περιέχει τέσσερις διαδικασίες: α) εντοπισμός του cluster που καλύπτει τις βέλτιστες τιμές των διαστάσεων - C0, β) εξαίρεση των cluster που δεν περιέχουν Skyline δεδομένα, γ) εξαίρεση των ομότιμων ενός cluster, που δεν περιέχουν Skyline δεδομένα και δ) υπολογισμός του Skyline για κάθε ομότιμο. Για το α) υπεύθυνος είναι ο ομότιμος που κάνει την ερώτηση. Στο β) εξαιρούνται όλα τα clusters των οποίων το ubound κυριαρχείται από το ubound του C0. Στα cluster τα οποία δεν κυριαρχούνται από το C0 το ερώτημα προωθείται στον ομότιμο που τα αντιπροσωπεύει. Στο cluster συνεχίζουμε να χρησιμοποιούμε το ubound για να βρούμε τους ομότιμους που κυριαρχούνται. Οι υπόλοιποι ομότιμοι επιστρέφουν το τοπικό τους Skyline στον ομότιμο που αντιπροσωπεύει το cluster, ο οποίος υπολογίζει το συνολικό Skyline του cluster του. Το τελικό Skyline υπολογίζεται στον ομότιμο που έθεσε το ερώτημα.

Το δεύτερο μοντέλο που θεωρεί το άρθρο είναι ένα πλήρως αδόμητο σύστημα ομότιμων, όπου κάθε ομότιμος συνδέεται σε κάποιους γειτονικούς του. Σε ένα τέτοιο σύστημα δεν είναι αποδοτικός ο ακριβής υπολογισμός του Skyline. Για κάθε ομότιμο θεωρούμε ότι γνωρίζουμε το κεντροειδές των δεδομένων του. Όταν ένας ομότιμος λαμβάνει ένα Skyline ερώτημα στο σύστημα, πρέπει να αποφασίσει σε ποιον γειτονικό του ομότιμο θα το προωθήσει. Ο ομότιμος υπολογίζει το σκορ του κάθε ομότιμου σύμφωνα με κάποια συνάρτηση η οποία λαμβάνει υπόψη το κεντροειδές του και επιλέγει να προωθήσει το ερώτημα στον ομότιμο με το μικρότερο σκορ. Κάθε ομότιμος που επισκεπτόμαστε αναφέρεται ως visited για να μην τον επισκεφθούμε πάνω από μία φορά. Κάθε ομότιμος που λαμβάνει το ερώτημα υπολογίζει το

Skyline του μαζί με τα αποτελέσματα που λαμβάνει από τον ομότιμο που το προώθησε και στέλνει τα αποτελέσματα στον ομότιμο που του έστειλε το ερώτημα.

Επειδή με αυτήν την τεχνική είναι πιθανόν να μην υπολογιστούν στο Skyline δεδομένα που μπορεί να είναι πολύ καλά, οι συγγραφείς προτείνουν μία βελτίωση αυτού του αλγόριθμου. Πλέον ο ομότιμος δεν στέλνει το ερώτημα σε έναν μόνο γείτονα του, αλλά στους j καλύτερους σύμφωνα με τα σκορ που έχει υπολογίσει. Αλλάζοντας το j μπορούμε να αλλάξουμε την ακρίβεια του αποτελέσματος.

Στις μετρήσεις που κάνουν υπολογίζουν α) τον μέσο αριθμό των hops για υπολογισμό ενός ερωτήματος, β) το φόρτο που προκαλούν στο δίκτυο από την μεταφορά των ενδιάμεσων αποτελεσμάτων, γ) το ποσοστό των cluster που επισκέπτονται και δ) το ποσοστό των ομότιμων που επισκέπτονται και τέλος ε) την ποιότητα των αποτελεσμάτων στο δεύτερο μοντέλο που προτείνουν.

Π.2.4. Processing Relaxed Skylines in PDMS Using Distributed Data Summaries

Οι συγγραφείς του συγκεκριμένου άρθρου [HoLS06] προσπαθούν να λύσουν αποδοτικά το πρόβλημα των Skyline ερωτήσεων σε ένα PDMS (Peer Data Management System). Το επιπλέον χαρακτηριστικό ενός τέτοιου συστήματος είναι ότι ο κάθε ομότιμος μπορεί να έχει διαφορετικό σχήμα στη βάση δεδομένων του. Λαμβάνοντας λοιπόν υπόψη τα χαρακτηριστικά των συστημάτων ομότιμων και θέτοντας ως στόχο την μείωση του χρόνου εκτέλεσης των Skyline ερωτήσεων καταλήγουν στη χρησιμοποίηση φίλτρων δρομολόγησης και στη μείωση των απαιτήσεων για ακριβείς απαντήσεις.

Βασικό στοιχείο των προσεγγίσεων που προτείνονται είναι ο τρόπος αναπαράστασης των δεδομένων σε κάθε ομότιμο. Σε κάθε ομότιμο υπάρχει μία «περίληψη» των δεδομένων του δικτύου καθώς και τα δικά του δεδομένα για τα οποία έχει πλήρη στοιχεία. Τα υπόλοιπα δεδομένα διατηρούνται σε μορφή «περιοχών» για τις οποίες κρατείται μόνο πληροφορία που είναι απαραίτητη για να βρούμε αν μία περιοχή κυριαρχεί ή κυριαρχείται από κάποια άλλη. Δίνεται ορισμός κυριαρχίας για περιοχές και κάθε περιοχή θεωρείται ως ένα σημείο. Ο αλγόριθμος που ακολουθείται έπειτα για τον υπολογισμό του Skyline περιλαμβάνει τρία

βήματα. Αρχικά ο ομότιμος υπολογίζει το τοπικό του Skyline, έπειτα προωθεί το ερώτημα μόνο στους ομότιμους οι οποίοι έχουν σημεία στις περιοχές που ανήκουν στο τοπικό Skyline. Τέλος ο ομότιμος συγκεντρώνει τα αποτελέσματα όλων των ομότιμων στους οποίους έστειλε το ερώτημα και είτε εμφανίζει το αποτέλεσμα στο χρήστη, είτε προωθεί το αποτέλεσμα στον ομότιμο ο οποίος του έστειλε το ερώτημα.

Στην απλή αυτή τεχνική που περιγράφηκε παραπάνω βασίζονται και στην επόμενη προσέγγιση η οποία προσπαθεί να μειώσει το κόστος εκτέλεσης των ερωτημάτων. Αρχικά ορίζουν το "relaxed Skyline", το οποίο διαισθητικά σημαίνει ότι για κάθε σημείο ή σύνολο σημείων που ανήκει στο Skyline ορίζεται ένας αντιπρόσωπος-σημείο, το οποίο βρίσκεται σε κάποια απόσταση από το πραγματικό/ά σημεία. Η απόσταση αυτή ορίζεται από τον χρήστη και προστίθεται στον ορισμό του εκάστοτε ερωτήματος. Σύμφωνα με αυτά, η διαφορά του νέου αλγόριθμου είναι ότι όταν υπολογίζεται το τοπικό Skyline, ο ομότιμος ψάχνει αν μπορεί να βρει κάποιον γνωστό αντιπρόσωπο για τις περιοχές που του είναι άγνωστες. Αν βρει κάποιον αντιπρόσωπο, τότε δεν προωθεί το ερώτημα στους ομότιμους αυτών των περιοχών και έτσι μειώνεται το κόστος υπολογισμού του ερωτήματος. Αν η περιοχή είναι αρκετά μεγάλη (σε σχέση με το λάθος που επιτρέπεται) τότε αναζητούνται πάνω από ένας αντιπρόσωποι. Τέλος, αφού έχει βρει το relaxed Skyline, προσπαθεί να βρει το minimal σύνολο των αντιπροσώπων που μπορούν να αναπαραστήσουν τις περιοχές.

Στην τελευταία παράγραφο του άρθρου περιγράφεται μία δομή δεδομένων που μπορεί να χρησιμοποιηθεί για την αναπαράσταση του DDS στους ομότιμους. Η δομή αυτή είναι το Q-tree, μία δομή που συνδυάζει τα πλεονεκτήματα των R-trees και των ιστογραμμάτων. Το Q-tree όμοια με το R-tree περιλαμβάνει MBB (Minimum Bounding Boxes) αλλά και κάδους. Οι κάδοι είναι τα φύλλα του δένδρου και μπορούν να περιλαμβάνουν επιπλέον πληροφορία η οποία στην προκειμένη περίπτωση είναι ο αριθμός των σημείων που περιέχει το MMB. Επιπλέον κάθε Q-tree χαρακτηρίζεται από δύο παραμέτρους: α) το μέγιστο αριθμό παιδιών που μπορεί να έχει ένας κόμβος και β) τον μέγιστο αριθμό κ. Τα δύο αυτά χαρακτηριστικά περιορίζουν το μέγεθος του. Σε αντίθεση με τα R-trees το Q-tree δεν χρειάζεται να είναι ισορροπημένο.

Στα πειράματα που παρουσιάζονται στο τέλος του άρθρου δείχνουν ότι πραγματικά μειώνεται το κόστος εκτέλεσης. Ο χρήστης ως αποτέλεσμα της ερώτησης του παίρνει μία γενική ιδέα

(ενημερώνεται για κάθε αντιπρόσωπο, ποια περιοχή αντιπροσωπεί και πόσα σημεία η περιοχή αυτή περιλαμβάνει). Αν για κάποια περιοχή ο χρήστης ενδιαφέρεται ιδιαίτερα, τότε έχει τη δυνατότητα να θέσει ένα πιο συγκεκριμένο ερώτημα. Επιπλέον αν θέλει ακριβή αποτελέσματα, ο χρήστης μπορεί εξ αρχής να τα έχει με το αντίστοιχο κόστος υπολογισμού, αφού αυτός είναι που καθορίζει την παράμετρο λάθους-απόστασης του ερωτήματος.

Π.2.5. Parallelizing Skyline Queries for Scalable Distribution

Σε αυτό το άρθρο [WZF+06] προσπαθούν να λύσουν το πρόβλημα των Skyline ερωτήσεων σε ένα σύστημα ομότιμων, όπου έχουμε διαμοιρασμό των δεδομένων στους κόμβους σύμφωνα με το περιεχόμενο τους. Κάτι τέτοιο συμβαίνει στο σύστημα ομότιμων CAN, πάνω στο οποίο υλοποιούν και αυτοί την τεχνική τους και κάνουν μετρήσεις. Ο αλγόριθμος που παρουσιάζουν ονομάζεται DSL (Distributed SkyLne Query) και έχει δύο βασικά βήματα. Πρώτα διαχωρίζει δυναμικά τα δεδομένα σε υποπεριοχές και έπειτα αναθέτει κωδικούς σε κάθε περιοχή έτσι ώστε κάθε περιοχή να γνωρίζει τι βήματα πρέπει να κάνει. Στόχος τους είναι να προτείνουν έναν προοδευτικό αλγόριθμο, scalable και ευέλικτο. Δίνουν σημασία στη δυνατότητα απάντησης constrained Skyline ερωτήσεων.

Στο πρώτο βήμα του αλγόριθμου γίνεται ένας διαχωρισμός των περιοχών σε μικρότερες με τέτοιον τρόπο ώστε κάθε περιοχή να εξαρτάται μόνο από κάποιες άλλες περιοχές. Οι περιοχές αυτές κατατάσσονται με τέτοιο τρόπο ώστε όταν μία περιοχή υπολογίζει το Skyline της να είναι σίγουρη ότι αυτό θα ανήκει στο ολικό Skyline (λόγω αυτής της κατάταξης μπορεί ο αλγόριθμος να επιστρέφει προοδευτικά τα αποτελέσματα στο χρήστη). Η κατάταξη αυτή γίνεται σύμφωνα τον Skyline Dependent ορισμό, ο οποίος ουσιαστικά καθορίζει ποια περιοχή κυριαρχεί κάποια άλλη. Αρχικά ο χώρος χωρίζεται σε 2d περιοχές σύμφωνα με το άνω δεξιά σημείο του κόμβου που βρίσκεται κάτω αριστερά στο χώρο που αφορά το ερώτημα. Αυτό που θέλουμε είναι κάθε υποπεριοχή που έχουμε ορίσει να είναι υπό την κατοχή ενός μόνο κόμβου. Έτσι η παραπάνω διαδικασία συνεχίζεται αναδρομικά μέχρι να ισχύει αυτή η συνθήκη.

Επειδή όλη αυτή η διαδικασία γίνεται δυναμικά κάθε φορά που θέτουμε ένα ερώτημα, κάθε κόμβος πρέπει να ενημερώνεται εκείνη τη στιγμή για το σε ποια περιοχή ανήκει, ποιους

πρέπει να περιμένει για να επιστρέψει το Skyline του κτλ. Για αυτόν τον λόγο καθορίζεται μία κωδικοποίηση κάθε φορά που γίνεται ένας διαχωρισμός και έτσι ο κόμβος παίρνει όλη την πληροφορία που χρειάζεται.

Ανακεφαλαιώνοντας, συνοπτικά ο αλγόριθμος λειτουργεί ως εξής: ο αρχικός κόμβος υπολογίζει το Skyline του, χωρίζει τον υπόλοιπο χώρο σε υποπεριοχές στέλνοντας στους αντίστοιχους κόμβους την κατάλληλη πληροφορία. Όταν ένα κόμβος λάβει ένα ερώτημα, πριν αρχίσει τους υπολογισμούς περιμένει μέχρι να λάβει δεδομένα από όλους τους προηγούμενους του. Αν η περιοχή του κυριαρχείται από κάποιους άλλους, δεν κάνει υπολογισμούς και επιστρέφει. Αλλιώς εκτελεί τα ίδια βήματα με τον πρώτο κόμβο, και στο τέλος επιστρέφει τον έλεγχο στον προκάτοχο του.

Με αυτόν τον αλγόριθμο επιτυγχάνονται σε δύο σημαντικά ζητήματα. Το πρώτο είναι ότι δεν γίνεται αλόγιστη χρησιμοποίηση bandwidth, επειδή σε κάθε κόμβο αποστέλλονται μόνο δεδομένα που ανήκουν σίγουρα στο Skyline. Το δεύτερο είναι ότι ελαχιστοποιούνται οι κόμβοι στους οποίους τίθεται το ερώτημα. Αυτό συμβαίνει επειδή μία περιοχή δεν χωρίζεται σε υποπεριοχές αν κυριαρχείται από Skyline σημεία που ήδη έχουν υπολογιστεί.

Τέλος χρησιμοποιούν μια τεχνική αντιγράφων (replication) για να επιτύχουν εξισορρόπηση φορτίου στους κόμβους, όσον αφορά τις τοπικές ερωτήσεις στις οποίες οι κόμβοι πρέπει να απαντήσουν. Κάθε κόμβος περιοδικά επιλέγει 10 τυχαία σημεία στο χώρο του CAN. Έπειτα ρωτά τους αντίστοιχους κόμβους για το φόρτο τους και συλλέγει τα αποτελέσματα. Αν παρατηρήσει ότι δέχεται περισσότερο φόρτο από τους μισούς, τότε δημιουργεί ένα αντίγραφο των δεδομένων του και το στέλνει στον λιγότερο φορτωμένο κόμβο από αυτούς που τυχαία επέλεξε. Κάθε κόμβος διατηρεί μία λίστα αντιγράφων. Όταν μία Skyline ερώτηση τίθεται σε έναν κόμβο τότε αυτός επιλέγει κυκλικά έναν κόμβο από αυτούς που κρατά στη λίστα των αντιγράφων του. Με αυτόν τον τρόπο επιτυγχάνεται καλύτερη κατανομή του φορτίου στους κόμβους.

Στις μετρήσεις που κάνουν συγκρίνουν τον DSL αλγόριθμο με τους Can-Multicast και Naïve. Ο Naïve χτίζει ουσιαστικά μία ιεραρχία με ρίζα τον κόμβο από τον οποίο ξεκίνησε το ερώτημα. Κάθε κόμβος περιμένει το Skyline των παιδιών του για να υπολογίσει το δικό του και ρωτούνται όλοι οι κόμβοι για να υπολογιστεί το συνολικό Skyline. Ο Can-Multicast

λειτουργεί με τον ίδιο τρόπο με τον Naïve με την διαφορά ότι τώρα τα δεδομένα έχουν ανατεθεί στους κόμβους ανάλογα με το περιεχόμενο τους. Έτσι ρωτώνται μόνο οι κόμβοι στους οποίους μπορεί να υπάρχουν δεδομένα που να ανήκουν στο Skyline. Οι μετρήσεις που γίνονται αφορούν α) το ποσοστό των κόμβων που επισκέπτονται σε κάθε ερώτημα, β) των αριθμό των δεδομένων που μεταδίδονται ανά ερώτημα και γ) τον μέσο χρόνο απόκρισης σε κάθε ερώτημα. Ο DSL αλγόριθμος είναι καλύτερος όσον αφορά και τις τρεις μετρήσεις, ενώ ακολουθούν στο β) ο Can-Multicast και έπειτα ο Naïve και στο γ) πρώτα ο Naïve και έπειτα ο Can-Multicast.

Π.2.6. SKYPEER: Efficient Subspace Skyline Computation over Distributed Data

Σε αυτό το άρθρο [VDKV07] προτείνεται ένας αποδοτικός τρόπος υπολογισμού Skyline ερωτημάτων που απευθύνονται σε ένα υποσύνολο των διαστάσεων των δεδομένων σε ένα σύστημα ομότιμων. Υποθέτουν ότι το ερώτημα γίνεται όσον αφορά το MIN και οι τιμές των δεδομένων είναι μη αρνητικές. Όσον αφορά το σύστημα ομότιμων υποθέτουν πως είναι αδόμητο και ότι υπάρχουν Super peers.

Κάθε ομότιμος που συνδέεται στο σύστημα υπολογίζει το τοπικό του Skyline και το extended Skyline του και το στέλνει στον Super peer που του αντιστοιχεί. Το extended Skyline του είναι τέτοιο ώστε για οποιοδήποτε υποσύνολο των διαστάσεων να μπορεί να υπολογιστεί το Skyline (ουσιαστικά περιέχει και δεδομένα τα οποία είναι χειρότερα από αυτά που ανήκουν στο Skyline, είναι όμως ίσα σε τουλάχιστον μία διάσταση). Οι Super peers διατηρούν μία λίστα από τα αυτά δεδομένα, η οποία είναι ταξινομημένη ως προς την f(p) τιμή τους. Το f(p) ορίζεται ως η μικρότερη τιμή που έχει μία πλειάδα σε οποιαδήποτε από τις διαστάσεις της. Επίσης διατηρεί ένα threshold το οποίο είναι η μέγιστη τιμή που εώς τώρα έχει δει να ανήκει σε κάποια διάσταση του Skyline. Αυτή η τιμή συγκρίνεται με τα f(p) και όταν βρει κάποιο μεγαλύτερο από το threshold σταματά να ψάχνει τα δεδομένα (τα υπόλοιπα δεν ανήκουν σίγουρα στο Skyline). Ο Super peer υπολογίζει το τοπικό του Skyline με κάποιον κεντρικοποιημένο αλγόριθμο και στέλνει τα αποτελέσματα σε αυτόν που του έστειλε το ερώτημα.

Όπως μπορούμε να παρατηρήσουμε, τα δεδομένα που μεταφέρονται στο δίκτυο είναι πολύ λιγότερα από ότι θα ήταν σε μία πιο αφελή προσέγγιση. Όσον αφορά όμως τις συγκρίσεις που θα γίνουν για τον υπολογισμό του Skyline παρατηρούμε ότι εξαρτώνται από το πόσο μεγάλο είναι το threshold. Έτσι προσπαθούν να υπολογίσουν ένα αρχικά μικρό threshold. Αυτό που γίνεται είναι πλέον το ερώτημα να προωθείται μαζί με το threshold ως όρισμα. Κάθε Super peer που βρίσκει ένα threshold μικρότερο από αυτό που του στάλθηκε το προωθεί στον επόμενο του έτσι ώστε το Skyline να υπολογιστεί πιο αποδοτικά. Επίσης προτείνουν και κάποιες παραλλαγές ως προς το κατά πόσον κάθε Super peer υπολογίζει τοπικά το Skyline και μετά το στέλνει ή στέλνει τα δεδομένα που μπορεί να ανήκουν και το Skyline υπολογίζεται τοπικά σε αυτόν που έθεσε το ερώτημα.

Στις μετρήσεις που κάνουν συγκρίνουν τις διάφορες παραλλαγές του αλγορίθμου τους, όσον αφορά τον ολικό χρόνο υπολογισμού του Skyline (με ή χωρίς καθυστερήσεις δικτύου), τον όγκο των δεδομένων που μεταφέρονται στο δίκτυο σε σχέση με το πόσες διαστάσεις έχουν τα δεδομένα. Επίσης μετρούν το πώς συμπεριφέρεται ο αλγόριθμος όταν αλλάζει το μέγεθος του δικτύου, όταν μεταβάλλεται ο αριθμός των Super peers, ή όταν μεταβάλλεται ο αριθμός των ομότιμων που είναι συνδεδεμένοι με κάθε Super peer.

Π.2.7. Efficient Skyline Query Processing on Peer-to-Peer Networks

Η εργασία αυτή [WOTX07] αναφέρεται στην επίλυση του προβλήματος της αποδοτικής απάντησης Skyline ερωτημάτων σε συστήματα ομότιμων. Αναφέρει πως τρία είναι τα σημεία τα οποία προσδοκά κάποιος από ένα τέτοιο σύστημα. Πρώτα από όλα να είναι όσο το δυνατό μικρός γίνεται ο αριθμός των κόμβων-ομότιμων που θα εμπλακούν στην απάντηση του ερωτήματος, δεύτερον, να σταλούν όσο το δυνατόν λιγότερα μηνύματα και τρίτον να υπάρχει εξισορρόπηση του φόρτου του ερωτήματος στο σύστημα. Η εργασία τους προσπαθεί και ως ένα σημείο λύνει τα παραπάνω προβλήματα με τους εξής τρόπους: αρχικά χρησιμοποιεί ένα BATON δένδρο ως δομή του συστήματος για αποθήκευση των δεδομένων, έπειτα χωρίζουν το χώρο αναζήτησης του ερωτήματος σε υποπεριοχές, μειώνοντας έτσι τους ομότιμους που εμπλέκονται στην απάντηση και τέλος η εξισορρόπηση του φόρτου γίνεται μόνη της από τους μηχανισμούς εξισορρόπησης φόρτου στο ίδιο το δένδρο κατά την εισαγωγή-διαγραφή κόμβων κτλ.

Αρχικά στην εργασία περιγράφεται συνοπτικά η BATON δομή. Κάθε ομότιμος είναι ένας από τους κόμβους του δένδρου και το σύνολο των δεδομένων κάθε ομότιμου περικλείεται σε ένα ορθογώνιο (hyper-rectangle). Τα ορθογώνια αυτά δεν μπορούν να είναι επικαλυπτόμενα. Επιπλέον κάθε ομότιμος κρατά δείκτες σε κάποιους γείτονες του και πληροφορίες που αφορούν τα δεδομένα του.

Όσον αφορά την εύρεση του Skyline, αυτή ξεκινά από τον κόμβο του οποίου τα τοπικά αποτελέσματα είναι εγγυημένο ότι θα ανήκουν στο αποτέλεσμα. Αυτό μπορεί να βρεθεί μέσω των περιοχών –ορθογωνίων που αντιστοιχούν στους ομότιμους. Από τα τοπικά του αποτελέσματα επιλέγεται το σημείο το έχει το μεγαλύτερο εύρος κυριαρχίας- αποκλείει δηλαδή πολλά σημεία από το να ανήκουν στο Skyline. Σύμφωνα με αυτό το σημείο επιλέγονται οι περιοχές οι οποίες θα ερευνηθούν για σημεία που ανήκουν στο Skyline. Για περαιτέρω αποδοτικότητα κάθε περιοχή χωρίζεται σε υποπεριοχές έτσι ώστε να γίνει παράλληλη σε αυτές η εκτέλεση του ερωτήματος. Το ερώτημα προωθείται στους ομότιμους των οποίων η περιοχή δεν κυριαρχείται από κάποια άλλη ώσπου να υπολογιστεί το αποτέλεσμα.

Τέλος, περιοδικά γίνεται έλεγχος για επιπλέον φόρτο. Αν παρατηρηθεί ότι κάποιος ομότιμος έχει περισσότερο φόρτο από όσο μπορεί να αντέξει, τότε μοιράζει μέρος του φορτίου του στους γειτονικούς του κόμβους.

ΣΥΝΤΟΜΟ ΒΙΟΓΡΑΦΙΚΟ

Η Φωτιάδου Αικατερίνη γεννήθηκε στη Δράμα το 1983. Το 2000 αποφοίτησε από το 2^{ο} Ενιαίο λύκειο Δράμας με βαθμό απολυτηρίου 18,4 και εισήχθη στο τμήμα Πληροφορικής του Πανεπιστημίου Ιωαννίνων. Ολοκλήρωσε τις σπουδές της το 2005 και εισήχθη στο Μεταπτυχιακό τμήμα του ίδιου Πανεπιστημίου όπου ακολούθησε εξειδίκευση «Λογισμικό». Το 2004 συμμετείχε στο πρόγραμμα απόκτησης επαγγελματικής εμπειρίας "Leonardo Da Vinci", με τοποθέτηση στον εκπαιδευτικό οργανισμό BFI του Linz της Αυστρίας.