ΕΘΝΙΚΌ ΜΕΤΣΌΒΙΟ ΠΟΛΥΤΕΧΝΕΊΟ
ΣΧΟΛΉ ΗΛΕΚΤΡΟΛΌΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΕΠΙΚΟΙΝΩΝΙΩΝ, ΗΛΕΚΤΡΟΝΙΚΗΣ ΚΑΙ
ΣΥΣΤΗΜΑΤΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ

Αποδοτικός σε Χώρο και Χρόνο Υπολογισμός των Βαθμών των Κόμβων Γράφων με Χρήση Πιθανοτικών Τεχνικών Data Sketching

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Αντώνης Δ. Μητρόπουλος

Επιβλέπων : Θεοδώρα Βαρβαρίγου
Καθηγήτρια Ε.Μ.Π.

Αθήνα, Μάρτιος 2014

ΕΘΝΙΚΌ ΜΕΤΣΌΒΙΟ ΠΟΛΥΤΕΧΝΕΊΟ
ΣΧΟΛΉ ΗΛΕΚΤΡΟΛΌΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΕΠΙΚΟΙΝΩΝΙΩΝ, ΗΛΕΚΤΡΟΝΙΚΗΣ ΚΑΙ
ΣΥΣΤΗΜΑΤΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ

Αποδοτικός σε Χώρο και Χρόνο Υπολογισμός των Βαθμών των Κόμβων Γράφων με Χρήση Πιθανοτικών Τεχνικών Data Sketching

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Αντώνης Δ. Μητρόπουλος

Επιβλέπων : Θεοδώρα Βαρβαρίγου
Καθηγήτρια Ε.Μ.Π.

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 10η Μαρτίου 2014.

.....	.....	.....
Θ. Βαρβαρίγου	Ε. Καγιάφας	Β. Λούμος
Καθηγήτρια Ε.Μ.Π.	Καθηγητής Ε.Μ.Π.	Καθηγητής Ε.Μ.Π.

Αθήνα, Μάρτιος 2014

.....

Αντώνης Δ. Μητρόπουλος

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

Ευχαριστίες

Με την ευκαιρία της ολοκλήρωσης της διπλωματικής μου εργασίας, θα ήθελα να εκφράσω τις θερμές μου ευχαριστίες προς την κ. Θεοδώρα Βαρβαρίγου, Καθηγήτρια Ε.Μ.Π., για τη δυνατότητα που μου παρείχε να εργαστώ ερευνητικά στον τομέα των Big Data.

Ακόμη, θα ήθελα να ευχαριστήσω ιδιαίτερα το διδάκτορα Κωνσταντίνο Τσερπέ για την επίβλεψη και την πολύτιμη συμβολή του στην εκπόνηση αυτής της διπλωματικής εργασίας. Οι κατευθύνσεις που παρείχε και το διαρκές ενδιαφέρον που έδειχνε υπήρξαν κρίσιμες.

Έπειτα, θα ήθελα να ευχαριστήσω και τα υπόλοιπα μέλη του εργαστηρίου Distributed Knowledge and Media Systems Group του τομέα Επικοινωνιών, Ηλεκτρονικής και Συστημάτων Πληροφορικής της σχολής Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών του Ε.Μ.Π.

Περίληψη

Η έννοια Big Data εμφανίζεται σε ολοένα και περισσότερους κλάδους στις μέρες μας. Η ποσότητα της πληροφορίας που παράγεται καθημερινά αυξάνει συνεχώς και μάλιστα με ρυθμούς ταχύτερους απ' ότι μεγαλώνουν τα διαθέσιμα μέσα αποθήκευσης ή βελτιώνονται οι γνωστές τεχνικές για την ανάλυση και την επεξεργασία τους.

Η ανάγκη λοιπόν για τεχνικές που να επιτυγχάνουν ανάλυση μεγάλων ποσοτήτων δεδομένων σε υψηλές ταχύτητες και καταλαμβάνοντας όσο το δυνατόν λιγότερο χώρο κρίνεται κρίσιμη. Για το σκοπό αυτό έχουν προταθεί διάφορες λύσεις. Μία εξ' αυτών, και αυτή με την οποία θα ασχοληθούμε στην παρούσα εργασία, είναι το Data Sketching. Η τεχνική αυτή επιτρέπει την αποθήκευση και ανάκτηση πληροφορίας με τρόπο πιθανοτικό, απαιτώντας πολύ μικρό χώρο συγκριτικά με πιο "παραδοσιακές" μεθόδους, και μάλιστα επιτυγχάνει κάτι τέτοιο διατηρώντας εξαιρετικά επίπεδα χρονικής πολυπλοκότητας.

Το πρόβλημα που θα χρησιμοποιηθεί ως οδηγός στην προσπάθεια βαθύτερης κατανόησης και στην έρευνα γύρω από το Data Sketching είναι αυτό του υπολογισμού των in-degrees των κόμβων ενός γράφου (κοινωνικού γράφου στη συγκεκριμένη περίπτωση). Για το σκοπό αυτό, επιλέγουμε τη μέθοδο Count Min Sketch η οποία επιτρέπει, για κάθε κόμβο, την αποθήκευση του πλήθους των εισερχόμενων σε αυτόν ακμών, με τρόπο πιθανοτικό, επιτυγχάνοντας υψηλά επίπεδα απόδοσης.

Καθώς η παραπάνω μέθοδος απαιτεί κατάλληλη, σε σχέση με τα χαρακτηριστικά του γράφου στον οποίο εφαρμόζεται, επιλογή των παραμέτρων της προκειμένου να επιστρέψει καλά αποτελέσματα, κρίνεται χρήσιμη η αναζήτηση ενός τρόπου που να μην απαιτεί πρότερη γνώση του γράφου, αλλά να αποτελεί universal λύση. Προτείνεται εδώ ένας αλγόριθμος ο οποίος δεσμεύοντας αρχικά μικρό ποσοστό της μνήμης, αυξάνει δυναμικά το χώρο που καταλαμβάνει ούτως ώστε να διατηρείται η απαραίτητη ακρίβεια των αποτελεσμάτων. Κάτι τέτοιο αυξάνει τις απαιτήσεις σε μνήμη, όμως ο τελικός χώρος που καταλαμβάνεται εξακολουθεί να είναι μικρότερος απ' αυτόν που θα απαιτούνταν από μια ντετερμινιστική δομή δεδομένων, ενώ ο χρόνος και η ακρίβεια προσεγγίζουν τα κλασσικά επίπεδα μιας Count Min υλοποίησης που εφαρμόζεται σε γνωστό γράφο.

Οι κοινωνικοί γράφοι ακολουθούν Power Law κατανομή. Το συγκεκριμένο τους χαρακτηριστικό, όπως θα φανεί και στη συνέχεια, καθιστά ιδιαίτερα υψηλές τις απαιτήσεις σε ακρίβεια του αλγορίθμου. Έτσι, τα αποτελέσματα της παρούσας εργασίας μπορούν, με μικρές αλλαγές, να γενικευτούν και να χρησιμοποιηθούν και για άλλες περιπτώσεις, πλην των κοινωνικών δικτύων, οι οποίες εμφανίζουν παρόμοιες ανάγκες σε ακρίβεια.

Λέξεις Κλειδιά

Count Min Sketch, Data Sketching, Big Data, Κοινωνικός Γράφος, Βαθμοί Κόμβων, Power Law Distribution

Abstract

The notion of Big Data becomes more and more important nowadays. The growth of data production is increasing significantly overcoming the increase in available data storages or the progress in known techniques for their analysis.

As a result, there's huge demand for research in techniques that can achieve efficient handling of such huge data volumes in a way that requests limited space and time recourses. Various solutions have been proposed; one of which is Data Sketching. This technique makes, in a probabilistic manner, the storage and retrieval of information possible in a way that is both space and time efficient in comparison to "traditional" solutions.

In order to better understand and research the idea of Data Sketching, we are going to use, as a guide, the problem of finding a graph's (a social graph is chosen here) in-degrees for each particular node. To achieve this, we have chosen the method Count Min Sketch (a stochastic Data Sketching algorithm) which is ideal for the storage of all incoming edges.

Count Min Sketch requires the best possible, according to the special characteristics of the graph in question, selection of its parameters. So, researching a universal solution which could achieve highly accurate results without the need of any pre-information would be hugely useful. In this thesis, an algorithm is proposed which, starting from allocating a small amount of memory, increases its space dynamically in a way that is preserving the accuracy necessary. The memory demands of such an application are higher but the total space needed remains less than the one of a deterministic database algorithm. The time complexity and the results' accuracy approximate the ones of a classic Count Min implementations on a pre-known large graph.

Social graphs (at least the ones that are used here) tend to follow the Power Law distribution. As explained later, this results in high needs of accuracy. As a result, the conclusions presented here and the algorithm proposed can be used, with small adjustments, in all applications (not only social network ones) with similar accuracy needs.

Keywords

Count Min Sketch, Data Sketching, Big Data, Social Graph, Nodes' Degrees, Power Law Distribution

Περιεχόμενα

Κατάλογος Σχημάτων

Κατάλογος Πινάκων

Κατάλογος Εξισώσεων

Εισαγωγή

Αντικείμενο της Διπλωματικής

Σκοπός της παρούσας διπλωματικής είναι να ερευνηθούν τρόποι με τους οποίους είναι δυνατό να αναλύονται μεγάλες ποσότητες δεδομένων σε περιπτώσεις όπου τα διαθέσιμα rescources είναι περιορισμένα τόσο σε χώρο όσο και σε χρόνο.

Συγκεκριμένα, στόχος είναι ο υπολογισμός των in-degrees όλων των κόμβων ενός γράφου, του οποίου το μέγεθος είναι τέτοιο που "παραδοσιακές" μέθοδοι δεν είναι αρκετές (όπως συμβαίνει για παράδειγμα στα κοινωνικά δίκτυα, με τα οποία θα ασχοληθούμε εδώ). Παράλληλα θα απαντώνται και ερωτήματα για τη συνολλική μορφή του γράφου, όπως "Πόσο είναι το πλήθος nodes με in-degree μεγαλύτερο ή ίσο με X".

Για το σκοπό αυτό, δοκιμάζεται η χρήση τεχνικών Data Sketching. Συγκεκριμένα, για την αποθήκευση των βαθμών των nodes επιλέγεται η μέθοδος Count Min Sketch. Σύμφωνα μ' αυτήν, με κατάλληλη επιλογή των παραμέτρων της καθίσταται εφικτή η αποδοτική, σε χώρο και χρόνο αποθήκευση της ζητούμενης πληροφορίας με τρόπο πιθανοτικό. Στην παρούσα εργασία γίνεται προσπάθεια οι παράμετροι αυτές να επιλέγονται αυτόματα από την εφαρμογή και να αλλάζουν δυναμικά ανάλογα με τις απαιτήσεις που δημιουργούνται σε real-time. Αν κάτι τέτοιο ήταν δυνατό, η υλοποίηση θα μπορούσε να χρησιμοποιείται για να απαντάει σε τέτοια ερωτήματα (πόση είναι η ποσότητα κάποιου συγκεκριμένου στοιχείου ή ποια η συχνότητα εμφάνισης στοιχείων με κάποια προκαθορισμένη τιμή) αποδοτικά, χωρίς να απαιτείται καμία πρότερη γνώση σχετικά με το είδος, το μέγεθος και τα ιδιαίτερα χαρακτηριστικά του data set στο οποίο εφαρμόζεται.

Οργάνωση Κειμένου

Η παρούσα εργασία χωρίζεται σε 3 μέρη. Το πρώτο μέρος αποτελείται από τα κεφάλαια 1, 2, 3, 4, 5, 6 και σ' αυτό παρουσιάζεται το απαραίτητο θεωρητικό υπόβαθρο. Το δεύτερο μέρος περιλαμβάνει τα κεφάλαια 7, 8 στα οποία περιέχεται αναλυτική περιγραφή της υλοποίησης και παρουσίαση των αποτελεσμάτων. Στο τρίτο μέρος παρατίθενται οι βιβλιογραφικές αναφορές και τα παραρτήματα.

Συγκεκριμένα:

1. Στο 1ο κεφάλαιο γίνεται μια γενική εισαγωγή στην έννοια Big Data. Περιγράφονται δημοφιλείς τεχνικές, αναφέρονται παραδείγματα όπου βρίσκουν εφαρμογή και αναλύονται τα κρισιμότερα ζητήματα σχετικά με αυτά. Στο 2ο κεφάλαιο επιλέγεται μία από τις τεχνικές που αναφέρθηκε παραπάνω, συγκεκριμένα το Data Sketching (που αποτελεί και το κεντρικό θέμα της παρούσας διπλωματικής), και περιγράφεται αναλυτικά. Δύο από τα είδη Data Sketching, το Bloom Filter και το Count Min Sketch, αποτελούν το κεντρικό θέμα των κεφαλαίων 3 και 4 αντίστοιχα. Στο κεφάλαιο 5 εξηγείται ο τρόπος με τον οποίο επιλέγονται οι hash functions ανάλογα με τις απαιτήσεις της εφαρμογής και στο κεφάλαιο 6 γίνεται αναλυτική παρουσίαση της Power Law κατανομής.

2. Στο κεφάλαιο 7 γίνεται σύντομη περιγραφή της υλοποίησης που θα χρησιμοποιηθεί ως Benchmark για σύγκριση των αποτελεσμάτων που θα προκύψουν από την Count Min υλοποίηση, της οποίας η κεντρική ιδέα καθώς και οι μετρήσεις που πάρθηκαν και τα

συμπεράσματα αυτών αναλύονται στο κεφάλαιο 8.

3. Στα παραρτήματα παρατίθεται ο κώδικας τόσο της Benchmark όσο και της Count Min υλοποίησης.

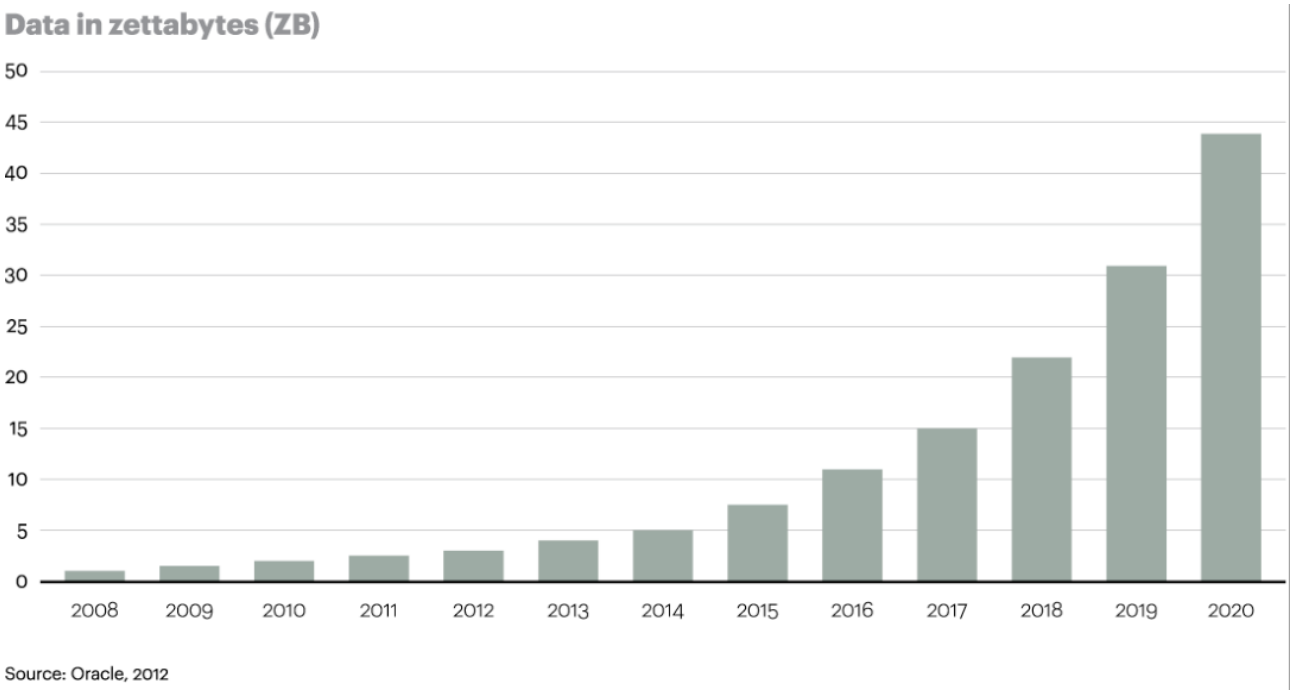
Μέρος 1ο: Θεωρητικό Υπόβαθρο

Κεφάλαιο 1: Big Data

Εισαγωγή

Ο όρος “Big Data” αναφέρεται σε συλλογές δεδομένων των οποίων ο όγκος είναι τέτοιος που καθιστά εξαιρετικά δύσκολη τη συλλογή, επεξεργασία, ανάλυση και αποθήκευσή τους χρησιμοποιώντας παραδοσιακές τεχνικές της επιστήμης της πληροφορικής.

Η ανάπτυξη της τεχνολογίας, σε συνδυασμό με την εισχώρησή της σε ολοένα και μεγαλύτερα τμήματα του πληθυσμού έχει οδηγήσει στη δημιουργία τεράστιων ποσοτήτων δεδομένων σε πληθώρα διαφορετικών περιοχών. Μάλιστα, η συγκεκριμένη αύξηση στα παραγόμενα δεδομένα δείχνει να ακολουθεί εκθετική κατανομή, πράγμα που προβλέπεται να συνεχιστεί τουλάχιστον βραχυπρόθεσμα. Τα τελευταία χρόνια, προβλήματα από το μεγάλο όγκο των δεδομένων που συλλέγονται, εμφανίζονται στους χώρους της πληροφορικής, της μετεωρολογίας, και άλλων επιστημονικών περιοχών, όμως ταυτόχρονα εμφανίζονται και σε λιγότερο αναμενόμενους τομείς, όπως είναι η υγεία, η εκπαίδευση, η οικονομία και οι επιχειρήσεις. Για το λόγο αυτό, οι περιοχές αυτές, αλλά και πολλές άλλες, εμφανίζουν τεράστια περιθώρια ανάπτυξης και βελτίωσης, από την πρόοδο στην έρευνα σχετικά με “Big Data”.



Σχήμα 1: Πρόβλεψη της αύξησης της παραγωγής δεδομένων τα επόμενα χρόνια.

Περιγραφή

Η έννοια “Big Data” δεν είναι απόλυτη και εξαρτάται σε μεγάλο βαθμό από το είδος της υπηρεσίας στην οποία απευθύνονται, από τις ανάγκες και τις δυνατότητες των ανθρώπων που τις

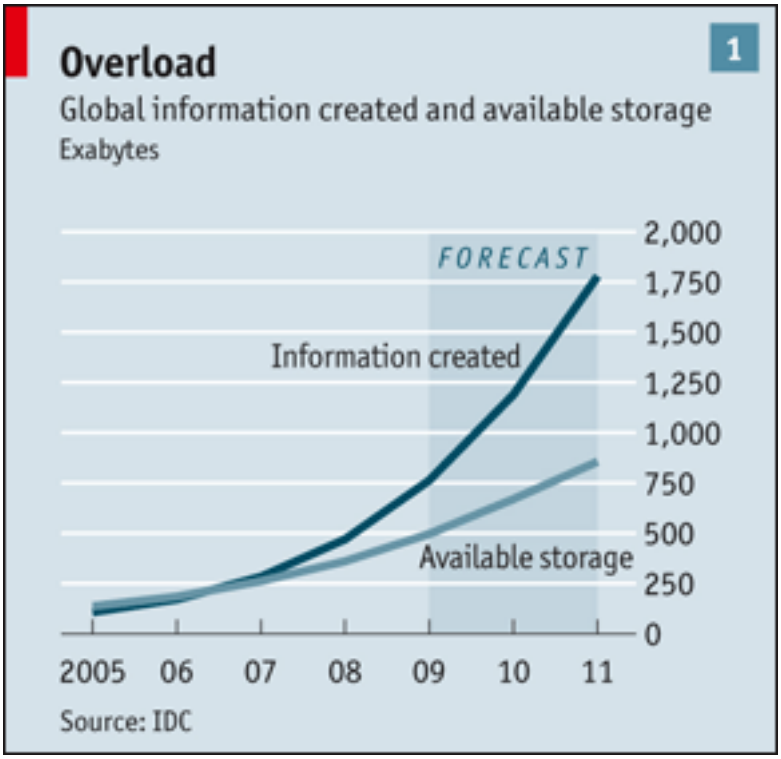
διαχειρίζονται, καθώς και από τις διαθέσιμες υποδομές. Δεν υπάρχει δηλαδή ένα όριο μεγέθους δεδομένων πάνω από το οποίο αποκαλούνται “Big Data”. Παρ’ όλ’ αυτά, υπολογίζεται ότι σήμερα με το συγκεκριμένο όρο αναφερόμαστε συνήθως σε όγκους δεδομένων που κυμαίνονται από μερικά terabytes έως δεκάδες ή και εκατοντάδες petabytes.

Όμως, οι προκλήσεις που προκύπτουν από τα “Big Data” δεν περιορίζονται μόνο στο μεγάλο όγκο των δεδομένων.

Προκλήσεις προκύπτουν κι από την πολύ υψηλή πλέον συχνότητα δημιουργίας νέων δεδομένων, καθώς κι από την ανάγκη για “γρήγορη” επεξεργασία. Οι ροές δεδομένων σε real time συστήματα απαιτούν ταχύτητα τόσο στη συλλογή όσο και στην επεξεργασία τους.

Ένα άλλο θέμα που σχετίζεται με το συγκεκριμένο όρο έχει να κάνει με την ποικιλία στα είδη των δεδομένων που συλλέγονται. Κάτι τέτοιο σημαίνει ότι τα συστήματα επεξεργασίας μεγάλων δεδομένων οφείλουν να είναι σχεδιασμένα με τέτοιον τρόπο ώστε να δέχονται και να είναι ικανά να αναλύσουν πληροφορίες, όχι μόνο από τις μέχρι πρότινως συνηθισμένες πηγές άντλησής τους (όπως είναι π.χ. οι υπολογιστές) αλλά και από νέες πηγές δεδομένων, όπως είναι οι κινητές συσκευές, τα κοινωνικά δίκτυα, τα e-mail, οι οικονομικές συναλλαγές, κ.ά.

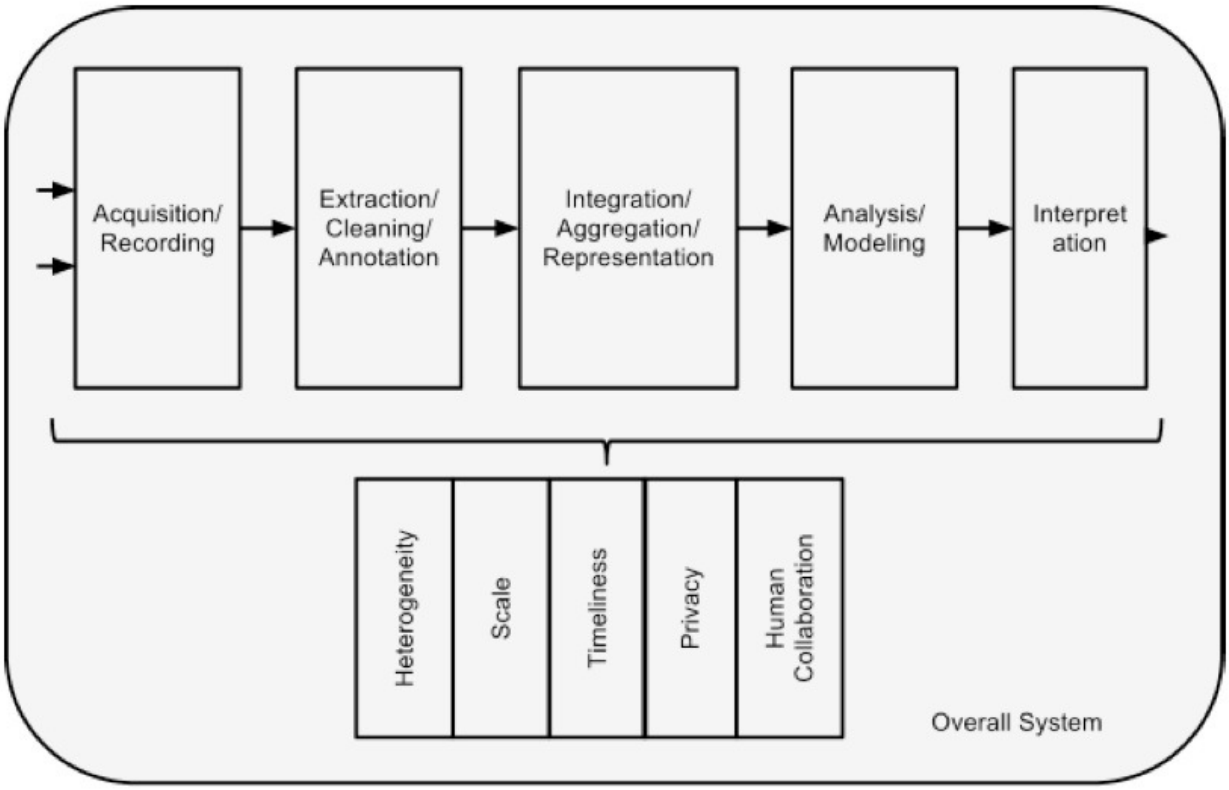
Στα παραπάνω οφείλεται ο όρος “3Vs” που συχνά χρησιμοποιείται στη βιβλιογραφία για να περιγράψει τον όρο “Big Data” και προκύπτει από τις λέξεις Volume, Velocity και Variety.



Σχήμα 2: Διαφορά μεταξύ της ποσότητας πληροφορίας που δημιουργείται και του αντίστοιχου διαθέσιμου χώρου αποθήκευσης

Ανάλυση Big Data και Ανοικτά Ζητήματα

Στο παρακάτω σχήμα, εμφανίζονται συνοπτικά τα διάφορα στάδια μιας “Big Data” ανάλυσης, καθώς επίσης οι τομείς που δημιουργούν τις προκλήσεις και το ερευνητικό ενδιαφέρον γύρω απ’ αυτήν.



Σχήμα 3: Στάδια "Big Data" ανάλυσης και προκλήσεις που προκύπτουν

Αναλυτικά τα θέματα που προκύπτουν στα διάφορα στάδια της ανάλυσης περιγράφονται παρακάτω.

Acquisition / Recording

Το πρώτο βήμα για μια "Big Data" ανάλυση είναι η συλλογή των προς επεξεργασία δεδομένων. Είναι κρίσιμο η συλλογή αυτή να γίνει σωστά, ώστε τα συλλεχθέντα δεδομένα να μπορούν να παράσχουν τα συμπεράσματα που ζητούνται, όμως ταυτόχρονα να μην είναι περισσότερα απ' όσα αρκούν για την εξαγωγή τους.

1. <u>Φιλτράρισμα των προς επεξεργασία δεδομένων.</u>
   Στις περισσότερες περιπτώσεις, είναι δύσκολο να αναγνωρίσουμε την "καθαρότητα" των δεδομένων που λαμβάνουν οι διάφορες πηγές. Είναι λοιπόν απαραίτητο να αναπτυχθούν τεχνικές οι οποίες να μπορούν να κρίνουν πότε η ύπαρξη "εξωπραγματικών" δεδομένων οφείλεται σε πραγματικές καταστάσεις και πότε αποτελούν λανθασμένες μετρήσεις.
   Είναι επίσης συχνό φαινόμενο, να υπάρχει επικάλυψη μεταξύ των δεδομένων που λαμβάνουμε από διαφορετικούς "αισθητήρες". Σ' αυτήν την περίπτωση, είναι σημαντικό κάτι τέτοιο να γίνεται αντιληπτό ώστε να μην αποθηκεύεται παραπάνω πληροφορία, η οποία πιάνει χώρο χωρίς να προσφέρει καινούργια δεδομένα.
   Από τα παραπάνω, κρίνεται επιτακτική η ανάγκη έρευνας σχετικά με το "φιλτράρισμα" των δεδομένων με τέτοιον τρόπο ώστε να μειώνεται σε "φυσιολογικά" επίπεδα ο όγκος τους, χωρίς όμως να χάνεται κρίσιμη πληροφορία. Μάλιστα, καθώς βρισκόμαστε μόλις στο στάδιο της συλλογής των δεδομένων, είναι σημαντικό τα παραπάνω να γίνονται on-the-fly, καθώς η αποθήκευσή τους, έστω και προσωρινή, είναι ακριβώς αυτό που επιδιώκεται να

αποφευχθεί. Πέρα λοιπόν από τη χωρική, μας ενδιαφέρει και η χρονική πολυπλοκότητα των μεθόδων που πρόκειται να προταθούν.

2. Δημιουργία metadata για τα δεδομένα που συλλέγονται.
Καθότι τα δεδομένα που προκύπτουν από την προηγούμενη διαδικασία εξακολουθούν να πιάνουν μεγάλο όγκο, με αποτέλεσμα να καθίσταται δύσκολη η κατανόησή τους, είναι αναγκαίο παράλληλα με τη συλλογή των δεδομένων αυτών καθ' αυτών, να συλλέγεται και πληροφορία που να περιγράφει τα δεδομένα αυτά και τον τρόπο που αυτά συλλέχθησαν ώστε να γίνεται περισσότερο εφικτή η επεξεργασία τους από τον άνθρωπο.
Μάλιστα, καθότι τα αρχικά αυτά δεδομένα, πρόκειται να περάσουν από αρκετά στάδια επεξεργασίας πριν την τελική εξαγωγή συμπερασμάτων, είναι χρήσιμο να υπάρχει τρόπος ώστε οι "περιγραφές" αυτές να διατηρούνται έως το τέλος της ανάλυσης και να μην "χάνονται" μεταξύ των διάφορων βημάτων.

Extraction / Cleaning / Annotation

Στο στάδιο αυτό, έχουμε συλλέξει μία ποσότητα δεδομένων και ενδιαφερόμαστε να την "ετοιμάσουμε" για την ανάλυση που ακολουθεί.

1. Μετατροπή των δεδομένων σε κατάλληλο format.
Τα δεδομένα που συλλέγουμε μπορεί να βρίσκονται σε διάφορες μορφές. Μπορεί να αποτελούν κείμενο, εικόνα, ήχο, βίντεο, κ.ά. Είναι λοιπόν απαραίτητο να υπάρχει μια διαδικασία "εξόρυξης" των απαιτούμενων πληροφοριών από τα παραπάνω δεδομένα και μετατροπή τους σε κατάλληλο για τη μετέπειτα επεξεργασία τους format. Φυσικά, η παραπάνω διαδικασία θα πρέπει να είναι προσαρμοσμένη στις απαιτήσεις της εκάστωτε εφαρμογής, αφού τα σημεία ενδιαφέροντος διαφέρουν ανάλογα με τους σκοπούς που αυτή εξυπηρετεί.

2. Αναγνώριση "θορύβου" στα αρχικά δεδομένα.
Είναι φυσιολογικό, πολλά από τα δεδομένα να περιέχουν παραποιημένη πληροφορία. Αυτό συχνά οφείλεται στον ανθρώπινο παράγοντα που παίζει ρόλο στην παραγωγή των συγκεκριμένων δεδομένων. Συγκεκριμένα, δεν είναι σπάνιο φαινόμενο οι άνθρωποι να παρέχουν ψευδή πληροφόρηση είτε λόγω συνηδειτής επιλογής είτε λόγω αθώου λάθους. Κάτι τέτοιο γίνεται ευκολότερα αντιληπτό αν σκεφτούμε την περίπτωση ασθενών καθώς παρέχουν το ιατρικό ιστορικό τους.
Τέτοιες περιπτώσεις θα πρέπει να λαμβάνονται υπ' όψιν κατά τη διαδικασία της ανάλυσης, είτε αναγνωρίζοντας τέτοιες συμπεριφορές είτε υπολογίζοντας και προσμετρώντας το αντίστοιχο σφάλμα στα τελικά αποτελέσματα.

Integration / Aggregation / Representation

Ένα τελευταίο στάδιο "προετοιμασίας" των δεδομένων απαιτείται πριν την έναρξη της ανάλυσής τους.

1. Ομογενοποίηση δεδομένων από διαφορετικές "συλλογές" για ομαδική επεξεργασία.
Σε περιπτώσεις όπου απαιτείται συνδυασμός περισσότερων από ένα data sets είναι

απαραίτητο, ειδικά όταν αυτά δεν έχουν δημιουργηθεί από την ίδια εφαρμογή, να υπάρχουν επαρκείς πληροφορίες metadata που να τα περιγράφουν, όπως αναφέραμε και προηγουμένως. Όμως αυτό δεν είναι αρκετό. Θα πρέπει τα δεδομένα που προέρχονται από διαφορετικά data sets να τροποποιηθούν κατά τέτοιον τρόπο ώστε να μπορούν να αναλυθούν συνολικά. Και φυσικά, αυτό θα πρέπει να μπορεί να γίνει αυτοματοποιημένα.

2. Σχεδίαση της βάσης δεδομένων.
Ακόμη και σε περιπτώσεις όπου έχουμε μόνο ένα data set, αλλά φυσικά όχι μόνον τότε, είναι σημαντική η επιλογή της μορφής της βάσης δεδομένων που θα χρησιμοποιηθεί. Η διαδικασία σχεδίασης της βάσης είναι κρίσιμη. Κάθε βάση έχει θετικά κι αρνητικά, όμως ανάλογα με τους σκοπούς της εφαρμογής και τους στόχους της ανάλυσης των δεδομένων η επιλογή λάθος βάσης μπορεί να αποβεί καταστροφική για το σύστημα.
Σύμφωνα με τα παραπάνω, προκύπτει ότι θα πρέπει να υπάρξει έρευνα σχετικά με τη σχεδίαση βάσεων δεδομένων, καθώς επίσης είναι αναγκαία η ανάπτυξη τεχνικών που να εκμεταλλεύονται όσο το δυνατόν αποδοτικότερα βάσεις δεδομένων που δεν έχουν σχεδιαστεί ιδανικά.

Analysis / Modelling

Τις περισσότερες φορές, ο όγκος των δεδομένων “Big Data” οδηγεί σε σημαντικές ανακρίβειες στην ποιότητα της πληροφορίας. Παρ’ όλ’ αυτά η ανάλυσή τους θεωρείται επιτακτική, καθώς σε αντίθεση με την, πράγματι ακριβέστερη, ανάλυση μικρότερων δειγμάτων, στην περίπτωση των “Big Data” το πλήθος της πληροφορίας είναι τέτοιο, που παρά το “θόρυβο” που περιέχει, επιτρέπει τον εντοπισμό μοτίβων και βοηθά σημαντικά στο σχηματισμό και την κατανόηση της “μεγάλης εικόνας”.

1. Ανάπτυξη αποδοτικών τεχνικών για πραγματοποίηση queries.
Έντονο ερευνητικό ενδιαφέρον υπάρχει στο “scale up” υπάρχοντων τεχνικών για απάντηση σε δημοφιλή queries, ώστε να υποστηρίζουν τα ίδια queries σε συλλογές δεδομένων όμως μεγέθους πολλών terabytes, με τρόπο που να τις καθιστά αποδοτικές για εφαρμογές που απαιτούν real-time αποκρίσεις. Τέτοιες τεχνικές θα πρέπει να μπορούν να υλοποιήσουν recommendation systems, να υπολογίζουν τις δημοφιλέστερες αναζητήσεις, αλλά και να ελέγχουν την αξία ενός data set ώστε να λαμβάνεται επιτόπου απόφαση σχετικά με την ανάγκη ή μη αποθήκευσής του.

2. Ανανέωση της SQL.
Η σημερινή έκδοση της SQL κρίνεται αναποδοτική για εφαρμογή queries πάνω σε δεδομένα μεγάλου όγκου. Το αποτέλεσμα είναι οι περισσότερες εφαρμογές σήμερα να υποχρεούνται κάθε φορά που πρόκειται να πραγματοποιήσουν κάποια ανάλυση, να “τραβάνε” τα απαιτούμενα δεδομένα από τη βάση τους, να τα επεξεργάζονται και στη συνέχεια να τα επιστρέφουν σε αυτήν. Φυσικά, η συγκεκριμένη διαδικασία είναι ιδιαίτερα κοστοβόρα.
Η ανανέωση λοιπόν των αλγορίθμων που χρησιμοποιεί η SQL θεωρείται επιτακτική ώστε μέρος της δουλειάς κατά την ανάλυση δεδομένων “Big Data” να μπορεί να γίνεται από την SQL με τα δεδομένα μέσα στη βάση, χωρίς να χρειάζεται να εξάγονται απ’ αυτήν.

Interpretation

Η παρουσίαση των αποτελεσμάτων αφού έχει προηγηθεί η ανάλυση των δεδομένων δεν μπορεί να είναι μια απλή διαδικασία. Τα αποτελέσματα δίνουν λίγες πληροφορίες από μόνα τους. Για να είναι ουσιώδη χρειάζεται να παρέχουν στο χρήστη λεπτομερή και κατανοητή περιγραφή του τι σημαίνουν, πώς προέκυψαν, κλπ.

1. <u>Λεπτομερής επεξήγηση της διαδικασίας εξαγωγής των συμπερασμάτων.</u>
   Είναι σημαντικό μαζί με τα αποτελέσματα να παρέχεται και μια περιγραφή του τι σημαίνουν, όμως ακόμη κι αυτό δεν είναι αρκετό. Είναι ανάγκη να δίνεται στο τέλος η πληροφορία του τι παραδοχές έγιναν στα διάφορα στάδια της ανάλυσης προκειμένου να εξαχθούν τα συγκεκριμένα συμπεράσματα. Ακόμη, είναι χρήσιμο να υπάρχει στο τέλος μια επεξήγηση των διάφορων σταδίων της ανάλυσης αυτής καθ' αυτής σε όρους όχι τεχνικούς ώστε να γίνονται εύκολα κατανοητοί από ένα μη εξειδικευμένο χρήστη, ο οποίος να μπορεί εν συνεχεία να κάνει μικροαλλαγές στην πορεία της ανάλυσης. Έτσι ο χρήστης αντιλαμβάνεται καλύτερα τα αποτελέσματα, ενώ ταυτόχρονα αν του δίνεται η δυνατότητα από το σύστημα, μπορεί αλλάζοντας κάποιες παραδοχές ή προσαρμόζοντας καλύτερα κάποιες παραμέτρους να βελιτώνει τα αποτελέσματα ή να δοκιμάζει άλλους τρόπους επεξεργασίας των δεδομένων.

2. <u>Ανάπτυξη εξειδικευμένων εργαλείων οπτικής αναπαράστασης των αποτελεσμάτων.</u>
   Σε αντίθεση με τα πιο "παραδοσιακά" συστήματα ανάλυσης δεδομένων, όπου τα αποτελέσματα εξάγονταν υπό τη μορφή πινάκων, στα "Big Data" συστήματα κάτι τέτοιο δεν είναι εύχρηστο. ο λόγος είναι ότι το μέγεθος και η πολυπλοκότητα των συμπερασμάταν είναι τέτοιο που ενδεχόμενη παρουσίασή τους σ' έναν τεράστιο πίνακα με άπειρους αριθμούς θα εκμηδένιζε κάθε αξία της ανάλυσης.
   Για το λόγο αυτό, είναι σημαντικό να αναπτυχθούν εργαλεία που να μπορούν να αναπαριστούν τα αποτελέσματα της ανάλυσης με τρόπο κατανοητό και οπτικά ορθό.

Αφού περιγράψαμε αναλυτικά τα διάφορα στάδια της ανάλυσης "Big Data" και τα πεδία ερευνητικού ενδιαφέροντος που ενυπάρχουν σε αυτά, στη συνέχεια θα αναφέρουμε συνοπτικά τα σημαντικότερα ζητήματα που εκκρεμούν συνολικά για τη βελτίωση των παραπάνω διαδικασιών.

Heterogeneity

Όπως αναφέρθηκε και παραπάνω, είναι πιθανό τα δεδομένα να μη βρίσκονται όλα σε μορφή σωστά δομημένη ώστε να είναι αξιοποιήσιμα από υπολογιστή. Ακόμη είναι δυνατόν κάποια από τα πεδία των δεδομένων που συλλέγουμε να είναι κενά.

Για περιπτώσεις σαν τις παραπάνω, κρίνεται επιτακτική η ανάγκη για έρευνα γύρω από τεχνικές "ομογενοποίησης" της μορφής αλλά και της πληροφορίας που περιέχεται στα συλλεχθέντα δεδομένα.

Scale

Ένα δεύτερο πεδίο έρευνας, που αναφέρθηκε και παραπάνω, είναι η ανάγκη για "βελτίωση" των υπάρχοντων τεχνικών ανάλυσης δεδομένων ώστε να μπορούν να υποστηρίζουν αντίστοιχη ανάλυση σε μεγάλα data sets.

Ο λόγος που το παραπάνω κρίνεται απαραίτητο είναι ότι, σε αντίθεση με τα προηγούμενα χρόνια όπου η αύξηση του όγκου των προς επεξεργασία δεδομένων αντισταθμιζόταν από τη βελτίωση του hardware του υπολογιστή, πλέον η αύξηση αυτή έχει ξεπεράσει την αντίστοιχη βελτίωση των resources. Συνεπώς, θα πρέπει να βρεθούν τρόποι ώστε να επιτευχθεί η επεξεργασία των δεδομένων στα νέα επίπεδα μεγέθους εφευρίσκοντας εναλλακτικές τεχνικές.

Κάτι τέτοιο μπορεί να γίνει είτε βελτιώνοντας τους αλγορίθμους που χρησιμοποιούντας έως σήμερα είτε παραλληλοποιώντας όσο είναι δυνατόν τις διαδικασίες ώστε να τρέχουν ταυτόχρονα σε πολλούς πυρήνες ή ακόμη σε πολλούς υπολογιστές (clusters).

Timeliness

Ο βασικός γνώμονας όσων ασχολούνται με έρευνα στην περιοχή των "Big Data" είναι να περιοριστεί ο χώρος που καταλαμβάνουν τα δεδομένα. Οι προσπάθειες που γίνονται έχουν να κάνουν κυρίως με το να μειωθεί ο όγκος των δεδομένων ώστε να αναλυθούν πιο αποτελεσματικά.

Παρ' όλ' αυτά, συνήθως η μείωση του χώρου που καταλαμβάνεται οδηγεί σε αύξηση της χρονικής πολυπλοκότητας του αλγορίθμου. Κάτι τέτοιο όμως δεν είναι αποδεκτό. Ειδικά όταν έχουμε να κάνουμε με εφαρμογές όπου είναι αναγκαία η απόκριση σε real time.

Συνεπώς, είναι σημαντικό να αναπτυχθούν μέθοδοι που να επιτυγχάνουν τη μείωση του όγκου, χωρίς όμως να αυξάνουν το χρόνο που απαιτείται για την ανάλυση και την εξαγωγή συμπερασμάτων.

Privacy

Ένα σημαντικό ζήτημα έχει να κάνει με την ιδιωτικότητα των δεδομένων που συλλέγονται. Ειδικά στην περίπτωση όπου κάποιο σύστημα λαμβάνει και αναλύει δεδομένα από data sets που προέρχονται από διαφορετικές πηγές, είναι αναγκαίο να βρεθεί τρόπος ώστε να τηρούνται όλες οι πολιτικές απορρήτου των επιμέρους πηγών.

Σε μια εποχή όπου οι άνθρωποι τείνουν να "μοιράζονται" ολοένα και περισσότερες λεπτομέρειες από τη καθημερινότητά τους, είναι κρίσιμο να γνωρίζουν με ποιον τρόπο αυτές αναλύονται από το διαχειριστή της εφαρμογής, καθώς και με ποιους άλλους τις μοιράζεται εκείνος με τη σειρά του.

Human Collaboration

Μεγάλο μέρος της έρευνας γύρω από τα "Big Data" έχει να κάνει με την αυτοματοποίηση των διάφορων σταδίων της ανάλυσης των δεδομένων. Όμως, δεν πρέπει να παραβλέπεται η συνεισφορά που θα μπορούσε να έχει ο ανθρώπινος παράγοντας στη διαδικασία.

Για το σκοπό αυτό είναι σημαντική η ορθή οπτική αναπαράσταση των διάφορων συμπερασμάτων, όπως αναφέρθηκε και παραπάνω. Μόνον έτσι είναι δυνατόν τα αποτελέσματα να γίνουν κατανοητά από τον άνθρωπο και στη συνέχεια να μπορέσει αυτός να επηρεάσει τη

διαδικασία, να εντοπίσει πιθανά σφάλματα και να την προσαρμόσει στις ειδικές συνθήκες της εκάστωτε περίπτωσης.

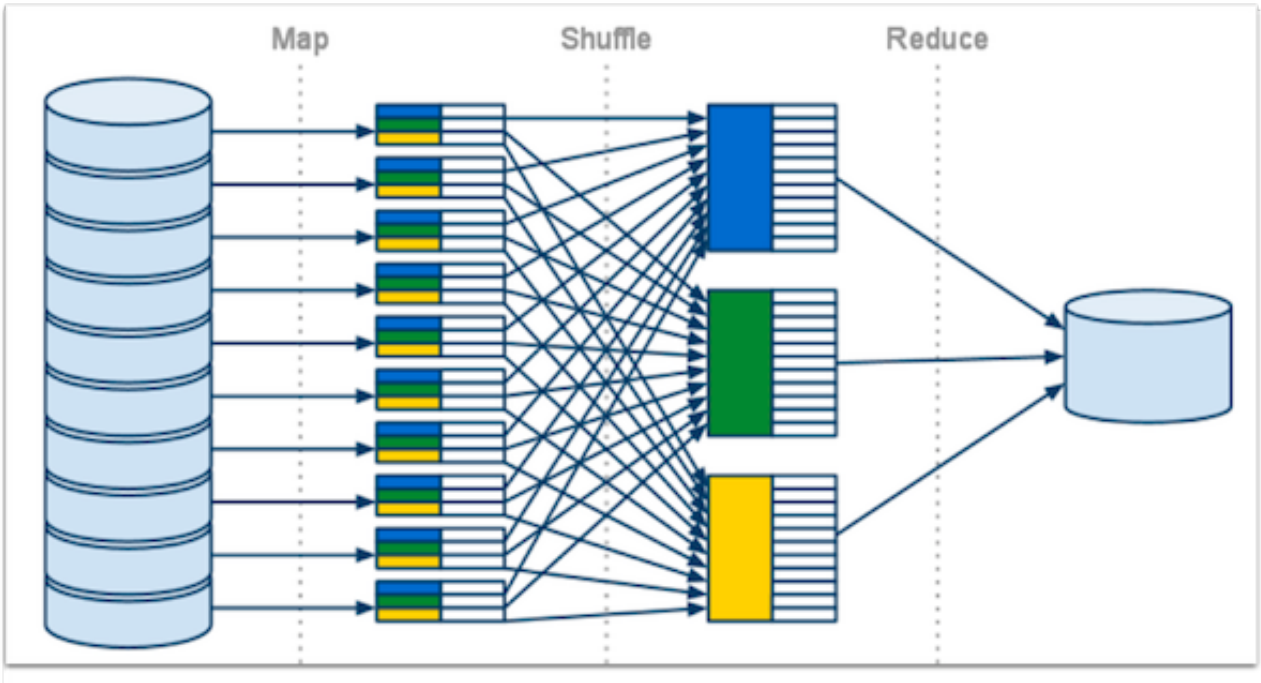
Ακόμη, είναι σημαντικό να μπορούν να αναγνωριστούν, είτε από το σύστημα αυτόματα είτε από ανθρώπινη παρέμβαση, τυχόν ανακρίβειες στα αρχικά δεδομένα που μπορεί να οφείλονται σε διαφορετικά συμφέροντα που τυχόν είχαν αυτοί που τα παρείχαν.

Γενικά, είναι κρίσιμο να βρεθεί τρόπος ώστε να λαμβάνεται υπ' όψιν ο ανθρώπινος παράγοντας τόσο κατά τη συλλογή των δεδομένων όσο και κατά την επιλογή της πορείας της ανάλυσης.

Τεχνικές Big Data

Οι προκλήσεις που περιγράφηκαν παραπάνω έχουν οδηγήσει πολλούς οργανισμούς και εταιρείες να ασχοληθούν σοβαρά με το θέμα και να προτείνουν λύσεις.

Μοντέλα MapReduce και Hadoop



Σχήμα 4: Στάδια MapReduce model

Το 2004, η Google παρουσίασε ένα μοντέλο για την επεξεργασία δεδομένων μεγάλου όγκου, το MapReduce. Το εν λόγω μοντέλο χρησιμοποιεί έναν αλγόριθμο ο οποίος παραλληλοποιεί και κατανέμει το συνολικό προς επεξεργασία όγκο, μοιράζοντας τμήματά του προς εξυπηρέτηση σε πολλαπλούς υπολογιστές. Η ονομασία του δεν είναι τυχαία. Το μοντέλο αρχικά κατανέμει το συνολικό όγκο των εργασιών σε πολλαπλούς υπολογιστές, οι οποίοι εκτελούν τα τμήματα που τους ανατίθενται παράλληλα (στάδιο map) κι εν συνεχεία τα διάφορα αποτελέσματα συλλέγονται και αναλύονται συνολικά πριν επιστραφούν (στάδιο reduce).

Το παραπάνω μοντέλο παρουσίασε μεγάλη επιτυχία και οδήγησε στην ανάπτυξη του open source framework Apache Hadoop, το οποίο χρησιμοποιείται σήμερα από πολλούς οργανισμούς κι επιχειρήσεις.

Data Sketching

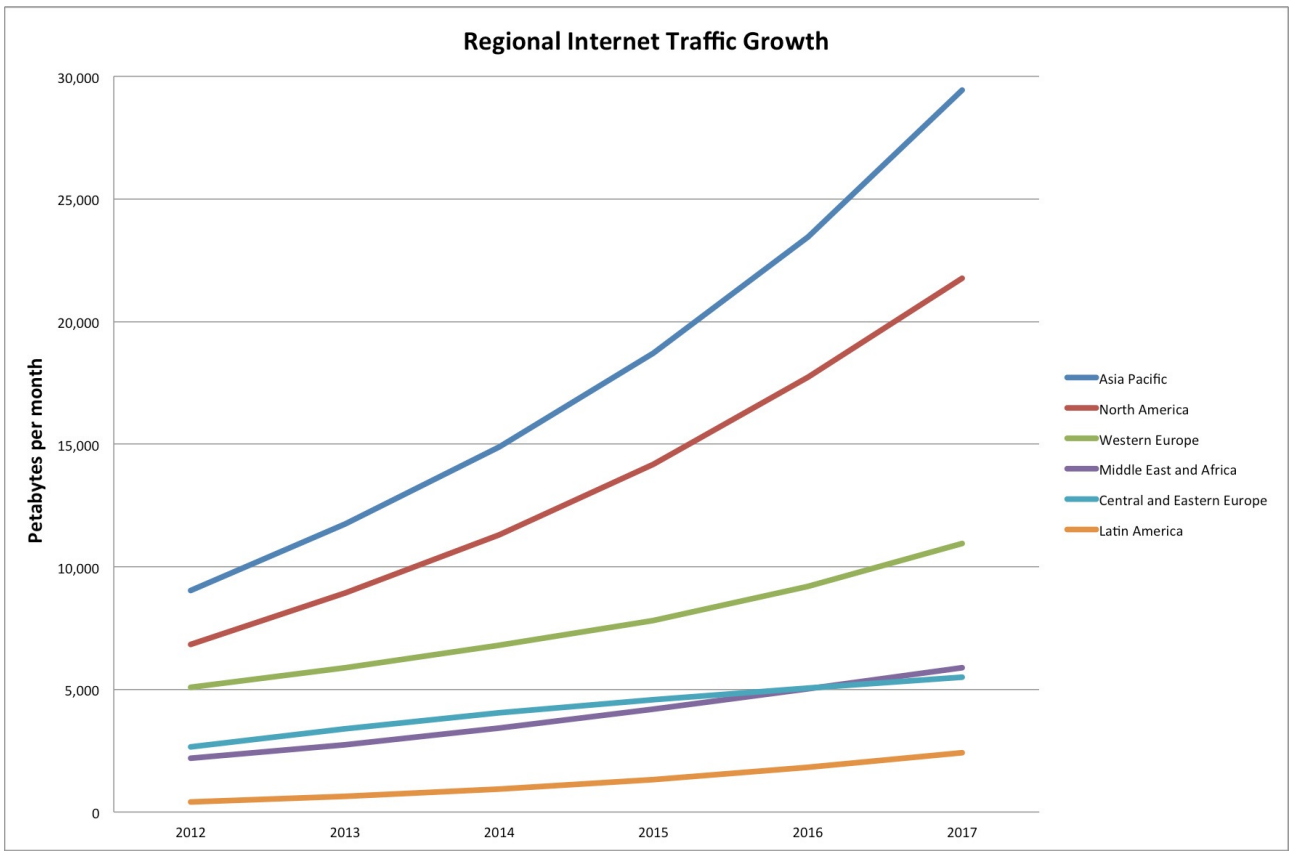
Πολλές εφαρμογές, ειδικά στο διαδίκτυο αλλά όχι μόνον εκεί, εμφανίζουν την ανάγκη για on-the-go ανάλυση δεδομένων, από δεδομένα που συλλέγουν real time και τα οποία σε πολλές περιπτώσεις δεν είναι καν δυνατόν να αποθηκευτούν για μετέπειτα επεξεργασία. Αυτό σημαίνει ότι οι εφαρμογές αυτές χρειάζεται να αναπτύξουν μεθόδους ώστε να μπορούν με ένα ή έστω "λίγα" περάσματα να αναλύουν την πληροφορία που φτάνει σε μορφή ροής δεδομένων (data stream).

Σε τέτοιες περιπτώσεις συνηθίζεται να χρησιμοποιούνται τεχνικές data sketching, οι οποίες επιδιώκουν να αποθηκεύουν μόνον την άκρως απαραίτητη πληροφορία που προκύπτει από τις συγκεκριμένες ροές. Τέτοιες τεχνικές έχουν αναπτυχθεί και αποθηκεύοντας ένα μέρος μόλις της πληροφορίας που λαμβάνεται από την εφαρμογή καταφέρνουν, με πιθανοτικό βέβαια τρόπο, να απαντάνε σε ερωτήματα όπου το σύνολο της πληροφορίας θα ήταν απαραίτητο αν χρησιμοποιούνταν τα "παραδοσιακά" μέσα.

Παραδείγματα Big Data

Η ανάγκη για περαιτέρω έρευνα και βελτίωση των υπάρχοντων τεχνικών στα παραπάνω θέματα φαίνεται στα επόμενα παραδείγματα.

1. Το eBay διαθέτει σχεδόν 90 petabytes δεδομένων, 40 από τα οποία είναι αποθηκευμένα σε Hadoop clusters, ενώ τα υπόλοιπα είναι αποθηκευμένα σε άλλες βάσεις δεδομένων.
2. Το Facebook έχει αποθηκευμένες περισσότερες από 50 δισεκατομμύρια φωτογραφίες χρηστών του, ενώ στο data center του είναι αποθηκευμένα πάνω από 300 petabytes δεδομένων, τα περισσότερα από τα οποία είναι επίσης σε Hadoop clusters.
3. Η μηχανή αναζήτησης της Google πραγματοποιεί περί τις 100 δισεκατομμύρια αναζητήσεις μηνιαίως ή 1.2 τρισεκατομμύρια αναζητήσεις το χρόνο, οι οποίες αποθηκεύονται όλες στα data centers της εταιρείας και αναλύονται για την εξαγωγή χρήσιμων πληροφοριών.



Σχήμα 5: Πρόβλεψη του όγκου δεδομένων που θα διακινούνται στο διαδίκτυο τα επόμενα έτη

Κεφάλαιο 2: Data Sketching

Εισαγωγή

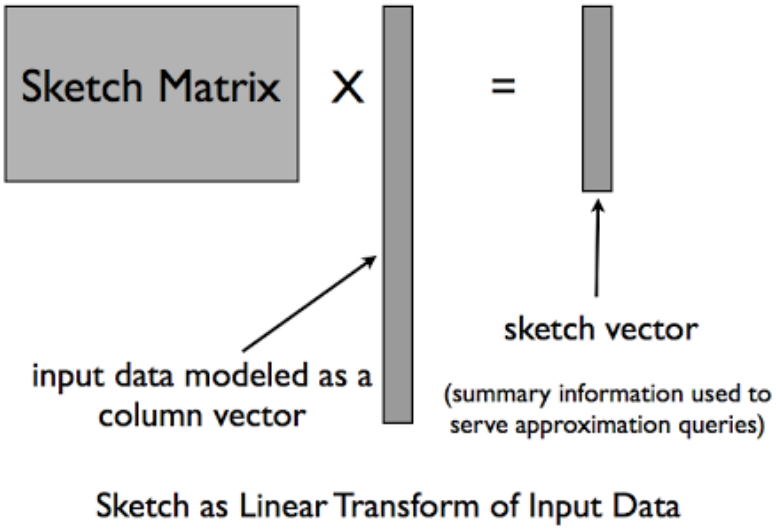
Η ανάγκη αποθήκευσης και ανάλυσης μεγάλων ποσοτήτων δεδομένων σε μικρό χώρο και χρόνο είναι δύσκολο να πραγματοποιηθεί με τις "παραδοσιακές" τεχνικές. Σε εφαρμογές που απαιτούν ανάλυση δεδομένων που έρχονται σε ροή (data streams), η αποθήκευσή τους, ακόμη κι αν αυτή γινόταν άμεσα θα απαιτούσε χώρο της τάξης του O(n) (n: το πλήθος των δεδομένων). Σε πρώτη σκέψη, το O(n) φαντάζει μια καλή χωρική πολυπλοκότητα, όμως στην πραγματικότητα, όταν έχουμε να κάνουμε με εφαρμογές μεγάλου όγκου δεδομένων, ακόμη και η γραμμική πολυπλοκότητα δεν επαρκεί για αποδοτικές υλοποιήσεις. Ο χώρος που καταλαμβάνουν τα δεδομένα δεν είναι όμως το μοναδικό πρόβλημα. Σε τόσο μεγάλα n, ο χρόνος που απαιτείται για την πραγματοποίηση queries είναι επίσης σημαντικός. Κατάλληλη μείωση του χώρου που καταλαμβάνουν τα δεδομένα θα μπορούσε να καταστήσει το μέγεθός τους κατάλληλο για μεταφορά των δεδομένων στην cache κατά τη διάρκεια της ανάλυσης, γλιτώνοντας έτσι χρονοβόρες μεταφορές από και προς τη βάση δεδομένων. Ακόμη, περιπτώσεις όπου απαιτούνται υπολογισμοί μεγεθών on-the-fly, από δεδομένα που καταφτάνουν σε real-time, καθιστούν ακόμη εντονότερη την ανάγκη για αποδοτικούς χρονικά αλγορίθμους.

Δημιουργούνται λοιπόν προβλήματα τόσο ως προς το χώρο όσο και ως προς το χρόνο της ανάλυσης. Τα παραπάνω, πέρα από αναποδοτικές εφαρμογές, οδηγούν και σε σημαντικές αυξήσεις στα κόστη του συστήματος. Ο λόγος είναι ότι η χρέωση στα, πολύ διαδεδομένα στις μέρες μας, cloud based infrastructures υπολογίζεται συνήθως με βάση ακριβώς αυτά τα μεγέθη (το χρόνο και το χώρο που καταλαμβάνουν τα δεδομένα).

Μια λύση που αρχικά προτάθηκε, είναι το random sampling. Σύμφωνα μ' αυτήν, δεν είναι ανάγκη να λαμβάνουμε υπ' όψιν όλα τα δεδομένα που καταφτάνουν, όμως αρκεί να εφαρμόζουμε την ανάλυσή μας πάνω σε κατάλληλο δείγμα. Ο τρόπος επιλογής των δεδομένων που απορρίπτονται ποικίλλει ανάλογα με τις ανάγκες της εφαρμογής.

Μια άλλη τεχνική, που είναι μάλιστα αρκετά διαδεδομένη σήμερα, είναι το data sketching. Η ιδέα της είναι, αντί να αποθηκεύεται το σύνολο των προς ανάλυση δεδομένων, είναι δυνατόν, ανάλογα με τις ανάγκες της εφαρμογής, να αποθηκεύεται κατάλληλη περίληψη αυτών.

Η παραπάνω τεχνική αποτελεί στην ουσία ένα μετασχηματισμό των συλλεχθέντων δεδομένων σε κατάλληλη περίληψή τους. Ο πίνακας μετασχηματισμού, μπορεί να διαφέρει από εφαρμογή σε εφαρμογή, όμως το γενικό σχήμα της τεχνικής, θεωρώντας τα δεδομένα ως διανύσματα, φαίνεται στο παρακάτω σχήμα.

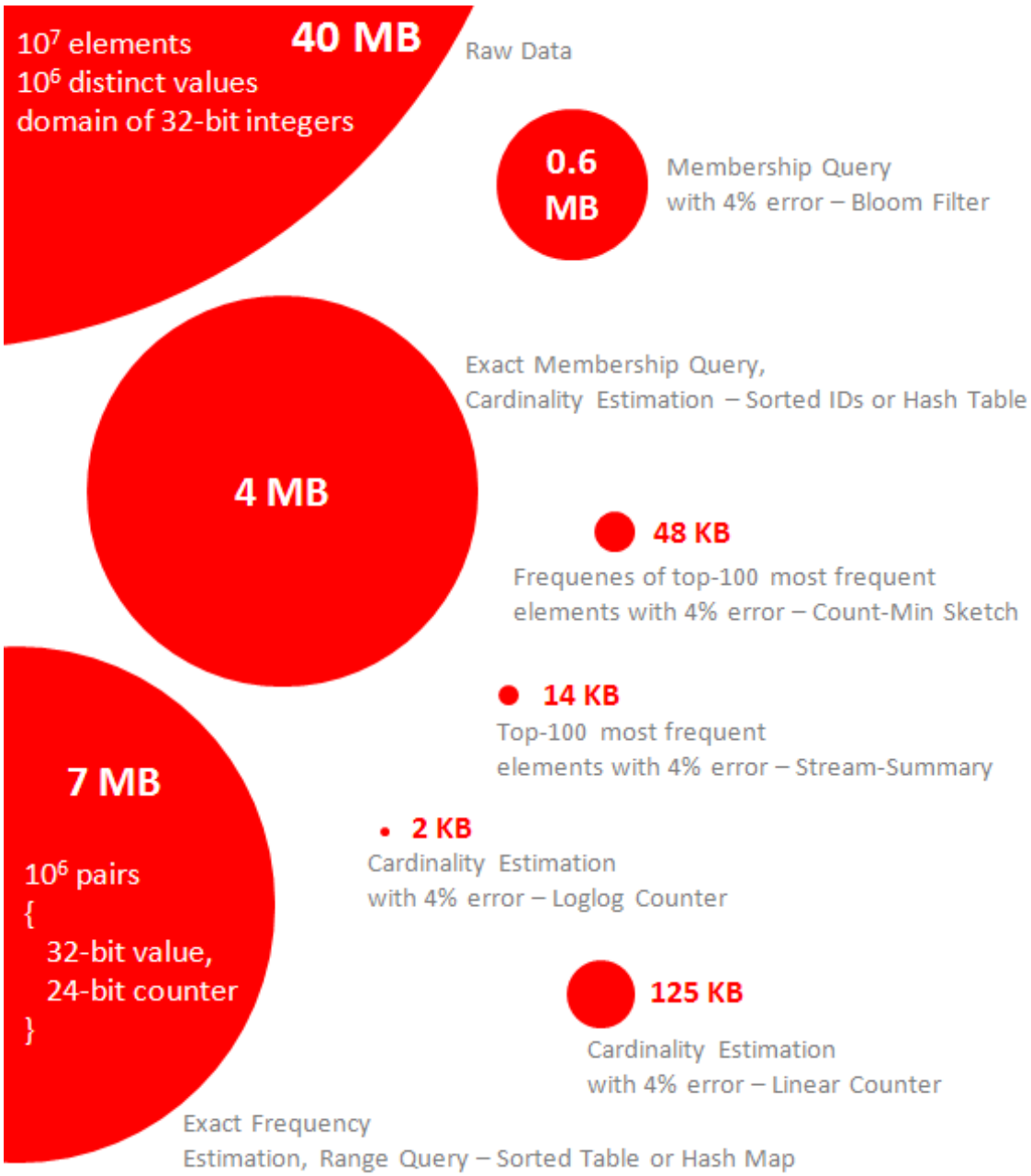


Σχήμα 6: Οπτική αναπαράσταση της μεθόδου data sketching

Περιγραφή Data Sketching

Η μέθοδος data sketching είναι παρόμοια με την τεχνική των hash structs και άλλων παραπλήσιων δέντρων, υπό την έννοια ότι διευκολύνουν σημαντικά την πρόσβαση στα δεδομένα. Παρ' όλ' αυτά υπάρχει μία σημαντική διαφορά. Το data sketching, σε αντίθεση με τις παραπάνω δομές, δεν αποθηκεύει το σύνολο της πληροφορίας. Αυτό που αποθηκεύει είναι, όπως τονίστηκε παραπάνω, μία περίληψη των δεδομένων κατάλληλη για την αποδοτική, σε χρόνο και χώρο, πραγματοποίηση πολύ συγκεκριμένω queries.

Φυσικά, υπάρχουν διαφόρων ειδών τεχνικές που στηρίζονται στην παραπάνω ιδέα. Οι σημαντικότερες εξ' αυτών θα αναφερθούν παρακάτω. Αν και διαφέρουν μεταξύ τους, όλες βασίζονται σε κάποιο είδος πιθανοτικού αλγορίθμου. Αυτό σημαίνει ότι, όπως ίσως ήταν αναμενόμενο, υπάρχει trade-off στις βελτιώσεις που το data sketching προσφέρει. Αυτό είναι ότι λόγω του πιθανοτικού χαρακτήρα του, η ακρίβεια των αποτελεσμάτων δεν είναι απόλυτη όπως συμβαίνει για παράδειγμα στις άλλες πιο χρονοβόρες και χωρικά πολύπλοκες δομές δεδομένων. Η περίληψη της πληροφορίας περιέχει σφάλμα. Σε εφαρμογές όμως όπου αυτό που ενδιαφέρει δεν είναι η ακρίβεια των δεδομένων αλλά η αίσθηση της τάξης μεγέθους τους, το παραπάνω "πρόβλημα" μπορεί να παραβλεφθεί.



Σχήμα 7: Σύγκριση χώρου που καταλαμβάνουν "παραδοσιακές" μέθοδοι σε σχέση με data sketching τεχνικές

Μάλιστα, ένα σημαντικό χαρακτηριστικό της μεθόδου είναι ότι το σφάλμα που προκύπτει μπορεί να προβλεφθεί συναρτήσει του διαθέσιμου χώρου και χρόνου του συστήματος. Όπως είναι φυσικό, στο όριο όπου η υποδομή του συστήματος παρέχει "άπειρο" χώρο, το σφάλμα τείνει στο μηδέν, η πληροφορία αποθηκεύεται αυτούσια και όσχι σε περίληψη και το data sketching μοιάζει πλέον αισθητά με κάποιας μορφής hash struct. Ακόμη όμως και στην περίπτωση μεγάλης ανακρίβειας, η θεωρία, όπως θα αναλυθεί και στη συνέχεια, εξασφαλίζει την ύπαρξη μόνο false positive "προβλημάτων".

Ένα ακόμη ιδιαίτερα σημαντικό χαρακτηριστικό της μεθόδου είναι ότι είναι εύκολα παραλληλοποιήσιμη. Κάτι τέτοιο είναι πολύ σημαντικό αν αναλογιστεί κανείς ότι οι υποδομές των μεγάλων συστημάτων στις μέρες μας αποτελούνται συνήθως από clusters υπολογιστών και άρα το παραπάνω στοιχείο επιτρέπει την καλύτερη εκμετάλλευση των δυνατοτήτων που παρέχονται.

Το data sketching παρουσιάζει σημαντική ανάπτυξη και προσελκύει έντονο ερευνητικό ενδιαφέρον τα τελευταία χρόνια κυρίως λόγω των παρακάτω στοιχείων.

1. Ανεκμετάλλευτες δυνατότητες
   Η έρευνα γύρω απ' τη συγκεκριμένη τεχνική ξεκίνησε μόλις πριν από μερικά χρόνια. Ως αποτέλεσμα, θεωρείται ότι απέχουμε ακόμη αρκετά από το να επέλθει κορεσμός γύρω απ' αυτήν. Αντίθετα, οι προοπτικές που προκύπτουν απ' τα σημαντικά πλεονεκτήματά της και οι δυαντότητες που φαίνεται να δημιουργούνται είναι αμέτρητες και ανατίθεται στην επιστημονική κοινότητα να εξερευντήσει τα όρι ατης μεθόδου και τις βελτιώσεις που θα μπορούσαν να γίνουν σε αυτήν.

2. Ευρήτητα και ωριμότητα
   Αν και νέα, η τεχνική του data sketching έχει ωριμάσει και διευρυνθεί αρκετά ώστε να έχουμε φτάσει πλέον στο σημείο να μπορούμε να την "εμπιστευτούμε" και να την χρησιμοποιήσουμε σε ποικίλες εφαρμογές και συστήματα Big Data Management. Η τεχνική είναι ήδη ικανοποιητικά λειτουργική. Αρκεί να βρεθούν περιοχές όπου να μπορεί να συνεισφέρει και να αποφέρει ρηξικέλευθα αποτελέσματα.

3. Ανάγκη για διαχείριση μεγάλων δεδομένων σε υψηλές ταχύτητες
   Η τάχυστη αύξηση στον όγκο των προς επεξεργασία δεδομένων δημιουργεί την ανάγκη για ολοένα και αποδοτικότερες τεχνικές. Συνεπώς, οι εφαρμογές που έχουν ανάγκη από μεθόδους sketching πληθαίνουν συνεχώς και άρα η σπουδαιότητα της μεθόδου αυξάνει.

4. Cloud computing
   Η υιοθέτηση από ποικίλες υπηρεσίες υποδομών Cloud, σε συνδυασμό με την, εξαρτώμενη από το χρόνο χρήσης, πολιτική χρέωσης τέτοιων υποδομών καθιστά επιτακτική την ανάγκη μείωσης του χρόνου ανάλυσης δεδομένων προκειμένου να διατηρηθεί το κόστος χρήσης τους σε ανεκτά επίπεδα.

5. Κινητές συσκευές
   Η ολοένα αυξανόμενη χρήση κινητών συσκευών, καθώς και η επικείμενη ανάπτυξη του Internet of Things, οδηγεί στην ανάγκη περιορισμού των ενεργοβόρων διαδικασιών. Οι συσκευές αυτές δεν έχουν τις ίδιες δυνατότητες με τους παραδοσιακούς υπολογιστές και άρα τεχνικές όπως το data sketching που περιορίζουν τις υπολογιστικές ανάγκες κρίνονται παραπάνω από απαραίτητες.

Είδη Data Sketching

Όπως αναφέρθηκε παραπάνω, το data sketching δεν αποτελεί μία συγκεκριμένη τεχνική αλλά είναι περισσότερο μια ιδέα πάνω στην οποία βασίζονται διάφορες τεχνικές. Μερικές από τις πιο σημαντικές είναι οι παρακάτω.

- Bloom Filter
  Το Bloom Filter είναι μια τεχνική data sketching κατάλληλη για περιπτώσεις όπου υπάρχει ανάγκη για queries ελέγχου ύπαρξης στοιχείων σε κάποιο set. Αποτελείται από έναν πίνακα μίας γραμμής και πλήθους στηλών που υπολογίζεται ανάλογα με το προσδοκώμενο από την εφαρμογή σφάλμα.

Αναλυτική περιγραφή του συγκεκριμένου τύπου sketching παρέχεται παρακάτω.

- Count Min Sketch
  Το Count Min Sketch αποτελεί μια κατηγορία data sketching κατάλληλη για εφαρμογές όπου απαιτείται η καταμέτρηση των ποσοτήτων διαφόρων στοιχείων. Αποτελείται από έναν πίνακα στον οποίο το πλήθος γραμμών και στηλών εξαρτάται από το προσδοκώμενο σφάλμα.
  Αναλυτική περιγραφή του συγκεκριμένου τύπου sketching παρέχεται παρακάτω.

- Count Sketch
  Η μέθοδος Count Sketch αποτελεί επίσης μία τεχνική data sketching για καταμέτρηση των ποσοτήτων διαφόρων στοιχείων. Για το λόγο ότι δεν παρουσιάζει κάποιο σημαντικό πλεονέκτημα σε σχέση με την Count Min Sketch (μάλιστα είναι χειρότερη στις περισσότερες περιπτώσεις ως προς το χώρο που καταλαμβάνει), δεν θα ασχοληθούμε με τη συγκεκριμένη τεχνική.

Εφαρμογές Data Sketching

Ενδεικτικές εφαρμογές όπου η ιδέα του data sketching εφαρμόζεται με επιτυχία είναι οι παρακάτω.

- Browser - κακόβουλα links — BLOOM FILTER
  Οι browsers είναι ανάγκη να διατηρούν στη μνήμη λίστα με τα γνωστά κακόβουλα sites προκειμένου να αποτρέπουν τους χρήστες από το να τα επισκέφτονται. Η αποθήκευση της πλήρους λίστας των sites θα έπιανε πολύ χώρο. Αντίθετα, με χρήση Bloom Filter είναι δυνατόν να ελέγχουμε αν ένα link ανήκει στο εν λόγω σύνολο, χωρίς να απαιτείται η πλήρης αποθήκευση των διευθύνσεων. Μάλιστα, στη συγκεκριμένη περίπτωση το false positive δεν είναι πρόβλημα. Κι αυτό γιατί τα περισσότερα links που θα ελέγχονται, όπως είναι φυσικό δεν θα ανήκουν στη λίστα. Τα λίγα που το Bloom Filter θα αποφαίνεται ότι ανήκουν, ακόμη κι αν τελικά κάνει λάθος, μπορούμε να τα ελέγχουμε και σε κάποια πλήρη λίστα των κακόβουλων sites που κρατείται κάπου στη μνήμη. Έτσι, στην πλειοψηφία των περιπτώσεων αποφεύγεται η “δύσκολη” αναζήτηση στην πλήρη λίστα και αυτή γίνεται μόνο σε ελάχιστες περιπτώσεις. Φυσικά είναι δυνατόν να μην κρατάμε καν πλήρη λίστα και να “εμπιστευόμαστε” μόνον το Bloom Filter. Κάτι τέτοιο παρουσιάζει επίσης καλά αποτελέσματα.

- Heavy hitters — COUNT MIN SKETCH
  Το πρόβλημα Heavy Hitters (λίστα στοιχείων με τιμή-ποσότητα στο top x% του συνόλου) μπορεί να προσεγγιστεί αποδοτικά με χρήση Count Min Sketch. Συγκεκριμένα, αποθηκεύουμε σε Count Min Sketch την τρέχουσα τιμή όλων των στοιχείων. Κάθε φορά που έρχεται ένα νέο στοιχείο, ανανεώνουμε την τιμή του και αν η νέα τιμή είναι μεγαλύτερη ή ίση του τρέχοντος threshold τότε προσθέτουμε το στοιχείο σε μια λίστα όπου κρατάμε τα αποτελέσματα. Το στοιχείο με τη μικρότερη τιμή στη λίστα κρατείται και σε κάθε επανάληψη ελέγχεται αν παραμένει στο top x%. Αν όχι αφαιρείται.

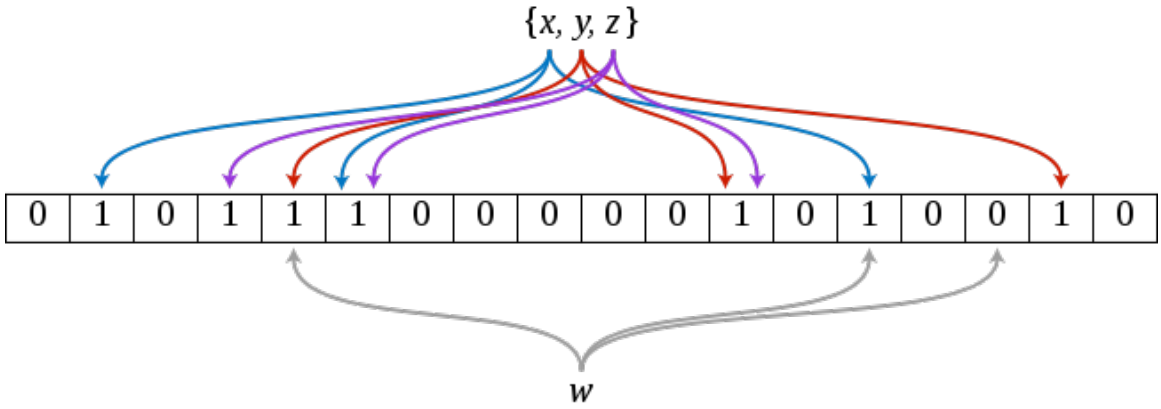
Άλλες περιοχές που το data sketching αποφέρει σημαντικές βελτιώσεις είναι οι τομείς matrix computations, network algorithms και machine learning.

Κεφάλαιο 3: Bloom Filter

Εισαγωγή

Το Bloom Filter αποτελεί μια τεχνική ελέγχου της ύπαρξης στοιχείων σε κάποιο σύνολο. Η μέθοδος αυτή είναι πιθανοτική, που σημαίνει ότι απαντά σε ερωτήματα τέτοιου τύπου με κάποια πιθανότητα και όχι ντετερμινιστικά. Το πλεονέκτημά της είναι ότι επιτυγχάνει κάτι τέτοιο σε χώρο σημαντικά μικρότερο απ' ότι μια πιο "παραδοσιακή" μέθοδος. Ακόμη, η χρονική πολυπλοκότητα των διάφορων queries (update και check) είναι επίσης εξαιρετικά αποδοτική, O(k) όπου k παράμετρος που θα αναλυθεί εκτενέστερα παρακάτω. Τέλος, ένα άλλο σημαντικό στοιχείο του Bloom Filter είναι ότι λόγω της φύσης του μπορεί να περιέχει πληροφορία για οσαδήποτε στοιχεία. Δεν υπάρχει δηλαδή περιορισμός στο πόσα στοιχεία "χωράει". Φυσικά όσο περισσότερα στοιχεία του προσθέτουμε τόσο μειώνεται η ακρίβεια της μεθόδου.

Τα παραπάνω καθιστούν τη συγκεκριμένη δομή ιδανική για εφαρμογές όπου το πλήθος των δεδομένων είναι "μεγάλο" (λόγω της καλής χωρικής πολυπλοκότητας) και η συχνότητα άφιξής τους είναι επίσης "μεγάλη" (λόγω της καλής χρονικής πολυπλοκότητας).



Σχήμα 8: Bloom Filter

Περιγραφή Bloom Filter

Η μέθοδος αποτελείται από έναν boolean πίνακα 1 γραμμής και n στηλών, καθώς και από ένα σύνολο από k hash functions.

Σε κάθε στοιχείο αντιστοιχίζονται μέσω των k hash functions, k αριθμοί που αντιστοιχούν σε k κελιά του πίνακα. Σε κάθε κελί περιέχεται η πληροφορία ύπαρξης ή μη του στοιχείου στο σύνολο (κάθε κελί περιέχει μία boolean μεταβλητή - 1 για ύπαρξη του στοιχείου στο σύνολο, 0 για μη ύπαρξή του). Φυσικά, από τη στιγμή που η χρησιμοποιούμενη δομή είναι στατική (πίνακας 1*n) και σε κάθε στοιχείο αντιστοιχίζονται k κελιά του πίνακα, από ένα σημείο και μετά είναι λογικό να υπάρχουν collisions μεταξύ των στοιχείων. Καθώς η μόνη "πράξη" που υποστηρίζεται από τη συγκεκριμένη τεχνική είναι αυτή της "προσθήκης" ενός στοιχείου στο set, το χειρότερο που μπορεί να συμβεί εξ' αιτίας των collisions είναι να θεωρηθεί λανθασμένα ότι κάποιο στοιχείο υπάρχει στο σύνολο (false positive), επειδή κάποιο από τα k κελιά του το "μοιράζεται" με κάποια άλλη μεταβλητή που πράγματι υπάρχει στο σύνολο.

Στην πραγματικότητα, σε αντίθεση με μια "παραδοσιακή" ντετερμινιστική τεχνική όπου θα απαιτούνταν μία boolean μεταβλητή για κάθε πιθανό στοιχείο, σε αυτήν την περίπτωση διαθέτουμε k boolean μεταβλητές για κάθε πιθανό στοιχείο, τις οποίες όμως "μοιράζεται" με άλλα στοιχεία (φυσικά είναι αρκετά σπάνιο 2 στοιχεία να έχουν όλα τα στοιχεία της k-άδας τους όμοια - από την άλλη είναι εξαιρετικά πιθανό να έχουν κοινά κάποια από τα κελιά τους).

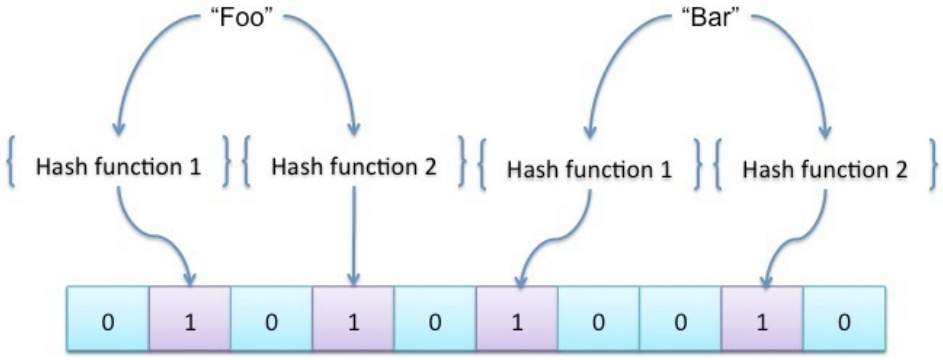
Bloom Filter Queries

Για να γίνουν πιο κατανοητά τα παραπάνω, θα εξηγήσουμε τώρα αναλυτικά τη διαδικασία που ακολουθείται για καθένα από τα 2 queries που υποστηρίζονται από τη συγκεκριμένη τεχνική. Η φύση του Bloom Filter είναι τέτοια που δεν επιτρέπει την πράξη της "αφαίρεσης" ενός στοιχείου από το set, καθώς αυτό μπορεί να επηρεάσει σημαντικά την πληροφορία σχετικά με την ύπαρξη ή μη πολλών άλλων στοιχείων και να "καταστρέψει" τελικά την αποτελεσματικότητα και την ακρίβεια του Bloom Filter.

Add

Για να προσθέσουμε την πληροφορία ύπαρξης κάποιου στοιχείου στο σύνολο, εργαζόμαστε ως εξής:

1. Τροφοδοτούμε τις k hash functions με το όνομα του στοιχείου και αυτές μας επιστρέφουν k ακεραίους.
2. Οι k ακέραιοι αντιστοιχίζονται σε k κελιά του πίνακα.
3. Θέτουμε τις τιμές αυτών των k κελιών σε 1.



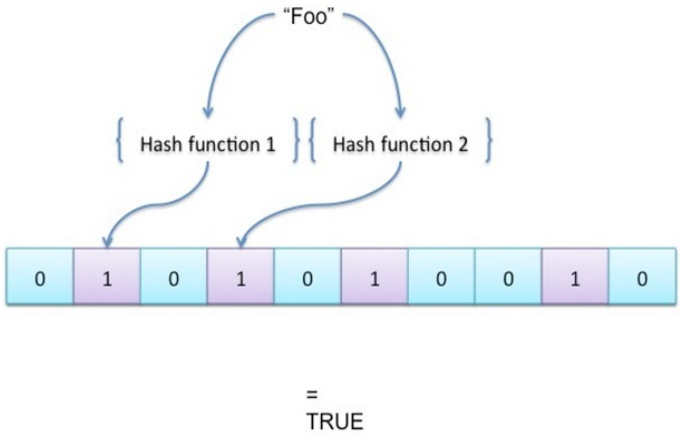
Σχήμα 9: Add query με k=2

Check

Για να ελέγξουμε αν κάποιο στοιχείο ανήκει στο σύνολο εργαζόμαστε ως εξής:

1. Τροφοδοτούμε τις k hash functions με το όνομα του στοιχείου και αυτές μας επιστρέφουν k ακεραίους.
2. Οι k ακέραιοι αντιστοιχίζονται σε k κελιά του πίνακα.

3. Αν έστω και ένα από τα k κελιά έχει τιμή 0, τότε απαντάμε ότι το στοιχείο δεν υπάρχει στο σύνολο.
Αν όλα τα κελιά έχουν τιμή 1, τότε απαντάμε ότι το στοιχείο ενδέχεται να ανήκει στο σύνολο.



Σχήμα 10: Check query με k=2

Η εξήγηση για τα παραπάνω έχει ως εξής:

Τα κελιά στα οποία αντιστοιχίζεται ένα στοιχείο αποτελούν k boolean μεταβλητές που δηλώνουν την ύπαρξη ή μη του εν λόγω στοιχείου στη λίστα.

Στην περίπτωση που δεν υπήρχαν collisions, θα έπρεπε αν το στοιχείο ανήκει πράγματι στο σύνολο, λαμβάνοντας υπ' όψιν και τη διαδικασία που ακολουθήθηκε στο add query, και οι k τιμές να είναι 1. Στην περίπτωση ύπαρξης collisions, αυτό δεν αλλάζει αφού η τιμή του "μοιραζόμενου" από περισσότερα στοιχεία κελιού δεν μπορεί να είναι 0 (αφού κατά την προσθήκη του στοιχείου έγινε 1) και φυσικά δεν μπορεί να έχει κάποια άλλη τιμή εκτός από 1 αφού πρόκειται για boolean μεταβλητή (επίσης δεν μπορεί να έχει ξαναλλάξει από 1 σε 0 αφού κάτι τέτοιο απαγορεύτηκε και είναι ο λόγος για τον οποίο δεν υπάρχει remove query, όπως εξηγήθηκε παραπάνω).

Αντίστοιχα, αν το στοιχείο δεν ανήκει στη λίστα, θα έπρεπε κανονικά και τα k κελιά που του αντιστοιχούν να έχουν τιμή 0. Λόγω ύπαρξης collisions όμως είναι δυνατόν κάποια απ' αυτά να έχουν γίνει 1 λόγω κάποιου άλλου στοιχείου που προστέθηκε στη λίστα και αντιστοιχίθηκε κι αυτό στο συγκεκριμένο κελί. Συνεπώς, απαντάμε ότι το στοιχείο δεν υπάρχει στο σύνολο αν έστω και ένα από τα κελιά που του αντιστοιχούν είναι 0 (αφού αν ανήκε στο σύνολο τότε κατά την προσθήκη του θα είχε κάνει όλα τα κελιά ίσα με 1). Φυσικά, ενδέχεται να είναι 1 και τα k κελιά, χωρίς το στοιχείο να ανήκει στο σύνολο (αυτό μπορεί να συμβεί αν υπάρχουν collisions σε όλα του τα κελιά από στοιχεία που ανήκουν στο σύνολο). Σε αυτήν την περίπτωση το bloom filter απαντά λανθασμένα.

Bloom Filter Accuracy

Η ακρίβεια ενός Bloom Filter αναλύεται παρακάτω.

Έστω m το μέγεθος του πίνακα και k το πλήθος των hash functions που χρησιμοποιούμε. Τότε η πιθανότητα ένα κελί του πίνακα να γίνει 1 από μία hash function κατά την προσθήκη ενός στοιχείου, είναι

$$\frac{1}{m}$$

Αντίστοιχα, η πιθανότητα ένα κελί να παραμείνει 0 μετά από την παραπάνω διαδικασία είναι φυσικά

$$1 - \frac{1}{m}$$

Λόγω του γεγονότος όμως ότι υπάρχουν k hash functions η πιθανότητα να παραμείνει ένα κελί στην τιμή 0 μετά την προσθήκη ενός νέου στοιχείου είναι

$$(1 - \frac{1}{m})^k$$

Αν το στοιχείο που προστίθεται δεν είναι το πρώτο, αλλά έχουν ήδη προστεθεί στο Bloom Filter πληροφορίες για n στοιχεία τότε η πιθανότητα (θεωρώντας ανεξάρτητα μεταξύ τους τα προγενέστερα στοιχεία - υπόθεση αρκετά αξιόπιστη δοθούσας της μεγάλης τιμής του n) γίνεται

$$(1 - \frac{1}{m})^{kn}$$

Συνεπώς, μετά την προσθήκη n στοιχείων, η πιθανότητα ένα οποιοδήποτε κελί να είναι 1 είναι

$$1 - (1 - \frac{1}{m})^{kn}$$

Με βάση τα παραπάνω, μπορούμε να υπολογίσουμε την πιθανότητα ενός false positive αποτελέσματος στο Bloom Filter. False positive έχουμε όταν και τα k κελιά στα οποία αντιστοιχίζεται μέσω των hash functions ένα στοιχείο έχουν την τιμή 1. Η συγκεκριμένη πιθανότητα είναι

$$(1 - (1 - \frac{1}{m})^{kn})^k \approx (1 - \epsilon^{-\frac{kn}{m}})^k$$

Παρατηρούμε λοιπόν ότι, όπως ήταν αναμενόμενο, όσο μεγαλύτερο είναι το μέγεθος m του πίνακα τόσο μικρότερη είναι η πιθανότητα λάθος εκτίμησης, ενώ όσο περισσότερα στοιχεία n έχουν προστεθεί στον πίνακα τόσο μεγαλύτερη γίνεται η πιθανότητα λάθος εκτίμησης.

Επιλογή Παραμέτρων

Από την παραπάνω σχέση σχετικά με την ακρίβεια του Bloom Filter, προκύπτει ότι για δοθέν μέγεθος πίνακα m, η τιμή του k που ελαχιστοποιεί την παραπάνω πιθανότητα είναι

$$k = \frac{m}{n} \ln 2$$

Εξίσωση 1: Καθορισμός πλήθους hash function σε bloom filter

Αντικαθιστώντας την παραπάνω τιμή στην αρχική πιθανότητα, έστω p η πιθανότητα αυτή, και μετά από πράξεις προκύπτει ότι το μέγεθος του πίνακα δίνεται με βάση την προσδοκόμενη πιθανότητα από τη σχέση

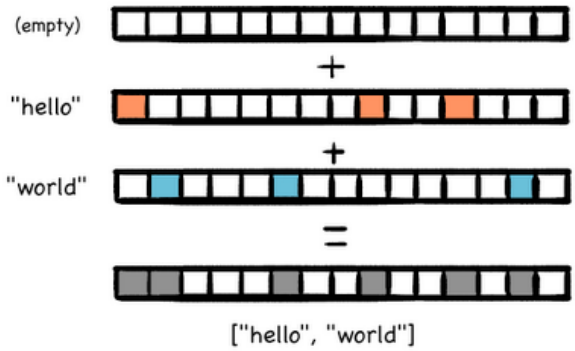
$$m = -\frac{n \ln p}{(\ln 2)^2}$$

Εξίσωση 2: Καθορισμός πλήθους στηλών σε bloom filter

Άλλες Ιδιότητες

Μερικές άλλες ενδιαφέρουσες ιδιότητες του Bloom Filter είναι:

1. Η πράξη UNION υποστηρίζεται από τα Bloom Filters. Συγκεκριμένα, αν έχουμε 2 ή περισσότερα Bloom Filters με ίδιου μεγέθους πίνακες (ίδια παράμετρο k) και προσθέσουμε τα στοιχεία τους, τότε ο πίνακας που προκύπτει πριέχει τη συνολική πληροφορία που περιείχε ο καθένας από τους αρχικούς πίνακες ξεχωριστά.
   Ο λόγος είναι ότι με ίδιες hash functions, τα στοιχεία που προστίθενται σε ένα Bloom Filter θα αντιστοιχούσαν στα ίδια κελιά και σ' ένα άλλο Bloom Filter με το ίδιο μέγεθος. Συνεπώς, αθροίζοντας τους 2 ή περισσότερους πίνακες προκύπτει ο πίνακας που θα προέκυπτε και στην περίπτωση που ανανεώναμε εξ' αρχής μόνον έναν πίνακα.
   Το παραπάνω είναι εξαιρετικά σημαντικό, λαμβάνοντας υπ' όψιν την ανάγκη των αλγορίθμων ανάλυσης "Μεγάλων Δεδομένων" για παραλληλοποιησιμότητα.



Σχήμα 11: Bloom Filter Union

2. Αν θεωρήσουμε X* μια εκτίμηση για το πλήθος των στοιχείων που έχουν προστεθεί στο set, N το μήκος του πίνακα, k το πλήθος των hash functions και X το πλήθος των κελιών του πίνακα που έχουν τεθεί σε 1, τότε έχει υπολογιστεί στη βιβλιογραφία ότι ο παρακάτω τύπος δίνει μια προσέγγιση του πλήθους των στοιχείων που έχουν προστεθεί στο σύνολο:

$$X^\star = -\frac{N \ln(1 - \frac{X}{N})}{k}$$

Κεφάλαιο 4: Count Min Sketch

Εισαγωγή

Το Count Min Sketch αποτελεί μια τεχνική αποθήκευσης δεδομένων, με τρόπο πιθανοτικό, ο οποίος επιτρέπει την αποθήκευση δεδομένων σε χώρο μικρότερο απ' ότι θα χρειαζόταν με κάποια άλλη πιο "παραδοσιακή" δομή δεδομένων. Παράλληλα, η συγκεκριμένη τεχνική επιτρέπει την πραγματοποίηση queries (update και find) με πολύ καλή χρονική πολυπλοκότητα O(k), όπου k σταθερά με τιμή συνήθως αρκετά "μικρή" που εξαρτάται αποκλειστικά από τις επιλεγμένες παραμέτρους με τις οποίες αρχικοποιούμε τη δομή μας. Περισσότερα σχετικά με τις παραμέτρους παρακάτω. Ακόμη, το Count Min Sketching δεν κάνει ποτέ overflow. Αυτό σημαίνει ότι για συγκεκριμένο μέγεθος του struct που χρησιμοποιεί το Count Min, μπορούμε να αποθηκεύσουμε άπειρο θεωρητικά πλήθος στοιχείων. Φυσικά, όσα περισσότερα στοιχεία αποθηκεύουμε τόσο μειώνεται η ακρίβεια των μετρητών.

Η τεχνική Count Min Sketching χρησιμοποιείται σε περιπτώσεις όπου απαιτείται η αποθήκευση των "ποσοτήτων" κάποιων στοιχείων. Είναι ιδανική όταν το πλήθος αυτών των στοιχείων είναι εξαιρετικά μεγάλο, με αποτέλεσμα να καθίσταται αδύνατη η δέσμευση μνήμης για την αποθήκευση των "ποσοτήτων" κάθε στοιχείου ξεχωριστά. Παράλληλα, η σταθερή χρονική πολυπλοκότητα επιτρέπει την αξιοποίησή της σε εφαρμογές που περιλαμβάνουν real time ροές δεδομένων, όπου ενδεχόμενη καθυστέρηση στα queries θα προκαλούσε σημαντικά προβλήματα αποδοτικότητας.

Περιγραφή Count Min Sketch

Η μέθοδος αποτελείται από έναν πίνακα k γραμμών και n στηλών, κι από ένα σύνολο από k hash functions.

Σε κάθε στοιχείο αντιστοιχίζονται μέσω των hash functions k αριθμοί στους οποίους κρατείται αποθηκευμένη η ζητούμενη "ποσότητα".

Φυσικά, από τη στιγμή που πρόκειται για μια δομή στατική (πίνακας), όσο αυξάνεται το πλήθος στοιχείων των οποίων τις "ποσότητες" αποθηκεύουμε στον πίνακα, τόσο αυξάνεται η πιθανότητα ύπαρξης collisions μεταξύ των στοιχείων. Παρ' όλ' αυτά, το γεγονός ότι για κάθε στοιχείο έχουμε αποθηκευμένη την αντίστοιχη "ποσότητα" σε k θέσεις του πίνακα μειώνει σημαντικά το ενδεχόμενο αλλοίωσης της πληροφορίας, αφού για να συμβεί κάτι τέτοιο θα πρέπει να υπάρξουν collisions και στα k στοιχεία που αντιστοιχούν σε κάθε στοιχείο.

Στην ουσία, σε αντίθεση με τις "παραδοσιακές" δομές δεδομένων στις οποίες θα απαιτούνταν ένας μετρητής για κάθε στοιχείο του οποίου την ποσότητα θέλουμε να αποθηκεύσουμε, σε αυτήν την περίπτωση κάθε στοιχείο αποθηκεύει την ποσότητά του σε k μετρητές, τους οποίους όμως "μοιράζεται" με άλλα στοιχεία. Έτσι, για να διατηρηθεί η πληροφορία για ένα στοιχείο αρκεί να διατηρηθεί η τιμή έστω σε έναν από τους k μετρητές που του αντιστοιχούν.

Επιλογή Παραμέτρων

Σύμφωνα με τη βιβλιογραφία η επιλογή των παραμέτρων k, n για την αρχικοποίηση του πίνακα μεγέθους k*n δίνονται από τις παρακάτω σχέσεις:

$$k = \ln \frac{1}{\delta}$$

$$n = \frac{e}{\epsilon}$$

Εξισώσεις 3-4: Καθορισμός πλήθους γραμμών και στηλών σε count min sketch

Οι παραπάνω σχέσεις είναι κρίσιμες για την τελική ακρίβεια του Count Min Sketch. Συγκεκριμένα, οι παραπάνω παράμετροι k, n εξασφαλίζουν ότι κάθε query απαντάται με σφάλμα ε σε πιθανότητα δ. Στην πραγματικότητα λοιπόν, το προσδοκόμενο ζεύγος (ε, δ) αποτελεί τις παραμέτρους του Count Min Sketch και με βάση αυτό καθορίζεται το μέγεθος του πίνακα που θα χρησιμοποιηθεί.

Count Min Sketch Queries

Για να γίνει πιο κατανοητή η παραπάνω τεχνική, θα περιγράψουμε στη συνέχεια τη διαδικασία που ακολουθείται στα 2 είδη queries που υποστηρίζει το Count Min Sketching.

Έστω διάνυσμα α μήκους m που δηλώνει ένα σύνολο από στοιχεία και έχει την παρακάτω μορφή

$$\alpha(t) = [\alpha_1(t), \ldots, \alpha_m(t)]$$

όπου το i-οστό στοιχείο συμβολίζει την ποσότητα του στοιχείου i την χρονική στιγμή t.

Αν χρησιμοποιούσαμε κάποια ντετερμινιστική μέθοδο, τότε η ανανέωση του διανύσματος κατά την άφιξη ενός ζεύγους (i, c) τη χρονική στιγμή t θα γινόταν ως εξής:

$$\alpha_i(t) = \alpha_i(t-1) + c$$

$$\alpha_j(t) = \alpha_j(t-1), j \neq i$$

Και φυσικά η αναζήτηση του στοιχείου i θα επέστρεφε την ποσότητα της i-οστής συνιστώσας του διανύσματος α.

Αντίστοιχα, στο Count Min Sketch, έστω οι παρακάτω hash functions.

$$h_1 \ldots h_k : \{1 \ldots m\} \rightarrow \{1 \ldots n\}$$

Τα queries έχουν λοιπόν ως εξής:

Update

Για να ανανεώσουμε τις τιμές που αντιστοιχούν σε κάποιο στοιχείο εργαζόμαστε ως εξής:

1. Τροφοδοτούμε τις k hash functions με το “όνομα” του εν λόγω στοιχείου και αυτές μας επιστρέφουν k αριθμούς:

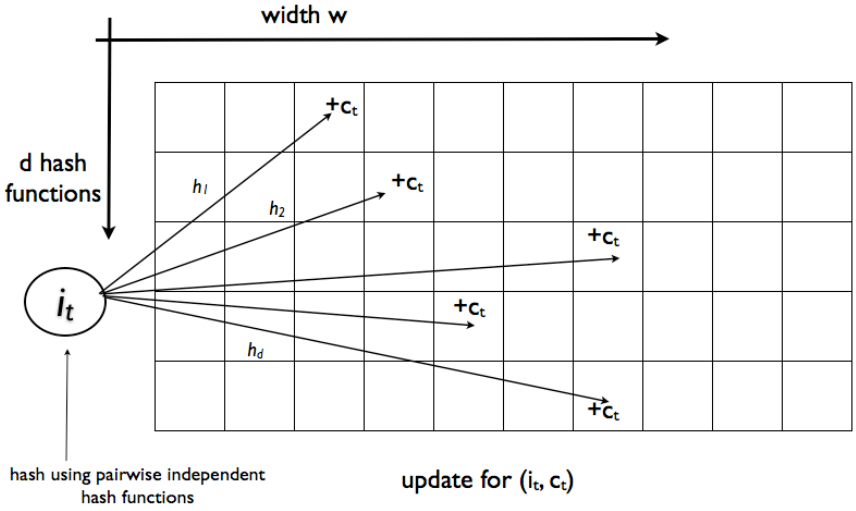
$$h_1(i), \ldots, h_k(i)$$

2. Οι k αριθμοί μας δίνουν τη στήλη στην οποία αντιστοιχίζεται το στοιχείο για καθεμία από τις k γραμμές του πίνακα.
3. Ανανεώνουμε τα k κελιά του πίνακα προσθέτοντας στις υπάρχουσες ποσότητες τη νέα τιμή.

$$count_t(1, h_1(i)) \leftarrow count_{t-1}(1, h_1(i)) + c$$

$$\ldots$$

$$count_t(k, h_k(i)) \leftarrow count_{t-1}(k, h_k(i)) + c$$



Σχήμα 12: Update Count Min

Find

Για να βρούμε την ποσότητα του στοιχείου i εργαζόμαστε ως εξής:

1. Ομοίως με το προηγούμενο query, τροφοδοτούμε τις k hash functions με το "όνομα" του εν λόγω στοιχείου και αυτές μας επιστρέφουν k αριθμούς.
2. Οι k αριθμοί μας δίνουν τη στήλη στην οποία αντιστοιχίζεται το στοιχείο για καθεμία από τις k γραμμές του πίνακα.
3. Επιστρέφουμε ως απάντηση στο query την ελάχιστη τιμή μεταξύ των k αυτών τιμών:

$$min_j count(j, h_j(i))$$

Η εξήγηση για τα παραπάνω είναι η εξής:

Παίρνουμε τις k τιμές για την ποσότητα του στοιχείου από τα αντίστοιχα κελιά.

Αν δεν υπήρχαν collisions τότε οποιαδήποτε από αυτές τις τιμές θα αποτελούσε την προς αναζήτηση ποσότητα.

Λόγω των collisions όμως, κάποιες από τις τιμές αυτές ενδέχεται να έχουν "πειραχθεί" από κάποιο άλλο στοιχείο που κάποια hash function έστειλε εκεί. Καθώς όμως κατά το update επιτρέψαμε μόνο αυξήσεις στις τιμές των κελιών, συμπεραίνουμε ότι μόνο μεγαλύτερη μπορεί να προκύψει κάποια "πειραγμένη" τιμή (το αντίστοιχο του false positive στο Bloom Filter).

Συνεπώς, από τις k τιμές που έχουμε για την "ποσότητα" του εν λόγω στοιχείου, η πιο "ακριβής" θα είναι αυτή με τη μικρότερη τιμή (αφού για να είναι η μικρότερη σημαίνει ότι είχε τα λιγότερα collisions).

Count Min Sketch Accuracy

Μια προσέγγιση της ακρίβειας στις τιμές που επιστρέφει το Count Min Sketch δίνεται παρακάτω.

Έστω Q(i) η απάντηση στην ερώτηση "Ποια η εκτίμηση για την τιμή του στοιχείου i". Φυσικά, σύμφωνα με τα παραπάνω, η τιμή του Q(i) θα είναι η ελάχιστη τιμή μεταξύ των τιμών του στοιχείου i στις διάφορες γραμμές του πίνακα. Έστω j η γραμμή του πίνακα με την ελάχιστη τιμή. Τότε θα ισχύει:

$$Q(i) = count(j, h_j(i))$$

Η παραπάνω ποσότητα όμως, περιλαμβάνει την ποσότητα που πράγματι ανήκει στο i, περιλαμβάνει όμως και τις ποσότητες διάφορων άλλων στοιχείων που επίσης αντιστοιχίζονται από την j-οστή hash function στο συγκεκριμένο κελί. Συνεπώς έχουμε:

$$Q(i) = count(j, h_j(i)) = \alpha_i(t) + \sum_{b \in h_j^{-1}(i) \setminus \{i\}} f(b)$$

Θεωρώντας ότι κάθε update του πίνακα γίνεται ανεξάρτητα από τα προηγούμενα, συμπεραίνουμε ότι για n στήλες, κάθε στήλη θα έχει κατά μέσο όρο τιμή ίση με

$$\frac{t}{n}, t = \sum_t c_t$$

Συμπεραίνουμε λοιπόν ότι η απόκλιση μεταξύ της εκτίμησης του Count Min Sketch και της πραγματικής τιμής του στοιχείου i θα είναι

$$Q(i) - \alpha_i(t) = \sum_{b \in h_j^{-1}(i) \setminus \{i\}} \leq t \times \frac{1}{n}$$

Αντικαθιστώντας στην παρακάτω ανισότητα τη σχέση που δίνει το πλήθος των στηλών n, προκύπτει ότι

$$Q(i) - \alpha_i(t) \leq t \times \frac{\epsilon}{e}$$

Σύμφωνα με την ανισότητα του Markov, έχουμε

$$P[Q(i) - \alpha_i(t) \geq \epsilon \times t] \leq \frac{t \times \frac{\epsilon}{e}}{\epsilon \times t} = \frac{1}{e}$$

Η παραπάνω σχέση δίνει την πιθανότητα της παραπάνω απόκλισης για μία συγκεκριμένη γραμμή του πίνακα. Δεδομένου όμως ότι για κάθε στοιχείο i, λαμβάνουμε υπ' όψιν k γραμμές, συμπεραίνουμε ότι η αντίστοιχη πιθανότητα γίνεται

$$P[Q(i) - \alpha_i(t) \geq \epsilon \times t] \leq \frac{1}{e^k}$$

Τέλος, αντικαθιστώντας στην παραπάνω ανισότητα τη σχέση από την οποία προκύπτει το πλήθος γραμμών k, έχουμε

$$P[Q(i) - \alpha_i(t) \geq \epsilon \times t] \leq \delta$$

Και άρα τελικά

$$P[Q(i) - \alpha_i(t) \leq \epsilon \times t] \geq 1 - \delta$$

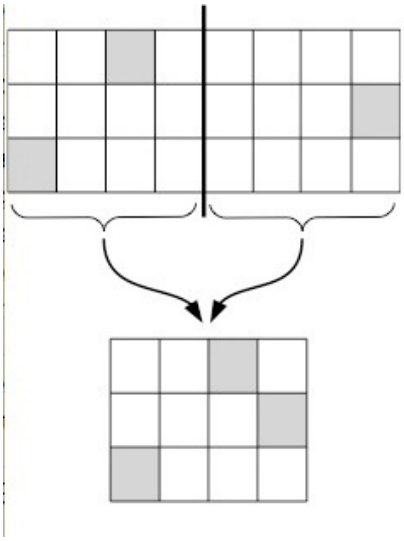
Εξίσωση 5: Εκτίμηση ακρίβειας της μεθόδου count min sketch

Count Min Sketch Union

Το Count Min Sketch επιτρέπει την ένωση 2 ή περισσότερων Count Min Struct με τις ίδιες παραμέτρους. Αυτό σημαίνει ότι αν έχουμε 2 ή περισσότερους πίνακες ίδιου μεγέθους τότε αθροίζοντας τα στοιχεία τους, ο πίνακας που προκύπτει περιέχει τη συνολική πληροφορία που περιείχε αρχικά ξεχωριστά ο κάθε πίνακας.

Ο λόγος είναι απλός. Για τις ίδιες παραμέτρους, και φυσικά τις ίδιες hash functions, ένα στοιχείο αντιστοιχίζεται στα ίδια κελιά σε όλους τους πίνακες. Συνεπώς, προσθέτοντας 2 πίνακες, προκύπτει ο πίνακας που θα προέκυπτε αν εξ' αρχής αντιστοιχίζαμε όλα τα στοιχεία στα κελιά του ίδιου πίνακα.

Κάτι τέτοιο είναι εξαιρετικά σημαντικό, καθώς επιτρέπει την παραλληλοποίηση των εφαρμογών.



Σχήμα 13: Count Min Union

Κεφάλαιο 5: Hash Functions

Εισαγωγή

Τόσο στο Bloom Filter όσο και στο Count Min Sketch, σημαντικότατο ρόλο στη σωστή υλοποίηση και τη βέλτιστη ακρίβεια παίζει η ορθή επιλογή των hash functions.

Απαιτήσεις των Hash Functions

Οι hash functions που θα επιλεγούν θα πρέπει να είναι ανά δύο ανεξάρτητες μεταξύ τους. Κάτι τέτοιο είναι σημαντικό προκειμένου να μειωθεί η πιθανότητα ύπαρξης στοιχείων που να έχουν κοινά περισσότερα από ένα κελιά, ενδεχόμενο στο οποίο θα υπήρχε μεγάλο πρόβλημα αφού η ιδέα των δύο αυτών τεχνικών είναι ότι ακόμη κι αν υπάρχουν collisions με άλλα στοιχεία, τα collisions αυτά θα είναι σε ένα το πολύ κελί ανά ζευγάρι στοιχείων (τουλάχιστον στην πλειοψηφία των περιπτώσεων).

Μορφή των Hash Functions

Προκειμένου να ικανοποιηθεί η παραπάνω απαίτηση, επιλέγονται hash functions της μορφής

$$(ax + b) mod\, p$$

όπου οι a, b είναι πρώτοι αριθμοί διαφορετικοί για κάθε hash function και ο p επίσης πρώτος αριθμός. Οι συγκεκριμένες συνθήκες θεωρούνται από τη βιβλιογραφία ικανές για την εξασφάλιση των παραπάνω περιορισμών.

Φυσικά, οι τιμές που προκύπτουν από τις παραπάνω συναρτήσεις “φιλτράρονται” με ένα ακόμα modulo δίνοντας τελικά τη σχέση

$$((ax + b) mod\, p) mod\, n$$

όπου n το πλήθος των στηλών. Η τελευταία κίνηση γίνεται προκειμένου να αποφεύγονται τα overflow στους χρησιμοποιούμενους πίνακες.

Είναι εμφανές ότι για τον πρώτο αριθμό p με τον οποίο κάνουμε αρχικά modulo θα πρέπει να ισχύει

$$p \geq n$$

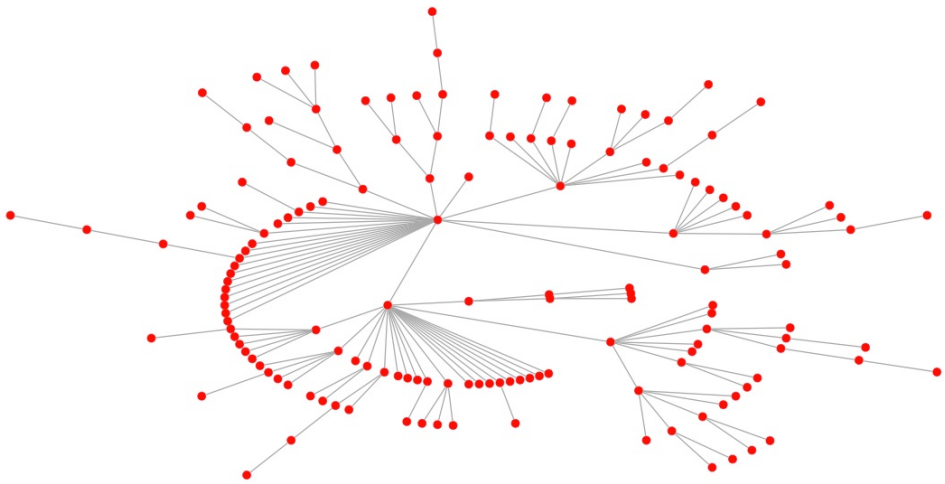
Διαφορετικά οι hash functions θα άφηναν μονίμως κενές (n-p) στήλες του πίνακα.

Κεφάλαιο 6: Power Law Distribution

Περιγραφή

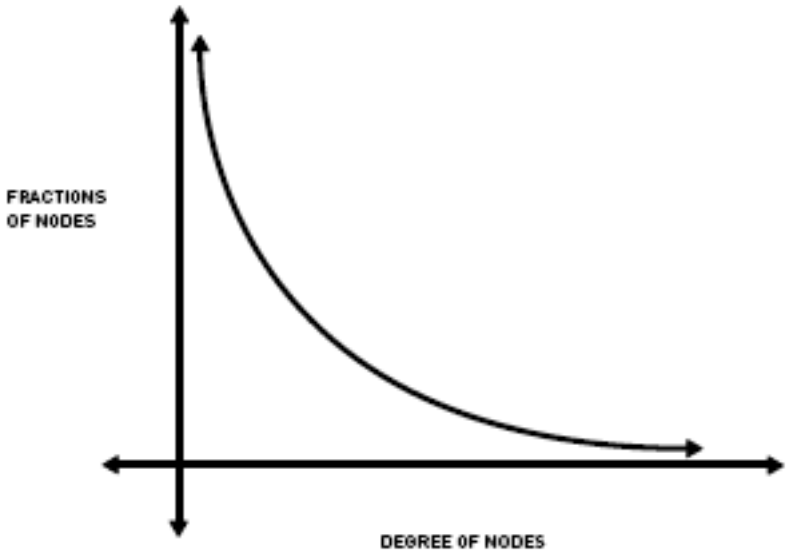
Η κατανομή αυτή αποτελεί μια σχέση μεταξύ δύο μεγεθών όπου το ένα μέγεθος μεταβάλλεται σαν μια αρνητική δύναμη της άλλης. Με άλλα λόγια το πλήθος των στοιχείων του ενός μεγέθους με “μεγάλη” αντίστοιχη τιμή στο άλλο μέγεθος είναι “μικρό” και το αντίστροφο.

Έχει παρατηρηθεί ότι το Power Law Distribution βρίσκει εφαρμογή σε πολλούς τομείς του περιβάλλοντος. Μεταξύ άλλων, η συγκεκριμένη κατανομή εμφανίζεται σε γράφους, και ιδιαίτερα σε κοινωνικούς γράφους, κατά τη γραφική αναπαράσταση των κόμβων και των αντίστοιχων βαθμών τους. Όπως είναι φυσικό, το μεγαλύτερο ποσοστό των κόμβων έχει “μικρό” σχετικά βαθμό, ενώ λίγοι μόνο ξεφεύγουν σε μεγαλύτερους βαθμούς.



Σχήμα 14: Power Law γράφος

Το ποσοστό μεταξύ δημοφιλών και μη-δημοφιλών στοιχείων διαφέρει ανάλογα με την εφαρμογή. Μια σχέση 80 προς 20 έχει παρατηρηθεί ότι ισχύει σε πολλές από τις εφαρμογές. Ειδικά σε κοινωνικούς γράφους, η συγκεκριμένη σχέση μπορεί να φτάσει ακόμη και 99 προς 1.

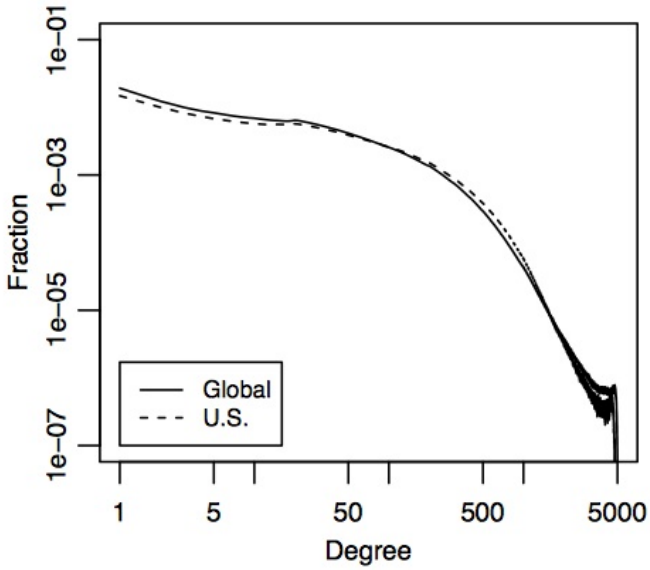


Σχήμα 15: Power Law Distribution μεταξύ του βαθμού και του αντίστοιχου πλήθους κόμβων

Παραδείγματα

Το παραπάνω γίνεται πιο εύκολα αντιληπτό αν αναλογιστούμε ορισμένα παραδείγματα.

1. Σε κοινωνικούς γράφους, όπου οι κόμβοι αναπαριστούν ανθρώπους και οι αντίστοιχοι βαθμοί αναπαριστούν τον αριθμό των φίλων τους, οι περισσότεροι άνθρωποι-κόμβοι έχουν "λίγους" σχετικά φίλους, ενώ ένα πολύ μικρό ποσοστό εμφανίζεται να διαθέτει "πολλούς" φίλους.
   Στο Facebook, αν και η παραπάνω πρόταση ισχύει, η κατανομή δεν είναι ακριβώς η ίδια αλλά μοιάζει περισσότερο με αυτήν στο παραπάνω σχήμα (σχεδιασμένη σε loglog κλίμακα). Ο λόγος είναι ότι ενώ συνεχίζει να ισχύει ότι οι περισσότεροι χρήστες έχουν "μικρό" αριθμό φίλων, η αντίστοιχη συνάρτηση δεν ακολουθεί αντίστροφα εκθετική κατανομή όπως ίσως θα περιμέναμε. Παρ' όλ' αυτά η γενική ιδέα ύπαρξης "πολλών" χρηστών με "λίγους" φίλους και το αντίστροφο εξακολουθεί να ισχύει. Παράλληλα, "μικρά" subsets του συνολικού γράφου του Facebook (ego networks) ακολουθούν κανονικά Power Law Distribution.



Σχήμα 16: Κατανομή κόμβων στο Facebook graph σε λογαριθμική κλίμακα

2. Κατά την αναπαράσταση του διαδικτύου σε μορφή γράφου, όπου οι κόμβοι αναπαριστούν τις ιστοσελίδες και οι αντίστοιχοι βαθμοί αντιστοιχούν στο πλήθος των συνδέσεων με άλλες ιστοσελίδες, οι "δημοφιλείς" (αυτές δηλαδή με μεγάλο βαθμό) ιστοσελίδες είναι φυσικά σημαντικά λιγότερες από αυτές με ελάχιστες συνδέσεις προς αυτές από άλλες ιστοσελίδες.

Μέρος 2ο: Υλοποίηση και Μετρήσεις

Κεφάλαιο 7: Benchmark Υλοποίηση

Εισαγωγή

Όπως έχει ήδη αναφερθεί, αναζητούμε έναν αλγόριθμο που να υπολογίζει το in-degree των κόμβων ενός γράφου, αλλά και αντίστροφα το πλήθος κόμβων με συγκεκριμένο in-degree καταναλώνοντας όσο το δυνατόν λιγότερους πόρους.

Προκειμένου να μπορούμε να ελέγχουμε την ακρίβεια των αποτελεσμάτων είναι αναγκαίο να έχουμε μία benchmark υλοποίηση του συγκεκριμένου αλγορίθμου.

Η υλοποίηση αυτή θα πρέπει να είναι όσο το δυνατόν αποδοτικότερη, χωρίς όμως να γίνεται πιθανοτική. Θα πρέπει δηλαδή να είναι απολύτως ακριβής ώστε να υπάρχει μέτρο σύγκρισης των πιθανοτικών υλοποιήσεων που θα περιγραφούν στη συνέχεια.

Για την αποθήκευση των in-degrees των nodes θα χρειαστεί να κρατάμε έναν μετρητή για κάθε node. Καθώς δε γνωρίζουμε εκ των προτέρων το πλήθος των διαφορετικών nodes του γράφου, κάτι τέτοιο δεν είναι εφικτό με χρήση στατικού πίνακα. Αντ' αυτού θα μπορούσε να χρησιμοποιηθεί linked-list ή hash-map για την αποθήκευση αυτών των δεδομένων. Οι δύο λύσεις περιγράφονται παρακάτω.

Linked List

Η ιδέα του αλγορίθμου είναι να χρησιμοποιήσουμε μία linked-list δομή δεδομένων για την αποθήκευση του in-degree των διάφορων nodes. Η λίστα θα περιέχει ένα στοιχείο για κάθε διαφορετικό node του γράφου.

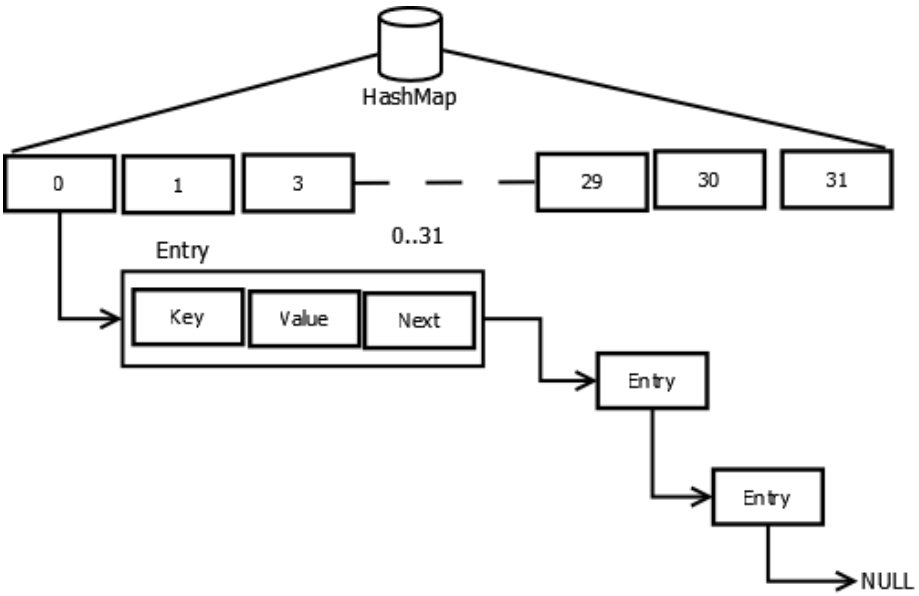
Η συγκεκριμένη δομή είναι αρκετά απλή στην υλοποίησή της και μάλιστα αν και καταλαμβάνει χώρο O(n) για την αποθήκευση των στοιχείων (n το πλήθος των διαφορετικών κόμβων), το καθένα από τα n στοιχεία της καταλαμβάνει χώρο μικρότερο απ' ότι αν χρησιμοποιούσαμε κάποιοα πιο πολύπλοκη δομή, όπως για παράδειγμα το hash-map. Παρουσιάζει λοιπόν ένα σημαντικό πλεονέκτημα από άποψη χώρου.

Όμως, η πραγματοποίηση των queries (update και find) θα απαιτεί το πέρασμα όλων των στοιχείων της λ΄ιστας προκειμένου να εντοπιστεί αυτό που αναζητείται. Συνεπώς, θα απαιτείται O(n) χρονική πολυπλοκότητα για κάθε query. Αυτό είναι φυσικά απαγορευτικό για εφαρμογές που λαμβάνουν δεδομένα από streams σε real-time και γι' αυτό το λόγο ο συγκεκριμένος αλγόριθμος απορρίπτεται.

Hash Map

Εισαγωγή

Η ιδέα του συγκεκριμένου αλγορίθμου είναι να χρησιμοποιήσουμε 2 hash-map δομές δεδομένων, μία για την αποθήκευση του in-degree των διάφορων nodes και μία για την αποθήκευση του πλήθους nodes για τα διάφορα in-degree.



Σχήμα 17: Σχήμα δομής hash-map

Περιγραφή Αλγορίθμου

Συγκεκριμένα, το πρώτο hash-map θα περιέχει n στοιχεία, ένα για κάθε διαφορετικό κόμβο του γράφου. Το id του κόμβου θα χρησιμοποιείται ως κλειδί για το συγκεκριμένο στοιχείο, ενώ η τιμή που θα λαμβάνει θα αποτελεί το in-degree του εν λόγω κόμβου.

Το δεύτερο hash-map θα περιέχει πλήθος στοιχείων ίσο με το μέγιστο in-degree του γράφου. Σε αυτήν την περίπτωση, το in-degree είναι αυτό που χρησιμοποιείται ως κλειδί, ενώ ως τιμή τίθεται το πλήθος κόμβων με in-degree μεγαλύτερο ή ίσο του κλειδιού.

Queries Αλγορίθμου

Τα βήματα του αλγορίθμου για την ανανέωση των δομών κατά την άφιξη μιας νέας ακμής καθώς και για την αναζήτηση του πλήθους κόμβων για δοθέν in-degree έχουν ως εξής:

Update

1. Άφιξη νέας ακμής και ανάγνωση του κόμβου στον οποίο εισέρχεται.
2. Αν ο συγκεκριμένος κομβος είναι καινούργιος, δημιουργία ενός νέου στοιχείου στο πρώτο hash-map με κλειδί το id του νέου κόμβου και αρχική τιμή 1.
   Διαφορετικά, εύρεση του εν λόγω κόμβου στο hash-map και ανανέωση της τιμής του προσθέτοντας +1 στην προηγούμενη τιμή του.
3. Ανανέωση του δεύτερου hash-map ως εξής:
   Αν υπάρχουν ήδη κόμβοι με in-degree ίσο με αυτό που προέκυψε από το βήμα 2 τότε εύρεση του αντίστοιχου στοιχείου στο δεύτερο hash-map και ανανέωση της τιμής του προσθέτοντας +1 στην προηγούμενη τιμή του.
   Διαφορετικά, δημιουργία στο δεύτερο hash-map ενός νέου στοιχείου με κλειδί το in-degree του βήματος 2 και αρχική τιμή 1.

Find

Για την αναζήτηση του πλήθους κόμβων με in-degree μεγαλύτερο ή ίσο του X εργαζόμαστε ως εξής:

1. Αναζήτηση στο δεύτερο hash-map του στοιχείου με κλειδί X. Αν υπάρχει τέτοιο στοιχείο τότε επιστρέφουμε την τιμή του, διαφορετικά επιστρέφουμε μηδέν.

Για την αναζήτηση του in-degree κάποιου συγκεκριμένου node, η διαδικασία είναι η ίδια με την προαναφερθείσα με τη διαφορά ότι η αναζήτηση γίνεται στο πρώτο hash-map.

Φυσικά, με μικρές μετατροπές είναι δυνατόν να επιστραφεί το πλήθος κόμβων με in-degree ίσο με X, μικρότερο από X, κ.ο.κ.

Για παράδειγμα, για τον υπολογισμό του πλήθους κόμβων με in-degree ίσο με X αρκεί να τρέξουμε το παραπάνω query για τις τιμές X και X+1 και η διαφορά των δύο τιμών που επιστρέφονται θα αποτελούν το ζητούμενο πλήθος.

Συγκεκριμένα, το query FIND για τιμή X θα επιστρέψει το πλήθος κόμβων με in-degree μεγαλύτερο ή ίσο του X.

$$FIND(X) = \#nodes(X) + \#nodes(X+1) + \#nodes(X+2) + ...$$

Το αντίστοιχο query για τιμή X+1 θα επιστρέψει το πλήθος κόμβων με in-degree μεγαλύτερο ή ίσο του X+1.

$$FIND(X+1) = \#nodes(X+1) + \#nodes(X+2) + \#nodes(X+3) + ...$$

Η διαφορά των δύο δίνει λοιπόν πράγματι τη ζητούμενη ποσότητα, όπως φαίνεται και παρακάτω.

$$FIND(X) - FIND(X+1) = \#nodes(X)$$

Χρονική Πολυπλοκότητα

Ο παραπάνω αλγόριθμος, τόσο για την προσθήκη ενός νέου στοιχείου όσο και για την αναζήτηση μιας τιμής σε κάποιο εκ των δύο hash-maps, ακολουθεί την πολυπλοκότητα της δομής hash-map. Εδώ φαίνεται η σημασία του δεύτερου hash-map αφού αν αυτό δεν υπήρχε τότε για τα queries αναζήτησης πλήθους κόμβων με in-degree μεγαλύτερο ή ίσο του X θα απαιτούνταν γραμμικό πέρασμα όλων των στοιχείων του πρώτου hash-map (O(n)).

Προκύπτει λοιπόν ότι οι παραπάνω διαδικασίες γίνονται σε σταθερό χρόνο, O(1), στη μέση περίπτωση.

Χωρική Πολυπλοκότητα

Το πρώτο hash-map έχει πλήθος στοιχείων ίσο με το πλήθος των διαφορετικών κόμβων του γράφου.

Το δεύτερο hash-map έχει πλήθος στοιχείων ίσο με το μέγιστο in-degree.

Για λόγους απλότητας θεωρούμε το χώρο που απαιτεί ο αλγόριθμος ίσο με το χώρο του πρώτου hash-map δεδομένου ότι το πλήθος κόμβων είναι τάξη μεγέθους μεγαλύτερο του μέγιστου in-degree λόγω της φύσης των γράφων. Συνεπώς, θεωρούμε τη χωρική πολυπλοκότητα του αλγορίθμου O(n), όπου n το πλήθος των διαφορετικών κόμβων του γράφου.

Κεφάλαιο 8: Count Min Υλοποίηση

Εισαγωγή

Επιδίωξη της παρούσας υλοποίησης είναι να λύσει το ζητούμενο πρόβλημα, όπως αυτό περιγράφηκε σε προηγούμενο κεφάλαιο, κάνοντας χρήση της μεθόδου Count Min ώστε, με τρόπο πιθανοτικό, να πετύχει έναν αποδοτικότερο σε χώρο, αλλά ταυτόχρονα όσο το δυνατόν ακριβέστερο, υπολογισμό των αποτελεσμάτων.

Περιγραφή Αλγορίθμου

Η ιδέα του αλγορίθμου είναι να χρησιμοποιήσει τη μέθοδο Count Min για την αποθήκευση των in-degrees των κόμβων. Συγκεκριμένα, θα υπάρχει ένας πίνακας, σύμφωνα με όσα προβλέπει η τεχνική του Count Min Sketch, στον οποίο θα αποθηκεύεται για κάθε διαφορετικό node του γράφου το αντίστοιχο in-degree.

Το παραπάνω αρκεί προκειμένου η υλοποίηση να απαντά σε ερωτήματα του τύπου “Ποιο είναι το in-degree του κόμβου X”. Για να μπορεί να απαντά επιπρόσθετα σε ερωτήματα του τύπου “Πόσα nodes έχουν in-degree ίσο / μεγαλύτερο ή ίσο / μικρότερο του X”, θα πρέπει να κρατάμε το σύνολο των nodes με in-degree μεγαλύτερο ή ίσο του X σε μια επιπλέον δομή δεδομένων, όπως κάναμε και στην benchmark υλοποίηση.

Αυτό μπορεί να γίνει, ομοίως με πριν, με χρήση ενός hash-map όπου στο X-οστό στοιχείο του θα κρατείται το πλήθος κόμβων με in-degree μεγαλύτερο ή ίσο του X. Αντίστοιχο αποτέλεσμα θα μπορούσαμε να πετύχουμε και με χρήση ενός δεύτερου Count Min στο οποίο θα κρατάμε την ίδια πληροφορία σε μικρότερο χώρο. Επειδή όμως, όπως εξηγήσαμε στο κεφάλαιο της benchmark υλοποίησης, το μέγεθος αυτού του hash-map δεν είναι πολύ μεγάλο στη συγκεκριμένη εφαρμογή, αποφεύγουμε το δεύτερο Count Min που ενδεχομένως να μειώσει την τελική ακρίβεια. Σε εφαρμογές όπου το εν λόγω μέγεθος είναι σημαντικό θα μπορούσε πράγματι να χρησιμοποιηθεί ένα δεύτερο Count Min έναντι του hash-map.

Queries Αλγορίθμου

Τα βήματα υλοποίησης των δύο queries της υλοποίησης περιγράφονται παρακάτω.

Update

Ανανέωση Count Min

1. Άφιξη νέας ακμής και ανάγνωση του κόμβου στον οποίο εισέρχεται.
2. Εύρεση μέσω των hash functions των κελιών του πίνακα Count Min στα οποία αντιστοιχίζεται ο συγκεκριμένος κόμβος.
3. Αύξηση κατά ένα της τιμής των κελιών αυτών.

Ανανέωση Hash Map

1. Αν υπάρχουν ήδη κόμβοι με in-degree ίσο με αυτό που προέκυψε από την προηγούμενη διαδικασία τότε εύρεση του αντίστοιχου στοιχείου στο hash-map και ανανέωση της τιμής

του προσθέτοντας +1 στην προηγούμενη τιμή του.
Διαφορετικά, δημιουργία στο hash-map ενός νέου στοιχείου με κλειδί το ανανεωμένο in-degree του συγκεκριμένου κόμβου και αρχική τιμή ίση με 1.

Find

Εύρεση in-degree του κόμβου X

1. Εύρεση μέσω των hash functions των κελιών του πίνακα Count Min στα οποία αντιστοιχίζεται ο κόμβος X.
2. Επιστροφή ως in-degree του κόμβου X, της μικρότερης τιμής μεταξύ των τιμών των παραπάνω κελιών.

Εύρεση πλήθους κόμβων με in-degree μεγαλύτερο ή ίσο του X

1. Αναζήτηση του στοιχείου με κλειδί X στο hash-map.
   Αν υπάρχει τέτοιο στοιχείο τότε επιστροφή της τιμής του.
   Αν όχι τότε επιστροφή της τιμής μηδέν.

Φυσικά, όμοια με την benchmark υλοποίηση, μπορούν να απαντηθούν ερωτήματα σχετικά με το πλήθος κόμβων με in-degree ίσο / μικρότερο του X.

Χρονική Πολυπλοκότητα

Ο παραπάνω αλγόριθμος για το query UPDATE αποτελείται όπως είδαμε από δύο στάδια, το στάδιο αναέωσης του Count Min πίνακα και το στάδιο ανανέωσης του hash-map. Και οι δύο αυτές ανανεώσεις απαιτούν, όπως γνωρίζουμε από τη θεωρία, σταθερό χρόνο για να πραγματοποιηθούν.

Για το query FIND ανάλογα με το ερώτημα που έχει γίνει από την εφαρμογή μπορεί να αποτελείται είτε από τη διαδικασία αναζήτησης της τιμής ενός κόμβου στο Count Min πίνακα είτε από τη διαδικασία αναζήτησης κάποιας τιμής στο hash-map. Όποιο από τα δύο ερωτήματα όμως και να τεθεί, ο χρόνος που χρειάζεται είναι σταθερός.

Με βάση τα παραπάνω συμπεραίνουμε ότι σε κάθε περίπτωση, όποιο κι αν είναι το query που τίθεται, ο αλγόριθμος απαιτεί σταθερή, O(1), χρονική πολυπλοκότητα για την πραγματοποίησή του. Φυσικά η συγκεκριμένη χρονική πολυπλοκότητα όταν εφαρμοστεί πάνω σ' ένα stream δεδομένων μήκους n, θα απαιτεί O(n) πολυπλοκότητα για το διάβασμα των δεδομένων.

Γενικά, το σημαντικό είναι ότι κάθε query γίνεται σε σταθερό χρόνο, καθώς επίσης ότι το σύνολο ενός stream μπορεί να αναλυθεί σε ένα μόνον πέρασμα.

Χωρική Πολυπλοκότητα

Λόγω του "μικρού" σχετικά, όπως εξηγήθηκε και σε προηγούμενο κεφάλαιο, αναμενόμενου μεγέθους του hash-map, θεωρούμε ότι ο χώρος που καταλαμβάνει ο αλγόριθμος εξαρτάται κυρίως από το μέγεθος του Count Min πίνακα.

Σύμφωνα με τη θεωρία, που αναλύθηκε στο αντίστοιχο κεφάλαιο, η επιλογή των παραμέτρων σχετικά με το μέγεθος του πίνακα έχει ως εξής:

$$\delta = 0.05$$

$$1 - \delta = 0.95$$

- Για 95% πιθανότητα μια τυχαία τιμή του πίνακα να είναι μέσα στα όρια του προβλεπόμενου πίνακα έχουμε
  Και άρα

  Συνεπώς, το πλήθος γραμμών του πίνακα είναι

  $$\ln \frac{1}{\delta} = 3$$

  Η επιλογή της συγκεκριμένης παραμέτρου έγινε προκειμένου να υπάρχει αρκετά καλή πιθανότητα (95%), χωρίς όμως να γίνει εξαιρετικά μεγάλος ο πίνακας όπως θα γινόταν για παράδειγμα αν επιλεγόταν πιθανότητα κοντά στο 100%.

- Σχετικά με το πλήθος των στηλών, αυτό όπως είδαμε εξαρτάται από το ανεχόμενο σφάλμα σε κάθε ανανέωση του πίνακα.

  $$n = \frac{e}{\epsilon}$$

  $$P[Q(i) - a_i(t) \leq \frac{e}{n} \times t] \geq 1 - \delta$$

  Επειδή όμως, όπως φαίνεται, το τελικό σφάλμα εξαρτάται από το πλήθος των ανανεώσεων (ή αλλιώς από το πλήθος των ακμών που έχουν διαβαστεί), το οποίο δεν μπορούμε να γνωρίζουμε εκ των προτέρων, επιλέγουμε πλήθος στηλών 200.000, το οποίο για ένα τυχαίο data set που περιέχει για παράδειγμα 1.000.000 ακμές, εξασφαλίζει ότι το σφάλμα στην πλειοψηφία των στοιχείων δεν θα υπερβαίνει την τιμή 13, με βάση τον παραπάνω τύπο. Φυσικά, η συγκεκριμένη επιλογή είναι αρκετά αυθαίρετη και δεν ικανοποιεί όλα τα data sets, αφού σε άλλα θα είναι υπερβολικά μεγάλη και σε άλλα θα είναι υπερβολικά μικρή. Αυτό φαίνεται καλύτερα στις μετρήσεις που παρατίθενται στη συνέχεια.

Συμπερασματικά, με τις συγκεκριμένες παραμέτρους προκύπτει πίνακας μεγέθους 3x200.000 και άρα ο χώρος που καταλαμβάνει η εφαρμογή θα είναι της τάξης του

$$600.000 \times sizeof(int)$$

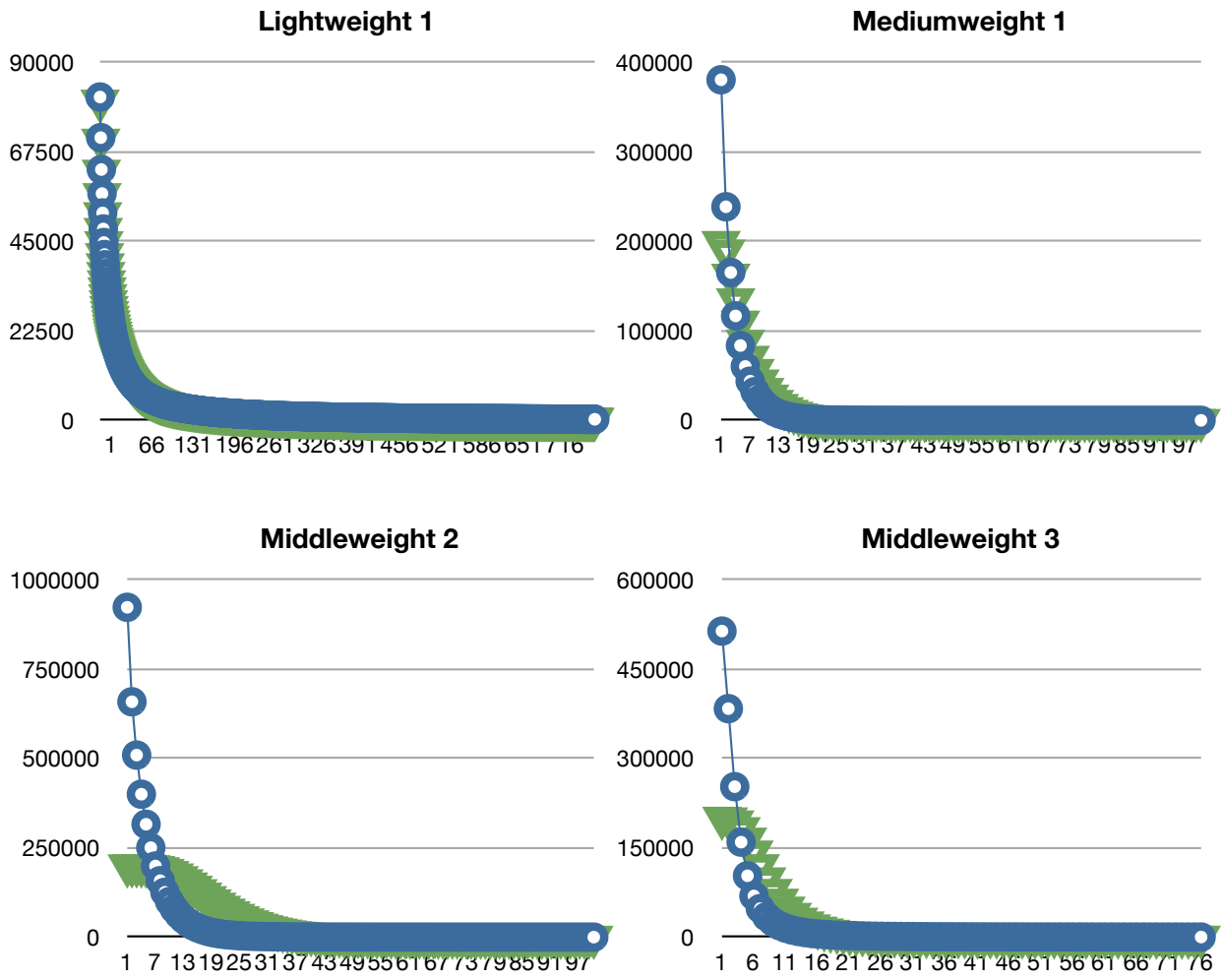
Είναι σημαντικό ότι το παραπάνω μέγεθος είναι σταθερό και δεν επηρεάζεται από το πόσα στοιχεία τελικά θα διαβαστούν από την εφαρμογή. Το μόνο που θα αλλάζει είναι η ακρίβεια των δεδομένων, όμως η χωρική πολυπλοκότητα θα παραμένει σταθερή και ίση με την τιμή που υπολογίστηκε παραπάνω για τις συγκεκριμένες παραμέτρους.

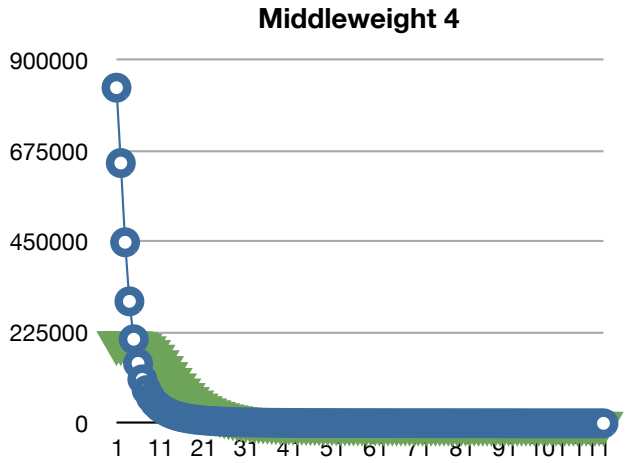
Στατικός Πίνακας Τυχαίου Μεγέθους

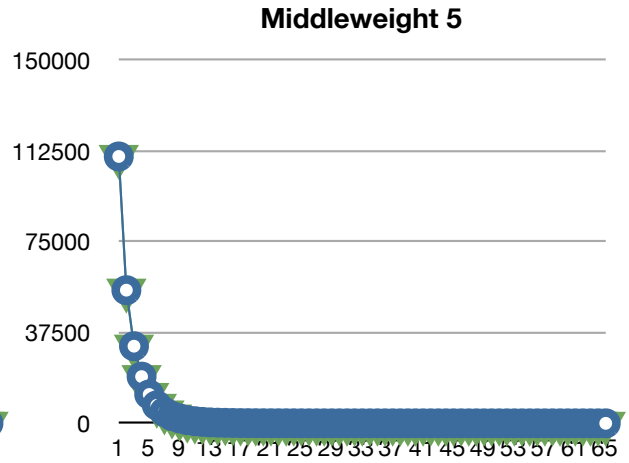
Εισαγωγή

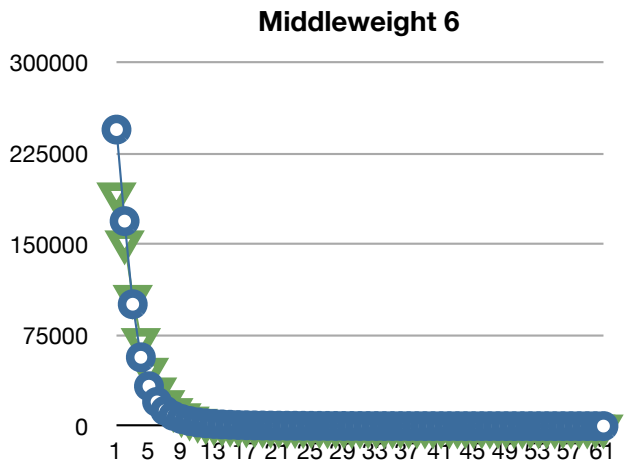
Η επιλογή των παραπάνω παραμέτρων για την υλοποίηση του Count Min αλγόριθμου οδήγησε στα παρακάτω αποτελέσματα.

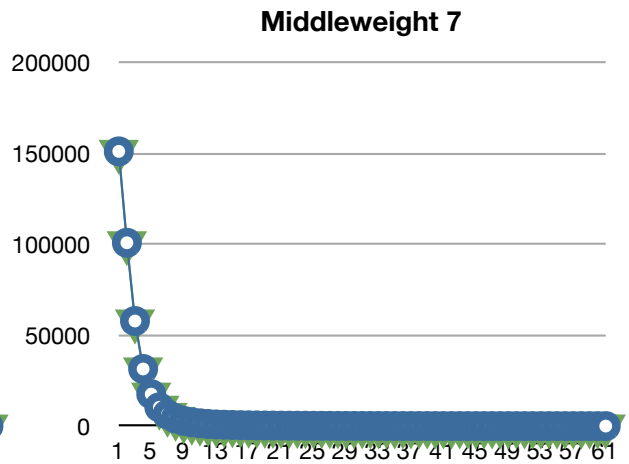
Μετρήσεις

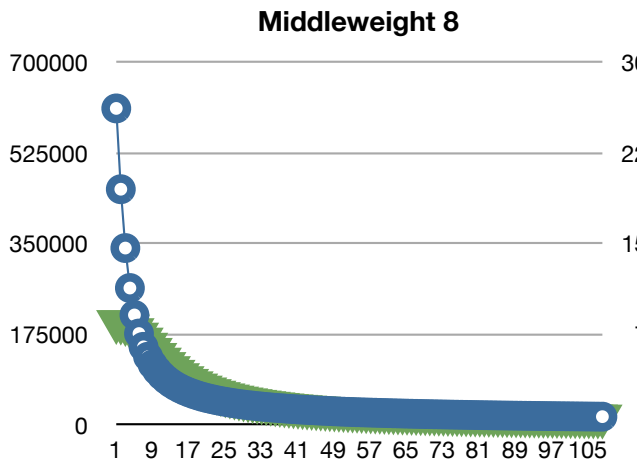


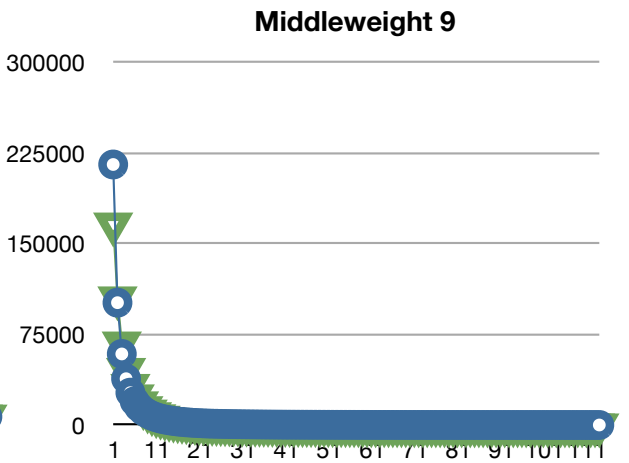

Middleweight 4
900000
675000
450000
225000
0
1 11 21 31 41 51 61 71 81 91 101 111

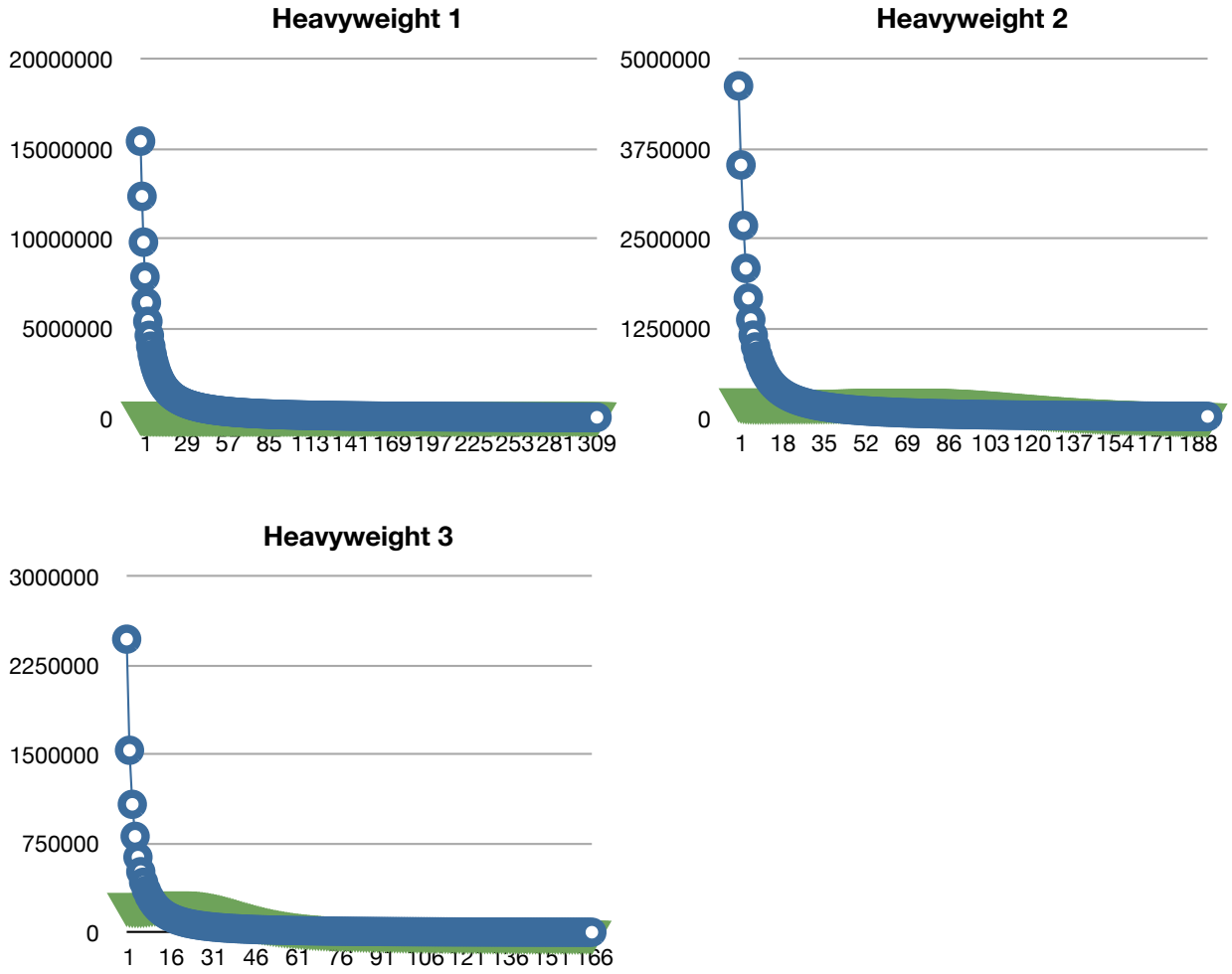

Middleweight 5
150000
112500
75000
37500
0
1 5 9 13 17 21 25 29 33 37 41 45 49 53 57 61 65


Middleweight 6
300000
225000
150000
75000
0
1 5 9 13 17 21 25 29 33 37 41 45 49 53 57 61

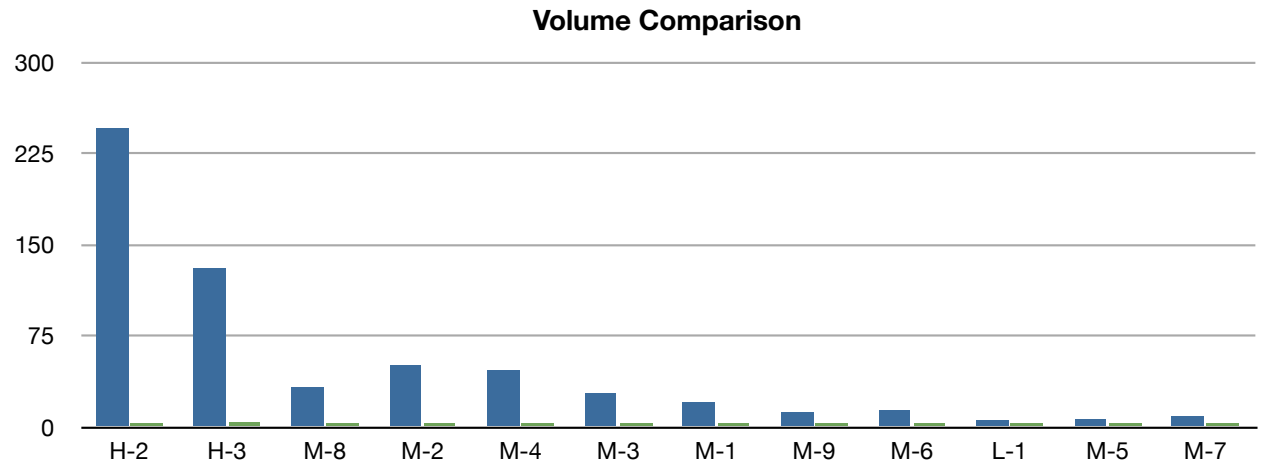

Middleweight 7
200000
150000
100000
50000
0
1 5 9 13 17 21 25 29 33 37 41 45 49 53 57 61


Middleweight 8
700000
525000
350000
175000
0
1 9 17 25 33 41 49 57 65 73 81 89 97 105


Middleweight 9
300000
225000
150000
75000
0
1 11 21 31 41 51 61 71 81 91 101 111



Σχήμα 18: Γραφικές παραστάσεις, για την περίπτωση στατικού πίνακα, της σχέσης μεταξύ in-degree και αντίστοιχου πλήθους κόμβων για τα διάφορα data sets που δοκιμάστηκαν - με “μπλε” εμφανίζονται τα πραγματικά αποτελέσματα, που προκύπτουν από την benchmark υλοποίηση και με “πράσινο” εμφανίζονται τα αποτελέσματα που προέκυψαν από την Count Min υλοποίηση



Σχήμα 19: Σύγκριση, για την περίπτωση στατικού πίνακα, του χώρου που απαιτεί η benchmark υλοποίηση σε σχέση με την Count Min - με "μπλε" εμφανίζονται τα πραγματικά αποτελέσματα, που προκύπτουν από την benchmark υλοποίηση και με "πράσινο" εμφανίζονται τα αποτελέσματα που προέκυψαν από την Count Min υλοποίηση

	Bottom 99% average abs(error) (%)	Top 1% average abs(error) (%)	Percentage of nodes with larger than expected error (%)	#edges / #columns	#nodes / #columns	Space used by benchmark (MB)	Space used by count min algorithm (MB)
Heavyweight 1	24471.0	290.0	0.33	1005.87	77.23	-	3.3
Heavyweight 2	6071.0	80.0	1.36	249.57	23.16	245.4	3.1
Heavyweight 3	2337.0	52.0	0.90	66.43	12.37	130.4	3.5
Middleweight 8	2276.0	2.0	5.42	58.47	3.06	32.9	2.9
Middleweight 2	779.0	60.0	0.17	20.21	4.62	50.6	2.6
Middleweight 4	592.0	33.0	0.66	15.86	4.16	46.4	2.6
Middleweight 3	291.0	73.0	0.83	8.57	2.57	27.9	2.6
Middleweight 1	184.0	15.0	0.73	6.08	1.90	19.8	2.6
Middleweight 9	93.0	4.0	2.36	2.82	1.08	12.3	2.7
Middleweight 6	72.0	6.0	0.94	3.36	1.22	13.6	2.6
Lightweight 1	7.0	0.0	0.10	12.10	0.41	5.1	3.1
Middleweight 5	0.0	0.0	0.00	1.25	0.55	6.4	2.6
Middleweight 7	0.0	0.0	0.00	1.97	0.76	8.3	2.6

Πίνακας 1: Πίνακας μετρήσεων, για την περίπτωση στατικού πίνακα, σημαντικών δεικτών σχετικά με την ακρίβεια των in-degrees για τα διάφορα data sets που δοκιμάστηκαν (το γεγονός ότι σε κάποια κελιά υπάρχει παύλα στη θέση της τιμής οφείλεται στη δυσκολία υπολογισμού της μνήμης που καταλαμβάνουν υπερβολικά μεγάλα data sets)

Στον παραπάνω πίνακα ως “average abs(error) (%)” θεωρείται ο μέσος όρος του ποσοστιαίου απόλυτου σφάλματος μεταξύ των τιμών in-degree που προκύπτουν από την Count Min υλοποίηση για κάθε κόμβο και των αντίστοιχων πραγματικών τιμών που έχουν υπολογιστεί με την benchmark υλοποίηση (αυτό σημαίνει ότι η τιμή 0.0 αποτελεί τη βέλτιστη ακρίβεια). Οι γραμμές είναι ταξινομημένες κατά φθίνουσα σειρά του “Bottom 99% average abs(error) (%)”, το οποίο αποτελεί και το βασικό δείκτη ακρίβειας των αποτελεσμάτων.

Παρατηρήσεις

Τα παραπάνω δεδομένα οδηγούν σε ορισμένες σημαντικές παρατηρήσεις.

Space

Το πρώτο στοιχείο που παρατηρούμε είναι ότι στην Count Min υλοποίηση, το hash-map στο οποίο αποθηκεύουμε το πλήθος nodes με in-degree μεγαλύτερο ή ίσο του X έχει πράγματι μέγεθος αμελητέο σε σχέση με το συνολικό χώρο που καταλαμβάνει η εφαρμογή. Στον παραπάνω πίνακα φαίνεται ότι στην πλειοψηφία των περιπτώσεων ο χώρος που απαιτείται είναι 2.6 MB, με μικρή διακύμανση σε περιπτώσεις με εξωπραγματικά μεγάλο max in-degree που οδηγεί σε ανάγκη για μεγάλο hash-map. Ακόμη όμως και σ’ αυτήν την περίπτωση, η αύξηση του χώρου δεν είναι σημαντική. Παρ’ όλ’ αυτά, όπως αναφέρθηκε και παραπάνω, με αντικατάσταση της hash-map δομής με ένα δεύτερο Count Min πίνακα μπορούμε να εξασφαλίσουμε ότι η υλοποίηση θα απαιτεί σταθερό χώρο σε κάθε περίπτωση.

Μία δεύτερη ενδιαφέρουσα παρατήρηση είναι ότι ακόμη και στην περίπτωση του “μικρού” (lightweight) data set, το οποίο διαθέτει περίπου 80.000 κόμβους και άρα αντίστοιχο πλήθος στοιχείων στη hash-map δομή της benchmark υλοποίησης, ο 3x200.000 πίνακας της count-min υλοποίησης παραμένει “οικονομικότερος” από άποψη χώρου. Αυτό σημαίνει ότι η χρήση 600.000 ακέραιων μετρητών εξακολουθεί να καταλαμβάνει λιγότερο χώρο απ’ ότι μια hash-map δομή 80.000 στοιχείων. Ο λόγος είναι ότι προκειμένου να εξασφαλίζεται η πραγματοποίηση των queries εισαγωγής νέου στοιχείου και αναζήτησης στοιχείου σε σταθερό χρόνο, η hash-map δομή είναι υλοποιημένη με κατάλληλο τρόπο που απαιτεί χώρο μεγαλύτερο απ’ αυτό ενός απλού integer για κάθε στοιχείο της.

Error

Στον παραπάνω πίνακα φαίνεται για κάθε data set το ποσοστό των κόμβων του αντίστοιχου γράφου των οποίων το in-degree που υπολογίζεται στην Count Min υλοποίηση απέχει από την πραγματική τιμή του κατά σφάλμα μεγαλύτερο αυτού που προβλέπεται από τη θεωρία, όπως αυτή αναλύθηκε σε προηγούμενο κεφάλαιο. Με βάση την επιλογή των παραμέτρων για τη συγκεκριμένη υλοποίηση (ο Count Min πίνακας αποτελείται από 3 γραμμές), αναμένεται το συγκεκριμένο ποσοστό να μην υπερβαίνει το 5%. Κάτι τέτοιο πράγματι ισχύει αφού μόνο σε μια περίπτωση το εν λόγω ποσοστό ξεπερνά την προβλεπόμενη τιμή, όμως ακόμη και τότε την ξεπερνά οριακά.

Accuracy

Από τις παραπάνω μετρήσεις γίνεται εύκολα αντιληπτό ότι η επιλογή των συγκεκριμένων παραμέτρων καθιστά τη συγκεκριμένη υλοποίηση περισσότερο ή λιγότερο κατάλληλη ανάλογα με το data set στο οποίο εφαρμόζεται. Δεν μπορούμε λοιπόν να "εμπιστευτούμε" τις συγκεκριμένες παραμέτρους για κάθε δυνατό data set.

Παράλληλα, παρατηρούμε στον παραπάνω πίνακα ότι από το σύνολο των κόμβων, το υποσύνολο που περιέχει το 1% αυτών με το μεγαλύτερο in-degree παρουσιάζει σαφώς καλύτερη ακρίβεια συγκριτικά με το υπόλοιπο 99%. Κάτι τέτοιο οφείλεται στο γεγονός ότι τα data sets που χρησιμοποιούμε αποτελούν γράφους που αναπαριστούν κοινωνικά δίκτυα και ακολουθούν Power Law Distribution. Αυτό σημαίνει ότι η πλειοψηφία των κόμβων έχει πολύ μικρό in-degree το οποίο σε πολλές περιπτώσεις (στα συγκεκριμένα data sets) δεν ξεπερνά ούτε καν τις μονοψήφιες τιμές. Έτσι, ακόμη και μονοψήφια σφάλματα μπορούν να οδηγήσουν σε δύο ή και περισσότερες φορές, μεγαλύτερα από τα αναμενόμενα, αποτελέσματα. Συνεπώς, το γεγονός ότι οι κοινωνικοί γράφοι ακολουθούν τέτοια κατανομή οδηγεί στην ανάγκη για εξαιρετικά "μικρά" σφάλματα, ώστε να μην επηρεάζονται οι "μικρές" τιμές της πλειοψηφίας των κόμβων τους. Από τον τύπο του σφάλματος (που επιβεβαιώνεται πειραματικά από το γεγονός ότι όσο μικρότερη η σχέση μεταξύ πλήθους ακμών και πλήθους στηλών, η οποία εμφανίζεται στον τύπο του σφάλματος, τόσο καλύτερη η ακρίβεια των αποτελεσμάτων) κάτι τέτοιο σημαίνει ότι θα πρέπει το πλήθος των στηλών να αυξηθεί σημαντικά και μάλιστα να προσεγγίσει επίπεδα έως και τρεις φορές του πλήθους των ακμών σε κάθε περίπτωση. Κάτι τέτοιο βέβαια θα οδηγούσε σε εξαιρετικά μεγάλους πίνακες και θα εξαφάνιζε τα όποια πλεονεκτήματα παρουσιάζει η χρήση της μεθόδου Count Min.

Στις παραπάνω γραφικές παραστάσεις φαίνεται να υπάρχει κάποιος περιορισμός στο μέγιστο πλήθος κόμβων που μπορεί να υπολογίσει η υλοποίηση για κάποιο in-degree. Συγκεκριμένα, είναι ελάχιστα τα data sets στα οποία ο αλγόριθμος εμφανίζει τιμές μεγαλύτερες από 200.000 και ακόμη και σε όσα το συγκεκριμένο όριο ξεπερνιέται κάτι τέτοιο είναι οριακό. Ο λόγος είναι ότι, όπως εξηγήθηκε στην περιγραφή του αλγορίθμου, το πλήθος κόμβων στο hash-map με in-degree ίσο με X ισούται με το πλήθος των μεταβάσεων από X-1 σε X της τιμής κάποιου κελιού στον πίνακα Count Min (και μάλιστα όχι οποιουδήποτε κελιού αλλά αυτού με τη μικρότερη τιμή μεταξύ της τριάδας που αντιστοιχεί στον κόμβο που ευθύνεται για τη συγκεκριμένη αλλαγή). Συμπεραίνουμε λοιπόν ότι το όριο 200.000 δεν είναι τυχαίο καθώς τόσο είναι το πλήθος των στηλών στη συγκεκριμένη υλοποίηση. Στις περιπτώσεις στις οποίες το όριο αυτό ξεπερνιέται, αυτό οφείλεται στη δημιουργία collisions σε κάποια από τις γραμμές του πίνακα που "αφήνει" ανέγγιχτα τα αντίστοιχα κελιά για χρήση (και αντίστοιχη μεταβολή του hash-map) από άλλους κόμβους. Σε κάθε περίπτωση όμως είναι αδύνατον οι τιμές αυτές να ξεπεράσουν κατά πολύ το όριο που δημιουργείται από το πλήθος των στηλών. Στο συγκεκριμένο χαρακτηριστικό οφείλεται το γεγονός ότι στις παραπάνω γραφικές παραστάσεις τα πολύ "μικρά" in-degrees εμφανίζονται με λιγότερο του αναμενόμενου, πλήθος κόμβων ενώ τα αμέσως επόμενα (τα οποία στην ουσία λαμβάνουν την αύξηση που θα έπρεπε να έχουν λάβει τα "μικρά") εμφανίζονται με πλήθος κόμβων μεγαλύτερο του πραγματικού. Στην πραγματικότητα, η false positive ιδιότητα του Count Min Sketch οδηγεί, κατά την ανάγνωση και προσθήκη μιας νέας ακμής στον Count Min πίνακα, στη λανθασμένη ανανέωση ενός μεγαλύτερου του σωστού στοιχείου του hash-map.

Η παραπάνω παρατήρηση συμβαδίζει με ένα ακόμη στοιχείο που προκύπτει από τον πίνακα. Αυτό είναι ότι η φθίνουσα ταξινόμηση αυτού με βάση την ακρίβεια του 99% των in-degrees των κόμβων φαίνεται να ακολουθείται από μία επίσης "σχεδόν" φθίνουσα ακολουθία των σχέσεων

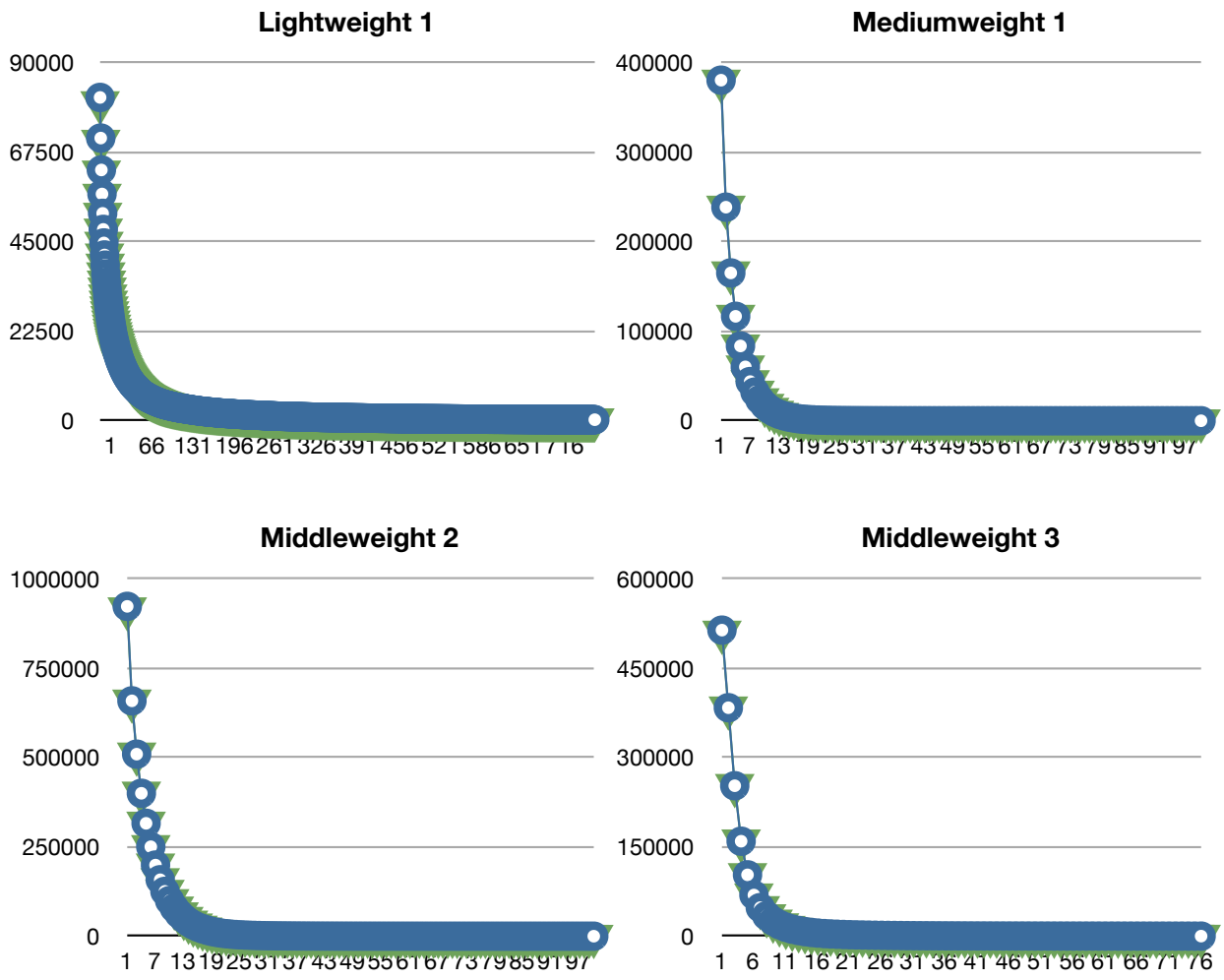
μεταξύ πλήθους κόμβων και πλήθους στηλών. Μάλιστα με βάση τα στοιχεία του πίνακα, η βέλτιστη σχέση των δύο αυτών μεγεθών φαίνεται να είναι κάποια τιμή κοντά στο 0.65.

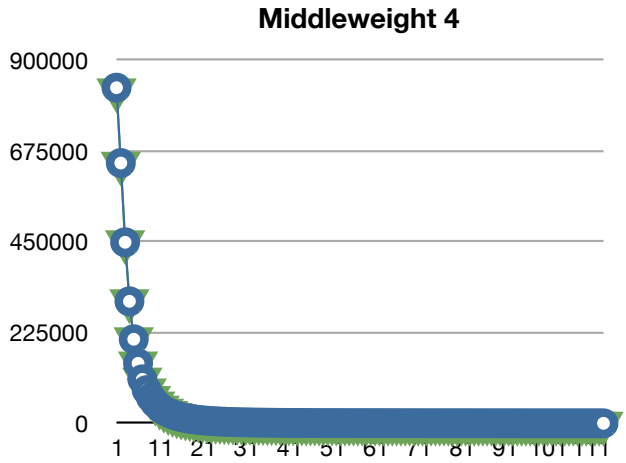
Προσέγγιση Ιδανικού Μεγέθους Στατικού Πίνακα

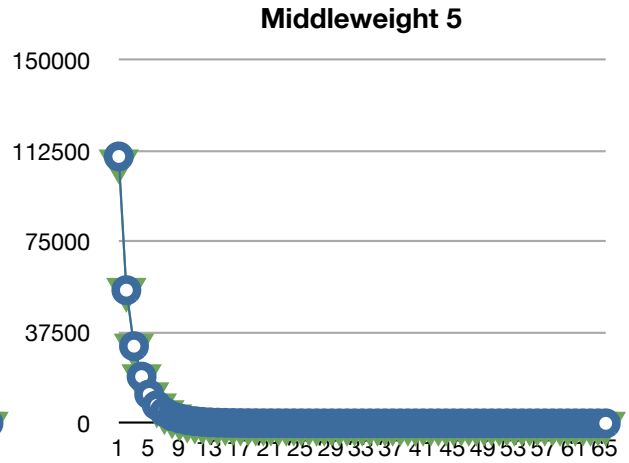
Περιγραφή

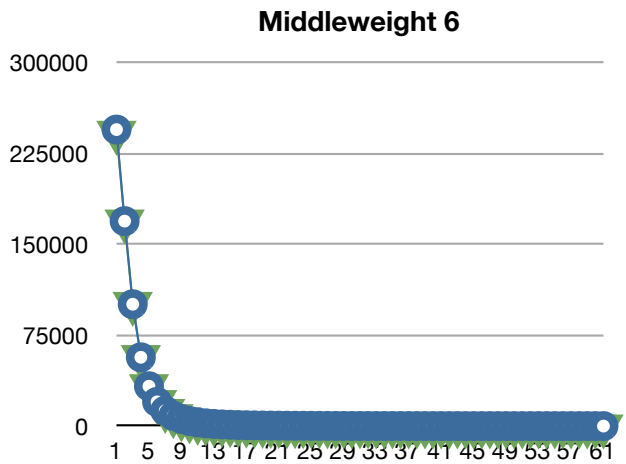
Στις παραπάνω μετρήσεις, παρατηρήθηκε ότι, όπως αναμενόταν, το απαιτούμενο πλήθος στηλών του Count Min πίνακα για καλή ακρίβεια στα αποτελέσματα διαφέρει ανάλογα με το μέγεθος του data set. Για το σκοπό αυτό, πραγματοποιήθηκαν πειράματα ώστε να βρεθεί για κάθε data set το ελάχιστο πλήθος στηλών που αποφέρει τη βέλτιστη ακρίβεια. Οι συγκεκριμένες μετρήσεις πάρθηκαν με σκοπό την ανεύρεση κάποιας σχέσης μεταξύ του πλήθους των στηλών και του μεγέθους του data set. Τα αποτελέσματα φαίνονται παρακάτω.

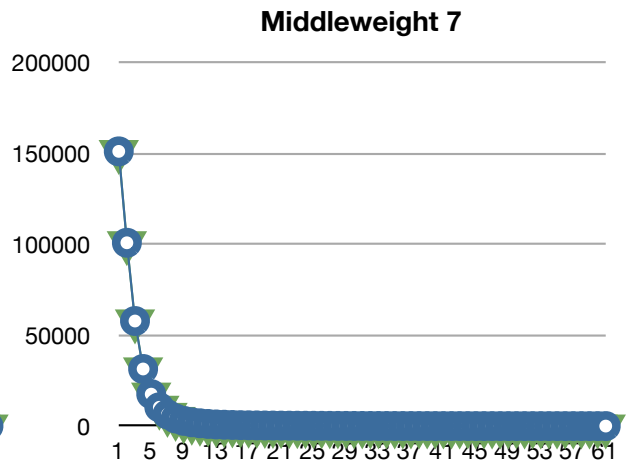
Μετρήσεις

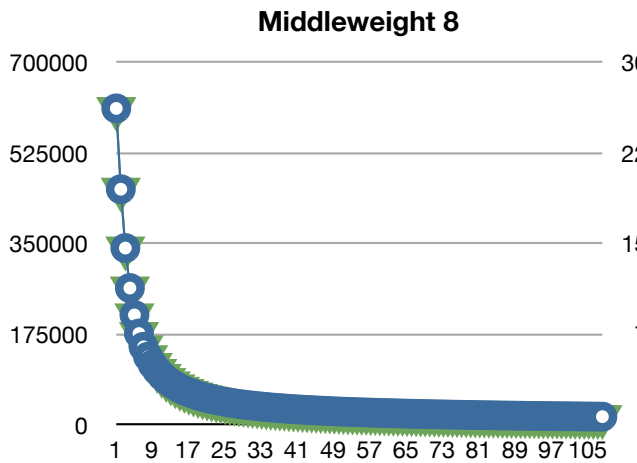


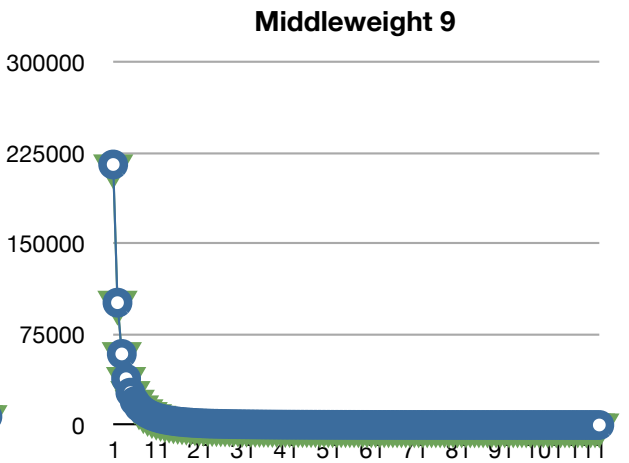

Middleweight 4
900000
675000
450000
225000
0
1 11 21 31 41 51 61 71 81 91 101 111

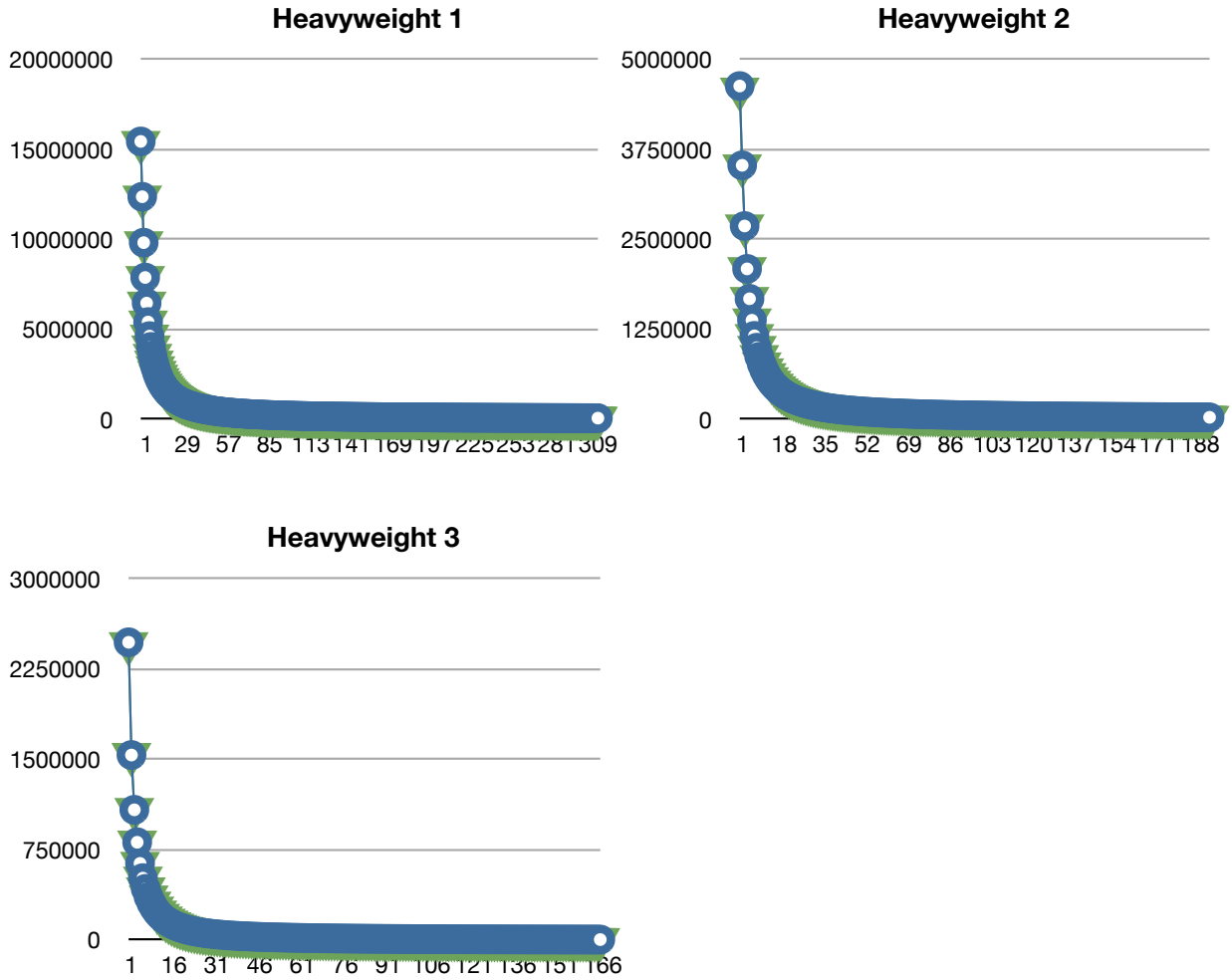

Middleweight 5
150000
112500
75000
37500
0
1 5 9 13 17 21 25 29 33 37 41 45 49 53 57 61 65


Middleweight 6
300000
225000
150000
75000
0
1 5 9 13 17 21 25 29 33 37 41 45 49 53 57 61

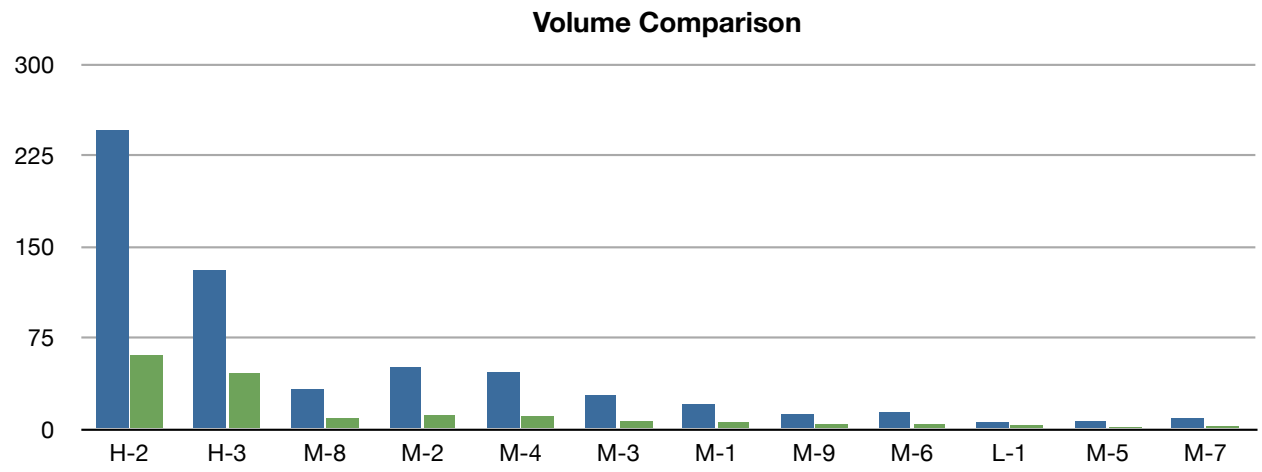

Middleweight 7
200000
150000
100000
50000
0
1 5 9 13 17 21 25 29 33 37 41 45 49 53 57 61


Middleweight 8
700000
525000
350000
175000
0
1 9 17 25 33 41 49 57 65 73 81 89 97 105


Middleweight 9
300000
225000
150000
75000
0
1 11 21 31 41 51 61 71 81 91 101 111



Σχήμα 20: Γραφικές παραστάσεις, για την περίπτωση βέλτιστης επιλογής πίνακα, της σχέσης μεταξύ in-degree και αντίστοιχου πλήθους κόμβων για τα διάφορα data sets που δοκιμάστηκαν - με "μπλε" εμφανίζονται τα πραγματικά αποτελέσματα, που προκύπτουν από την benchmark υλοποίηση και με "πράσινο" εμφανίζονται τα αποτελέσματα που προέκυψαν από την Count Min υλοποίηση



Σχήμα 21: Σύγκριση, για την περίπτωση βέλτιστης επιλογής πίνακα, του χώρου που απαιτεί η benchmark υλοποίηση σε σχέση με την Count Min - με “μπλε” εμφανίζονται τα πραγματικά αποτελέσματα, που προκύπτουν από την benchmark υλοποίηση και με “πράσινο” εμφανίζονται τα αποτελέσματα που προέκυψαν από την Count Min υλοποίηση

	Bottom 99% average abs(error) (%)	Top 1% average abs(error) (%)	Percentage of nodes with larger than expected error (%)	Bottom 99% average in-degree	#edges / #columns	#nodes / #columns	Space used by benchmark (MB)	Space used by count min (MB)
H-1	7.0	0.0	0.08	7	11.85	0.91	-	-
H-2	7.0	0.0	0.04	6	9.51	0.88	245.4	60.8
H-3	1.0	0.0	0.00	4	3.41	0.63	130.4	45.8
M-8	4.0	0.0	0.02	10	16.13	0.84	32.9	8.9
M-2	2.0	0.0	0.03	4	4.21	0.96	50.6	11.3
M-4	3.0	0.0	0.05	4	3.69	0.97	46.4	10.2
M-3	3.0	0.0	0.04	3	3.17	0.95	27.9	6.5
M-1	1.0	0.0	0.01	3	2.96	0.93	19.8	5.0
M-9	3.0	0.0	0.08	2	1.95	0.74	12.3	2.7
M-6	3.0	0.0	0.04	3	2.53	0.92	13.6	3.4
L-1	9.0	0.0	0.10	22	13.45	0.45	5.1	2.9
M-5	5.0	0.0	0.09	2	2.05	0.90	6.4	1.7
M-7	0.0	0.0	0.00	3	2.32	0.89	8.3	2.3

Πίνακας 2: Πίνακας μετρήσεων, για την περίπτωση βέλτιστης επιλογής πίνακα, σημαντικών δεικτών σχετικά με την ακρίβεια των in-degrees για τα διάφορα data sets που δοκιμάστηκαν

Παρατηρήσεις

Οι παρατηρήσεις που προκύπτουν από τα παραπάνω δεδομένα εξηγούνται παρακάτω.

Space

Από τις παραπάνω μετρήσεις προκύπτει ότι ακόμη και στην περίπτωση που ρυθμίσουμε κατάλληλα το πλήθος των στηλών στην Count Min υλοποίηση, αυτή εξακολουθεί να καταλαμβάνει σημαντικά λιγότερο χώρο σε σχέση με την benchmark υλοποίηση.

Error

Από τον παραπάνω πίνακα γίνεται φανερό ότι σε καμία περίπτωση το ποσοστό των κόμβων με απόκλιση μεγαλύτερη του αναμενόμενου σφάλματος, δεν ξεπερνά το προβλεπόμενο, από την επιλογή παραμέτρων, 5%.

Accuracy

Όπως φαίνεται, τόσο από τις γραφικές παραστάσεις όσο και από τα στοιχεία του πίνακα, η ακρίβεια των αποτελεσμάτων σ' αυτήν την περίπτωση είναι ιδιαίτερα υψηλή, όπως φυσικά αναμενόταν αφού οι παράμετροι επιλέχθηκαν με αυτόν ακριβώς το στόχο.

Παράλληλα, στον τελευταίο πίνακα, έχει προστεθεί μία επιπλέον στήλη σε σχέση με τον πίνακα των προηγούμενων μετρήσεων. Πρόκειται για την τέταρτη στήλη στην οποία αναγράφεται για κάθε data set, το μέσο in-degree του 99% των nodes (εξαιρείται το 1% με τις "εξωπραγματικά" μεγάλες τιμές).

Παρατηρούμε ότι παρά το γεγονός ότι το σύνολο των data sets παρουσιάζει "καλή" ακρίβεια, ο λόγος μεταξύ του πλήθους ακμών και του αριθμού των στηλών διαφέρει μεταξύ αυτών. Κάτι τέτοιο έρχεται σε αντίθεση με τη θεωρία που προβλέπει ότι το μέσο σφάλμα εξαρτάται από το συγκεκριμένο λόγο πολλαπλασιαζόμενο με τη σταθερά e (η οποία περιγράφηκε αναλυτικά σε προηγούμενο κεφάλαιο). Παρ' όλ' αυτά, παρατηρώντας τις συγκεκριμένες δύο στήλες του πίνακα (αυτήν του μέσου in-degree και αυτήν της σχέσης μεταξύ ακμών και στηλών) φαίνεται να ακολουθούν παρόμοιες μεταβολές. Συγκεκριμένα, εξαιρουμένων ελάχιστων περιπτώσεων, σε όλα τα data sets η βέλτιστη αναλογία ακμών και στηλών προκύπτει ίση με τη μέση τιμή του in-degree των κόμβων του αντίστοιχου γράφου. Αυτή η παρατήρηση είναι ιδιαίτερα σημαντική καθώς σημαίνει ότι αν υπάρχει εκ των προτέρων γνώση του πλήθους των ακμών και του μέσου in-degree του γράφου, μπορεί να γίνει μία πολύ καλή εκτίμηση του βέλτιστου πλήθους στηλών. Όσον αφορά τις "λίγες" περιπτώσεις στις οποίες οι δύο τιμές δείχνουν να μην είναι ακριβώς ίσες, παρατηρούμε ότι μόνο σε μία περίπτωση η μέση τιμή in-degree είναι μεγαλύτερη από την αναλογία ακμών-στηλών. Αυτό σημαίνει ότι, εκτός αυτής, σε όλες τις άλλες περιπτώσεις ακόμη κι αν δεν είναι ίσες, μπορούμε και πάλι να βασιστούμε στην τιμή του μέσου όρου in-degree για τον προσδιορισμό του πλήθους στηλών ώστε να επιτευχθεί η βέλτιστη ακρίβεια. Το μοναδικό μειονέκτημα είναι ότι σ' αυτές τις περιπτώσεις θα δεσμευθεί περισσότερος χώρος απ' όσος χρειάζεται. Η ακρίβεια όμως θα διατηρηθεί στα αναμενόμενα επίπεδα και σε αυτήν την περίπτωση.

Ακόμη, εξηγείται γιατί, όπως αναφέρθηκε παραπάνω, οι μετρήσεις έρχονται σε αντίθεση με τη θεωρία περί σφαλμάτων. Ο λόγος είναι ότι η ακρίβεια των αποτελεσμάτων εξαρτάται, όχι από το απόλυτο μέγεθος του μέσου σφάλματος, αλλά από το μέγεθος αυτού σε σχέση με τη μέση τιμή του συνόλου των κόμβων. Δηλαδή, στην περίπτωση που το μέσο in-degree είναι "μικρό" απαιτείται

“μεγαλύτερη” ακρίβεια, αφού ακόμη κι ένα “μικρό” σφάλμα είναι δυνατόν να επηρεάσει σημαντικά τα αποτελέσματα. Συνεπώς, οι μετρήσεις δεν αντιτίθενται στη θεωρία, απλώς το ύψος του επιτρεπόμενου σφάλματος διαφέρει ανάλογα με τη μέση τιμή της πλειοψηφίας των κόμβων.

Μία ακόμη παρατήρηση είναι ότι η αναλογία μεταξύ πλήθους κόμβων και πλήθους στηλών δείχνει να είναι σταθερή (πλην μίας περίπτωσης που καθώς είναι μοναδική μπορεί να θεωρηθεί εξαίρεση). Αυτό σημαίνει ότι αν θέλουμε να πετύχουμε βέλτιστη ακρίβεια, η σχέση μεταξύ κόμβων και στηλών θα πρέπει να κυμαίνεται γύρω από την τιμή 0.85 (μέση τιμή της αναλογίας, όπως αυτή προκύπτει από τον παραπάνω πίνακα), δηλαδή λίγο κάτω απ’ τη μονάδα.

Αν θεωρήσουμε ότι

$$\#nodes \simeq \frac{\#edges}{average_{in-degree}}$$

τότε προκύπτει ότι

$$\frac{\#nodes}{\#columns} \simeq \frac{\#edges}{\#columns \times average_{in-degree}}$$

και άρα ότι

$$\frac{\#edges}{\#columns} \simeq \frac{\#nodes}{\#columns} \times 0.85$$

Η τελευταία σχέση συμβαδίζει με την προηγούμενη παρατήρηση, ότι δηλαδή η αναλογία ακμών-στηλών είναι περίπου ίση με το μέσο in-degree (η διαφορά των δύο συμπερασμάτων είναι η σταθερά 0.85 η οποία όμως μπορεί να θεωρηθεί σχεδόν ίση με τη μονάδα). Συνεπώς, μπορούμε να θεωρήσουμε τις δύο παρατηρήσεις (αναλογία ακμών-στηλών περίπου ίση με μέση τιμή in-degree και αναλογία κόμβων-στηλών περίπου ίση με 0.85) ισοδύναμες.

Πίνακας Δυναμικά Μεταβαλλόμενου Μεγέθους - 1η Προσέγγιση

Περιγραφή

Είναι σημαντικό η Count Min υλοποίηση να πετυχαίνει υψηλή ακρίβεια ακόμη κι αν δεν υπάρχει καμία εκ των προτέρων πληροφορία σχετική με τα data sets. Θα ήταν ιδανικό δηλαδή ο αλγόριθμος να δεσμεύει μνήμη δυναμικά όσο αυξάνει το πλήθος ακμών που καταφτάνουν με σκοπό τη διατήρηση της βέλτιστης ακρίβειας. Κάτι τέτοιο εμφανίζει δύο σημαντικές δυσκολίες.

1. <u>Με ποια διαδικασία θα επιτυγχάνεται η αύξηση του μεγέθους του πίνακα;</u>
   Στη συγκεκριμένη υλοποίηση ως αύξηση του μεγέθους του πίνακα θεωρείται ο διπλασιασμός του πλήθους των στηλών του που πραγματοποιείται με τη δημιουργία ενός δεύτερου πίνακα ίδιου μεγέθους.
   Για το σκοπό αυτό χρησιμοποιείται μια linked-list στην οποία αρχικά αποθηκεύεται ο

πρώτος πίνακας και στη συνέχεια προστίθενται οι νέοι πίνακες που δημιουργούνται. Κάθε νέος πίνακας έχει πλήθος στηλών ίσο με το συνολικό πλήθος στηλών όλων των προηγούμενων πινάκων ώστε να επιτευχθεί ο απαραίτητος διπλασιασμός. Καθώς είναι απαραίτητο το σύνολο των πινάκων να μπορεί να προσπελαστεί ως ενιαίος πίνακας, έχουν δημιουργηθεί ειδικές συναρτήσεις για την ανάθεση τιμής στο συνολικό πίνακα, καθώς και για την ανεύρεση της τιμής κάποιου συγκεκριμένου κελιού. Φυσικά, η ύπαρξη linked-list αυξάνει τη χρονική πολυπλοκότητα των queries, όμως καθώς το μέγεθος του πίνακα αυξάνεται εκθετικά μπορεί να θεωρηθεί ότι το πλήθος των πινάκων που μαζί αποτελούν το συνολικό πίνακα δεν θα είναι ποτέ ικανό να αυξήσει σημαντικά τον απαιτούμενο χρόνο των queries. Αναλυτική περιγραφή των συναρτήσεων μπορεί να βρεθεί στο παράρτημα όπου παρατίθεται το σύνολο του κώδικα που χρησιμοποιήθηκε. Φυσικά, όταν ένας πίνακας διπλασιάζεται τίθεται το ερώτημα πώς θα αρχικοποιηθεί ο νέος πίνακας.
Έστω κόμβος με id ίσο με X. Πριν την αύξηση του πλήθους των στηλών, η hash function τον έστελνε στο κελί

$$hash(X)$$

Με την προσθήκη του νέου πίνακα, ο ίδιος κόμβος θα αντιστοιχίζεται πλέον στο κελί

$$hash'(X)$$

Ο λόγος που αλλάζει η τιμή είναι ο όρος

$$mod(\#columns)$$

που υπάρχει στον τύπο όλων των hash functions. Καθώς σε κάθε αύξηση ο νέος πίνακας που προκύπτει έχει πλήθος στηλών ακριβώς διπλάσιο του προηγούμενου, συμπεραίνουμε ότι για τις δύο παραπάνω τιμές θα ισχύει

$$hash'(X) = hash(X)$$

ή

$$hash'(X) = hash(X) + \#columns$$

Φυσικά δεν υπάρχει κανένας τρόπος να γίνει γνωστό ποια από τις δύο ισότητες ισχύει για κάθε κόμβο. Επίσης, σε κάθε κελί του αρχικού πίνακα μπορεί να υπάρχει, όπως είναι γνωστό, πληροφορία για περισσότερους από έναν κόμβους. Κατά την αύξηση του πλήθους στηλών είναι δυνατόν κάποιοι απ’ αυτούς τους κόμβους να εξακολουθούν να αντιστοιχίζονται στο ίδιο κελί (πρώτη ισότητα) ενώ κάποιοι άλλοι να αντιστοιχίζονται πλέον στο κατάλληλο κελί του νέου πίνακα (δεύτερη ισότητα). Καθώς τη στιγμή του διπλασιασμού ο πίνακας έχει “καλή” ακρίβεια, που σημαίνει ότι για κάθε κόμβο τα αντίστοιχα κελιά διατηρούν μια αρκετά καλή εκτίμηση του in-degree του, θα πρέπει τουλάχιστον οι ήδη υπάρχοντες κόμβοι να διατηρούν αυτήν την ακρίβεια και μετά την

αύξηση. Συνεπώς, θα πρέπει το in-degree τους που προκύπτει από τον Count Min πίνακα να είναι το ίδιο τόσο πριν όσο και μετά την αύξηση. Για το λόγο αυτό, στη συγκεκριμένη υλοποίηση θα θεωρήσουμε ότι κάθε φορά που δημιουργείται ένας νέος πίνακας, ο πίνακας αυτός θα αρχικοποιείται με τέτοιον τρόπο ώστε να αποτελεί αντίγραφο του προηγούμενου. Φυσικά, κάτι τέτοιο θα απαιτούσε τετραγωνική πολυπλοκότητα σε κάθε αύξηση. Καθώς όμως ο διπλασιασμός δε συμβαίνει "συχνά", η συγκεκριμένη πολυπλοκότητα δεν καθίσταται απαγορευτική.
Παρ' όλ' αυτά, στην παρούσα υλοποίηση το παραπάνω αποφεύγεται. Αντί να αρχικοποιείται ο νέος πίνακας κατά τη δημιουργία του, η συγκεκριμένη διαδικασία γίνεται με "lazy" τρόπο. Μετά την αύξηση, κάθε φορά που γίνεται ένα query (είτε αυτό είναι update query είτε είναι find query), αν για την απάντησή του χρειαστεί η τιμή κάποιου κελιού του τελευταίου δημιουργημένου πίνακα, η οποία είναι ακόμη μηδενική-μη αρχικοποιημένη, τότε επιστρέφεται η τιμή του κελιού στο οποίο θα αντιστοιχιζόταν ο συγκεκριμένος κόμβος πριν το διπλασιασμό και ταυτόχρονα γίνεται και η αρχικοποίηση του νέου κελιού.
Μ' αυτή τη διαδικασία είναι δυνατόν να μην πραγματοποιηθεί ποτέ η πλήρης αρχικοποίηση του νέου πίνακα (αν για παράδειγμα χρειαστεί να γίνει νέα αύξηση χωρίς να έχει γίνει query σε όλα τα κελιά του πίνακα), όμως αποφεύγεται η τετραγωνική πολυπλοκότητα της αρχικής προσέγγισης.

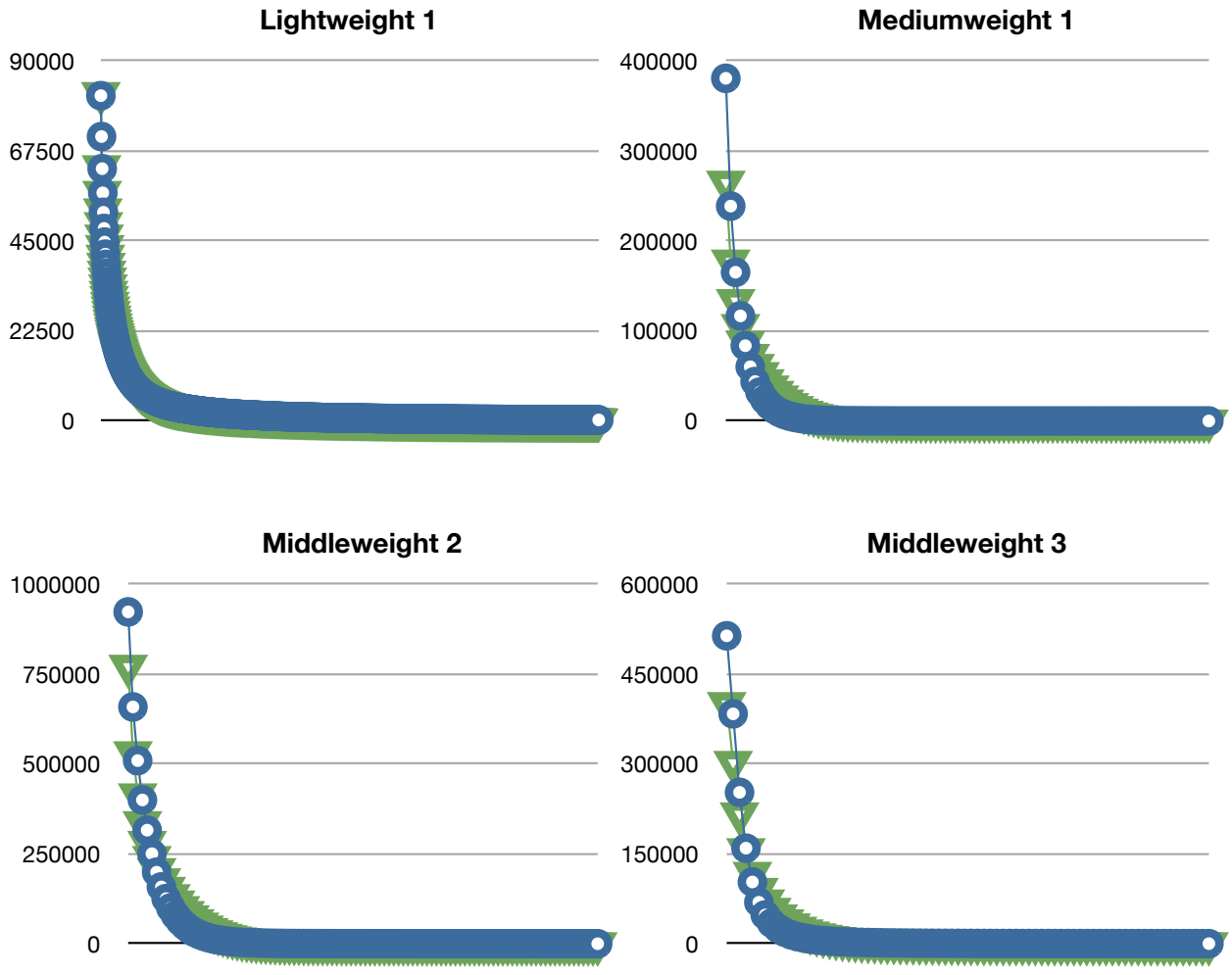
2. <u>Με ποιο κριτήριο θα "πυροδοτείται" μια τέτοια αύξηση;</u>
Με βάση τις παρατηρήσεις των προηγούμενων μετρήσεων, υπάρχουν δύο κριτήρια που θα μπορούσαν να χρησιμοποιηθούν.
Το ένα είναι να γίνεται αύξηση κάθε φορά που η αναλογία κόμβων-στηλών ξεπερνά τον αριθμό 0.85 και το δεύτερο είναι η αύξηση να πραγματοποιείται όταν η αναλογία ακμών-στηλών ξεπερνά τη μέση τιμή του in-degree του γράφου. Και οι δύο συνθήκες όμως παρουσιάζουν προβλήματα.
Στην πρώτη περίπτωση, για τον υπολογισμό της αναλογίας κόμβων-στηλών είναι απαραίτητη η γνώση του πλήθους των διαφορετικών κόμβων που έχουν ήδη διαβαστεί. Η συγκεκριμένη ποσότητα είναι φυσικά ίση με το πλήθος κόμβων με in-degree μεγαλύτερο ή ίσο του 1. Συνεπώς, από το ανάλογο query μπορούμε κάθε φορά να υπολογίζουμε το αντίστοιχο πλήθος. Όπως αναφέρθηκε και παραπάνω, το πλήθος κόμβων με in-degree μεγαλύτερο ή ίσο του 1 ισούται με το πλήθος των φορών στις οποίες έγινε ανανέωση κάποιου κελιού από 0 σε 1. Όμως, από τη στιγμή που σε κάθε αύξηση αρχικοποιούμε το νέο πίνακα ώστε να είναι ίδιος με τον προηγούμενο (ο οποίος προφανώς έχει σχεδόν "γεμίσει", δηλαδή όλα τα κελιά του έχουν τιμή μεγαλύτερη από 0 - γι' αυτό γίνεται η αύξηση εξάλλου), το πλήθος των κελιών με τιμή 0 θα είναι ελάχιστο μόλις μετά τον πρώτο διπλασιασμό. Κάτι τέτοιο σημαίνει ότι το συγκεκριμένο query αναμένεται να έχει κακή ακρίβεια, σημαίνει όμως ακόμη ότι δεν υπάρχει τρόπος να γνωρίζουμε το πλήθος των διαφορετικών κόμβων με καλή ακρίβεια μετά την πρώτη αύξηση του μεγέθους του πίνακα.
Στη δεύτερη περίπτωση, η δυσκολία έγκειται στο γεγονός ότι δεν μπορεί να είναι εκ προοιμίου γνωστή η μέση τιμή του in-degree του γράφου, από τη στιγμή που αυτός δεν έχει καν ακόμη διαβαστεί.

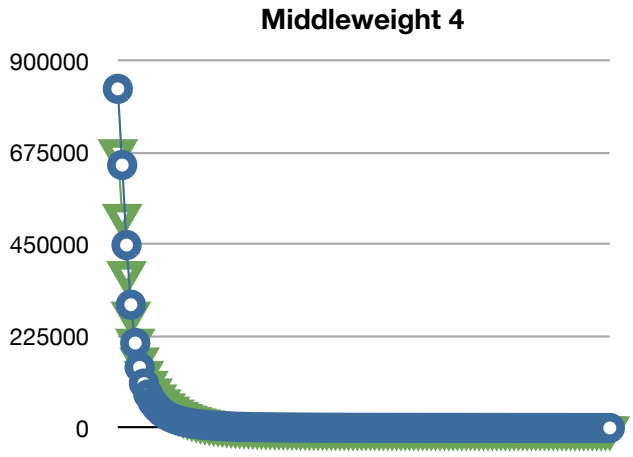
Για τους παραπάνω λόγους, η συγκεκριμένη υλοποίηση χρησιμοποιεί ένα συνδυασμό των παραπάνω κριτηρίων. Συγκεκριμένα, αρχικά χρησιμοποιείται η πρώτη συνθήκη. Αυτό σημαίνει ότι η πρώτη αύξηση πραγματοποιείται όταν η αναλογία κόμβων-στηλών

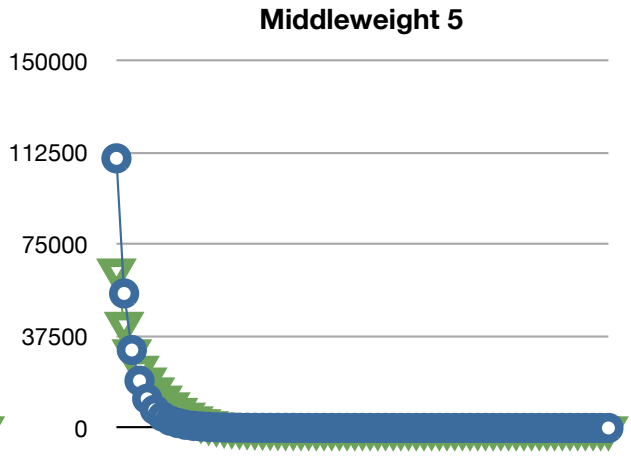
ξεπεράσει τον αριθμό 0.65 (και όχι 0.85 όπως αρχικά είχε εκτιμηθεί - ο λόγος είναι ότι παρατηρώντας τον πίνακα των προηγούμενων μετρήσεων, η ελάχιστη τιμή της σχετικής αναλογίας (πλην μίας περίπτωσης που απέχει αρκετά απ' τις υπόλοιπες) ήταν 0.65 - προτιμούμε να επιλέξουμε την ελάχιστη τιμή καθώς αυτή μας εξασφαλίζει υψηλή ακρίβεια σε κάθε περίπτωση με μοναδικό μειονέκτημα ότι απαιτεί παραπάνω χώρο - καθώς όμως το συγκεκριμένο κριτήριο χρησιμοποιείται μόνο για την πρώτη αύξηση, επιλέγουμε την τιμή που παρέχει την καλύτερη ακρίβεια). Αμέσως πριν απ' αυτήν την αύξηση γίνεται υπολογισμός του, μέχρι εκείνη τη στιγμή, μέσου in-degree του γράφου. Όλες οι επόμενες αυξήσεις γίνονται κάθε φορά που η αναλογία ακμών-στηλών ξεπερνά τη συγκεκριμένη εκτίμηση της μέσης τιμής. Φυσικά, για την επιτυχία της συγκεκριμένης υλοποίησης είναι σημαντικό η αρχική εκτίμηση του μέσου in-degree να είναι ακριβής.

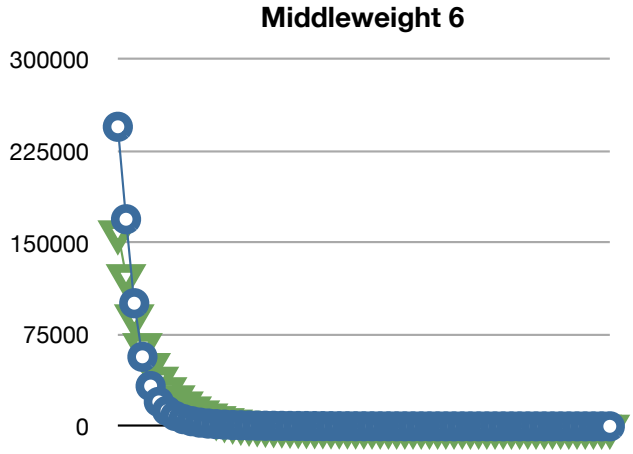
Η παραπάνω ιδέα υλοποιήθηκε και εφαρμόστηκε στο σύνολο των data sets. Ο αρχικός πίνακας ορίστηκε να έχει 3 γραμμές και 10.000 στήλες. Τα αποτελέσματα φαίνονται στις παρακάτω μετρήσεις.

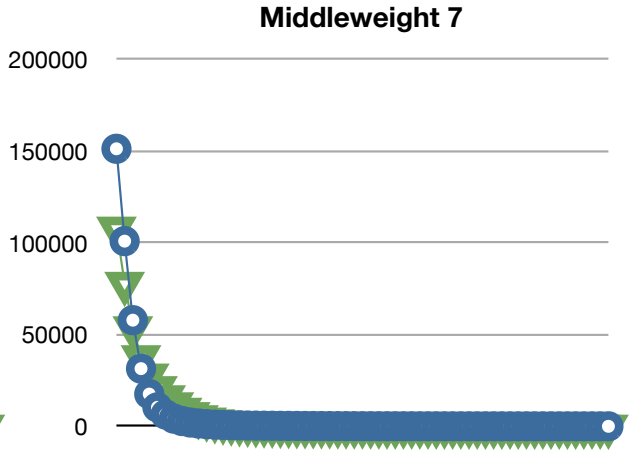
Μετρήσεις

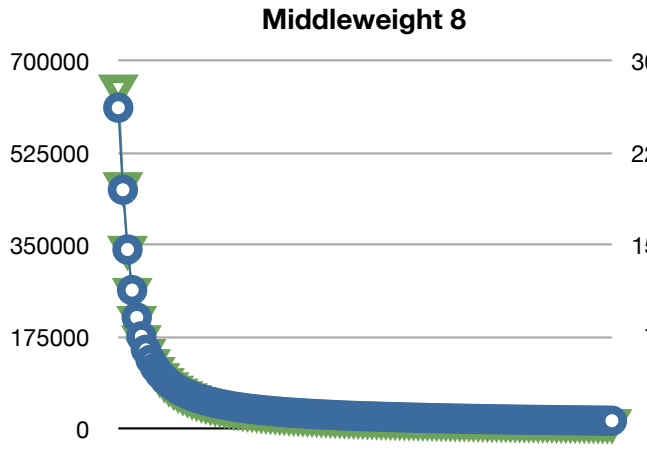


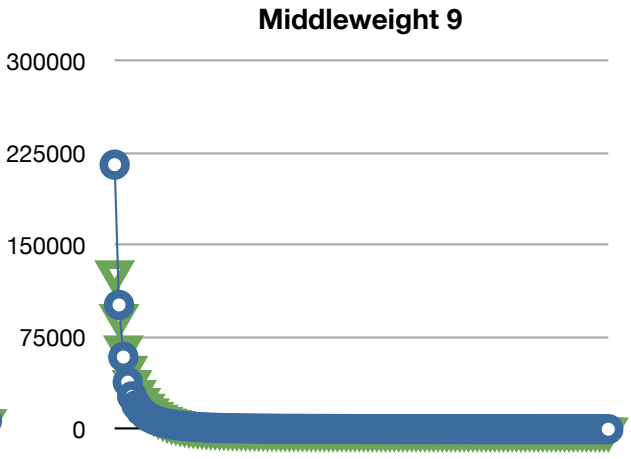

Middleweight 4
900000
675000
450000
225000
0

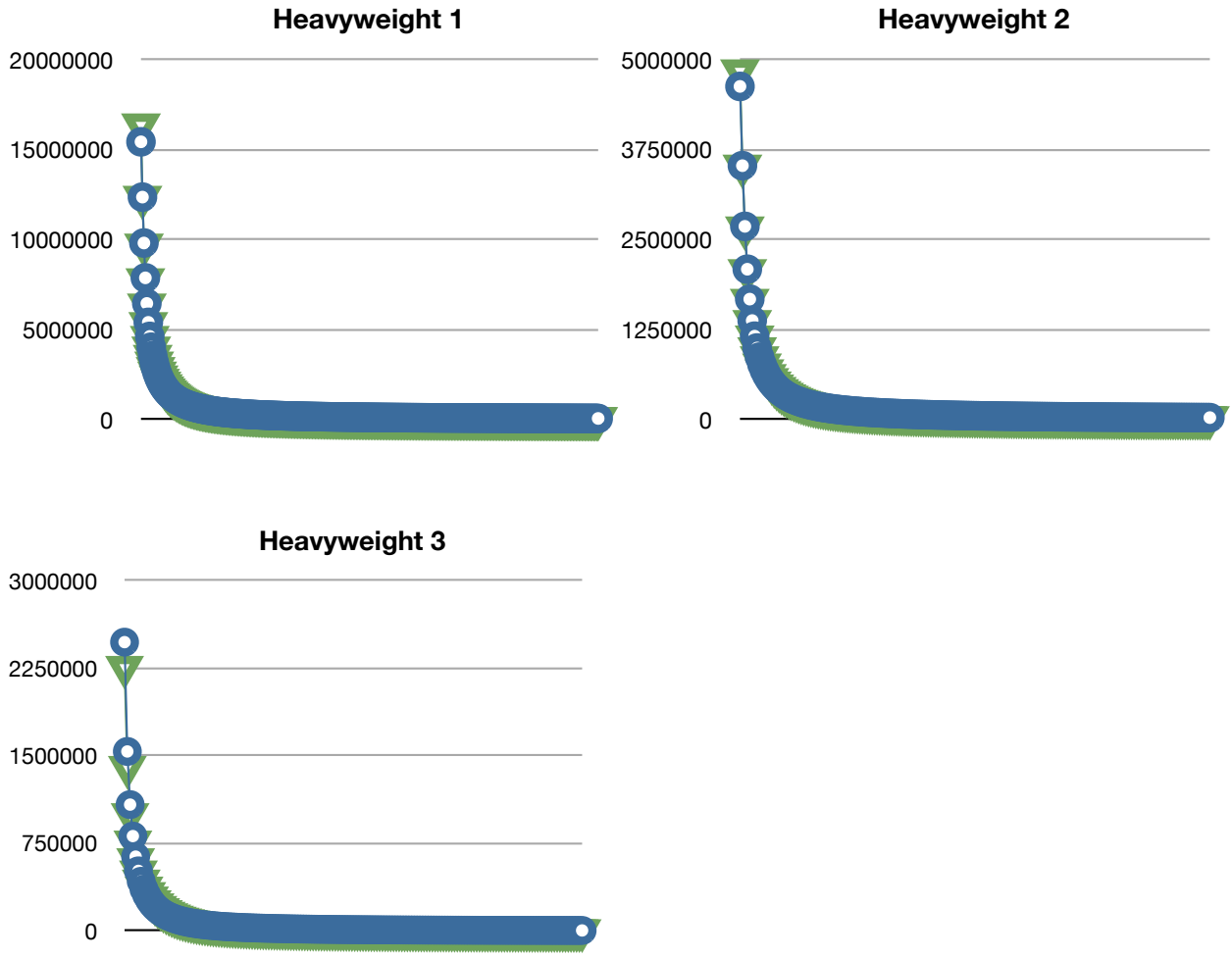

Middleweight 5
150000
112500
75000
37500
0


Middleweight 6
300000
225000
150000
75000
0

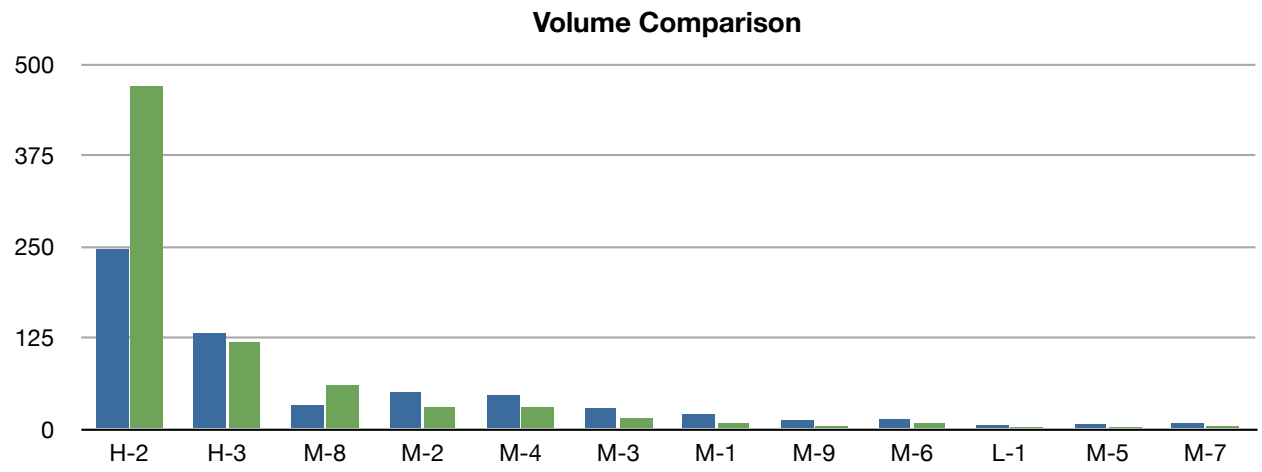

Middleweight 7
200000
150000
100000
50000
0


Middleweight 8
700000
525000
350000
175000
0


Middleweight 9
300000
225000
150000
75000
0



Σχήμα 22: Γραφικές παραστάσεις, για την περίπτωση πίνακα δυναμικού μεγέθους που λαμβάνει υπ' όψιν και τα 2 κριτήρια, της σχέσης μεταξύ in-degree και αντίστοιχου πλήθους κόμβων για τα διάφορα data sets που δοκιμάστηκαν - με "μπλε" εμφανίζονται τα πραγματικά αποτελέσματα, που προκύπτουν από την benchmark υλοποίηση και με "πράσινο" εμφανίζονται τα αποτελέσματα που προέκυψαν από την Count Min υλοποίηση



Σχήμα 23: Σύγκριση, για την περίπτωση πίνακα δυναμικού μεγέθους που λαμβάνει υπ' όψιν και τα 2 κριτήρια, του χώρου που απαιτεί η benchmark υλοποίηση σε σχέση με την Count Min - με "μπλε" εμφανίζονται τα πραγματικά αποτελέσματα, που προκύπτουν από την benchmark υλοποίηση και με "πράσινο" εμφανίζονται τα αποτελέσματα που προέκυψαν από την Count Min υλοποίηση

	Bottom 99% average abs(error) (%)	Top 1% average abs(error) (%)	Percentage of nodes with larger than expected error (%)	Bottom 99% average in-degree	#edges / #columns	#nodes / #columns	Space used by benchmark (MB)	Space used by count min (MB)
H-1	2.8	0.5	0.17	7	0.61	0.05	-	-
H-2	253.5	96.6	20.79	6	1.22	0.11	245.4	469.5
H-3	43.7	4.0	4.43	4	1.30	0.24	130.4	118.4
M-8	3.6	6.0	0.47	10	2.28	0.12	32.9	59.2
M-2	46.1	16.2	9.04	4	1.58	0.36	50.6	29.6
M-4	59.6	39.0	9.35	4	1.24	0.32	46.4	29.6
M-3	62.5	38.9	7.51	3	1.34	0.40	27.9	15.0
M-1	96.8	22.3	11.84	3	1.90	0.59	19.8	7.6
M-9	130.2	12.2	8.16	2	1.76	0.67	12.3	4.0
M-6	90.7	56.2	12.42	3	1.05	0.38	13.6	7.6
L-1	99.0	3.2	1.83	22	15.13	0.51	5.1	2.6
M-5	118.3	20.6	13.46	2	1.56	0.69	6.4	2.2
M-7	69.0	43.1	9.40	3	1.23	0.47	8.3	4.0

Πίνακας 3: Πίνακας μετρήσεων, για την περίπτωση πίνακα δυναμικού μεγέθους που λαμβάνει υπ' όψιν και τα 2 κριτήρια, σημαντικών δεικτών σχετικά με την ακρίβεια των in-degrees για τα διάφορα data sets που δοκιμάστηκαν

Παρατηρήσεις

Space

Παρατηρούμε ότι στις περισσότερες περιπτώσεις η εφαρμογή εξακολουθεί να καταλαμβάνει λιγότερο χώρο από την benchmark υλοποίηση. Παρ' όλ' αυτά, αυτή τη φορά υπάρχουν και περιπτώσεις, αποτελούν βέβαια μειοψηφία, όπου η μνήμη που καταλαμβάνεται ξεπερνά την αντίστοιχη στην benchmark υλοποίηση.

Σ' αυτήν την υλοποίηση, το μέγεθος του πίνακα εξαρτάται από το πλήθος των ακμών (όπως εξηγήθηκε και παραπάνω). Κάτι τέτοιο, μπορεί να οδηγήσει σε, χωρίς λόγο, εξωπραγματικά μεγάλο πίνακα, στην περίπτωση για παράδειγμα ενός γράφου με λίγους κόμβους και πολλές ακμές. Τότε, από ένα σημείο κι έπειτα δεν θα προστίθενται νέοι κόμβοι στον πίνακα, όμως το μέγεθος του θα συνεχίζει να αυξάνεται.

Error

Παρατηρούμε ότι σ' αυτήν την υλοποίηση το ποσοστό των κόμβων με σφάλμα μεγαλύτερο του αναμενόμενου ξεπερνά το προβλεπόμενο από τη θεωρία. Ο λόγος είναι ότι κάθε φορά που πραγματοποιείται αρχικοποίηση σε κάποιο νέο πίνακα προστίθεται ένα extra σφάλμα στα αποτελέσματα.

Accuracy

Στις παραπάνω γραφικές παραστάσεις παρατηρούμε ότι η υλοποίηση έχει αρκετά καλή ακρίβεια στα queries του τύπου "Πόσα nodes έχουν in-degree μεγαλύτερο ή ίσο του X".

Αντίθετα, στον πίνακα φαίνεται ότι η αντίστοιχη ακρίβεια στα queries του τύπου "Πόσο είναι το in-degree του κόμβου X" δεν είναι αντίστοιχη της προηγούμενης "βέλτιστης" υλοποίησης. Αυτό έχει να κάνει κυρίως με το σφάλμα που προκύπτει κατά την αρχικοποίηση των νέων πινάκων. Παρ' όλ' αυτά, η ακρίβεια των αποτελεσμάτων δεν είναι απαγορευτική (σφάλμα 83% κατά μέσο όρο). Εδώ πρέπει να ληφθεί υπ' όψιν ότι στη συγκεκριμένη υλοποίηση δεν υπήρχε καμία πρότερη γνώση σχετικά με τους γράφους στους οποίους εφαρμόστηκε. Συνεπώς, η παραπάνω ανακρίβεια μπορεί να θεωρηθεί αποδεκτή τηρουμένων των περιστάσεων.

Ακόμη, παρατηρούμε ότι η αναλογία ακμών-στηλών απέχει σημαντικά από τον αναμενόμενο μέσο όρο in-degree. Συμπεραίνουμε λοιπόν ότι η εκτίμηση του αλγορίθμου για τη μέση τιμή in-degree, με βάση τις πρώτες ακμές που καταφτάνουν, δεν μπορεί να θεωρηθεί ακριβής. Παρ' όλ' αυτά, το γεγονός ότι η υπολογιζόμενη από τον αλγόριθμο μέση τιμή in-degree είναι σε κάθε περίπτωση μικρότερη από την πραγματική οδηγεί σε μεγαλύτερο από το προβλεπόμενο τελικό μέγεθος πίνακα. Συνεπώς, αφού η ακρίβεια δεν είναι βέλτιστη για τον μεγαλύτερο πίνακα, σίγουρα θα ήταν ακόμη χειρότερη αν αυτός ήταν μικρότερος (αν δηλαδή είχε υπολογιστεί σωστά η μέση τιμή) και άρα καταλήγουμε ότι το κριτήριο αύξησης του μεγέθους του πίνακα με βάση την αναλογία ακμών-στηλών δεν μπορεί να οδηγήσει σε καλά αποτελέσματα.

Πίνακας Δυναμικά Μεταβαλλόμενου Μεγέθους - 2η Προσέγγιση

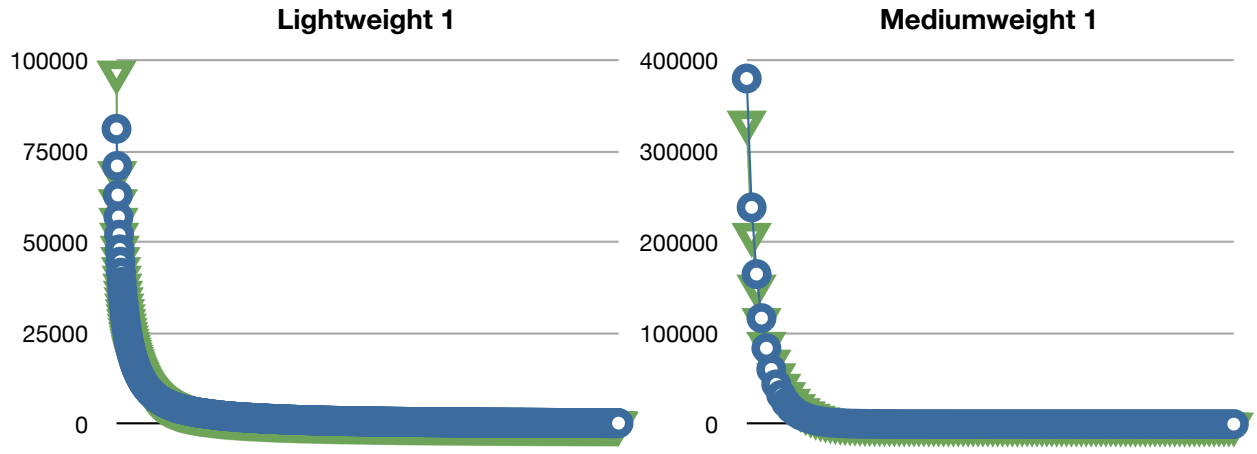
Περιγραφή

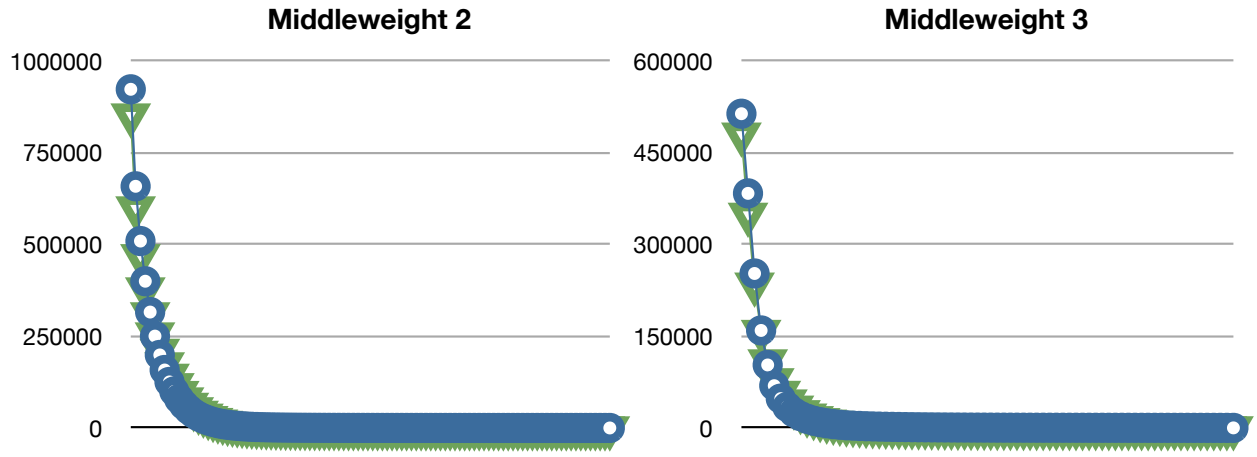
Στην προηγούμενη υλοποίηση έγιναν δύο σημαντικές παρατηρήσεις. Η μία ήταν ότι ενδεχομένως να είναι "επικίνδυνο" το μέγεθος του πίνακα να αυξάνεται ανάλογα με το πλήθος των ακμών (καθώς αυτό μπορεί να οδηγήσει σε εξωπραγματικά μεγάλο μέγεθος πίνακα) και η δεύτερη ήταν ότι, λόγω του extra σφάλματος που εμφανίζεται κατά τη δημιουργία νέων πινάκων και την αρχικοποίηση αυτών, η ακρίβεια των αποτελεσμάτων φαίνεται να μη συνδέεται με την αναλογία ακμών-στηλών. Ακόμη, παρατηρήθηκε ότι το πρόβλημα που φαινόταν να υπήρχε ως προς τη χρήση του πρώτου κριτηρίου ως μοναδικό κριτήριο (κακή ακρίβεια του query "Πόσα nodes έχουν in-degree μεγαλύτερο ή ίσο του 1) δεν είναι ιδιαίτερα σημαντικό.

Λαμβάνοντας υπ' όψιν τα παραπάνω, σ' αυτήν την υλοποίηση θα αγνοηθεί το δεύτερο κριτήριο αύξησης μεγέθους της προηγούμενης περίπτωσης (αυτό δηλαδή που επεδίωκε την ισότητα μεταξύ αναλογίας ακμών-στηλών και μέσης τιμής in-degree, αφού ούτε η απαιτούμενη αναλογία ακμών-στηλών στην περίπτωση της δυναμικής αύξησης μεγέθους φαίνεται να ταυτίζεται με αυτήν των αρχικών "στατικών" περιπτώσεων, ούτε όμως και η μέση τιμή in-degree είναι δυνατόν να υπολογιστεί με ακρίβεια). Παράλληλα, θα ενδυναμωθεί το, μοναδικό πλέον κριτήριο, ώστε να επιτυγχάνει ακόμη καλύτερη ακρίβεια. Συγκεκριμένα, αντί η αύξηση των στηλών του πίνακα να πραγματοποιείται κάθε φορά που η αναλογία κόμβων-στηλών ξεπερνά την τιμή 0.65, σ' αυτήν την υλοποίηση η σταθερά που θα χρησιμοποιηθεί θα ισούται με 0.4. Η συγκεκριμένη επιλογή δεν έγινε τυχαία. Ο πρώτος λόγος έχει να κάνει με το γεγονός ότι κατά την προηγούμενη υλοποίηση παρατηρήθηκε ότι τα κριτήρια που χρησιμοποιήθηκαν δεν ήταν αρκετά ισχυρά ώστε να επιστρέφουν καλής ακρίβειας αποτελέσματα. Ο δεύτερος λόγος είναι ότι στις μετρήσεις που έγιναν παραπάνω για την περίπτωση του στατικού πίνακα βέλτιστου μεγέθους, η συγκεκριμένη τιμή ήταν η μικρότερη που εμφανίστηκε στα αποτελέσματα.

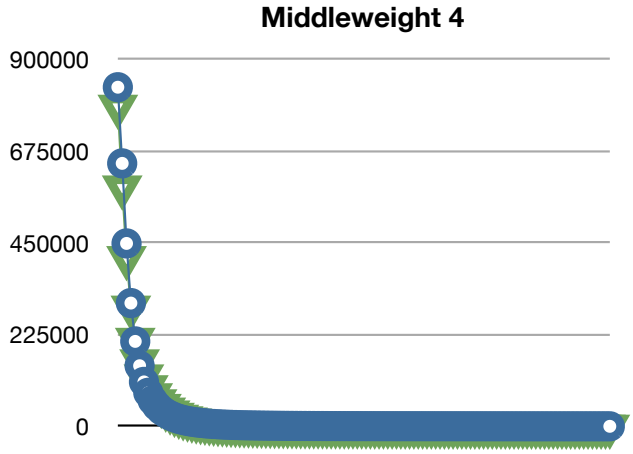
Τα αποτελέσματα του συγκεκριμένου αλγορίθμου φαίνονται παρακάτω.

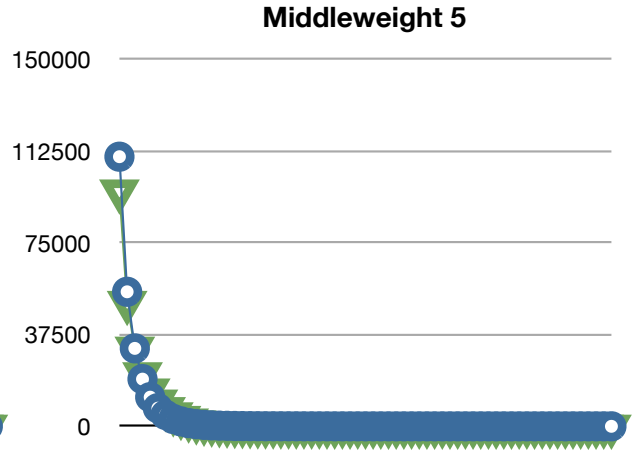
Μετρήσεις

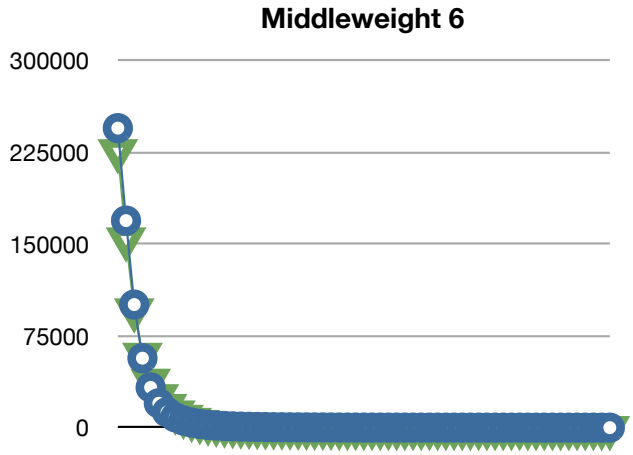


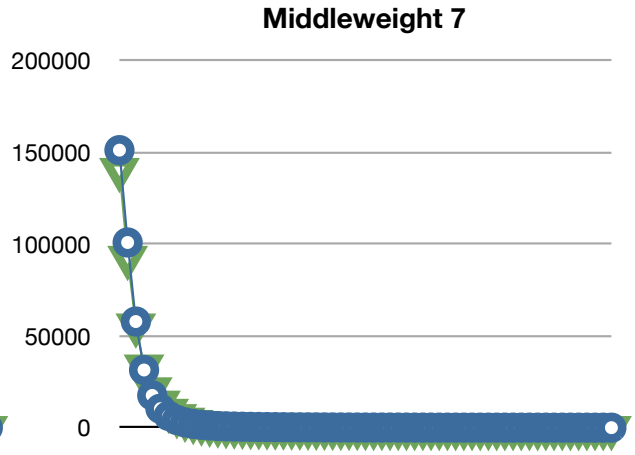

Middleweight 2
1000000
750000
500000
250000
0

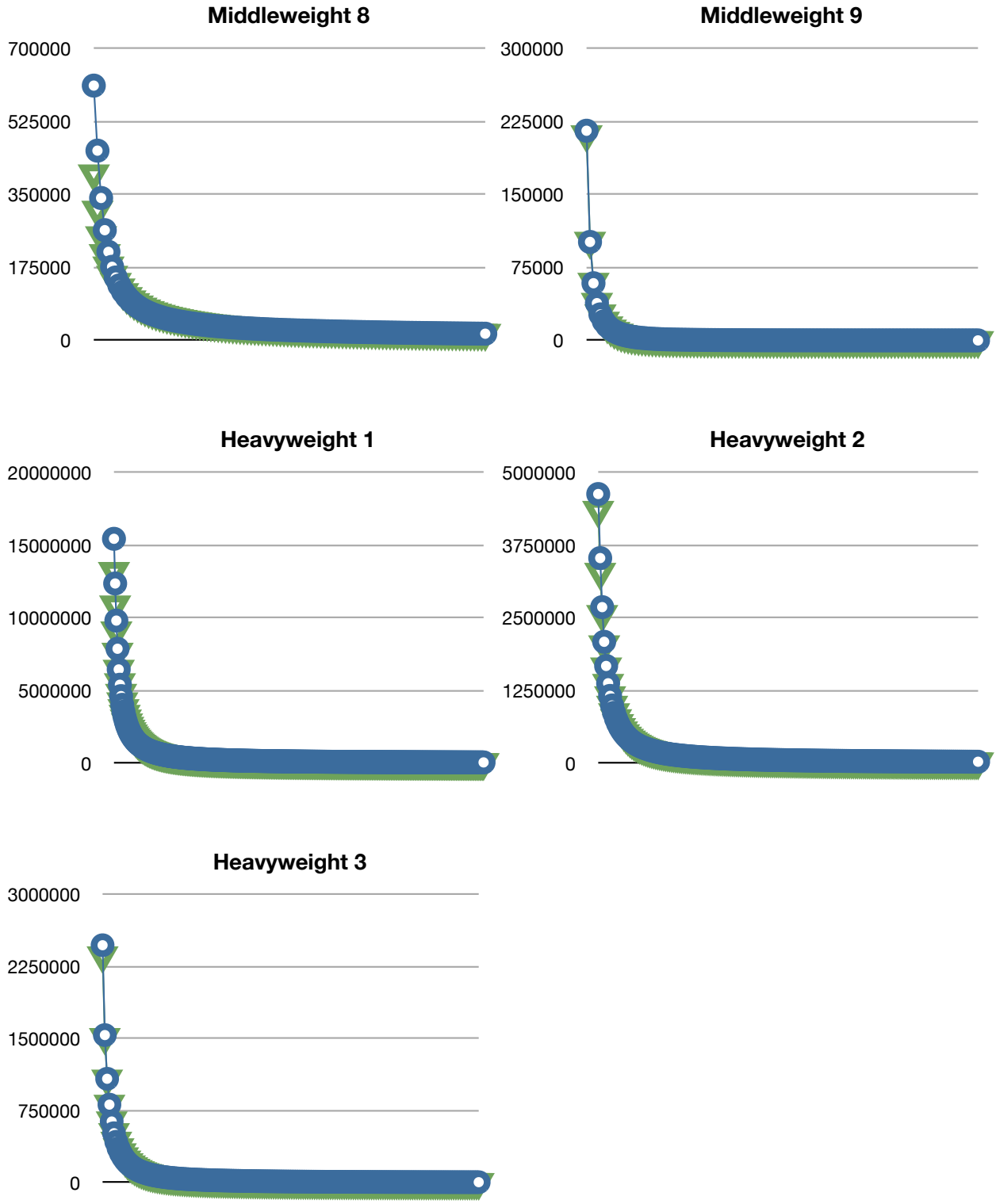
Middleweight 3
600000
450000
300000
150000
0


Middleweight 4
900000
675000
450000
225000
0

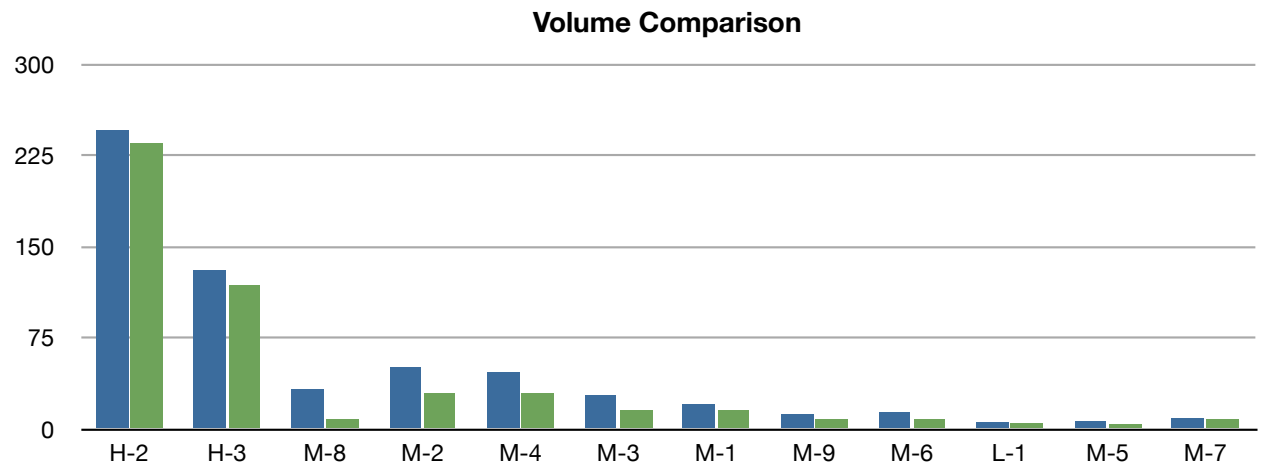

Middleweight 5
150000
112500
75000
37500
0


Middleweight 6
300000
225000
150000
75000
0


Middleweight 7
200000
150000
100000
50000
0



Σχήμα 24: Γραφικές παραστάσεις, για την περίπτωση πίνακα δυναμικού μεγέθους που λαμβάνει υπ' όψιν μόνο το 1 κριτήριο, της σχέσης μεταξύ in-degree και αντίστοιχου πλήθους κόμβων για τα διάφορα data sets που δοκιμάστηκαν - με "μπλε" εμφανίζονται τα πραγματικά αποτελέσματα, που προκύπτουν από την benchmark υλοποίηση και με "πράσινο" εμφανίζονται τα αποτελέσματα που προέκυψαν από την Count Min υλοποίηση



Σχήμα 25: Σύγκριση, για την περίπτωση πίνακα δυναμικού μεγέθους δυναμικού μεγέθους που λαμβάνει υπ' όψιν μόνο το 1 κριτήριο, του χώρου που απαιτεί η benchmark υλοποίηση σε σχέση με την Count Min - με "μπλε" εμφανίζονται τα πραγματικά αποτελέσματα, που προκύπτουν από την benchmark υλοποίηση και με "πράσινο" εμφανίζονται τα αποτελέσματα που προέκυψαν από την Count Min υλοποίηση

	Bottom 99% average abs(error) (%)	Top 1% average abs(error) (%)	Percentage of nodes with larger than expected error (%)	Bottom 99% average in-degree	#edges / #columns	#nodes / #columns	Space used by benchmark (MB)	Space used by count min (MB)
H-1	57.0	5.0	2.87	7	4.91	0.38	-	-
H-2	44.0	23.0	2.12	6	2.44	0.23	245.4	235.1
H-3	29.0	2.0	1.58	4	1.30	0.24	130.4	118.4
M-8	977.0	39.0	8.66	10	9.14	0.48	32.9	7.7
M-2	26.0	7.0	3.55	4	1.58	0.36	50.6	29.6
M-4	33.0	23.0	4.29	4	1.24	0.32	46.4	29.6
M-3	31.0	24.0	2.75	3	1.34	0.40	27.9	15
M-1	39.0	15.0	5.42	3	0.95	0.30	19.8	15
M-9	34.0	18.0	1.74	2	0.88	0.34	12.3	7.7
M-6	34.0	31.0	2.98	3	1.05	0.38	13.6	7.6
L-1	46.0	3.0	1.45	22	7.56	0.25	5.1	4.5
M-5	41.0	15.0	5.33	2	0.78	0.34	6.4	4
M-7	41.0	53.0	2.51	3	0.62	0.24	8.3	7.6

Πίνακας 4: Πίνακας μετρήσεων, για την περίπτωση πίνακα δυναμικού μεγέθους που λαμβάνει υπ' όψιν μόνο το 1 κριτήριο, σημαντικών δεικτών σχετικά με την ακρίβεια των in-degrees για τα διάφορα data sets που δοκιμάστηκαν

Παρατηρήσεις

Space

Από το σχήμα όπου συγκρίνεται η μνήμη που καταλαμβάνει η κάθε υλοποίηση προκύπτει ότι η παρούσα εφαρμογή απαιτεί σε κάθε περίπτωση λιγότερο χώρο απ' ότι η benchmark υλοποίηση. Κάτι τέτοιο οφείλεται σημαντικά στο γεγονός ότι πλέον το μέγεθος του Count Min πίνακα είναι ανάλογο του πλήθους κόμβων (ακόμη και σε μικρή αναλογία - 0.4). Έτσι, εξασφαλίζεται ότι το πλήθος των στηλών δεν θα "ξεφύγει" σε περιπτώσεις γράφων με μεγάλη αναλογία ακμών προς κόμβους.

Error

Από τον παραπάνω πίνακα παρατηρούμε ότι το ποσοστό κόμβων με σφάλμα μεγαλύτερο του αναμενόμενου ικανοποιεί, πλην μίας εξαιρέσεως, την προβλεπόμενη συνθήκη (<5%). Κάτι τέτοιο είναι σημαντικό λαμβάνοντας υπ' όψιν ότι ο τρόπος με τον οποίο γίνεται η δυναμική αύξηση του μεγέθους του πίνακα οδηγεί θεωρητικά στην προσθήκη extra σφάλματος. Παρ' όλ' αυτά, η απόκλιση των αποτελεσμάτων από τις πραγματικές τιμές φαίνεται να διατηρείται σε φυσιολογικά επίπεδα.

Accuracy

Στις γραφικές παραστάσεις που παρατίθενται παραπάνω, φαίνεται ότι τα queries αναζήτησης πλήθους κόμβων με συγκεκριμένο in-degree επιστρέφουν αποτελέσματα με καλή ακρίβεια, ειδικά αν ληφθεί υπ' όψιν ότι δεν υπάρχει καμία πρότερη γνώση των χαρακτηριστικών των data sets στα οποία εφαρμόζεται ο αλγόριθμος.

Παράλληλα, από τα στοιχεία του πίνακα παρατηρείται ότι, πλην μίας εξαιρέσεως (data set M-8), η ακρίβεια στα in-degrees των nodes είναι εντυπωσιακά καλύτερη απ' ότι στην προηγούμενη προσέγγιση (38% σφάλμα έναντι 83% της προηγούμενης υλοποίησης). Κάτι τέτοιο παρατηρείται τόσο για το 99% των λιγότερο "δημοφιλών" κόμβων, όσο και για το 1% των "κορυφαίων".

Όμως, παρατηρούμε ότι η απόκλιση στην "κακή" περίπτωση είναι σημαντική (σφάλμα 977% έναντι μέσης τιμής 38% όλων των υπολοίπων). Κάτι τέτοιο δεν υπήρχε στην προηγούμενη υλοποίηση όπου μπορεί η ακρίβεια γενικά να ήταν χειρότερη αλλά δεν υπήρχαν μεγάλες διακυμάνσεις στο ύψος της ακρίβειας των διαφόρων περιπτώσεων. Το συγκεκριμένο χαρακτηριστικό οφείλεται ενδεχομένως στο γεγονός ότι σε ορισμένα data sets (με μεγάλη αναλογία ακμών-κόμβων) η αύξηση του μεγέθους του πίνακα με βάση το κριτήριο του πλήθους κόμβων δεν είναι η ιδανική. Σε τέτοιες περιπτώσεις, ένας αλγόριθμος που να λαμβάνει υπ' όψιν ως κριτήριο το πλήθος των ακμών ίσως να επιστρέφει καλύτερα αποτελέσματα. Στη γενική περίπτωση όμως, η παρούσα υλοποίηση φαίνεται να είναι ακριβέστερη.

Συμπεράσματα και Ανοικτά Ζητήματα

Με βάση τις παραπάνω μετρήσεις οδηγούμαστε στα εξής συμπεράσματα.

1. Καθώς οι γράφοι των κοινωνικών δικτύων, των οποίων data sets χρησιμοποιούνται εδώ, ακολουθούν Power Law κατανομή, και μάλιστα με μονοψήφια στις περισσότερες περιπτώσεις μέση τιμή in-degree, απαιτείται πολλή υψηλή ακρίβεια στο Count Min Sketch, καθώς ακόμη και μονοψήφιο σφάλμα μπορεί να αποφέρει σημαντική απόκλιση στα αποτελέσματα.
   Παρ' όλ' αυτά, η χρήση Count Min για τον υπολογισμό των in-degrees των κόμβων σε μεγάλους γράφους προκύπτει ότι, ακόμη κι έτσι, αποτελεί μια ιδιαίτερα αποτελεσματική μέθοδο, τόσο χωρικά όσο και χρονικά, όταν αυτή εφαρμόζεται σε γράφο του οποίου τα χαρακτηριστικά (έστω η τάξη μεγέθους) είναι γνωστά εκ των προτέρων.

2. Στην περίπτωση που αυτά δεν είναι γνωστά, τότε μπορεί να χρησιμοποιηθεί πίνακας του οποίου το μέγεθος παραμένει σταθερό καθ' όλη τη διάρκεια του αλγορίθμου. Όσο μεγαλύτερο το μέγεθος του γράφου στον οποίο εφαρμόζεται τόσο μικρότερη η ακρίβεια των αποτελεσμάτων.

3. Εναλλακτικά, μπορεί να χρησιμοποιηθεί δυναμική αύξηση του μεγέθους του πίνακα. Σ' αυτήν την περίπτωση, όπως φάνηκε παραπάνω, τα καλύτερα αποτελέσματα προκύπτουν όταν ο πίνακας αυξάνει αναλογικά με το πλήθος διαφορετικών κόμβων που καταφτάνουν. Η συγκεκριμένη τακτική επιτυγχάνει, όχι μόνο υψηλή ακρίβεια, αλλά ταυτόχρονα διατηρεί το χώρο που καταλαμβάνει η εφαρμογή ανάλογο του αριθμού των κόμβων του γράφου, γεγονός σαφώς προτιμότερο από το να ήταν ανάλογος του πλήθους των ακμών. Η συγκεκριμένη στρατηγική πετυχαίνει αρκετά καλή ακρίβεια στα αποτελέσματα που επιστρέφει, απαιτεί όμως σημαντικά μεγαλύτερο χώρο από την περίπτωση όπου υπάρχει εξ' αρχής γνώση των χαρακτηριστικών του γράφου. Σε περιπτώσεις όμως όπου τέτοια πληροφόρηση είναι αδύνατη, η εν λόγω υλοποίηση είναι προτιμότερη από τη χρήση μιας "παραδοσιακής" δομής hash-map, αφού εξακολουθεί να καταναλώνει λιγότερη μνήμη απ' αυτήν, διατηρώντας παράλληλα σταθερή χρονική πολυπλοκότητα.

Φυσικά, η πρόταση της παρούσας εργασίας για δυναμική αύξηση του μεγέθους του Count Min πίνακα επιδέχεται σημαντικών βελτιώσεων. Τα κυριότερα σημεία του αλγορίθμου στα οποία μελλοντική έρευνα θα μπορούσε να αποφέρει σημαντική βελτίωση περιγράφονται αναλυτικά παρακάτω.

1. Η επιλογή του κριτηρίου αύξησης του μεγέθους του πίνακα αποτελεί κρίσιμο παράγοντα τόσο στην τελική ακρίβεια των αποτελεσμάτων όσο και στο συνολικό χώρο που καταλαμβάνει η εφαρμογή.
   Πειραματισμοί σχετικά με την ιδανική σταθερά για την αναλογία κόμβων-στηλών (στην παρούσα υλοποίηση χρησιμοποιήθηκε η σταθερά 0.4) θα μπορούσαν να βελτιώσουν σημαντικά τον αλγόριθμο.
   Ακόμη, η αναζήτηση κάποιου καλύτερου κριτηρίου είναι δυνατόν να αποτελέσει σημαντικό παράγοντα στη βελτίωση του αλγορίθμου.

2. Ο τρόπος με τον οποίο “μεταφέρεται” η πληροφορία από τον παλιό στο νέο/μεγαλύτερο πίνακα κατά την αύξηση του πλήθους στηλών επηρεάζει καθοριστικά την τελική ακρίβεια των αποτελεσμάτων. Μία νέα διαδικασία “μεταφοράς” θα μπορούσε να καταστήσει το συγκεκριμένο αλγόριθμο ακόμη ακριβέστερο.

3. Μία τεχνική που ενδεχομένως θα μπορούσε να μειώσει τον απαιτούμενο χώρο για την Count Min υλοποίηση είναι το graph sampling. Συγκεκριμένα, ο αλγόριθμος θα μπορούσε ενδεχομένως να “αγνοεί” ορισμένες από τις ακμές που καταφτάνουν και να διατηρεί έτσι αποτελέσματα για ένα sample του γράφου (που φυσικά θα καταλαμβάνουν μικρότερο ποσοστό της διαθέσιμης μνήμης), τα οποία εν συνεχεία με scale up είναι δυνατόν να πλησιάζουν τα πλήρη αποτελέσματα.

Μέρος 3ο

Βιβλιογραφία

[1] Burton H. Bloom. “Space/Time Trade-offs in Hash Coding with Allowable Errors” July 1970, Magazine Communications of the ACM, Volume 13 Issue 7, Pages 422-426.

[2] Graham Cormode. “Count-Min Sketch” AT&T Labs-Research.

[3] Flavio Bonomi, Michael Mitzenmacher, Rina Panigrahy, Sushil Singh, George Varghese. “An Improved Construction for Counting Bloom Filters” ESA’06 Proceedings of the 14th conference on Annual European Symposium - Volume 14.

[4] Moses Charikar, Kevin Chen, Martin Farach-Colton. “Finding Frequent Items in Data Streams” Automata, Languages and Programming, 29th International Colloquium, ICALP 2002 Malaga, Spain, July 8-13, 2002 Proceedings.

[5] Graham Cormode, S. Muthukrishnan. “An Improved Data Stream Summary: The Count-Min Sketch and its Applications” April 2005, Journal of Algorithms Volume 55 Issue 1 Pages 58-75.

[6] Noga Alon, Yossi Matias, Mario Szegedy. “The space complexity of approximating the frequency moments” Theory of computing, STOC ’96 Proceedings of the twenty-eighth annual ACM symposium on, Pages 20-29.

[7] Richard M. Karp, Christos H. Papadimitriou, Scott Shenker. “A simple algorithm for finding frequent elements in streams and bags” March 2003, Database Systems, ACM Transactions on (TODS), Volume 28 Issue 1 Pages 51-55.

[8] Supratik Bhattacharyya, Andre Madeira, S. Muthukrishnan, Tao Ye. “How to (accurately) skip past streams” April 2007 Data Engineering Workshop, IEEE 23rd International Conference on, Pages 654-663.

[9] Kook Jin Ahn, Sudipto Guha, Andrew McGregor. “Graph Sketches: Sparsification, Spanners and Subgraphs” Principles of Database Systems, PODS ’12 Proceedings of the 31st symposium on, Pages 5-14.

[10] Michael Mitzenmacher, Eli Upfal. “Probability and Computing: Randomized Algorithms and Probabilistic Analysis” Pages 107-111, Cambridge University Press, 2005.

[11] Bahman Bahmani, Ravi Kumar, Sergei Vassilvitskii. “Densest Subgraph in Streaming and MapReduce” January 2012, Proceedings of the VLDB Endowment Volume 5 Issue 5 Pages 454-465.

[12] Li Fan, Pei Cao, Jussara Almeida, Andrei Z. Broder. “Summary Cache: A Scalable Wide-Area Web Cache Sharing Protocol” June 2000, Networking, IEEE/ACM Transactions on (TON), Volume 8 Issue 3 Pages 281-293.

[13] L. A. N. Amaral, A. Scala, M. Barthelemy, H. E. Stanley. “Classes of small-world networks” October 2000, Proceedings of the National Academy of Sciences of the United States of America, Vol. 97 No. 21.

[14] S. N. Dorogovtsev. “Principles of statistical mechanics of uncorrelated random networks” September 2003, Nuclear Physics B, Volume 666 Issue 3 Pages 396-416.

[15] J.-P. Onnela, J. Saramaki, J. Hyvonen, G. Szabo, D. Lazer, K. Kaski, J. Kertesz, A.-L. Barabasi. “Structure and tie strengths in mobile communication networks” April 2007, Proceedings of the National Academy of Sciences of the United States of America, Vol. 104 No. 18.

[16] Oliver Hein, Michael Schwind, Wolfgang Konig. “Scale-Free Networks: The Impact of Fat Tailed Degree Distribution on Diffusion and Communication Processes” 2006, Wirtschaftsinformatik, Vol. 48 No. 4, Pages 267-275.
[17] Kenneth Cukier. “Data, data everywhere” February 2010, The Economist.
[18] James Manyika, Michael Chui, Brad Brown, Jacques Bughin, Richard Dobbs, Charles Roxburgh, Angela Hung Byers. “Big data: The next frontier for innovation, competition, and productivity” May 2011, McKinsey Global Institute.
[19] Doug Laney. “3D Data Management: Controlling Data Volume, Velocity, and Variety.” March 2012, Magazine Queue - Development, Volume 10 Issue 3, Page 20.
[20] “Gartner Says Solving ‘Big Data’ Challenge Involves More Than Just Managing Volumes of Data” June 2011, Gartner Press Release.
[21] Jeff Bertolucci. “Hadoop: From Experiment To Leading Big Data Platform” June 2013, InformationWeek.
[22] Jeffrey Dean, Sanjay Ghemawat. “MapReduce: Simplified Data Processing on Large Clusters” January 2008, Magazine Communications of the ACM - 50th anniversary issue: 1958-2008, Volume 51 Issue 1, Pages 107-113.
[23] Liz Tay. “Inside eBay’s 90PB data warehouse” May 2010, www.itnews.com.au.
[24] Julia Layton. “How Amazon Works”, www.howstuffworks.com.
[25] Steve Rosenbush. “Here’s How Facebook Manages Big Data” October 2013, The Wall Street Journal
[26] D. Agrawal, P. Bernstein, E. Bertino, S. Davidson, U. Dayal, M. Franklin, J. Gehrke, L. Haas, A. Halevy, J. Han, H. V. Jagadish, A. Labrinidis, S. Madden, Y. Papakonstantinou, J. M. Patel, R. Ramakrishnan, K. Ross, C. Shahabi, D. Suciu, S. Vaithyanathan, J. Widom. “Challenges and Opportunities with Big Data” February 2012.
[27] “Cisco Visual Networking Index: Forecast and Methodology, 2012-2017” May 2013, Executive Summary, Cisco.
[28] Derek Mead, “The Next Five Years of Explosive Internet Growth, In Seven Graphs” June 2013, motherboard.vice.com.
[29] Atish Das Sarma, Sreenivas Gollapudi, Rina Panigrahy. “Estimating PageRank on Graph Streams” Principles of database systems, PODS ’08 Proceedings of the twenty-seventh ACM SIGMOD-SIGACT-SIGART symposium on, Pages 69-78.
[30] Sudipto Guha, Adam Meyerson, Nina Mishra, Rajeev Motwani. “Clustering Data Streams: Theory and Practice” June 2003, Knowledge and Data Engineering, IEEE Transactions on, Volume 15 Issue 3, Pages 515-528.
[31] Johan Ugander, Brian Karrer, Lars Backstrom, Cameron Marlow. “The Anatomy of the Facebook Social Graph” November 2011
[32] Lars Backstrom, Paolo Boldi, Marco Rosa, Johan Ugander, Sebastiano Vigna. “Four Degrees of Separation” November 2011
[33] Aniket Mahanti, Niklas Carlsson, Anirban Mahanti, Martin Arlitt, Carey Williamson. “A Tale of the Tails: Power-laws in Internet Measurements” February 2013, Network IEEE, Volume 27 Issue 1, Pages 59-64.
[34] Michalis Faloutsos, Petros Faloutsos, Christos Faloutsos. “On Power-Law Relationships of the Internet Topology” Applications, technologies, architectures, and protocols for computer communication, SIGCOMM ’99 Proceedings of the conference on, Pages 251-262.
[35] Albert-Laszlo Barabasi, Reka Albert. “Emergence of Scaling in Random Networks” October 1999, Science Vol. 286 No. 5439, Pages 509-512.
[36] S. N. Dorogovtsev, J. F. F. Mendes, A. N. Samukhin. “Generic scale of the “scale-free” growing networks” May 2001, Physical Review E Volume 63 Issue 6.

[37] Jure Leskovec, Christos Faloutsos. “Sampling from Large Graphs” Knowledge discovery and data mining, KDD ’06 Proceedings of the 12th ACM SIGKDD international conference, Pages 631-636.
[38] Tamas Sarlos, Andras A. Benczur, Karoly Csalogany, Daniel Fogaras, Balazs Racz. “To Randomize or Not To Randomize: Space Optimal Summaries for Hyperlink Analysis” World Wide Web, WWW ’06 Proceedings of the 15th international conference on, Pages 297-306.
[39] “Count-Min Sketch”, www.wikipedia.com

Παραρτήματα

Κώδικας Benchmark Υλοποίησης

```
public Hashmap(String path, int time) throws FileNotFoundException {
		super();

		// Initializations
		long startTime = System.nanoTime();
		int node = 0, prevValue = 0;
		HashMap<Integer, Integer> nodeSet = new HashMap<Integer, Integer>(100000);
		HashMap<Integer, Integer> degreeSet = new HashMap<Integer, Integer>(1000);
		Scanner scanner = new Scanner(new File(path));
		// Ignore first integer
		node = scanner.nextInt();

		// Main algorithm
		while (scanner.hasNextInt()) {
			// Ignore source node
			node = scanner.nextInt();

			// Read destination node
			node = scanner.nextInt();

			// Add or update nodeSet
			if (!nodeSet.containsKey(node))
				prevValue = 0;
			else
				prevValue = nodeSet.get(node);
			nodeSet.put(node, (prevValue + 1));

			// Add or update degreeSet
			if (!degreeSet.containsKey(prevValue + 1))
				degreeSet.put((prevValue + 1), 1);
			else
				degreeSet.put((prevValue + 1), (degreeSet.get(prevValue + 1)
+ 1));
		}
		scanner.close();

		// Print results
		PrintWriter out = new PrintWriter("/Users/antones1/Desktop/Diplomatiki/
outputs/hashmap-" + time + ".txt");
		for (int i=1; i<=(degreeSet.size()); i++)
			out.println(i + "\t" + degreeSet.get(i));
		out.close();

		// Print node-degree pairs
		PrintWriter out2 = new PrintWriter("/Users/antones1/Desktop/Diplomatiki/
outputs/hashmapList-" + time + ".txt");
		for (int elem : nodeSet.keySet())
			out2.println(elem + "\t" + nodeSet.get(elem));
		out2.close();

		System.out.println("Total time needed: " + ((System.nanoTime() -
startTime) / 1000000000.0) + " seconds.");
	}
```

Κώδικας Count Min Υλοποίησης

Παρατίθεται αυτούσιος ο κώδικας της τελευταίας αναφερθείσας υλοποίησης. Οι προηγούμενες υλοποιήσεις αποτελούν στην ουσία ειδικές περιπτώσεις του παρακάτω κώδικα και προκύπτουν εύκολα απ' αυτόν.

```
public int hash(int nd, int index, int size) {
		int result = 0;
		int [] primes = {907, 17, 827, 31, 769, 37, 269, 41, 111, 59, 107, 67,
103, 89, 101, 97, 3, 5, 11, 19, 7};
		int modPrime = 1327217909;

		/* Avoid out of bounds integer */
		if (nd >= Integer.MAX_VALUE / primes[index])
			nd = nd % (Integer.MAX_VALUE / primes[index]);

		/* Hash function */
		result = ((primes[index] * nd + primes[index + 1]) % modPrime) % size;

		return result;
	}

	public int get(List<int[][]> list, int row, int column, int n) {
		// Get totalArray[i][j]
		if ((column / n) == 0)
			return list.get(0)[row][column];
		else {
			int array = (int) (Math.log(column / n) / (Math.log(2))) + 1;
			return list.get(array)[row][(column - (((int) Math.pow(2, (array -
1))) * n))];
		}
	}

	public void set(int value, List<int[][]> list, int row, int column, int n) {
		// Set totalArray[i][j] = value
		if ((column / n) == 0)
			list.get(0)[row][column] = value;
		else {
			int array = (int) (Math.log(column / n) / (Math.log(2))) + 1;
			list.get(array)[row][(column - (((int) Math.pow(2,  (array - 1))) *
n))] = value;
		}
	}

	public void increase1(List<int[][]>list, int row, int column, int n) {
		// totalArray[i][j]++
		if ((column / n) == 0)
			list.get(0)[row][column]++;
		else {
			int array = (int) (Math.log(column / n) / (Math.log(2))) + 1;
			list.get(array)[row][(column - (((int) Math.pow(2,  (array - 1))) *
n))]++;
		}
	}

	public int findInList(int target, List<int[][]> list, int k, int n, int curN,
HashMap<Integer, Integer> set) {
		// Return minimum from totalArray
```

```
		int min = -1, i = 0, j = 0, tmp = 0;

		for (i=0; i<k; i++) {
			j = hash(target, i, curN);

			// Initialize new array if needed
			// In each increase duplicate
			if ((j >= (curN / 2)) && (curN > n) && (get(list, i, j, n) == 0)) {
				tmp = get(list, i, (j % (curN / 2)), n);
				set(tmp, list, i, j, n);
			}

			/* Find minimum */
			if (min < 0)
				min = get(list, i, j, n);
			else if (get(list, i, j, n) < min)
				min = get(list, i, j, n);
		}

		return min;
	}

	public int average(HashMap<Integer, Integer> set, double pct) {
		int total = set.get(1), average = 0, count = 0, c1 = 0, c2 = 0;
		for (int elem : set.keySet()) {
			if (set.get(elem) > (pct * (double) total)) {
				c1 = set.get(elem);
				c2 = set.get(elem + 1);
				average += elem * (c1 - c2);
				count += c1 - c2;
			}
			else
				break;
		}
		return ((int) Math.round((float) average / count));
	}

	public Countmin(String source, int time) throws FileNotFoundException {
		super();

		// Initializations
		long startTime = System.nanoTime();
		int k = 3;
		int n = 10000;
		int [][] nodeArray = new int [k][n];
		List<int[][]> nodeSet = new LinkedList<int[][]>();
		nodeSet.add(nodeArray);
		nodeArray = null;
		HashMap<Integer, Integer> degreeSet = new HashMap<Integer, Integer>(1000);
		int node = 0, i = 0, j = 0, degree = 0, curN = n, tmp = 0, count = 0,
edges = 0, totalEdges = 0;
		int incLimit = 10000;
		Scanner scanner = new Scanner(new File(source));
		// Ignore first integer
		node = scanner.nextInt();

		// Main algorithm
		while (scanner.hasNextInt()) {
			// Ignore source node
			node = scanner.nextInt();
```

```
                    // Read destination node
                    node = scanner.nextInt();
                    edges++;
                    totalEdges++;

                    // Update degreeArray
                    for (i=0; i<k; i++) {
                          j = hash(node, i, curN);
                          // Initialize new array if needed and then update
                          // In each increase duplicate
                          if ((j >= (curN / 2)) && (curN > n) && (get(nodeSet, i, j, n)
== 0)) {
                                tmp = get(nodeSet, i, (j % (curN / 2)), n);
                                set((tmp + 1), nodeSet, i, j, n);
                          }
                          else
                                increase1(nodeSet, i, j, n);
                    }

                    // Update degreeSet
                    degree = findInList(node, nodeSet, k, n, curN, degreeSet);
                    if (!degreeSet.containsKey(degree))
                          degreeSet.put(degree, 1);
                    else
                          degreeSet.put(degree, (degreeSet.get(degree) + 1));

                    // Estimation for number of nodes
                    if (degree == 1)
                          count++;
                    // Increase array if needed
                    // Using just one criteria
                    if (count > (curN * 0.4)) {
                    // Using both criteria
                    //if (((count > (curN * 0.65)) && (curN == n)) || (edges >= (curN *
incLimit / 2))) {
                          if (curN == n) {
                                incLimit = average(degreeSet, 0.01);
                                System.out.println("Average: " + incLimit);
                          }
                          nodeArray = new int[k][curN];
                          nodeSet.add(nodeArray);
                          nodeArray = null;
                          curN = curN * 2;
                          edges = 0;
                    }
              }
              scanner.close();

              // Print results
              PrintWriter out = new PrintWriter("/Users/antones1/Desktop/Diplomatiki/
outputs/countmin-H-" + time + ".txt");
              for (i=1; i<=degreeSet.size(); i++)
                    out.println(2*degreeSet.get(i));
              out.close();

              System.out.println(totalEdges);

              // Print node-degree pairs
```

```
            PrintWriter out2 = new PrintWriter("/Users/antones1/Desktop/Diplomatiki/
outputs/countminList-H-" + time + ".txt");
            scanner = new Scanner(new File("/Users/antones1/Desktop/Diplomatiki/
outputs/hashmapList-H-" + time + ".txt"));
            while (scanner.hasNextInt()) {
                  node = scanner.nextInt();
                  out2.println(2*findInList(node, nodeSet, k, n, curN, degreeSet));
                  node = scanner.nextInt();
            }
            scanner.close();
            out2.close();

            System.out.println("Size: 3x" + curN);
            System.out.println("Total time needed: " + ((System.nanoTime() -
startTime) / 1000000000.0) + " seconds.");
      }
```