ΑΝΑΖΗΤΗΣΗ ΠΕΡΙΕΧΟΜΕΝΟΥ ΣΕ ΑΔΟΜΗΤΑ ΔΙΚΤΥΑ ΟΜΟΤΙΜΩΝ

Η
ΜΕΤΑΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ ΕΞΕΙΔΙΚΕΥΣΗΣ

Υποβάλλεται στην

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

από τον

Βασίλειο Φωτόπουλο

ως μέρος των Υποχρεώσεων

για τη λήψη

του

ΜΕΤΑΠΤΥΧΙΑΚΟΥ ΔΙΠΛΩΜΑΤΟΣ ΣΤΗΝ ΠΛΗΡΟΦΟΡΙΚΗ

ΜΕ ΕΞΕΙΔΙΚΕΥΣΗ ΣΤΟ ΛΟΓΙΣΜΙΚΟ

Μάρτιος 2008

ΠΕΡΙΕΧΟΜΕΝΑ

ΕΥΡΕΤΗΡΙΟ ΠΙΝΑΚΩΝ

ΕΥΡΕΤΗΡΙΟ ΣΧΗΜΑΤΩΝ

ΠΕΡΙΛΗΨΗ

Βασίλειος Φωτόπουλος του Γεωργίου και της Παναγιώτας. MSc, Τμήμα Πληροφορικής, Πανεπιστήμιο Ιωαννίνων, Μάρτιος, 2008. Αναζήτηση Περιεχομένου σε Αδόμητα Δίκτυα Ομότιμων. Επιβλέποντες: Ευαγγελία Πιτουρά & Απόστολος Ζάρρας.

Ένα από τα βασικότερα προβλήματα που έχει να αντιμετωπίσει η κοινότητα των αδόμητων ομότιμων συστημάτων (P2Ps) είναι η εύρεση αποτελεσματικών μεθόδων για την ανάκτηση πληροφορίας που διαμοιράζεται μεταξύ των κόμβων του δικτύου. Οι μέχρι τώρα προτεινόμενες μέθοδοι δεν κρίνονται αποτελεσματικές μιας και στηρίζονται κυρίως στην λογική της πλημμύρας ή σε κάποιο είδος γενικής (global) γνώσης που έχει κάθε κόμβος για το δίκτυο.

Σε αυτή την διατριβή προτείνουμε μια καινούργια τεχνική αναζήτησης με πληροφορία, το Inverse σχήμα, για ανάκτηση πληροφορίας σε αδόμητα P2P δίκτυα. Η λογική του αλγορίθμου περιλαμβάνει δύο επιμέρους ιδέες: (1) Οι κόμβοι καταγράφουν σε μια cache που διατηρούν τοπικά τα ερωτήματα που υποβάλλονται στο δίκτυο από τους κόμβους της κοινότητας. (2) Κάθε νέο ερώτημα που υποβάλλεται, καθοδηγείται σε εκείνο το κομμάτι-τμήμα του δικτύου από όπου προέρχονται κόμβοι που υπέβαλλαν κατά το παρελθόν παρόμοια ερωτήματα. Το όνομα της τεχνικής αποδίδεται στο γεγονός ότι οι ερωτήσεις δεν κατευθύνονται προς τους κόμβους που έχουν την πληροφορία, αλλά σε αυτούς που σε μια προηγούμενη χρονική στιγμή την εντόπισαν.

Ο κύριος ανταγωνιστής του Inverse σχήματος είναι ένα Forward σχήμα αναζήτησης το οποίο ακολουθεί την αντίστροφη λογική του Inverse. Πιο συγκεκριμένα, κάθε κόμβος καταγράφει στην cache του τα QueryHits μηνύματα που δημιουργούν και μεταδίδουν οι κόμβοι όταν απαντούν σε μια ερώτηση, γνωστοποιόντας με αυτόν τον τρόπο ένα κομμάτι της γνώσης τους. Στην συνέχεια κάθε νέο ερώτημα κατευθύνεται προς εκείνο το τμήμα του δικτύου στο οποίο κατά το παρελθόν υπήρξαν κόμβοι που ικανοποίησαν ένα παρόμοιο ερώτημα.

Οι παραπάνω τεχνικές υλοποιήθηκαν με χρήση του προσομοιωτή OMNeT++ και συγκρίθηκαν σε διάφορα περιβάλλοντα. Τα αποτελέσματα δείχνουν πως αν και απλό, το Inverse σχήμα αναζήτησης χωρίς να κάνει χρήση QueryHits μηνυμάτων κατορθώνει και υπερτερεί έναντι του Forward σχήματος ως προς το recall. Δηλαδή, στο πλήθος των απαντήσεων που εντοπίζει μια ερώτηση ως προς το συνολικό πλήθος των αποτελεσμάτων που υπάρχουν για την συγκεκριμένη ερώτηση σε όλο το δίκτυο.

EXTENDED ABSTRACT IN ENGLISH

Fotopoulos Bill. MSc, Computer Science Department, University of Ioannina, Greece. March 2008. Content Based Searching on Unstructured P2P Systems. Thesis Supervisors: Evaggelia Pitoura & Apostolos Zarras.

Locating content in unstructured peer-to-peer networks is a challenging problem. Existing search mechanisms rely on either blind flooding queries to all peers, or routing queries to peers based on some form of global knowledge that each peer has about the network. Although flooding is robust and simple, it is not scalable.

In this dissertation, we propose the Inverse searching schema which is an efficient, scalable yet simple mechanism for improving the information retrieval problem in unstructured P2P networks. The main idea of the Inverse schema consists of the following two concepts: (1) Each peer in the network keeps a local cache which contains the most recently submitted queries. (2) Each new query is forwarded to that region of the network where there is a peer which had previously submitted a similar query.

The Inverse mechanism consists of four components: A *cache* which logs query messages coming from neighbors, a *query similarity function* which calculates the similarity of past queries against a new query, a *ranking function* which is an online neighbor ranking criterion and a *basic searching mechanism* which defines the way peers forward queries to selected neighbors.

The main competitor of the Inverse schema is the Forward searching mechanism that follows a reverse logic. In particular, each peer forwards a new query to this region of the network where there is a peer which had previously provided an answer for a similar query. Each node reveals a piece of his content creating and forwarding a QueryHit message to the other peers of the community when he resolves a query.

We implemented and compared the two searching mechanisms over various environments, using the OMNeT++ simulator. Our results indicate that Inverse mechanism outperforms Forward with respect to the well-known recall criterion (number of answers that a query found to the total number of results that exist in the network), without using QueryHits messages.

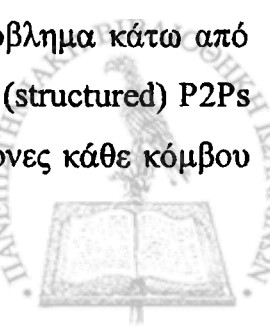
ΚΕΦΑΛΑΙΟ 1. ΕΙΣΑΓΩΓΗ

1.1. Εισαγωγή

Τα δίκτυα ομότιμων κόμβων (peer-to-peer systems (P2Ps)) είναι δίκτυα στα οποία κόμβοι έχοντας τον ρόλο τόσο του πελάτη (client) όσο και του εξυπηρετητή (server) ενώνονται με ακμές πάνω από το Διαδίκτυο (Internet) σχηματίζοντας με αυτόν τον τρόπο ένα ιδεατό (overlay) δίκτυο. Μέσα σε αυτά τα δίκτυα, κάθε κόμβος προσφέρει και αναζητεί ένα σύνολο πόρων που μπορεί να είναι από μια απλή συλλογή εγγράφων (μουσικά κομμάτια, βίντεο κτλ.) μέχρι και ένα σύνολο από υπηρεσίες (services). Τα κύρια χαρακτηριστικά των P2Ps είναι ο διπλός ρόλος του κάθε κόμβου στο δίκτυο, η μεγάλη κλιμάκωση (scale) όσον αφορά το πλήθος των κόμβων που συνδέονται μέσα σε αυτό και τέλος η δυναμικότητα ενός τέτοιου περιβάλλοντος μιας και κόμβοι εισέρχονται και εξέρχονται προς και από το δίκτυο κατά βούληση.

Ένα από τα βασικότερα προβλήματα που έχει να αντιμετωπίσει ένας σχεδιαστής μιας P2P εφαρμογής είναι με ποιόν τρόπο θα δρομολογούνται τα ερωτήματα μέσα στο δίκτυο έτσι ώστε κάθε ερώτηση να εξετάζει (προωθείται σε) όσο το δυνατόν λιγότερους κόμβους μέχρι να ικανοποιηθεί.

Αρχικά, η επιστημονική κοινότητα αντιμετώπισε το παραπάνω πρόβλημα κάτω από δυο οπτικές γωνίες. Στην πρώτη εξ αυτών συναντάμε τα δομημένα (structured) P2Ps π.χ CHORD [1] και CAN [2]. Σε αυτά τα συστήματα τόσο οι γείτονες κάθε κόμβου



που εισέρχεται στο σύστημα όσο και τα ίδια τα δεδομένα ορίζονται ρητά από το δοθέν πρωτόκολλο για το που οφείλουν να τοποθετηθούν. Το πλεονέκτημα τους είναι ότι η αναζήτηση είναι λογαριθμική, το μειονέκτημα τους συναντάται στο ότι το σύστημα δεν μπορεί να ανταπεξέλθει αποτελεσματικά όταν έχουμε μεγάλο ρυθμό αφίξεων και αναχωρήσεων κόμβων στοιχείο που είναι στενά συνδεδεμένο με την φύση αυτών των δικτύων.

Στη δεύτερη περίπτωση συναντάμε τα αδόμητα (unstructured) P2Ps στα οποία δεν ορίζεται καμία δομή για το πως πρέπει να είναι συνδεδεμένοι οι κόμβοι στο δίκτυο ενώ η πληροφορία μπορεί να είναι οπουδήποτε. Οι λύσεις που έχουν δοθεί μπορεί να χωριστούν σε δύο κατηγορίες, *στις τεχνικές αναζήτησης χωρίς πληροφορία (blind) και σε εκείνες με πληροφορία (informed)*. Χαρακτηριστικό παράδειγμα αναζήτησης χωρίς πληροφορία εκτελεί το σύστημα Gnutella [3] στο οποίο κάθε ερώτηση εξετάζει όλους τους κόμβους του δικτύου εωσότου ικανοποιηθεί. Μια παραλλαγή της πλημμύρας που εκτελεί η Gnutella και που σαν στόχο έχει να βελτιώσει την κλιμάκωση είναι οι k-random walkers [4] καθώς επίσης και παραλλαγές αυτού του σχήματος. Στις τεχνικές αναζήτησης με πληροφορία, στόχος είναι κάθε κόμβος να συντηρεί ευρετήρια (indexes) για το περιεχόμενο που φέρουν κάποιοι από τους κόμβους του συστήματος (π.χ οι γείτονες του) και έτσι με αυτόν τον τρόπο όταν έρχεται μια ερώτηση να συμβουλεύονται το ευρετήριο τους και να προωθούν το ερώτημα στους πιο κατάλληλους κόμβους, ήτοι σε αυτούς τους κόμβους που με μεγαλύτερη πιθανότητα σε σχέση με κάποιους άλλους θα ικανοποιήσουν την δοθείσα ερώτηση.

Μια κοινή αδυναμία όλων των παραπάνω προσεγγίσεων είναι ότι περιορίζονται σε αναζήτηση βασιζόμενη σε κάποιο είδος αναγνωριστικού (match id-based searching) όπως για παράδειγμα το όνομα ενός αρχείου κάτι που σε σύγκριση με τις τωρινές μηχανές αναζήτησης θέτει έναν περιορισμό στις απαιτήσεις που έχει ένας χρήστης που επιθυμεί να πραγματοποιήσει πάνω σε ένα P2P σύστημα μια αναζήτηση περιεχομένου (content-based) μέσω ερωτήσεων που εκφράζονται με πολλαπλούς όρους (terms) ή λέξεις (words).

Σε αυτήν την εργασία διερευνώνται τεχνικές αναζήτησης για ανάκτηση περιεχομένου σε αδόμητα P2P δίκτυα με τις ερωτήσεις να εκφράζονται μέσω λέξεων.

Αναφερόμαστε σε τεχνικές με πληροφορία στις οποίες οι κόμβοι στο δίκτυο καταγράφουν τοπικά στις caches τους ερωτήσεις που υποβλήθηκαν κατά το παρελθόν και στην συνέχεια παρατηρώντας ομοιότητα μεταξύ νέων και παλιών ερωτημάτων προσπαθούν να κατευθύνουν το δοθέν ερώτημα σε εκείνον τον κόμβο που το περιεχόμενο του έχει μεγαλύτερη πιθανότητα να το ικανοποιήσει.

Προτείνουμε μια καινούργια τεχνική αναζήτησης με πληροφορία, το Inverse σχήμα αναζήτησης, για ανάκτηση πληροφορίας σε αδόμητα P2P δίκτυα. Η λογική του αλγορίθμου βασίζεται σε δύο επιμέρους ιδέες: (1) Οι κόμβοι καταγράφουν σε μια cache που διατηρούν τοπικά τα ερωτήματα που υποβάλλονται στο δίκτυο από τους κόμβους της κοινότητας. (2) Κάθε νέο ερώτημα που υποβάλλεται, καθοδηγείται σε εκείνο το κομμάτι-τμήμα του δικτύου από όπου προέρχονται κόμβοι που ρώτησαν κατά το παρελθόν παρόμοια ερωτήματα. Το όνομα της τεχνικής αποδίδεται στο γεγονός ότι οι ερωτήσεις δεν κατευθύνονται προς τους κόμβους που έχουν την πληροφορία, αλλά σε αυτούς που σε μια προηγούμενη χρονική στιγμή την εντόπισαν.

Ο κύριος ανταγωνιστής του Inverse σχήματος είναι ένα Forward σχήμα αναζήτησης το οποίο ακολουθεί την αντίστροφη λογική του Inverse. Πιο συγκεκριμένα, κάθε κόμβος καταγράφει στην cache του τα QueryHits μηνύματα που δημιουργούν και μεταδίδουν οι κόμβοι όταν απαντούν σε μια ερώτηση, γνωστοποιόντας με αυτόν τον τρόπο ένα κομμάτι της γνώσης τους. Στην συνέχεια κάθε νέο ερώτημα κατευθύνεται προς εκείνο το τμήμα του δικτύου στο οποίο κατά το παρελθόν υπήρξαν κόμβοι που ικανοποίησαν ένα παρόμοιο ερώτημα.

Τόσο το Inverse όσο και το Forward σχήμα αναζήτησης ανήκουν στην κατηγορία των lazy αλγορίθμων μιας και η πληροφορία που κρατά και διατηρεί κάθε κόμβος στο σύστημα δεν απαιτεί για να αποκτηθεί κανένα επιπρόσθετο κόστος επικοινωνίας μεταξύ των κόμβων του P2P. Προφανώς η υπόθεση πάνω στην οποία στηρίζεται η παραπάνω λογική είναι ότι μέσα στο δίκτυο υποβάλλονται παρόμοιες μεταξύ τους ερωτήσεις (μεγάλο term locality) κάτι όμως που παρατηρείται εν γένει στα σημερινά συστήματα.

Ο στόχος της παρούσας εργασίας είναι διττός. Αρχικά θέλουμε να μελετήσουμε την συμπεριφορά του Inverse και Forward μηχανισμού αναζήτησης καθώς και να τα συγκρίνουμε σε διαφορετικά περιβάλλοντα. Επιπλέον, θέλουμε να εξετάσουμε γενικότερα αν η εφαρμογή της lazy-based λογικής σε ένα μηχανισμό αναζήτησης σε αδόμητα P2P δίκτυα μπορεί να βελτιώσει το recall του συστήματος σε σχέση με ένα τυχαίο μηχανισμό προώθησης των μηνυμάτων-ερωτημάτων.

1.2. Δομή της Διατριβής

Η εργασία ακολουθεί στην συνέχεια την εξής δομή: Στο κεφάλαιο 2 αρχικά ορίζουμε το περιβάλλον κάτω από το οποίο θα εξετάσουμε τις προτεινόμενες τεχνικές αναζήτησης και στην συνέχεια προχωρούμε στην λεπτομερή περιγραφή τους. Στο κεφάλαιο 3 ακολουθεί ένα σύνολο πειραμάτων που πραγματοποιήθηκαν με σκοπό την σύγκριση των προτεινόμενων τεχνικών αναζήτησης. Στην συνέχεια στο κεφάλαιο 4 περιγράφεται ένα σύνολο σχετικών δουλειών. Τέλος, η διατριβή ολοκληρώνεται με την καταγραφή συμπερασμάτων και μελλοντικών επεκτάσεων της.

ΚΕΦΑΛΑΙΟ 2. ΜΕΘΟΔΟΙ ΑΝΑΖΗΤΗΣΗΣ

2.1. Περιβάλλον και Ορισμοί

Θεωρούμε ένα αδόμητο P2P δίκτυο το οποίο αποτελείται από ένα σύνολο κόμβων $\{p_1,p_2,\ldots,p_n\}$. Κάθε p_i φέρει ένα μοναδικό id και συνδέεται άμεσα με ένα σύνολο κόμβων που αποτελούν τους γείτονες του στο δίκτυο. Επιπλέον κάθε p_i φέρει μια συλλογή από έγγραφα (documents) τα οποία μπορεί να είναι οποιοσδήποτε πόρος που μπορεί να χαρακτηριστεί από ένα σύνολο όρων π.χ απλά έγγραφα, μουσικά αρχεία, βίντεο, υπηρεσίες κτλ. Στο εξής θα αναφερόμαστε στη συλλογή του p_i ως Τοπική Συλλογή Εγγράφων (Local Document Collection (LDC)) θεωρώντας ότι αποτελείται από ένα σύνολο από έγγραφα $LDC_i = \{d_1,d_2,\ldots,d_m\}$.

Όμοια με τους κόμβους έτσι και κάθε d_i χαρακτηρίζεται από ένα id που είναι μοναδικό στο δίκτυο. Κάθε έγγραφο χαρακτηρίζεται από ένα σύνολο όρων (terms) από τα οποία αποτελείται-περιγράφεται, πιο συγκεκριμένα γράφουμε $d_i=\{t_1,t_2,\ldots t_n\}$.

Μέσα στο P2P, κόμβοι υποβάλλουν ερωτήματα τα οποία αποτελούνται από ένα σύνολο όρων. Πιο συγκεκριμένα κάθε ερώτημα αναγνωρίζεται από ένα μοναδικό id και περιγράφεται ως εξής $q_i=\{t_1,t_2,..t_l\}$.

Θεωρούμε ότι ένα d_i ικανοποιεί (ταιριάζει με) ένα q_i αν και μόνο αν το q_i είναι υποσύνολο του d_i. Συνεπώς αναφερόμαστε σε συζευκτικές ερωτήσεις. Αντίστοιχα ένας κόμβος θεωρούμε ότι ικανοποιεί (είναι σχετικός με) ένα q_i αν στην LDC του υπάρχει τουλάχιστον ένα d_i το οποίο ταιριάζει με το q_i.

Θεωρώντας ότι μέσα σε ένα ρεύμα ερωτήσεων (query stream) υπάρχει μεγάλος βαθμός locality (όροι επαναλαμβάνονται συχνά) στόχος μας είναι να προτείνουμε lazy μεθόδους αναζήτησης οι οποίες θα εκμεταλλεύονται ομοιότητες μεταξύ των ερωτημάτων που υποβάλλονται με στόχο κάθε φορά να βελτιώνεται το recall. Το recall για μια ερώτηση q_i ορίζεται ως το πλήθος των εγγράφων που εντοπίζει ο αλγόριθμος αναζήτησης ως προς το πλήθος των συνολικών, σχετικών ως προς την q_i, εγγράφων που υπάρχουν σε όλο το δίκτυο. Θέτοντας το διαφορετικά, αν στο δίκτυο υποβληθούν n ως προς το πλήθος ίδιες (παρόμοιες) ερωτήσεις σε n διαφορετικές χρονικές στιγμές, επιθυμούμε το σύστημα μας να παρουσιάζει αύξηση στο recall.

Ιδιότητα εκμάθησης: Θα λέμε ότι ένας προτεινόμενος μηχανισμός αναζήτησης εμφανίζει την ιδιότητα εκμάθησης αν το σύστημα που τον εφαρμόζει παρατηρεί αύξηση στο recall όταν υποβάλλονται σε αυτό ίδιες (παρόμοιες) ερωτήσεις.

2.2. Forward Σχήμα Αναζήτησης

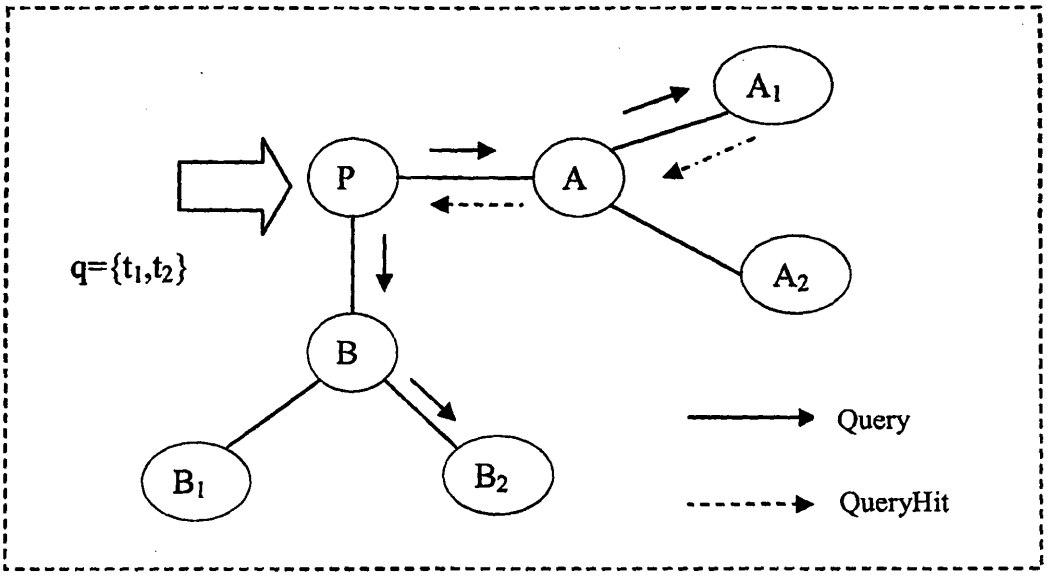
Ο Forward μηχανισμός αναζήτησης θεωρεί πως κάθε κόμβος στο δίκτυο διατηρεί μια cache στην οποία αποθηκεύει τις πιο πρόσφατες ερωτήσεις που υποβλήθηκαν κατά το παρελθόν. Βασική ιδέα του αλγορίθμου είναι η εξής: *Μια δοθείσα ερώτηση q που φτάνει σε έναν κόμβο προωθείται προς εκείνον τον γείτονα του οποίου το υπογράφημα που ξεκινάει από αυτόν παρείχε απάντηση (ήταν σχετικό) ως προς μια ερώτηση q' που υποβλήθηκε κατά το παρελθόν και είναι παρόμοια της q.* Συνεπώς, η υπόθεση που κρύβεται πίσω από τη κεντρική ιδέα του Forward σχήματος είναι πως αν ένας κόμβος p είναι σχετικός ως προς μια ερώτηση q, ο ίδιος ο p θα είναι σε θέση να απαντήσει και μια ερώτηση q' παρόμοια της q.

2.2.1. Γενική Περιγραφή

Χωρίς να περιγράψουμε τον τρόπο με τον οποίο εκτελείται ο μηχανισμός προώθησης των ερωτημάτων του Forward σχήματος, στο σχήμα 2.1 δίνεται μια γενική εικόνα εκτέλεσης του μηχανισμού. Ο P υποβάλει μια δεδομένη χρονική στιγμή t στο δίκτυο μια ερώτηση q. Μια ερώτηση-μήνυμα q αποτελείται από τα παρακάτω πεδία:
<QID, List of Terms, List of Peers, TTL}
Το QID αποτελεί το αναγνωριστικό της ερώτησης και θεωρείται μοναδικό για κάθε ερώτηση που υποβάλλεται στο δίκτυο. Θεωρούμε ότι το QID αποτελείται από το id του κόμβου που υποβάλλει την ερώτηση σε συνδυασμό με έναν αύξοντα αριθμό για κάθε ερώτηση στο δίκτυο που υποβάλει ο συγκεκριμένος κόμβος. Στην συνέχεια υπάρχει μια λίστα από όρους στην οποία ο χρήστης περιγράφει τα έγγραφα που επιθυμεί να εντοπίσει. Το τρίτο πεδίο είναι μια λίστα από κόμβους οι οποίοι μέχρι στιγμής έχουν λάβει την ερώτηση. Τέλος, το TTL ορίζει το πλήθος των κόμβων που οφείλει να εξετάσει μια ερώτηση (βάθος αναζήτησης). Πιο συγκεκριμένα, ο κόμβος ερώτησης (querying) ορίζει μια συγκεκριμένη τιμή, στην συνέχεια κάθε ένας ενδιάμεσος κόμβος που λαμβάνει την ερώτηση ελαττώνει την τιμή του TTL κατά ένα, μόλις η τιμή του γίνει ίση με το 0 η ερώτηση διαγράφεται και η αναζήτηση σταματάει.

Μόλις ένας κόμβος λάβει μια ερώτηση, αρχικά εξετάζει αν ο ίδιος έχει στην LDC του έγγραφα που την ικανοποιούν. Αν ναι (στο παράδειγμα ο A_1), τότε πλέον γίνεται απαντών (answering) κόμβος και δημιουργεί ένα QueryHit μήνυμα το οποίο ακολουθεί το αντίστροφο μονοπάτι της ερώτησης με στόχο να φτάσει στον κόμβο ερώτησης (στο παράδειγμα ο P). Το QueryHit μήνυμα αποτελείται από τα παρακάτω πεδία:
<QID, List of Terms, List of Peers, Answering peer ID, List Of Relevant Documents>
Τα πρώτα τρία πεδία είναι ίδια με αυτά του μηνύματος της ερώτησης και αντιγράφονται από αυτό στο QueryHit μήνυμα. Τα υπόλοιπα δυο πεδία είναι το id του κόμβου που παρέχει την απάντηση καθώς και μια λίστα με τα ids των εγγράφων που ικανοποιούν την ερώτηση.

Μόλις ο ερωτών κόμβος λάβει το QueryHit μήνυμα από έναν από τους γείτονες του ελέγχει να δει αν τα ids των εγγράφων που παρέχει ο απαντών κόμβος βρίσκονται ήδη στην LDC του. Αν όχι, δημιουργεί μια σύνδεση με τον απαντών κόμβο και κατεβάζει τα έγγραφα που δεν βρήκε στην LDC του. Διαφορετικά δεν κάνει καμία ενέργεια.



Σχήμα 2.1 Παράδειγμα εκτέλεσης του Forward μηχανισμού αναζήτησης. Ο P υποβάλει την ερώτηση q μία χρονική στιγμή t. Ο A_1 είναι ο μοναδικός κόμβος στο δίκτυο που έχει στη LDC του έγγραφα που ικανοποιούν την q

2.2.2. Βασικός Μηχανισμός Προώθησης Ερωτημάτων

Όταν ένας κόμβος λάβει ένα ερώτημα q, εκτελεί τοπικά και εντελώς αυτόνομα έναν αλγόριθμο μέσω του οποίου ταξινομεί τους γείτονες του ως προς το q. Στην ουσία αποδίδει σε κάθε έναν από τους γείτονες του ένα σκορ. Ο γείτονας με το μεγαλύτερο σκορ κρίνεται πιο σχετικός ως προς το δοθέν q και αυτό με την σειρά του προωθείται προς αυτόν. Η καρδιά του μηχανισμού προώθησης των ερωτημάτων στο Forward σχήμα αποτελείται από τα παρακάτω 4 τμήματα:

1. Μια *cache* στην οποία ο κάθε κόμβος καταγράφει για κάθε έναν από τους γείτονες του τα πιο πρόσφατα ερωτήματα για τα οποία παρείχαν απάντηση.
2. Μια *συνάρτηση ομοιότητας* για τον υπολογισμό ομοιότητας μεταξύ διαφορετικών ερωτημάτων που υποβάλλονται στο δίκτυο.

3. Μια *συνάρτηση σκορ* μέσω της οποίας κάθε κόμβος μόλις λαμβάνει μια ερώτηση θα υπολογίζει το σκορ που αντιστοιχεί σε κάθε γείτονα του ως προς το δοθέν ερώτημα.
4. Τέλος, ένας *βασικός μηχανισμός αναζήτησης* που θα ορίζει με ποιό τρόπο θα επικοινωνούν οι κόμβοι μεταξύ τους στο δίκτυο και δεν θα είναι άλλο από δυο ήδη γνωστά σχήματα αναζήτησης σε αδόμητα P2P: Τους k-walkers και ένα constrained flooding παρόμοιο με αυτό που χρησιμοποιεί η Gnutella [3].

Δομή και Διαχείριση της Cache

Η cache του κάθε κόμβου στο δίκτυο γεμίζει μέσω των QueryHits μηνυμάτων. Για παράδειγμα η cache του P στο σχήμα 2.1 μετά το τέλος της ερώτησης q που υπέβαλλε θα φαίνεται όπως στον πίνακα 2.1:

Πίνακας 2.1 Περιεχόμενο της cache του κόμβου P μετά το τέλος εκτέλεσης της ερώτησης q

Query ID	**Set of Terms**	**Connections & Hits**	**Timestamp**
q	$\{t_1,t_2\}$	B(0), A(5)	T_1
.....	.....	.....	.....

Κάθε εγγραφή της cache αντιστοιχεί σε μια ερώτηση που υποβλήθηκε στο δίκτυο. Για κάθε μια ερώτηση εκτός από το id της και το σύνολο των όρων που ουσιαστικά την αναγνωρίζουν μοναδικά καταγράφεται το πλήθος των εγγράφων που είδε ο κόμβος από κάθε γείτονα του. Τέλος, καταγράφεται και ένα timestamp που αφορά την πρώτη εμφάνιση QueryHit για μια ερώτηση.

Η σωστή ερμηνεία του πίνακα 2.1 είναι η εξής: Από το υπογράφημα που ξεκινάει από τον γείτονα A του P βρέθηκαν συνολικά 5 έγγραφα σχετικά με την q. Ενώ από το υπογράφημα που ξεκινάει από το γείτονα B του P δεν βρέθηκε κανένα έγγραφο σχετικό με την q. Με αυτόν τον τρόπο εάν λάβει ο P στο μέλλον μια ερώτηση q' παρόμοια ως προς την q θα είναι σε θέση να εκτιμήσει ποιος από τους γείτονες του (ή ποιο σωστά ποιο από τα υπογραφήματα που ξεκινούν από τους γείτονες του) είναι καταλληλότερος για να ικανοποιήσει την q'.

Θα πρέπει να τονίσουμε ότι για την κάθε καταγεγραμμένη ερώτηση στην cache του κάθε κόμβου δεν υπάρχει η έννοια του βάθους της αναζήτησης. Για παράδειγμα ο P δεν γνωρίζει σε πόσα βήματα (hops) μακριά από τον γείτονα του A εντοπίστηκαν τα 5 σχετικά έγγραφα, ούτε επιπλέον γνωρίζουμε για το εάν αυτά τα 5 έγγραφα ανήκουν σε έναν ή παραπάνω από έναν κόμβους.

Τέλος, όσον αφορά την replacement πολιτική που πρέπει να εφαρμοστεί επιλέγουμε να εκτελέσουμε τον αλγόριθμο LRU. Ποιο συγκεκριμένα κρατάμε στην cache ένα μέγιστο πλήθος από τις ποιο πρόσφατες ερωτήσεις που υποβλήθηκαν στο δίκτυο κατά το παρελθόν. Ανανεώνουμε το timestamp μιας ερώτησης q αν η τομή της με μια καινούργια q' είναι διαφορετική του κενού συνόλου (ομοιότητα διαφορετική του 0). Ο λόγος της επιλογής του LRU σχήματος πέραν της απλότητας του ως προς την υλοποίηση κρίνεται κατάλληλος στο περιβάλλον μας αναλογιζόμενοι πρώτον τις συχνές αναχωρήσεις/εισόδους κόμβων από και προς το δίκτυο και δεύτερον για να 'πιάνουμε' την term locality, παρόμοιες ερωτήσεις υποβάλλονται σε σχετικά κοντινό χρονικό διάστημα.

Συνάρτηση Ομοιότητας

Όπως τονίσαμε στην εισαγωγή της περιγραφής της Forward μεθόδου, όταν ένας κόμβος λάβει μια ερώτηση q οφείλει να υπολογίσει την ομοιότητα της q με όλες τις q' ερωτήσεις που είναι την δεδομένη χρονική στιγμή στην cache του έτσι ώστε να εντοπίσει ποιος από τους γείτονες του έχει απαντήσει κατά το παρελθόν παρόμοιες ερωτήσεις ως προς την q. Αυτός λογικά θα μπορεί να απαντήσει καταλληλότερα και την ερώτηση q. Για να μπορούμε να υπολογίσουμε ομοιότητα μεταξύ ερωτημάτων θα χρησιμοποιήσουμε την ομοιότητα συνημίτονου.

Η ομοιότητα συνημίτονου μεταξύ 2 διανυσμάτων-ερωτημάτων ορίζεται από την σχέση: $sim(q,q') = \frac{q * q'}{|q| * |q'|}$. Είναι το εσωτερικό γινόμενο 2 διανυσμάτων προς το γινόμενο των νορμών τους, δίνοντας σαν σύνολο τιμών μια τιμή που ανήκει στο [0,1]. Όσο πιο κοντά στο 1 είναι η τιμή δυο ερωτημάτων τόσο πιο παρόμοια είναι μεταξύ τους.

Στο δικό μας περιβάλλον τα ερωτήματα που υπάρχουν στην cache ενός κόμβου μια δεδομένη χρονική στιγμή ορίζουν ένα διανυσματικό χώρο, όπου συνιστώσα-διάσταση είναι ένας όρος. Έτσι για παράδειγμα αν θεωρήσουμε ότι ένας κόμβος έχει στην cache του μια δεδομένη χρονική στιγμή 3 ερωτήματα τα $q_1=\{t_1,t_2\}$, $q_2=\{t_1,t_3\}$ και $q_3=\{t_4,t_5\}$ τότε οποιοδήποτε ερώτημα λάβει ο κόμβος έστω το $q'=\{t_4,t_7\}$ θα εκφραστεί με βάση το διάνυσμα $\{t_1,t_2,t_3,t_4,t_5,t_7\}$. Συνεπώς το q' θα γραφεί ως $q'=\{0,0,0,1,0,1\}$ ενώ τα υπάρχοντα ερωτήματα στην cache θα γραφούν ως $q_1=\{1,1,0,0,0,0\}$, $q_2=\{1,0,1,0,0,0\}$ και $q_3=\{0,0,0,1,1,0\}$.

Να σημειώσουμε πως το μέτρο ομοιότητας που χρησιμοποιούμε είναι ανεξάρτητο της γενικής τεχνικής κάτι που σημαίνει ότι μπορεί να αντικατασταθεί από ένα οποιοδήποτε άλλο μέτρο και να αρχίσει πλέον να υπολογίζεται τοπικά από το κάθε κόμβο του συστήματος χωρίς να αλλάξει ο ευρύτερος μηχανισμός αναζήτησης. Παρόμοιες μετρικές που παρατίθενται στην λίστα [5], μπορεί να είναι το Jaccard coefficient, dice coefficient και το inner product.

Επιπλέον, όπως τονίστηκε στην ενότητα 2.1, στην περίπτωση μας προσπαθούμε να επιλύουμε συζευκτικές ερωτήσεις συνεπώς δεν μας απασχολεί να χρησιμοποιήσουμε κάποιο μέτρο ομοιότητας μεταξύ ερωτημάτων και εγγράφων. Όπως ήδη έχουμε αναφέρει ένα έγγραφο θεωρείται σχετικό ως προς μια ερώτηση q αν απλά περιέχει όλους τους όρους της q.

Συνάρτηση Σκορ

Μόλις ένας κόμβος λάβει μια ερώτηση q υπολογίζει δυναμικά την συγκεκριμένη χρονική στιγμή, με βάση το περιεχόμενο της cache, ένα σκορ για κάθε έναν από τους γείτονες του. Οι γείτονες με το μεγαλύτερο σκορ αναγνωρίζονται ποιο σχετικοί ως προς την q συνεπώς σε αυτούς προωθείται η ερώτηση. Το μέτρο δίνεται από την σχέση:

$$Score\ (P_i, q) = \sum_{j\ queries\ \ answeared\ \ by\ p_i} sim\ (q_j, q) * R(P_i, q_j)\cdot$$

Για να υπολογίσουμε την σχετικότητα ενός γείτονα P_i ως προς μια q, υπολογίζουμε συναθροίστηκα την ομοιότητα ανάμεσα στην q και σε όλες τις ερωτήσεις (q_j) που

είδαμε να απαντιόνται από το υπογράφημα που ξεκινάει από τον P_i επί το πλήθος των αποτελεσμάτων ($R(P_i,q_j)$) που λάβαμε από τον P_i. Είναι φανερό πως όσο περισσότερες όμοιες ερωτήσεις έχει απαντήσει ένας γείτονας και ταυτόχρονα όσο περισσότερα αποτελέσματα έχουμε λάβει από αυτόν τόσο μεγαλύτερο σκορ θα λάβει στην βαθμολόγηση (ranking).

Με αυτόν τον τρόπο το σύστημα μας εκτελεί αναζήτηση με πληροφορία. Να σημειώσουμε τέλος πως στην περίπτωση που η cache είναι άδεια ή η νέα q που λαμβάνουμε είναι εντελώς ανόμοια με τις υπάρχουσες στην cache ερωτήσεις όλοι οι κόμβοι φέρουν το ίδιο σκορ που είναι ίσο με το μηδέν. Σε αυτήν την περίπτωση ο όλος μηχανισμός γίνεται τυχαίος. Δηλαδή η επιλογή των γειτόνων στους οποίους πρέπει να προωθηθεί η ερώτηση γίνεται με τυχαίο τρόπο.

Βασικός Μηχανισμός Αναζήτησης

Ο βασικός μηχανισμός αναζήτησης υλοποιεί τον τρόπο με τον οποίο οι κόμβοι επικοινωνούν-ανταλλάσουν μεταξύ τους μηνύματα. Σε αυτήν την περίπτωση θα θεωρήσουμε δύο υπάρχοντα γνωστά σχήματα τους k-walkers και ένα σχήμα constrained flooding. Στην ουσία μέσω αυτών των μηχανισμών ορίζεται το μέγεθος της αναζήτησης (search size) δηλαδή το πλήθος των κόμβων που εξετάζει κάθε ερώτημα που υποβάλλεται στο δίκτυο.

Στην περίπτωση των k-walkers, ο ερωτών κόμβος αρχικά επιλέγει το πολύ k από τους γείτονες του για να προωθήσει την ερώτηση. Στην συνέχεια κάθε ένας από τους ενδιάμεσους κόμβους των k μονοπατιών επιλέγει έναν γείτονα του (διαφορετικό από αυτόν από τον οποίο έλαβε το ερώτημα) για να προωθήσει το ερώτημα έως ότου το πεδίο TTL γίνει ίσο με 0. Συνεπώς σε αυτήν την περίπτωση το μέγεθος της αναζήτησης είναι το πολύ k*TTL βήματα. Να σημειώσουμε ότι στην υλοποίηση μας θεωρούμε την απλή εκδοχή αυτού του σχήματος κάτι που σημαίνει ότι δεν ελέγχουμε για παράδειγμα αν τα μονοπάτια συμπίπτουν για να εξετάσουμε ακόμα περισσότερους διαφορετικούς κόμβους κ.ο.κ.

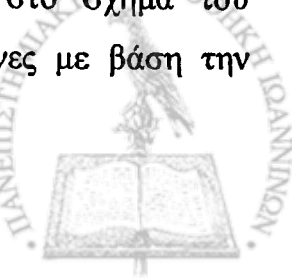
Στην περίπτωση του constrained flooding, ο κόμβος ερώτησης αλλά και κάθε ενδιάμεσος κόμβος που λαμβάνει μια ερώτηση την προωθεί σε ένα υποσύνολο των γειτόνων του. Στην υλοποίηση αυτού του μηχανισμού επιλέξαμε να είναι το 50% των γειτόνων του κάθε κόμβου. Ξανά όπως και πριν ο κόμβος που υποβάλλει το ερώτημα ορίζει το TTL που θα εκτελέσει η ερώτηση στο δίκτυο συνεπώς στην περίπτωση αυτή το μέγεθος της αναζήτησης ορίζεται σαν $(d/2)^0+(d/2)^1+\ldots+(d/2)^{TTL}$, όπου d το μέσο πλήθος γειτόνων του κάθε κόμβου στο δίκτυο.

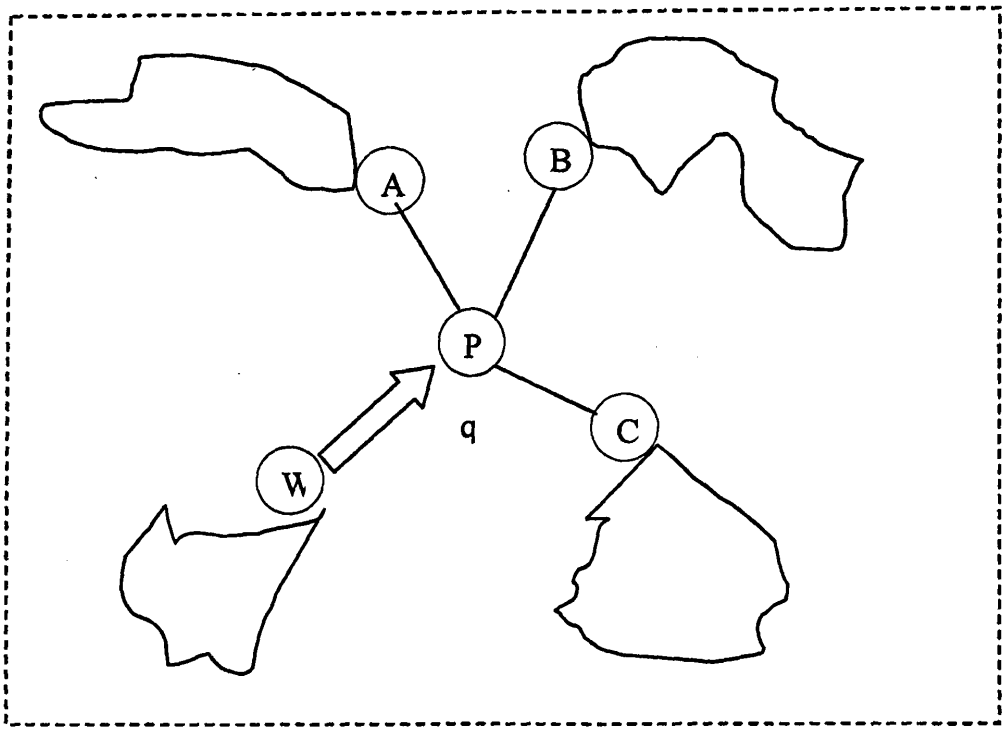
Συνεπώς τα παραπάνω σχήματα ορίζουν το πλήθος των γειτόνων στους οποίους πρέπει να προωθηθεί μια ερώτηση, το με ποιόν τρόπο προκύπτουν αυτοί γίνεται με βάση την συνάρτηση σκορ που αναλύσαμε.

2.2.3. Ιδιότητα Εκμάθησης και Τυχαίες Προωθήσεις Ερωτημάτων

Επειδή θέλουμε ο Forward μηχανισμός να έχει την ιδιότητα εκμάθησης οι πιο υποσχόμενες διαδρομές δεν μπορεί να επιλέγονται πάντα με βάση το σκορ που αποδίδει ένας κόμβος στους γείτονες του. Για να κατανοήσουμε το πρόβλημα θεωρούμε για παράδειγμα το P2P του σχήματος 2.2. Όταν ο P λάβει την ερώτηση q και έχει άδεια την cache ή υπάρχουν ερωτήσεις στην cache αλλά δεν είναι παρόμοιες με την q τότε η επιλογή του P γίνεται τυχαία για το που θα προωθηθεί η q. Αν επιλέξει τον A και C και λάβει από αυτούς κάποια αποτελέσματα στην συνέχεια οποιαδήποτε q' παρόμοια της q θα οδηγείται συνεχώς προς τα υπογραφήματα που ορίζουν οι κόμβοι A και C. Αυτό έχει σαν αποτέλεσμα οι άλλοι γείτονες (π.χ ο B) να μην εξερευνηθούν ποτέ αν κάθε φορά η επιλογή γίνεται με βάση την συνάρτηση σκορ.

Συνεπώς στην περίπτωση των k-walkers ο κάθε κόμβος που αρχικοποιεί μια αναζήτηση επιλέγει οι μισοί από τους περίπατους να ακολουθήσουν τυχαίο μονοπάτι και οι άλλοι μισοί να προωθούν τα ερωτήματα χρησιμοποιώντας την πληροφορία της cache υπολογίζοντας δηλαδή σκορ για κάθε γείτονα. Όμοια στο σχήμα του constrained flooding ο κάθε κόμβος επιλέγει τους μισούς γείτονες με βάση την συνάρτηση σκορ ενώ τους υπόλοιπους τυχαία.





Σχήμα 2.2 Το πρόβλημα της επιλογής γειτόνων στο Forward σχήμα. Η πρώτη τυχαία επιλογή του P για μια q θα επηρεάσει μελλοντικές q' παρόμοιες της q με αποτέλεσμα κάποιες διαδρομές να μην εξερευνώνται ποτέ

Τέλος, θα δείξουμε μέσω παραδείγματος πως το Forward σχήμα στους k-walkers εμφανίζει την ιδιότητα εκμάθησης. Έστω ότι ο κόμβος C του δικτύου του σχήματος 2.3 συναντάται σε περιπάτους της ίδιας ερώτησης $q=\{t_1,t_2\}$ η οποία υποβάλλεται στο δίκτυο 4 φορές από έστω 4 διαφορετικούς κόμβους τις χρονικές στιγμές T_1, T_2, T_3 και T_4. Τότε μια πιθανή εκτέλεση για τον C μπορεί να είναι η εξής:

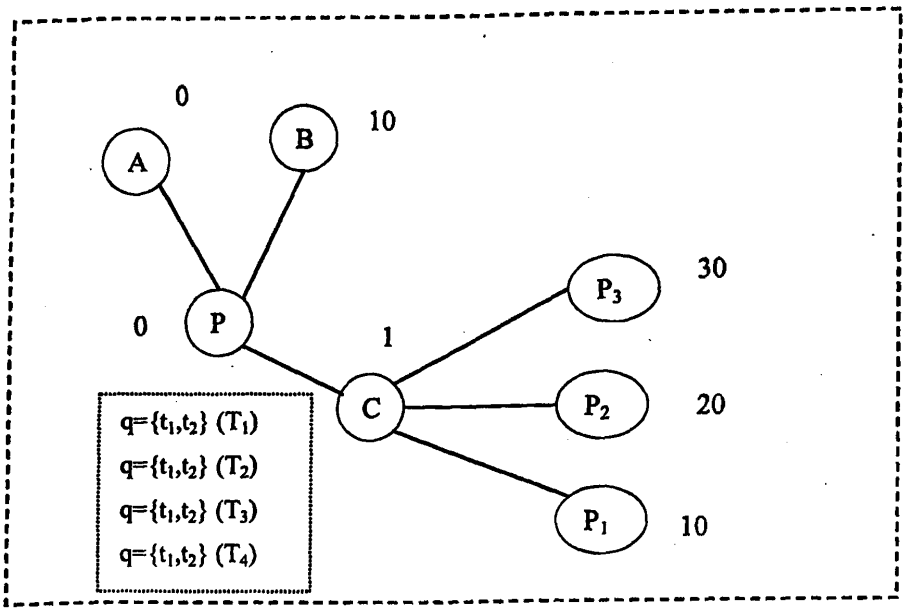
*1. Ο C είναι στο μονοπάτι και επιβάλλεται να κάνει **τυχαία** επιλογή και έστω επιλέγει τον P_1 →10*

*2. Ο C είναι στο μονοπάτι και επιβάλλεται να κάνει επιλογή **με πληροφορία**, επιλέγει τον P_1 → 10*

*3. Ο C είναι στο μονοπάτι και επιβάλλεται να κάνει **τυχαία** επιλογή και έστω επιλέγει τον P_2 →20*

*4. Ο C είναι στο μονοπάτι και επιβάλλεται να κάνει επιλογή **με πληροφορία**, επιλέγει τον P_2 → 20*

Οποιαδήποτε μελλοντική επιλογή με πληροφορία για όμοια ερώτηση θα δίνει 20 αποτελέσματα.



Σχήμα 2.3 Ο κόμβος C λαμβάνει σε 4 διαφορετικές χρονικές στιγμές T_1, T_2, T_3 και T_4 την ίδια ερώτηση $q=\{t_1,t_2\}$. Δίπλα σε κάθε κόμβο του δικτύου απεικονίζεται το πλήθος των εγγράφων που είναι σχετικά ως προς την q. Π.χ ο κόμβος A δεν έχει κανένα σχετικό έγγραφο ενώ ο κόμβος P_3 έχει 30

2.3. Inverse Σχήμα Αναζήτησης

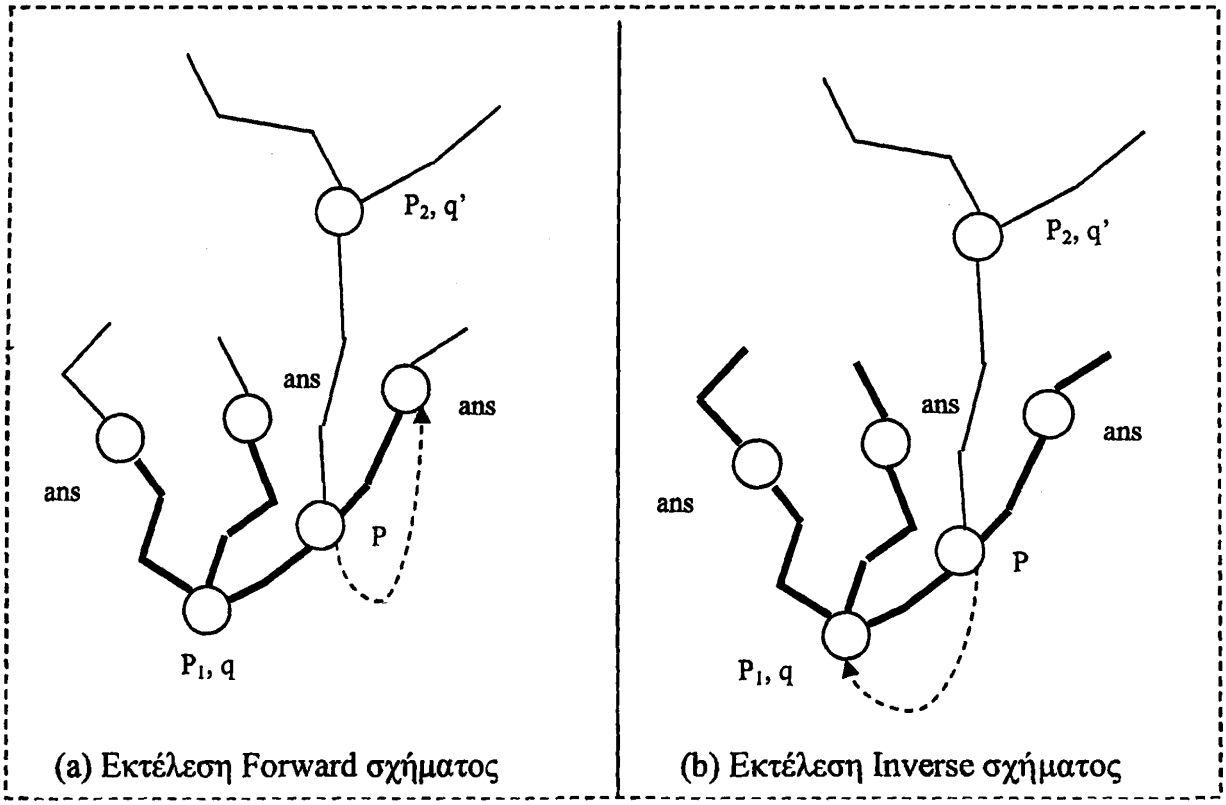
Η βασική ιδέα του Inverse σχήματος αναζήτησης αποτελεί στην ουσία μια αντιστροφή του σκεπτικού που αναλύσαμε για το Forward σχήμα. *Στο Inverse σχήμα οι κόμβοι καταγράφουν για κάθε ερώτηση που συναντούν, τον γείτονα από τον οποίο έφτασε η ερώτηση. Συνεπώς, στόχος του Inverse είναι να οδηγήσει τα ερωτήματα στους κόμβους εκείνους που προηγουμένως είχαν υποβάλλει παρόμοια ερωτήματα.* Στο Inverse σχήμα αναζήτησης η υπόθεση του Forward αντιστρέφεται και αυτή με την σειρά της και ορίζεται πλέον ως εξής: ο κόμβος που ρωτάει κατά το παρελθόν ένα ερώτημα q, είναι σε θέση στο μέλλον να δώσει απάντηση για q' παρόμοια του q. Με άλλα λόγια, σε αυτήν την περίπτωση γίνεται μια καταγραφή των ενδιαφερόντων που έχει κάθε κόμβος στο δίκτυο και εκφράζεται με βάση τις ερωτήσεις που αυτός πραγματοποιεί.

2.3.1. Σύγκριση Ανάμεσα στο Inverse και στο Forward Σχήμα Αναζήτησης

Για να κατανοήσουμε καλύτερα το Inverse σχήμα και να εντοπίσουμε τις διαφορές του σε σχέση με το Forward θα περιγράψουμε δυο σενάρια εκτέλεσης των δύο τακτικών σε δύο διαφορετικά περιβάλλοντα όπως αυτά απεικονίζονται στα σχήματα 2.4 και 2.5 αντίστοιχα. Στα 2.4 (a) και 2.5 (a) εκτελείται το Forward σχήμα ενώ στα 2.4 (b) και 2.5 (b) το Inverse.

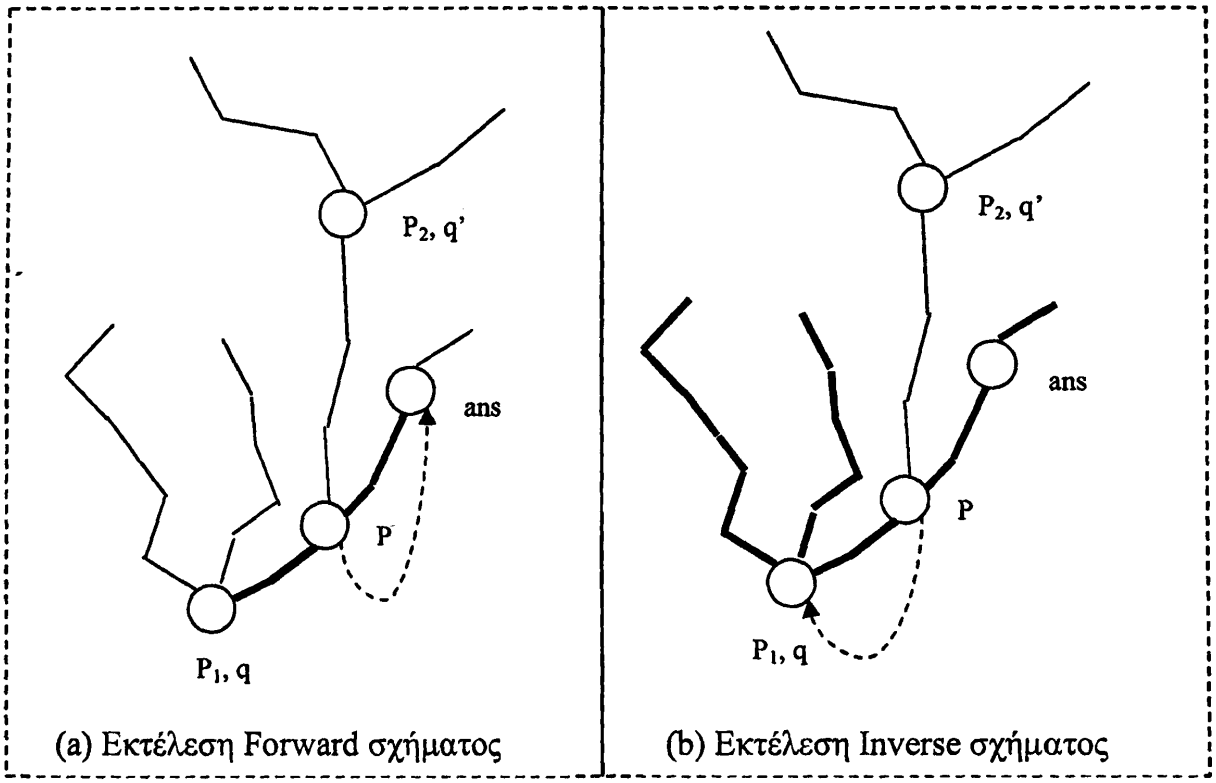
Στο πρώτο περιβάλλον (σχήμα 2.4) θεωρούμε ότι υποβάλλεται μια ερώτηση q η οποία ικανοποιείται από ένα μεγάλο πλήθος κόμβων του δικτύου (*διάσημη ερώτηση*). Στο σχήμα 2.4 (a) εκτελείται το Forward σχήμα αναζήτησης, ο P_1 υποβάλει την q εκτελώντας έστω 3 περίπατους και έστω ότι σε κάθε περίπατο υπάρχει ένας κόμβος που ικανοποιεί την q (συμβολίζονται ως ans (answering peers)). Με έντονη γραμμή συμβολίζονται οι κόμβοι οι οποίοι εισάγουν στη cache τους την πληροφορία ότι ο κόμβος ans παρείχε απάντηση στην q. Στην συνέχεια ο P_2 υποβάλει μια ερώτηση q' όμοια της q και εκτελεί και αυτός 3 έστω περιπάτους. Μόλις η q' φτάνει στον P την οδηγεί προς τον κόμβο ans. Το ίδιο σενάριο εξελίσσεται και στο σχήμα 2.4 (b) με την διαφορά όμως ότι εδώ ακολουθείται το Inverse σχήμα. Συνεπώς μόλις η q' προωθηθεί

στον P, ο P σε αυτήν την περίπτωση θα την οδηγήσει στον κόμβο που υπέβαλε την ερώτηση q κατά το παρελθόν, ήτοι στον P_1. Σε αυτό το σενάριο παρατηρούμε πως η πληροφορία που γίνεται cache λόγω της ερώτησης q είναι ίδια όσον αφορά στο πλήθος των κόμβων που την κατέγραψαν (οι caches γεμίζουν με τον ίδιο ρυθμό). Από την άλλη όμως παρατηρούμε πως το recall στη 2.4 (b) περίπτωση ενδέχεται να είναι μεγαλύτερο σε σχέση με αυτό της 2.4 (a). Θεωρείστε την περίπτωση που η q' είναι ακριβώς ίδια με την q (similarity(q,q')=1). Σε αυτήν την περίπτωση ο P_2 θα συγκεντρώσει όλα τα έγγραφα που συγκέντρωσε ο P_1 μαζί με όλα εκείνα που θα συγκεντρώσει από τους δικούς του k τυχαίους περιπάτους.



Σχήμα 2.4 Ο P_1 υποβάλλει την q μια χρονική στιγμή t_1. Την χρονική στιγμή $t_2 > t_1$ ο P_2 υποβάλλει την q' παρόμοια της q. Σε αυτό το περιβάλλον θεωρούμε ότι οι ερωτήσεις είναι διάσημες. Στα αριστερά απεικονίζεται η εκτέλεση των ερωτημάτων στο Forward μηχανισμό ενώ στα δεξιά στο Inverse. Με ans συμβολίζονται οι κόμβοι απάντησης ενώ με έντονη γραμμή απεικονίζονται μονοπάτια στα οποία οι κόμβοι εισάγουν στην cache ένα ερώτημα

Στο δεύτερο περιβάλλον (σχήμα 2.5) θεωρούμε ότι υποβάλλεται μια ερώτηση q η οποία ικανοποιείται από λίγους κόμβους του δικτύου (*σπάνια ερώτηση*). Αυτή η περίπτωση απεικονίζεται στα σχήματα 2.5 (a) και 2.5 (b) για Forward και Inverse σχήμα αναζήτησης αντίστοιχα. Σε αυτήν την περίπτωση αν η νέα ερώτηση q' συναντήσει τον P, σε οποιοδήποτε από τα 2 σχήματα αναζήτησης το πλήθος των εγγράφων που θα εντοπίσει θα είναι το ίδιο. Όμως το πλήθος των κόμβων που καταγράφουν στις caches τους την ερώτηση q είναι μεγαλύτερο στο Inverse σχήμα παρά στο Forward (έντονες γραμμές). Αυτό σημαίνει ότι η πιθανότητα κάποια παρόμοια q' να εντοπίσει έναν κόμβο P όπου κατέχει στην cache του πληροφορία για την q είναι μεγαλύτερη. Συνεπώς και εδώ το recall για την q στο Inverse σχήμα αναζήτησης φαίνεται να είναι μεγαλύτερο σε σχέση με αυτό του Forward.



Σχήμα 2.5 Ο P_1 υποβάλλει την q μια χρονική στιγμή t_1. Την χρονική στιγμή $t_2 > t_1$ ο P_2 υποβάλλει την q' παρόμοια της q. Σε αυτό το περιβάλλον θεωρούμε ότι οι ερωτήσεις είναι σπάνιες. Στα αριστερά απεικονίζεται η εκτέλεση των ερωτημάτων στο Forward μηχανισμό ενώ στα δεξιά στο Inverse. Με ans συμβολίζονται οι κόμβοι απάντησης ενώ με έντονη γραμμή απεικονίζονται μονοπάτια στα οποία οι κόμβοι εισάγουν στην cache ένα ερώτημα

Η υπόθεση που γίνεται και στα δύο παραπάνω περιβάλλοντα, δηλαδή και στην περίπτωση κατά την οποία υποβάλλεται μια διάσημη ερώτηση αλλά και όταν υποβάλλεται μια σπάνια ερώτηση, είναι πως ο κόμβος ερώτησης P_1 βρίσκει πάντα έγγραφα κατά την αρχική αναζήτηση της q. Πρέπει να σημειώσουμε ότι στην περίπτωση κατά την όποια μια αναζήτηση είναι μη επιτυχής για μια ερώτηση q, τόσο το Forward όσο και το Inverse σχήμα δεν θα είναι σε θέση να εκτελέσουν αναζήτηση με πληροφορία για μελλοντική q' παρόμοια της q. Πιο συγκεκριμένα, στην περίπτωση του Forward η cache θα είναι κενή, ενώ στην περίπτωση του Inverse η cache θα περιέχει μεν την q αλλά στην ουσία η αναζήτηση κατευθύνεται προς έναν κόμβο που δεν έχει σχετικά έγγραφα. Το τελευταίο όμως δεν σημαίνει ότι η q στην περίπτωση του Forward θα εντοπίσει μεγαλύτερο πλήθος σχετικών εγγράφων σε σχέση με την q, αντίθετα και οι δύο αναζητήσεις είναι τυχαίες. Απλά στην περίπτωση του Inverse υπάρχει ένα έξτρα αποθηκευτικό (storage) και επεξεργαστικό (processing) κόστος χωρίς κάποιο όφελος. Όπου όφελος, η εκτέλεση αναζήτησης με πληροφορία.

2.3.2. Βασικός Μηχανισμός Προώθησης Ερωτημάτων

Στο Inverse σχήμα αναζήτησης δεν υπάρχει QueryHit μήνυμα όπως είχαμε στην περίπτωση του Forward. Εδώ οι caches των κόμβων γεμίζουν απλά και μόνο από την ερώτηση που λαμβάνουν. Μια ερώτηση-μήνυμα q αποτελείται από τα παρακάτω πεδία:

<QID, List of Terms, TTL>

Είναι ακριβώς το ίδιο μήνυμα με αυτό του Forward σχήματος μόνο που εδώ δεν εισέρχεται η λίστα με τους κόμβους που έχουν είδη λάβει την ερώτηση μιας και δεν υπάρχει QueryHit μήνυμα συνεπώς δεν υπάρχει και λόγος να κρατείται το αντίστροφο μονοπάτι που ακολούθησε η ερώτηση. Εδώ μόλις ένας κόμβος ικανοποιήσει μια q δημιουργεί σύνδεση με τον κόμβο που υπέβαλλε την ερώτηση δίνοντας του τα σχετικά ως προς την q έγγραφα.

Ο βασικός μηχανισμός του Inverse σχήματος αποτελείται από 4 τμήματα:

1. Μια *cache* στην οποία ο κάθε κόμβος καταγράφει για κάθε έναν από τους γείτονες του τις πιο πρόσφατες ερωτήσεις τις οποίες έλαβε από αυτούς.
2. Μια *συνάρτηση ομοιότητας* για τον υπολογισμό ομοιότητας μεταξύ διαφορετικών ερωτημάτων που υποβάλλονται στο δίκτυο.
3. Μια *συνάρτηση σκορ* μέσω της οποίας κάθε κόμβος μόλις λαμβάνει μια ερώτηση θα υπολογίζει το σκορ που αντιστοιχεί σε κάθε γείτονα του ως προς μια συγκεκριμένη ερώτηση.
4. Ένα *βασικός μηχανισμός αναζήτησης* που θα ορίζει με ποιόν τρόπο θα επικοινωνούν οι κόμβοι του δικτύου μεταξύ τους.

Τα 2 και 4 είναι ακριβώς τα ίδια όπως περιγράφτηκαν στο Forward σχήμα αναζήτησης. Εκείνα τα τμήματα που διαφοροποιούνται λίγο είναι τα 1 και 3 τα οποία ορίζουμε παρακάτω.

Δομή και Διαχείριση της Cache

Η cache του κάθε κόμβου στο δίκτυο γεμίζει μέσω των ερωτημάτων-μηνυμάτων. Για παράδειγμα η cache του κόμβου A στο σχήμα 2.1 αφότου λάβει την q από τον κόμβο P θα φαίνεται όπως στον πίνακα 2.2:

Πίνακας 2.2 Περιεχόμενο της cache του κόμβου A μετά το τέλος εκτέλεσης της ερώτησης q

Query ID	**Set of Terms**	**Peer from which query received**	**Timestamp**
q	$\{t_1,t_2\}$	P	T_1
....	.....	.....	.....

Κάθε εγγραφή της cache αντιστοιχεί σε μια ερώτηση που υποβλήθηκε στο δίκτυο. Στην περίπτωση κατά την οποία ένας κόμβος λάβει την ίδια ερώτηση περισσότερες από μία φορές τότε θα εισάγει στην cache μόνο έναν γείτονα και συγκεκριμένα από τον πρώτο που την έλαβε. Η cache replacement πολιτική που θα εφαρμόσουμε και εδώ θα είναι ο LRU παρόμοια με το Forward σχήμα.

Η σωστή ερμηνεία του πίνακα 2.2 αναφέρει πως από το υπογράφημα που ξεκινάει από τον γείτονα P του A υπάρχει ένας κόμβος μέσα σε αυτό που τον απασχόλησε κατά το παρελθόν (υπέβαλλε) μια ερώτηση q. Με αυτόν τον τρόπο εάν λάβει ο A στο μέλλον μια ερώτηση q' παρόμοια ως προς την q θα την οδηγήσει προς το υπογράφημα που ξεκινάει από τον P.

Συνάρτηση Σκορ

Όταν ένας κόμβος λάβει μια ερώτηση q τότε υπολογίζει το σκορ για κάθε γείτονα του με βάση την εξής συνάρτηση:

$$Score(P_i, q) = \sum_{j \text{ queries answeared by } p_i} sim(q_j, q)$$

Σε σχέση με το αντίστοιχο κριτήριο του Forward μηχανισμού δεν υπάρχει στην συνάρτηση σκορ το πλήθος των εγγράφων που τελικά συγκέντρωσε κάθε ένας από τους κόμβους ερώτησης των q_j ερωτημάτων που βρίσκονται στην cache. Συνεπώς σε αυτήν την περίπτωση, η q προωθείται σε εκείνο το υπογράφημα του όποιου οι κόμβοι ρώτησαν κατά το παρελθόν τις περισσότερες παρόμοιες ως προς την q ερωτήσεις.

2.3.3. Ιδιότητα Εκμάθησης και Τυχαίες Προωθήσεις Ερωτημάτων

Στο Inverse σχήμα αναζήτησης επιθυμούμε όπως και στο Forward το recall του συστήματος να αυξάνεται όταν παρόμοιες ερωτήσεις υποβάλλονται σε αυτό. Όταν υποβάλλει ένας κόμβος μια ερώτηση στο δίκτυο και οι ενδιάμεσοι κόμβοι που λαμβάνουν την ερώτηση δεν μπορούν να επιτελέσουν προώθηση με πληροφορία (π.χ έχουν κενή cache) σε αυτήν την περίπτωση η προώθηση γίνεται με τυχαίο τρόπο. Αν και αυτή η τυχαία επιλογή στην αρχή επηρεάζει το recall για όλες τις επόμενες παρόμοιες ερωτήσεις που θα υποβληθούν, ο μηχανισμός του Inverse δεν χρησιμοποιεί QueryHits μηνύματα για να γνωρίζει πόσο είναι το πλήθος των εγγράφων που τελικά συγκέντρωσε ένα ερώτημα που υποβλήθηκε κατά το παρελθόν, απλά οδηγεί την νέα παρόμοια ερώτηση σε αυτόν. Συνεπώς με αυτόν τον τρόπο η πρώτη τυχαία επιλογή επηρεάζει το recall όλων των μελλοντικών παρόμοιων ερωτημάτων.

Πέρα όμως της πρώτης τυχαίας επιλογής εδώ παρατηρείται και ένα άλλο φαινόμενο που εμποδίζει την αύξηση του recall. Αν υποθέσουμε ότι έχουν υποβληθεί πολλές ίδιες ερωτήσεις στο δίκτυο τότε προωθώντας ένα ερώτημα μόνο προς κόμβους που κατά το παρελθόν υπέβαλαν παρόμοια ερωτήματα τότε στην ουσία η ερώτηση κατευθύνεται προς περιοχές που έχουν ήδη εξερευνηθεί. Συνεπώς κάθε νέα ερώτηση δεν θα είναι σε θέση να ανακαλύπτει καινούργια έγγραφα πλην εκείνων που συγκέντρωσαν οι προηγούμενοι κόμβοι ερώτησης.

Για τους δυο παραπάνω λόγους και στο Inverse σχήμα αναζήτησης εκτός από τις προωθήσεις με πληροφορία επιλέγουμε να γίνονται και τυχαίες επιλογές γειτόνων για την προώθηση των ερωτημάτων. Ποιο συγκεκριμένα στην υλοποίηση των k-walkers ο κόμβος που υποβάλει την ερώτηση επιλέγει οι k/2 περίπατοι να επιτελέσουν αναζήτηση με πληροφορία (εάν καθίσταται δυνατό) ενώ οι άλλοι k/2 να κάνουν τυχαίες επιλογές. Όμοια στην περίπτωση του constrained flooding επιλέγουμε ο κάθε ενδιάμεσος κόμβος να επιλέγει τους μισούς γείτονες του με βάση την συνάρτηση σκορ (εάν καθίσταται δυνατό) και τους άλλους μισούς με τυχαίο τρόπο.

ΚΕΦΑΛΑΙΟ 3. ΠΕΙΡΑΜΑΤΙΚΑ ΑΠΟΤΕΛΕΣΜΑΤΑ

3.1. Εισαγωγικά

Σε αυτό το κεφάλαιο εκτελούμε μια σειρά πειραμάτων που σαν στόχο έχουν να συγκρίνουν κάτω από διαφορετικές οπτικές γωνίες τους αλγορίθμους αναζήτησης που περιγράψαμε στο 3° κεφάλαιο. Υλοποιήσαμε 3 διαφορετικούς αλγορίθμους αναζήτησης:

1. Random σχήμα αναζήτησης.
2. Inverse σχήμα αναζήτησης.
3. Forward σχήμα αναζήτησης.

Ο κάθε αλγόριθμος υλοποιήθηκε με βασικό σχήμα αναζήτησης τους k-walkers και το constrained flooding σχήμα. Συνεπώς εξετάζουμε 6 διαφορετικές υλοποιήσεις.

Για την υλοποίηση των μεθόδων μας, χρησιμοποιήσαμε έναν ανοιχτού κώδικα προσομοιωτή τον OMNeT++[6]. Πρόκειται για ένα διακριτό βασιζόμενο σε γεγονότα (event-based) περιβάλλον προσομοίωσης υλοποιημένο σε γλώσσα C++ που χρησιμοποιείται ευρέως σε προσομοιώσεις δικτύων επικοινωνίας.
Θεωρούμε 2 μετρικές πάνω στις οποίες θα πραγματοποιηθεί η σύγκριση των παραπάνω τεχνικών αναζήτησης:

1. *Recall*, ως προς μια ερώτηση q, είναι ο λόγος του πλήθους των εγγράφων που ανέκτησε ο αλγόριθμος ως προς το συνολικό πλήθος των σχετικών ως προς την q εγγράφων που υπάρχουν σε όλο το δίκτυο.
2. *Overlapping recall*, ορίζεται στην ουσία με τον ίδιον τρόπο που ορίζεται το recall μόνο που εδώ στον αριθμητή αθροίζονται όλες οι εμφανίσεις των εγγράφων ακόμα και για ένα συγκεκριμένο που εντοπίζεται πολλαπλές φορές. Διαφορετικά, στον αριθμητή δεν παίρνουμε τις μοναδικές εμφανίσεις κάθε εγγράφου αλλά το σύνολο των εμφανίσεων.

Η πρώτη μετρική αποτελεί μια από τις κυριότερες μετρικές για αλγορίθμους αναζήτησης στο πεδίο της ανάκτησης πληροφορίας. Η δεύτερη μετρική αποτελεί για μας το μέτρο σύγκρισης ανάμεσα στον Random αλγόριθμο και τις 2 μεθόδους αναζήτησης με πληροφορία. Θέλουμε να τονίσουμε ότι οι 2 μέθοδοι αναζήτησης με πληροφορία που υλοποιούμε, τόσο δηλαδή το Forward όσο και το Inverse σχήμα σαν στόχο τους έχουν να αυξήσουν το overlapping recall. Το οποίο σημαίνει ότι εάν εξετάζουμε συνολικά x σε πλήθος κόμβους για μια ερώτηση q θέλουμε όσο το δυνατόν μεγαλύτερο ποσοστό από τους x να είναι σχετικοί ως προς την q. Θέλουμε δηλαδή να πούμε ότι οι μέθοδοι με αναζήτηση δεν κάνουν κάτι ιδιαίτερο ώστε να βρίσκουν συνεχώς και ξεχωριστά έγγραφα ο βασικός στόχος τους είναι απλά να βρίσκουν σχετικά έγγραφα. Προφανώς όσο ποιο μεγάλο είναι το overlapping recall τόσο θα αυξάνει και το recall. Συνεπώς αναφερόμαστε σε 2 κριτήρια τα οποία το μεν πρώτο δείχνει το πραγματικό κέρδος, όσο μεγαλύτερο τόσο καλύτερη και η μέθοδο που το εμφανίζει, ενώ το δεύτερο κριτήριο μαρτυρά την πληροφορία που χρησιμοποιεί κατά την προώθηση των ερωτημάτων η μέθοδος αναζήτησης.

Για να μπορέσουμε να συγκρίνουμε τους παραπάνω αλγορίθμους θα πρέπει όλοι να παρουσιάζουν το ίδιο μέγεθος αναζήτησης. Δηλαδή το μέσο πλήθος κόμβων που

εξετάζει κατά μέσο όρο κάθε ερώτηση πρέπει για όλες τις μεθόδους να είναι αν όχι ακριβώς το ίδιο, παρόμοιο. Όπως έχουμε ήδη αναφέρει το μέγεθος αναζήτησης για κάθε μέθοδο ορίζεται από το σχήμα των k-walkers ή αυτό του constrained flooding που θα χρησιμοποιήσουμε ανά περίσταση.

Σε όλα τα πειράματα που πραγματοποιήσαμε, το ενδιαφέρον μας επικεντρώθηκε στους εξής δύο βασικούς άξονες: Πρώτον, θέλουμε να παρατηρήσουμε πως συμπεριφέρεται το Inverse σε σχέση με το Forward σχήμα ως προς το recall κάτω από διαφορετικά περιβάλλοντα. Δεύτερον, επιθυμούμε να δούμε τη συμπεριφορά του Random σχήματος σε σχέση με τις τακτικές αναζήτησης με πληροφορία. Θέλουμε δηλαδή να δούμε πόσο μια lazy-based αναζήτηση μπορεί να βελτιώσει το recall σε σχέση με μια απλοϊκή τυχαία επιλογή προώθησης μηνυμάτων.

3.2. Δημιουργία Δικτύου και Ερωτημάτων

Θεωρούμε ως P2P δίκτυο ένα τυχαίο, μη κετευθυνόμενο γράφημα με συνολικό πλήθος κόμβων ίσο με 1000 και μέσο πλήθος γειτόνων για κάθε κόμβο ίσο με 5. Πιο συγκεκριμένα κάθε κόμβος έχει πιθανότητα 0.5% να συνδεθεί με έναν οποιοδήποτε άλλο κόμβο στο δίκτυο πλην του εαυτό του. Να σημειώσουμε ότι το δίκτυο σε όλα τα πειράματα-περιβάλλοντα που θα εκτελέσουμε θα είναι το ίδιο.

Κάθε Σύνολο Ερωτήσεων (Query Set (Q.S)) που θα χρησιμοποιήσουμε αποτελείται από ένα σύνολο ξεχωριστών ερωτήσεων που θα επαναλαμβάνονται η κάθε μια εξ αυτών ένα σταθερό πλήθος φορών. Με αυτόν τον τρόπο αφενός μεν θα μπορέσουμε να επιτύχουμε ισχυρή locality στο ρεύμα ερωτήσεων, αφετέρου δε θα είναι και ελεγχόμενη, θα μπορούμε δηλαδή να διαπιστώσουμε πόσο μεγάλη η μικρή locality χρειάζονται οι αλγόριθμοι μας με πληροφορία για να βελτιώνουν το recall.

Επιπλέον κάθε Q.S ορίζει και τον τρόπο με τον οποίο κατανέμονται τα έγγραφα στους κόμβους του δικτύου. Κάθε ένα Q.S έχει το ίδιο ποσοστό κόμβων που παρέχουν απάντηση σε κάθε μια από τις ερωτήσεις του Q.S. Αυτό το γεγονός το ορίζουμε ως Q.S x%, όπου x δηλώνει το % ποσοστό των κόμβων στο δίκτυο που

είναι σχετικοί ως προς κάθε μια ερώτηση του Q.S. Πιο συγκεκριμένα, η δημιουργία ενός Q.S γίνεται ως εξής: Αρχικά δημιουργούμε ένα σύνολο από ξεχωριστές ερωτήσεις. Η κάθε μια εξ αυτών αποτελείται από 2 όρους (σύμφωνα με το [7] το μέσο πλήθος όρων που είχαν τα ερωτήματα που υποβλήθηκαν στο http://www.yahoo.co.uk από Νόεμβριο του 2005 εώς και Νοέμβριο του 2006 ήταν 2.5) και κάθε όρος αναπαρίσταται από 4 χαρακτήρες. Για κάθε μια από τις ξεχωριστές ερωτήσεις του Q.S, κάθε κόμβος στο δίκτυο επιλέγει έναν αριθμό από [1,100]. Εάν η τιμή που προκύπτει είναι μικρότερη του x τότε αυτός ο κόμβος οφείλει να είναι κόμβος απάντησης ως προς την συγκεκριμένη ερώτηση. Εν συνεχεία μόλις ένας κόμβος επιλέγεται ως απαντών για μια ερώτηση επιλέγει ομοιόμορφα στο διάστημα [1,30] για το πλήθος των σχετικών εγγράφων που τελικά θα εισαχθούν στην LDC του.

Είναι σημαντικό να τονίσουμε πως το πλήθος των κόμβων του δικτύου με τον τρόπο με τον οποίο αυτοί συνδέονται μεταξύ τους καθώς και ένα δοθέν Q.S ορίζουν στο περιβάλλον μας το δίκτυο. Το πρώτο ορίζει την φυσική αναπαράσταση του δικτύου ενώ το δεύτερο το περιεχόμενο (LDC) των κόμβων.

Για να μπορέσουμε να συγκρίνουμε τις 3 τεχνικές αναζήτησης μεταξύ τους, το Q.S δημιουργείται εκ των προτέρων και υποβάλλεται το ίδιο σε κάθε έναν από τους αλγορίθμους αναζήτησης σε ένα δοθέν δίκτυο. Συνεπώς όταν 'ρίχνουμε' ένα Q.S σε ένα δίκτυο που ακολουθεί ένα από τα 3 σχήματα αναζήτησης, όποιο και εάν είναι αυτό, πάντα θα υπάρχει ο ίδιος κόμβος που θα υποβάλλει την ίδια ερώτηση την ίδια χρονική στιγμή. Επιπρόσθετα, κάθε μια από τις ερωτήσεις στο Q.S υποβάλλεται σε σχέση με την προηγούμενη της με διαφορά 200 χρονικών στιγμών αυτό το επιβάλλαμε κυρίως λόγω του Forward σχήματος έτσι ώστε να του δώσουμε το χρονικό περιθώριο να αποθηκεύει στην cache του ο κάθε κόμβος τα ερωτήματα. Να σημειώσουμε ότι λόγω του ότι η cache στο Forward σχήμα 'γεμίζει' μέσω των QueryHits μηνυμάτων απαιτεί μεγαλύτερο χρονικό διάστημα για να σταθεροποιηθεί σε σχέση με το Inverse σχήμα.

Για παράδειγμα, αν αναφερόμαστε σε μια ερώτηση που εξετάζει 30 κόμβους και κάθε βήμα (hop) στην αναζήτηση απαιτεί μια χρονική στιγμή προσομοίωσης (χρόνος

επικοινωνίας για να αποσταλεί ένα μήνυμα μεταξύ δύο γειτόνων στον προσομοιωτή) τότε μέσα σε 30 χρονικές στιγμές οι κόμβοι στο Inverse σχήμα αναζήτησης που οφείλουν να αποθηκεύσουν το ερώτημα (αυτοί που το έλαβαν) στην cache τους, έχουν ολοκληρώσει την διαδικασία εισαγωγής. Αντίθετα, στο Forward σχήμα στην χειρότερη περίπτωση αυτός ο χρόνος μπορεί να είναι ο διπλάσιος αν σκεφτούμε την πιθανότητα ο μεγαλύτερος σε βάθος αναζήτησης κόμβος να είναι και απαντών κάτι που σημαίνει ότι θα απαιτηθούν άλλα 30 βήματα για να μεταδοθεί το QueryHit προς το αντίστροφο μονοπάτι και να φτάσει στον κόμβο που υπέβαλε την ερώτηση. Συνεπώς η διαφορά των 200 χρονικών στιγμών ανά υποβολή ερώτησης στο δίκτυο συνεπάγεται ότι σε κανένα από τα πειράματα μας δεν θα έχουμε μέγεθος αναζήτησης μεγαλύτερο από 100 χρονικές στιγμές προσομοίωσης.

Στο πίνακα 3.1 παρουσιάζονται οι συμβολισμοί και οι ερμηνείες όλων των παραμέτρων των πειραμάτων. Επιπλέον, παρουσιάζονται και οι τιμές εκείνων που παραμένουν σταθερές κατά την εκτέλεση όλων των πειραμάτων.

Πίνακας 3.1 Συμβολισμοί και ερμηνεία των παραμέτρων καθώς και τιμές για τις παραμέτρους που παραμένουν σταθερές σε όλες τις εκτελέσεις των πειραμάτων

Συμβολισμός Παραμέτρου	Ερμηνεία Παραμέτρου	Τιμή παραμέτρου
N	Πλήθος κόμβων στο δίκτυο	1000
d	Μέσο πλήθος γειτόνων για κάθε κόμβο	5
k	Πλήθος περιπάτων	Ανάλογα με την εκτέλεση
TTL	Βάθος αναζήτησης εκφρασμένο σε πλήθος βημάτων	Ανάλογα με την εκτέλεση
%forward	Το % ποσοστό των γειτόνων κάθε κόμβου που προωθεί ένα ερώτημα (αφορά το constrained flooding σχήμα)	50%
Q.S AxB	Σύνολο Ερωτήσεων που αποτελείται από Α μοναδικές ερωτήσεις που επαναλαμβάνονται Β φορές η κάθε μια	Ανάλογα με την εκτέλεση
Q.S %x	Το % ποσοστό των κόμβων του δικτύου που είναι σχετικοί ως προς κάθε μια ερώτηση του Q.S	Ανάλογα με την εκτέλεση

3.3. Σύγκριση των Σχημάτων με Βασικό Μηχανισμό Αναζήτησης k-walkers

Το κύριο χαρακτηριστικό του Inverse σχήματος έναντι του Forward είναι πως οι caches στο Inverse γεμίζουν με μεγαλύτερο ρυθμό σε σχέση με το Forward. Πιο συγκεκριμένα, δοθείσας μιας ερώτησης q που υποβάλλεται στο δίκτυο το πλήθος των κόμβων που θα εισάγουν την q στην cache τους είναι ανάλογο του πλήθους των συνολικών κόμβων που θα εξετάσει η q, δηλαδή ο αριθμός των εισαγωγών εξαρτάται από το μέγεθος της αναζήτησης.

Αντίθετα, στην περίπτωση του Forward σχήματος για να εισάγει ένας κόμβος μια q στην cache του θα πρέπει πρώτα να βρεθεί κατά την αναζήτηση της κάποιος απαντών κόμβος ο οποίος και θα παράγει το QueryHit μήνυμα το οποίο κατά την δρομολόγηση του προς τα πίσω οι κόμβοι θα το εισάγουν στις caches τους. Συνεπώς σε αυτήν την

περίπτωση όσο περισσότεροι είναι οι κόμβοι απάντησης που συναντά μια ερώτηση τόσο μεγαλώνει και το πλήθος των κόμβων που την εισάγουν στις cache τους. Το πλήθος των κόμβων απάντησης εξαρτάται από δύο παράγοντες: Πρώτον από το μέγεθος της αναζήτησης και δεύτερον από το πόσοι κόμβοι στο δίκτυο έχουν σχετικά έγγραφα ως προς μία ερώτηση. Όσο μεγαλύτερη τιμή φέρει κάθε μια από τις παραπάνω παραμέτρους τόσο μεγαλύτερο είναι και το πλήθος των κόμβων απάντησης που συναντά μια ερώτηση.

Το χαρακτηριστικό του πόσο γρήγορα-εύκολα γεμίζουν οι caches των κόμβων του δικτύου επηρεάζει το recall και το overlapping recall του κάθε μηχανισμού. Όσο μεγαλύτερη πληροφορία έχει στην cache ένας κόμβος τόσο μεγαλύτερο αριθμό προωθήσεων των ερωτημάτων με πληροφορία είναι σε θέση να εκτελέσει.

Σε αυτήν την ενότητα έχοντας σαν βασικό μηχανισμό αναζήτησης τους k-walkers διεξάγουμε 4 πειράματα με στόχο να εξετάσουμε τη συμπεριφορά του Inverse και Forward μηχανισμού όταν μεταβάλλουμε το πλήθος των κόμβων απάντησης που συναντά κατά μέσο όρο κάθε ερώτημα. Με αυτόν τον τρόπο είμαστε σε θέση να ελέγχουμε τον ρυθμό με τον οποίο γεμίζουν οι caches στο Forward σχήμα. Στον πίνακα 3.2 παρουσιάζονται συγκεντρωτικά οι τιμές των παραμέτρων για κάθε ένα από τα 4 πειράματα.

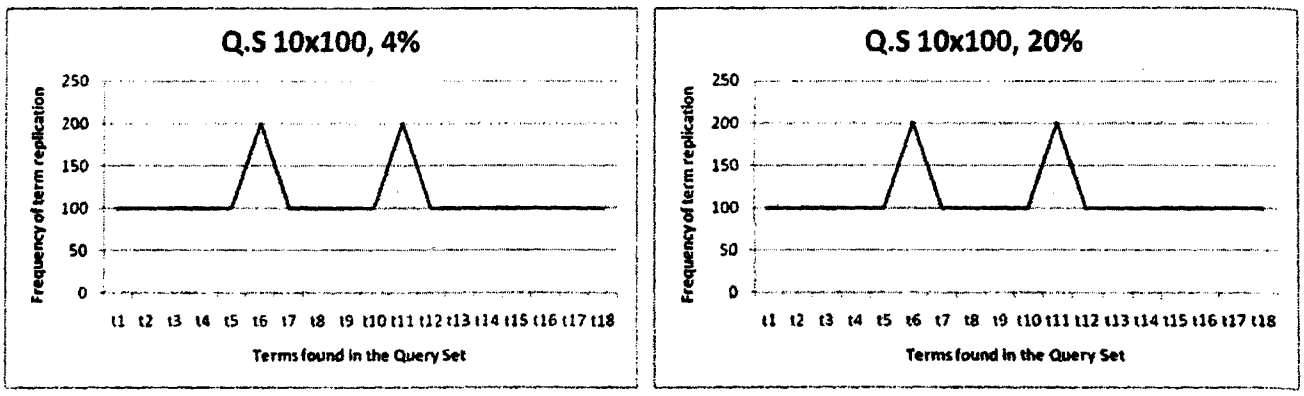
Πίνακας 3.2 Τιμές παραμέτρων για τα πειράματα των ενοτήτων 3.3.1, 3.3.2, 3.3.3 και 3.3.4 αντίστοιχα

	3.3.1	**3.3.2**	**3.3.3**	**3.3.4**
k	3	5	5	5
TTL	3	5	10	20
Q.S	4%	4%	4%	20%
Num of queries	10x100	10x100	10x100	10x100

Τα πειράματα 3.3.1, 3.3.2 και 3.3.3 εκτελούνται πάνω στο ίδιο δίκτυο με το ίδιο Q.S και προσομοιώνουν περιπτώσεις κατά τις οποίες οι caches των κόμβων του Inverse σχήματος γεμίζουν με μεγαλύτερο ρυθμό σε σχέση με το Forward σχήμα.

Στο πείραμα 3.3.4 προσομοιώνουμε την περίπτωση κατά την οποία τόσο οι caches του Inverse όσο και αυτές του Forward γεμίζουν με τον ίδιο ρυθμό. Όπως φαίνεται και στον πίνακα 3.2 για να το πετύχουμε αυτό δημιουργούμε ένα νέο Q.S στο οποίο το πλήθος των κόμβων που απαντάνε σε κάθε μια ερώτηση είναι πενταπλάσιο σε σχέση με πριν και το μέγεθος της αναζήτησης διπλάσιο σε σχέση με αυτό που θέσαμε στο πείραμα 3.3.3.

Τα δυο σύνολα ερωτήσεων που δημιουργούμε αποτελούνται από 10 ξεχωριστές ερωτήσεις που η κάθε μια εξ αυτών υποβάλλεται στο δίκτυο 100 φορές. Συνεπώς αναφερόμαστε σε 1000 ερωτήσεις συνολικά όπου κάθε κόμβος στο δίκτυο υποβάλλει και από μια ερώτηση. Η διαφορά των δυο Q.S εντοπίζεται στο πλήθος των κόμβων απάντησης, στο μεν πρώτο έχουμε Q.S 4%, δηλαδή υπάρχουν 40 κόμβοι που απαντάνε κάθε μια από τις ερωτήσεις του Q.S. Ενώ στο δεύτερο, έχουμε αυξήσει το πλήθος των κόμβων που παρέχουν απάντηση στο 20% των κόμβων του δικτύου, ήτοι σε 200. Η term locality και για τα δυο Q.S που δημιουργήσαμε είναι ίδια και εμφανίζεται στο σχήμα 3.1.



Σχήμα 3.1 Δεξιά εμφανίζεται η term locality για το Q.S 4% και αριστερά η term locality για Q.S 20%, κάθε μια ερώτηση αποτελείται από 2 όρους

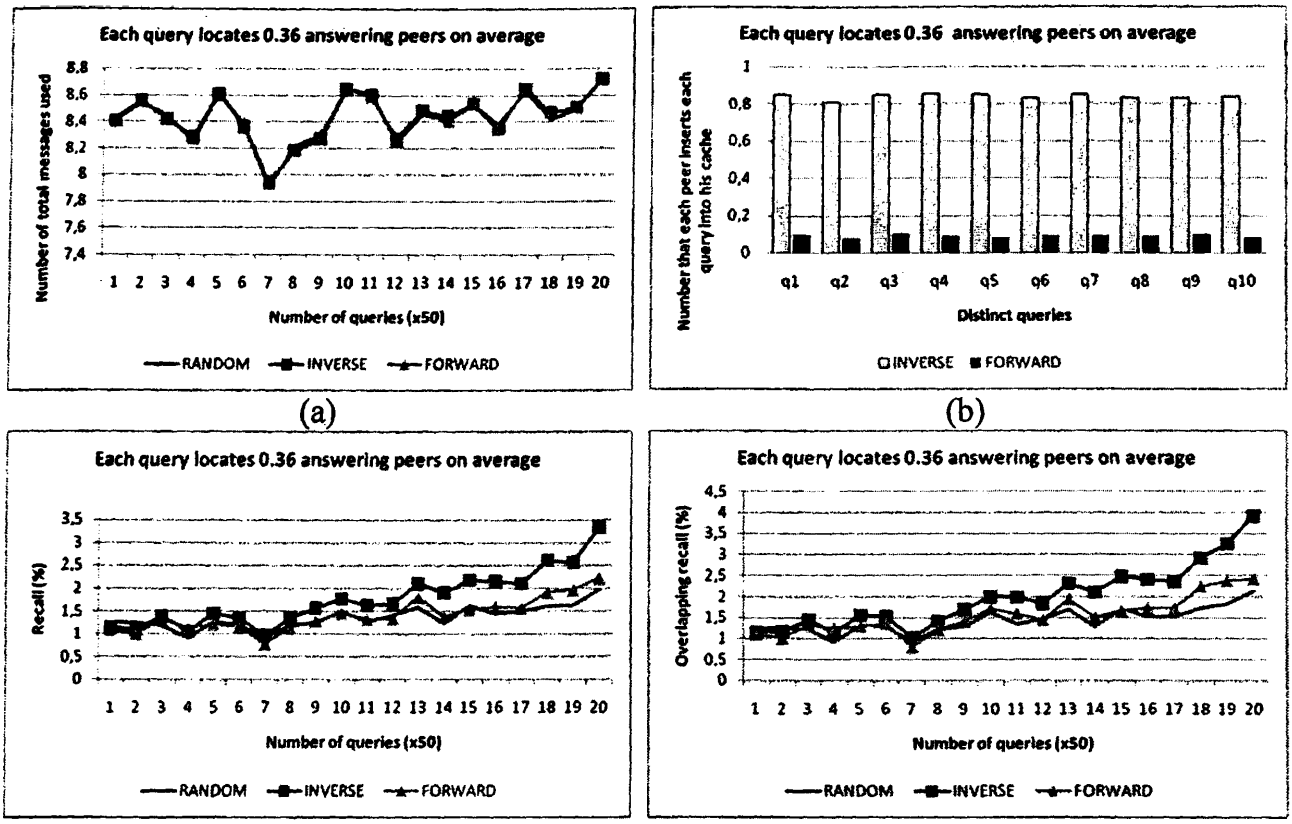
Τέλος, επειδή μας ενδιέφερε σε όλα τα πειράματα αυτής της ενότητας να γνωρίζουμε το περιεχόμενο των caches, για κάθε μια από τις δύο μεθόδους αναζήτησης με πληροφορία, αφήσαμε τις caches να γεμίζουν χωρίς κανέναν περιορισμό χωρίς δηλαδή να εφαρμόσουμε τον LRU αλγόριθμο.

3.3.1. Κάθε Ερώτηση στο Q.S Απαντάται Δύσκολα

Σε αυτό το πείραμα στόχος μας είναι να παρατηρήσουμε την συμπεριφορά των 2 τεχνικών αναζήτησης με πληροφορία όταν το πλήθος των κόμβων απάντησης που συναντά κάθε ερώτηση στο Q.S είναι πάρα πολύ μικρό. Πιο συγκεκριμένα θέλουμε να δούμε εάν και κατά πόσο το Inverse σχήμα που αποθηκεύει μη χρήσιμη πληροφορία στις caches των κόμβων επηρεάζει και κατά πόσο την αναζήτηση. Οι παράμετροι του πειράματος δίνονται στην στήλη 3.3.1 του πίνακα 3.2.

Θέτοντας πλήθος περιπάτων ίσο με 3 και βάθος αναζήτησης ίσο με 3 το μέγεθος της αναζήτησης θα είναι το πολύ 9 κάτι που συνεπάγεται ότι το πλήθος των κόμβων απάντησης που θα εντοπίζει κάθε μια ερώτηση θα είναι ίσο με (40*9)/1000=0.36. Αυτό σημαίνει ότι στην πρώτη υποβολή κάθε μιας από τις 10 ξεχωριστές ερωτήσεις είναι δύσκολο να βρεθούν απαντήσεις. Βέβαια στην συνέχεια και εφόσον η κάθε μια ερώτηση επαναλαμβάνεται 100 φορές το πλήθος των κόμβων απάντησης μεγαλώνει μιας και όσοι κόμβοι ερώτησης κατάφεραν κατά το παρελθόν να εντοπίσουν σχετικά έγγραφα τα αποθηκεύουν μόνιμα στην LDC τους. Συνεπώς στην χειρότερη περίπτωση αν θεωρήσουμε ότι οι προηγούμενοι 99 κόμβοι ερώτησης βρήκαν απαντήσεις τότε το πλήθος των κόμβων απάντησης όταν θα υποβληθεί για $100^{η}$ φορά μια οποιαδήποτε ξεχωριστή ερώτηση θα είναι (40+99)*9/1000=1.251.

Στο σχήμα 3.2 απεικονίζονται 4 γραφικές παραστάσεις. Σε κάθε μια γραφική παράσταση εμφανίζονται τιμές για κάθε ένα από τα τρία διαφορετικά σχήματα αναζήτησης που συγκρίνουμε, ήτοι το Random, Inverse και Forward, εκτός από την γραφική (b) μιας και το Random σχήμα δεν χρησιμοποιεί cache για να αποθηκεύει τις ερωτήσεις.



Σχήμα 3.2 Σε δίκτυο με 1000 κόμβους υποβάλλονται 10 ξεχωριστές ερωτήσεις που επαναλαμβάνονται 100 φορές η κάθε μια. Κάθε μια ξεχωριστή ερώτηση την πρώτη φορά που υποβάλλεται βρίσκει κατά μέσο όρο 0.36 κόμβους απάντησης. Οι παράμετροι του πειράματος απεικονίζονται στη στήλη 3.3.1 του πίνακα 3.2

Στο σχήμα 3.2 (a) απεικονίζεται το πλήθος των συνολικών μηνυμάτων που παράγουν κατά μέσο όρο οι ερωτήσεις που υποβάλλονται στο δίκτυο. Επειδή στο σύνολο έχουμε 1000 ερωτήματα, σε κάθε σημείο του άξονα x σχεδιάζουμε το μέσο πλήθος κόμβων που εξετάζεται σε 5 επαναλήψεις κάθε μιας από τις 10 ξεχωριστές ερωτήσεις, αναφερόμαστε δηλαδή σε σύνολο 50 ερωτήσεων. Συγκεκριμένα το σημείο 1 του άξονα x αφορά το πλήθος κόμβων που εξετάστηκε στις πρώτες 5 επαναλήψεις κάθε μιας από τις 10 ξεχωριστές ερωτήσεις, το σημείο 2 αφορά την δεύτερη πεντάδα επανάληψης κ.ο.κ μέχρι να φτάσουμε στις τελευταίες 5 επαναλήψεις [96-100] για κάθε μια από τις ξεχωριστές ερωτήσεις του Q.S. Όπως αναμέναμε κάθε μια από τις ερωτήσεις εξετάζει έναν αριθμό 9 συνολικά κόμβων. Να σημειώσουμε πως το 99% των εξεταζόμενων κόμβων ήταν και ξεχωριστοί. Το σημαντικό σε αυτήν την γραφική παράσταση είναι πως για κάθε ερώτηση κάθε ένα από τα 3 εξεταζόμενα σχήματα έχει το ίδιο σχεδόν μέγεθος αναζήτησης γεγονός που τα καθιστά απόλυτα συγκρίσιμα ως προς το recall και το overlapping recall.

Στο σχήμα 3.2 (b) παρουσιάζουμε πόσες φορές κατά μέσο όρο κάθε κόμβος στο δίκτυο εισήγαγε στην cache του κάθε μια από τις 10 ξεχωριστές ερωτήσεις. Μιας και κάθε ερώτηση επαναλαμβάνεται 100 φορές και η κάθε μία έχει ένα μέγεθος αναζήτησης ίσο με 9 κάθε ξεχωριστή ερώτηση εξέτασε το πολύ 900 κόμβους. Για το μεν Inverse που κάθε βήμα (hop) συνεπάγεται και εισαγωγή στην cache, κάθε κόμβος στο δίκτυο είδε κάθε μια ερώτηση κοντά στις 0.8 φορές γεγονός που δικαιολογεί την ανάλυση μας. Αντίθετα στο Forward σχήμα αναζήτησης στο οποίο οι caches γεμίζουν μέσω των QueryHits μηνυμάτων, σε ένα περιβάλλον που κάθε ερώτηση συναντά 0.36 κόμβους απάντησης οι caches στην ουσία δεν γεμίζουν καθόλου κάτι που απεικονίζεται και στην γραφική του σχήματος 3.2 (b) με την ερώτηση q_3 να πιάνει μια μέγιστη τιμή και αυτή ίση με 0.1 φορές.

Στα σχήματα 3.2 (c) και 3.2 (d) εμφανίζεται το recall και το overlapping recall αντίστοιχα. Όσων αφορά τις 2 μεθόδους αναζήτησης με πληροφορία σε σχέση με την Random μέθοδο αναζήτησης σε ένα τέτοιο περιβάλλον οι διαφορές είναι πάρα πολύ μικρές μιας και στην ουσία οι δυο μέθοδοι με πληροφορία μετατρέπονται σε Random. Ποιο συγκεκριμένα το Forward σχήμα δεν εισάγει τίποτα μέσα στην cache, ήτοι για τις μελλοντικές παρόμοιες ερωτήσεις δεν μπορεί να επιτελέσει προωθήσεις με πληροφορία. Από την άλλη το Inverse σχήμα αποθηκεύει στην cache πληροφορία που όμως δεν οδηγεί σε κόμβους που είχαν βρει κατά το παρελθόν σχετικά έγγραφα.

Πάρα ταύτα το Inverse σχήμα μετά τις 35 πρώτες επαναλήψεις της κάθε μιας από τις ξεχωριστές ερωτήσεις αρχίζει πλέον και βελτιώνει το recall σε σχέση με τα άλλα 2 σχήματα αναζήτησης. Αυτό οφείλεται στο ότι λόγω των επαναλήψεων κάποιοι από τους προηγούμενους κόμβους ερώτησης εντόπισαν κάποια σχετικά έγγραφα συνεπώς η πληροφορία της cache αρχίζει πλέον και χρησιμοποιείται προς όφελος του αλγορίθμου. Προφανώς για να φανεί κάτι τέτοιο πρέπει στο δοθέν περιβάλλον με τους λίγους κόμβους απάντησης να εμφανιστεί μεγάλη locality στο ρεύμα ερωτήσεων.

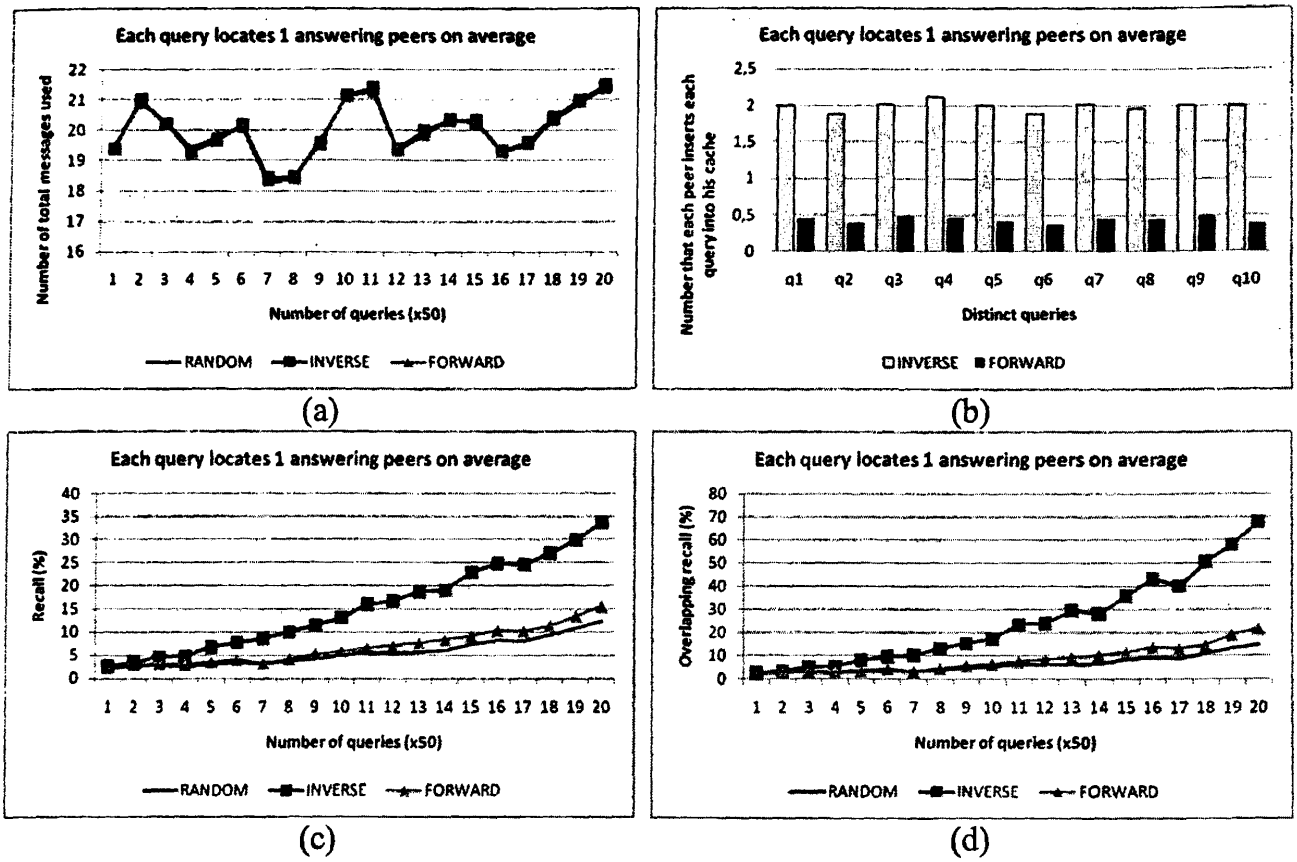
Συνεπώς το συμπέρασμα που προκύπτει από τις γραφικές του σχήματος 3.2 είναι πως το Inverse σχήμα ακόμα και όταν επιτελεί λανθασμένες προωθήσεις ερωτημάτων, προωθώντας τες σε κόμβους που δεν έχουν σχετικά έγγραφα δεν επηρεάζει το recall

του συστήματος μιας και κατά την πορεία της η ερώτηση συνεχίζει και εξετάζει με τυχαίο τρόπο κόμβους. Συνεπώς είναι πιο σωστό να διατυπώνουμε ότι σε αυτήν την περίπτωση επιτελεί τυχαίες προωθήσεις (και όχι λανθασμένες) εις βάρος ενός έξτρα επεξεργαστικού και αποθηκευτικού κόστους σε σχέση με το Forward σχήμα.

3.3.2. Κάθε Ερώτηση στο Q.S Βρίσκει Απάντηση

Σε αυτό το πείραμα θα μεγαλώσουμε το μέγεθος της αναζήτησης που κάναμε στο πείραμα 3.3.1 διατηρώντας τις άλλες παραμέτρους ίδιες με στόχο να αυξήσουμε το πλήθος των κόμβων απάντησης που κάθε ερώτημα που ανήκει στο Q.S συναντά. Οι παράμετροι του πειράματος δίνονται στην στήλη 3.3.2 του πίνακα 3.2.

Θέτοντας τον αριθμό των περιπάτων ίσο με 5 και το βάθος της αναζήτησης ίσο με 5 κάθε αναζήτηση θα εξετάζει το πολύ 25 κόμβους. Αυτό συνεπάγεται πως κάθε μια από τις ξεχωριστές ερωτήσεις την πρώτη φορά που υποβάλλονται στο δίκτυο θα είναι σε θέση να εντοπίζουν 1 κόμβο απάντησης. Διαφορετικά, μπορούμε να πούμε ότι κατά μέσο όρο ένας περίπατος θα καταλήξει σε επιτυχή αναζήτηση. Βέβαια ξανά σε κάθε επανάληψη το πλήθος των κόμβων απάντησης θα μεγαλώνει και συγκεκριμένα στην $100^{η}$ επανάληψη κάθε μιας από τις ξεχωριστές ερωτήσεις το πλήθος των απαντών κόμβων θα φτάσει στους 3.475.



Σχήμα 3.3 Σε δίκτυο με 1000 κόμβους υποβάλλονται 10 ξεχωριστές ερωτήσεις που επαναλαμβάνονται 100 φορές η κάθε μια. Κάθε μια ξεχωριστή ερώτηση την πρώτη φορά που υποβάλλεται βρίσκει κατά μέσο όρο 1 κόμβο απάντησης. Οι παράμετροι του πειράματος απεικονίζονται στη στήλη 3.3.2 του πίνακα 3.2

Στο σχήμα 3.3 εμφανίζονται οι γραφικές παραστάσεις του πειράματος 3.3.2. Για άλλη μια φορά το μέγεθος της αναζήτησης είναι κοινό και για τους τρεις αλγορίθμους όπως φαίνεται στην εικόνα 3.3 (a) ή πιο συγκεκριμένα οι αυξομειώσεις που παρατηρούνται είναι κοινές. Το γεγονός ότι παρουσιάζονται αυξομειώσεις στο πλήθος των εξεταζόμενων κόμβων ανά σύνολο ερωτήσεων καθώς και ότι δεν παίρνουμε μέγεθος αναζήτησης ίσο με 25 όπως αναλύσαμε παραπάνω οφείλεται σε τρεις λόγους: (1) Λόγω κύκλων, ένας κόμβος λαμβάνει παραπάνω από μια φορά το ίδιο μήνυμα-ερώτημα (στην υλοποίηση μας σε κάθε μια τεχνική αναζήτησης οι κύκλοι εντοπίζονται και τα διπλότυπα ερωτήματα-μηνύματα απορρίπτονται). (2) Ένας κόμβος ερώτησης ενδεχομένως να μην έχει 5 γείτονες για να ορίσει εξ αρχής 5 περιπάτους αναζήτησης και (3) επειδή ένας ενδιάμεσος κόμβος μπορεί να μην έχει άλλο γείτονα για να προωθήσει το ερώτημα πλην από αυτόν από τον οποίο το έλαβε.

Οι caches και στους δυο αλγορίθμους αναζήτησης με πληροφορία βλέπουν πλέον την κάθε μια από τις ξεχωριστές ερωτήσεις περισσότερες φορές σε σχέση με τις caches του πειράματος 3.3.1. Το Inverse σχήμα συνεχίζει όμως να γεμίζει τις caches με μεγαλύτερο ρυθμό σε σχέση με το Forward μηχανισμό αναζήτησης.

Το Inverse σχήμα αναζήτησης σε αυτήν την περίπτωση ότι ερώτημα αποθηκεύει στην cache χρησιμοποιείται εις όφελος όλων των μελλοντικών παρόμοιων ερωτημάτων κάτι που το καθιστά καλύτερο σχήμα αναζήτησης ως προς το recall από την $25^{η}$ επανάληψη κάθε ξεχωριστής ερώτησης του Q.S και μετά, πολύ γρηγορότερα δηλαδή σε σχέση με το πρώτο πείραμα. Αυτό δικαιολογείται από το γεγονός ότι κάθε κόμβος εισάγει στην cache του κατά μέσο όρο 2 φορές κάθε μια από τις 10 ξεχωριστές ερωτήσεις συνεπώς οι μισές από αυτές, δηλαδή 500 σε πλήθος, προωθούνται με σωστό τρόπο προς κόμβους ερώτησης που προηγουμένως πραγματοποίησαν επιτυχείς αναζητήσεις. Τώρα με την σειρά τους ένα ποσοστό από αυτές όντως καταφέρνουν και εντοπίζουν αυτούς τους κόμβους. Για να γίνει κάτι τέτοιο πρέπει το τρέχων TTL να έχει τέτοια τιμή ικανή για να προσεγγίσει τον κόμβο που υπέβαλε κατά το παρελθόν την ερώτηση.

Το Forward με το Random σχήμα αναζήτησης παρουσιάζουν σχεδόν αντίστοιχη συμπεριφορά. Παρατηρούμε ότι όσο υποβάλλονται παρόμοιες ερωτήσεις στο δίκτυο το recall τους αυξάνει όχι όμως για λόγους καθοδήγησης των ερωτημάτων με πληροφορία αλλά λόγω του ότι οι προηγούμενοι κόμβοι ερώτησης έβρισκαν αποτελέσματα και τα αποθήκευαν μόνιμα στην LDC τους (στο τέλος της προσομοίωσης κάθε μια ερώτηση του Q.S θα απαντάται από 140 κόμβους, 40 αρχικά συν 100 κόμβοι που υπέβαλαν την ερώτηση).

Η τυχαία συμπεριφορά του Forward σχήματος οφείλεται στο γεγονός ότι κάθε κόμβος εισάγει στην cache του την ίδια ερώτηση λιγότερο από 0.5 φορές. Αυτό μπορεί να ερμηνευτεί ως εξής: Από τους 5 περιπάτους κάθε αναζήτησης ένας μόνο εντοπίζει κόμβο απάντησης κάτι που σημαίνει ότι το πολύ 5 κόμβοι ανά αναζήτηση βλέπουν το QueryHit μήνυμα συνεπώς εισάγουν και στις caches τους το ερώτημα. Το κάθε ένα ερώτημα επαναλαμβάνεται 100 φορές άρα 500 κόμβοι στο σύνολο θα το εισάγουν στις caches τους. Συνεπώς το φαινόμενο να έχουμε προώθηση των

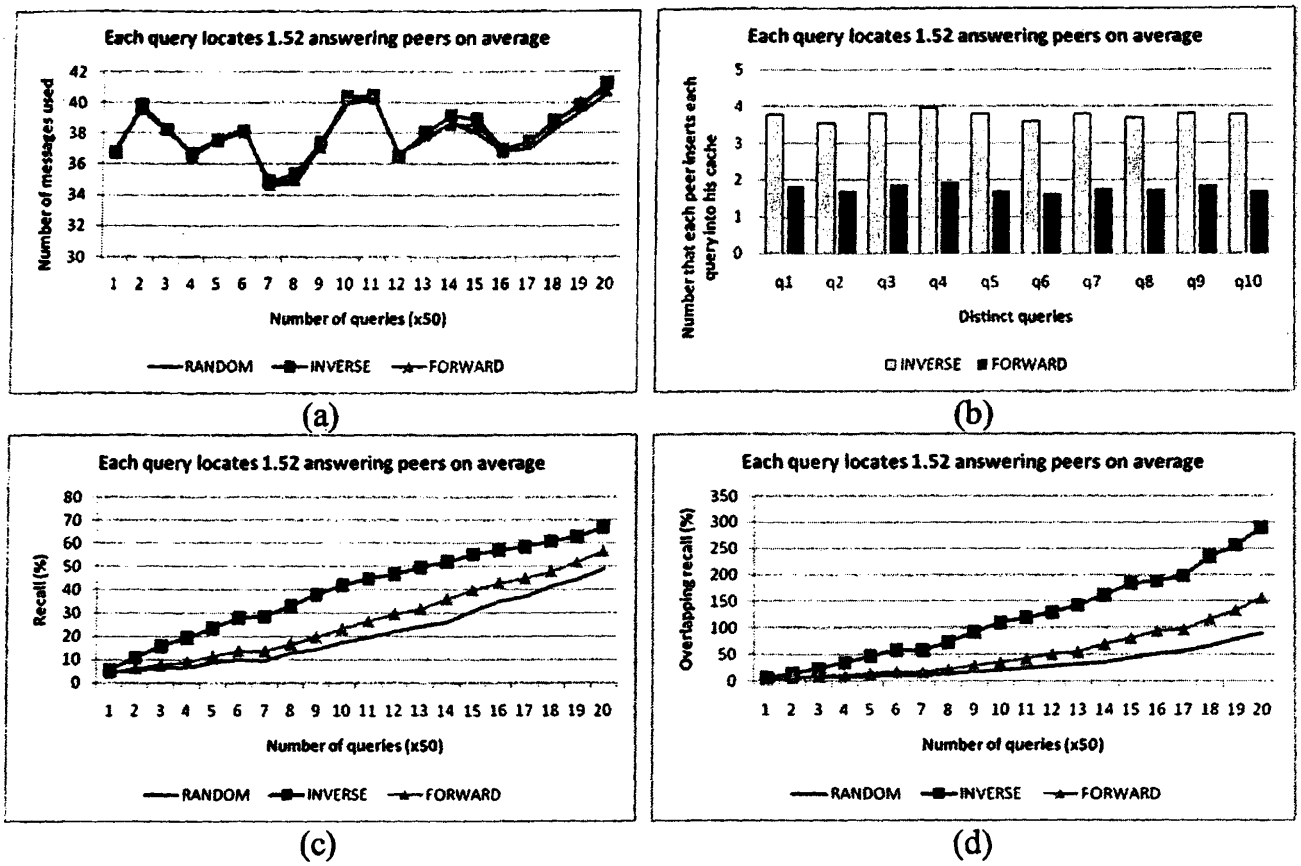
ερωτήσεων με πληροφορία παρουσιάζεται ελάχιστα τόσο ώστε μόλις μετά την 60η επανάληψη κάθε ξεχωριστού ερωτήματος να παρουσιάζεται μια ισχνή αύξηση στο recall του Forward σε σχέση με αυτή του Random. Η διαφορά τους προσεγγίζει στις τελευταίες επαναλήψεις το 3% υπέρ του Forward.

Το συμπέρασμα που προκύπτει από τις γραφικές του σχήματος 3.3 είναι ότι το Inverse σχήμα όταν καταχωρεί στις caches των κόμβων ωφέλιμη πληροφορία, δηλαδή ερωτήματα που κατά το παρελθόν ήταν επιτυχή, και οι caches γεμίζουν με μεγαλύτερο ρυθμό σε σχέση με το Forward είναι σε θέση να αξιοποιήσει αυτήν την πληροφορία-γνώση της cache και να σημειώσει μεγαλύτερο recall σε σχέση με το Forward σχήμα.

3.3.3. Αυξάνοντας το Πλήθος των Κόμβων Απάντησης ανά Ερώτημα στο Q.S

Στα προηγούμενα πειράματα η συμπεριφορά του Forward μηχανισμού ήταν παρόμοια με αυτή του Random. Ο λόγος που εμφάνιζε αυτήν την τυχαία συμπεριφορά οφειλόταν στο χαμηλό ρυθμό με τον οποίον οι κόμβοι του δικτύου εισήγαγαν στις caches τους τα ερωτήματα. Σε αυτό το πείραμα θέλουμε να εμφανίσουμε τον με πληροφορία χαρακτήρα του Forward μηχανισμού. Για να το πετύχουμε αυτό αυξάνουμε το πλήθος των κόμβων απάντησης που συναντά κατά μέσο όρο κάθε ερώτηση στο Q.S διπλασιάζοντας το μέγεθος της αναζήτησης. Με αυτόν τον τρόπο κάθε κόμβος θα εισάγει κάθε ερώτηση περισσότερες φορές συνεπώς η μάθηση του κάθε κόμβου θα μεγαλώνει και έτσι θα γίνονται 'σωστότερες' προωθήσεις των μηνυμάτων-ερωτημάτων. Οι παράμετροι του πειράματος δίνονται στην στήλη 3.3.3 του πίνακα 3.2.

Σύμφωνα με τη γραφική παράσταση του σχήματος 3.4 (a) κάθε ερώτηση εξετάζει 38 κόμβους κατά μέσο όρο συνεπώς το πλήθος των κόμβων απάντησης στην πρώτη υποβολή κάθε μιας από τις ξεχωριστές ερωτήσεις είναι κατά μέσο όρο 1.52 που όμως και εδώ στην συνέχεια μεγαλώνει και φτάνει στην 100η επανάληψη της κάθε μιας ξεχωριστής ερώτησης στους 5.282.



Σχήμα 3.4 Σε δίκτυο με 1000 κόμβους υποβάλλονται 10 ξεχωριστές ερωτήσεις που επαναλαμβάνονται 100 φορές η κάθε μια. κάθε μια ξεχωριστή ερώτηση την πρώτη φορά που υποβάλλεται βρίσκει κατά μέσο όρο 1.52 κόμβους απάντησης. Οι παράμετροι του πειράματος απεικονίζονται στη στήλη 3.3.3 του πίνακα 3.2

Όσον αφορά το Forward σχήμα αναζήτησης, στο τέλος της προσομοίωσης κάθε ένας κόμβος στο δίκτυο έβαλε στην cache του κάθε μια από τις ξεχωριστές ερωτήσεις 2 φορές. Συνεπώς όπως φαίνεται στην γραφική του σχήματος 3.4 (c) το recall του Forward μετά τις πρώτες 35 επαναλήψεις για κάθε μια από τις ξεχωριστές ερωτήσεις υπερτερεί σε σχέση με αυτό του Random σχήματος. Η αύξηση του recall που εμφανίζει το Random σχήμα δηλώνει το γεγονός ότι οι προηγούμενοι κόμβοι ερώτησης εντόπιζαν σχετικά έγγραφα και τα αποθήκευαν μόνιμα στην LDC τους. Άρα η διαφορά της τάξεως [5%-10%] που παρουσιάζει το Forward έναντι του Random μηχανισμού μαρτυρά το ποσοστό αύξησης που καρπώνεται ο Forward μηχανισμός λόγω της πληροφορίας που χρησιμοποιεί για την δρομολόγηση των ερωτημάτων. Ποιο έντονα ο με πληροφορία χαρακτήρας του Forward σχήματος εμφανίζεται στο overlapping recall που η διαφορά τους είναι στο διάστημα [30%-70%].

Όσον αφορά το Inverse σχήμα αναζήτησης, παρατηρούμε πως συνεχίζει να υπερτερεί έναντι των άλλων δυο σχημάτων ως προς το recall, συνεχίζοντας με αυτόν τον τρόπο να εκμεταλλεύεται με σωστό τρόπο την πληροφορία της cache που είναι και σε αυτήν την περίπτωση μεγαλύτερη σε σχέση με αυτή του Forward σχήματος. Μάλιστα σε αυτό το πείραμα η διαφορά του Inverse σε σχέση με το Forward και Random εμφανίζεται από τις 10 πρώτες επαναλήψεις κάθε μιας από τις ξεχωριστές ερωτήσεις.

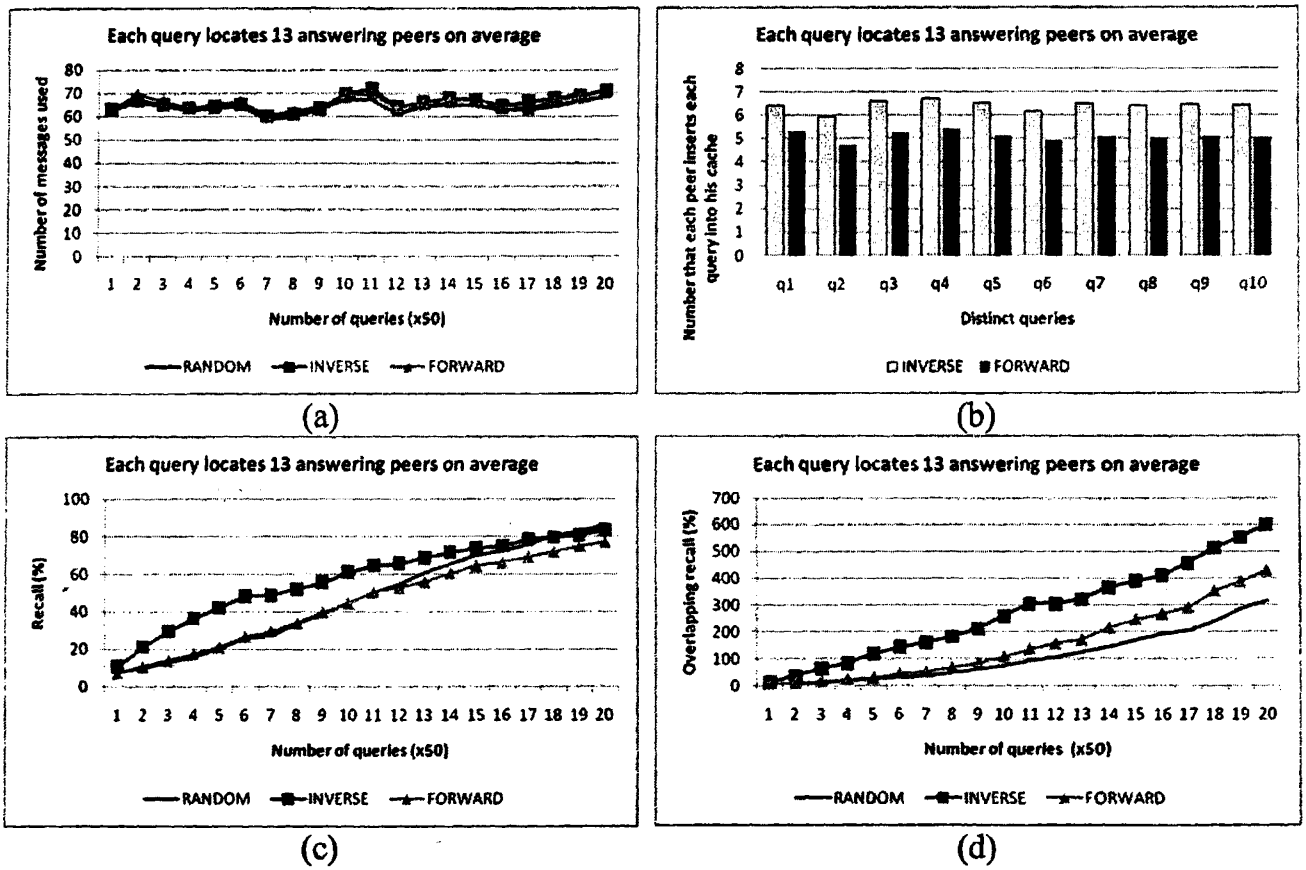
Επιπλέον, ο ρυθμός αύξησης του recall για το Inverse μηχανισμό ελαττώνεται μετά τις 500 ερωτήσεις. Αυτό το γεγονός επιβεβαιώνει την ανάλυση που κάναμε στην ενότητα 2.3 μιας και τώρα οι τελευταίες αναζητήσεις καταλήγουν για κάθε έναν από τους 5 περιπάτους να οδηγούνται σε περιοχές που προηγουμένως είχαν εξερευνηθεί. Συνεπώς από την $50^{η}$ επανάληψη κάθε μιας ξεχωριστής ερώτησης και μετά, το πλήθος των σχετικών εγγράφων που είναι διαφορετικό από αυτά που συγκέντρωσαν οι προηγούμενοι κόμβοι ερώτησης μειώνεται. Προφανώς κάτι τέτοιο δεν εμφανίζεται στο overlapping recall που η αύξηση του συνεχίζεται σχεδόν γραμμικά.

Συνεπώς αυξάνοντας το πλήθος των κόμβων απάντησης ο μεν Forward μηχανισμός γεμίζοντας πιο εύκολα τις caches είναι σε θέση να ξεχωρίσει από το Random σχήμα επιδεικνύοντας τον με πληροφορία χαρακτήρα του. Από την άλλη το Inverse σχήμα υπερτερεί έναντι των ανταγωνιστών του μιας και συνεχίζει να εκμεταλλεύεται σωστά τον μεγαλύτερο ρυθμό εισαγωγών ερωτημάτων στις caches των κόμβων του δικτύου.

3.3.4. Οι Caches 'Γεμίζουν' με τον ίδιο Ρυθμό

Μέχρι στιγμής οι caches στο Inverse σχήμα αναζήτησης γέμιζαν με μεγαλύτερο ρυθμό σε σχέση με το Forward σχήμα κάτι που όπως είδαμε οδηγούσε το πρώτο να αποδίδει πολύ καλυτέρα ως προς το δεύτερο όσον αφορά το recall. Σε αυτό το πείραμα θα αλλάξουμε το περιβάλλον και θα 'επιβάλουμε' οι caches να γεμίζουν σχεδόν το ίδιο και στα 2 σχήματα αναζήτησης με πληροφορία. Με δεδομένο ότι κάθε κόμβος θα λαμβάνει την ίδια ερώτηση ίσο πλήθος φορών στόχος είναι να παρατηρήσουμε πιο από τα σχήματα παρουσιάζει πιο γρήγορη συμπεριφορά εκμάθησης. Οι παράμετροι του πειράματος δίνονται στην στήλη 3.3.4 του πίνακα 3.2.

Για να έχουμε τον ίδιο ρυθμό εισαγωγών ερωτημάτων στις caches των κόμβων και στα δύο σχήματα αναζήτησης με πληροφορία, υποβάλουμε στο δίκτυο ένα νέο Q.S πάλι με 10 ξεχωριστές ερωτήσεις που υποβάλλονται από 100 φορές η κάθε μια στο δίκτυο, άλλα εδώ θα έχουμε Q.S 20%. Δηλαδή κάθε μια ερώτηση στο Q.S μπορεί να απαντηθεί από 200 κόμβους του δικτύου. Επιπλέον, θα θέσουμε το k=5 και το TTL=20.



Σχήμα 3.5 Σε δίκτυο με 1000 κόμβους υποβάλλονται 10 ξεχωριστές ερωτήσεις που επαναλαμβάνονται 100 φορές η κάθε μια. Κάθε μια ξεχωριστή ερώτηση την πρώτη φορά που υποβάλλεται βρίσκει κατά μέσο όρο 13 κόμβους απάντησης. Οι παράμετροι του πειράματος απεικονίζονται στη στήλη 3.3.4 του πίνακα 3.2

Κάθε ερώτηση εξετάζει συνολικά κατά μέσο όρο 65 κόμβους συνεπώς κάθε μια από τις 10 ξεχωριστές ερωτήσεις την πρώτη φορά που υποβάλλονται στο δίκτυο εντοπίζουν 13 κόμβους απάντησης. Σε αυτό το περιβάλλον και οι δυο αλγόριθμοι αναζήτησης με πληροφορία εισάγουν στις caches τους το ίδιο σχεδόν πλήθος ερωτήσεων (σχήμα 3.5 (b)).

Παρατηρούμε πως το Inverse σχήμα αναζήτησης εμφανίζει καλύτερο recall και overlapping recall σε σχέση με το Forward ακόμα και στην περίπτωση που οι caches γεμίζουν με τον ίδιο ρυθμό και στα δύο σχήματα. Το γεγονός αυτό μπορεί να ερμηνευτεί ως εξής: Θεωρούμε έστω ότι τόσο στο Inverse όσο και στο Forward σχήμα μια ερώτηση που υποβάλλεται στο δίκτυο συναντά έναν ενδιάμεσο κόμβο που φέρει μέσα στην cache του μια παρόμοια ερώτηση. Στο μεν Inverse σχήμα εάν το τρέχων ερώτημα έχει τέτοια τιμή στο TTL ικανή για να φτάσει τον προηγούμενο κόμβο ερώτησης είναι αρκετό για να αυξήσει το recall του νέου ερωτήματος. Στο δε Forward σχήμα όμως, ένας ενδιάμεσος κόμβος θα πρέπει να λάβει περισσότερες φορές την ίδια (παρόμοια) ερώτηση για να εντοπίσει ποιος από τους γείτονες του αποτελεί την πραγματικά καλύτερη διαδρομή για να αυξήσει το recall του νέου ερωτήματος.

Επιπλέον, η αργή μάθηση που παρουσιάζει το Forward σχήμα οφείλεται και στο ότι το περιβάλλον είναι δυναμικό λόγω της μετακίνησης περιεχομένου ανάμεσα στους κόμβους του δικτύου. Έστω για παράδειγμα ότι έχουμε έναν κόμβο p ο οποίος έχει 3 γείτονες και μια δεδομένη χρονική στιγμή ο p έχει εντοπίσει ότι κάθε ένα από τα υπογραφήματα των γειτόνων του έχουν 10 σχετικά έγγραφα ως προς μια συγκεκριμένη ερώτηση q. Αν μια μελλοντική χρονική στιγμή υπάρξει κόμβος που υποβάλλει την q και κατοικεί σε κάποιο από αυτά τα υπογραφήματα ο p δεν θα είναι σε θέση να επιλέξει τον καταλληλότερο γείτονα για να προωθήσει την q μιας και η κατανομή των εγγράφων έχει αλλάξει (αν φυσικά ο κόμβος που ρώτησε κατά το παρελθόν βρήκε αποτελέσματα).

Συνεπώς το συμπέρασμα που προκύπτει από το πείραμα 3.3.4 είναι ότι και στην περίπτωση που οι caches γεμίζουν με τον ίδιο ρυθμό το Inverse σχήμα εμφανίζει μεγαλύτερο recall σε σχέση με το Forward.

3.4. Σύγκριση των Σχημάτων με Βασικό Μηχανισμό Αναζήτησης k-walkers σε Δυναμικό Περιβάλλον

Στα πειράματα που εκτελέσαμε στην προηγούμενη ενότητα θεωρούσαμε ότι το περιβάλλον ήταν στατικό, δηλαδή οι κόμβοι στο δίκτυο παρέμεναν σταθεροί σε όλη τη διάρκεια της προσομοίωσης. Σε αυτή την ενότητα θα παρατηρήσουμε την συμπεριφορά των τεχνικών μας σε ένα δυναμικό περιβάλλον στο οποίο κόμβοι εισέρχονται και εξέρχονται προς και από το δίκτυο κατά βούληση. Ένα περιβάλλον το οποίο προσεγγίζει πιο ρεαλιστικά την πραγματική φύση των P2P δικτύων που χαρακτηρίζονται από αυτήν την δυναμικότητα-κινητικότητα των κόμβων. Στόχος μας σε αυτό το πείραμα είναι να εντοπίσουμε κάτω από διαφορετικούς ρυθμούς κινητικότητας των κόμβων εάν οι αλγόριθμοι αναζήτησης με πληροφορία είναι σε θέση να συνεχίζουν να εκτελούν σωστές προωθήσεις των ερωτημάτων και να μην εμφανίζουν τυχαία συμπεριφορά.

Αρχικά, θεωρούμε το ίδιο δίκτυο που χρησιμοποιήσαμε στα παραπάνω πειράματα, ήτοι ένα τυχαίο γράφημα με 1000 κόμβους και μέσο πλήθος γειτόνων ανά κόμβο ίσο με 5. Για την προσομοίωση των αφίξεων/αποχωρήσεων θεωρούμε ότι κάθε κόμβος ανά κάποιο συγκεκριμένο χρονικό διάστημα (το θέσαμε ίσο με 400 χρονικές στιγμές προσομοίωσης) 'ρίχνει' ένα ζάρι και αποφασίζει ανεξάρτητα για το αν πρέπει να βρίσκεται εντός ή εκτός δικτύου. Δοκιμάσαμε τρία διαφορετικά drop rate και συγκεκριμένα για 10%,20% και 30%. Αυτό σημαίνει πως στην περίπτωση π.χ του 30% κάθε κόμβος ανά 400 χρονικές στιγμές έχει πιθανότητα 30% να μεταβληθεί η κατάσταση του. Επιπλέον, να υπενθυμίσουμε όπως είχαμε τονίσει στην ενότητα 3.2 πως οι ερωτήσεις που υποβάλλονται στο δίκτυο απέχουν χρονική απόσταση μεταξύ τους 200 χρονικές στιγμές. Αυτό σημαίνει πως ανά δυο οι ερωτήσεις υποβάλλονται σε διαφορετικό δίκτυο.

Αν ένας κόμβος πρέπει να εισέλθει στο δίκτυο τότε επιστρέφει πάλι στην ίδια θέση που βρισκόταν και πριν την αποχώρηση του. Όπου ίδια θέση εννοούμε ότι θα συνδέεται με τους ίδιους γείτονες που είχε και πριν. Στην ουσία αναφερόμαστε σε on και off λειτουργία του κάθε κόμβου. Επιπλέον, μόλις ένας κόμβος εξέρχεται του δικτύου δεν διαγράφει το περιεχόμενο της cache του κάτι που σημαίνει ότι σε μια

μελλοντική άφιξη του στο δίκτυο θα έχει το περιεχόμενο της cache που είχε και πριν αποχωρίσει. Αν και από τη μια πλευρά, αυτό το στοιχείο από μόνο του ισχυροποιεί τις δυο μεθόδους αναζήτησης με πληροφορία σε σύγκριση με το Random σχήμα, από την άλλη πλευρά θα λέγαμε ότι είναι μια λογική υπόθεση αν αυτές οι αποχωρίσεις/αφίξεις γίνονται μέσα σε ένα κοντινό μεταξύ τους χρονικό διάστημα.

Οι παράμετροι του πειράματος που χρησιμοποιήθηκαν είναι ίδιοι με αυτούς του πειράματος 3.3.3 και απεικονίζονται στην στήλη 3.3.3 του πίνακα 3.2. Στο σχήμα 3.7 απεικονίζονται οι γραφικές παραστάσεις για μέγεθος αναζήτησης και recall για κάθε ένα από τα τρία σχήματα αναζήτησης για drop rate ίσο με 0%,10%,20% και 30%. Στην περίπτωση που δεν παρατηρείται κινητικότητα των κόμβων του δικτύου τα αποτελέσματα είναι ίδια με αυτά που δώσαμε στο πείραμα 3.3.3.

Αρχικά παρατηρούμε πως το πλήθος των εξεταζόμενων κόμβων όσο το drop rate μεγαλώνει ελαττώνεται σταδιακά μιας και στο δίκτυο υπάρχουν λιγότεροι κόμβοι. Με την σειρά του αυτό επηρεάζει και το recall που σε όλα τα σχήματα αναζήτησης ελαττώνεται με την αύξηση του drop rate.

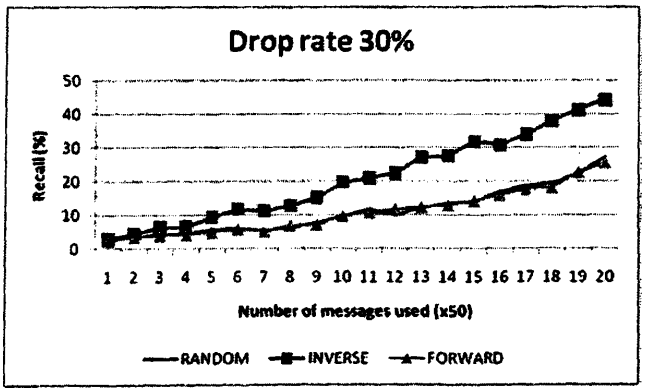
Όσον αφορά τα δυο σχήματα αναζήτησης με πληροφορία δεν κάνουν κάτι στον μηχανισμό τους για να μπορέσουν να αντιμετωπίσουν την δυναμικότητα που παρουσιάζεται. Οι caches πλέον περιέχουν πληροφορία η οποία δεν είναι σίγουρο ότι θα οδηγήσει σε κάποιον απαντών κόμβο διότι δεν υπάρχει κάποια πληροφορία για το εάν αυτός είναι μέσα στο δίκτυο μια δεδομένη χρονική στιγμή.

Ένα πλεονέκτημα του Inverse σχήματος έναντι του Forward οφείλεται στην χρήση του τελευταίου των QueryHits μηνυμάτων. Κόμβοι που είναι ενδιάμεσοι για μια ερώτηση ενδεχομένως να έχουν αποχωρίσει από το δίκτυο όταν ένα QueryHit μήνυμα ακολουθεί το αντίστροφο μονοπάτι και περνά πάλι από αυτούς. Αυτό πιθανόν να εμποδίζει το Forward μηχανισμό να γεμίζουν οι caches των κόμβων και βέβαια να χάνονται έγγραφα.

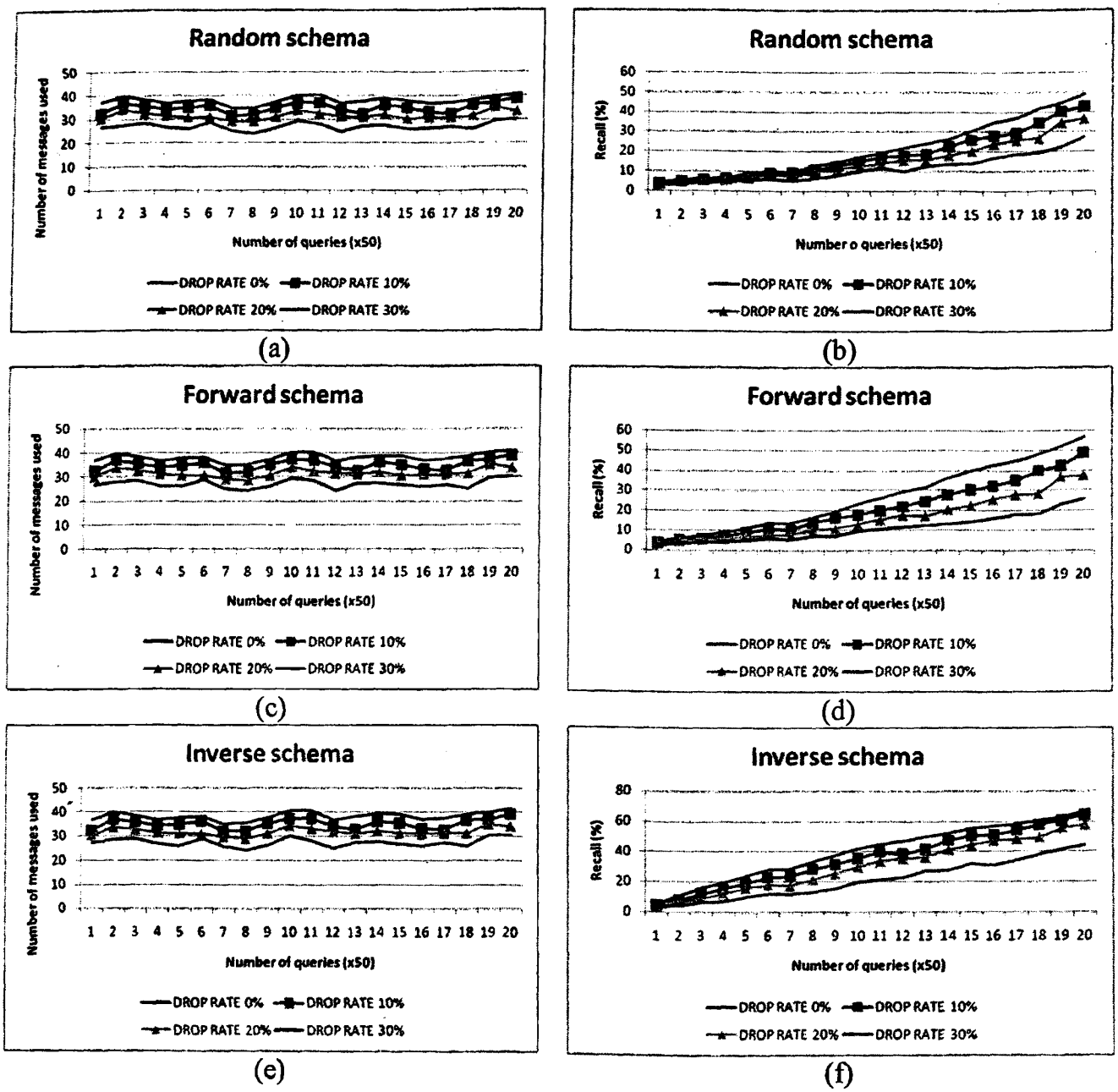
Ένα μειονέκτημα του Inverse σχήματος είναι ότι προωθεί τα ερωτήματα προς προηγούμενος κόμβους ερώτησης που αυτοί ως προς το πλήθος είναι λιγότεροι σε

σχέση με τους κόμβους απάντησης, συνεπώς με μεγαλύτερη πιθανότητα οι caches του Forward σχήματος ενδεχομένως να εντοπίζουν κάποιους από τους κόμβους που φέρουν σχετικά έγγραφα.

Τα παραπάνω όμως δεν μπορούν να φανούν τόσο έντονα μέσα από την εκτέλεση του πειράματος, είναι φανερό πως η διαφορά του Inverse σχήματος σε σχέση με το Random ήταν τόσο μεγάλη, όπως φαίνεται στο σχήμα 3.3.3, που ακόμα και σε drop rate 30% διατηρεί τον με πληροφορία χαρακτήρα του και προηγείται ως προς το recall έναντι των άλλων δυο σχημάτων. Αντίθετα, το Forward σχήμα στο ίδιο drop rate η συμπεριφορά του γίνεται πανομοιότυπη με αυτή του Random σχήματος. Το σχήμα 3.6 παρουσιάζει αυτό το γεγονός.



Σχήμα 3.6 Το recall για τα τρία σχήματα αναζήτησης στη μέγιστη τιμή του drop rate ίση με 30%



Σχήμα 3.7 Για κάθε ένα από τα τρία σχήματα αναζήτησης στα δεξιά απεικονίζονται τα συνολικά μηνύματα που παρήγαγε κατά μέσο όρο κάθε ερώτημα ενώ στα δεξιά το recall. Κάθε καμπύλη σε κάθε γραφική αντιστοιχεί σε drop rate της τάξεως του 0%,10%,20% και 30%

3.5. Σύγκριση των Σχημάτων με Βασικό Μηχανισμό Αναζήτησης Constrained Flooding

Σε όλα τα πειράματα που περιγράψαμε μέχρις στιγμής χρησιμοποιούσαμε βασικό μηχανισμό αναζήτησης τους k-walkers. Αυτό από μόνο του όριζε ένα συγκεκριμένο τρόπο επικοινωνίας μεταξύ των κόμβων του δικτύου. Σε αυτήν την ενότητα θα αλλάξουμε αυτόν τον μηχανισμό και θα χρησιμοποιήσουμε στην θέση των k-walkers ένα constrained flooding σχήμα. Στόχος μας είναι να δημιουργήσουμε περιβάλλοντα παρόμοια με αυτά της ενότητας 3.3 έτσι ώστε να παρατηρήσουμε πως αυτή η αλλαγή μπορεί να επηρεάσει και με ποιόν τρόπο τα δυο σχήματα αναζήτησης με πληροφορία.

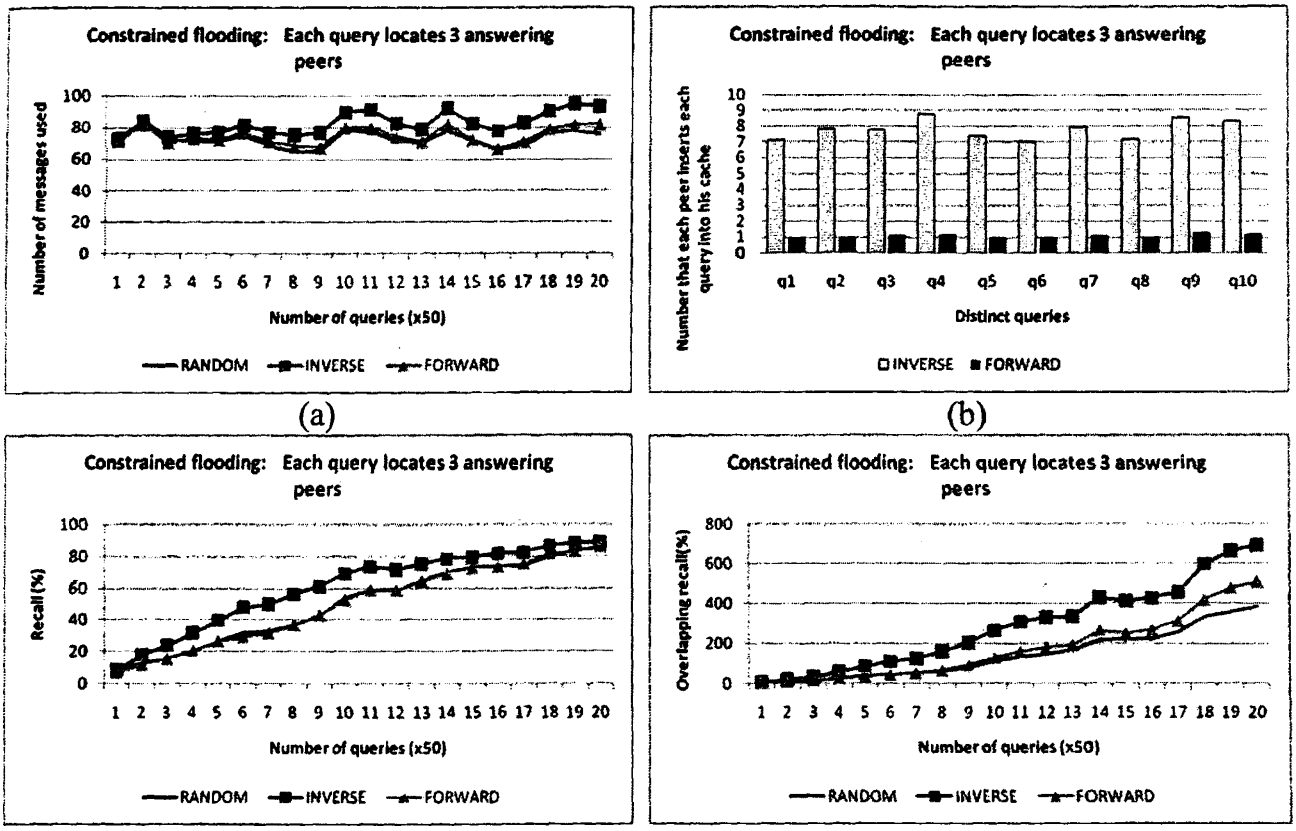
Χρησιμοποιούμε το ίδιο δίκτυο με αυτό του πειράματος 3.3.3, δηλαδή δίκτυο με 1000 κόμβους, μέσο πλήθος γειτόνων ανά κόμβο ίσο με 5 και Q.S 4% με 10 ξεχωριστές ερωτήσεις να επαναλαμβάνονται 100 φορές η κάθε μια. Στόχος μας είναι να δημιουργήσουμε ένα παρόμοιο μέγεθος αναζήτησης με αυτό του πειράματος 3.3.3, ήτοι κοντά στο 50. Για αυτό τον λόγο θέτουμε TTL ίσο με 4 και όπως έχουμε αναφέρει το %forward ίσο με 50%. Οι τιμές των παραμέτρων φαίνονται συγκεντρωτικά στον πίνακα 3.3.

Πίνακας 3.3 Παράμετροι για το πείραμα 3.5

%forward	50%
TTL	4
Q.S	4%
Num of queries	10x100

Όπως φαίνεται στο σχήμα 3.8 κατά μέσο όρο κάθε αναζήτηση παράγει 75 μηνύματα κάτι που σημαίνει ότι κάθε μια από τις ξεχωριστές ερωτήσεις όταν υποβάλλεται για πρώτη φορά στο δίκτυο συναντάει κατά μέσο όρο 3 κόμβους απάντησης. Προφανώς σε αυτήν την περίπτωση σε σχέση με το πείραμα 3.3.3 η αναζήτηση είναι μεγαλύτερη και το πλήθος των κόμβων απάντησης μεγαλύτερο και αυτό κατά ένα. Η παρατήρηση μας δεν εστιάζεται τόσο στο recall, αλλά στο πόσες φορές κάθε κόμβος εισάγει στην cache του κατά μέσο όρο κάθε μια από τις ξεχωριστές ερωτήσεις. Σε σχέση με το Forward μηχανισμό, οι κόμβοι στο constrained flooding εισάγουν στις caches τους

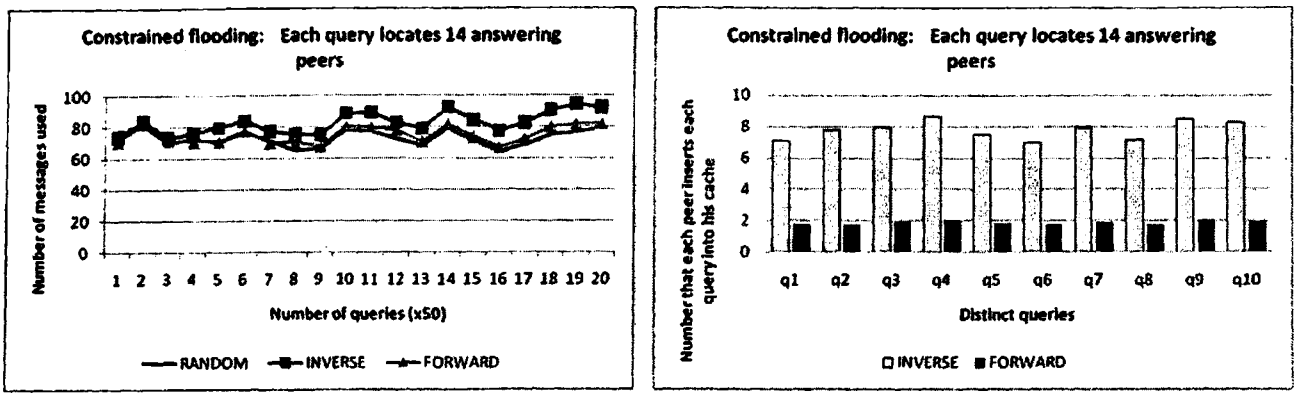
κάθε μια μοναδική ερώτηση κοντά στην 1 φορά, ενώ στο σχήμα με τους walkers για το ίδιο δίκτυο έφτανε κοντά στις 2 φορές. Αυτό μάλιστα στην περίπτωση κατά την οποία το μέγεθος της αναζήτησης (και συνεπώς και το πλήθος των κόμβων απάντησης) στους walkers ήταν μικρότερο από αυτό που συναντάμε στο constrained flooding σχήμα. Δηλαδή, θα περιμέναμε μια αύξηση στο πλήθος των εισαγόμενων ερωτήσεων.



Σχήμα 3.8 Σε δίκτυο με 1000 κόμβους υποβάλλονται 10 ξεχωριστές ερωτήσεις που επαναλαμβάνονται 100 φορές η κάθε μια. Κάθε μια ξεχωριστή ερώτηση την πρώτη φορά που υποβάλλεται βρίσκει κατά μέσο όρο 3 κόμβους απάντησης. Οι παράμετροι του πειράματος απεικονίζονται στον πίνακα 3.3

Το φαινόμενο παρατηρήθηκε ακόμα πιο έντονα όταν εκτελέσαμε με constrained flooding βασικό μηχανισμό αναζήτησης ένα πείραμα αντίστοιχο με αυτό της ενότητα 3.3.4. Δηλαδή με το ίδιο Q.S 20% και με TTL ίσο με 4, παίρνοντας ένα μέσο μέγεθος αναζήτησης ίσο με 70. Στο σχήμα 3.9 απεικονίζονται το μέσο πλήθος κόμβων που εξετάζει κάθε μια ερώτηση καθώς και το πως γέμισαν οι caches των κόμβων του δικτύου. Όπως παρατηρείται κάθε κόμβος κατά μέσο όρο είδε κάθε μια από τις μοναδικές ερωτήσεις κοντά στις 2 φορές την στιγμή που στο αντίστοιχο περιβάλλον

με τους k-walkers ως βασικό σχήμα αναζήτησης, το πλήθος των εισαγωγών έφτανε τις 5 φορές.



(a) (b)

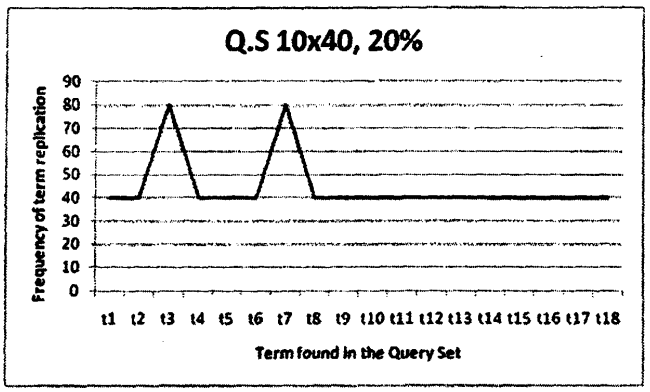
Σχήμα 3.9 Το Q.S είναι το ίδιο που χρησιμοποιήθηκε στο πείραμα 3.3.4. Το μέγεθος της αναζήτησης που πετυχαίνουν οι τεχνικές για κάθε μια ερώτηση είναι συγκρίσιμο με το πείραμα 3.3.4

Το ότι το Forward σχήμα γεμίζει τις caches πιο εύκολα όταν χρησιμοποιείται πάνω από τους k-walkers σε σχέση με το constrained flooding ερμηνεύεται από τον τρόπο με τον οποίο οι δυο μηχανισμοί προωθούν τα ερωτήματα. Στην πρώτη περίπτωση η αναζήτηση εξελίσσεται σαν k μεγάλες 'γραμμές' οι οποίες ξεκινούν από τον κόμβο ερώτησης και εξερευνούν το δίκτυο. Αυτό έχει σαν αποτέλεσμα πως ένας μόνο κόμβος απάντησης να εμφανιστεί σε κάθε ένα από τα k μονοπάτια είναι αρκετό για να γεμίσουν οι caches των κόμβων. Μάλιστα όσο μεγαλύτερο βάθος έχει στο μονοπάτι ο απαντών κόμβος τόσο το καλύτερο για το Forward σχήμα. Στην δεύτερη περίπτωση αντίθετα, η αναζήτηση είναι κάπως 'σφαιρική' με κέντρο τον κόμβο που υποβάλλει την ερώτηση συνεπώς εδώ το πλήθος των κόμβων απάντησης πρέπει να είναι μεγαλύτερο του k για να μπορέσουν οι caches να δουν το ίδιο πλήθος ερωτημάτων που εισήγαγαν στην περίπτωση των walkers.

Συνεπώς το Forward σχήμα στην περίπτωση που χρησιμοποιείται με βασικό μηχανισμό αναζήτησης το constrained flooding παρουσιάζει μικρότερο recall σε σχέση με αυτό που εμφανίζεται όταν χρησιμοποιείται σαν βασικό σχήμα αναζήτησης οι k-walkers.

3.6. Σύγκριση Ανάμεσα στο Forward και Random Σχήμα με Βασικό Μηχανισμό Αναζήτησης k-walkers

Σε αυτήν την ενότητα θα πραγματοποιήσουμε ένα πείραμα με διαφορετική λογική σε σχέση με τα προηγούμενα. Εδώ κατασκευάζουμε ένα Q.S το οποίο αποτελείται από 10 ξεχωριστές ερωτήσεις που υποβάλλονται από 40 φορές η κάθε μια από 10 peers. Δηλαδή, σε αυτήν την περίπτωση έχουμε 10 μόνο διαφορετικούς κόμβους ερώτησης που υποβάλλουν το ίδιο ερώτημα 40 φορές. Το term locality του Q.S που δημιουργήσαμε απεικονίζεται στο σχήμα 3.10.



Σχήμα 3.10 Term locality για το Q.S 10x40 με 20%, κάθε μια ερώτηση αποτελείται από 2 όρους

Σε ένα τέτοιο περιβάλλον το Inverse σχήμα αναζήτησης δεν βρίσκει εφαρμογή μιας και η λογική του στηρίζεται στο να οδηγούνται παρόμοιες ερωτήσεις προς προηγούμενους κόμβους ερώτησης, εδώ όμως οι κόμβοι ερώτησης είναι ίδιοι για κάθε μια από τις 10 ξεχωριστές ερωτήσεις. Συνεπώς στο τρέχων πείραμα θα μας απασχολήσουν μόνο το Random και το Forward σχήμα και συγκεκριμένα θέλουμε να δούμε την με πληροφορία φύση του Forward αλγορίθμου σε ένα περιβάλλον που δεν αυξάνονται οι κόμβοι απάντησης μιας και ένας συγκεκριμένος κόμβος επαναλαμβάνει ο ίδιος πολλαπλές φορές ένα ερώτημα. Οι τιμές των παραμέτρων που χρησιμοποιήθηκαν απεικονίζονται στο πίνακα 3.4.

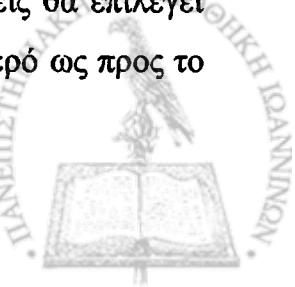
Πίνακας 3.4 Παράμετροι για το Πείραμα 3.6

k	4
TTL	10
Q.S	20%
Num of queries	10x40

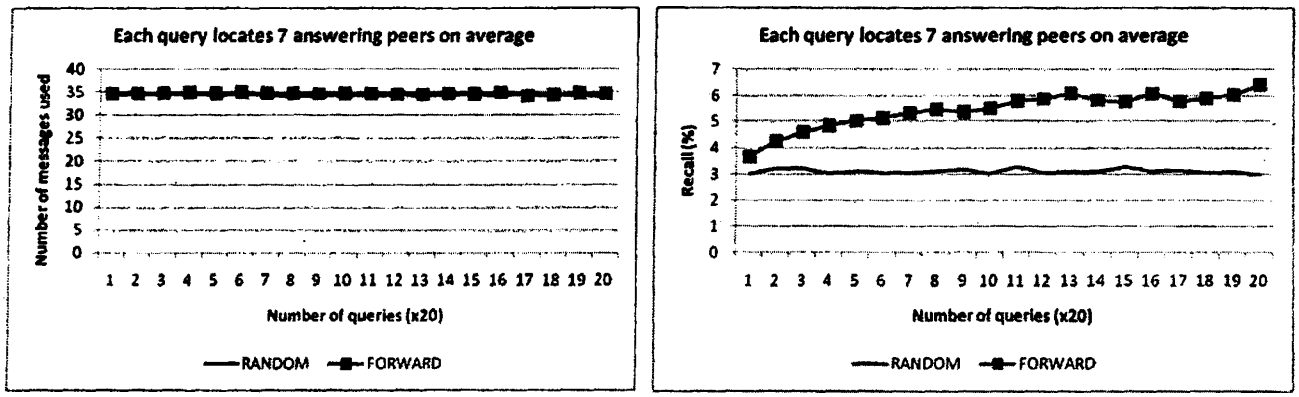
Το μέσο πλήθος κόμβων που κατά μέσο όρο εξετάζει κάθε ερώτηση εμφανίζεται στο σχήμα 3.11 (a) και είναι ίσο με 35 και για τα δύο σχήματα αναζήτησης, κάτι που σημαίνει πως κάθε ερώτηση του Q.S θα εντοπίζει σε κάθε αναζήτηση 7 κόμβους απάντησης. Σε αυτό το πείραμα κάθε σημείο του άξονα x αντιστοιχεί σε 20 ερωτήσεις που είναι 2 επαναλήψεις κάθε μιας από τις 10 ξεχωριστές ερωτήσεις. Συνεπώς το σημείο 1 αφορά τις δύο πρώτες επαναλήψεις για κάθε μια από τις ξεχωριστές ερωτήσεις, το σημείο 2 αντιστοιχεί στις επόμενες 2 κοκ. Στο σχήμα 3.11 (b) απεικονίζεται το recall το οποίο είναι ακριβώς το ίδιο και με το overlapping recall σε αυτήν την περίπτωση μιας και εξ αρχής τα έγγραφα που φέρει ο κάθε ένας κόμβος στην LDC του είναι μοναδικά (δεν υπάρχουν διπλότυπα).

Σε αυτό το πείραμα παρατηρούμε αρχικώς πως το recall στην Random αναζήτηση δεν αυξάνει μιας και εδώ επειδή κάθε κόμβος υποβάλλει ο ίδιος επαναληπτικά την ίδια ερώτηση, η αναζήτηση και στις 40 επαναλήψεις θα συναντά το ίδιο πλήθος κόμβων απάντησης. Αντίθετα, το recall στην περίπτωση του Forward σχήματος μεγαλώνει και στις επαναλήψεις 39 και 40 για τις 10 ξεχωριστές ερωτήσεις γίνεται διπλάσιο σε σχέση με αυτό του Random σχήματος.

Αυτό το περιβάλλον κρίνεται ελκυστικότατο για να αποδώσει καλύτερα το Forward σχήμα σε σχέση με το Random για δυο λόγους. (1) Κάθε ενδιάμεσος κόμβος λαμβάνει την ίδια ερώτηση πολλαπλές φορές μιας και ο ερωτών κόμβος είναι ίδιος και για τις 40 επαναλήψεις της ερώτησης. Φανταστείτε ότι οι γείτονες ενός κόμβου ερώτησης θα λάβουν σχεδόν 40 φορές την ίδια ερώτηση συνεπώς θα κάνουν πολλαπλάσια βήματα εκμάθησης σε σύγκριση με αυτά που γινόντουσαν στα προηγούμενα πειράματα άρα ο κάθε γείτονας του κόμβου ερώτησης θα εξετάσει όλους σχεδόν τους γείτονες του συνεπώς μετά από κάποιες επαναλήψεις θα επιλέγει τον πραγματικά καταλληλότερο γείτονα. (2) Το περιβάλλον είναι σταθερό ως προς το



περιεχόμενο κάτι που βοηθάει στην εκμάθηση κάθε κόμβου για το περιεχόμενο των υπογραφημάτων που ξεκινούν από τους γείτονες τους.



(a) (b)

Σχήμα 3.11 Σε δίκτυο με 1000 κόμβους υποβάλλονται 10 ξεχωριστές ερωτήσεις από 10 διαφορετικούς κόμβους που ο κάθε ένας υποβάλλει κάθε ένα από τα ερωτήματα 40 Φορές. Οι παράμετροι του πειράματος απεικονίζονται στο πίνακα 3.4

Το συμπέρασμα που προκύπτει από το συγκεκριμένο πείραμα είναι πως το Forward σχήμα αναζήτησης μπορεί να είναι αποδοτικό ακόμα και στην περίπτωση όπου οι κόμβοι του δικτύου ρωτάνε διαφορετικά πράγματα μεταξύ τους, ή αλλιώς δεν μοιράζονται κοινά ενδιαφέροντα. Στο πείραμα αυτό, το μεγάλο term locality το δημιούργησε ο κάθε ένας από τους κόμβους ερώτησης για την δικιά του ερώτηση που υπέβαλλε, αδιαφορώντας για το τι ρώτησαν οι υπόλοιποι κόμβοι της κοινότητας. Να σημειώσουμε πως κάτι τέτοιο είναι αδύνατο να συμβεί στο Inverse σχήμα που στην ουσία βασίζεται στο γεγονός ότι υπάρχουν πολλοί διαφορετικοί κόμβοι που υποβάλλουν ίδια (παρόμοια) ερωτήματα και που ο επόμενος ερωτών κόμβος επωφελείται από τον προηγούμενο.

3.7. Σύνοψη Πειραμάτων

Σε όλα τα πειράματα που εκτελέστηκαν η συμπεριφορά ως προς το recall του Inverse σχήματος ήταν καλύτερη σε σχέση με αυτή του Forward μηχανισμού. Με δεδομένο ότι οι προηγούμενοι κόμβοι ερώτησης βρίσκουν απαντήσεις, το Inverse σχήμα εκμεταλλεύεται το γεγονός ότι οι caches των κόμβων γεμίζουν πιο εύκολα σε σχέση με το Forward σχήμα, εκτελώντας με αυτόν τον τρόπο περισσότερες σε πλήθος με πληροφορία προωθήσεις των μηνυμάτων-ερωτημάτων. Αυτό παρατηρήθηκε ακόμα πιο έντονα όταν αλλάξαμε τον βασικό μηχανισμό αναζήτησης από k-walkers σε constrained flooding.

Για τον ίδιο λόγο το Inverse ακόμα και σε ένα δυναμικό περιβάλλον που κόμβοι εισέρχονται και αποχωρούν προς και από το δίκτυο είναι σε θέση να κρατήσει τον με πληροφορία χαρακτήρα του σε αντίθεση με το Forward που η συμπεριφορά του γίνεται τυχαία.

Όμως και στην περίπτωση κατά την οποία οι προηγούμενοι κόμβοι ερώτησης δεν βρίσκουν απαντήσεις, δεν λειτουργεί αρνητικά για το Inverse μιας και επιτελεί τυχαίες προωθήσεις όπως ακριβώς κάνουν και τα άλλα δυο σχήματα αναζήτησης.

Η ίδια συμπεριφορά παρατηρήθηκε και κατά την περίπτωση εκείνη όπου οι caches γέμιζαν με τον ίδιο ρυθμό και για τα 2 σχήμα αναζήτησης με πληροφορία. Ο Forward μηχανισμός με τον τρόπο που λειτουργεί, πρέπει να πάρει το ίδιο (παρόμοιο) ερώτημα περισσότερες φορές σε πλήθος σε σχέση με το Inverse ώστε να μάθει για μια δοθείσα ερώτηση πια είναι όντος η καλύτερη διαδρομή για να την ικανοποιήσει.

Τέλος, στο τελευταίο πείραμα δείξαμε ότι το Forward σχήμα μπορεί να εφαρμοστεί αποδοτικά σε ένα περιβάλλον στο οποίο οι κόμβοι δεν μοιράζονται κοινά ενδιαφέροντα και ένας κόμβος ρωτά επαναληπτικά ο ίδιος μια ερώτηση. Λόγω του τρόπου λειτουργίας του αλγορίθμου μπορεί ο κόμβος να παρατηρήσει αύξηση στο recall. Σε ένα τέτοιο περιβάλλον το Inverse σχήμα είναι αδύνατον να εφαρμοστεί.

ΚΕΦΑΛΑΙΟ 4. ΣΧΕΤΙΚΗ ΔΟΥΛΕΙΑ

Στην εργασία [8] προτείνεται ένας μηχανισμός αναζήτησης με πληροφορία για αδόμητα P2P δίκτυα. Θεωρεί πως κάθε κόμβος στο δίκτυο φέρει μια τοπική συλλογή από έγγραφα στα οποία ορίζεται μια γενική (global) επίπεδη κατηγοριοποίηση, π.χ έγγραφα ανήκουν σε κατηγορίες όπως Θεωρία, Δίκτυα, Μαθηματικά κτλ. Συνεπώς μέσω αυτής της τεχνικής απαντώνται ερωτήματα περιεχομένου που διατυπώνονται ως εξής: Εντόπισε 10 έγγραφα που έχουν να κάνουν με Θεωρία & Δίκτυα. Όπου 10 εκφράζει το θεμιτό από το χρήστη πλήθος σχετικών εγγράφων που επιθυμεί να εντοπίσει η αναζήτηση του.

Κάθε κόμβος στο δίκτυο συντηρεί ένα Routing Index (Ευρετήριο Διαδρομών) στο οποίο αποθηκεύει για κάθε έναν από τους γείτονες του πόσα έγγραφα συνολικά, αλλά και ανά κατηγορία μπορεί να βρει ένα ερώτημα αν ακολουθήσει αυτή την διαδρομή που ορίζει ο γείτονας του. Λόγω του ότι αποθηκεύονται ο αριθμός εγγράφων ανά διαδρομή με συμπαγή τρόπο η δομή καλείται Compound Index (CI). Συνεπώς όταν ένας κόμβος λάβει ένα ερώτημα ταξινομεί τους γείτονες του με βάση το περιεχόμενο του CI και αυτός που λαμβάνει το μεγαλύτερο σκορ γίνεται αυτόματα ο πιο υποσχόμενος κόμβος για την δοθείσα ερώτηση, ή διαφορετικά το υπογράφημα που ξεκινάει από αυτόν τον γείτονα έχει και την μεγαλύτερη σε ποσότητα πληροφορία για να καλύψει την δοθείσα ερώτηση.

Επιπλέον, οι συγγραφείς του [8] προχωράνε ένα βήμα παραπάνω και εισάγουν και 2 πιο ενισχυμένα είδη ευρετηρίων, τα hop-count RIs και τα exponentially RIs. Τα πρώτα σε αντίθεση με τα CI αναφέρονται σε πλήθος εγγράφων ανά βήμα (hop) για κάποιο συγκεκριμένο βάθος-ορίζοντα. Με αυτόν τον τρόπο όταν γίνεται η ταξινόμηση των γειτόνων ενός κόμβου ως προς μια ερώτηση λαμβάνεται υπόψη το

πλήθος των σχετικών εγγράφων ανά βήμα που θα έχει μια συγκεκριμένη διαδρομή. Τα exponentially RIs αποτελούν ένα συνδυασμό των δυο πρώτων RIs εμπεριέχοντας σαν πληροφορία το πλήθος των εγγράφων ανά βήμα αλλά ταυτόχρονα χωρίς να φράζεται από κάποιον συγκεκριμένο ορίζοντα. Το τελευταίο κατορθώνεται με απώλεια στην ακρίβεια της πληροφορίας.

Τα RIs παρουσιάζουν ομοιότητα με τον Forward μηχανισμό αναζήτησης όσον αφορά την έννοια της δρομολόγησης (routing), δηλαδή τα ερωτήματα προωθούνται στις πιο υποσχόμενες διαδρομές. Η διαφορά τους έγκειται στο πως γεμίζουν οι caches, στην μεν περίπτωση του Forward αυτή η διαδικασία γίνεται μέσω των QueryHits μηνυμάτων που παράγουν οι κόμβοι που απαντούν και στην ουσία μαρτυρούν το περιεχόμενο τους, αντίθετα στα RIs αυτό γίνεται ρητά μέσω ενός μηχανισμού που τρέχει στο παρασκήνιο και οποίος αναλαμβάνει να γίνεται η ενημέρωση των RIs όταν κόμβοι αποχωρούν/εισέρχονται από και προς το δίκτυο καθώς και όταν έχουμε αλλαγές περιεχομένου σε έναν κόμβο (προστίθενται/αφαιρούνται έγγραφα). Το τελευταίο είναι και το μεγαλύτερο μειονέκτημα των RIs, διότι απαιτούνται O(n) μηνύματα για κάθε μια ενημέρωση.

Στο [4] παρουσιάζεται για πρώτη φορά η αναζήτηση των k random walkers, την οποία χρησιμοποιούμε και εμείς σαν βασικό τρόπο επικοινωνίας μεταξύ των κόμβων του δικτύου. Σε αυτήν την αναζήτηση κάθε ερωτών κόμβος προωθεί την ερώτηση σε k το πολύ από τους γείτονες του. Στη συνέχεια κάθε ένας από τους ενδιάμεσους κόμβους επιλέγει στην τύχη έναν μόνο από τους γείτονες του για να προωθήσει το ερώτημα. Η αναζήτηση συνεχίζεται έως ένα βάθος TTL το οποίο ορίζει εξ' αρχής ο κόμβος ερώτησης. Συνεπώς μιλάμε στην ουσία για το πολύ k περιπάτους που εξερευνούν το δίκτυο.

Στο [9] οι συγγραφείς προτείνουν τον APS που αποτελεί επέκταση της ιδέα των k walkers προσδίδοντας σε αυτόν έναν με πληροφορία χαρακτήρα έτσι ώστε οι ενδιάμεσοι κόμβοι μιας ερώτησης να επιλέγουν κάθε χρονική στιγμή τον πιο υποσχόμενο γείτονα για να προωθούν το ερώτημα. Το APS εκτελεί αναζήτηση βάση αναγνωριστικού (id-based) π.χ με βάση το όνομα ενός αρχείου, σε αδόμητα P2P δίκτυα έχοντας τον κάθε κόμβο να διατηρεί μια cache στην οποία αποθηκεύονται

ερωτήματα που κατά το παρελθόν υποβλήθηκαν στο δίκτυο. Πιο συγκεκριμένα για κάθε ερώτημα της cache αντιστοιχίζεται σε κάθε έναν γείτονα ένα σκορ. Προφανώς αν η ερώτηση βρήκε απάντηση από την διαδρομή που ορίζει ένας γείτονας τότε το σκορ του είναι μεγαλύτερο σε σχέση με τους υπόλοιπους από τους οποίους είτε το ερώτημα προωθήθηκε αλλά δεν προέκυψε απάντηση από το υπογράφημα τους είτε το ερώτημα δεν προωθήθηκε καθόλου. Στην συνέχεια αν ένας κόμβος λάβει ακριβώς την ίδια ερώτηση με κάποια που είχε υποβληθεί κατά το παρελθόν και είναι ήδη στην cache του είναι σε θέση να προωθήσει την ερώτηση στην πιο 'κατάλληλη' διαδρομή.

Η λογική του APS είναι παρόμοια με εκείνη του Forward σχήματος και τα δύο στην ουσία είναι lazy-based τεχνικές για αναζήτηση πληροφορίας σε αδόμητα P2P. Επιπλέον και τα δύο σχήματα στηρίζονται στην έννοια του QueryHit, οι κόμβοι που απαντάνε οφείλουν να στείλουν προς το αντίστροφο μονοπάτι ένα μήνυμα μέσω του οποίου γνωστοποιούν στους προηγούμενους κόμβους του μονοπατιού κατά πόσο η αρχική τυχαία επιλογή τους ήταν ή όχι σωστή. Επίσης τόσο το Forward όσο και το Inverse θα μπορούσαμε να τα δούμε σαν δύο σχήματα αναζήτησης που επεκτείνουν την βασική ιδέα του APS από τον κόσμο της αναζήτησης βάση κάποιου αναγνωριστικού στον κόσμο της αναζήτησης περιεχομένου.

Στο [10] προτείνεται ο Intelligent Search Mechanism (ISM) για συζευκτικές ερωτήσεις σε αδόμητα P2P. Ο μηχανισμός αναζήτησης τους είναι παρόμοιος σε αρκετά σημεία με τον Forward μηχανισμό. Δηλαδή πάλι μέσω QueryHits μηνυμάτων το σύστημα προσπαθεί να αυξήσει το recall για παρόμοιες ερωτήσεις που υποβάλλονται στο δίκτυο. Οι διαφορές εστιάζονται σε δύο σημεία: (1) Σαν βασικό σχήμα αναζήτησης θεωρούν την περίπτωση του constrained flooding, ενώ στην περίπτωση μας θεωρούμε τους k-walkers. (2) Τα πειραματικά τους αποτελέσματα στηρίζονται μόνο κατά την περίπτωση που ένας κόμβος υποβάλλει την ίδια ερώτηση πολλαπλές φορές. Το τελευταίο οδηγεί κατά την άποψη μας σε μονοδιάστατα συμπεράσματα για την συμπεριφορά του αλγορίθμου κάτι που φάνηκε στην εργασία μας.

Στο [11] προτείνεται μια διαφορετική προσέγγιση για αναζήτηση πληροφορίας σε αδόμητα P2P δίκτυα. Οι συγγραφείς θεωρούν ότι ο βασικός μηχανισμός αναζήτησης

παραμένει ένα σχήμα πλημμύρας παρόμοιο με αυτό που χρησιμοποιεί η Gnutella [3]. Όμως ένας κόμβος κατά την διάρκεια παραμονής του στο σύστημα προσπαθεί να εντοπίσει ποιοι είναι εκείνοι οι κόμβοι που απαντάνε στα ερωτήματα του. Η υπόθεση που γίνεται είναι η εξής: Εάν ένας κόμβος Β απαντήσει σε ένα ερώτημα που υποβάλλει ένας κόμβος Α μια δεδομένη χρονική στιγμή, τότε ο Β θα είναι σε θέση να απαντήσει και σε ένα διαφορετικό ερώτημα που θα υποβληθεί μελλοντικά από τον Α. Στην ουσία το σύστημα προσπαθεί να εντοπίσει κοινά ενδιαφέροντα μεταξύ των κόμβων του δικτύου. Οι κόμβοι εισέρχονται μέσα σε μια λίστα που καλείται shortcut list (λίστα συντομεύσεων). Στην συνέχεια όταν ο Α επιθυμεί να υποβάλει μια ερώτηση στο δίκτυο αρχικά εξετάζονται όλοι οι κόμβοι που υπάρχουν εκείνη την στιγμή στην shortcut list, αν η απάντηση δεν βρεθεί τότε θα εκτελεστεί πλημμύρα. Να σημειώσουμε πως με αυτόν τον τρόπο ο κάθε κόμβος εκτός από τις ιδεατές (overlay) συνδέσεις που φέρει στην αρχή κατά την είσοδο του στο δίκτυο, αποκτά δυναμικά κατά την διάρκεια του συστήματος και άλλες άμεσες (direct) συνδέσεις με κόμβους που μοιράζεται κοινά ενδιαφέροντα. Αυτό έχει σαν αποτέλεσμα να μεταβάλλεται η δομή του δικτύου όσον αφορά τους γείτονες που θα φέρει κάθε κόμβος. Τέλος, ανεύρεση νέων shortcuts επιτελείται και με την αντίστροφη λογική όταν δηλαδή κάποιος κόμβος ικανοποίει ένα ερώτημα.

Η κεντρική ιδέα του [11], έχει εμφανιστεί με μικρές παραλλαγές και σε άλλες εργασίες ([12], [13]) και εμπεριέχεται κατά κάποιον τρόπο και στα δικά μας προτεινόμενα σχήματα αναζήτησης. Η διαφορά μας σε σχέση με την ιδέα των shortcuts εντοπίζεται πως στην περίπτωση μας δεν μεταβάλλονται οι γείτονες ενός κόμβου.

Στο [14] οι συγγραφείς θέτουν το πρόβλημα της αναζήτησης σε αδόμητα P2P δίκτυα σε μια άλλη βάση παρουσιάζοντας την ιδέα των σημασιολογικών ιδεατών δικτύων (Semantic Overlay Networks (SONs)). Η ιδέα των SONs αναφέρει πως κόμβοι με παρόμοιο περιεχόμενο οφείλουν να μαθαίνουν ο ένας τον άλλον και να σχηματίζουν πάνω στο ιδεατό δίκτυο το δικό τους σημασιολογικό ιδεατό δίκτυο ανάλογα με το κοινό τους περιεχόμενο που διαθέτουν προς διαμοιρασμό. Ένας κόμβος μπορεί να συμμετέχει σε παραπάνω από ένα SON, ενώ από την άλλη όταν αναφέρουμε ότι σχηματίζεται ένα SON δεν σημαίνει πως αναφερόμαστε σε ένα πλήρες γράφημα,

αντίθετα μιλάμε για ένα συνεκτικό γράφημα. Η ιδέα των SONs έχει το πλεονέκτημα πως μια ερώτηση απαντάται πιο γρήγορα και σε λιγότερα βήματα. Με δεδομένο ότι γνωρίζουμε από ποιο SON μπορεί να απαντηθεί, το μόνο που χρειάζεται είναι να κατευθυνθεί στο συγκεκριμένο SON και να εξετάσει πλέον μόνο τους κόμβους που ανήκουν σε αυτό.

Στο [14] ορίζονται και αναλύονται 4 βασικά σχεδιαστικά προβλήματα που πρέπει να επιλύσει ένας σχεδιαστής μιας P2P εφαρμογής αν θέλει να αναπτύξει ένα P2P που να βασίζεται στην ιδέα των SONs. (1) Πρέπει να βρεθεί μια γενική ιεραρχία-κατηγοριοποίηση της συλλογής των εγγράφων που φέρει κάθε κόμβος. (2) Να οριστεί μια μέθοδος ταξινόμησης των εγγράφων σε κατηγορίες. (3) Να οριστεί ένας ταξινομητής που θα ορίζει ένας κόμβος σε ποιά SONs οφείλει να ενωθεί και (4) ένας ταξινομητής που θα ορίζει από ποια SONs μπορεί να επιλυθεί μια ερώτηση καθώς και με ποιόν τρόπο θα εκτελείται η αναζήτηση.

Το SETS [15] είναι μια πρώτη υλοποίηση ενός μηχανισμού αναζήτησης σε αδόμητα P2P που στηρίζεται στην ιδέα των SONs. Σε αυτήν την εργασία οι κόμβοι του συστήματος δημιουργούν 2 ειδών συνδέσεων (links) τα short distance και τα long distance. Τα πρώτα αναφέρονται σε συνδέσεις με κόμβους που έχουν παρόμοιο περιεχόμενο, ενώ τα δεύτερα σε συνδέσεις με κόμβους όπου το περιεχόμενο τους διαφέρει σημαντικά.

Οι συγγραφείς αναφέρουν τις ομάδες ως topic segments. Για να μάθει κάθε νέος κόμβος που εισάγεται στο σύστημα με ποια segments πρέπει να ενωθεί (να αποκτήσει short distance συνδέσεις) υπολογίζει τα term vectors για κάθε ένα από τα έγγραφα του με χρήση του μοντέλου unigram [16] και εν συνεχεία πραγματοποιώντας μια ένωση αυτών υπολογίζει μια πιο συμπαγή αναπαράσταση όλων των εγγράφων της συλλογής του που οι συγγραφείς ονομάζουν site vector. Αυτό το διάνυσμα από όρους το στέλνουν σε μία κεντρική (dedicated) εκ των προτέρων γνωστή τοποθεσία (site) η οποία εκτελώντας έναν αλγόριθμο ομαδοποίησης (clustering) υπολογίζει σε ποιο-α segment ένας κόμβος οφείλει να ενωθεί.

Το σύστημα απαντά συζευκτικά ερωτήματα εκφρασμένα με όρους-λέξεις όμοια με αυτά των δικών μας μεθόδων. Πιο συγκεκριμένα η διαδικασία της αναζήτησης εκτελείται ως εξής: Όταν ένας κόμβος υποβάλλει μια ερώτηση q αρχικά υπολογίζει με χρήση της συνάρτηση ομοιότητας συνημίτονου ποιά είναι η σχέση της q με κάθε ένα από τα segment που υπάρχουν στο δίκτυο. Να σημειώσουμε ότι κάθε κόμβος στο σύστημα γνωρίζει το κέντρο (περιγραφή) του κάθε segment και το οποίο του το παρέχει εξ αρχής η κεντρική τοποθεσία. Τα R πιο κοντινά ως προς την ερώτηση segments επιλέγονται ως τα πιο κατάλληλα, συνεπώς ο ερωτών κόμβος υποβάλλει R ερωτήματα που μέσω των long distance συνδέσεων προσπαθούν να προσεγγίσουν έναν κόμβο που να ανήκει σε αυτά τα segments. Τέλος, μόλις μια q φτάσει σε ένα κόμβο που ανήκει στο θεμιτό segment εκτελείται πλημμύρα, έχοντας την q να προωθείται σε όλες τις short distance συνδέσεις των κόμβων του segment.

Ένα από τα μειονεκτήματα του SETS είναι η ύπαρξη της κεντρικής τοποθεσίας η οποία και αναλαμβάνει να υπολογίζει τα segments που κάθε κόμβος οφείλει να ενωθεί, επιπλέον η συμμετοχή της είναι θεμελιώδη και στις ενημερώσεις που γίνονται κατά την διάρκεια της ζωής του συστήματος καθώς κόμβοι εισέρχονται/εξέρχονται και το περιεχόμενο των κόμβων μεταβάλλεται. Το GES [17] έρχεται να καλύψει αυτό το κενό στηριζόμενο πάνω στην ιδέα του SETS αλλά αντικαθιστώντας την κεντρική τοποθεσία με πιο κατανεμημένους μηχανισμούς. Τα πειράματα του GES έδειξαν ότι σε αρκετές περιπτώσεις η κατανεμημένη εκδοχή κατάφερνε και υπερίσχυε έναντι του SETS.

Στο Distributed Cache Table (DCT) [18] οι συγγραφείς προτείνουν ένα query-driven indexing σχήμα για την επεξεργασία (retrieval) multi-terms ερωτήσεων σε ένα δομημένο P2P δίκτυο. Η βασική τους ιδέα βασίζεται στην εξής παρατήρηση: Δεν έχει νόημα να δημιουργούμε και να συντηρούμε ανάστροφες λίστες (inverted lists) για όρους ή σύνολα από όρους τα οποία δεν εμφανίζονται συχνά σε ένα δοθέν ρεύμα ερωτήσεων (query stream), διότι οι ανάστροφες λίστες δεν θα χρησιμοποιηθούν σχεδόν ποτέ για την επίλυση των ερωτήσεων του συστήματος. Πάνω σε αυτό, το DCT προτείνει ο κάθε κόμβος να εισάγει στην cache αποτελέσματα (απαντήσεις) που συγκέντρωσε για ερωτήσεις που έκανε ο ίδιος κατά το παρελθόν. Συνεπώς όταν υποβάλλεται μια ερώτηση q στο δίκτυο εξετάζονται οι caches των κόμβων μήπως και

βρεθεί έστω και μια cache που περιέχει τα αποτελέσματα για μια ερώτηση q' που υποβλήθηκε κατά το παρελθόν και είναι υποσύνολο της q. Το βασικό πρόβλημα που είχαν να αντιμετωπίσουν είναι πως η cache κάθε κόμβου είναι άνω φραγμένη (δεν μπορεί κάθε κόμβος να εισάγει στην cache κάθε ερώτηση που υποβάλλεται στο σύστημα). Το κριτήριο για το αν μια q πρέπει να εισαχθεί στην cache ή όχι δίνεται μέσα από την εκτέλεση ενός άπληστου (greedy) αλγορίθμου έτσι ώστε να είναι κάθε φορά στην cache εκείνα τα ερωτήματα τα οποία επιλύουν εν γένει μεγάλο πλήθος ερωτήσεων, ήτοι έχουμε μεγάλο cachehit.

Αν και η μέθοδος DCT είναι προσαρμόσιμη στο ρεύμα ερωτήσεων και δημιουργεί indices που κατά βάση χρησιμοποιούνται κατά την επίλυση των ερωτημάτων το μειονέκτημα της μεθόδου εντοπίζεται στο γεγονός ότι αν μια q δεν μπορεί να επιλυθεί μέσω των κατανεμημένων caches αναγκαστικά θα εκτελεστεί πλημμύρα σε όλο το δίκτυο. Πάνω σε αυτήν την παρατήρηση το [19] συνδυάζει την δυναμικότητα της ιδέας του [18] με την αποφυγή των πλημμύρων προτείνοντας μια άλλη μέθοδο για indexing η οποία είναι η εξής: Αρχικά φτιάχνονται ανάστροφες λίστες για κάθε μοναδικό (single) όρο που ανήκει στην συλλογή των εγγράφων και στην συνέχεια βασιζόμενοι στο ρεύμα ερωτήσεων δημιουργούνται indices και για σύνολα από όρους τα οποία αποκαλούν keys (κλειδιά). Η βασική συνεισφορά του [19] έγκειται στον καθορισμό κριτηρίων για την δημιουργία ανάστροφων λιστών για σύνολα όρων. Το πιο βασικό εξ αυτών είναι το πόσο συχνά εμφανίστηκε το key στο μέχρι τώρα παρατηρούμενο ρεύμα ερωτήσεων.

Μια παρόμοια λογική που προηγήθηκε χρονικά του DCT εξετάζεται στο [20]. Οι συγγραφείς παρατηρούν το εξής: Αν το σύστημα διατηρεί ανάστροφες λίστες για τους όρους (a), (b) και (c) και υποβάλλεται μια συζευκτική ερώτηση q={a,b,c} από τον κόμβο p τότε οι ανάστροφες λίστες των (a) (b) και (c) πρέπει να μεταφερθούν στον p ώστε να απαντηθεί το ερώτημα (αυτός εν συνεχεία τοπικά να παράγει την τομή και να εμφανίσει τα αποτελέσματα στον χρήστη). Αν όμως υπήρχε έτοιμη μια ανάστροφη λίστα για το (a,b) τότε ο p χρειαζόταν να μεταφέρει μόνο 2 λίστες την (a,b) και την (c). Προφανώς στην δεύτερη περίπτωση η αξιοποίηση του εύρους ζώνης (bandwidth) γίνεται πιο αποτελεσματικά.

Για τον παραπάνω λόγο το σύστημα προσπαθεί να κάνει cache αποτελέσματα (views) ερωτήσεων που υποβλήθηκαν στο δίκτυο κατά το παρελθόν. Η συνεισφορά του [20] έγκειται στο ότι προτείνει μια καινούργια δομή την view tree (δένδρο όψεων) η οποία εμπεριέχει κάθε view που έχει παραχθεί στο σύστημα. Μέσω αυτής της δομής κάθε κόμβος που θέλει να επιλύσει μια q διασχίζει το δένδρο μέχρι να εντοπίσει τις πιο κατάλληλες q' υποσύνολα της q ώστε να ανακτήσει τις απαντήσεις. Ένα από τα μειονεκτήματα της προτεινόμενης μεθόδου είναι ότι αναφέρεται σε κεντρικοποιημένο P2P, ένας κόμβος πρέπει να αποθηκεύει το view tree.

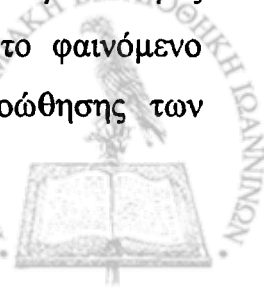
ΚΕΦΑΛΑΙΟ 5. ΣΥΜΠΕΡΑΣΜΑΤΑ

Σε αυτήν την εργασία συγκρίναμε δύο lazy-based μηχανισμούς για αναζήτηση πληροφορίας σε αδόμητα P2P δίκτυα, το Forward σχήμα και το Inverse. Το βασικό τους χαρακτηριστικό είναι πως οι κόμβοι του συστήματος αποθηκεύουν στις caches τους ερωτήματα τα οποία υποβάλλονται στο δίκτυο με στόχο αν στο μέλλον υποβληθούν παρόμοια ερωτήματα με τα παλιά, τα νέα να μπορούν να κατευθυνθούν σε κόμβους σχετικούς ως προς τις ερωτήσεις. Με αυτόν τον τρόπο πραγματοποιείται καθοδήγηση των ερωτημάτων με πληροφορία.

Η βασική τους διαφορά εντοπίζεται πως στο μεν Forward μηχανισμό αναζήτησης τα νέα ερωτήματα προωθούνται σε περιοχές του δικτύου που προηγουμένως είχαν απαντήσει σε παρόμοια ερωτήματα, στο δε Inverse σχήμα η προώθηση ενός ερωτήματος γίνεται προς εκείνο το μέρος του δικτύου από το οποίο κόμβοι κατά το παρελθόν ρώτησαν παρόμοια ερωτήματα.

Από το σύνολο των πειραμάτων που πραγματοποιήθηκαν το Inverse σχήμα αναζήτησης παρουσίασε μεγαλύτερο recall σε σχέση με το Forward, μάλιστα η διαφορά τους έφτανε αρκετές φορές και στο 50%. Ενώ η διαφορά που παρατηρήθηκε στο overlapping recall ήταν πολλές φορές πάνω από 100%. Το γεγονός αυτό είναι ιδιαίτερα σημαντικό αν αναλογιστούμε ότι το Inverse σχήμα σε αντίθεση με το Forward, δεν χρειάζεται QueryHits μηνύματα για να λειτουργήσει συνεπώς αξιοποιεί καλύτερα το εύρος ζώνης του δικτύου.

Το παραπάνω οφείλεται στο ότι οι caches του Inverse μηχανισμού γεμίζουν πιο γρήγορα σε σχέση με αυτές του Forward συνεπώς το Inverse επιτελεί περισσότερες προωθήσεις ερωτημάτων-μηνυμάτων με πληροφορία. Επιπρόσθετα, το φαινόμενο αυτό έγινε πιο έντονο όταν αλλάξαμε το βασικό μηχανισμό προώθησης των



μηνυμάτων και από k walkers κάναμε την σύγκριση πάνω σε ένα constrained flooding σχήμα.

Επιπλέον, όταν οι caches γέμιζαν με τον ίδιο ρυθμό και για τα δύο σχήματα το Inverse σχήμα συνέχιζε να παρουσιάζει μεγαλύτερο recall σε σχέση με αυτό του Forward. Αυτό οφείλεται στο ότι ο Forward μηχανισμός απαιτεί περισσότερα βήματα εκμάθησης, ένας κόμβος πρέπει να λάβει το ίδιο-παρόμοιο ερώτημα πολλές φορές για να μπορέσει να επιλέξει τον πραγματικά 'καλύτερο' γείτονα για να προωθήσει ένα δοθέν ερώτημα.

Τα παραπάνω φάνηκαν ακόμα και στην προσομοίωση ενός δυναμικού περιβάλλοντος στο οποίο κόμβοι αποχωρούν και εισέρχονται από και προς το δίκτυο κατά βούληση. Η διαφορά του Inverse σχήματος ως προς το Random ήταν τόσο μεγάλη που ακόμα και σε ένα drop rate της τάξεως του 30% ήταν σε θέση να διατηρήσει τον με πληροφορία χαρακτήρα του. Σε αντίθεση με το Forward σχήμα αναζήτησης του οποίου τα ποσοστά στο recall είχαν την ίδια απόδοση με το Random.

Το Inverse σχήμα αναζήτησης βρίσκει εφαρμογή σε περιβάλλον στο οποίο κόμβοι μοιράζονται κοινά ενδιαφέροντα, υποβάλλουν παρόμοιες ερωτήσεις. Σε ένα τέτοιο περιβάλλον η απόδοση του κρίνεται καλύτερη σε σχέση με το Forward μηχανισμό. Όμως το Forward σχήμα βρίσκει εφαρμογή και σε ένα περιβάλλον στο οποίο οι κόμβοι του δικτύου δεν μοιράζονται κοινά ενδιαφέροντα. Σε αυτήν την περίπτωση αν ένας κόμβος δράσει εγωιστικά και υποβάλλει ο ίδιος ένα δικό του ερώτημα πολλαπλές φορές θα πετύχει μια τεχνίτη αύξηση στο locality του ρεύματος των ερωτήσεων με αποτέλεσμα να παρουσιαστεί αύξηση στο recall.

Σαν μελλοντική δουλειά επιθυμούμε να συγκρίνουμε τα δυο σχήματα αναζήτησης και σε διαφορετικά δίκτυα όπως δένδρα και power-law γραφήματα και να καταγράψουμε πιο λεπτομερώς την συμπεριφορά τους.

ΑΝΑΦΟΡΕΣ

[1] I. Stoica, R. Morris, D. Karger, M. Kaashoek, H. Balakrishnan, "Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications", ACM SIGCOMM, San Diego, CA, pp. 149-160, August 2001.

[2] S. Ratnasamy, P. Francis, M. Handley, R. Karp, "A Scalable Content-Addressable Network", ACM SIGCOMM, San Diego, CA, pp. 161-172, August 2001.

[3] HTTP://WWW.GNUTELLA.COM/

[4] Q. Lv, P. Cao, E. Cohen, K. Li, S. Shenker, "Search and Replication in Unstructured Peer-to-Peer Computing", ICS02, New York, USA, June 2002.

[5] G. Salton, "Automatic Text Processing: The Transformation, Analysis, and Retrieval of Information by Computer", Addison-Wesley, Reading, MA, 1989.

[6] HTTP://WWW.OMNETPP.ORG/

[7] R. Baeza-Yates, A. Gionis, F. Junqueira, "The impact of Caching on Search Engines", SIGIR, Amsterdam, The Netherlands, July 23-27, 2007.

[8] A. Crespo, H. Garcia-Molina, "Routing Indices For Peer-to-Peer Systems", Proc of Int. Conf. On Distributed Computing Systems, Vienna, Austria, 2002.

[9] D. Tsoumakos, N. Roussopoulos, "Adaptive Probabilistic Search for Peer-to-Peer Networks", IEEE Third International Conference on P2P computing, P2P2003, 2003.

[10] V. Kalogeraki, D. Gunopoulos, D. Zeinalipour-atzi, "Local Search Mechanism for Peer-to-Peer Networks", ACM CIKM, Virginia, USA, 2002.

[11] K. Sripanidkulchai, B. Maggs, H. Zhang, "Efficient Content Location Using Interest-Based Locality in Peer-to-Peer Systems", IEEE INFOCOM, 2003.

[12] A. Löser, C. Tempich, B. Quilitz, W. Balke, S. Staab, W. Nejdl, "Searching Dynamic Communities with Personal Indexes", ISWC 2005, Vol 3729/2005, pp 491-505, October 17, 2005.

[13] C. Tempich, S. Staab, A. Wranik, "REMINDIN: Semantic Query Routing in Peer-to-Peer Networks based on Social Metaphors", ACM, WWW Conference, New York, 2004.

[14] A. Crespo, H. Garcia-Molina, "Semantic Overlay Networks for P2P Systems", 29th VLDB, Berlin, Germany, 2003.

[15] M. Bawa, G. Manku, P. Raghavan, "SETS: Search Enhanced by Topic Segmentation", SIGIR03, Torronto, Canada, July 28-August 1, 2003.

[16] J. Callan. "Distributed Information Retrieval", Advances in Information Retrieval, pages 127-150, 2000.

[17] Y. Zhu, X. Yang, Y. Hu, "Making Search Efficient on Gnutella-like P2P Systems", IEEE IPDPS, 2005.

[18] G. Skobeltsyn, K. Aberer, "Distributed Cache Table: Efficient Query-Driven Processing of Multi-Term Queries in P2P Networks", P2PIR, Arlington, Virginia, USA, November 11, 2006.

[19] G. Skobeltsyn, T. Luu, I. Podnar, M. Rajman, K. Aberer "Web Text Retrieval with a P2P Query-Driven Index", SIGIR, Amsterdam, Netherlands, 2007.

[20] B. Bhattacharjee, S. Chawathe, V. Gopalakrishnan, P. Keleher, B. Silaghai, "Efficient Peer-to-Peer Searches Using Result-Caching", IPTPS, Berkeley, CA, USA, 2003.

ΣΥΝΤΟΜΟ ΒΙΟΓΡΑΦΙΚΟ

Ο Φωτόπουλος Βασίλειος γεννήθηκε το 1981 στην Πάτρα. Εισήχθη στο τμήμα Πληροφορικής της Σχολής Θετικών Επιστημών του Πανεπιστημίου Ιωαννίνων το 2000 από το οποίο αποφοίτησε το 2005. Στο διάστημα 2005 με 2008 παρακολούθησε το μεταπτυχιακό πρόγραμμα σπουδών του ιδίου τμήματος.